

本扫描仅提供00083. com网站内部会员交流学习使用

如果觉得好请购买原版杂志

www.00083.com

经典计算机科学著作最新修订版

计算机程序设计艺术

第2卷 半数值算法

(第3版)

The Art of Computer
Programming

苏运霖 译

[美] DONALD E. KNUTH 著


Addison
Wesley

国防工业出版社
National Defence Industry Press
<http://www.ndip.com.cn>

计算机科学/程序设计

它本来是作为参考书撰写的，但有人却发现每一卷都可以饶有兴致地从头读到尾。一位中国的程序员甚至把他的阅读经历比做读诗。

如果你认为你确实是一个好的程序员，读一读Knuth的《计算机程序设计艺术》吧，要是你真把它读通了，你就可以给我递简历了。——Bill Gates

这一鸿篇巨制被广泛誉为对经典计算机科学的权威描述，头三卷在几十年来一直是学生、研究人员和业内人士的无价财富。

一部所有基础算法的宝典，今天的许多软件开发关于计算机程序设计的绝大多数知识都是从该宝典中获得的。——Byte, September 1995

不计其数的读者深受Knuth著作的影响。科学家们惊讶于他的分析的精美与雅致，而普通程序员每天都从其提供的“菜谱”中获得问题解决方案。书的恢宏、透彻、精确与幽默赢得了所有人的尊敬。

我无法描述它们在我学习和娱乐中伴我度过的愉悦时光。我在车里，在餐馆里，在家里……甚至在我儿子棒球小联赛的间隙都忘不了带上它们，一有空就捧出来阅读。——Charles Long

不管你基础如何，倘若你想认真地编写任何计算机程序，你都有理由把这套书的任何一卷抱回家，以便在你学习和工作的时候随时翻阅。

要是有一个问题难到要把Knuth的著作请下书架，那就太令人愉悦了。我发现，人们只要翻一翻它，就会对计算机产生极其有用的令人“恐慌”的影响。——Jonathan Laventhol

为反映该领域的最新发展，Knuth二十多年来第一次将三卷书全部做了修订。他的修订主要集中在上一版以来趋于一致的知识、已经解决的问题、以及有所变化的问题。为保持本书的权威性，关于该领域先驱工作的历史信息都做了更新。为维护作者苦心孤诣追求至善至美的盛誉，新的版本将敏锐和苛刻的读者发现的少量技术错误都做了更正。增加了上百的新习题，对您也是新的挑战。

责任编辑：辛再甫 zfxin@ndip.com.cn

ISBN 7-118-02707-3/TP·672

定价：98.00 元

计算机程序设计艺术

第2卷

半数值算法

(第3版)

[美] Donald E. Knuth 著
苏运霖 译

国防工业出版社

著作权合同登记号 图字:军-2001-019 号

图书在版编目(CIP)数据

计算机程序设计艺术. 第2卷, 半数值算法 / (美)
克努特(Knuth, D. E.); 苏运霖译. —3版. —北京: 国防
工业出版社, 2002.8
ISBN 7-118-02707-3

I. 计... II. ①克... ②苏... III. ①电子计算机
算法理论 ②电子计算机 - 随机数产生法 IV. TP301.6

中国版本图书馆 CIP 数据核字 (2001) 第 078811 号

Simplified Chinese edition copyright ©2002 by Pearson Education North Asia Limited
and National Defense Industry Press.

Original English language title: The Art of Computer Programming, Vol. 2, Seminumerical
Algorithms 3rd Edition by Donald E. Knuth

Copyright ©1998 by Addison Wesley Longman
All Rights Reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing
as Addison Wesley Longman, Inc.

This edition is authorized for sale only in the People's Republic of China (excluding the
Special Administrative Region of Hong Kong and Macau).

互联网网页 <http://www-cs-faculty.stanford.edu/~knuth/taocp.html> 包含本书及相关著作的最新
信息。

国防工业出版社出版发行

(北京市海淀区紫竹院南路 23 号)

(邮政编码 100044)

北京奥隆印刷厂印刷

新华书店经营

*

开本 787×1092 1/16 印张 48½ 959 千字

2002 年 8 月第 1 版 2002 年 8 月北京第 1 次印刷

印数: 1—4000 册 定价: 98.00 元

(本书如有印装错误, 我社负责调换)

前 言

O dear Ophelia!
I can ill do these numbers;
I have not art to reckon my groans.
亲爱的奥菲利娅;
这些数真让人烦恼;
我可没有计算我的愁怀的技巧。^{*}
— Hamlet: (Act II, Scene 2, Line 120)

本书所讨论的算法直接地涉及数。但我相信把它们叫做半数值算法是适当的,因为它们处于数值和符号计算的边界线上。每个算法不仅计算数值问题所要求的答案,而且也应与一台数字计算机的内部操作很好地融合。在许多情况下,人们都不可能充分品味某个算法的美,除非他也懂计算机机器语言;相应的机器程序的有效性是不能同算法本身分开的一个重要因素。问题是寻找出计算机处理数的最佳方法,这既要考虑数值,又要研究策略。因此本书的主题显然既是数值数学的一个部分,也是计算机科学的一个部分。

在数值分析的“高层次”上工作的某些人将把这里处理的课题当做是系统程序员的领域,而工作在系统程序设计“高层次”上的其他人将把这里处理的课题当做数值分析的领域。但我相信,还会剩下一些人,他们将愿意仔细地考察这些基本方法。尽管这些方法或许处于低层次上,但是它们却奠定了计算机在数值问题上的所有更强大应用的基础,因此了解它们就很重要了。在这里我们最关心的是数值数学和计算机程序设计之间的界面。正是这两种技巧的结合,才使这一课题变得如此有趣。

本书比起本丛书的其他卷来,其数学内容所占比例要显著地高得多。这是由于所处理的课题所致。大多数情况下,在这里所展开的必要的数学课题几乎都是从很皮毛的内容开始的(或者从第1卷证明的结果开始的)。但是在若干部分,显然仍需要读者具有一定的微积分学知识。

本卷是由整套丛书的第3章和第4章组成的。第3章涉及“随机数”:它不单单是对生成随机序列的各种方法的研究,它还研究随机性的统计检验,以及一致随机数与其它类型随机量间的转换;后一课题说明在实践中如何使用随机数。本章还有

* 朱生豪译文为:“亲爱的奥菲利娅啊,我的诗写得太坏;我不会用诗句来抒写我的愁怀。”(《哈姆莱特》,朱生豪译,吴兴华校,1997年人民出版社出版。)鉴于译文难以准确表达作者的意旨,本书特将作者引文原文附上。以下同。——本书责任编辑

一节包括了随机数本身的性质。第4章的意图是讲述经历了数百年的进步之后,人类对算术运算的有趣发现;它论述了表示数的各种系统,以及在這些系统之间如何进行转换;而且它还处理关于浮点数、高精度整数、有理数、多项式及幂级数的算术运算,也包括因子分解和求最大公因子等问题在内。


第3章和第4章均可作为从大学三年级到研究生层次的一学期课程的基础。尽管“随机数”和“算术”的课程现在都不是许多大学课程表的一部分,但我相信,读者会发现这两章的学科内容本身是有实际教育价值的,非常适合于统一论述。我本人的经验是,这些课程是向大学生们介绍初等概率论和数论的很好的手段。通常,在这样的人门性课程中讨论的几乎所有课题都很自然地在同应用相关联中出现,而这些应用可以成为促进学生学习和鉴赏理论的重要因素。其次,每一章都给出一些更深入课题的提示,它们将激发许多学生进行进一步研究的兴趣。

本书的大部分内容都是自成体系的,除了偶尔涉及在第1卷中说明的 MIX 计算机的讨论外。附录 B 列出了本书所用的数学符号,其中一些符号与传统数学书中略有差别。

第3版前言

本书的第2版完成于1980年,它实际上也是电子排版系统 $T_E X$ 和 METAFONT 的第一个重大的测试实例。现在我高兴地用她(正是为了她我萌发了开发电子排版系统的念头)的第3版庆祝 $T_E X$ 和 METAFONT 的全面开发成功。最终我将能以一致的格式拥有《计算机程序设计艺术》的所有各卷,而这将使它们很容易适应未来在印刷和显示技术上的变化。这些进展已经使我得以把长期以来一直想要做的数以千计的改进收到这些书中。

我逐字逐句地审阅了新版的所有文字,在或许加上一些更为深思熟虑的论述的同时,我试图保留原来句子的朝气;增添了几十道新习题,还对几十道旧的习题给出了新的和改进了的答案。变动随处可见,但最主要的是在3.5节(关于随机性的理论保证),3.6节(关于可移植的随机数生成程序),4.5.2小节(关于二进制的最大公因子算法,以及4.7节(关于幂级数的合成和迭代)。

 然而,《计算机程序设计艺术》的写作仍然是进行中的工作。关于半数值算法的研究继续以非凡的速度在发展着。因此,本书的某些部分被冠以“正在施工”的图标,用于对该部分的内容还不是最新表示歉意。我的文件中已经挤满了许多重要的素材。我打算从现在开始用大约16年的时间,把这些素材包含在第2卷最后的辉煌的第4版中,但是,我必须首先完成第4卷和第5卷,而且除非绝对需要,我一

刻也不想拖延它们的出版。

我要向在过去 35 年来帮助我搜集和改进这些素材的数百位人们致以最衷心的感谢。准备新版本的大多数困难工作是由 Silvio Levy 完成的,他熟练地编辑电子文本,而 Jeffrey Oldham 则负责把几乎全部图形转换成 METAFONT 的格式。我已经把机敏的读者们在第 2 版中所发现的每一个错误(以及,竟无人发现的一些错误也在内)都作了改正;而且我已经力图在新的材料中不引进新的错误。然而,我想仍然会存在某些缺点,我要尽快地改进它们。因此我将高兴地支付 2.56 美元给每一个技术、印刷或历史错误的头一位发现者,在网页 <http://www-cs-faculty.stanford.edu/~knuth/taocp.html> 列出了向我报告过的所有错误的更正。*

Stanford, California

D. E. K.

July 1997

*When a book has been eight years in the making,
there are too many colleagues, typists, students, teachers, and friends to thank*

*Besides, I have no intention of giving such people
the usual exoneration from responsibility for errors which remain*

They should have corrected me!

*And sometimes they are even responsible for ideas
which may turn out in the long run to be wrong*

Anyway, to such fellow explorers, my thanks.

当一本书已经编写了 8 年时,需要感谢的

同事、打字员、学生、老师和朋友真太多了。

此外,我无意对这些人宣布,对于遗留的错误,他们不负责任。

他们应当帮助我改正它们!

而且有时他们也应对一些经过长时间考验被证实是错的那些思想负责。

但无论如何,对于这样的同行探索者,我都要表示衷心感谢!

— EDWARD F. CAMPBELL, JR. (1975)

*'Defendit numerus,' [there is safety in numbers]
is the maxim of the foolish;*

*'Deperdit numerus,' [there is ruin in numbers]
of the wise*

“数能保平安”是极大的愚蠢;

“数中有祸”是明智的。

— C. C. COLTON (1820)

* 中文版已按 2001 年 11 月的最新勘误表更正了原书的错误。网页 <http://www.ndcp.com.cn/computer> 上也将列出对中文版的勘误表,我将很诚挚地对发现中文版错误的读者表示感谢,但我不敢保证也支付您 2.56 美元。——本书责任编辑

关于习题的说明

本套书的习题既可用于自学,也可用于课堂练习。无论是谁,如果想纯粹地通过阅读,而不将所阅读的信息应用到特定问题上,并由此牵引思考先前阅读的内容,就想学到一门学问,纵然可能,那也是很困难的。其次,对于我们自己的发现,我们总是领会得最透。因此,习题形成了这一套书的一个重要部分;我着意使这些习题含有丰富的信息,而且也尽量选择既有趣又有启发性的习题。

在许多书里,容易的习题被随机地混杂于极端困难的问题当中。有时这是不合适的,因为读者预先想要知道一道习题花多少时间——否则他们可能跳过这些习题了事。这一情况的典型例子是由 Richard Bellman 所著的 *Dynamic Programming* 一书。这是一本重要的先驱性的论著,其中在某些章的末尾,在“习题和研究题”的标题下,把一组问题收集在一起,而且一些极其平凡的问题出现在一些深入的、还未被解决的问题当中。据说,有人曾问过 Bellman 博士,怎样区分习题和研究题。他回答说:“如果你能解决它,那它就是一道习题,否则它就是一个研究题。”

但在这一类书里把研究问题和很容易的习题都包括进来确有其道理;因此,为使读者免于陷入确定哪是习题哪是研究题的困境,我给习题注明了等级分,以指出困难的程度。这些分数大体具有下列意义:

分数	解释
00	一个极其容易的问题,如果你已理解正文的内容,就可立即做出回答。这样一道题几乎总是可以“眉头一皱”就把它做出。
10	一个简单的问题。它要求你去思考刚刚学过的内容,但绝不意味着是困难的。你应当有能力在顶多一分钟之内就把它做出。在获得解答的过程中可能要用到笔和纸。
20	一个普通的问题。它检查你对正文内容的基本理解,但你可能需要十五或二十分钟才能完整地回答它。
30	一个中等难度或中等复杂的问题。这个题目可能需要两个小时以上的工作才能令人满意地解决。或者甚至更长时间,如果电视机在开着的话。
40	确实是一个十分困难或冗长的问题。在学校里,它将适合于作为一个学期的课程设计。一个学生应当有能力在相当长的时间里解决它,但这个解不会是平凡的。
50	就作者在编写本书时所知,这是一个还未令人满意地解决的问题,尽管已经有很多人做了尝试。如果你已经找到了这样一个问题的答案,你应当把它写出来发表。其次,本书的作者将乐意尽快地听到关

于这一解答的消息(当然,假定它是正确的)。

通过在这个“对数”尺上的内插,其它分数的意义也就清楚了。例如 17 分将表示比普通的题更为简单的一道习题。一个随后被某个读者解决了的 50 分的题在本书后来的版本中将以 45 的分数出现,而且会在互联网上的勘误表中刊出(参见版权页)。

分数除以 5 的余数表示所要求的详细工作量。因此分数是 24 的习题可能要比分数是 25 的题花费更长的时间来求解,但后者将要求更多的创造性。

作者已经认真地试图指定精确的分数,但是提出问题的人要想确切地知道,所提问题对求解的另一个人难度如何,肯定是困难的;而且每个人都会有较其他人更为适应的问题类型。只能希望这些分数较好地反映了习题的困难程度,希望读者把它们当做一般的导引,而不应作为绝对的指标。

本书是为具有不同程度的数学训练和素养的读者写的;有些习题是特意为有更多数学基础的读者提供的。如果一道题的出题动机或涉及的数学概念超越了主要兴趣仅仅是算法编程本身的读者应掌握的程度,则在分数前边加上 M 。如果一道习题的答案必须涉及在本书中未予提供的微积分或其它高等数学知识,则对这道题标以“ HM ”的字母。“ HM ”标记并不必定意味着困难。

某些习题的前边标有三角符号“ \blacktriangleright ”;这说明是特别有启发性的问题,因而特别予以推荐。当然,并不期望读者/学生来求解所有的习题,所以标出了那些看起来最有价值的部分(这并不意味着贬低其它习题!)。每个读者至少应该尝试去解分数是 10 或者更低的所有问题;三角符号可以帮助指出应该对具有较高分数的哪些问题予以优先考虑。

在答案部分中已经列出大多数习题的解答,请明智地使用它们。在你已真正地做出努力亲自解决问题之前,不要去翻答案,除非你确实没有时间来做这一特定的习题。在获得了你自己的解答或者对该题做了郑重的尝试之后,你可能会感到答案是有启发和有帮助的。给出的解答通常十分简短,作者认为你已经认真地通过自己的方法做过求解尝试,因而略去了其细节。有时解答所提供的信息比所要求的为少,但通常都是更多的。十分可能,你有比这里给的更好的答案,或者在答案中你发现一个错误。在这样的情况下,作者将乐意知道详细的情况。在本书以后的版本中将在适当地方刊登出这些改进了的答案以及提供这些解的人的姓名。

当做一道习题时,一般地你可以使用前边习题的答案,除非明确地禁止这样做。在对习题打分时,已经考虑到这一点了,因此很可能第 $n+1$ 题的分数比第 n 题还要低,尽管它把第 n 题的结果作为一个特殊情况包括进来。

代码含义	00 立即可解的
	10 简单的(一分钟)
	20 一般水平(一刻钟)
\blacktriangleright 特别推荐的	30 难度适中
M 面向数学的	40 学期设计
HM 要求“高等数学”的	50 研究题

习 题

►1. [00] 分数“M20”意味着什么?

2. [10] 在一本教科书中的习题对读者来说有什么价值?

3. [34] Leonhard Euler 在 1772 年猜测, 方程 $w^4 + x^4 + y^4 = z^4$ 没有正整数解, 但 Noam Elkies 在 1987 年证明, 它有无穷多个解[见 *Math. Comp.* **51** (1988), 825 ~ 835]。试求使得 $0 \leq w \leq x \leq y < z < 10^6$ 的所有整数解。

4. [M50] 试证明, 当 n 是整数, $n > 4$ 时, 方程 $w^n + x^n + y^n = z^n$ 没有正整数的 w, x, y, z 的解。

Exercise is the beste instrument in learnyng.

习题是学习的最好手段。

ROBERT RECORDE, *The Whetstone of Witte* (1557)

内 容 简 介

本书是国内外业界广泛关注的 7 卷本《计算机程序设计艺术》第 2 卷的最新版。本卷对半数值算法领域做了全面介绍,分“随机数”和“算术”两章。本卷总结了主要算法范例及这些算法的基本理论,广泛剖析了计算机程序设计与数值分析间的相互联系,其中特别值得注意的是作者对随机数生成程序的重新处理和对形式幂级数计算的讨论。

本书附有大量习题和答案,标明了难易程度及数学概念的使用。

本书内容精辟,语言流畅,引人入胜,可供从事计算机科学、计算数学、计算技术诸方面的工作人员参考、研究和借鉴,也是相关专业高等院校的理想教材和教学参考书。

目 录

第3章 随 机 数

3.1 引言	1
3.2 生成一致随机数	8
3.2.1 线性同余法	8
3.2.1.1 模数的选择	10
3.2.1.2 乘数的选择	15
3.2.1.3 效能	21
3.2.2 其它方法	23
3.3 统计检验	35
3.3.1 研究随机数据的一般检验方法	36
3.3.2 经验检验	53
3.3.3 理论检验	70
3.3.4 谱检验	82
3.4 其它类型的随机量	104
3.4.1 数值分布	105
3.4.2 随机抽样和洗牌	125
*3.5 什么是随机序列	131
3.6 小结	163

第4章 算 术

4.1 定位计数系统	175
4.2 浮点算术	192
4.2.1 单精度计算	193
4.2.2 浮点算术的精确度	207
*4.2.3 双精度计算	222
4.2.4 浮点数的分布	229
4.3 多精度算术	239
4.3.1 经典算法	239
*4.3.2 模算术	259
*4.3.3 乘法能有多快?	267

4.4 进制转换	289
4.5 有理算术	299
4.5.1 分数	299
4.5.2 最大公因子	302
4.5.3 欧几里得算法的分析	322
4.5.4 分解素因子	344
4.6 多项式算术	380
4.6.1 多项式除法	382
4.6.2 多项式的因子分解	400
4.6.3 求幂值	421
4.6.4 多项式求值	444
*4.7 幂级数的操作	480
习题答案	492
附录 A 数值数量表	718
附录 B 符号索引	723
索引与词汇表	727

第3章 随 机 数

*Any one who considers arithmetical methods
of producing random digits is, of course, in a state of sin.*

任何考虑用算术方法产生随机数字的人
都是在做错事。

— J. VON NEUMANN (1951)

*Let men suspect so that truth
keep probability in view.*

为了避免人们怀疑你的故事的真实性,
请不要忘了概率。

— JOHN GAY (1727)

*There is need not some beams of light
to guide men in the exercise of their Stochastic faculty.*

不需要什么光芒来照引人们
去使用他们的随机本领。

— JOHN OWEN (1662)

3.1 引 言

“随机地选择”的数,有许多不同类型的应用。例如:

a) 仿真 当使用一台计算机仿真自然现象时,就需要用随机数使事情变得逼真。仿真涉及从核物理(其中粒子受到随机的碰撞)直到运筹学(比如说,人们在随机的时间里进入一个飞机场)的许多领域。

b) 抽样 通常,要考察所有可能的情况是不实际的,而随机的抽样将使我们能了解“典型”的行为。

c) 数值分析 利用随机数已经想出了许多巧妙的技术来解复杂的数值问题。关于这个课题,已有不少专著。

d) 计算机程序设计 为检验计算机算法的有效性,随机值乃是数据的好来源。更重要的是,随机值对于随机化算法的运算是很要紧的,随机化算法经常比确定性算法要优越得多。随机数的使用是我们在这套书中所关注的主要应用,所以我们要在第3章,即在讲其它大多数计算机算法之前,先讲随机数。

e)决策 据说,有许多决策人通过抛硬币或掷飞镖等等来做出他们的判断。还听说某些大学教授也以此为基础来评分。有时,做出这样完全“无偏见”的判断是重要的。随机性也是矩阵游戏理论中优化策略的必不可少的部分。

f)美学 一点儿随机性就会使计算机生成的图形和音乐似乎更活泼。例如,下面左边的图案在某些场合就比右边的更有吸引力。



[请见 D. E. Knuth, *Bull. Amer. Math. Soc.* 1 (1979), 369.]

g)娱乐 摇骰子、洗扑克牌、转轮盘等,是人们很喜爱的娱乐方式。随机数的这些传统用法,已经被命名为“蒙特卡罗方法”,它已成为描述任何使用随机数的算法的通用术语。

考虑这一课题的人们几乎总是碰到关于“随机”一词含义的哲学性讨论。在某种意义下不存在什么随机数。例如,2 是一个随机数吗? 我们宁愿谈论具有一个确定分布的独立的随机数序列。这大致意味着,每个数的出现都只是偶然的,并且与这序列中的其它数无关,每个数落入任何给定的范围都有一个确定的概率。

在一个有限数集上的一个一致分布是一个这样的分布,其中每个可能的数都有相等的概率。凡分布一般都认为是一致的,除非特别说明是某种其它分布。

在一个(一致的)随机数字序列中,0 到 9 这 10 个数中每一个数出现的次数约为 $1/10$,每一对连续数字出现的次数约为 $1/100$,等等。然而,如果我们取一个有 100 万个数字的真正的随机序列,则它总不会恰好有 10 万个 0,10 万个 1,等等。事实上,这种机会是十分微小的;而由这种序列组成的序列,则平均地具有这一特征。

100 万个数字的任何一个特定的序列,和任何其它序列都是同等可能的。因此,如果我们正随机地选择 100 万个数字,而其中的头 999 999 个碰巧已经是零,则最后一个数字是零的机会,在一个真正随机的状态下,仍然恰恰是 $1/10$ 。这些说法对很多人来说似乎是自相矛盾的,但事实上并非如此。

对于一个随机序列的抽象定义,有许多好的描述方式。我们将在 3.5 节回头来讨论这个有趣的问题。但眼前,还是让我们来直观地了解这一概念吧!

许多年前,那些在科学工作中需要随机数的人们,用从一个“充分转动起来”的坛子中抓出球来或摇骰子、分牌之类的办法获得随机数。1927 年, L. H. C. Tippett 发布了超过 40 000 个随机数字的一张表,那些数字是“从人口统计调查报告中随机地取得的”。自那以后,已经造出了一些特殊的装置,用来机械地生成随机数。1939 年, M. G. Kendall 和 B. Babington-Smith 使用头一部这样的机器造出了有 100 000 个随机数字的一张表。1951 年首次安装的 Ferranti Mark I 计算机有一个内置指令,它使用一个电阻噪声生成器将 20 个随机位放进累加器中;这一功能受到了 A. M. Turing 的推荐。1955 年, RAND 公司又公布了一张被广泛使用的有百万个随机数字的表,这张表是利用另一部特殊装置得到的。一台取名为 ERNIE(厄尼)的著名随机数机器被使用了许多年,用于产生英国政府有奖债券的中奖号码。[见 Kendall 和

Babington-Smith 的论文, *J. Royal Stat. Soc.* **A101** (1938), 147~166; **B6** (1939), 51~61。也见 S. H. Lavington 对 Mark 1 的讨论, *CACM* **21** (1978), 4~12; 以及对 RAND 表的评论: *Math. Comp.* **10** (1956), 39~43。也见 W. E. Thomson 对 ERNIE 的讨论, *J. Royal Stat. Soc.* **A122** (1959), 301~333。]

在计算机问世之后不久,人们开始探求在计算机程序中求随机数的有效方法。可以使用一张表,但由于内存空间和输入时间的要求,这种方法的实用性有限,因为这种表可能太短了,而且编制和维护这种表也是一件麻烦的事情。可以把诸如 ERNIE 这样的机器联到计算机上,如同在 Ferranti Mark 1 中所做的那样,但这也并不令人满意,因为当校验一个程序时,它实际上不可能再一次精确地重复以前的计算;而且,这类机器还经常会发生不易查出的故障。20 世纪 90 年代技术的发展又使表变得有用起来,因为 CDROM 上可以分布(存储)十亿字节的经过测试的随机数。1995 年,George Marsaglia 将一个噪声二极管的电流输出与确定的经过扰频的 rap 音乐叠加在一起生成了 650MB 的随机值(他称之为黑白噪声)并把它们做成演示光盘,因而促进了随机数表的复兴。

早期用机械方式生成随机数的缺点,引起了人们利用一台计算机的普通算术操作来产生随机数的兴趣。1946 年前后,John von Neumann(冯诺伊曼)头一个建议利用这种方法;他的办法是取前面的随机数的平方,并抽取中部的数字。例如,如果正在生成 10 位数字,而且先前的值是 5772156649,则把它平方得到

33317792380594909201

因此下一个数就是 7923805949。

对于这种技术,有十分明显的异议:既然每个数完全由它先前的数所决定,那么以这样的方法产生的序列怎么会是随机的呢?(请见本章开始处冯诺伊曼的评述。)回答是,这个序列不是随机的,仅仅像是随机的而已。在典型的应用中,一个数与跟在它后面的那个数之间的真实关系并无客观的意义,因此,这种非随机的特征并不真是不可取的。从直观上看,平方取中似乎相当充分地搅乱了前面的数。

用这样一种确定的方法生成的序列,在高深的学术著作中通常都叫做伪随机或拟随机序列,但在本书中我们简单地称之为随机序列,只不过把它们理解成只是看起来像随机的罢了。也许任何随机序列最多只能说是“显然随机的”。凡是仔细地选择了适当的方法的,在计算机上以确定方式生成的随机数,都已十分成功地用于几乎每一项应用之中。当然,确定的序列并不总是答案,它们肯定不应代替 ERNIE 来抽奖。

已经证明,冯诺伊曼原来的“平方取中法”并不是求随机数的好方法。危险在于这个序列容易出现重复元素的短循环。例如,如果 0 一旦作为这个序列的一个数出现,则它将不断地重现本身。

一些人在 20 世纪 50 年代初曾以“平方取中法”进行了实验。G. E. Forsythe 用 4 位数字代替 10 位数字,试验了 16 个不同的初始值。结果发现,其中的 12 个导致了以循环 6100, 2100, 4100, 8100, 6100, ... 为结局的序列,而其中有两个退化成零。N. Metropolis 对平方取中法进行了广泛的试验,试验中大多采用二进数系统。他证

明了:用 20 位数字进行工作时,序列可能退化成 13 个不同的循环,其中最长的循环周期为 142。

当发现了 0 时,很容易用一个新的值重新开始平方取中法,但不容易避免长的循环。习题 6 和 7 讨论了确定循环序列周期的某些有趣方法,只须使用很少的存储空间。

在习题 9 和 10 中,指出了平方取中法的一个理论上的缺陷。另一方面, N. Metropolis 用 38 位数工作,在出现退化之前得到了大约 750 000 个数的序列,而且得到的 $750\,000 \times 38$ 位数字满意地通过了随机性的统计检验。[Symp. on Monte Carlo Methods (Wiley, 1956), 29~36。]这个实验证明,平方取中法能给出有用的结果,但是在没有实施精心的计算之前,对它过于信任还是危险的。

在最初编写本章时,人们所使用的许多随机数生成程序都不是很好的。人们习惯于不去研究这样的子程序,某些相当不令人满意的老方法仍被盲目地从一个程序员传到另一个程序员,以至用户对它原有的局限性一无所知。在本章我们将看到,掌握有关随机数生成程序的要点并不困难,尽管为避免通常易犯的错误,谨慎从事是必要的。

要想出一个十分简单明了的随机数生成程序并不容易。作者在 1959 年就对此有深刻体会,当时试图利用下列特定的方法来建立一个非常好的随机数生成程序:

算法 K(“超随机”数生成程序) 给定一个 10 位的十进制数 X ,本算法可以用来把 X 变为一个像是随机序列中的下一个数。尽管可以预期这个算法能得出一个相当随机的序列,但后面提到的理由说明了它事实上并非十全十美。(读者除了注意这个算法的复杂性外,不必详尽地研究它。请特别注意 K1 和 K2。)

K1. [选择迭代次数] 置 $Y \leftarrow \lfloor X/10^9 \rfloor$, 即置 X 的最高位有效数字。(我们将执行步骤 K2 到 K13 $Y+1$ 次;亦即,我们应用随机变换随机次。)

K2. [选择随机步骤] 置 $Z \leftarrow \lfloor X/10^8 \rfloor \bmod 10$, 亦即,置 X 的第二位最高有效数字,转向步骤 $K(3+Z)$ 。(亦即,我们转到程序中的一个随机步骤。)

K3. [确保 $\geq 5 \times 10^9$] 如果 $X < 5000000000$, 则置 $X \leftarrow X + 5000000000$ 。

K4. [平方取中] 以 $\lfloor X^2/10^5 \rfloor \bmod 10^{10}$ 代替 X , 亦即,以 X 平方取中来代替 X 。

K5. [乘] 以 $(1001001001X) \bmod 10^{10}$ 代替 X 。

K6. [伪补数] 如果 $X < 1000000000$, 则置 $X \leftarrow X + 9814055677$; 否则置 $X \leftarrow 10^{10} - X$ 。

K7. [互换两半] 把 X 的后五位数字与前五位数字互换, 即置 $X \leftarrow 10^5(X \bmod 10^5) + \lfloor X/10^5 \rfloor$, 即 $(10^{10} + 1)X$ 的中间 10 位数字。

K8. [乘] 同步骤 K5。

K9. [减小数字] 把 X 的十进表示的每个非 0 数字减 1。

K10. [99999 修改] 如果 $X < 10^5$, 则置 $X \leftarrow X^2 + 99999$, 否则置 $X \leftarrow X - 99999$ 。

K8. [乘] 同步骤 K5。

- K9.** [减小数字] 把 X 的十进表示的每个非 0 数字减 1。
- K10.** [99999 修改] 如果 $X < 10^5$, 则置 $X \leftarrow X^2 + 99999$, 否则置 $X \leftarrow X - 99999$ 。
- K11.** [规格化] (这时 X 不能为 0。) 如果 $X < 10^9$, 则置 $X \leftarrow 10X$ 并重复这一步骤。
- K12.** [修改过的平方取中法] 以 $\lfloor X(X-1)/10^5 \rfloor \bmod 10^{10}$ 代替 X , 亦即, 以 $X(X-1)$ 的中间 10 位数字代替 X 。
- K13.** [重复?] 如果 $Y > 0$, 则 Y 减 1 并返回到步骤 K2。如果 $Y = 0$, 则本算法终止, 以 X 作为所求的“随机”值。 ■

(对应于上述算法的机器语言程序竟是如此复杂, 以至不借助于注解来读它的清单时, 人们将不知道这个程序是干什么的。)

考虑到算法 K 的复杂设计, 这一算法似乎能产生无穷尽的令人难以置信的随机数, 但事实上, 当把这个算法头一次放到计算机上时, 它几乎立即收敛到 10 位数值 6065038420——非常巧合, 本算法把这个数转换成它自己(见表 1)。若以另外一个数开始, 则这个序列在 7 401 个值之后, 开始以长度为 3 178 的周期进行循环。

表 1 非常巧合: 算法 K 把数 6065038420 变成它自己

步骤	X (之后)	步骤	X (之后)
K1	6065038420	K9	1107855700
K3	6065038420	K10	1107755701
K4	6910360760	K11	1107755701
K5	8031120760	K12	1226919902 $Y = 3$
K6	1968879240	K5	0048821902
K7	7924019688	K6	9862877579
K8	9631707688	K7	7757998628
K9	8520606577	K8	2384626628
K10	8520506578	K9	1273515517
K11	8520506578	K10	1273415518
K12	0323372207 $Y = 6$	K11	1273415518
K6	9676627793	K12	5870802097 $Y = 2$
K7	2779396766	K11	5870802097
K8	4942162766	K12	3172562687 $Y = 1$
K9	3831051655	K4	1540029446
K10	3830951656	K5	7015475446
K11	3830951656	K6	2984524554
K12	1905867781 $Y = 5$	K7	2455429845
K12	3319967479 $Y = 4$	K8	2730274845
K6	6680032521	K9	1620163734
K7	3252166800	K10	1620063735
K8	2218966800	K11	1620063735
		K12	6065038420 $Y = 0$

这个事例的教益是,随机数并不是通过随机地选择的方法就可生成的,应当使用某些理论。

在以下的诸节中,我们将考虑比平方取中法和算法 K 更为优越的随机数生成程序,确保对应的序列具有某些所希望的性质,并且不出现退化。我们将稍微详细地剖析一下这种类随机行为的原由,而且也将考虑对随机数进行操作的技术。例如,我们要研究的问题之一是:在一个计算机程序内洗一副模拟的扑克牌。

3.6 节将对本章进行小结并列出若干参考文献。

习 题

►1. [20] 假设你希望不使用计算机来随机地得到一个十进数,问下列方法哪一个合适?

a) 随机地打开一本电话号码簿(亦即,随意地翻开一页)并且采用所选页上找到的头一个数的个位数字。

b) 和 a) 一样,但所用的是页号的个位数字。

c) 掷一个具有正二十面体形状的骰子,它的二十个面已经标以数字 0,0,1,1,...,9,9,当这个骰子停下来时使用出现于顶部的数字。(建议在铺有毡子的硬质桌面上掷骰子。)

d) 让一个盖氏计数器在放射性源下暴露一分钟(注意自身防护),并采用所得到计数的个位数字。假设盖氏计数器以十进制显示所记的数,并假设计数从 0 开始。

e) 看看你的手表,如果秒针的位置在 $6n$ 和 $6(n+1)$ 之间,则选择数字 n 。

f) 让一个朋友来想一个随机数字,并采用他说出的这个数字。

g) 让一个“敌人”来想一个随机数字,并采用他说出的这个数字。

h) 假设有十四马进行竞赛,而你对于这十四匹马的资历毫无所知。以任意的方式把数字 0~9 赋予这些马,并在竞赛结束后采用优胜者的数字。

2. [M22] 在一个有 100 万个十进数字的随机序列中,每个可能的数字出现 100 000 次的概率是多少?

3. [10] 按平方取中法,1010101010 之后是什么数?

4. [20] (a) 当实施了算法 K 的步骤 K11 之后,为什么 X 的值不能为 0? 如果 X 竟然为 0,则算法出了什么问题? (b) 使用表 1 推导,当以初始值 $X = 3830951656$ 重复应用算法 K 时会发生什么情况。

5. [15] 鉴于预先就知道由算法 K 所生成的任何序列最终都将是周期的(即使不出现如表 1 所示的巧合),试说明为什么在任何情况下都不能期望算法 K 提供无限多的随机数。

►6. [M21] 假设我们要生成一个在 $0 \leq X_n < m$ 范围内的整数序列 X_0, X_1, X_2, \dots , 令 $f(x)$ 是任一函数,使得若 $0 \leq x < m$, 则 $0 \leq f(x) < m$ 。试考虑由规则 $X_{n+1} = f(X_n)$ 形成的一个序列。(例子是平方取中法和算法 K。)

a) 说明这个序列在下述意义下终归是周期的:存在数 λ 和 μ , 对于它们,值 $X_0, X_1, \dots, X_\mu, \dots, X_{\mu+\lambda-1}$ 是不同的;但当 $n \geq \mu$ 时,有 $X_{n+\lambda} = X_n$ 。试求出 μ 和 λ 的极大和极小的可能值。

b) (R. W. Floyd) 证明存在一个 $n > 0$, 使得 $X_n = X_{2n}$, 且这种 n 的最小值在 $\mu \leq n \leq \mu + \lambda$ 的范围内。而且,在下述意义下 X_n 的值是惟一的:如果 $X_n = X_{2n}$, 且 $X_r = X_{2r}$, 则 $X_r = X_n$ 。

c) 使用 b) 的思想,试设计对于任何给定的函数 f 和给定的 X_0 来计算 μ 和 λ 的一个算法,且仅使用 $O(\mu + \lambda)$ 步和有限数量的内存单元。

►7. [M21] (R. P. Brent, 1977) 令 $\ell(n)$ 是小于或等于 n 的 2 的最大次幂, 例如, $\ell(15) = 8$ 和 $\ell(\ell(n)) = \ell(n)$ 。

a) 借助于习题 6 中的符号, 证明存在一个 $n > 0$, 使得 $X_n = X_{\ell(n)-1}$ 。求一公式, 它用 μ 和 λ 表示 n 的最小值。

b) 应用这一结果来设计一个算法, 把此算法和形为 $X_{n+1} = f(X_n)$ 的任何随机数生成程序联用, 即可防止它无限地循环。你的算法应该计算周期长度 λ , 并且它只应当使用少量的存储空间——你不能简单地存储所有计算出来的序列值!

8. [23] 用两位十进数对平方取中法做全面检验。

a) 我们可以从 100 个可能的值 00, 01, ..., 99 中的任何一个来开始这个过程, 这些值中有多少最终将导致重复的循环 00, 00, ...? [例子: 从 43 开始, 我们得到序列 43, 84, 05, 02, 00, 00, 00, ...]

b) 有多少可能的最终循环? 最长的循环是多长?

c) 在这个序列重复以前, 哪个(或哪些)开始值将使序列中不同元素的个数为最大?

9. [M14] 证明用 b 进制的 $2n$ 位数的平方取中法有下列缺点: 如果序列包括有其最高 n 位有效数字为 0 的任何数, 则后继数将越变越小, 直到重复地出现 0 为止。

10. [M16] 在上题的假定下, 如果 X 的最低 n 位有效数字为 0, 则关于 X 之后的数列有何特性? 如果最低 $n+1$ 位有效数字为 0, 又如何?

►11. [M26] 考虑有习题 6 中所述那种形式的随机数生成程序序列。如果我们随机地选择 $f(x)$ 和 X_0 , 换句话说, 如果我们假定 m^m 个可能的函数 $f(x)$ 的每一个都是同等可能的, 而且假定 X_0 的 m 个可能的值的每一个也都是同等可能的。问这个序列最终退化为一个长度 $\lambda = 1$ 的循环的概率是多少? (注: 在这个问题的假定中给出了考虑这类“随机的”随机数生成程序的一个自然的方式。可以考虑一个像算法 K 这样的方法, 这个方法具有某些类似于这里所考虑的平均生成程序的特性。从这个问题的答案可以看出表 1 异常地巧合的程度。)

►12. [M31] 在上题的假定之下, 最终循环的平均长度是多少? 在开始循环之前, 序列的平均长度是多少? (使用习题 6 的符号, 我们希望考察 λ 和 $\mu + \lambda$ 的平均值。)

13. [M42] 如果 $f(x)$ 是在习题 11 的意义下随机地选定的, 则通过改变开始值 X_0 , 得到的最长循环的平均长度是多少? (注: 在 $f(x)$ 是一个随机排列的情况下, 我们已经考虑了类似的问题。见习题 1.3.3-23。)

14. [M38] 如果 $f(x)$ 是在习题 11 的意义下随机地选定的, 则通过改变开始值而得到不同的最终循环的平均个数是多少? [参考习题 8 b)。]

15. [M15] 如果 $f(x)$ 是在习题 11 的意义下随机地选定的, 则不管 X_0 如何选择, 不存在长度为 1 的最终循环的概率是多少?

16. [15] 如习题 6 中那样生成的序列, 在至多生成 m 个值后, 必然开始重复。假设我们推广这个方法, 使得 X_{n+1} 既依赖于 X_n 也依赖于 X_{n-1} ; 形式地说, 令 $f(x, y)$ 是一个这样的函数, 即如果 $0 \leq x, y < m$, 则 $0 \leq f(x, y) < m$ 。任选 X_0 和 X_1 , 然后令

$$X_{n+1} = f(X_n, X_{n-1}), \quad n > 0$$

试问这样构造出来的序列可能达到的极大周期是多少?

17. [10] 推广上题的情况, 使得 X_{n+1} 依赖于这个序列的前 k 个值

18. [M20] 试创立一个类似于习题 7 的方法, 以找出习题 17 中所讨论的随机数生成程序的一般形式下的循环。

19.[M48] 在更一般的情况下, X_{n+1} 依赖于这个序列前面的 k 个值; m^m 个函数 $f(x_1, \dots, x_k)$ 的每一个都认为是同等可能的。试对这种情况求解习题 11 到习题 15。(注意, 在习题 2.3.4.2-23 中分析了产生极大周期的函数个数。)

20.[30] 求所有的非负 $x < 10^{10}$, 它们经由算法 K 最终导致表 1 中的自再现的数。

21.[42] 证明或反驳: 由算法 K 定义的 $X \mapsto f(X)$ 的映射恰有长度为 3178, 1606, 1024, 943 和 1 的五个循环。

22.[21](H. Rolletschek) 通过使用序列 $f(0), f(1), f(2), \dots$ 而非 $x_0, f(x_0), f(f(x_0)), \dots$ 生成随机数, 是否一个好想法? 其中 f 是一个随机函数。

► 23.[M26](D. Foata 和 A. Fuchs, 1970) 证明在习题 6 中考虑的 m^m 个函数 $f(x)$ 的每一个都可表示为有下列性质的一个序列 $(x_0, x_1, \dots, x_{m-1})$ 。

- i) $(x_0, x_1, \dots, x_{m-1})$ 是 $(f(0), f(1), \dots, f(m-1))$ 的一个排列。
- ii) $(f(0), \dots, f(m-1))$ 可以从 $(x_0, x_1, \dots, x_{m-1})$ 惟一地构造出来。
- iii) 出现在 f 的循环元素是 $\{x_0, x_1, \dots, x_{k-1}\}$, 其中 k 是使得这 k 个元素都不相同的最大下标。

iv) $x_j \in \{x_0, x_1, \dots, x_{j-1}\}$ 意味着 $x_{j-1} = f(x_j)$, 除非 x_j 是在 f 的循环中的最小元素。

v) $(f(0), f(1), \dots, f(m-1))$ 是 $(0, 1, \dots, m-1)$ 的一个排列, 当且仅当通过 1.3.3 小节的“非寻常对应”, $(x_0, x_1, \dots, x_{m-1})$ 是该排列的逆。

vi) $x_0 = x_1$ 当且仅当通过习题 2.3.4.4-18 的构造, $f(x)$ 是 x 的父亲, (x_0, \dots, x_{m-1}) 表示一个有向树。

3.2 生成一致随机数

在这一节里, 我们将考虑生成一随机分数(一致地分布于 0 和 1 之间的随机实数) U_n 的序列的方法。由于一台计算机仅能以有限的精度来表示一个实数, 故我们实际上将生成在 0 和某个数 m 之间的整数 X_n 。分数

$$U_n = X_n/m$$

便处于 0 和 1 之间。通常, m 是计算机的字长, 所以 X_n 可以(保守地)看成是一个计算机字的整数内容, 小数点假定是在最右边; 而 U_n 可以(大体上)看成是同一字的内容, 它的小数点假定在最左边。

3.2.1 线性同余法

当今使用的最流行的随机数生成程序是 D. H. Lehmer 1949 年介绍过的下列方案的特殊情况[见 *Proc. 2nd Symp. on Large-Scale Digital Computing Machinery* (Cambridge, Mass.: Harvard University Press, 1951), 141~146]。我们选择 4 个魔术整数:

$$\begin{array}{ll}
 m, \text{模数}; & 0 < m \\
 a, \text{乘数}; & 0 \leq a < m \\
 c, \text{增量}; & 0 \leq c < m \\
 X_0, \text{开始值}; & 0 \leq X_0 < m
 \end{array} \quad (1)$$

然后通过置

$$X_{n+1} = (aX_n + c) \bmod m, \quad n \geq 0 \quad (2)$$

而得到所求的随机数序列 $\langle X_n \rangle$ 。这个序列称做线性同余序列。对 m 求余有点像确定转动的轮盘上球的落点。

例如,当 $m=10$ 和 $X_0=a=c=7$ 时,得到的序列是

$$7, 6, 9, 0, 7, 6, 9, 0, \dots \quad (3)$$

如此例所示,对于 m, a, c 和 X_0 的所有选择,这个序列并不总是“随机”的。本章的后面部分将仔细探讨适当选择这些魔术数的原则。

例子(3)说明了同余序列总是进入一个循环的事实,亦即,它最终必定在 n 个数之间无休止地重复循环。这个性质对于具有一般形式 $X_{n+1} = f(X_n)$ 的任何序列都是共同的,这里 f 把一个有限集合转换成它自身;参见习题3.1-6。这种重复的循环称为周期,序列(3)有一个长度为4的周期。一个有用的序列当然应有相当长的周期。

$c=0$ 的特殊情况值得明确指出,因为 $c=0$ 时数的生成过程比 $c \neq 0$ 时要稍微快些。我们后面看到, $c=0$ 的限制缩短了序列的周期长度,但是它仍然有可能得到一个相当长的周期。Lehmer原来的生成方法只有 $c=0$ 的情况,尽管他提出了 $c \neq 0$ 的可能性。取 $c \neq 0$ 来得到更长的周期的想法是分别由Thomson[Comp. J. 1 (1958), 83, 86]和Rotenberg[JACM 7 (1960), 75~77]提出来的。许多作者用术语乘同余法和混合同余法来分别表示 $c=0$ 和 $c \neq 0$ 的线性同余法。

这一章自始至终都将在上述意义下来使用字母 m, a, c 和 X_0 。而且,我们发现为了简化公式,定义

$$b = a - 1 \quad (4)$$

将是有益的。

我们可以立即排除 $a=1$ 的情况,因为这将意味着 $X_n = (X_0 + nc) \bmod m$,因而可以肯定它不具有随机序列的特性。而 $a=0$ 的情况甚至更坏。因此为实用起见可以假定

$$a \geq 2, \quad b \geq 1 \quad (5)$$

现在我们可以证明等式(2)的一个推广

$$X_{n+k} = (a^k X_n + (a^k - 1)c/b) \bmod m, \quad k \geq 0, n \geq 0 \quad (6)$$

它借助于第 n 项来直接表达第 $(n+k)$ 项。(在这个等式中 $n=0$ 的特殊情况值得注

* 本书“模”与 mod, modulo 或 modulus 对应,可作名词、动词,也可作介词。作动词时含义为“做模运算”;作介词时含义为“以……为模”。modulo 在算式中出现时一般未予译出。——本书责任编辑

意。)由此得出,由 $\langle X_n \rangle$ 的每第 k 项组成的子序列是另一个线性同余序列,该子序列具有乘数 $a^k \bmod m$ 和增量 $((a^k - 1)c/b) \bmod m$ 。

等式(6)的一个重要推论是:由 m, a, c 和 X_0 定义的一般序列,能非常简单地借助 $c=1$ 和 $X_0=0$ 的特殊情况来表示。设

$$Y_0 = 0, \quad Y_{n+1} = (aY_n + 1) \bmod m \quad (7)$$

根据等式(6),我们得到 $Y_k \equiv (a^k - 1)/b \pmod{m}$,因此,等式(2)中定义的一般序列满足

$$X_n = (AY_n + X_0) \bmod m, \text{ 其中 } A = (X_0b + c) \bmod m \quad (8)$$

习 题

1. [10] 例(3)显示 $X_4 = X_0$ 这样一个情况,所以这个序列再次从起点开始。试举出 $m=10$ 的线性同余序列的一个例子,对于这样一个例子, X_0 绝不会在该序列中再次出现。

► 2. [M20] 证明:如果 a 和 m 是互素的,则数 X_0 将总是出现于周期中。

3. [M10] 如果 a 和 m 不互素,试说明为什么这个序列不太完备并且大概不是很随机的,因此我们一般都要求乘数 a 和模数 m 互素。

4. [11] 证明等式(6)。

5. [M20] 对于 $k \geq 0$,等式(6)成立。如果可能,试对于负的 k 值,给出借助于 X_n 表示 X_{n+k} 的公式。

3.2.1.1 模数的选择 我们当前的目标是为定义线性同余序列的参数寻求好的值。让我们首先考虑数 m 的适当选择。既然一个周期不可能多于 m 个元素,我们希望 m 的值稍微大些。(即使一个人只想生成随机的0和1,他也不应取 $m=2$,因为这个序列最好也只有形式 $\dots, 0, 1, 0, 1, 0, 1, \dots$! 在3.4节,讨论了通过修改随机数来得到随机的0和1的方法。)

影响对 m 的选择的另一个因素是生成速度:我们要挑选一个值,使得 $(aX_n + c) \bmod m$ 的计算很快。

以MIX为例,我们可以通过把 y 放进寄存器A和X中然后除以 m 来计算 $y \bmod m$ 。假定 y 和 m 为正,我们将看到 $y \bmod m$ 出现在寄存器X中。可是除法操作比较慢,如果我们把 m 取成特别方便的值,例如取成计算机的字长,则可避免用除法。

令 w 是计算机的字长, w 在 e 位的二进制计算机上是 2^e ,在 e 位的十进制计算机上是 10^e 。(在本书中我们将经常使用字母 e 表示任意的整指数,而不是自然对数的底,希望这一符号在文中不至于引起歧义。*物理学家在使用 e 表示电子的电荷时,也会遇到同样的问题。)加法运算的结果通常在模 w 后给出,除非在1的补码机器上;而乘法模 w 也很简单,因为所求结果是乘积的低位一半。因此,下面的程序

* 中译本用正体的 e 表示自然对数的底,而用斜体的 e 表示任意整指数。——本书责任编辑

有效地计算 $(aX + c) \bmod w$:

```
LDA  A    rA ← a
MUL  X    rAX ← (rA) · X
SLAX 5    rA ← rAX mod w
ADD  C    rA ← (rA + c) mod w    |
```

(1)

结果出现在寄存器 A 中。溢出标志在这些指令结束后可能是开的;如果不希望出现这种情况,可以在代码后加上“JOV * + 1”关闭溢出标志。

一个不太为人所知的聪明技术可用于模 $w + 1$ 的计算。鉴于后文将解释的原因,我们一般希望 $m = w + 1$ 时, $c = 0$, 所以我们只须计算 $(aX + c) \bmod (w + 1)$ 。下面的程序完成这一计算:

```
01  LDAN  X    rA ← -X
02  MUL   A    rAX ← (rA) · a
03  STX   TEMP
04  SUB   TEMP  rA ← rA - rX
05  JANN  * + 3  如果 rA ≥ 0, 则出口
06  INCA  2     rA ← rA + 2
07  ADD   = w - 1 = rA ← rA + w - 1    |
```

(2)

寄存器 A 现在即含有值 $(aX) \bmod (w + 1)$ 。当然,这个值可以位于 0 和 w 之间的任何一点,包括 0 和 w 在内,因而读者自然会产生疑问,寄存器 A 怎么表示这么多的值! (这个寄存器显然不能保存大于 $w - 1$ 的数。)回答是:当且仅当结果等于 w 时,上面的程序才会导致溢出(假定开始时溢出标志是闭的)。我们可以以 0 来表示 w , 因为当 $X = 0$ 时,通常将不使用程序(2)。如果溢出出现于同余序列模 $(w + 1)$ 中,最方便的就是简单地拒绝 w 的值。于是,我们也就可以避免溢出。这只需把程序(2)的行 05 和 06 改成“JANN * + 4; INCA 2; JAP * - 5”就可以了。

为了证明程序(2)果真确定 $(aX) \bmod (w + 1)$, 请注意,在行 04 中我们从乘积的上半部减去下半部,这一步不会出现溢出;而且如果 $aX = qw + r$, 其中 $0 \leq r < w$, 则在 04 行之后,在寄存器 A 中我们得到量 $r - q$, 现在

$$aX = q(w + 1) + (r - q)$$

而且因为 $q < w$, 我们就有 $-w < r - q < w$; 因此 $(aX) \bmod (w + 1)$ 或者等于 $r - q$, 或者等于 $r - q + (w + 1)$, 视 $r - q \geq 0$ 或 $r - q < 0$ 而定。

采用类似的技术可以得到两个数的乘积对 $(w - 1)$ 的模;见习题 8。

在后几节中,要求有关于 m 的素因子的一些知识,以便正确地选择乘数 a 。表 1 列出了对于几乎每一个已知的计算机字长 $w \pm 1$ 的完全素因子分解,如果需要,4.5.4 小节的方法可用来扩充这个表。

读者很可能会问,既然选择 $m = w$ 明明方便得多,何以还要自寻烦恼地考虑使用 $m = w \pm 1$ 呢? 原因在于,当 $m = w$ 时, X_n 右边的数字与左边的数字相比,其随

机性要弱得多。如果 d 是 m 的一个因子,而且

$$Y_n = X_n \bmod d \quad (3)$$

则我们可以容易地证明

$$Y_{n+1} = (aY_n + c) \bmod d \quad (4)$$

(因为对于某个整数 q , $X_{n+1} = aX_n + c - qm$, 当 d 是 m 的因子时, 两边对 d 取模就消去了 qm 。)

为了举例说明等式(4)的意义, 假设我们有一台二进制的计算机。如果 $m = w = 2^e$, 则 X_n 的低四位是 $Y_n = X_n \bmod 2^4$ 。等式(4)的要旨是, X_n 的低四位形成其长度为 16 或更短的周期的一个同余序列。类似地, 低五位也是周期的, 其周期至多是 32, 而且 X_n 的最低有效位或者是常数, 或者严格地在 0 和 1 之间交替着。

当 $m = w \pm 1$ 时, 就不出现这个情况。这时, X_n 的低位的特性恰像高位一样地随机。例如, 假定 $w = 2^{35}$ 及 $m = 2^{35} - 1$, 而我们又只考虑它们模 31, 71, 127 或 122921 的余数(参考表 1), 则这个序列的数就不是很随机的了; 但低位(它表示这个序列取模 2 的数)却是令人满意地随机的。

表 1 $w \pm 1$ 的素因子分解

$2^e - 1$	e	$2^e + 1$
7·31·151	15	$3^2 \cdot 11 \cdot 331$
3·5·17·257	16	65537
131071	17	$3 \cdot 43691$
$3^3 \cdot 7 \cdot 19 \cdot 73$	18	$5 \cdot 13 \cdot 37 \cdot 109$
524287	19	$3 \cdot 174763$
$3 \cdot 5^2 \cdot 11 \cdot 31 \cdot 41$	20	$17 \cdot 61681$
$7^2 \cdot 127 \cdot 337$	21	$3^2 \cdot 43 \cdot 5419$
$3 \cdot 23 \cdot 89 \cdot 683$	22	$5 \cdot 397 \cdot 2113$
47·178481	23	$3 \cdot 2796203$
$3^2 \cdot 5 \cdot 7 \cdot 13 \cdot 17 \cdot 241$	24	$97 \cdot 257 \cdot 673$
31·601·1801	25	$3 \cdot 11 \cdot 251 \cdot 4051$
$3 \cdot 2731 \cdot 8191$	26	$5 \cdot 53 \cdot 157 \cdot 1613$
$7 \cdot 73 \cdot 262657$	27	$3^4 \cdot 19 \cdot 87211$
$3 \cdot 5 \cdot 29 \cdot 43 \cdot 113 \cdot 127$	28	$17 \cdot 15790321$
$233 \cdot 1103 \cdot 2089$	29	$3 \cdot 59 \cdot 3033169$
$3^2 \cdot 7 \cdot 11 \cdot 31 \cdot 151 \cdot 331$	30	$5^2 \cdot 13 \cdot 41 \cdot 61 \cdot 1321$
2147483647	31	$3 \cdot 715827883$
$3 \cdot 5 \cdot 17 \cdot 257 \cdot 65537$	32	$641 \cdot 6700417$
$7 \cdot 23 \cdot 89 \cdot 599479$	33	$3^2 \cdot 67 \cdot 683 \cdot 20857$
$3 \cdot 43691 \cdot 131071$	34	$5 \cdot 137 \cdot 953 \cdot 26317$
$31 \cdot 71 \cdot 127 \cdot 122921$	35	$3 \cdot 11 \cdot 43 \cdot 281 \cdot 86171$
$3^3 \cdot 5 \cdot 7 \cdot 13 \cdot 19 \cdot 37 \cdot 73 \cdot 109$	36	$17 \cdot 241 \cdot 433 \cdot 38737$
223·616318177	37	$3 \cdot 1777 \cdot 25781083$
$3 \cdot 174763 \cdot 524287$	38	$5 \cdot 229 \cdot 457 \cdot 525313$
$7 \cdot 79 \cdot 8191 \cdot 121369$	39	$3^2 \cdot 2731 \cdot 22366891$

(续)

$2^e - 1$	e	$2^e + 1$
$3 \cdot 5^2 \cdot 11 \cdot 17 \cdot 31 \cdot 41 \cdot 61681$	40	$257 \cdot 4278255361$
$13367 \cdot 164511353$	41	$3 \cdot 83 \cdot 8831418697$
$3^2 \cdot 7^2 \cdot 43 \cdot 127 \cdot 337 \cdot 5419$	42	$5 \cdot 13 \cdot 29 \cdot 113 \cdot 1429 \cdot 14449$
$431 \cdot 9719 \cdot 2099863$	43	$3 \cdot 2932031007403$
$3 \cdot 5 \cdot 23 \cdot 89 \cdot 397 \cdot 683 \cdot 2113$	44	$17 \cdot 353 \cdot 2931542417$
$7 \cdot 31 \cdot 73 \cdot 151 \cdot 631 \cdot 23311$	45	$3^2 \cdot 11 \cdot 19 \cdot 331 \cdot 18837001$
$3 \cdot 47 \cdot 178481 \cdot 2796203$	46	$5 \cdot 277 \cdot 1013 \cdot 1657 \cdot 30269$
$2351 \cdot 4513 \cdot 13264529$	47	$3 \cdot 283 \cdot 165768537521$
$3^2 \cdot 5 \cdot 7 \cdot 13 \cdot 17 \cdot 97 \cdot 241 \cdot 257 \cdot 673$	48	$193 \cdot 65537 \cdot 22253377$
$179951 \cdot 3203431780337$	59	$3 \cdot 2833 \cdot 37171 \cdot 1824726041$
$3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 31 \cdot 41 \cdot 61 \cdot 151 \cdot 331 \cdot 1321$	60	$17 \cdot 241 \cdot 61681 \cdot 4562284561$
$7^2 \cdot 73 \cdot 127 \cdot 337 \cdot 92737 \cdot 649657$	63	$3^3 \cdot 19 \cdot 43 \cdot 5419 \cdot 77158673929$
$3 \cdot 5 \cdot 17 \cdot 257 \cdot 641 \cdot 65537 \cdot 6700417$	64	$274177 \cdot 67280421310721$
$10^e - 1$	e	$10^e + 1$
$3^3 \cdot 7 \cdot 11 \cdot 13 \cdot 37$	6	$101 \cdot 9901$
$3^2 \cdot 239 \cdot 4649$	7	$11 \cdot 909091$
$3^2 \cdot 11 \cdot 73 \cdot 101 \cdot 137$	8	$17 \cdot 5882353$
$3^4 \cdot 37 \cdot 333667$	9	$7 \cdot 11 \cdot 13 \cdot 19 \cdot 52579$
$3^2 \cdot 11 \cdot 41 \cdot 271 \cdot 9091$	10	$101 \cdot 3541 \cdot 27961$
$3^2 \cdot 21649 \cdot 513239$	11	$11^2 \cdot 23 \cdot 4093 \cdot 8779$
$3^3 \cdot 7 \cdot 11 \cdot 13 \cdot 37 \cdot 101 \cdot 9901$	12	$73 \cdot 137 \cdot 99990001$
$3^2 \cdot 11 \cdot 17 \cdot 73 \cdot 101 \cdot 137 \cdot 5882353$	16	$353 \cdot 449 \cdot 641 \cdot 1409 \cdot 69857$

另一个方案是令 m 为小于 w 的最大素数,利用 4.5.4 小节的技术,可以找到这个素数,而且在该节中有充分大的素数的表。

在大多数应用中,低位是不重要的,因此假如使用随机数的程序员明智地选择 $m = w$,则结果便十分令人满意了。

我们迄今的讨论都是以 MIX 这样的“带符号量”的计算机为基础的。尽管有某些指令的变化,但类似的想法也适用于使用补码的机器。例如,DEC 20 计算机有 36 位二进制,并使用 2 的补码的算术运算;当它计算两个非负整数的乘积时,低半部包含低有效位的 35 位和一个加号。因此,在这样的机器上,我们将取 $w = 2^{35}$,而不是 2^{36} 。在 IBM System/370 计算机上,32 位的 2 的补码算术是不同的:一个乘积的低半部含完全的 32 位。某些程序员觉得这是一个缺点,因为当操作数为正时低半部可能为负,而改正它却是一件麻烦的事;但是从随机数生成的观点来看,这实际上是一个明显的优点,因为我们可以取 $m = 2^{32}$ 而不是 2^{31} (见习题 4)。

习 题

1. [M12] 在习题 3.2.1-3 中,我们知道最好的同余生成程序要求 a 与 m 互素。试证明在这种情况下,有可能仅用三条 MIX 指令来计算 $(aX + c) \bmod w$,而不是像程序(1)中那样用四条。计

算结果在寄存器 X 中。

2. [16] 试写出具有下列特征的一个 MIX 子程序:

调用序列: JMP RANDOM

入口条件: 单元 XRAND 含一整数 X

出口条件: $X \leftarrow rA \leftarrow (aX + c) \bmod w$, $rX \leftarrow 0$, 溢出关闭

(因此, 调用这个子程序将产生一个线性同余序列的下一个随机数。)

► 3. [M25] 许多计算机都不提供以一个字长的数来除两个字长的数的能力; 它们仅提供单字长的数的运算, 例如当 x 和 y 都是小于字长 w 的非负整数时, $\text{himult}(x, y) = \lfloor xy/w \rfloor$, $\text{lomult}(x, y) = xy \bmod w$; 假定 $0 \leq a, x < m < w$, 而且 $m \perp w$, 试说明怎样借助于 himult 和 lomult 来计算 $ax \bmod m$ 。你可以使用依赖于 a, m 和 w 的预先计算的常数。

► 4. [21] 在诸如 IBM System/370 系列这样的 2 的补码机器上, 对于 $m = 2^v$, 讨论线性同余序列的计算。

5. [20] 假定 m 小于字长, 而且 x, y 是小于 m 的非负整数, 试证明差 $(x - y) \bmod m$ 可以仅由四条 MIX 指令来计算, 而不用做任何除法。试问, 为求和式 $(x + y) \bmod m$, 最好的程序是什么?

► 6. [20] 上题指出, 减法模 m 比加法模 m 更易于实现。试讨论由规则

$$X_{n+1} = (aX_n - c) \bmod m$$

生成的序列。这些序列与在上文中定义的线性同余序列有实质的差别吗? 它们是否更适合于在计算机上有效地计算?

7. [M24] 在表 1 中你能看出什么模式?

► 8. [20] 写出计算 $(aX) \bmod (w - 1)$ 的类似于程序 (2) 的一个 MIX 程序。在你的程序的输入和输出中, 值 0 和 $w - 1$ 应看做是等价的。

► 9. [M25] 大多数高级程序设计语言都不提供以单字长的整数来除两个字长的整数的好方法, 它们也不提供习题 3 中的 himult 运算。本题的目的是对于变量 x 和对于常数 $0 < a < m$, 当我们希望计算 $ax \bmod m$ 时, 找出对付这样的局限的一个合理的方法。

a) 证明如果 $q = \lfloor m/a \rfloor$, 我们有 $a(x - (x \bmod q)) = \lfloor x/q \rfloor(m - (m \bmod a))$ 。

b) 使用 a) 的恒等式来计算 $ax \bmod m$, 而无须计算绝对值超过 m 的任何数, 假定 $a^2 \leq m$ 。

10. [M26] 习题 9 b) 的解答有时在 $a^2 > m$ 时也有效。对于 0 和 m 之间的所有 x , 精确地说, 有多少个这样的乘数 a , 对于它说来, 在该方法下中间结果绝不超过 m ?

11. [M30] 继续习题 9, 说明有可能仅仅使用下列的基本运算就能计算 $ax \bmod m$:

i) $u \times v$, 其中 $u \geq 0, v \geq 0$ 和 $uv < m$;

ii) $\lfloor u/v \rfloor$, 其中 $0 < v \leq u < m$;

iii) $\lfloor u - v \rfloor \bmod m$, 其中 $0 \leq u, v < m$ 。

事实上, 通过 i) 和 ii) 的至多 12 种运算, 以及 iii) 的有限个数的运算, 而不用预先计算依赖于 a 和 m 的常数, 就总有可能来做这件事。例如, 试说明, 当 a 是 62089911 和 m 是 $2^{31} - 1$ 时, 应如何进行。(这些常数出现在表 3.3.4-1 中。)

► 12. [M28] 考虑用笔和纸或算盘的计算。

a) 什么是用 10 来乘一个给定的 10 位数, 再对 999998999 取模的好方法?

b) 同样的问题, 但代之以乘 99999900 (模 999998999)。

c) 对于 $n = 1, 2, 3, \dots$, 说明怎样计算 $99999900^n \bmod 999998999$ 。

d) 把这样的计算同 $1/999998999$ 的小数展开联系起来。

e) 说明, 这些思想使得: 通过对每个生成的数只使用一些运算, 就有可能实现某种类型的线性同余生成程序。

13. [M24] 重复上一道题,但对模 9999999001 以及乘数 10 和 8999999101 进行。

14. [M25] 推广上两道题的思想,得到有非常大的模的一大类线性同余生成程序。

3.2.1.2 乘数的选择 在这一小节中,我们将说明怎样选择乘数 a ,以给出具有极大长度的周期。长的周期对于任何用作随机数来源的序列来说是必不可少的。确实,我们总是希望这个周期所含的数比任何具体应用中所需求的多得多。因此,在本节中,我们将研究周期长度的问题。然而,读者应该记住,长周期对于序列的随机性说来,仅仅是所希望的准则之一。例如,当 $a = c = 1$ 时,这个序列只不过是 $X_{n+1} = (X_n + 1) \bmod m$,它显然有长度为 m 的周期,但它绝不是随机的。有关选择一个乘数的其它考虑,将在这一章的后面部分给出。

由于只可能有 m 个不同的值,故周期确实不可能大于 m 。我们能达到极大长度 m 吗? 上面的例子说明这总是可能的,但是选取 $a = c = 1$ 不能产生所求的序列。让我们来研究得到周期长度 m 的 a, c 和 X_0 的所有可能的选择。结果是,所有这样的参数值都可非常简单地加以表征;当 m 是不同素数的积时,只有 $a = 1$ 才产生完全的周期,但当 m 可被某个素数的高次幂整除时, a 的选择才有相当大的范围。下面的定理使我们很容易指出周期是否为极大。

定理 A 由 m, a, c 和 X_0 所定义的线性同余序列有周期长度 m 当且仅当

- i) c 与 m 互素;
- ii) 对于整除 m 的每个素数 $p, b = a - 1$ 是 p 的倍数;
- iii) 如果 m 是 4 的倍数,则 b 也是 4 的倍数。

用于证明这个定理的思想至少可追溯到一百年前。但在 $m = 2^e$ 的特殊情况下,在这种具体形式下的定理的头一个证明是由 M. Greenberger 给出的[见 JACM 8 (1961), 383~389],而 Hull 和 Dobell 证明了在一般情况下条件 i), ii) 和 iii) 的充分性[见 SIAM Review 4 (1962), 230~254]。为证明这个定理,我们首先要考虑本身就很有趣的某些辅助性的数论结果。

引理 P 设 p 是一个素数, e 是一个正整数,其中 $p^e > 2$, 如果

$$x \equiv 1 \pmod{p^e}, \quad x \not\equiv 1 \pmod{p^{e+1}} \quad (1)$$

则

$$x^p \equiv 1 \pmod{p^{e+1}}, \quad x^p \not\equiv 1 \pmod{p^{e+2}} \quad (2)$$

证明 对于不是 p 的倍数的某个整数 q , 我们有 $x = 1 + qp^e$ 。由二项式公式

$$x^p = 1 + \binom{p}{1} qp^e + \cdots + \binom{p}{p-1} q^{p-1} p^{(p-1)e} + q^p p^{pe} =$$

$$1 + qp^{e+1} \left(1 + \frac{1}{p} \binom{p}{2} qp^e + \frac{1}{p} \binom{p}{3} q^2 p^{2e} + \cdots + \frac{1}{p} \binom{p}{p} q^{p-1} p^{(p-1)e} \right)$$

圆括弧中的量是一个整数,而且事实上圆括弧中除去第一项之外,各项都是 p 的倍数,因为如果 $1 < k < p$, 则二项式系数 $\binom{p}{k}$ 可为 p 所整除(参见习题 1.2.6-10)。因

此

$$\frac{1}{p} \binom{p}{k} q^{k-1} p^{(k-1)e}$$

可为 $p^{(k-1)e}$ 所整除。最后一项是 $q^{p-1} p^{(p-1)e-1}$, 由于当 $p^e > 2$ 时, $(p-1)e > 1$, 因而它可为 p 所整除。于是 $x^p \equiv 1 + qp^{e+1} \pmod{p^{e+2}}$, 证毕。(注: 习题 3.2.2-11 (a) 是这个结果的推广。) ■

引理 Q 设把 m 分解成素因子

$$m = p_1^{e_1} \cdots p_t^{e_t} \quad (3)$$

则由 (X_0, a, c, m) 所确定的线性同余序列的周期长度 λ , 是诸线性同余序列 $(X_0 \bmod p_j^{e_j}, a \bmod p_j^{e_j}, c \bmod p_j^{e_j}, p_j^{e_j})$ 的周期长度 λ_j 的最小公倍数, 其中, $1 \leq j \leq t$ 。

证明 对 t 用归纳法, 只须证明, 如果 m_1 与 m_2 互素, 则由 $(X_0, a, c, m_1 m_2)$ 所确定的线性同余序列的周期长度 λ 是序列 $(X_0 \bmod m_1, a \bmod m_1, c \bmod m_1, m_1)$ 和 $(X_0 \bmod m_2, a \bmod m_2, c \bmod m_2, m_2)$ 的周期长度 λ_1 和 λ_2 的最小公倍数。根据上一节的等式(4), 如果这三个序列的元素分别记为 X_n, Y_n, Z_n , 则将有

$$Y_n = X_n \bmod m_1 \text{ 和 } Z_n = X_n \bmod m_2, \text{ 对于所有 } n \geq 0$$

因此, 由 1.2.4 小节的定律 D, 我们发现

$$X_n = X_k \text{ 当且仅当 } Y_n = Y_k \text{ 和 } Z_n = Z_k \quad (4)$$

设 λ' 是 λ_1 和 λ_2 的最小公倍数; 我们希望证明 $\lambda' = \lambda$ 。由于对于所有适当大的 $n, X_n = X_{n+\lambda}$, 我们有 $Y_n = Y_{n+\lambda}$ (因此 λ 是 λ_1 的倍数), 且 $Z_n = Z_{n+\lambda}$ (因此 λ 是 λ_2 的倍数), 所以必然 $\lambda \geq \lambda'$ 。其次, 我们知道, 对于所有适当大的 $n, Y_n = Y_{n+\lambda'}$, $Z_n = Z_{n+\lambda'}$; 因此由式(4), $X_n = X_{n+\lambda'}$ 。这证明 $\lambda \leq \lambda'$ 。所以 $\lambda = \lambda'$ 。■

现在着手证明定理 A。由引理 Q 知, 只须对 m 是一个素数的乘幂这种特殊情况证明定理 A 就够了。这是因为, 当且仅当对于 $1 \leq j \leq t, \lambda_j = p_j^{e_j}$ 时,

$$p_1^{e_1} \cdots p_t^{e_t} = \lambda = \text{lcm}(\lambda_1, \cdots, \lambda_t) \leq \lambda_1 \cdots \lambda_t \leq p_1^{e_1} \cdots p_t^{e_t}$$

为真。

因此, 假定 $m = p^e$, 其中 p 是素数, e 是正整数。当 $a = 1$ 时定理显然为真, 所以可取 $a > 1$ 。当且仅当每个可能的整数 $0 \leq x < m$ 都出现于这个周期中时, 周期长度才能为 m 。因为在周期中没有出现一次以上的值, 因此周期长度为 m 当且仅当 $X_0 = 0$ 的序列的周期长度为 m , 而且有理由假设 $X_0 = 0$ 。由公式 3.2.1-(6), 我们有

$$X_n = \left(\frac{a^n - 1}{a - 1} \right) c \bmod m \quad (5)$$

如果 c 不与 m 互素, 则 X_n 绝不能为 1, 所以定理的条件 i) 是必要的。周期长度为 m 的充要条件是, 使 $X_n = X_0 = 0$ 的 n 的最小正值是 $n = m$ 。由关系式(5)和条件 i), 定理的证明便归结为证明如下事实:

引理 R 假定 $1 < a < p^e$, 其中 p 为素数, 如果 λ 是使得 $(a^\lambda - 1)/(a - 1) \equiv$

0(modulo p^e)的最小正整数,则

$$\lambda = p^e \quad \text{当且仅当} \quad \begin{cases} a \equiv 1 \pmod{p}, & \text{当 } p > 2 \text{ 时} \\ a \equiv 1 \pmod{4}, & \text{当 } p = 2 \text{ 时} \end{cases}$$

证明 假设 $\lambda = p^e$, 如果 $a \not\equiv 1 \pmod{p}$, 则当且仅当 $a^n - 1 \equiv 0 \pmod{p^e}$ 时, $(a^n - 1)/(a - 1) \equiv 0 \pmod{p^e}$ 。于是, 条件 $a^{\lambda} - 1 \equiv 0 \pmod{p^e}$ 意味着 $a^{\lambda} \equiv 1 \pmod{p}$; 但由定理 1.2.4F, $a^{\lambda} \equiv a \pmod{p}$, 因此 $a \not\equiv 1 \pmod{p}$ 导致矛盾。如果 $p = 2$ 和 $a \equiv 3 \pmod{4}$, 则由习题 8 有

$$(a^{2^{e-1}} - 1)/(a - 1) \equiv 0 \pmod{2^e}$$

这些论证说明, 每当 $\lambda = p^e$ 时, 一般地必有 $a = 1 + qp^f$, 其中 $p^f > 2$ 且 q 不是 p 的倍数。

剩下要证明的是, 这个条件对于导致 $\lambda = p^e$ 是充分的。通过重复地应用引理 P, 可以发现对所有 $g \geq 0$,

$$a^{p^g} \equiv 1 \pmod{p^{f+g}}, \quad a^{p^g} \not\equiv 1 \pmod{p^{f+g+1}}$$

因此,

$$\begin{aligned} (a^{p^g} - 1)/(a - 1) &\equiv 0 \pmod{p^g} \\ (a^{p^g} - 1)/(a - 1) &\not\equiv 0 \pmod{p^{g+1}} \end{aligned} \quad (6)$$

特别是, $(a^{p^e} - 1)/(a - 1) \equiv 0 \pmod{p^e}$ 。现在, 同余序列 $(0, a, 1, p^e)$ 有 $X_n = (a^n - 1)/(a - 1) \pmod{p^e}$; 因此, 它有长度为 λ 的周期, 即当且仅当 n 是 λ 的倍数时 $X_n = 0$ 。因此 p^e 是 λ 的倍数。仅当对于某个 g , $\lambda = p^e$ 时这才可能出现, 而且关系式(6)意味着 $\lambda = p^e$, 从而完成了证明。 ■

至此, 定理 A 的证明即告完成。 ■

在结束本节之前, 让我们来考察一下 $c = 0$ 时纯乘法性生成程序的特殊情况。尽管在这种情况下, 随机数生成的过程稍微快些, 但定理 A 说明了极大周期长度不可能达到。事实上, 这是十分明显的。这是因为, 现在这一序列满足关系式

$$X_{n+1} = aX_n \pmod{m} \quad (7)$$

而且 $X_n = 0$ 的值绝不会出现, 以免这个序列退化成 0。一般说来, 如果 d 是 m 的任何因子, 而且 X_n 是 d 的倍数, 则乘法性序列的所有随后的元素 X_{n+1}, X_{n+2}, \dots , 都是 d 的倍数。所以当 $c = 0$ 时, 要求对所有的 n , X_n 与 m 互素, 而这就限制了周期的长度至多为 $\varphi(m)$, 即 0 与 m 之间同 m 互素的整数个数。

即使我们约定 $c = 0$, 仍有可能达到一个可接受的长周期。现在让我们来寻找关于乘数的条件, 使得在这个特殊情况下, 周期尽可能地长。

根据引理 Q, 当 $m = p^e$ 时这个序列的周期完全依赖于诸序列的周期。我们来考虑该情况。我们有 $X_n = a^n X_0 \pmod{p^e}$, 显然, 如果 a 是 p 的倍数, 则这周期的长度将是 1, 所以我们取 a 与 p 互素。于是, 周期是使得 $X_0 = a^\lambda X_0 \pmod{p^e}$ 的最小整数

λ 。如果 X_0 与 p^e 的最大公因子是 p^f , 则这个条件就等价于

$$a^\lambda \equiv 1 \pmod{p^{e-f}} \quad (8)$$

由欧拉定理(习题 1.2.4-28), $a^{\varphi(p^{e-f})} \equiv 1 \pmod{p^{e-f}}$; 因此, λ 是

$$\varphi(p^{e-f}) = p^{e-f-1}(p-1)$$

的一个因子。当 a 与 m 互素时, 使 $a^\lambda \equiv 1 \pmod{m}$ 的最小整数 λ 通常称为 a 模 m 的阶。使模 m 的阶达到最大的 a 的任何值称为模 m 的一个本原元(素)。

设 $\lambda(m)$ 表示一本原元的阶, 即模 m 的阶的极大值, 则据上边的分析说明 $\lambda(p^e)$ 是 $p^{e-1}(p-1)$ 的一个因子; 只须稍为留意(见以下的习题 11~16), 即可给出在所有情形下 $\lambda(m)$ 的精确值如下:

$$\begin{aligned} \lambda(2) &= 1, \quad \lambda(4) = 2, \quad \lambda(2^e) = 2^{e-2}, \text{ 如果 } e \geq 3 \\ \lambda(p^e) &= p^{e-1}(p-1), \quad \text{如果 } p > 2 \\ \lambda(p_1^{e_1} \cdots p_t^{e_t}) &= \text{lcm}(\lambda(p_1^{e_1}), \dots, \lambda(p_t^{e_t})) \end{aligned} \quad (9)$$

我们的论述可以总结为下面的定理:

定理 B [C. F. Gauss, *Disquisitiones Arithmeticae* (1801), § 90 ~ § 92] 当 $c = 0$ 时, 可能的极大周期是 $\lambda(m)$, 其中 $\lambda(m)$ 在式(9)中定义。如果

- i) X_0 与 m 互素,
- ii) a 是以 m 为模的本原元,

则可实现这一周期。 ■

注意, 如果 m 为素数, 则可得长度为 $m-1$ 的周期; 这恰恰比极大长度小 1, 所以, 从实用的观点看, 这样一个周期已有我们所期望的长度。

现在的问题是怎样才能找到以 m 为模的本原元。这一节结尾的诸习题告诉我们, 当 m 是素数或素数的幂时, 有一个相当简单的答案, 这即是在以下定理中所述的结果。

定理 C 数 a 是以 p^e 为模的本原元, 当且仅当下列情况成立:

- i) $p=2, e=1$, 且 a 是奇数;
- ii) $p=2, e=2$, 且 $a \bmod 4 = 3$;
- iii) $p=2, e=3$, 且 $a \bmod 8 = 3, 5$ 或 7 ;
- iv) $p=2, e \geq 4$, 且 $a \bmod 8 = 3$ 或 5 ;
- v) p 是奇数, $e=1, a \not\equiv 0 \pmod{p}$, 而且对 $p-1$ 的任何素因子 $q, a^{(p-1)/q} \not\equiv 1 \pmod{p}$;
- vi) p 是奇数, $e > 1, a$ 满足 v), 而且 $a^{p-1} \not\equiv 1 \pmod{p^2}$ 。 ■

对于充分大的 p 值, 如果我们知道 $p-1$ 的因子的话, 通过使用 4.6.3 小节讨论的计算乘方的有效方法, 很容易在一台计算机上检验这个定理的条件 v) 和 vi)。

定理 C 只可应用于素数的幂。但如果已给出模 p^e 本原的值 a_j , 则采用在 4.3.2 节中讨论的中国剩余算法, 有可能找出一个 a 的值, 使得 $a \equiv a_j \pmod{p^e}$

p_j^e), $1 \leq j \leq t$, 而这个数 a 将是以 $p_1^e \cdots p_t^e$ 为模的本原元。因此, 对任何适度大小的模数 m , 有一个相当有效的方法来构造满足定理 B 的条件的那些乘数, 尽管这些计算在一般情况下是比较冗长的。

在 $m = 2^e, e \geq 4$ 的普通情况下, 上述条件简化成一个要求, 即 $a \equiv 3$ 或 $5 \pmod{8}$ 。这时, 所有可能的乘数的四分之一将使周期长度等于 $m/4$, 而当 $c = 0$ 时, $m/4$ 是可能的极大值。

第二个最普通的情况是 $m = 10^e$ 。应用引理 P 和 Q, 不难得到在一台十进制的计算机上达到极大周期的必要和充分条件(参见习题 18):

定理 D 如果 $m = 10^e, e \geq 5, c = 0$ 且 X_0 不是 2 或 5 的倍数, 则当且仅当 $a \pmod{200}$ 等于下列 32 个值之一时, 线性同余序列的周期是 $5 \times 10^{e-2}$:

$$\begin{aligned} &3, 11, 13, 19, 21, 27, 29, 37, 53, 59, 61, 67, 69, 77, 83, 91, 109, 117, \\ &123, 131, 133, 139, 141, 147, 163, 171, 173, 179, 181, 187, 189, 197 \end{aligned} \quad (10)$$

习 题

1. [10] 具有 $X_0 = 5772156648, a = 3141592621, c = 2718281829$ 和 $m = 10000000000$ 的线性同余序列的周期长度是多少?

2. [10] 当 m 是 2 的一个乘幂时, 下列两个条件是否足以保证极大长度的周期? “(i) c 是奇数; (ii) $a \pmod{4} = 1$ 。”

3. [13] 假设 $m = 10^e$, 其中 $e \geq 2$, 并进一步假设 c 是奇数, 而且不是 5 的倍数。证明: 当且仅当 $a \pmod{20} = 1$ 时, 该线性同余序列有极大长度的周期。

4. [M20] 假设 $m = 2^e$ 和 $X_0 = 0$, 如果数 a 和 c 满足定理 A 的条件, 问 $X_{2^{e-1}}$ 的值是多少?

5. [14] 当 $m = 2^{35} + 1$ 时, 试求满足定理 A 的条件的所有乘数 a (m 的素因子可从表 3.2.1.1-1 中找到)。

► 6. [20] 当 $m = 10^6 - 1$ 时, 求出满足定理 A 的条件的所有乘数 a (见表 3.2.1.1-1)。

► 7. [M23] 一个同余序列的周期不必以 X_0 开始, 但总能找到下标 $\mu \geq 0, \lambda > 0$, 使得每当 $n \geq \mu$ 时 $X_{n+\lambda} = X_n$, 而且 μ 和 λ 是具有这个性质的最小可能的值(参考习题 3.1-6 和 3.2.1-1)。如果 μ_i 和 λ_i 是对应于序列 $(X_i \pmod{p_i^e}, a \pmod{p_i^e}, c \pmod{p_i^e}, p_i^e)$ 的下标, 而且 μ 和 λ 对应于合成序列 $(X_0, a, c, p_1^e \cdots p_t^e)$, 则引理 Q 指出, λ 是 $\lambda_1, \dots, \lambda_t$ 的最小公倍数。问如何借助于 μ_1, \dots, μ_t 表示 μ 的值? 当 $m = p_1^e \cdots p_t^e$ 固定时, 通过改变 X_0, a 和 c , 可以得到的 μ 的最大值等于多少?

8. [M20] 证明: 如果 $a \pmod{4} = 3$, 则当 $e > 1$ 时, $(a^{2^{e-1}} - 1)/(a - 1) \equiv 0 \pmod{2^e}$ 。(用引理 P₀)

► 9. [M22] (W. E. Thomson) 当 $c = 0$ 和 $m = 2^e \geq 16$ 时, 定理 B 和 C 指出, 当且仅当乘数 a 满足 $a \pmod{8} = 3$, 或者 $a \pmod{8} = 5$ 时周期的长度为 2^{e-2} 。证明每个这样的序列在下列意义下, 实质上是一个 $m = 2^{e-2}$ 的满周期线性同余序列:

a) 如果 $X_{n+1} = (4c + 1)X_n \pmod{2^e}$, 而且 $X_n = 4Y_n + 1$, 则

$$Y_{n+1} = ((4c + 1)Y_n + c) \pmod{2^{e-2}}$$

b) 如果 $X_{n+1} = (4c - 1)X_n \pmod{2^e}$, 而且 $X_n = ((-1)^n(4Y_n + 1)) \pmod{2^e}$, 则

$$Y_{n+1} = ((1 - 4c)Y_n - c) \bmod 2^{e-2}$$

[注:在这些公式中, c 是一个奇整数。在文献中有过若干论断,意思是说,满足定理 B 且 $c=0$ 的序列比满足定理 A 的序列更为随机,尽管在定理 B 的情况下它的周期长度只达到四分之一。这一习题反驳了这样的论断;实质上,当 M 是 2 的幂时,人们必须抛弃字长中的两位,以节省对 c 的加法。]

10. [M21] 当 m 为什么值时,有 $\lambda(m) = \varphi(m)$?

► 11. [M28] 设 x 是大于 1 的奇整数。(a) 证明存在惟一的整数 $f > 1$, 使得 $x \equiv 2^f \pm 1 \pmod{2^{f+1}}$ 。(b) 给定 $1 < x < 2^e - 1$, 而 f 是由 (a) 得到的相应的整数, 证明 x 模 2^e 的阶是 2^{e-f} 。(c) 特别是, 这证明了定理 C i) ~ iv)。

12. [M26] 设 p 为一奇素数。如果 $e > 1$, 证明: 当且仅当 a 为以 p 为模的本原元且 $a^{p-1} \not\equiv 1 \pmod{p^2}$ 时, a 也是以 p^e 为模的本原元。(在本题中, 假设 $\lambda(p^e) = p^{e-1}(p-1)$ 。这个事实将在下面的习题 14 和 16 中证明。)

13. [M22] 设 p 是素数。若 a 不是以 p 为模的本原元, 证明: 或者 a 是 p 的倍数, 或者对于整除 $p-1$ 的某个素数 q , $a^{(p-1)/q} \equiv 1 \pmod{p}$ 。

14. [M18] 如果 $e > 1$ 且 p 是一个奇素数, 而且如果 a 是以 p 为模的本原元, 证明 a 或者 $a+p$ 是以 p^e 为模的本原元。[提示: 见习题 12。]

15. [M29] (a) 设 a_1, a_2 与 m 互素, 并令它们模 m 的阶分别是 λ_1 和 λ_2 。如果 λ 是 λ_1 和 λ_2 的最小公倍数, 证明: 对于适当的整数 κ_1 和 κ_2 , $a_1^{\kappa_1} a_2^{\kappa_2}$ 有模 m 的阶 λ 。[提示: 首先考虑 λ_1 与 λ_2 互素的情况。] (b) 设 $\lambda(m)$ 是任何元素的模 m 极大阶, 证明 $\lambda(m)$ 是每个元素模 m 的阶的倍数, 即证明每当 a 与 m 互素时, $a^{\lambda(m)} \equiv 1 \pmod{m}$ 。(不要使用定理 B。)

► 16. [M24] (原根的存在性) 设 p 是一素数。

a) 考虑多项式 $f(x) = x^n + c_1 x^{n-1} + \cdots + c_n$, 其中诸 c 为整数。假定 a 是使得 $f(a) \equiv 0 \pmod{p}$ 的整数, 证明存在一个整系数的多项式 $q(x) = x^{n-1} + q_1 x^{n-2} + \cdots + q_{n-1}$, 使得对所有整数 x , $f(x) \equiv (x-a)q(x) \pmod{p}$ 。

b) 设 $f(x)$ 是如同 a) 中那样的多项式, 证明 $f(x)$ 至多有 n 个模 p 的不同的“根”; 即, 至多有 n 个整数 a ($0 \leq a < p$), 使得 $f(a) \equiv 0 \pmod{p}$ 。

c) 由习题 15(b), 多项式 $f(x) = x^{\lambda(p)} - 1$ 有 $p-1$ 个不同的根, 因此有一个阶为 $p-1$ 的整数 a 。

17. [M26] 并非定理 D 中所列的值全都能通过文中的方法构造出来; 例如, 11 就不是模 5^e 本原的, 但根据定理 D, 11 是模 10^e 的本原元, 试解释为什么会有上述结论? 列在定理 D 中的那些值中, 哪些同时是模 2^e 和 5^e 这两者的本原元?

18. [M25] 证明定理 D (参考上题)。

19. [40] 假定 $c=0$, 试对列在表 3.2.1.1-1 中的每个 m 的值, 制作一张适当的乘数 a 的表。

► 20. [M24] (G. Marsaglia) 本题的目的是研究一任意线性同余序列的周期长度。令 $Y_n = 1 + a + \cdots + a^{n-1}$, 使得对于由等式 3.2.1-(8) 得到的某个常数 A , $X_n \equiv (AY_n + X_0) \bmod m$ 。

a) 证明 $\langle X_n \rangle$ 的周期长度即是 $\langle Y_n \bmod m' \rangle$ 的周期长度, 其中 $m' = m/\gcd(A, m)$ 。

b) 证明当 p 为素数时 $\langle Y_n \bmod p^e \rangle$ 的周期长度满足下列条件: (i) 若 $a \bmod p = 0$, 则它为 1。(ii) 若 $a \bmod p = 1$, 则它为 p^e , 但 $p=2$ 和 $e \geq 2$ 及 $a \bmod 4 = 3$ 时例外。(iii) 若 $p=2, e \geq 2$ 及 $a \bmod 4 = 3$, 则它是 a 模 p^e 阶的两倍 (参考习题 11), 但 $a \equiv -1 \pmod{2^e}$ 时例外, 那时它为 2。(iv) 若 $a \bmod p > 1$, 则它为 a 模 p^e 的阶。

21. [M25] 在极大周期的一个线性同余序列中, 令 $X_0 = 0$ 和令 s 为使得 $a^s \equiv 1 \pmod{m}$ 的最小正整数, 证明 $\gcd(X_s, m) = s$ 。

► 22. [M25] 试讨论求模 $m = b^k \pm b' \pm 1$ 的问题, 使得习题 3.2.1.1-14 的带借位减法和带进位加法生成程序将有很长的周期。

3.2.1.3 效能 在上一节, 我们证明了, 当 $b = a - 1$ 是每一个整除 m 的素数的倍数 (且若 m 是 4 的倍数, 则 b 也是 4 的倍数) 时, 可以达到极大周期。如果 z 是正在使用的机器的进制——对于一台二进制的计算机 $z = 2$, 而对于一台十进制的计算机 $z = 10$ ——而且如果 m 是字长 z^e , 则乘数

$$a = z^k + 1, \quad 2 \leq k < e \quad (1)$$

满足这些条件。定理 3.2.1.2A 也指出, 我们可以取 $c = 1$ 。现在有形如

$$X_{n+1} = ((z^k + 1)X_n + 1) \bmod z^e \quad (2)$$

的递推关系, 而且这个等式提示我们可以避免乘法, 只须使用移位和加法就足够了。

例如, 假设 $a = B^2 + 1$, 其中 B 是 MIX 的字节大小, 则代码

$$\text{LDA } X; \quad \text{SLA } 2; \quad \text{ADD } X; \quad \text{INCA } 1 \quad (3)$$

可用来代替在 3.2.1.1 小节中给出的一串指令, 且执行时间从 $16u$ 减少到 $7u$ 。

由于这个原因, 文献中对于具有形式 (1) 的乘数已广泛地进行了讨论, 而且它们确实已为许多作者所推荐。然而, 早年使用这个方法的经验表明, 应该避免 (1) 中的简单形式的乘数, 它所生成的数并不是很随机的。

在这一章稍后, 我们将讨论某个稍微复杂的理论, 它解释被认为是坏的所有线性同余随机数生成程序为什么是坏的。然而, 某些生成程序 (例如 (2)) 是非常糟的, 一个比较简单的理论就可以把它否定掉。这样的简单理论, 同“效能”的概念有关, 我们现在就要来讨论它。

具有极大周期的线性同余序列的效能被定义为使得

$$b^s \equiv 0 \pmod{m} \quad (4)$$

的最小整数 s (当乘数满足定理 3.2.1.2A 的条件时, 这样一个整数 s 总是存在的, 因为 b 是整除 m 的每个素数的倍数)。

我们可以通过取 $X_0 = 0$ 来分析这个序列的随机性, 因为 0 总要出现于这周期中的某处。在这个假定下, 等式 3.2.1-(6) 简化为 $X_n = ((a^n - 1)c/b) \bmod m$, 而且如果我们用二项式定理展开 $a^n - 1 = (b + 1)^n - 1$, 就得到

$$X_n = c \left(n + \binom{n}{2}b + \cdots + \binom{n}{s}b^{s-1} \right) \bmod m \quad (5)$$

b^s, b^{s+1} 等等的项可忽略不计, 因为它们都是 m 的倍数。

等式 (5) 是有启发性的, 所以我们将考虑某些特殊情况。如果 $a = 1$, 则效能是 1; 而且如我们已注意到的, $X_n \equiv cn \pmod{m}$, 所以这个序列肯定不是随机的。

如果效能为 2, 则有 $X_n \equiv cn + cb \binom{n}{2}$, 这个序列还不是很随机的。其实,

$$X_{n+1} - X_n \equiv c + cbn$$

所以,在这一情况下,连续地生成的数之间的差以一种简单的方式从 n 的一个值变为下一个值。点 (X_n, X_{n+1}, X_{n+2}) 总是处于三维空间中的四个平面之一上:

$$\begin{aligned} x - 2y + z &= d + m, & x - 2y + z &= d - m \\ x - 2y + z &= d, & x - 2y + z &= d - 2m \end{aligned}$$

其中 $d = cb \bmod m$ 。

如果效能等于 3,则这个序列开始时看起来更随机些,但在 X_n, X_{n+1} 和 X_{n+2} 之间有很强的相依性;检验说明,具有效能 3 的序列仍然不是十分好的。当效能为 4 或更大时,已经有关于其合理结果的报道,但其它人对此有不同看法。为了达到充分随机的值,似乎至少需要有效能 5。

例如,假设 $m = 2^{35}$ 和 $a = 2^k + 1$ 。于是 $b = 2^k$, 可见当 $k \geq 18$ 时,值 $b^2 = 2^{2k}$ 是 m 的倍数;效能是 2。如果 $k = 17, 16, \dots, 12$, 则效能是 3, 而对于 $k = 11, 10, 9$, 效能达到 4。因此,从效能的观点看,可取的乘数仅有 $k \leq 8$ 。这意味着 $a \leq 257$ 。稍后我们将看到,也应避免小的乘数。因此当 $m = 2^{35}$ 时,我们已经去掉了所有形如 $2^k + 1$ 的乘数。

当 m 等于 $w \pm 1$ 时(其中 w 是字长), m 一般地不为素数的高次幂所整除,因而高的效能(见习题 6)是不可能的。所以在这种情况下,不应当使用极大周期方法,而应当代之以应用 $c = 0$ 时的纯粹乘法的方法。

必须强调的是,对于随机性说来,高的效能是必要的,但不是充分的。我们使用效能的概念仅仅是为了拒绝不起作用的生成程序,而不是用于接受起作用的生成程序。在线性同余序列的随机性被认为是可接受的之前,它们须通过 3.3.4 小节中讨论的“谱检验”。

习 题

1. [M10] 证明,不管 MIX 的字节大小 B 如何,代码(3)都提供具有极大周期的随机数生成程序。

2. [10] 由 MIX 代码(3)所表示的生成程序的效能是多少?

3. [11] 当 $m = 2^{35}$ 时,具有 $a = 3141592621$ 的线性同余序列的效能是多少? 如果乘数是 $a = 2^{23} + 2^{13} + 2^2 + 1$, 则效能是多少?

4. [15] 试证明:如果 $m = 2^e \geq 8$, 则当 $a \bmod 8 = 5$ 时,可达到极大的效能。

5. [M20] 给定 $m = p_1^{e_1} \cdots p_t^{e_t}$, 且 $a = 1 + kp_1^{f_1} \cdots p_t^{f_t}$, 其中 a 满足定理 3.2.1.2A 的条件, 而且 k 与 m 互素, 试证明效能是 $\max(\lceil e_1/f_1 \rceil, \dots, \lceil e_t/f_t \rceil)$ 。

► 6. [20] 在表 3.2.1.1-1 里 $m = w \pm 1$ 的值中, 哪个可用于产生其效能是 4 或更大的极大周期的线性同余序列(利用习题 5 的结果)?

7. [M20] 当 a 满足定理 3.2.1.2A 的条件时, 它与 m 互素; 因此有一个数 a' , 使得 $aa' \equiv 1 \pmod{m}$ 。证明 a' 可以简单地借助于 b 来表示。

► 8. [M26] 由 $X_{n+1} = (2^{17} + 3)X_n \bmod 2^{35}$ 和 $X_0 = 1$ 所定义的一个随机数生成程序受到如下的检验: 设 $Y_n = \lfloor 10X_n/2^{35} \rfloor$; 则 Y_n 应是在 0-9 之间的一个随机数字, 且三元组 $(Y_{3n}, Y_{3n+1}, Y_{3n+2})$ 应该以相同的概率来取从 $(0, 0, 0) \sim (9, 9, 9)$ 这 1000 个可能值的每一个值。但用 30000

个 n 的值进行检验时,某些三元组几乎不出现,而其它的三元组则经常比它们应有的出现次数要多。你能说明这个失误的原因何在吗?

3.2.2 其它方法

当然,线性同余序列并不是被提出来供计算机使用的随机数的惟一来源。在这一小节里,我们将考察其它一些最有意义的方法;其中有些是十分重要的,而另一些之所以引起我们的兴趣,主要是由于它们不像人们期望的那么好。

同随机数生成相关联常见的谬误之一,是认为:为了得到一个“甚至更随机”的序列,我们可以采取一个好的生成程序并对它略事修改。这往往是不正确的。例如,已知

$$X_{n+1} = (aX_n + c) \bmod m \quad (1)$$

导致相当好的随机数,那么,由

$$X_{n+1} = ((aX_n) \bmod (m+1) + c) \bmod m \quad (2)$$

产生的序列是否更随机些呢?回答是,这个新的序列可能大大减少了随机性。因为整个理论破坏了,而且在缺乏任何关于序列(2)的特性的理论的情况下,我们进入了 $X_{n+1} = f(X_n)$ 类型的生成程序的领域,其中函数 f 是随机地选择的;习题 3.1-11~3.1-15 说明,这些序列的特性大概都比由更受限制的函数(1)得到的序列差得多。

我们来考虑另一种方法,以图获得对序列(1)的一个真正的改进。线性同余方法可以被推广成(比如说)二次同余方法:

$$X_{n+1} = (dX_n^2 + aX_n + c) \bmod m \quad (3)$$

习题 8 推广了定理 3.2.1.2A,以得到有关 a, c 和 d 的必要和充分条件,这些条件使由(3)确定的序列的周期达到极大长度 m ,这些限制并不比线性方法更严。

R. R. Coveyou 提出了当 m 是 2 的幂时的一个有趣的二次方法:设

$$X_0 \bmod 4 = 2, \quad X_{n+1} = X_n(X_n + 1) \bmod 2^e, n \geq 0 \quad (4)$$

这个序列大约可以像(1)一样有效地来计算,而无须担忧会产生任何溢出。它同冯诺伊曼的原始平方取中法有着有趣的联系。如果设 Y_n 为 $2^e X_n$, 即 Y_n 是通过在 X_n 的二进表示的右边放上 e 个 0 而得到的一个双精度数,于是 Y_{n+1} 恰好是由 $Y_n^2 + 2^e Y_n$ 的中间 $2e$ 个字组成!换句话说, Coveyou 方法几乎等同于一个稍微退化的双精度平方取中法,而且它保证有长周期,在习题 8 的答案中所引用的 Coveyou 的论文中证明了它的随机性进一步的证据。

还可考虑等式(1)的其它推广。例如,我们可以试图扩展这个序列的周期长度。一个线性同余序列的周期是相当长的;当 m 近似于计算机的字长时,通常得到 10^9 阶的或更高阶的周期,因而在计算时通常仅仅使用这个序列非常小的一部分。另一方面,在 3.3.4 小节中讨论有关精确度的思想时,我们将看到,周期长度影响着一个序列可以达到的随机性程度。因此,有时希望探求一个更长的周期。有许多方法都可做到这一点。有一项技术是让 X_{n+1} 依赖于 X_n 和 X_{n-1} , 而不仅仅依赖于 X_n ; 这

样,周期的长度可以高达 m^2 。因为这个序列在 $(X_{n+\lambda}, X_{n+\lambda+1}) = (X_n, X_{n+1})$ 之前不会开始重复。John Mauchly 在 1949 年提交给一个统计会议而未发表的一篇论文中,使用递推关系 X_n 等于 $(X_{n-1} \cdot X_{n-6})$ 取中,推广了平方取中方法。

X_{n+1} 依赖于前边一个以上的值的最简单序列是斐波那契(Fibonacci)序列

$$X_{n+1} = (X_n + X_{n-1}) \bmod m \quad (5)$$

人们在 20 世纪 50 年代初就考虑过这个生成程序,而且它给出的周期长度常常大于 m ;但是检验表明,由斐波那契递推关系(5)所产生的数的随机性肯定不能令人满意,所以现在对以关系(5)作为随机数来源的主要兴趣在于,它构成了一个很好的“坏例子”。我们还可以考虑当 k 有比较大的值时,形如

$$X_{n+1} = (X_n + X_{n-k}) \bmod m \quad (6)$$

的生成程序。Green, Smith 和 Klem 对此作过介绍[JACM 6 (1959), 527~537],他们报告说,当 $k \leq 15$ 时,这个序列没能通过 3.3.2 小节所述的间隔检验,尽管当 $k = 16$ 时,检验是令人满意的。

1958 年 G. J. Mitchell 和 D. P. Moore 想出了一个好得多的加法生成程序(未发表),他们提出由

$$X_n = (X_{n-24} + X_{n-55}) \bmod m, \quad n \geq 55 \quad (7)$$

定义的稍微不寻常的序列,其中 m 为偶数, X_0, \dots, X_{54} 是不全为偶数的任意整数。在这个定义中,常数 24 和 55 不是随机地选择的,它们是特殊的值,它们碰巧定义了最低有效二进位值 $\langle X_n \bmod 2 \rangle$ 的周期长度正好是 $2^{55} - 1$ 的一个序列。因此,序列 $\langle X_n \rangle$ 必定有至少这样长的一个周期。习题 30 说明,当 $m = 2^e$ 时,式(7)有长度为 $2^{e-1}(2^{55} - 1)$ 的周期。

乍一看,等式(7)似乎对丁机器实现极不适合,但事实上使用一个循环表就可非常有效地生成这个序列。

算法 A(加数生成程序) 开始时,内存单元 $Y[1], Y[2], \dots, Y[55]$ 分别置成值 $X_{54}, X_{53}, \dots, X_0$, j 开始时等于 24, k 开始时等于 55。本算法的逐次实施将产生数 X_{55}, X_{56}, \dots 作为输出。

A1. [加] (如果这时我们正要输出 X_n , $Y[j]$ 现在等于 X_{n-24} , $Y[k]$ 等于 X_{n-55} 。)置 $Y[k] \leftarrow (Y[k] + Y[j]) \bmod 2^e$, 并输出 $Y[k]$ 。

A2. [前进] j 和 k 减 1。如果现在 $j = 0$, 则置 $j \leftarrow 55$, 否则如果 $k = 0$, 则置 $k \leftarrow 55$ 。 ─

在 MIX 中这个算法简单地写为如下的程序 A。

程序 A(加数生成程序) 假定变址寄存器 5 和 6 代表 j 和 k , 都不为包含这个子程序的程序的其余部分所触及, 下面的代码实施算法 A 并把结果留在寄存器 A 中。

LDA Y, 6 A1. 加

ADD	Y, 5	$Y_k + Y_j$ (可能溢出)
STA	Y, 6	$\rightarrow Y_k$
DEC5	1	A2. 前进。 $j \leftarrow j - 1$
DEC6	1	$k \leftarrow k - 1$
J5P	* + 2	
ENT5	55	如果 $j = 0$, 则置 $j \leftarrow 55$
J6P	* + 2	
ENT6	55	如果 $k = 0$, 则置 $k \leftarrow 55$ ┃

这个生成程序通常比我们前边讨论过的方法要快些,因为它不要求任何乘法。除了速度外,它还具有我们见过的最长的周期,但在习题 3.2.1.2-22 中例外。其次,如 Richard Brent 已经发现的,可以把它弄成对于浮点数也能正确地工作,而避免在整数和分数之间的转换(见习题 23)。因此,在实际应用中它很可能被证明是非常好的随机数来源。难以毫无保留地推荐序列(7)的主要原因是,只有很少理论来证明它们具有还是不具有所需的随机性质;实际上,我们所确实知道的是这个周期非常长,而这是不够的。然而 John Reiser(斯坦福大学博士论文,1977)已经证明,假定某个似真的猜测为真,则像等式(7)这样的加法序列将在高维中很好地分布(请见习题 26)。

(7)中的数(24,55)普通称做延搁(lag),由(7)所定义的数 X_n 构成了延搁的斐波那契序列。以下某些习题中提供的理论结果证实,像(24,55)这样的延搁很有效。当然,当一个应用碰巧在一个时刻使用比如说 55 个值的组,则使用稍微更大的延搁要更好些;由(7)所生成的数将肯定没有严格地处于 X_{n-24} 和 X_{n-55} 之间的 X_n (见习题 2)。J.-M. Normand, H. J. Herrmann 和 M. Hajjar 在进行要求 10^{11} 个随机数的高精度蒙特卡罗方法的详尽研究时发现,由(7)所生成的数的稍微偏离 [J. Statistical Physics 52 (1988), 441 ~ 446]; 但是很大的 k 值减少了坏的效果。表 1 列出了若干有用的 (l, k) 偶。对于这些值偶来说, $X_n = (X_{n-l} + X_{n-k}) \bmod 2^e$ 有周期长度 $2^{e-1} (2^k - 1)$ 。对于大多数应用说来, $(l, k) = (30, 127)$ 的情况应是足够大了,特别是和我们后面将要讨论的其它增强随机性技术相结合之后。

表 1 产生长周期模 2 的延搁

(24,55)	(37,100)	(83,258)	(273,607)	(576,3217)	(7083,19937)
(38,89)	(30,127)	(107,378)	(1029,2281)	(4187,9689)	(9739,23209)

对于本表的扩充,请见 N. Zierler 和 J. Brillhart, *Information and Control* 13 (1968), 541 ~ 554, 14 (1969), 566 ~ 569, 15 (1969), 67 ~ 69; 栗田良春 (Y. Kurita) 和 松本真 (M. Matsumoto), *Math. Comp.* 56 (1991), 817 ~ 821; Heringa, Blöte 和 Compagner, *Int. J. Mod. Phys. C3* (1992), 561 ~ 564

George Marsaglia [Comp. Sci. and Statistics: Symposium on the Interface 16 (1984), 3 ~ 10] 建议以

$$X_n = (X_{n-24} + X_{n-55}) \bmod m, \quad n \geq 5 \quad (7')$$

来代替(7)。其中 m 是 4 的倍数,而 X_0 到 X_{54} 是奇数,不全同余于 1 (模 4)。于是

第二最低有效位有 $2^{55} - 1$ 的周期,而最高有效位比以前更彻底地混杂在一起。因为它们以一种基本方式依赖于 X_{n-24} 和 X_{n-55} 的所有二进位。习题 31 表明序列(7')的周期长度只是比(7)的周期长度稍短一点点。

自 1958 年以来延搁的斐波那契生成程序已被成功地使用于许多情况,因此当在 1990 年发现它们实际上不能通过对于随机性的一个极其简单的非针对性的检验时,确实引起了震动(请见习题 3.3.2-31)。在本小节接近结尾处描述了通过丢弃序列的适当元素而避免这样的问题的解决方案。

不用纯粹的加法和纯粹的乘法的序列,我们可以通过对小的 k 取 X_{n-1}, \dots, X_{n-k} 的一般线性组合来构造有用的随机数生成程序。在这种情况下当模 m 是很大的素数时出现最好的结果:例如, m 可以选择成可装入一个计算机字的最大素数(见表 4.5.4-2)。当 $m = p$ 是素数时,有限域理论告诉我们,有可能来求乘数 a_1, \dots, a_k , 使得

$$X_n = (a_1 X_{n-1} + \dots + a_k X_{n-k}) \bmod p \quad (8)$$

所定义的序列有周期长度 $p^k - 1$; 这里 X_0, \dots, X_{k-1} 可以任意选择但不全为 0 (特殊情况 $k=1$ 对应于一个具有素数模的乘法同余序列,这是我们已熟悉的)。当且仅当多项式

$$f(x) = x^k - a_1 x^{k-1} - \dots - a_k \quad (9)$$

是一个“模 p 本原多项式”时,即,当且仅当这个多项式以有 p^k 个元素的域中的一个本原元作为根时,式(8)中常数 a_1, \dots, a_k 有所希望的性质(见习题 4.6.2-16)。

当然,对于实际应用来说,仅仅假定有适当的常数 a_1, \dots, a_k 使周期长度为 $p^k - 1$, 这是不够的;我们必须有能力来找到它们,而且我们不能简单地尝试所有 p^k 个可能性,因为 p 有计算机字长那么大的阶。幸而, (a_1, \dots, a_k) 恰有 $\varphi(p^k - 1)/k$ 种适当的选择,所以在做了一些随机的试验之后,有相当好的机会选中一个。但是我们也需要有一种方法能迅速地告知(9)是否是一个模 p 本原多项式;逐个生成这个序列的 $p^k - 1$ 个元素并等候出现重复肯定是不可想像的! 在 Sankhyā A26 (1964), 305~328 中, Alanen 和 Knuth 讨论了检验模 p 本原性的一些方法;可以使用以下的准则:令 $r = (p^k - 1)/(p - 1)$ 。

i) $(-1)^{k-1} a_k$ 必定是模 p 原根(参考 3.2.1.2 小节)。

ii) 多项式 x^r 必然同余于 $(-1)^{k-1} a_k$, 模 $f(x)$ 和 p 。

iii) 利用模 p 多项式算术,对于 r 的每个本原因子 q , $x^{r/q} \bmod f(x)$ 的阶必定为正数。

4.6.2 小节中讨论了利用模给定素数 p 多项式算术,计算多项式 $x^n \bmod f(x)$ 的有效方法。

为了进行这个检验,需要知道 $r = (p^k - 1)/(p - 1)$ 的素因子分解,而在计算中这是一个限制因素;当 $k=2, 3$ 或许 4 时, r 可在相当时间内被分解,但当 p 充分大时,更高的 k 值就难于处理了。 $k=2$ 实际上已把 $k=1$ 可以达到的“有效随机数字”

个数增加一倍,所以很少有必要取更大的 k 值。

修改后的谱检验(3.3.4 小节)可用来估价由(8)生成的数的序列;见习题 3.3.4-24。该节的考虑表明,当该形式的本原多项式存在时,我们不应做 $a_1 = +1$ 或 -1 这样明显的选择;最好挑选大的、实质上“随机的”满足条件的 a_1, \dots, a_k 的值,并应用谱检验来验证这个选择。求 a_1, \dots, a_k 涉及相当大的计算量,但所有已知证据表明,其结果将是随机数的非常令人满意的来源。我们实质上只用单精度运算就实现了有 k 倍精度的一个线性同余生成程序的随机性。

特殊情况 $p=2$ 有独特的意义。有时,希望有这样一种随机数生成程序,它仅仅产生一些二进位——一些 0 和一些 1——的随机序列,而不是一个介于 0 和 1 之间的小数。有一个简单的办法,通过 k 个二进位字的操作来生成一台二进计算机上的高度随机的二进序列:从一个任意的非 0 二进字 x 开始。为了得到这个序列的下一个随机位,进行(以 MIX 语言表示的)下列操作(见习题 16):

LDA X	(假设现在溢出开关是“关闭”的)	
ADD X	左移一位	
JNOV *+2	如果高位原来为 0 则转移	(10)
XOR A	否则以“异或”来调整这个数	
STA X	↓	

这里第四条指令是“异或”操作,几乎所有的二进计算机上都具备(参考习题 2.5-28 和 7.1 节);它改变 rA 中与单元 A 中每一个为“1”的位对应的位的状态。单元 A 中的值是二进常数 $(a_1 \dots a_k)_2$, 这里 $x^k - a_1 x^{k-1} - \dots - a_k$ 是如上所说的模 2 的本原多项式。在执行代码(10)后,所生成的序列的下一位可取作 x 的最低位,或者也可以总使用 x 的最高位,如果最高位更方便的话。

例如,考虑图 1,它说明了对于 $k=4$, $\text{CONTENTS}(A) = (0011)_2$ 生成的序列。当然,这里 k 的值非常小。右边的列表示这个序列的位顺序,它以长度为 $2^k - 1 = 15$ 的周期重复,即 1101011110001001…。考虑到它是仅由内存的四个二进位生成的,这个序列是十分随机的。为了看出这一点,考虑出现于周期中的相邻四位组,即 1101, 1010, 0101, 1011, 0111, 1111, 1110, 1100, 1000, 0001, 0010, 0100, 1001, 0011, 0110。因为周期的长度恰为 $2^k - 1$,一般地说,除了全 0 的组之外,每个可能的相邻 k 位组在周期中都恰好出现一次;所以相邻的 k 位组

1011
0101
1010
0111
1110
1111
1101
1001
0001
0010
0100
1000
0011
0110
1100
1011

图 1 在二进方法中计算机字 x 的逐个的内容,假设 $k=4$ 且 $\text{CONTENTS}(A) = (0011)_2$

实质上是独立的。我们将在 3.5 节中看到, 当 k 为 30 或更大时, 这是一个非常强的随机性准则。R. C. Tausworthe 在一篇论文中从理论上说明了这个序列的随机性 [Math. Comp. 19 (1965), 201 ~ 209]。

W. Stahnke 编制了模 2 的次数 ≤ 168 的本原多项式表 [Math. Comp. 27 (1973), 977~980]。当 $k=35$ 时, 我们可以取

[illegible]

但习题 18 和 3.3.4-24 告诉我们,最好还是找出定义模 2 本原多项式的实质上“随机”的常数。

正告:若干人曾经误认为本随机位生成技术可以用来生成随机的全字长的小数, $(.X_0X_1\cdots X_{k-1})_2, (.X_kX_{k+1}\cdots X_{2k-1})_2, \cdots$;但这实际上是随机小数的一个拙劣的来源,尽管这些位个别地是十分随机的!习题 18 说明了为什么如此。

Mitchell 和 Moore 的加法生成程序(7)实质上是以本原多项式的概念为基础的;多项式 $x^{55} + x^{24} + 1$ 是本原的,而且表 1 实质上是所有模 2 本原三项式的一张表。几乎和 Mitchell 和 Moore 的这个生成程序相同的生成程序独立地为 T. G. Lewis 和 W. H. Payne 于 1971 年所发现[JACM 20 (1973), 456~468],但用的是“异或”而不是加法,使得周期恰为 $2^{55} - 1$ 。在他们的序列中,每个二进位都跑遍同样的周期序列,但有各自的起始点。经验表明,(7)给出更好的结果。

我们现在已经看到,当 X_n 是 X_{n-1}, \dots, X_{n-k} 的适当函数和当 m 是素数时,可以不太费劲就构造出 $0 \leq X_n \leq m$ 且周期为 $m^k - 1$ 的序列。不难看出,由形如

$$X_n = f(X_{n-1}, \dots, X_{n-k}), \quad 0 \leq X_n < m \quad (11)$$

的关系定义的任何序列的最高周期是 m^k 。M. H. Martin [*Bull. Amer. Math. Soc.* **40** (1934), 859~864] 首先证明了: 对于所有的 m 和 k , 达到这个极大周期的函数都是可能的。他的方法容易叙述 (见习题 17), 而且对于编写程序也相当有效 (见习题 29), 但不适合于随机数的生成, 因为它非常缓慢地改变 $X_{n-1} + \dots + X_{n-k}$ 的值: 所有的 k 元组都出现, 但并非以很随机的顺序。习题 21 考虑了产生极大周期的更好的函数 f 的类。一般说来, 相应的程序不如我们已经描述过的其它方法那样有效地生成随机数。但是如果整体来考虑周期, 它们确实具有很明显的随机性。

随机数生成的许多其它方案已经被提了出来。在这些可选择的方法中最让人感兴趣的可能是由 Eichenauer 和 Lehn 所提出的逆同余序列 [Statistische Hefte 27 (1986), 315~326]

$$X_{v+1} = (aX_v^{-1} + c) \bmod p \quad (12)$$

这里 p 是素数, X_n 取遍集合 $\{0, 1, \dots, p-1, \infty\}$, 而逆由 $0^{-1} = \infty, \infty^{-1} = 0$, 否则 $X^{-1}X \equiv 1 \pmod{p}$ 来定义。由于在这个序列中 0 后面总是跟随着 ∞ 而后跟着 c , 因此从实现的目的考虑我们可以简单地定义 $0^{-1} = 0$; 但当 $0^{-1} = \infty$ 时在理论上更清晰也更易于发展。适合于硬件实现的有效算法可利用来计算 $X^{-1} \pmod{p}$; 比如说, 请见习题 4.5.2-39。但不幸的是这个运算不是大多数计算机的保留节目。习题 35 说明, a 和 c 的许多选择产生长度为 $p+1$ 的极大周期。习题 37 显示了重要的性

质:逆同余序列完全没有格(lattice)结构,而格结构正是线性同余序列的特征。

另一类重要的技术涉及随机数生成程序的组合。总有那样一些人,感到线性同余方法,加法方法,等等,都太简单了,给不出充分随机的序列;而且也许永远无法证明,他们的怀疑是不正确的——确实,他们可能是对的——因此争论这一点确实毫无用处。有相当有效的方法来把两个序列组合成第三个序列来。它应是足够杂乱的,几乎能令最坚定的怀疑论者满意。

假设我们有在 0 和 $m-1$ 之间的两个随机数序列 X_0, X_1, \dots 和 Y_0, Y_1, \dots , 最好是由两个无关的方法生成。于是,如同 M. D. MacLam 和 G. Marsaglia 所建议的那样[JACM 12 (1965), 83~89; 也可见 Marsaglia 和 Bray, CACM 11 (1968), 757~759], 我们可以用一个随机序列来排列另一个的元素。

算法 M(通过洗牌实现随机化) 给定生成两个序列 $\langle X_n \rangle$ 和 $\langle Y_n \rangle$ 的方法, 本算法将逐个地输出“颇为更随机”的序列的项。我们使用一张辅助表 $V[0], V[1], \dots, V[k-1]$, 其中 k 为某个适当选择的数, 通常在 100 左右。开始时, 以 X 序列的头 k 个值来填 V 表。

M1. [生成 X, Y] 置 X 和 Y 分别等于序列 $\langle X_n \rangle$ 和 $\langle Y_n \rangle$ 的下一项。

M2. [抽取 j] 置 $j \leftarrow \lfloor kY/m \rfloor$, 其中 m 是用于序列 $\langle Y_n \rangle$ 中的模, 即 j 是一随机值, $0 \leq j < k$, 它由 Y 确定。

M3. [交换] 输出 $V[j]$ 而后置 $V[j] \leftarrow X$ 。 ─

作为一个例子, 假定把算法 M 应用于以下两个序列, 且 $k=64$ 。

$$\begin{aligned} X_0 &= 5772156649 & X_{n+1} &= (3141592653 X_n + 2718281829) \bmod 2^{35} \\ Y_0 &= 1781072418 & Y_{n+1} &= (2718281829 Y_n + 3141592653) \bmod 2^{35} \end{aligned} \quad (13)$$

凭直觉, 看来可以保险地预测, 通过应用算法 M 于(13)上所得到的序列实际上将满足任何人对于计算机生成的序列的随机性要求, 因为输出的邻近项之间的关系几乎完全被抹掉了。而且为生成这个序列所需要的时间仅仅比单独生成序列 $\langle X_n \rangle$ 所花时间的两倍略多一些而已。

习题 15 证明: 算法 M 输出的周期长度, 从实用考虑, 在大多数情况下, 将是 $\langle X_n \rangle$ 和 $\langle Y_n \rangle$ 的周期长度的最小公倍数。特别是, 如果当值 0 出现在 Y 序列中我们就拒绝它, 使得 $\langle Y_n \rangle$ 有 $2^{35}-1$ 的周期长度, 则从(13)通过应用算法 M 生成的数有长度为 $2^{70}-2^{35}$ 的周期[请见 J. Arthur Greenwood, *Comp. Sci. and Statistics: Symp. on the Interface* 9 (1976), 222]。

然而, 有一个更好的方法来洗一个序列的元素, 它是由 Carter Bays 和 S. D. Durham 发现的[ACM Trans. Math. Software 2 (1976), 59~64]。他们的方法尽管看起来同算法 M 十分类似, 而且只要求一个输入序列 $\langle X_n \rangle$ 而不是两个, 但是可给出令人惊异的更好的性能。

算法 B(通过洗牌实现随机化) 给定生成序列 $\langle X_n \rangle$ 的方法, 这个算法像算法 M

一样利用一个辅助表 $V[0], V[1], \dots, V[k-1]$, 逐个地输出“颇为更随机的”序列的项。开始时, 用 X 序列的头 k 个值来填满 V 表, 而辅助变量 Y 置成等于第 $k+1$ 个值。

B1. [抽取 j] 置 $j \leftarrow \lfloor kY/m \rfloor$, 其中 m 是用于序列 $\langle X_n \rangle$ 的模, 即 j 是由 Y 确定的一个随机值, $0 \leq j < k$ 。

B2. [交换] 置 $Y \leftarrow V[j]$, 输出 Y , 而后置 $V[j]$ 为序列 $\langle X_n \rangle$ 的下一元素。 **I**

做一下习题 3 和习题 5, 以便对算法 M 和算法 B 之间的差别有一些感性认识。

通过取 k 等于字节大小, 在 MIX 上我们可以实现算法 B。一旦已经完成了初始化, 即可得到以下简单的生成方案:

LD6	Y(1:1)	$j \leftarrow Y$ 的高阶字节	
LDA	X	$rA \leftarrow X_n$	
INCA	1	(参考习题 3.2.1.1-1)	
MUL	A	$rX \leftarrow X_{n+1}$	(14)
STX	X	$n \leftarrow n+1$	
LDA	V, 6		
STA	Y	$Y \leftarrow V[j]$	
STX	V, 6	$V[j] \leftarrow X_n$	I

输出在寄存器 A 中。注意, 对于每个生成的数, 算法 B 仅要求四条指令的开销。

F. Gebhardt [Math. Comp. 21 (1967), 708~709] 发现, 即使把算法 M 应用于像斐波那契序列这样非随机的序列, 并取 $X_n = F_{2n} \bmod m$ 和 $Y_n = F_{2n+1} \bmod m$, 算法 M 也可产生出令人满意的随机序列来。然而, 如果像习题 3 中所示那样, $\langle X_n \rangle$ 和 $\langle Y_n \rangle$ 是密切相关的话, 算法 M 也可能产生出不如原来序列那样随机的序列。对于算法 B 似乎不会引起这样的问题。因为算法 B 不使任何序列减少随机性, 而且由于它可能以非常小的额外代价增强随机性, 故可以推荐它和任何其它的随机数生成程序组合在一起使用。

然而洗牌的方法有一个固有的缺点: 它们只改变所生成数的顺序, 而不是数的本身。对于大多数的目的而言, 顺序是关键的事, 但如果一个随机数生成程序不能通过在 3.3.2 小节中讨论的“生日间隔”测试或者习题 3.3.2-31 的随机漫步测试, 那么在洗牌之后, 其际遇也不会更好。洗牌还有一个相对的缺点, 即它不允许我们在周期中的一个给定位置开始, 或者对于很大的 k , 从 X_n 迅速地跳到 X_{n+k} 。

因此许多人已经建议以简单得多的方式来组合两个序列 $\langle X_n \rangle$ 和 $\langle Y_n \rangle$ 。它可以避免洗牌方法的这两个缺点: 当 $0 \leq X_n < m$ 且 $0 \leq Y_n < m' \leq m$ 时, 我们可以使用像

$$Z_n = (X_n + Y_n) \bmod m \quad (15)$$

这样的组合。习题 13 和 14 讨论了这样的序列的周期长度; 习题 3.3.2-23 证明, 当

种子(初始值) X_0 和 Y_0 独立地选定时,(15)趋向于增强随机性。

去掉算术生成数的结构性偏倚的一个甚至更为简单的方法,在计算的早期岁月就已由 J. Todd 和 O. Taussky Todd 提出[*Symp. on Monte Carlo Methods* (Wiley, 1956), 15~28]:我们能够把序列中的某些数扔掉。他们的建议对于线性同余生成程序用途不大,但同像(7)这样有极长的周期的生成程序相关联,就变得十分合适了,因为我们可丢弃大量的数。

改进(7)的随机性的最简单的方法是对于某个小的 j ,仅使用每第 j 项。但是一个更好的方案,它甚至也更简单,是使用(7)来产生,比如说,在一个数组中的500个随机数,而且仅仅用它们中的头55个。在消耗了这55个数之后,我们用同样的方法再产生500个。这个思想是在动态系统中的混沌理论的激励之下,由 Martin Lüscher 提出的[*Computer Physics Communications* 79 (1994), 100~110]。我们可以认为(7)是把55个值 (X_n, \dots, X_{n+54}) 映射到55个值 $(X_{n+55}, \dots, X_{n+109})$ 的另一个向量的过程。假定生成了 $t \geq 55$ 个值并使用其中的头55个;那么如果 $t = 55$,则这个新的值向量稍微接近于旧的,但如果 $t \approx 500$,则在旧的和新的之间几乎就没有关联了(见习题33)。事实上,对于带进位加和带借位减的生成程序的类似情况(习题3.2.1.1-14),这些向量已知为一个线性同余生成程序中的数的 b 进表示,而且在某个时刻当我们生成 t 个数时相关的乘数是 b^{-t} 。关于这一情况的 Lüscher 理论因此可通过3.3.4小节的谱检验得到证实。基于经 Lüscher 方法增强了的斐波那契延搁系统产生的一个可移植的随机数生成程序,出现于3.6节,并附有进一步的评述一起给出。

随机数生成程序一般只要做少量的乘法和/或加法就能从序列的一个元素得到下一个。当把这样的生成程序同上面建议的组合在一起时,普通常识告诉我们,得到的序列应当是同真正的随机数没有区别的。但是直观的预感不能代替严格的数学证明。如果我们愿意做更多的工作——比如说,是现在工作的1000倍或1000000倍那么多——我们就可以得到这样的序列,对于它们,可以有对随机性好得多的理论保证。

例如,考虑由

$$X_{n+1} = X_n^2 \bmod M \quad B_n = X_n \bmod 2 \quad (16)$$

生成的二进位 B_1, B_2, \dots 的序列[Blum 和 Shub, *SICOMP* 15 (1986), 364~383],或者由

$$X_{n+1} = X_n^2 \bmod M \quad B_n = X_n \cdot Z \bmod 2 \quad (17)$$

生成的更复杂的序列,其中 r 位二进数 $(x_{r-1} \dots x_0)_2$ 和 $(z_{r-1} \dots z_0)_2$ 的点乘乘积是 $x_{r-1}z_{r-1} + \dots + x_0z_0$;这里 Z 是一个 r 位的“屏蔽”(mask),而且 r 是 M 中的二进位数。模 M 应是形如 $4k+3$ 的两个很大素数的乘积。而且起始值 x_0 应该同 M 互素。由 Leonid Levin 所建议的规则(7),是从冯诺伊曼原来的平方取中法出发的;我们称它为平方搅混法(muddle-square method),因为它把平方的二进位搅混。规则(16)当然是 $Z=1$ 的特殊情况。

3.5F 小节包含这样一个证明,即当 X_0, Z 和 M 是随机选取的时,由(16)和(17)所生成的序列通过关于随机性的所有统计检验,所需工作都不会多于对大数的因子分解。换句话说,当 M 足够大时产生的二进位数,今天最快的计算机计算 100 年也算不出其与真正的随机数有什么分别,除非可以找到更快的方法求出这种数的非平凡因子。式(16)比(17)要简单些,但是如果我们要想达到相同的统计保证,则(16)中的模 M 必须比(17)中的更大些才行。

习 题

►1.[12] 在实践中,我们利用 $X_{n+1} = (aX_n + c) \bmod m$ 来形成随机数,其中诸 X 是整数,然后把它们处理成小数 $U_n = X_n/m$ 。 U_n 的递推关系实际上是

$$U_{n+1} = (aU_n + c/m) \bmod 1$$

利用计算机上的浮点算术,试讨论直接利用这个关系的随机序列的生成。

►2.[M20] 一个好的随机数源大约有六分之一的时间有 $X_{n-1} < X_{n+1} < X_n$, 因为 X_{n-1}, X_n 和 X_{n+1} 的六种可能的相对次序的每一种应当是同等可能的。然而,试证,如果使用斐波那契序列(5),则上述次序绝不出现。

3.[23] (a)如果 $X_0 = 0, X_{n+1} = (5X_n + 3) \bmod 8, Y_0 = 0, Y_{n+1} = (5Y_n + 1) \bmod 8$, 且 $k = 4$, 则算法 M 将生成什么序列?(注意,因为效能为 2, 所以 $\langle X_n \rangle$ 和 $\langle Y_n \rangle$ 开始时不是极端随机的。)
(b)如果把算法 B 应用于同一个序列 $\langle X_n \rangle$ 且 $k = 4$, 则会发生什么情况?

4.[00] 为什么用于程序(13)的头一行是最高位字节,而不是其它字节?

►5.[20] 试讨论在算法 M 中利用 $X_n = Y_n$ 来改进生成的速度。结果类似于算法 B 吗?

6.[10] 正文指出,在二进方法(10)中,如果反复地执行这段程序,则 X 的低位是随机的。为什么不是整个字 X 是随机的?

7.[20] 试证:如果把程序(10)改成如下:

```
LDA X          LDA A          JNOV   * + 3          XOR A
JANZ * + 2     ADD X          JAZ    * + 2          STA X    I
```

则可得到长度为 2^e 的全序列(即,在周期中, 2^e 个可能的 e 个相邻位组的每一个,恰恰出现一次)。

8.[M39] 证明:二次同余序列(3)有长度 m 的周期,当且仅当下列条件成立:

i) c 与 m 互素;

ii) 对于所有整除 m 的奇素数 p , d 和 $a - 1$ 都是 p 的倍数;

iii) 如果 m 是 4 的倍数,则 d 是偶数,而且 $d \equiv a - 1 \pmod{4}$; 如果 m 是 2 的倍数,则 $d \equiv a - 1 \pmod{2}$;

iv) 如果 m 是 9 的倍数,则 $d \not\equiv 3c \pmod{9}$ 。

[提示:由 $X_0 = 0, X_{n+1} = dX_n^2 + aX_n + c$ 模 m 定义的序列有长度为 m 的周期,仅当同一个序列模 m 的任何因子 r 有长度为 r 的周期。]

►9.[M24] (R. R. Coveyou) 利用习题 8 的结果证明:修改的平方取中法(4)有长度为 2^{e-2} 的周期。

10.[M29] 证明:如果 X_0 和 X_1 不全为偶数,且 $m = 2^e$, 则斐波那契序列(5)的周期是 $3 \times 2^{e-1}$ 。

11. [M36] 这个习题的目的是分析满足递推关系

$$X_n = a_1 X_{n-1} + \cdots + a_k X_{n-k}, \quad n \geq k$$

的整数序列的若干性质; 当 p 是素数时, 如果我们能计算这个序列模 $m = p^e$ 的周期长度, 则关于任意模 m 的周期长度, 是对于 m 的诸素数幂因子的诸周期长度的最小公倍数。

a) 如果 $f(z), a(z), b(z)$ 是整系数的多项式, 且对于某些整系数多项式 $u(z), v(z)$, 有 $a(z) = b(z) + f(z)u(z) + mv(z)$, 则我们写 $a(z) \equiv b(z) \pmod{f(z) \text{ 和 } m}$ 。证明当 $f(0) = 1$ 和 $p^e > 2$ 时, 下列命题成立: 如果 $z^A \equiv 1 \pmod{f(z) \text{ 和 } p^e}$, $z^B \not\equiv 1 \pmod{f(z) \text{ 和 } p^{e-1}}$, 则 $z^{2A} \equiv 1 \pmod{f(z) \text{ 和 } p^{e+1}}$, $z^{2B} \not\equiv 1 \pmod{f(z) \text{ 和 } p^{e+2}}$ 。

b) 设 $f(z) = 1 - a_1 z - \cdots - a_k z^k$, 且

$$G(z) = 1/f(z) = A_0 + A_1 z + A_2 z^2 + \cdots$$

令 $\lambda(m)$ 表示 $\langle A_n \bmod m \rangle$ 的周期长度, 证明 $\lambda(m)$ 是使得 $z^\lambda \equiv 1 \pmod{f(z) \text{ 和 } m}$ 的最小正整数 λ 。

c) 给定素数 $p, p^e > 2$ 且 $\lambda(p^e) \neq \lambda(p^{e-1})$, 证明对于所有 $r \geq 0, \lambda(p^{e+r}) = p^r \lambda(p^e)$ 。(因此为求序列 $\langle A_n \bmod 2^e \rangle$ 的周期长度, 可以计算 $\lambda(4), \lambda(8), \lambda(16), \dots$, 直到求出使得 $\lambda(2^e) \neq \lambda(4)$ 的不小于 3 的最小 e 值; 于是对于所有的 e , 即可定出模 2^e 周期长度。习题 4.6.3-26 说明当 n 很大时如何用 $O(\log n)$ 步操作计算 X_n 。)

d) 证明: 满足在本题开始时提出的递推关系的任何整数序列, 对于某个整系数多项式 $g(z)$, 有生成函数 $g(z)/f(z)$ 。

e) 设 d) 中的多项式 $f(z)$ 和 $g(z)$ 对于模 p 互素 (参照 4.6.1 节), 证明序列 $\langle X_n \bmod p^e \rangle$ 有和 b) 中的特殊序列 $\langle A_n \bmod p^e \rangle$ 完全相同的周期长度。(不管如何选择 X_0, \dots, X_{k-1} , 都不可能得到更长的周期, 因为一般的序列是特殊序列的“移位”的线性组合。)[提示: 习题 4.6.2-22 (Hensel 引理) 表明: 存在多项式使得 $a(z)f(z) + b(z)g(z) \equiv 1 \pmod{p^e}$ 。]

► 12. [M28] 求整数 X_0, X_1, a, b 和 c , 使得序列

$$X_{n+1} = (aX_n + bX_{n-1} + c) \bmod 2^e, \quad n \geq 1$$

在这种类型的序列中有最长的周期。[提示: 由此得出, $X_{n+2} - ((a+1)X_{n+1} + (b-a)X_n - bX_{n-1}) \bmod 2^e$; 见习题 11c。]

13. [M20] 设 $\langle X_n \rangle$ 和 $\langle Y_n \rangle$ 是整数模 m 的序列, 且周期长度为 λ_1 和 λ_2 。通过令 $Z_n = (X_n + Y_n) \bmod m$ 把两者组合在一起。证明, 如果 λ_1 与 λ_2 互素, 则序列 $\langle Z_n \rangle$ 的周期长度是 $\lambda_1 \lambda_2$ 。

14. [M24] 设 $X_n, Y_n, Z_n, \lambda_1, \lambda_2$ 如上题所述。假定 λ_1 的素因子分解为 $2^{e_2} 3^{e_3} 5^{e_5} \dots$, 类似地, 假设 $\lambda_2 = 2^{f_2} 3^{f_3} 5^{f_5} \dots$, 令 $g_p = (\max(e_p, f_p), \text{如果 } e_p \neq f_p; \text{ 否则为 } 0)$, 而且令 $\lambda_0 = 2^{g_2} 3^{g_3} 5^{g_5} \dots$ 。证明序列 $\langle Z_n \rangle$ 的周期 λ' 是 λ_0 的倍数, 而且它是 $\lambda = \text{lcm}(\lambda_1, \lambda_2)$ 的一个因子。特别是, 如果对于每个素数 $p, e_p \neq f_p$ 或 $e_p = f_p = 0$, 则 $\lambda' = \lambda$ 。

15. [M27] 设算法 M 中的序列 $\langle X_n \rangle$ 有周期长度 λ_1 , 并假定它的周期的所有元素都不同。令 $q_n = \min\{r \mid r > 0 \text{ 且 } \lfloor kY_{n-r}/m \rfloor = \lfloor kY_n/m \rfloor\}$ 。假定对于所有 $n \geq n_0, q_n < \frac{1}{2}\lambda_1$, 且序列 $\langle q_n \rangle$ 有周期长度 λ_2 。设 λ 是 λ_1 和 λ_2 的最小公倍数, 证明由算法 M 产生的输出序列 $\langle Z_n \rangle$ 的周期长度是 λ 。

► 16. [M28] 设在二进记法下, 方法 (10) 中的 $\text{CONTENTS}(A)$ 是 $(a_1 a_2 \dots a_k)_2$ 。证明低阶二进位 X_0, X_1, \dots 的生成序列满足关系

$$X_n = (a_1 X_{n-1} + a_2 X_{n-2} + \cdots + a_k X_{n-k}) \bmod 2$$

[这可以看成是定义序列的另一种方法, 尽管乍一看来, 这个关系式和有效代码 (10) 之间的联系并不明显。]

17. [M33] (M. H. Martin, 1934) 设 m 和 k 为正整数, 并令 $X_1 = X_2 = \cdots = X_k = 0$ 。对于所有 $n > 0$, 令 X_{n+k} 为小于 m 的最大非负值 y , 使得 k 元组 $(X_{n+1}, \cdots, X_{n+k-1}, y)$ 未曾出现于这序列中; 换言之, 对于 $0 \leq r < n$, $(X_{n+1}, \cdots, X_{n+k-1}, y)$ 必须不同于 $(X_{r+1}, \cdots, X_{r+k})$ 。这样, 每个可能的 k 元组在序列中至多出现一次。当我们达到这样一个 n 值, 即对于所有非负的 $y < m$, $(X_{n+1}, \cdots, X_{n+k-1}, y)$ 在这个序列中已经出现, 则这个过程结束。例如, 如果 $m - k = 3$, 则这个序列是

00022212202112102012001110100

而且这个过程结束于此。(a)证明当这个序列终止时,有 $X_{n+k-1} = \cdots = X_{n+k-1} = 0$ 。(b)证明,满足 $0 \leq a_j < m$ 的每个 k 元组 (a_1, a_2, \cdots, a_k) 都出现于序列中;因此当 $n = m^k$ 时,序列终止。[提示:对 s 用归纳法,证明当 $a_s \neq 0$ 时 $(a_1, \cdots, a_s, 0, \cdots, 0)$ 出现。]注意,如果对于 $1 \leq n \leq m^k$, 定义 $f(X_n, \cdots, X_{n+k-1}) = X_{n+k}$, 并置 $X_{m^k+k} = 0$, 则我们得到一个有极大周期的函数。

[illegible]

19. [M41] 对于 4.5.4 小节中表 2 头一列中所确定的每个素数 p , 求出如同正文所建议的适当常数 a_1, a_2 , 使得当 $k=2$ 时, 式 (8) 的周期长度是 p^2-1 。(例子见等式 3.3.4-(39)。)

20. [M40] 对于 $2 \leq k \leq 64$, 试计算适合于用作方法 (10) 中的 $\text{CONTENTS}(A)$ 的常数, 并且要有和那些数差不多一样多的 0 和 1 的个数。

21. [M35] (D. Rees) 正文说明了怎样来求使得序列(11)的周期长度为 $m^k - 1$ 的函数 f , 假定 m 是素数且 X_0, \dots, X_{k-1} 不全为 0。证明: 这样的函数可修改, 以得到对于所有 m , 周期长度为 m^k 的类型(11)的序列。[提示: 考虑习题 7 和 13 的结果, 以及像 $\langle \rho X_{2n} + X_{2n+1} \rangle$ 这样的序列。]

►22.[M24] 正文把推广线性序列(8)的讨论限于 m 是素数的情况。证明当 m “无平方”时, 即 m 是不同素数的乘积时, 也能得到相当长的周期。(检查表 3.2.1.1-1 可以看出, $m = w \pm 1$ 经常满足这个假设; 因此正文的许多结果都可以运用于这种情况, 这对于计算更方便。)

►23.[20] 作为不同于式(7)的一种选择,讨论由 $X_n = (X_{n-55} \quad X_{n-24}) \bmod m$ 所定义的序列。

24. [M20] 设 $0 < l < k$ 。证明: 当由 $Y_n = (Y_{n-l} + Y_{n-k}) \bmod 2$ 定义的序列有周期长度 2^{k-1} 时, 由递推式 $X_n = (X_{n-k+l} + X_{n-k}) \bmod 2$ 所定义的二进制序列也有同样的周期长度。

25.[26] 试讨论程序 A 的一种变例,每当第 55 次要求一个随机数时,它改变 Y 表中的所有 55 个表项。

26. [M48] (J. F. Reiser) 设 p 是素数并设 k 是整数。给定整数 a_1, \dots, a_k 和 x_1, \dots, x_k , 令 λ_a 是由递推式

$$X_n = x_n \bmod p^a, 0 \leq n < k; \quad X_n = (a_1 X_{n-1} + \dots + a_k X_{n-k}) \bmod p^2, n \geq k$$

生成的序列 $\langle X_n \rangle$ 的周期, 并设 N_a 是在这个周期中出现的 0 的个数 (即使得 $\mu_a \leq j \leq \mu_a + \lambda_a$ 且 $X_j = 0$ 的下标 j 的个数), 试证明或否定下列猜测: 对所有 a 和所有 x_1, \dots, x_k , 存在一个常数 c (可能依赖于 p 和 k 及 a_1, \dots, a_k) 使得 $N_a \leq c p^{p^{(k-2)/(k-1)}}$ 。

[注: Reiser 证明, 如果这个递推式有模 p 极大周期长度 (即如果 $\lambda_1 = p^k - 1$), 且如果这个猜测成立, 则当 $\alpha \rightarrow \infty$ 时 $\langle X_n \rangle$ 的 k 维差将是 $O(\alpha^k p^{-\alpha/(k-1)})$ 。于是, 当 $m = 2^e$ 和考虑整个周期时, 像 (7) 这样的加法生成程序将在 55 维中很好地分布 (关于在 k 维中差异的定义见 3.3.4 节)。这个猜测是很弱的情况, 因为如果 $\langle X_n \rangle$ 差不多同样经常地取每个值, 且如果 $\lambda_0 = p^{e-1}(p^k - 1)$, 则当 α 增加时, 量 $N_n \approx (p^k - 1)/p$ 根本不增加。Reiser 对于 $k = 3$ 已经验证了这个猜测。另一方面, 他已经证明, 假定 $\lambda_0 = p^{e-1}(p^k - 1)$ 及 $k \geq 3$ 且 α 充分大, 则有可能找到异常坏的开始值 x_1, \dots, x_k 。

(同 a 有关),使得 $N_{2a} \geq p^a$.]

27. [M30] 假设把算法 B 应用于周期长度为 λ 的一个序列 $\langle X_n \rangle$, 其中 $\lambda \gg k$. 证明对于固定的 k 和所有充分大的 λ , 这个序列的输出最终将是周期的且有相同的周期长度 λ , 除非 $\langle X_n \rangle$ 在开始时不是非常随机的。[提示: 找出 $\lfloor kX_n/m \rfloor$ 的连续值的一个模式, 该模式能引起算法 B“同步”它随后的特性。]

28. [40] (A. G. Waterman) 以 m 为计算机字长的平方或立方, 而 a 和 c 为单精度数, 试实验线性同余序列。

► 29. [40] 只给定 k 元组 (x_1, \dots, x_k) , 试求出计算习题 17 中由 Martin 的序列所定义的函数 $f(x_1, \dots, x_k)$ 的一个好方法。

30. [M37] (R. P. Brent) 令 $f(x) = x^k + a_1 x^{k-1} + \dots + a_k$ 是模 2 本原多项式, 并假设 X_0, \dots, X_{k-1} 是不全为偶数的整数。

a) 证明对于所有 $e \geq 1$, 递推式 $X_n = (a_1 X_{n-1} + \dots + a_k X_{n-k}) \bmod 2^e$ 的周期是 $2^{e-1}(2^k - 1)$, 当且仅当 $f(x)^2 + f(-x)^2 \not\equiv 2f(x^2)$ 和 $f(x)^2 + f(-x)^2 \not\equiv 2(-1)^k f(-x^2) \pmod{8}$ 。[提示: 我们有 $x^{2^k} \equiv -x \pmod{4}$ 和 $f(x)$ 当且仅当 $f(x)^2 + f(-x)^2 \equiv 2f(x^2) \pmod{8}$ 。]

b) 证明当多项式 $f(x) = x^k \pm x^l \pm 1$ 是模 2 本原的且 $k > 2$ 时, 这个条件总成立。

31. [M30] (G. Marsaglia) 当 $m = 2^e \geq 8$ 时序列 (7') 的周期长度是多少? 假设 X_0, \dots, X_{54} 不全都模 8 同余于 ± 1 。

32. [M21] 当 $X_n = (X_{n-24} + X_{n-55}) \bmod m$ 时子序列 $\langle X_{2n} \rangle$ 和 $\langle X_{3n} \rangle$ 的元素满足什么递推式?

► 33. [M23] (a) 令 $g_n(z) = X_{n+30} + X_{n+29}z + \dots + X_{n+30}z^{30} + X_{n+54}z^{31} + \dots + X_{n+31}z^{54}$, 其中诸 X 满足延搁的斐波那契序列 (7)。试求 $g_n(z)$ 和 $g_{n+1}(z)$ 之间的一个简单关系。(b) 借助于 X_0, \dots, X_{54} 来表示 X_{500} 。

34. [M25] 证明逆同余序列 (12) 有 $p+1$ 的周期当且仅当多项式 $f(x) = x^2 - cx - a$ 有以下两个性质: (i) 当以模 p 多项式算术进行计算时, $x^{p+1} \bmod f(x)$ 是一个非零常数; (ii) 对于每一个整除 $p+1$ 的素数 q , $x^{(p+1)/q} \bmod f(x)$ 有度数 1。[提示: 考虑矩阵 $\begin{pmatrix} 0 & 1 \\ a & c \end{pmatrix}$ 的乘幂。]

35. [HM35] 有多少数对 (a, c) 满足习题 34 的条件?

36. [M25] 试证明逆同余序列 $X_{n+1} = (aX_n^{-1} + c) \bmod 2^e$, $X_0 = 1$, $e \geq 1$, 每当 $a \bmod 4 = 1$ 和 $c \bmod 4 = 2$ 时, 有 2^{e-1} 的周期长度。

► 37. [HM32] 令 p 是素数并假定 $X_{n+1} = (aX_n^{-1} + c) \bmod p$ 定义周期为 $p+1$ 的一个逆同余序列。还设 $0 \leq b_1 < \dots < b_d \leq p$ 并考虑集合

$$V = \{(X_{n+b_1}, X_{n+b_2}, \dots, X_{n+b_d}) \mid 0 \leq n \leq p \text{ 且对于 } 1 \leq j \leq d, X_{n+b_j} \neq \infty\}$$

这个集合包含 $p+1-d$ 个向量, 其中的任何 d 个位于某 $(d-1)$ 维超平面 $H = \{(v_1, \dots, v_d) \mid r_1 v_1 + \dots + r_d v_d = r_0 \pmod{p}\}$ 中, 其中 $(r_1, \dots, r_d) \neq (0, \dots, 0)$ 。试证 V 中没有 $d+1$ 个向量位于相同的超平面中。

3.3 统计检验

我们的主要目的是得到一些序列, 它们的特性好像是随机的。至今, 我们已经看到怎样使序列的周期长到在实际应用中绝不出现重复; 这是一个重要的准则, 但

是它并不意味着这样的序列在应用中保证是有用的。那么,如何判定一个序列是否充分地随机呢?

如果给某人笔和纸,要求他写下 100 个随机的十进数字,则他写出令人满意的结果的机会是十分微小的。人们倾向于避免似乎是非随机的一些事情,例如相等的两个相邻数字(尽管每十个数字中大约有一个应等于它的前面的数字)。而且,如果我们给某人一张真正随机数字的表,则他十分可能告诉我们,它们全然不是随机的;他的眼睛将看出某些表面上的规律性。

根据 I. J. Matrix 博士(Martin Gardner 摘录于 *Scientific American*, 1965 年 1 月)的说法,“数学家把 π 的十进制展开当做是随机序列,而对于一个现代数值逻辑学家说来,它有极丰富的值得注意的模式。”例如,Matrix 博士指出,在 π 的展开式中头一次重复的两位数字是 26,而它的第二次出现是在一个奇妙的重复模式中间:

$$3.14159265358979323846264338327950 \quad (1)$$

在列出许多位数字或它们的其它性质之后,人们就会发现,如果正确地解释的话, π 可以反映人类经历的整个历史!

人们都会注意自己的电话号码、执照号码等的模式,以帮助记忆。这些陈述都集中到一点,就是我们不能自信地去判定一个数列是否随机。因此,必须应用某些无偏倚的机械检验。

统计学理论为我们提供了随机性的某些定量的测度。可以想像到的检验多得不胜枚举;我们将讨论那些已经证明是最有用、最有效益和最易于为计算机计算所采用的检验。

如果一个序列对于检验 T_1, T_2, \dots, T_n 有随机特性,那么一般我们不能保证,当施以下一个检验 T_{n+1} 时,它不致惨遭失败;不过,每一次检验都越来越增强我们关于序列的随机性的信心。实践中,我们应把五六个不同类型的统计检验用于一个序列上,如果它都令人满意地通过了这些检验,则我们就认为它是随机的——在证明有罪之前,宜于假定是无辜的。

应该对每一个要广泛使用的序列都进行细心的检验,所以下列各节说明怎样以正确的方式进行这些检验。要区别两种类型的检验:经验检验,即让计算机来处理序列中出现的数组,并计算若干统计量;理论检验,即使用以形成该序列的递推规律为基础的数论方法,来确定序列的特性。

如果根据还不够充分,则读者可以尝试 Darrell Huff 的 *How to Lie With Statistics* (Norton, 1954) 一书中所介绍的一些技术。

3.3.1 研究随机数据的一般检验方法

A. χ^2 检验 χ^2 检验也许是所有统计检验中最著名的,而且在与许多别的检验

一起使用时,它是一个基本的方法。在一般地考虑这个方法之前,先来考虑 χ^2 检验的一个特例,即应用于掷骰子的情形。利用两个“真正的”骰子(假定它们每一个都独立地以同样的概率指示 1, 2, 3, 4, 5 或 6), 以下给出了一次投掷得到给定总和 s 的概率:

$$\begin{array}{cccccccccccc} s \text{ 的值} &= & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ \text{概率, } p_s &= & \frac{1}{36} & \frac{1}{18} & \frac{1}{12} & \frac{1}{9} & \frac{5}{36} & \frac{1}{6} & \frac{5}{36} & \frac{1}{9} & \frac{1}{12} & \frac{1}{18} & \frac{1}{36} \end{array} \quad (1)$$

(例如, 4 的值可以三种方式掷成: $1+3, 2+2, 3+1$; 这组成了 $\frac{3}{36} = \frac{1}{12}$, 即 36 种可能结果中的 p_4 。)

如果掷骰子 n 次, 则平均说来得到值 s 的次数近似于 np_s 。例如, 在 144 次投掷中, 我们将得到值 4 大约 12 次。以下说明了在 144 次投掷的具体序列中真正得到的结果是什么:

$$\begin{array}{cccccccccccc} s \text{ 的值} &= & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ \text{观察的数, } Y_s &= & 2 & 4 & 10 & 12 & 22 & 29 & 21 & 15 & 14 & 9 & 6 \\ \text{期望的数, } np_s &= & 4 & 8 & 12 & 16 & 20 & 24 & 20 & 16 & 12 & 8 & 4 \end{array} \quad (2)$$

注意, 观察的数在所有情况下都不同于期望的数; 事实上, 随机地抛骰子从来难得恰好以正确的概率出现: 掷骰子 144 次, 有 36^{144} 个可能的序列, 它们全都有同等可能。这些序列之一由所有的 2 组成(“对么”, 两点), 而且任何人如果一连串掷出 144 次对么, 则可以确信这骰子装铅了。但如果我们列出每次投掷每个骰子的结果的话, 则全是 2 的序列将同任何特定的序列具有完全相同的可能性。

由此看来如何能检验出给定的骰子有没有装铅呢? 回答是, 不可能做出一个确定的是否判断, 但可以给出一个概率的回答, 即我们可以说某些事件是如何地可能或不可能的。

处理上面的例子的一个相当自然的方式, 是考虑观察数 Y_s 和期望数 np_s 之间差数的平方。我们可以把这些平方加在一起, 得到

$$V = (Y_2 - np_2)^2 + (Y_3 - np_3)^2 + \cdots + (Y_{12} - np_{12})^2 \quad (3)$$

坏的骰子应当得到相对高的 V 值; 而且对于任何给定的 V 值, 我们可以问, “使用真正的骰子时, V 这样高的值的概率是什么?” 如果这个概率很小, 比如说 $\frac{1}{100}$, 我们就知道, 在 100 次中仅有一次将给出同期望数如此背离的结果, 这样我们就完全有了猜疑的理由。(不过请记住, 即使是好的骰子, 100 次中也会有一次给出这样高的 V 值, 所以一个细心的人, 应再次重复这个检验, 看看这个高的 V 值是否再现。)

式(3)中的统计 V 对于 $(Y_7 - np_7)^2$ 和 $(Y_2 - np_2)^2$ 给出相等的数, 尽管 $(Y_7 - np_7)^2$ 可能要比 $(Y_2 - np_2)^2$ 高得多, 因为 7 的出现次数大约是 2 出现次数的 6 倍。

所以,“正确”的统计(至少已证明为最重要的一个统计)将对 $(Y_7 - np_7)^2$ 仅仅赋予相当于 $(Y_2 - np_2)^2$ 的 $\frac{1}{6}$ 重要性,而且我们将把(3)换成下列公式

$$V = \frac{(Y_2 - np_2)^2}{np_2} + \frac{(Y_3 - np_3)^2}{np_3} + \cdots + \frac{(Y_{12} - np_{12})^2}{np_{12}} \quad (4)$$

这就是所谓在这个掷骰子的实验中,观察量 Y_2, \dots, Y_{12} 的 χ^2 统计。对于(2)中的数据,我们得出

$$V = \frac{(2-4)^2}{4} + \frac{(4-8)^2}{8} + \cdots + \frac{(9-8)^2}{8} + \frac{(6-4)^2}{4} = 7 \frac{7}{48} \quad (5)$$

现在重要的问题当然是“ $7 \frac{7}{48}$ 是否是 V 不可能取到的高值呢?”在回答这个问题以前,让我们考虑 χ^2 方法的一般应用。

一般地说,假设每个观察可以落入 k 个范畴之一。我们作 n 个独立的观察。这意味着,一次观察的结果必然对于任何其它次的结果绝无影响。设 p_s 是每次观察落入范畴 s 的概率,并设 Y_s 是真正落入范畴 s 的观察数,于是我们形成统计

$$V = \sum_{s=1}^k \frac{(Y_s - np_s)^2}{np_s} \quad (6)$$

在上面的例子中,每次掷骰子共有 11 种可能的结果,所以 $k=11$ 。[等式(6)是由等式(4)稍作改动得到的,因为我们把可能性编号改为 1 至 k 以代替从 2 至 12。]

展开(6)中的 $(Y_s - np_s)^2 = Y_s^2 - 2np_s Y_s + n^2 p_s^2$, 并利用事实

$$\begin{aligned} Y_1 + Y_2 + \cdots + Y_k &= n \\ p_1 + p_2 + \cdots + p_k &= 1 \end{aligned} \quad (7)$$

得到公式

$$V = \frac{1}{n} \sum_{s=1}^k \left(\frac{Y_s^2}{p_s} \right) - n \quad (8)$$

该式通常使 V 的计算稍容易些。

现在我们回到重要的问题:“什么是 V 的一个合理的值?”这可以通过参考表 1 这样的表来求得,表 1 对于各种 ν 的值给出“自由度 ν 的 χ^2 分布”的值。要使用表中 $\nu = k - 1$ 的行;“自由度”的数是 $k - 1$, 比范畴数小 1。(直观地看,这意味着 Y_1, Y_2, \dots, Y_k 不完全独立,因为(7)表明,如果 Y_2, \dots, Y_k 已知,则 Y_1 即可计算出来;

因此,有 $k-1$ 个自由度。这个论证是不严格的,但是下面的理论可以证实它。)

表 1 χ^2 分布的选定的百分值

	$p=1\%$	$p=5\%$	$p=25\%$	$p=50\%$	$p=75\%$	$p=95\%$	$p=99\%$
$\nu=1$	0.00016	0.00393	0.1015	0.4549	1.323	3.841	6.635
$\nu=2$	0.02010	0.1026	0.5754	1.386	2.773	5.991	9.210
$\nu=3$	0.1148	0.3518	1.213	2.366	4.108	7.815	11.34
$\nu=4$	0.2971	0.7107	1.923	3.357	5.385	9.488	13.28
$\nu=5$	0.5543	1.1455	2.675	4.351	6.626	11.07	15.09
$\nu=6$	0.8721	1.635	3.455	5.348	7.841	12.59	16.81
$\nu=7$	1.239	2.167	4.255	6.346	9.037	14.07	18.48
$\nu=8$	1.646	2.733	5.071	7.344	10.22	15.51	20.09
$\nu=9$	2.088	3.325	5.899	8.343	11.39	16.92	21.67
$\nu=10$	2.558	3.940	6.737	9.342	12.55	18.31	23.21
$\nu=11$	3.053	4.575	7.584	10.34	13.70	19.68	24.72
$\nu=12$	3.571	5.226	8.438	11.34	14.85	21.03	26.22
$\nu=15$	5.229	7.261	11.04	14.34	18.25	25.00	30.58
$\nu=20$	8.260	10.85	15.45	19.34	23.83	31.41	37.57
$\nu=30$	14.95	18.49	24.48	29.34	34.80	43.77	50.89
$\nu=50$	29.71	34.76	42.94	49.33	56.33	67.50	76.15
$\nu > 30$	$\nu \cdot 2\sqrt{\nu}x_p + \frac{2}{3}x_p^2 - \frac{2}{3} + O(1/\sqrt{\nu})$						
$x_p =$	-2.33	-1.64	-0.674	0.00	0.674	1.64	2.33

更多的值见 M. Abramowitz 和 I. A. Stegun 编的 *Handbook of Mathematical Functions* (Washington, D. C.; U.S. Government Printing Office, 1964), 表 26.8, 也见式(22)和习题 16

如果表中第 p 列第 ν 行的项是 x , 则它意味着“如果 n 充分大, 式(8)中的量 V 将以大约 p 的概率小于或等于 x ”。例如, 第 10 行的 95% 那一列的值是 18.31, 这就是说, 将仅有 5% 的次数取 $V > 18.31$ 。

现在让我们假定, 利用某个假设是随机数的序列, 在一台计算机上模拟上述掷骰子实验, 得到下列结果:

$$\begin{array}{cccccccccccc}
 s= & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
 \text{实验 1, } Y_i = & 4 & 10 & 10 & 13 & 20 & 18 & 18 & 11 & 13 & 14 & 13 \\
 \text{实验 2, } Y_i = & 3 & 7 & 11 & 15 & 19 & 24 & 21 & 17 & 13 & 9 & 5
 \end{array} \quad (9)$$

在第一种情况下, 可以计算 χ^2 统计, 得到 $V_1 = 29 \frac{59}{120}$, 而在第二种情况下, 则得到

$V_2 = 1 \frac{17}{120}$ 。参考表中自由度为 10 的项, 我们看到 V_1 太高; V 大约只有 1% 的机会大于 23.21! (通过使用更详尽的表, 可以发现事实上 V 仅有 0.1% 的机会这样高。) 因此, 实验 1 表示了对随机特性的重大偏离。

另一方面, V_2 十分小, 因为在实验 2 中 Y_i 十分接近于 (2) 中的期望值 np_i 。 χ^2 表告诉我们, V_2 的值太低了: 观察的值竟如此接近于期望的值, 我们不能认为这结果是随机的! (其实由其它的表可见, V 的这样低的值, 当自由度是 10 时, 仅有 0.03% 的机会。) 最后, (5) 中计算的值 $V = 7 \frac{7}{48}$ 也可以用表 1 进行校验。它落在 25% 和 50% 的表项之间, 所以不能认为它是过于高或过于低的; 根据这个检验, (2) 中的观察是令人满意地随机的。

值得注意的是, 不管 n 的值是什么, 也不管概率 p_i 是多少, 都使用同一张表中的值。只有数 $\nu = k - 1$ 对结果有影响。但是实际上, 这张表的值不是绝对正确的: χ^2 分布是仅对足够大的 n 值才正确的一个近似值。那么, n 应该多大呢? 一个经验规则是把 n 取得充分大, 使得每个期望值 np_i 是 5 或更大; 而更可取的是, 取的 n 比这大得多, 以得到更强有力的检验。在上面的例子中, 我们取 $n = 144$, 所以 np_2 仅为 4, 结果违背了上述的“经验规则”。这只是由于作者厌倦掷骰子而造成的; 它使得表 1 中的项对于我们的应用来说不大精确。在一台计算机上, 以 $n = 1000$ 或 10000 或甚至 100000 来进行这个实验, 将比这好得多。我们也可以把 $s = 2$ 和 $s = 12$ 的数据合并在一起; 于是, 这个检验将仅有 9 个自由度, 但 χ^2 的近似将更精确。

通过考虑仅有两个范畴、其概率分别为 p_1 和 p_2 的情形, 我们可以得知所涉及近似的粗略程度。假设 $p_1 = \frac{1}{4}$ 和 $p_2 = \frac{3}{4}$ 。按所述经验规则, 为了得到一个令人满意的近似, 应当有 $n \geq 20$ 。让我们来检验一下。当 $n = 20$ 时, 对于 $-5 \leq r \leq 15$, V 可能的值是 $(Y_1 - 5)^2/5 + (5 - Y_1)^2/15 = \frac{4}{15}r^2$ 。我们希望知道表 1 的 $\nu = 1$ 行对 V 的分布的描述好到什么程度。 χ^2 分布连续地变化, 而实际的 V 分布则有稍大的跳跃, 所以, 为了表示确切的分布我们需要某些约定。设这个试验的 (可能是不同的) 结果分别以概率 $\pi_0, \pi_1, \dots, \pi_n$ 导致值 $V_0 \leq V_1 \leq \dots \leq V_n$, 假设给定的百分比 p 落入范围 $\pi_0 + \dots + \pi_{j-1} < p < \pi_0 + \dots + \pi_j$ 。我们将通过“百分比点” x 来表示 p , 使得 V 小于 x 的概率小于等于 p , 而大于 x 的概率小于等于 $1 - p$ 。不难看出, 惟一这样的数是 $x = V_j$ 。在 $n = 20$ 和 $\nu = 1$ 的例子中, 结果是, 当 $p = 1\%, 5\%, 25\%, 50\%, 75\%, 95\%, 99\%$ 时, 与表 1 中的近似值相应的精确分布的百分比点分别为 (精确到小数点后两位)

0, 0, .27, .27, 1.07, 4.27, 6.67

例如, $p = 95\%$ 时的百分比点是 4.27, 而表 1 给出 3.841 的估计。后一个值太低; 它 (不正确地) 告诉我们以 95% 的水平拒绝值 $V = 4.27$, 而事实上 $V \geq 4.27$ 的概率大

于 6.5%。当 $n=21$ 时情况稍有变化,因为所预期的值 $np_1=5.25$ 和 $np_2=15.75$ 绝不可能精确地得到; $n=21$ 的百分比点是

.02, .02, .14, .40, 1.29, 3.57, 5.73

当 $n=50$ 时我们期望表 1 是更好的近似,但对应的值实际上在某些方面比 $n=20$ 更远离表 1:

.03, .03, .03, .67, 1.31, 3.23, 6

以下是 $n=300$ 时的值:

0, 0, .07, .44, 1.44, 4, 6.42

在这种情况下,甚至在每个范畴中, $np_i \geq 75$ 时,表 1 的表项也仅好到大约一位有效数字。

n 的适当选择是不大容易办到的。如果骰子实际上是有偏倚的,则当 n 越来越大时,即可发现这一事实(参照习题 12)。但是当具有一个很强倾向的数区紧跟有一个相反倾向的数区时,很大的 n 值将倾向于平滑掉局部地非随机的行动。当转动实际的骰子时,局部的非随机的行动并不是一个问题,因为贯穿于整个检验使用的是同一个骰子。但是由计算机生成的数的一个序列却可能很清楚地显示这样的异常性。或许应该对若干个不同的 n 值来做 χ^2 检验。无论如何, n 总是应该稍大些。

我们可以将 χ^2 检验总结如下:做相当多(n)次独立的观察。(重要的是,除非观察都是独立的,否则就避免使用 χ^2 方法。例如,拿习题 10 来说,它考虑的是当观察的一半依赖于另一半时的情况。)我们计算落入 k 个范畴中的每一个观察数,并计算式(6)和(8)中给出的量 V 。然后对于 $\nu=k-1$,把 V 同表 1 的数进行比较。如果 V 小于 1% 那列的值或大于 99% 那列的值,则我们拒绝这个数并认为它是不够随机的。如果它处于 1% 和 5% 的表项之间或 95% 和 99% 的表项之间,则这些数是“可疑的”;如果(通过插值) V 在 5% 和 10% 的表项之间,或 90% 与 95% 的表项之间,则这些数可能是“几乎可疑的”。 χ^2 检验通常至少要对不同的数据组进行三次,而且如果在三个结果中至少有两个是可疑的,则这些数即被认为是不够随机的。

例如,图 2 说明了对六个随机数序列的每一个应用五种不同类型的 χ^2 检验的结果。图中每个检验被应用于这序列的数中三个不同的块区。生成程序 A 是 MacLaren-Marsaglia 方法(算法 3.2.2M 应用于 3.2.2-(13)中的序列),生成程序 E 是斐波那契方法,3.2.2-(5),而其余的生成程序是具有下列参数的线性同余序列:

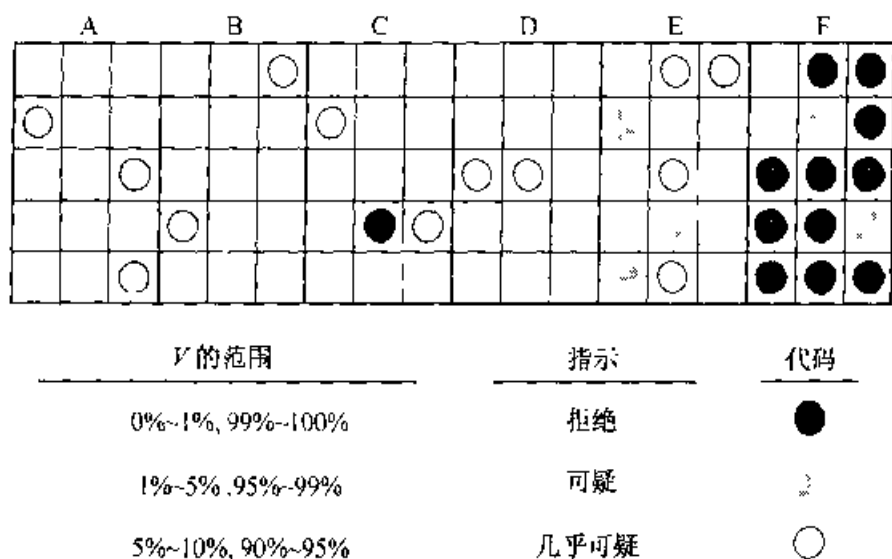
生成程序 B: $X_0=0, a=3141592653, c=2718281829, m=2^{35}$ 。

生成程序 C: $X_0=0, a=2^7+1, c=1, m=2^{35}$ 。

生成程序 D: $X_0=47594118, a=23, c=0, m=10^8+1$ 。

生成程序 F: $X_0=314159265, a=2^{18}+1, c=1, m=2^{35}$ 。

由图 2 可以作出结论(仅就这些检验来说),生成程序 A, B, D 是令人满意的,而生成程序 C 处于两可之间,因而大概它将被拒绝,生成程序 E 和 F 肯定不能令人满意。当然,生成程序 F 效能低。生成程序 C 和 D 已在文献中讨论过,但它们的乘数太小。(生成程序 D 是 Lehmer 于 1948 年提出的原始乘法生成程序;生成程序 C 是

图2 在90个 χ^2 检验中的“显著”偏离的图示(参看图5)

Rotenberg 于1960年提出的 $c \neq 0$ 的原始线性同余生成程序。)

除了用“可疑”、“几乎可疑”等等作为判断 χ^2 检验结果的原则外,还有一个不那么专用的方法可资利用,我们将在这一节的稍后部分来讨论它。

B. Kolmogorov-Smirnov 检验 我们已经看到,当观察可以落入有限的 k 个范畴时,即可应用 χ^2 检验。然而,考虑可以取无限多个值的随机量,例如随机分数(即0和1之间的一个随机实数)的情况也不鲜见。尽管在计算机里只能表示有限多个实数,但我们希望随机值本质上具有 $[0,1)$ 中的随机实数的特性。

不论概率分布是有限的还是无限的,有一种统一的描述方法普遍地用于概率和统计的研究中。假设要描述一个随机量 X 的值的分布,我们可以借助分布函数 $F(x)$ 来做,其中

$$F(x) = \Pr(X \leq x) = (X \leq x) \text{ 的概率}$$

图3示出了三个例子。首先,我们看到一个随机二进位的分布函数,即在 X 仅取两个值0和1的情况下,每一个有概率 $\frac{1}{2}$ 。图3(b)部分显示了在0与1之间一致地分布的随机实数的分布函数;这里当 $0 \leq x \leq 1$ 时, $X \leq x$ 的概率简单地就等于 x ,例如 $X \leq \frac{2}{3}$ 的概率自然为 $\frac{2}{3}$ 。而图3(c)显示了在 χ^2 检验中值 V 的极限分布(这里以10个自由度来表示);我们已经看到这个分布在表1中以另一种方式表示。注意,当 x 从 $-\infty$ 变到 $+\infty$ 时, $F(x)$ 总是从0增加到1。

如果我们对随机量 X 做 n 次独立观察,由此得到 X_1, X_2, \dots, X_n 的值,则可形成经验分布函数 $F_n(x)$,即

$$F_n(x) = \frac{\text{小于或等于 } x \text{ 的 } X_1, X_2, \dots, X_n \text{ 的数目}}{n} \quad (10)$$

图4示出了三个经验分布函数(呈锯齿形,尽管严格说来垂直线并不是 $F_n(x)$ 图的

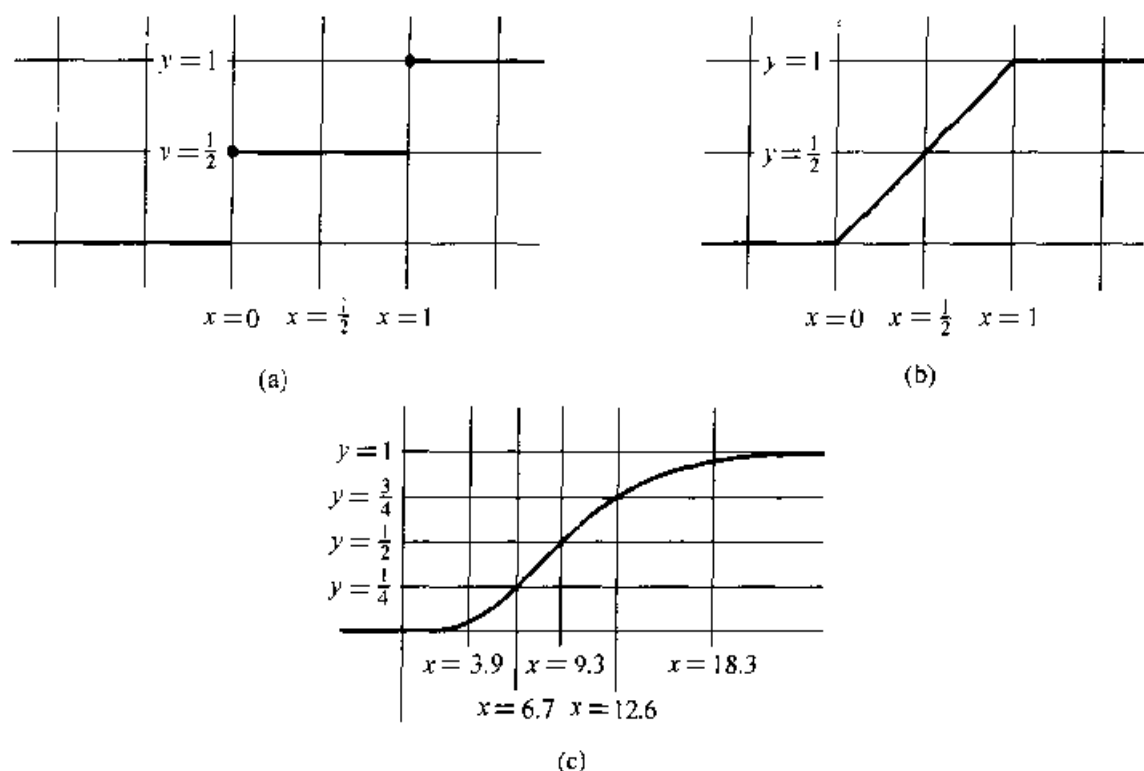


图3 分布函数的例子

一部分), 叠印在假定的实际分布函数 $F(x)$ 的图上的情况。当 n 充分大时, $F_n(x)$ 将越来越很好地近似于 $F(x)$ 。

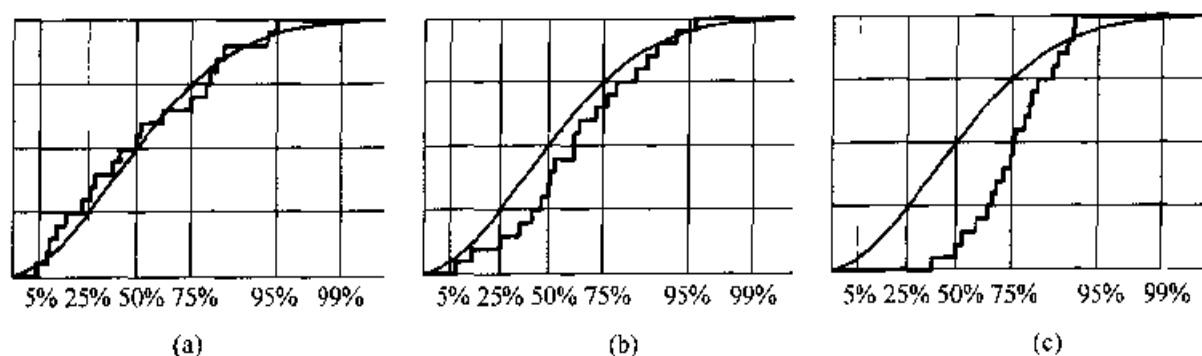


图4 经验分布的例子

当 $F(x)$ 没有跳跃时, 可以使用 Kolmogorov-Smirnov 检验 (KS 检验)。该检验是以 $F(x)$ 与 $F_n(x)$ 之间的差为基础的。坏的随机数生成程序给出不太近似于 $F(x)$ 的经验分布函数。在图 4(b) 的例子中 X_i 一直太高, 所以经验分布函数就太低。图 4(c) 给出了一个甚至更坏的例子; 很明显, $F_n(x)$ 和 $F(x)$ 之间这样大的偏离是极不可能的, KS 检验能告诉我们这种不可能性的程度。

为进行 KS 检验, 我们构造下面的统计:

$$K_n^+ = \sqrt{n} \max_{-\infty < x < +\infty} (F_n(x) - F(x))$$

$$K_n^- = \sqrt{n} \max_{-\infty < x < +\infty} (F(x) - F_n(x))$$
(11)

这里 K_n^+ 度量当 F_n 大于 F 时的最大偏离量, 而 K_n^- 度量当 F_n 小于 F 时的极大偏离。对于图 4 的例子的统计是

图 4(a)	图 4(b)	图 4(c)	
K_{20}^+ 0.492	0.134	0.313	(12)
K_{20}^- 0.536	1.027	2.101	

(注: 出现于式(11)中的因子 \sqrt{n} 最初可能使人感到莫名其妙。习题 6 说明, 对于固定的 x , $F_n(x)$ 对 $F(x)$ 的标准偏离与 $1/\sqrt{n}$ 成比例; 因此, \sqrt{n} 的作用是放大统计量 K_n^+ 与 K_n^- , 以使这个标准偏离与 n 无关。)

和在 χ^2 测试中一样, 我们现在可以通过在“百分比”表中考察 K_n^+ 和 K_n^- 来确定它们是否特别高或特别低。对于 K_n^+ 和 K_n^- 表 2 均可用于这个目的。例如, K_{20}^- 小于或等于 0.7975 的概率是 75%。与 χ^2 检验不同, 这个表中的项不仅仅是 n 充分大时的近似值; 表 2 给出了精确值(当然舍入误差除外), 因而对于任意的 n 值 KS 检验都能可靠地使用。

表 2 分布 K_n^+ 和 K_n^- 的部分百分比点

	$p=1\%$	$p=5\%$	$p=25\%$	$p=50\%$	$p=75\%$	$p=95\%$	$p=99\%$
$n=1$	0.01000	0.05000	0.2500	0.5000	0.7500	0.9500	0.9900
$n=2$	0.01400	0.06749	0.2929	0.5176	0.7071	1.0980	1.2728
$n=3$	0.01699	0.07919	0.3112	0.5147	0.7539	1.1017	1.3589
$n=4$	0.01943	0.08789	0.3202	0.5110	0.7642	1.1304	1.3777
$n=5$	0.02152	0.09471	0.3249	0.5245	0.7674	1.1392	1.4024
$n=6$	0.02336	0.1002	0.3272	0.5319	0.7703	1.1463	1.4144
$n=7$	0.02501	0.1048	0.3280	0.5364	0.7755	1.1537	1.4246
$n=8$	0.02650	0.1086	0.3280	0.5392	0.7797	1.1586	1.4327
$n=9$	0.02786	0.1119	0.3274	0.5411	0.7825	1.1624	1.4388
$n=10$	0.02912	0.1147	0.3297	0.5426	0.7845	1.1658	1.4440
$n=11$	0.03028	0.1172	0.3330	0.5439	0.7863	1.1688	1.4484
$n=12$	0.03137	0.1193	0.3357	0.5453	0.7880	1.1714	1.4521
$n=15$	0.03424	0.1244	0.3412	0.5500	0.7926	1.1773	1.4606
$n=20$	0.03807	0.1298	0.3461	0.5547	0.7975	1.1839	1.4698
$n=30$	0.04354	0.1351	0.3509	0.5605	0.8036	1.1916	1.4801
$n > 30$	$y_p = 1/(6\sqrt{n}) + O(1/n)$, 其中 $y_p^2 = \frac{1}{2} \ln(1/(1-p))$						
$y_p =$	0.07089	0.1601	0.3793	0.5887	0.8326	1.2239	1.5174

为扩充这个表, 请见式(25)和(26)以及习题 20 的答案

就其本身而言,式(11)不易于用计算机计算,因为我们要求无限多个 x 值的极大值。然而 $F(x)$ 是递增的且 $F_n(x)$ 只在有限步中增长,由此可以导出求统计量 K_n^+ 和 K_n^- 值的简单步骤:

步骤 1. 获取独立的观察量 X_1, X_2, \dots, X_n 。

步骤 2. 重新排列这些观察量,使它们排成递增的次序,即使得 $X_1 \leq X_2 \leq \dots \leq X_n$ (有效的排序算法是第 5 章的课题,但如同习题 23 中所示,在这种情况下有可能避免排序)。

步骤 3. 通过下列公式得出所求的统计量:

$$\begin{aligned} K_n^+ &= \sqrt{n} \max_{1 \leq j \leq n} \left(\frac{j}{n} - F(X_j) \right) \\ K_n^- &= \sqrt{n} \max_{1 \leq j \leq n} \left(F(X_j) - \frac{j-1}{n} \right) \end{aligned} \quad (13)$$

观察次数 n 的适当选择,对于这个检验来说,比 χ^2 检验稍容易些,尽管某些考虑是类似的。如果诸随机变量 X_j 实际上属于概率分布 $G(x)$,而却被假定属于由 $F(x)$ 所给出的分布,则为否定 $G(x) = F(x)$ 的假设将须取相当大的 n 值;因为我们需要 n 充分大,以期望经验分布 $G_n(x)$ 和 $F_n(x)$ 观察起来不同。另一方面, n 的充分大值趋向于把局部非随机的特性修平,而这样不合要求的特性在随机数的大多数计算机应用中具有相当大的危险;这又要求我们取较小的 n 值。一个好的折中是取 n 等于(比方说)1000,并且在一个随机序列的不同部分上,来进行对于 K_{1000}^+ 的大量计算,由此得到值

$$K_{1000}^+(1), K_{1000}^+(2), \dots, K_{1000}^+(r) \quad (14)$$

我们也可以再次对这些结果应用 KS 检验:这里设 $F(x)$ 是 K_{1000}^+ 的分布函数,并确定根据式(14)中观察的值得到的经验分布 $F_r(x)$ 。很幸运,这种情况下的 $F(x)$ 是很简单的;对于像 $n=1000$ 这样充分大的 n 值, K_n^+ 的分布由

$$F_\infty(x) = 1 - e^{-2x^2}, \quad x \geq 0 \quad (15)$$

充分逼近。同样的论述也可以应用于 K_n^- , 因为 K_n^+ 和 K_n^- 有着相同的期望特性。对于适当大小的 n 做若干次检验,然后在另一个 KS 检验中组合这些观察量,这种方法将趋向于检测局部和全局两者的非随机特性。

例如,当编写这一章时,作者做了以下简单的试验:把下一小节中所述的“5 的极大值”检验用于一组 1000 个一致的随机数上,得到 200 个观察量 X_1, X_2, \dots, X_{200} , 假设对于 $0 \leq x \leq 1$ 它们属于分布 $F(x) = x^5$ 。把这些观察量分成 20 组,每组 10 个,计算每组的统计量 K_{10}^+ 。如此得到 20 个 K_{10}^+ 的值,导致了图 4 所示的经验分布。图 4 的每一个图形中所示的光滑曲线,是 K_{10}^+ 应有的真正的分布。图 4(a)示出了由序列

$$Y_{n+1} = (3141592653 Y_n + 2718281829) \bmod 2^{35}, \quad U_n = Y_n / 2^{35}$$

得出的 K_{10}^+ 的经验分布, 而且它是令人满意地随机的。图 4(b) 是由斐波那契方法得到的, 这个序列有全局非随机的特性, 即, 可以证明, 在“5 的极大值”检验中, 诸观察量 X_n 不具有正确的分布 $F(x) = x^5$ 。图 4(c) 来自口碑不好且无效的线性同余序列 $Y_{n+1} = ((2^{18} + 1) Y_n + 1) \bmod 2^{35}, U_n = Y_n / 2^{35}$ 。

应用 KS 检验于图 4 中的数据, 就得出了(12)中所示的结果。就 $n = 20$ 参照表 2, 我们看到对于图 4(b), K_{20}^+ 和 K_{20}^- 的值是几乎可疑的(它们处于大约 5% 和 88% 的水平), 但还没有坏到要立即拒绝它的程度。当然, 图 4(c) 的 K_{20}^- 完全不协调, 所以“5 的极大值”的检验揭示了该随机数生成程序肯定通不过。

我们预料, 在这个实验中的 KS 检验在确定全局非随机性方面比确定局部非随机性要更困难, 因为图 4 的基本观察所用的抽样中, 每个抽样仅包括 10 个观察量。如果我们取 20 个组, 每组 1000 个观察量, 则图 4(b) 将显示一个更加显著的偏离。为了说明这点, 应用单个的 KS 检验于导致图 4 的所有 200 个观察量, 得到下列结果:

	图 4(a)	图 4(b)	图 4(c)	
K_{200}^+	0.477	1.537	2.819	(16)
K_{200}^-	0.817	0.194	0.058	

这里已经确定地检测出斐波那契生成程序的全局非随机性。

我们可以总结 KS 检验如下: 给出 n 个独立的观察量 X_1, \dots, X_n , 它们取自某个由一个连续函数 $F(x)$ 所确定的分布, 即, $F(x)$ 必须像图 3(b) 和 (c) 所示的函数那样, 没有图 3(a) 中那样的跳跃。把紧挨在等式(13)之前说明的步骤用于这些观察量上, 就得到统计量 K_n^+ 和 K_n^- 。这些统计量应当按照表 2 来分布。

现在可对 KS 检验与 χ^2 检验作某些比较。首先, 我们应注意到, KS 检验可以与 χ^2 检验一起使用, 以给出一个比我们在总结 χ^2 检验时提到的特殊方法更好的过程(即, 比起进行三次检验并考虑这些结果有多少是“可疑的”来, 有一个更好的方法可以使用)。假设我们对于一个随机数序列的不同部分, 已经做了比如说 10 次独立的 χ^2 检验, 并得到值 V_1, V_2, \dots, V_{10} 。简单地计算有多少个 V 是可疑地大或小并不是一个好策略。这个方法在一些极端情形下是有效的, 而非常大和非常小的值可能意味着序列有过多的局部非随机性; 但更好且更一般的方法是画出这 10 个值的经验分布函数, 并把它同正确分布(可以从表 1 得到)进行比较。这将给出 χ^2 检验结果的更清晰的写照, 而且事实上统计量 K_{10}^+ 和 K_{10}^- 可作为成功或失败的标志来确定。如果仅有 10 个值或甚至多到 100 个值, 这些都能通过手算和利用图形的方法容易地完成; 对于更大量的 V , 就需要有计算 χ^2 分布的计算机子程序了。注意, 图 4(c) 中所有 20 个观察量落入 5% 和 95% 水平之间, 所以我们在个体上并未认为它们中任何一个可疑的; 但在总体上看, 经验分布却表明这些观察量是不成问题的。

KS 检验和 χ^2 检验之间的一个重要差别在于, KS 检验应用于没有跳跃的分布

$F(x)$, 而 χ^2 检验应用于只是跳跃的分布(因为所有观察都分成 k 个范畴)。因此这两个检验是为不同种类的应用而设置的。然而甚至当 $F(x)$ 连续时, 如果把 $F(x)$ 的区域分成 k 部分, 并忽略在每个部分内的变化, 则也有可能应用 χ^2 检验。例如, 如果我们要来检验 U_1, U_2, \dots, U_n 是否可认为是 0 与 1 之间的一致分布, 则可以检验它们对于 $0 \leq x \leq 1$, 是否有 $F(x) = x$ 分布。这是 KS 检验的一个很自然的应用。但是我们也可以把从 0 到 1 的区间分成 100 个相等部分, 计算有多少个 U 落到每个部分, 并且以自由度 99 来应用 χ^2 检验。对于比较 KS 检验和 χ^2 检验的有效性, 目前, 还没有许多可资利用的理论结果。作者找到了某些例子, 其中 KS 检验比 χ^2 检验更明确地指出非随机性, 而在其它例子中, 则是 χ^2 检验给出了更有意义的结果。例如, 如果把上边提到的 100 个范畴编号为 $0, 1, \dots, 99$, 而且如果同期望值的偏差在 0 到 49 的间隔是正的, 而在 50 到 99 的间隔是负的, 则经验分布函数将更多地由 $F(x)$ 的而不是由 χ^2 的值所指出; 但如果在间隔 $0, 2, \dots, 98$ 中出现正的偏离, 而在间隔 $1, 3, \dots, 99$ 中出现负的偏离, 则经验分布函数将趋向于更紧密地靠拢 $F(x)$ 。因此, 测定的偏差种类稍有不同。应用 χ^2 检验形成图 4 的 200 个观察量, 其中 $k = 10$, 所得的 V 的值分别是 9.4, 17.7 和 39.3; 所以在这个具体的情况下这些值同在 (16) 中给出的 KS 的值是十分相近的。由于 χ^2 检验本来就不太精确, 而且又要求有充分大的 n 值, 因此当有待检验的是一个连续分布时, KS 检验就具有若干优点了。

还有一个有趣的例子。导致图 2 的数据是对于 $1 \leq t \leq 5$, 以“ t 的极大值”为原则, 以 $n = 200$ 个观察量为基础的 χ^2 统计, 且这个范围分成 10 个同等可能的部分。KS 统计量 K_{200}^+ 和 K_{200}^- 可由完全相同的 200 个观察量的诸组来计算, 而且这些结果可以以和我们在图 2 中所采用的完全相同的方式造表(指出哪些 KS 值超过 99% 的水平, 等等)。图 5 中示出了在这种情况下结果。注意, 生成程序 D(Lehmer 原来的方法)在图 5 中显得非常坏, 而对于相同的数据, χ^2 检验在图 2 上则显得非常顺利; 相反, 生成程序 E(斐波那契方法)在图 5 中看起来还不那么坏。好的生成程序 A 和 B, 令人满意地通过所有检验。图 2 和图 5 之间出现矛盾的原因, 主要是: (a) 观察量个数 200, 对于一个功能很强的检验来说实际上不够大; (b) “拒绝”、“可疑”、“几乎可疑”的排列规则本身就是可疑的。

(附带说一下, 由于在 20 世纪 40 年代使用一个“坏的”随机数生成程序而责备 Lehmer 是不公正的, 因为他对生成程序 D 的实际使用十分有效。ENIAC 计算机是一台高度并行的机器, 它借助于一个线路连接板进行程序设计; Lehmer 把它设置成使它的累加器之一重复地以 23 乘它自己的内容(模 $10^8 + 1$), 在几毫秒内就得到一个新值。由于乘数 23 太小了, 我们知道, 通过这个过程得到的每个值和它前边得到的值就太密切相关了, 因而难以认为它是充分随机的; 但是由伴随的程序所使用的特殊累加器, 其中实际使用的值之间的持续时间是相对较长的, 而且有某些变动。所以对于很大的而且变化的 k 值有效的乘数是 23^k 。)

C. 历史, 文献和理论 1900 年, Karl Pearson 引进了 χ^2 检验 [*Philosophical Magazine*, Series 5, 50, 157~175]。Pearson 的重要论文被认为是现代统计学的基础

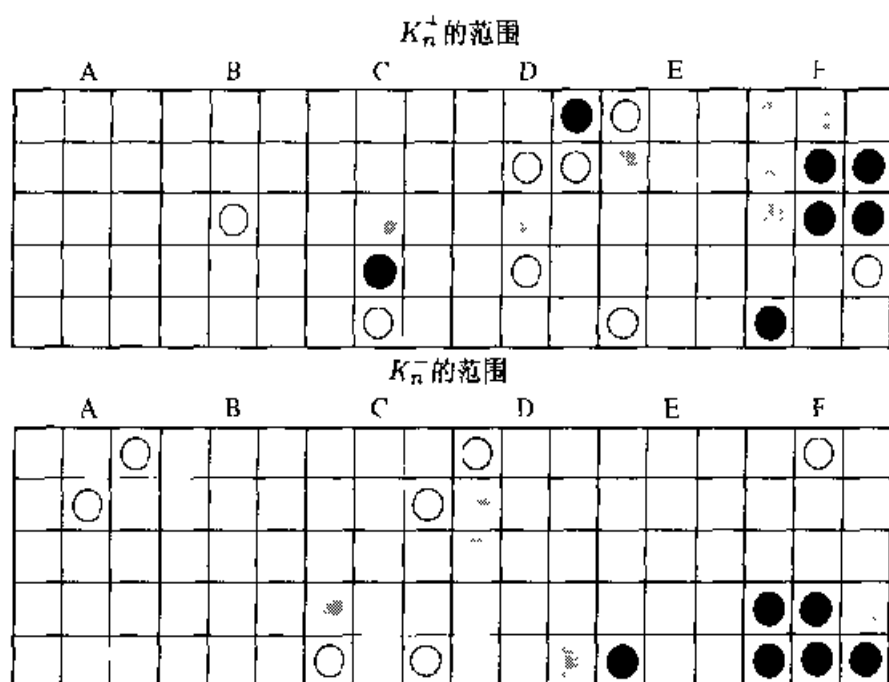


图5 KS检验应用于和图2相同的数据

之一,因为在这以前,人们只是简单地用图形来画出实验的结果,并断定它们是正确的。在这篇论文中,Pearson 举出了以前滥用统计学的若干有趣的例子;他还证明了轮盘赌中的某些机遇(1892 年他在蒙特卡罗实验了两周时间),同期望的频率相差竟如此之远,以致对于一个公正的轮子也只能以大约 10^{29} 对 1 的赌注打赌! William G. Cochran 的评述论文中有关于 χ^2 检验的一般讨论和广泛的文献。见 *Annals of Math. Stat.* **23** (1952), 315~345。

现在我们给出关于 χ^2 检验的简单理论推导。容易看出, $Y_1 = y_1, \dots, Y_k = y_k$ 的精确概率是

$$\frac{n!}{y_1! \cdots y_k!} p_1^{y_1} \cdots p_k^{y_k} \quad (17)$$

如果我们假定, Y_s 以泊松(Poisson)概率

$$\frac{e^{-np_s} (np_s)^{y_s}}{y_s!}$$

有值 y_s , 而且这些 Y 是独立的, 则 (Y_1, \dots, Y_k) 等于 (y_1, \dots, y_k) 的概率是

$$\prod_{s=1}^k \frac{e^{-np_s} (np_s)^{y_s}}{y_s!}$$

而且 $Y_1 + \dots + Y_k$ 等于 n 的概率是

$$\sum_{\substack{y_1 + \dots + y_k = n \\ y_1, \dots, y_k \geq 0}} \prod_{s=1}^k \frac{e^{-np_s} (np_s)^{y_s}}{y_s!} = \frac{e^{-n} n^n}{n!}$$

如果我们假定, 除了条件 $Y_1 + \dots + Y_k = n$ 外, 它们是独立的, 则 $(Y_1, \dots, Y_k) =$

(y_1, \dots, y_k) 的概率是商

$$\left(\prod_{s=1}^k \frac{e^{-np_s} (np_s)^{y_s}}{y_s!} \right) / \left(\frac{e^{-n} n^n}{n!} \right)$$

这等于式(17)。因此,除了它们有固定的和这一事实之外,我们可以认为这些 Y 是独立的泊松分布。

做一下变量的改动是方便的,令

$$Z_s = \frac{Y_s - np_s}{\sqrt{np_s}} \quad (18)$$

使得 $V = Z_1^2 + \dots + Z_k^2$ 。条件 $Y_1 + \dots + Y_k = n$ 等价于要求

$$\sqrt{p_1} Z_1 + \dots + \sqrt{p_k} Z_k = 0 \quad (19)$$

让我们考察使得式(19)成立的所有向量 (Z_1, \dots, Z_k) 构成的 $(k-1)$ 维空间 S 。对于大的 n 值,每个 Z_s 有近乎正态的分布(参考习题 1.2.10-15);因此,在 S 的一个微分体积 dz_1, \dots, dz_k 内的点以近似正比于 $\exp(-(z_1^2 + \dots + z_k^2)/2)$ 的概率出现(在推导中,正是这一点使 χ^2 方法对于大的 n 仅仅成为一个近似方法)。于是 $V \leq v$ 的概率为

$$\frac{\int_{(z_1^2, \dots, z_k^2) \text{ 在 } S \text{ 中 } z_1^2 + \dots + z_k^2 \leq v} \exp(-(z_1^2 + \dots + z_k^2)/2) dz_1 \dots dz_k}{\int_{(z_1, \dots, z_k) \text{ 在 } S \text{ 中} \exp(-(z_1^2 + \dots + z_k^2)/2) dz_1 \dots dz_k} \quad (20)$$

由于超平面(19)通过 k 维空间的原点,故(20)中的分子是对一个中心在原点的 $(k-1)$ 维超球面内部的一个积分。对于某个函数 f ,以径 χ 和角度 $\omega_1, \dots, \omega_{k-2}$ 做适当的广义极坐标变换,把式(20)变成

$$\frac{\int_{\chi^2 \leq v} e^{-\chi^2/2} \chi^{k-2} f(\omega_1, \dots, \omega_{k-2}) d\chi d\omega_1 \dots d\omega_{k-2}}{\int_0^\infty e^{-\chi^2/2} \chi^{k-2} f(\omega_1, \dots, \omega_{k-2}) d\chi d\omega_1 \dots d\omega_{k-2}}$$

(见习题 15);然后对角度 $\omega_1, \dots, \omega_{k-2}$ 的积分给出一个常数因子,从分子和分母中消去之。最后我们得到 $V \leq v$ 的近似概率的公式

$$\frac{\int_0^{\sqrt{v}} e^{-\chi^2/2} \chi^{k-2} d\chi}{\int_0^\infty e^{-\chi^2/2} \chi^{k-2} d\chi} \quad (21)$$

上述的推导使用了符号 χ 来表示径长度,恰如 Pearson 在他的开创性论文中所做的那样;这就是 χ^2 检验得名的由来。代入 $t = \chi^2/2$,这一积分可以借助于不完备的伽玛函数来表示,这就是我们在 1.2.11.3 节中讨论的:

$$\lim_{n \rightarrow \infty} \Pr(V \leq v) = \gamma\left(\frac{k-1}{2}, \frac{v}{2}\right) / \Gamma\left(\frac{k-1}{2}\right) \quad (22)$$

这正是具有 $k-1$ 自由度的 χ^2 分布的定义。

再来看 KS 检验。1933 年 A. N. Kolmogorov 提出了一个以统计

$$K_n = \sqrt{n} \max_{-\infty < x < +\infty} |F_n(x) - F(x)| = \max(K_n^+, K_n^-) \quad (23)$$

为基础的检验。N. V. Smirnov 于 1939 年对这个检验做了若干修正,包括我们在上边已经提到的个别地考察 K_n^+ 和 K_n^- 在内。类似的检验有很多,但 K_n^+ 和 K_n^- 似乎最便于计算机应用。J. Durbin 在 *Regional Conf. Series on Applied Math.* 9 (SIAM, 1973) 的专题文章中,详尽地评述了有关 KS 检验及它们的推广的文献,其中包括一份广泛的文献目录。

为了研究 K_n^+ 和 K_n^- 的分布,我们从下面的基本事实出发:如果 X 是具有连续分布的 $F(x)$ 的一个随机变量,则 $F(X)$ 是一致分布于 0 与 1 之间的实数。为证明这一点,仅仅需要验证,如果 $0 \leq y \leq 1$,则 $F(X) \leq y$ 的概率为 y 。由于 F 是连续的,故对于某个 x_0 有 $F(x_0) = y$,于是 $F(X) \leq y$ 的概率是 $X \leq x_0$ 的概率。由定义,后者的概率为 $F(x_0)$,即它为 y 。

对于 $1 \leq j \leq n$,令 $Y_j = nF(X_j)$,其中诸 X 如同在上边的步骤 2 中那样是已排好序的。于是诸变量 Y_j 和独立的、已经排序成为非递减次序的 0 和 n 之间的一致分布随机数一样, $Y_1 \leq Y_2 \leq \dots \leq Y_n$; 而且式(13)的头一个等式现在可以变换成为

$$K_n^+ = \frac{1}{\sqrt{n}} \max(1 - Y_1, 2 - Y_2, \dots, n - Y_n)$$

如果 $0 \leq t \leq n$,则 $K_n^+ \leq t/\sqrt{n}$ 的概率是对于 $1 \leq j \leq n$, $Y_j \geq j - t$ 的概率。借助于 n 维积分这不难表示

$$\frac{\int_{a_n}^n dy_n \int_{a_{n-1}}^{y_n} dy_{n-1} \dots \int_{a_1}^{y_2} dy_1}{\int_0^n dy_n \int_0^{y_n} dy_{n-1} \dots \int_0^{y_2} dy_1}, \text{ 其中 } a_j = \max(j - t, 0) \quad (24)$$

这里分母可以立即算出:它等于 $n^n/n!$,这是有意义的,因为具有 $0 \leq y_j < n$ 的所有向量 (y_1, y_2, \dots, y_n) 的超立方体的体积是 n^n ,而且它可以划分成对应于诸 y 每个可能的排序的 $n!$ 个相等部分。分子中的积分稍难些,但可按习题 17 中提示的方法求解,使我们得到一般的公式

$$\Pr\left(K_n^+ \leq \frac{t}{\sqrt{n}}\right) = \frac{t}{n^n} \sum_{0 \leq k \leq t} \binom{n}{k} (k - t)^k (t + n - k)^{n-k-1} = \quad (25)$$

$$1 - \frac{t}{n^n} \sum_{t < k \leq n} \binom{n}{k} (k - t)^k (t + n - k)^{n-k-1} \quad (26)$$

K_n^- 的分布与此完全一样。式(26)首先由 N. V. Smirnov 得出 [Uspekhi Mat. Nauk 10(1944), 176~206]; 也可参考 Z. W. Birnbaum 和 Fred H. Tingey, *Annals Math. Stat.* 22 (1951), 592~596。Smirnov 对于所有固定的 $s \geq 0$, 推导出下列近似公式:

$$\Pr(K_n^+ \leq s) = 1 - e^{-2s^2} \left(1 - \frac{2}{3}s/\sqrt{n} + O(1/n)\right) \quad (27)$$

这产生出在表 2 中出现的对于很大的 n 的近似值。

阿贝尔的二项式定理,即式 1.2.6-(16)证明了(25)和(26)的等价性。使用这两个公式的任何一个我们可以扩充表 2,但有一个有趣的折中:当给定 $s = t/\sqrt{n}$ 时,尽管(25)中的和只有大约 $s\sqrt{n}$ 项,但必须以多精度算术来计算之,因为这些项很大,而且它们的前导数字都消失掉了。在(26)中却没有这样的问题,因为它的项都是正的,但(26)有 $n - s\sqrt{n}$ 项。

习 题

1.[00] 应当使用 χ^2 表的哪一行来检验式(5)的值 $V = 7 \frac{7}{48}$ 是否是过于高了?

2.[20] 如果两个骰子被“装铅”,使得一个骰子上值为 1 的那一面朝上的次数恰是其它任何一面朝上次数的两倍,而另一个骰子则类似地偏倚于 6,试计算概率 p_s ,即对于 $2 \leq s \leq 12$,两个骰子的和数为 s 的概率。

►3.[23] 两个如上题所述那样装了铅的骰子被投掷 144 次,并且观察到下列的一些值:

s 值 = 2 3 4 5 6 7 8 9 10 11 12

观察数, $Y_s = 2 6 10 16 18 32 20 13 16 9 2$

利用式(1)中的概率对这些值进行 χ^2 检验,并假装不知道这个骰子事实上是有毛病的。试问它能否发现这个骰子是坏的? 如果不能,说明原因。

►4.[23] 通过模拟这样一对骰子,其中一个是正常的,而另一个被装铅,使得它经常使 1 或 6 朝上(后两者的概率是同等),作者实际得到了(9)中实验 1 的数据。试计算在这种情况下的概率以代替(1),并利用 χ^2 检验判断这个实验的结果与如此装铅的骰子是否一致?

5.[22] 设 $F(x)$ 为一致分布,如图 3(b)所示。请对下列 20 个观察值求 K_{20}^+ 和 K_{20}^- :

0.414, 0.732, 0.236, 0.162, 0.259, 0.442, 0.189, 0.693, 0.098, 0.302.

0.442, 0.434, 0.141, 0.017, 0.318, 0.869, 0.772, 0.678, 0.354, 0.718

并指出这些观察量相对于这两个检验的每一个是否同预期的特性有重大差异。

6.[M20] 对于固定的 x ,考虑式(10)中给出的 $F_n(x)$ 。给定一整数 s ,问 $F_n(x) = s/n$ 的概率是多少? $F_n(x)$ 的平均值是多少? 标准差是多少?

7.[M15] 证明 K_n^+ 和 K_n^- 绝不能为负, K_n^+ 最大可能的值是多少?

8.[00] 正文中描述了一个实验,实验中在研究一个随机序列时得到了统计量 K_{10}^+ 的 20 个值。画出这些值,得到图 4,并从得到的图形算出了统计量 KS。问为什么用 $n=20$ 的表项来研究得到的统计,而不用 $n=10$ 的表项?

►9.[20] 正文中叙述的实验由画出 K_{10}^+ 的 20 个值组成。它们是把 5 的极大值检验应用于一个随机序列的不同部分面算出的。也可以计算 K_{10} 的相应的 20 个值;因为 K_{10} 与 K_{10}^+ 有相同的分布,所以可以把这样得到的 40 个值(即 K_{10}^+ 的 20 个值和 K_{10}^- 的 20 个值)总括起来,并可应用 KS 检验得到 K_{40}^+ 、 K_{40}^- 的新值。试讨论这一设想的优点。

►10.[20] 假设通过做 n 次观察来完成一个 χ^2 检验,并得到值 V 。若再用同样的 n 次观察重复这个检验(当然,得到相同的结果),并且把两次检验的数据放在一起,把它当做对 $2n$ 次观察的一次单独的 χ^2 检验(这个过程违背了正文中指出的所有观察必须彼此独立的约定)。试问第二个

V 值如何相关于头一个值?

11. [10] 以 KS 检验代替 χ^2 检验, 解习题 10。

12. [M28] 假设对 n 个观察量的一个集合进行 χ^2 检验, 同时假定 p_i 是每个观察量落入范畴 s 的概率; 但又假设事实上这些观察量落入范畴 s 的概率为 $q_i, q_i \neq p_i$ (参看习题 3)。当然, 我们希望 χ^2 检验能检测出关于 p_i 的假定是不正确的这一事实。试证, 当 n 充分大时能做到这一点。并证明对于 KS 检验也有类似的结果。

13. [M24] 证明式 (13) 等价于式 (11)。

► 14. [HM26] 设 Z_i 由式 (18) 给出。如果当 $n \rightarrow \infty$ 时 Z_1, Z_2, \dots, Z_k 有界, 试利用斯特林近似公式直接证明多项式概率

$$n! p_1^{Y_1} \cdots p_k^{Y_k} / Y_1! \cdots Y_k! = e^{-V/2} / \sqrt{(2n\pi)^{k-1} p_1 \cdots p_k} + O(n^{-k/2})$$

(这个思想导致了 χ^2 检验的一个更接近于“基本原则”的证明, 而且比正文中的推导省事。)

15. [HM24] 二维极坐标习惯上都由等式 $x = r \cos \theta$ 和 $y = r \sin \theta$ 定义。为了积分的目的, 我们取 $dx dy = r dr d\theta$ 。更一般地说, 在 n 维空间中, 我们可以命

$$\begin{aligned} x_k &= r \sin \theta_1 \cdots \sin \theta_{k-1} \cos \theta_k, \quad 1 \leq k < n \\ x_n &= r \sin \theta_1 \cdots \sin \theta_{n-1} \end{aligned}$$

试证在这种情况下

$$dx_1 dx_2 \cdots dx_n = |r^{n-1} \sin^{n-2} \theta_1 \cdots \sin \theta_{n-2} dr d\theta_1 \cdots d\theta_{n-1}|$$

► 16. [HM35] 推广定理 1.2.11.3A 以求出对于大 x 和固定的 y, z ,

$$\gamma(x+1, x+z\sqrt{2x+y})/\Gamma(x+1)$$

的值。忽略答案中的 $O(1/x)$ 项, 用这个结果来找出对于大 x 和固定的 p , 方程

$$\gamma\left(\frac{x}{2}, \frac{t}{2}\right) / \Gamma\left(\frac{x}{2}\right) = p$$

的渐近解 t , 由此解释表 1 中指出的渐近公式。[提示: 见习题 1.2.11.3-8.]

17. [MH26] 设 t 是一个固定的实数, 对于 $0 \leq k \leq n$, 设

$$P_{nk}(x) = \int_{n-t}^t dx_n \int_{n-1-t}^{x_n} dx_{n-1} \cdots \int_{k+1-t}^{x_{k+2}} dx_{k+1} \int_0^{x_{k+1}} dx_k \cdots \int_0^{x_2} dx_1$$

约定 $P_{00}(x) = 1$ 。证明下列关系:

$$a) P_{nk}(x) = \int_x^{x+t} dx_n \int_{n-1}^{x_n} dx_{n-1} \cdots \int_{k+1}^{x_{k+2}} dx_{k+1} \int_t^{x_{k+1}} dx_k \cdots \int_t^{x_2} dx_1.$$

$$b) P_{n0}(x) = (x+t)^n/n! - (x+t)^{n-1}/(n-1)!.$$

$$c) P_{nk}(x) - P_{n(k-1)}(x) = \frac{(k-t)^k}{k!} P_{(n-k)0}(x-k), \text{ 其中 } 1 \leq k \leq n.$$

d) 求得 $P_{nk}(x)$ 的一个一般公式, 并应用它计算式 (24)。

18. [M20] 说明为什么 K_n 与 K_n^+ 有同样的概率分布的“简单”原因。

19. [HM48] 提出类似于 KS 检验的检验, 以便用于多元分布 $F(x_1, \dots, x_r) = \Pr(X_1 \leq x_1, \dots, X_r \leq x_r)$ 。(例如, 这样的过程可用来代替下节中的“序列检验”。)

20. [HM41] 导出 KS 分布的渐近特性的高次项并推广 (27)。

21. [M40] 尽管正文指出, KS 检验仅可应用于 $F(x)$ 是连续的分布函数的情况, 但是甚至当分布有跳跃时, 仍有可能试图计算 K_n^+ 和 K_n^- 。试分析对于各种不连续的分布 $F(x)$, K_n^+ 和 K_n^- 的概率特性。对于若干随机数的抽样, 试把所得到的统计检验的有效性与 χ^2 检验作比较。

22. [HM46] 试研究在习题 6 的答案中建议的“改进了的”KS 检验。

23. [M22] (T. Gonzalez, S. Sahni 及 W. R. Franta) (a) 假设 KS 统计量 K_n^+ 的公式(13)中的极大值在一个给定的下标 j 处出现, 其中 $\lfloor nF(X_j) \rfloor = k$. 证明 $F(X_j) = \max_{1 \leq i \leq n} \{ F(X_i) \mid \lfloor nF(X_i) \rfloor = k \}$. (b) 设计在 $O(n)$ 步内计算 K_n^+ 和 K_n^- 的一个算法(不用排序)。

► 24. [40] 对不同的 n 计算 χ^2 统计量 V 的精确分布, 实验在三个范畴上的各种概率分布(p, q, r), 其中 $p + q + r = 1$, 由此确定具有两个自由度的一个近似 χ^2 分布的精确程度。

25. [HM26] 假设对于 $1 \leq i \leq m$, $Y_i = \sum_{j=1}^n a_{ij} X_j + \mu_i$, 其中 X_1, \dots, X_n 是均值为 0、方差为 1 的独立随机变量, 而且矩阵 $A = (a_{ij})$ 有阶 n 。

a) 借助于矩阵 A 表达协方差矩阵 $C = (c_{ij})$, 其中 $c_{ij} = E(Y_i - \mu_i)(Y_j - \mu_j)$ 。

b) 证明: 如果 $\bar{C} = (\bar{c}_{ij})$ 是使得 $CCC = C$ 的任何矩阵, 则统计

$$W = \sum_{i=1}^m \sum_{j=1}^m (Y_i - \mu_i)(Y_j - \mu_j) \bar{c}_{ij}$$

等于 $X_1^2 + \dots + X_n^2$ 。[因此, 如果 X_j 有正态分布, 则 W 是有 n 个自由度的 χ^2 分布。]

*The equanimity of your average tosser of coins
depends upon a law . . . which ensures that
he will not upset himself by losing too much
nor upset his opponent by winning too often.*

你对于普通的硬币投掷持顺其自然的态度
同一个定律有关……这个定律确保
你将不会由于输得太多而心烦意乱,
也不会由于赢得太频繁而使你的对手烦恼不宁。

—TOM STOPPARD, *Rosencrantz & Guildenstern are Dead* (1966)

3.3.2 经验检验

本节我们讨论 11 种已被用于观察序列的随机性的特殊检验。对于每一检验的讨论都有两个部分:(a)对实施该检验的“插入式”描述;(b)关于这个检验的理论基础的研究。(缺乏数学训练的读者,可以跳过理论的讨论。反之,对于有数学基础的读者,即使他们从来未曾打算检验随机数生成程序,仍可以发现有关的理论是十分有趣的,因为其中包含了某些有益的组合问题。事实上本节涉及的某些话题在稍后的不同内容中也是很重要的。)

每一个检验都旨在应用于 0 和 1 之间独立和一致分布的一个实数序列

$$\langle U_n \rangle = U_0, U_1, U_2, \dots \quad (1)$$

某些检验并不是为实数序列(1),而主要是为取整数值的序列设计的。在这种情况下,检验中使用的辅助序列

$$Y_n = Y_0, Y_1, Y_2, \dots \quad (2)$$

由规则

$$\langle Y_n \rangle = \lfloor dU_n \rfloor \quad (3)$$

定义。这是个整数序列,在 0 与 $d-1$ 之间独立和一致地分布。数 d 是为了方便而

选择的;例如,在一台二进计算机上,可以有 $d = 64 = 2^6$,使得 Y_n 表示 U_n 的二进表示的六位最高位。 d 的值应当充分大,以使这个检验富有意义,但也不能大到使检验变得难以进行而不实用。

在这一节,量 U_n, Y_n 和 d 自始至终有上文的意义,但 d 的值在不同的检验中可以是不同的。

A. 等分布检验(频率检验) 序列(1)必须满足的第一个要求是它的数实际上一致分布于 0 与 1 之间。有两种方式进行这种检验:(a)对 $F(x) = x, 0 \leq x \leq 1$, 用 KS 检验。(b)命 d 是一个方便的数,例如在一台十进制计算机上的 100,在一台二进制计算机上的 64 或 128,而且使用序列(3)来代替序列(1)。对于每个整数 $r, 0 \leq r < d$ 及 $0 \leq j < n$, 计算 $Y_j = r$ 的次数,而后对于每个范畴利用 $k = d$ 和概率 $p_s = 1/d$,应用 χ^2 检验。

这个检验背后的理论已在 3.3.1 节叙述过了。

B. 序列检验 更一般地,我们要求相继的数偶独立地一致分布。太阳升起的次数总是与它落下的次数一样多,但这并不使它的运动成为随机的。

为进行序列检验,我们只需计算对于 $0 \leq j < n$, 数偶 $(Y_{2j}, Y_{2j+1}) = (q, r)$ 出现的次数;这些计数是对 $0 \leq q, r < d$ 的每个整数偶 (q, r) 进行的,而 χ^2 检验被应用于这些 $k = d^2$ 个范畴上,其中每个范畴的概率是 $1/d^2$ 。和等分布检验一样, d 可以是任何方便的数,但它将稍微小于上面提议的值,因为一个有效的 χ^2 检验应当使 n 相对于 k 说来足够大(比如说,至少 $n \geq 5d^2$)。

显然,我们可以把这个检验推广到三元组、四元组等等,以代替数偶(见习题 2);然而,必须急剧地减小 d 的值以避免范畴太多。因此,当考虑四元组和更大量的相邻元素时,我们利用像下边所述的扑克检验或极大值检验这类精确度较差的检验。

注意,在这个检验中,使用了序列(2)的 $2n$ 个数来进行 n 次观察。对于数偶 $(Y_0, Y_1), (Y_1, Y_2), \dots, (Y_{n-1}, Y_n)$ 实行序列检验是错误的,读者能看出为什么吗?我们倒可以对数偶 (Y_{2j+1}, Y_{2j+2}) 实行另一个序列检验,而且期望这个序列同时通过这两个检验,请记住这些检验并不是彼此独立的。代替地,George Marsaglia 已经证明如果使用数偶 $(Y_0, Y_1), (Y_1, Y_2), \dots, (Y_{n-1}, Y_n)$, 并且如果对相同的 d 值来对 Y_0, \dots, Y_{n-1} 使用通常的 χ^2 检验方法计算其对于序列检验的统计 V_2 和对于频率检验的统计 V_1 , 则当 n 很大时, $V_2 - V_1$ 应当有 χ^2 分布和 $d(d-1)$ 的自由度(参见习题 24)。

C. 间隔检验 间隔检验用来考察在某个范围内 U_j 出现的“间隔”长度。如果 α 和 β 是满足 $0 \leq \alpha < \beta \leq 1$ 的两个实数,我们要考虑连续的子序列 $U_j, U_{j+1}, \dots, U_{j+r}$ 的长度,其中 U_{j+r} 位于 α 和 β 之间,但其它的 U 则不然(这 $r+1$ 个数的子序列表示长度为 r 的一个间隔)。

算法 G(间隔检验的数据) 任意给定 α 和 β 的值,把下列算法应用于序列(1),它计算长度为 $0, 1, \dots, t-1$ 的间隔个数,以及长度大于或等于 t 的间隔的个数,直到造出 n 个间隔的表格来。

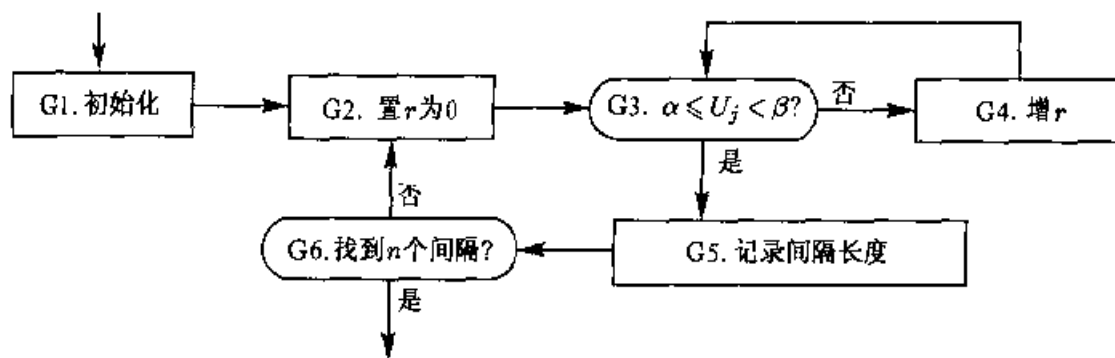


图6 搜集间隔检验的数据(“集券检验”和“运行检验”的算法均类似)

G1. [初始化] 置 $j \leftarrow -1, s \leftarrow 0$, 并且对于 $0 \leq r \leq t$, 置 $\text{COUNT}[r] \leftarrow 0$ 。

G2. [置 r 为 0] 置 $r \leftarrow 0$ 。

G3. [$\alpha \leq U_j < \beta$?] j 增 1, 如果 $U_j \geq \alpha$ 且 $U_j < \beta$, 则转到 G5。

G4. [增 r] r 增 1, 并返回 G3。

G5. [记录间隔长度] (长度为 r 的一个间隔现已找到。)如果 $r \geq t$, 则 $\text{COUNT}[t]$ 增 1, 否则 $\text{COUNT}[r]$ 增 1。

G6. [找到 n 个间隔了?] s 增 1。如果 $s < n$, 则返回步骤 G2。 |

实施了此算法之后,应用 χ^2 检验于 $\text{COUNT}[0], \text{COUNT}[1], \dots, \text{COUNT}[t]$, 共 $k = t+1$ 个值,并采用下面的概率:

$$p_r = p(1-p)^r, \quad 0 \leq r \leq t-1; \quad p_t = (1-p)^t \quad (4)$$

这里 $p = \beta - \alpha$, 即 $\alpha \leq U_j < \beta$ 的概率。通常, n 和 t 的值应选择成使 $\text{COUNT}[r]$ 的每个值都预期为 5 或更大, 愈大愈好。

为了省去步骤 G3 中的比较之一, 常以 $\alpha = 0$ 或 $\beta = 1$ 应用间隔检验。特殊情况 $(\alpha, \beta) = (0, \frac{1}{2})$ 或 $(\frac{1}{2}, 1)$ 的检验, 有时分别称做“中下运行”或“中上运行”。

式(4)中的概率很容易导出, 因此把推导留给读者。注意, 如上所述的间隔检验考察了 n 个间隔的长度, 它并不考察 n 个数之间的间隔长度。如果序列 $\langle U_n \rangle$ 是充分非随机的, 则算法 G 可能不终止。除此之外, 还有考察一个固定数目的 U 的其它间隔检验(见习题 5)。

D. 扑克检验(分划检验) “经典的”扑克检验考虑由五个相继整数组成的 n 个组 $(Y_{5j}, Y_{5j+1}, \dots, Y_{5j+4}), 0 \leq j < n$, 并观察每一个五元组(无顺序)与下列七种形式中哪一种相匹配:

全不同:	$abcde$	客满:	$aaabb$
一对:	$aabcd$	四个同:	$aaaab$

两对: $aabbc$ 五个同: $aaaaa$

三个同: $aaabc$

χ^2 检验是以在每个范畴中五元组的数目为基础的。

为了便于程序设计,需要将这种检验改得简单一点,一个好的折中方案是仅仅计算五元集合中不同值的个数。这样,我们有五个范畴:

五个值 = 全不同;

四个值 = 一对;

三个值 = 两对,或三个同;

二个值 = 客满,或四个同;

一个值 = 五个同类。

这种分类更易于系统地确定,而且检验也几乎同样好。

一般地说,我们可以考虑 n 个由 k 个相继的数组成的组,而且我们可以计算具有 r 个不同值的 k 元组的个数。然后利用有 r 个不同值的概率

$$p_r = \frac{d(d-1)\cdots(d-r+1)}{d^k} \left\{ \begin{matrix} k \\ r \end{matrix} \right\} \quad (5)$$

做 χ^2 检验。(斯特林数 $\left\{ \begin{matrix} k \\ r \end{matrix} \right\}$ 是在 1.2.6 节中定义的,利用那里给出的公式容易计算它们。)由于当 $r=1$ 或 2 时,概率 p_r 非常小,所以在应用 χ^2 检验之前,一般把低概率的一些范畴归并在一起。

为了推导出 p_r 的适当公式,必须计算在 0 与 $d-1$ 之间数的 d^k 个 k 元组当中,有多少个恰有 r 个不同的元素,并以 d^k 来除这个总数。由于 $d(d-1)\cdots(d-r+1)$ 是从 d 个对象的一个集合中选出 r 个有序事物的数目,故我们仅须证明 $\left\{ \begin{matrix} k \\ r \end{matrix} \right\}$ 是把 k 个元素的一个集合恰巧分成 r 个部分的方法的数目。因此,习题 1.2.6-64 就完成了式(5)的推导。

E. 集券检验 这一个检验与扑克检验有关,就像间隔检验与频率检验有关那样。使用序列 Y_0, Y_1, \dots , 并观察为得到由 0 到 $d-1$ 的整数的“完备的集合”所要求的区段 $Y_{j+1}, Y_{j+2}, \dots, Y_{j+r}$ 的长度。算法 C 精确地描述了这个方法:

算法 C(集券检验的数据) 给定满足 $0 \leq Y_j < d$ 的整数序列 Y_0, Y_1, \dots , 本算法计算 n 个相继的“集券”区段的长度。待这个算法结束时, $\text{COUNT}[r]$ 包含有长度 r 的区段个数,其中 $d \leq r < t$, 而 $\text{COUNT}[t]$ 含有长度 $\geq t$ 的区段个数。

C1. [初始化] 置 $j \leftarrow -1, s \leftarrow 0$, 且对于 $d \leq r \leq t$, 置 $\text{COUNT}[r] \leftarrow 0$ 。

C2. [置 q, r 为 0] 置 $q \leftarrow r \leftarrow 0$, 且对于 $0 \leq k < d$, 置 $\text{OCCURS}[k] \leftarrow 0$ 。

C3. [下一次观察] r 和 j 增 1, 如果 $\text{OCCURS}[Y_j] \neq 0$, 则重复这一步骤。

C4. [完备的集合?] 置 $\text{OCCURS}[Y_j] \leftarrow 1, q \leftarrow q + 1$ (所观察的这个于序列至此包含 q 个不同的值; 如果 $q = d$, 则得到一个完备的集合)。如果 $q < d$, 则

返回步骤 C3。

C5. [记录长度] 如果 $r \geq t$, 则 $\text{COUNT}[t]$ 增 1, 否则 $\text{COUNT}[r]$ 增 1。

C6. [找到了 n 个?] s 增加 1, 如果 $s < n$, 则返回步骤 C2。 ▮

该算法的一个例子见习题 7。可以想像, 一个小孩要搜集 d 种类型的赠券, 这些赠券随机地分布在他的早餐麦片盒中。他必须不断地吃麦片, 直到找齐所有类型的赠券为止。

在算法 C 计算了 n 个长度之后, 即可应用 χ^2 检验于 $\text{COUNT}[d], \text{COUNT}[d+1], \dots, \text{COUNT}[t]$, 其中 $k = t - d + 1$ 。相应的概率是

$$p_r = \frac{d!}{d^r} \left\{ \frac{r-1}{d-1} \right\}, \quad d \leq r < t; \quad p_t = 1 - \frac{d!}{d^{t-1}} \left\{ \frac{t-1}{d} \right\} \quad (6)$$

为推导这些概率, 我们略作说明: 如果 q_r 是表示长度为 r 的一个子序列是不完备的概率, 则由式(5)得

$$q_r = 1 - \frac{d!}{d^r} \left\{ \frac{r}{d} \right\}$$

因为不完备意味着有一个 r 元组, 其元素没有全都不同的 d 个值。于是由关系 $p_t = q_{t-1}$ 和对于 $d \leq r < t$, $p_r = q_{r-1} - q_r$ 得到式(6)。

与集券检验的推广有关, 可得到一些公式, 请见习题 9 和 10, 以及 George Pólya, *Zeitschrift für angewandte Math. und Mech* 10 (1930), 96 ~ 97; Hermann von Schelling, *AMM* 61 (1954), 306 ~ 311。

F. 排列检验 把输入序列分为 n 组, 每组 t 个元素, 即 $(U_{jt}, U_{jt+1}, \dots, U_{jt+t-1})$, 其中 $0 \leq j < n$ 。在每组中的元素可以有 $t!$ 种可能的相对顺序; 计算每种排序出现的次数, 以 $k = t!$ 并以每种排序的概率 $1/t!$, 应用 χ^2 检验。

例如, 如果 $t = 3$, 根据 $U_{3j} < U_{3j+1} < U_{3j+2}$ 或 $U_{3j} < U_{3j+2} < U_{3j+1}$ 或 \dots 或 $U_{3j+2} < U_{3j+1} < U_{3j}$, 我们将有六个范畴。在这种检验中, 假定诸 U 不相等。由于两个 U 相等的概率是 0, 这样的假定是正当的。

在一台计算机上实施此检验的一种方便的方式是使用以下的算法, 它本身就很令人感兴趣。

算法 P(分析一个排列) 给定不同元素 (U_1, \dots, U_t) 的一个序列, 计算一个整数 $f(U_1, \dots, U_t)$ 使得

$$0 \leq f(U_1, \dots, U_t) < t!$$

而且 $f(U_1, \dots, U_t) = f(V_1, \dots, V_t)$, 当且仅当 (U_1, \dots, U_t) 和 (V_1, \dots, V_t) 有相同的相对排序。

P1. [初始化] 置 $r \leftarrow t, f \leftarrow 0$ 。(在算法执行期间, 我们有 $0 \leq f < t!/r!$ 。)

P2. [找极大值] 找 $\{U_1, \dots, U_r\}$ 的极大值, 并假设 U_s 是极大值。置 $f \leftarrow r \cdot f + s - 1$ 。

P3. [交换] 交换 $U_r \leftrightarrow U_s$ 。

P4. [减少 r] r 减 1。如果 $r > 1$, 则返回步骤 P2。 ▮

当这个算法停止时, 序列 (U_1, \dots, U_t) 将已经排为递增次序的。为证明结果 f 惟一地表征 (U_1, \dots, U_t) 的初始次序, 注意算法 P 可以向后运行:

对于 $r = 2, 3, \dots, t$,
置 $s \leftarrow f \bmod r, f \leftarrow \lfloor f/r \rfloor$,
并交换 $U_r \leftrightarrow U_{s+1}$ 。

容易看出, 这将消除步骤 P2~P4 的影响; 因此没有两个排列可以产生相同的 f 值, 因而算法 P 的效果与前面所说的一致。

奠定算法 P 的基本思想是称做“阶乘数系统”的混合进制表示。在 $0 \leq f < t!$ 范围中的每个整数可惟一地写成形式

$$f = (\dots(c_{t-1} \times (t-1) + c_{t-2}) \times (t-2) + \dots + c_2) \times 2 + c_1 = \\ (t-1)!c_{t-1} + (t-2)!c_{t-2} + \dots + 2!c_2 + 1!c_1 \quad (7)$$

其中“数字” c_j 是对于 $1 \leq j < t$ 满足

$$0 \leq c_j \leq j \quad (8)$$

的整数。在算法 P 中, 当对于给定的 r 值执行步骤 P2 时, $c_{r-1} = s - 1$ 。

G. 运行检验 一个序列也可以做“上行运行”和“下行运行”检验。这意味着, 我们考察原来序列的单调子序列的长度, 即递增或递减的区段的长度。

作为运行的精确定义的一个例子, 我们来考虑十个数的序列“1298536704”。在左端和右端, 以及每当 $X_j > X_{j+1}$ 时在 X_j 和 X_{j+1} 之间置一竖线, 我们得到

$$1 \ 1 \ 2 \ 9 \ 8 \ 5 \ 3 \ 6 \ 7 \ 0 \ 4 \mid \quad (9)$$

它显示了“上行运行”: 有一个长度为 3 的运行, 后边跟着长度为 1 的两个运行, 后边又接着长度为 3 的另一个运行, 再接着长度为 2 的一个运行, 习题 12 的算法说明怎样来造“上行运行”长度的表。

不像间隔检验和集券检验那样(在许多方面它们类似于这个检验), 我们将不应用 χ^2 检验于运行计算, 因为相邻的运行不是独立的。一个长的运行势必跟随一个短的运行, 而且反过来也是一样。这种独立性的缺乏, 足以使一个直截了当的 χ^2 检验失效。替代的是当运行长度已经如习题 12 那样确定时, 可以计算下面的统计量:

$$V = \frac{1}{n - 6} \sum_{1 \leq i, j \leq 6} (\text{COUNT}[i] - nb_i)(\text{COUNT}[j] - nb_j) a_{ij} \quad (10)$$

其中 n 是这个序列的长度, 系数矩阵 $A = (a_{ij})_{1 \leq i, j \leq 6}$ 和 $B = (b_i)_{1 \leq i \leq 6}$ 由

$$A = \begin{pmatrix} 4529.4 & 9044.9 & 13568 & 18091 & 22615 & 27892 \\ 9044.9 & 18097 & 27139 & 36187 & 45234 & 55789 \\ 13568 & 27139 & 40721 & 54281 & 67852 & 83685 \\ 18091 & 36187 & 54281 & 72414 & 90470 & 111580 \\ 22615 & 45234 & 67852 & 90470 & 113262 & 139476 \\ 27892 & 55789 & 83685 & 111580 & 139476 & 172860 \end{pmatrix} \quad B = \begin{pmatrix} \frac{1}{6} \\ \frac{5}{24} \\ \frac{11}{120} \\ \frac{19}{720} \\ \frac{29}{5040} \\ \frac{1}{840} \end{pmatrix} \quad (11)$$

给出。(这里所示的 a_{ij} 的值仅仅是近似的;利用以下推导的公式,可以求得其精确值。)当 n 很大时,式(10)中的统计量 V 应具有自由度 6(不是 5)的 χ^2 分布。比如说, n 的值应当是 4000 或更大。同样的检验可应用于“下行运行”。

习题 14 中有一个更简单和更实用的运行检验,所以,只对检验随机数生成程序感兴趣的读者可跳过以下数页,并在考虑习题 14 之后就转到“ t 的极大值检验”去。另一方面,对于相互关联的运行,从数学的观点探讨如何建立一个复杂的运行检验是有启发性的,所以我们现在将偏离主题片刻。

给定 n 个元素的任何排列,如果位置 i 是长度 p 或更大的一个递增运行的开头,命 $Z_{pi} = 1$,否则命 $Z_{pi} = 0$ 。例如,考虑 $n = 10$ 的排列(9),有

$$Z_{11} = Z_{21} = Z_{31} = Z_{14} = Z_{15} = Z_{16} = Z_{26} = Z_{36} = Z_{19} = Z_{29} = 1$$

面所有其它的 Z 皆为 0。在此记号下

$$R'_p = Z_{p1} + Z_{p2} + \cdots + Z_{pn} \quad (12)$$

是长度 $\geq p$ 的运行的个数,而且

$$R_p = R'_p - R'_{p+1} \quad (13)$$

是长度恰为 p 的运行的个数。我们的目标是要计算 R_p 的平均值,还要计算协方差

$$\text{covar}(R_p, R_q) = \text{mean}((R_p - \text{mean}(R_p))(R_q - \text{mean}(R_q)))^{\text{①}}$$

它测量 R_p 和 R_q 的相关性。这些平均值应计算作所有 $n!$ 个排列的集合上的平均值。

式(12)和(13)说明,问题的答案可借助于 Z_{pi} 和 $Z_{pi}Z_{qj}$ 的平均值来表达,于是作为推导的头一步,我们得到下列的结果(假定 $i < j$):

$$\frac{1}{n!} \sum Z_{pi} = \begin{cases} \frac{p + \delta_{i1}}{(p+1)!}, & \text{如果 } i \leq n - p + 1 \\ 0, & \text{否则} \end{cases} \quad (14)$$

$$\frac{1}{n!} \sum Z_{pi}Z_{qj} = \begin{cases} \frac{(p + \delta_{i1})q}{(p+1)!(q+1)!}, & \text{如果 } i + p < j \leq n - q + 1 \\ \frac{p + \delta_{i1}}{(p+1)!q!} - \frac{p + q + \delta_{i1}}{(p+q+1)!}, & \text{如果 } i + p = j \leq n - q + 1 \\ 0, & \text{否则} \end{cases}$$

符号 \sum 表示对所有可能的排列求和。为了解释这里所涉及的计算,我们来处理最困难的情况,即当 $i + p = j \leq n - q + 1$, 且 $i > 1$ 时。量 $Z_{pi}Z_{qj}$ 或者是 0 或者为 1, 所以求和应由计算所有使 $Z_{pi} = Z_{qj} = 1$ 的排列 $U_1 U_2 \cdots U_n$ 的计数所组成,即使得

$$U_{i-1} > U_i < \cdots < U_{i+p-1} > U_{i+p} < \cdots < U_{i+p+q-1} \quad (15)$$

的所有排列。这样排列的个数可枚举如下:有 $\binom{n}{p+q+1}$ 种方式来选择具有式(15)

① covar 表示协方差, mean 表示平均值。——译者注

中所示状况的元素。有

$$(p+q+1) \binom{p+q}{p} - \binom{p+q+1}{p+1} - \binom{p+q+1}{1} + 1 \quad (16)$$

种方式把它们安排成次序(15),如同习题(13)中所示;而且有 $(n-p-q-1)!$ 种方式来排列余下的元素。于是,总共有 $\binom{n}{p+q+1}(n-p-q-1)!$ 乘(16)种方式,将其除以 $n!$,即得到所求的公式。

从关系(14),一个稍长的计算给出

$$\text{mean}(R'_p) = \frac{(n+1)p}{(p+1)!} - \frac{(p-1)}{p!}, 1 \leq p \leq n \quad (17)$$

$$\begin{aligned} \text{covar}(R'_p, R'_q) &= \text{mean}(R'_p R'_q) - \text{mean}(R'_p) \text{mean}(R'_q) = \\ &= \sum_{1 \leq i, j \leq n} \frac{1}{n!} \sum Z_{pi} Z_{qj} - \text{mean}(R'_p) \text{mean}(R'_q) = \\ &= \begin{cases} \text{mean}(R'_t) + f(p, q, n), & \text{若 } p+q \leq n \\ \text{mean}(R'_t) - \text{mean}(R'_p) \text{mean}(R'_q), & \text{若 } p+q > n \end{cases} \end{aligned} \quad (18)$$

其中 $t = \max(p, q)$, $s = p+q$, 而且

$$\begin{aligned} f(p, q, n) &= (n+1) \left(\frac{s(1-pq) + pq}{(p+1)!(q+1)!} - \frac{2s}{(s+1)!} \right) + 2 \left(\frac{s-1}{s!} \right) + \\ &= \frac{(s^2 - s - 2)pq - s^2 - p^2 q^2 + 1}{(p+1)!(q+1)!} \end{aligned} \quad (19)$$

可惜的是,协方差这一表达式过于复杂,但它对于如上所述运行检验的成功是必要的。从这些公式容易计算

$$\begin{aligned} \text{mean}(R_p) &= \text{mean}(R'_p) - \text{mean}(R'_{p+1}) \\ \text{covar}(R_p, R'_q) &= \text{covar}(R'_p, R'_q) - \text{covar}(R'_{p+1}, R'_q) \\ \text{covar}(R_p, R_q) &= \text{covar}(R_p, R'_q) - \text{covar}(R_p, R'_{q+1}) \end{aligned} \quad (20)$$

在 *Annals Math. Stat.* **15** (1944), 163~165 中, J. Wolfowitz 证明了假定平均值和协方差表达如上, 则当 $n \rightarrow \infty$ 时, 量 $R_1, R_2, \dots, R_{t-1}, R'_t$ 变成正态分布的。这意味着如下的运行检验是有效的: 给定 n 个随机数的一个序列, 计算对于 $1 \leq p < t$, 长度为 p 的运行 R_p 的个数, 以及长度为 t 或更大的运行 R'_t 的个数。令

$$\begin{aligned} Q_1 &= R_1 - \text{mean}(R_1), \dots, Q_{t-1} = R_{t-1} - \text{mean}(R_{t-1}) \\ Q_t &= R'_t - \text{mean}(R'_t) \end{aligned} \quad (21)$$

构造诸 R' 的协方差的矩阵 C ; 例如, $C_{13} = \text{covar}(R_1, R_3)$, 而 $C_{1t} = \text{covar}(R_1, R'_t)$ 。当 $t=6$ 时, 如果 $n \geq 12$, 我们有

$$C = nC_1 + C_2 \quad (22)$$

其中

$$C_1 = \begin{bmatrix} \frac{23}{180} & \frac{-7}{360} & \frac{5}{336} & \frac{-433}{60480} & \frac{-13}{5670} & \frac{-121}{181440} \\ \frac{-7}{360} & \frac{2843}{20160} & \frac{-989}{20160} & \frac{-7159}{362880} & \frac{-10019}{1814400} & \frac{1303}{907200} \\ \frac{-5}{336} & \frac{989}{20160} & \frac{54563}{907200} & \frac{-21311}{1814400} & \frac{62369}{19958400} & \frac{-7783}{9979200} \\ \frac{-433}{60480} & \frac{-7159}{362880} & \frac{21311}{1814400} & \frac{886657}{39916800} & \frac{257699}{239500800} & \frac{-62611}{239500800} \\ \frac{-13}{5670} & \frac{-10019}{1814400} & \frac{-62369}{19958400} & \frac{-257699}{239500800} & \frac{29874811}{5448643200} & \frac{-1407179}{21794572800} \\ \frac{-121}{181440} & \frac{-1303}{907200} & \frac{-7783}{9979200} & \frac{-62611}{239500800} & \frac{-1407179}{21794572800} & \frac{2134697}{1816214400} \end{bmatrix}$$

$$C_2 = \begin{bmatrix} \frac{83}{180} & \frac{-29}{180} & \frac{-11}{210} & \frac{-41}{12096} & \frac{91}{25920} & \frac{41}{18144} \\ \frac{-29}{180} & \frac{-305}{4032} & \frac{319}{20160} & \frac{2557}{72576} & \frac{10177}{604800} & \frac{413}{64800} \\ \frac{-11}{210} & \frac{319}{20160} & \frac{-58747}{907200} & \frac{19703}{604800} & \frac{239471}{19958400} & \frac{39517}{9979200} \\ \frac{-41}{12096} & \frac{2557}{72576} & \frac{19703}{604800} & \frac{-220837}{4435200} & \frac{1196401}{239500800} & \frac{360989}{239500800} \\ \frac{91}{25920} & \frac{10177}{604800} & \frac{239471}{19958400} & \frac{1196401}{239500800} & \frac{-139126639}{7264857600} & \frac{4577641}{10897286400} \\ \frac{41}{18144} & \frac{413}{64800} & \frac{39517}{9979200} & \frac{360989}{239500800} & \frac{4577641}{10897286400} & \frac{-122953057}{21794572800} \end{bmatrix}$$

现在构造矩阵 C 的逆 $A = (a_{ij})$, 并计算 $\sum_{i,j=1}^t Q_i Q_j a_{ij}$ 。对于大的 n , 结果有近似于自由度 t 的 χ^2 分布。

前面给出的矩阵 (11) 是 C_1 的逆, 计算至五位有效数字。真正的逆 A 是 $n^{-1}C_1^{-1} - n^{-2}C_1^{-1}C_2C_1^{-1} + n^{-3}C_1^{-1}C_2C_1^{-1}C_2C_1^{-1} - \dots$, 结果 $C_1^{-1}C_2C_1^{-1}$ 非常接近于 $-6C^{-1}$ 。因此 $V \approx Q^T C_1^{-1} Q / (n-6)$ 。

H. t 的极大值检验 对于 $0 \leq j < n$, 令 $V_j = \max(U_j, U_{j+1}, \dots, U_{j+t-1})$ 。现在, 对于有分布函数 $F(x) = x^t$ ($0 \leq x \leq 1$) 的序列 V_0, V_1, \dots, V_{n-t} 应用 KS 检验。或者, 把等分布检验应用于序列 $V'_0, V'_1, \dots, V'_{n-1}$ 。

为验证这个检验, 须证明 V_j 的分布函数是 $F(x) = x^t$ 。 $\max(U_1, U_2, \dots, U_t) \leq x$ 的概率是 $U_1 \leq x$ 且 $U_2 \leq x$ 且 \dots 且 $U_t \leq x$ 的概率, 这是诸个别概率的乘积, 即 $x \cdot x \cdot \dots \cdot x = x^t$ 。

I. 冲突检验 仅当在每个范畴中预期有非平凡个数的项时, 才可进行 χ^2 检验。但是, 当范畴数比观察数大得多时, 可以使用另一种类型的检验; 这一检验同“散列”(hashing) 有关, 散列是我们将第 6 章中加以研究的信息检索的一个重要方法。

假设有 m 个瓮, 我们随意地把 n 个球投入到这些瓮中, 这里 m 比 n 大得多。大多数球将落入原先是空的瓮中, 但若一个球落入至少已经含一个球的瓮中, 我们

便说出现一个“冲突”。冲突检验计算冲突的次数,而且如果一个生成程序不引起太多或太少的冲突,则说它通过这个检验。

为了确定这些思想,假设 $m = 2^{20}$ 和 $n = 2^{14}$ 。那么,平均每瓮仅仅接收一个球的 $1/64$ 。一个给定的瓮恰含 k 个球的概率为 $p_k = \binom{n}{k} m^{-k} (1 - m^{-1})^{n-k}$, 所以每个瓮的期望冲突数为

$$\sum_{k \geq 1} (k-1) p_k = \sum_{k \geq 0} k p_k - \sum_{k \geq 1} p_k = \frac{n}{m} - 1 + p_0$$

由于 $p_0 = (1 - m^{-1})^n = 1 - nm^{-1} + \binom{n}{2} m^{-2} - \text{较小的项}$, 故我们求得所有 m 个瓮发生的总冲突数平均比 $n^2/(2m) = 128$ 少一点点(实际的值是 ≈ 127.33)。

我们可以使用冲突检验来测量在很大维数下的一个随机数生成程序。例如,当 $m = 2^{20}$ 和 $n = 2^{14}$ 时,我们可以通过令 $d = 2$ 并对 $0 \leq j < n$ 构造 20 维向量 $V_j = (Y_{20j}, Y_{20j+1}, \dots, Y_{20j+19})$ 检验一个数值生成程序的 20 维随机性。为了确定冲突,保持 $m = 2^{20}$ 个二进位的一张表,对向量 V_j 的每一可能的值用一个二进位表示;在一台每个字有 32 个二进位的计算机上,这总计为 2^{15} 个字。开始时,这个表的所有 2^{20} 个二进位清为 0;然后对于每个 V_j , 如果对应的二进位已为 1, 我们就记录一个冲突,否则就把这个二进位置为 1。这个检验也可用于 $d = 4$ 的 10 维向量,等等。

为了判断这个检验是否通过,我们可以使用当 $m = 2^{20}$ 和 $n = 2^{14}$ 时的下列百分点表:

冲突 \leq	101	108	119	126	134	145	153
概率	.009	.043	.244	.476	.742	.946	.989

奠定这些概率的理论基础和我们在扑克检验中所使用的,即等式(5),是一样的; c 个冲突出现的概率就是 $n - c$ 个瓮被占据的概率,即

$$\frac{m(m-1)\cdots(m-n+c+1)}{m^n} \left\{ \begin{matrix} n \\ n-c \end{matrix} \right\}$$

尽管 m 和 n 非常大,但不难利用以下方法计算这些概率:

算法 S(关于冲突检验的百分点) 给定 m 和 n , 这个算法确定当 n 个球散列于 m 个瓮时冲突数的分布。为进行计算,使用浮点数的一个辅助数组 $A[0], A[1], \dots, A[n]$; 实际上,仅当 $j_0 \leq j \leq j_1$ 时, $A[j]$ 才将是非零的,而 $j_1 - j_0$ 将至多有 $\log n$ 的阶,所以只要使用相当少的存储就可能得到它。

S1. [初始化] 对于 $0 \leq j \leq n$ 置 $A[j] \leftarrow 0$; 然后置 $A[1] \leftarrow 1$ 和 $j_0 \leftarrow j_1 \leftarrow 1$ 。然后执行步骤 S2 恰 $n-1$ 次并转到步骤 S3 继续进行。

S2. [修改概率] (每对应于投掷一个球到一个瓮中就执行这步一次; $A[j]$ 表示恰有 j 个瓮被占据的概率。)置 $j_1 \leftarrow j_1 + 1$, 然后对于 $j \leftarrow j_1, j_1 - 1, \dots, j_0$ (以此顺序), 置 $A[j] \leftarrow (j/m)A[j] + ((1 + 1/m) - (j/m))A[j-1]$ 。如果随着这个计算的结果 $A[j]$ 变成非常小,比方说 $A[j] < 10^{-20}$, 则置

$A[j] \leftarrow 0$; 而且在这种情况下, 若 $j = j_1$, 则 j_1 减 1, 或若 $j = j_0$ 则 j_0 加 1。

S3. [计算答案] 在这步中, 利用含有我们感兴趣的百分比的一张辅助表 $(T_1, T_2, \dots, T_{\text{tmax}}) = (.01, .05, .25, .50, .75, .95, .99, 1.00)$ 。置 $p \leftarrow 0, t \leftarrow 1$ 和 $j \leftarrow j_0 - 1$ 。进行下列的迭代直到 $t = \text{tmax}$ 为止: j 增 1, 并置 $p \leftarrow p + A[j]$; 然后如果 $p > T_t$, 则输出 $n - j - 1$ 和 $1 - p$ (意味着至多有 $n - j - 1$ 个冲突的概率是 $1 - p$), 并重复地对 t 加 1 直到 $p \leq T_t$ 为止。 ▮

J. 生日间隔检验 George Marsaglia 在 1984 年引进了一个新型的检验: 如同在冲突检验一样, 我们把 n 个球投入到 m 个瓮中, 但现在我们把瓮想像为“一年的每一天”, 而把球想像为“生日”。假设生日是 (Y_1, \dots, Y_n) , 其中 $0 \leq Y_i < m$ 。按非递减次序对这些生日排序 $Y_{(1)} \leq \dots \leq Y_{(n)}$; 然后定义 n 个“间隔” $S_1 = Y_{(2)} - Y_{(1)}, \dots, S_{n-1} = Y_{(n)} - Y_{(n-1)}, S_n = Y_{(1)} + m - Y_{(n)}$; 最后对这些间隔排序, $S_{(1)} \leq \dots \leq S_{(n)}$ 。令 R 是间隔相等的个数, 即使得 $1 < j \leq n$ 且 $S_{(j)} = S_{(j-1)}$ 的下标 j 的个数。当 $m = 2^{25}$ 和 $n = 512$ 时, 我们将有

$R =$	0	1	2	3 或以上
概率	.368801577	.369035243	.183471182	.078691997

(对于这里的 m 和 n 值相等间隔的平均数应近似于 1。)重复此检验, 比方说 1000 次, 并进行具有 3 个自由度的 χ^2 检验, 来把经验的 R 和正确的分布作比较; 这将告知这个生成程序是否相当合理地产生随机生日间隔。习题 28~30 建立了关于这个检验的理论并对其它的 m 和 n 的值给出了若干公式。

这样一个生日间隔检验之所以重要是由于这样一个奇怪的事实, 即延搁的斐波那契生成程序一致地不能通过它, 尽管它们都很好通过其它传统的检验。[这样的失败的引人注目的例子是由 Marsaglia, Zaman 和 Tsang 在 *Stat. and Prob. Letters* 8 (1990), 35~39 上报告的。]例如考虑式 3.3.2-(7) 的序列

$$X_n = (X_{n-24} + X_{n-55}) \bmod m$$

这个序列的数满足

$$X_n + X_{n-86} \equiv X_{n-24} + X_{n-31} \pmod{m}$$

因为两边都同余于 $X_{n-24} + X_{n-55} + X_{n-86}$, 因此两对差相等:

$$X_n - X_{n-24} \equiv X_{n-31} - X_{n-86}$$

和

$$X_n - X_{n-31} \equiv X_{n-24} - X_{n-86}$$

每当 x 相当接近于 X_{n-24} 或 X_{n-31} (因为它应当是在一个真正随机的序列中) 时, 这个差有在两个间隔中出现的很好的机会。因此我们就有大得多的相等的情况。平均说来典型地 $R \approx 2$, 而不是 1。但是如果我们从 R 中扣除由所述的同余产生的任何相等的间隔, 则得到的统计 R' 通常确实通过生日检验。(避免失败的一个方法是抛弃序列的某些元素, 比如说只使用 X_0, X_2, X_4, \dots 作为随机数; 然后我们绝不取集

合 $|X_n, X_{n-24}, X_{n-31}, X_{n-86}|$ 的所有四个元素, 因而生日间隔就没有问题了。避免问题的一个甚至更好的方法, 是如 Lüscher 所建议的那样, 抛弃连续的成批的数, 请参见 3.2.2 小节。) 类似的评述也适用于习题 3.2.1.1-14 的带借位减法和带进位加法生成程序。

K. 序列相关检验 我们也可以计算下列统计数字:

$$C = \frac{n(U_0 U_1 + U_1 U_2 + \cdots + U_{n-2} U_{n-1} + U_{n-1} U_0) - (U_0 + U_1 + \cdots + U_{n-1})^2}{n(U_0^2 + U_1^2 + \cdots + U_{n-1}^2) - (U_0 + U_1 + \cdots + U_{n-1})^2} \quad (23)$$

这是“序列相关系数”, 它是测量 U_{j+1} 依赖于 U_j 的程度的一个量度。

相关系数经常出现于统计学中; 如果我们有 n 个量 U_0, U_1, \dots, U_{n-1} 和 n 个其它的量 V_0, V_1, \dots, V_{n-1} , 则它们之间的相关系数定义为

$$C = \frac{n \sum (U_j V_j) - (\sum U_j)(\sum V_j)}{\sqrt{(n \sum U_j^2 - (\sum U_j)^2)(n \sum V_j^2 - (\sum V_j)^2)}} \quad (24)$$

在此公式中, 所有的求和都跑遍范围 $0 \leq j < n$; 式 (23) 是特殊情况 $V_j = U_{(j+1) \bmod n}$; 当 $U_0 = U_1 = \cdots = U_{n-1}$ 或 $V_0 = V_1 = \cdots = V_{n-1}$ 时, 式 (24) 的分母为 0; 我们的讨论排除这一情况。

相关系数总处于 -1 与 $+1$ 之间。当它为 0 或非常小时, 它指示量 U_j 和 V_j (相对地说) 是彼此独立的, 但当相关系数为 ± 1 时, 则它指示完全线性相关; 事实上, 在这种情况下, 对所有 j , 对于某常数 α 和 β , $V_j = \alpha \pm \beta U_j$ (见习题 17)。

因此, 希望式 (23) 中的 C 接近于 0。实际上, 由于 $U_0 U_1$ 不完全独立于 $U_1 U_2$, 故不能期望序列相关系数恰巧为 0 (见习题 (18))。 C 的一个“好”的值将在 $\mu_n - 2\sigma_n$ 和 $\mu_n + 2\sigma_n$ 之间, 其中

$$\mu_n = \frac{-1}{(n-1)}, \quad \sigma_n^2 = \frac{n^2}{(n-1)^2(n-2)}, \quad n > 2 \quad (25)$$

我们预期 C 大约在 95% 的时间内处于两个极限之间。

(25) 中的 σ_n^2 的公式是一个上限, 它对来自于任何分布的独立随机变量之间的序列相关都有效。当诸 U 是一致分布的时, 真正的方差由减去 $\frac{24}{5}n^{-2} + O(n^{-7/3} \cdot \log n)$ 得到 (见习题 20)。

代替简单地计算观察值 $(U_0, U_1, \dots, U_{n-1})$ 和它们的直接后继者 $(U_1, \dots, U_{n-1}, U_0)$ 之间的相关系数, 我们也可以在 $(U_0, U_1, \dots, U_{n-1})$ 和任何循环移位序列 $(U_q, \dots, U_{n-1}, U_0, \dots, U_{q-1})$ 之间来计算它。对于 $0 < q < n$, 循环相关应该是很小的。对于所有的 q , 对式 (24) 的一个直截了当的计算将要求大约 n^2 次乘法。但通过使用“快速傅里叶变换”实际上有可能仅在 $O(n \log n)$ 步内计算所有的相关 (见 4.6.4 小节; 也可参见 L.P. Schmid, CACM 8 (1965), 115)。

L. 子序列检验 外部程序通常成批地调用随机数。例如,如果一个程序对三个随机变量 X, Y 和 Z 进行处理,则它可能一次就要生成三个随机数。在这样的应用中,重要的是由原来序列的每个第三项^①组成的子序列是随机的。如果这个程序一次要求 q 个数,则诸子序列

$$U_0, U_q, U_{2q}, \dots; U_1, U_{q+1}, U_{2q+1}, \dots; \dots; U_{q-1}, U_{2q-1}, U_{3q-1}, \dots$$

中的每一个都可施以对原来的序列 U_0, U_1, U_2, \dots 的上述那种检验。

使用线性同余序列的经验已经表明,这些导出的子序列很少(如果有的话)有可能不如原来的序列随机,除非 q 和周期长度有共同的大因子。例如,在 m 等子字长的一台二进计算机上,对于 $q=8$ 的一个子序列的检验,势必给出对所有 $q < 16$ 的子序列检验中最差的随机性;而在一台十进计算机上, $q=10$ 产生似乎最不能令人满意的子序列。(在某种程度上可用效能来解释,因为 q 的这些值一般会降低效能。习题 3.2.1.2-20 提供了更详细的说明。)

M. 历史评述和进一步讨论 在统计学家“证明”或“否定”关于各种观察数据的假设的工作中,自然地出现了统计检验。讨论人工生成的数的随机性检验早期最著名的论文是 M.G.Kendall 和 B.Babington-Smith 的两篇,发表在 *Journal of the Royal Statistical Society* **101** (1938), 147~166, 和该杂志的增刊, **6** (1939), 51~61 上。这些论文讨论了 0 与 9 之间随机数字的检验,而不是随机的实数的检验;为此目的,两位作者引进了频率检验、序列检验、间隔检验以及扑克检验,然而他们滥用了序列检验。Kendall 和 Babington-Smith 也用过集券检验的一种变形;前文所述的方法是由 R.E.Greenwood 引进的 [*Math. Comp.* **9** (1955), 1~5]。

运行检验有一段颇为有趣的历史。开头,这些检验对上行运行和下行运行同时进行:一个上行运行跟着一个下行运行,然后是另一个上行运行等等。注意,运行检验和排列检验都不依赖于诸 U 的一致分布性,它们仅仅依赖于这样一个事实,即当 $i \neq j$ 时, $U_i = U_j$ 的概率为 0;因此,这些检验可以应用于许多类型的随机序列。原始形式的运行检验是由 J.Bienaymé 创立的 [*Comptes Rendus Acad. Sci. Paris* **81** (1875), 417~423]。大约 60 年后, W.O.Kermack 和 A.G.McKendrick 发表了关于这一课题的两篇综述论文 [*Proc. Royal Society Edinburgh* **57** (1937), 228~240, 332~376];作为一个例子,他们指出,在 1785 年到 1930 年间,爱丁堡的雨量相对于运行检验说来本质上是完全随机的(尽管他们仅仅考察了运行长度的平均值和标准差)。若干其他人开始使用这个检验,但直到 1944 年,人们才发现把使用 χ^2 方法与这一检验相联系是不正确的。H.Levenc 和 J.Wolfowitz 在 *Annals Math. Stat.* **15** (1944), 58~69 的论文中引进了正确的运行检验(交替地进行上下行运行检验),并且讨论了早先滥用该检验的错误。如同在上面的正文中所指出的,对于上行运行和下行运行分开检验,更适合于计算机应用,所以我们没有给出在上下交替情况下更复杂的公式。请看 D.E.Barton 和 C.L.Mallows 的评述论文, *Annals Math. Stat.* **36**

① 即每隔三项。——译者注

(1965), 236~260。

在我们已描述过的所有检验中,几乎所有的随机数生成程序都通过了频率检验和序列相关检验,从这个意义上说,频率检验和序列相关检验似乎是最弱的。它们较弱的理论根据,将在3.5节进行简要的讨论(参考习题3.5-26)。另一方面,运行检验是一个略强的检验:习题3.3.3-23和24的结果提示,如果乘数不是太大,则线性同余生成程序趋于有比通常略长的运行,所以习题14的运行检验肯定是值得推荐的。

冲突检验也是很值得推荐的,因为它已特别地设计成用以发现广为流传的许多差的生成程序的缺陷。这一检验以 H. Delgas Christiansen 的思想 [Inst. Math. Stat. and Oper. Res., Tech. Univ. Denmark (1975 年 10 月), 未发表] 为根据。它是在计算机发明之后开发的头一个检验,特别用于计算机计算,不适合于手算。

读者大概会惊异,“为什么会有这样多的检验?”据说,花在检验随机数上的计算机执行时间超过了应用这些随机数的时间!这是不真实的,尽管确有可能对检验过分热衷了。

已经有大量文献论述了进行若干检验的必要。例如,人们已经发现,由平方取中法的一个变形所产生的某些数已经通过频率检验、间隔检验和扑克检验,但却通不过序列检验。人们知道,具有小乘数的线性同余序列通过了许多检验,但是对于运行检验却失败了,因为长度为1的运行太少了。 t 的极大值检验也已经用来探测出某些坏的生成程序,不然这些生成程序会被误认为可正当实施。当极大间隔的长度超过最大延搁时,带借位减法生成程序通不过间隔检验,参见 Vettulainen, Kankaala, Saarinen 和 Ala-Nissila, *Computer Physics Communications* **86** (1999), 209~226, 其中还报告了其它形形色色的检验。延搁斐波那契生成程序,它在理论上保证有相同分布的最低有效位,但仍然不能通过1位相同分布检验的一些简单的变形(见习题31和35,还有3.6-14)。

也许,对随机数生成程序进行广泛检验的主要原因是,滥用 X 先生的随机数生成程序的人们,几乎从不承认他们的程序是不行的:他们将责难这个随机数生成程序,直到 X 先生能向他们证明,他的数确是充分随机的为止。另一方面,如果随机数的来源仅为 X 先生个人使用,则他可能会决定不去费心来检验它们,因为本章所推荐的一些技术有令人满意的高概率。

由于计算机变得更快了,因此比起以前,随机数的使用也更多了,于是曾经令人满意的随机数生成程序对于在物理学、组合学、随机几何等等的复杂应用说来不再是足够好的了。因此 George Marsaglia 介绍了一些严格的检验,它们大大超出像间隔和扑克检验这样经典的方法,以便满足新的挑战。例如,他发现序列 $X_{n+1} = (62605X_n + 113218009) \bmod 2^{29}$ 在下列的实验中有值得注意的偏倚:生成 2^{21} 个随机数 X_n 并抽取它们的10个前导位 $Y_n = \lfloor X_n/2^{19} \rfloor$ 。统计10位二进位数的 2^{20} 个可能的数偶 (y, y') 中有多少在 $(Y_1, Y_2), (Y_2, Y_3), \dots, (Y_{2^{21}-1}, Y_{2^{21}})$ 当中不出现。应当有大约 141909.33 对不出现的数偶,并有标准差 ≈ 290.46 (请见习题34)。但是以

$X_1 = 1234567$ 开始的六个连续的测试值产生出全在 1.5 和 3.5 之间太低的标准差的计数。分布有点太“平”了而不是随机的——大概是因为 2^{21} 个数是整个周期一个有意义的分数 $1/256$ 。一个带有乘数 69069 和模数 2^{30} 的类似的生成程序被证明是更好些的。Marsaglia 和 Zaman 把它叫做“猴子检验”，因为它统计一只猴子随机地在有 1024 个键的键盘上敲键，敲不到的两字符组合的个数，关于对若干个猴子检验的分析，请见 *Computers and Math.* **26**, 9 (1993 年 9 月), 1~10。

习 题

1. [10] 为什么在 B 部分中描述的序列检验应该应用于 $(Y_0, Y_1), (Y_2, Y_3), \dots, (Y_{2n-2}, Y_{2n-1})$ ，而不是 $(Y_0, Y_1), (Y_1, Y_2), \dots, (Y_{n-1}, Y_n)$ ？

2. [10] 指出把序列检验从数偶推广到三元组、四元组的一个适当方法。

► 3. [M20] 假定序列是随机的，平均地说，在间隔检验（算法 G）中找到 n 个间隔之前需要考察多少个 U ？什么是这个量的标准差？

4. [12] 证明 (4) 中的概率对于间隔检验是正确的。

5. [M23] Kendall 和 Babington-Smith 所使用的“经典的”间隔检验考虑数 U_0, U_1, \dots, U_{N-1} 是一个循环序列，而且 U_{N+j} 与 U_j 相等，这里 N 是接受检验的诸 U 的固定个数。如果数 U_0, \dots, U_{N-1} 中有 n 个落入范围 $\alpha \leq U_j < \beta$ ，则在循环序列中有 n 个间隔。设 Z_r 是长度为 r 的间隔个数，其中 $0 \leq r < t$ ，并设 Z_t 是长度 $\geq t$ 的间隔的个数；证明在 N 趋于无穷的极限下，量 $V = \sum_{0 \leq r < t} (Z_r - np_r)^2 / np_r$ 将有自由度 t 的 χ^2 分布，其中 p_r 是式 (4) 给出的。

6. [40] (H. Geiringer) 在 $e = 2.71828\dots$ 的表示中，头 2000 位十进数字的频率计数给出 1.06 的 χ^2 值。这说明，对于数字 0, 1, \dots , 9，实际频率太接近于预期的值了，因之不能认为是随机分布的（事实上， $\chi^2 \geq 1.15$ 的频率为 99.9%）。同样的检验应用于 e 的头 10000 个数字，给出比较合理的值 $\chi^2 = 8.61$ ；但是头 2000 个数字如此均匀地分布的事实仍然是令人惊异的。在 e 对于其它进制的表示中，是否有同样的现象？[见 *AMM* **72** (1965), 483~500。]

7. [08] 以 $d = 3$ 和 $n = 7$ 应用集券检验程序（算法 C）于序列 1101221022120202001212201010201121，七个子序列长度是多少？

► 8. [M22] 假定序列是随机的，平均说来，在找到 n 个完备的集合之前，在集券检验（算法 C）中，有多少个 U 需要检验？标准差等于什么？[提示：见等式 1.2.9-(28)。]

9. [M21] 推广集券检验，使得一旦找到 w 个不同的值之后，即停止检索，其中 w 是一个小于或等于 d 的固定正整数。问应当以什么概率代替 (6)？

10. [M23] 对于习题 9 中所描述的更一般的集券检验，解习题 8。

11. [00] (9) 中示出一个特殊排列中的“上行运行”，问该排列中的“下行运行”是什么？

12. [20] 设 U_0, U_1, \dots, U_{n-1} 是 n 个不同的数，写出一个算法，确定序列中所有递增运行的长度。当这个算法终止时，对于 $1 \leq r \leq 5$ ， $\text{COUNT}[r]$ 应是长度为 r 的运行的个数，而 $\text{COUNT}[6]$ 应该是长度为 6 或更大的运行的个数。

13. [M23] 证明 (16) 是有形式 (15) 的 $p + q + 1$ 个不同元素的排列的个数。

► 14. [M15] 如果“抛弃”紧接于一个运行后的那个元素，使得当 X_j 大于 X_{j+1} 时，以 X_{j+2} 开始下一个运行，则运行长度就都是独立的了，而且可以使用一个简单的 χ^2 检验（代替正文中所推导的令人生畏的复杂方法）。对于这一简单的运行检验，什么是相应的运行长度的概率？

15. [M10] 在“ t 的极大值”检验中,为什么要假定 $V_0^t, V_1^t, \dots, V_{n-1}^t$ 在 0 与 1 之间一致分布?

► 16. [15] J. H. Quick 先生(一位大学生)要对 t 的各种值实施“ t 的极大值”检验。

a) 令 $Z_{jt} = \max(U_j, U_{j+1}, \dots, U_{j+t-1})$, 他发现一个从序列 $Z_{0(t-1)}, Z_{1(t-1)}, \dots$ 变到序列 Z_{0t}, Z_{1t}, \dots 的只使用少量时间和空间的巧妙方法。他的聪明的方法是怎样的?

b) 他决定修改“ t 的极大值”的方法,以使第 j 个观察量为 $\max(U_j, \dots, U_{j+t-1})$; 换言之,他取 $V_j = Z_{jt}$, 而不是如正文中所说的 $V_j = Z_{(tj)t}$ 。他推断,所有的 Z 都应该有相同的分布,所以,如果不是每次取第 t 个值,而是取每个 $Z_{jt}, 0 \leq j < n$, 则这个检验应该更强,但当他对于 V_j^t 的值进行一次 χ^2 等分布检验时,却得到极其高的统计值 V , 当 t 增大时,甚至得到更高的值 V 。这是为什么?

17. [M25] 给定任意数 $U_0, \dots, U_{n-1}, V_0, \dots, V_{n-1}$, 令

$$\bar{u} = \frac{1}{n} \sum_{0 \leq k < n} U_k, \quad \bar{v} = \frac{1}{n} \sum_{0 \leq k < n} V_k$$

a) 设 $U'_k = U_k - \bar{u}, V'_k = V_k - \bar{v}$ 。证明式(24)中给出的相关系数 C 等于

$$\frac{\sum_{0 \leq k < n} U'_k V'_k}{\sqrt{\sum_{0 \leq k < n} U'^2_k} \sqrt{\sum_{0 \leq k < n} V'^2_k}}$$

b) 设 $C = N/D$, 其中 N 和 D 表示 a) 部分中表达式的分子和分母。证明 $N^2 \leq D^2$, 因此 $-1 \leq C \leq 1$; 并且求出关于差 $D^2 - N^2$ 的一个公式。[提示: 见习题 1.2.3-30。]

c) 如果 $C = \pm 1$, 证明对于某些不全为 0 的常数 α, β 和 τ , 有 $\alpha U_k + \beta V_k = \tau, 0 \leq k < n$ 。

18. [M20] (a) 证明, 如果 $n=2$, 则序列相关系数(23)总是等于 -1 (除非分母为 0)。(b) 类似地, 证明当 $n=3$ 时, 序列相关系数总等于 $-\frac{1}{2}$ 。(c) 证明, 当且仅当 $U_0 = U_1 = \dots = U_{n-1}$ 时, 式(23)中的分母为 0。

19. [M30] (J. P. Butler) 设 U_0, \dots, U_{n-1} 是有相同分布的独立随机变量, 试证明序列相关系数(23)的期望值, 对分母非零的所有情况取平均, 是 $-1/(n-1)$ 。

20. [HM41] 继续上一道题, 试证明(23)的方差等于 $n^2/(n-1)^2(n-2) - n^3 E((U_0 - U_1)^4/D^2)/2(n-2)$, 其中 D 是(23)的分母而 E 表示对于 $D \neq 0$ 的所有情况的期望值。当每个 U_j 一致地分布时 $E((U_0 - U_1)^4/D^2)$ 的近似值是多少?

21. [19] 如果对算法 P 提供排列 $(1, 2, 9, 8, 5, 3, 6, 7, 0, 4)$, 问用它算出的 f 值等于多少?

22. [18] 如何排列整数 $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, 才能使算法 P 产生值 $f=1024$?

23. [M22] 设 $\langle Y_n \rangle$ 和 $\langle Y'_n \rangle$ 是分别有周期长度 λ 和 λ' 的整数序列, 且 $0 \leq Y_n, Y'_n < d$; 还设 $Z_n = (Y_n + Y'_{n+r}) \bmod d$, 其中 r 是在 0 和 $\lambda' - 1$ 之间随机地选择的。试证明, 在下列意义下 $\langle Z_n \rangle$ 至少也像 $\langle Y_n \rangle$ 一样好地通过 t 维序列检验: 设 $P(x_1, \dots, x_t)$ 和 $Q(x_1, \dots, x_t)$ 是 t 元组 (x_1, \dots, x_t) 在 $\langle Y_n \rangle$ 和 $\langle Z_n \rangle$ 中出现的概率:

$$P(x_1, \dots, x_t) = \frac{1}{\lambda} \sum_{n=0}^{\lambda-1} [(\langle Y_n, \dots, Y_{n+t-1} \rangle) = (x_1, \dots, x_t)]$$

$$Q(x_1, \dots, x_t) = \frac{1}{\lambda \lambda'} \sum_{n=0}^{\lambda-1} \sum_{r=0}^{\lambda'-1} [(\langle Z_n, \dots, Z_{n+t-1} \rangle) = (x_1, \dots, x_t)]$$

则
$$\sum_{(x_1, \dots, x_t)} (Q(x_1, \dots, x_t) - d^{-t})^2 \leq \sum_{(x_1, \dots, x_t)} (P(x_1, \dots, x_t) - d^{-t})^2$$

24. [HM35] (G. Marsaglia) 试证明对于 n 个重叠 t 元组 $(Y_1, Y_2, \dots, Y_t), (Y_2, Y_3, \dots, Y_{t+1}), \dots, (Y_n, Y_1, \dots, Y_{t-1})$ 的序列检验可如下进行: 对于每个满足 $0 \leq a_i < d$ 的串 $\alpha = a_1 \dots a_n$,

令 $N(\alpha)$ 是 α 作为 $Y_1 Y_2 \cdots Y_n Y_1 \cdots Y_{m-1}$ 的一个子串出现的次数, 并设 $P(\alpha) = P(a_1) \cdots P(a_m)$ 是 α 出现在任何给定的位置的概率; 个别的数字可以以不同的概率 $P(0), P(1), \cdots, P(d-1)$ 出现, 计算统计

$$V = \frac{1}{n} \sum_{|\alpha|=r} \frac{N(\alpha)}{P(\alpha)} - \frac{1}{n} \sum_{|\alpha|=r-1} \frac{N(\alpha)}{P(\alpha)}$$

那么当 n 很大时 V 应有自由度为 $d^r - d^{r-1}$ 的 χ^2 分布。[提示, 使用习题 3.3.1-25。]

25. [M46] 当 C_1 和 C_2 是 (22) 之后定义的矩阵时, 为什么 $C_1^{-1} C_2 C_1^{-1} \approx -6C_1^{-1}$?

26. [HM30] 设 U_1, U_2, \cdots, U_n 是在 $[0, 1)$ 中的独立一致偏离, 并设 $U_{(1)} \leq U_{(2)} \leq \cdots \leq U_{(n)}$ 是排序之后他们的值, 并定义间隔 $S_1 = U_{(2)} - U_{(1)}, \cdots, S_{n-1} = U_{(n)} - U_{(n-1)}, S_n = U_{(1)} + 1 - U_{(n)}$ 以及和生日间隔检验一样间隔 $S_{(1)} \leq \cdots \leq S_{(n)}$ 。在下列的计算中使用记号 x^+ 作为表达式 $x^+[x \geq 0]$ 的缩写。

a) 给定任何实数 s_1, s_2, \cdots, s_n , 证明联立不等式 $S_1 \geq s_1, S_2 \geq s_2, \cdots, S_n \geq s_n$ 以 $(1 - s_1 - s_2 - \cdots - s_n)^+_{n-1}$ 的概率出现。

b) 结果最小的间隔 $S_{(1)}$ 以 $1 - (1 - ns)^+_{n-1}$ 的概率小于等于 s 。

c) 对于 $1 \leq k \leq n$, 什么是分布函数 $F_k(s) = P_r(S_{(k)} \leq s)$?

d) 计算每个 $S_{(k)}$ 的均值和方差。

► 27. [HM26] (迭代的间隔) 在上题记号下, 证明数 $S'_1 = nS_{(1)}, S'_2 = (n-1)(S_{(2)} - S_{(1)}), \cdots, S'_n = 1(S_{(n)} - S_{(n-1)})$ 和 n 个一致偏离的原来的间隔 S_1, \cdots, S_n 有相同的联合概率分布。因此我们可以对它们进行排序, $S'_{(1)} \leq \cdots \leq S'_{(n)}$, 并重复这个转换来得出另一个随机间隔的集合 S''_1, \cdots, S''_n 等等。使用

$$K_{n-1}^+ = \sqrt{n-1} \max_{1 \leq j \leq n} \left(\frac{j}{n-1} - S_1^{(k)} - \cdots - S_j^{(k)} \right)$$

$$K_{n-1}^- = \sqrt{n-1} \max_{1 \leq j \leq n} \left(S_1^{(k)} + \cdots + S_j^{(k)} - \frac{j-1}{n-1} \right)$$

每个逐次的间隔 $S_1^{(k)}, \cdots, S_n^{(k)}$ 的集合都可经受 Kolmogorov-Smirnov 检验。在 $n=2$ 和 $n=3$ 的情况下详细考察从 (S_1, \cdots, S_n) 到 (S'_1, \cdots, S'_n) 的转换; 说明当把它应用于具有有限精度的计算机生成的数时, 为什么连续重复此过程最终要崩溃。(比较随机数生成程序的一个方法是看它们对于这样的折磨检验, 能继续生存多久。)

28. [M26] 设 $b_{nr}(m)$ 是满足 $0 \leq y_j < m$ 的 n 元组 (y_1, \cdots, y_n) 的个数, 它们恰有 r 个相等的间隔和 s 个零间隔。因此, 在生日间隔检验中 $R=r$ 的概率是 $\sum_{s=0}^{r-1} b_{nr}(m)/m^n$ 。还设 $p_n(m)$ 是把 m 分划成至多 n 个部分的分划数 (见习题 5.11-15)。(a) 用分划表达 $b_{n0}(m)$ 。[提示: 考虑有小的 m 和 n 的情况。] (b) 证明, 当 $s > 0$ 时 $b_{nr}(m)$ 和 $b_{(n-s)(r+1-s)0}(m)$ 之间有一个简单的关系。(c) 试推导没有间隔相等的概率的公式。

29. [M35] 继续习题 28, 试求当 $r=0, 1, 2$ 时, 对于生成函数 $b_{nr}(z) = \sum_{m \geq 0} b_{nr0}(m) z^m / m$ 的简单表达式。

30. [HM41] 继续上道题, 证明如果 $m = n^3/\alpha$, 当 $n \rightarrow \infty$ 时对于固定的 α , 我们有

$$p_n(m) = \frac{m^{n-1} e^{\alpha/4}}{n!(n-1)!} \left(1 - \frac{13\alpha^2}{288n} + \frac{169\alpha^4 + 2016\alpha^3 - 1728\alpha^2 - 41472\alpha}{165888n^2} + O(n^{-3}) \right)$$

求对于 $q_n(m)$ 的一个类似的公式, 这里 $q_n(m)$ 为把 m 分成 n 个不同的正的的部分的分划数。推导生日间隔检验发现 R 等于 0, 1 和 2 到 $O(1/n)$ 之内的近似概率。

►31.[M21] 递推式 $Y_n = (Y_{n-24} + Y_{n-55}) \bmod 2$ 描述延搁斐波那契生成程序 3.2.2-(7) 的最低有效位以及 3.2.2-(7) 的次低有效位, 已知有周期长度 $2^{55} - 1$; 因此位 $(Y_n, Y_{n+1}, \dots, Y_{n+54})$ 的每一个可能的非零模式同样经常地出现。尽管如此, 试证明如果在周期中的一个随机点开始, 我们生成 79 个连续的随机二进位 Y_n, \dots, Y_{n+78} , 则 1 的个数比 0 多的概率大过 51%。如果我们使用这样的二进位来定义一个“随机的走动”, 即当二进位是 1 时则向右移动, 而如果二进位是 0 时则向左移动, 则我们将以大大超过一半的时间在我们出发点的右边结束。[提示: 求生成函数 $\sum_{k=-79}^{79} \Pr(Y_n + \dots + Y_{n+78} = k) z^k$ 。]

32.[M20] 真或假: 如果 X 和 Y 是具有均值 0 的独立相等分布的随机变量, 而且如果它们更可能为正, 则 $X + Y$ 也是更可能为正的。

33.[HM32] 当 $k > 2l$ 且递推式 $Y_n = (Y_{n-l} + Y_{n-k}) \bmod 2$ 的周期长度是 $2^k - 1$, 并假定 k 很大时, 试求由这个递推式生成的 $k + l$ 个连续的二进位中 1 的个数多于 0 的个数的概率的近似值。

34.[HM29] 试说明如何估计在一个 m 个字母的字符集上长度为 n 的一个随机串中两个字母的组合不连续地出现的个数的均值和误差。假定 m 很大并且 $n \approx 2m^2$ 。

►35.[HM32] (J. H. Lindholm, 1968) 假设对于 a_1, \dots, a_k 的某个选择使用递推式

$$Y_n = (a_1 Y_{n-1} + a_2 Y_{n-2} + \dots + a_k Y_{n-k}) \bmod 2$$

我们生成随机二进位 $\langle Y_n \rangle$, 使得周期长度是 $2^k - 1$; 以 $Y_0 = 1$ 和 $Y_1 = \dots = Y_{k-1} = 0$ 开始。令 $Z_n = (-1)^{Y_{n+1}} = 2Y_{n+1} - 1$ 是一个随机符号, 并考虑统计 $S_m = Z_n + Z_{n+1} + \dots + Z_{n+m-1}$, 其中 n 是这个周期中的一个随机点。

- 证明 $ES_m = m/N$, 其中 $N = 2^k - 1$ 。
- 什么是 ES_m^2 ? 假设 $m \leq N$ 。提示: 请见习题 3.2.2-16。
- 如果诸 Z 真正地随机, 则 ES_m 和 ES_m^2 将如何?
- 假定 $m \leq N$, 试证明 $ES_m^3 = m^3/N - 6B(N+1)/N$, 其中

$$B = \sum_{0 \leq i < j < m} [(Y_{i+1} Y_{i+2} \dots Y_{i+k-1})_2 = (Y_{j+1} Y_{j+2} \dots Y_{j+k-1})_2] (m-j)$$

- 计算在习题 31 考虑的特殊情况中的 B ; $m = 79$ 和 $Y_n = (Y_{n-24} + Y_{n-55}) \bmod 2$ 。

*3.3.3 理论检验

尽管利用上一节的方法, 总可能检验任何随机数生成程序, 但是如果有一些先验的检验, 即能预先告诉我们这些检验的结果将会是如何的理论, 那就更好得多。这样的理论要比经验的、反复实验的结果更能增进我们对生成方法的理解。在这一小节, 我们将更详细地研究线性同余序列; 如果在我们实际生成这些数之前就知道某些检验将有什么结果, 则我们就有更好的机会来适当地选择 a , m 和 c 。

尽管已经取得了某些进展, 发展这一类型的理论却十分困难。至今所得到的结果一般都是关于对整个周期所做的统计检验。当把统计检验应用于整个一个周期上时, 并非所有的统计检验都有意义——例如, 用等分布检验所得出的结果就太理想化了——然而, 序列检验、间隔检验、排列检验、极大值检验等方法却能有效地进

行分析。这些研究将用来检验一个序列的全局非随机性,即在非常大的抽样中的不适当行为。

我们将要讨论的理论是十分富于启发性的,但它并不排除通过 3.3.2 节的方法检验局部非随机性的必要。确实,证明关于短的子序列的任何有用的性质看来都是极其困难的。关于线性同余序列在比一个全周期短的区间上的特性仅仅知道一些理论结果;这些将在 3.3.4 节末尾处讨论(也见习题 18)。

让我们从对于最不复杂的排列检验的情况,证明一个简单的先验法则开始。第一个定理的要点是:假定序列有高的效能,则大约有一半的时间有 $X_{n+1} < X_n$ 。

定理 P 设 a, c 和 m 生成一个极大周期的线性同余序列;设 $b = a - 1$, 并设 d 是 m 和 b 的最大公因子,则 $X_{n+1} < X_n$ 的概率等于 $1/2 + r$, 其中

$$r = (2(c \bmod d) - d)/2m \quad (1)$$

因此 $|r| < d/2m$ 。

证明 这个定理的证明涉及某些技术,这些技术本身就是有趣的。首先,定义

$$s(x) = (ax + c) \bmod m \quad (2)$$

于是, $X_{n+1} = s(X_n)$, 定理归结为计算使 $0 \leq x < m$ 和 $s(x) < x$ 的整数 x 的个数,因为每个这样的整数皆出现于周期中的某处。我们要证明这个数是

$$\frac{1}{2}(m + 2(c \bmod d) - d) \quad (3)$$

当 $x > s(x)$ 时, 函数 $\lceil (x - s(x))/m \rceil$ 等于 1, 否则它为 0; 因此我们希望计算的个数可以简单地写作

$$\sum_{0 \leq x < m} \left\lceil \frac{x - s(x)}{m} \right\rceil = \sum_{0 \leq x < m} \left\lceil \frac{x}{m} - \left(\frac{ax + c}{m} - \left\lfloor \frac{ax + c}{m} \right\rfloor \right) \right\rceil = \sum_{0 \leq x < m} \left(\left\lfloor \frac{ax + c}{m} \right\rfloor - \left\lfloor \frac{bx + c}{m} \right\rfloor \right) \quad (4)$$

(回想 $\lceil -y \rceil = -\lfloor y \rfloor$ 及 $b = a - 1$ 。)通过习题 1.2.4-37 的方法可以计算这样的和式,在那里我们已经证明当 h 和 k 是整数且 $k > 0$ 时,

$$\sum_{0 \leq j < k} \left\lfloor \frac{hj + c}{k} \right\rfloor = \frac{(h-1)(k-1)}{2} + \frac{g-1}{2} + g \lfloor c/g \rfloor, \quad g = \gcd(h, k) \quad (5)$$

因为 a 与 m 互素,故由此得出

$$\begin{aligned} \sum_{0 \leq x < m} \left\lfloor \frac{ax + c}{m} \right\rfloor &= \frac{(a-1)(m-1)}{2} + c \\ \sum_{0 \leq x < m} \left\lfloor \frac{bx + c}{m} \right\rfloor &= \frac{(b-1)(m-1)}{2} + \frac{d-1}{2} + c - (c \bmod d) \end{aligned}$$

因此,立即得出(3)。 ■

定理 P 的证明指出,假设我们有能力满意地处理含有 $\lfloor \cdot \rfloor$ 和 $\lceil \cdot \rceil$ 函数的和,则确实可进行预先检验。在许多情况下,处理底限和顶限函数最强有力的技术是以两个稍微更对称的公式来代替它们

$$\delta(x) = \lfloor x \rfloor + 1 - \lceil x \rceil = [x \text{ 是整数}] \quad (6)$$

$$\begin{aligned} ((x)) &= x - \lfloor x \rfloor - \frac{1}{2} + \frac{1}{2}\delta(x) = x - \lceil x \rceil + \frac{1}{2} - \frac{1}{2}\delta(x) = \\ &= x - \frac{1}{2}(\lfloor x \rfloor + \lceil x \rceil) \end{aligned} \quad (7)$$

后一函数是在研究傅里叶级数时熟知的“锯齿”函数,它的图形如图7所示。选择

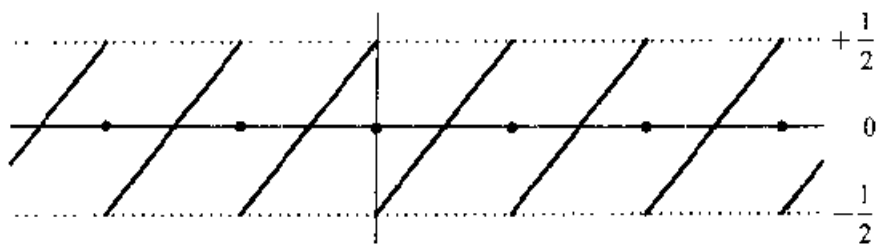


图7 锯齿函数 $((x))$

$((x))$ 而不选择 $\lfloor x \rfloor$ 或 $\lceil x \rceil$ 的原因是 $((x))$ 具有若干非常有用的性质:

$$((-x)) = -((x)) \quad (8)$$

$$((x+n)) = ((x)), \text{ 整数 } n \quad (9)$$

$$((nx)) = ((x)) + \left(\left(x + \frac{1}{n} \right) \right) + \cdots + \left(\left(x + \frac{n-1}{n} \right) \right), \quad \text{整数 } n \geq 1 \quad (10)$$

(见习题 1.2.4-38 和 1.2.4-39(a,b,g).)

为了得到用这些函数进行工作的某些实践经验,我们再次证明定理 P,这次不依赖于习题 1.2.4-37。借助于等式(7),(8),(9),我们可以证明

$$\begin{aligned} \left\lceil \frac{x-s(x)}{m} \right\rceil &= \frac{x-s(x)}{m} - \left(\left(\frac{x-s(x)}{m} \right) \right) + \frac{1}{2} - \frac{1}{2}\delta\left(\frac{x-s(x)}{m}\right) = \\ &= \frac{x-s(x)}{m} - \left(\left(\frac{x-(ax+c)}{m} \right) \right) + \frac{1}{2} = \\ &= \frac{x-s(x)}{m} + \left(\left(\frac{bx+c}{m} \right) \right) + \frac{1}{2} \end{aligned} \quad (11)$$

因为 $(x-s(x))/m$ 绝不是一个整数。现在

$$\sum_{0 \leq x < m} \frac{x-s(x)}{m} = 0$$

因为 x 和 $s(x)$ 都取 $\{0, 1, \dots, m-1\}$ 的每个值恰一次,因此由(11)得出

$$\sum_{0 \leq x < m} \left\lceil \frac{x-s(x)}{m} \right\rceil = \sum_{0 \leq x < m} \left(\left(\frac{bx+c}{m} \right) \right) + \frac{m}{2} \quad (12)$$

令 $b = b_0 d$, $m = m_0 d$, 其中 b_0 与 m_0 互素。我们知道,当 x 从 0 变到 $m_0 - 1$ 时, $(b_0 x) \bmod m_0$ 以某一顺序取数值 $\{0, 1, \dots, m_0 - 1\}$ 。由(9)和(10)以及

$$\left(\left(\frac{b(x+m_0)+c}{m} \right) \right) = \left(\left(\frac{bx+c}{m} \right) \right)$$

这一事实,我们有

$$\sum_{0 \leq x < m} \left(\left(\frac{bx+c}{m} \right) \right) = d \sum_{0 \leq x < m_0} \left(\left(\frac{bx+c}{m} \right) \right) = d \sum_{0 \leq x < m_0} \left(\left(\frac{c}{m} + \frac{b_0 x}{m} \right) \right) = d \left(\left(\frac{c}{d} \right) \right) \quad (13)$$

由(12)和(13)立即得到定理 P。

定理 P 的一个推论是,实际上,除了对于那些有很大的 d 的情况, a 和 c 的任何选择都将给出 $X_{n+1} < X_n$ 的相当大的概率,至少对于整个周期是如此。很大的 d 值对应于低效能,而我们已经知道低效能的生成程序是不合意的。

下一定理给我们提供了选择 a 和 c 的更严格的条件,我们将考虑应用到整个周期上的序列相关检验。在 3.3.2 小节,式(23)定义的量是

$$C = \left(m \sum_{0 \leq x < m} xs(x) - \left(\sum_{0 \leq x < m} x \right)^2 \right) / \left(m \sum_{0 \leq x < m} x^2 - \left(\sum_{0 \leq x < m} x \right)^2 \right) \quad (14)$$

设 x' 是使得 $s(x') = 0$ 的元素,我们有

$$s(x) = m \left(\left(\frac{ax+c}{m} \right) \right) + \frac{m}{2} [x \neq x'] \quad (15)$$

借助于和数

$$\sigma(h, k, c) = 12 \sum_{0 \leq j < k} \left(\left(\frac{j}{k} \right) \right) \left(\left(\frac{hj+c}{k} \right) \right) \quad (16)$$

可以最容易地表达我们有待导出的这些公式。这个和数是在若干重要的数学问题中出现的一个重要函数,它称做广义戴德金和数,因为 Richard Dedekind 于 1876 年在评述 Riemann 的一篇未完成的手稿时引进了函数 $\sigma(h, k, 0)$ 。[见 B. Riemann 的 *Gesammelte Math. Werke*, 第 2 版, (1892), 466~478。]

利用已知的公式

$$\sum_{0 \leq x < m} x = \frac{m(m-1)}{2} \quad \text{及} \quad \sum_{0 \leq x < m} x^2 = \frac{m(m-\frac{1}{2})(m-1)}{3}$$

很容易把式(14)变换成为

$$C = \frac{m\sigma(a, m, c) - 3 + 6(m - x' - c)}{m^2 - 1} \quad (17)$$

(见习题 5。)因为 m 通常很大,所以可以抛弃阶为 $1/m$ 的项,因而我们有近似值

$$C \approx \sigma(a, m, c)/m \quad (18)$$

而且误差的绝对值小于 $6/m$ 。

序列相关检验现在归结为确定戴德金和数 $\sigma(a, m, c)$ 的值。从其定义式(16)直接地计算 $\sigma(a, m, c)$, 几乎一点也不比直接计算相关系数容易,但幸而有简单的方法可十分迅速地计算戴德金和数。

引理 B(戴德金和数的“互反律”) 设 h, k, c 为整数,如果 $0 \leq c < k, 0 \leq h < k$, 而且如果 h 与 k 互素,则

$$\sigma(h, k, c) + \sigma(k, h, c) = \frac{h}{k} + \frac{k}{h} + \frac{1}{hk} + \frac{6c^2}{hk} \cdot 6 \left\lfloor \frac{c}{k} \right\rfloor - 3e(h, c) \quad (19)$$

其中

$$e(h, c) = [c = 0] + [c \bmod h \neq 0] \quad (20)$$

证明 我们把对

$$\begin{aligned} \sigma(h, k, c) + \sigma(k, h, c) &= \sigma(h, k, 0) + \sigma(k, h, 0) + \frac{6c^2}{hk} - \\ &\quad 6 \left\lfloor \frac{c}{h} \right\rfloor - 3e(h, c) + 3 \end{aligned} \quad (21)$$

的证明留给读者(见习题6)。现在仅须在 $c=0$ 的情况下证明引理足矣。

我们的证明以单位的复数根为基础,它主要归功于 L. Carlitz。实际上,有一个仅使用和数的初等操作的更简单证明(见习题7)——但下边的方法展示了对于这类问题可利用的更多的数学工具,因此它更有用处。

设 $f(x)$ 和 $g(x)$ 是如下定义的多项式:

$$\begin{aligned} f(x) &= 1 + x + \cdots + x^{k-1} = (x^k - 1)/(x - 1) \\ g(x) &= x + 2x^2 + \cdots + (k-1)x^{k-1} = xf'(x) = \\ &\quad kx^k / ((x-1) - x(x^k - 1)/(x-1)^2) \end{aligned} \quad (22)$$

如果 ω 是单位的第 k 个复根 $e^{2\pi i/k}$, 则由式 1.2.9-(13) 有

$$\frac{1}{k} \sum_{0 \leq j < k} \omega^{-jr} g(\omega^j x) = rx', \quad \text{如果 } 0 \leq r < k \quad (23)$$

置 $x=1$; 若 $j \neq 0$ 则 $g(\omega^j x) = k/(\omega^j - 1)$, 否则它等于 $k(k-1)/2$ 。因此

$$r \bmod k = \sum_{0 < j < k} \frac{\omega^{-jr}}{\omega^j - 1} + \frac{1}{2}(k-1), \quad \text{若 } r \text{ 为整数}$$

(式(23)表明,当 $0 \leq r < k$ 时,右边等于 r , 而且即使 r 加上 k 的倍数它也不变。)因此

$$\left(\left(\frac{r}{k} \right) \right) = \frac{1}{k} \sum_{0 < j < k} \frac{\omega^{-jr}}{\omega^j - 1} - \frac{1}{2k} + \frac{1}{2} \delta \left(\frac{r}{k} \right) \quad (24)$$

这是个重要的公式,只要 r 为整数它就成立,它使我们得以把涉及 $((r/k))$ 的许多计算归纳为包含 1 的第 k 次根的和式,而且它将引进一整套全新的技术。特别是,当 $h \perp k$ 时我们得到如下的公式:

$$\sigma(h, k, 0) + \frac{3(k-1)}{k^2} = \frac{12}{k^2} \sum_{0 < r < k} \sum_{0 < s < k} \sum_{0 < j < k} \frac{\omega^{-ir}}{\omega^s - 1} \frac{\omega^{-jh}}{\omega^j - 1} \quad (25)$$

通过对 r 求和,可以简化这公式的右边;若 $s \bmod k \neq 0$, 我们就有 $\sum_{0 \leq r < k} \omega^{rs} = f(\omega^s) = 0$ 。式(25)现在归结为

$$\sigma(h, k, 0) + \frac{3(k-1)}{k} = \frac{12}{k} \sum_{0 < j < k} \frac{1}{(\omega^{-jh} - 1)(\omega^j - 1)} \quad (26)$$

用 $\zeta = e^{2\pi i/h}$ 代替 ω , 可对 $\sigma(k, h, 0)$ 得到一个类似的公式。

对于(26)中的和式我们能做什么并不是显然的,但是根据这个和数的每项都是 ω^j 的一个函数(其中 $0 < j < k$)这一事实,却有一个很好的方法可用;因此这个和实质上是对单位的不等于1的第 k 个根进行的。每当 x_1, x_2, \dots, x_n 为不同复数时,我们就有恒等式

$$\sum_{j=1}^n \frac{1}{(x_j - x_1) \cdots (x_j - x_{j-1})(x - x_j)(x_j - x_{j+1}) \cdots (x_j - x_n)} = \frac{1}{(x - x_1) \cdots (x - x_n)} \quad (27)$$

这是由通常的把右边展开成部分分式的方法得出的,而且,如果 $q(x) = (x - y_1)(x - y_2) \cdots (x - y_m)$,则有

$$q'(y_j) = (y_j - y_1) \cdots (y_j - y_{j-1})(y_j - y_{j+1}) \cdots (y_j - y_m) \quad (28)$$

这个恒等式通常可用来简化像(27)左边那样的一些表达式。当 h 与 k 互素时,数 $\omega, \omega^2, \dots, \omega^{k-1}, \zeta, \zeta^2, \dots, \zeta^{h-1}$ 都是不同的;因而我们可以在多项式 $(x - \omega) \cdots (x - \omega^{k-1})(x - \zeta) \cdots (x - \zeta^{h-1}) = (x^k - 1)(x^h - 1)/(x - 1)^2$ 的特殊情况下考虑(27),得到下列 x 的恒等式:

$$\frac{1}{h} \sum_{0 < j < h} \frac{\zeta^j (\zeta^j - 1)^2}{(\zeta^{jk} - 1)(x - \zeta^j)} + \frac{1}{k} \sum_{0 < j < k} \frac{\omega^j (\omega^j - 1)^2}{(\omega^{jh} - 1)(x - \omega^j)} = \frac{(x - 1)^2}{(x^h - 1)(x^k - 1)} \quad (29)$$

这个恒等式有许多有趣的推论,它导出许多像式(26)中那样的和式的互反公式。例如,如果我们对 x 微分(29)两次,并设 $x \rightarrow 1$,则得

$$\frac{2}{h} \sum_{0 < j < h} \frac{\zeta^j (\zeta^j - 1)^2}{(\zeta^{jk} - 1)(1 - \zeta^j)^3} + \frac{2}{k} \sum_{0 < j < k} \frac{\omega^j (\omega^j - 1)^2}{(\omega^{jh} - 1)(1 - \omega^j)^3} = \frac{1}{6} \left(\frac{h}{k} + \frac{k}{h} + \frac{1}{hk} \right) + \frac{1}{2} - \frac{1}{2h} - \frac{1}{2k}$$

在这些求和中用 $h-j$ 和 $k-j$ 代替 j ,并使用(26)得到

$$\frac{1}{6} \left(\sigma(k, h, 0) + \frac{3(h-1)}{h} \right) + \frac{1}{6} \left(\sigma(h, k, 0) + \frac{3(k-1)}{k} \right) = \frac{1}{6} \left(\frac{h}{k} + \frac{k}{h} + \frac{1}{hk} \right) + \frac{1}{2} - \frac{1}{2h} - \frac{1}{2k}$$

它等价于所求的结果。 \square

引理B给了我们一个显函数 $f(h, k, c)$,使得当 $0 < h \leq k, 0 \leq c < k$ 及 h 与 k 互素时,

$$\sigma(h, k, c) = f(h, k, c) - \sigma(k, h, c) \quad (30)$$

由定义(16),显然

$$\sigma(k, h, c) = \sigma(k \bmod h, h, c \bmod h) \quad (31)$$

因此,我们可以迭代地使用(30)来计算 $\sigma(h, k, c)$,并使用欧几里得算法简化参数。

当我们更仔细地考察这个迭代过程时,就出现进一步的简化。让我们置 $m_1 =$

$k, m_2 = h, c_1 = c$, 并做成下表:

$$\begin{aligned} m_1 &= a_1 m_2 + m_3 & c_1 &= b_1 m_2 + c_2 \\ m_2 &= a_2 m_3 + m_4 & c_2 &= b_2 m_3 + c_3 \\ m_3 &= a_3 m_4 + m_5 & c_3 &= b_3 m_4 + c_4 \\ m_4 &= a_4 m_5 & c_4 &= b_4 m_5 + c_5 \end{aligned} \quad (32)$$

这里

$$\begin{aligned} a_j &= \lfloor m_j / m_{j+1} \rfloor, & b_j &= \lfloor c_j / m_{j+1} \rfloor \\ m_{j+2} &= m_j \bmod m_{j+1}, & c_{j+1} &= c_j \bmod m_{j+1} \end{aligned} \quad (33)$$

而且由此得出

$$0 \leq m_{j+1} < m_j, \quad 0 \leq c_j < m_j \quad (34)$$

为方便起见, 我们已经假定在(32)中, 在四次迭代之后欧几里得算法结束; 这个假定将揭示在一般情况下成立的模式。因为开始时 h 和 k 互素, 所以在式(32)中我们必定有 $m_5 = 1$ 和 $c_5 = 0$ 。

我们进一步假定 $c_3 \neq 0$ 而 $c_4 = 0$, 以看看这对于递推式有什么影响。等式(30)和(31)得出

$$\begin{aligned} \sigma &= (h, k, c) = \sigma(m_2, m_1, c_1) = \\ &= f(m_2, m_1, c_1) - \sigma(m_3, m_2, c_2) = \cdots = \\ &= f(m_2, m_1, c_1) - f(m_3, m_2, c_2) + \\ &= f(m_4, m_3, c_3) - f(m_5, m_4, c_4) \end{aligned}$$

式(19)中的 $f(h, k, c)$ 的公式之头一部分 $h/k + k/h$ 对总和贡献了

$$\frac{m_2}{m_1} + \frac{m_1}{m_2} - \frac{m_3}{m_2} - \frac{m_2}{m_3} + \frac{m_4}{m_3} + \frac{m_3}{m_4} - \frac{m_5}{m_4} - \frac{m_4}{m_5}$$

它可简化成为

$$\frac{h}{k} + \frac{m_1}{m_2} - \frac{m_3}{m_2} - \frac{m_2}{m_3} + \frac{m_3}{m_4} - \frac{m_4}{m_3} + \frac{m_4}{m_5} - \frac{m_5}{m_4} = \frac{h}{k} + a_1 - a_2 + a_3 - a_4$$

式(19)的第二部分 $1/hk$ 也有一个简单的贡献; 根据等式 4.5.3-(9)和 4.5.3 小节的其它公式, 我们有

$$\frac{1}{m_1 m_2} - \frac{1}{m_2 m_3} + \frac{1}{m_3 m_4} - \frac{1}{m_4 m_5} = \frac{h'}{k} - 1 \quad (35)$$

其中 h' 是满足

$$h'h \equiv 1 \pmod{k}, \quad 0 < h' \leq k \quad (36)$$

的惟一整数。把所有贡献加起来, 并记住我们的假定 $c_4 = 0$ (使得 $e(m_4, c_3) = 0$, 参考式(20)), 借助于假定的表(32), 我们求得

$$\begin{aligned} \sigma(h, k, c) &= \frac{h + h'}{k} + (a_1 - a_2 + a_3 - a_4) - 6(b_1 - b_2 + b_3 - b_4) + \\ &+ 6 \left(\frac{c_1^2}{m_1 m_2} - \frac{c_2^2}{m_2 m_3} + \frac{c_3^2}{m_3 m_4} - \frac{c_4^2}{m_4 m_5} \right) + 2 \end{aligned}$$

一般来说有如下的类似结果:

定理 D 设 h, k, c 是整数, 并有 $0 < h \leq k, 0 \leq c < k$, 且 h 与 k 互素。按上面式 (33) 中定义的方式构造“欧几里得表”, 并假定此过程在 t 步之后停止并有 $m_{t+1} = 1$ 。设 s 是使得 $c_s = 0$ 的最小下标, 并设 h' 由 (36) 来定义, 则

$$\sigma(h, k, c) = \frac{h + h'}{k} + \sum_{1 \leq j \leq t} (-1)^{j+1} \left(a_j - 6b_j + 6 \frac{c_j^2}{m_j m_{j+1}} \right) + 3((-1)^s + \delta_{s1}) - 2 + (-1)^s \quad \blacksquare$$

在 4.5.3 小节中仔细地分析了欧几里得算法; 量 a_1, a_2, \dots, a_t 称做 h/k 的部分商。定理 4.5.3F 告诉我们, 迭代次数 t 绝不会超过 $\log_2 k$; 因此戴德金和数可很快地计算。项 $c_j^2/m_j m_{j+1}$ 可进一步简化, 习题 17 是计算 $\sigma(h, k, c)$ 的一个有效算法。

既然我们已经分析了广义戴德金和数, 就让我们应用已有知识来确定序列相关系数。

例 1 求当 $m = 2^{35}, a = 2^{34} + 1, c = 1$ 时的序列相关性。

解 由等式 (17) 我们有

$$C = (2^{35} \sigma(2^{34} + 1, 2^{35}, 1) - 3 + 6(2^{35} - (2^{34} - 1) - 1)) / (2^{70} - 1)$$

为计算 $\sigma(2^{34} + 1, 2^{35}, 1)$, 我们可以构造表

$$\begin{array}{llll} m_1 = 2^{35} & & c_1 = 1 & \\ m_2 = 2^{34} + 1 & a_1 = 1 & c_2 = 1 & b_1 = 0 \\ m_3 = 2^{34} - 1 & a_2 = 1 & c_3 = 1 & b_2 = 0 \\ m_4 = 2 & a_3 = 2^{33} - 1 & c_4 = 1 & b_3 = 0 \\ m_5 = 1 & a_4 = 2 & c_5 = 0 & b_4 = 1 \end{array}$$

由于 $h' = 2^{34} + 1$, 根据定理 D, 这个值为 $2^{33} - 3 + 2^{-32}$ 。于是

$$C = (2^{68} + 5) / (2^{70} - 1) = \frac{1}{4} + \epsilon, \quad |\epsilon| < 2^{-67} \quad (37)$$

这样的相关性对于随机性而言太高了。当然, 这个生成程序的效能非常低, 因为我们已经把它看做非随机的而拒绝了。

例 2 求当 $m = 10^{10}, a = 10001, c = 2113248653$ 时的近似序列相关性。

解 我们有 $C \approx \sigma(a, m, c)/m$, 且计算过程如下:

$$\begin{array}{llll} m_1 = 10000000000 & & c_1 = 2113248653 & \\ m_2 = 10001 & a_1 = 999900 & c_2 = 7350 & b_1 = 211303 \\ m_3 = 100 & a_2 = 100 & c_3 = 50 & b_2 = 73 \\ m_4 = 1 & a_3 = 100 & c_4 = 0 & b_3 = 50 \end{array}$$

$$\sigma(m_2, m_1, c_1) = -31.6926653544, \quad C \approx -3 \cdot 10^{-9} \quad (38)$$

这确实是 C 的非常不错的值。但是这个生成程序的效能仅为 3, 所以虽然它的序列相关性很低, 却不是随机数的好来源。序列相关性低是必要的但不是充分的。

例3 对一般的 a, m 和 c 估计序列相关性。

解 如果我们仅仅考虑式(30)的一个应用,则有

$$\sigma(a, m, c) \approx \frac{m}{a} + 6 \frac{c^2}{am} - 6 \frac{c}{a} - \sigma(m, a, c)$$

由习题12,现在 $|\sigma(m, a, c)| < a$, 因此

$$C \approx \frac{\sigma(a, m, c)}{m} \approx \frac{1}{a} \left(1 - 6 \frac{c}{m} + 6 \left(\frac{c}{m} \right)^2 \right) \quad (39)$$

在这个近似中,误差绝对值小于 $(a+6)/m$ 。

(39)中的结果是关于同余生成程序的随机性已知的头一个理论结果。R. R. Coveyou[JACM 7 (1960), 72~74]通过对0和 m 之间的所有实数 x 取平均值而不是仅仅考虑整数得到了这个结果(参考习题21);然后, Martin Greenberger[Math. Comp. 15(1961), 383~389]给出了包括一个误差项估计的严格推导。

于是开始了计算机科学历史上最可悲的章回之一! 尽管上面的近似十分正确,但在实践中它却使人伤心地被滥用了;人们摒弃了他们正在使用的极好的生成程序,而以从(39)的观点看像是好的而实际上却是糟糕的生成程序代替它们。有十年以上,仅仅由于理论发展的原因,日常使用的最普通的随机数生成程序严重地缺乏。

A Little learning is a dang'rous Thing.

学习太少是一件危险的事情。

Alexander Pope, *An Essay on Criticism*, 215 (1711)

如果我们要从过去的错误中学习,那么最好仔细地考察一下(39)是怎样被滥用的。首先,人人漫不经心地假定:如果在全周期上的序列相关性很小,则将是随机性的一个很好的保证;但是事实上,甚至对于这个序列的1000个连续的元素,序列相关性都不能保证是小的(参见习题14)。

其次,仅当 $a \approx \sqrt{m}$ 时,(39)和它的误差项才确保相对小的 C 值;因此,人们提议选择接近 \sqrt{m} 的乘数。事实上,我们将看到,几乎所有的乘数都给出实质上小于 $1/\sqrt{m}$ 的 C 值,因此,(39)不是对于真正的特性的一个非常好的近似,极小化 C 的一个粗略的上界并不极小化 C 。

第三,人们发现当

$$c/m \approx \frac{1}{2} \pm \frac{1}{6} \sqrt{3} \quad (40)$$

时,(39)给出最好的估计,因为这些值是 $1 - 6x + 6x^2 = 0$ 的根。“在不存在选择 c 的任何别的准则时,我们倒不如用这一个。”这个说法并非错误,但却会误导人们,因为经验已经表明,当 a 是一个好的乘数时 c 的值几乎对序列相关性没有任何影响;(40)的选择仅在像上边的例2的情况下才能大大地简化 C 。而且,在这样的情况下,我们已在愚弄自己,因为坏的乘数将在其它方面暴露它的缺陷。

显然,我们需要一个比(39)更好的估计;而且由于定理D,现在已有这样的一个估计可用,它主要是从 Ulrich Dieter 的工作导出的[Math. Comp. 25 (1971), 855~

883]。定理 D 意味着,如果 a/m 的部分商很小,则 $\sigma(a, m, c)$ 也将很小。确实,通过更加仔细地分析广义戴德金和数,有可能得到十分精确的估计。

定理 K 在定理 D 的假设之下,我们总有

$$-\frac{1}{2} \sum_{\substack{1 \leq j \leq t \\ j \text{ 奇}}} a_j - \sum_{\substack{1 \leq j \leq t \\ j \text{ 偶}}} a_j \leq \sigma(h, k, c) \leq \sum_{\substack{1 \leq j \leq t \\ j \text{ 奇}}} a_j + \frac{1}{2} \sum_{\substack{1 \leq j \leq t \\ j \text{ 偶}}} a_j - \frac{1}{2} \quad (41)$$

证明 见 D.E.Knuth, *Acta Arithmetica* 33 (1978), 297~325, 其中进一步证明了当有很大的部分商时,这些界实际上是最好的。 ■

例 4 对于 $a = 3141592621$, $m = 2^{35}$, c 为奇,估计序列相关性。

解 a/m 的部分商是 10, 1, 14, 1, 7, 1, 1, 1, 3, 3, 3, 5, 2, 1, 8, 7, 1, 4, 1, 2, 4, 2; 因此由定理 K,

$$-55 \leq \sigma(a, m, c) \leq 67.5$$

因此,保证了对于所有的 c , 序列相关性是极低的。

注意,这个界比我们能从(39)得到的要好得多,因为(39)中的误差的阶为 a/m ; 我们的“随机”乘数比根据(39)特别选择的看起来好的乘数要好得多。事实上,有可能证明:对所有同 m 互素的乘数 a 来说, $\sum_{j=1}^t a_j$ 的平均值是(见习题 4.5.3-35)

$$\frac{6}{\pi^2} (\ln m)^2 + O((\log m)(\log \log m)^4)$$

因此,一个随机乘数有很大的 $\sum_{j=1}^t a_j$, 比如说对于固定的 $\epsilon > 0$, 大于 $(\log m)^{2+\epsilon}$, 当 $m \rightarrow \infty$ 时,其概率趋于 0。这体现了这样一个经验论据,即在整个周期上几乎所有线性同余序列的序列相关性都极低。

以下的习题表明,其它一些预先检验,例如在整个周期上的序列检验,也可借助于一些广义戴德金和数来表达。从定理 K 得出,假定某些确定的分数(依赖于 a 和 m 而不依赖于 c)的部分商很小,则线性同余序列将通过这些检验。特别是,习题 19 的结果意味着当且仅当 a/m 的部分商不是很大时,成对的序列检验将被满意地通过。

Hans Rademacher 与 Emil Grosswald 所著的 *Dedekind Sums* 一书(Math. Assoc. of America, Carus Monograph No. 16, 1972), 讨论了戴德金和数的历史与性质及它们的推广。3.3.4 小节讨论了进一步的理论检验,包括在较高维上的序列检验。

习题——第一组

1. [M10] 借助于锯齿函数与 δ 函数表达“ $x \bmod y$ ”。
2. [HM22] 什么是函数 $((x))$ 的傅里叶级数展开式(借助于正弦与余弦)?
3. [M23] (N.J.Fine) 证明对所有实数 x , $\left| \sum_{k=0}^{n-1} \left(2^k x + \frac{1}{2} \right) \right| < 1$ 。

►4. [M19] 如果 $m = 10^{10}$, 假定生成程序的效能为 10, 则 d 的最高值是多少(用定理 P 的表示方法)?

5. [M21] 推导出式(17)来。

6. [M27] 设 $hh' + kk' = 1$ 。

a) 不使用引理 B, 证明对于所有整数 $c \geq 0$,

$$\sigma(h, k, c) = \sigma(h, k, 0) + 12 \sum_{0 \leq j < c} \left(\left(\frac{h'j}{k} \right) \right) + 6 \left(\left(\frac{h'c}{k} \right) \right)$$

b) 证明如果 $0 < j < k$,

$$\left(\left(\frac{h'j}{k} \right) \right) + \left(\left(\frac{k'j}{h} \right) \right) = \frac{j}{hk} - \frac{1}{2} \delta \left(\frac{j}{h} \right)$$

c) 在引理 B 的假定下, 证明式(21)。

►7. [M24] 当 $c = 0$ 时, 利用习题 1.2.4-45 的一般互反律给出互反律(19)的一个证明。

►8. [M34] (L. Carlitz) 设

$$\rho(p, q, r) = 12 \sum_{0 \leq j < r} \left(\left(\frac{jp}{r} \right) \right) \left(\left(\frac{jq}{r} \right) \right)$$

通过推广引理 B 中所用的证明方法, 证明下列由 H. Rademacher 给出的漂亮恒等式: 如果 p, q, r 中每一个都与其它两个互素, 则

$$\rho(p, q, r) + \rho(q, r, p) + \rho(r, p, q) = \frac{p}{qr} + \frac{q}{rp} + \frac{r}{pq} - 3$$

($c = 0$ 时的戴德金和数的互反律是 $r = 1$ 的特殊情况。)

9. [M40] 沿着习题 7 中那个特例的证明路线, 能否给出 Rademacher 恒等式(习题 8)的一个简单证明?

10. [M20] 证明当 $0 < h < k$ 时, 有可能很容易地借助于 $\sigma(h, k, c)$ 来表达 $\sigma(k - h, k, c)$ 和 $\sigma(h, k, -c)$ 。

11. [M30] 正文中给出的公式说明当 h 与 k 互素时, 如何计算 $\sigma(h, k, c)$ 。对于一般情况, 证明

a) $\sigma(dh, dk, dc) = \sigma(h, k, c)$, 整数 $d > 0$;

b) $\sigma(h, k, c + \theta) = \sigma(h, k, c) + 6 \left(\left(\frac{h'c}{k} \right) \right)$, 对于整数 c , 实数 $0 < \theta < 1$, $h \perp k$ 且 $hh' \equiv 1 \pmod{k}$ 。

12. [M24] 证明如果 h 与 k 互素且 c 是整数, 则 $|\sigma(h, k, c)| \leq (k-1)(k-2)/k$ 。

13. [M24] 推广等式(26), 使得它给出 $\sigma(h, k, c)$ 的一个表达式。

►14. [M20] 对于有 $m = 2^{35}$, $a = 2^{18} + 1$, $c = 1$ 的线性同余生成程序, 对它进行三批有 1000 个连续的数的序列相关检验。而且在每种情况下, 在 0.2 和 0.3 之间, 结果有非常高的相关。取遍周期的所有 2^{35} 个数, 问这个生成程序的序列相关是什么?

15. [M21] 推广引理 B 使得它能应用于所有 c 的实数值, $0 \leq c < k$ 。

16. [M24] 给定(33)中定义的欧几里得表, 令 $p_0 = 1$, $p_1 = a_1$, 且对于 $1 < j \leq t$, $p_j = a_j p_{j-1} + p_{j-2}$ 。证明定理 D 中的和式的复杂部分可以重写如下, 以避免非整数计算:

$$\sum_{1 \leq j \leq t} (-1)^{j+1} \frac{c_j^2}{m_j m_{j+1}} = \frac{1}{m_1} \sum_{1 \leq j \leq t} (-1)^{j+1} b_j (c_j + c_{j+1}) p_{j-1}$$

[提示: 证明对于 $1 \leq r \leq t$ 我们有 $\sum_{1 \leq j \leq r} (-1)^{j+1} / m_j m_{j+1} = (-1)^{r+1} p_{r-1} / m_1 m_{r+1}$ 。]

17. [M22] 试设计一个算法,它对于满足定理 D 的假设的整数 h, k, c , 计算 $\sigma(h, k, c)$ 。你的算法应当只使用(未限定精确值的)整数算术,而且它应当以 $A + B/k$ 的形式产生答案,其中 A 和 B 是整数(参考习题 16)。如果可能,只使用有限个变量临时存储,而不使用像 a_1, a_2, \dots, a_t 这样的数组。

► 18. [M23] (U. Dieter) 给定正整数 h, k, z , 设

$$S(h, k, c, z) = \sum_{0 \leq j < z} \left(\left\lfloor \frac{hj + c}{k} \right\rfloor \right)$$

证明这个函数可借助于广义戴德金和数及锯齿函数以“封闭形式”计算。[提示:当 $z \leq k$ 时,对于 $0 \leq j < z$, 量 $\lfloor j/k \rfloor - \lfloor (j-z)/k \rfloor$ 等于 1, 且对于 $z \leq j < k$ 它等于 0, 所以我们可以引进这个分数以及在 $0 \leq j < k$ 上的和。]

► 19. [M23] 证明:借助于广义戴德金和数,通过求当 a, β, a', β' 是给定整数且 $0 \leq a < \beta \leq m$, $0 \leq a' < \beta' \leq m$ 时, $a \leq X_n < \beta$ 和 $a' \leq X_{n+1} < \beta'$ 的概率的一个公式,可以在全周期上分析序列检验。[提示:考虑量 $\lfloor (x-a)/m \rfloor - \lfloor (x-\beta)/m \rfloor$ 。]

20. [M29] (U. Dieter) 借助于广义戴德金和数,求出 $X_n > X_{n+1} > X_{n+2}$ 的概率公式,推广定理 P。

习题——第二组

在许多情况下,对于整数的精确计算是十分困难的,但我们可以尝试研究当对 x 的所有实数值取平均,以代替把计算限制为整数值时出现的概率。尽管这些结果仅仅是近似的,但它们还是能展示出此课题的一点面貌。

处理 0 和 1 之间的数 U_n 是方便的;而且对于线性同余序列 $U_n = X_n/m$, 我们有 $U_{n+1} = \{aU_n + \theta\}$, 其中 $\theta = c/m$ 且 $\{x\}$ 表示 $x \bmod 1$ 。例如,序列相关的公式现在变成

$$C = \left(\int_0^1 x \{ax + \theta\} dx - \left(\int_0^1 x dx \right)^2 \right) / \left(\int_0^1 x^2 dx - \left(\int_0^1 x dx \right)^2 \right)$$

► 21. [HM23] (R. R. Coveyou) 在刚才给出的公式中, C 的值是多少?

► 22. [M22] 设 a 是一个整数,且 $0 \leq \theta < 1$ 。如果 x 是 0 与 1 之间的一个实数,且如果 $s(x) = \{ax + \theta\}$, 问 $s(x) < x$ 的概率是多少? (这是定理 P 的“实数模拟”。)

23. [28] 以前的习题给出了 $U_{n+1} < U_n$ 的概率。假定 U_n 是 0 与 1 之间的随机实数,问 $U_{n+2} < U_{n+1} < U_n$ 的概率是多少?

24. [M29] 在上题的假设下,但 $\theta = 0$ 除外,证明 $U_n > U_{n+1} > \dots > U_{n+t-1}$ 以

$$\frac{1}{t!} \left(1 + \frac{1}{a} \right) \dots \left(1 + \frac{t-2}{a} \right)$$

的概率出现。假定 U_n 是在 0 与 1 之间随机选择的,问在 U_n 处开始的递减运行的平均长度是多少?

► 25. [M25] 设 a, β, a', β' 是实数,且 $0 \leq a < \beta \leq 1, 0 \leq a' < \beta' \leq 1$ 。在习题 22 的假设之下, $a \leq x < \beta$ 和 $a' \leq s(x) < \beta'$ 的概率是多少? (这是习题 19 的“实数”模拟。)

26. [M21] 考虑一个“斐波那契”生成程序,其中 $U_{n+1} = \{U_n + U_{n-1}\}$ 。假定 U_1 和 U_2 是在 0 与 1 之间独立地随机选择的。求 $U_1 < U_2 < U_3, U_1 < U_3 < U_2, U_2 < U_1 < U_3$ 等的概率。[提示:根据 x, y 和 $\{x+y\}$ 的相对顺序,把单位正方形 $\{(x, y) | 0 \leq x, y < 1\}$ 分成六部分,并确定每个

部分的面积。]

27. [M32] 在上题的斐波那契生成程序中, 设除 $U_0 > U_1$ 外, U_0 和 U_1 是在单位正方形中独立选择的。试确定 U_1 是长度为 k 的上行运行的开头而使 $U_0 > U_1 < \cdots < U_k > U_{k+1}$ 的概率。试把这同一个随机序列的相应概率作一比较

28. [M35] 根据式 3.2.1.3-(5), 效能为 2 的一个线性同余生成程序满足条件 $X_{n+1} = 2X_n + X_{n-1} \equiv (a+1)c \pmod{m}$ 。把这个式子抽象化为 $U_{n+1} = \{a + 2U_n - U_{n-1}\}$, 考虑这种情况下的生成程序。像在习题 26 中那样, 把单位正方形分成对于每对 (U_1, U_2) , 表明 U_1, U_2 和 U_3 的相对顺序的诸部分。假定 U_1, U_2 在单位正方形中随机地选择, 是否有这样的 a 的值, 对于它, 所有六个可能的顺序都以 $\frac{1}{6}$ 的概率被达到?

3.3.4 谱检验

这一小节我们将研究校验线性同余随机数生成程序质量的一个特别重要的方式。不仅所有好的生成程序都能通过这个检验, 现在已知的所有坏的生成程序实际上都不能通过它。因此, 到目前为止, 它是已知最强有力的检验, 值得特别注意。我们的讨论也将明确提出, 由线性同余序列及其推广, 能够预期的随机性程度的一些基本限制。

谱检验体现了上边各节中所研究的经验检验和理论检验两者的一些方面。它像理论检验, 因为它处理序列全周期的性质, 同时, 它也像经验检验, 因为它需要一个计算机程序来判定结果。

A. 检验的基本思想 最重要的随机性准则似乎依赖于序列的 t 个相继的元素分布的一些性质, 而谱检验直接地处理这个分布。如果我们有一个周期为 m 的序列, 它的思想是分析 t 维空间中所有 m 个点的集合

$$\{(U_n, U_{n+1}, \dots, U_{n+t-1}) \mid 0 \leq n < m\} \quad (1)$$

为简便起见, 我们将假定有一个(使得 $c \neq 0$ 的)极大周期长度为 m 的线性同余序列 (X_0, a, c, m) , 或者假定 m 为素数及 $c = 0$ 且周期长度为 $m - 1$ 。在后一情况下, 我们将把点 $(0, 0, \dots, 0)$ 加到集合(1)中, 使得总是总共有 m 个点; 当 m 很大时, 这额外的点的影响可忽略, 而且它使理论简单得多。在这些假定之下, 集合(1)可重写做

$$\left\{ \frac{1}{m} (x, s(x), s(s(x)), \dots, s^{(t-1)}(x)) \mid 0 \leq x < m \right\} \quad (2)$$

其中

$$s(x) = (ax + c) \bmod m \quad (3)$$

是 x 的“后继”。我们仅仅考虑 t 维中所有这样的点的集合, 而不是这些点实际上被生成的顺序。但是生成的顺序被反映在向量分量之间的相关性上; 而且谱检验通过处理所有点(2)的总体性来研究对各种维数 t 的这种相关性。

例如, 图 8 示出了对于具有

$$s(x) = (137x + 187) \bmod 256 \quad (4)$$

的生成程序,这是在二维和三维下的一个典型的小例子。当然,周期长度为 256 的一个生成程序几乎不可能是随机的,但 256 是很小的数,因而可以画出这个框图,以便在我们转到有实际意义的更大的 m 之前,首先做一些分析。

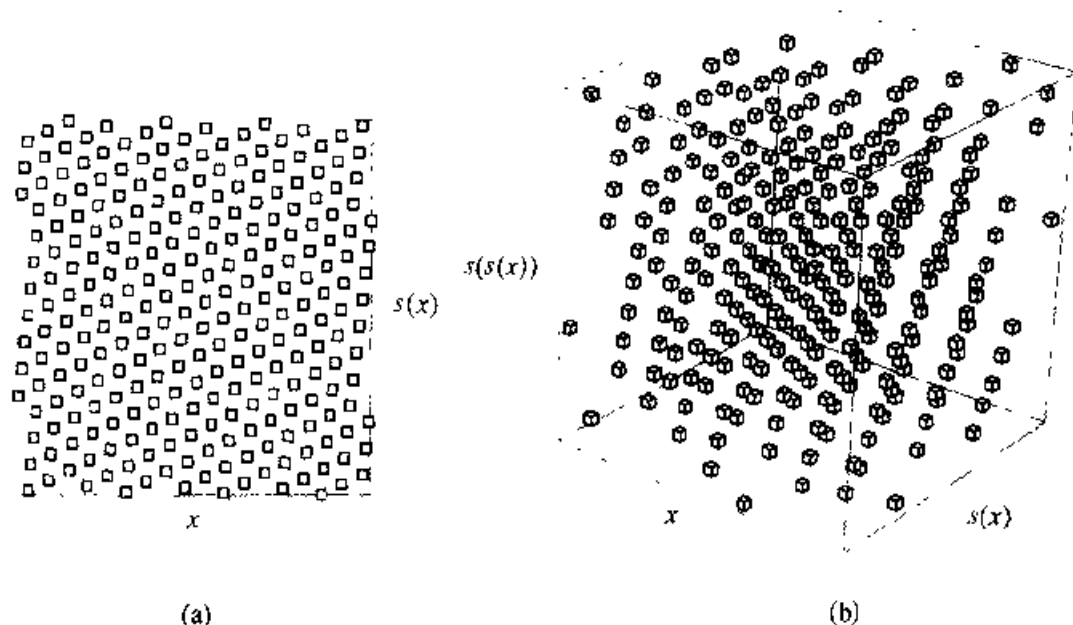


图 8

(a) 当 $X_{n+1} = (137X_n + 187) \bmod 256$ 时,由所有相继的点偶 (X_n, X_{n+1}) 形成的二维的栅格;

(b) 三元组 (X_n, X_{n+1}, X_{n+2}) 的三维栅格。

或许,关于图 8(a)的方框模式最引人注目的事,是我们可以通过相当少量的平行线来全部覆盖它们;确实,有许多不同的平行线族,它们将碰上所有点。例如,一组 20 条近乎垂直的线就行,一组 21 条向上倾斜大约 30° 的线也行。当驾车经过已经播种的大片农田时,我们普遍地发现类似的情形。

如果在三维中考虑同样的生成程序,我们就得到一个立方体中的 256 个点,它是通过如同在图 8(b)中所示那样,对图 8(a)中的平面的 256 个点 $(x, s(x))$ 都附加一个“高”分量 $s(s(x))$ 得到的。想像这三维的晶体结构已经做成一个物理模型,一个我们可以在手上转动的立方体;当转动它时,我们注意到包含所有点的各种平行平面族。按 Wallace Givens 的话说,随机数“主要保持在平面内”。

乍看起来我们可能想,这种系统的行为是如此的非随机,弄得同余生成程序十分没有价值了;但若更仔细地考虑,并注意到在实际使用中 m 是很大的,对它就能有更好地的洞察。图 8 中的正规结构实际上是当在高效能的显微镜下考察我们的随机数时看到的“颗粒”。如果我们真正地取 0 和 1 之间的随机数,并且舍入或截取它们成有限的精度,使得它们每一个都是 $1/\nu$ 的整数倍(其中 ν 是给定的数),则当用一个显微镜观察时,我们得到的 t 维点(1)将有极其规则的特征。

设 $1/\nu_2$ 是遍历包含二维中的点 $\{(x/m, s(x)/m)\}$ 的所有平行直线族时,直线间的极大距离。我们将称 ν_2 为随机数生成程序的二维精度,因为相继的数偶有一

个精良的结构,它大约能精确到 ν_2 分之一的程度。类似地,设 $1/\nu_3$ 是遍历包括所有点 $\{(x/m, s(x)/m, s(s(x))/m)\}$ 的所有平行平面族时,平面之间的极大距离;我们将称 ν_3 为三维精度。 t 维精度 ν_t 是遍历包括所有点 $\{(x/m, s(x)/m, \dots, s^{[t-1]}(x)/m)\}$ 的所有平行的 $(t-1)$ 维超平面族时,超平面间的极大距离的倒数。

周期序列同已经截取成为 $1/\nu$ 的倍数的真正随机序列之间的实质性差别,在于真正随机序列的“精度”在所有维中都是相同的,而一个周期序列的“精度”却随着 t 增加而减少。其实,由于当 m 是周期长度时在 t 维立体中仅有 m 个点,我们不可能达到多于大约 $m^{1/t}$ 的 t 维精度。

当考虑 t 个相继值的独立性时,计算机生成的随机数的性能实质上就像将真正随机的数截取成 $\lg \nu_t$ 个二进位那样,其中 ν_t 随 t 的增加而减少。在实践中,我们所需要的也就是这样变化的精度。我们并不一定要求 10 维精度为 2^{32} 。也就是说,在一台 32 位字长机器上所有 $(2^{32})^{10}$ 个可能的 10 元组 $(U_n, U_{n+1}, \dots, U_{n+9})$ 都将是同样可能的。对于这样大的 t 值,我们仅仅要求 $(U_n, U_{n+1}, \dots, U_{n+t-1})$ 的一些前导位具有独立随机的性质。

另一方面,当实际应用中要求高分辨率的随机数序列时,简单的线性同余序列肯定不适用;应当使用具有更大周期的生成程序,尽管它实际上将仅仅生成这个周期的一小部分。把这个周期取平方实质上等于把更高维下的精度取平方,即,它将把精度的二进位有效数字加倍。

谱检验是以小的 t (比如说 $2 \leq t \leq 6$) 的 ν_t 值为基础的。为了发现在一个序列中重要的缺陷,维数 2, 3 和 4 似乎是适当的,但由于我们正在考虑整个周期,明智的是稍微谨慎些并增加另一维或两维;另一方面,对 $t \geq 10$ 的 ν_t 值,似乎无论如何都没有实际意义(这是幸运的,因为当 $t \geq 10$ 时计算 ν_t 就显得相当困难了)。

在谱检验和序列检验之间有一个含糊的关系;例如,像在习题 3.3.3-19 中那样在整个周期上进行的序列检验的一个特殊情况,它计算图 8(a) 中的 64 个方格中每个方格里边的小方格数。主要差别在于谱检验转动这些点以便发现最不合意的方向。在本节稍后,我们将回过头来考虑序列检验。

乍一看,我们似应仅仅对于适当高的一个 t 值应用谱检验;如果一个生成程序通过三维中的检验,则似乎它也应通过二维检验。因此,我们倒不如省去后一个检验。在这个推理中出现了谬误,因为我们在较低维中应用了更严格的条件。对于序列检验也出现类似的情况:考虑这样一个生成程序,当单位立方体已经分成大小为 $1/4 \times 1/4 \times 1/4$ 的 64 个小立方体时,它十分合适地在单位立方体的每个小立方体中(非常严格地)产生几乎相同的点数;当单位正方形已被分成 64 个 $1/8 \times 1/8$ 的小正方形时,这同样的生成程序可能产生单位正方形的一些完全空的子正方形。由于我们增加了在较低维中的期望,故要求对每维进行分开的检验。

$\nu_t \leq m^{1/t}$ 不总是对的,尽管当诸点形成一个矩形栅格时这个上界是正确的。例如,在图 8 中,原来 $\nu_2 = \sqrt{274} > \sqrt{256}$, 因为接近六边形的结构使 m 个点比在一个

严格的矩形结构中更加靠在一起。

为了建立有效的计算 ν_i 的算法,我们必须更深入地考察相关的数学理论。因此,建议不擅长数学的读者跳到本小节的 D 部分,在那里,谱检验是随着若干例子作为一个“插件”进行介绍的。但是,谱检验的数学背景仅仅要求某些初等向量运算。

有些作者已经建议,使用平行覆盖直线或覆盖超平面的极小数 N_i 作为准则,以代替它们之间的极大距离 $1/\nu_i$ 。然而这个数并不显得像上边定义的精确概念那样重要;因为它受到直线或超平面的斜率和立方体的坐标轴相符到什么程度这一偏倚问题的影响。例如,按照下面的等式(14)以及 $(u_1, u_2) = (18, -2)$, 20 条覆盖图 8(a)的所有点的近于垂直的线实际上有 $1/\sqrt{328}$ 的偏离,而这可能错误地蕴涵 $1/\sqrt{328}$ 的精度,也许甚至 $1/20$ 的精度。而对于具有 $7/15$ 的斜率的更大的 21 条直线族仅仅实现 $1/\sqrt{274}$ 的精度;有 $-11/13$ 的斜率的另一个 24 条直线族,也有比 20 条直线族更大的直线间距离,因为 $1/\sqrt{290} > 1/\sqrt{328}$ 。直线族在单位超立方体的边界中起作用的确切方式似乎并非是一个特别“清楚”或重要的准则;然而喜欢计算超平面的人们,有可能用十分类似于我们计算 ν_i 的方法来计算 N_i (见习题 16)。

***B. 检验后面的理论** 为了分析基本集合(2),我们从观察

$$\frac{1}{m}s^{[j]}(x) = \left(\frac{a^j x + (1 + a + \cdots + a^{j-1})c}{m} \right) \bmod 1 \quad (5)$$

开始,通过周期地扩充集合我们可以去掉“mod 1”运算,并构造原来的 t 维超立方体在所有方向的无限多个复制品。这给出集合

$$L = \left\{ \left(\frac{x}{m} + k_1, \frac{s(x)}{m} + k_2, \cdots, \frac{s^{(t-1)}(x)}{m} + k_t \right) \mid \text{整数 } x, k_1, k_2, \cdots, k_t \right\} = \\ \left\{ V_0 + \left(\frac{x}{m} + k_1, \frac{ax}{m} + k_2, \cdots, \frac{a^{t-1}x}{m} + k_t \right) \mid \text{整数 } x, k_1, k_2, \cdots, k_t \right\}$$

其中

$$V_0 = \frac{1}{m}(0, c, (1+a)c, \cdots, (1+a+\cdots+a^{t-2})c) \quad (6)$$

是一个常向量。在 L 的这个表示中变量 k_1 是冗余的,因为我们可以把 $(x, k_1, k_2, \cdots, k_t)$ 变为 $(x + k_1 m, 0, k_2 - ak_1, \cdots, k_t - a^{t-1}k_1)$, 不失一般性,把 k_1 变成 0。因此我们得到比较简单的公式

$$L = \{ V_0 + y_1 V_1 + y_2 V_2 + \cdots + y_t V_t \mid \text{整数 } y_1, y_2, \cdots, y_t \} \quad (7)$$

其中

$$V_1 = \frac{1}{m}(1, a, a^2, \cdots, a^{t-1}) \quad (8)$$

$$V_2 = (0, 1, 0, \cdots, 0), V_3 = (0, 0, 1, \cdots, 0), \cdots, V_t = (0, 0, 0, \cdots, 1) \quad (9)$$

对于所有的 j , 满足 $0 \leq x_j < 1$ 的 L 的点 (x_1, x_2, \cdots, x_t) 恰是我们原集合(2)的 m 个点。

注意,增量 c 仅出现于 V_0 ,而且 V_0 的效果仅仅是移动 L 的所有元素,而不改变它们的相对距离;因此 c 无论如何不影响谱检验,而且当我们计算 ν_t 时,也可假定 $V_0 = (0, 0, \dots, 0)$ 。当 V_0 为 0 向量时我们有一个所谓的点格

$$L_0 = \{y_1 V_1 + y_2 V_2 + \dots + y_t V_t \mid \text{整数 } y_1, y_2, \dots, y_t\} \quad (10)$$

而且我们的目标是研究在包括 L_0 的所有点的平行超平面族中,相邻的 $(t-1)$ 维超平面间的距离。

平行的 $(t-1)$ 维超平面的一个族可由垂直于它们的一个非 0 向量 $U = (u_1, \dots, u_t)$ 来定义;而且在一个特殊超平面上的点集是:

$$\{(x_1, \dots, x_t) \mid x_1 u_1 + \dots + x_t u_t = q\} \quad (11)$$

其中 q 是对于族中每个超平面都不一样的一个常数。换言之,每个超平面是使得点积 $X \cdot U$ 有一给定值 q 的所有向量 X 的集合。在我们的情况下,超平面都以一个固定的距离分开,而且它们之一含 $(0, 0, \dots, 0)$;因此我们可以调整 U 的量度使得所有整数 q 值的集合给出族中的所有超平面。然后相邻超平面间的距离是从 $(0, 0, \dots, 0)$ 到 $q=1$ 的超平面的极小距离,即

$$\min_{\text{实数 } x_1, \dots, x_t} \left\{ \sqrt{x_1^2 + \dots + x_t^2} \mid x_1 u_1 + \dots + x_t u_t = 1 \right\} \quad (12)$$

柯西不等式(参考习题 1.2.3-30)告诉我们

$$(x_1 u_1 + \dots + x_t u_t)^2 \leq (x_1^2 + \dots + x_t^2)(u_1^2 + \dots + u_t^2) \quad (13)$$

因此当每个 $x_j = u_j / (u_1^2 + \dots + u_t^2)$ 时, (12) 中极小值出现;邻近超平面之间的距离是

$$1 / \sqrt{u_1^2 + \dots + u_t^2} = 1/U \text{ 的长度} \quad (14)$$

换言之,我们寻求的量 ν_t 恰是定义含有 L_0 的所有元素的一个超平面族 $\{X \cdot U = q \mid \text{整数 } q\}$ 的最短向量 U 的长度。

这样一个向量 $U = (u_1, \dots, u_t)$ 必须非零,而且对于 L_0 中所有的 V 必定满足 $V \cdot U = \text{整数}$ 。特别是因为点 $(1, 0, \dots, 0), (0, 1, 0, \dots, 0), \dots, (0, 0, \dots, 0, 1)$ 都在 L_0 中,所有的 u_j 必为整数。其次,由于 V_1 在 L_0 中,我们必定有 $\frac{1}{m}(u_1 + au_2 + \dots + a^{t-1}u_t)$ 为整数,即

$$u_1 + au_2 + \dots + a^{t-1}u_t \equiv 0 \pmod{m} \quad (15)$$

反之,任何满足 (15) 的非 0 整向量 $U = (u_1, \dots, u_t)$ 定义了具有所要求的性质的一个超平面族,因为 L_0 的所有点都将被包括:对于所有整数 y_1, \dots, y_t , 点积 $(y_1 V_1 + \dots + y_t V_t) \cdot U$ 将是一个整数。我们已经证明

$$\begin{aligned} \nu_t^2 &= \min_{(u_1, \dots, u_t) \neq (0, \dots, 0)} \{u_1^2 + \dots + u_t^2 \mid u_1 + au_2 + \dots + a^{t-1}u_t \equiv 0 \pmod{m}\} = \\ &\quad \min_{(x_1, \dots, x_t) \neq (0, \dots, 0)} ((mx_1 - ax_2 - a^2x_3 - \dots - a^{t-1}x_t)^2 + x_2^2 + x_3^2 + \dots + x_t^2) \end{aligned} \quad (16)$$

C. 推导一个计算方法 我们现在已经把谱检验的问题归结为求极小值(16)的问题,但是我们究竟怎样才能不太长的时间内确定极小值呢?硬求是不行的,因为实际使用中 m 非常大。

如果我们建立解一个甚至更为一般的问题的计算方法,它将是有趣的而且大概更有用:给定任何非奇异系数矩阵 $U = (u_{ij})$,对所有非0整向量 (x_1, \dots, x_t) 求

$$f(x_1, \dots, x_t) = (u_{11}x_1 + \dots + u_{t1}x_t)^2 + \dots + (u_{1t}x_1 + \dots + u_{tt}x_t)^2 \quad (17)$$

的极小值。表达式(17)称做 t 个向量的一个“正定二次型”。因为 U 非奇异,因此除非 x_j 全为0,否则(17)不可能为0。

把 U 的行写作 U_1, \dots, U_t , 则(17)可以写成

$$f(x_1, \dots, x_t) = (x_1 U_1 + \dots + x_t U_t) \cdot (x_1 U_1 + \dots + x_t U_t) \quad (18)$$

即向量 $(x_1 U_1 + \dots + x_t U_t)$ 长度的平方。非奇异矩阵 U 有一个逆,它意味着我们可以惟一地确定向量 V_1, \dots, V_t , 使得

$$U_i \cdot V_j = \delta_{ij}, \quad 1 \leq i, j \leq t \quad (19)$$

例如,在谱检验中出现的特殊形式(16)中,我们有

$$\begin{aligned} U_1 &= (m, 0, 0, \dots, 0), & V_1 &= \frac{1}{m}(1, a, a^2, \dots, a^{t-1}) \\ U_2 &= (-a, 1, 0, \dots, 0), & V_2 &= (0, 1, 0, \dots, 0) \\ U_3 &= (-a^2, 0, 1, \dots, 0), & V_3 &= (0, 0, 1, \dots, 0) \\ &\dots\dots\dots & & \\ U_t &= (-a^{t-1}, 0, 0, \dots, 1) & V_t &= (0, 0, 0, \dots, 1) \end{aligned} \quad (20)$$

这些 V_j 恰是我们用来定义原来的格 L_0 的向量(8), (9)。读者可能怀疑这是一个巧合——其实,如果以一个任意的格 L_0 开始,它是由任意一组线性无关向量 V_1, \dots, V_t 定义的,则我们上边已经使用的论证可加以推广,以证明在一个覆盖族中,把超平面之间的距离极大化等价于把(17)极小化,这里系数 u_{ij} 由(19)定义(见习题2)。

我们极小化(18)的头一步是把它归结为一个有穷问题,即证明,当求极小值时我们不需要检验无限多的向量 (x_1, \dots, x_t) 。正是在这里,我们引入向量 V_1, \dots, V_t , 我们有

$$x_k = (x_1 U_1 + \dots + x_t U_t) \cdot V_k$$

而且柯西不等式告诉我们

$$((x_1 U_1 + \dots + x_t U_t) \cdot V_k)^2 \leq f(x_1, \dots, x_t) (V_k \cdot V_k)$$

因此我们已经推导出关于每个坐标 x_k 的一个有用的上界:

引理 A 设 (x_1, \dots, x_t) 是极小化(18)的一个非0向量,并设 (y_1, \dots, y_t) 是任何非0整向量,则

$$x_k^2 \leq f(y_1, \dots, y_t) (V_k \cdot V_k), \quad \text{对于 } 1 \leq k \leq t \quad (21)$$

特别是,对所有 i 令 $y_i = \delta_{ij}$,

$$x_k^2 \leq (U_j \cdot U_j)(V_k \cdot V_k), \quad \text{对于 } 1 \leq j, k \leq t \quad | \quad (22)$$

引理 A 把问题归结为有限的查找,但是(21)右边通常太大了,使得不可能进行一个穷尽的查找;我们至少还需要另一个思想。在这样的时刻,一句老的格言提供了很好的忠告:“如果你不能像问题所述的那样解决这个问题,就把它改成为有同样答案的较简单的问题。”例如,欧几里得算法有这个形式;如果我们不知道输入数的 gcd,我们把它变成有同样 gcd 的较小数。(事实上,一个稍微更一般的方法大概是发现几乎所有算法的基础:“如果你不能直接地解一个问题,就把它变成为一个或多个较简单的问题,由它们的解你可以解原来的问题。”)

在我们的情况下,一个较简单的问题是要求较少查找的一个问题,因为(22)右边是较小的。我们将使用的关键思想是有可能把一个二次形改变成为对所有实用目的而言和它等价的另一个二次形。设 j 是任何固定的下标, $1 \leq j \leq t$; 设 $(q_1, \dots, q_{j-1}, q_{j+1}, \dots, q_t)$ 是 $t-1$ 个整数的任何序列;并考虑下列向量的变换:

$$\begin{aligned} V'_i &= V_i - q_i V_j, & x'_i &= x_i - q_i x_j, & U'_i &= U_i, \text{ 对于 } i \neq j; \\ V'_j &= V_j, & x'_j &= x_j, & U'_j &= U_j + \sum_{i \neq j} q_i U_i \end{aligned} \quad (23)$$

容易看出,新的向量 U'_1, \dots, U'_t 定义一个二次型 f' , 它有 $f'(x'_1, \dots, x'_t) = f(x_1, \dots, x_t)$; 其次,基本正交条件保持(19)成立,因为容易校验 $U'_i \cdot V'_j = \delta_{ij}$ 。由于 (x_1, \dots, x_t) 跑遍所有非零整向量,所以 (x'_1, \dots, x'_t) 也是;因此新的形式 f' 有和 f 一样的极小值。

我们的目标是使用变换(23),而且对于所有的 i ,以 U'_i 代替 U_i ,以 V'_i 代替 V_i ,以便使(22)右边成为小值;而且当 $U_j \cdot U_j$ 和 $V_k \cdot V_k$ 都小时,(22)右边也小。因此,关于变换(23)自然要问下列两个问题:

- 选什么样的 q_i 能使 $V'_i \cdot V'_i$ 尽可能小?
- 选什么样的 $q_1, \dots, q_{j-1}, q_{j+1}, \dots, q_t$ 能使 $U'_j \cdot U'_j$ 尽可能小?

首先,对 q_1 的实数值解决这些问题是最容易的。问题 a) 十分简单,因为

$$\begin{aligned} (V_i - q_i V_j) \cdot (V_i - q_i V_j) &= V_i \cdot V_i - 2q_i V_i \cdot V_j + q_i^2 V_j \cdot V_j = \\ &= (V_j \cdot V_j)(q_i - (V_i \cdot V_j / V_j \cdot V_j))^2 + \\ &+ V_i \cdot V_i - (V_i \cdot V_j)^2 / V_j \cdot V_j \end{aligned}$$

而且当

$$q_i = V_i \cdot V_j / V_j \cdot V_j \quad (24)$$

时出现极小值。从几何上看,我们问的是应从 V_i 减去 V_j 的多少倍,使得得到的向量 V'_i 有极小长度,而答案是,选择 q_i 使得 V'_i 垂直于 V_j (即使得 $V'_i \cdot V_j = 0$); 下面的(25)使这个问题一目了然。

转到问题 b),我们要选择 q_i 使得 $U_j + \sum_{i \neq j} q_i U_i$ 有极小的长度;从几何上说,我们要从 U_j 开始,并且把某个向量加到它的点是 $\{U_i | i \neq j\}$ 的倍数之和的 $(t-1)$ 维超平面中。再次,最好的解是选择使得 U'_j 垂直于超平面的那些东西,使得对于所

$$(25)$$

有 $k \neq j, U_j' \cdot U_k = 0$:

$$U_j \cdot U_k + \sum_{i \neq j} q_i (U_i \cdot U_k) = 0, \quad 1 \leq k \leq t, \quad k \neq j \quad (26)$$

[关于问题 b) 的一个解必须满足这些 $t-1$ 个方程的严格证明请见习题 12.]

既然我们已经回答了问题 a) 和 b), 我们就有点左右为难了; 我们是应该按 (24) 来选择 q_i , 使得 $V_i' \cdot V_i'$ 极小呢, 还是应该按 (26), 使得 $U_j' \cdot U_j'$ 变极小? 这两个选择的任何一个都对 (22) 的右边做了改进, 因而并不能立即明了哪一个应获优先。幸而, 对于此两难处境有一个非常简单的答案: 条件 (24) 和 (26) 完全相同 (请见习题 7), 因此问题 a) 和 b) 有相同答案; 我们很高兴, 可以同时减少 U 和 V 的长度。确实, 我们刚刚重新发现了 Gram-Schmidt 正交过程 [见 Crelle 94 (1883), 41~73]。

认识到我们仅仅对于 q_i 的实数值处理了问题 a) 和 b), 我们的高兴心情就要受到节制。我们的应用把我们限制于整数值, 所以不能使 V_i' 精确地垂直于 V_j 。对于问题 a) 我们所能做的最好的事是令 q_i 是最接近于 $V_i \cdot V_j / V_i \cdot V_i$ 的整数 (请见 (25))。结果是这并非总是对于问题 b) 的最好的解; 事实上, U_j' 有时可能长于 U_j 。然而, 界 (21) 绝不增加, 因为我们能够记得迄今找到的 $f(y_1, \dots, y_t)$ 的最小的值。因此单独基于问题 a) 对 q_i 的一个选择是十分令人满意的。

如果我们以这样一种方式, 即向量 V_i 没有一个变得更长, 而且至少有一个要变短些, 重复应用变换 (23), 那么就绝不会陷入一个循环中去; 即是, 在进行了这种类型的一系列非平凡的变换之后, 我们将不会再考虑相同的二次型了。但是实际上我们总会被搁浅的, 因为对于 $1 \leq j \leq t$, (23) 的变换将不能缩短向量 V_1, \dots, V_t 中的任何一个。这时, 利用引理 A 的界——在大多数情况下它是十分小的——我们可以逆转到一个穷尽查找上。有时这些界 (21) 将是很差的, 而另一类型的变换通常将使算法再次开动并把界减少 (见习题 18)。然而, 变换 (23) 本身已经证明, 当如同下面讨论的算法那样安排计算时, 它是令人惊讶地强有力的。

D. 如何实施谱检验 现在由我们的讨论得出一个有效的计算过程。R.W.Gosper 和 U.Dieter 已经发现, 有可能使用较低维数的结果来使谱检验在更高维中运行得快得多。这个改进已连同高斯对二维情况的重大简化被加入到下列算法中 (请见习题 5)。

算法 S (谱检验) 对于 $2 \leq t \leq T$, 给定 a, m 和 T , 其中 $0 < a < m$ 且 a 与 m 互素, 这个算法确定

$$\nu_t = \min \left\{ \sqrt{x_1^2 + \cdots + x_t^2} \mid x_1 - ax_2 + \cdots + a^{t-1}x_t \equiv 0 \pmod{m} \right\} \quad (27)$$

的值。(极小是对于所有非零整向量 (x_1, \dots, x_t) 来选取的, 而且数 ν_t 如同在上面的正文中所讨论的那样, 测量随机数生成程序的 t 维精度。) 本算法所进行的所有算术运算都是对于其量级很少(如果有的话)超过 m^2 的整数进行的, 步骤 S7 除外; 事实上, 在计算期间几乎所有整数变量的绝对值将小于 m 。

当对于 $t \geq 3$, 计算 ν_t 时, 这个算法对两个 $t \times t$ 的矩阵 U 和 V 有效, 对于 $1 \leq i \leq t$, 它们的行向量以 $U_i = (u_{i1}, \dots, u_{it})$ 和 $V_i = (v_{i1}, \dots, v_{it})$ 表示, 这些向量满足下列条件:

$$u_{i1} + au_{i2} + \cdots + a^{t-1}u_{it} \equiv 0 \pmod{m}, \quad 1 \leq i \leq t \quad (28)$$

$$U_i \cdot V_j \equiv m\delta_{ij}, \quad 1 \leq i, j \leq t \quad (29)$$

(因此, 我们以前讨论的 V_j 已被乘以 m , 以确保它们的分量是整数。) 有三个其它的辅助向量 $X = (x_1, \dots, x_t)$, $Y = (y_1, \dots, y_t)$ 以及 $Z = (z_1, \dots, z_t)$ 。在整个算法期间, r 将表示 $a^{t-1} \bmod m$, s 将表示迄今已经发现的 ν_t^2 的最小上界。

S1. [初始化] 置 $t \leftarrow 2, h \leftarrow a, h' \leftarrow m, p \leftarrow 1, p' \leftarrow 0, r \leftarrow a, s \leftarrow 1 + a^2$ (这个算法的头一步通过一个特殊的方法处理 $t = 2$ 的情况。这个方法非常像欧几里得算法, 在计算阶段将有

$$h - ap \equiv h' - ap' \equiv 0 \pmod{m} \text{ 和 } hp' - h'p = \pm m \quad (30)$$

S2. [欧几里得步骤] 置 $q \leftarrow \lfloor h'/h \rfloor, u \leftarrow h' - qh, v \leftarrow p' - qp$ 。如果 $u^2 + v^2 < s$, 置 $s \leftarrow u^2 + v^2, h' \leftarrow h, h \leftarrow u, p' \leftarrow p, p \leftarrow v$ 并重复步骤 S2。

S3. [计算 ν_2] 置 $u \leftarrow u - h, v \leftarrow v - p$; 而且如果 $u^2 + v^2 < s$ 则置 $s \leftarrow u^2 + v^2, h' \leftarrow u, p' \leftarrow v$ 。然后输出 $\sqrt{s} = \nu_2$ 。(对于二维的情况这个计算的正确性在习题 5 中证明。现在我们将建立满足(28)和(29)的 U 和 V 矩阵, 为更高维的计算做准备。)置

$$U \leftarrow \begin{pmatrix} -h & p \\ -h' & p' \end{pmatrix}, \quad V \leftarrow \pm \begin{pmatrix} p' & h' \\ -p & -h \end{pmatrix}$$

其中, 当且仅当 $p' > 0$ 时用负号。

S4. [推进 t] 如果 $t = T$, 算法结束。(否则我们要对 t 加 1, 这时 U 和 V 是满足(28)和(29)的 $t \times t$ 矩阵, 因而我们必须通过加上一个适当的新行和新列来扩大它们。)置 $t \leftarrow t + 1$ 和 $r \leftarrow (ar) \bmod m$ 。置 U_t 为 t 个元素的新行 $(-r, 0, 0, \dots, 0, 1)$, 且对于 $1 \leq i < t$ 置 $u_{it} \leftarrow 0$ 。置 V_t 成为新行 $(0, 0, 0, \dots, 0, m)$ 。最后, 对于 $1 \leq i < t$, 置 $q \leftarrow \text{round}(v_{i1}r/m), v_{it} \leftarrow v_{i1}r - qm$, 以及 $U_i \leftarrow U_i + qU_t$ 。(这里“ $\text{round}(x)$ ”表示最接近于 x 的整数, 例如 $\lfloor x + 1/2 \rfloor$ 。我们实质上置 $v_{it} \leftarrow v_{i1}r$ 并且对于 $j = t$ 立即应用变换(23), 因为数 $|v_{i1}r|$ 太大, 因此它们应立即被减小。)最后置 $s \leftarrow \min(s, U_t \cdot U_t), k \leftarrow t$ 以及 $j \leftarrow 1$ 。(在下列步骤中, j 表示变换(23)的当前行下标, k 表示最后的这样的下标,

即其中变换至少缩短 V_i 之一。

S5. [变换] 对于 $1 \leq i \leq t$, 进行下列运算: 如果 $i \neq j$ 和 $2|V_i \cdot V_j| > V_j \cdot V_j$, 置 $q \leftarrow \text{round}(V_i \cdot V_j / V_j \cdot V_j)$, $V_i \leftarrow V_i - qV_j$, $U_i \leftarrow U_i + qU_j$, $s \leftarrow \min(s, U_j \cdot U_j)$, 以及 $k \leftarrow j$ 。(当 $2|V_i \cdot V_j|$ 精确地等于 $V_j \cdot V_j$ 时, 我们省略这一变换; 习题 19 证明这个预防措施防止算法无穷地循环。)

S6. [推进 j] 如果 $j = t$, 则置 $j \leftarrow 1$; 否则置 $j \leftarrow j + 1$ 。现在如果 $j \neq k$, 则返回步骤 S5。(如果 $j = k$, 我们已经通过了无变换的 $t - 1$ 个连续的循环, 因此变换过程搁浅。)

S7. [为查找做准备] (现在使用对满足引理 A 的条件 (21) 的所有 (x_1, \dots, x_t) 的穷尽查找, 将确定出绝对极小。)置 $X \leftarrow Y \leftarrow (0, \dots, 0)$, 置 $k \leftarrow t$, 并置

$$z_j \leftarrow \lfloor \sqrt{\lfloor (V_j \cdot V_j) s / m^2 \rfloor} \rfloor, \quad \text{其中 } 1 \leq j \leq t \quad (31)$$

(对于 $1 \leq j \leq t$ 我们将考察 $|x_j| \leq z_j$ 的所有 $X = (x_1, \dots, x_t)$ 。通常 $|z_j| \leq 1$, 但 L. C. Killingbeck 在 1999 年注意到当 $m = 2^{64}$ 时, 所有乘数的大约 0.00001 会出现大的值。在穷尽查找期间, 向量 Y 将总是等于 $x_1 U_1 + \dots + x_t U_t$ 使得 $f(x_1, \dots, x_t) = Y \cdot Y$ 。由于 $f(-x_1, \dots, -x_t) = f(x_1, \dots, x_t)$, 因此我们将仅仅考察其头一个非零分量为正的那些向量。这实质上是在如下算法各步骤中的计数方法: 它把 (x_1, \dots, x_t) 当做在具有混合进制 $(2z_1 + 1, \dots, 2z_t + 1)$ 的平衡计数系统中的数字处理; 请见 4.1 节。)

S8. [推进 x_k] 如果 $x_k = z_k$, 则转到 S10。否则 x_k 加 1 并置 $Y \leftarrow Y + U_k$ 。

S9. [推进 k] 置 $k \leftarrow k + 1$ 。然后如果 $k \leq t$, 则置 $x_k \leftarrow -z_k$, $Y \leftarrow Y - 2z_k U_k$ 并重复步骤 S9。但如果 $k > t$, 则置 $s \leftarrow \min(s, Y \cdot Y)$ 。

S10. [减少 k] 置 $k \leftarrow k - 1$ 。如果 $k \geq 1$, 则返回 S8。否则输出 $\nu_t = \sqrt{s}$ (穷尽查找完成) 并返回 S4。 ■

实践中, 一般对 $T = 5$ 或 6 应用算法 S; 当 $T = 7$ 或 8 时, 它通常工作得相当好, 但当 $T \geq 9$ 时它可能非常慢, 因为穷尽查找趋于使运行时间随 3^T 而增长。(如果在许多不同的点出现极小值 ν_t , 则穷尽查找将把它们全部找出; 因此我们通常发现, 对于很大的 t , 典型地都有 $z_k = 1$ 。如上所述, 当 t 很大时, ν_t 的值一般不适合于实用目的。)

一个例子将有助于弄清算法 S, 考虑由

$$m = 10^{10}, \quad a = 3141592621, \quad c = 1, \quad X_0 = 0 \quad (32)$$

定义的线性同余序列, 步骤 S2 和 S3 中的欧几里得算法的六个循环足以证明满足

$$x_1 + 3141592621x_2 \equiv 0 \pmod{10^{10}}$$

的 $x_1^2 + x_2^2$ 的极小非零值当 $x_1 = 67654$, $x_2 = 226$ 时出现; 因此这个生成程序的二维精度是

$$\nu_2 = \sqrt{67654^2 + 226^2} \approx 67654.37748$$

对于三维,我们求使得

$$x_1 + 3141592621x_2 + 3141592621^2x_3 \equiv 0 \pmod{10^{10}} \quad (33)$$

的极小非零值 $x_1^2 + x_2^2 + x_3^2$ 。步骤 S4 建立矩阵

$$U = \begin{bmatrix} -67654 & -226 & 0 \\ -44190611 & 191 & 0 \\ 5793866 & 33 & 1 \end{bmatrix} \quad V = \begin{bmatrix} -191 & -44190611 & 2564918569 \\ -226 & 67654 & 1307181134 \\ 0 & 0 & 10000000000 \end{bmatrix}$$

当 $i=2$ 时以 $q=1$, 当 $i=3$ 时以 $q=4$, 来进行步骤 S5 的头一次迭代, 把它们改变成为

$$U = \begin{bmatrix} -21082801 & 97 & 4 \\ -44190611 & 191 & 0 \\ 5793866 & 33 & 1 \end{bmatrix} \quad V = \begin{bmatrix} -191 & -44190611 & 2564918569 \\ -35 & 44258265 & -1257737435 \\ 764 & 176762444 & -259674276 \end{bmatrix}$$

(注意头一行 U_1 在这一变换中实际上已变得更长了, 尽管最终 U 的诸行将缩短。)

步骤 S5 的以下 14 次迭代有 $(j, q_1, q_2, q_3) = (2, -2, *, 0), (3, 0, 3, *), (1, *, -10, -1), (2, -1, *, -6), (3, -1, 0, *), (1, *, 0, 2), (2, 0, *, -1), (3, 3, 4, *), (1, *, 0, 0), (2, -5, *, 0), (3, 1, 0, *), (1, *, -3, -1), (2, 0, *, 0), (3, 0, 0, *)$ 。现在变换过程搁浅了, 但矩阵的诸行已经变得相当短了:

$$U = \begin{bmatrix} -1479 & 616 & -2777 \\ -3022 & 104 & 918 \\ -227 & -983 & -130 \end{bmatrix} \quad V = \begin{bmatrix} -888874 & 601246 & -2994234 \\ -2809871 & 438109 & 1593689 \\ -854296 & -9749816 & -1707736 \end{bmatrix} \quad (34)$$

步骤 S7 中的查找极限 (z_1, z_2, z_3) 结果是 $(0, 0, 1)$, 所以 U_3 是 (33) 的最短的解, 我们有

$$\nu_3 = \sqrt{227^2 + 983^2 + 130^2} \approx 1017.21089。$$

为求这个值仅仅需要少数迭代, 尽管乍一看条件 (33) 十分难以处理。我们的计算已经证明由随机数生成程序 (32) 产生的所有点 (U_n, U_{n+1}, U_{n+2}) 位于相隔大约 0.001 单位的平行平面族上, 但不会全落在平面之间相隔超过 0.001 单位的平面族上。

步骤 S8~S10 中的穷尽查找仅仅很稀少地减少 s 的值, 一个这样的情况是由 R. Carling 和 K. Levine 在 1982 年发现的, 当 $a = 464680339$, $m = 2^{29}$ 和 $t = 5$ 时它出现; 另一个情况是当作者对本小节稍后表 1 的行 21 计算 ν_6^2 时出现的。

E. 确定各种生成程序的等级 迄今我们实际上还未给出一个准则, 来指出一个特殊的随机数生成程序是否“通过”或者“通不过”谱检验。事实上, 谱检验的成功取决于应用, 因为某些应用要求比其它应用有更高的分辨率。对于大多数应用来说, 对于 $2 \leq t \leq 6$, $\nu_t \geq 2^{30/t}$ 将是十分适当的 (尽管作者必须承认选定这个准则部分地是因为 30 可方便地为 2, 3, 5 和 6 所整除)。

对于某些目的而言, 我们将喜欢相对地独立于 m 的准则, 所以我们可以说对于

已知的 m , 一个具体的乘数相对于所有其它乘数的集合是好还是坏, 而无须考察任何其它乘数。对一个具体乘数的美好性评等级的一个合理度量似乎是由下列关系所定义的, t 维空间中椭球的体积

$$(x_1 m - x_2 a - \cdots - x_t a^{t-1})^2 + x_2^2 + \cdots + x_t^2 \leq \nu_t^2$$

因为这个体积趋于表示非零整数点 (x_1, \cdots, x_t) ——对应于(15)的解——在该椭球中的可能性有多大。因此我们建议计算这个体积, 即

$$\mu_t = \frac{\pi^{t/2} \nu_t^t}{(t/2)! m} \quad (35)$$

作为给定 m 的乘数 a 的有效性的一个标志。在这个公式中

$$\left(\frac{t}{2}\right)! = \left(\frac{t}{2}\right)\left(\frac{t}{2}-1\right)\cdots\left(\frac{1}{2}\right)\sqrt{\pi}, \quad \text{当 } t \text{ 为奇数时} \quad (36)$$

因此, 在六维或较小的维数中:

$$\mu_2 = \pi \nu_2^2 / m, \quad \mu_3 = \frac{4}{3} \pi \nu_3^3 / m, \quad \mu_4 = \frac{1}{2} \pi^2 \nu_4^4 / m$$

$$\mu_5 = \frac{8}{15} \pi^2 \nu_5^5 / m, \quad \mu_6 = \frac{1}{6} \pi^3 \nu_6^6 / m$$

我们可以说对于 $2 \leq t \leq 6$, 如果 μ_t 是 0.1 或更多, 则乘数 a 通过谱检验, 而且对于所有这些 t , 如果 $\mu_t \geq 1$, 则它“出色地通过”。 μ_t 的低值意味着我们大概已经挑上了一个非常倒霉的乘数, 因为很少的格会有如此接近原点的整数点。反过来, μ_t 的高值意味着我们对于给定的 m , 已经找到异常好的乘数; 但它并不表明随机数必然非常好, 因为 m 可能太小。只有 ν_t 的值才真正地指出随机性的程度。

表 1 表明在典型的序列中出现的值的类型。这个表的每行考虑一个具体的生成程序, 并且列出 ν_t^2 , μ_t 以及“精确度的二进位数” $\lg \nu_t$ 。行 1 到行 4 示出 3.3.1 小节中图 2 和图 5 中提到的生成程序。行 1 和 2 的生成程序由于乘数太小而质量不佳; 像图 8 那样的图形当 a 很小时有几乎垂直的“条纹”。行 3 上的糟糕的生成程序有好的 μ_2 , 但 μ_3 和 μ_4 却很差; 和几乎所有效能为 2 的生成程序一样, 它有 $\nu_3 = \sqrt{6}$ 和 $\nu_4 = 2$ (见习题 3)。行 4 给出一个“随机的”乘数; 这个生成程序已经令人满意地通过大量的关于随机性的经验检验, 但它没有特别高的 μ_2, \cdots, μ_6 值。事实上, μ_5 的值通不过我们的准则。

行 5 给出图 8 的生成程序。当考虑 μ_2 直到 μ_6 时, 它出色地通过谱检验, 当然 m 太小了, 因此这些数几乎不能叫做随机数; ν_t 的值低得不像话。

行 6 是上边(32)中讨论的生成程序; 行 7 是类似的例子, 并且有一个异常低的 μ_3 值。行 8 给出对于同一个模数 m 的一个非随机的乘数; 所有它的部分商都是 1, 2 或 3。这些乘数是由 I. Borosh 和 H. Niederreiter 建议的, 因为戴德金和数可能特别小, 而且因为它们二维序列检验中产生最好的结果(参考 3.3.3 小节和习题 30)。行 8 中的具体例子仅有一个“3”作为一个部分商; 没有这样的乘数, 它模 20 同余于 1, 而且它关于 10^{10} 的部分商仅由一些 1 和 2 组成。行 9 的生成程序给出恶意预谋

表1 谱检验的结果样例
($\epsilon = \frac{1}{10}$)

行	a	m	v_2^1	v_3^1	v_4^1	v_5^1	v_6^1	$\lg v_2$	$\lg v_3$	$\lg v_4$	$\lg v_5$	$\lg v_6$	μ_2	μ_3	μ_4	μ_5	μ_6	行
1	23	$10^8 + 1$	530	530	530	530	447	4.5	4.5	4.5	4.5	4.4	2_i^5	5_i^4	0.01	0.34	4.62	1
2	$2^7 + 1$	2^{35}	16642	16642	16642	16642	252	7.0	7.0	7.0	7.0	4.0	2_i^6	3_i^4	0.04	4.66	2_i^3	2
3	$2^{18} + 1$	2^{35}	34359738368	6	4	4	4	17.5	1.3	1.0	1.0	1.0	$1.0 \cdot 3_{i-1}^4$	2_i^9	2_i^9	5_i^8	5_i^8	3
4	3141592653	2^{35}	2997222016	1026050	27822	1118	1118	15.7	10.0	7.4	5.1	5.1	$5.1 \cdot 10_{i-1}^7$	0.13	0.11	0.01	0.21	4
5	137	2^{56}	274	30	14	6	4	4.0	2.5	1.9	1.3	1.3	$1.0 \cdot 3_{i-1}^6$	2.69	3.78	1.81	1.29	5
6	3141592621	10^{10}	4577114792	1034718	62454	1776	542	16.0	10.0	8.0	5.4	4.5	$4.5 \cdot 1_{i-1}^4$	0.44	1.92	0.07	0.08	6
7	3141592221	10^{10}	4293881050	276266	97450	3366	2382	16.0	9.0	8.3	5.9	5.6	1.35	0.06	4.69	0.35	6.98	7
8	4219755981	10^{10}	10721093248	2595578	49362	5868	820	16.7	10.7	7.8	6.3	4.8	3.37	1.75	1.20	1.39	0.28	8
9	4160984121	10^{10}	9183801602	4615650	16686	6840	1344	16.5	11.1	7.0	6.4	5.2	2.89	4.15	0.14	2.04	1.25	9
10	$2^{24} + 2^{13} + 5$	2^{35}	8364058	8364058	71476	16712	1496	11.5	11.5	7.2	7.0	5.3	8_i^3	2.95	0.07	5.53	0.50	10
11	5^{13}	2^4	33161885770	2925242	113374	13070	2256	17.5	10.7	8.4	6.8	5.6	3.03	0.61	1.85	2.99	1.73	11
12	$2^{16} + 3$	2^{29}	536936458	118	116	116	116	14.5	3.4	3.4	3.4	3.4	$3.4 \cdot 3_{i-1}^4$	1.5	1.4	1.1	0.02	12
13	1812433253	2^{32}	4326934538	1462856	15082	4866	906	16.0	10.2	6.9	6.1	4.9	3.16	1.73	0.26	2.02	0.89	13
14	1566083041	2^{32}	4659748970	2079590	44902	4652	662	16.1	10.5	7.7	6.1	4.7	3.41	2.92	2.32	1.81	0.35	14
15	69069	2^{32}	4243209856	2072544	52804	6990	242	16.0	10.5	7.8	6.4	4.0	3.10	2.91	3.20	5.01	0.02	15
16	1664525	2^{17}	4938916874	2322494	63712	4092	1038	16.1	10.6	8.0	6.0	5.0	3.61	3.45	4.66	1.31	1.35	16
17	314159269	$2^{31} - 1$	1432232969	899290	36985	3427	1144	15.2	9.9	7.6	5.9	5.1	2.10	1.66	3.14	1.69	3.60	17
18	62089911	$2^{31} - 1$	1977289717	1662317	48191	6101	1462	15.4	10.3	7.8	6.3	5.3	2.89	4.18	5.34	7.13	7.52	18
19	16807	$2^{31} - 1$	282475250	408197	21682	4439	895	14.0	9.3	7.2	6.1	4.9	0.41	0.51	1.08	3.22	1.73	19
20	48271	$2^{31} - 1$	1990735345	1433881	47418	4404	1402	15.4	10.2	7.8	6.1	5.2	2.91	3.35	5.17	3.15	6.63	20
21	40692	$2^{31} \cdot 249$	1655838865	1403422	42475	6507	1438	15.3	10.2	7.7	6.3	5.2	2.42	3.24	4.15	8.37	7.16	21
22	44485709377909	2^{46}	5.6×10^{13}	1180915002	1882426	279928	26230	22.8	15.1	10.4	9.0	7.3	2.48	2.42	0.25	3.10	1.33	22
23	31167285	2^{48}	3.2×10^{14}	4111841446	17341510	306326	59278	24.1	16.0	12.0	9.1	7.9	3.60	3.92	5.27	0.97	3.82	23
24	参见(38)	2^{48}	2.4×10^{18}	4.7×10^{11}	1.9×10^9	3194548	1611610	30.5	19.4	15.4	10.8	10.3	1.65	0.29	3.88	0.02	4.69	24
25	参见(39)	2^{64}	$(2^{31} - 1)^2$	1.4×10^{12}	643578623	12930027	837632	31.0	20.2	14.6	11.8	9.8	3.14	1.49	0.44	0.69	0.66	25
26	参见正文	2^{64}	8.8×10^{18}	6.4×10^{12}	4.1×10^9	45662836	1846368	31.5	21.3	16.0	12.7	10.4	1.50	3.68	4.52	4.02	1.76	26
27	参见正文	$\approx 2^{78}$	$2^{62} + 1$	4281084902	2.2×10^9	1.8×10^9	1862407	31.0	16.0	15.5	15.4	10.4	5_i^5	4_i^9	8_i^5	2.56	2.56	27
28	$2^{24} \cdot 389$	$\approx 2^{576}$	1.8×10^{173}	3.5×10^{115}	4.4×10^{86}	2×10^{69}	5×10^{57}	288.192	144.115	95.9	2.27	3.46	3.92	2.49	2.98	2.98	2.98	28
29	$(2^{32} \cdot 5) \cdot 400$	$\approx 2^{1376}$	1.6×10^{414}	8.6×10^{275}	1×10^{207}	2×10^{165}	8×10^{137}	688.458	344.275	229.3	10.2	2.04	2.85	1.15	1.33	1.33	1.33	29

由(40)得到的上限: 3.63 5.92 9.87 14.89 23.87

L.C. Killingbeck 完成了 $m = 2^{32}$ 时所有 $a \equiv 1 \pmod{4}$ 的乘数的搜索。他发现乘数 $n = 2650845021$ 有 $v_2^1 = 4938969760$, $v_3^1 = 2646962$, $v_4^1 = 68342$, $v_5^1 = 8778$ 和 $v_6^1 = 1506$, 所以它超过了该模列出的所有乘数。的确, 它的引入注目的 μ 值 (3.61, 4.20, 5.37, 8.85, 4.11) 超过了整个表中任何模的所有 μ_2, μ_3, μ_4 和 μ_5 值。

选定的另一个乘数,这是 A. G. Waterman 建议的,它保证相当高的 μ_2 值(见习题 11)。行 10 是有趣的,因为尽管 μ_2 非常低,但 μ_3 却很高(参见习题 8)。

表 1 的行 11 是对过去的好时光的提醒——自 O. Taussky 于 20 世纪 50 年代初提议以来,它曾经被广泛地使用过。但在 60 年代末, 2^{35} 是一个合适的模数的那些计算机开始淡出,而且随着 32 位算术运算的机器开始激增,到 80 年代它们几乎完全绝迹了。这种朝着相对小的字长的改变要求相对更大的小心。天啊,行 12 的生成程序实际上已经被世界上大多数科学计算中心这样的机器使用了 10 年以上;它奇特的名字 RANDU 足以使许多计算机科学家大跌眼镜和大倒胃口! 实际的生成程序是由

$$X_0 \text{ 奇}, \quad X_{n+1} = (65539X_n) \bmod 2^{31} \quad (37)$$

定义的,而习题 20 指出, 2^{29} 是对于谱检验的适当模数。由于 $9X_n - 6X_{n+1} + X_{n+2} \equiv 0 \pmod{2^{31}}$, 因此这个生成程序不能通过随机性的大多数三维准则,因此它本不应被使用。几乎任何同余于 5(模 8)的乘数都会要好些。(R. W. Gosper 注意到关于 RANDU 的一个奇怪的事实,即 $\nu_4 = \nu_5 = \nu_6 = \nu_7 = \nu_8 = \nu_9 = \sqrt{116}$, 因此 μ_9 是人瞩目的 11.98。)行 13 和行 14 是对于模 2^{32} 的 Borosh-Niederreiter 和 Waterman 乘数。行 16 和 23 是 M. Lavaux 和 F. Janssens 在计算机查找中发现的,他们的目的是查找具有很高的 μ_2 值且谱检验很好的乘数;行 22 在 Cray X-MP 库中与 $c=0$ 和 $m=2^{48}$ 一起作为乘数使用;行 26(它的杰出的乘数 6364136223846793005 太大了,在一栏中装不下)是由 C. E. Haynes 给出的。行 15 是由 George Marsaglia 在进行了二维到五维的接近立方的格的计算机查找之后,作为“所有乘数中最好者的候选者”而提出的,部分原因是它便于记忆。[*Applications of Number Theory to Numerical Analysis*, S. K. Zaremba 编(New York: Academic Press, 1972), 275]。

行 17 使用一个随机原根,对素数 $2^{31}-1$ 的模作为乘数。行 18 示出对于 $2^{31}-1$ 谱上最好的原根,它是由 G. S. Fishman 和 L. R. Moore III 通过穷尽查找发现的 [*SIAM J. Sci. Stat. Comput.* 7 (1986), 24~45]。在行 19 中的适当的但不太突出的乘数 $16807=7^5$, 在由 Lewis, Goodman 和 Miller 于 *IBM Systems J.* 8 (1969), 136~146 上提出之后,实际上最经常地用于此模数;自 1971 年以来它已经成为流行的 IMSL 子程序库中主要的生成程序之一。 $a=16807$ 之所以被继续使用是因为 a^2 小于模数 m , 因此 $ax \bmod m$ 可以使用习题 3.2.1.1-9 的技术以相当高的效率在高级语言中实现。然而这样小的乘数有已知的一些缺点。S. K. Park 和 K. W. Miller 注意到同样的实现技术也可应用到大于 \sqrt{m} 的某些乘数上。所以他们要求 G. S. Fishman 在这更宽的表中找到最好的“可有效移植的”乘数;结果出现于行 20 上 [*CACM* 31(1988), 1192~1201]。行 21 示出由 P. L'Ecuyer 给出的另一个好的乘数 [*CACM* 31(1988), 742~749, 774]; 它使用稍微更小的素模数。

当如同在等式 3.2.2-(15)中建议的那样,通过减法把行 20 和行 21 的生成程序组合在一起,使得被生成的数 $\langle Z_n \rangle$ 满足

$$\begin{aligned} X_{n+1} &= 48271 X_n \bmod (2^{31} - 1), \quad Y_{n+1} = 40692 Y_n \bmod (2^{31} - 249) \\ Z_n &= (X_n - Y_n) \bmod (2^{31} - 1) \end{aligned} \quad (38)$$

时,习题 32 证明,用对于 $m = (2^{31} - 1)(2^{31} - 249)$ 和 $a = 1431853894371298687$ 的谱检验来对 $\langle Z_n \rangle$ 评级是合理的。(a 的这个值满足 $a \bmod (2^{31} - 1) = 48271$ 和 $a \bmod (2^{31} - 249) = 40692$)。结果见于行 24 上。对于低的 μ_5 的值我们不必太担心,因为 $\nu_5 > 1000$ 。生成程序(38)有长度 $(2^{31} - 2)(2^{31} - 250)/62 \approx 7 \times 10^{16}$ 的周期。

表的行 25 表示序列

$$X_n = (271828183 X_{n-1} - 314159269 X_{n-2}) \bmod (2^{31} - 1) \quad (39)$$

可以证明它有 $(2^{31} - 1)^2 - 1$ 的周期长度;它已通过习题 24 的广义谱检验做了分析。

表 1 的最后三行是基于带进位加法和带借位减法得出的;它们模拟有极其大的模数的线性同余序列(见习题 3.2.1.1-14)。行 27 是对于生成程序

$$\begin{aligned} X_n &= (X_{n-1} + 65430 X_{n-2} + C_n) \bmod 2^{31} \\ C_{n+1} &= \lfloor (X_{n-1} + 65430 X_{n-2} + C_n) / 2^{31} \rfloor \end{aligned}$$

的,它对应于 $\chi_{n+1} = (65430 \cdot 2^{31} + 1) \chi_n \bmod (65430 \cdot 2^{62} + 2^{31} - 1)$;表中的这些数称做“超值”

$$\chi_n = (65430 \cdot 2^{31} + 1) X_{n-1} + 65430 X_{n-2} + C_n$$

而不是真正被计算和被使用作随机数的值 X_n 。行 28 表示更典型的带借位减法的生成程序

$$X_n = (X_{n-10} - X_{n-24} - C_n) \bmod 2^{24}, \quad C_{n+1} = [X_{n-10} < X_{n-24} + C_n]$$

但通过生成序列的 389 个元素并且仅仅使用开头(或最后的)24 个做了修改。这个生成程序,在通过了以前的生成程序没有通过的许多严厉的检验之后,被 Martin Lüscher 所推荐,并取名为 RANLUX [Computer Physics Communications 79 (1994), 100~110]。一个类似的序列

$$X_n = (X_{n-22} - X_{n-43} - C_n) \bmod (2^{32} - 5), \quad C_{n+1} = [X_{n-22} < X_{n-43} + C_n]$$

出现在行 29 上,而且在生成 400 个之后只使用 43 个元素;在习题 3.2.1.2-22 的答案中对这个序列做了讨论。在两种情况下表的各项都称为对多精度数 χ_n ,而不是对个别“数字” X_n 的谱检验,但是高的 μ 值指出,在选择 24 个或 43 个之前生成 389 个或 400 个数的过程是去掉由于生成方案的极端简单性所引起偏倚的杰出方法。

对于 μ_t 的理论上限在表 1 的紧底下示出,对于任何的 m 它们都不能被超越。已知每个单位体积中有 m 个点的每个格有

$$\nu_t \leq \gamma_t^{1/2} m^{1/t} \quad (40)$$

其中 γ_t 对于 $t=2, \dots, 8$, 分别取

$$(4/3)^{1/2}, 2^{1/3}, 2^{1/2}, 2^{3/5}, (64/3)^{1/6}, 4^{3/7}, 2 \quad (41)$$

的值。[见习题 9 和 J.W.S.Cassels, *Introduction to the Geometry of Numbers* (Berlin: Springer, 1959), 332; J.H.Conway 和 N.J.A.Sloane, *Sphere Packings, Lattices and Groups* (New York: Springer, 1988), 20。]这些界对于由任何实坐标向量生

成的格均成立。例如,对于 $t=2$ 的最优格是六面体的,而它是由形成一个等腰三角形的两边且长为 $2/\sqrt{3}m$ 的向量生成的。在三维中最优的格是由可被转动成形式 $(v, v, -v), (v, -v, v), (-v, v, v)$ 的向量 V_1, V_2, V_3 生成的,其中 $v = 1/\sqrt[3]{4m}$ 。

***F. 同序列检验的关系** 在自 20 世纪 70 年代以来发表的一系列重要文章中,Harald Niederreiter 已经说明如何借助指数和来分析 t 维向量(1)的分布。他的理论的一个主要推论是若干维中的序列检验将为通过谱检验的任何生成程序所通过,甚至当我们仅考虑周期的充分大的一部分而不是全部周期时也是如此。我们现在将转而简单地研究一下,在周期长度为 m 的线性同余序列 (X_0, a, c, m) 的情况下,他的有趣的方法。

我们需要的头一个思想是在 t 维中的差异的思想,我们将它定义为落入到一个超矩形区域的 t 维向量 $(x_n, x_{n+1}, \dots, x_{n+t-1})$ 的预期数和实际数之间的差别(对所有这样的区域取极大)。精确地说,设 $\langle x_n \rangle$ 是在范围 $0 \leq x_n < m$ 中的一个整数序列,我们定义

$$D_N^{(t)} = \max_R \left| \frac{0 \leq n < N \text{ 时 } R \text{ 中的 } (x_n, \dots, x_{n+t-1}) \text{ 的个数}}{N} - \frac{R \text{ 的体积}}{m^t} \right| \quad (42)$$

其中 R 取遍形式为

$$R = \{(y_1, \dots, y_t) \mid \alpha_1 \leq y_1 < \beta_1, \dots, \alpha_t \leq y_t < \beta_t\} \quad (43)$$

的所有点的集合,这里 α_j 和 β_j 是对于 $1 \leq j \leq t$, 在 $0 \leq \alpha_j < \beta_j \leq m$ 的范围中的所有整数。 R 的体积显然是 $(\beta_1 - \alpha_1) \cdots (\beta_t - \alpha_t)$ 。为得到差异 $D_N^{(t)}$, 我们想像考察所有这些集合 R 并且从中求出对点集 (x_n, \dots, x_{n+t-1}) 来说超量或缺额为最大的一个集合。

通过使用指数和可以找到差异的一个上界,令 $\omega = e^{2\pi i/m}$ 是一个 m 次本原单位根。如果 (x_1, \dots, x_t) 和 (y_1, \dots, y_t) 是两个向量,它们的所有分量都在 $0 \leq x_j, y_j < m$ 范围中,我们有

$$\sum_{0 \leq u_1, \dots, u_t < m} \omega^{(x_1 - y_1)u_1 + \dots + (x_t - y_t)u_t} = \begin{cases} m^t & \text{若 } (x_1, \dots, x_t) = (y_1, \dots, y_t) \\ 0 & \text{若 } (x_1, \dots, x_t) \neq (y_1, \dots, y_t) \end{cases}$$

因此当 R 由(43)定义时,对于 $0 \leq n < N$, R 中向量 (x_n, \dots, x_{n+t-1}) 的个数可表达为

$$\frac{1}{m^t} \sum_{0 \leq n < N} \sum_{0 \leq u_1, \dots, u_t < m} \omega^{x_n u_1 + \dots + x_{n+t-1} u_t} \sum_{\alpha_1 \leq y_1 < \beta_1} \cdots \sum_{\alpha_t \leq y_t < \beta_t} \omega^{-(y_1 u_1 + \dots + y_t u_t)}$$

在这个和中当 $u_1 = \dots = u_t = 0$ 时,我们得到 N/m^t 乘以 R 的体积;因此我们可以把 $D_N^{(t)}$ 表达为

$$\left| \frac{1}{Nm^t} \sum_{0 \leq n < N} \sum_{\substack{0 \leq u_1, \dots, u_t < m \\ (u_1, \dots, u_t) \neq (0, \dots, 0)}} \omega^{x_n u_1 + \dots + x_{n+t-1} u_t} \sum_{\alpha_1 \leq y_1 < \beta_1} \cdots \sum_{\alpha_t \leq y_t < \beta_t} \omega^{-(y_1 u_1 + \dots + y_t u_t)} \right|$$

在 R 上的极大值。由于复数满足 $|\omega + z| \leq |\omega| + |z|$ 和 $|\omega z| = |\omega| |z|$, 由此得出

$$\begin{aligned}
D_N^{(t)} &\leq \max_R \frac{1}{m^t} \sum_{\substack{0 \leq u_1, \dots, u_t < m \\ (u_1, \dots, u_t) \neq (0, \dots, 0)}} \left| \sum_{\alpha_1 \leq \beta_1} \cdots \sum_{\alpha_t \leq \beta_t} \omega^{-(y_1 u_1 + \dots + y_t u_t)} \right| g(u_1, \dots, u_t) \leq \\
&\frac{1}{m^t} \sum_{\substack{0 \leq u_1, \dots, u_t < m \\ (u_1, \dots, u_t) \neq (0, \dots, 0)}} \max_R \left| \sum_{\alpha_1 \leq \beta_1} \cdots \sum_{\alpha_t \leq \beta_t} \omega^{-(y_1 u_1 + \dots + y_t u_t)} \right| g(u_1, \dots, u_t) = \\
&\sum_{\substack{0 \leq u_1, \dots, u_t < m \\ (u_1, \dots, u_t) \neq (0, \dots, 0)}} f(u_1, \dots, u_t) g(u_1, \dots, u_t) \quad (44)
\end{aligned}$$

其中

$$\begin{aligned}
g(u_1, \dots, u_t) &= \left| \frac{1}{N} \sum_{0 \leq n < N} \omega^{x_n u_1 + \dots + x_{n+t-1} u_t} \right| \\
f(u_1, \dots, u_t) &= \max_R \frac{1}{m^t} \left| \sum_{\alpha_1 \leq \beta_1} \cdots \sum_{\alpha_t \leq \beta_t} \omega^{-(y_1 u_1 + \dots + y_t u_t)} \right| = \\
&\max_R \left| \frac{1}{m} \sum_{\alpha_1 \leq \beta_1} \omega^{-u_1 y_1} \right| \cdots \left| \frac{1}{m} \sum_{\alpha_t \leq \beta_t} \omega^{-u_t y_t} \right|
\end{aligned}$$

为了得到关于 $D_N^{(t)}$ 的好上界, f 和 g 两者都可进一步简化。当 $u \neq 0$ 时我们有

$$\left| \frac{1}{m} \sum_{\alpha \leq \beta} \omega^{-u y} \right| = \left| \frac{1}{m} \frac{\omega^{-\beta u} - \omega^{\alpha u}}{\omega^{-u} - 1} \right| \leq \frac{2}{m |\omega^{-u} - 1|} = \frac{1}{m \sin(\pi u/m)}$$

而且当 $u = 0$ 时这个和小于等于 1; 因此

$$f(u_1, \dots, u_t) \leq r(u_1, \dots, u_t) \quad (45)$$

其中

$$r(u_1, \dots, u_t) = \prod_{\substack{1 \leq k \leq t \\ u_k \neq 0}} \frac{1}{m \sin(\pi u_k/m)} \quad (46)$$

而且当 $\langle x_n \rangle$ 是由一个线性同余序列模 m 生成时, 我们有

$$\begin{aligned}
x_n u_1 + \dots + x_{n+t-1} u_t &= x_n u_1 + (a x_n + c) u_2 + \dots + (a^{t-1} x_n + c(a^{t-2} + \dots + 1)) u_t = \\
&= (u_1 + a u_2 + \dots + a^{t-1} u_t) x_n + h(u_1, \dots, u_t)
\end{aligned}$$

其中 $h(u_1, \dots, u_t)$ 与 n 无关; 因此

$$g(u_1, \dots, u_t) = \left| \frac{1}{N} \sum_{0 \leq n < N} \omega^{q(u_1, \dots, u_t) x_n} \right| \quad (47)$$

其中

$$q(u_1, \dots, u_t) = u_1 + a u_2 + \dots + a^{t-1} u_t \quad (48)$$

于是这里引出了和谱检验的关系: 我们将证明, 和数 $g(u_1, \dots, u_t)$ 是相当小的, 除非 $q(u_1, \dots, u_t) \equiv 0 \pmod{m}$; 换句话说, 对 (44) 的贡献主要来自于 (15) 的解。其次, 习题 27 表明, 当 (u_1, \dots, u_t) 是 (15) 的“很大的”解时, $r(u_1, \dots, u_t)$ 是相当小的。因此, 当 (15) 仅有“很大的”解时, 即当通过谱检验时, 差异 $D_N^{(t)}$ 将是相当小的。我们剩下的任务是通过进行仔细的计算来对这些定性论述进行定量。

首先让我们考虑 $g(u_1, \dots, u_t)$ 的大小。当 $N = m$ 时, 即 (47) 是对整个周期求和时, 除了当 (u_1, \dots, u_t) 满足 (15) 外, 我们有 $g(u_1, \dots, u_t) = 0$, 所以在这种情况下

通过遍取(15)的所有非 0 解对 $r(u_1, \dots, u_t)$ 求和, 就得到差异的一个上界。但我们也考虑当 N 小于 m 和 $q(u_1, \dots, u_t)$ 不是 m 的一个倍数时, 在像(47)这样的和一个数中会发生什么情况, 我们有

$$\begin{aligned} \frac{1}{N} \sum_{0 \leq n < N} \omega^{x_n} &= \frac{1}{N} \sum_{0 \leq n < N} \frac{1}{m} \sum_{0 \leq k < m} \omega^{-nk} \sum_{0 \leq j < m} \omega^{x_j + jk} = \\ &= \frac{1}{N} \sum_{0 \leq k < m} \left(\frac{1}{m} \sum_{0 \leq n < N} \omega^{-nk} \right) S_{k0} \end{aligned} \quad (49)$$

其中

$$S_{kl} = \sum_{0 \leq j < m} \omega^{x_{j+l} + jk} \quad (50)$$

现在 $S_{kl} = \omega^{-lk} S_{k0}$, 所以对所有的 l , $|S_{kl}| = |S_{k0}|$, 而且我们能通过进一步的指数对称性来计算这个共同的值:

$$\begin{aligned} |S_{k0}|^2 &= \frac{1}{m} \sum_{0 \leq l < m} |S_{kl}|^2 = \frac{1}{m} \sum_{0 \leq l < m} \sum_{0 \leq j < m} \omega^{x_{j+l} + jk} \sum_{0 \leq i < m} \omega^{-x_{i+l} - ik} = \\ &= \frac{1}{m} \sum_{0 \leq j, i < m} \omega^{(j-i)k} \sum_{0 \leq l < m} \omega^{x_{j+l} - x_{i+l}} = \\ &= \frac{1}{m} \sum_{0 \leq j < m} \sum_{i \leq j < m+i} \omega^{(j-i)k} \sum_{0 \leq l < m} \omega^{(a^{j-i}-1)x_{i+l} + (a^{j-i}-1)c/(a-1)} \end{aligned}$$

设 s 是使得 $a^s \equiv 1 \pmod{m}$ 的极小值, 且设

$$s' = (a^s - 1)c/(a - 1) \pmod{m}$$

则 s 是 m 的一个因子(请见引理 3.2.1.2P), 且 $x_{n+js} \equiv x_n + js' \pmod{m}$ 。除非 $j-i$ 是 s 的一个倍数, 否则对 l 的求和等于 0, 所以我们求得

$$|S_{k0}|^2 = m \sum_{0 \leq j < m/s} \omega^{jsk + js'}$$

我们有 $s' = q's$, 其中 q' 与 m 互素(参考习题 3.2.1.2-21), 所以结果是

$$|S_{k0}| = \begin{cases} 0 & \text{若 } k + q' \not\equiv 0 \pmod{m/s} \\ m/\sqrt{s} & \text{若 } k + q' \equiv 0 \pmod{m/s} \end{cases} \quad (51)$$

把这信息放回(49)中, 并回想起(45)的推导, 即得

$$\left| \frac{1}{N} \sum_{0 \leq n < N} \omega^{x_n} \right| \leq \frac{m}{N\sqrt{s}} \sum_k r(k) \quad (52)$$

其中求和是对使得 $k + q' \equiv 0 \pmod{m/s}$ 的 $0 < k < m$ 进行的。现在习题 25 可用来估计剩下的和数, 我们求得

$$\left| \frac{1}{N} \sum_{0 \leq n < N} \omega^{x_n} \right| \leq \frac{2}{\pi} \frac{\sqrt{s}}{N} \ln s + O\left(\frac{m}{N\sqrt{s}}\right) \quad (53)$$

同样的界可用来估计对任意 $q \not\equiv 0 \pmod{m}$ 的 $|N^{-1} \sum_{0 \leq n < N} \omega^{qx_n}|$, 因为其效果只是在这个推导中以 m 的一个因子代替 m 。事实上, 当 q 和 m 有公因子时, 上界甚至更小, 因为 s 和 m/\sqrt{s} 一般地变得更小(见习题 26)。

我们现在已经证明如果 N 足够大且如果 (u_1, \dots, u_t) 不满足谱检验同余(15), 则差异的上限(44)的 $g(u_1, \dots, u_t)$ 部分很小。习题 27 证明, 当对所有满足(15)的非零向量 (u_1, \dots, u_t) 求和时, 假定所有这样的向量都离 $(0, \dots, 0)$ 很远, 则我们的上界的 $f(u_1, \dots, u_t)$ 部分很小。把这些结果归并在一起, 就得到下面的 Niederreiter 的定理:

定理 N 设 $\langle X_n \rangle$ 是周期长度为 m 的一个线性同余序列 (X_0, a, c, m) , 并设 s 是使得 $a^s \equiv 1 \pmod{m}$ 的最小正整数, 则如同在(42)中所定义的那样, 对应于 $\langle X_n \rangle$ 的头 N 个值的 t 维差异 $D_N^{(t)}$, 满足

$$D_N^{(t)} = O\left(\frac{\sqrt{s} \log s (\log m)^t}{N}\right) + O\left(\frac{m (\log m)^t}{N \sqrt{s}}\right) + O((\log m)^t r_{\max}) \quad (54)$$

$$D_m^{(t)} = O((\log m)^t r_{\max}) \quad (55)$$

这里 r_{\max} 是在(46)中定义的量 $r(u_1, \dots, u_t)$ 的极大值, 是对满足(15)的所有非0整向量 (u_1, \dots, u_t) 得到的。

证明 (54)中的头两个 O 项来自(44)中不满足(15)的向量 (u_1, \dots, u_t) , 因为习题 25 证明对于所有 (u_1, \dots, u_t) 求和之 $f(u_1, \dots, u_t)$ 是 $O(((2/\pi) \ln m)^t)$, 而且习题 26 给出每个 $g(u_1, \dots, u_t)$ 的界。(这些项在(55)中见不到, 因为在该情况下 $g(u_1, \dots, u_t) = 0$ 。) (54)和(55)中剩下的 O 项来自满足(15)的非零向量 (u_1, \dots, u_t) , 并使用习题 27 中导出的界。(通过仔细地考察这个证明, 我们可以把这些公式中的每个 O 代之以 t 的一个显函数。) ■

式(55)同整个周期上的 t 维序列检验有关, 而式(54)给出了当 N 小于 m 且不是太小时, 关于头 N 个生成值的分布的有用信息。注意, (54)仅当 s 充分大时才保证低差异, 否则 m/\sqrt{s} 项将占主导地位。如果 $m = p_1^{e_1} \cdots p_r^{e_r}$ 和 $\gcd(a-1, m) = p_1^{f_1} \cdots p_r^{f_r}$, 则由引理 3.2.1.2P, s 等于 $p_1^{e_1 - f_1} \cdots p_r^{e_r - f_r}$; 于是, s 的极大值对应于高效能。在 $m = 2^e$ 和 $a \equiv 5 \pmod{8}$ 的普通情况下, 我们有 $s = \frac{1}{4} m$, 所以 $D_N^{(t)}$ 是 $O(\sqrt{m} (\log m)^{t+1}/N) + O((\log m)^t r_{\max})$ 。不难证明

$$r_{\max} \leq \frac{1}{\sqrt{8} \nu_t} \quad (56)$$

(见习题 29)。因此等式(54)特别说明, 如果谱检验能够通过且如果 N 稍大于 $\sqrt{m} (\log m)^{t+1}$, 则在 t 维中差异将是低的。

在某种意义上, 定理 N 几乎太强了, 因为习题 30 的结果表明, 像表 1 的行 8 和行 13 那样的线性同余序列在二维中有阶为 $(\log m)^2/m$ 的差异, 这种差异是极小的, 尽管存在一些面积约为 $1/\sqrt{m}$ 的平行四边形区域不包含点 (U_n, U_{n+1}) 。当把点转动时, 差异可以如此急剧地改变, 这一事实警告我们, 序列检验作为对随机性的

一种度量,可能不像转动不变量谱检验那样有意义。

G. 历史评述 1959年,在推导用蒙特卡罗方法计算 t 维积分中误差的上界时, N.M. Korobov 设计了对一个线性同余序列的乘数评级的方法。他的稍微复杂的公式同谱检验有关,因为它受(15)的“小”解的强烈影响;但不完全相同。Korobov 的检验是 Kuipers 和 Niederreiter 在 *Uniform Distribution of Sequences* (New York: Wiley, 1974, § 2.5) 中综述的大量文献的主题。

谱检验首先是由 R.R. Coveyou 和 R.D. MacPherson [JACM 14 (1967), 100 ~ 119] 系统阐述的,他们以间接有趣的方式介绍了它。他们不是利用逐点的栅格结构,而是把随机数生成程序看做是 t 维“波”的源。他们当初的论述中使 $x_1 + \cdots + a^{t-1}x_t \equiv 0 \pmod{m}$ 的数 $\sqrt{x_1^2 + \cdots + x_t^2}$ 是波的“频率”,或者是由随机数生成程序定义的“谱”中的点,而且低频率的波对随机性是最有损害的;谱检验的名称由此而来。Coveyou 和 MacPherson 介绍了实施他们的检验的类似于算法 S 的一个过程,其基础是引理 A 的原理。然而最初的方法(使用矩阵 UU^T 和 VV^T 而不是 U 和 V) 处理的是非常大的数;直接用 U 和 V 的思想是由 F. Janssens 和 U. Dieter 独立提出的。[见 *Math. Comp.* 29 (1975), 827 ~ 833。]

若干其他作者指出,谱检验可以以具体得多的术语加以理解;通过引入对应于线性同余序列的格栅和格结构的研究,关于随机性的基本极限在图形上就变得清楚了,见 G. Marsaglia, *Proc. Nat. Acad. Sci.* 61 (1968), 25 ~ 28; W.W. Wood, *J. Chem. Phys.* 48 (1968), 427; R.R. Coveyou, *Studies in Applied Math.* 3 (Philadelphia: SIAM, 1969), 70 ~ 111; W.A. Beyer, R.B. Roof 及 D. Williamson, *Math. Comp.* 25 (1971), 345 ~ 360; G. Marsaglia 和 W.A. Beyer, *Applications of Number Theory to Numerical Analysis*, S.K. Zaremba 编 (New York: Academic Press, 1972), 249 ~ 285, 361 ~ 370。

通过使用指数估计, R.G. Stoneham 证明,当 a 是模素数 p 的一个原根时,序列 $a^k X_0 \pmod{p}$ 的 $p^{1/2+}$ 或更多的元素有渐近的小的分布 [Acta Arithmetica 22 (1973), 371 ~ 389]。如同在上面所说明的那样,这一工作为 Harald Niederreiter 的一些论文所推广 [Math. Comp. 28 (1974), 1117 ~ 1132; 30 (1976), 571 ~ 597; Advances in Math. 26 (1977), 99 ~ 181; Bull. Amer. Math. Soc. 84 (1978), 957 ~ 1041]。也可参见 Niederreiter 的著作 *Random Number Generation and Quasi-Monte Carlo Methods* (Philadelphia: SIAM, 1992)。

习 题

1. [M10] 在一维中谱检验归结成什么? (换言之,当 $t=1$ 时发生什么情况?)

2. [HM20] 设 V_1, \dots, V_t 是 t 维空间中的线性无关向量, 设 L_0 是由(10)定义的点的格, 并设 U_1, \dots, U_t 由(19)定义。证明对于包括 L_0 的所有平行超平面族, $t-1$ 维超平面间的极大距离是 $1/\min\{f(x_1, \dots, x_t)^{1/2} \mid (x_1, \dots, x_t) \neq (0, \dots, 0)\}$, 其中 f 在(17)中定义。

3. [M24] 对于效能为 2 和周期是 m 的所有线性同余生成程序确定 ν_3 和 ν_4 。

► 4. [M23] 设 $u_{11}, u_{12}, u_{21}, u_{22}$ 是使得 $u_{11} + au_{12} \equiv u_{21} + au_{22} \equiv 0 \pmod{m}$ 及 $u_{11}u_{22} - u_{21}u_{12} \equiv m$ 的一个 2×2 整数矩阵的元素。

a) 证明同余 $y_1 + ay_2 \equiv 0 \pmod{m}$ 的所有整数解 (y_1, y_2) 有形式 $(y_1, y_2) = (x_1u_{11} + x_2u_{21}, x_1u_{12} + x_2u_{22})$, 其中 x_1, x_2 是整数。

b) 另外, 如果 $2 | u_{11}u_{21} + u_{12}u_{22} | \leq u_{11}^2 + u_{12}^2 \leq u_{21}^2 + u_{22}^2$ 也成立, 证明在同余式的所有非零解中, $(y_1, y_2) = (u_{11}, u_{12})$ 使 $y_1^2 + y_2^2$ 达到极小。

5. [M30] 证明算法 S 的步骤 S1 到 S3 正确地实施二维中的谱检验。[提示: 见题 4 并证明在步骤 S2 的开始 $(h' + h)^2 + (p' + p)^2 \geq h^2 + p^2$ 。]

6. [M30] 设 a_0, a_1, \dots, a_{t-1} 是 3.3.3 小节中所定义的 a/m 的部分商, 并设 $A = \max_{0 \leq j < t} a_j$ 。证明 $\mu_2 > 2\pi/(A + 1 + 1/A)$ 。

7. [HM22] 证明当 $q_1, \dots, q_{t-1}, q_{t+1}, \dots, q_t$ 是实数时, 式 (23) 后面的问题 a) 和问题 b) 有同样的解 (参见 (24), (26))。

8. [M18] 表 1 的行 10 有一个非常低的 μ_2 值, 但 μ_3 是令人满意的。当 $\mu_2 = 10^{-6}$ 和 $m = 10^{10}$ 时, μ_3 可能的最高值是什么?

9. [HM32] (C. Hermite, 1846) 设 $f(x_1, \dots, x_t)$ 是如何在 (17) 中那样由矩阵 U 定义的一个正定二次型, 并设 θ 是 f 在非零整数点上的极小值。证明 $\theta \leq \left(\frac{4}{3}\right)^{(t-1)/2} |\det U|^{2/t}$ 。[提示: 如果 W 是行列式为 1 的任何整矩阵, 则矩阵 WU 定义等价于 f 的一个二次型; 而且如果 S 是任何正交矩阵 (即 $S^{-1} = S^T$), 则矩阵 US 定义恒等于 f 的一个二次型。证明有一个等价的二次型 g , 其极小值 θ 在 $(1, 0, \dots, 0)$ 处出现。然后通过归纳法证明一般结果, 写 $g(x_1, \dots, x_t) = \theta(x_1 + \beta_2x_2 + \dots + \beta_tx_t)^2 + h(x_2, \dots, x_t)$, 其中 h 是 $t-1$ 个变量的一个正定二次型。]

10. [M28] 设 y_1 和 y_2 是使得 $y_1 + ay_2 \equiv 0 \pmod{m}$ 和 $y_1^2 + y_2^2 < \sqrt{4/3}m$ 的互素整数。证明存在整数 u_1 和 u_2 使得 $u_1 + au_2 \equiv 0 \pmod{m}$, $u_1y_2 - u_2y_1 = m$, $2 | u_1y_1 + u_2y_2 | \leq \min(u_1^2 + u_2^2, y_1^2 + y_2^2)$ 且 $(u_1^2 + u_2^2)(y_1^2 + y_2^2) \geq m^2$ (因此由习题 4, $\nu_2^2 = \min(u_1^2 + u_2^2, y_1^2 + y_2^2)$)。

► 11. [HM30] (Alan G. Waterman, 1974) 试设计出计算乘数 $a \equiv 1 \pmod{4}$ 的一个相当有效的过程, 对于这个乘数, 同余式 $y_1 + ay_2 \equiv 0 \pmod{m}$ 存在互素的解且有 $y_1^2 + y_2^2 = \sqrt{4/3}m - \epsilon$, 其中给定 $m = 2^e$, $\epsilon > 0$ 尽可能小。(由习题 10, a 的这一选择将保证 $\nu_2^2 \geq m^2/(y_1^2 + y_2^2) > \sqrt{3/4}m$ 而且存在 ν_2^2 接近它的最优值 $\sqrt{4/3}m$ 的可能。实践中, 我们将计算若干个有这样的小的 ϵ 的乘数, 并选择有最好的谱值 ν_2, ν_3, \dots 的一个。

12. [HM23] 不用几何图示, 证明正文的问题 b) 的解必定也满足方程组 (26)。

13. [HM22] 引理 A 利用 U 的非奇异性来证明一个正定二次型在非零整数点上达到一个确定的非零极小值。给出一个系数矩阵为奇异的二次型 (19), 使得在非零整数点 (x_1, \dots, x_t) 处, $f(x_1, \dots, x_t)$ 的值任意地接近零 (但不达到零), 以证明这个假设是必要的。

14. [24] 对于 $m = 100, a = 41, T = 3$, 手工实施算法 S。

► 15. [M20] 设 U 是满足 (15) 的一个整向量。问有多少个由 U 定义的 $t-1$ 维超平面交于单位超体 $\{(x_1, \dots, x_t) | \text{对于 } 1 \leq j \leq t, 0 \leq x_j < 1\}$? (这近似于族中足以覆盖 L_0 的超平面的个数。)

16. [M30] (U. Dieter) 说明如何修改算法 S 以便计算如题 15 中那样交于单位超体的平行超平面的极小个数 N_t (取遍满足 (15) 的所有 U)。[提示: 正定二次型和引理 A 的适当的类比是什

么?]

17. [20] 修改算法 S, 使得除计算量 v_t 外, 它还输出满足 (15) 的所有整向量, 使得对于 $2 \leq t \leq T$, $u_1^2 + \cdots + u_t^2 = v_t^2$ 。

18. [M30] 本习题是关于算法 S 的最坏情况的。

a) 通过考虑其元素有 $y + x\delta_j$ (请见习题 1.2.3-39) 形式的“组合矩阵”, 找出满足 (29) 的 3×3 整数矩阵 U 和 V , 使得对于任何 j , 步骤 S5 的变换什么也不做, 但是 (31) 中对应的 z_k 值是如此之大, 以致穷尽查找根本不可能。(矩阵 U 不必满足 (28); 这里我们感兴趣的是行列式 m 的任何正定二次型。)

b) 尽管变换 (23) 对于 a) 中所构造的矩阵没有什么用, 但试求出确实产生相当大简化的另一个变换。

► 19. [HM25] 假设稍微改变步骤 S5, 使得当 $2V_i \cdot V_j = V_j \cdot V_j$ 时对 $q=1$ 做一变换。(于是每当 $i \pm j$ 时 $q = \lfloor (V_i \cdot V_j / V_j \cdot V_j) + 1/2 \rfloor$ 。) 问算法 S 是否可能进入一个无穷循环?

20. [M23] 试讨论对于 $c=0$, X_0 为奇数, $m=2^e$, $a \bmod 8=5$ 的线性同余序列, 如何实施适当的谱检验。

21. [M20] (R. W. Gosper) 以 4 个一批来使用随机数, 但“抛弃”每组中的第二个。给定周期 $m=2^e$ 的一个线性同余生成程序, 如何研究 $\left\{ \frac{1}{m} (X_{4n}, X_{4n+2}, X_{4n+3}) \right\}$ 的栅格结构?

22. [M46] 假定 μ_2 非常接近于它的极大值 $\sqrt{4/3}\pi$, 什么是 μ_3 的最好上界? 假定 μ_3 非常接近于它的极大值 $\frac{4}{3}\pi\sqrt{2}$, 什么是 μ_2 的最好上界?

23. [M46] 设 U_i, V_j 是对于 $1 \leq i, j \leq t$ 满足 $U_i \cdot V_j = \delta_{ij}$ 的实数向量, 且使得对于 $i \neq j$, $U_i \cdot U_j = 1$, $2|U_i \cdot U_j| \leq 1$, $2|V_i \cdot V_j| \leq V_j \cdot V_j$ 。问 $V_1 \cdot V_1$ 能有多大? (如果 (23) 和习题 18b) 的变换都不能做任何约简, 则这个问题同步骤 S7 中的界有关。已知要达到的极大值是 $(t+2)/3$, 对于 $2 \leq j \leq t$, 当 $U_1 = I_1$, $U_j = \frac{1}{2}I_1 + \frac{1}{2}\sqrt{3}I_j$, $V_1 = I_1 - (I_2 + \cdots + I_t)/\sqrt{3}$, $V_j = 2I_j/\sqrt{3}$ 时它出现, 其中 (I_1, \cdots, I_t) 是单位矩阵; 这个构造属于 B. V. Alexeev。

► 24. [M28] 把谱检验推广到形如 $X_n = (aX_{n-1} + bX_{n-2}) \bmod p$, 并有周期长度 $p^2 - 1$ 的二阶序列 (参考式 3.2.2-(8)), 应如何修改算法 S?

25. [HM24] 设 d 是 m 的一个因子, 且设 $0 \leq q < d$, 证明遍取使得 $k \bmod d = q$ 的所有 $0 \leq k < m$ 所求的和数 $\sum r(k)$, 其值顶多是 $(2/d\pi) \ln(m/d) + O(1)$ 。(这里当 $t=1$ 时 $r(k)$ 在等式 (46) 中定义。)

26. [M22] 说明对于 $0 < q < m$, 为什么 (53) 的推导导致对

$$\left| N^{-1} \sum_{0 \leq n < N} \omega^{qn} \right|$$

的一个类似上界。当 $m=1$ 时, (53) 的推导在何处出问题?

27. [HM39] (E. Hlawka, H. Niederreiter) 设 $r(u_1, \cdots, u_t)$ 是 (46) 中定义的函数。试证明, 若对使得 $(u_1, \cdots, u_t) \neq (0, \cdots, 0)$ 和 (15) 成立的所有 $0 \leq u_1, \cdots, u_t < m$ 取和 $\sum r(u_1, \cdots, u_t)$, 其值至多是 $2((\pi + 2\pi \lg m)^t r_{\max})$, 其中 r_{\max} 是在和数中的极大项 $r(u_1, \cdots, u_t)$ 。

► 28. [M28] (H. Niederreiter) 在 m 为素数, $c=0$, a 为取模 m 的原根, $X_0 \not\equiv 0 \pmod{m}$ 的情况下, 求与定理 N 相类似的一个定理。[提示: 你的指数和应当含 ω , 也含 $\zeta = e^{2\pi i/(m-1)}$ 。] 证明在此情况下, “平均”原根有差异 $D_m^{(t)} - 1 = O(t(\log m)^t / \varphi(m-1))$, 因此对于所有 m , 存在好的原

根。

29. [HM22] 证明习题 27 的量 r_{\max} 绝不大于 $1/\sqrt{8}v_i$ 。

30. [M33] (S. K. Zaremba) 证明在二维中 $r_{\max} = O(\max(a_1, \dots, a_s)/m)$, 其中 a_1, \dots, a_s 是应用欧几里得算法于 m 和 a 时得到的部分商。[提示: 在 4.5.3 小节的记号下, 我们有 $a/m = //a_1, \dots, a_s//$; 应用习题 4.5.3-42。]

31. [HM47] (I. Borosh) 证明对于所有充分大的 m , 存在一个数 a 与 m 互素, 使得 a/m 的所有部分商小于等于 3。其次满足这个条件但所有部分商小于等于 2 的所有 m 的集合有正的密度。

▶ 32. [M21] 设 $m_1 = 2^{31} - 1$ 和 $m_2 = 2^{31} - 249$ 是生成程序 (38) 的模数。

a) 证明如果 $U_n = (X_n/m_1 - Y_n/m_2) \bmod 1$, 则我们有 $U_n \approx Z_n/m_1$ 。

b) 设 $W_0 = (X_0 m_2 - Y_0 m_1) \bmod m$ 和 $W_{n+1} = a W_n \bmod m$, 其中 a 和 m 有在 (38) 之后的正文中所述的值。试证明在 W_n 和 U_n 之间存在一个简单的关系。



在本书的下一版中, 我计划引入一个题目为“ L^3 算法”的新的 3.3.5 小节。它将从随机数的一般课题扯开去, 但它将继续 3.3.4 小节中格约简的讨论, 它的主要课题是 A. K. Lenstra, H. W. Lenstra, Jr. 及 L. Lovász [Math. Annalen 261 (1982), 515 ~ 534] 用于求基向量接近最优的集合的现代经典算法, 以及随后其他研究者对该算法的改进。后者的例子可从下列论文及他们的参考文献中找到: M. Seysen, Combinatorica 13 (1993), 363 ~ 375; C. P. Schnorr 和 H. H. Hörner, Lecture Notes in Comp. Sci. 921 (1995), 1 ~ 12。

3.4 其它类型的随机量

我们现在已经看到如何使一台计算机生成序列 U_0, U_1, U_2, \dots , 这个序列的特性是似乎每个数都以一致分布的方式在 0 和 1 之间随机地选择。然而, 随机数的应用通常要求其它类型的分布; 例如, 如果我们要从 k 个可能中进行随机选择, 我们就要有 1 和 k 之间的随机整数。如果某个模拟过程要求在独立事件的出现之间有一个随机等候时间, 则需要具有指数分布的一个随机数。有时我们甚至不要随机数——我们要一个随机排列 (即 n 个对象的一个随机安排) 或一个随机组合 (即从 n 个对象的集合中选择 k 个对象)。

从原理上说, 任何这些其它的随机量都可以从一致偏离 U_0, U_1, U_2, \dots 中得到。人们已经想出一些重要的“随机技巧”, 它用于对一致偏离进行有效的变换。对于这些技术的研究使我们对在蒙特卡罗方法的应用中如何适当地使用随机数有了进一步的洞察。

可以想像, 有一天某人将发明一个随机数生成程序, 它能直接地, 而不是间接地通过一致分布产生出这些其它的随机量之一。但是除了在 3.2.2 小节所描述的“随机位”生成程序之外, 至今还没有已被证明是实用的直接方法。(请见习题 3.4.1-31, 其中一致分布主要用于初始化, 在那之后这个方法就几乎完全直接了。)

下面一节的讨论假定在 0 与 1 之间一致分布的一个实数随机序列的存在性。当我们需要时,就可以生成一个新的一致偏离 U 。这些数通常以一个计算机字表示,并假定小数点在左边。

3.4.1 数值分布

这一节综述由各种重要的分布类型产生数的已知的最好的技术。其中许多方法最初是由 John von Neumann 于 20 世纪 50 年代初提出的,而后逐渐地由其他人主要是 George Marsaglia, J. H. Ahrens 及 U. Dieter 加以改进。

A. 从一个有限集合随机选择 实用上要求的最简单最普通的分布类型是一个随机整数。0 与 7 之间的一个整数可以在一台二进计算机上从 U 的三个二进位中抽取出来;在这种情况下,这些二进位应从计算机字的最高有效位(左边)部分抽出,因为由许多随机数生成程序产生的低有效位是不充分随机的(见 3.2.1.1 小节中的讨论)。

一般地说,为得到 0 与 $k-1$ 之间的一个随机数 X ,我们可以乘以 k ,并设 $X = \lfloor kU \rfloor$ 。在 MIX 上,我们将写成

$$\begin{array}{ll} \text{LDA} & U \\ \text{MUL} & K \end{array} \quad (1)$$

在这两条指令被执行之后,所需要的整数将出现于寄存器 A 中。如果所需的是 1 与 k 之间的随机整数,那么往这个结果上加 1(把指令“INCA 1”接在(1)之后)。

这个方法以几乎相等的概率给出每个整数。由于计算机字长的有限性而略有误差(见习题 2)。但如果 k 是小的,例如,如果 $k/m < 1/10000$,则这个误差是完全可以忽略的。

在更一般的情况下,可能要求对于不同的整数给出不同的权。假设得到值 $X = x_1$ 的概率为 p_1 ,得到值 $X = x_2$ 的概率为 p_2, \dots ,得到值 $X = x_k$ 的概率为 p_k 。我们可以生成一致的数 U ,并令

$$X = \begin{cases} x_1, & \text{如果 } 0 \leq U < p_1 \\ x_2, & \text{如果 } p_1 \leq U < p_1 + p_2 \\ \vdots & \\ x_k, & \text{如果 } p_1 + p_2 + \dots + p_{k-1} \leq U < 1 \end{cases} \quad (2)$$

(注意: $p_1 + p_2 + \dots + p_k = 1$ 。)

有一个“最好”的方法来进行(2)中所隐含的 U 与 $p_1 + p_2 + \dots + p_k$ 的各种值的比较;在 2.3.4.5 小节中讨论了这种情况。特殊情况则可以用更有效的方法处理;例如,为了分别以骰子概率 $\frac{1}{36}, \frac{2}{36}, \dots, \frac{6}{36}, \dots, \frac{2}{36}, \frac{1}{36}$ 得到 11 个值 2, 3, \dots , 12 之一,我们可以计算 1 与 6 之间的两个独立的随机整数,并把它们加在一起。

然而,基于由 A. J. Walker 引入的巧妙方法[*Electronics Letters* 10, 8(1974), 127

~ 128 ; ACM Trans. Math. Software 3(1977), 253~256], 实际上有一个以任意给定的概率来选择 x_1, \dots, x_k 的快速方法。假设我们形成 kU 并分别地考虑整数部分 $K = \lfloor kU \rfloor$ 和分数部分 $V = (kU) \bmod 1$; 例如, 在代码(1)之后我们将有 K 在寄存器 A 中和 V 在寄存器 X 中。然后通过进行下列操作我们总能得到所需的分布: 对于某些适当的表 (P_0, \dots, P_{k-1}) 和 (Y_0, \dots, Y_{k-1}) ,

$$\text{如果 } V < P_K, \text{ 则 } X \leftarrow x_{K+1}, \text{ 否则 } X \leftarrow Y_K \quad (3)$$

习题 7 说明一般地应如何计算这样的表。Walker 的方法有时称做“别名”方法。

在一台二进计算机上, 假定 k 是 2 的幂通常是有帮助的, 这使得乘法可用移位来代替; 不失一般性, 通过引进以概率 0 出现的另外的一些 X 值可以做到这一点。例如, 再次考虑骰子; 假设我们要 $X=j$ 以下列 16 个概率出现:

$$\begin{array}{cccccccccccccccc} j = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ p_j = & 0 & 0 & \frac{1}{36} & \frac{2}{36} & \frac{3}{36} & \frac{4}{36} & \frac{5}{36} & \frac{6}{36} & \frac{5}{36} & \frac{4}{36} & \frac{3}{36} & \frac{2}{36} & \frac{1}{36} & 0 & 0 & 0 \end{array}$$

如果 $k=16$ 和对于 $0 \leq j < 16, x_{j+1} = j$, 而且如果 P 表和 Y 表建立如下:

$$\begin{array}{cccccccccccccccc} j = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ P_j = & 0 & 0 & \frac{4}{9} & \frac{8}{9} & 1 & \frac{7}{9} & 1 & 1 & 1 & \frac{7}{9} & \frac{7}{9} & \frac{8}{9} & \frac{4}{9} & 0 & 0 & 0 \\ Y_j = & 5 & 9 & 7 & 4 & * & 6 & * & * & * & 8 & 4 & 7 & 10 & 6 & 7 & 8 \end{array}$$

则我们可以用(3)来做此事。(当 $P_j = 1$ 时, 不用 Y_j 。)例如, 如所要求的那样, 值 7 以概率 $\frac{1}{16} \cdot ((1 - P_2) + P_7 + (1 - P_{11}) + (1 - P_{14})) = \frac{6}{36}$ 出现。它是掷骰子的一个特殊方式, 但这些结果并不异于实际事物。

概率 p_j 可以通过非负的权 w_1, w_2, \dots, w_k 含蓄地表示; 如果我们以 W 表示诸权的和, 则 $p_j = w_j / W$ 。在许多应用中个别的权动态地变化。Matias, Vitter 和 Ni [SODA 4 (1993), 361~370] 已经说明如何在一个固定的预期时间内修改一个权和生成 X 。

B. 连续分布的一般分法 借助于“分布函数” $F(x)$, 可以表达最一般的实值分布。这个函数确定随机量 X 不超过 x 的概率:

$$F(x) = \Pr(X \leq x) \quad (4)$$

这个函数总是单调地从 0 增到 1, 即

$$F(x_1) \leq F(x_2), \quad \text{如果 } x_1 \leq x_2; \quad F(-\infty) = 0, \quad F(+\infty) = 1 \quad (5)$$

在 3.3.1 小节图 3 中给出了分布函数的一些例子。如果 $F(x)$ 是连续和严格递增的 (使得当 $x_1 < x_2$ 时, $F(x_1) < F(x_2)$), 它取 0 与 1 之间的所有值, 而且有一个逆函数 $F^{[-1]}(y)$, 使得如果 $0 < y < 1$,

$$y = F(x) \quad \text{当且仅当} \quad x = F^{[-1]}(y) \quad (6)$$

一般来说, 当 $F(x)$ 是连续和严格递增时, 通过置

$$X = F^{[-1]}(U) \quad (7)$$

我们可以用分布 $F(x)$ 计算一个随机量 X , 其中 U 是一致的。可以这样做是因为 $X \leq x$ 的概率是 $F^{[-1]}(U) \leq x$ 的概率, 即 $U \leq F(x)$ 的概率, 而这是 $F(x)$ 。

现在问题归结为一个数值分析问题, 就是找出计算 $F^{[-1]}(U)$ 到所需精度的好方法。数值分析不属于本书所谈的半数值的范围; 但其中有一些重要的可利用的捷径来加速(7)这一一般方法, 因而我们将在这里考虑它们。

首先, 如果 X_1 是分布为 $F_1(x)$ 的一个随机变量, 而且如果 X_2 是分布为 $F_2(x)$ 的一个独立随机变量, 则

$$\begin{aligned} \max(X_1, X_2) \text{ 的分布为 } & F_1(x)F_2(x) \\ \min(X_1, X_2) \text{ 的分布为 } & F_1(x) + F_2(x) - F_1(x)F_2(x) \end{aligned} \quad (8)$$

(见习题 4。)例如, 一致离差 U 对于 $0 \leq x \leq 1$ 的分布是 $F(x) = x$; 如果 U_1, U_2, \dots, U_t 是独立的一致离差, 则 $\max(U_1, U_2, \dots, U_t)$ 对于 $0 < x \leq 1$ 的分布函数是 $F(x) = x^t$ 。这个公式是 3.3.2 小节中给出的“ t 的极大值”检验的基础。它的逆函数是 $F^{[-1]}(y) = \sqrt[t]{y}$ 。因此, 在 $t=2$ 的特殊情况下, 我们可以看到两个公式

$$X = \sqrt{U} \text{ 和 } X = \max(U_1, U_2) \quad (9)$$

给出随机变量 X 的等价分布, 尽管乍一看去这并不显然。我们不需要取一致离差的平方根。

像这样的技巧是不胜枚举的: 任何运用随机数作为输入的算法都给出具有某种分布的随机变量作为输出。问题是给定输出分布函数, 如何找出构造这个算法的一般方法。我们将避开以纯粹抽象的术语讨论这样的方法, 而是将研究在一些重要情况下如何应用它们。

C. 正态分布 也许最重要的非一致的连续分布是具有均值 0 和标准差 1 的正态分布

$$F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-t^2/2} dt \quad (10)$$

在 1.2.10 小节中指出了这个分布的重要性。在这种情况下, 逆函数 $F^{[-1]}$ 不是特别容易计算的, 但我们将看到有若干其它技术可以利用。

1) 配极法是由 G.E.P.Box, M.E.Muller 及 G.Marsaglia 给出的 (见 *Annals Math. Stat.* **29** (1958), 610 ~ 611; 以及波音科学研究实验室报告 D1-82-0203 (1962))。

算法 P(正态离差的配极法) 这个算法计算两个独立的正态分布变量 X_1 和 X_2 。

P1. [得到一致变量] 生成 0 与 1 之间一致分布的两个独立的一致分布变量 U_1 和 U_2 。置 $V_1 \leftarrow 2U_1 - 1$, $V_2 \leftarrow 2U_2 - 1$ 。(现在 V_1 和 V_2 是 -1 和 +1 之间一致分布的, 在大多数计算机上, 以浮点形式表示 V_1 和 V_2 是更可取的。)

P2. [计算 S] 置 $S \leftarrow V_1^2 + V_2^2$ 。

P3. [$S \geq 1$?] 如果 $S \geq 1$, 返回步骤 P1。(步骤 P1 到 P3 平均执行 1.27 次, 标准差为 0.587; 见习题 6。)

P4. [计算 X_1, X_2] 如果 $S = 0$, 置 $X_1 \leftarrow X_2 \leftarrow 0$; 否则置

$$X_1 \leftarrow V_1 \sqrt{\frac{-2 \ln S}{S}}, X_2 \leftarrow V_2 \sqrt{\frac{-2 \ln S}{S}} \quad (11)$$

这些是所需的正态分布变量。 I

为证明这个方法的正确性, 我们使用初等解析几何和微积分。如果在步骤 P3 中 $S < 1$, 则在平面中具有笛卡尔坐标 (V_1, V_2) 的点是在单位圆内一致分布的随机点。变换成极坐标 $V_1 = R \cos \Theta, V_2 = R \sin \Theta$, 我们求得

$$S = R^2, X_1 = \sqrt{-2 \ln S} \cos \Theta, X_2 = \sqrt{-2 \ln S} \sin \Theta$$

也使用极坐标 $X_1 = R' \cos \Theta', X_2 = R' \sin \Theta'$, 我们求得 $\Theta' = \Theta$ 和 $R' = \sqrt{-2 \ln S}$ 。显然 R' 和 Θ' 是独立的, 因为 R 和 Θ 是在单位圆内独立的, 而且, Θ' 在 0 和 2π 之间一致分布。 $R' < r$ 的概率是 $-2 \ln S \leq r^2$ 的概率, 即 $S \geq e^{-r^2/2}$ 的概率。这等于 $1 - e^{-r^2/2}$, 因为 $S = R^2$ 是在 0 和 1 之间一致分布的。因此, R' 位于 r 和 $r + dr$ 之间的概率是 $1 - e^{-r^2/2}$ 的微分, 即 $r e^{-r^2/2} dr$ 。类似地, Θ' 位于 θ 和 $\theta + d\theta$ 之间的概率是 $(1/2\pi)d\theta$ 。 $X_1 \leq x_1$ 和 $X_2 \leq x_2$ 的联合概率现在可以计算出来; 它是

$$\begin{aligned} \int_{\{(r, \theta) | r \cos \theta \leq x_1, r \sin \theta \leq x_2\}} \frac{1}{2\pi} e^{-r^2/2} r dr d\theta = \\ \frac{1}{2\pi} \int_{\{(x, y) | x \leq x_1, y \leq x_2\}} e^{-(x^2+y^2)/2} dx dy = \\ \left(\sqrt{\frac{1}{2\pi}} \int_{-\infty}^{x_1} e^{-x^2/2} dx \right) \left(\sqrt{\frac{1}{2\pi}} \int_{-\infty}^{x_2} e^{-y^2/2} dy \right) \end{aligned}$$

像所希望的那样, 这证明了 X_1 和 X_2 是独立的。

2) 由 G. Marsaglia 引进的矩形-楔形-尾形方法。在这个方法中, 我们使用分布

$$F(x) = \operatorname{erf}(x/\sqrt{2}) = \sqrt{\frac{2}{\pi}} \int_0^x e^{-t^2/2} dt, \quad x \geq 0 \quad (12)$$

$F(x)$ 给出正态离差的绝对值的分布。在按照分布 (12) 计算了 X 之后, 我们就对它的值附加一个随机的正负号, 这使它成为一个真正的正态离差。

矩形-楔形-尾形方法是以若干重要的一般技术为基础的, 我们将在随后建立此方法的过程中剖析这些技术。主要的思想是把 $F(x)$ 看成若干其它函数的一个混合, 即写成

$$F(x) = p_1 F_1(x) + p_2 F_2(x) + \cdots + p_n F_n(x) \quad (13)$$

其中 F_1, F_2, \dots, F_n 是适当的分布, p_1, p_2, \dots, p_n 是和为 1 的非负概率。如果我们通过选择具有概率 p_j 的分布 F_j 生成一个随机变量 X , 容易看出 X 整个地将有分布 F 。某些分布 $F_j(x)$ 可能稍难处理, 甚至比 F 本身更难; 但是我们通常可以把事情

安排成使得在这种情况下概率 p_j 是非常小的。大多数分布 $F_j(x)$ 将十分易于调节,因为它们只是对一致分布稍加修改而已。得到的方法提供了一个非常有效的程序,因为它的平均运行时间非常短。

如果我们以分布的导数而不是以分布本身进行处理,则易于理解这个方法。设

$$f(x) = F'(x), \quad f_j(x) = F'_j(x)$$

这些称为概率分布的密度函数。等式(13)变为

$$f(x) = p_1 f_1(x) + p_2 f_2(x) + \cdots + p_n f_n(x) \quad (14)$$

每个 $f_j(x)$ 非负,而且,在 $f_j(x)$ 的图形之下的整个面积为 1;所以有一个方便的图形方式来表示关系(14): $f(x)$ 之下的区域分成 n 部分,而且对应于 $f_j(x)$ 的部分之面积为 p_j 。请看图 9,这里它描述了我们感兴趣的情形,其中 $f(x) = F'(x) = \sqrt{2/\pi} \cdot e^{-x^2/2}$; 曲线之下的区域已经分成 $n=31$ 部分。有 15 个矩形,它们代表 $p_1 f_1(x), \cdots, p_{15} f_{15}(x)$; 有 15 个楔形块,它们表示 $p_{16} f_{16}(x), \cdots, p_{30} f_{30}(x)$; 还有剩下部分 $p_{31} f_{31}(x)$ 是“尾形”,这即当 $x \geq 3$ 时 $f(x)$ 的整个图形。

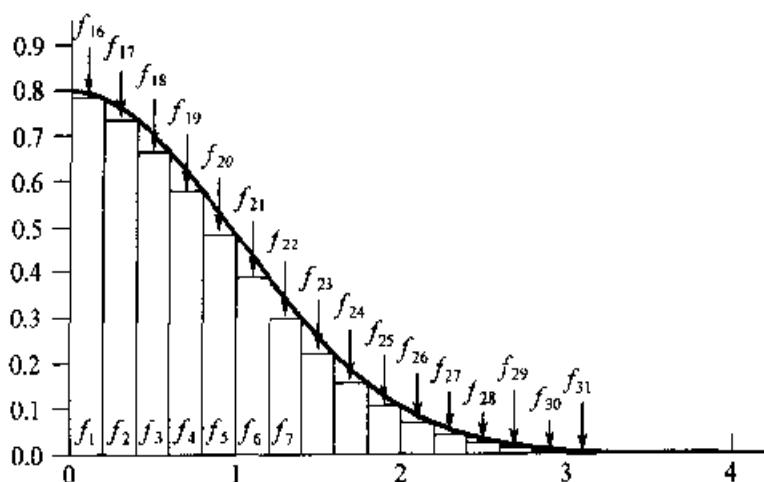


图 9 密度函数分成 31 部分,每个部分的面积表示具有该密度的一个随机数的有待计算的平均次数

矩形部分 $f_1(x), \cdots, f_{15}(x)$ 表示一致分布。例如, $f_3(x)$ 表示在 $\frac{2}{5}$ 和 $\frac{3}{5}$ 之间一致分布的随机变量。 $p_j f_j(x)$ 的高度是 $f(j/5)$, 因此第 j 个矩形的面积是

$$p_j = \frac{1}{5} f(j/5) = \sqrt{\frac{2}{25\pi}} e^{-j^2/50}, \quad \text{对于 } 1 \leq j \leq 15 \quad (15)$$

为了生成分布的这种矩形部分,我们只须计算

$$X = \frac{1}{5} U + S \quad (16)$$

其中 U 是一致的,而 S 以概率 p_j 取 $(j-1)/5$ 的值。由于 $p_1 + \cdots + p_{15} = 0.9183$, 因此我们可以在大约 92% 的时间里使用像这样的简单一致离差。

在剩下的 8% 当中, 我们通常将要生成楔形分布 F_{16}, \dots, F_{30} 之一。我们需要做的事情的典型例子如图 10 所示。当 $x < 1$ 时, 曲线部分是向下凹的, 而当 $x > 1$ 时它是向上凸的, 但在每种情况下曲线部分相当接近于直线, 因而它可以包括在两条平行线之间, 如图所示的那样。

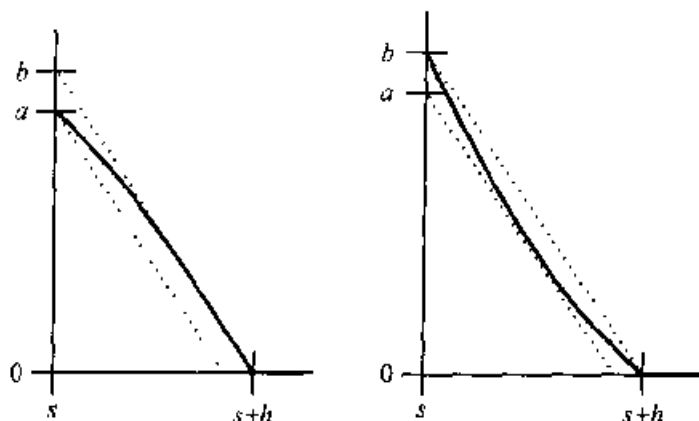


图 10 密度函数, 它们可被算法 L 用来生成随机数

为了处理这些楔形分布, 我们将依赖于另一项一般技术, 即冯诺伊曼为从“包围”一个复杂密度的另一个密度得到这个复杂密度的拒绝方法。上边描述的配极法是这种方法的简单例子: 步骤 P1~P3 首先在一个较大的正方形内生成一个随机点, 如果该点在单位圆的外部就拒绝它并重复该步骤, 最后得到在单位圆内的一个随机点。

一般的拒绝方法甚至比这还更加有效, 为生成具有密度 f 的随机变量 X , 设 g 是使得对所有 t

$$f(t) \leq cg(t) \quad (17)$$

的另一个概率密度函数, 其中 c 是常数。现在按照密度 g 生成 X , 还生成一个独立的一致离差 U 。如果 $U \geq f(X)/cg(X)$, 则拒绝 X 并再次以另一个 X 和 U 开始。当条件 $U < f(X)/cg(X)$ 最终出现时, 得到的 X 将有所需的密度 f 。[证明: $X \leq x$

将以 $p(x) = \int_{-\infty}^x (g(t)dt \cdot f(t)/cg(t)) + qp(x)$ 的概率出现, 其中量 $q = \int_{-\infty}^{\infty} (g(t)dt \cdot (1 - f(t)/cg(t))) = 1 - 1/c$ 是拒绝的概率, 因此 $p(x) = \int_{-\infty}^x f(t)dt$ 。]

当 c 很小时, 拒绝技术是最有效的, 因为在接受一个值之前, 平均将有 c 次迭代 (见习题 6)。在某些情况下, $f(x)/cg(x)$ 总是 0 或 1; 此时 U 不必被生成。在其它情况下, 如果 $f(x)/cg(x)$ 难以计算, 我们可以在两个界函数之间来“挤”它:

$$r(x) \leq f(x)/cg(x) \leq s(x) \quad (18)$$

这些界函数是简单得多的函数, 而且除非 $r(x) \leq U \leq s(x)$, 否则 $f(x)/cg(x)$ 的精确值不必计算。以下的算法通过进一步建立拒绝方法, 解决楔形问题。

算法 L(近乎线性的密度) 这个算法可以用来生成一个任何分布的随机变量 X , 其密度 $f(x)$ 满足以下条件(参考图 10):

$$\begin{aligned} f(x) &= 0, \quad \text{对于 } x < s \text{ 及 } x > s+h \\ a - b(x-s)/h &\leq f(x) \leq b - b(x-s)/h, \quad \text{对于 } s \leq x \leq s+h \end{aligned} \quad (19)$$

L1. [获得 $U \leq V$] 生成在 0 和 1 之间的一致分布的两个随机变量 U, V 。如果 $U > V$, 则交换 $U \leftrightarrow V$ 。

L2. [容易的情况?] 如果 $V \leq a/b$, 转向 L4。

L3. [再试?] 如果 $V > U + (1/b)f(s+hU)$, 则转回到步骤 L1。(如果 a/b 接近于 1, 则算法的这一步经常是不必要的。)

L4. [计算 X] 置 $X \leftarrow s + hU$ 。 ■

当达到步骤 L4 时, 点 (U, V) 是图 11 中阴影区域中的一个随机点, 即 $0 \leq U \leq V \leq U + (1/b)f(s+hU)$ 。条件(19)确保

$$\frac{a}{b} \leq U + \frac{1}{b}f(s+hU) \leq 1$$

现在对于 $0 \leq x \leq 1$, $X \leq s+hx$ 的概率是图 11 中垂直线 $U=x$ 左边的面积, 除以整个的面积, 即

$$\int_0^x \frac{1}{b}f(s+hu)du \bigg/ \int_0^1 \frac{1}{b}f(s+hu)du = \int_s^{s+hx} f(v)dv$$

因此 X 有正确的分布。

使用适当的常数 a_j, b_j, s_j , 算法 L 将对 $1 \leq j \leq 15$ 处理图 9 的楔形密度 f_{j+15} 。最后的分布 F_{31} 大约每隔 370 次仅须处理一次; 即仅当要计算 $X \geq 3$ 的结果时才使用它。习题 11 说明对于这个“尾巴”, 可以使用的一个标准的拒绝方案。因此我们可以全面考虑这个过程:

算法 M(正态离差的矩形-楔形-尾形方法) 这个算法使用一些辅助表 $(P_0, \dots, P_{31}), (Q_1, \dots, Q_{15}), (Y_0, \dots, Y_{31}), (Z_0, \dots, Z_{31}), (S_1, \dots, S_{16}), (D_{16}, \dots, D_{30}), (E_{16}, \dots, E_{30})$, 其构造在习题 10 中有说明; 例子见表 1。这里假定使用的是一台二进制计算机; 对于十进计算机也可构造一个类似的过程。

M1. [得到 U] 生成一个一致的随机数 $U = (.b_0b_1b_2 \dots b_t)_2$ 。(这里诸 b 是 U 的二进表示的诸位。为了达到合理的精度, t 至少应是 24。)置 $\phi \leftarrow b_0$ 。(以后 ϕ 将用来确定结果的符号。)

M2. [矩形?] 置 $j \leftarrow (b_1b_2b_3b_4b_5)_2$, 这是由 U 的前五位确定的一个二进制数, 同时置 $f \leftarrow (.b_6b_7 \dots b_t)_2$, 这是由剩余的位确定的小数。如果 $f \geq P_j$, 则置 $X \leftarrow Y_j + fZ_j$ 并转到 M9。否则如果 $j \leq 15$ (即 $b_1 = 0$), 则置 $X \leftarrow S_j + fQ_j$

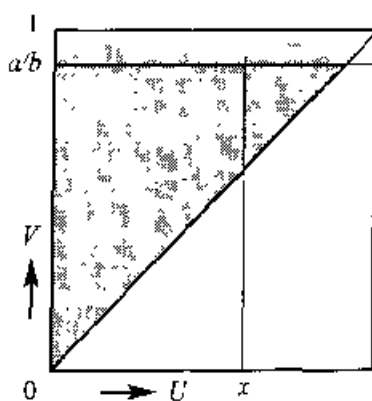


图 11 算法 L 中“接受”的区域

并转到 M9。(这是 Walker 的别名方法(3)的改编。)

M3. [楔形或尾形?] (现在 $16 \leq j \leq 31$ 而且每个具体值 j 以 p_j 的概率出现。)

若 $j = 31$ 则转到 M7。

M4. [得到 $U \leq V$] 生成两个新的一致离差 U 和 V ; 若 $U > V$, 则交换 $U \leftrightarrow V$ 。(我们现在正实施算法 L。)置 $X \leftarrow S_{j-15} + \frac{1}{5}U$ 。

M5. [容易的情况?] 若 $V \leq D_j$, 则转到 M9。

M6. [另一次试验?] 若 $V > U + E_j(e^{(S_{j-15}^2 - X^2)/2} - 1)$, 转回步骤 M4; 否则转到 M9。(这一步被执行的概率很小。)

M7. [得到超尾离差] 生成两个新的一致离差 U 和 V , 并置 $X \leftarrow \sqrt{9 - 2 \ln V}$ 。

M8. [拒绝?] 若 $UX \geq 3$, 则转回到步骤 M7。(这种情况出现的次数大约仅是我们达到步骤 M8 的次数的 $1/12$ 。)

M9. [加上正负号] 若 $\phi = 1$ 置 $X \leftarrow -X$ 。 ■

这个算法是数学理论同程序设计才智紧密结合完全融汇的极好例子——这是计算机程序设计艺术的一个极好说明! 在大多数时间里仅须实施步骤 M1, M2 和 M9, 而其它步也并不太慢。关于矩形-楔形-尾形方法最先的著作有 G. Marsaglia, *Annals Math. Stat.* **32** (1961), 894 ~ 899; G. Marsaglia, M. D. MacLaren 及 T. A. Bray, *CACM* **7** (1964), 4 ~ 10。G. Marsaglia, K. Anathanarayanan 及 N. J. Paul 对算法 M 做了进一步的改进, *Inf. Proc. Letters* **5** (1976), 27 ~ 30。

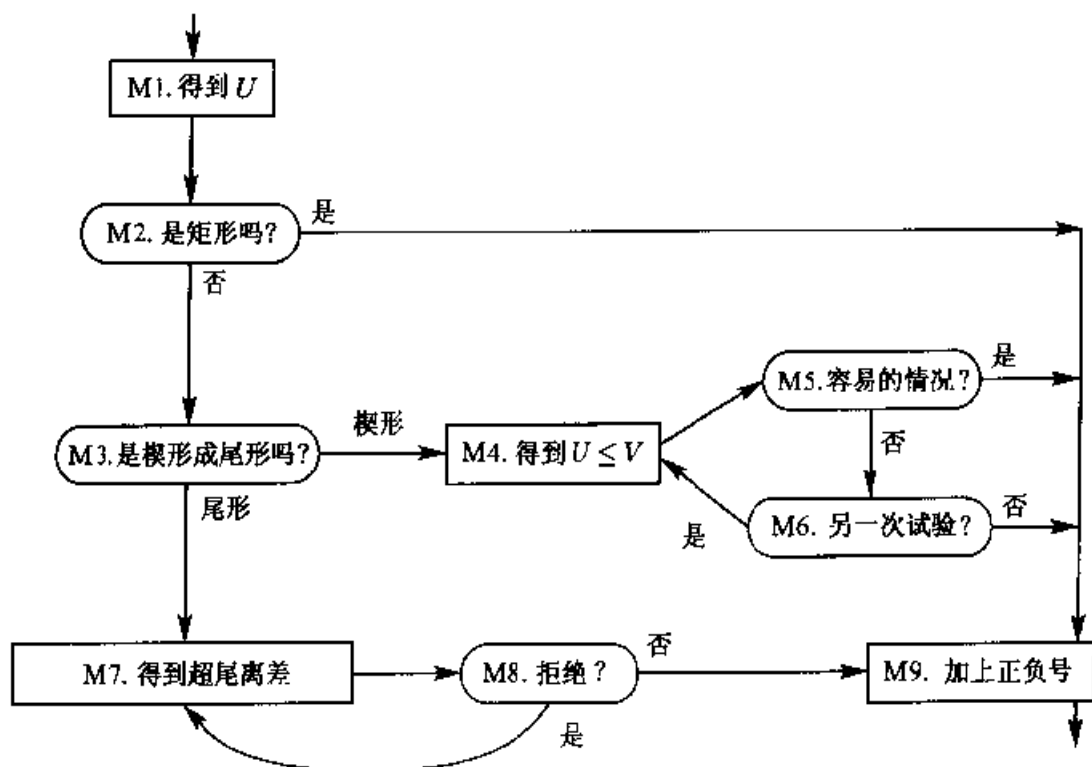


图 12 生成正态离差的“矩形-楔形-尾形”算法

3)由 G. E. Forsythe 给出的奇偶方法。1950 年左右, John von Neumann 和 G. E. Forsythe 发现,在具有一般指数形式

$$f(x) = Ce^{h(x)}[a \leq x < b] \quad (20)$$

的密度下,当

$$0 \leq h(x) \leq 1, \quad \text{对于 } a \leq x < b \quad (21)$$

时,有着惊人简单的生成随机离差的技术。这一思想是以前边描述的拒绝方法为基础的,令 $g(x)$ 是 $[a, b)$ 上的一致分布;置 $X \leftarrow a + (b - a)U$, 其中 U 是一致离差,然后我们要以 $e^{-h(X)}$ 的概率接受 X 。当 V 是另一个一致离差时,对 $e^{-h(X)}$ 和 V , 或对 $h(X)$ 和 $-\ln V$ 做比较,可以完成后一操作,但无须应用任何超越函数,而以下列有趣的方式也可做这件事。置 $V_0 \leftarrow h(X)$, 然后生成一致离差 V_1, V_2, \dots 直到求得某个 $K \geq 1$, 且 $V_{K-1} < V_K$ 为止。对于固定的 X 和 k , $h(X) \geq V_1 \geq V_2 \geq \dots \geq V_k$ 的概率是 $1/k!$ 乘以 $\max(V_1, \dots, V_k) \leq h(X)$ 的概率,即 $h(X)^k/k!$; 因此 $K = k$ 的概率是 $h(X)^{k-1}/(k-1)! - h(X)^k/k!$, 而且 K 为奇数的概率为

$$\sum_{k \text{ 奇}, k \geq 1} \left(\frac{h(X)^{k-1}}{(k-1)!} - \frac{h(X)^k}{k!} \right) = e^{-h(X)} \quad (22)$$

因此,当 K 为偶数时,我们拒绝 X 并再次试验;如果 K 为奇数,我们就接受 X 作为具有密度(20)的一个随机变量。我们通常不用生成许多 V 以确定 K , 因为 K 的平均值(给定 X)是 $\sum_{k \geq 0} \Pr(K > k) = \sum_{k \geq 0} h(X)^k/k! = e^{-h(X)} \leq e$ 。

表 1 和算法 M 一起使用的表举例 *

j	P_j	P_{j+15}	Q_j	Y_j	Y_{j+15}	Z_j	Z_{j+15}	S_{j+1}	D_{j+15}	E_{j+15}
0	.000	.067		0.00	0.59	0.20	0.21	0.0		
1	.849	.161	.236	-0.92	0.96	1.32	0.24	0.2	.505	25.00
2	.970	.236	.206	-5.86	-0.06	6.66	0.26	0.4	.773	12.50
3	.855	.285	.234	-0.58	0.12	1.38	0.28	0.6	.876	8.33
4	.994	.308	.201	-33.16	1.31	34.96	0.29	0.8	.939	6.25
5	.995	.304	.201	-39.51	0.31	41.31	0.29	1.0	.986	5.00
6	.933	.280	.214	-2.57	1.12	2.97	0.28	1.2	.995	4.06
7	.923	.241	.217	-1.61	0.54	2.61	0.26	1.4	.987	3.37
8	.727	.197	.275	0.67	0.75	0.73	0.25	1.6	.979	2.86
9	1.000	.152	.200		0.56		0.24	1.8	.972	2.47
10	.691	.112	.289	0.35	0.17	0.65	0.23	2.0	.966	2.16
11	.454	.079	.440	-0.17	0.38	0.37	0.22	2.2	.960	1.92
12	.287	.052	.698	0.92	-0.01	0.28	0.21	2.4	.954	1.71
13	.174	.033	1.150	0.36	0.39	0.24	0.21	2.6	.948	1.54
14	.101	.020	1.974	-0.02	0.20	0.22	0.20	2.8	.942	1.40
15	.057	.086	3.526	0.19	0.78	0.21	0.22	3.0	.936	1.27

* 实际应用中,这些数据将以更高得多的精度给出;本表仅示出足够的数字使得感兴趣的读者可以检查他们自己的程序以便更精确地计算这些值

若干年以后, Forsythe 认识到这个方法导出了计算正态离差的一个非常有效的方法,而无须像在算法 P 和 M 中那样,使用任何辅助的子程序来计算平方根或对数。他的过程连同 J. H. Ahrens 和 U. Dieter 给出的改进了的区间 $[a, b)$ 的选择,可

综述如下:

算法 F(正态离差的奇偶法) 在一台近似地有 $t+1$ 位精度的二进计算机上, 本算法生成正态离差。本算法要求有一张对于 $1 \leq j \leq t+1$ 的值 $d_j = a_j - a_{j-1}$ 的表, 其中 a_j 是由关系

$$\sqrt{\frac{2}{\pi}} \int_{a_j}^{\infty} e^{-x^2/2} dx = \frac{1}{2^j} \quad (23)$$

定义的。

F1. [得到 U] 生成一个一致随机数 $U = (.b_0 b_1 \cdots b_t)_2$, 其中 b_0, \dots, b_t 表示在二进制下的诸位, 置 $\psi \leftarrow b_0, j \leftarrow 1$ 及 $a \leftarrow 0$ 。

F2. [求头一个为 0 的 b_j] 若 $b_j = 1$, 置 $a \leftarrow a + d_j, j \leftarrow j + 1$, 并重复此步。(若 $j = t + 1$, 则视 b_j 为 0。)

F3. [生成候选者] (现在 $a = a_{j-1}$, 而且 j 的当前值以约等于 2^{-j} 的概率出现。使用上边叙述的拒绝法, 我们将在 $[a_{j-1}, a_j)$ 的范围内生成 X , 且 $h(x) = x^2/2 - a^2/2 = y^2/2 + ay$, 其中 $y = x - a$ 。习题 12 证明, 如(21)中所要求的, $h(x) \leq 1$ 。)置 $Y \leftarrow d_j$ 乘以 $(.b_{j+1} \cdots b_t)_2$ 且 $V \leftarrow \left(-\frac{1}{2}Y + a\right)Y$ 。(由于 j 的平均值是 2, 通常在 $(.b_{j+1} \cdots b_t)_2$ 中有足够的有效值以提供合适的精度。在定点算术中做这些计算比较容易。)

F4. [拒绝?] 生成一个一致离差 U , 如果 $V < U$, 则转向步骤 F5。否则置 U 成为一个新的一致离差; 如果现在 $U < V$ (即如果在上边的讨论中, K 为偶), 则转回 F3, 否则重复步骤 F4。

F5. [返回 X] 置 $X \leftarrow a + Y$ 。如果 $\psi = 1$, 则置 $X \leftarrow -X$ 。 ▮

在 Ahrens 和 Dieter 的文章 [Math. Comp. 27 (1973), 927~937] 中, 给出了对于 $1 \leq j \leq 47$ 时 d_j 的值; 他们的文章讨论了以更多的表为代价来改进算法的速度。算法 F 是吸引人的, 因为它几乎和算法 M 一样快, 而且它更易于实现。每个正态离差的一致离差的平均个数是 2.53947; R. P. Brent [CACM 17(1974), 704~705] 已经指出怎样以每节省一个一致离差花费两个减法和一个除法的代价把这个数减少到 1.37446。

4) 一致离差的比例。还有另一个生成正态离差的好方法, 这是 1976 年由 A. J. Kinderman 和 J. F. Monahan 发现的。他们的思想是在由

$$0 < u \leq 1, \quad -2u \sqrt{\ln(1/u)} \leq v \leq 2u \sqrt{\ln(1/u)} \quad (24)$$

定义的范围内生成一个随机点 (U, V) , 而后输出比例 $X \leftarrow V/U$ 。图 13 的阴影区是使这个方法有效的魔术区域。在研究有关理论之前, 我们首先叙述这个算法, 以显示它的效率和简洁性。

算法 R(正态离差的比例方法) 本算法生成正态离差 X 。

R1. [得到 U, V] 生成两个独立的一致离差 U 和 V , 其中 U 非零, 并置 $X \leftarrow$

$\sqrt{8/e} \left(V - \frac{1}{2} \right) / U$ 。(现在 X 是包含图 13 的阴影区域部分的矩形中一个随机点的坐标 $\left(U, \sqrt{8/e} \left(V - \frac{1}{2} \right) \right)$ 之比率。如果对应的点实际上位于“阴影”内,我们将接受 X ,否则我们将再次试验。)

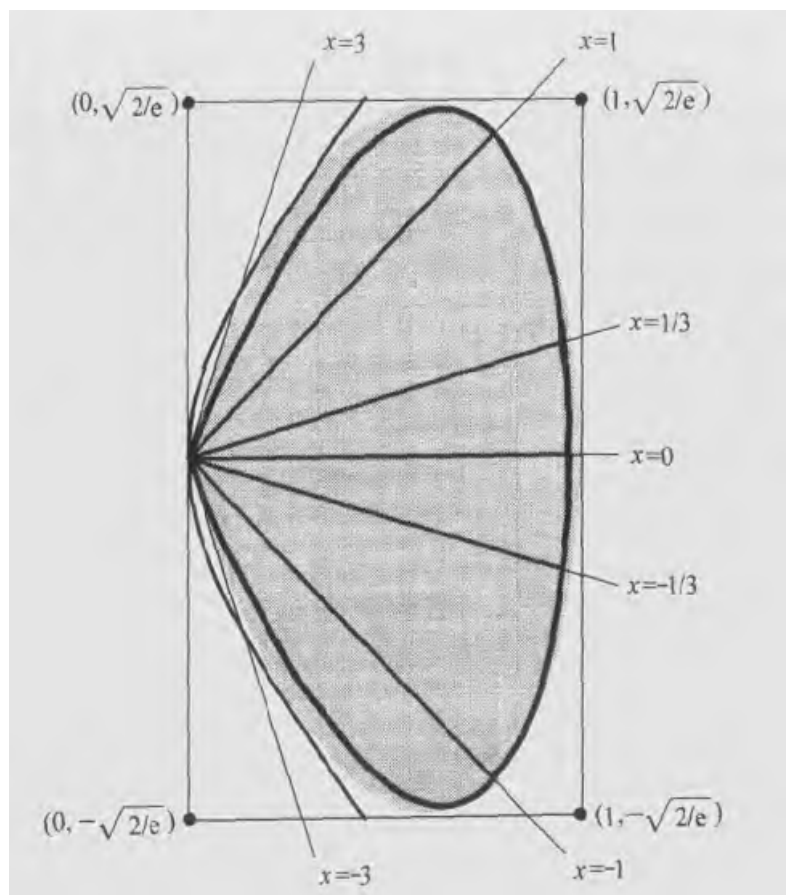


图 13 在正态离差的一致比例的方法中的“接受”区域,具有坐标比值 x 的线段长度有正态分布

- R2.** [任选的上界测试] 如果 $X^2 \leq 5 - 4e^{1/4} U$, 输出 X 并结束此算法。(如果愿意,此步可以省去;它测试所选择的点是否在图 13 的内部区域,以便省掉一次对数计算。)
- R3.** [任选的下界测试] 如果 $X^2 \geq 4e^{-1.35}/U + 1.4$, 则转回 R1。(这步也可省去;它测试所选择的点是否在图 13 的外部区域,以便省掉一次对数计算。)
- R4.** [最后的测试] 如果 $X^2 \leq -4 \ln U$, 则输出 X 并结束此算法,否则转回 R1。 ■

习题 20 和 21 给出计时分析;分析了四个不同的算法,因为步骤 R2 和 R3 根据个人的爱好可以包括进来或省去。下表给出每步将被执行的平均次数,它取决于选用什么样的测试。

步骤	全不用	仅用 R2	仅用 R3	全用
R1	1.369	1.369	1.369	1.369
R2	0	1.369	0	1.369 (25)
R3	0	0	1.369	0.467
R4	1.369	0.467	1.134	0.232

因此,如果有非常快的对数运算,省去任选的测试是合算的,但如果对数子程序比较慢,则采用它们就比较合算。

但为什么它行得通呢?一个原因是我们可计算 $X \leq x$ 的概率,而且结果它是正确值(10)。但除非已碰巧触到了正确的“窍门”,不然这样一个计算是很不容易的,而且最好首先理解最初是如何发现这个算法的。Kinderman 和 Monahan 是通过解决下列理论导出它的,这一理论对于任何特性很好的密度函数 $f(x)$ 都可使用[见 *ACM Trans. Math. Software* 3 (1977), 257~260]。

一般说来,假设一个点 (U, V) 已在由

$$u > 0, \quad u^2 \leq g(v/u) \quad (26)$$

定义的 (u, v) 平面的区域上一致地生成,其中 g 是某个非负可积函数。如果置 $X \leftarrow V/U$, 在由(26)中的两个关系加上辅助条件 $v/u \leq x$ 所定义的区域上对 $du dv$ 积分,然后除以不带这额外条件的同样积分,可以计算 $X \leq x$ 的概率。令 $v = tu$, 使得 $dv = u dt$, 这个积分变成

$$\int_{-\infty}^x dt \int_0^{\sqrt{g(t)}} u du = \frac{1}{2} \int_{-\infty}^x g(t) dt$$

因此 $X \leq x$ 的概率是

$$\int_{-\infty}^x g(t) dt / \int_{-\infty}^{+\infty} g(t) dt \quad (27)$$

当 $g(t) = e^{-t^2/2}$ 时得出正态分布;而且条件 $u^2 \leq g(v/u)$ 在此情况下简化成 $(v/u)^2 \leq -4 \ln u$ 。容易看出,满足这一关系的所有 (u, v) 的集合整个地包含在图 13 的矩形中。

在步骤 R2 与 R3 中的界定义了具有较简单边界方程的内部与外部区域。对所有实数 x 成立的众所周知的不等式

$$e^x \geq 1 + x$$

可用来证明,对任何常数 $c > 0$,

$$1 + \ln c - cu \leq -\ln u \leq 1/(cu) - 1 + \ln c \quad (28)$$

习题 21 证明 $c = e^{1/4}$ 是用于步骤 R2 的最好常数。在步骤 R3 中情况要复杂,而且在此情况下似乎没有一个最优的 c 的简单表达式,但是计算实践表明对于 R3 的最好值近似于 $e^{1/35}$ 。当 $u = 1/c$ 时,近似曲线(28)与真正的边界相切。

通过把区域划分成小区域有可能得到一个更快的方法,大多数这些小区域可以

更快地处理。当然,这意味着,如同在算法 M 和 F 中一样,需要辅助表。Ahrens 和 Dieter 在 CACM 31 (1988), 1330~1337 上已经提出了要求较少的辅助表表项的一个有趣的替代方法。

5) 由正态离差出发的正态离差。习题 31 讨论了一个有趣的方法,通过直接地以正态离差进行工作,而不是把每件事情都基于一致离差,它可节省时间。这个由 C.S. Wallace 于 1996 年引入的方法在现时有相对较少的理论支持,但它成功地通过了一些经验检验。

6) 正态分布的变形。至今我们考虑了均值为 0 和标准差为 1 的正态分布。如果 X 有此分布,则

$$Y = \mu + \sigma X \quad (29)$$

有均值为 μ 和标准差为 σ 的正态分布。其次,如果 X_1 和 X_2 是具有均值为 0 和标准差为 1 的独立的正态离差,而且如果

$$Y_1 = \mu_1 + \sigma_1 X, \quad Y_2 = \mu_2 + \sigma_2(\rho X_1 + \sqrt{1-\rho^2} X_2) \quad (30)$$

则 Y_1 和 Y_2 是相关随机变量,它们有正态分布,且均值为 μ_1, μ_2 , 标准差为 σ_1, σ_2 , 相关系数为 ρ (关于对 n 个变量的推广,见习题 13)。

D. 指数分布 在一致离差和正态离差之后,下一个重要的随机量是指数离差。这样的数出现于“到达时间”情况中。例如,如果一个放射体以某种速度放出 α 粒子,它平均每 μ 秒钟放出一个粒子,则在两次连续的放射之间的时间有指数分布且均值为 μ 。这个分布由公式

$$F(x) = 1 - e^{-x/\mu}, \quad x \geq 0 \quad (31)$$

定义。

1) 对数方法。显然,如果 $y = F(x) = 1 - e^{-x/\mu}$, 则 $x = F^{[-1]}(y) = -\mu \ln(1-y)$ 。因此由等式(7), $-\mu \ln(1-U)$ 有指数分布。由于当 U 是一致分布时, $1-U$ 也是一致分布,我们的结论是:

$$X = -\mu \ln U \quad (32)$$

是均值为 μ 的指数分布。($U=0$ 的情况必须特别地处理,我们可以以任何方便的值 ϵ 代替 0,因为这个情况的概率是极其小的。)

2) 随机极小化方法。在算法 F 中,我们看到有一些计算一致离差的对数的简单而快速的方法。下边特别有效的方法是由 G. Marsaglia, Sibuya 和 J. H. Ahrens 建立的[见 CACM 15 (1972), 876~877]:

算法 S(具有均值 μ 的指数分布) 使用具有 $(t+1)$ 个二进位精度的一致离差,本算法在一台二进计算机上产生指数离差。常数

$$Q[k] = \frac{\ln 2}{1!} + \frac{(\ln 2)^2}{2!} + \dots + \frac{(\ln 2)^k}{k!}, \quad k \geq 1 \quad (33)$$

应预先算出,展开直到 $Q[k] > 1 - 2^{-t}$ 为止。

S1.[得到 U 和移位] 生成一个 $(t+1)$ 位一致随机二进分数 $U = (.b_1 b_2 \dots$

$b_i)_2$;找出头一个0位 b_j ,并移走前导 $j+1$ 位,置 $U \leftarrow (.b_{j+1} \cdots b_i)_2$ 。(像在算法 F 中一样,被抛弃的二进位的平均个数为2。)

S2. [立即接受?] 如果 $U < \ln 2$,置 $X \leftarrow \mu(j \ln 2 + U)$ 并结束此算法(注意 $Q[1] = \ln 2$)。

S3. [极小值] 求使得 $U < Q[k]$ 的最小的大于等于2的 k ,生成 k 个新的一致离差 U_1, \dots, U_k ,并置 $V \leftarrow \min(U_1, \dots, U_k)$ 。

S4. [派送答案] 置 $X \leftarrow \mu(j + V) \ln 2$ 。 ▮

生成指数离差也可使用另一些方法(例如,如同在算法 R 中的一致比例方法)。

E. 其它连续分布 现在我们简略地考虑如何处理在实践中相当经常地出现的某些其它分布。

1) 阶 $a > 0$ 的伽玛分布由

$$F(x) = \frac{1}{\Gamma(a)} \int_0^x t^{a-1} e^{-t} dt, \quad x \geq 0 \quad (34)$$

定义。当 $a=1$ 时,这是具有均值1的指数分布;当 $a = \frac{1}{2}$ 时,它是 $\frac{1}{2} Z^2$ 的分布,其中 Z 有正态分布(均值0,方差1)。如果 X 和 Y 分别是阶为 a 和 b 的独立伽玛分布随机变量,则 $X+Y$ 有阶为 $a+b$ 的伽玛分布。于是,例如,具有均值1的 k 个独立指数离差之和有阶为 k 的伽玛分布。如使用对数方法(32)来生成这些指数离差,则我们仅仅需要计算一个对数: $X \leftarrow -\ln(U_1 \cdots U_k)$, 其中 U_1, \dots, U_k 是非零一致离差。这一技术处理所有的整数阶 a ;为完成这个描述,对于 $0 < a < 1$ 的一个适当方法,可在习题16中找到。

当 a 很大时,简单的对数方法太慢,因为它要求 $\lfloor a \rfloor$ 个一致离差。而且有一个相当大的风险,即乘积 $U_1 \cdots U_{\lfloor a \rfloor}$ 将引起浮点溢出。对于很大的 a ,下列由 J. H. Ahrens 给出的算法是相当有效的,而且很容易借助标准子程序来写[见 Ann. Inst. Stat. Math. 13 (1962), 231~237]。

算法 A(阶 $a > 1$ 的伽玛分布)

A1. [生成候选者] 置 $Y \leftarrow \tan(\pi U)$, 其中 U 是一个一致离差,而且置 $X \leftarrow \sqrt{2a-1} Y + a - 1$ 。(代替 $\tan(\pi U)$ 我们可以使用一个配极方法,如同在算法 P 的步骤 P4 中那样计算比值 V_2/V_1 。)

A2. [接受?] 若 $X \leq 0$,则返回 A1。否则生成一个一致离差 V ,并且若 $V > (1 + Y^2) \exp((a-1) \ln(X/(a-1)) - \sqrt{2a-1} Y)$ 就返回 A1。否则接受 X 。

▮

当 $a \geq 3$ 时,步骤 A1 被执行的平均次数小于 1.902。

基于以下值得注意的事实,即伽玛离差近似地等于 aX^3 ,其中 X 是有均值 $1 -$

$1/(9a)$ 和标准差 $1/\sqrt{9a}$ 的正态分布,对于很大的 a ,也有一个吸引人的方法;请见 E. B. Wilson 和 M. M. Hilferty, *Proc. Nat. Acad. Sci.* **17** (1931), 684 ~ 688; G. Marsaglia, *Computers and Math.* **3** (1977), 321 ~ 325.*

关于稍微复杂点但要快得多的算法,请见 J. H. Ahrens 和 V. Dieter 的论文, *CACM* **25** (1982), 47 ~ 54。这个算法生成伽玛离差的速度大约是生成正态离差的两倍。这篇论文包含了用于构造这个算法的设计原理的有启发性的讨论。

2) 带有正参数 a 和 b 的贝塔分布由

$$F(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{a-1}(1-t)^{b-1} dt, \quad 0 \leq x \leq 1 \quad (35)$$

定义。令 X_1 和 X_2 分别为阶 a 和阶 b 的独立的伽玛离差,并置 $X \leftarrow X_1/(X_1 + X_2)$ 。另一个对于小的 a 和 b 有用的方法,是反复地置 $Y_1 \leftarrow U_1^{1/a}$ 和 $Y_2 \leftarrow U_2^{1/b}$ 直到 $Y_1 + Y_2 \leq 1$ 为止;然后 $X \leftarrow Y_1/(Y_1 + Y_2)$ [见 M. D. Jöhrnk, *Metrika* **8** (1964), 5 ~ 15]。如果 a 和 b 是整数(不太大),则还有另外一个方法,即置 X 成为 $a + b - 1$ 个独立一致离差中的第 b 个最大者(参考第 5 章开始的习题 9)。也见 R. C. H. Cheng 所描述的直接方法, *CACM* **21** (1978), 317 ~ 322。

3) 具有 ν 个自由度的 χ^2 分布(等式 3.3.1 - (22))是通过置 $X \leftarrow 2Y$ 得到的,其中 Y 是阶为 $\nu/2$ 的伽玛分布。

4) 自由度为 ν_1 和 ν_2 的 F 分布(方差比分布)由

$$F(x) = \frac{\nu_1^{\nu_1/2} \nu_2^{\nu_2/2} \Gamma((\nu_1 + \nu_2)/2)}{\Gamma(\nu_1/2) \Gamma(\nu_2/2)} \int_0^x t^{\nu_1/2-1} (\nu_2 + \nu_1 t)^{-\nu_1/2-\nu_2/2} dt \quad (36)$$

所定义,其中 $x \geq 0$ 。设 Y_1 和 Y_2 是独立的且其自由度分别为 ν_1 和 ν_2 的 χ^2 分布;置 $X \leftarrow Y_1 \nu_2 / Y_2 \nu_1$, 或者置 $X \leftarrow \nu_2 Y / \nu_1 (1 - Y)$, 其中 Y 是具有参数 $\nu_1/2$ 和 $\nu_2/2$ 的贝塔变量。

5) 具有 ν 个自由度的 t 分布是由

$$F(x) = \frac{\Gamma((\nu+1)/2)}{\sqrt{\pi\nu} \Gamma(\nu/2)} \int_{-\infty}^x (1+t^2/\nu)^{-(\nu+1)/2} dt \quad (37)$$

定义的。设 Y_1 是一个正态离差(均值 0, 方差 1), 并设 Y_2 是与 Y_1 无关, 并有 ν 个自由度的 χ^2 分布;置 $X \leftarrow Y_1/\sqrt{Y_2/\nu}$ 。或者当 $\nu > 2$ 时, 设 Y_1 是一个正态离差并设 Y_2 独立地有均值 $2/(\nu-2)$ 的指数分布;置 $Z \leftarrow Y_1^2/(\nu-2)$ 并且如果 $e^{-Y_2-Z} \geq 1 - Z$, 就拒绝 (Y_1, Y_2) , 否则置 $X \leftarrow Y_1/\sqrt{(1-2Z/\nu)(1-Z)}$ 。后一方法是 G. Marsaglia 给出的, 见 *Math. Comp.* **34** (1980), 235 ~ 236。[也见 A. J. Kinderman, J. F. Monahan 和 J. G. Ramage, *Math. Comp.* **31** (1977), 1009 ~ 1018。]

6) 半径为 1 的 n 维球体上的随机点。设 X_1, X_2, \dots, X_n 是独立的正态离差(均值 0, 方差 1), 在这单位球上所寻求的点是

* 在 323 页上的算法中, 把“ $+(3a-1)$ ”改成“ $-(3a-1)$ ”。——原注

$$(X_1/r, X_2/r, \dots, X_n/r), \quad \text{其中 } r = \sqrt{X_1^2 + X_2^2 + \dots + X_n^2} \quad (38)$$

如果利用配极方法,即算法 P,计算诸 X ,我们每次计算两个独立的 X ,而且在该算法的记号下,有 $X_1^2 + X_2^2 = -2 \ln S$;这就节省了一点为计算 r 所需要的时间。(38)的正确性是从这样一个事实得来的,即点 (X_1, \dots, X_n) 的分布函数的密度仅仅依赖于它和原点的距离,所以当把它投影到单位球上时,它有一致分布。这个方法首先是由 G. W. Brown 提出的,见 *Modern Mathematics for the Engineer*, 第 1 辑, E. F. Beckenbach 编辑 (New York: McGraw-Hill, 1956), 302。为得到在 n 维球内的一个随机点, R. P. Brent 提出取曲而上的一点,并把它乘以 $U^{1/n}$ 。

在三维中,可以使用一个简单得多的方法,因为每个个别的坐标在 -1 和 $+1$ 之间一致分布:通过算法 P 的步骤 P1 ~ P3 找到 V_1, V_2 和 S ,则所求的球面上的随机点是 $(\alpha V_1, \alpha V_2, 2S - 1)$, 其中 $\alpha = 2 \sqrt{1 - S}$ 。[Robert. E. Knop, CACM 13 (1970), 326。]

F. 重要的整值分布 仅由整值组成的一个概率分布实质上可由本小节开始时所描述的技术来加以处理;但这些分布中有些在实践中十分重要,因此值得在这里特别提及。

1) 几何分布。如果某个事件以概率 p 出现,则这一事件的出现之间(或直到这一事件头一次出现之前)所需的独立试验数 N 有几何分布。我们有 $N=1$ 的概率是 p , $N=2$ 的概率是 $(1-p)p$, \dots , $N=n$ 的概率是 $(1-p)^{n-1}p$ 。这实质上是 3.3.2 小节的间隔检验中已经考虑过的状况;它也直接同这小节的算法的某些循环被执行的次数有关,例如配极方法步骤 P1 ~ P3 的执行次数。

生成具有这种分布的一个变量的一个方便方法是置

$$N \leftarrow \lceil \ln U / \ln(1-p) \rceil \quad (39)$$

为校验这一公式,我们发现 $\lceil \ln U / \ln(1-p) \rceil = n$, 当且仅当 $n-1 < \ln U / \ln(1-p) \leq n$, 即 $(1-p)^{n-1} > U \geq (1-p)^n$, 而这发生的概率如同我们要求的那样,为 $p(1-p)^{n-1}$ 。量 $\ln U$ 可以任意地以 $-Y$ 代替,其中 Y 有均值为 1 的指数分布。

$p = \frac{1}{2}$ 的特殊情况在一台二进计算机上可以非常简单地处理,因为公式(39)简化成置 $N \leftarrow \lceil -\lg U \rceil$;即 N 比 U 的二进制表示中的前导 0 位个数大 1。

2) 二项分布 (t, p) 。如果某个事件以概率 p 出现,且我们进行 t 次独立的试验,则事件出现的总数 N 等于 n 的概率为 $\binom{t}{n} p^n (1-p)^{t-n}$ (见 1.2.10 小节)。换言之,如果我们生成 U_1, \dots, U_t , 我们要计算这些当中有多少小于 p 。对于小的 t , 以这方式恰能得到 N 。

对于很大的 t , 我们可以以整参数 a 和 b 生成一个贝塔变量 X , 其中 $a+b-1=t$; 这样可以有效地给出 t 个元素的第 b 个最大者,省去了生成其它元素的麻烦。现在如果 $X \geq p$, 我们就置 $N \leftarrow N_1$, 其中 N_1 有二项分布 $(a-1, p/X)$, 因为这告诉我

们在范围 $[0, X)$ 中的 $a-1$ 个随机数中有多少是小于 p 的; 而如果 $X < p$, 我们就置 $N \leftarrow a + N_1$, 其中 N_1 有二项分布 $(b-1, (p-X)/(1-X))$, 因为 N_1 告诉我们在 $[X, 1)$ 的范围内的 $b-1$ 个随机数中有多少小于 p 。通过选择 $a = 1 + \lfloor t/2 \rfloor$, 在大约 $\lg t$ 次这类约简之后, 参数 t 将被减少到相当的大小。(这一方法是由 J. H. Ahrens 给出的, 他还提出对于中等大小的 t 的一个替换方案, 见习题 27。)

3) 具有均值 μ 的泊松分布 这个分布同指数分布有关, 就如二项分布同几何分布有关一样: 它表示在任何时刻都可能出现的一个事件在每个单位时间里出现的次数。例如, 在一秒钟的时间里, 由放射体放出的 α 粒子的个数是一个泊松分布。

根据这一原理, 我们可以首先生成具有均值 $1/\mu$ 的独立指数离差 X_1, X_2, \dots , 而且一旦 $X_1 + \dots + X_m \geq 1$ 即停止; 然后令 $N \leftarrow m-1$, 即得到一个泊松离差。 $X_1 + \dots + X_m \geq 1$ 的概率是阶为 m 的一个伽玛离差 $\geq \mu$ 的概率, 亦即 $\int_{\mu}^{\infty} t^{m-1} e^{-t} dt / (m-1)!$; 因此 $N = n$ 的概率是

$$\frac{1}{n!} \int_{\mu}^{\infty} t^n e^{-t} dt = \frac{1}{(n-1)!} \int_{\mu}^{\infty} t^{n-1} e^{-t} dt = e^{-\mu} \frac{\mu^n}{n!}, \quad n \geq 0 \quad (40)$$

如果我们通过对数方法生成指数离差, 则上面的方法告诉我们当 $-(\ln U_1 + \dots + \ln U_m)/\mu \geq 1$ 时停止。简化这个表达式, 即可看到通过计算 $e^{-\mu}$, 把它换成定点表示, 然后生成一个或多个一致离差 U_1, U_2, \dots 直到乘积满足 $U_1 \cdots U_m \leq e^{-\mu}$, 最后置 $N \leftarrow m-1$, 即得到所需的泊松离差。平均来说, 这要求生成 $\mu+1$ 个一致离差, 所以当 μ 不太大时, 它是非常有用的方法。

我们知道如何处理很大阶的伽玛分布和二项式分布, 利用这一点, 当 μ 很大时我们可以得到阶为 $\log \mu$ 的一个方法。首先生成 X , 它具有阶 $m = \lfloor \alpha \mu \rfloor$ 的伽玛分布, 其中 α 是一个适当的常数。(因为 X 等价于 $-\ln(U_1 \cdots U_m)$, 我们实质上绕过了前面方法中的 m 步。) 如果 $X < \mu$, 则置 $N \leftarrow m + N_1$, 其中 N_1 是具有均值 $\mu - X$ 的一个泊松离差; 如果 $X \geq \mu$, 则置 $N \leftarrow N_1$, 其中 N_1 有二项分布 $(m-1, \mu/X)$ 。这个方法是由 J. H. Ahrens 和 U. Dieter 给出的, 他们的实验说明, $7/8$ 是对 α 的一个好的选择。

当 $X \geq \mu$ 时, 上述约简的正确性是下列重要原理的一个推论: “设 X_1, \dots, X_m 是具有相同均值的独立指数离差; 对于 $1 \leq j \leq m$, 设 $S_j = X_1 + \dots + X_j$, 且设 $V_j = S_j/S_m$ 。则 V_1, V_2, \dots, V_{m-1} 的分布和排成递增次序的 $m-1$ 个独立的一致离差的分布一样。”为了形式地建立这个原理, 给定 $S_m = s$ 的值, 我们对任意的值 $0 \leq v_1 \leq \dots \leq v_{m-1} \leq 1$, 计算 $V_1 \leq v_1, \dots, V_{m-1} \leq v_{m-1}$ 的概率: 设 $f(v_1, v_2, \dots, v_{m-1})$ 是 $m-1$ 重积分

$$\int_0^{v_1 s} \mu e^{-t_1/\mu} dt_1 \int_0^{v_2 s - t_1} \mu e^{-t_2/\mu} dt_2 \cdots \times \\ \int_0^{v_{m-1} s - t_1 - \dots - t_{m-2}} \mu e^{-t_{m-1}/\mu} dt_{m-1} \cdot \mu e^{-(s - t_1 - t_2 - \dots - t_{m-1})/\mu}$$

则通过替换 $t_1 = su_1, t_1 + t_2 = su_2, \dots, t_1 + \dots + t_{m-1} = su_{m-1}$, 有

$$\frac{f(v_1, v_2, \dots, v_{m-1})}{f(1, 1, \dots, 1)} = \frac{\int_0^{v_1} du_1 \int_{u_1}^{v_2} du_2 \dots \int_{u_{m-2}}^{v_{m-1}} du_{m-1}}{\int_0^1 du_1 \int_{u_1}^1 du_2 \dots \int_{u_{m-2}}^1 du_{m-1}}$$

后边的比是在 $U_1 \leq \dots \leq U_{m-1}$ 的条件下, 一致离差 U_1, \dots, U_{m-1} 满足 $U_1 \leq v_1, \dots, U_{m-1} \leq v_{m-1}$ 的对应概率。

习题 22 概述了一项更有效的但稍微更复杂的关于二项式和泊松离差的技术。

G. 更多文献 1947 年 5 月 21 日, John von Neumann 所写的一封信的传真, 出现在 Los Alamos Science 特刊 *Stanislaw Ulam 1909 - 1984* 上 (Los Alamos National Lab., 1987), 135 ~ 136。就在这封信中, 拒绝方法问世。由 L. Devroye 所写的书 *Non-Uniform Random Variate Generation* (Springer, 1986) 讨论了更多的有不一致分布的随机变量的生成算法, 以及每种技术在典型计算机上的有效性的详细研究。

W. Hörmann 和 G. Derflinger [ACM Trans. Math. Software **19** (1993), 489 ~ 495] 指出, 在同有小的乘数 $a \approx \sqrt{m}$ 的线性同余生成程序相关联时, 使用拒绝方法可能是危险的。

从理论观点来看, 在使用可能的最小个数的随机二进位产生所要求的结果这一意义下, 考虑生成具有给定分布的随变量的最优方法, 是有益的。关于处理这样一些问题的一个理论的雏形, 请见 D. E. Knuth 和 A. C. Yao 著, J. F. Traub 编辑, *Algorithms and Complexity* (New York: Academic Press, 1976), 357 ~ 428。

作为对本小节许多技术的复习, 推荐做一做习题 16。

习 题

1. [10] 如果 α 和 β 是实数, 且 $\alpha < \beta$, 你将如何生成在 α 和 β 之间一致分布的一个随机实数?

2. [M16] 假定 mU 是 0 和 $m-1$ 之间的一个随机整数, 如果 $0 \leq r < k$, 问 $[kU] = r$ 的精确概率是什么? 把这同期望的概率 $1/k$ 进行比较。

► 3. [14] 如果把 U 当做一个整数并计算其模 k 剩余, 以得到 0 和 $k-1$ 之间的一个随机整数, 而不是像正文中所建议那样做乘法。这样 (1) 将变为

```

ENTA 0
LDX   U
DIV   K

```

且结果出现于寄存器 X 中。这是一个好方法吗?

4. [M20] 证明 (8) 中的两个关系。

► 5. [21] 提出一个有效方法计算具有分布 $px + qx^2 + rx^3$ 的一个随机变量, 其中 $p \geq 0, q \geq 0, r \geq 0$ 且 $p + q + r = 1$ 。

6. [HM21] 通过下列方法来计算量 X :

步骤 1. 生成两个独立一致离差 U 和 V 。

步骤 2. 如果 $U^2 + V^2 \geq 1$, 则返回步骤 1, 否则置 $X \leftarrow U$ 。

什么是 X 的分布函数? 步骤 1 将被执行多少次? (给出均值和标准差。)

►7. [20] (A. J. Walker) 假设我们有一堆 k 种不同颜色的立方体, 比方说, 对于 $1 \leq j \leq k$, 有 n_j 个颜色 C_j 的立方体, 同时还有 k 个盒子 B_1, \dots, B_k , 它们每个恰可装 n 个立方体。而且 $n_1 + \dots + n_k = kn$, 所以这些立方体正好装得进这些盒子中。试(构造性地)证明总有一个方法来把这些立方体放进这些盒子中, 使得每个盒子中最多有两种不同颜色; 事实上, 存在一种办法使得每当盒子 B_j 含两个颜色时, 颜色之一为 C_j 。给定一个概率分布 (p_1, \dots, p_k) , 说明如何使用这个原理来计算(3)中所需求的 P 和 Y 表。

8. [M15] 证明: 运算(3)可以改为运算

如果 $U < P_K$, 则 $X \leftarrow r_{K+1}$, 否则 $X \leftarrow Y_K$

(即用 U 原来的值代替 V), 如果通过适当更改 P_0, P_1, \dots, P_{r-1} 能更方便地实现后者的话。

9. [HM10] 为什么对于 $x < 1$, 图 9 的曲线 $f(x)$ 向下凹, 而对于 $x > 1$ 却向上凸?

►10. [HM24] 说明怎样计算辅助常数 $P_j, Q_j, Y_j, Z_j, S_j, D_j, E_j$, 使得算法 M 提供具有正确分布的答案。

►11. [HM27] 证明算法 M 的步骤 M7~M8 生成一个随机变量, 它具有正态分布的适当尾形; 换句话说, $X \leq x$ 的概率应精确地为

$$\int_3^x e^{-t^2/2} dt \Big/ \int_3^\infty e^{-t^2/2} dt, \quad x \geq 3$$

[提示: 证明它是拒绝方法对于某个 C , 满足 $g(t) = Cte^{-t^2/2}$ 的一个特殊情况。]

12. [HM23] (R. P. Brent) 证明(23)中定义的数 a_j 满足关系 $a_j^2 - a_{j-1}^2 < 2 \ln 2$ (对于所有 $j \geq 1$)。

[提示: 若 $f(x) = e^{x^2/2} \int_x^\infty e^{-t^2/2} dt$, 证明对于 $0 \leq x < y$, $f(x) > f(y)$ 。]

13. [HM25] 给定一组 n 个独立正态离差 X_1, X_2, \dots, X_n , 且均值为 0 和方差为 1, 试说明如何求常数 b_j 和 a_{ij} , $1 \leq j \leq i \leq n$, 使得如果

$$Y_1 = b_1 + a_{11}X_1, \quad Y_2 = b_2 + a_{21}X_1 + a_{22}X_2, \dots, Y_n = b_n + a_{n1}X_1 + \dots + a_{nn}X_n$$

则 Y_1, Y_2, \dots, Y_n 是相关正态分布变量, Y_i 有均值 μ_i , 且诸 Y 有一给定的协方差矩阵 (c_{ij}) 。(Y_i 和 Y_j 的协方差 c_{ij} 定义为 $(Y_i - \mu_i)(Y_j - \mu_j)$ 的平均值。特别是, c_{ij} 是 Y_j 的方差, 即它的标准差的平方。并非所有矩阵 (c_{ij}) 都可以是协方差矩阵, 因而每当对于给定条件可能存在一个解时, 你的构造才能有效。)

14. [M21] 如果 X 是具有连续分布 $F(x)$ 的一个随机变量, 而且如果 c 是一个(可能为负的)常数, 问 cX 的分布是什么?

15. [HM21] 如果 X_1 和 X_2 是分别具有分布 $F_1(x)$ 和 $F_2(x)$, 且具有密度 $f_1(x) = F'_1(x)$, $f_2(x) = F'_2(x)$ 的独立随机变量, 量 $X_1 + X_2$ 的分布和密度函数是什么?

►16. [HM22] (J. H. Ahrens) 利用当 $0 < t < 1$ 时为 $cg(t) = t^{a-1}/\Gamma(a)$, 当 $t \geq 1$ 时为 $cg(t) = e^{-t}/\Gamma(a)$ 的拒绝方法建立求阶为 a (当 $0 < a \leq 1$ 时) 的伽玛离差的一个算法。

►17. [M24] 什么是具有概率 p 的几何分布的分布函数 $F(x)$? 什么是生成函数 $G(x)$? 什么是这个分布的均值和标准差?

18. [M24] 提出一个方法来计算一个随机整数 N , 对于这一方法而言, N 以 $np^2 \times (1-p)^{n-1}$ ($n \geq 0$) 的概率取值 n 。(特别有趣的情况是当 p 比较小时。)

19. [22] 负的二项分布 (t, p) 有整数值 $N = n$ 的概率为 $\binom{t-1+n}{n} p^t (1-p)^n$. (不像通常的二项分布, t 不必是一个整数, 因为当 $t > 0$ 时, 这个量对所有 n 都是非负的). 推广习题 18, 说明当 t 是小的正整数时, 如何生成具有这个分布的整数 N . 如果 $t = p = \frac{1}{2}$, 你将提出什么方法?

20. [M20] 设 A 是图 13 中的阴影区域内的面积, 并设 R 是包含它的矩形的面积. 设 I 是为步骤 R2 所接受的内部区域的面积, 而 E 是在步骤 R3 拒绝的外部区域和外部矩形之间的面积. 借助于 A, R, I 和 E , 试如 (25) 中那样对于它的四个变形的每一个, 确定算法 R 的每步被执行的次数.

21. [HM29] 导出计算习题 20 中定义的量 A, R, I 和 E 的公式 (对于 I 以及特别是 E , 你可以使用一个交互的计算机代数系统). 说明对于形如 " $X^2 \leq 4(1 + \ln c) - 4cU$ " 的测试, $c = e^{1/4}$ 是在步骤 R2 中最好的常数.

22. [HM40] 通过生成一个适当的正态离差, 以某种方便的方式把它转换成一个整数, 并偶尔应用一个 (可能很复杂的) 修正, 问能够得到对于很大的 μ 的精确泊松分布吗?

23. [HM23] (J. von Neumann) 下面两个生成随机量 X 的方法是等价的吗 (即量 X 是否有同样的分布)?

方法 1: 置 $X \leftarrow \sin((\pi/2)U)$, 其中 U 是一致的.

方法 2: 生成两个一致离差 U 和 V , 而且如果 $U^2 + V^2 \geq 1$, 则重复执行, 直到 $U^2 + V^2 < 1$.

然后置 $X \leftarrow |U^2 - V^2| / (U^2 + V^2)$.

24. [HM40] (S. Ulam, J. von Neumann) 设 V_0 是在 0 和 1 之间随机选择的一个实数, 通过规则 $V_{n+1} = 4V_n(1 - V_n)$ 定义序列 $\langle V_n \rangle$. 如果以完全的精度来进行这个计算, 则其结果将是具有分布 $\sin^2 \pi U$ 的一个序列, 其中 U 是一致的, 即具有分布函数 $F(x) = \int_0^x dx / \sqrt{2\pi x(1-x)}$. 因为如果我们令 $V_n = \sin^2 \pi U_n$, 即知 $U_{n+1} = (2U_n) \bmod 1$, 而且因为几乎所有的实数都有一个随机的二进展开 (见 3.5 节), 所以序列 U_n 是等分布的. 但如果 V_0 的计算仅以有限精度进行, 则上述论证就失败了, 因为我们很快就受到舍入误差的干扰 [von Neumann, *Collected Works* 5, 768 ~ 770].

当仅给出有限精度时, 试从经验上 (对于 V_0 的各种不同选择) 和从理论上分析上段定义的序列 $\langle V_n \rangle$. 这一序列是否有类似于预期的分布那样的分布?

25. [M25] 设 X_1, X_2, \dots, X_5 是二进字, 它们的每一个二进位都独立地以 $\frac{1}{2}$ 的概率为 0 或 1. 问什么是 $X_1 \vee (X_2 \wedge (X_3 \vee (X_4 \wedge X_5)))$ 的 - 给定位置含 1 的概率? 试推广之.

26. [M18] 设 N_1 和 N_2 是分别有均值 μ_1 和 μ_2 的独立的泊松离差, 其中 $\mu_1 > \mu_2 \geq 0$. 证明或否定: (a) $N_1 + N_2$ 有均值 $\mu_1 + \mu_2$ 的泊松分布. (b) $N_1 - N_2$ 有均值 $\mu_1 - \mu_2$ 的泊松分布.

27. [22] (J. H. Ahrens) 在大多数二进计算机上, 有一个有效的方法来计算一个二进字中 1 的个数 (参见 7.1 节). 因此当 $p = 1/2$ 时, 有一个很好的方法来得到二项分布 (t, p) , 只须生成 t 个随机二进位并计算 1 的个数就行.

试设计对于任意的 p 产生二项分布 (t, p) 的一个算法, 仅仅使用对于 $p = 1/2$ 的特殊情况的一个子程序作为随机数据源. [提示: 模拟一个过程, 该过程首先观察 t 个一致离差的最高有效位, 然后观察其前导的二进位是否小于 p 的那些离差中的第二个二进位, 等等.]

28. [HM35] (R. P. Brent) 试建立一个方法生成由 $\sum a_k x_k^2 = 1$ 定义的椭圆体曲面上的一个随

机点,其中 $a_1 \geq \dots \geq a_n > 0$ 。

29. [M20] (J. L. Bentley 和 J. B. Saxe) 试找出生成在 0 和 1 之间一致的 n 个数 X_1, \dots, X_n 的一个简单方法,但不要求它们是排了序的,即 $X_1 \leq \dots \leq X_n$ 这个事实,你的算法应当只花费 $O(n)$ 步。

30. [M30] 解释如何生成随机点 (X_j, Y_j) 集,使得如果 R 是单位方阵中包含的任何面积为 α 的矩形,则 R 中的 (X_j, Y_j) 数有均值 $\alpha\mu$ 的泊松分布。

31. [HM39] (正态离差的直接生成)

a) 证明如果 $a_1^2 + \dots + a_k^2 = 1$ 且如果 X_1, \dots, X_k 是有均值 0 和方差 1 的独立正态离差,则 $a_1 X_1 + \dots + a_k X_k$ 是有均值 0 和方差 1 的正态离差。

b) a) 的结果提示,如同我们从老的一致离差能得到新的一致离差一样,我们能从老的正态离差得到新的正态离差。例如,在一组正态离差 X_0, \dots, X_{54} 已初始地被计算之后,我们可以使用 3.2.2-(7) 的思想,但通过像下面这样的递推式:

$$X_n = (X_{n-24} + X_{n-55})/\sqrt{2} \quad \text{或} \quad X_n = \frac{3}{5} X_{n-24} + \frac{4}{5} X_{n-55}$$

试说明为什么这不是一个好想法。

c) 然而,试说明,有一个适当的方法,通过使用 a) 和 b) 中的思想的一个改进,可以从其它的正态离差迅速地生成正态离差。[提示:如果 X 和 Y 是独立正态离差,则对于任何角度 θ , $X' = X \cos \theta + Y \sin \theta$ 以及 $Y' = -X \sin \theta + Y \cos \theta$ 也是。]

32. [HM30] (C. S. Wallace) 设 X 和 Y 是具有均值 1 的独立指数离差。试证明,如果以下列任何方式从 X 和 Y 得到 X' 和 Y' ,则类似地 X' 和 Y' 是具有均值 1 的独立指数离差:

a) 给定 $0 < \lambda < 1$,

$$X' = (1-\lambda)X - \lambda Y + (X+Y)[(1-\lambda)X < \lambda Y], \quad Y' = X + Y - X'$$

$$b) (X', Y') = \begin{cases} (2X, Y-X), & \text{如果 } X \leq Y \\ (2Y, X-Y), & \text{如果 } X > Y \end{cases}$$

c) 如果在二进制号下, $X = (\dots x_2 x_1 x_0 \cdot x_{-1} x_{-2} x_{-3} \dots)_2$ 和 $Y = (\dots y_2 y_1 y_0 \cdot y_{-1} y_{-2} y_{-3} \dots)_2$, 则 X' 和 Y' 有“洗过了”的值

$$X' = (\dots x_2 y_1 x_0 \cdot y_{-1} x_{-2} y_{-3} \dots)_2 \quad \text{和} \quad Y' = (\dots y_2 x_1 y_0 \cdot x_{-1} y_{-2} x_{-3} \dots)_2$$

33. [20] 通过消耗未知其个数的一致随机变量 U_1, U_2, \dots , 算法 P, M, F 和 R 生成正态离差, 如何能修改它们使得输出仅是一个 U 的函数?

3.4.2 随机抽样和洗牌

许多数据处理应用都要求从含有 N 个记录的一个文件中随机地无偏倚地选择 n 个记录。例如,这个问题出现于需要有抽样的质量控制或其它统计计算中。通常 N 非常大,以致不可能一次在存储器中存下所有数据;而且个别的记录本身通常也非常大,因此我们甚至不可能在内存中容下 n 个记录。因而我们要寻求一个有效的选择 n 个记录的过程,通过这一过程,当每个记录到来时,判断是接受还是拒绝它,并把所接受的记录写到一个输出文件上。

对于这个问题已经想出了许多方法。最明显的方法是以概率 n/N 来选择每个记录;这有时可能是合适的,但它仅仅能使抽样中的记录个数平均为 n 。标准差是

$\sqrt{n(1-n/N)}$, 而且样本可能或者对于我们所希望的应用来说太大, 或者对于给出必需的结果来说又太小了。

幸而, 只需对这个“显然”的过程做一简单的修正即可给出所需要的结果: 如果已经选择了 m 个项目, 则对第 $t+1$ 个记录应以概率 $(n-m)/(N-t)$ 加以选择。这是适当的概率, 因为在从 N 个事物中选择 n 个并使 m 个值出现在头 t 个当中的所有可能的方式中恰有

$$\binom{N-t-1}{n-m-1} / \binom{N-t}{n-m} = \frac{n-m}{N-t} \quad (1)$$

种方式选择第 $t+1$ 个元素。

在上一段中提出的思想立即引出了下列算法:

算法 S(选择抽样技术) 从 N 个记录的一个集合中随机地选择 n 个记录, 其中 $0 < n \leq N$ 。

S1. [初始化] 置 $t \leftarrow 0, m \leftarrow 0$ 。(在本算法中, m 表示已选得的记录数, 而 t 表示我们已经处理过的输入记录的总数。)

S2. [生成 U] 生成在 0 与 1 之间一致地分布的一个随机数 U 。

S3. [检验] 如果 $(N-t)U \geq n-m$, 则转到步骤 S5。

S4. [选择] 把下一个记录选为样本, m 和 t 加 1。如果 $m < n$, 则转到步骤 S2; 否则取样完成, 算法终止。

S5. [跳] 跳过下一个记录(不把它选为样本), t 加 1, 并转到步骤 S2。|

乍一看, 这个算法可能显得不可靠而且简直是不正确的; 但经过细心的分析(见下面的习题), 证明它是完全信赖的。不难验证:

a) 至多输入 N 个记录(在选择 n 个项目之前, 我们绝不会跑出文件的末尾)。

b) 这个抽样是完全无偏倚的; 特别是, 任何给定的元素(例如文件的最后元素)被选择的概率是 n/N 。

虽然有这样的事实: 即我们不以概率 n/N , 而是以等式(1)中的概率选择第 $t+1$ 项, 但命题 b) 是真的! 这在发表的著作中引起了某些混乱。读者能否说明这种表面上的矛盾?

(注: 当使用算法 S 时, 每次运行程序时, 都应该小心使用随机数 U 的不同来源, 以消除不同时候得到的抽样之间的联系。例如, 这可以通过每次使用线性同余方法选择不同的 X_0 值来实现。种子值 X_0 可以赋值为当前的日期, 或者赋值成上次运行该程序时生成的最后的 X 值。)

我们通常将不扫描所有 N 个记录; 事实上, 因为上边的 b) 指出最后的记录是以概率 n/N 被选择的, 故在恰考虑最后记录 $(1-n/N)$ 时机之前, 就已终止这个算法了。当 $n=2$ 时所考虑的记录平均数大约是 $\frac{2}{3}N$, 习题 5 和 6 给出了一般的公式。

C. T. Fan, Mervin E. Muller 和 Ivan Rezucha 在一篇文章中讨论了算法 S 和一些其它的抽样技术, 见 *J. Amer. Stat. Assoc.* 57(1962), 387~402。这个方法独立地为

T. G. Jones 所发现, 见 CACM 5 (1962), 343。

如果我们预先不知道 N 的值, 就出现问题了, 因为在算法 S 中 N 的准确值是至关重要的。假设我们要从一个文件中随机选择 n 项, 但不知道在该文件中恰有多少项, 我们可以首先扫描并计算这些记录, 然后用第二遍扫描来选择它们; 但是一般来说更好的是在第一遍扫描时就抽取 $m \geq n$ 个原始项目, 其中 m 比 N 小得多, 所以在第二遍扫描时只需考虑 m 项。当然, 以这样一种途径来做这项工作的技巧是, 要使最后的结果是原始文件的一个真正随机的抽样。

因为我们不知道输入何时结束, 故必须记住迄今所看到的输入记录的一个随机抽样, 这样随时都可以结束。当读输入时, 我们将构造 m 个记录的一个“水库”(储蓄器), 这个水库只含在前边的抽样中已出现过的记录。头 n 个记录总是进入水库中。当输入第 $t+1$ 个记录而 $t \geq n$ 时, 我们将在内存中有 n 个索引的一张表, 它们指向水库中属于我们从头 t 个记录中已经选定为随机抽样的那些记录。问题就成了当 t 增 1 时仍要维持这一状态, 就是说从现在已知存在的 $t+1$ 个记录当中选择一个新的随机抽样。不难看出, 我们应当以 $n/(t+1)$ 的概率来把新记录包括到新抽样当中, 而且在这种情况下, 应该去掉以前抽样中的一个随机元素。

因此, 下列过程完成这一工作。

算法 R(水库抽样) 给定 $n > 0$, 从其确切大小未知但大于等于 n 的一个文件中随机选择 n 个记录。一个称做“水库”的辅助文件存放有作为最后抽样的候选者的所有记录。本算法使用具有不同索引 $I[j]$ 的一张表, 其中 $1 \leq j \leq n$, 每一个索引指向水库的一个记录。

- R1. [初始化]** 输入头 n 个记录, 并把它们复制到水库中。对于 $1 \leq j \leq n$, 置 $I[j] \leftarrow j$, 并置 $t \leftarrow m \leftarrow n$ 。(如果被抽样文件少于 n 个记录, 当然有必要中断算法并报告失败。在算法运行期间, 索引 $I[1], \dots, I[n]$ 指向当前抽样中的记录; m 是水库的大小; 而 t 是迄今已处理过的输入记录数。)
- R2. [文件结束?]** 如果已无记录输入, 则转到步骤 R6。
- R3. [生成并检验]** t 增 1, 然后生成 1 和 t (含 t) 之间的一个随机整数 M 。如果 $M > n$, 则转到 R5。
- R4. [加到水库中]** 复制输入文件的下一记录到水库中, m 增 1 并置 $I[M] \leftarrow m$ 。(以前由 $I[M]$ 指示的记录现在在抽样中为新的记录所代替。) 转到 R2。
- R5. [跳]** 跳过输入文件的下一记录(不把它包括在水库中), 并返回步骤 R2。
- R6. [第二次扫描]** 对 I 表的项进行排序使得 $I[1] < \dots < I[n]$; 然后扫描水库, 并把具有这些索引的记录复制到保存最后抽样的输出文件中。 ▮

算法 R 是由 Alan G. Waterman 给出的, 读者可以把习题 9 中运行该算法的例子做一遍。

如果诸记录充分短, 当然完全没有必要“建水库”, 我们可以在所有时候都把当前抽样的 n 个记录存放在内存中, 因而算法就变得简单得多(见习题 10)。

关于算法 R,我们自然要提这样的问题:“水库预期的大小要多大?”习题 11 表明 m 的平均值恰是 $n(1 + H_N - H_n)$;这近似于 $n(1 + \ln(N/n))$ 。所以如果 $N/n = 1000$,这水库将仅仅包含原来文件的大约 $1/125$ 的项目。

注意,算法 S 和 R 可用来同时得到若干独立范畴的一些抽样。例如,如果有美国居民的名字和地址的一份大文件,我们可以从 50 个州的每一个当中挑出恰有 10 个人的随机抽样,而不必对此文件进行 50 次扫描,而且不必首先按州对这文件进行排序。

如果我们生成一个随机变量来告知我们自己应当跳过多少记录,而不是决定是否跳过每一个记录,则当 n/N 很小时,对算法 S 和算法 R 两者的重大改进是有可能的(请见习题 8)。

按照从 N 个事物中每次取 n 个加以组合的通常定义(见 1.2.6 小节),抽样问题可以认为是进行一个随机组合的计算。现在我们考虑计算 t 个对象的一个随机排列的问题;我们把这称为洗牌问题,因为洗一副牌,无非是对它进行一次随机的排列。

稍加思索就能使任何玩牌者认识到,传统地用来洗牌的一些方法是极不适当的;通过这样的方法根本不能在任何地方都以近乎相等的概率得到 $t!$ 个排列的每一个。据说老练的桥牌行家在决定是否“叫”牌时便利用这一事实。为了达到 10% 以内的一致分布,对 52 张的一副牌至少需要洗 7 次,而 14 次随机的洗牌可保证之[见 Aldous 和 Diaconis, *AMM* 93(1986), 333~348]。

如果 t 很小,我们可以通过在 1 与 $t!$ 之间生成一个随机整数非常迅速地得到一个随机排列。例如,当 $t=4$ 时,在 1 与 24 之间的一个随机数足以从包含所有可能的排列的一张表中选择一个随机排列。但是对很大的 t ,如果我们要声言每个排列是同样可能的,就有必要更仔细些,因为 $t!$ 比个别的随机数的精度要大得多。

一个适当的洗牌过程可以通过回想算法 3.3.2P 来得到,它给出 $t!$ 个可能的排列的每一个排列同一个序列 $(c_1, c_2, \dots, c_{t-1})$ 之间的简单对应,这里 $0 \leq c_j \leq j$ 。随机地计算这样一组数是很容易的,因而我们可以使用这个对应来产生一个随机排列。

算法 P(洗牌) 设 X_1, X_2, \dots, X_t 是要洗的 t 个数的一個集合。

P1.[初始化] 置 $j \leftarrow t$ 。

P2.[生成 U] 生成在 0 与 1 之间一致分布的随机数 U 。

P3.[交换] 置 $k \leftarrow \lfloor jU \rfloor + 1$ 。(现在 k 是 1 与 j 之间的一个随机整数。习题 3.4.1-3 说明不应取模 j 剩余以计算 k 。)交换 $X_k \leftrightarrow X_j$ 。

P4.[减小 j] j 减 1。如果 $j > 1$,则返回步骤 P2。■

这个算法首先是由 R. A. Fisher 和 F. Yates 以普通语言发表的[*Statistical Tables* (London, 1938), 例 12], 而由 R. Durstenfeld 以计算机语言发表[*CACM* 7(1964), 420]。如果我们只想生成 $\{1, \dots, t\}$ 的一个随机排列而不是洗一个给定的序列 $(X_1, \dots,$

X_i), 我们可以通过令 j 由 1 增加到 t 并置 $X_j \leftarrow X_t, X_t \leftarrow j$ 来避免交换操作 $X_k \leftrightarrow X_j$; 见 D. E. Knuth, *The Stanford GraphBase* (New York: ACM Press, 1994), 104。

R. Salfi [COMPSTAT 1974 (Vienna: 1974), 28~35] 已经指出, 当我们以模 m 的一个线性同余序列得到一致的 U 时, 算法 P 可能不能生成多于 m 个的不同排列, 也就是说其实当我们使用一个递推 $U_{n+1} = f(U_n)$ 时, 对于它 U_n 仅可取 m 个不同的值, 因为在这样的情况下最后的排列完全由被生成的头一个 U 的值所确定。因此, 例如, 如果 $m = 2^{32}$, 则 13 个元素的某些排列将不会出现, 因为 $13! \approx 1.45 \times 2^{32}$ 。在大多数应用中, 我们并不真正要看到所有 $13!$ 的排列; 但是知道被排除的排列是由像一个格结构这样相当简单的数学规则确定的, 却是令人不安的 (请见 3.3.4 小节)。

当我们使用像 3.2.2-(7) 那样具有充分长周期的延搁斐波那契生成程序时, 不出现这个问题。但即使使用这样一些方法, 我们也不能一致地得到所有的排列, 除非我们能够确定至少 $t!$ 个不同初始值来初始化这个生成程序。换句话说, 除非我们把 $\lg t!$ 个真正随机二进位放进去, 否则就得出 $\lg t!$ 个真正随机的二进位。3.5 节说明, 我们不必对此感到绝望。

可以很容易地修改算法 P 来产生一个随机组合的随机排列 (见习题 15)。关于其它类型的随机组合对象 (例如分划) 的讨论, 请见 7.2 节和/或 Nijenhuis 和 Wilf 的书 *Combinatorial Algorithms* (New York: Academic Press, 1975)。

习 题

1. [M12] 解释等式 (1)。

2. [20] 证明算法 S 绝不试图去读它的输入文件的 N 个以上的记录。

▶ 3. [22] 算法 S 中的第 $t+1$ 项以概率 $(n-m)/(N-t)$ 而不是以 n/N 被选择, 但正文中断言这个抽样是无偏倚的, 所以每项都应以相同的概率被选择! 这两个命题怎么能同时为真?

4. [M23] 设 $p(m, t)$ 是在选择抽样技术中从头 t 项当中恰巧选择 m 项的概率。由算法 S 直接证明:

$$p(m, t) = \binom{t}{m} \binom{N-t}{n-m} / \binom{N}{n}, \quad \text{对于 } 0 \leq t \leq N$$

5. [M24] 当算法 S 终止时, t 的平均值是什么? (换句话说, 在抽样完成之前, 平均已经扫描过 N 个记录中的多少个?)

6. [M24] 上题所计算的值的标准差是多少?

7. [M25] 证明, 算法 S 以概率 $1/\binom{N}{n}$ 得到由 N 个记录的集合中取 n 个记录的任何给定的选择。因此这个抽样是完全无偏倚的。

8. [M39] (J. S. Vitter) 算法 S 对于它处理的每个输入记录计算一个一致离差。本习题的目的是考虑一个更有效的方法, 以便更快速地计算它在做头一个选择之前要跳过的输入记录的适当个数 X 。

a) 给定 $k, X \geq k$ 的概率是多少?

b) 证明, a) 的结果允许我们只须生成一个一致的 U , 然后进行 $O(X)$ 的其它计算就可计算

X_0

c) 证明, 我们也可以置 $X \leftarrow \min(Y_N, Y_{N-1}, \dots, Y_{N-n+1})$, 其中诸 Y 是独立的, 而且在 $0 \leq Y_i < t$ 的范围内每个 Y_i 是一个随机整数。

d) 为了获得极大速度, 证明, 使用等式 3.4.1-(18) 那样的一个“挤压”方法, 平均说来, 也可以在 $O(1)$ 步内计算 X 。

9. [12] 设 $n=3$ 。如果把算法 R 应用于含有编号为 1 到 20 的 20 个记录的文件上, 而且如果在步骤 R3 中生成的随机数分别是

4,1,6,7,5,3,5,11,11,3,7,9,3,11,4,5,4

问哪些记录进入水库？哪些是在最后的抽样中？

10. [15] 假定当前抽样中的 n 个记录可以放在内存中, 试修改算法 R 使得能消去该水库。

► 11. [M25] 设 p_m 是在算法 R 的头一次扫描期间恰有 m 个元素被存入水库的概率。试确定生成函数 $G(z) = \sum p_m z^m$, 并求均值和标准差。(利用 1.2.10 小节的思想。)

12. [M26] 算法 P 的主旨是任何排列 π 都可惟一地写成在 $\pi = (a_t t) \cdots (a_3 3)(a_2 2)$ 的形式下的置换的乘积, 其中当 $t \geq j > 1$ 时, $1 \leq a_j \leq j$ 。证明也存在惟一的形如 $\pi = (b_2 2)(b_3 3) \cdots (b_t t)$ 的一个表示, 其中当 $1 < j \leq t$ 时, $1 \leq b_j \leq j$, 并设计在 $O(t)$ 步内从诸 a 计算诸 b 的算法。

13. [M23] (S. W. Golomb) 洗牌最普遍的方法之一是尽可能地把牌分成相等的两部分, 并且把它们“洗”到一起。(按照打牌游戏的 Hoyle 规则中玩牌规矩的讨论, “这种洗牌应当进行大约三次, 以使这些牌彻底地搅乱”。) 考虑 $2n-1$ 张卡片的一副牌 $X_1, X_2, \dots, X_{2n-1}$; 一个“大清洗”, 把这些牌分成为 X_1, X_2, \dots, X_n 和 X_{n+1}, \dots, X_{2n-1} , 然后完全地相互交错它们以得到 $X_1, X_{n+1}, X_2, X_{n+2}, \dots, X_{2n-1}, X_n$ 。 “切”牌操作 c' 把 $X_1, X_2, \dots, X_{2n-1}$ 变成为 $X_{j+1}, \dots, X_{2n-1}, X_1, \dots, X_j$ 。说明当 $n \geq 1$ 时, 通过彻底的洗和切, 顶多可能有 $(2n-1)(2n-2)$ 种不同的安排。

[4.22] $a_0 a_1 \cdots a_{n-1}$ 的一个切和洗排列, 把它变成这样一个序列, 它包含对于某个 x 和 y , 相互交换的子序列

$$a_x a_{(x+1) \bmod n} \cdots a_{(y-1) \bmod n} \text{ 和 } a_y a_{(y+1) \bmod n} \cdots a_{(x-1) \bmod n}$$

因此, 3890145267 是对于 $x=3$ 和 $y=8$ 的 0123456789 的一个切和洗。

a) 以在标准次序

2 3 4 5 6 7 8 9 10 J Q K A 2 3 4 5 6 7 8 9 10 J Q K A 2 3 4 5 6 7 8 9 10 J Q K A 2 3 4 5 6 7 8 9 10 J Q K A

下的 52 张牌开始, J. H. Quick(一名学生)做了一次切和洗, 然后他撤去最左边的一张牌并把它插进一个随机的位置, 得到序列

9 10 K J Q A K A 2 Q 3 2 3 4 5 6 7 4 8 9 5 10 6 J 7 Q 8 K 9 10 J Q A K 2 3 A 4 2 3 4 5 6 5 6 7 8 7 9 10 J S

问他从最左边的位置移动的是哪张牌？

b)再次从这组牌原始顺序开始,Quick 现在做了3次切和洗后把最左边的牌移动到新位置

10 J Q 3 4 5 6 J J Q 4 6 K A 2 3 K 4 7 5 6 Q A 7 5 A 8 7 6 K K 9 A 7 8 9 10 8 10 8 2 5 J 2 3 Q 4 9 3 2 9 10

问这次他动的是哪张牌？

15. [30] (Ole-Johan Dahl) 在算法 P 的开始, 如果对于 $1 \leq k \leq t$, $X_k = k$, 而且当 j 达到 $t - n$ 的值时, 我们终止这个算法, 则序列 X_{t-n+1}, \dots, X_t 是 n 个元素的一个随机组合的随机排列。试说明如何只使用 $O(n)$ 个内存单元模拟这个过程的效果。

16. [M25] 基于散列(6.4节)的思想,给定 N 和 n ,试设计一个方法进行从 N 抽样 n 个记录的计算,你的算法应当使用 $O(n)$ 个存储单元和平均 $O(n)$ 个时间单位,而且它应当把抽样表现

为一个排了序的整数集合 $1 \leq X_1 < X_2 < \cdots < X_n \leq N$ 。

17. [M22] (R. W. Floyd) 试证明, 下列算法从 $\{1, \cdots, N\}$ 生成 n 个整数的一个随机抽样: 置 $S \leftarrow \emptyset$; 然后对于 $j \leftarrow N - n + 1, N - n + 2, \cdots, N$ (按此顺序), 置 $k \leftarrow \lfloor jU \rfloor + 1$ 且

$$S \leftarrow \begin{cases} S \cup \{k\}, & \text{如果 } k \notin S \\ S \cup \{j\}, & \text{如果 } k \in S \end{cases}$$

* 3.5 什么是随机序列

A. 引论 在这一章中, 我们已经看到了怎样生成在 $0 \leq U_n < 1$ 范围内的实数序列

$$\langle U_n \rangle = U_0, U_1, U_2, \cdots \quad (1)$$

而且我们把它们称做“随机”序列, 尽管在本质上它们是完全确定的。为了说明这个术语是合理的, 我们断言这些数的特性“很像是真正的随机数”。这个命题对于实用的目的(在现时)可能是令人满意的, 但它回避了一个非常重要的哲学问题和理论问题: 我们所说的“随机行为”精确地说究竟指的是什么? 我们需要有随机行为的定量的定义。谈论那些我们并不真正理解的概念是不合意的, 特别是因为关于随机数会做出许多明显似是而非的论述。

概率和统计的数学理论审慎地回避回答这个问题; 它避免做绝对的陈述, 只是对那些涉及随机事件序列的命题以它的概率有多大来表达事物。概率论公理的建立, 使得抽象概率的计算变得容易了, 但是关于概率真正意味着什么, 或者这些概念怎样能有意义地应用于现实世界, 却没有说及。在 *Probability, Statistics, and Truth* (New York: Macmillan, 1957) 一书中, R. von Mises 详细地讨论了这一情况, 而且提出了如下观点, 即为了要给出概率的适当定义, 首先要搞清楚随机序列的适当定义。

许多作者都已对这一问题表达了他们的看法, 这里我们来解释他们中的两位关于随机序列的论述。

D. H. Lehmer (1951): “随机序列是一个含糊的概念, 它具体是指, 在一序列中, 每一项对于非初始项来说都是不可预料的, 而且它的数字通过了一定数量的检验, 这些检验又是统计学家沿袭用的, 并且与该序列的用途有关。”

J. N. Franklin (1962): “如果序列(1)具有这样的特点, 即凡是从一致分布的随机变量中独立抽样获得的每个无限序列都有的性质, 它都有, 则序列(1)是随机的。”

Franklin 的命题实际上推广了 Lehmer 的命题, 说的是这个序列必须满足所有的统计检验。他的定义不是完全精确的, 我们以后将看到, 他的命题的一个合理的解释会使我们得出根本就没有随机序列这样的结论来! 因此我们还是从限制较少的 Lehmer 的论述开始, 看看能否把它弄精确。实际上我们真正要的是一张相当短的表, 即某些数学性质的一览表, 我们关于随机数列的直观概念满足所有这些性质; 而且这张表应完备到足以使我们同意, 满足这些性质的任何序列都是“随机的”。在这一节里, 我们将按照这些准则提出随机性的适当定义, 尽管还有许多有趣的问题

有待回答。

设 u 和 v 是实数, $0 \leq u < v \leq 1$ 。如果 U 是一个随机变量, 它在 0 与 1 之间一致分布, 则 $u \leq U < v$ 的概率等于 $v - u$ 。例如, $\frac{1}{5} \leq U < \frac{3}{5}$ 的概率是 $\frac{2}{5}$ 。怎样才能把单个数 U 的这个性质转换成无限序列 U_0, U_1, U_2, \dots 的一个性质呢? 明显的答案是计算 U_n 有多少次位于 u 和 v 之间, 其平均次数应等于 $v - u$ 。我们关于概率的直观思想就是这样以出现的频率为基础的。

更确切地说, 设 $\nu(n)$ 是当 $0 \leq j < n$ 时使得 $u \leq U_j < v$ 的 j 的个数; 当 n 趋向无穷大时我们要使比值 $\nu(n)/n$ 趋于值 $v - u$:

$$\lim_{n \rightarrow \infty} \nu(n)/n = v - u \quad (2)$$

如果对于所有的 u 和 v 的选择, 这个条件都成立, 那么就说这个序列是等分布的。

设 $S(n)$ 是关于整数 n 和序列 U_1, U_2, \dots 的一个命题。例如, $S(n)$ 可以是上边考虑的命题“ $u \leq U_n < v$ ”。我们可以推广前一段中所采用的思想, 对于一个特定的无限序列来定义 $S(n)$ 为真的概率。

定义 A 设 $\nu(n)$ 是对于 $0 \leq j < n$, 使得 $S(j)$ 为真的 j 的个数。如果当 n 趋于无穷大时 $\nu(n)/n$ 等于 λ , 我们就说 $S(n)$ 以 λ 的概率为真。用符号表示, 如果 $\lim_{n \rightarrow \infty} \nu(n)/n = \lambda$, 则 $\Pr(S(n)) = \lambda$ 。

在这个记号下, 当且仅当对所有实数 $u, v, 0 \leq u < v \leq 1$, 有 $\Pr(u \leq U_n < v) = v - u$ 时, 序列 U_0, U_1, \dots 是等分布序列。

一个序列可以是等分布序列而不是随机序列。例如, 如果 U_0, U_1, \dots 和 V_0, V_1, \dots 是等分布序列, 不难证明, 序列

$$W_0, W_1, W_2, W_3, \dots = \frac{1}{2}U_0, \frac{1}{2} + \frac{1}{2}V_0, \frac{1}{2}U_1, \frac{1}{2} + \frac{1}{2}V_1, \dots \quad (3)$$

也是等分布的, 因为子序列 $\frac{1}{2}U_0, \frac{1}{2}U_1, \dots$ 在 0 与 $\frac{1}{2}$ 之间是等分布的, 而交替的项 $\frac{1}{2} + \frac{1}{2}V_0, \frac{1}{2} + \frac{1}{2}V_1, \dots$ 是在 $\frac{1}{2}$ 与 1 之间等分布的。但在诸 W 的序列中, 小于 $\frac{1}{2}$ 的一个值总是有一个大于或等于 $\frac{1}{2}$ 的值跟随其后, 反之亦然; 因此, 对于任何合理的定义, 这个序列都不是随机的。可见, 需要有比等分布更强的性质。

等分布性质的自然推广, 是消除上一段中所述的缺陷来考虑序列中相邻的数对。对于任何四个数 u_1, v_1, u_2, v_2 , 其中 $0 \leq u_1 < v_1 \leq 1, 0 \leq u_2 < v_2 \leq 1$ 我们可以要求这个序列满足条件

$$\Pr(u_1 \leq U_n < v_1 \text{ 和 } u_2 \leq U_{n+1} < v_2) = (v_1 - u_1)(v_2 - u_2) \quad (4)$$

一般来说, 对于任何正整数 k , 可以要求, 序列在下列意义下是 k 分布的:

定义 B 当 $1 \leq j \leq k$ 时, 如果对于实数 u_j, v_j 的所有选择, 其中 $0 \leq u_j < v_j \leq 1$,

皆有

$$\Pr(u_1 \leq U_n < v_1, \dots, u_k \leq U_{n+k-1} < v_k) = (v_1 - u_1) \cdots (v_k - u_k) \quad (5)$$

成立,则说序列(1)是 k 分布的。

一个等分布序列是1分布序列。注意,如果 $k > 1$,则 k 分布序列总是 $k-1$ 分布的,因为可以在等式(5)中置 $u_k = 0$ 和 $v_k = 1$ 。因此,特别地任何序列若已知是4分布的序列,必定也是3分布的,2分布的和1分布的。我们可以研究使给定的序列是 k 分布的最大的 k ;这就导致下面的定义:

定义 C 如果对于所有正整数 k ,一个序列是 k 分布的,就说它是 ∞ 分布的。

至今我们已经考虑了“ $[0,1)$ 序列”,即处于0与1之间的实数序列。同样的思想亦可应用于整值序列;我们说如果一个序列 $\langle X_n \rangle = X_0, X_1, X_2, \dots$ 的每个 X_n 都是整数 $0, 1, \dots, b-1$ 之一,就说它是一个 b 进序列,于是二进序列是0和1的序列。

我们也定义,一个 k 位 b 进数是 k 个整数 $x_1 x_2 \cdots x_k$ 的一个串,其中对于 $1 \leq j \leq k$ 有 $0 \leq x_j < b$ 。

定义 D 如果对于所有 b 进数 $x_1 x_2 \cdots x_k$ 有

$$\Pr(x_n x_{n+1} \cdots x_{n+k-1} = x_1 x_2 \cdots x_k) = 1/b^k \quad (6)$$

则说一个 b 进序列是 k 分布的。

根据这个定义,显然,如果 U_0, U_1, \dots 是一个 k 分布的 $[0,1)$ 序列,则 $\lfloor bU_0 \rfloor, \lfloor bU_1 \rfloor, \dots$ 是一个 k 分布的 b 进序列。(如果置 $u_j = x_j/b, v_j = (x_j + 1)/b, X_n = \lfloor bU_n \rfloor$,则等式(5)即变成等式(6)。)而且,每个 k 分布的 b 进序列也是 $(k-1)$ 分布的,当 $k > 1$ 时;我们把诸 b 进数 $x_1 \cdots x_{k-1} 0, x_1 \cdots x_{k-1} 1, \dots, x_1 \cdots x_{k-1} (b-1)$ 的概率加在一起,得到

$$\Pr(X_n \cdots X_{n+k-2} = x_1 \cdots x_{k-1}) = 1/b^{k-1}$$

(不相交事件的概率是可加的;见习题5。)因此上面定义C中谈到 ∞ 分布的 b 进序列是很自然的事。

在 b 进数系下,一个正实数的表示可以看成是一个 b 进序列;例如 π 对应于10进数序列3,1,4,1,5,9,2,6,5,3,5,8,9,……。人们已经猜测,这个序列是 ∞ 分布的,但还没有人能够证明它甚至是1分布的。

现在我们在 $k = 1000000$ 的情况下更周密地来分析这些概念。1000000分布的一个二进序列在一行中要有一百万个0的运行!类似地,一个1000000分布的 $[0,1)$ 序列有一百万个相继值的运行,其中每个值都小于 $\frac{1}{2}$ 。平均说来,发生这种情况的可能性只有 $\left(\frac{1}{2}\right)^{1000000}$,但事实是它确实发生了。实际上,利用我们“真正随机”

的直观概念,这个现象将在任何真正随机的序列中出现。人们很容易想像,如果把这一百万个“真正随机”的数的集合用于一个计算机模拟实验中,这样一种情况将会产生多么巨大的影响;有充分的理由来抱怨随机数生成程序!然而,如果在一个数

列中没有一百万个小于 $\frac{1}{2}$ 的相继 U 的运行,则这序列就不是随机的,而且它也不适合作为数据源用于任何需要极长的 U 的区段作为输入的场所。总之,一个真正的随机数序列将显示局部的非随机性。在某些应用中,局部非随机性是必要的。但在其它一些应用中它是灾难性的。我们不得不作出结论:不存在对于每个应用都适当的“随机数”序列。

类似地,人们可以论证,没有办法判断一个有限序列的随机与否;任何特定序列与任何其它的序列的可能性一样大。如果我们真想要有一个有用的随机性定义,则上述事实肯定是绊脚石,但也不必过于惊慌。我们仍然有可能给出实数的无限序列的随机性定义,而相应的理论(在适当的观点下)将使我们更透彻地理解在一台计算机上生成的通常的有理数的有限序列。而且,我们在这一节的后边将看到,对于有限序列,有若干可行的随机性定义。

B. ∞ 分布序列 现在我们就来对 ∞ 分布序列的理论进行简略的研究。为适当地描述这一理论,需要使用高等数学中的有关知识,所以在这节的剩下部分假定,读者已经知道通常在“高等微积分”课程中讲授的内容。

首先推广定义 A 是方便的,因为那里出现的极限并非对所有序列都存在。我们定义

$$\overline{\text{Pr}}(S(n)) = \limsup_{n \rightarrow \infty} \frac{\nu(n)}{n}, \quad \underline{\text{Pr}}(S(n)) = \liminf_{n \rightarrow \infty} \frac{\nu(n)}{n} \quad (7)$$

因此,如果 $\text{Pr}(S(n))$ 存在,它就是 $\underline{\text{Pr}}(S(n))$ 与 $\overline{\text{Pr}}(S(n))$ 的共同的值。

我们已经看到,如果以 $\lfloor bU \rfloor$ 来代替 U , 一个 k 分布 $[0,1)$ 序列导致一个 k 分布 b 进序列。我们的头一个定理证明反过来的结果也是真的。

定理 A 设 $\langle U_n \rangle = U_0, U_1, U_2, \dots$ 是一个 $[0,1)$ 序列,如果序列

$$\langle \lfloor b_j U_n \rfloor \rangle = \lfloor b_j U_0 \rfloor, \lfloor b_j U_1 \rfloor, \lfloor b_j U_2 \rfloor, \dots$$

对于在一个无限整数序列 $1 < b_1 < b_2 < b_3 < \dots$ 中的所有 b_j 是一个 k 分布的 b_j 进序列,则原来的序列 $\langle U_n \rangle$ 是 k 分布的。

作为这个定理的一个例子,假设 $b_j = 2^j$, 序列 $\lfloor 2^j U_0 \rfloor, \lfloor 2^j U_1 \rfloor, \dots$ 实质上是 U_0, U_1, \dots 的二进表示的头 j 位的序列。如果所有这些整数序列在定义 D 的意义下都是 k 分布的,则实数值序列 U_0, U_1, \dots 在定义 B 的意义下也必是 k 分布的。

定理 A 的证明 如果序列 $\lfloor b U_0 \rfloor, \lfloor b U_1 \rfloor, \dots$ 是 k 分布的,则通过概率的加法可以证明:每当每个 μ_j 和 v_j 是具有分母 b 的一个有理数时,等式(5)成立。现在设 u_j, v_j 是任何实数, u'_j, v'_j 是具有分母 b 的有理数,使得

$$u'_j \leq u_j < u'_j + 1/b, \quad v'_j \leq v_j < v'_j + 1/b$$

设 $S(n)$ 是命题 $u_1 \leq U_n < v_1, \dots, u_k \leq U_{n+k-1} < v_k$, 我们有

$$\overline{\text{Pr}}(S(n)) \leq \text{Pr}\left(u'_1 \leq U_n < v'_1 + \frac{1}{b}, \dots, u'_k \leq U_{n+k-1} < v'_k + \frac{1}{b}\right) =$$

$$\begin{aligned} & \left(v'_1 - u'_1 + \frac{1}{b}\right) \cdots \left(v'_k - u'_k + \frac{1}{b}\right) \\ \Pr(S(n)) & \geq \Pr\left(u'_1 + \frac{1}{b} \leq U_n < v'_1, \dots, u'_k + \frac{1}{b} \leq U_{n+k-1} < v'_k\right) = \\ & \left(v'_1 - u'_1 - \frac{1}{b}\right) \cdots \left(v'_k - u'_k - \frac{1}{b}\right) \end{aligned}$$

此刻 $|(v'_j - u'_j \pm 1/b) - (v_j - u_j)| \leq 2/b$ 。由于我们的不等式对于所有 $b = b_j$ 成立, 而且当 $j \rightarrow \infty$ 时 $b_j \rightarrow \infty$, 故有

$$(v_1 - u_1) \cdots (v_k - u_k) \leq \Pr(S(n)) \leq \overline{\Pr}(S(n)) \leq (v_1 - u_1) \cdots (v_k - u_k) \quad \text{I}$$

下一个定理是证明有关 k 分布序列的一些事实的主要工具。

定理 B 设 $\langle U_n \rangle$ 是一个 k 分布 $[0, 1)$ 序列, 而且设 $f(x_1, x_2, \dots, x_k)$ 是 k 个变量的黎曼可积函数, 则

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{0 \leq j < n} f(U_j, U_{j+1}, \dots, U_{j+k-1}) = \\ \int_0^1 \cdots \int_0^1 f(x_1, x_2, \dots, x_k) dx_1 \cdots dx_k \end{aligned} \quad (8)$$

证明 k 分布序列的定义指出, 这个结果对于某些常数 $u_1, v_1, \dots, u_k, v_k$ 在

$$f(x_1, \dots, x_k) = [u_1 \leq x_1 < v_1, \dots, u_k \leq x_k < v_k] \quad (9)$$

的特殊情况下为真。因此, 每当 $f = a_1 f_1 + a_2 f_2 + \dots + a_m f_m$ 且每个 f_j 是类型(9)的一个函数时, 等式(8)为真; 换句话说, 每当 f 是通过下列步骤得到的“阶梯函数”时, 等式(8)成立: 把单位 k 维立方体划分成一些子胞(subcell), 这些子胞的表面平行于坐标轴, 并在每个子胞上对 f 赋予一个常数值。

现在设 f 是任何黎曼可积函数。如果 ϵ 是任何正数, 则我们知道(由黎曼可积定义)存在阶梯函数 \underline{f} 和 \bar{f} , 使得 $\underline{f}(x_1, \dots, x_k) \leq f(x_1, \dots, x_k) \leq \bar{f}(x_1, \dots, x_k)$, 而且使得 \underline{f}, f 和 \bar{f} 的积分之差小于 ϵ 。由于等式(8)对于 \underline{f} 和 \bar{f} 成立, 且由于

$$\begin{aligned} \frac{1}{n} \sum_{0 \leq j < n} \underline{f}(U_j, \dots, U_{j+k-1}) & \leq \frac{1}{n} \sum_{0 \leq j < n} f(U_j, \dots, U_{j+k-1}) \leq \\ & \frac{1}{n} \sum_{0 \leq j < n} \bar{f}(U_j, \dots, U_{j+k-1}) \end{aligned}$$

可以断定等式(8)对 f 亦真。I

定理 B 可应用于例如 3.3.2 小节中的排列检验。假设 (p_1, p_2, \dots, p_k) 是数 $\{1, 2, \dots, k\}$ 的任何排列; 我们要证明

$$\Pr(U_{n+p_1-1} < U_{n+p_2-1} < \dots < U_{n+p_k-1}) = 1/k! \quad (10)$$

为证此, 假设序列 $\langle U_n \rangle$ 是 k 分布的, 并设

$$f(x_1, \dots, x_k) = [x_{p_1} < x_{p_2} < \dots < x_{p_k}]$$

我们有

$$\begin{aligned} \Pr(U_{n+p_1-1} < U_{n+p_2-1} < \cdots < U_{n+p_k-1}) &= \int_0^1 \cdots \int_0^1 f(x_1, \cdots, x_k) dx_1 \cdots dx_k = \\ &= \int_0^1 dx_{p_k} \int_0^{x_{p_k}} \cdots \int_0^{x_{p_3}} dx_{p_2} \int_0^{x_{p_2}} dx_{p_1} = \frac{1}{k!} \end{aligned}$$

推论 P 如果一个 $[0,1)$ 序列是 k 分布的,则在等式(10)的意义下,它满足阶为 k 的排列检验。■

我们也可以证明,序列相关检验也能通过:

推论 S 如果一个 $[0,1)$ 序列是 $(k+1)$ 分布的,则 U_n 与 U_{n+k} 之间的序列相关系数趋于 0:

$$\lim_{n \rightarrow \infty} \frac{\frac{1}{n} \sum U_j U_{j+k} - \left(\frac{1}{n} \sum U_j \right) \left(\frac{1}{n} \sum U_{j+k} \right)}{\sqrt{\left(\frac{1}{n} \sum U_j^2 - \left(\frac{1}{n} \sum U_j \right)^2 \right) \left(\frac{1}{n} \sum U_{j+k}^2 - \left(\frac{1}{n} \sum U_{j+k} \right)^2 \right)}} = 0$$

(这里所有求和都对于 $0 \leq j < n$ 进行。)

证明 由定理 B,当 $n \rightarrow \infty$ 时,量

$$\frac{1}{n} \sum U_j U_{j+k}, \quad \frac{1}{n} \sum U_j^2, \quad \frac{1}{n} \sum U_{j+k}^2, \quad \frac{1}{n} \sum U_j, \quad \frac{1}{n} \sum U_{j+k}$$

分别趋于极限 $\frac{1}{4}, \frac{1}{3}, \frac{1}{3}, \frac{1}{2}, \frac{1}{2}$ 。■

现在我们考虑序列的某些稍微更一般的分布性质。通过考虑所有相邻的 k 元组,我们已经定义了 k 分布的概念。例如,当且仅当点

$$(U_0, U_1), (U_1, U_2), (U_2, U_3), (U_3, U_4), (U_4, U_5), \cdots$$

在单位正方形中等分布时,一个序列是 2 分布的。然而,很可能交替的点偶 $(U_1, U_2), (U_3, U_4), (U_5, U_6), \cdots$ 本身并不是等分布的;如果点 (U_{2n-1}, U_{2n}) 的密度在某个面积中是不足的,则其它的点 (U_{2n}, U_{2n+1}) 可以弥补它。例如,周期长度为 16 的一个周期二进序列

$$\langle X_n \rangle = 0, 0, 0, 1, \quad 0, 0, 0, 1, \quad 1, 1, 0, 1, \quad 1, 1, 0, 1, \quad 0, 0, 0, 1, \cdots \quad (11)$$

看得出是 3 分布的,而且偶数号的元素的子序列 $\langle X_{2n} \rangle = 0, 0, 0, 0, 1, 0, 1, 0, \cdots$ 中 0 的个数是 1 的个数的 3 倍,而奇数号元素的子序列 $\langle X_{2n+1} \rangle = 0, 1, 0, 1, 1, 1, 1, 1, \cdots$ 中 1 的个数是 0 的个数的 3 倍。

假设序列 $\langle U_n \rangle$ 是 ∞ 分布的,例(11)表明,交替项子序列 $\langle U_{2n} \rangle = U_0, U_2, U_4, U_6, \cdots$ 并非显然地保证是 ∞ 分布的,或者甚至是 1 分布的。但我们将看到,事实上, $\langle U_{2n} \rangle$ 是 ∞ 分布的,而且甚至有多得多为真的事实。

定义 E 如果对满足 $0 \leq u_r < v_r \leq 1$ (其中 $1 \leq r \leq k$) 的所有实数 u_r, v_r 的选择以及对于 $0 \leq j < m$ 的所有整数 j , 有

$$\Pr(u_1 \leq U_{mn+j} < v_1, u_2 \leq U_{mn+j+1} < v_2, \dots, u_k \leq U_{mn+j+k-1} < v_k) = (v_1 - u_1) \cdots (v_k - u_k)$$

则说一个 $[0, 1)$ 序列 $\langle U_n \rangle$ 是 (m, k) 分布的。

于是一个 k 分布序列是定义 E 中的 $m=1$ 的特殊情况; 情况 $m=2$ 意味着在偶位置上开始的 k 元组必然和在奇位置上开始的 k 元组有相同的密度, 等等。

定义 E 的下列性质是显然的:

当 $1 \leq \kappa \leq k$ 时, 一个 (m, k) 分布序列是 (m, κ) 分布的。 (12)

对于 m 的所有因子 d , 一个 (m, k) 分布序列是 (d, k) 分布的。 (13)

(见习题 8。)和定义 D 中一样, 我们也可以定义 (m, k) 分布的 b 进序列的概念, 而且对于 (m, k) 分布序列, 定理 A 仍保持成立。

下一个定理在许多方面有些令人惊奇, 它说明了 ∞ 分布的性质确实是很强的, 比起我们先前考虑这个概念时所想像的要强得多。

定理 C (Ivan Niven 和 H. S. Zuckerman) 一个 ∞ 分布的序列对于所有正整数 m 和 k 都是 (m, k) 分布的。

证明 利用刚才提到的定理 A 的推广, 对 b 进序列证明这个定理就足够了。而且, 还可以假定 $m=k$, 因为 (12) 和 (13) 指出, 如果一个序列是 (mk, mk) 分布的, 则它也是 (m, k) 分布的。

所以, 我们将证明任何 ∞ 分布的 b 进序列 X_0, X_1, \dots 对于所有正整数 m 都是 (m, m) 分布的。我们的证明是 Niven 和 Zuckerman 在 *Pacific J. Math.* 1 (1951), 103 ~ 109 上给出的证明的简化形式。

证明用到的一个关键思想是: “如果 m 个量的和以及它们平方的和, 两者都符合 m 个量是相等的假设, 则该假设就是正确的。”这一思想在许多数学论证中都是非常有用的。这个原理可以以更充分的形式叙述如下:

引理 E 对于 $1 \leq j \leq m$, 给定 m 个数列 $\langle y_{jn} \rangle = y_{j0}, y_{j1}, \dots$, 假设

$$\begin{aligned} \lim_{n \rightarrow \infty} (y_{1n} + y_{2n} + \dots + y_{mn}) &= ma \\ \limsup_{n \rightarrow \infty} (y_{1n}^2 + y_{2n}^2 + \dots + y_{mn}^2) &\leq ma^2 \end{aligned} \quad (14)$$

则对于每个 j , $\lim_{n \rightarrow \infty} y_{jn}$ 存在并等于 a 。

习题 9 给出了这个引理的一个极其简单的证明。■

现在继续证明定理 C, 设 $x = x_1 x_2 \dots x_m$ 是一个 b 进数, 而且如果 $X_{p-m+1} X_{p-m+2} \dots X_p = x$, 则说 x 出现在这个序列的位置 p 上。设 $\nu_j(n)$ 是当 $p < n$ 和 $p \bmod m = j$ 时, x 出现于位置 p 的次数。设 $y_{jn} = \nu_j(n)/n$, 我们希望证明

$$\lim_{n \rightarrow \infty} y_{jn} = \frac{1}{mb^m} \quad (15)$$

首先我们知道

$$\lim_{n \rightarrow \infty} (y_{0n} + y_{1n} + \cdots + y_{(m-1)n}) = \frac{1}{b^m} \quad (16)$$

因为这个序列是 m 分布的。由引理 E 和等式(16)可知,如果能证明

$$\limsup_{n \rightarrow \infty} (y_{0n}^2 + y_{1n}^2 + \cdots + y_{(m-1)n}^2) \leq \frac{1}{mb^{2m}} \quad (17)$$

则本定理即得证。

这个不等式还不是显然的;在我们能够证明它之前,有必要采取某些稍微巧妙的策略。设 q 是 m 的一个倍数,考虑

$$C(n) = \sum_{0 \leq j < m} \left(\frac{\nu_j(n) - \nu_j(n-q)}{2} \right) \quad (18)$$

这是在位置 p_1, p_2 中出现的 x 的数偶的个数,其中 $n-q \leq p_1 < p_2 < n$, 且 $p_2 - p_1$ 是 m 的倍数。现在考虑和数

$$S_N = \sum_{n=1}^{N+q} C(n) \quad (19)$$

在位置 p_1, p_2 上出现的 x 的每个数偶都在总和 S_N 中被计算 $p_1 + q - p_2$ 次,其中 $p_1 < p_2 < p_1 + q$, $p_2 - p_1$ 是 m 的倍数,且 $p_1 \leq N$ (即,当 $p_2 < n \leq p_1 + q$ 时);而且对于 $N < p_1 < p_2 < N + q$, 这样出现的数偶被精确地计算了 $N + q - p_2$ 次。

设 $d_t(n)$ 是当 $p_1 + t = p_2 < n$ 时,位置 p_1, p_2 上 x 的数偶出现的个数。上边的分析表明

$$\sum_{0 < t < q/m} (q - mt) d_{mt}(N + q) \geq S_N \geq \sum_{0 < t < q/m} (q - mt) d_{mt}(N) \quad (20)$$

由于原来的序列是 q 分布的,所以对所有的 $t, 0 < t < q/m$,

$$\lim_{N \rightarrow \infty} \frac{1}{N} d_{mt}(N) = \frac{1}{b^{2m}} \quad (21)$$

因此由(20),

$$\lim_{N \rightarrow \infty} \frac{S_N}{N} = \sum_{0 < t < q/m} \frac{q - mt}{b^{2m}} = \frac{q(q-m)}{2mb^{2m}} \quad (22)$$

在做了某些处理之后,这一事实即能证明定理。

由定义知

$$2S_N = \sum_{n=1}^{N+q} \sum_{0 \leq j < m} ((\nu_j(n) - \nu_j(n-q))^2 - (\nu_j(n) - \nu_j(n-q)))$$

我们可以通过应用(16)消去非平方项来得到

$$\lim_{N \rightarrow \infty} \frac{T_N}{N} = q(q-m)/mb^{2m} + q/b^m \quad (23)$$

其中

$$T_N = \sum_{n=1}^{N+q} \sum_{0 \leq j < m} (\nu_j(n) - \nu_j(n-q))^2$$

利用不等式

$$\frac{1}{r} \left(\sum_{j=1}^r a_j \right)^2 \leq \sum_{j=1}^r a_j^2$$

(参考习题 1.2.3-30)求得

$$\limsup_{N \rightarrow \infty} \sum_{0 \leq j < m} \frac{1}{N(N+q)} \left(\sum_{n=1}^{N+q} (\nu_j(n) - \nu_j(n-q)) \right)^2 \leq \frac{q(q-m)}{mb^{2m}} + \frac{q}{b^m} \quad (24)$$

我们还有

$$q\nu_j(N) \leq \sum_{N < n \leq N+q} \nu_j(n) = \sum_{n=1}^{N+q} (\nu_j(n) - \nu_j(n-q)) \leq q\nu_j(N+q)$$

而且把这代入(24)给出

$$\limsup_{N \rightarrow \infty} \sum_{0 \leq j < m} (\nu_j(N)/N)^2 \leq \frac{q-m}{qmb^{2m}} + \frac{1}{qb^m} \quad (25)$$

当 q 是 m 的倍数时, 这个公式是已经确立了的; 令 $q \rightarrow \infty$, 即得到(17), 这就完成了证明。

关于一个更简单的证明, 见 J. W. S. Cassels, *Pacific J. Math.* 2 (1952), 555 ~ 557. |

习题 29 和 30 说明了这个定理的不平凡性, 而且它们也说明了: 一个 q 分布序列的概率, 同真正的 (m, n) 分布概率相比, 实际上顶多偏离 $1/\sqrt{q}$ (参考(25))。对于这个定理的证明来说, ∞ 分布的全部假设都是必要的。

作为定理 C 的一个结果, 我们可以证明, 一个 ∞ 分布序列能通过序列检验、 t 的极大值检验、冲突检验以及 3.3.2 小节中提出的对子序列的一些检验; 不难证明, 它也能通过间隔检验、扑克检验以及运行检验 (见习题 12 到 14)。集券检验处理起来要难得多, 但也能通过 (见习题 15 和 16)。

下一个定理保证了一类较为简单的 ∞ 分布序列的存在。

定理 F (J. N. Franklin) 对于几乎所有的实数 $\theta > 1$, 满足

$$U_n = \theta^n \bmod 1 \quad (26)$$

的 $[0, 1)$ 序列 U_0, U_1, U_2 是 ∞ 分布的。即, 集合 $\{\theta \mid \theta > 1 \text{ 且 (26) 不是 } \infty \text{ 分布}\}$ 的测度为 0。

这个定理的证明和某些推广在 *Math. Comp.* 17 (1963), 28 ~ 59 中给出。|

Franklin 证明了, 为使 (26) 是 ∞ 分布的, θ 必须是超越数。20 世纪 60 年代初利用多精度算术, 人们已经费力地计算出 $n \leq 10000$ 时的所有乘方 $\langle \pi^n \bmod 1 \rangle$; 而且对于每个这些数的最高 35 位, 已存储在一个磁盘文件中, 并被成功地用做一个一致离差的来源。由定理 F 看到, 乘方 $\langle \pi^n \bmod 1 \rangle$ 是 ∞ 分布的概率等于 1; 但由于有不可数多个实数, 因此, 关于 π 的序列是否真正 ∞ 分布的, 它什么也没有告诉我们。可以相

当保险地打赌:在我们的有生之年,没有人会证明这个特殊序列不是 ∞ 分布的,但也可能果真不是。由于这些考虑,人们当然要问是否有任何明确的 ∞ 分布的序列,即:是否有一个算法,对于所有 $n \geq 0$,来计算实数 U_n ,使得序列 $\langle U_n \rangle$ 是 ∞ 分布的?回答是肯定的,如 D. E. Knuth 在 *BIT* 5 (1965), 246 ~ 250 中给出的例子所示的那样。在那里构造的序列完全由有理数组成;事实上,每个数 U_n 在二进制数系中都有一个有限的表示。一个明确的 ∞ 分布序列的另一种构造,比刚才引证的序列略复杂些,可以从下面的定理 W 得出。也见 N. M. Korobov, *Izv. Akad. Nauk SSSR* 20 (1956), 649 ~ 660。

C. ∞ 分布 = 随机吗? 综观以上关于 ∞ 分布序列的全部理论,有一件事我们可以确信: ∞ 分布序列的概念是数学上一个重要的概念。同时也有大量的证据说明,下列命题是随机性的直观思想的一种正确的阐述。

定义 R1 如果一个 $[0, 1)$ 序列是一个 ∞ 分布序列,则定义它为“随机”序列。

我们已经看到,满足这个定义的序列能通过 3.3.2 小节的所有统计检验,而且还能通过更多的检验。

我们来试一下客观地鉴定这个定义。首先,每个“真正随机”的序列是 ∞ 分布的吗?在 0 与 1 之间有不可数多的实数序列 U_0, U_1, \dots 。如果选出一个真正的随机数生成程序,并令它生成值 U_0, U_1, \dots ,则任何可能的序列都可以看成是同等可能的,而且,某些序列(其实,是它们中的不可数多个)甚至不是等分布的。另一方面,利用在所有可能序列的空间上的概率的任何合理的定义,我们就能得出结论:一个随机序列是 ∞ 分布的概率为 1。因此这导致我们以如下的方式来形式化 Franklin 随机数定义(如本小节开头所给出的那样):

定义 R2 设 $\langle U_n \rangle$ 为 $[0, 1)$ 的序列, $\langle V_n \rangle$ 是一致分布的随机变量独立抽样的序列,如果 P 是这样一种性质,即每当 $P(\langle V_n \rangle)$ 以 1 的概率成立时, $P(\langle U_n \rangle)$ 也为真,则我们定义 $\langle U_n \rangle$ 为随机序列。

定义 R1 或许可能等价于定义 R2 吗?现在让我们彻底考察定义 R1 的某些可能的异议,看看这些批评是否适当。

首先,定义 R1 仅涉及当 $n \rightarrow \infty$ 时这个序列的极限性质。有一些 ∞ 分布序列,其中头一百万个元素全为 0,这样一个序列也应看成是随机的吗?

这个异议不很得要领。如果 ϵ 是任何正数,没有理由说一个序列的头一百万个元素不应全小于 ϵ 。一个真正的随机序列以概率 1 包含无限多个运行,其中每个都含有小于 ϵ 的百万个相继元素,所以为什么这不能发生在这个序列的开头呢?

另一方面,考虑定义 R2,并设 P 表示“一个序列的所有元素都不同”这一性质; P 以概率 1 为真,所以按这个准则,具有百万个 0 的任何序列都不是随机的。

现在设 P 表示序列没有元素等于 0 这一性质; P 又以概率 1 为真,所以由定义 R2,任何具有一个零元素的序列都不是随机的。然而,更一般地,令 x_0 是 0 与 1 之间的任何固定的数,并设 P 是序列的元素都不等于 x_0 这一性质;定义 R2 又指出,

没有随机序列能包含元素 x_0 ! 我们现在可以证明,没有满足定义 R2 的条件的序列。(因为如果 U_0, U_1, \dots 是这样一序列,则取 $x_0 = U_0$ 即否定了它。)

因此,如果 R1 是一个太弱的定义,那么 R2 肯定太强了。“正确”的定义必须不如 R2 苛刻。然而,我们并未真正证明 R1 太弱,所以让我们进一步处置它。如同上边所指出的,我们已经构造了一个有理数的 ∞ 分布序列(其实,这并不那样令人惊讶;见习题 18)。几乎所有实数都是无理数,也许我们应坚持,对于一个随机序列来说:

$$\Pr(U_n \text{ 是有理数}) = 0$$

等分布的定义等式(2)指出 $\Pr(u \leq U_n < v) = v - u$ 。利用测度论,有一个明显的方式来推广这个定义:“如果 $S \subseteq [0, 1)$ 是测度为 μ 的集合,则对所有随机序列 $\langle U_n \rangle$,

$$\Pr(U_n \in S) = \mu \quad (27)$$

成立”。特别是,如果 S 是有理数的集合,则它的测度为 0,所以在这种推广的意义下,没有有理数的序列是等分布的。如果约定性质(27),则预期定理 B 可以推广到勒贝格积分,以代替黎曼积分是合理的。然而,我们再次发现,定义(27)是太苛刻了,因为没有序列满足该性质! 如果 U_0, U_1, \dots 是任意序列,则集合 $S = \{U_0, U_1, \dots\}$ 的测度为 0,而且 $\Pr(U_n \in S) = 1$ 。因此借助于我们用来把有理数排除于随机序列之外的同样论证,也可以排除所有的随机序列。

到目前为止,已证明定义 R1 是站得住脚的。然而,对于它仍有某些完全有道理的异议。例如,如果我们有在直观意义下的一个随机序列,则无穷子序列

$$U_0, U_1, U_4, U_9, \dots, U_{n^2}, \dots \quad (28)$$

也应是一个随机序列。对于一个 ∞ 分布序列,这并不总是正确的,事实上,如果我们任取一个 ∞ 分布序列,并且对所有 n 置 $U_n' \leftarrow 0$,则在 k 分布检验中的计数 $v_k(n)$ 至多改变 \sqrt{n} ,所以比值 $v_k(n)/n$ 的极限保持不变。定义 R1 不幸不能满足这个随机性准则。

也许我们应如下来加强定义 R1:

定义 R3 如果一个 $[0, 1)$ 序列的每一个无限子序列都是 ∞ 分布的,则定义这个序列是“随机”序列。

但是我们再次发现,这个定义太苛刻了;任何等分布的序列 $\langle U_n \rangle$ 都有一个单调子序列且 $U_{s_0} < U_{s_1} < U_{s_2} < \dots$ 。

奥妙在于要限定这些子序列,使得它们在被定义时,不必考察 U_n 就能决定 U_n 是否应属于这个子序列 U_n 。现在引出下边的定义:

定义 R4 令 $\langle U_n \rangle$ 是 $[0, 1)$ 的序列,如对于 $n \geq 0$,对于确定不同非负整数 s_n 的无限序列的每个能行算法,对应于这个算法的子序列 $U_{s_0}, U_{s_1}, U_{s_2}, \dots$ 是 ∞ 分布的,则说序列 $\langle U_n \rangle$ 是“随机的”。

定义 R4 中所指的算法是给定 n 后计算 s_n 的能行过程(见 1.1 节中的讨论)。于是例如,序列 $\langle \pi^n \bmod 1 \rangle$ 不满足 R4,因为它或者不是等分布的,或者没有一个能行的算法来确定一个无限子序列 s_n ,使得 $(\pi^{s_0} \bmod 1) < (\pi^{s_1} \bmod 1) < (\pi^{s_2} \bmod 1) < \dots$ 。类似地,任何明确定义的序列都不能满足定义 R4;如果我们同意,所有明显地定义的序列都不是真正随机的,则这个结论就是适当的。然而,对于几乎所有实数 $\theta > 1$,看似明显的序列 $\langle \theta^n \bmod 1 \rangle$ 实际上确实都满足定义 R4;这并不矛盾,因为几乎所有 θ 都不能由算法计算。J.F. Koksma 证明了,如果 s_n 是不同正整数的任何序列,则几乎对于所有 $\theta > 1$, $\langle \theta^{s_n} \bmod 1 \rangle$ 是 1 分布的[Compositio Math. 2 (1935), 250 ~ 258];H. Niederreiter 和 R. F. Tichy 以 ∞ 分布代替 1 分布,加强了 Koksma 的定理[Mathematika 32 (1985), 26 ~ 32]。仅仅可数多个序列 $\langle s_n \rangle$ 是可有效地定义的,因此 $\langle \theta^n \bmod 1 \rangle$ 几乎总是满足 R4。

定义 R4 比定义 R1 强得多;但断言定义 R4 太弱仍然有道理。例如,设 $\langle U_n \rangle$ 是一个真正随机的序列,而且以下列规则定义子序列 $\langle U_{s_n} \rangle$: $s_0 = 0$;而且如果 $n > 0$, s_n 是大于等于 n 的最小整数,使得 $U_{s_n-1}, U_{s_n-2}, \dots, U_{s_n-n}$ 全部小于 $1/2$ 。因此,我们考察的是紧接于头一个连续的 n 个小于 $1/2$ 的值的运行之后的值子序列。假设“ $U_n < 1/2$ ”对应于在一次硬币投掷中的“正面”值。假定使用的是一个真正的硬币,赌者势必感到,在“正面”多次出现以后,另一面,即“背面”出现的可能性就增加了。刚才定义的子序列 $\langle U_{s_n} \rangle$ 对应于一个人的赌博系统,这个人在紧接着 n 个相继的“正面”的头一个运行之后,下他的第 n 个赌注。赌者可能想 $\Pr(U_{s_n} \geq 1/2)$ 是大于 $1/2$ 的,但当然在一个真正随机序列里 $\langle U_{s_n} \rangle$ 将是完全随机的。没有一个赌博系统能稳操胜券! 定义 R4 没有说及根据这样一个赌博系统所形成的子序列,所以明显地我们仍需要更进一步的東西。

让我们定义一个“子序列规则” \mathcal{R} 是函数 $\langle f_n(x_1, \dots, x_n) \rangle$ 的一个无穷序列,其中对子 $n \geq 0$, f_n 是 n 个变量的一个函数,而且 $f_n(x_1, \dots, x_n)$ 的值或为 0 或为 1。这里 x_1, \dots, x_n 是某个集合 S 的元素。(于是,特别是, f_0 为一常数函数,或为 0 或为 1。)一个子序列规则 \mathcal{R} 定义任何无穷序列 $\langle X_n \rangle$ 的一个子序列如下:当且仅当 $f_n(X_0, X_1, \dots, X_{n-1}) = 1$ 时,第 n 项 X_n 在子序列 $\langle X_n \rangle_{\mathcal{R}}$ 中。注意,这样定义的子序列 $\langle X_n \rangle_{\mathcal{R}}$ 不必是无穷的,事实上它可以是空子序列。

例如,刚才所述的“赌者的子序列”对应于以下的子序列规则:“ $f_0 = 1$;且对于 $n > 0$, $f_n(x_1, \dots, x_n) = 1$ 当且仅当在 $0 < k \leq n$ 的范围中有某个 k ,使得当 $m = n$ 但不是当 $k \leq m < n$ 时, k 个连续的参数 $x_m, x_{m-1}, \dots, x_{m-k+1}$ 全都小于 $1/2$ 。”

如果有一个能行的算法,当给定 n 和 x_1, \dots, x_n 作为输入时,它确定 $f_n(x_1, \dots, x_n)$ 的值,就说子序列规则 \mathcal{R} 是可计算的。当试图定义随机性时,我们最好把我们自己限制在可计算的子序列规则上,免得我们得到定义 R3 那样过于苛刻的定义。但是能行算法不能很好地处理作为输入的任意实数;例如,如果通过一个无穷的十

进制展开来确定一个实数 x , 则没有算法可确定 x 是否小于 $\frac{1}{3}$, 因为需要考察数 $0.333\cdots$ 的所有数字。因此, 可计算子序列规则不能应用于所有的 $[0, 1)$ 序列。所以, 把我们下一个定义建立在 b 进序列基础上是适宜的。

定义 R5 如果一个 b 进序列的每一个由可计算子序列规则定义的无穷子序列都是 1 分布的, 则这个 b 进序列是“随机”的序列。

令 $\langle U_n \rangle$ 是一个 $[0, 1)$ 序列, 如果对于所有整数 $b \geq 2$, b 进序列 $\langle \lfloor bU_n \rfloor \rangle$ 都是“随机”的, 则 $\langle U_n \rangle$ 是“随机”的。

注意定义 R5 仅说“1 分布”, 没说“ ∞ 分布”。验证这样做这并未失去一般性是有趣的。因为对于任何 b 进数 a_1, \cdots, a_k , 我们可定义一个明显可计算的子序列规则 $\mathcal{R}(a_1, \cdots, a_k)$ 如下: 当且仅当 $n \geq k+1$ 且 $x_{n-k+1} = a_1, \cdots, x_{n-1} = a_k, x_n = a_{k+1}$ 时, $f_n(x_1, \cdots, x_n) = 1$ 。现在如果 $\langle X_n \rangle$ 是一个 k 分布的 b 进序列, 则这个规则 $\mathcal{R}(a_1, \cdots, a_k)$ ——它选择由那些恰恰在 a_1, \cdots, a_k 的一个出现之后的项组成的子序列——定义一个无穷子序列; 而且如果这个子序列是 1 分布的, 则对于 $0 \leq a_{k+1} < b$, 每个 $k+1$ 元组 $a_1, \cdots, a_k, a_{k+1}$ 在 $\langle X_n \rangle$ 中出现的概率都是 $1/b^{k+1}$ 。于是, 我们可以对 k 施用归纳法证明, 满足定义 R5 的一个序列对所有 k 都是 k 分布的。类似地, 通过考虑子序列的“合成”规则——如果 \mathcal{R}_1 定义了一个无穷子序列 $\langle X_n \rangle_{\mathcal{R}_1}$, 则我们可以定义 $\mathcal{R}_1 \mathcal{R}_2$ 作为使得 $\langle X_n \rangle_{\mathcal{R}_1 \mathcal{R}_2} = (\langle X_n \rangle_{\mathcal{R}_1})_{\mathcal{R}_2}$ 的子序列的规则——可以证明, 在定义 R5 中考虑的所有子序列都是 ∞ 分布的 (见习题 32)。

∞ 分布是定义 R5 的一个非常特殊的情况, 这一事实是令人鼓舞的; 它是一个好迹象, 表明我们最后可以找出所探求的随机性定义。但可惜仍然有问题。满足定义 R5 的序列是否必定满足定义 R4, 这点尚不清楚。我们刚才描述的“可计算子序列规则”总是枚举子序列 $\langle X_{s_n} \rangle$, 其中 $s_0 < s_1 < \cdots$, 但在 R4 中并不要求 $\langle s_n \rangle$ 是单调的, 它仅须满足条件: 当 $n \neq m$ 时, $s_n \neq s_m$ 。

为解决这个异议, 我们可以把定义 R4 和 R5 组合如下:

定义 R6 设 $\langle X_n \rangle$ 是一个 b 进序列。如果对于把不同非负整数的一个无穷序列 $\langle s_n \rangle$ 描述为 n 和值 $X_{s_0}, \cdots, X_{s_{n-1}}$ 的一个函数的每个能行算法, 在定义 R5 的意义下, 对应于这个算法的子序列 $\langle X_{s_n} \rangle$ 是“随机的”, 则说 $\langle X_n \rangle$ 是“随机”的。

如果对所有整数 $b \geq 2$, b 进序列 $\langle \lfloor bU_n \rfloor \rangle$ 是“随机”的, 则说 $[0, 1)$ 序列 $\langle U_n \rangle$ 是“随机”的。

作者认为* 这个定义确实满足随机性的所有合乎哲理的要求, 因此它回答了这节中所提出的主要问题。

* 至少当作者于 1966 年准备这一节的内容时, 他是这么认为的。——作者原注

D. 随机序列的存在性 我们看到,在不存在能满足定义 R3 的序列的意义下,定义 R3 是太强了;而上边的定义 R4, R5 和 R6 的努力企图抓住定义 R3 的本质特征。为了证明定义 R6 所作的限制并不过分,我们仍必须证明,满足所有这些条件的序列是存在的。直观上,我们十分确信这样序列的存在是没有问题的,因为我们坚信存在一个真正的随机序列,而且它满足定义 R6;但是还是必须有一个证明,来说明这个定义是无矛盾的。

A. Wald 给出了构造满足定义 R5 的序列的一个有趣方法,该方法从一个非常简单的 1 分布序列开始。

引理 T 设实数序列 $\langle V_n \rangle$ 借助于二进系统定义如下:

$$V_0 = 0, \quad V_1 = .1, \quad V_2 = .01, \quad V_3 = .11, \quad V_4 = .001, \quad \dots$$

$$V_n = .c_r \cdots c_1 1, \quad \text{如果 } n = 2^r + c_1 2^{r-1} + \cdots + c_r \quad (29)$$

命 $I_{b_1 \cdots b_r}$ 表示 $[0, 1)$ 中其二进表示以 $0.b_1 \cdots b_r$ 开始的所有实数的集合,于是

$$I_{b_1 \cdots b_r} = [(0.b_1 \cdots b_r)_2, (0.b_1 \cdots b_r)_2 + 2^{-r}) \quad (30)$$

这样,如果对于 $0 \leq k < n$, $\nu(n)$ 表示 $I_{b_1 \cdots b_r}$ 中 V_k 的个数,则我们有

$$|\nu(n)/n - 2^{-r}| \leq 1/n \quad (31)$$

证明 因为 $\nu(n)$ 是使 $k \bmod 2^r = (b_r \cdots b_1)_2$ 的 k 的个数,所以当 $\lfloor n/2^r \rfloor = t$ 时,有 $\nu(n) = t$ 或 $t+1$, 因此 $|\nu(n) - n/2^r| \leq 1$ 。■

由(31)得出,序列 $\langle \lfloor 2^r V_n \rfloor \rangle$ 是一个等分布的 2^r 进序列;因此由定理 A, $\langle V_n \rangle$ 是一个等分布的 $[0, 1)$ 序列。其实,很显然, $\langle V_n \rangle$ 的等分布性几乎和一个 $[0, 1)$ 序列之为等分布差不多! (关于这个序列及有关序列的进一步讨论,见 J. G. van der Corput, *Proc. Koninklijke Nederl. Akad. Wetenschappen* **38** (1935), 813 ~ 821, 1058 ~ 1066; J. H. Halton, *Numerische Math.* **2** (1960), 84 ~ 90, 196; S. Haber, *J. Research National Bur. Standards* **B70** (1966), 127 ~ 136; R. Bézian 和 H. Faure, *Comptes Rendus Acad. Sci. Paris* **A285** (1977), 313 ~ 316; H. Faure, *J. Number Theory* **22** (1986), 4 ~ 20; Stezuka, *ACM Trans. Modeling and Comp. Simul.* **3** (1993), 99 ~ 107。L. H. Ramshaw 已经证明,序列 $\langle \phi n \bmod 1 \rangle$ 比 $\langle V_n \rangle$ 稍微更多地是等分布的;见 *J. Number Theory* **13** (1981), 138 ~ 175。)

现在设 $\mathcal{R}_1, \mathcal{R}_2, \dots$ 是无穷多个子序列规则,我们希望求这样一个序列 $\langle U_n \rangle$, 使得所有无穷子序列 $\langle U_n \rangle_{\mathcal{R}_j}$ 都是等分布的。

算法 W (Wald 序列) 给定定义有理数的 $[0, 1)$ 序列的子序列之子序列规则 $\mathcal{R}_1, \mathcal{R}_2, \dots$ 的一个无穷序列,下述过程定义一个 $[0, 1)$ 序列 $\langle U_n \rangle$ 。在计算中涉及无穷多个辅助变量 $C[a_1, \dots, a_r]$, 其中 $r \geq 1$, 且对于 $1 \leq j \leq r$, $a_j = 0$ 或 1 。这些变量开始时全为 0。

W1. [初始化 n] 置 $n \leftarrow 0$ 。

W2. [初始化 r] 置 $r \leftarrow 1$ 。

W3. [检验 \mathcal{R}_r] 如果根据 U_k (对于 $0 \leq k < n$) 的值, 元素 U_n 是在由 \mathcal{R}_r 定义
的子序列中, 则置 $a_r \leftarrow 1$, 否则置 $a_r \leftarrow 0$ 。

W4. [$[a_1, \dots, a_r]$ 的情况尚未完成?] 如果 $C[a_1, \dots, a_r] < 3 \cdot 4^{r-1}$, 则转向
W6。

W5. [r 增值] 置 $r \leftarrow r + 1$, 并且转到 W3。

W6. [置 U_n] $C[a_1, \dots, a_r]$ 的值加 1 并命 k 是 $C[a_1, \dots, a_r]$ 的新值。置 $U_n \leftarrow$
 V_k , 其中 V_k 在上边的引理 T 中定义。

W7. [推进 n] n 增加 1 并返回 W2。■

严格地说, 这不是一个算法, 因为它不终止。但是, 当然可以很容易地修改这个
过程使 n 达到一个给定值时就停止。读者如想掌握此构造的思想, 建议用手工试
验, 在试验前以 2^r 代替步骤 W4 的数 $3 \cdot 4^{r-1}$ 。

算法 W 并非打算作为随机数的一个实际来源。它只试图为理论的目的服务。

定理 W 设 $\langle U_n \rangle$ 是由算法 W 定义的有理数序列, 并设 k 是一个正整数, 如果子
序列 $\langle U_n \rangle_{\mathcal{R}_k}$ 是无穷的, 则它是 1 分布的。

证明 命 $A[a_1, \dots, a_r]$ 表示 $\langle U_n \rangle$ 的 (可能是空的) 子序列, 它恰巧包含具有如
下特征的元素: 对于所有 $j \leq r$, 如果 $a_j = 1$, 它们属于子序列 $\langle U_n \rangle_{\mathcal{R}_j}$, 如果 $a_j = 0$, 则
它们不属于 $\langle U_n \rangle_{\mathcal{R}_j}$ 。

只须证明, 对于所有 $r \geq 1$ 和所有二进数数偶 $a_1 \dots a_r$ 和 $b_1 \dots b_r$, 每当子序列 A
[a_1, \dots, a_r] 是无穷时, 相对于此子序列有 $\Pr(U_n \in I_{b_1 \dots b_r}) = 2^{-r}$ [见等式 (30)]。因
为如果 $r \geq k$, 则无穷序列 $\langle U_n \rangle_{\mathcal{R}_k}$ 是所有 $a_k = 1$ 和 $a_j = 0$ 或 1 (其中 $1 \leq j \leq r, j \neq k$)
的不相交子序列 $A[a_1, \dots, a_r]$ 的有限并。由此推出相对于 $\langle U_n \rangle_{\mathcal{R}_k}$ 有 $\Pr(U_n \in$
 $I_{b_1 \dots b_r}) = 2^{-r}$ (见习题 33)。由定理 A, 就足以证明这序列是 1 分布的。

命 $B[a_1, \dots, a_r]$ 表示 $\langle U_n \rangle$ 的这样的子序列, 它由在算法步骤 W6 中使 $C[a_1,$
 $\dots, a_r]$ 加 1 的那些 n 的值组成。由这个算法, $B[a_1, \dots, a_r]$ 是至多有 $3 \times 4^{r-1}$ 个元
素的有限序列。 $A[a_1, \dots, a_r]$ 除了有限的成员外, 全都来自子序列 $B[a_1, \dots, a_r,$
 $\dots, a_t]$, 其中对于 $r < j \leq t, a_j = 0$ 或 1。

现在假定 $A[a_1, \dots, a_r]$ 是无穷的, 且设 $A[a_1, \dots, a_r] = \langle U_{s_i} \rangle$, 其中 $s_0 < s_1 <$
 $s_2 < \dots$ 。如果 N 是一个大整数, 且 $4^r \leq 4^q < N \leq 4^{q+1}$, 则由此推知, 使得 U_{s_k} 在 $I_{b_1 \dots b_r}$
中的 k ($k < N$) 的值的个数 (除了在于子序列开头处的有限个元素外) 是

$$\nu(N) = \nu(N_1) + \dots + \nu(N_m)$$

这里 m 是上边所列子序列 $B[a_1, \dots, a_r]$ 的个数, 其中 U_{s_k} 对于某个 k ($k < N$) 出现,
 N_j 是使 U_{s_k} 在对应子序列中的 k 值的个数, 而且 $\nu(N_j)$ 也是在 $I_{b_1 \dots b_r}$ 中的这样的值
的个数。因此, 由引理 T,

$$\begin{aligned} |\nu(N) - 2^{-r}N| &= |\nu(N_1) - 2^{-r}N_1 + \cdots + \nu(N_m) - 2^{-r}N_m| \leq \\ &|\nu(N_1) - 2^{-r}N_1| + \cdots + |\nu(N_m) - 2^{-r}N_m| \leq \\ &m \leq 1 + 2 + 4 + \cdots + 2^{q-r+1} < 2^{q+1} \end{aligned}$$

关于 m 的不等式,是从这样的事实得出的:由我们对于 N 的选择,对于某个 $t \leq q+1$,元素 U_{s_N} 在 $B[a_1, \cdots, a_t]$ 中。

我们证得 $|\nu(N)/N - 2^{-r}| \leq 2^{q+1}/N < 2/\sqrt{N}$ 。■

为了最后证明满足定义 R5 的序列是存在的,我们首先说明,如果 $\langle U_n \rangle$ 是有理数的一个 $[0,1)$ 序列,而且如果 \mathcal{R} 是一个 b 进序列的可计算子序列规则,则通过使 \mathcal{R}' 中的 $f'(x_1, \cdots, x_n)$ 等于 \mathcal{R} 中的 $f_n(\lfloor bx_1 \rfloor, \cdots, \lfloor bx_n \rfloor)$,可以使 \mathcal{R}' 成为 $\langle U_n \rangle$ 的一个可计算子序列规则 \mathcal{R}' 。如果 $[0,1)$ 序列 $\langle U_n \rangle$ 是等分布的,则 b 进序列 $\langle \lfloor bU_n \rfloor \rangle_{\mathcal{R}}$ 也是等分布的。现在 b 进序列的所有可计算的子序列规则的集合(对于 b 的所有值来说)是可数的(因为仅仅可能有可数多个能行的算法),所以它们可以列成某个序列 $\mathcal{R}_1, \mathcal{R}_2, \cdots$;因此算法 W 定义了一个 $[0,1)$ 序列,它在定义 R5 的意义下是随机的。

这把我们带到有些似是而非的境地。如同我们早先所述,没有一个能行算法能确定一个满足定义 R4 的序列。由于同样的原因,也没有能行的算法能确定一个满足 R5 的序列。这样的随机序列的存在性证明必然是非构造性的,那么算法 W 怎么能构造这样一个序列呢?

这里并没有矛盾,我们先前的困难仅仅在于:所有能行算法的集合不能被一个能行算法所枚举。换句话说,没有能行的算法来选择第 j 个可计算的子序列规则 \mathcal{R}_j ;这是由于不存在确定一计算方法是否终止的能行算法的缘故。但是我们能够系统地枚举很广的重要算法类。举例来说,算法 W 证明,如果我们限于考虑“本原递归”的子序列规则,则通过一个能行的算法,可能构造一个满足定义 R5 的序列。

通过修改算法 W 的步骤 W6,使其置 $U_n \leftarrow V_{k+t}$ 而不是 V_k ,其中 t 是与 a_1, \cdots, a_r 相关的任何非负整数,我们就可以证明,有不可数多个满足定义 R5 的 $[0,1)$ 序列。

以下定理证明存在另一个方法,可以证明不可数多个随机序列的存在性,这里用的是以测度论为基础的较为间接的证明,可是使用了强定义 R6。

定理 M 如果实数 $x (0 \leq x < 1)$ 的二进表示是 $(0.X_0X_1\cdots)_2$, 命 x 对应于二进序列 $\langle X_n \rangle$ 。在这个对应之下,几乎所有的 x 都对应于在定义 R6 的意义下随机的二进序列。(换句话说,所有对应于按定义 R6 为非随机的二进序列的实数 x 的集合,有测度 0。)

证明 命 \mathcal{S} 是一个能行算法,它确定不同非负整数 $\langle s_n \rangle$ 的一个无穷序列,其中 s_n 的选择仅仅依赖于 n 和 $X_{s_k}, 0 \leq k < n$; 且设 \mathcal{R} 是一个可计算的子序列规则。这样,任何二进序列 $\langle X_n \rangle$ 导致一个子序列 $\langle X_{s_n} \rangle_{\mathcal{R}}$ 。定义 R6 指出,这个子序列必须是

有限的或是1分布的。只须证明,对于固定的 \mathcal{R} 和 \mathcal{S} ,对应于 $\langle X_n \rangle$ 而且使得 $\langle X_n \rangle_{\mathcal{R}}$ 是无穷和非1分布的所有实数 X 的集合 $N(\mathcal{R}, \mathcal{S})$,有测度0。因为当且仅当 x 在 $\bigcup N(\mathcal{R}, \mathcal{S})$ 中(这里的并取遍所有可数多个 \mathcal{R} 和 \mathcal{S})时, x 有一个非随机的二进表示。

于是命 \mathcal{R} 和 \mathcal{S} 固定,把对于所有二进数 $a_1 a_2 \cdots a_r$ 定义的集合 $T(a_1 a_2 \cdots a_r)$ 当做是对应于 $\langle X_n \rangle$ 的所有 x 的集合,使得 $\langle X_n \rangle_{\mathcal{R}}$ 有大于等于 r 个元素,它的头 r 个元素分别等于 a_1, a_2, \cdots, a_r 。我们的头一个结果是:

$$T(a_1 a_2 \cdots a_r) \text{ 的测度 } \leq 2^{-r} \quad (32)$$

为证此,我们从观察 $T(a_1 a_2 \cdots a_r)$ 是一个可测集合开始: $T(a_1 a_2 \cdots a_r)$ 的每个元素是一个实数 $x = (0. X_0 X_1 \cdots)_2$,对于它存在一个整数 m 使得算法 \mathcal{S} 确定不同的值 s_0, s_1, \cdots, s_m ,而且规则 \mathcal{R} 确定 $X_{s_0}, X_{s_1}, \cdots, X_{s_m}$ 的一个子序列,使得 X_{s_m} 是这个序列的第 r 个元素。所有使得对于 $0 \leq k \leq m$ 有 $Y_{s_k} = X_{s_k}$ 的实数 $y = (0. Y_0 Y_1 \cdots)_2$ 的集合,也属于 $T(a_1 a_2 \cdots a_r)$,而且这是由二进子区间 $I_{b_1 \cdots b_k}$ 的有限的并组成的可测集合。由于仅有可数多个这样的二进区间,我们看到, $T(a_1 a_2 \cdots a_r)$ 是可数多个二进子区间的并,因此它是可测的。其次,这个论证可加以扩充证明 $T(a_1 \cdots a_{r-1} 0)$ 的测度等于 $T(a_1 \cdots a_{r-1} 1)$ 的测度,因为后者是从前者通过要求对于 $0 \leq k < m$ 有 $Y_{s_k} = X_{s_k}$ 且 $Y_{s_m} \neq X_{s_m}$ 而得到的二进区间的并。现在由于

$$T(a_1 \cdots a_{r-1} 0) \cup T(a_1 \cdots a_{r-1} 1) \subseteq T(a_1 \cdots a_{r-1})$$

$T(a_1 a_2 \cdots a_r)$ 的测度至多等于 $T(a_1 \cdots a_{r-1})$ 测度的一半,不等式(32)由对 r 用归纳法推得。

现在已经证得(32),剩下的实际上是要证明:几乎所有实数的二进表示都是等分布的。对于 $0 < \epsilon < 1$,命 $B(r, \epsilon)$ 是 $\bigcup T(a_1 \cdots a_r)$,这里的并取遍所有这样的二进数串 $a_1 \cdots a_r$,对于这些 $a_1 \cdots a_r$ 当中1的个数 $\nu(r)$ 满足

$$\left| \nu(r) - \frac{1}{2}r \right| \geq \epsilon r$$

这样的二进数的个数是 $C(r, \epsilon) = \sum \binom{r}{k}$,这里对满足 $\left| k - \frac{1}{2}r \right| \geq \epsilon r$ 的 k 值求和。

习题 1.2.10-21 证明 $C(r, \epsilon) \leq 2^{r+1} e^{-\epsilon^2 r}$;因此由(32),

$$B(r, \epsilon) \text{ 的测度 } \leq 2^{-r} C(r, \epsilon) \leq 2e^{-\epsilon^2 r} \quad (33)$$

下一步是定义

$$B^*(r, \epsilon) = B(r, \epsilon) \cup B(r+1, \epsilon) \cup B(r+2, \epsilon) \cup \cdots$$

$B^*(r, \epsilon)$ 测度至多是 $\sum_{k \geq r} 2e^{-\epsilon^2 k}$,而这是一个收敛级数的余部,所以

$$\lim_{r \rightarrow \infty} (B^*(r, \epsilon) \text{ 的测度 }) = 0 \quad (34)$$

现在如果 x 是这样一实数, 即其二进展开 $(0.X_0X_1\cdots)_2$ 导致不是 1 分布的一个无穷序列 $\langle X_n \rangle_{n \geq 0}$, 而且如果 $\nu(r)$ 表示在该序列的头 r 个元素中 1 的个数, 则对于某个 $\epsilon > 0$ 和无穷多个 r ,

$$\left| \nu(r)/r - \frac{1}{2} \right| \geq \epsilon$$

这意味着对于所有 r, x 在 $B^*(r, \epsilon)$ 中。所以最后我们求得

$$N(\mathcal{R}, \delta) = \bigcup_{t \geq 2} \bigcap_{r \geq 1} B^*(r, 1/t)$$

而且由 (34), 对所有 t , $\bigcap_{r \geq 1} B^*(r, 1/t)$ 的测度为 0, 因此 $N(\mathcal{R}, \delta)$ 有测度 0。■

由满足定义 R6 的二进序列的存在性, 我们可以证明在这个意义下随机的 $[0, 1)$ 序列的存在性, 详见习题 36。由此确立了定义 R6 的无矛盾性。

E. 随机有限序列 上边给出的论证指出, 不可能对有限序列定义随机性的概念; 任何给定的有限序列与任何其它的序列是同样可能的。然而, 几乎每个人都会同意, 序列 011101001 要比 101010101 “更随机”些, 而且甚至这后一序列也比 000000000 “更随机些”。尽管真正随机的序列确会显示出局部的非随机特性, 但我们期望这样的特性仅仅出现在一个长的有限序列中, 而不是在一个短的有限序列中。

人们提出了若干方法来定义一个有限序列的随机性, 这里将仅仅概述其中的某些思想。为方便起见, 只限于考虑 b 进序列的情况。

给定一个 b 进序列 $X_1, X_2, \cdots, X_{N-1}$, 我们可以说

$$\Pr(S(n)) \approx p, \quad \text{如果 } |\nu(N)/N - p| \leq 1/\sqrt{N} \quad (35)$$

其中 $\nu(n)$ 是这一节开始处定义 A 中出现的量。如果对于所有 b 进数 $x_1x_2\cdots x_k$ 有

$$\Pr(X_nX_{n+1}\cdots X_{n+k-1} = x_1x_2\cdots x_k) \approx 1/b^k \quad (36)$$

上边的序列可称为“ k 分布”的。(同定义 D 做比较。不幸的是, 按这个新定义, 一个不是 $(k-1)$ 分布的序列, 却可以是 k 分布的。)

现在可以模仿定义 R1 来给出一个随机性的定义。

定义 Q1 如果对于所有正整数 $k \leq \log_b N$, 一个长度为 N 的 b 进序列是 k 分布的 (在上述意义下), 则说它是随机的序列。

例如按照这个定义, 有 178 个长度为 11 的非随机二进序列:

00000001111	10000000111	11000000011	11100000001	11110000000
00000001110	10000000110	11000000010	11100000000	11010000000
00000001101	10000000101	11000000001	10100000001	10110000000
00000001011	10000000011	01000000011	01100000001	01110000000
00000000111				

加上 01010101010, 以及有 9 个或更多个 0 的所有序列, 加上从上边诸序列中通过交换 1 和 0 而得到的所有序列。

类似地,我们可以叙述对应于定义 R6 的有限序列的定义。命 A 是一组算法,其中每一个算法都像在定理 M 的证明中那样给出子序列 $\langle X_{i_n} \rangle \in \mathcal{R}$ 的一个组合挑选和选择过程。

定义 Q2 b 进序列 $X_0, X_1, X_2, \dots, X_{N-1}$ 称为相对于算法集 A 是 (n, ϵ) 随机的,如果对于由 A 的算法确定的每一个子序列 $X_{i_1}, X_{i_2}, \dots, X_{i_m}$, 我们有 $m < n$ 或者

$$\left| \frac{1}{m} \nu_a(x_{i_1}, \dots, x_{i_m}) - \frac{1}{b} \right| \leq \epsilon, \quad \text{对于 } 0 \leq a < b$$

这里 $\nu_a(x_1, \dots, x_m)$ 是序列 x_1, \dots, x_m 中 a 的个数。

(换句话说,由 A 的一个算法确定的每一个充分长的子序列,都必须是近似地等分布的。)在这种情况下基本思想是命 A 为一组“简单”的算法; A 中算法的个数(及复杂性)可以随 N 的增长而增长。

作为定义 Q2 的一个例子,我们考虑二进序列,并设 A 就是以下四个算法:

- a) 取整个序列。
- b) 由头一项开始,取序列的交替项。
- c) 取序列的在一个 0 之后的项。
- d) 取序列的在一个 1 之后的项。

现在一个序列 X_1, X_2, \dots, X_7 相对于 A 是 $\left(4, \frac{1}{8}\right)$ 随机的,如果:

由 a), $\left| \frac{1}{8}(X_0 + X_1 + \dots + X_7) - \frac{1}{2} \right| \leq \frac{1}{8}$, 即如果有 3, 4 或 5 个 1;

由 b), $\left| \frac{1}{4}(X_0 + X_2 + X_4 + X_6) - \frac{1}{2} \right| \leq \frac{1}{8}$, 即如果在偶数号数的位置中恰有两个 1;

由 c), 根据有多少个 0 占据位置 X_0, \dots, X_6 , 存在着三种可能性:如果在这里有 2 或 3 个 0, 则没有条件要检验(因为 $n=4$); 如果有 4 个 0, 则它们必须分别跟有 2 个 0 和 2 个 1; 而如果有 5 个 0, 则它们必须分别跟有 2 个或 3 个 0;

由 d), 我们得到类似于由 c) 所蕴涵的条件。

结果是,对于这些规则,仅有下列长度为 8 的二进序列是 $\left(4, \frac{1}{8}\right)$ 随机的,即

00001011	00101001	01001110	01101000
00011010	00101100	01011011	01101100
00011011	00110010	01011110	01101101
00100011	00110011	01100010	01110010
00100110	00110110	01100011	01110110
00100111	00111001	01100110	

加上由一致地交换 0 和 1 得到的那些序列。

显然,我们可以把这算法的集合扩充成如此之大,以至当 n 和 c 相当小时,没有序列能满足这个定义。A. H. Kolmogorov 已经证明,对于任何给定的 N ,如果 A 中算法的个数不超过

$$\frac{1}{2} e^{2nc^2(1/c)} \quad (37)$$

则将总存在一个 (n, c) 随机二进序列。这个结果还未强到足以证明满足定义 Q1 的序列的存在。但利用习题 3.2.2-21 中 Rees 的过程,即可有效地构造出这样的序列来。基于离散傅里叶变换的一个广义谱检验可用来检验根据定义 Q1 对一个序列的测定好到什么程度[请见 A. Compagner, *Physical Rev.* **52** (1995), 5634~5645]。

Per Martin-Löf 采取了另外一种有趣的方法来定义随机性[*Information and Control* **9** (1966), 602~619]。给定一个有限的 b 进序列 X_1, \dots, X_N , 令 $l(X_1, \dots, X_N)$ 是生成这个序列的最短的图灵机程序的长度。(或者,我们可以使用其它类的能行算法,如在 1.1 节中讨论的那些。)于是, $l(X_1, \dots, X_N)$ 是这个序列的“无模式”的一个测度,而且我们可以把这个思想和随机性等同起来。使 $l(X_1, \dots, X_N)$ 取极大值的长度 N 的序列可称做随机的。(从通过计算机生成实用随机数的观点看,当然,这是可以想像的“随机性”的最坏的定义!)

大约在同一时间 G. Chaitin 独立地给出实质上相同的随机性定义;见 *JACM* **16** (1969), 145~159。有趣的是,尽管这个定义没有像我们的其它定义那样提到等分布的性质, Martin-Löf 和 Chaitin 已经揭示,在适当的意义下,这种类型的随机序列也有短期的等分布的一些性质。事实上, Martin-Löf 已经证明,在适当的意义下,这样的序列满足关于随机性的所有可计算的统计检验。

关于随机有限序列的定义方面进一步的发展,请见 Zvonkin 和 L. A. Levin, *Uspekhi Mat. Nauk* **25**, 6 (1970 年 11 月), 85~127 (英文翻译见 *Russian Math. Surveys* **25**, 6 (1970 年 11 月), 83~124]; L. A. Levin, *Doklady Akad. Nauk SSSR* **212** (1973), 548~550; L. A. Levin, *Information and Control* **61** (1984) 15~37。

F. 拟随机数 从理论观点了解到,存在各种风格的随机有限序列,是令人愉快的。但是这样一些定理并没有回答现实生活中的程序员们所面临的各种问题。基于有限序列集合的研究,最新的发展已经导致一个更恰当理论的出现。更精确地说,我们考虑序列可以出现一次以上的多重集合。

设 S 是含有长度为 N 的二进位串(二进制序列)的一个多重集合;我们称 S 为一个 N 源。设 S_N 表示包含所有 2^N 个可能的 N 位串的 N 源。 S 的每个元素表示我们可以用做拟随机二进位的一个来源的序列;选择不同的“种子”值导致 S 的不同元素。例如,在由 $X_{j+1} = (aX_j + c) \bmod 2^c$ 定义的线性同余序列中 S 可以是

$$\{B_1, B_2, \dots, B_N \mid B_j \text{ 是 } X_j \text{ 的最高有效位}\} \quad (38)$$

其中对于 2^c 个起始值 X_0 的每一个,有一个串 $B_1 B_2 \dots B_N$ 。

如同本章所见到的那样,拟随机码序列的基本思想是得到看上去是随机的 N 个二进位,尽管当选择种子值时我们仅仅依赖于一些“真正的”随机二进位。在刚刚

考虑的例子中,我们需要 e 个真正随机的二进位来选择 X_0 ;一般来说,选择 S 的一个成员相当于使用 $\lg|S|$ 个真正随机的二进位,在那之后我们就确定地进行了。如果 $N=10^6$ 和 $|S|=2^{32}$,那我们对于扩展的每个真正随机的二进位就要得到多于 30 000 个“看起来随机”的二进位。对于 $\$N$ 而不是 N ,我们就得不到这样的扩大,因为 $\lg|\$N|=N$ 。

“看起来随机”意味着什么呢? 1982 年 A. C. Yao 提出了一个很好的定义:考虑任何算法 A ,它考察一个二进位串 $B=B_1\cdots B_N$,而且输出 $A(B)=0$ 或 1 ,我们可以把 A 想像为对随机性的一个检验;例如, A 可以计算连续的 0 和 1 的运行的分布。而且如果运行的长度上分不同于预期的分布则输出 1。每当 A 如此进行时,我们可以考虑当 B 是 S 的一个随机选定的元素时 $A(B)=1$ 的概率 $P(A, S)$,而且我们可以把它同当 B 是长度为 N 的一个真正的随机二进位串时 $A(B)=1$ 的概率 $P(A, \$N)$ 做比较。如果对于所有的统计检验 A , $P(A, S)$ 极其接近 $P(A, \$N)$,我们就无法区别 S 的序列和真正随机的二进位序列。

定义 P 如果 $|P(A, S) - P(A, \$N)| < \epsilon$,则我们说对于容差 ϵ ,一个 N 源 S 通过统计检验 A ;如果 $|P(A, S) - P(A, \$N)| \geq \epsilon$,则它不通过这个检验。

算法 A 不必由统计学家来设计。按照定义 P,任何算法都可认为是随机性的一个统计检验。当 A 实施它的计算时,我们允许 A 投掷硬币(即使用真正的随机二进位)。惟一的要求是 A 必须输出 0 或 1。

当然,实际上还有另一个要求:我们坚持, A 必须在一个合理的时间之内派送它的输出,至少是平均说来。我们对于要花费许多年来运行的算法不感兴趣。因为如果我们的计算机在我们的有生之年不能发现 S 和 $\$N$ 的区别,那我们也就永远不会注意到它们的区别。 S 的序列仅包含信息的 $\lg|S|$ 位,所以肯定地会有这样的算法,它们将最终会检测到冗余性;但我们不在乎,真正要紧的是,只要 S 能够通过所有的检验。

如同我们将要看到的那样,这些定性的思想可加以定量化。这个理论是相当微妙的,但它是充分漂亮和重要的。因此花上时间仔细地研究其细节的读者必将获益匪浅。

在以下的讨论中,把一个算法 A 在 N 个二进位串上的运行时间 $T(A)$ 定义为输出 $A(B)$ 预期需要的步骤数的极大值。取极大是对所有 $B \in \$N$ 进行的;预期数是对算法所做的所有硬币投掷取平均的。

在我们的定量分析中的头一步是来说明,我们可以把检验限制为有非常特殊类型的。令 A_k 是这样一算法,它只依赖于输入串 $B=B_1\cdots B_N$ 的头 k 个二进位,其中 $0 \leq k < N$,并设 $A_k^P(B) = (A_k(B) + B_{k+1} + 1) \bmod 2$ 。因此当且仅当 A_k 成功地预测 B_{k+1} 时 A_k^P 输出 1;我们称 A_k^P 为一个预测检验。

引理 P1 设 S 是一个 N 源,如果 S 不能通过带有容差 ϵ 的检验 A ,则存在一个整数 $k \in \{0, 1, \dots, N-1\}$ 和满足 $T(A_k^P) \leq T(A) + O(N)$ 的一个预测检验 A_k^P ,使得 S

不能通过带有容差 ϵ/N 的 A_k^P 。

证明 如果必要,通过补充 A 的输出,我们可以假定 $P(A, S) - P(A, \$N) \geq \epsilon$ 。考虑算法 F_k ,它通过投掷 $N-k$ 次硬币并且在执行 A 之前以随机二进位 $B'_{k+1} \cdots B'_N$ 来替代 $B_{k+1} \cdots B_N$ 。算法 F_N 和算法 A 是一样的,而 F_0 对 S 的动作就如同 A 对 $\$N$ 的动作那样。设 $p_k = P(F_k, S)$ 。由于 $\sum_{k=0}^{N-1} (p_{k+1} - p_k) = p_N - p_0 = P(A, S) - P(A, \$N) \geq \epsilon$, 因此存在某个 k 使得 $p_{k+1} - p_k \geq \epsilon/N$ 。

设 A_k^P 是实施 F_k 的计算并且预测值 $(F_k(B) + B'_{k+1} + 1) \bmod 2$ 的算法;换句话说,它输出

$$A_k^P(B) = (F_k(B) + B_{k+1} + B'_{k+1}) \bmod 2 \quad (39)$$

对概率的仔细分析表明 $P(A_k^P, S) - P(A_k^P, \$N) = p_{k+1} - p_k$ (见习题 40)。■

长度为 k 的每一个子串 $B_1 \cdots B_k, B_2 \cdots B_{k+1}, \dots, B_{N-k+1} \cdots B_N$ 都有相同的概率分布。在这个意义下,有实用价值的大多数 N 源都是移位对称的。例如,当 S 对应于(38)中的一个线性同余序列时,这一点成立。在这样一些情况下,我们通过取 $k = N-1$ 可对引理 P1 进行改进。

引理 P2 如果 S 是一个移位对称的 N 源,它不能通过带有容差 ϵ 的检验 A ,则存在满足 $T(A') \leq T(A) + O(N)$ 的一个算法 A' ,它以至少 $\frac{1}{2} + \epsilon/N$ 的概率从 $B_1 \cdots B_{N-1}$ 中预测 B_N 。

证明 如果 $P(A, S) > P(A, \$N)$, 设 A' 是在引理 P1 的证明中的 A_k^P , 但被应用于 $B_{N-k} \cdots B_{N-1} 0 \cdots 0$ 而不是 $B_1 \cdots B_N$ 。于是 A' 由于移位对称性,有相同的平均行为。如果 $P(A, S) < P(A, \$N)$, 设 A' 是在同样的方式下的 $1 - A_k^P$, 显然, $P(A', \$N) = 1/2$ 。■

通过假定序列 $B_1 B_2 \cdots B_N$ 的每一个在 X_0 取遍某个集合 X 时,都有 $f(g(X_0)) f(g(g(X_0))) \cdots f(g^{[N]}(X_0))$ 的形式,其中 g 是 X 的一个排列而且对于所有的 $x \in X$, $f(x)$ 为 0 或 1。我们进一步来确定 S 。对于 $X = \{0, 1, \dots, 2^e - 1\}$, $g(x) = (ax + c) \bmod 2^e$, 以及 $f(x)$ 等于 x 的最高有效位,我们的线性同余的例子满足这个限制。这样一种 N 源称为是迭代的。

引理 P3 如果 S 是一个迭代的 N 源,它不能通过带有容差 ϵ 的检验 A ,则存在一个满足 $T(A') \leq T(A) + O(N)$ 的算法 A' ,它以至少 $\frac{1}{2} + \epsilon/N$ 的概率从 $B_2 \cdots B_N$ 中预测 B_1 。

证明 一个迭代的 N 源是移位对称的,因此它的反射 $S^R = \{B_N \cdots B_1 \mid B_1 \cdots B_N \in S\}$ 也是。因此引理 P2 适用于 S^R 。■

在排列 $g(x) = (ax + c) \bmod 2^e$ 中,只要 a 为奇数,我们就可以由 $g(x)$ 确定 x ,

在这个意义下,它容易反演。但很多容易计算的排列函数却是“单向”的——即难以反演的——因而我们将看到,这使得它们成为可证实的拟随机数的好来源。

引理 P4 设 S 是对应于 f, g 和 X 的一个迭代的 N 源。如果 S 不能通过带有容差 ϵ 的检验 A , 则存在一个算法 G , 当 x 是 X 的一个随机元素时, 给定 $g(x)$, 它以大于等于 $\frac{1}{2} + \epsilon/N$ 的概率正确地猜测 $f(x)$ 。运行时间 $T(G)$ 至多是 $T(A) + O(N) \cdot (T(f) + T(g))$ 。

证明 给定 $y' = g(x)$, 所求算法 G 计算 $B_2 = f(g(x)), B_3 = f(g(g(x))), \dots, B_N = f(g^{[N-1]}(x))$, 并且应用引理 P3 的算法 A' , 它以大于等于 $\frac{1}{2} + \epsilon/N$ 的概率猜测 $f(x) = B_1$, 因为 g 是 X 的一个排列, 而且 $B_1 \dots B_N$ 是对应于种子值 X_0 的 S 的元素, 对于它我们有 $g(X_0) = x$ 。■

为了使用引理 P4, 我们需要把猜测单个二进位 $f(x)$ 的能力增大到猜测 x 本身的能力, 而只须给出 $g(x)$ 的值。如果我们扩充 S 使得需要猜测许多不同的函数 $f(x)$, 则使用布尔函数的性质, 有一个很好的一般的方法来做这件事。(然而, 这个方法是稍微技术性的, 因此头一次阅读的读者在仔细考察以下的细节之前可以跳到后面的定理 G 去。)

假设 $G(z_1, \dots, z_R)$ 是在 R 个二进位串上的一个二进制值的函数, 对于某个固定的 $x = x_1 \dots x_R$, 它在猜测形如 $f(z_1 \dots z_R) = (x_1 z_1 + \dots + x_R z_R) \bmod 2$ 的一个函数方面是很好的。通过计算预期值

$$s = E((-1)^{G(z_1 \dots z_R) + x_1 z_1 + \dots + x_R z_R}) \quad (40)$$

并对 $z_1 \dots z_R$ 的所有可能性取均值, 来测量 G 的成功是很方便的。这是正确猜测之和减去不正确的猜测, 除以 2^R ; 所以, 如果 p 是 G 正确的概率, 我们有 $s = p - (1 - p)$ 或 $p = \frac{1}{2} + \frac{1}{2}s$ 。

例如, 假设 $R=4$ 和 $G(z_1 z_2 z_3 z_4) = [z_1 \neq z_2][z_3 + z_4 < 2]$ 。如果 $x = 1100$, 此函数有成功率 $s = \frac{3}{4}$ (和 $p = \frac{7}{8}$)。因为对于所有 4 个二进位的串 z , 除 0111 或 1011 之外, 它等于 $x \cdot z \bmod 2 = (z_1 + z_2) \bmod 2$ 。当 $x = 0000, 0011, 1101$ 或 1110 时, 它也有成功率 $\frac{1}{4}$ 。所以对于 x 有 5 个似真的可能性。其他 11 个 x 使 $s \leq 0$ 。

在刚才描述的意义下, 当 G 是一个成功的猜测程序时, 下列算法在大多数情况下都奇迹般地发现 x 。更确切地说, 这个算法构造出极可能包含 x 的短的表。

算法 L(线性猜测的扩大) 给定一个二进制值函数 $G(z_1 \dots z_R)$ 和一个正整数 k , 本算法输出 2^k 个二进序列 $x = x_1 \dots x_R$ 的一个表而且有这样—个性质, 即当 $G(z_1 \dots z_R)$ 是函数 $(x_1 z_1 + \dots + x_R z_k) \bmod 2$ 的一个好的近似时, x 可能是输出。

L1. [构造一个随机矩阵] 对于 $1 \leq i \leq k$ 和 $1 \leq j \leq R$ 生成随机二进位 B_{ij} 。

L2. [计算符号] 对于 $1 \leq i \leq R$ 和对于所有的二进位串 $b = b_1 \cdots b_k$, 计算

$$h_i(b) = \sum_{c \neq 0} (-1)^{b \cdot c + G(cB + e_i)} \quad (41)$$

其中 e_i 是有 1 在位置 i 的 R 个二进位的串 $0 \cdots 010 \cdots 0$, 而且其中 cB 是有 $d_j = (B_{1j}c_1 + \cdots + B_{kj}c_k) \bmod 2$ 的串 $d_1 \cdots d_R$ 。(换句话说, 二进制向量 $c_1 \cdots c_k$ 乘以 $k \times R$ 二进制矩阵 B 。)求和是对于所有不等于 $0 \cdots 0$ 的所有 $2^k - 1$ 个二进位串 $c_1 \cdots c_k$ 进行的。利用对于阿达马变换的 Yates 方法, 对于每个 i 它可以用 $k \cdot 2^k$ 个加法和减法来计算。见等式 4.6.4-(38) 下面的说明。

L3. [输出猜测] 对于 $b = b_1 \cdots b_k$ 的所有 2^k 种选择, 输出串 $x(b) = [h_1(b) < 0] \cdots [h_R(b) < 0]$ 。■

为了证明算法 L 适当地工作, 我们必须证明, 一个给定的串 x 大概将是输出, 每当它值得成为输出时。首先注意, 如果我们把 G 变成 G' , 其中 $G'(z) = (G(z) + z_j) \bmod 2$, 则原来的 $G(z)$ 是 $x \cdot z \bmod 2$ 的一个好的近似当且仅当新的 $G'(z)$ 是 $(x + e_j) \cdot z \bmod 2$ 的好的近似时, 其中 e_j 是在步骤 L2 中定义的单位向量的串。而且, 如果我们应用算法到 G' 上以代替 G , 我们得到

$$h'_i(b) = \sum_{c \neq 0} (-1)^{b \cdot c + G(cB + e_i) + (cB + e_i) \cdot e_i} = (-1)^{\delta_i} h_i((b + B_j) \bmod 2)$$

其中 B_j 是 B 的列 j 。因此步骤 L3 输出向量 $x'(b) = x((b + B_j) \bmod 2) + e_j$, 对 2 取模。当 b 跑遍所有 k 个二进位的串时, $(b + B_j) \bmod 2$ 也是, 其效果是在输出中对每个 x 的位 j 取补。

因此我们只需证明, 每当 $G(z)$ 是常量函数 0 的一个好的近似时, 向量 $x = 0 \cdots 0$ 可能是输出。事实上, 我们将证明, 每当 $G(z)$ 相比之于为 1 更有可能为 0 时以及当 k 充分大时, 在步骤 L3 中 $x(0 \cdots 0)$ 以高的概率等于 $0 \cdots 0$ 。更精确地说, 当对于 z 的所有 2^k 种可能取平均值而且 k 足够大时, 如果 $s = E((-1)^{G(z)})$ 为正, 则对于 $1 \leq i \leq R$,

$$\sum_{c \neq 0} (-1)^{G(cB + e_i)} > 0$$

以大于 $1/2$ 的概率成立。

关键的发现是, 对于每个固定的 $c = c_1 \cdots c_k \neq 0 \cdots 0$, 串 $d = cB$ 一致地分布: d 的每个值以 $1/2^R$ 的概率出现, 因为 B 的二进位是随机的。其次, 当 $c \neq c' = c'_1 \cdots c'_k$ 时, 串 $d = cB$ 和 $d' = c'B$ 是独立的: 串偶 (d, d') 的每一个值以 $1/2^{2R}$ 的概率出现。因此, 如同在切比雪夫不等式的证明中那样, 我们可以论证, 对于任何固定的 i , 和 $\sum_{c \neq 0} (-1)^{G(cB + e_i)}$, 以至多 $1/((2^k - 1)s^2)$ 的概率为负。(习题 42 包含其细节。)由此得出, $R/((2^k - 1)s^2)$ 是在步骤 L3 中 $x(0)$ 非负的概率的上界。

定理 G 如果 $s = E((-1)^{G(z) + x \cdot z}) > 0$ 且 $2^k > \lceil 2R/s^2 \rceil$, 则算法 L 以大于等于 $1/2$ 的概率输出 x 。运行时间是 $O(k2^k R)$ 加上对 G 做 $2^k R$ 个计算的时间。■

现在我们已经做好了证明等式 3.2.2-(17) 的平方搅混序列是(拟)随机数的一个好的来源的准备。假设 $2^{R-1} < M = PQ < 2^R$, 其中 P 和 Q 是分别在范围 $2^{(R-2)/2} < P < 2^{(R-1)/2}$, $2^{R/2} < Q < 2^{(R+1)/2}$ 中形如 $4k+3$ 的素数。我们将称 M 是一个 R 个二进位的 Blum 整数。因为这样的数对于密码学的重要性首先是由 Manuel Blum 指出的 [COMPCON 24 (1982 年春), 133~137]。Blum 原来提议, P 和 Q 两者都有 $R/2$ 个二进位, 但是算法 4.5.4D 表明, 最好是如这里所述选择 P 和 Q , 使得 $Q - P > .29 \times 2^{R/2}$ 。

在 $0 < X_0 < M$ 的范围内随机地选择 X_0 且 $X_0 \perp M$; 也设 Z 是一个随机的 R 个二进位的屏蔽。通过设 X 是对于作为 (X_0, Z, M) 的可能性的所有 (x, z, m) 的集合, 并且加上进一步的限制, 即对于某个 a , $x \equiv a^2 \pmod{m}$, 我们可以构造一个迭代的 N 源 S 。容易证明函数 $g(x, z, m) = (x^2 \bmod m, z, m)$ 是 X 的一个排列 (例如, 请见习题 4.5.4-35)。在这个迭代的源中抽取二进位的函数 $f(x, z, m)$ 是 $x \cdot z \bmod 2$ 。我们的初始值 (X_0, Z, M) 不必在 X 中, 但 $g(X_0, Z, M)$ 是在 X 中一致分布的, 因为恰好有 4 个 X_0 的值有一个给定的平方 $X_0^2 \bmod M$ 。

定理 P 设 S 是在 R 个二进位的模上由平方搅混方法定义的 N 源, 并假设 S 不能通过带有容差 $\epsilon \geq 1/2^N$ 的某个统计检验 A 。于是我们可以构造一个这样的算法 F , 它找出有上面所描述的形式随机 R 个二进位的 Blum 整数 $M = PQ$ 的因子, 而且其成功概率至少为 $\epsilon/(4N)$, 其运行时间为 $T(F) = O(N^2 R^2 \epsilon^{-2} T(A) + N^3 R^4 \epsilon^{-2})$ 。

证明 乘法模 M 可以在 $O(R^2)$ 步内完成; 因此 $T(f) + T(g) = O(R^2)$ 。因此引理 P4 断言带有成功率 ϵ/N 的一个猜测算法 G 的存在性且 $T(G) \leq T(A) + O(NR^2)$ 。利用习题 41 的方法我们可以由 A 构造 G 。这个算法 G 有这样一个性质, 即 $s = E((-1)^{G(y, z, m) + z \cdot x}) \geq (\frac{1}{2} + \epsilon/N) - (\frac{1}{2} - \epsilon/N) = 2\epsilon/N$, 其中期望值是对所有 $(x, z, m) \in X$ 取的, 且其中 $(y, z, m) = g(x, z, m)$ 。

所要求的算法 F 进行如下: 给定一个随机的 $M = PQ$, 且 P, Q 是未知的, 它在 0 与 M 之间计算一个随机的 X_0 , 而且如果 $\gcd(X_0, M) \neq 1$ 就以一个已知的因子分解立即停止。否则它以 $G(z) = G(X_0^2 \bmod M, z, M)$ 和 $k = \lceil \lg(1 + 2N^2 R / \epsilon^2) \rceil$ 应用算法 L 。如果通过该算法所产生的 2^k 个值 x 的输出之一满足 $x^2 \equiv X_0^2 \pmod{M}$, 则 $x \not\equiv \pm X_0$ 的机会是 50:50; 于是 $\gcd(X_0 - x, M)$ 和 $\gcd(X_0 + x, M)$ 都是 M 的素因子。(请见 4.5.4 小节中 Rabin 的“SQRT 盒”。)

由于 $\epsilon \geq 2^{-N}$, 因此这个算法的运行时间显然是 $O(N^2 R^2 \epsilon^{-2} T(A) + N^3 R^4 \epsilon^{-2})$ 。它在对 M 进行因子分解中获得成功的概率可以估计如下: 设 $n = |X|/2^R$ 是对 (x, m) 进行选择的次数, 并设 $s_{xm} = 2^{-R} \sum (-1)^{G(y, z, m) + z \cdot x}$, 其中求和是对于所有 R 个二进位的数 z 进行的; 于是 $s = \sum_{x, m} s_{xm} / n \geq 2\epsilon/N$ 。设 t 是使得 $s_{xm} \geq \epsilon/N$ 的 (x, m) 的个数。我们的算法处理这样的数偶的概率是

$$\frac{t}{n} \geq \sum_{x, m} [s_{xm} \geq \epsilon/N] \frac{s_{xm}}{n} = \sum_{x, m} 1 - [s_{xm} < \epsilon/N] \frac{s_{xm}}{n} \geq$$

$$\frac{2\epsilon}{N} - \sum_{r,m} [s_{x,m} < \epsilon/N] \frac{s_{xm}}{n} \geq \frac{\epsilon}{N}$$

而且在这样一情况下由定理 G 它以大于等于 1/2 的概率找到 x , 因为我们有 $2^k > \lceil 2R/s_{xm}^2 \rceil$; 因此它以大于等于 1/4 的概率求出一个因子。■

从实用的观点看, 定理 P 意味着什么呢? 我们的证明显示, 由 O 所蕴涵的常数是很小的; 让我们假定进行因子分解的运行时间顶多为 $10(N^2 R^2 \epsilon^{-2} T(A) + N^3 R^4 \epsilon^{-2})$ 。世界上许多最伟大的数学家都已经在从事对大的数进行因子分解的研究, 特别是在 20 世纪 70 年代末期发现因子分解同密码学高度相关之后。由于它们还未找到一个好的解答, 我们有充分理由相信因子分解是很难的; 因此定理 P 将表明, 对于探测平方搅混二进位的非随机性来说, $T(A)$ 对于所有算法都是很大的。

长的计算可以方便地以机器指令年(MIP 年)来度量, 即由每 Gregorian 秒执行百万条指令的一台机器每 Gregorian 年执行的指令数, 这个数即 $31\,556\,952\,000\,000 \approx 3.16 \times 10^{13}$ 。1995 年, 利用最高性能算法分解一些 120 个十进位(即 400 个二进位)的数, 要花费多达 250 机器指令年以上。如果发现一个算法, 当 $R \rightarrow \infty$ 时, 它只要求 $\exp(R^{1/4}(\ln R)^{3/4})$ 条指令, 则从事因子分解的最乐观的研究者们将会大吃一惊。但是让我们假定, 这样一个突破已经实现, 因为至少对于 R 个二进位的 Blum 整数的一个不太小的因子是如此。那么我们就可在 2×10^{25} 个 MIP 年内分解大约 50 000 个二进位(15 000 个十进位)的数。如果对于 $R = 50\,000$, 通过平方搅混方法我们生成 $N = 1\,000$ 个随机二进位, 而且如果假定, 好到足以对 50 000 个二进位的 Blum 整数的至少 $\frac{1}{400000}$ 进行因子分解的算法必须至少运行 2×10^{25} MIP 年, 定理 P 告诉我们, 每个这样 1000 个二进位的集合将通过随机性的所有统计检验, 而其运行时间少于 70 000 机器指令年。没有这样的算法 A 将能以大于等于 $\epsilon = \frac{1}{100}$ 的概率来把这样的二进位同真正随机序列区别开来。

印象深刻吗? 否。这样一个结果几乎不会令人感到惊讶: 因为当 $R = 50\,000$ 时仅仅为了以 X_0, Z 和 M 来启动平方搅混方法, 我们就需要确定大约 150 000 个真正随机的二进位。从这样一个投资我们当然应该能够得到 1 000 个随机位的回报!

但是一般来说, 当 $\epsilon = \frac{1}{100}$ 时, 在我们保守的假定下, 这个公式变成

$$T(A) \geq \frac{1}{100000} N^{-2} R^{-2} \exp(R^{1/4}(\ln R)^{3/4}) - NR^2$$

当 R 很大时, NR^2 项可以忽略不计。所以让我们设 $R = 200\,000$ 和 $N = 10^{10}$ 。于是从 $3R \approx 600\,000$ 个真正随机的二进位, 我们得到 100 亿拟随机搅混位, 并且通过所有的统计检验。而所需要时间则少于 7.486×10^{10} MIP 年 = 74.86 吉 MIP 年。对于 $N = 10^{13}$ 和 $R = 333\,333$, 为探测任何统计偏倚所需要的计算时间增加到 53.5 太 (10^{12}) MIP 年。

如果因子分解是不可解的, 避免随机屏蔽 Z 的简单拟随机生成程序 3.2.2-

(16)也可被证明通过对于随机性的所有多项式时间的检验(见习题 4.5.4-13)。但是对更简单的方法已知的性能保证,要比平方搅混的性能保证稍微弱些;当前两者的关系是 $O(N^4 R \epsilon^{-4} \log(NR \epsilon^{-1}))$ 对定理 P 中的 $O(N^2 R^2 \epsilon^{-2})$ 。

每个人都相信对于 R 个二进位的数,没有这样的因子分解算法,其运行时间是关于 R 的多项式的。如果该猜测在一个更强的形式下成立,使得我们对于任何固定的 k ,都不能在多项式时间内对于 R 个二进位的 Blum 数的哪怕是其 $1/R^k$ 来进行因子分解,则定理 P 证明,平方搅混方法生成的拟随机数,能通过随机性的所有多项式时间的统计检验。

以另外一种方式来表述:如果对于适当地选定的 N 和 R ,你使用平方搅混方法生成随机二进位,那你将或者得到通过所有合理的统计检验的数,或者你获得发现新的因子分解算法的运气和荣誉。

G. 概述,历史和文献 我们已经定义了一个序列所可能具有的若干不同程度的随机性。

∞ 分布的无穷序列满足很多期望于随机序列的有用性质,而且有一套关于 ∞ 分布序列的内容充实的理论(后面的习题提出了 ∞ 分布的若干重要的性质,它们是正文中还未提到的)。因此,定义 R1 是随机性的理论研究的一个适当基础。

∞ 分布 b 进序列的概念是由 Emile Borel 于 1909 年引进的。他实际上定义了 (m, k) 分布序列的概念,而且他证明了几乎所有实数的 b 进表示对于所有 m 和 k 都是 (m, k) 分布的。他称这样的数对于基数 b 是完全正规的,而且他非正式地表述了定理 C,但没有明确认识到它需要证明[*Rendiconti Circ. Mat. Palermo* **27** (1909), 247~271, § 12]。关于这个课题的精彩讨论,出现于他的名著 *Lecons sur la Théorie des Fonctions*,第 2 版(1914),182~216 中。

实数的 ∞ 分布序列——也称做完备等分布序列——的概念,首先由 N. M. Korobov 发表于 *Doklady Akad. Nauk SSSR* **62** (1948), 21~22 的一篇笔记上。Korobov 与他的同事们在 20 世纪 50 年代的一系列文章中十分广泛地发展了关于这种序列的理论。Joel N. Franklin 独立地研究了完备等分布序列,见 *Math. Comp.* **17** (1963), 28~59。这是一篇特别值得注意的论文,因为它是由随机数生成的问题引起的。L. Kuipers 和 H. Niederreiter 所著 *Uniform Distribution of Sequences* (New York: Wiley, 1974)是一本关于所有类型的 k 分布序列的丰富数学论著的非常完备的信息源泉。

然而,我们看到, ∞ 分布序列并不一定要充分地杂乱才可以认为是“随机”的。上面描述的三个定义 R4, R5 和 R6 都提供了附加的条件,特别是定义 R6 看来是定义一个无穷随机序列概念的合适形式。它是一个同真正随机的直观概念完全相符的精确的定量陈述。

从历史上看,这些定义的发展主要是受了 R. von Mises 探索“概率”的一个好定义的影响。在 *Math. Zeitschrift* **5** (1919), 52~99 中, von Mises 提出了一个在想法上类似于定义 R5 的定义,但这个定义太苛刻了(像我们的定义 R3),以致不可能有满

足这些条件的序列存在。许多人注意到了这个矛盾, A. H. Copeland [Amer. J. Math. 50 (1928), 535~552] 建议, 通过采用他称做“可允许的数”(或伯努利序列), 来减弱 von Mises 的定义。这些都等价于这样的 ∞ 分布 $[0, 1)$ 序列, 即对于一个给定的概率 p , 序列中所有元素 U_n 都已用 1 (如果 $U_n < p$) 或用 0 (如果 $U_n \geq p$) 代替。于是, Copeland 实际上是建议退回到定义 R1 上。而后 Abraham Wald 指出没有必要如此急剧地减弱 von Mises 的定义, 他提出采用子序列规则的一个可数集合。在一篇重要的论文 *Ergebnisse eines math. Kolloquiums* 8 (Vienna: 1937), 38~72] 中, Wald 实际上证明了定理 W, 但他做出错误的论断说, 对于所有勒贝格可测的 $A \subseteq [0, 1)$, 由算法 W 构造的序列也满足更强的条件: $\Pr(U_n \in A)$ 等于 A 的测度。我们已经发现, 没有序列能满足这个性质。

当 Wald 写他的论文时, “可计算性”的概念尚完全处于幼年时代, 而 A. Church [Bull. Amer. Math. Soc. 46 (1940), 130~135] 说明了怎样可以把“能行算法”的精确思想加进 Wald 的理论中去, 以使他的定义成为完全严格的。对定义 R6 的推广实质上归功于 A. N. Kolmogorov [Sankhyā A25 (1963), 369~376], 他同时提出了对于有限序列的定义 Q2。关于有限序列随机性的另一个定义, 它“介于”定义 Q1 和 Q2 之间, 早在许多年前就已为 S. Besicovitch 所阐述 [Math. Zeitschrift 39 (1934), 146~156]。

Church 和 Kolmogorov 的著作仅仅考虑这样的二进序列, 即对于一给定的概率 p , 有 $\Pr(X_n = 1) = p$ 。我们在本节的讨论, 是稍微更一般的, 因为一个 $[0, 1)$ 序列实际上一次就表示所有的 p 。von Mises-Wald-Church 的定义已由 J. V. Howard 以另一种有趣的方式所改进, 见 *Zeitschr. für math. Logik und Grundlagen der Math.* 21 (1975), 215~224。

另一个重要的贡献是由 Donald W. Loveland 做出的 [Zeitschr. für math. Logik und Grundlagen der Math. 12 (1966), 279~294], 他讨论了定义 R4, R5, R6 和若干中间概念。Loveland 证明了存在不满足 R4 的 R5 随机序列, 由此确定了需要像 R6 这样更强的定义。事实上, 他定义了一个稍微简单的非负整数排列 $\langle f(n) \rangle$ 和类似于算法 W 的算法 W' , 使得当给出一个子序列规则的无穷集合 \mathcal{R}_k 时, 对于由算法 W' 产生的每一 R5 随机序列,

$$\bar{\Pr}\left(U_{f(n)} \geq \frac{1}{2}\right) - \Pr\left(U_{f(n)} \geq \frac{1}{2}\right) \geq \frac{1}{2}$$

尽管定义 R6 直观上比 R4 强得多, 但是要严格地证明这一点显然不是一件简单的事, 而且若干年来关于 R4 是否蕴涵 R6 一直是未解决的问题。最后 Thomas Herzog 和 James C. Owings, Jr. 发现了怎样构造满足 R4 而不满足 R6 的一大类序列 [见 *Zeitschr. für math. Logik und Grundlagen der Math.* 22 (1976), 385~389]。

Kolmogorov 写了另一篇重要的文章 [Problemy Peredači Informatsii 1 (1965), 3~11], 其中, 他考虑了定义一个序列的“信息内容”问题, 而这一工作促成了 Chaitin

和 Martin-Löf 通过“无模式性”来给出有限随机序列的有趣定义[见 *IEEE Trans. IT-14* (1968), 662~664]。这一思想也可以追溯到 R. J. Solomonoff, *Information and Control* 7 (1964), 1~22, 224~254; *IEEE Trans. IT-24* (1978), 422~432; *J. Comp. System Sci.* 55 (1997), 73~88。

关于随机序列的哲学讨论, 见 K. R. Popper, *The Logic of Scientific Discovery* (London, 1959), 特别是 162~163 页上的有趣构造, 是他在 1934 年首先发表的。

关于随机序列与递归函数之间的进一步联系, 已由 D. W. Loveland 所剖析, 见 *Trans. Amer. Math. Soc.* 125 (1966), 497~510。也见 C. - P. Schnorr [*Zeitschr. Wahr. verw. Geb.* 14 (1969), 27~35], 他发现随机序列和由 L. E. J. Brouwer 在 1919 年定义的“测度 0 类”之间的强烈关系。Schnorr 随后的书 *Zufälligkeit und Wahrscheinlichkeit* [*Lecture Notes in Math.* 218 (Berlin: Springer, 1971)] 给出了关于随机性的整个问题的详尽讨论, 而且对这个课题的一直增长着的最新文献也做了出色的介绍。李明和 Paul. M. B. Vitányi 在 *An Introduction to Kolmogorov Complexity and Its Applications* (Springer, 1993) 一书中, 对于此后的 20 年间的重要发展做了综述。

Manuel Blum, Silvio Micali 和 Andrew Yao [*FOCS* 23 (1982), 80~91, 112~117; *SICOMP* 13 (1984), 850~864] 奠定了拟随机序列和有效信息理论的基础。他们构造了通过所有能行的统计检验的明显序列。Blum 和 Micali 引进了“硬核二进位”的概念, 即使得 $f(x)$ 和 $g(x)$ 都可容易地计算 (尽管 $f(g^{[-1]}(x))$ 却不) 的布尔函数 f ; 他们的论文是引理 P4 的起源。Leonid Levin 进一步发展了这一理论 [*Combinatorica* 7 (1987), 357~363], 而后他和 Oded Goldreich [*STOC* 21 (1989), 25~32] 分析了诸如平方搅混这样一些算法并说明对一个屏蔽的类似使用在许多进一步的情况中产生硬核二进位。最后 Levin [*J. Symbolic Logic* 58 (1993), 1102~1103] 通过引进和分析算法 L 从而改进了该篇论文的诸方法。

许多作者都对这个理论做出了贡献——他们主要有 Impagliazzo, Levin, Luby 以及 Håstad, 他们证明 [*SICOMP* 28 (1999), 1364~1396] 拟随机序列可以由任何单向函数构造出来——但这里不讨论这样一些结果, 因为他们主要是用于抽象理论而不是实际的随机数生成。关于拟随机数性的理论工作在实践中含义首先是由 P. L'Ecuyer 和 R. Proulx 从经验上进行研究的, 见 *Winter Simulation Conf.* 22 (1989), 467~476。

If the numbers are not random,
they are at least higgledy-piggledy.

如果这些数不随机,
他们至少是杂乱无章的。

—GEORGE MARSAGLIA (1984)

习 题

1. $\lfloor 10 \rfloor$ 一个周期序列能是等分布的吗?

2. $\lfloor 10 \rfloor$ 考虑周期的二进序列 $0, 0, 1, 1, 0, 0, 1, 1, \dots$, 它是 1 分布的吗? 是 2 分布的吗? 是 3 分布的吗?

3. $\lfloor M22 \rfloor$ 构造一个 3 分布的周期三元序列.

4. $\lfloor HM14 \rfloor$ 对于任意两个命题 $S(n)$ 和 $T(n)$, 假定极限 $\Pr(S(n) \text{ 且 } T(n))$, $\Pr(S(n) \text{ 或 } T(n))$, $\Pr(S(n))$ 和 $\Pr(T(n))$ 中至少有三个存在, 试证明等式 $\Pr(S(n) \text{ 且 } T(n)) + \Pr(S(n) \text{ 或 } T(n)) = \Pr(S(n)) + \Pr(T(n))$ 成立. 例如, 如果一个序列是 2 分布的, 则我们将求得

$$\Pr(u_1 \leq U_n < v_1 \text{ 或 } u_2 \leq U_{n+1} < v_2) = v_1 - u_1 + v_2 - u_2 - (v_1 - u_1)(v_2 - u_2)$$

► 5. $\lfloor HM22 \rfloor$ 设 $U_n = (2^{\lfloor \log_2(n+1) \rfloor} / 3) \bmod 1$, 问 $\Pr(U_n < \frac{1}{2})$ 是多少?

6. $\lfloor HM23 \rfloor$ 设 $S_1(n), S_2(n), \dots$ 是一个关于互不相交事件命题的无穷序列, 即如果 $i \neq j$, 则 $S_i(n)$ 和 $S_j(n)$ 不能同时为真. 假设对于每一个 $j \geq 1$, $\Pr(S_j(n))$ 存在. 证明: $\Pr(\text{对于某个 } j \geq 1, S_j(n) \text{ 为真}) \geq \sum_{j \geq 1} \Pr(S_j(n))$, 并给出一个例子来说明等式不必成立.

7. $\lfloor HM27 \rfloor$ 设 $\{S_{ij}(n)\}$ 是对于所有 $i, j \geq 1$ 使得 $\Pr(S_{ij}(n))$ 存在的一族命题. 假定对所有的 $n > 0$, $S_{ij}(n)$ 恰巧对于一对整数 i, j 为真. 如果 $\sum_{i,j \geq 1} \Pr(S_{ij}(n)) = 1$, 则能否得出结论: 对于所有的 $i \geq 1$, $\Pr(S_{ij}(n) \text{ 对某个 } j \geq 1 \text{ 为真})$ 存在并且等于 $\sum_{j \geq 1} \Pr(S_{ij}(n))$?

8. $\lfloor M15 \rfloor$ 证明 (13).

9. $\lfloor HM20 \rfloor$ 证明引理 E [提示: 考虑 $\sum_{j=1}^m (y_{jn} - a)^2$].

► 10. $\lfloor HM22 \rfloor$ 在定理 C 的证明中, m 整除 q 这一事实用在什么地方?

11. $\lfloor M10 \rfloor$ 用定理 C 证明: 如果一个序列 $\langle U_n \rangle$ 是 ∞ 分布的, 则子序列 $\langle U_{2n} \rangle$ 也是.

12. $\lfloor HM20 \rfloor$ 试证 k 分布序列在下列的意义下通过“ k 的极大值”检验:

$$\Pr(u \leq \max(U_n, U_{n+1}, \dots, U_{n+k-1}) < v) = v^k - u^k$$

► 13. $\lfloor HM27 \rfloor$ 试证在下列意义下一个 ∞ 分布 $[0, 1]$ 序列通过“间隔检验”: 如果 $0 \leq \alpha < \beta \leq 1$ 且 $p = \beta - \alpha$, 则命 $f(0) = 0$, 而且对于 $n \geq 1$, 命 $f(n)$ 是使得 $\alpha \leq U_n < \beta$ 的最小整数 m ($m > f(n-1)$), 于是 $\Pr(f(n) - f(n-1) = k) = p(1-p)^{k-1}$.

14. $\lfloor HM25 \rfloor$ 试证在下列定义下一个 ∞ 分布序列通过“运行检验”: 如果 $f(0) = 0$ 而且对于 $n \geq 1$, $f(n)$ 是使得 $U_{m-1} > U_m$ 的最小整数 m ($m > f(n-1)$), 则

$$\Pr(f(n) - f(n-1) = k) = 2k/(k+1)! \cdot 2^{(k+1)/(k+2)!}$$

► 15. $\lfloor HM30 \rfloor$ 试证在下列定义下一个 ∞ 分布序列通过仅有两类赠券时的“集券检验”: 设 X_1, X_2, \dots 是一个 ∞ 分布序列. 令 $f(0) = 0$, 而且对于 $n \geq 1$, 令 $f(n)$ 是使得 $\{X_{f(n-1)+1}, \dots, X_m\}$ 是集合 $\{0, 1\}$ 的最小整数 m ($m > f(n-1)$). 证明对于 $k \geq 2$, $\Pr(f(n) - f(n-1) = k) = 2^{1-k}$ (参见习题 7).

16. $\lfloor HM38 \rfloor$ 当有两类以上的赠券时, 集券检验对于 ∞ 分布序列是否成立? (参考上题.)

17. $\lfloor HM50 \rfloor$ 如果 r 为任何给定的有理数, Franklin 已经证明, 序列 $\langle r^n \bmod 1 \rangle$ 不是 2 分布的.

但是有没有有理数 r 使得这个序列是等分布的呢? 特别是, 当 $r = \frac{3}{2}$ 时, 这个序列是等分布的吗? [参考 K. Mahler, *Mathematika* 4 (1957), 122-124.]

► 18. [HM22] 证明: 如果 U_0, U_1, \dots 是 k 分布的, 则序列 V_0, V_1, \dots 也是 k 分布的, 其中 $V_n = \lfloor nU_n \rfloor / n$ 。

19. [HM35] 考虑对定义 R4 的这样一个修改, 它要求子序列仅是“1 分布”, 而不是“ ∞ 分布”的。有无满足这个较弱定义的一个序列, 但它却不是 ∞ 分布的? (较弱的定义是否真正较弱?)

20. [HM36] (N. G. de Bruijn 和 P. Erdos) 任何有 $U_0 = 0$ 的 $[0, 1)$ 序列 $\langle U_n \rangle$ 的头 n 个点把区间 $[0, 1)$ 分成 n 个子区间; 设这些子区间有长度 $l_n^{(1)} \geq l_n^{(2)} \geq \dots \geq l_n^{(n)}$ 。显然 $l_n^{(1)} \geq \frac{1}{n} \geq l_n^{(n)}$, 因为 $l_n^{(1)} + \dots + l_n^{(n)} = 1$ 。测量 $\langle U_n \rangle$ 的分布的相等性的一个方法是考虑

$$\bar{L} = \limsup_{n \rightarrow \infty} n l_n^{(1)} \quad \text{和} \quad \underline{L} = \liminf_{n \rightarrow \infty} n l_n^{(n)}$$

a) 对于 van der Corput 序列 (29) 说来, 什么是 \bar{L} 和 \underline{L} ?

b) 证明对于 $1 \leq k \leq n$, $l_{n+k-1}^{(1)} \geq l_n^{(k)}$ 。利用这一结果来证明 $\bar{L} \geq 1/\ln 2$ 。

c) 证明 $\underline{L} \leq 1/\ln 4$ 。[提示: 对于每个 n , 有数 a_1, \dots, a_{2n} 使得对于 $1 \leq k \leq 2n$, $l_{2n}^{(k)} \geq l_{n+a_k}^{(n+a_k)}$ 。

而且, 在 $\{a_1, \dots, a_{2n}\}$ 中每个整数 $2, \dots, n$ 至多出现两次。]

d) 证明对于所有的 n , 由 $W_n = \lg(2n+1) \bmod 1$ 所定义的序列 $\langle W_n \rangle$ 满足 $1/\ln 2 > n l_n^{(1)} \geq n l_n^{(n)} > 1/\ln 4$ 。因此它实现最优的 \bar{L} 和 \underline{L} 。

21. [HM40] (L. H. Ramshaw)

a) 继续上一题, 序列 $\langle W_n \rangle$ 是等分布的吗?

b) 证明 $\langle W_n \rangle$ 是惟一这样的 $[0, 1)$ 序列, 对于它, 每当 $1 \leq k \leq n$ 时, 我们有

$$\sum_{j=1}^k l_n^{(j)} \leq \lg(1 + k/n)$$

c) 设 $\langle f_n(l_1, \dots, l_n) \rangle$ 是在 n 元组 $\{(l_1, \dots, l_n) \mid l_1 \geq \dots \geq l_n \text{ 和 } l_1 + \dots + l_n = 1\}$ 的集合上的连续函数的任何序列, 且满足以下两个性质:

$$f_{mn}\left(\frac{1}{m}l_1, \dots, \frac{1}{m}l_1, \dots, \frac{1}{m}l_n, \dots, \frac{1}{m}l_n\right) = f_n(l_1, \dots, l_n)$$

若对于 $1 \leq k \leq n$, $\sum_{j=1}^k l_j \geq \sum_{j=1}^k l'_j$, 则 $f_n(l_1, \dots, l_n) \geq f_n(l'_1, \dots, l'_n)$ 。[例子是: $n l_n^{(1)}$; $-n l_n^{(n)}$; $l_n^{(1)}/l_n^{(n)}$; $n(l_n^{(1)2} + \dots + l_n^{(n)2})$ 。] 设对于序列 $\langle W_n \rangle$,

$$\bar{F} = \limsup_{n \rightarrow \infty} f_n(l_n^{(1)}, \dots, l_n^{(n)})$$

证明对于所有的 n , 相对于 $\langle W_n \rangle$, $f_n(l_n^{(1)}, \dots, l_n^{(n)}) \leq \bar{F}$; 而且相对子其它每个 $[0, 1)$ 序列, $\limsup_{n \rightarrow \infty} f_n(l_n^{(1)}, \dots, l_n^{(n)}) \geq \bar{F}$ 。

► 22. [HM30] (Hermann Weyl) 证明 $[0, 1)$ 序列 $\langle U_n \rangle$ 是 k 分布的充分必要条件是: 对于每一组不全为 0 的整数 c_1, c_2, \dots, c_k ,

$$\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{0 \leq n < N} \exp(2\pi i(c_1 U_n + \dots + c_k U_{n+k-1})) = 0$$

23. [M32] (a) 证明每当 c_1, c_2, \dots, c_k 是不全为零的整数时, 当且仅当所有序列 $\langle (c_1 U_n + c_2 U_{n+1} + \dots + c_k U_{n+k-1}) \bmod 1 \rangle$ 是 1 分布时, $[0, 1)$ 序列 $\langle U_k \rangle$ 是 k 分布的。(b) 每当 c_1, c_2, \dots, c_k 是整数且 $\gcd(c_1, c_2, \dots, c_k) = 1$ 时, 当且仅当所有序列 $\langle (c_1 X_n + c_2 X_{n+1} + \dots + c_k X_{n+k-1}) \bmod b \rangle$ 是 1 分布时, 一个 b 进序列 $\langle X_n \rangle$ 是 k 分布的。

►24. [M35] (J. G. van der Corput) (a) 证明每当对于所有的 $k > 0$, 序列 $\langle (U_{n+k} - U_n) \bmod 1 \rangle$ 是等分布时, $[0, 1)$ 序列 $\langle U_n \rangle$ 是等分布的。(b) 因此, 当 $d > 0$ 和 α_d 是非有理数时, $\langle (\alpha_d n^d + \cdots + \alpha_1 n + \alpha_0) \bmod 1 \rangle$ 是等分布的。

25. [HM20] 如果所有的顺序相关系数均为 0, 即, 如果对所有 $k \geq 1$, 推论 S 中的等式均为真, 则说一个序列是“白序列”(由推论 S, 一个 ∞ 分布序列是白色的)。证明: 如果一个 $[0, 1)$ 序列是等分布的, 则它是白色的充要条件是

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{0 \leq j < n} \left(U_j - \frac{1}{2} \right) \left(U_{j+k} - \frac{1}{2} \right) = 0, \quad \text{对所有 } k \geq 1$$

26. [HM34] (J. Franklin) 像上题所定义的那样, 一个白序列可以完全不是随机的。设 U_0, U_1, \dots 是一个 ∞ 分布序列, 定义序列 V_0, V_1, \dots 如下:

$$\begin{aligned} (V_{2n-1}, V_{2n}) &= (U_{2n-1}, U_{2n}) && \text{如果 } (U_{2n-1}, U_{2n}) \in G \\ (V_{2n-1}, V_{2n}) &= (U_{2n}, U_{2n-1}) && \text{如果 } (U_{2n-1}, U_{2n}) \notin G \end{aligned}$$

其中 G 是集合 $\{(x, y) \mid x - \frac{1}{2} \leq y \leq x \text{ 或 } x + \frac{1}{2} \leq y\}$ 。

证明: (a) V_0, V_1, \dots 是等分布的和白色的; (b) $\Pr(V_n > V_{n+1}) = \frac{5}{8}$ (这指出序列相关检验的弱点)。

27. [HM48] 在一个等分布的白序列中, $\Pr(V_n > V_{n+1})$ 的最高可能值是多少? [D. Copper-smith 已经构造了一个这样的序列, 并取得了值 $7/8$ 。]

►28. [HM21] 用序列 (11) 来构造 3 分布的一个 $[0, 1)$ 序列, 使得 $\Pr(U_{2n} \geq \frac{1}{2}) = \frac{3}{4}$ 。

29. [HM34] 设 X_0, X_1, \dots 是一个 $(2k)$ 分布序列, 证明

$$\overline{\Pr}(X_{2n} = 0) \leq \frac{1}{2} + \binom{2k-1}{k} / 2^{2k}$$

►30. [M39] 构造一个 $(2k)$ 分布的二进序列, 并使得对于它 $\Pr(X_{2n} = 0) = \frac{1}{2} + \binom{2k-1}{k} / 2^{2k}$ 。

(因此, 上题中的不等式是最好的)。

31. [M30] 试证: 存在满足定义 R5 的一个 $[0, 1)$ 序列, 而且对于所有的 $n > 0$, $\nu_n/n \geq \frac{1}{2}$, 其中 ν_n 是使得 $U_j < \frac{1}{2}$ 的 $j < n$ 的个数 (这可以认为是序列的一个非随机性质)。

32. [M24] 令 $\langle X_n \rangle$ 是按照定义 R5 为“随机”的一个 b 进序列, 而且 \mathcal{R} 是确定一无穷子序列 $\langle X_n \rangle_{\mathcal{R}}$ 的可计算子序列规则, 证明该子序列不仅是 1 分布的, 而且按定义 R5 它是“随机的”。

33. [HM22] 设 $\langle U_{r_n} \rangle$ 和 $\langle U_{s_n} \rangle$ 是一个序列 $\langle U_n \rangle$ 的无穷不相交子序列 (于是 $r_0 < r_1 < r_2 < \dots$ 和 $s_0 < s_1 < s_2 < \dots$ 都是整数的递增序列, 而且对于任何 $m, n, r_m \neq s_n$)。设 $\langle U_{t_n} \rangle$ 是组合成的子序列, 它使得 $t_0 < t_1 < t_2 < \dots$, 并且使集合 $t_n = \{r_n\} \cup \{s_n\}$ 。试证: 如果 $\Pr(U_{r_n} \in A) = \Pr(U_{s_n} \in A) = p$, 则 $\Pr(U_{t_n} \in A) = p$ 。

►34. [M25] 定义子序列规则 $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \dots$ 使得借助这些规则可以使用算法 W, 以给出一个能行算法来构造满足定义 R1 的 $[0, 1)$ 序列。

►35. [HM35] (D. W. Loveland) 证明: 如果一个二进序列 $\langle X_n \rangle$ 是 R5 随机的, 而且如果 $\langle s_n \rangle$ 是如

同定义 R4 中那样的任何可计算序列, 则 $\overline{\text{Pr}}(X_n = 1) \geq \frac{1}{2}$ 且 $\text{Pr}(X_n = 1) \leq \frac{1}{2}$ 。

36. [HM30] 设 $\langle X_n \rangle$ 是按照定义 R6 为“随机”的一个二进制序列。试证, 按照方案 $U_0 = (0, X_0)_2, U_1 = (0, X_1 X_2)_2, U_2 = (0, X_3 X_4 X_5)_2, U_3 = (0, X_6 X_7 X_8 X_9)_2, \dots$

在二进制号下定义的 $[0, 1]$ 序列 $\langle U_n \rangle$ 在定义 R6 的意义下是随机的。

37. [M37] (D. Coppersmith) 试定义满足定义 R4 但不满足定义 R5 的一个序列。[提示: 在一个真正的随机序列中考虑改变 $U_0, U_1, U_4, U_9, \dots$]

38. [M49] (A. H. Kolmogorov) 给定 N, n, ϵ 和算法集合 A , 使得相对于 A 不存在长度为 N 的 (n, ϵ) 随机二进制序列的 A 中算法的最小个数是多少 (如果不能给出精确的公式, 能否找到渐近公式? 问题的要点是去发现界 (37) 如何接近于成为“最佳可能的界”)?

39. [HM45] (W. M. Schmidt) 设 U_n 是一个 $[0, 1]$ 序列, 并设 $\nu_n(u)$ 是使 $0 \leq U_j < u$ 的非负整数 j ($j \leq n$) 的个数。证明有一个正常数 c 使得对于任何 N 和任何 $[0, 1]$ 序列 $\langle U_n \rangle$, 对于满足 $0 \leq n < N, 0 \leq u < 1$ 的某个 n 和 u 有

$$|\nu_n(u) - un| > c \ln N$$

(换句话说, 没有 $[0, 1]$ 序列可能是太等分布的。)

40. [M28] 完成引理 P1 的证明。

41. [M21] 引理 P2 证明一个预测检验的存在性, 但它的证明依赖于一个适当的 k 的存在性, 却没有说明我们如何可以从 A 构造性地求出 k 。试证明任何算法 A 都可能转换成这样一个算法 A' , 且有 $T(A') \leq T(A) + O(N)$; 对于任何移位对称 N 源 S , 它以至少为 $\frac{1}{2} + (P(A, S) - P(A, S_N))/N$ 的概率从 $B_1 \dots B_{N-1}$ 预测 B_N 。

► 42. [M28] (成对独立性)

a) 设 X_1, \dots, X_n 是对于 $1 \leq j \leq n$ 有均值 $\mu = E X_j$ 和方差 $\sigma^2 = E X_j^2 - (E X_j)^2$ 的随机变量, 在下列附加假定, 即每当 $i \neq j$ 时 $E(X_i X_j) = (E X_i)(E X_j)$ 之下, 证明切比雪夫不等式

$$\text{Pr}((X_1 + \dots + X_n - n\mu)^2 \geq t n \sigma^2) \leq 1/t$$

b) 设 B 是一个随机的 $k \times R$ 二进制矩阵。证明如果 c 和 c' 是固定的非零 k 个二进制位的向量, 且 $c \neq c'$, 则向量 cB 和 $c'B$ 是独立随机 R 个二进制位向量 (模 2)。

c) 应用 a) 和 b) 到算法 L 的分析上。

43. [20] 看起来求任何固定的 R 个二进制 Blum 整数 M 的因子和求一个随机的 R 个二进制整数的因子一样地难。那么为什么定理 P 是对随机的 M 而不是对固定的 M 论述呢?

44. [16] (I. J. Good) 一张正确的随机数字表能仅含一个印刷错误吗?

3.6 小结

在这一章中我们叙述了相当大量的课题: 怎样生成随机数, 怎样检验它们, 怎样在应用中修正它们, 以及怎样推导关于它们的理论等。也许在许多读者的心目中, 主要的问题是: “什么是整个理论的结果? 什么是在我的程序中可以使用的简单而良好的生成程序, 以便有一个可靠的随机数的来源?”

这一章中的详细研究表明, 下列过程给出了对于大多数计算机的机器语言来说是最“最好”和“最简单”的随机数生成程序: 在程序的开头, 把一个整型变量 X 置为某

个值 X_0 。变量 X 仅用于随机数生成的目的。每当程序要求一个新的随机数时,即置

$$X \leftarrow (aX + c) \bmod m \quad (1)$$

并且使用 X 的新值作为随机值。应该适当地选择 X_0, a, c 和 m , 而且按下列原则明智地使用随机数:

(i) “种子”数 X_0 可以任意地选择。如果这个程序运行若干次, 而且每次都希望有不同的随机数来源, 则置 X_0 为上次赋予运行中由 X 得到的最后值; 或者(如果更方便的话)置 X_0 为当前的日期和时间。如果这个程序以后可能要以同样的随机数重新运行(例如当调试时), 而又不知道 X_0 的值是什么, 就应该打印出来看看。

(ii) 数 m 应是大的, 比如说至少是 2^{30} 。取计算机的字长可能是方便的, 因为这会使 $(aX + c) \bmod m$ 的计算十分高效。3.2.1.1 小节更详细地讨论了 m 的选择。 $(aX + c) \bmod m$ 的计算必须精确, 不带舍去误差。

(iii) 如果 m 是 2 的一个乘方(即如果正使用一台二进制计算机), 则可挑选 a 使得 $a \bmod 8 = 5$ 。如果 m 是 10 的一个乘方(即如果正使用一台十进制计算机), 则可选择 a 使得 $a \bmod 200 = 21$ 。 a 的这一选择和以下给出的 c 的选择一起, 保证了这个随机数生成程序在它开始重复以前, 产生 X 的全部 m 个可能的不同值(见 3.2.1.2 小节)以及保证了高的“效能”(见 3.2.1.3 小节)。

(iv) 乘数 a 选择在 $.01m$ 和 $.99m$ 之间是可取的, 而且它的二进表示或十进表示数字不应有一个简单的正规的模式。通过选择某些像 $a = 3141592621$ 这样的任意常数(它同时满足(iii)中的两个条件), 几乎总能得到相当好的乘数。如果要广泛使用这个随机数生成程序, 当然应该做进一步的检验; 例如, 当使用欧几里得算法来求 a 和 m 的 gcd(见 3.3.3 小节)时, 不应有太大的商。在乘数被认为真正合格之前它应通过谱检验(3.3.4 小节)和 3.3.2 小节的若干检验。

(v) 当 a 是一个好的乘数时, 除了当 m 是计算机的字的大小时不能和 m 有公因子外, c 的值是无所谓的。因此我们可以选择 $c = 1$ 或 $c = a$ 。许多人已经把 $c = 0$ 和 $m = 2^e$ 放在一起使用, 但它们牺牲两位精度和一半的种子值, 却只不过节省了几纳秒的运行时间(请见习题 3.2.1.2-9)。

(vi) X 的最低位(右边)数字不是非常随机的, 所以以数 X 为基础的判定总是主要受最高位的影响。一般说来, 最好是把 X 当做 0 与 1 之间的随机分数 X/m , 即假定在 X 的左边有一个小数点, 而不把 X 看做 0 与 $m-1$ 之间的一个随机整数。为了计算 0 与 $k-1$ 之间的一个随机整数, 人们应该乘之以 k , 并截取这结果的整数部分(不要除以 k , 见习题 3.4.1-3)。

(vii) 3.3.4 小节中讨论了关于序列(1)的随机性的一个重要极限, 其中证明了在 t 维中的“精度”大约仅是 \sqrt{m} 中的一部分。利用 3.2.2 小节中所讨论的一些技术, 要求较高分辨率的蒙特卡罗的应用可以改进随机性。

(viii) 至多应生成大约 $m/1000$ 个数, 否则未来的特性将越来越像过去那样; 如果 $m = 2^{32}$, 这意味着在消耗了几百万个随机数之后应该采取一个新的方案(例如一

个新的乘数 a)。

上述评述主要适用于机器语言的编码。对于程序设计来说,某些思想在高级程序设计语言中也很有效;例如,如果 x 是无符号长类型(unsigned long)的而且 m 是无符号长类型的算术运算的模数(通常是 2^{32} 或 2^{64}),则在 C 语言中(1)就变成“ $x = a * x + c$ ”。但 C 语言不像上面的(vi)中所要求的那样,为我们提供把 x 当做一个分数的好方法,除非我们转换成双精度的浮点数。

因此(1)的另外的形式经常被用于像 C 这样的语言当中:我们把 m 选择成一个接近易于计算的最大整数的素数,而且令 a 是 m 的一个原根;此种情况的适当增量 c 是零。于是(1)可以利用习题 3.2.1.1-9 的技术,通过对保持在 $-m$ 和 $+m$ 之间的数的简单算术运算完全地实现。例如,当 $a = 48271$ 和 $m = 2^{32}-1$ (见表 3.3.4-1 中的行 20),我们可以下述的 C 程序计算 $X \leftarrow aX \bmod m$:

```
#define MM 21474783647      /* 一个梅森素数 */
#define AA 48271             /* 这在谱检验中干得很好 */
#define QQ 44488             /* MM/AA */
#define RR 3399              /* MM % AA;重要的是 RR < QQ */
X = AA * (X % QQ) - RR * X/QQ;
if (X < 0) X += MM;
```

这里 x 是长类型的,因此 x 应该被初始化为小于 MM 的一个非零种子值。由于 MM 是素数, x 的最低有效位和最高有效位一样随机。所以不再需要采取(vi)的预设措施。

如果需要数以亿计的随机数,可以通过写类似于下面的附加的代码,如同在等式 3.3.4-(38)中那样,把这个程序同另一个程序组合在一起。

```
#define MMM 2147483399       /* 一个非梅森素数 */
#define AAA 40692            /* 另一个谱检验的成功故事 */
#define QQQ 52774           /* MMM/AAA */
#define RRR 3791             /* MMM % AAA;再次小于 QQQ */
Y = AAA * (Y % QQQ) - RRR * (Y/QQQ);
if (Y < 0) Y += MMM;
Z = X - Y; if (Z <= 0) Z += MM;
```

和 x 一样,变量 y 开始时需要是非零的。这个代码稍微偏离 3.3.4-(38)使得输出 z 总不为零; z 总是严格地处于 0 和 2^{31} 之间。 z 序列的周期长度大约是 74×10^{24} ,因此它的数字的精度差不多是 x 的数字的精确度的两倍。

这个方法是可移植的而且相当简单,但不是很快的。基于延搁斐波那契序列和减法(习题 3.2.2-23)的另外一个方案是更吸引人的,因为它不仅允许在计算机之间的可移植性,而且它要快得多。它还派送质量更好的随机数,因为对于 $t \leq 100$ 来说 t 维精度大概是好的。以下是一个 C 子程序 `ran_array(long aa[], int n)`,它使用递推式

$$X_j = (X_{j-100} - X_{j-37}) \bmod 2^{30} \quad (2)$$

生成 n 个新的随机数并且把它们放在一个给定的数组 aa 中。这个递推式特别适合于现代计算机。 n 的值必须至少是 100; 建议使用像 1000 这样更大的值。

```
#define KK 100 /* 长的延搁 */
#define LL 37 /* 短的延搁 */
#define MM (1L<<30) /* 模数 */
#define mod_diff(x,y) (((x)-(y))&(MM-1)) /* (x-y)mod MM */
long ran_x[KK]; /* 生成程序状态 */
void ran_array(long aa[],int n) {
    register int i,j;
    for (j=0;j<KK;j++) aa[j]=ran_x[j];
    for (;j<n;j++) aa[j]=mod_diff(aa[j-KK],aa[j-LL]);
    for (i=0;i<LL;i++,j++) ran_x[i]=mod_diff(aa[j-KK],aa[j-LL]);
    for (;i<KK;i++,j++) ran_x[i]=mod_diff(aa[j-KK],ran_x[i-LL]);
}
```

所有通过未来对 ran_array 的调用而将生成的关于数的信息都出现于 ran_x 中,所以如果要在以后的某个时候重新启动而不必全程跑回到这个序列的开始处,你可以在一个计算的当中做这个数组的一个备份。当然,使用像(2)这样的递推式的技巧是通过建立适当的值 X_0, \dots, X_{99} , 首先使每件事情都适当地开始。当给定 0 和 $2^{30}-3=1\,073\,741\,821$ (含)之间任何种子数时,下列子程序 $ran_start(long\ seed)$ 能够很好地初始化生成程序:

```
#define TT 70 /* 保证流之间的分开 */
#define is_odd(x) ((x)&1) /* x 的单位的二进位 */
#define evenize(x) ((x)(MM-2)) /* 使 x 成为偶数 */
void ran_start(long seed) /* 使用这建立 ran_array */
{
    register int t,j;
    long x[KK+KK-1]; /* 预备缓冲 */
    register long ss=evenize(seed+2);
    for (j=0;j<KK;j++) {
        x[j]=ss; /* 缓冲的自导引 */
        ss<<=1; if (ss>=MM) ss-=MM-2; /* 循环移 29 个二进位 */
    }
    for (;j<KK+KK-1;j++) x[j]=0;
    x[1]++; /* 使 x[1] (且仅 x[1]) 为奇 */
    ss=seed&(MM-1); t=TT-1; while (t) {
```



```

for (j = KK - 1; j > 0; j--) x[j + j] = x[j];           /* “平方” */
for (j = KK + KK - 2; j > KK - LL; j -= 2) x[KK + KK - 1 - j] = evenize(x[j]);
for (j = KK + KK - 2; j > = KK; j--) if (is_odd(x[j])) {
    x[j - (KK - LL)] = mod_diff(x[j - (KK - LL)], x[j]);
    x[j - KK] = mod_diff(x[j - KK], x[j]);
}
if (is_odd(ss)) {                                       /* “乘以 z” */
    for (j = KK; j > 0; j--) x[j] = x[j - 1];
    x[0] = x[KK];                                     /* 循环移缓冲区 */
    if (is_odd(x[KK])) x[LL] = mod_diff(x[LL], x[KK]);
}
if (ss) ss >> = 1; else t--;
}
for (j = 0; j < LL; j++) ran_x[j + KK - LL] = x[j];
for (; j < KK; j++) ran_x[j - LL] = x[j];
}

```

习题 9 中说明了 *ran_start* 有些奇怪的操作,它证明由不同起始种子生成的数的序列是彼此独立的:在 *ran_array* 的子序列输出中每 100 个连续值 $X_n, X_{n+1}, \dots, X_{n+99}$ 的块将不同于通过另外的种子而出现的块。(严格地说,仅当 $n < 2^{70}$ 时才知道这是对的;但一年里也不过 2^{55} ns。)若干过程因此可通过不同的种子并行地开始并且应确保它们进行独立的计算;在不同的计算机中心工作在同一问题的不同科学家群体可以保证,他们不重复其他人的工作,如果他们把自己限定在不相交的种子组的话。因此单个的例程 *ran_array* 和 *ran_start* 提供了 10 亿个以上实质上不相交的随机数组。而且如果这还不足够,你还可以从表 3.2.2-1 中选取其它值来代替程序参数 100 和 37。

为有效起见,这些 C 程序使用位的与操作“&”,所以它们并非严格地可移植的,除非计算机对于整数使用 2 的补码表示。几乎所有的现代计算机都是基于 2 的补码算术的,但是“&”对于这个算法说来并非真正必要。习题 10 说明不使用这样的技巧,怎样在 FORTRAN 中得到完全相同的数的序列。尽管在这里示出的程序被设计成生成 30 个二进位的整数,但也很容易把它们修改成生成在有可靠的浮点算术运算的计算机上的 0 和 1 之间的 52 个二进位的分数,请见习题 11。

你可能希望把 *ran_array* 包括在一个子程序库中,或者你可能发现已有另外某个人已经这样做了。检查 *ran_array* 和 *ran_start* 的一个实现是否同上面给出的一致的一个方法是运行下列未完善的检查程序:

```

int main() { register int m; long a[2009];
    ran_start(310952);

```

```

for (m = 0; m < 2009; m++) ran_array(a, 1009);
printf("%1d\n", ran_x[0]);
ran_start(310952)
for (m = 0; m < 1009; m++) ran_array(a, 2009);
printf("%1d\n", ran_x[0]); return 0;
}

```

打印输出应是 461390032(两次)。

告诫:由 *ran_array* 生成的数通不过 3.3.2J 小节中的生日间隔检验,而且它们还有其它一些缺点,这些缺点有时在高分辨率的模拟中出现(请见习题 3.3.2-31 和 3.3.2-35)。避免生日间隔问题的一个方法是简单地只使用一半的数(跳过奇数编号的数);但这并不解决其它问题。一个更好的方法是如同在 3.2.2 小节中那样,遵循 Martin Lüscher 的建议:使用 *ran_array* 来生成,比如说 1009 个数,但只使用这些数中的头 100 个(见习题 15)。这个方法有适度的理论支持而且没有已知的那些缺点。大多数用户将不需要这样的预防措施,但它肯定地是较少风险的,而且允许在随机性和速度方面方便地折中。

关于像(1)这样的线性同余序列已经知道很多了,但是关于像(2)这样的延搁斐波那契序列的随机性性质已经证明的东西相对要少些。在实际中这两个方法似乎都是可靠的,假如以已经论述的那些告诫来使用它们的话。

当在 20 世纪 60 年代刚开始写本章时,在世界上大多数计算机上普遍使用称做 *RANDU* 的真正可怕的随机数生成程序(请见 3.3.4 小节)。许多对于随机数生成科学做出贡献的作者通常不知道他们所使用的一些特定方法将被证明是不适当的。一个特别值得注意的例子是 Alan M. Ferrenberg 和他的同事们在 *Physical Review Letters* **69** (1992), 3382~3384 上报告的经验:他们通过首先考虑有一个已知答案的相关的二维问题,来检验对于一个三维问题他们的算法,从中发现,设想的高质量的现代随机数生成程序在第五个十进数位置中给出错误的结果。而形成对照的是,一个老式的历经磨炼的线性同余生成程序 $X \leftarrow 16807X \bmod (2^{31}-1)$ 却工作得很好。或许进一步的研究将表明甚至这里推荐的一些生成程序也是不能令人满意的;我们希望情况不是这样,但这个课题的历史警告我们要谨慎。对于每个人说来最谨慎的做法是在认真地采用程序的答案之前,至少再次使用十分不同的随机数来源,来运行每个蒙特卡罗程序;这不仅给出对于结果的稳定性的一个指示,而且也防止信任有隐藏缺陷的生成程序的危险(每一个随机数生成程序至少将在一个应用中失灵)。

关于随机数生成的文献,1972 年以前已由下述人员编辑了很好的文献目录: Richard E. Nance 和 Claude Overstreet, Jr., *Computing Reviews*, **13** (1972), 495~508; E. R. Soley, *International Stat. Review* **40** (1972), 355~371。1972 年—1984 年期间则由 Soley 作了报道, *International Stat. Review* **46** (1978), 89~102; J. Royal Stat. Soc. **A149** (1986), 83~107。随后的发展由 Shu Tezuka 在 *Uniform Random Numbers* (Boston: Kluwer, 1995) 上作了讨论。

关于在数值分析中随机数的使用的详细研究,请见 J. M. Hamersley 和 D. C. Handscomb, *Monte Carlo Methods* (London: Methuen, 1964)。这本书说明,通过使用专门为某种目的设计的“亚随机”数(不必满足我们已经讨论的一些统计检验)可加强某些数值方法。N. Metropolis 和 R. Eckhardt 在 *Los Alamos Science* **15** (1987), 125~136) 的专集 *Stanislaw Ulam 1909-1984* 中,讨论了用于计算机的蒙特卡罗方法的起源。

鼓励每一位读者都来做一做下面这组问题中的习题 6。

习 题

1. [21] 用方法(1)写出一个具有如下特征的 MIX 子程序:

调用程序: JMF RANDI

入口条件: $rA = k$, 一个小于 5000 的正整数

出口条件: $rA \leftarrow$ 一个随机整数 Y , $1 \leq Y \leq k$, 而且每个整数是同等可能的; $rX = ?$; 不考虑溢出

► 2. [15] 某些人害怕计算机将在某一大统治整个世界;但他们由于这样一个论点而放心了,即一台机器不能做任何真正新的事情,因为它仅仅服从于它的主人,即程序员的命令。Lovelace 女士于 1844 年写到:“分析机并不能主动做任何事情,它只能做我们知道如何命令它去做的那些事情。”她的论点已经进一步为许多哲学家所精心推敲。试从随机数生成程序的观点来讨论这一问题。

3. [32] (一个骰子游戏) 写出一个程序,它模拟两个骰子的投掷,其中每一个以相等的概率取值 1, 2, ..., 6。如果在头一次投掷时总数是 7 或 11, 则游戏为胜;总数是 2, 3 或 12 为败;对于任何其它的总数,称该总数为“点”,然后继续投掷骰子,直至或者出现一个 7(失败)或者再次出现点(胜利)。

玩十次游戏,骰子每次投掷的结果应以 mn 形式打印,其中 m 和 n 是两个骰子的内容,后边再附加以适当的解释(比方说,“对幺”或“四点”或“对二”,等等)。

4. [40] (单人纸牌游戏) 有些人花费大量宝贵的时间来玩单人纸牌游戏,也许自动化将大量侵入这个领域。试写出一个程序:(a)洗一副模拟的纸牌;(b)根据这副牌中纸牌的顺序,玩某种普通的单人纸牌游戏;(c)打印出这个游戏的结果,即怎样使这个程序以获胜结束。应该保持累计的统计数字,需要时要使程序能“作弊”。

5. [46] (由计算机实现创造性写作) 1960 年 10 月 26 日哥伦比亚广播公司(CBS)电视网播出名为“思想机器”的一个电视节目(穿插在其它节目当中),连续放映了两场西方风格的独幕剧,它们是由一个计算机程序写成的。下边是用计算机打印出来的两个剧本。

故事 #1 (枪在右手;钱在左手;玻璃杯在桌子上;瓶子在桌子上;手枪套在强盗身上,警长的枪在警长的右手中;警长的手枪套在警长的身上。)

强盗:(强盗在窗户旁)走向门;开门;进门;关门;走到角落处;把钱放到角落处;走向桌子;把枪放在桌上;坐下,心情烦躁;坐在桌旁;右手拿起玻璃杯(空的);把玻璃杯放在桌上;右手拿起瓶子;倒;把瓶子放在桌上;右手拿起玻璃杯;举起玻璃杯饮;把玻璃杯放在桌上;右手拿起瓶子;坐在桌旁;走向角落;走向窗户;走向桌边,把瓶子放到桌上;坐下,心情烦躁;坐在桌旁;坐着,心情烦躁;走向窗户;走向桌子;右手拿起玻璃杯。

警长:(警长在窗户旁)看着强盗;(强盗看着警长);走向门。

强盗:把玻璃杯放到桌上;右手拿起枪;检查枪。

警长:等候;开门;看着强盗;(强盗看着警长);进门。

强盗:走向窗户;瞄准;开枪;警长躲过。

警长:走向窗户;瞄准;开枪;未中;走向门;走向窗户。

强盗:走向门;瞄准;瞄准。

警长:瞄准;开枪;未中。

强盗:开枪;警长躲过。

警长:走向门;瞄准;开枪;未中;出门;瞄准。

强盗:瞄准;开枪;未中。瞄准;开枪;未中。

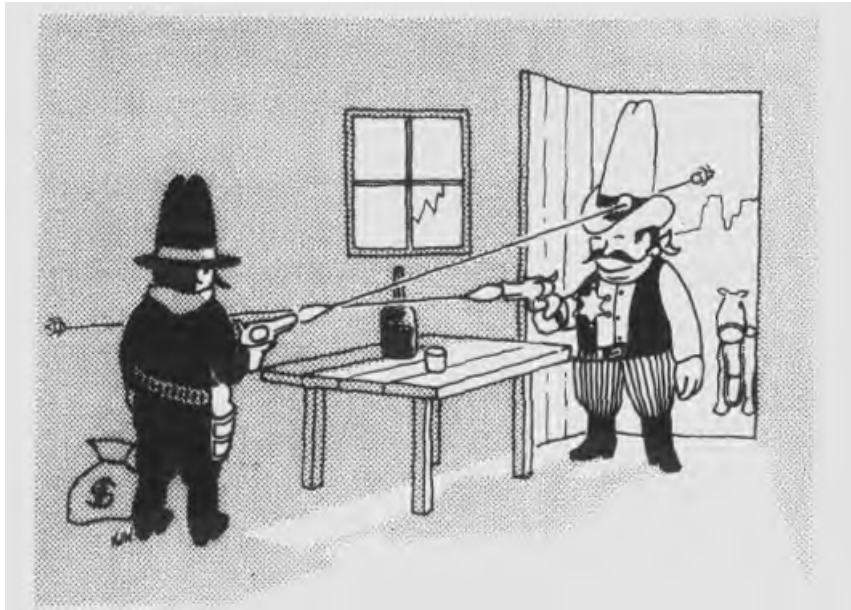
警长:开枪;未中;走向窗户;瞄准;开枪;未中。

强盗:瞄准;开枪;未中;瞄准;开枪;未中;瞄准;开枪;警长躲过。

警长:瞄准;开枪;强盗中弹。

强盗:枪落地;强盗死。

警长:把枪放入枪套中;走向桌子;右手拿起玻璃杯(空的);把右手的玻璃杯换到左手;右手拿起瓶子;倒;把瓶子放到桌上;玻璃杯从左手换到右手;举玻璃杯饮;玻璃杯从右手换到左手;右手拿起瓶子;倒;把瓶子放到桌上;玻璃杯从左手换到右手;举玻璃杯饮;把玻璃杯放到桌上;走向角落;右手拿起钱;走向门;过门;关门;幕落。



故事#2 (枪在右手;钱在左手;玻璃杯在桌上;瓶子在桌上;手枪套在强盗身上;警长的枪在警长的右手口;警长的手枪套在警长身上。)

强盗:(强盗在窗户旁边)走向门;开门;过门;关门;走到角落;在角落处把钱放下;走向窗户;在窗户旁把枪放下;靠着窗户观望;靠着窗户观望;走向角落;数钱;走向桌子;右手拿起玻璃杯(空的);把玻璃杯从右手换到左手;右手拿起瓶子;倒;把瓶子放到桌上;把玻璃杯从左手换到右手;举玻璃杯饮;把玻璃杯放到桌上;右手拿起瓶子;倒;走向角落;在角落处把瓶子放下;走向窗户;右手拿起枪;验枪;把枪放进手枪套;走向桌子,右手拿起玻璃杯;举玻璃杯饮;走向窗户;在窗户旁把玻璃杯放下。

警长:(警长在窗户旁)看着强盗;(强盗看着警长);走到门口。

强盗:右手从手枪套取枪;验枪;走到门口;验枪;在门口把枪放下。

警长:开门;看着强盗;(强盗看着警长);过门;走向窗户。

强盗:右手拿起枪。

警长:走到桌旁。

强盗:瞄准;开枪;未中;瞄准;开枪;警长中弹;吹枪管;把枪放进手枪套。

警长:枪落地;警长死去。

强盗:走到角落;右手拿起钱袋;走到门口;出门;关门;幕落。

小心阅读上面的剧本会发现,这里有高度激烈的戏剧场面。计算机程序仔细记住每个演员的位置,他手中的东西等等。演员所采取的动作是随机的,由某个概率支配;其中愚蠢动作的概率随演员喝过多少酒及在一次射击中他多少次没有射中而增加。研究上面的剧本应能推断出本程序进一步的性质。

当然,甚至最好的剧本在它们上演之前也要重写,当一个没有经验的作者准备原始剧本时,尤其如此。下面是真实地用来演出的剧本。

故事#1 音乐起。

中景 强盗通过小屋子的窗户盯着看。

特写 强盗的脸。

中景 强盗进入小屋。

特写 强盗看见桌上的威士忌酒瓶。

特写 警长在小屋外边。

中景 强盗看见警长。

远景 警长在强盗肩膀对过的门口处,两人掏枪。

中景 警长掏枪。

远景 射击,强盗中弹。

中景 警长取钱包。

中景 强盗打着趔趄。

中景 强盗垂死挣扎。在试图向警长作最后的射击之后,倒向桌子。

中景 警长带着钱走出门。

中景 强盗的身体现在仍然倒在桌子上。摄影机摇回。(笑声)

故事#2 音乐起。

特写 窗户。强盗出现。

中景 强盗带着两袋钱进入小屋。

中景 强盗把钱放在桶上。

特写 强盗——看见桌上的威士忌。

中景 强盗自己在桌边倒酒喝。数钱。笑。

中景 警长在小屋外边。

中景 掠过窗户。

中景 强盗通过窗户看警长。

远景 警长进入小屋,掏枪,射击。

特写 警长因中弹而痛苦地扭动。

中景/2 镜头 中弹的警长摇晃着到桌旁去喝酒……摔倒死去。

中景 强盗带着钱袋离开小屋。*

[说明:近景=CS="close up",中景=MS="medium shot",等等。感谢电视剧制片人 Thomas

* 1962年由哥伦比亚广播公司出版。版权所有。未经允许不得使用。关于进一步的信息,请看,J. E. Pfeiffer, *The Thinking Machine* (New York: J. B. Lippincott, 1962)。——作者原注

H. Wolf 为作者提供了上述细节。他首先提出了计算机编写剧本的想法,提出这一想法的还有 Douglas T. Ross 和 Harrison R. Morse,他们制作了计算机程序。]

1952 年夏天,Christopher Strachey 用 Ferranti Mark I 计算机的硬件随机数生成器写出了如下信件:

Honey Dear

My sympathetic affection beautifully attracts your affectionate enthusiasm. You are my loving adoration: my breathless adoration. My fellow feeling breathlessly hopes for your dear eagerness. My lovesick adoration cherishes your avid ardour.

亲爱的:

我敏感的爱美好地吸引着你的深情。你是我至爱的敬慕者:我至死不渝的敬慕者。我的那份感情至死不渝地盼望着你的亲爱的热情。我相思病似地敬慕珍爱着你强烈的热诚。

Yours wistfully M. U. C.

[Encounter 3 (1954), 4, 25~31; 另外一个例子出现于 Pears Cyclopedia 第 64 版关于电子计算机的文章中 (London, 1995), 190~191.]

毫无疑问读者会有许多关于怎样来教计算机进行创造性写作的想法;本题的要旨正在于此。

► 6. [40] 考察在你的单位里的每台计算机装置的子程序库,并且用好的随机数生成程序来代替原来的程序。你会有惊人的发现。

► 7. [M40] 一位程序员决定通过使用由带有 $m = 2^{32}$ 的 (1) 生成的周期为 2^{32} 的一个线性同余序列 $\langle X_n \rangle$, 以对他的文件进行加密。他取最高有效位 $\lfloor X_n/2^{16} \rfloor$ 并同他的数据做异或运算,但对于参数 a, c 和 X_0 保密。

通过设计一个方法,它只给出对于 $0 \leq n < 150$ 的 $\lfloor X_n/2^{15} \rfloor$ 的值,就可在一个合理的时间范围内推出乘数 a 和头一个差 $X_1 - X_0$,由此证明这不是一个很安全的方案。

8. [M15] 试提出一个检验线性同余生成程序的实现是否适当地工作的好方法。

9. [HM32] 设 X_0, X_1, \dots 是在 *ran - start* 已经以种子 s 初始化生成过程之后由 *ran - array* 产生的数,并考虑多项式

$$P_n(z) = X_{n+62}z^{99} + X_{n+61}z^{98} + \dots + X_nz^{37} + X_{n+99}z^{36} + \dots + X_{n+64}z + X_{n+63}$$

a) 证明,对于某个指数 $h(s)$, $P_n(z) \equiv z^{h(s)-n} \pmod{2 \text{ 和 } z^{100} + z^{37} + 1}$ 。

b) 借助于 s 的二进表示来表达 $h(s)$ 。

c) 证明,如果 X'_0, X'_1, \dots 是由种子 $s' \neq s$ 通过相同的程序产生的数的序列,我们有:对于 $0 \leq k < 100$, $X_{n+k} \equiv X'_{n+k} \pmod{2}$ 仅当 $|n - n'| \geq 2^{70} - 1$ 时。

10. [22] 把 *ran - array* 和 *ran - start* 的 C 代码转换成精确地生成相同的数的序列的 FORTRAN77 子程序。

► 11. [M25] 假定对于双精度(double)类型的浮点算术运算在 4.2.2 小节的意义下被适当地舍入(因此当对这些值作适当约束时它们是精确的)。试把 C 程序 *ran - array* 和 *ran - start* 转换为提供在 $[0, 1)$ 范围内的双精度随机分数,而不是 30 个二进位整数的类似程序。

► 12. [M21] 对于只对 $[-32768, 32767]$ 范围内的整数进行算术运算的一个小型计算机来说,什么随机数生成程序将是适当的。

13. [M25] 试把习题 3.2.1.1~12 的带借位减法的生成程序同在本小节的程序中实现的延搁斐波那契生成程序做比较。

► 14. [M35] (未来与过去) 令 $X_n = (X_{n-37} + X_{n-100}) \bmod 2$, 考虑序列

$$\langle Y_0, Y_1, \dots \rangle = \langle X_0, X_1, \dots, X_{69}, X_{200}, X_{201}, \dots, X_{299}, X_{400}, X_{401}, \dots, X_{499}, X_{600}, \dots \rangle$$

(这个序列对应于在抛弃一半的元素之后,重复地调用 `ran_array(a,200)` 而且只考察最低有效位。)使用序列 $\langle Y_n \rangle$, 下列实验重复进行一百万次:“生成 100 个随机位;然后如果它们中有 60 个以上为零,就再生成一位并打印它。”这个结果打印 14527 个 0 和 13955 个 1;但是 28482 个随机二进制位包含至多 13955 个 1 的概率仅大约为 .000358。

试给出为什么输出中有这么多 0 的数学解释。

►15.[25] 试编写一个 C 代码,通过如同正文中所建议的那样抛弃除了每 1009 个元素的头 100 个之外的所有其它元素,使得可以方便地生成由 `ran_array` 所得到的随机数。

第4章 算 术

*Seeing there is nothing (right well beloued Students in the Mathematickes)
that is so troublesome to Mathematicall practise, nor that doth more molest
and hinder Calculators, then the Multiplications, Diuisions, square and
cubical Extractions of great numbers, which besides the tedious
expencc of time, are for the most part subiect to many slippery errors.*

*I began therefore to consider in my minde, by what certaine and
ready Art I might remoue those hindrances.*

由于认识到这样一点(学数学的学生们在这点上看法很对),即,
在数学的实践中,没有什么比大数的乘法、除法、开平方和立方更麻烦,
更能折磨和困扰搞计算的人了,

它们除了花费大量的时间外,
还成为引起许多不可琢磨的错误的大部分来源。

我开始在脑海中琢磨,
通过什么知识和现成的技巧,才能消除这些障碍。

——JOHN NEPAIR [NAPIER](1616)

*I do hate sums. There is no greater mistake than to call arithmetic an exact
science. There are. . . hidden laws of Number which it requires a mind
like mine to perceive. For instance, if you add a sum from the bottom up,
and then again from the top down, the result is always different.*

我憎恶求和。再没有比把算术称做精确的科学更大的错误了。

……存在着一些关于数的潜在法则要求像我这样的人去理解。
例如,假定你由下而上地求某个和,而后再由上而下地求它,结果总是不同的。

——M. P. LA TOUCHE (1878)

*I cannot conceive that anybody will require multiplications at the rate
of 40,000, or even 4,000 per hour; such a revolutionary change as the
octonary scale should not be imposed upon mankind in general
for the sake of a few individuals.*

我不能想像,有人需要以每小时 40 000 次
或者即使是 4 000 次的速度来做乘法计算;
对于像八进制换算所引起的这种革命性的变化,

一般不应为了少数人而强加于全人类。

——F. H. WALES (1936)

Most numerical analysts have no interest in arithmetic.

大多数数值分析家都对算术不感兴趣。

——B. PARLETT (1979)

本章的主要目的是对四种基本算术运算——加法、减法、乘法和除法,进行仔细的研究。许多人以为,算术只不过是小孩子学的和计算机做的微不足道的事情。但我们将看到算术是一个迷人的课题,有着许多有趣的方面。对算术也就是数值计算的有效方法进行透彻的研究是很重要的,因为算术是许多计算机应用的基础。

事实上,算术是在世界历史上起着重要作用的一个活生生的课题,而且它仍然在继续急速发展。本章我们将分析对许多类型的量进行算术操作的算法,这些量包括“浮点”数、极其大的数、分数(有理数)、多项式和幂级数等;我们还将讨论诸如进制转换、数的因子分解和多项式求值等这样一些有关的课题。

4.1 定位计数系统

我们进行算术运算的方法,与表示所处理数的方式密切相关。所以,从讨论表示数的主要方式来着手进行算术的研究,是适宜的。

用基数 b (或者进制 b) 的定位记数法由规则

$$(\cdots a_3 a_2 a_1 a_0 . a_{-1} a_{-2} \cdots)_b = \cdots + a_3 b^3 + a_2 b^2 + a_1 b^1 + a_0 + a_{-1} b^{-1} + a_{-2} b^{-2} + \cdots \quad (1)$$

来定义,例如, $(520.3)_6 = 5 \cdot 6^2 + 2 \cdot 6^1 + 0 + 3 \cdot 6^{-1} = 192 \frac{1}{2}$ 。当然,通常的十进计数法,是 b 等于 10,且诸 a 是从“十进数”0,1,2,3,4,5,6,7,8,9 中选取的特殊情况;在这种情况下,(1)中下标 b 可以省略。

当取 b 为一大于 1 的整数,而且要求诸 a 是在 $0 \leq a_k < b$ 的范围内的整数时,我们就得到了十进计数系统的最简单推广。这就为我们提供了标准的二进($b=2$),三进($b=3$),四进($b=4$),五进($b=5$), \cdots 数系统。一般说来,我们可以取 b 为任意非零数,而且可以从任何特定的一组数中选择 a ;如同我们将看到的,这导致某些有趣的情况。

(1)中 a_0 和 a_{-1} 之间出现的点,称为进制数点(当 $b=10$ 时,它亦叫做小数点,而当 $b=2$ 时,它有时叫做二进小数点,等等)。欧洲大陆的人经常使用逗号来代替句点以表示小数点。英国人以前使用抬高的点。

(1)中诸 a 称为表示的数字。通常对于大的 k 的数字 a_k 要比对于小的 k 的数字 a_k “更重要”,因此最左边的或“前导的”数字称为最高有效位数字,而最右边的或“尾部”数字称为最低有效位数字。在标准的二进系统中,二进数字通常称做位。在

标准的十六进系统(进制为16)中,通常以

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$$

表示从0到15的十六个数字。

数表示的历史发展是一段引人入胜的故事,其发展与文明本身的发展并驾齐驱。如果我们要详细地考察这段历史,那就会扯远了,但考察一下它的主要特征却是有教益的。

数表示的最早形式,仍然可以在原始的文化中找到,它一般都是以手指头或一堆石头等等为基础的,通常伴以特殊约定,就是以—个特殊类型的对象或者—个特殊位置来代替更大的一堆或—组对象,比如说五个或十个。这样的系统就自然地导致了以书写形式来表示数的早期形式,诸如巴比伦的、埃及的、希腊的、中国的和罗马的数字系统。但是,除了在最简单的情况下外,这样的记号对于实施算术运算来说,是十分不便的。

20世纪以来,数学史家们对于考古学家在中东发现的早期的楔形文字表进行了广泛的研究。这些研究表明巴比伦人实际上有两种不同的数表示系统:日常商务交易中使用的数,是按十、百等等分组为基础的记号写出的,这个记号继承了更早的美索不达米亚文化,它很少用到大的数。然而,当考虑更困难的数学问题时,巴比伦数学家就广泛地利用六十进(进制为60)的定位计数法,至少早在公元前1750年,它就已高度地发展了。这个计数法是独特的,因为它实质上是省略了指数的浮点表示形式;六十的适当的比例因子或乘幂由语境提供,使得例如数2, 120, 7200, $\frac{1}{30}$ 等等都以相同的方式写成。利用辅助的表,这个记号对于乘法和除法是特别方便的,因为小数点的调整不影响答案;作为这个巴比伦记号的例子,考虑早期表的下列摘录:30的平方是15(它也可以读成“ $\frac{1}{2}$ 的平方是 $\frac{1}{4}$ ”); $81 = (1\ 21)_{60}$ 的倒数是 $(44\ 26\ 40)_{60}$;而后者的平方是 $(32\ 55\ 18\ 31\ 6\ 40)_{60}$ 。巴比伦人对于0有一个符号,但由于他们的“浮点”原理,它仅用于数的中间,而不在右端用来表示比例因子。关于早期巴比伦数学的有趣历史,请看 O. Neugebauer, *The Exact Sciences in Antiquity* (Princeton, N. J.: Princeton University Press, 1952), 以及 B. L. van der Waerden, *Science Awakening*, 由 A. Dresden 译成英文 (Groningen: P. Noordhoff, 1954); 也见 D. E. Knuth, *CACM* **15** (1972), 671~677; **19** (1976), 108。

定点的定位计数法看来首先是由玛雅印第安人于约2000年前在中美洲建立起来的;它们的二十进系统是高度发展了的,特别是在联系到天文记录和日历方面。他们在大约公元200年就开始用书写符号表示零。但是西班牙征服者毁掉了玛雅关于历史和科学的几乎所有书籍,所以关于美洲土人已经如何精通算术,我们知之甚少;已经发现了一些专用的乘法表,但未见有除法的例子。[见 J. Eric S. Thompson, *Contributions to Amer. Anthropology and History* **7** (Carnegie Inst. of Washington, 1941), 37~67; J. Justeson, “Ancient Mesoamerican computing practices”, *History of*

Science 3 (Rome: Istituto della Enciclopedia Italiana), 待出。]

在公元前若干世纪,希腊人应用早期形式的算盘来进行他们的算术运算。这种算盘是一块板,上面放着沙子和(或)卵石,它们的行和列以自然的方式对应于我们的十进制系统。也许使我们感到惊奇的是,这种定位计数法从未应用于数的书写形式,因为我们已经非常习惯于使用笔和纸来进行十进计算了;但用算盘计算的极大方便(因为当时手写还不是一项普遍的技巧,而且因为算盘计算可以不需要记忆加法和乘法表)大概使希腊人觉得,那种认为在“草稿纸”上才能更好地完成计算的主张是愚蠢的。同一时期,希腊天文学家确曾使用了对分数的六十进的定位计数法,这是他们从巴比伦人那儿学来的。

我们的十进记号与古老的形式不同,主要是因为它有固定的小数点,它还有表示零的符号以标记一个空位置。零符号首先是在印度的印度教文化中建立起来的。这一记号最初出现的确切日期不能十分确定,一个比较可靠的猜测是大约在公元600年。在这个时期印度的科学相当发达,特别是在天文学方面。现在已知的使用十进记号的最早的印度手稿上,数字是逆着写的(即最高位数字在右边),但不久最高位数字在左边就成了标准。

大约在公元750年,随着许多重要的著作翻译成阿拉伯文,印度的十进算术的原理传到了波斯,在一份 Abraham Ibn Ezra 所写的希伯来文件中,生动地叙述了这一发展,这一文件已译成英文,见 *AMM* 25 (1918), 99 ~ 108。这以后不久, al-Khwārizmī 写出了关于这个课题的阿拉伯文的课本(如同第1章所说明的,我们的“algorithm”(算法)一词来自于 al-Khwārizmī 的名字)。他的书被译成拉丁文,而且对 Leonardo Pisano(斐波那契)有巨大的影响。斐波那契关于算术的著作(公元1202年),在把印度-阿拉伯数字传播到欧洲这一点上起了重大作用。有趣的是,自左到右地写数的顺序,在从印度到阿拉伯和从阿拉伯到拉丁这两次传播期间,都没有改变,尽管阿拉伯文是从右往左写的,而印度文和拉丁文是从左往右写的。David Eugene Smith 在他的 *History of Mathematics* 1 (Boston: Ginn and Co., 1923) 第6章和第8章中,记述了随后在1200年到1600年期间十进记数法和算术传遍于欧洲的详细历史。

十进制记法最初仅仅用于整数,而不是小数。阿拉伯天文学家在他们的行星图和其它表中需要使用小数,他们继续使用以六十进小数为基础的 Ptolemy(著名的希腊天文学家)的记号。六十进系统,作为原始的巴比伦六十进记法的一个残迹,今天还幸存下来在“度分秒”的三角单位中及时间单位中使用。早期的欧洲数学家在处理非负整数时,也使用六十进小数;例如,斐波那契给出值

$$1^{\circ} 22' 7'' 42''' 33^{iv} 4^v 40^{vi}$$

作为方程 $x^3 + 2x^2 + 10x = 20$ 的根的近似值(正确的答案是 $1^{\circ} 22' 7'' 42''' 33^{iv} 4^v 38^{vi} 30^{vii} 50^{viii} 15^i 43^j \dots$)。

对于十分之一、百分之一等等,以类似的方式使用十进记号,似乎只是一个相当小的变化;但是,破坏传统当然是困难的,而且六十进小数比起十进小数来,有其优点,即诸如 $1/3$ 这样的数,可以用一种简单的方式精确地表达出来。

中国数学家(他们从未使用六十进制)是首先使用十进小数的等价记号的创始人,尽管他们的计数系统(没有0)严格说来原先并不是定位计数系统。中国的重量和度量单位是十进的,所以祖冲之(卒于公元501年)能把 π 近似地表达为

3丈,1尺,4寸,1分,5厘,9毫,2丝,7忽

这里丈, …, 忽都是长度的单位;1忽(一根丝线的直径)等于1/10丝等等。大约在公元1250年后,中国就相当广泛地使用了这样一种类似于十进的小数。

真正的十进小数定位计数系统的萌芽形式见于一位名为 al-Uqlīdisi(“欧几里得的门徒”)的无名数学家在大马士革写的一部10世纪的算术教科书中。他有时对小数点的位置作了标记,比如说在同复利有关的问题中,对于 $1 \leq n \leq 5$ 计算 135 乘 $(1.1)^n$ 。[见 A.S.Saidan, *The Arithmetic of al-Uqlīdisi* (Dordrecht: D. Reidel, 1975), 110, 114, 343, 355, 481~485。]但他没有充分地发展这一思想,而且他的技巧不久就被忘却了。巴格达和巴库的 Al-Samaw' al 在1172年写到,他懂得 $\sqrt{10} = 3.162277\dots$ 但他没有方便的方法把这个近似值写下来。过了好几个世纪之后一位波斯数学家 al-Kāshī 才重新发明了十进小数, al-Kāshī 死于1429年。al-Kāshī 是一位有高度技巧的计算家,他给出了 2π 精确到16位的值如下:

整 数		小 数															
0	6	2	8	3	1	8	5	3	0	7	1	7	9	5	8	6	5

这是直到1596—1610年间, Ludolph van Ceulen 费尽气力计算到35位之前,所知的 π 的最好的近似值。

十进小数开始零星地出现于欧洲,例如,一种所谓的“土耳其方法”曾被用于计算 $153.5 \times 16.25 = 2494.375$; 1450年前, Giovanni Bianchini 将它用于测量,进一步发展了这一方法,但和 al-Uqlīdisi 一样,他的工作似乎影响不大。Christof Rudolff 和 Francois Viète 于1525和1579年再次提出这一思想。最后,比利时人 Simon Stevin 于1585年独立地提出了关于十进小数的思想,他写的算术教科书流传很广。他的工作和不久以后对数的发现,使得十进小数于17世纪在欧洲已非常普及。[关于进一步的评述和参考文献,见 D.E. Smith, *History of Mathematics* 2 (1925), 228~247; V.J. Katz, *A History of Mathematics* (1993), 225~228; 及 G. Rosińska, *Quart. J. Hist. Sci. Tech.* 40 (1995), 17~32。]

二进制计数系统有它自己有趣的历史,据了解,今天还存在的许多原始部落,都使用二进的或“对偶”计数系统(即以两个为一组而不是以五个或十个为一组)计数,但他们并不是以真正的二进系统来计数,因为他们并不以一种特殊的方式来处理2的幂。关于原始计数系统的有趣细节,请见 Abraham Seidenberg, *The Diffusion of Counting Practices*, Univ. of Calif. Publ. in Math. 3 (1960), 215~300。另一个实质上是二进系统的“原始”例子,是通常表达节奏和时间持续的音乐记号。

欧洲在17世纪讨论了非十进的数字系统。在很长时间内,天文学家有时用六十进数来表示整数和小数,这主要是当进行乘法运算时[见 John Wallis, *Treatise of*

Algebra (Oxford:1685),18~22,30]。关于任何大于1的整数都可作为进制这一事实首先是在 Blaise Pascal(帕斯卡)的著作 *De Numeris Multiplicibus* 中明确提出的,它大约写于1658年[见 Pascal, *Œuvres Complètes* (Paris:Éditions du Seuil,1963),84~89]。帕斯卡写到:“十进系统已经建立起来,当然有点愚蠢,这是按照人的习惯,而不是像大多数人所想像的根据自然的需要确定的。”他指出,改用十二进系统将是受欢迎的,而且他给出了检验一个十二进的数被9除的可除性的规则。Erhard Weigel 于1673年开始在一些出版物中大力鼓吹四进系统。Joshua Jordaine 对十二进算术进行了详细的讨论,见 *Duodecimal Arithmetick* (London,1687)。

在那个时代,尽管算术已经几乎全部使用十进记号,但其它的重量和度量系统却很少以10的倍数为基础,而且许多商业事务在进行诸如磅、先令及便士这些量的加法中需要大量的技能。因此,若干世纪以来,商人们学会了计算以货币、重量和尺寸的特殊单位表示的数量的和与差;因而他们采用的算术运算都是非十进计数系统的。从13世纪或更早些,在英国通用的液体量度,是特别值得注意的:

2 及耳 = 1 焦品	2 半蒲式耳 = 1 蒲式耳或弗肯
2 焦品 = 1 品脱	2 弗肯 = 1 克德肯
2 品脱 = 1 夸脱	2 克德肯 = 1 巴力尔
2 夸脱 = 1 勃特	2 巴力尔 = 1 霍士赫德
2 勃特 = 1 加仑	2 霍士赫德 = 1 贝北
2 加仑 = 1 配克	2 贝北 = 1 担
2 配克 = 半蒲式耳	

表示成加仑、勃特、夸脱、品脱等等的液体的数量实质上是以二进记号写成的。也许二进算术的真正发明者竟是英国的酒商哩!

已知的头一个纯粹的二进记号,大约出现于1605年的 Thomas Harriot(1560—1621)的某些未发表的手稿中。Harriot 是一个有创造性的人,他由于作为 Walter Raleigh 爵士的代表来到美国而闻名。他发明了(随其它发明一起)类似于现在用做“小于”和“大于”关系的记号;但由于某种原因他决定不把他的许多发明发表出来。关于他的二进算术笔记的摘录已由 John W. Shirley 加以发表 [*Amer. J. Physics* 19 (1951),452~454]; Harriot 关于二进记号的发现是由 Frank Morley 首先提及的 [*The Scientific Monthly* 14 (1922),60~66]。

关于二进系统的头一个公开的讨论见于一位杰出的西斯特主教 Juande Caramuel Lobkowitz 的著作 *Mathesis Biceps* 1 (Campaniae:1670),45~48。Caramuel 相当详细地讨论了在2,3,4,5,6,7,8,9,10,12和60进制下的数的表示,但除了在60进制的情况下之外,没有给出非十进系统中的算术运算的例子。

最后,由 G. W. Leibniz(莱布尼茨)所写的一篇论文 [*Mémoires de l'Académie Royale des Sciences* (Paris:1703),110~116],阐明了二进加法、减法、乘法和除法,真正地把二进制计数法引入殿堂,因而这篇论文通常被看成是二进制算术的起源。莱布尼茨后来十分经常地引用二进系统。他并不推荐用二进制来做实际计算,但他强调了二进制在数论研究中的重要性,因为数列在二进制计数法下的样式通常比它们

在十进制记法下更为明显;他也看到了每一事物都可借助于0和1来表达这一事实的神秘意义。莱布尼茨未发表的手稿表明,早在1679年,他就已经对二进计数系统感兴趣,那时他把它叫做“bimal”(双的)系统(类似于“decimal”(十的)系统)。

Hans J. Zacher 对莱布尼茨早期关于二进系统的工作进行了仔细的研究[Die *Hauptschriften zur Dyadik von G. W. Leibniz* (Frankfurt am Main: Klostermann, 1973)]。Zacher 指出,莱布尼茨熟悉 John Napier 的所谓“局部算术”,即用石头来进行计算的一种方法,这等于是用一种二进制的算盘。[Napier 把他的局部算术的思想作为他的小书 *Rhabdologia* 的一个附录发表于1617年;它可以称做世界上的头一个“二进计算机”,而且它当然是世界上最便宜的,尽管 Napier 感到它的实用性还不如它的趣味性。见 Martin Gardner 在 *Knotted Doughnuts and Other Mathematical Entertainments* (New York: Freeman, 1986)第8章中的讨论。]

有趣的是对于小数点右边的负幂次这一重要概念,那时还未被很好认识。莱布尼茨曾要求 James Bernoulli (伯努利)在二进系统中计算 π ,而伯努利是这样来“解决”这个问题的,即,通过对 π 取35位十进位的近似值,并把它乘以 10^{35} ,然后以二进系统来表达这个整数,这就是他的答案。如果只考虑前几位,这等于说 $\pi \approx 3.14$,而且 $(314)_{10} = (100111010)_2$;因此在二进制下 π 是 100111010! [见 Leibniz, *Math. Schriften*, K. Gehrhardt 编辑,3 (Halle: 1855), 97; 由于计算错误,118位答案中有2位是不正确的。]伯努利的计算的动机显然是要看看在 π 的这种表示中能否观察到什么简单的模式。

瑞典的查尔斯十二世的数学才能也许超过了世界上所有其他国王。在大约1717年,他提出了八进制算术的思想。这大概是他自己的发明,尽管他于1707年曾经同莱布尼茨短暂地见过面。查尔斯觉得8或64进制比起十进制来更便于计算,因此他考虑在瑞典引进八进算术;但他在进行这样一项变革之前死于战争中[见 *The Works of Voltaire* 21 (Paris: E. R. DuMont, 1901), 49; E. Swedenborg, *Gentleman's Magazine* 24 (1754), 423~424]。

1750年前,威廉和玛丽学院(College of William and Mary)的教授 Hugh Jones 牧师也在殖民地的美国提出了八进计数法[见 *Gentleman's Magazine* 15 (1745), 377~379; H. R. Phalen, *AMM* 56 (1949), 461~465]。

大约一个多世纪以后,一位名叫 John W. Nystrom 的杰出的瑞典裔美国土木工程师决定把查尔斯十二世的计划推进一步,而且他以十六进算术为基础设计了计数、重量和度量的一个完备系统。他写道:“我不害怕或者说不踌躇来鼓吹二进制的算术和度量衡系统。我知道大自然在我这边。若我不能成功地使你们对其方便性和于人类的巨大重要性留下深刻印象,那么就说明我太不信任我们这一代,我们的科学人员和我们的哲学家了。”Nystrom 想出了对于十六进数读音的特殊方法;例如 $(C0160)_{16}$ 被读做“vybong, bysanton”。他的整个系统称为托纳尔(Tónal)系统,并在 *J. Franklin Inst.* 46 (1863), 263~275, 337~348, 402~407 中对它作了描述。一个类似的系统,但使用八进制,大约在同时为 Alfred B. Taylor 提出[*Proc. Amer. Phar-*

maceutical Assoc. 8 (1859), 115~216; *Proc. Amer. Philosophical Soc.* 24 (1887), 296~366]。在那个时代,法国度量衡公制系统的推广使用,引起了关于使用十进算术优点的许多辩论。事实上,当时在法国就已有有人建议使用八进制[J. D. Collenne, *Le Système Octaval* (Paris: 1845); Aimé Mariage, *Numération par Huit* (Paris: Le Nonnant, 1857)]。

远自莱布尼茨时代以来,二进系统就已经作为新奇的事物而闻名,而且 R. C. Archibald 汇集了大约 20 篇关于它的早期文献[*AMM* 25 (1918), 139~142]。它主要应用于乘幂的计算,如同在 4.6.3 小节所说明的那样,也应用于分析某些游戏和难题。Giuseppe Peano[*Atti della R. Accademia delle Scienze di Torino* 34 (1898), 47~55]使用二进位记号作为 256 个符号的“逻辑”字符集的基础。Joseph Bowden[*Special Topics in Theoretical Arithmetic* (Garden City: 1936), 49]给出了十六进制数的术语系统。

Anton Glaser 的著作 *History of Binary and Other Nondecimal Numeration* (Los Angeles: Tomash, 1981), 包含有关二进计数法发展的丰富和近乎完整的讨论,包括以上引证的许多著作的英文翻译[见 *Historia Math.* 10 (1983), 236~243]。

计数系统的许多近来的历史都同计算机器的发展相联系。Charles Babbage 在 1838 年的笔记表明,他考虑过把非十进数用于他的分析机 (Analytical Engine) 中 [M. V. Wilkes, *Historia Math.* 4 (1977), 421]。由于对实现算术运算,特别是实现乘法的机械设备的兴趣日益增加,导致了 20 世纪 30 年代很多人考虑把二进系统用于实现这一目的。E. William Phillips 在题为“Binary Calculation”的文章 [*Journal of the Institute of Actuaries* 67 (1936), 187~221] 中,以及在他关于这一问题的讲座之后的讨论记录中,对这样的活动做了可喜的记述。Phillips 的开场白是:“(本文的)最终目的是说服整个文明世界抛弃十进数而用八进数来代替它。”

Phillips 的论文的现代读者也许会惊奇地发现,在那时的所有英语字典上,就如同十进数系统被正确地称做“denary”或“decimal”(“十进位的”)那样,八进数系统已被正确地称做“octonary”或“octonal”(“八进位的”);直到 1961 年以前,在英语字典中一直未出现有“octal”(八进制)一词,而且它显然是作为某类真空管的“基极”的一个术语而开始的。“hexadecimal”(十六进制的)一词出现于我们语言中的时间更晚,它是希腊语和拉丁语词干的混合;更恰当的术语是“senidenary”或“sedecimal”或甚至“sexadecimal”(十六进位的)。但后者对于计算机程序员来说或许有伤风化(sex 含有性的含义——译者注)。

本章开始处所摘录的 Wales 先生的评述,就是从同 Phillips 的论文一起印刷的讨论记录中摘出的。参加同一讲演的另一个人指出了八进制系统对于商业有一个缺点:“5% 变成了 64 分之 3.1463,它听上去未免太可怕了。”

Phillips 从能够进行二进制计数的一个电子线路中得到了对于他的建议的鼓舞 [C. E. Wynn-Williams, *Proc. Roy. Soc. London* A136 (1932), 312~324]。自 20 世纪 30 年代以来,用于一般算术运算的电子机械和电子线路开始发展,主要贡献是由美

国的 John V. Atanasoff 和 George R. Stibitz, 法国的 L. Couffignal 和 R. Valtat 以及德国的 Helmut Schreyer 和 Konrad Zuse 做出的。这些发明家都用到了二进系统, 尽管 Stibitz 后来发明了余 3 二-十进制计数法。所有这些早期发展的动人记述, 包括当时重要资料的再版和译介, 见 Brian Randell, *The Origins of Digital Computer* (Berlin: Springer, 1973)。

20 世纪 40 年代初建造的美国第一台高速计算机, 用的是十进算术。但在 1946 年, A. W. Burks, H. H. Goldstine 及 J. von Neumann 在同第一台存储程序计算机的设计有关的一份重要的备忘录中, 对于决定背离传统进制而使用基数为 2 的计数制做了详细的论证[见 John von Neumann, *Collected Works* 5, 41~65]。自那以后, 二进计算机已经成倍发展, 在有了使用二进机器十多年的经验之后。W. Buchholz 在他的文章“Fingers or Fist?”[CACM 2 (1959 年 12 月), 3~11]中, 讨论了二进制的相对优点和缺点。

本书中所使用的 MIX 计算机, 已经定义成既可以是二进的, 也可以是十进的。说明这样一点是有趣的, 即几乎所有的 MIX 程序全都可以在无须知道正在使用的是二进的还是十进的符号的情况下表达出来——甚至当我们进行多精度算术运算时也如此。因而, 我们发现, 进制的选择对于计算机程序设计并没有重大的影响。(然而, 对于这一陈述, 有一个值得注意的例外, 那就是 7.1 节中所讨论的“布尔”数算法; 也见算法 4.5.2B。)

在一台计算机中, 负数的表示有若干不同的方法, 而且这有时会影响完成算术运算的方式。为了了解这些记号, 让我们首先把 MIX 看成一台十进制的计算机, 于是每个字包含 10 个数字和一个正负号, 例如

$$-12345\ 67890 \quad (2)$$

这叫做带符号的量表示法。这样一种表示同普通的符号约定是一致的, 所以许多程序员都喜欢它。一个潜在的缺点是有负零和正零两种表示, 而通常它们应表示同一个数; 这种可能性要求在实践中稍留点心, 尽管它只是有时有用。

进行十进算术的大多数机械计算器, 都使用另一个叫做 10 的补码记法的系统。如果我们从 00000 00000 减去 1, 在这种记法下我们得到 99999 99999; 换言之, 对于这个数不附加明显的正负号, 而且计算是模 10^{10} 进行的。数 $-12345\ 67890$ 在 10 的补码记法下将呈现为

$$87654\ 32110 \quad (3)$$

在这种记法下, 通常把前导数字为 5, 6, 7, 8 或 9 的任何数看成是一个负的值, 尽管相对于加法和减法, 把(3)看成是数 $+87654\ 32110$ 并无妨碍, 如果这样做是方便的话。对于 10 的补码记法没有“负零”的问题。

在实践中, 带符号的量表示法和 10 的补码记法之间的主要差别, 是右移并不使补码记法的量除以 10; 例如, 数 $-11 = \cdots 99989$, 右移一位, 得出 $\cdots 99998 = -2$ (假定当所移的数为负时, 右移一位就插入“9”作为前导数字)。一般来说, 在 10 的补码记法下, 无论 x 为正还是为负, x 每右移一位都给出 $\lfloor x/10 \rfloor$ 。

10 的补码系统的一个可能的缺点是:它关于零不对称; p 位数字负数 $500\cdots 0$ 不是任何 p 位数字的正数的负值。因此把 x 变成 $-x$ 可能引起溢出。(关于有无限精度的进制补码记号的讨论,见习题 7 和 31。)

高速计算机刚问世后,就已经使用了另一种称为 9 的补码表示的记法。在这种情况下,数 $-12345\ 67890$ 将呈现为

$$87654\ 32109 \quad (4)$$

负数 $-x$ 的每位数字等于 9 减去 x 的对应数字。不难看出,一个负数的 9 的补码记号总比对应的 10 的补码的记号小 1。加法和减法是以模 $10^{10}-1$ 进行的,它意味着离开左端的一个进位被加到右端上(参考 3.2.1.1 小节模 $w-1$ 算术的讨论)。这样,对于负零又产生了潜在的问题,因为 $99999\ 99999$ 和 $00000\ 00000$ 都表示同一个值。

刚才对 10 进制算术所说明的思想可以类似的方式应用于 2 进制算术上,对于它们,我们有带符号的量,2 的补码和 1 的补码记法。对于 n 个二进位数的 2 的补码算术是模 2^n 的算术;1 的补码算术是模 2^n-1 的算术。如同在本章中的例子中所使用的那样,MIX 计算机仅仅处理带符号的量的算术运算;然而,当有必要时,在有关的正文里也会讨论关于补码记法的另外的一些过程。

喜欢细节的读者和版面编辑应当注意在像“2 的补码”和“1 的补码”这样的术语中撇号的位置(指英文表示——译者):2 的补码关于 2 的一个次幂求补,而 1 的补码是关于 1 的一个长的序列求补。其实,也还有 3 进制的 2 的补码和关于 $(2\cdots 22)_3$ 的补码。

机器语言的描述经常告诉我们,计算机的线路在每个数值的字中的一个特定位置处设置进制点。对这种陈述通常可不予理睬。如果我们预先假定小数点处于某个位置,则最好知道关于在一个指令的结果中小数点将在何处出现的规则。例如,在 MIX 的情况下,我们可以认为我们的操作数是小数点在极右的整数,或者是小数点在极左边的分数,或者作为这两个极端的某种混合。关于在加法、减法、乘法或除法之后小数点的出现的规则则是直截了当的。

容易看出,在进制 b 与进制 b^k 之间有一个简单的关系:

$$(\cdots a_3 a_2 a_1 a_0 . a_{-1} a_{-2} \cdots)_b = (\cdots A_3 A_2 A_1 A_0 . A_{-1} A_{-2} \cdots)_{b^k} \quad (5)$$

其中

$$A_j = (a_{kj+k-1} \cdots a_{kj+1} a_{kj})_b$$

见习题 8。于是我们一眼就能看出,比如说,在 2 进制与 16 进制之间简单的转换技术。

除了至今讨论的标准 b 进制系统之外,定位计数法还有许多有趣的变形。例如,我们可以有进制为 (-10) 的数,使得

$$\begin{aligned} (\cdots a_3 a_2 a_1 a_0 . a_{-1} a_{-2} \cdots)_{-10} &= \cdots + a_3(-10)^3 + a_2(-10)^2 + a_1(-10)^1 + a_0 + \cdots = \\ &\cdots - 1000a_3 + 100a_2 - 10a_1 + a_0 - \frac{1}{10}a_{-1} + \frac{1}{100}a_{-2} - \cdots \end{aligned}$$

这里,恰如在十进系统中那样,各个数字都满足 $0 \leq a_k \leq 9$,在“负十进”系统中数 1234567890 将呈现为

$$(1\ 93755\ 73910)_{-10} \quad (6)$$

因为后者表示 $10305070900 - 9070503010$ 。有趣的是,注意这个数的相反数,即 -1234567890 ,将被写成

$$(28466\ 48290)_{-10} \quad (7)$$

而且,事实上,在负十进系统中,每个实数,无论是正还是负,都可以表示出来,而不用带符号。

负进制系统首先是由 Vittorio Grünwald 考虑的 [*Giornale di Matematiche di Battaglini* 23 (1885), 203~221, 367], 他说明在这样的系统中如何执行四种算术运算。Grünwald 还讨论了求根、可除性检验及进制转换。然而,由于他的文章发表在不太出名的杂志上,似乎对别人的研究工作没有影响,而且很快就被忘却了。关于负进系统的下一个著作看来是 A. J. Kempner 的 [*AMM* 43 (1936), 610~617]。他讨论了非整数进制的性质并且在脚注中解释说负进制也是可行的。又过了 20 年,这一思想重新被发现,这次是由 Z. Pawlak 和 A. Wakulicz [*Bulletin de l'Académie Polonaise des Sciences, Classe III*, 5 (1957), 233~236; *Série des sciences techniques* 7 (1959), 713~721] 以及 L. Wacel [*IRE Transactions EC-6* (1957), 123] 发现的。20 世纪 50 年代后期,在波兰研制了叫做 SKRZAT 1 和 BINEG 的实验性计算机,它们使用 -2 作为算术运算的进制;见 N. M. Blachman, *CACM* 4 (1961), 257; R. W. Marczyński, *Ann. Hist. Computing* 2 (1980), 37~48。进一步的文献见 *IEEE Transactions EC-12* (1963), 274~276; *Computer Design* 6 (1967 年 5 月), 52~63。有证据表明好些人都独立地萌生过负进制的思想。例如,在 1955 年的一次高年级高中生的“寻找科学天才”竞赛中, D. E. Knuth 提交的一篇短篇论文就曾讨论过负进制系统,并且进一步推广到复数进制。

$2i$ 进制产生出称做“虚 4”计数系统的一个系统(与 4 进制相对照),它有异常的特性:每一个复数都可以不带符号地用数字 0, 1, 2, 3 来表示。[见 D. E. Knuth, *CACM* 3 (1960), 245~247。]例如,

$$\begin{aligned} (11210.31)_{2i} &= 1 \cdot 16 + 1 \cdot (-8i) + 2 \cdot (-4) + 1 \cdot (2i) + \\ &\quad 3 \cdot \left(-\frac{1}{2}i\right) + 1 \cdot \left(-\frac{1}{4}\right) = 7\frac{3}{4} - 7\frac{1}{2}i \end{aligned}$$

这里数 $(a_{2n} \cdots a_1 a_0 \cdot a_{-1} \cdots a_{-2k})_{2i}$ 等于

$$(a_{2n} \cdots a_2 a_0 \cdot a_{-2} \cdots a_{-2k})_{-4} + 2i(a_{2n-1} \cdots a_3 a_1 \cdot a_{-1} \cdots a_{-2k+1})_{-4}$$

所以,将一个数转换到虚 4 进记法去和从虚 4 进记法转换回来,就归结为转换成实部与虚部的负 4 进表示,和从实部和虚部的负 4 进表示转换回来的问题。这个系统的有趣性质是它允许在一种十分统一的方式下完成复数的乘法和除法,而无须单独地处理实部和虚部。例如,在此系统下可以像我们在任何进制之下所做的那样把两

个数相乘,只是使用不同的“进位”规则:每当一个数字超过3时,我们就将其减4,并且在它的左边第二列“进位”-1;当一个数字为负时,我们就对它加4并对它左边第二列“进位”+1。下列例子说明了如何运用这个特殊的进位规则

$$\begin{array}{r}
 12231 \\
 12231 \\
 \hline
 12231 \\
 10320213 \\
 13022 \\
 13022 \\
 \hline
 12231 \\
 \hline
 021333121
 \end{array}
 \begin{array}{l}
 [9-10i] \\
 [9-10i] \\
 \\
 \\
 \\
 \\
 \\
 [-19-180i]
 \end{array}$$

一个类似的仅仅使用数字0和1的系统可以以 $\sqrt{2}i$ 为进制,但它要求对简单的数“i”本身的一个无穷的非重复的展开。Vittorio Grünwald 提出在奇数位置中使用数字0和 $1/\sqrt{2}$,以避免这个问题,但这实际上糟蹋了整个系统[见*Commentari dell'Ateneo di Brescia* (1886), 43~54]。

如 W. Penney[JACM 12 (1965), 247~248]所建议的,通过使用进制 $i-1$,也可以得到另一个“二进”复数系统:

$$(\cdots a_4 a_3 a_2 a_1 a_0 \cdot a_{-1} \cdots)_{i-1} = \cdots - 4a_4 + (2+2i)a_3 - 2ia_2 + (i-1)a_1 + a_0 - \frac{1}{2}(i+1)a_{-1} + \cdots$$

在这个系统中,仅仅需要数字0和1。证明每个复数都有这样一个表示的一种方式,是考虑图1中所示的有趣的集合 S ;由定义,这个集合由所有可以写成 $\sum_{k \geq 1} a_k (i-1)^{-k}$ 的点组成,其中 a_1, a_2, a_3, \cdots 是0和1的无限序列,它也叫做“双龙不规则碎片”(twindragon fractal)[见 M. F. Barnsley, *Fractals Everywhere*, 第2版 (Academic Press, 1993), 306, 310]。图1说明了 S 可以分解成同余于 $\frac{1}{16}S$ 的256个小块。注意,如果 S 的图形以顺时针方向转动 135° ,则我们就得到两个同余于 $(1/\sqrt{2})S$ 的相邻集合,因为 $(i-1)S = S \cup (S+1)$ 。关于 S 包含所有绝对值充分小的复数的详细证明,见习题18。

也许所有数系当中最漂亮的是平衡三进制。它由用“三元”(三进制数字)-1, 0, +1代替0, 1和2的三进制表示组成。如果使用 $\bar{1}$ 来代表-1,则我们就有下列的平衡三进制数的例子:

平衡的三进制	十进制
$10\bar{1}$	8
$1110.\bar{1}\bar{1}$	$32\frac{5}{9}$

$$\bar{1}110.11 \quad -32\frac{5}{9}$$

$$1110 \quad -33$$

$$0.11111\dots \quad \frac{1}{2}$$

求一个数在平衡的二进制系统中的表示的一个方法是首先把它表示成三进制数。例如，

$$208.3 = (21201.022002200220\dots)_3$$

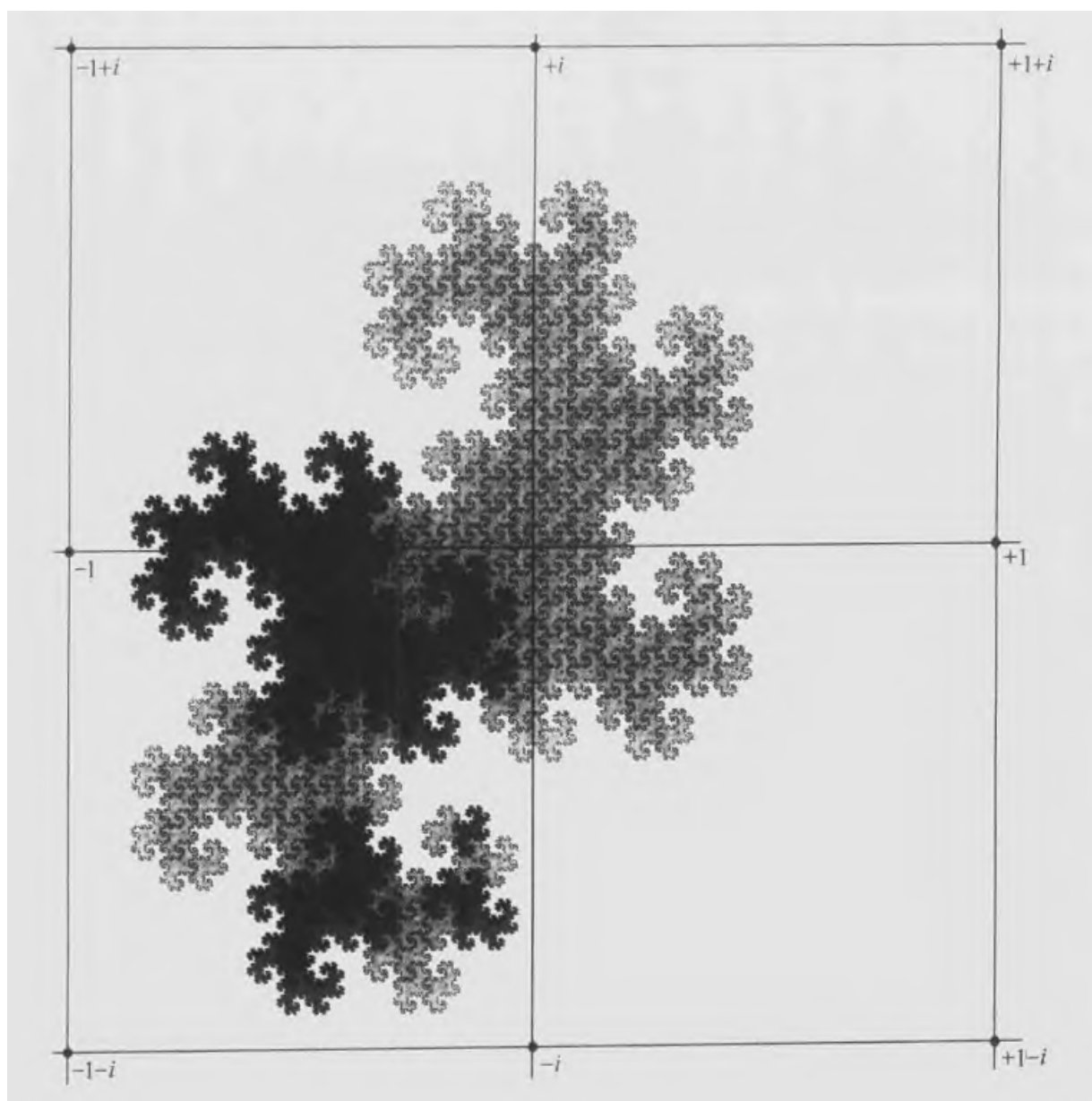


图1 称为“双龙”的不规则碎片集合 S

(用纸和笔转换成三进制数的一个非常简单的方法在习题 4.4-12 中给出。)现在加上二进制表示中的无穷的数 $\dots 11111.11111\dots$ 。在上边的例子中,我们得到无穷的数

$$(\cdots 11111210012.210121012101\cdots)_3$$

最后,用从每个数字减 1 的办法来减去数 $\cdots 11111.11111\cdots$,我们得到

$$208.3 = (10\bar{1}\bar{1}01.10\bar{1}010\bar{1}010\bar{1}0\cdots)_3 \quad (8)$$

显然,如果我们用具有适当多个 1 的数来代替人为的无穷数 $\cdots 11111.11111\cdots$,就可以使这个过程严格化。

平衡三进制数系有许多有趣的性质:

a)通过交换 1 和 $\bar{1}$,就得到一个数的负值。

b)一个数的正负号由它的最高位的非零的“三元”量给出,一般说来我们可以像在十进制系统中那样,通过从左到右地读任何两个数来使用字典序比较它们。

c)舍入成最接近的整数的运算,和截断是一样的;换言之,只须删去小数点右边的数。

利用下面的表,平衡三进制系统的加法是十分简单的:

$$\begin{array}{cccccccccccccccccccccccccccc}
 \bar{1} & \bar{1} & \bar{1} & \bar{1} & \bar{1} & \bar{1} & \bar{1} & \bar{1} & \bar{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 \bar{1} & \bar{1} & \bar{1} & 0 & 0 & 0 & 1 & 1 & 1 & \bar{1} & \bar{1} & \bar{1} & 0 & 0 & 0 & 1 & 1 & 1 & \bar{1} & \bar{1} & \bar{1} & 0 & 0 & 0 & 1 & 1 & 1 \\
 \bar{1} & 0 & 1 & \bar{1} & 0 & 1 & \bar{1} & 0 & 1 & \bar{1} & 0 & 1 & \bar{1} & 0 & 1 & \bar{1} & 0 & 1 & \bar{1} & 0 & 1 & \bar{1} & 0 & 1 & \bar{1} & 0 & 1 \\
 \hline
 10 & 11 & \bar{1}\bar{1}1 & \bar{1} & 0 & \bar{1} & 0 & \bar{1}\bar{1}1 & 1 & 0 & \bar{1} & 0 & 1 & 0 & 11\bar{1} & \bar{1} & 0 & 1 & 0 & 11\bar{1} & \bar{1}\bar{1}1 & 10
 \end{array}$$

(加法的三个输入,是要相加的数的数字和进位数字。)减法就是取负之后再进行加法;而乘法也归结为取负和加法;如下例所示:

$$\begin{array}{r}
 \bar{1}\bar{1}0\bar{1} \quad [17] \\
 \bar{1}\bar{1}0\bar{1} \quad [17] \\
 \hline
 \bar{1}\bar{1}01 \\
 \bar{1}\bar{1}01 \\
 \hline
 1\bar{1}0\bar{1} \\
 \hline
 011\bar{1}\bar{1}01 \quad [289]
 \end{array}$$

数在平衡的三进制系统中的表示隐含于一个有名的普通叫做“Bachet 重量问题”的数学难题中,尽管在 Bachet 写他的书之前 4 个世纪,斐波那契就已经指出了它,而在斐波那契之前 100 多年,波斯的 Tabari 也已提到。[见 W. Ahrens, *Mathematische Unterhaltungen and Spiele* 1 (Leipzig: Teubner, 1910), 3.4 节; H. Hermelink, *Janus* 65 (1978), 105~117.] 带有负数的定位计数系统是 J. Colson 发明的 [*Philos. Trans* 34 (1726), 161~173], 而后它被忘却, 在大约 100 年之后重新由 John Leslie 爵士 [*The Philosophy of Arithmetic* (Edinburgh: 1817), 33~34, 54, 64~65, 117, 150] 及 A. Cauchy (柯西) [*Comptes Rendus Acad. Sci. Paris* 11 (1840), 789~798] 发现。柯西指出, 负数字使得人们没有必要去记忆超过 5×5 的乘法表。有人断言, 这样带有负数字的定位计数系统在印度很久以前就为人所知了 [见 J. Bharati, *Vedic Mathematics* (Delhi: Motilal Banarsidass, 1965)], 但已被 K. S. Shukla 所否定 [*Mathematical Education* 5, 3 (1989), 129~133]。头一个“纯粹”的平衡三进制的真正出现是在

Léon Lalanne 的一篇论文中[Comptes Rendus Acad. Sci. Paris 11 (1840), 903~905], 他是算术运算的机械设备的的设计者。在 Lalanne 的文章发表之后 100 年内, 这个系统很少被提及, 一直到 1945—1946 年间穆尔电子工程学校(Moore School of Electrical Engineering)的第一台电子计算机问世时为止; 那时, 它连同二进系统作为十进系统可能的替代者引起了认真的注意。进行平衡的三进制算术运算的算术线路的复杂性并不比进行二进系统算术运算的算术线路大很多, 而且表示一个给定的数所需的数字个数仅仅相当于用二进表示所需数字个数的 $\ln 2 / \ln 3 \approx 63\%$ 。关于平衡的三进制系统的讨论见于 AMM 57 (1950), 90~93, 以及 High-speed Computing Devices, Engineering Research Associates(McGraw-Hill, 1950), 287~289 中。试验性的俄国的计算机 SETUN 是以平衡的三进制系统为基础的[见 CACM 3 (1960), 149~150]。而且, 或许这个计数系统的对称性和简单的算术运算有朝一日将证明是十分重要的——当“触发器”为一个“三态器”所代替时。

还有另一种重要的方式可把简单的定位计数法推广成一个混合进制系统。给定一个数的序列 $\langle b_n \rangle$ (其中 n 可以为负), 我们定义

$$\left[\begin{array}{c} \cdots, a_3, a_2, a_1, a_0; a_{-1}, a_{-2}, \cdots \\ \cdots, b_3, b_2, b_1, b_0; b_{-1}, b_{-2}, \cdots \end{array} \right] = \cdots + a_3 b_2 b_1 b_0 + a_2 b_1 b_0 + a_1 b_0 + a_0 + a_{-1}/b_{-1} + a_{-2}/b_{-1}b_{-2} + \cdots \quad (9)$$

在最简单的混合进制系统中, 我们仅以整数进行工作; 我们设 b_0, b_1, b_2, \cdots 为大于 1 的整数, 而且仅仅处理没有小数点的数, 其中要求 a_n 处于 $0 \leq a_n < b_n$ 的范围之内。

最重要的混合进制系统之一是阶乘数系, 其中 $b_n = n + 2$ 。利用这一系统, 我们可以以形式

$$c_n n! + c_{n-1} (n-1)! + \cdots + c_2 2! + c_1 \quad (10)$$

来惟一地表示每一个非负整数, 其中对于 $1 \leq k \leq n, 0 \leq c_k \leq k$, 而 $c_n \neq 0$ (见算法 3.3.2P)。

当我们涉及度量单位时, 混合进制系统在日常生活中是常见的。例如, 量“3 个星期, 2 天, 9 小时, 22 分, 57 秒和 492 毫秒”等于

$$\left[\begin{array}{cccccc} 3, & 2, & 9, & 22, & 57, & 492 \\ & & 7, & 24, & 60, & 60; & 1000 \end{array} \right] \text{秒}$$

量“10 磅, 6 先令和 3 又 $\frac{1}{2}$ 便士”, 在英国变成纯粹十进制的货币系统之前, 就成为英

国货币的 $\left[\begin{array}{ccc} 10, & 6, & 3; & 1 \\ & 20, & 12; & 2 \end{array} \right] \text{便士}。$

直接推广通常的加法和减法, 就可能进行混合进制数的加法和减法, 当然须假定两个操作数都使用相同的混合进制系统(见习题 4.3.1-9)。类似地, 简单地推广熟知的手算方法, 即可容易地用小的整常数去乘和除混合进制的数。

第一个详尽地讨论混合进制系统的是 Georg Cantor[Zeitschrift für Math. und Physik 14 (1869), 121~128]。习题 26 和 29 给出了关于混合进制系统进一步的信

息。

涉及无理进制的某些问题已由 W. Parry 做了研究[Acta Math. Acad. Sci. Hung. 11 (1960), 401~416]。

除了这一节所述的这些系统之外,在本套书的其它地方,还要提到表示数的其它途径。这些途径有组合数系统(习题 1.2.6-56),斐波那契数系统(习题 1.2.8-34, 5.4.2-10), Φ 数系统(习题 1.2.8-35),模表示(4.3.2 小节),Gray 码(7.2.1 小节)以及罗马记数(9.1 节)等。

习 题

1. [15] 在 -2 进制的数系中表达数 $-10, -9, \dots, 9, 10$ 。

► 2. [24] 考虑下列四个数系:(a)二进制(带符号的量);(b)负二进制(-2 进制);(c)平衡三进制;(d)进制 $b = \frac{1}{10}$ 。用上述的每一种数系表达下列三个数:(i) -49 ;(ii) $-3\frac{1}{7}$ (表示成重复的循环);(iii) π (表示到一些有效位)。

3. [20] 在虚四数系中表达 $-49 + i$ 。

4. [15] 假设我们有一个 MIX 程序,其中单元 A 包含一个其小数点在第三字节和第四字节之间的数,而单元 B 包含一个其小数点在第二字节和第三字节之间的数(最左边字节编号为 1)。试问在执行了指令

(a) LDA A; MUL B (b) LDA A; SRAX 5; DIV B

之后,在寄存器 A 和 X 中,小数点该在什么地方?

5. [00] 如果把表示看成是正确的,说明为什么一个负整数在 9 的补码记法下的表示总比它在 10 的补码记法下的表示小 1?

6. [16] 在(a)带符号的量的二进法(包括一位符号),(b)2 的补码法,(c)1 的补码法中,所能表示的最大和最小的 p 位整数各是什么?

7. [M20] 正文仅对在一个计算机字中表示的整数定义了 10 的补码记法,有没有办法对所有有“无限的精度”的实数定义 10 的补码记法,并且与正文定义相似?有没有一个类似的方法,对于所有实数,来定义 9 的补码?

8. [M10] 证明等式(5)。

► 9. [15] 利用十六进数字 0, 1, \dots , 9, A, B, C, D, E, F 把下列的八进制数改变成十六进制记号: 12; 5655; 2550276; 76545336; 3726755。

10. [M22] 把等式(5)推广成如同(9)中那样的混合进制记法。

11. [22] 给出一个算法,它利用 -2 数系计算 $(a_n \cdots a_1 a_0)_{-2}$ 和 $(b_n \cdots b_1 b_0)_{-2}$ 之和,得到答案 $(c_{n+2} \cdots c_1 c_0)_{-2}$ 。

12. [23] 给出算法,(a)把二进制带符号的数 $\pm (a_n \cdots a_0)_2$ 转换成它的负二进制形式 $(b_{n+2} \cdots b_0)_{-2}$;(b)把负二进制数 $(b_{n+1} \cdots b_0)_{-2}$ 转换成它的带符号的量的形式 $\pm (a_{n+1} \cdots a_0)_2$ 。

► 13. [M21] 在十进数系中有些数有两个无穷的十进展开,例如 $2.3599999 \cdots = 2.3600000 \cdots$ 。负十进制(-10 进制)系统是否有惟一的展开,或者在这进制之下是否也存在具有两种不同的无限展开的实数?

14. [14] 利用正文中所述的方法,在虚四数系中对 $(11321)_2$ 做自乘。

15. [M24] 对于负 i -进制和对于虚四数系, 什么是类似于图 1 的集合 $S = \{\sum_{k \geq 1} a_k b^{-k} \mid a_k \text{ 为一允许的数字}\}$?

16. [M24] 试设计一个算法, 在 $i-1$ 数系中, 把 1 加到 $(a_n \cdots a_1 a_0)_{i-1}$ 上去。

17. [M30] 数 $i-1$ 已经被提议作为数系的进制, 但类似的直观上更简单的 $i+1$ 却没有, 这可能显得很独特。能否仅仅利用数字 0 和 1, 把每个复数 $a+bi$ (其中 a 和 b 是整数) 表示成进制为 $i+1$ 的定位计数系统?

18. [HM32] 证明图 1 中的双龙是包含原点的一个邻域的封闭集 (因而, 每个复数对于进制 $i-1$ 都有一个“二进”表示)。

► 19. [23] (David W. Matula) 设 D 是 b 个整数的集合, 恰含同余式 $x \equiv j \pmod{b}$ 的一个解, 其中 $0 \leq j < b$ 。证明所有的整数 m (正的, 负的或 0) 都可以以形式 $m = (a_n \cdots a_0)_b$ 表示, 其中所有 a_j 在 D 中, 当且仅当 $l \leq m \leq u$ 范围中的所有整数都可如此表示, 其中 $l = -\max\{|a| \mid a \in D\} / (b-1)$, $u = \min\{|a| \mid a \in D\} / (b-1)$ 。例如对所有的 $b \geq 3$, $D = \{-1, 0, \dots, b-2\}$ 满足条件。[提示: 设计一个算法, 使它能构造一个适当的表示。]

20. [HM28] (David W. Matula) 考虑使用数字 $D = \{-1, 0, 8, 17, 26, 35, 44, 53, 62, 71\}$ 代替 $\{0, 1, \dots, 9\}$ 的十进系统。习题 19 的结果意味着 (和在习题 18 中一样), 所有实数都可用取自 D 的数字实行无穷十进展开。

习题 13 指出, 在通常的十进系统中, 某些数有两种表示。(a) 找出有两种以上 D -十进表示的一个实数。(b) 证明不存在无穷多个 D -十进表示的实数。(c) 证明有不可数多个数有两种或更多的 D -十进表示。

► 21. [M22] (C. E. Shannon) 是否每个实数 (正的, 负的或 0) 都能在一个平衡的十进数系中表示, 即对于某个整数 n 和某个序列 $a_n, a_{n-1}, a_{n-2}, \dots$, 表示成为 $\sum_{k \leq n} a_k 10^k$ 的形式。其中每个 a_k 是 10 个数 $\left\{-4\frac{1}{2}, -3\frac{1}{2}, -2\frac{1}{2}, -1\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, 1\frac{1}{2}, 2\frac{1}{2}, 3\frac{1}{2}, 4\frac{1}{2}\right\}$ 中之一个 (尽管 0 不是允许的数字之一, 但我们隐含地假定 a_{n+1}, a_{n+2}, \dots 都是 0)。求出 0 在这个数系下的所有表示, 并求出 1 的所有表示。

22. [HM25] 设 $\alpha = -\sum_{m \geq 1} 10^{-m^2}$ 。给定 $\epsilon > 0$ 以及任何实数 x , 证明有一个“十进”表示使得 $0 < \left|x - \sum_{k=0}^n a_k 10^k\right| < \epsilon$, 其中每个 a_k 仅允许是三个值 0, 1 或 α 中之一。(在这个表示中没有使用 10 的负次幂!)

23. [HM30] 设 D 是一个 b 个实数的集合, 使得每个正实数都有一个表示 $\sum_{k \leq n} a_k b^k$ 且所有 $a_k \in D$ 。习题 20 表明, 可能有许多实数没有唯一的表示; 但如果 $0 \in D$, 试证明所有这样的数的集合 T 的测度为 0。并证明如果 $0 \notin D$, 这个结论不必成立。

24. [M35] 求满足下列三个条件的无穷多个不同集合, 其中每个集合包括 10 个非负整数: (i) $\gcd(D) = 1$; (ii) $0 \in D$; (iii) 每个正实数都可表示成形式 $\sum_{k \leq n} a_k 10^k$, 其中 $a_k \in D$ 。

25. [M25] (S. A. Cook) 设 b, u 和 v 是正整数, 其中 $b \geq 2$ 且 $0 < v < b^m$ 。试证 u/v 的 b 进表示在小数点右边任何地方都不包含接连 m 个等于 $b-1$ 的数字 (根据约定, 在标准的 b 进表示中不允许无限多个 $b-1$)。

► 26 [HM30] (N. S. Mendelsohn) 设 $\langle \beta_n \rangle$ 是对全体整数 $n, -\infty < n < +\infty$, 有定义的一个实数序列, 它使得

$$\beta_n < \beta_{n+1}; \quad \lim_{n \rightarrow \infty} \beta_n = \infty; \quad \lim_{n \rightarrow -\infty} \beta_n = 0$$

设 $\langle c_n \rangle$ 是对所有整数 $n, -\infty < n < +\infty$, 有定义的正整数的任意序列。如果有一个整数 n 和一个整数的无穷序列 $a_n, a_{n-1}, a_{n-2}, \dots$ 使得 $x = \sum_{k=-n}^{\infty} a_k \beta_k$, 其中 $a_n \neq 0, 0 \leq a_k \leq c_k$ 且对无穷多的 k , 有 $a_k < c_k$, 则我们就说整数 x 有一个“广义的表示”。

试证, 每个实数 x 恰有一个广义的表示的充分必要条件是, 对所有 $n, \beta_{n+1} = \sum_{k \leq n} c_k \beta_k$ 。(因此, 具有整数基的混合进制系统都有这一性质; 而且具有 $\beta_1 = (c_0 + 1)\beta_0, \beta_2 = (c_1 + 1)(c_0 + 1)\beta_0, \dots, \beta_{-1} = \beta_0/(c_{-1} + 1), \dots$ 的混合进制系统是这种类型的最一般的数系。)

27. [M21] 证明每个非 0 整数都有惟一的“倒转的二进表示” $2^{e_0} - 2^{e_1} + \dots + (-1)^{e_t} 2^{e_t}$, 其中 $e_0 < e_1 < \dots < e_t$ 。

28. [M24] 证明形如 $a + bi$ 的每个非 0 复数, 其中 a 和 b 是整数, 都有惟一的“旋转的二进表示”

$$(1+i)^{e_0} + i(1+i)^{e_1} - (1+i)^{e_2} - i(1+i)^{e_3} + \dots + i^t(1+i)^{e_t}$$

其中 $e_0 < e_1 < \dots < e_t$ (参照习题 27)。

29. [M35] (N. G. de Bruijn) 设 S_0, S_1, S_2, \dots 是非负整数的集合, 如果每个非负整数 n 都恰有一种方式写成形式

$$n = s_0 + s_1 + s_2 + \dots, \quad s_j \in S_j$$

则说汇集 $\{S_0, S_1, S_2, \dots\}$ 有性质 B (性质 B 意味着: 对于所有 $j, 0 \in S_j$, 因为 $n=0$ 仅可表示成 $0+0+0+\dots$)。具有进制 b_0, b_1, b_2, \dots 的任何混合进制数系都有满足性质 B 的汇集的例子, 为此我们只需设 $S_j = \{0, B_j, \dots, (b_j-1)B_j\}$ 即可, 其中 $B_j = b_0 b_1 \dots b_{j-1}$; 这里, 表示 $n = s_0 + s_1 + s_2 + \dots$ 以一种明显的方式对应于它的混合进制表示 (9)。此外, 如果汇集 $\{S_0, S_1, S_2, \dots\}$ 有性质 B, 而且如果 A_0, A_1, A_2, \dots 是非负整数的任何分划 (使得 $A_0 \cup A_1 \cup A_2 \cup \dots = \{0, 1, 2, \dots\}$, 且对于 $i \neq j$, $A_i \cap A_j = \emptyset$; 某些 A_j 可以是空的), 则“塌陷的”集组 $\{T_0, T_1, T_2, \dots\}$ 也有性质 B, 其中 T_j 是取遍所有可能的选择 $s_i \in S_i$ 的所有和 $\sum_{i \in A_j} s_i$ 的集合。

试证, 任何满足性质 B 的汇集 $\{T_0, T_1, T_2, \dots\}$ 均可通过使对应于一个混合进制数系的某个汇集 $\{S_0, S_1, S_2, \dots\}$ 塌陷而得到。

30. [M39] (N. G. de Bruijn) -2 进制数系表明, 每个整数 (正的, 负的或 0) 都有形如

$$(-2)^{e_1} + (-2)^{e_2} + \dots + (-2)^{e_t}, \quad e_1 > e_2 > \dots > e_t \geq 0, \quad t \geq 0$$

的惟一表示。本题的目的是探索这一现象的推广。

a) 设 b_0, b_1, b_2, \dots 是一整数序列, 它使每个整数 n 都有形如

$$n = b_{e_1} + b_{e_2} + \dots + b_{e_t}, \quad e_1 > e_2 > \dots > e_t \geq 0, t \geq 0$$

的惟一表示 (这样的序列 $\langle b_n \rangle$ 叫做“二进基底”)。证明有一个下标 j , 使得 b_j 是奇数, 但对于所有 $k \neq j, b_k$ 是偶数。

b) 证明一个二进基底 $\langle b_n \rangle$ 总能重新排列成形式 $d_0, 2d_1, 4d_2, \dots = \langle 2^n d_n \rangle$, 其中每个 d_k 都是奇数。

c) 如果 b) 中的每个 d_0, d_1, d_2, \dots 都是 ± 1 , 试证: 当且仅当有无限多个 $+1$ 和无限多个 -1 时, $\langle b_n \rangle$ 是二进基底。

d) 证明 $7, -13 \cdot 2, 7 \cdot 2^2, -13 \cdot 2^3, \dots, 7 \cdot 2^{2k}, -13 \cdot 2^{2k+1}, \dots$ 是二进基底, 并求 $n=1$ 的表示法。

► 31. [M35] 2 的补的算术的一种推广, 称做“2-adic 数”, 是由 K. Hensel [Crelle 127 (1904), 51~84] 发现的 (Hensel 事实上处理的是 p -adic 数, 其中 p 为任何素数)。一个 2-adic 的数可以看成二进制数

$$u = (\cdots u_3 u_2 u_1 u_0 \cdot u_{-1} u_{-2} \cdots u_{-n})_2$$

且其表示无限地向左展开,但在小数点右边仅有有限多位。2-adic 数的加法、减法及乘法按照通常的算术过程进行,原则上它可以无限地向左边扩展。例如,

$$\begin{aligned} 7 &= (\cdots 000000000000111)_2 & \frac{1}{7} &= (\cdots 110110110110111)_2 \\ 7 &= (\cdots 111111111111001)_2 & -\frac{1}{7} &= (\cdots 001001001001001)_2 \\ \frac{7}{4} &= (\cdots 00000000000001.11)_2 & \frac{1}{10} &= (\cdots 110011001100110.1)_2 \\ \sqrt{-7} &= (\cdots 100000010110101)_2 \quad \text{或} \quad (\cdots 011111110100101)_2 \end{aligned}$$

这里,7 是通常的二进制整数 7,而 -7 是它的 2 的补(无限地向左扩展);容易证明,当这过程无限地继续下去时,通常的二进制数加法过程将给出 $-7+7=(\cdots 00000)_2=0$ 。 $\frac{1}{7}$ 和 $-\frac{1}{7}$ 的值是惟一这样的 2-adic 数,当形式地乘以 7 时,它们分别给出 +1 和 -1 的值。 $\frac{7}{4}$ 和 $\frac{1}{10}$ 的值是非 2-adic “整数”的 2-adic 的数的例子,因为它们在小数点右边有非零的数字。 $\sqrt{-7}$ 的两个值彼此互负,它们是惟一的这样的 2-adic 的数,即当形式地平方时,它得到值 $(\cdots 111111111111001)_2$ 。

a) 证明任何 2-adic 的数 u 均可被任何非 0 的 2-adic 数 v 整除,并得到一个满足 $u = \nu w$ 的惟一的 2-adic 数 w (因此,2-adic 数集合构成一个“域”;参考 4.6.1 小节)。

b) 证明当 n 是一个正整数时,有理数 $-1/(2n+1)$ 的 2-adic 数表示可以以如下方式得到:首先求 $+1/(2n+1)$ 的通常的二进制展开,它对于 0 和 1 的某个串 α 有周期形式 $(0.\alpha\alpha\alpha\cdots)_2$,于是 $-1/(2n+1)$ 就是 2-adic 数 $(\cdots \alpha\alpha\alpha)_2$ 。

c) 证明 2-adic 数 u 的表示最终必是周期的(即,对所有大的 N 和某个 $\lambda \geq 1$, $u_{N+\lambda} = u_N$),其充要条件是, u 是有理的(即对某个整数 m 和 n , $u = m/n$)。

d) 试证当 n 为一个整数时, \sqrt{n} 是一个 2-adic 数的充分必要条件是,对于某个非负整数 k ,它满足 $n \bmod 2^{2k+3} = 2^{2k}$ (于是,可能性是 $n \bmod 8 = 1$ 或 $n \bmod 32 = 4$,等等)。

32. [M40] (I. Z. Ruzsa) 试构造无穷多个整数,其三进表示仅用 0 和 1,而且其四进表示仅用 0, 1 和 2。

33. [M40] (D. A. Klarner) 设 D 是任何整数的集合,且设 b 是任何正整数,又设 k_n 是对于进制 b 及在 D 中的数字 a_i 可以写作 n 位整数 $(a_{n-1} \cdots a_1 a_0)_b$ 的不同整数的个数。证明序列 $\langle k_n \rangle$ 满足一个线性递推关系,并说明如何计算生成函数 $\sum_n k_n z^n$ 。通过说明 k_n 是在 $b=3$ 和 $D=\{-1, 0, 3\}$ 的情况下的斐波那契数,说明你的算法。

► 34. [22] (G. W. Reitwiesner, 1960) 使用最少的非零数字,说明怎样以 $(\cdots a_2 a_1 a_0)_2$ 的形式表示一个特定的整数 n ,其中每个 a_i 是 $-1, 0$ 或 1 。

4.2 浮点算术

在这一节中,我们将通过分析奠定浮点数计算基础的内部机制,来研究对这种数进行算术运算的基本原理。也许许多读者对这样的课题兴趣不大,因为他们的计算机内已经有浮点的指令了,或者他们的操作系统包括了适当的子程序。但是,事实上,这一节的技术不只关系到计算机设计工程师或者为新的机器编写库程序的那

部分人,而且每一个熟练的程序员都应当有浮点算术的基本步骤在执行期间是如何进行的知识。这一课题并不像大多数人所想像的那样全然是微不足道的,它包含多得惊人的有趣信息。

4.2.1 单精度计算

A. 浮点记法 在 4.1 节我们已经讨论了数的“定点”记法;在这种情况下,程序员知道在正在操作的数中,假定小数点是在什么位置上。但是从许多目的来看,当程序运行时,如果让小数点灵活变动或者“浮动”,并让每个数带一个相应的小数点的位置标志,有着很大的方便性。这一想法,在科学计算中已经用了许多年,特别是适用于表达像阿伏加德罗常数 $N = 6.02214 \times 10^{23}$ 这样非常大的数,或者像普朗克常数 $h = 6.6261 \times 10^{-27}$ 尔格·秒这样非常小的数。

在这一节中,我们将以进制 b , 余量 q , 有 p 位数字的浮点数来进行工作。这样的数被表示成一对值 (e, f) , 它表示

$$(e, f) = f \times b^{e-q} \quad (1)$$

这里 e 是有一确定范围的整数,而 f 是带符号小数。我们约定

$$|f| < 1$$

换言之,小数点出现在 f 的定位表示的左边,更确切地说,关于 p 位数的约定总意味着 $b^p f$ 是一个整数,而且

$$-b^p < b^p f < b^p \quad (2)$$

“浮点二进数”的术语意味着 $b = 2$, “浮点十进数”意味着 $b = 10$, 等等。例如,利用 8 位数字的余 50 的十进数系,我们可以把阿伏加德罗常数 N 和普朗克常数 h 分别写为

$$\begin{aligned} N &= (74, +.60221400) \\ h &= (24, +.66261000) \end{aligned} \quad (3)$$

一个浮点数的两个分量 e 和 f 分别称为指数和小数部分。(有时也使用其它名称,主要有“特征阶”(characteristic)和“尾数”(mantissa);但把小数部分说成尾数乃是术语的滥用,因为尾数的概念在联系到对数时有十分不同的含义,而且英语中尾数一词指的是“一个微不足道的附加物”。)*

MIX 计算机假定它的浮点数形式是

±	e	f	f	f	f
---	---	---	---	---	---

(4)

* 在本书第 4 版作者将改变定义使得 $|f| < b$, 这样式(2)将变为“ $-b^p < b^{p-1}f < b^p$ ”, 而式(5)将变为“ $1 \leq |f| < b$ ”, 阿伏加德罗常数的余 50 十进表示将是 $(73, +6.0221400)$, 等等。其纯粹效果将是余量概念的改变, 在第 3 版中的余 q 在第 4 版中将称为余 $q+1$ 。这种术语的改变将与 IEEE 标准浮点协定更一致(而且也与 4.2.2 小节定义的术语“ulp”相匹配), 尽管(21)~(24)的定义将稍稍凌乱些。——本书责任编辑自“Errata to Volume 2”

这里我们用精度为四个字节的进制 b 余 q 的浮点记法, 其中 b 是字节大小(例如 $b = 64$ 或 $b = 100$), 而 $q = \lfloor \frac{1}{2} b \rfloor$ 。小数部分是 $\pm ffff$, 而 e 是指数, 它处于范围 $0 \leq e < b$ 之中。在大多数现有的计算机中, 这个内部表示是典型的约定, 但这里 b 比通常的进制大得多。

B. 规格化计算 如果 f 的表示的最高位非 0, 从而

$$1/b \leq |f| < 1 \quad (5)$$

或者如果 $f=0$ 且 e 有它的最小的值, 则说一个浮点数 (e, f) 是规格化的。通过首先比较两个规格化浮点数的指数部分, 而后仅当指数相等时才检查小数部分, 能够指出这两个数中哪一个具有更大的绝对值。

现在使用中的大多数浮点程序几乎全都处理规格化的数: 对程序的输入假定为规格化的, 而且输出总是规格化的。在这些约定之下, 我们就失去了表示一些非常小的量的能力——例如值 $(0, .00000001)$ 如不产生一个负指数就不能规格化——但我们获得了速度、一致性, 以及在计算中对相对误差给出相对简单的界的能力(非规格化浮点算术在 4.2.2 小节讨论)。

现在来详细研究浮点算术的操作。同时我们可以考虑构造这些操作的子程序, 这里假定我们的计算机没有内部的浮点硬件。

浮点算术的机器语言子程序通常都编写得非常依赖于机器, 它们使用该计算机的许多最原始的特性; 因此, 从表面上看, 两台不同的机器的浮点加法子程序往往很不一样。但是, 仔细研究二进制和十进制计算机两者的许多子程序就会发现, 这些程序实际上有许多共同点, 而且有可能在一种不依赖机器的方式下来讨论这些课题。

在这一节中我们将要讨论的第一个(而且是最最困难的一个)算法, 是进行浮点加法的一个过程:

$$(e_u, f_u) \oplus (e_v, f_v) = (e_w, f_w) \quad (6)$$

由于浮点算术本身固有地是近似的, 不精确的, 我们将使用“圆”符号

$$\oplus, \ominus, \otimes, \oslash$$

来分别表示浮点的加、减、乘、除, 以便把近似运算同真正运算相区别。

关于浮点加法的基本思想是十分简单的: 假定 $e_u \geq e_v$, 我们取 $e_w = e_u$, $f_w = f_u + f_v/b^{e_u - e_v}$ (从而调整小数点位置, 以进行有意义的加法), 并把结果规格化。可能出现若干情况, 而使这个过程显得颇不平凡, 下面的算法更精确地说明了这个方法。

算法 A(浮点加法) 给定进制 b , 余 q , p 位规格化的浮点数 $u = (e_u, f_u)$ 和 $v = (e_v, f_v)$, 本算法形成和 $w = u \oplus v$ 。对于浮点减法可使用同样的算法, 只须以 $-v$ 替换 v 即可。

A1. [拆开] 分开 u 和 v 表示的指数和小数部分。

A2. [假定 $e_u \geq e_v$] 如果 $e_u < e_v$, 则交换 u 和 v (在许多情况下, 最好是把步骤

A2 和步骤 A1 或者与后边的某些步骤合并在一起)。

A3. [置 e_w] 置 $e_w \leftarrow e_u$ 。

A4. [测试 $e_u - e_v$] 如果 $e_u - e_v \geq p + 2$ (指数的差很大) 则置 $f_u \leftarrow f_u$, 并转到步骤 A7。 (实际上, 由于我们假定 u 是规格化的, 所以我们本来可以就此结束这个算法, 但是要是能通过加 0 来对一个可能是非规格化的数进行规格化, 有时是有用的。)

A5. [右调] 把 f_v 右移 $e_u - e_v$ 个位置, 即用 $b^{e_u - e_v}$ 来除它。 [注: 这样至多进行 $p + 1$ 位的移位, 而且因此下一步 (把 f_u 加到 f_v) 就要求有一个累加器, 能用于保存小数点右边的 $2p + 1$ 位 b 进数字。如果没有这么大的累加器可用, 只要预先采取适当的措施, 也可以把要求缩短到只保存 $p + 2$ 位或 $p + 3$ 位。其细节在习题 5 中给出。]

A6. [相加] 置 $f_w \leftarrow f_u + f_v$ 。

A7. [规格化] (这时 (e_w, f_w) 表示 u 和 v 的和, 但 $|f_w|$ 可能有多于 p 位的数字, 而且它可能大于 1 或小于 $1/b$ 。) 实施以下的算法 N, 规格化和舍入 (e_w, f_w) , 成为最后的答案。 ■

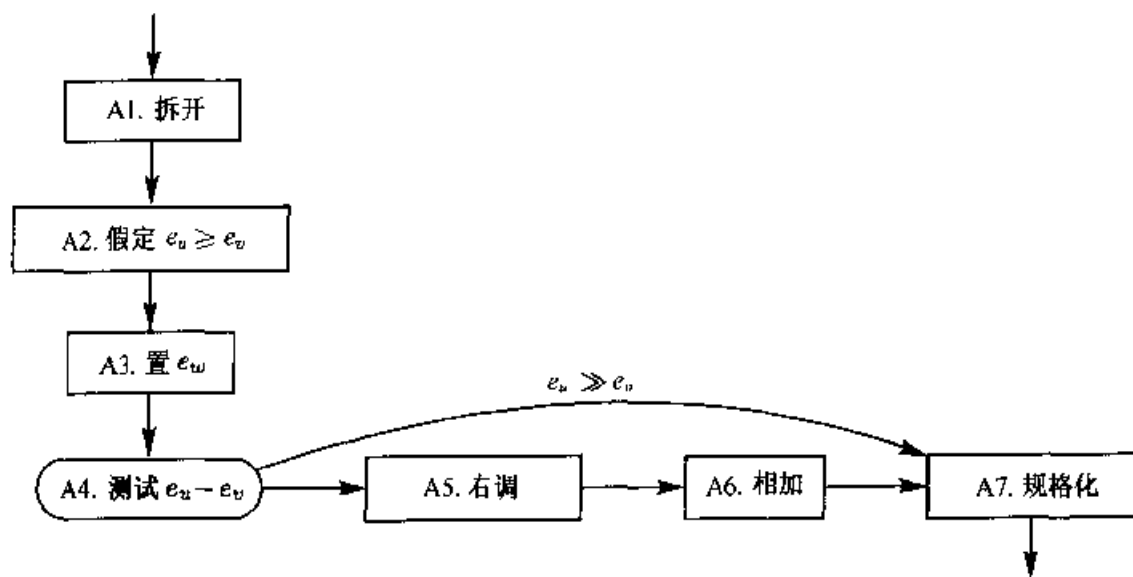


图 2 浮点加法

算法 N (规格化) “未加工的”指数 e 和“未加工的”小数 f 被转换成规格化的形式, 必要时把它舍入成 p 位数字。本算法假定 $|f| < b$ 。

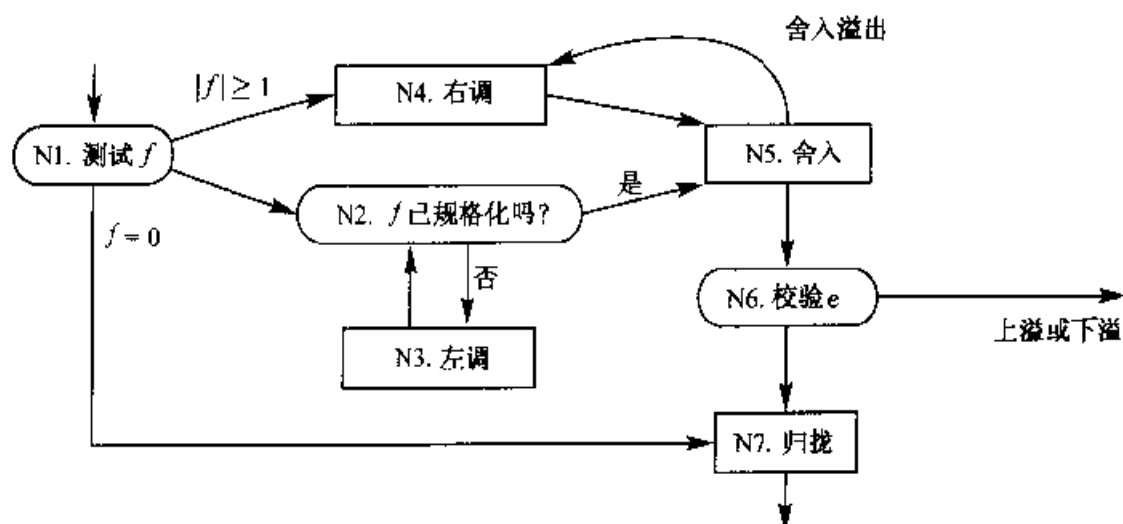
N1. [测试 f] 如果 $|f| \geq 1$ (“小数上溢”), 则转到步骤 N4。如果 $f = 0$, 则置 e 成为其最小可能值, 并转到步骤 N7。

N2. [f 已规格化?] 如果 $|f| \geq 1/b$, 则转到步骤 N5。

N3. [左调] 把 f 左移一位 (即乘以 b), 并把 e 减 1, 返回步骤 N2。

N4. [右调] 把 f 右移一位 (即以 b 来除它), 并把 e 加 1。

- N5. [舍入] 把 f 舍入成 p 位。(我们这样做意味着把 f 变成 b^{-p} 的最接近的倍数。有可能 $(b^p f) \bmod 1 = \frac{1}{2}$, 因而有两个最接近的倍数; 如果 b 是偶数, 选择把 f 变成 b^{-p} 最接近的倍数 f' 使得 $b^p f' + \frac{1}{2}b$ 为奇数。关于舍入的讨论, 见 4.2.2 小节。)重要的是, 这里的舍入运算可以使 $|f| = 1$ (“舍入上溢”); 在这种情况下, 返回步骤 N4。
- N6. [校验 e] 如果 e 太大, 即大于它所允许的范围, 则建立指数上溢标记。如果 e 太小, 则建立指数下溢标记。(见以下的讨论; 由于结果不能表示成所要求范围内的规格化浮点数, 因此需要特殊的动作。)
- N7. [归拢] 把 e 和 f 合在一起, 构成符合输出要求的表示。■

图3 (e, f) 的规格化

习题 4 中给出了浮点加法的某些简单的例子。

下面的 MIX 子程序是做形如(4)的数的加法和减法的, 它说明算法 A 和 N 可以怎样表达成计算机程序。以下的子程序被设计成: 在进入子程序时从符号单元 ACC 取一个输入 u , 又从寄存器 A 取另一个输入 v 。输出 w 既出现于寄存器 A 中, 也出现于单元 ACC 中。于是, 定点代码序列

LDA A; ADD B; SUB C; STA D (7)

对应于下列浮点代码序列:

LDA A, STA ACC; LDA B, JMP FADD; LDA C, JMP FSUB; STA D (8)

程序 A(加法, 减法和规格化) 下列程序是实现算法 A 的一个子程序, 而且已把它设计成使得规格化部分亦可为出现在这一节后边的其它子程序所使用。在这个程序中以及贯穿于这一章的许多其它的程序中, OFLO 都表示一个这样的子程序, 它打印出一个信息, 告知 MIX 的溢出标记已意外地被发现是接通的。字节大小 b 假定是 4 的倍数。规格化子程序 NORM 假定 $r12 = e$ 和 $rAX = f$, 其中 $rA = 0$ 意味着 $rX = 0$ 和 $r12 < b$ 。

00	BYTE	EQU	1(4:4)	字节大小 b
01	EXP	EQU	1:1	指数字段的定义
02	FSUB	STA	TEMP	浮点减法子程序
03		LDAN	TEMP	改变操作数符号
04	FADD	STJ	EXITF	浮点加法子程序
05		JOV	OFLO	确保溢出标记关闭
06		STA	TEMP	$TEMP \leftarrow v$
07		LDX	ACC	$rX \leftarrow u$
08		CMPA	ACC(EXP)	<u>这里把步骤 A1, A2, A3 合并</u>
09		JGE	1F	如果 $e_v \geq e_u$, 则转移
10		STX	FU(0:4)	$FU \leftarrow \pm f f f f 0$
11		LD2	ACC(EXP)	$rI2 \leftarrow e_u$
12		STA	FV(0:4)	
13		LD1N	TEMP(EXP)	$rI1 \leftarrow -e_v$
14		JMP	4F	
15	1H	STA	FU(0:4)	$FU \leftarrow \pm f f f f 0 (u, v \text{ 交换})$
16		LD2	TEMP(EXP)	$rI2 \leftarrow e_w$
17		STX	FV(0:4)	
18		LD1N	ACC(EXP)	$rI1 \leftarrow -e_v$
19	4H	INC1	0, 2	$rI1 \leftarrow e_u - e_v$ (步骤 A4 不必要)
20	5H	LDA	FV	<u>A5. 右调</u>
21		ENTX	0	清 rX
22		SRAX	0, 1	右移 $e_u - e_v$ 位
23	6H	ADD	FU	<u>A6. 加</u>
24		JOV	N4	<u>A7. 规格化</u> 。若小数上溢则转移
25		JXZ	NORM	是容易的情况?
26		CMPA	$= 0 = (1:1)$	f 是规格化的吗?
27		JNE	N5	如果是, 舍入之
28		SRC	5	$ rX \leftrightarrow rA $
29		DECX	1	(rX 为正的)
30		STA	TEMP	(操作数有相反的符号,
31		STA	HALF(0:0)	在舍入和规格化之前寄存器
32		LDAN	TEMP	必须调整)

33		ADD	HALF	
34		ADD	HALF	低有效位部分求补
35		SRC	4	转移到规格化子程序
36		JMP	N3A	
37	HALF	CON	1//2	字大小的一半(符号改变)
38	FU	CON	0	小数部分 f_u
39	FV	CON	0	小数部分 f_v
40	NORM	JAZ	ZR0	<u>N1. 测试 f</u>
41	N2	CMPA	= 0 = (1:1)	<u>N2. f 是规格化的吗?</u>
42		JNE	N5	如果前导字节非零, 则转 N5
43	N3	SLAX	1	<u>N3. 左调</u>
44	N3A	DEC2	1	e 减 1
45		JMP	N2	返回 N2
46	N4	ENTX	1	<u>N4. 右调</u>
47		SRC	1	右移, 插入带适当符号的“1”
48		INC2	1	e 加 1
49	N5	CMPA	= BYTE/2 = (5:5)	<u>N5. 舍入</u>
50		JL	N6	尾部 < (1/2) b 吗?
51		JG	5F	
52		JXNZ	5F	尾部 > (1/2) b 吗?
53		STA	TEMP	尾部 = (1/2) b ; 舍入成奇数
54		LDX	TEMP(4:4)	
55		JXO	N6	如果 rX 为奇数, 则转 N6
56	5H	STA	* + 1(0:0)	存 rA 的符号
57		INCA	BYTE	加 b^{-4} 到 $ f $ 上(符号改变)
58		JOV	N4	校验舍入上溢
59	N6	J2N	EXPUN	<u>N6. 检验 e。若 $e < 0$, 则下溢</u>
60	N7	ENTX	0, 2	<u>N7. 归拢。 $rX \leftarrow e$</u>
61		SRC	1	
62	ZR0	DEC2	BYTE	$rI2 \leftarrow e - b$
63	8H	STA	ACC	
64	EXITF	J2N	*	跳出, 除非 $e \geq b$
65	EXPOV	HLT	2	发现指数上溢
66	EXPUN	HLT	1	发现指数下溢
67	ACC	CON	0	浮点累加器

25~37 行的代码部分是长了一点,但这是需要的,因为 MIX 只有一个 5 字节累加器可用于带符号数的相加,同时算法 A 一般地要求精度为 $2p+1=9$ 个位置。如果我们愿意牺牲一点它的精度,这个程序就可以缩短大约现有长度的一半,但我们在下一节将看到,充分的精度是重要的。第 55 行使用在 4.5.2 小节中定义的非标准 MIX 指令。浮点加法和减法的运行时间依赖于在 4.2.4 小节中分析的若干因素。

现在让我们研究乘法和除法,它们比加法更简单,而且它们彼此有些类似。

算法 M(浮点乘法和除法) 给定进制 b , 余 q, p 位数表示的规格化浮点数 $u = (e_u, f_u)$ 和 $v = (e_v, f_v)$ 。本算法形成乘积 $w = u \otimes v$ 或商 $w = u \oslash v$ 。

M1. [拆开] 把 u 和 v 表示的指数和小数部分分开(有时在这一步中测试操作数是否为 0 是方便的,但不是必要的)。

M2. [运算] 置

$$\begin{aligned} e_w &\leftarrow e_u + e_v - q, & f_w &\leftarrow f_u f_v & (\text{对于乘法}) \\ e_w &\leftarrow e_u - e_v + q + 1, & f_w &\leftarrow (b^{-1} f_u) / f_v & (\text{对于除法}) \end{aligned} \quad (9)$$

(因为假定了输入数是规格化的,由此推出, $f_w = 0$ 或 $1/b^2 \leq |f_w| < 1$, 或者已出现了除以 0 的错误。)如果必要,这时 f_w 可以缩短成 $p+2$ 位或 $p+3$ 位数字,如同习题 5 中那样。

M3. [规格化] 对 (e_w, f_w) 实施算法 N, 以规格化、舍入和归拢结果。(注:在这种情况下,规格化是比较简单的。因为左调顶多出现一次,而在除法之后,不会出现“舍入溢出”。) **||**

以下的 MIX 子程序可与程序 A 联合使用,它说明与算法 M 有关的必需的关于机器方面的考虑。

程序 M(浮点乘法和除法)

01	Q	EQU	BYTE/2	q 是字节大小之半
02	FMUL	STJ	EXITF	浮点乘法子程序
03		JOV	OFLO	确保上溢标记关闭
04		STA	TEMP	$TEMP \leftarrow v$
05		LDX	ACC	$rX \leftarrow u$
06		STX	FU(0:4)	$FU \leftarrow \pm f f f f 0$
07		LD1	TEMP(EXP)	
08		LD2	ACC(EXP)	
09		INC2	- Q, 1	$rI2 \leftarrow e_u + e_v - q$
10		SLA	1	
11		MUL	FU	f_u 乘以 f_v
12		JMP	NORM	规格化,舍入并跳出
13	FDIV	STJ	EXITF	浮点除法子程序
14		JOV	OFLO	确保清除溢出标记

15	STA	TEMP	$TEMP \leftarrow v$
16	STA	FV(0:4)	$FV \leftarrow \pm ffff0$
17	LD1	TEMP(EXP)	
18	LD2	ACC(EXP)	
19	DEC2	-Q,1	$rI2 \leftarrow e_u - e_v + q$
20	ENTX	0	
21	LDA	ACC	
22	SLA	1	$rA \leftarrow f_u$
23	CMFA	FV(1:5)	
24	JL	* + 3	如果 $ f_u < f_v $ 则转移
25	SRA	1	否则,右调 f_u
26	INC2	1	并把 rI2 加 1
27	DIV	FV	除
28	JNOV	NORM	规格化,舍入和跳出
29	DVZRO	HLT 3	没有规格化或以 0 作除数

这个程序最值得注意的特性是在行 23~26 中进行除法的措施,这样做是为了确保舍入后的答案有足够的精度。如果 $|f_u| < |f_v|$,则直截了当地应用算法 M 将在寄存器 A 中保留形如“ $\pm 0ffff$ ”的结果,而如果不对余数进行仔细的分析,就不能够进行正确的舍入(余数出现在寄存器 X 中)。所以在这种情况下,这个程序计算 $f_w \leftarrow f_u/f_v$,并确保在所有情况下 f_w 或者为 0,或者是规格化的。舍入可对 5 个有效字节进行,并且可能测试余数是否为 0。

我们有时需要进行定点和浮点表示之间的转换。“定化浮”的转换程序借助于上边的规格化算法容易得到。例如,在 MIX 中,下列子程序把一个整数转换成浮点数的形式:

01	FLOT	STJ	EXITF	假设 $rA = u$, 一个整数	
02		JOV	OFLO	确保清除溢出标记	
03		ENT2	Q + 5	置未加工的指数	(10)
04		ENTX	0		
05		JMP	NORM	规格化,舍入并跳出	■

“浮点化定点”子程序是习题 14 的课题。

浮点子程序的排错通常是一项困难的工作,因为需要考虑很多情况。这里是一份通常易于出错的“陷阱”的清单,这些“陷阱”常常使设计浮点程序的程序员或机器设计者落入圈套。

1) 丢失符号。在许多机器上(不是 MIX),寄存器之间的移位指令将影响符号,因此必须仔细地分析用于规格化和左调或右调的移位操作。当出现负 0 时,符号也屡被丢失(例如,程序 A 在 30~34 行中小心地保留寄存器 A 的符号,也见习题 6)。

2) 没有适当地处理指数上溢或下溢。只有在舍入和规格化之前,才能校验 e_w 的大小,因为先期测试可能给出一个错误的指示。指数下溢和上溢不仅在做浮点乘法和除法时可以出现,在做浮点加法和减法时也可出现;而且即使出现得很少也必须每次加以测试。在出现上溢或下溢之后,应保留足够的信息使得有可能采取有意义的改正动作。

可惜的是,在许多情况下人们已经习惯于忽略指数的下溢并简单地把下溢的结果置成 0,而不给出出错标志。这在大多数情况下引起了精度的严重损失(其实是损失了所有的有效位),并且破坏了奠定浮点算术的假定,所以当出现了下溢时,必须实际地告知程序员。只有在某些情况下,当结果以后要被加进一个相当大的量时,把结果置成 0 才是适当的。如果对指数的下溢不加检测,我们会发现不可思议的情况,即 $(u \otimes v) \otimes w$ 为 0,但 $u \otimes (v \otimes w)$ 则不然,因为 $u \otimes v$ 使指数下溢,但仍可计算 $u \otimes (v \otimes w)$ 而不使任何指数越出范围之外。类似地,如果不去检测指数下溢,则我们可以找到正数 a, b, c, d 和 y ,使得

$$\begin{aligned}(a \otimes y \oplus b) \otimes (c \otimes y \oplus d) &\approx \frac{2}{3} \\ (a \oplus b \otimes y) \otimes (c \oplus d \otimes y) &= 1\end{aligned}\quad (11)$$

(见习题 9。)即使浮点程序不是完全精确的,当 a, b, c, d 和 y 都是正数时,像(11)这样的矛盾也是十分意外的!程序员通常不预先处理指数下溢,所以关于这点应该告知他。*

3) 插入无用信息。当左调时,重要的是要防止在右边引进除 0 外的任何东西。例如,注意程序 A 第 21 行中的“ENTX 0”指令以及在 FLOT 子程序(10)的行 04 中的最易于忘记的“ENTX 0”指令。(但在除法子程序中的行 27 之后清除寄存器 X 将是一个错误。)

4) 非预见的舍入溢出。当类似 .999999997 的数被舍入成 8 位时,在小数点左边将出现一个进位,因而结果必须右调。许多人都错误地下结论说,做乘法时舍入溢出是不可能的,因为他们看到 $|f_u f_v|$ 的极大值是 $1 - 2b^{-p} + b^{-2p}$,因而这不能舍入成 1。这个推理的错误在习题 11 中示出。奇妙的是,在浮点除法过程中却不会出

* 另一方面,我们必须承认,今天的高级程序设计语言对于如何利用一个浮点程序要提供的信息,没有向程序员介绍令人满意的方法,或者只介绍很少的方法。本节中的 MIX 程序,当发现错误时,它干脆停机,这甚至更坏。在许多重要的应用中指数下溢是相对无害的,因此对于程序员来说,找出容易的和安全的对付这种情况的方法是值得的。径直用零代替下溢的做法已绝对地被怀疑。但还有另一个选择,它近来已经深受欢迎,这就是修改我们对于浮点数已经给出的定义。当指数有它最小可能的值时,允许有一个未规格化的小数部分。这个“逐渐下溢”的思想首先体现在 Electrologica X8 计算机的硬件上。它仅仅对算法增加少量的复杂性,而且使得在加法或减法期间下溢不可能出现。在存在有“逐渐下溢”的情况下 4.2.2 小节中关于相对误差的简单公式不再成立,所以这个题目超出了本书的范围。但是使用像 $\text{round}(x) = x(1 - \delta) + \epsilon$, 其中 $|\delta| < b^{1-p}/2$ 和 $|\epsilon| < b^{-p-q}/2$, 这样的公式,人们能够证明在许多重要情况下逐渐下溢是成功的。见 W. M. Kahan 和 J. Palmer, ACM SIGNUM Newsletter (1979 年 10 月), 13~21。——作者原注

现舍入溢出的现象(见习题 12)。

有一种学派说,把像 .999999997 这样的值舍入成 .99999999 而不是 1.0000000 是无害的,因为这不增加相对误差最坏情况的界。浮点数 1.0000000 可以说代表了区间 $[1.0000000 - 5 \times 10^{-8}, 1.0000000 + 5 \times 10^{-8}]$ 中的全部数,而 .99999999 代表在小得多的区间 $(.99999999 - 5 \times 10^{-9}, .99999999 + 5 \times 10^{-9})$ 中的所有数。尽管后一区间不包含原来的 .999999997,但后一区间的每一个数都包含在头一个区间当中,所以用第二个区间做以后的计算不会比第一个不精确。但是这个巧妙的论断同在 4.2.2 小节中所表达的浮点算术的数学原理不相容。

5)规格化之前舍入。不精确来自过早地在错误的数字位置上进行舍入。当舍入在适当位置的左边进行时,这个错误是明显的;但在不太明显的情况下,即先在右边很远的地方进行舍入然后再在正确的位置舍入,那也是危险的。由于这一原因,在步骤 A5 的右调操作过程中进行舍入是错误的,除非像习题 5 所给出的那样。(但是在 N5 中舍入的特殊情况下,当出现了舍入溢出之后再次进行舍入,却是无害的,因为舍入溢出总产生 ± 1.0000000 ,而这样的值不受以后的舍入过程的影响。)

6)在中间计算中没有保留足够的精度。下节进行的浮点算术的精确度的详细分析,有力地提示了规格化浮点程序应当总是产生适当地舍入的结果。产生这一论断应该没有什么例外,即使是在出现概率极小的情况下也是这样。如同在算法 A 和算法 M 中所指出的那样,在整个计算过程中,都应该保留适当个数的有效数字。

C. 浮点硬件 几乎每一台用于科学计算的大型计算机,都包含浮点算术作为它内部操作的指令表的一部分。可惜这样的硬件设计通常总有些毛病,它在某些情况下引起可怕地拙劣的性能。因此我们希望未来的计算机设计者们能比过去更加注意为计算机提供适当的性能。这只需多花不多的钱来造出正确的机器,而下一节的一些考虑表明得到的好处却很大。就我们现在所知来说,过去的折中妥协对于现代的机器不再是合适的了。

MIX 计算机在本套书中是当做“典型”机器的一个例子,它有一套包括以下 7 个操作的任选的“浮点附件”(另付费用可买到):

• FADD, FSUB, FMUL, FDIV, FLOT, FCMP (分别有 $C = 1, 2, 3, 4, 5, 56$; $F = 6$)。在“FADD V”操作之后 rA 的内容和经过操作

STA ACC; LDA V; JMP FADD

之后 rA 的内容完全相同,其中 FADD 是在这节前边出现的子程序。不同之处是,如果两个操作数在进入子程序之前还不具规格化的形式,则它们都将被自动地进行规格化(在进行这个预先的规格化期间而不是在对答案进行规格化期间出现指数下溢,则不给出下溢标志)。类似的说法,亦可应用于 FSUB, FMUL 和 FDIV。在执行“FLOT”操作之后, rA 的内容是执行上述子程序(10)中的“JMP FLOT”后的内容。

rA 的内容经“FCMP V”操作后不变,这条指令依赖 rA 的内容是否“确定地小于”,“近似地等于”或“确定地大于”V 的内容,而把比较指示器置为 LESS, EQUAL 或 GREATER。下节将讨论这个课题。而且其确切的动作由习题 4.2.2-17 的子程序

FCMP 定义,其中 EPSILON 在单元 0 中。

rA 以外的寄存器均不受任何浮点操作的影响。如果出现指数上溢或下溢,则置上溢出标记,而且答案中的指数是以模字节大小的形式给出的。以 0 做除数在 rA 中产生无定义的无用信息。执行时间分别为 $4u, 4u, 9u, 11u, 3u, 4u$ 。

• FIX($C=5; F=7$)。rA 的内容用整数“round(rA)”来代替,像在算法 N 的步骤 N5 中那样舍入成最接近的整数。然而如果这个答案太大,在寄存器中装不下,则置溢出标记且结果无定义,执行时间为 $3u$ 。

有时以非标准的方式来使用浮点操作是有帮助的。例如,如果操作 FLOT 未被作为 MIX 的浮点附件的一部分包括进来,则编写

```

FLOT  STJ    9F
      SLA    1
      ENTX   Q+4
      SRC    1
      FADD   =0 =
9H    JMP    *

```

(12)

就能容易地实现它对 4 字节数的作用。这个程序并不严格地等价于 FLOT 操作,因为它假定 rA 的 1:1 字节为 0,而且它会破坏 rX。处理更一般的情况有点窍门,因为甚至在一个 FLOT 操作期间也可能出现舍入溢出。

类似地,假设 MIX 有一个 FADD 操作,但没有 FIX 操作。如果我们要把一个数 u 从浮点形式舍入成为最接近的定点数,而且如果我们知道这个数是非负的,并且至多占据三个字节,则可以写

FADD FUDGE

其中 FUDGE 包含常数

+	Q+4	1	0	0	0
---	-----	---	---	---	---

rA 中的结果将是

+	Q+4	1	round(u)值
---	-----	---	---------------

(13)

D. 历史和文献 浮点记法的起源可追溯到巴比伦时代(公元前 1800 年或更早些)的数学家们,他们广泛使用 60 进制的浮点算术,但没有关于指数的记法。进行这些计算的人们对于指数应该是什么好像总是“清楚”的。虽然至少已经发现了一个例子,说明由于在实施加法时不适当地调整操作数,因而给出了错误的答案,但这种例子是很稀少的;见 O. Neugebauer, *The Exact Sciences in Antiquity* (Princeton, N. J.: Princeton University Press, 1952), 26~27。对于浮点记法另一个早期的贡献应归功于希腊数学家 Apollonius(公元前 3 世纪),至少在一些简单情况下,他显然是第一个说明怎样分别地从 10 的幂的系数来累加这些幂以简化乘法的人。[关于 Apollonius 方法的讨论,见 Pappus, *Mathematical Collections* (公元 4 世纪)。]在巴比伦文

化消亡以后,直到很晚,大约在发明对数(1600年)及不久之后 Oughtred 发明计算尺(1630年)之前,积和商的浮点记法一直很少使用。大约同一时间,引进了指数的现代记号“ x^n ”;在这以前,已经使用了 x 的平方, x 的立方等符号。

浮点算术已被某些最早的计算机的设计所采用。1914年,Leonardo Torres y Quevedo 在马德里,1936年 Konrad Zuse 在柏林,1939年 George Stibitz 在新泽西,都独立地提出了它。Zuse 的机器使用他称做半对数记号的浮点二进表示,还可以处理像“ ∞ ”和“无定义”这样的特殊量。用浮点算术硬件进行操作的头一台美国计算机是贝尔实验室的 Model V 和哈佛大学的 Mark II,这两台机器都是在1944年设计的继电计算器。[见 B. Randell, *The Origins of Digital Computers* (Berlin: Springer, 1973), 100, 155, 163~164, 259~260; *Proc. Symp. Large-Scale Digital Calculating Machinery* (Harvard, 1947), 41~68, 69~79; *Datamation* 13 (1967年4月), 35~44 (1967年5月), 45~49; *Zeit. für angew. Math. und Physik* 1 (1950), 345~346。]


在1944年—1946年期间,穆尔学校的研究者在制造头一台电子数字计算机的计划中,极认真地考虑了浮点二进制算术的用法,但他们发现,以电子管来实现浮点线路比起用继电器来更困难得多。这个小组认识到,调整比例在程序设计中是个问题;但他们知道在那时它不过是整个程序设计工作中很小的一部分;确实,明确的定点比例似乎值得花费时间和气力来做,因为它使程序员对所得数值的精确度保持了解。其次,机器的设计者论断说,浮点表示将耗费大量宝贵的存储空间,因为必须存储指数,而且对于多精度计算采取浮点算术运算是困难的。[见 von Neumann, *Collected Works* 5 (New York: Macmillan, 1963), 43, 73~74。]当然,这时候他们正在设计头一台存储程序计算机以及第二台电子计算机,而他们的选择不得不或者采用定点,或者采用浮点,而不是两者兼用。他们已先期编制了计算浮点二进数的子程序,而且事实上把“左移”和“右移”指令放进他们的机器中,主要是为了使这样的子程序更为有效。在硬件中同时具有两种类型的算术运算的头一台机器显然是通用电子公司(General Electric Company)研制的一台计算机[见 *Proc. 2nd Symp. Large-Scale Digital Calculating Machinery* (Cambridge, Mass.: Harvard University Press, 1951), 65~69]。

早期机器的浮点子程序和解释系统是由 D. J. Wheeler 和其他一些人编码的,这些子程序首先发表在 Wilkes, Wheeler 和 Gill 所著的 *The Preparation of Programs for an Electronic Digital Computer* 上(Reading, Mass.: Addison-Wesley, 1951),子程序 A1~A11, 35~37, 105~117。有趣的是,这里描述了浮点十进算术运算,尽管所使用的是一台二进制的计算机;换言之,数被表示成 10^f ,而不是 2^f ,因此调整比例操作需要乘以10或除以10。在这台具体的机器上这种十进制的调整比例大约就像移位那样容易,而且十进方法大大地简化了输入输出转换。

大多数涉及浮点运算程序细节的公开文献,都散见于各计算机厂家分发的技术备忘录中,偶尔也出现在公开的著作中。除了上述的文献外,下面一些是有历史价值的:R. H. Stark 和 D. B. MacMillan, *Math. Comp.* 5 (1951), 86~92,其中描述了一个线路板程序;D. McCracken, *Digital Computer Programming* (New York: Wiley,

1957), 121~131; J. W. Carr III, CACM 2, 5 (1959 年 5 月), 10~15; W. G. Wadey, JACM 7 (1960), 129~139; D. E. Knuth, JACM 8 (1961), 119~128; O. Kesner, CACM 5 (1962), 269~271; F. P. Brooks 和 K. E. Iverson, Automatic Data Processing (New York: Wiley, 1963), 184~199。关于从计算机设计者的观点进行浮点算术的讨论, 见 W. Buchholz 所编 *Planning a Computer System* [(New York: McGraw-Hill, 1962), 92~121] 一书中 S. G. Campbell 写的“Floating point operation”; A. Padegs, IBM Systems J. 7 (1968), 22~29。另外一些主要涉及浮点方法精度的文献, 在 4.2.2 小节中给出。

在 20 世纪 80 年代末, 当大多数厂家开始采用 ANSI/IEEE 标准 754 之后, 在浮点硬件方面发生了革命性变化。有关的文献是: IEEE Micro 4 (1984), 86~100; W. J. Cody, Comp. Sci. and Statistics: Symp. on the Interface 15 (1983), 133~139; W. M. Kahan, Mini/Micro West-83 Conf. Record (1983), 论文 16/1; D. Goldberg, Computing Surveys 23 (1991), 5~48, 413; W. J. Cody 和 J. T. Coonen, ACM. Trans. Math. Software 19 (1993), 443~451。

 将在本书的下一版中替代 MIX 的 MMIX 计算机, 自然地将符合新标准。

习 题

1. [10] 怎样以 100 进制, 余量 50, 四位浮点记法表示阿伏加德罗数和普朗克常数 (如果字节大小为 100, 则像在 (4) 中那样, 这将是 MIX 所使用的表示)?

2. [12] 假设指数 e 限定在 $0 \leq e \leq E$ 的范围中, 能写成 b 进制, 余 q, p 位数字的浮点数的最大和最小正值是多少? 在这些规定下, 可以写成规格化的浮点数的最大和最小正值是什么?

3. [11] (K. Zuse, 1936) 证明, 如果使用规格化的浮点二进算术运算, 则有一个办法可以稍稍提高精度而不损失存储空间; 即如果指数值的范围减少一点点的话, 则可仅用 $p-1$ 位来表示 p 位小数部分。

► 4. [16] 假定 $b=10, p=8$ 。对于 $(50, +.98765432) \oplus (49, +.33333333)$, 算法 A 给出什么结果? 对于 $(53, -.99987654) \oplus (54, +.10000000)$ 呢? 对于 $(45, -.50000001) \oplus (54, +.10000000)$ 呢?

► 5. [24] 我们说 $x \sim y$ (关于一个给定的进制 b) 的充分必要条件是, x 和 y 是满足下列条件的实数:

$$\begin{aligned} \lfloor x/b \rfloor &= \lfloor y/b \rfloor \\ x \bmod b = 0 &\Leftrightarrow y \bmod b = 0 \\ 0 < x \bmod b < \frac{1}{2}b &\Leftrightarrow 0 < y \bmod b < \frac{1}{2}b \\ x \bmod b = \frac{1}{2}b &\Leftrightarrow y \bmod b = \frac{1}{2}b \\ \frac{1}{2}b < x \bmod b < b &\Leftrightarrow \frac{1}{2}b < y \bmod b < b \end{aligned}$$

证明如果在算法 A 的步骤 A5 与 A6 之间以 $b^{-p-2}F_v$ 代替 f_v , 其中 $F_v \sim b^{p+2}f_v$, 则该算法的结果将不变。(如果 F_v 是一个整数而 b 为偶数, 这一操作实际上把 f_v 截成 $p+2$ 位, 同时记住是否有

任何非零数字被去掉了,因此步骤 A6 中进行加法所需要的寄存器长度达到极小。)

6. [20] 如果一个 FADD 指令的结果为 0,则按照本节中给出的 MIX 浮点指令的定义 rA 的符号为何?

7. [27] 利用平衡的三进制系统,讨论浮点算术运算。

8. [20] 假定指数必须满足 $0 \leq e < 100$,试给出规格化的 8 位数字的浮点十进数 u 和 v ,对于它们,加法产生 (a) 指数下溢, (b) 指数上溢。

9. [M24] (W. M. Kahan) 假定在指数下溢时以 0 来代替结果,同时又不给出错误指示。试利用余 0.8 位数字的,且 e 在范围 $-50 \leq e < 50$ 内的浮点十进数,找出使得 (11) 成立的正的 a, b, c, d 和 y 的值。

10. [12] 给出规格化的 8 位数字的浮点十进数 u 和 v 的一个例子,使得它们在加法中出现舍入溢出。

► 11. [M20] 给出规格化的余 50 的 8 位数字的浮点十进数 u 和 v 的一个例子,使得它们在乘法中出现舍入溢出。

12. [M25] 证明在浮点除法的规格化阶段中不能出现舍入溢出。

13. [30] 在“区间算术”中,我们不对浮点计算的结果实行舍入;而是要实现像 ∇ 和 ∇ 这样的运算,它们给出对于真正的和最严格的可表示的界:

$$u \nabla v \leq u + v \leq u \Delta v$$

为此目的,本节的算法应做怎样的修改?

14. [25] 写出一个 MIX 子程序,它从寄存器 A 中的任意浮点数(不必是规格化的)开始,并把这个数转换成最接近的定点整数(或者确认其绝对值太大,从而不可能来做这样的转换)。

► 15. [28] 写出一个 MIX 子程序,对于给定的浮点数 u ,它能在这一节的其它子程序的帮助下,计算 $u \pmod{1}$,即计算 $u - \lfloor u \rfloor$ 并舍入到最近的浮点数。注意,当 u 是一个非常小的负数时, $u \pmod{1}$ 可能被舍入成为结果是 1 的数(即使 $u \pmod{1}$ 已定义成总是小于 1 的一个实数,也不例外)。

16. [HM21] (Robert L. Smith) 给定实的浮点值 a, b, c 和 d ,试设计一个计算复数 $(a + bi)/(c + di)$ 的实部和虚部的算法。避免计算 $c^2 + d^2$,因为它将引起浮点溢出,甚至当 $|c|$ 或 $|d|$ 是允许的极大浮点值的近似平方根时,也是如此。

17. [40] (John Cocke) 定义一种单字长表示,在这种表示下,其小数部分的精度随指数量度的增加而减少。由此探讨推广浮点数的范围的思想。

18. [25] 考虑字长 36 位的一台二进计算机,其正浮点二进制数表示为 $(0e_1e_2 \cdots e_8f_1f_2 \cdots f_{27})_2$; 这里 $(e_1e_2 \cdots e_8)_2$ 是一个余 $(10000000)_2$ 的指数,而 $(f_1f_2 \cdots f_{27})_2$ 是一个 27 位的小数。负的浮点数用相应正表示的 2 的补码给出(见 4.1 节)。因此 1.5 在八进记法下是 $201 \mid 600000000$,而 -1.5 是 $576 \mid 200000000$; 1.0 和 -1.0 的八进表示分别为 $201 \mid 400000000$ 和 $576 \mid 400000000$ (这里的垂直线用来表示指数和小数部分的界)。注意,一个规格化的正数的位 f_1 必为 1,而对于负数它几乎总为 0;例外情况是 2^{-k} 的表示。

假设一个浮点运算的准确结果有八进码 $572 \mid 740000000 \mid 01$; 这(负的)33 位小数必须规格化和舍入成为 27 位。如果左移直到前导小数位为 0,我们得到 $576 \mid 000000000 \mid 20$,但这会舍入成不合法的值 $576 \mid 000000000$; 我们已经过分规格化了,因为正确答案是 $575 \mid 400000000$ 。另一方面,如果(在某个其它的问题中)我们以值 $572 \mid 740000000 \mid 05$ 开始,而且在过分规格化之前停止,就会得到 $575 \mid 400000000 \mid 50$,它舍入到非规格化的数 $575 \mid 400000001$; 随后的规格化得出 576

|000000002,而正确的答案为 576|000000001。

给出一个在这样的机器上解决这个矛盾的简单而又正确的舍入规则(不抛弃 2 的补码记号)。

19.[24] 就数据的有关特征来说,程序 A 中的 FADD 指令的运行时间是多少?对于不引起指数上溢或下溢的所有输入而言,极大运行时间是多少?

Round numbers are always false.

舍入的数总是不真实的。

——SAMUEL JOHNSON (1750)

*I shall speak in round numbers, not absolutely accurate,
yet not so wide from truth as to vary the result materially.*

我将在演讲中使用舍入的数,它们不是绝对精确的,
但也没有远离真实到如此的地步,以致会显著地改变结果。

——THOMAS JEFFERSON (1824)

4.2.2 浮点算术的精确度

就本质说来,浮点算术是不精确的,而且程序员们很容易就会滥用它,从而使计算的答案几乎全由“噪声”所组成。数值分析的主要问题之一,是确定某个数值方法的结果的精确度如何。这里有“信用差距”的问题:即我们不知道计算机的答案有多少是可信的。计算机的新用户无保留地信任计算机,把它当做不出错的权威;他们倾向于相信打印出的答案的所有数字都是有意义的。醒悟了的计算机用户恰持相反的态度,他们经常怕他们的答案几乎是无意义的。许多认真的数学家曾试图对于浮点操作的序列给出严格的分析,结果发现这个任务实在人得惊人,只好做些含糊其词的论证来聊以自慰。

当然,对于误差分析技术的彻底考察,超出了本书的范围,但我们将在这一小节中研究浮点技术的误差的某些特征。我们的目标是来发现应以什么样一种方式实施浮点算术,使得误差传播的合理分析尽可能简便。

表达浮点算术运算特性的一个粗糙的(但相当有用的)方式,可以以“有效位”或相对误差的概念为基础。如果我们用近似值 $\hat{x} = x(1 + \epsilon)$ 来表示在一台计算机内的一个精确实数 x ,则量 $\epsilon = (\hat{x} - x)/x$ 称为该近似值的相对误差。粗略地说,浮点乘法和除法的运算并不使相对误差扩大很多;但近于相等的量的减法(以及浮点相加 $u \oplus v$,其中 u 近似地等于 $-v$)则可大大增加其相对误差。所以我们有一个一般的经验法则,即从这样的加法和减法中,而不是从乘法和除法中去寻找主要的精度损失。另一方面,这一情况有点不合常人的想法,它需要适当地加以理解,因为“坏”的加法和减法是以完全的精度被执行的(见习题 25)。

浮点加法可能的不可靠性的后果之一,是破坏了结合律:

$$(u \oplus v) \oplus w \neq u \oplus (v \oplus w), \quad \text{对许多 } u, v, w \quad (1)$$

例如

$$(11111113. \oplus -11111111.) \oplus 7.5111111 = 2.0000000 \oplus 7.5111111 = 9.5111111$$

$$11111113.\ominus(-11111111.\oslash 7.5111111) = 11111113.\oplus -11111103. = 10.000000$$

(这节中的所有例子都是以8位浮点十进算术给出的;而且以明显的小数点位置来指出指数。回忆一下,如同在4.2.1小节那样,符号 $\oplus, \ominus, \otimes, \oslash$ 是用来代表浮点运算的,它们对应于精确运算 $+, -, \times, /$ 。)

就破坏结合律这一点来说,在这一章开头出现的 La Touche 夫人的评论对于浮点算术很有意义。像“ $a_1 + a_2 + a_3$ ”或“ $\sum_{k=1}^n a_k$ ”这样的数学记号本来就是以结合律的假定为基础的,所以一个程序员必须特别小心,不得暗中假定结合律的正确性。

A. 一个公理化的方法 尽管结合律不是正确的,但交换律

$$u \oplus v = v \oplus u \quad (2)$$

是成立的,而且这一定律对于程序设计和程序分析来说,在概念上不失为一种有价值的财富。等式(2)提示,我们应寻找一些为 \oplus, \ominus, \otimes 和 \oslash 所满足的重要定律;而且,我们认为,应该把浮点程序设计成为使尽可能多的通常数学定律保持有效,这才是合理的。如果有更多的公理成立,则就更容易写出好的程序来,而且程序也就变得更从一台机器移植到另一台机器上去。

因此现在让我们考虑某些其它的基本定律,它们对于前一节所述的规格化的浮点运算是正确的。首先我们有

$$u \ominus v = u \oplus -v \quad (3)$$

$$-(u \oplus v) = -u \oplus -v \quad (4)$$

$$u \ominus v = 0 \quad \text{当且仅当} \quad v = -u \quad (5)$$

$$u \oplus 0 = u \quad (6)$$

从这些定律我们可以导出进一步的恒等式;例如(习题1)

$$u \ominus v = -(v \ominus u) \quad (7)$$

恒等式(2)到(6)容易从4.2.1小节的算法导出,下列规则稍微不大显然:

$$\text{如果 } u \leq v \quad \text{则} \quad u \oplus w \leq v \oplus w \quad (8)$$

我们可以不通过分析算法4.2.1A来证明这个规则,而是回过头去看奠定该算法的设计原理(算法的证明并不总比数学的证明更容易些),我们的思想是浮点运算应满足

$$u \oplus v = \text{round}(u + v), \quad u \ominus v = \text{round}(u - v) \quad (9)$$

$$u \otimes v = \text{round}(u \times v), \quad u \oslash v = \text{round}(u/v)$$

如同在算法4.2.1N中定义的,round(x)代表 x 的最好的浮点近似。我们有

$$\text{round}(-x) = -\text{round}(x) \quad (10)$$

$$x \leq y \quad \text{蕴涵} \quad \text{round}(x) \leq \text{round}(y) \quad (11)$$

而且这些基本的关系直接推出性质(2)到(8)。我们也可以写出更多的恒等式:

$$u \otimes v = v \otimes u, \quad (-u) \otimes v = -(u \otimes v), \quad 1 \otimes v = v$$

$$u \otimes v = 0 \quad \text{当且仅当} \quad u = 0 \text{ 或 } v = 0$$

$$(-u) \oslash v = u \oslash (-v) = -(u \oslash v)$$

$$0 \oslash v = 0, \quad u \oslash 1 = u, \quad u \oslash u = 1$$

如果 $u \leq v$ 和 $w > 0$, 则 $u \otimes w \leq v \otimes w$, 且 $u \odot w \leq v \odot w$; 当 $v \geq u > 0$ 时, $w \odot u \geq w \odot v$ 。如果 $u \oplus v = u + v$, 则 $(u \oplus v) \ominus v = u$; 且如果 $u \otimes v = u \times v \neq 0$, 则 $(u \otimes v) \odot v = u$ 。我们看到, 如果定义下得适当, 则尽管浮点运算是精确的, 仍有大量规则存在。

当然, 若干熟悉的代数规则仍然显然地不存在于上边的恒等式集合中; 如习题 3 中所示, 浮点乘法的结合律不是严格正确的。而且稍微更糟的是 \otimes 和 \oplus 之间的分配律不成立。设 $u = 20000.000$, $v = -6.0000000$ 以及 $w = 6.0000003$; 则

$$(u \otimes v) \oplus (u \otimes w) = -120000.00 \oplus 120000.01 = .010000000$$

$$u \otimes (v \oplus w) = 20000.000 \otimes .00000030000000 = .0060000000$$

所以

$$u \otimes (v \oplus w) \neq (u \otimes v) \oplus (u \otimes w) \quad (12)$$

另一方面, 当 b 是浮点进位时, 我们有 $b \otimes (v \oplus w) = (b \otimes v) \oplus (b \otimes w)$, 因为

$$\text{round}(bx) = b \text{round}(x) \quad (13)$$

(严格地说, 我们在这一节里考虑的恒等式和不等式隐含地假定指数的下溢和上溢不出现。当 $|x|$ 太小或太大时函数 $\text{round}(x)$ 是无定义的, 而且像 (13) 这样的等式仅当两边有定义时才成立。)

浮点算术使传统代数被破坏的另一个重要例子是柯西基本不等式

$$(x_1^2 + \cdots + x_n^2)(y_1^2 + \cdots + y_n^2) \geq (x_1 y_1 + \cdots + x_n y_n)^2$$

出问题。习题 7 表明, 甚至在 $n = 2, x_1 = x_2 = 1$ 的简单情况下, 它也不成立。使用教科书上的公式

$$\sigma = \sqrt{\left(n \sum_{1 \leq k \leq n} r_k^2 - \left(\sum_{1 \leq k \leq n} x_k\right)^2\right) / n(n-1)} \quad (14)$$

来计算某些观察的标准差的新程序员经常发现他们取的是一个负数的平方根! 用浮点算术来计算均值和标准差的好得多的方式是使用递推式

$$M_1 = x_1, \quad M_k = M_{k-1} \oplus (x_k \ominus M_{k-1}) \odot k \quad (15)$$

$$S_1 = 0, \quad S_k = S_{k-1} \oplus (x_k \ominus M_{k-1}) \otimes (x_k \ominus M_k) \quad (16)$$

其中 $2 \leq k \leq n$, $\sigma = \sqrt{S_n / (n-1)}$ [参考 B. P. Welford, *Technometrics* 4 (1962), 419~420]。由这个公式可知, S_n 绝不能是负的, 而且如习题 16 中所证明的那样, 我们可以避免通常的累计求和方法遇到的其它严重问题(关于能更好保证精度的一个求和技术, 见习题 19)。

尽管代数法则不一定都成立, 但我们通常可以证明它们也不是太离谱的。当

$b^{e-1} \leq x < b^e$ 时, 我们有 $\text{round}(x) = x + \rho(x)$, 其中 $|\rho(x)| \leq \frac{1}{2} b^{e-p}$, 因此

$$\text{round}(x) = x(1 + \delta(x)) \quad (17)$$

其中相对误差有与 x 无关的界:

$$|\delta(x)| = \frac{|\rho(x)|}{|x|} \leq \frac{|\rho(x)|}{b^{e-1} + |\rho(x)|} \leq \frac{\frac{1}{2}b^{e-p}}{b^{e-1} + \frac{1}{2}b^{e-p}} < \frac{1}{2}b^{1-p} \quad (18)$$

我们可以使用这个不等式来简单地估计规格化浮点计算中的相对误差, 因为 $u \oplus v = (u + v)(1 + \delta(u + v))$, 等等。

作为典型的误差估计过程的一个例子, 我们考虑乘法的结合律。习题 3 表明 $(u \otimes v) \otimes w$ 一般不等于 $u \otimes (v \otimes w)$, 但是这里的形势比加法的结合律(1)和分配律(12)面临的形势要好得多。事实上, 假定不出现指数下溢或上溢, 对于某些 $\delta_1, \delta_2, \delta_3, \delta_4$, 我们有

$$(u \otimes v) \otimes w = ((uv)(1 + \delta_1)) \otimes w = uvw(1 + \delta_1)(1 + \delta_2)$$

$$u \otimes (v \otimes w) = u \otimes ((vw)(1 + \delta_3)) = uvw(1 + \delta_3)(1 + \delta_4)$$

其中 $|\delta_j| < \frac{1}{2}b^{1-p}$, 对于每一个 j 。因此

$$\frac{(u \otimes v) \otimes w}{u \otimes (v \otimes w)} = \frac{(1 + \delta_1)(1 + \delta_2)}{(1 + \delta_3)(1 + \delta_4)} = 1 + \delta$$

其中

$$|\delta| < 2b^{1-p}/(1 - \frac{1}{2}b^{1-p})^2 \quad (19)$$

在这种分析中, 数 b^{1-p} 出现得如此频繁, 因而对它已经起了一个特殊名字, 叫一个 ulp, 意思是在分数部分的“最后位置中的一个单位”。浮点运算正确到半个 ulp 以内, 而且用两个浮点乘法计算 uvw 将正确到大约一个 ulp 之内(忽略二次项), 因此乘法的结合律在大约两个 ulp 的相对误差之内成立。

我们已经证明 $(u \otimes v) \otimes w$ 近似地等于 $u \otimes (v \otimes w)$, 除非发生指数上溢或下溢。值得更详细地研究近似相等的直观概念, 我们能以一个合理的方式更精确地给出这样的陈述吗?

使用浮点算术的程序员几乎总不希望测试是否两个被计算的数彼此完全相等(或至少他几乎不需要尝试这样做), 因为这是极不可能出现的事。例如, 在使用一个递推式

$$x_{n+1} = f(x_n)$$

时, 如果因为某一教科书的理论指出, 当 $n \rightarrow \infty$ 时, x_n 趋于一个极限, 你就去等候使得 $x_{n+1} = x_n$ 的 n 出现, 这通常是一个错误。因为由于中间结果的舍入, 序列 x_n 可能是周期的且周期较长。正确的方法是, 对于某个适当选定的数 δ , 等候到 $|x_{n+1} - x_n| < \delta$; 但因为我们并不需要预先知道量 x_n 的阶, 等候到

$$|x_{n+1} - x_n| \leq \epsilon |x_n| \quad (20)$$

甚至更好。现在 ϵ 是一个更易于选定的数。关系(20)是叙述 x_{n+1} 和 x_n 近似相等的另一个方式, 而且我们的讨论表明, 当涉及浮点计算时, 如果我们仅能定义一个适

当的近似关系,则一个“近似相等”的关系将比传统的相等关系更加有用。

换句话说,浮点值的严格相等不那么重要,这一事实意味着我们应当有一个新的运算,即浮点比较,其目的在于帮助评价两个浮点量的相对值。对于进制 b 、余 q 的浮点数 $u = (e_u, f_u)$ 和 $v = (e_v, f_v)$, 下列定义似乎是适当的:

$$u \prec v(\epsilon) \text{ 当且仅当 } v - u > \epsilon \max(b^{e_u - q}, b^{e_v - q}) \quad (21)$$

$$u \sim v(\epsilon) \text{ 当且仅当 } |v - u| \leq \epsilon \max(b^{e_u - q}, b^{e_v - q}) \quad (22)$$

$$u \succ v(\epsilon) \text{ 当且仅当 } u - v > \epsilon \max(b^{e_u - q}, b^{e_v - q}) \quad (23)$$

$$u \approx v(\epsilon) \text{ 当且仅当 } |v - u| \leq \epsilon \min(b^{e_u - q}, b^{e_v - q}) \quad (24)$$

这些定义既可应用于规格化的值,也可应用于非规格化的值。注意,对于给定的任何一对值 u 和 v , 必定恰有条件 $u \prec v$ (确定地小于), $u \sim v$ (近似地等于) 或 $u \succ v$ (确定地大于) 之一成立。关系 $u \approx v$ 比 $u \sim v$ 稍微更强些,它可以读作“ u 实质上等于 v ”。所有这些关系都是借助于一个正实数 ϵ 给出的,这个 ϵ 表示所考虑的近似程度。

观察上述定义的一个方法是把每个浮点数 u 同个“邻域”集合 $N(u) = \{x \mid |x - u| \leq b^{e_u - q}\}$ 结合起来;于是, $N(u)$ 表示按照 u 的浮点表示的指数计算接近于 u 的一个值集。在这些术语之下,我们有 $u \prec v$ 当且仅当 $N(u) < v$ 及 $u < N(v)$; $u \sim v$ 当且仅当 $u \in N(v)$ 或 $v \in N(u)$; $u \succ v$ 当且仅当 $u > N(v)$ 及 $N(u) > v$; $u \approx v$ 当且仅当 $u \in N(v)$ 和 $v \in N(u)$ 。(这里假定,测定近似程度的参数 ϵ 是一个常数;一个更完备的记号将指出 $N(u)$ 对 ϵ 的依赖性。)

下边是上述定义的一些简单推论:

$$\text{如果 } u \prec v(\epsilon) \text{ 则 } v \succ u(\epsilon) \quad (25)$$

$$\text{如果 } u \approx v(\epsilon) \text{ 则 } u \sim v(\epsilon) \quad (26)$$

$$u \approx u(\epsilon); \quad (27)$$

$$\text{如果 } u \prec v(\epsilon) \text{ 则 } u < v \quad (28)$$

$$\text{如果 } u \prec v(\epsilon_1) \text{ 且 } \epsilon_1 \geq \epsilon_2 \text{ 则 } u \prec v(\epsilon_2) \quad (29)$$

$$\text{如果 } u \sim v(\epsilon_1) \text{ 且 } \epsilon_1 \leq \epsilon_2 \text{ 则 } u \sim v(\epsilon_2) \quad (30)$$

$$\text{如果 } u \approx v(\epsilon_1) \text{ 且 } \epsilon_1 \leq \epsilon_2 \text{ 则 } u \approx v(\epsilon_2) \quad (31)$$

$$\text{如果 } u \prec v(\epsilon_1) \text{ 且 } v \prec w(\epsilon_2) \text{ 则 } u \prec w(\min(\epsilon_1, \epsilon_2)) \quad (32)$$

$$\text{如果 } u \approx v(\epsilon_1) \text{ 且 } v \approx w(\epsilon_2) \text{ 则 } u \sim w(\epsilon_1 + \epsilon_2) \quad (33)$$

而且,我们可以毫无困难地证明

$$|u - v| \leq \epsilon |u| \text{ 且 } |u - v| \leq \epsilon |v| \text{ 意味着 } u \approx v(\epsilon) \quad (34)$$

$$|u - v| \leq \epsilon |u| \text{ 或 } |u - v| \leq \epsilon |v| \text{ 意味着 } u \sim v(\epsilon) \quad (35)$$

反过来,当 $\epsilon < 1$ 时,对于规格化的浮点数 u 和 v ,

$$u \approx v(\epsilon) \text{ 意味着 } |u - v| \leq b\epsilon |u| \text{ 且 } |u - v| \leq b\epsilon |v| \quad (36)$$

$$u \sim v(\epsilon) \text{ 意味着 } |u - v| \leq b\epsilon |u| \text{ 或 } |u - v| \leq b\epsilon |v| \quad (37)$$

假设 $\epsilon_0 = b^{1-p}$ 是一个 ulp。(17)的推导建立了不等式 $|x - \text{round}(x)| =$

$|\rho(x)| < \frac{1}{2}\epsilon_0 \min(|x|, |\text{round}(x)|)$, 因此

$$x \approx \text{round}(x) \quad \left(\frac{1}{2}\epsilon_0\right) \quad (38)$$

由此得出, $u \oplus v \approx u + v \quad (\frac{1}{2}\epsilon_0)$, 等等。上边导出的乘法的近似结合律也可以由下列推算得到: 由(19), 我们有

$$|(u \otimes v) \otimes w - u \otimes (v \otimes w)| < \frac{2\epsilon_0}{\left(1 - \frac{1}{2}\epsilon_0\right)^2} |u \otimes (v \otimes w)|$$

把 $(u \otimes v) \otimes w$ 和 $u \otimes (v \otimes w)$ 互换, 同样的不等式成立。因此由(34), 当 $\epsilon \geq 2\epsilon_0 / \left(1 - \frac{1}{2}\epsilon_0\right)^2$ 时,

$$(u \otimes u) \otimes w \approx u \otimes (v \otimes w) \quad (\epsilon) \quad (39)$$

例如, 如果 $b=10, p=8$, 我们可以取 $\epsilon=0.00000021$ 。

在数值算法中, 关系 \prec, \sim, \succ 和 \approx 是有用的, 因此提供浮点数算术运算以及对它们进行比较的一些子程序是一个好主意。

现在我们把注意力转回到寻求浮点运算所满足的精确关系的问题。有趣的是从一种合理观点来看, 浮点加法和减法并不是完全无法对付的; 因为它们都满足下列一些定理所述的并非一目了然的恒等式。

定理 A 设 u 和 v 是规格化的浮点数。假定不出现指数上溢和下溢, 则

$$((u \oplus v) \ominus u) + ((u \oplus v) \ominus ((u \oplus v) \ominus u)) = u \oplus v \quad (40)$$

这个看上去稍微麻烦的恒等式, 可以改写为如下较简单的形式, 设

$$\begin{aligned} u' &= (u \oplus v) \ominus v, & v' &= (u \oplus v) \ominus u \\ u'' &= (u \oplus v) \ominus v', & v'' &= (u \oplus v) \ominus u' \end{aligned} \quad (41)$$

直观上, u' 和 u'' 应该近似于 u , 而 v' 和 v'' 应该近似于 v 。定理 A 告诉我们

$$u \oplus v = u' + v'' = u'' + v' \quad (42)$$

这是比起恒等式

$$u \oplus v = u' \oplus v'' = u'' \oplus v' \quad (43)$$

更强的命题, 后者由舍入(42)得到。

证明 如果

$$t \equiv x \pmod{b^e}, \quad |t| \leq \frac{1}{2}b^e \quad (44)$$

则我们说 t 是 x 的一个模 b^e 尾部。于是 $x - \text{round}(x)$ 总是 x 的一个尾部。定理 A 的证明大部分依赖于在习题 11 中证明的以下简单事实。

引理 T 如果 t 是浮点数 x 的一个尾部, 则 $x \ominus t = x - t$ 。■

设 $w = u \oplus v$, 当 $w = 0$ 时, 定理 A 显然成立。通过以 b 的适当次幂来乘所有变

量,我们可以不失一般性地假定 $e_w = p$ 。于是 $u + v = w + r$, 其中 r 是 $u + v$ 的一个模 1 尾部。而且 $u' = \text{round}(w - v) \sim \text{round}(u - r) = u - r - t$, 其中 t 是 $u - r$ 的一个模 b^p 尾部, 而且 $e = e_{u'} - p$ 。

如果 $e \leq 0$, 则 $t \equiv u - r \equiv -v \pmod{b^e}$, 因此 t 是 $-v$ 的一个尾部且 $v'' = \text{round}(w - u') = \text{round}(v + t) = v + t$; 这就证明了 (40)。如果 $e > 0$, 则 $|u - r| \geq b^p - \frac{1}{2}$; 而且由于 $|r| \leq \frac{1}{2}$, 我们有 $|u| \geq b^p - 1$ 。由此得出 u 是一个整数, 所以 r 是 v 的一个模 1 尾部。如果 $u' = u$, 则 $t = -r$ 是 $-v$ 的一个尾部。否则关系 $\text{round}(u - r) \neq u$ 意味着 $|u| = b^p - 1$, $|r| = \frac{1}{2}$, $|u'| = b^p$, $t = r$; 再次 t 是 $-v$ 的一个尾部。|

定理 A 揭示了浮点加法的一个规律性, 但它似乎不是一个特别有用的结果。下面的恒等式是更有意义的。

定理 B 在定理 A 和 (41) 的假设之下,

$$u + v = (u \oplus v) + ((u \ominus u') \oplus (v \ominus v'')) \quad (45)$$

证明 事实上, 我们可以证明, $u \ominus u' = u - u'$, $v \ominus v'' = v - v''$, 而且 $(u - u') \oplus (v - v'') = (u \oplus v) + ((u \ominus u') \oplus (v \ominus v''))$, 因此从定理 A 可得出 (45)。利用前边证明中使用的记号, 这些关系分别等价于

$$\text{round}(t + r) = t + r, \quad \text{round}(t) = t, \quad \text{round}(r) = r \quad (46)$$

习题 12 在 $|e_u - e_v| \geq p$ 的特殊情况下证明了定理。否则 $u + v$ 至多有 $2p$ 个有效位, 而且容易看出: $\text{round}(r) = r$ 。现在如果 $e > 0$, 则定理 A 的证明表明 $t = -r$ 或 $t = r = \pm \frac{1}{2}$ 。如果 $e \leq 0$, 我们有 $t + r \equiv u$ 和 $t \equiv -v \pmod{b^e}$; 这足以证明, 假定 $e_u \geq e$ 和 $e_v \geq e$, 则 $t + r$ 和 t 舍入成它们本身。但 $e_u < 0$ 或 $e_v < 0$ 将同我们关于 $|e_u - e_v| < p$ 的假设矛盾, 因为 $e_w = p$ 。|

定理 B 给出了 $u + v$ 与 $u \oplus v$ 之间差的一个显式公式, 并用浮点算术的五个运算可以直接算出的量来表达它。如果进制 b 是 2 或 3, 我们可以对这个结果进行改进, 并只需用两个浮点运算和一个绝对值的(定点)比较即可给出修正项的精确值。

定理 C 如果 $b \leq 3$ 和 $|u| \geq |v|$, 则

$$u + v = (u \oplus v) + (u \ominus (u \oplus v)) \oplus v \quad (47)$$

证明 仍旧沿用上边证明的约定, 我们希望证明: $v \ominus v' = r$ 。但只须证明 $v' = w - u$ 即可, 因为利用 (46) 将得出 $v \ominus v' = \text{round}(v - v') = \text{round}(u + v - w) = \text{round}(r) = r$ 。

事实上, 我们将证明每当 $b \leq 3$ 和 $e_u \geq e_v$ 时, (47) 成立。如果 $e_u \geq p$, 则 r 是 v 的一个模 1 尾部, 因此, 如所希望的那样, $v' = w \ominus u = v \ominus r = v - r = w - u$ 。如果 $e_u < p$, 则我们必然有 $e_u = p - 1$, 而且 $w - u$ 是 b^{-1} 的一个倍数; 因此如果它的量少

于 $b^{p-1} + b^{-1}$, 则它将会舍入为它本身。由于 $b \leq 3$, 我们确实有 $|w - u| \leq |w - u - v| + |v| \leq \frac{1}{2} + (b^{p-1} - b^{-1}) < b^{p-1} + b^{-1}$ 。这就完成了证明。■

定理 A, B 和 C 的证明在 x 恰处于两个浮点数中间时的歧义情况下并不真正地依赖于 $\text{round}(x)$ 的精确定义; 解决歧义性的任何方法都足以保证我们至今已经证明的每件事情的正确性。

没有一种舍入规则对每种应用来说都是最好的。例如, 当计算收入税时, 我们一般地需要一个特殊的规则。但是对于大多数数值计算来说, 最好的办法乃是在算法 4.2.1N 中所确定的舍入方案, 该方案认为, 当舍入的是一个歧义值时, 最低有效数字应弄成总是偶数(或总是奇数)。这不是只有吹毛求疵的人才感兴趣的雕虫小技, 它是一个重要的实际考虑。因为歧义的情况发生得惊人地频繁, 而且一种有偏倚的舍入规则产生相当低劣的结果。例如, 考虑十进算术并假定 5 的余数总是往上舍入, 则如果 $u = 1.0000000$ 和 $v = 0.5555555$, 我们有 $u \oplus v = 1.5555556$; 如果我们用浮点减法从这个结果中减去 v , 我们得到 $u' = 1.0000001$ 。从 u' 加 v 和减 v 给出 1.0000002 , 下次我们得到 1.0000003 , 等等; 尽管我们总在加和减相同的值, 但这一结果保持增长。

这个现象称做漂移, 如果使用基于最低有效数字的奇偶性的稳定的舍入规则, 它就不会出现, 更精确地说:

定理 D $((u \oplus v) \ominus v) \oplus v = (u \oplus v) \ominus v$ 。

例如, 如果 $u = 1.2345679$ 和 $v = -0.23456785$, 我们求得

$$u \oplus v = 1.0000000, \quad (u \oplus v) \ominus v = 1.2345678,$$

$$((u \oplus v) \ominus v) \oplus v = 0.9999995, \quad (((u \oplus v) \ominus v) \oplus v) \ominus v = 1.2345678$$

对于一般的 u 和 v 的证明似乎要求比在上述一些定理中更为详细的情况分析; 见下面的文献。■

定理 D 对于“舍入到偶数”和“舍入到奇数”两者都是正确的; 在这两种可能性之间我们应当如何选择呢? 当进制 b 为奇数时, 除了在进行浮点除法的期间外, 绝不会出现歧义的情况, 而在这种情况下舍入相对地不大重要。对于偶的进制, 有理由优先采用下列规则: “当 $b/2$ 为奇数时舍入到偶数, 当 $b/2$ 为偶数时舍入到奇数。”一个浮点小数的最低有效数字经常成为在随后的计算中要舍去的余数, 而这个规则只要可能就避免在最低有效位生成数字 $b/2$; 它的效果是提供对于歧义舍入的某些信息, 使得随后的舍入将趋于非歧义的。例如, 如果我们在十进系统中舍入成奇数, 则对 2.44445 重复进行舍入的结果是每舍入一次就少去一位, 而导致了序列 2.4445 , 2.445 , 2.45 , 2.5 , 3 ; 如果舍入成偶数就不会出现这种情况, 尽管重复舍入像 2.5454 这样一个数将导致许多错误。[见 Roy A. Keir, *Inf. Proc. Letters* 3 (1975), 188 ~ 189.] 另一方面, 有些人喜欢在所有情况下都舍入成偶数, 使得最低有效数字为 0 的可能性更大。习题 23 说明了舍入成偶数的这一优点。没有一种选择是确定地优于另一种的; 幸而进制通常是 $b=2$ 或 $b=10$, 这时每一个人都同意舍入成偶数是最好

的。

如果读者检验一下上述几个证明的一些细节,将不难发现由简单规则 $u \oplus v = \text{round}(u + v)$ 带来的大量简化。要是我们的浮点加法子程序没能给出这一结果,即便只是在某些少数情况下,这些证明将变得更加复杂化,或许它们将完全失败。

如果用截取运算代替舍入,即若令 $u \oplus v = \text{trunc}(u + v)$ 和 $u \ominus v = \text{trunc}(u - v)$, 其中 $\text{trunc}(x)$ 把所有正实数 x 变成为不大于 x 的最大浮点数,则定理 B 不成立。当 $u + v$ 和 $u \oplus v$ 之间的差不能精确地表达为一个浮点数时,则定理 B 对于像 $(20, +.10000001) \oplus (10, -.10000001) = (20, +.10000000)$ 这样一些情况将出现例外;如果这个差能表达为浮点数,则对于像 $12345678 \oplus .012345678$ 这样的情况也要出现例外。

许多人觉得,由于浮点算术从本质上说就是不精确的,因此在一些个别的情况下使得它仅仅稍不精确也无妨,假如这样做方便的话。这种策略在设计计算机硬件中能节省些钱,或者节约一个子程序的少量的平均运行时间。但是上边的讨论表明这样一种策略是错误的。如果在一些情况下任意采用不正确的舍入,我们可能节省 FADD 子程序,即程序 4.2.1A 的 5% 的运行时间,以及节省它 25% 的空间,但是还是不要这样做为好。原因并不在于崇尚对位数的追求;一个更基本的论点在这里是至关重要的;只要可能,数值子程序提供的结果应当满足那些简单的、有用的数学法则。关键性的公式 $u \oplus v = \text{round}(u + v)$ 是一个“规律性”的性质,它告诉我们,计算算法的数学分析做还是不做,其差别是很大的。如果没有任何基本的对称性质,则证明有趣的结果的工作就变成为极端无聊的。人们对于工具的欣赏乃是取得工作成功的一个必不可少的成分。

B. 非规格化的浮点算术 对所有浮点数进行规格化的方针可以以两种方法来解释。从有利方面看,是试图得到在一给定精度范围内所能达到的极大精确性,但是我们也可以把它当成潜在危险,因为它给出的结果的精确度比实际上应有的精度更加精确。当把 $(1, +.31428571) \ominus (1, +.31415927)$ 的结果规格化为 $(-2, +.12644000)$ 时,我们把结果量不精确性可能更大的信息抑制了。而如果答案保留为 $(1, +.00012644)$ 的话,则这样的信息就保留下来了。

一个问题的输入数据,经常并不如浮点表示所允许的那样精确,例如,阿伏加德罗常数和普朗克常数的值,我们仅知道 8 位有效数字,如果用

$$(27, +.00060221) \quad \text{和} \quad (-23, +.00066261)$$

而不是通过 $(24, +.60221400)$ 和 $(-26, +.66261000)$ 来表示它们,可能更为适当。如果对每个问题都以一种非规格化的形式给出我们的输入数据,并指明假定的精度是多少,而且输出时也指明在答案中的精度是多少,那就太好了。可惜这是一个非常困难的问题,尽管使用非规格化的算术能提供某些信息。例如,我们可以相当肯定地说,阿伏加德罗常数与普朗克常数的积是 $(1, +.00039903)$, 和是 $(27, +.00060221)$ 。(这个例子的目的并不是想说明这些基本常数的和与积有任何重要的物理意义;要点在于,当原始的操作数彼此无关时,就有可能在对于不精确量

进行计算的结果中,保留关于精度的一点信息.)

有关非规格化算术的规则简单地说是:设 l_u 是 $u = (e_u, f_u)$ 的小数部分中前导 0 的个数,因此 l_u 是在小于等于 p 的条件下使 $|f_u| < b^{-l_u}$ 的最大整数。于是加法和减法恰如在算法 4.2.1A 中那样实施,但是去掉了所有左调。乘法和除法如同在算法 4.2.1M 中那样实施,但是要右调或左调答案,使得恰好出现 $\max(l_u, l_v)$ 个前导 0。实际上,这个规则在手工计算中已经使用了好多年。

由此得出,对于非规格化的计算

$$e_{u \oplus v}, e_{u \ominus v} = \max(e_u, e_v) - (0 \text{ 或 } 1) \quad (48)$$

$$e_{u \odot v} = e_u + e_v - q - \min(l_u, l_v) - (0 \text{ 或 } 1) \quad (49)$$

$$e_{u \oslash v} = e_u - e_v + q - l_u + l_v + \max(l_u, l_v) + (0 \text{ 或 } 1) \quad (50)$$

当一个计算的结果为 0 时,即以非规格化的 0 (通常称为“量 0 的阶”)作为答案给出;这表示答案可能不真正为 0,我们只是不知道它的任何有效数字。

误差分析有着和非规格化浮点算术稍微不同的形式,我们定义

$$\delta_u = \frac{1}{2} b^{e_u - q - p} \quad \text{如果 } u = (e_u, f_u) \quad (51)$$

这个量依赖于 u 的表示,而不仅仅依赖于值 $b^{e_u - q} f_u$ 。舍入规则告诉我们

$$|u \oplus v - (u + v)| \leq \delta_{u \oplus v}, \quad |u \ominus v - (u - v)| \leq \delta_{u \ominus v}$$

$$|u \otimes v - (u \times v)| \leq \delta_{u \otimes v}, \quad |u \oslash v - (u/v)| \leq \delta_{u \oslash v}$$

这些不等式既适用于规格化算术,也适用于非规格化算术;两种类型的误差分析之间的主要差别在于每个运算(等式(48)至(50))的结果的指数之定义。

我们已经阐述本节早些时候定义的关系 \prec, \sim, \succ 以及 \approx , 既对规格化数有效和有意义,对非规格化数也亦然。作为使用这些关系的一个例子,我们来证明类似于 (39) 的对于非规格化加法的一个近似结合律:对适当的 ϵ ,

$$(u \oplus v) \oplus w \approx u \oplus (v \oplus w) \quad (\epsilon) \quad (52)$$

我们有

$$\begin{aligned} |(u \oplus v) \oplus w - (u + v + w)| &\leq |(u \ominus v) \oplus w - ((u \ominus v) + w)| + \\ &\quad |u \oplus v - (u + v)| \leq \delta_{(u \oplus v) \oplus w} + \delta_{u \oplus v} \leq \\ &\quad 2\delta_{(u \oplus v) \oplus w} \end{aligned}$$

对于 $|u \oplus (v \oplus w) - (u + v + w)|$ 类似的公式成立。现在,由于 $e_{(u \oplus v) \oplus w} = \max(e_u, e_v, e_w) + (0, 1 \text{ 或 } 2)$, 我们有 $\delta_{(u \oplus v) \oplus w} \leq b^2 \delta_{u \oplus (v \oplus w)}$ 。因此我们发现,当 $\epsilon \geq b^{2-p} + b^{-p}$ 时 (52) 成立;相对于结合律,非规格化加法不像规格化加法那样不规律。

应当强调的是,非规格化算术绝不是万灵药。有若干例子,它给出比现有精确度更高的精确度(例如,大量具有大致相同绝对值的量的加法,或者对于很大的 n , 计算 x^n);但却有更多的例子,表明它有很差的精确度,而规格化算术却确定能产生相当好的结果。为什么“每次一个操作”的直接误差分析方法没有一个是令人满意

的? 一个重要的原因就是操作数通常都不是彼此无关的, 这意味着误差趋向于以非寻常的方式彼此抵消或者加强。例如, 假设 x 近似于 $1/2$, 并有其近似值 $y = x + \delta$, δ 为绝对误差。如果现在希望计算 $x(1-x)$, 则我们可以形成 $y(1-y)$; 如果 $x = \frac{1}{2} + \epsilon$, 则我们发现 $y(1-y) = x(1-x) - 2\epsilon\delta - \delta^2$, 这时误差的绝对值已经减少相当多: 它已经乘以因子 $2\epsilon + \delta$ 。这仅仅是当操作数不是彼此独立时, 不精确量的乘法可以导致一个十分精确结果的一种情况。一个更明显的例子是 $x \odot x$ 的计算, 它可以以完全的精确度求得, 而不管我们开始时从多么坏的 x 的近似值着手。

非规格化算术给予我们的额外信息, 通常可能比在进行一项持久的计算时被它破坏的信息更为重要, 但是(和通常一样), 使用时务必小心谨慎。R. L. Ashenurst 和 N. Metropolis 在 *Computer and Computing, AMM, Slaught Memorial Papers* **10** (1965 年 2 月), 47~59; N. Metropolis 在 *Numer. Math.* **7** (1965), 104~112 以及 R. L. Ashenurst 在 *Error in Digital Computation* **2**, L. B. Rall 编辑 (New York: Wiley, 1965), 3~37 中讨论了适当地使用非规格化算术的一些例子。R. L. Ashenurst 在 *JACM* **11** (1964), 168~187 中给出了对于非规格化形式的输入和输出计算标准数学函数的适当方法。N. Metropolis 在 *IEEE Trans. C-22* (1973), 573~576 中讨论了非规格化算术的一个扩充, 它能记住某些已知是精确的值。

C. 区间算术 对于误差决定问题的另一个方法是所谓的“区间”或“范围”算术, 在此算术之下, 在进行计算期间要保持每个数的上界和下界。于是, 例如, 如果知道 $u_0 \leq u \leq u_1$ 和 $v_0 \leq v \leq v_1$, 我们以区间记号 $u = [u_0, u_1]$, $v = [v_0, v_1]$ 来表示它们。和 $u \oplus v$ 为 $[u_0 \nabla v_0, u_1 \Delta v_1]$, 其中 ∇ 表示“低浮点加法”, 即小于或等于真正的和的最大可表示数, 而 Δ 类似地定义(见习题 4.2.1-13)。而且 $u \ominus v = [u_0 \nabla v_1, u_1 \Delta v_0]$; 还有, 如果 u_0 和 v_0 为正, 我们有 $u \otimes v = [u_0 \nabla v_0, u_1 \Delta v_1]$, $u \oslash v = [u_0 \nabla v_1, u_1 \Delta v_0]$ 。例如, 我们可以表示阿伏加德罗数和普朗克常数为

$$N = [(24, +.60221331), (24, +.60221403)]$$

$$h = [(-26, +.66260715), (-26, +.66260795)]$$

它们的和与积为

$$N \oplus h = [(24, +.60221331), (24, +.60221404)]$$

$$N \otimes h = [(-2, +.39903084), (-2, +.39903181)]$$

当 $v_0 < 0 < v_1$ 时, 如果我们试图以 $[v_0, v_1]$ 来除, 则有除数为 0 的可能性。因为奠定区间算术的原理是提供严格的误差估计, 在这种情况下应当标出除以 0 的错误。然而, 如果像在习题 24 中所讨论的那样引进特殊的约定, 则在区间算术中上溢和下溢不必当做严重的错误。

区间算术大约仅仅比通常的算术长两倍的样子, 而且它提供真正可靠的误差估计。考虑到数学误差分析的困难, 它确实是值得支付的小代价。像上边所说的那样, 由于在一个计算中中间值通常是互相依赖的, 故由区间算术所得到的最后估计势必将是悲观的; 如果我们要处理区间算术, 则通常要重新设计迭代数值方法。然

而,有效利用区间算术的前景看来非常好,因而应努力提高它的可利用性,并使它尽可能地成为对用户友好的。

D. 历史和文献 Jules Tannery 关于十进计算的经典论述, *Lecons d'Arithmétique* (Paris: Colin, 1894) 指出, 如果头一个被抛弃的数字大于等于 5, 正数应该往上舍入; 因为恰有一半的十进数字大于等于 5。他觉得这个规则平均将向上舍入一半次数, 所以它的误差将得到补偿。在歧义的情况下“舍入到偶数”的思想似乎首先是由 James B. Scarborough 在他的先驱性的作品 *Neumerical Mathematical Analysis* 的头一版 (Baltimore: Johns Hopkins Press, 1930), 2 中提到的; 在第 2 版 (1950) 中他进一步阐明了这个观点, 指出任何有头脑的人都应该明白: 当 5 被去掉时, 仅在一半的情况下前一位数字应加 1, 因此他建议舍入成偶数以便达到这一点。

对浮点算术的头一个分析是由 F. L. Bauer 和 K. Samelson 给出的, *Zeitschrift für angewandte Math. und Physik* 4 (1953), 312~316。此后过了 5 年以上才有第二篇作品出现: J. W. Carr III, *CACM* 2, 5 (1959 年 5 月), 10~15。也见 P. C. Fischer, *Proc. ACM Nat. Meeting* 13 (1958), 论文 39。由 J. H. Wilkinson 著的 *Rounding Errors in Algebraic Processes* (Englewood Cliffs: Prentice-Hall, 1963), 说明了怎样把个别算术运算的误差分析应用到大型问题上去。也见他在 *The Algebraic Eigenvalue Problem* (Oxford: Clarendon Press, 1965) 上的论文。

有两篇重要的文章可以特别推荐来供进一步研究: W. M. Kahan, *Proc. IFIP Congress* (1971), 2, 1214~1239; *IEEE Trans. C-22* (1973), 601~607。这两篇论文都包括有用的理论并且指出这一理论能用于实践。

这一小节中引进的关系 $\prec, \sim, \succ, \approx$ 类似于由 A. van Wijngaarden 在 *BIT* 6 (1966), 66~81 中所发表的一些思想。上边的定理 A 和定理 B 受到 Ole Møller 的某些有关工作的启示, 见 *BIT* 5 (1965), 37~50, 251~255; 定理 C 是由 T. J. Dekker 给出的, *Numer. Math.* 18 (1971), 224~242。S. Linnainmaa 发表了关于这三个定理的推广和改进, 见 *BIT* 14 (1974), 167~202。W. M. Kahan 在某些未发表的笔记中给出定理 D。关于完备的证明及进一步的评述, 见 J. F. Reiser 和 D. E. Knuth, *Inf. Proc. Letters* 3 (1975), 84~87, 164。

非规格化的算术是由 F. L. Bauer 和 K. Samelson 在上边引证的论文中推荐的, 而且它为 J. W. Carr III 于 1953 年在密执安大学单独使用过。若干年后, MANIAC III 计算机的设计中, 将两种类型的算术都包括在了硬件中; 见 R. L. Ashenurst 和 N. Metropolis, *JACM* 6 (1959), 415~428; *IEEE Trans. EC-12* (1963), 896~901; R. L. Ashenurst, *Proc. Spring Joint Computer Conf.* 21 (1962), 195~202。早期关于非规格化算术进一步的讨论也见 H. L. Gray 和 C. Harrison, Jr., *Proc. Eastern Joint Computer Conf.* 16 (1959), 244~248, 以及 W. G. Wadey, *JACM* 7 (1960), 129~139。

关于区间算术的早期发展, 以及某些变型, 见 A. Gibb, *CACM* 4 (1961), 319~320; B. A. Chartres, *JACM* 13 (1966), 386~403; 以及 Ramon E. Moore, *Interval*

Analysis (Prentice-Hall, 1966)。Moore 后来的书, *Methods and Applications of Interval Analysis* (SIAM, 1979) 描述了关于这个课题后续的成果。

20 世纪 80 年代初在卡尔斯鲁赫(Karlsruhe)大学研制了对 Pascal 语言的一个扩充, 它允许变量有“区间”类型。这个语言还包括了科学计算的许多其它特征。关于该语言的描述, 请见 Bohlender, Ullrich, Wolff von Gudenberg 及 Rall, *Pascal-SC* (Academic Press, 1987)。

Ulrich Kulisch 所著的书 *Grundlagen des numerischen Rechnens: Mathematische Begründung der Rechnerarithmetik* (Mannheim: Bibl. Inst., 1976) 完全致力于浮点算术系统的研究。也可见 Kulisch 在 *IEEE Trans. C-26* (1977), 610~621 上的一篇论文, 以及他最近和 W. L. Miranker 写的书 *Computer Arithmetic in Theory and Practice* (New York: Academic Press, 1981)。

关于浮点误差分析最新工作的杰出概述, 请见 N. J. Higham, *Accuracy and Stability of Numerical Algorithms* (Philadelphia: SIAM, 1996)。

习 题

注: 以下习题除有相反的说明外, 都假定采取规格化的浮点算术。

1. [M18] 证明恒等式(7)是(2)到(6)的一个推论。

2. [M20] 利用恒等式(2)到(8)证明, 当 $x \geq 0$ 和 $y \geq 0$ 时, $(u \oplus x) \oplus (v \oplus y) \geq u \oplus v$ 。

3. [M20] 找出 8 位数字的浮点十进数 u, v 和 w , 使得

$$u \otimes (v \otimes w) \neq (u \otimes v) \otimes w$$

且在这些计算中, 无指数上溢和下溢出现。

4. [10] 是否可能有浮点数 u, v 和 w , 对于它们, 在进行 $u \otimes (v \otimes w)$ 的计算中有指数上溢出现, 但在进行 $(u \otimes v) \otimes w$ 的计算中则不出现。

5. [M20] 对于所有不出现指数上溢和下溢的浮点数 u 和 $v \neq 0$ 说来, $u \oslash v = u \otimes (1 \oslash v)$ 是否一个恒等式?

6. [M22] 对于所有浮点数 u 来说, 下列两个恒等式之一是否成立? (a) $0 \ominus (0 \ominus u) = u$; (b) $1 \oslash (1 \oslash u) = u$ 。

7. [M21] 令 u^{\otimes} 代表 $u \otimes u$, 求浮点二进制数 u 和 v 使得 $(u + v)^{\otimes} > 2(u^{\otimes} + v^{\otimes})$ 。

► 8. [20] 设 $\epsilon = 0.0001$; 对于下列的进制 10, 余 0, 8 位数字浮点数的数对, 关系式 $u \prec v(\epsilon)$, $u \sim v(\epsilon)$, $u \succ v(\epsilon)$, $u \approx v(\epsilon)$ 中哪一个成立?

a) $u = (1, +.31415927)$, $v = (1, +.31416000)$;

b) $u = (0, +.99997000)$, $v = (1, +.10000039)$;

c) $u = (24, +.60221400)$, $v = (27, +.00060221)$;

d) $u = (24, +.60221400)$, $v = (31, +.00000006)$;

e) $u = (24, +.60221400)$, $v = (28, +.00000000)$ 。

9. [M22] 证明(33)并说明为什么这个结论不能被加强成为 $u \approx w(\epsilon_1 + \epsilon_2)$?

► 10. [M25] (W. M. Kahan) 某台计算机实施浮点算术, 而无适当的舍入, 而且事实上它的浮点乘法程序忽略了 $2p$ 位数字的乘积 $f_u f_v$ 中除 p 位最高位外的所有位(于是, 当 $f_u f_v < 1/b$ 时, $u \otimes v$ 的最低位数字由于随后进行的规格化而总是成为 0)。证明这将导致乘法单调性的破坏; 换言之,

示出正的规格化浮点数 u, v, w 使得在这台机器上有 $u < v$, 但 $u \odot w > v \odot w$

11. [M20] 证明引理 T。

12. [M24] 当 $|e_u - e_v| \geq p$ 时, 给出定理 B 和 (46) 的证明。

► 13. [M25] 某些程序设计语言(甚至某些计算机)仅仅利用浮点算术, 而没有提供整数的精确计算的设备。如果希望对整数的操作, 我们当然可以把一个整数表示为一个浮点数; 而且当浮点操作满足 (9) 中的基本定义时, 我们知道, 假定操作数和答案都恰能以 p 位有效数字精确表示, 则所有浮点运算都将是精确的。因此只要知道数不太大, 我们就能进行整数的加、减或乘, 而没有舍入误差引起的不精确性。

但假设一个程序员想要确定, 当 m 和 $n \neq 0$ 都是整数时, m 是否 n 的一个整倍数。并进一步假设, 如同在习题 4.2.1-15 中那样, 有一个子程序, 对于任何给定的浮点数 u , 可以用它来计算量 $\text{round}(u \bmod 1) = u \pmod{1}$ 。确定 m 是否 n 的倍数的一个好方法是利用假定的子程序, 测试是否 $(m \oslash n) \pmod{1} = 0$; 但是或许在某些情况下, 浮点计算中的舍入误差将使这个测试无效。

试寻找关于整数值 $n \neq 0$ 和 m 的范围的适当条件, 使得 m 是 n 的倍数的充分必要条件是 $(m \oslash n) \pmod{1} = 0$ 。换句话说, 证明如果 m 和 n 不太大, 则这个测试是正确的。

14. [M27] 当使用非规格化的乘法时, 试找出一个适当的 ϵ 使得 $(u \odot v) \otimes w \approx u \otimes (v \odot w)$ (ϵ)。(这推广了 (39), 因为当输入操作数 u, v 和 w 都已规格化时, 非规格化乘法恰和规格化乘法相同。)

► 15. [M24] (H. Bjork) 如果计算出一个区间的中点, 它是否总处于两个端点之间? (换言之, $u \leq v$ 是否意味着 $u \leq (u \oplus v) \oslash 2 \leq v$?)

16. [M28] (a) 若 $n = 10^6$ 且对于所有 $k, x_k = 1.1111111$, 则在 8 位数字的浮点十进算术下, $(\cdots((x_1 \ominus x_2) \ominus x_3) \oplus \cdots \oplus x_n)$ 等于多少? (b) 当等式 (14) 用来计算这些特殊值 x_k 的标准差时, 将发生什么情况? 当使用等式 (15) 和 (16) 时又发生什么情况? (c) 证明对于所有的 x_1, \cdots, x_k 的选择, 在 (16) 中 $S_k \geq 0$ 。

17. [28] 写出一个 MIX 子程序 FCMP, 它比较单元 ACC 中的浮点数 u 和寄存器 A 中的浮点数 v , 并且按照 $u < v, u = v$ 或 $u > v$ (ϵ), 把比较指示器置成 LESS, EQUAL 或 GREATER; 这里 ϵ 存在单元 EPSILON 中作为一个非负定点量, 并假定小数点在该字的左边。假定输入是规格化的。

18. [M40] 在非规格化算术中, 是否有一个适当的数 ϵ 使得 $u \otimes (v \oplus w) \approx (u \otimes v) \oplus (u \otimes w)$ (ϵ)?

► 19. [M30] (W. M. Kahan) 考虑对 x_1, \cdots, x_n 进行浮点加法的下列过程:

$$s_0 = c_0 = 0;$$

$$y_k = x_k \ominus c_{k-1}, \quad s_k = s_{k-1} \oplus y_k, \quad c_k = (s_k \ominus s_{k-1}) \ominus y_k, \quad \text{对于 } 1 \leq k \leq n.$$

设在这些运算中的相对误差由等式

$$y_k = (x_k - c_{k-1})(1 + \eta_k), \quad s_k = (s_{k-1} + y_k)(1 + \sigma_k)$$

$$c_k = ((s_k - s_{k-1})(1 + \gamma_k) - y_k)(1 + \delta_k)$$

定义, 其中 $|\eta_k|, |\sigma_k|, |\gamma_k|, |\delta_k| \leq \epsilon$, 证明 $s_n = \sum_{k=1}^n (1 + \theta_k)x_k$, 其中 $|\theta_k| \leq 2\epsilon + O(n\epsilon^2)$ 。[定理 C 指出: 如果 $b-2$ 和 $|s_{k-1}| \geq |y_k|$, 则我们恰有 $s_{k-1} + y_k = s_k - c_k$ 。但在本题中我们要得到一个估计, 它只假定每个运算有有界的相对误差, 而甚至当浮点运算没有仔细地进行舍入时, 这个估计仍然正确。]

20. [25] (S. Linnainmaa) 试寻找使得 $|u| \geq |v|$ 且 (47) 不成立的所有 u, v 。

21. [M35] (T. J. Dekker) 定理 C 指出怎样进行浮点二进数的精确加法。试说明如何进行精

确的乘法:以 $w \cdot w'$ 的形式表达乘积 uv , 其中 w 和 w' 是从两个给定的浮点二进制数 u 和 v , 仅仅使用运算 \oplus 、 \ominus 和 \otimes 计算出来的。

22. [M30] 在浮点乘/除中能出现漂移吗? 给定 u 和 $v \neq 0$, 考虑序列 $x_0 = u, x_{2n+1} = x_{2n} \otimes v, x_{2n+2} = x_{2n+1} \oslash v$; 使得 $x_k \neq x_{k+2}$ 成为可能的最大下标 k 是什么?

► 23. [M26] 证明或否定: 对所有浮点数 u , $u \ominus (u \text{ mod } 1) = \lfloor u \rfloor$ 。

24. [M27] 考虑所有区间 $[u_l, u_r]$ 的集合, 其中 u_l 和 u_r 或者是非零浮点数或者是特殊符号 $0, -0, +\infty, -\infty$; 每个区间必须有 $u_l \leq u_r$, 而且仅当 u_l 是有限的和非零的时, 才允许 $u_l = u_r$ 。区间 $[u_l, u_r]$ 代表使得 $u_l \leq x \leq u_r$ 的所有浮点 x , 其中我们认为对于所有正的 x ,

$$-\infty < x < -0 < +0 < +x < +\infty$$

(于是, $[1, 2]$ 意味着 $1 \leq x \leq 2$; $[+0, 1]$ 意味着 $0 < x \leq 1$; $[-0, 1]$ 意味着 $0 \leq x \leq 1$; $[0, +0]$ 表示单个值 0; 而 $[-\infty, +\infty]$ 则代表每个数。)说明怎样对所有这样的区间定义适当的算术运算, 而无须依靠“上溢”或“下溢”或其它异常的指示, 除非以包括 0 的区间做除数时。

► 25. [15] 当人们谈及浮点算术的不正确性时, 他们通常把误差归结为在进行近乎相等的量的减法期间出现的“抵消”。而当 u 和 v 近似相等时, 差 $u \ominus v$ 被精确地得到而没有误差。这些人们实际上想说的是什么呢?

26. [M21] 给定正的浮点数 u, u', v 和 v' 而且 $u \sim u'(\epsilon)$ 和 $v \sim v'(\epsilon)$ 并假定采用规格化的算术。试证明有一个小的 ϵ' 使得 $u \ominus v \sim u' \ominus v'(\epsilon')$ 。

27. [M27] (W. M. Kahan) 对于所有 $u \neq 0$, 证明 $1 \oslash (1 \oslash (1 \oslash u)) - 1 \oslash u$ 。

28. [HM30] (H. G. Diamond) 假设 $f(x)$ 是在某个区间 $[x_0, x_1]$ 上严格地递增的函数, 且设 $g(x)$ 为逆函数(例如, f 和 g 可以是“exp”和“ln”或“tan”和“arctan”)。如果 x 是一个浮点数, 使得 $x_0 \leq x \leq x_1$, 设 $f(x) = \text{round}(f(x))$, 而且如果 y 是一个浮点数, 使得 $f(x_0) \leq y \leq f(x_1)$, 令 $\hat{g}(y) = \text{round}(g(y))$; 其次令 $h(x) = \hat{g}(\hat{f}(x))$, 条件是当它有定义时。尽管 $h(x)$ 由于舍入不总是等于 x , 我们还是期望 $h(x)$ 很“接近于” x 。

证明如果精度 b^p 至少是 3, 而且如果 f 严格凹或严格凸(即对于所有在 $[x_0, x_1]$ 中的 x , $f''(x)$ 有相同符号, 则在对于所有 x ,

$$h(h(h(x))) = h(h(x))$$

且等式两边都有定义的意义下, h 的重复应用将是稳定的。换句话说, 如果适当地执行这些子程序, 将没有“漂移”存在。

► 29. [M25] 给出一个例子说明在上道题中 $b^p \geq 3$ 的条件是必要的。

► 30. [M30] (W. M. Kahan) 设对于 $x < 1$, $f(x) = 1 + x + \cdots + x^{106} = (1 - x^{107})/(1 - x)$, 且设对于 $0 < y < 1$, $g(y) = f((\frac{1}{3} - y^2)(3 + 3.45y^2))$ 。对 $y = 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$, 在一台或多台“袖珍计算器”上计算 $g(y)$, 并解释你在结果中得到的所有不精确性(因为今天的大多数计算器都不正确地舍入, 因此这些结果通常是令人惊异的。注意 $g(\epsilon) = 107 - 10491.35\epsilon^2 + 659749.9625\epsilon^4 - 30141386.26625\epsilon^6 + O(\epsilon^8)$)。

31. [M25] (U. Kulisch) 当对于 $x = 408855776$ 和 $y = 708158977$, 利用标准的 53 个二进位的双精度浮点算术计算多项式 $2y^2 + 9x^4 - y^4$ 时, 结果是 $\approx -3.7 \times 10^{19}$ 。以另一个形式 $2y^2 + (3x^2 - y^2)(3x^2 + y^2)$ 计算它给出 $\approx 1.0 \times 10^{18}$ 。然而, 真正的结果是 1.0(精确地)。试说明如何构造数值不稳定性的类似的例子。

*4.2.3 双精度计算

到现在为止,我们考虑了“单精度”浮点算术,这实质上意味着,我们所处理的浮点值均可存入单个的机器字中。对于一个给定的应用,当单精度浮点算术不产生足够的精确度时,通过适当的程序设计技巧还可提高精度,这些技巧使用两个或多个存储字来表示每一个数。

尽管我们将在4.3节讨论高精度计算的一般问题,但在这里对于双精度给以单独的讨论仍然是适当的。应用于双倍精度的特殊技术,对于更高精度说来,相对地是不适当的;而且双精度本身也是一个相当重要的课题,因为它是越过单精度之后的第一步,而且可应用于不要求极高精度的许多问题中。



是的,当作者在20世纪60年代撰写本书的头一版时,上一段话是正确的。但是计算机已经以这样一种方式发展起来,使得双精度浮点运算的旧的动机大都已经消失;因此,本小节主要是出于历史的兴趣。在计划中的本书的第4版中,4.2.1小节将改称“规格化计算”,而本节4.2.3小节将被关于“例外的数”的讨论所代替。新的内容将专注于ANSI/IEEE标准754的一些特殊方面:非规范的数,以及表示无穷、无定义或者不然就是非寻常的量的所谓NaNs。(请参阅4.2.1小节末尾的参考文献。)同时,让我们对于较老的思想做最后的回顾,以便了解它们仍可以为我们提供什么教训。

浮点算术,但不是定点算术,几乎总要求双精度计算,也许统计工作除外,在那里,普遍使用定点双精度算术计算平方和与叉积;由于双精度算术的定点形式比浮点形式要更简单些,因此在这里仅限于讨论浮点形式。

人们经常需要采用双精度,这不仅用来扩充浮点数小数部分的精度,而且用来增加指数部分的范围。因此,我们将在这小节讨论在MIX计算机中双精度浮点数的如下双字格式:

$$\begin{array}{|c|c|c|c|c|c|} \hline \pm & e & e & f & f & f \\ \hline \end{array} \quad \begin{array}{|c|c|c|c|c|c|} \hline & f & f & f & f & f \\ \hline \end{array} \quad (1)$$

这里两个字节用做指数,八个字节用做小数。指数是“余 $b^2/2$ ”的,其中 b 是字节大小。正负号将出现在最高位的字中;完全忽略掉另一个字的符号是适宜的。

关于双精度算术的讨论将是十分面向机器的,因为只有研究这些子程序编码时涉及的问题,我们才能适当地评价这个课题。因此仔细地研究一下MIX程序,对于理解这一内容是必不可少的。

在这一节中我们将离开在前两小节中提出的精确性的理想目标;我们的双倍精度于程序将不舍入它们的结果,有时允许掺进一点误差。用户不敢太信任这些程序。在单精度的情况下,我们有充足的理由排除丢失精度的每个可能性,但现在我们面对着一一种不同的形势:(a)为保证在所有情况下都采用双精度舍入,其额外的程序工作量是可观的;完全精确的子程序将花费比如说两倍的空间以及另外一半的时间。使得我们的单精度程序完善是比较容易的,但双精度使我们直接面对机器的限

制。相对于其它浮点子程序,情况与此类似;我们不能预期余弦子程序对于所有的 x 都精确地计算 $\text{round}(\cos x)$,因为这实际上是不可能的。代替的是,对所有合理的 x 值,余弦子程序能以合理的速度提供它所能达到的最好相对误差。当然,只要可能子程序的设计者应当力图使所计算的函数尽量满足简单的数学定律——例如,

$$\cos(-x) = \cos x; |\cos x| \leq 1; \text{ 对 } 0 \leq x \leq y < \pi, \cos x \geq \cos y$$

(b)单精度算术是我们的“主食”,每个要使用浮点算术的人都必须使用它,但双精度通常用于此种干净利落的结果并不重要的情形。7位和8位精度之间的区别可能是值得注意的,但我们很少注意15位和16位精度之间的差别。在计算单精度的结果期间,双精度最常用于中间步骤;其全部潜力并非必需的。(c)分析这些过程以便看看它们可能不精确到什么程度,对我们将是有益的,因为它们一般地代表在坏的单精度子程序中所采取的截断的类型(见习题7和8)。

现在我们从上述观点来考虑加法和减法运算。当然,通过改变第二个操作数的符号,减法可转换为加法。分别地把低有效位部分和高有效位部分加到一起,适当地传送“进位”,就可实现加法。

然而,出现了一个困难,因为我们做的是“带符号的量”的算术:有可能在把两个低有效位部分相加时得到错误的符号(即当操作数的符号相反而且较小操作数的低有效位部分大于较大操作数的低有效位部分时)。最简单的解法是预先采取正确符号;所以在算法4.2.1A的步骤A2中,我们不仅假定 $e_u \geq e_v$,还假定 $|u| \geq |v|$ 。这意味着我们可以确信,最后的符号将是 u 的符号。在其它方面,双精度加法很像单精度加法,只是每件事需要做两次而已。

程序A(双精度加法)子程序DFADD把形式为(1)的一个双精度浮点数 v 加到一个双精度浮点数 u 上。假定 v 开始时在 rAX 中(即寄存器A和X),而 u 开始时存放在单元 ACC 和 ACCX 中,答案出现在 rAX 和 $(\text{ACC}, \text{ACCX})$ 中。子程序DFSUB在同样的约定下从 u 减去 v 。

假定两个输入操作数是规格化的,而且答案也被规格化。这个程序的最后部分是一个双精度规格化过程,它也被这一节的其它子程序所使用。习题5说明怎样对这个程序进行重大改进。

01	ABS	EQU	1:5	绝对值的字段定义
02	SIGN	EQU	0:0	符号的字段定义
03	EXPD	EQU	1:2	双精度指数字段
04	DFSUB	STA	TEMP	双精度减法:
05		LDAN	TEMP	改变 v 的符号
06	DFADD	STJ	EXITDF	双精度加法:
07		CMPA	ACC(ABS)	比较 $ u $ 和 $ v $
08		JG	1F	
09		JL	2F	
10		CMPX	ACCX(ABS)	

11		JLE	2F	
12	1H	STA	ARG	如果 $ v > u $, 交换 $u \leftrightarrow v$
13		STX	ARGX	
14		LDA	ACC	
15		LDX	ACCX	
16		ENT1	ACC	(ACC 和 ACCX 处于相邻的单元中)
17		MOVE	ARG(2)	
18	2H	STA	TEMP	
19		LD1N	TEMP(EXPD)	$r11 \leftarrow -e_v$
20		LD2	ACC(EXPD)	$r12 \leftarrow e_u$
21		INC1	0,2	$r11 \leftarrow e_u - e_v$
22		SLAX	2	去掉指数
23		SRAX	1,1	右调
24		STA	ARG	$0 \ v_1 \ v_2 \ v_3 \ v_4$
25		STX	ARGX	$v_5 \ v_6 \ v_7 \ v_8 \ v_9$
26		STA	ARGX(SIGN)	在两半都存上真正的符号
27		LDA	ACC	(我们知道 u 有答案的符号)
28		LDX	ACCX	$rAX \leftarrow u$
29		SLAX	2	去掉指数
30		STA	ACC	$u_1 \ u_2 \ u_3 \ u_4 \ u_5$
31		SLAX	4	
32		ENTX	1	
33		STX	EXPO	$EXPO \leftarrow 1$ (见下)
34		SRC	1	$1 \ u_5 \ u_6 \ u_7 \ u_8$
35		STA	1F(SIGN)	一个技巧,见正文中注释
36		ADD	ARGX(0:4)	加 $0 \ v_5 \ v_6 \ v_7 \ v_8$
37		SRAX	4	
38	1H	DECA	1	从插入的 1 恢复(符号改变)
39		ADD	ACC(0:4)	把两个高有效位的部分相加
40		ADD	ARG	(上溢不可能出现)
41	DNORM	JANZ	1F	规格化子程序:
42		JXNZ	1F	f_w 在 rAX 中, $e_w = EXPO + r12$
43	DZERO	STA	ACC	如果 $f_w = 0$, 则置 $e_w \leftarrow 0$
44		JMP	9F	
45	2H	SLAX	1	向左规格化
46		DEC2	1	

47	1H	CMPA	= 0 = (1:1)	前导字节为 0 吗?
48		JE	2B	
49		SRAZ	2	(舍入省略)
50		STA	ACC	
51		LDA	EXPO	计算最后的指数
52		INCA	0,2	
53		JAN	EXPUND	它是负的吗?
54		STA	ACC(EXPD)	
55		CMPA	= 1(3:3) =	是否多于两个字节?
56		JL	8F	
57	EXPOVD	HLT	20	
58	EXPUND	HLT	10	
59	8H	LDA	ACC	把答案送入 rA
60	9H	STX	ACCX	
61	EXITDF	JMP	*	子程序出口
62	ARG	CON	0	
63	ARGX	CON	0	
64	ACC	CON	0	浮点累加器
65	ACCX	CON	0	
66	EXPO	CON	0	“未加工的指数”部分

当在这个程序中把两个低位部分加到一起时,一个额外的数字“1”插入到已知有正确符号的那个字的左边。做完加法后,依照不同情况这个字节可以是 0,1 或 2,以这种方式同时处理所有三种情况(试把这个方法同用于程序 4.2.1A 中的颇为麻烦的求补方法做一比较)。

值得指出的是,在执行了行 40 的指令之后,寄存器 A 可为 0;而且,由于当结果为 0 时,MIX 定义零结果符号的方式,累加器包含着当寄存器 X 非 0 时应附加给结果的正确符号。如果交换行 39 和行 40,则程序将是不正确的,尽管两条指令都是“ADD”!

现在来考虑双精度的乘法。乘积有四个部分,如图 4 所示。由于我们仅需要最左边的 8 个字节,所以,仅仅处理图解中垂直线左边的数字是适当的。特别是,这意味着我们甚至不需要计算两个低位部分的乘积。

程序 M(双精度乘法) 这个程序的输入和输出的约定,与程序 A 相同。

01	BYTE	EQU	1(4:4)	字节大小
02	QQ	EQU	BYTE * BYTE/2	双精度指数的余量
03	DFMUL	STJ	EXITDF	双精度乘法:
04		STA	TEMP	
05		SLAX	2	去掉指数

06	STA	ARG	v_m
07	STX	ARGX	v_l
08	LDA	TEMP(EXPD)	
09	ADD	ACC(EXPD)	
10	STA	EXPO	$EXPO \leftarrow e_u + e_v$
11	ENT2	-QQ	$r12 \leftarrow -QQ$
12	LDA	ACC	
13	LDX	ACCX	
14	SLAX	2	去掉指数
15	STA	ACC	u_m
16	STX	ACCX	u_l
17	MUL	ARGX	$u_m \times v_l$
18	STA	TEMP	
19	LDA	ARG(ABS)	
20	MUL	ACCX(ABS)	$ v_m \times u_l $
21	SRA	1	0 x x x x
22	ADD	TEMP(1:4)	(上溢不可能出现)
23	STA	TEMP	
24	LDA	ARG	
25	MUL	ACC	$v_m \times u_m$
26	STA	TEMP(SIGN)	把结果的真正符号存起来
27	STA	ACC	现在准备把各部分乘积加在一起
28	STX	ACCX	
29	LDA	ACCX(0:4)	0 x x x x
30	ADD	TEMP	(不可能上溢)
31	SRAX	4	
32	ADD	ACC	(上溢不可能出现)
33	JMP	DNORM	规格化并出口

		$u \ u \ u \ u \ u$	$u \ u \ u \ 0 \ 0$	$= u_m + \epsilon u_l$
		$v \ v \ v \ v \ v$	$v \ v \ v \ 0 \ 0$	$= v_m + \epsilon v_l$
		$x \ x \ x \ x \ x$	$x \ 0 \ 0 \ 0 \ 0$	$= \epsilon^2 u_l \times v_l$
	$x \ x \ x \ x \ x$	$x \ x \ x \ 0 \ 0$		$= \epsilon u_m \times v_l$
	$x \ x \ x \ x \ x$	$x \ x \ x \ 0 \ 0$		$= \epsilon u_l \times v_m$
$x \ x \ x \ x \ x$	$x \ x \ x \ x \ x$			$= u_m \times v_m$
$w \ w \ w \ w \ w$	$w \ w \ w \ w \ w$	w	$w \ w \ w \ w \ w$	$w \ 0 \ 0 \ 0 \ 0$

图4 8字节小数部分的双精度乘法

注意,在这个程序中小心地处理了符号,还要注意,指数的范围使它不可能利用一个变址寄存器来计算最后的指数。程序 M 也许在精确度方面稍微马虎了点,因为它仅仅使用图 4 中垂线左边的信息,而这会使得最低位字节有多至 2 的误差。按照习题 4 中的讨论,可以达到更高的精确度。

双精度浮点除法是最困难的子程序,至少是在这一章中到现在为止我们所遇到的最棘手的情况。但实际上,一旦我们看到怎么来做它,就知道它并不复杂得吓人;现在把要进行除法的数写成形式 $(u_m + \epsilon u_l)/(v_m + \epsilon v_l)$, 其中 ϵ 是计算机字长的倒数,而且假定 v_m 已规格化。这个分数现在可以展开如下:

$$\frac{u_m + \epsilon u_l}{v_m + \epsilon v_l} = \frac{u_m + \epsilon u_l}{v_m} \left(\frac{1}{1 + \epsilon(v_l/v_m)} \right) = \frac{u_m + \epsilon u_l}{v_m} \left(1 - \epsilon \left(\frac{v_l}{v_m} \right) + \epsilon^2 \left(\frac{v_l}{v_m} \right)^2 - \dots \right) \quad (2)$$

由于 $0 \leq |v_l| < 1$ 且 $1/b \leq |v_m| < 1$, 我们有 $|v_l/v_m| < b$, 而且删去含有 ϵ^2 的项所产生的误差可被忽略。因此我们的方法是计算 $w_m + \epsilon w_l = (u_m + \epsilon u_l)/v_m$, 然后从结果中减去 ϵ 乘 $w_m v_l/v_m$ 。

在下列程序中,行 27~32 做双精度加法的较低位的一半,它采用的技巧不同于程序 A,而是用另一种方法来保证得到正确的符号。

程序 D(双精度除法) 这一程序采用与程序 A 和 M 相同的约定。

01	DEFDIV	STJ	EXITDF	双精度除法:
02		JOV	OFL0	确保上溢标记清除
03		STA	TEMP	
04		SLAX	2	去掉指数
05		STA	ARG	v_m
06		STX	ARGX	v_l
07		LDA	ACC(EXPD)	
08		SUB	TEMP(EXPD)	
09		STA	EXPO	$EXPO \leftarrow e_u - e_v$
10		ENT2	QQ + 1	$r12 \leftarrow QQ + 1$
11		LDA	ACC	
12		LDX	ACCX	
13		SLAX	2	去掉指数
14		SRAX	1	(参考算法 4.2.1M)
15		DIV	ARG	如果上溢,则下面会被发现
16		STA	ACC	w_m
17		SLAX	5	余数用于进一步的除法中
18		DIV	ARG	

19	STA	ACCX	$\pm w_l$
20	LDA	ARGX(1:4)	
21	ENTX	0	
22	DIV	ARG(ABS)	$rA \leftarrow \lfloor b^4 v_l / v_m \rfloor / b^5$
23	JOV	DVZROD	除法引起溢出吗?
24	MUL	ACC(ABS)	近似地 $rAX \leftarrow \lfloor w_m v_l / b v_m \rfloor$
25	SRAX	4	乘以 b , 并保存前导字节于
26	SLC	5	rX 中
27	SUB	ACCX(ABS)	减 $ w_l $
28	DECA	1	加负号
29	SUB	WM1	
30	JOV	$* + 2$	如无上溢, 则再进到高位那
31	INCX	1	一半
32	SLC	5	(现在 $rA \leq 0$)
33	ADD	ACC(ABS)	$rA \leftarrow \lfloor w_m \rfloor - \lfloor rA \rfloor$
34	STA	ACC(ABS)	(现在 $rA \geq 0$)
35	LDA	ACC	$rA \leftarrow w_m$ 连同正确符号
36	JMP	DNORM	规格化并出口
37	DVZROD	HLT	30
38	1H	EQU	1(1:1)
39	WM1	CON	1B - 1, BYTE - 1(1:1) 字大小减 1

下边是这些双精度子程序同相应的 4.2.1 节中出现的单精度子程序近似平均计算时间的对比。

	单精度	双精度
加法	45.5 μ	84 μ
减法	49.5 μ	88 μ
乘法	48 μ	109 μ
除法	52 μ	126.5 μ

关于把这节的方法推广到三精度浮点小数部分, 见 Y. Ikebe, CACM 8(1965), 175~177。

习 题

1. [16] 当 180000 除以 314159 时, 对于 $\epsilon = \frac{1}{1000}$, 试用手算来实施双精度除法的技术。(于是, 设 $(u_m, u_l) = (.180, .000)$ 和 $(v_m, v_l) = (.314, .159)$, 并用正文中(2)下边一段提出的方法来求商。)

2.[20] 在程序 M 的行 30 和 31 之间,插入指令“ENTX 0”,以便防止留在寄存器 X 中的不需要的信息干扰结果的精确度,这是一个好想法吗?

3.[M20] 说明为什么在执行程序 M 期间不会出现上溢?

4.[22] 应该怎样改变程序 M 以实现更大精确性?这实质上是把图 4 中的垂线右移一位。详述所要求的所有变动,并确定由于这些变动所引起的执行时间上的差别。

►5.[24] 应该怎样改变程序 A 以实现更大精确性?这实质上是利用小数点右边 9 个字节的累加器而不是 8 个字节的累加器进行处理。详述所要求的所有变动,并确定由于这些变动所引起的执行时间上的差别。

6.[23] 假定这小节的双精度子程序和 4.2.1 小节的单精度子程序被用于同一主程序中。试写出一个子程序,把单精度浮点数转换为双精度形式(1),并写出另一个子程序,它把双精度浮点数转换为单精度形式(如果转换不可能时,它报告指数上溢或下溢)。

►7.[M30] 试通过如下的办法来估计这小节中的双精度子程序的精确度,即通过找出相对误差

$$\begin{aligned} &|((u \oplus v) - (u + v))/(u + v)|, \quad |((u \otimes v) - (u \times v))/(u \times v)| \\ &|((u \oslash v) - (u/v))/(u/v)| \end{aligned}$$

的界 δ_1, δ_2 和 δ_3 来做到这一点。

8.[M28] 在习题 7 的意义下,估计习题 4 和习题 5 的“改进的”双精度子程序的精确度。

9.[M42] T.J.Dekker[Numer. Math. 18 (1971), 224~242]提出了实现双精度的另一种方法,它完全以单精度浮点的二进算法为基础。例如,定理 4.2.2C 指出,如果 $|u| \geq |v|$ 和进制为 2,则 $u + v = w + r$, 其中 $w = u \oplus v$ 和 $r = (u \ominus w) \oplus v$; 这里 $|r| \leq |w|/2^p$, 所以数偶 (w, r) 可当做 $u + v$ 的一个双精度的变形。为了把两个这样的数偶加起来,即 $(u, u') \oplus (v, v')$, 其中 $|u'| \leq |u|/2^p$ 和 $|v'| \leq |v|/2^p$ 且 $|u| \geq |v|$, Dekker 建议(精确地)计算 $u + v = w + r$, 然后计算 $s = (r \oplus v') \oplus u'$ (一个近似的剩余), 最后得到 $(w \oplus s, (w \ominus (w \oplus s)) \oplus s)$ 。

当这个方法递归地用于产生四倍精度的计算时,研究它的精确度和有效性。

4.2.4 浮点数的分布

为了分析浮点算术算法的平均特性(特别是确定它们的平均运行时间),我们需要某些统计信息,以便确定各种情况出现的频繁程度。这一小节的目的是讨论浮点数分布的经验的和理论的性质。

A. 加法和减法程序 一个浮点加法和减法的执行时间在很大程度上依赖于指数的初始差,也依赖于(向右或向左)规格化所需的步骤数。尚不知道有什么方法来给出一个好的理论模型,能告诉我们可预期的特征是什么,但是 D.W.Sweeney 已经做了广泛的经验研究[IBM Systems J. 4 (1965), 31~42]。

借助于一个特殊的跟踪程序, Sweeney 运行了 6 个选自不同计算实验室的“典型”大型数值程序,并十分仔细地考察了每一个浮点加法或减法运算。为收集这些数据做了 25 万次以上的浮点加减,受检验的程序所执行的每 10 条指令中大约有一条是 FADD 或 FSUB。

通过先对第二个操作数取负,可以把减法看做加法;因此,可以像只做加法那样来给出所有的统计, Sweeney 的结果可以综述如下:

大约有 9% 的时间,发现要相加的两个操作数之一等于零,而这通常是累加器 (ACC)。其它的 91% 的情况大致相等地分成为运算数有相同的符号和相反的符号两种情况,以及大致相等地分成 $|u| \leq |v|$ 和 $|v| \leq |u|$ 两种情况,大约有 1.4% 的时间计算答案为零。

对于各种进制的 b ,指数之间的差近似地有由表 1 所示的概率给出的一个特性(此表“5 以上”的行实质上包括当一个操作数为 0 时的所有情况,但“平均行”不包括这些情况)。

表 1 在加法之前进行操作数调整的经验数据

$ e_u - e_v $	$b=2$	$b=10$	$b=16$	$b=64$
0	0.33	0.47	0.47	0.56
1	0.12	0.23	0.26	0.27
2	0.09	0.11	0.10	0.04
3	0.07	0.03	0.02	0.02
4	0.07	0.01	0.01	0.02
5	0.04	0.01	0.02	0.00
5 以上	0.28	0.13	0.11	0.09
平均	3.1	0.9	0.8	0.5

当 u 和 v 有相同的符号且均已规格化时, $u + v$ 或者需要右移一位(对于小数上溢),或者根本不需要什么规格化移位。当 u 和 v 有相反的符号时,在规格化期间要进行零次或更多次左移,表 2 给出了所观察到的需要的移位次数;这个表的最后一行包括结果为零的所有情况,每次规格化时平均移位的次数,当 $b=2$ 时大约为 0.9,当 $b=10$ 时或 $b=16$ 时大约为 0.2,而 $b=64$ 时大约为 0.1。

表 2 做了加法之后进行规格化的经验数据

	$b=2$	$b=10$	$b=16$	$b=64$
右移 1 位	0.20	0.07	0.06	0.03
不移位	0.59	0.80	0.82	0.87
左移 1 位	0.07	0.08	0.07	0.06
左移 2 位	0.03	0.02	0.01	0.01
左移 3 位	0.02	0.00	0.01	0.00
左移 4 位	0.02	0.01	0.00	0.01
左移 > 4 位	0.06	0.02	0.02	0.02

B. 小数部分 对浮点程序的进一步分析可以以随机选择的规格化浮点数小数部分的统计分布为基础。在这种情况下,事实是十分令人惊奇的,并且有有趣的理论来解释观察到的不寻常现象。

为方便起见,暂且假定我们正处理浮点十进算术(即以 10 为进制);下列的讨论可以非常直截了当地修改成任何其它正整数进制 b 。假定我们有一个随机的正的规格化的数 $(e, f) = 10^e \cdot f$ 。由于 f 是规格化的,我们知道它的前导数字为 1, 2, 3, 4, 5, 6, 7, 8 或 9, 而且假定前导数字中的每一个都将有 $1/9$ 的机会出现似乎是合乎自然的。但是事实上,实际的特性却并非如此。例如,竟有 30% 以上的时间前导数字趋向等于 1!

检查刚才所作论断的一种方法,是从某些标准参考文献取一张物理常数(例如,光速、重力加速度)表。例如,如果考察 *Handbook of Mathematical Functions* (U. S. Dept of Commerce, 1964), 我们发现在表 2.3 中给出的 28 个不同物理常数中竟有 8 个,大约占 29%,其前导数字等于 1。对于 $1 \leq n \leq 100$ 的 $n!$ 的十进值恰有 30 个以上以 1 开头;对于 $1 \leq n \leq 100$, 2^n 和 F_n 的十进制也如此,我们也可以考察人口调查报告,或者一份农业年鉴(但不是电话号码簿)。

在袖珍计算器出现之前,一些对数表前头的若干页势必用得十分脏,而后边的部分则保持得相当干净和整洁。这一现象首先是由美国天文学家 Simon Newcomb 明确提到的 [*Amer. J. Math.* 4 (1881), 39~40]。他给出了很好的论据使人们坚信,前导数字 d 以 $\log_{10}(1 + 1/d)$ 的概率出现。许多年以后, Frank Benford 才从实践经验中发现了相同的分布,他报告了取自不同来源的 20 229 个观察结果 [*Proc. Amer. Philosophical Soc.* 78 (1938), 551~572]。

为了解释这个前导数字法则,让我们对于以浮点记号写数做更仔细的观察。如果取任何正数 u , 则它的前导数字由公式 $10f_u = 10^{(\log_{10} u) \bmod 1}$ 确定;因此前导数字小于 d 的充要条件是:

$$(\log_{10} u) \bmod 1 < \log_{10} d \quad (1)$$

现在如果有从在自然界中可能出现的某个合理分布中选择的“随机”正数 U , 则可以预期,至少很近似地, $(\log_{10} U) \bmod 1$ 将一致地分布于 0 和 1 之间(类似地,我们可预期 $U \bmod 1$, $U^2 \bmod 1$, $\sqrt{U + \pi} \bmod 1$ 等等,是一致分布的。按实质上相同的理由,我们可以预期轮盘赌的轮子是无偏倚的)。因此,由(1),前导数字将是 1 的概率为 $\log_{10} 2 \approx 30.103\%$ 。数字是 2 的概率为 $\log_{10} 3 - \log_{10} 2 \approx 17.609\%$;一般地,如果 r 是 1 与 10 之间的实数值,则我们应该近似地有 $\log_{10} r$ 的机会得到 $10f_U \leq r$ 。

前导数字趋向于是小的数字这一事实,使得浮点计算的“平均误差”估计这一最明显的技术成为无效的。

当然只可以说,上边的启发式的论证并未能证明所述的法则。它仅仅向我们说明了为什么前导数字有这种特性的一个似乎有理的原因。R. Hamming 已经提出了分析前导数字的一个有趣的方法:设 $p(r)$ 是 $10f_U \leq r$ 的概率,其中 $1 \leq r \leq 10$, 且 f_U 是一个随机规格化浮点数 U 的规格化的小数部分。如果考虑现实世界的随机量,则我们发现,它们是通过任意的单位来测量的;而且,如果我们去改变一米或一克的定义,则许多基本物理常数都将有不同的值。现在假设,宇宙中的所有数突然地乘以一个常数因子 c ;则随机浮点量的世界在这个变换下实质上应该保持不变,

所以 $p(r)$ 不应受影响。

每个数都乘以 c 的结果是把 $(\log_{10} U) \bmod 1$ 变成 $(\log_{10} U + \log_{10} c) \bmod 1$ 。现在该是建立描述所希望特性的公式的时候了；可以假定 $1 \leq c \leq 10$ ，由定义，

$$p(r) = \Pr((\log_{10} U) \bmod 1 \leq \log_{10} r)$$

由我们的假定，我们也应有

$$\begin{aligned} p(r) &= \Pr((\log_{10} U + \log_{10} c) \bmod 1 \leq \log_{10} r) \\ &= \begin{cases} \Pr((\log_{10} U \bmod 1) \leq \log_{10} r - \log_{10} c \\ \quad \text{或 } (\log_{10} U \bmod 1) \geq 1 - \log_{10} c), & \text{如果 } c \leq r \\ \Pr((\log_{10} U \bmod 1) \leq \log_{10} r + 1 - \log_{10} c \\ \quad \text{且 } (\log_{10} U \bmod 1) \geq 1 - \log_{10} c), & \text{如果 } c \geq r \end{cases} \\ &= \begin{cases} p(r/c) + 1 - p(10/c), & \text{如果 } c \leq r \\ p(10r/c) - p(10/c), & \text{如果 } c \geq r \end{cases} \end{aligned} \quad (2)$$

现在通过定义 $p(10^n r) = p(r) + n$ ，把函数 $p(r)$ 推广到范围 $1 \leq r \leq 10$ 之外的值去；这样，如果以 d 代替 $10/c$ ，则(2)的最后一个等式可以写成

$$p(rd) = p(r) + p(d) \quad (3)$$

如果我们所做的关于乘以一个常因子后分布不变性的假定成立，则等式(3)必然对所有 $r > 0$ 和 $1 \leq d \leq 10$ 成立。现在 $p(1) = 0$ ， $p(10) = 1$ 的事实意味着

$$1 = p(10) = p((\sqrt[n]{10})^n) = p(\sqrt[n]{10}) + p((\sqrt[n]{10})^{n-1}) = \cdots = np(\sqrt[n]{10})$$

因此，对于所有正整数 m 和 n ，我们导出 $p(10^{m/n}) = m/n$ 。如果现在决定要求 p 是连续的，则我们不得不作出这样的结论： $p(r) = \log_{10} r$ ，而这正是所需的定律。

尽管这一论断比头一个可能更使人信服些，但如果我们坚持概率的习惯概念，则在深究之下，它并不真正成立。使上述论证严格化的传统方法是，假定有某个基本数字分布 $F(u)$ ，使得给定的正数 $U \leq u$ 的概率为 $F(u)$ ；则我们所关心的概率是

$$p(r) = \sum_m (F(10^m r) - F(10^m)) \quad (4)$$

其中对于所有的值 $-\infty < m < \infty$ 求和。我们关于比例不变量和连续性的假定导致了下列结论：

$$p(r) = \log_{10} r$$

利用同一论断，我们可以“证明”对每个正数 $b \geq 2$ ，当 $1 \leq r \leq b$ 时有

$$\sum_m (F(b^m r) - F(b^m)) = \log_b r \quad (5)$$

但是不存在分布函数 F ，它对所有这样的 b 和 r 满足这个等式！（见习题 7。）

摆脱这种困境的一种方法，是把对数法则 $p(r) = \log_{10} r$ 仅仅看成是真正分布的一个非常接近的近似。真正的分布本身也许随宇宙的扩展而可以改变，并且随着时间的推移变成越来越好地近似；而且如果我们以一个任意的进制 b 来代替 10，则随着 b 之增大，近似反而变得更不精确了（在任意给定的时间里）。摆脱这一困境的另一比较合适的途径，是抛弃分布函数的传统思想，这已由 R. A. Raimi 提出来，见

AMM 76 (1969), 342~348。

上段的模棱两可的说法,大概是非常令人不满意的说明。所以下边进一步的计算(它坚持严格的数学论证,避免任何直觉的然而似是而非的关于概率的观念)应该是受欢迎的:考虑正整数前导数字的分布,而不考虑某个想像的实数集合的分布。这一课题的研究是十分有趣的,不仅是由于它在一定程度上阐明了浮点数据的概率分布,而且也构成了一个特别有教益的例子,说明如何把离散数学的方法同无穷小的微积分方法结合起来。

在下边的讨论中,设 r 是一个固定的实数, $1 \leq r \leq 10$; 我们将试图给出对 $p(r)$ 的一个合理定义,即在一个“随机”的正整数 N 的表示 $10^e \cdot f_N$ 中, $10f_N < r$ 的概率,并假定其精度是无限的。

为了开始,让我们试验像 3.5 节的“Pr”定义那样,利用极限方法来找这个概率,不妨把该定义重新表述如下:

$$P_0(n) = [n = 10^e \cdot f, \text{ 其中 } 10f < r] = [(\log_{10} n) \bmod 1 < \log_{10} r] \quad (6)$$

现在 $P_0(1), P_0(2), \dots$ 是 0 与 1 的无穷序列,同时诸 1 表示贡献于我们正在寻求的这一概率的情况,我们可以通过定义

$$P_1(n) = \frac{1}{n} \sum_{k=1}^n P_0(k) \quad (7)$$

来对这个序列取“平均”,因此如果利用第 3 章的技术生成 1 与 n 之间的随机整数,并把它转换成浮点十进形式 (e, f) , 则 $10f < r$ 的概率恰是 $P_1(n)$ 。令 $\lim_{n \rightarrow \infty} P_1(n)$ 是我们所需要的“概率” $p(r)$ 是很自然的,而这恰恰是我们在 3.5 节中所做过的。

但在这种情况下极限并不存在,例如,考虑子序列

$$P_1(s), P_1(10s), P_1(100s), \dots, P_1(10^n s), \dots$$

其中 s 是一个实数, $1 \leq s \leq 10$ 。如果 $s \leq r$, 则我们发现

$$\begin{aligned} P_1(10^n s) &= \frac{1}{10^n s} (\lceil r \rceil - 1 + \lceil 10r \rceil - 10 + \dots + \lceil 10^{n-1} r \rceil - 10^{n-1} + \lfloor 10^n s \rfloor + 1 - 10^n) = \\ &= \frac{1}{10^n s} (r(1 + 10 + \dots + 10^{n-1}) + O(n) + \lfloor 10^n s \rfloor - 1 - 10 - \dots - 10^n) = \\ &= \frac{1}{10^n s} \left(\frac{1}{9} (10^n r - 10^{n+1}) + \lfloor 10^n s \rfloor + O(n) \right) \end{aligned} \quad (8)$$

因此,当 $n \rightarrow \infty$ 时, $P_1(10^n s)$ 趋向于极限值 $1 + (r - 10)/9s$ 。只要我们以 $\lceil 10^n r \rceil$ 代替 $\lfloor 10^n s \rfloor + 1$, 上述对于 $s \leq r$ 的计算就可修改成使它对于 $s > r$ 也成立; 所以当 $s \geq r$ 时, 我们求得极限值是 $10(r - 1)/9s$ 。[见 J. Franel, *Naturforschende Gesellschaft, Vierteljahrsschrift* 62 (Zürich: 1917), 286~295。]

换言之, 序列 $\langle P_1(n) \rangle$ 有子序列 $\langle P_1(10^n s) \rangle$, 当 s 从 1 变到 r 又变到 10 时, 它的极限从 $(r - 1)/9$ 上升到 $10(r - 1)/9r$ 而后又下降到 $(r - 1)/9$ 。我们看到 $P_1(n)$ 当 $n \rightarrow \infty$ 时没有极限; 而且对于很大的 n , $P_1(n)$ 的值也并不特别近似于我们猜测

的极限 $\log_{10} r!$

由于 $P_1(n)$ 并不趋于一个极限, 我们可以再次尝试使用和(7)同样的思想, 用平均的办法来消除异常特性。一般来说, 设

$$P_{m+1}(n) = \frac{1}{n} \sum_{k=1}^n P_m(k) \quad (9)$$

则 $P_{m+1}(n)$ 将趋于成为比 $P_m(n)$ 特性更好的序列, 我们试一下用定量的计算来证实这一点; 对于 $m=0$ 的特殊情况, 我们的经验指出, 可能值得考虑子序列 $P_{m+1}(10^n s)$ 。事实上, 可以导出下列结果:

引理 Q 对于任意的整数 $m \geq 1$ 和任意实数 $\epsilon > 0$, 存在函数 $Q_m(s)$, $R_m(s)$ 和一个整数 $N_m(\epsilon)$, 使得当 $n > N_m(\epsilon)$ 和 $1 \leq s \leq 10$ 时, 有

$$|P_m(10^n s) - Q_m(s) - R_m(s)[s > r]| < \epsilon \quad (10)$$

而且函数 $Q_m(s)$ 和 $R_m(s)$ 满足关系

$$\begin{aligned} Q_m(s) &= \frac{1}{s} \left(\frac{1}{9} \int_1^{10} Q_{m-1}(t) dt + \int_1^r Q_{m-1}(t) dt + \frac{1}{9} \int_r^{10} R_{m-1}(t) dt \right) \\ R_m(s) &= \frac{1}{s} \int_r^s R_{m-1}(t) dt \\ Q_0(s) &= 1, \quad R_0(s) = -1 \end{aligned} \quad (11)$$

证明 考虑由(11)定义的函数 $Q_m(s)$ 和 $R_m(s)$, 并命

$$S_m(t) = Q_m(t) + R_m(t)[t > r] \quad (12)$$

我们将对 m 用归纳法来证明这一引理。

首先注意 $Q_1(s) = (1 + (s-1) - (10-r)/9)/s = 1 + (r-10)/9s$, 及 $R_1(s) = (r-s)/s$ 。由(8)我们求得

$$|P_1(10^n s) - S_1(s)| = O(n)/10^n$$

当 $m=1$ 时, 就证明了引理。

现在对于 $m > 1$, 我们有

$$\begin{aligned} P_m(10^n s) &= \frac{1}{s} \left(\sum_{0 \leq j < n} \frac{1}{10^{n-j}} \sum_{10^{j-1} \leq k < 10^j} \frac{1}{10^j} P_{m-1}(k) + \right. \\ &\quad \left. \sum_{10^n \leq k < 10^{n+1}} \frac{1}{10^n} P_{m-1}(k) \right) \end{aligned}$$

我们要逼近这个量。由归纳法, 当 $1 \leq q \leq 10$ 且 $j > N_{m-1}(\epsilon)$ 时,

$$\left| \sum_{10^{j-1} \leq k < 10^j} \frac{1}{10^j} P_{m-1}(k) - \sum_{10^{j-1} \leq k < 10^j} \frac{1}{10^j} S_{m-1}\left(\frac{k}{10^j}\right) \right| \quad (13)$$

小于 $q\epsilon$ 。由于 $S_{m-1}(t)$ 是连续的, 故它是黎曼可积函数, 而且由积分定义, 对于大于某个数 N 的所有 j , 与 q 无关, 差

$$\left| \sum_{10^{j-1} \leq k < 10^j} \frac{1}{10^j} S_{m-1}\left(\frac{k}{10^j}\right) - \int_1^q S_{m-1}(t) dt \right| \quad (14)$$

小于 ϵ 。我们可以选择 N 为大于 $N_{m-1}(\epsilon)$ 者。因此,如果 M 是对所有正整数 j 成立的(13)+(14)的上限,则对于 $n > N$,差

$$\left| P_m(10^n s) - \frac{1}{s} \left(\sum_{0 \leq j < n} \frac{1}{10^{n-j}} \int_1^{10} S_{m-1}(t) dt + \int_1^s S_{m-1}(t) dt \right) \right| \quad (15)$$

以

$$\sum_{j=0}^N \frac{M}{10^{n-j}} + \sum_{N < j < n} \frac{11\epsilon}{10^{n-j}} + 11\epsilon$$

为界。最后,出现于(15)中的和 $\sum_{0 \leq j < n} (1/10^{n-j})$ 等于 $(1 - 1/10^n)/9$; 所以如果 n 取得足够大,就可以使

$$\left| P_m(10^n s) - \frac{1}{s} \left(\frac{1}{9} \int_1^{10} S_{m-1}(t) dt + \int_1^s S_{m-1}(t) dt \right) \right|$$

小于比如说 20ϵ 。把这同(10)和(11)比较就完成了证明。 ■

引理 Q 的要旨在于我们有极限关系

$$\lim_{n \rightarrow \infty} P_m(10^n s) = S_m(s) \quad (16)$$

而且,由于当 s 变动时 $S_m(s)$ 不是常数,故极限

$$\lim_{n \rightarrow \infty} P_m(n)$$

(它本应是我们所希望的概率)对任何 m 都不存在。这一情况展示于图 5 中,它示出了当 m 小且 $r=2$ 时 $S_m(s)$ 的值。

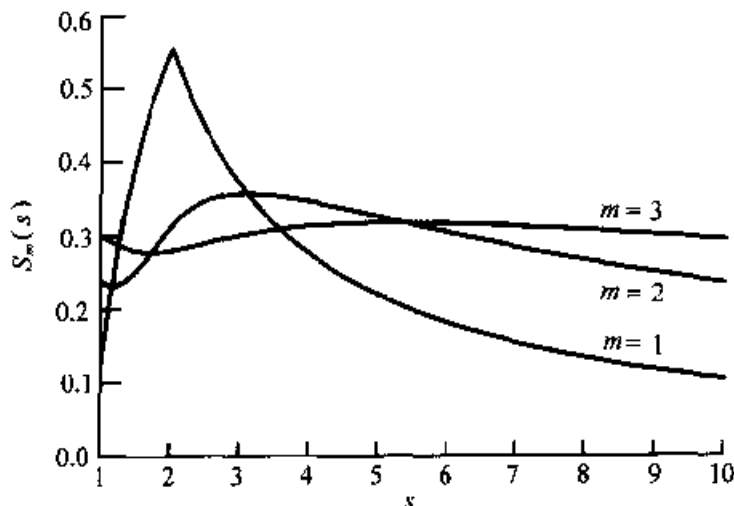


图 5 前导数字为 1 的概率

即使 $S_m(s)$ 不是一个常数,以致于我们对于 $P_m(n)$ 不能有一个确定的极限,只要注意图 5 便可知,对于 $m=3$, $S_m(s)$ 的值已经非常接近于 $\log_{10} 2 = 0.30103 \dots$ 。因此,我们有充分理由猜测,对所有大的 m , $S_m(s)$ 非常接近于 $\log_{10} r$, 而且事实上,函数 $\langle S_m(s) \rangle$ 的序列一致地收敛到常数函数 $\log_{10} r$ 。

明确地对所有 m 计算 $Q_m(s)$ 和 $R_m(s)$, 如同在证明以下的定理中那样, 来证明上述猜想是有趣的:

定理 F 设 $S_m(s)$ 是 (16) 中定义的极限, 对所有 $\epsilon > 0$, 存在一个数 $N(\epsilon)$, 使得当 $m > N(\epsilon)$ 时, 对于 $1 \leq s \leq 10$,

$$|S_m(s) - \log_{10} r| < \epsilon \quad (17)$$

证明 鉴于引理 Q, 如果能证明存在一个与 ϵ 有关的 M , 使得对于 $1 \leq s \leq 10$, 和对于所有 $m > M$, 我们有

$$|Q_m(s) - \log_{10} r| < \epsilon \quad \text{及} \quad |R_m(s)| < \epsilon \quad (18)$$

则我们就能证明这个结果。

不难对 R_m 来解递推公式 (11): 我们有 $R_0(s) = -1$, $R_1(s) = -1 + r/s$, $R_2(s) = -1 + (r/s)(1 + \ln(s/r))$, 而一般地

$$R_m(s) = -1 + \frac{r}{s} \left(1 + \frac{1}{1!} \ln \left(\frac{s}{r} \right) + \cdots + \frac{1}{(m-1)!} \left(\ln \left(\frac{s}{r} \right) \right)^{m-1} \right) \quad (19)$$

对于所述的 s 的范围, 这一致地收敛到 $-1 + (r/s) \exp(\ln(s/r)) = 0$ 。

Q_m 的递推公式 (11) 具有形式

$$Q_m(s) = \frac{1}{s} \left(c_m + 1 + \int_1^s Q_{m-1}(t) dt \right) \quad (20)$$

其中

$$c_m = \frac{1}{9} \left(\int_1^{10} Q_{m-1}(t) dt + \int_r^{10} R_{m-1}(t) dt \right) - 1 \quad (21)$$

再次地通过试验开头少数情况, 并推测一个可由归纳法证明的公式, 容易求出递推式 (20) 的解; 我们发现

$$Q_m(s) = 1 + \frac{1}{s} \left(c_m + \frac{1}{1!} c_{m-1} \ln s + \cdots + \frac{1}{(m-1)!} c_1 (\ln s)^{m-1} \right) \quad (22)$$

对我们说来, 剩下的是计算系数 c_m , 由 (19), (21) 和 (22), 它满足关系式

$$\begin{aligned} c_m &= (r - 10)/9 \\ c_{m+1} &= \frac{1}{9} \left(c_m \ln 10 + \frac{1}{2!} c_{m-1} (\ln 10)^2 + \cdots + \frac{1}{m!} c_1 (\ln 10)^m + \right. \\ &\quad \left. r \left(1 + \frac{1}{1!} \ln \frac{10}{r} + \cdots + \frac{1}{m!} \left(\ln \frac{10}{r} \right)^m \right) - 10 \right) \end{aligned} \quad (23)$$

这个序列最初从外表上看非常复杂。实际上不难借助于生成函数对它进行分析。

命

$$C(z) = c_1 z + c_2 z^2 + c_3 z^3 + \cdots$$

则由于 $10^z = 1 + z \ln 10 + (1/2!)(z \ln 10)^2 + \cdots$, 我们导出

$$\begin{aligned} c_{m+1} &= \frac{1}{10} c_{m+1} + \frac{9}{10} c_{m+1} = \\ &= \frac{1}{10} \left(c_{m+1} + c_m \ln 10 + \cdots + \frac{1}{m!} c_1 (\ln 10)^m \right) + \end{aligned}$$

$$\frac{r}{10} \left(1 + \cdots + \frac{1}{m!} \left(\ln \frac{10}{r} \right)^m \right) - 1$$

是函数

$$\frac{1}{10} C(z) 10^z + \frac{r}{10} \left(\frac{10}{r} \right)^z \left(\frac{z}{1-z} \right) - \frac{z}{1-z} \quad (24)$$

中 z^{m+1} 的系数。这个条件对 m 的所有值都成立,所以(24)必然等于 $C(z)$,而且我们得到显式

$$C(z) = \frac{z}{1-z} \left(\frac{(10/r)^{z-1} - 1}{10^{z-1} - 1} \right) \quad (25)$$

我们要来研究 $C(z)$ 的系数的渐近性质,以完成我们的分析。(25)中的大圆括号中的因子当 $z \rightarrow 1$ 时趋于 $\ln(10/r)/\ln 10 = 1 - \log_{10} r$,所以我们看到

$$C(z) + \frac{1 - \log_{10} r}{1-z} = R(z) \quad (26)$$

是圆

$$|z| < \left| 1 + \frac{2\pi i}{\ln 10} \right|$$

中复变量 z 的解析函数。特别是,对于 $z=1$, $R(z)$ 收敛,所以它的系数趋于零。这就证明了 $C(z)$ 的系数有像 $(\log_{10} r - 1)/(1-z)$ 的系数那样的特性,即是

$$\lim_{m \rightarrow \infty} c_m = \log_{10} r - 1$$

最后,我们可以把这同(22)结合起来,以证明 $Q_m(s)$ 对于 $1 \leq s \leq 10$ 一致地趋于

$$1 + \frac{\log_{10} r - 1}{s} \left(1 + \ln s + \frac{1}{2!} (\ln s)^2 + \cdots \right) = \log_{10} r \quad \blacksquare$$

因此,我们已经通过直接计算,建立了对于整数的对数定律。同时看到,它是对于平均行为的极好的近似,尽管它绝不能精确地达到。

上边给出的引理 Q 和定理 F 的证明是 B. J. Flehinger 在 AMM 73 (1966), 1056~1061 中的方法的简化和扩充。许多作者已经写了关于初始数字分布的论著,并证明对数法则是许多奠基性的分布的一个好的近似;关于这方面著作的广泛评论,见 Ralph A. Raimi 的综述, AMM 83 (1976), 521~538, 以及 Peter Schatte, *J. Information Processing and Cybernetics* 24 (1988), 443~455。

习题 17 讨论了定义概率的一个方法,在这方法之下,对数定律对整数精确地成立。而且习题 18 揭示了对于整数的概率的任何合理的定义必然导致对数定律成立,如果它对前导数的概率赋一个值的话。

当然,浮点计算主要地是对非整数进行运算;由于对于整数的熟悉和其简单性,我们已经研究了整数。当考虑任意实数时,理论结果更难得到,但是对于实数的计算将接近于总是趋于产生对于小数部分的对数分布越来越好的近似,在这个意义下有越来越多的证据表明,相同的统计是适用的。例如, Peter Schatte [*Zeitschrift für angewandte Math. und Mechanik* 53 (1973), 553~565] 证明,在温和的限制下,独立的相等分布的随机实变量的乘积趋向对数分布。这种变量的和也如此,只不过是

重复平均的意义下。*J. L. Barlow* 和 *E. H. Bareiss* 已经得到类似的结果[*Computing* 34(1985), 325~347]。

习 题

1. [13] 如果 u 和 v 是有相同符号的浮点十进制数, 则按表 1 和表 2, 在计算 $u \oplus v$ 时出现小数溢出的概率是多少?

2. [42] 试对浮点加法和减法做进一步的检验, 以确认或改进表 1 和表 2 的精度。

3. [15] 按对数定律, 浮点十进制数的两个前导数字是“23”的概率等于多少?

4. [M18] 正文中指出经常使用的对数表前边一些页要比后边一些页脏。如果我们有一份反对数表, 即当给定 $\log_{10} x$ 时, 给出 x 的值的一份表, 则情况如何? 这样一份表的哪些页最脏?

► 5. [M20] 假设 U 是在区间 $0 < U < 1$ 中一致分布的一个实数。问 U 的前导数字的分布是什么?

6. [23] 如果我们有 $n+1$ 位的二进制计算机字, 则我们可以使用 p 位于浮点二进制的小数部分, 使用 1 位于符号部分, 使用 $n-p$ 位于指数部分。这意味着可表示的值的范围, 即最大的与最小的规格化正值的比例, 实质上是 $2^{2^{n-p}}$ 。同样的计算机字可以用来表示浮点十六进制的数, 即进制为 16, 小数部分 $p+2$ 位 ($(p+2)/4$ 个十六进数字), 以及指数部分 $n-p-2$ 位的浮点数; 此时值的范围和以前一样, 是 $16^{2^{n-p-2}} = 2^{2^{n-p}}$, 只是小数部分的位数更多了。这看起来好像我们不花任何代价而得到了收获, 但是进制 16 的规格化条件是较弱的, 原因在于在小数部分中可以有多达三个前导 0 数字。于是, 并非所有 $p+2$ 位都是“有效位”。

在对数定律的基础上, 一个十六进制正的规格化浮点数的小数部分恰有 0 个, 1 个, 2 个和 3 个前导 0 数字的概率是多少? 试讨论十六进制相对于二进制的优点。

7. [HM28] 证明, 对于每个整数 $b \geq 2$, 以及对于所有在范围 $1 \leq r \leq b$ 中的实数值 r , 没有满足 (5) 的分布函数 $F(u)$ 。

8. [HM23] 当 $m \rightarrow 0$ 时, (10) 对于适当的 $N_0(\epsilon)$ 是否成立?

9. [HM25] (P. Diaconis) 设 $P_1(n), P_2(n), \dots$ 是根据等式 (9) 重复地平均一个给定函数 $P_0(n)$ 所定义的任何函数序列。证明对所有固定的 $n, \lim_{m \rightarrow \infty} P_m(n) = P_0(1)$ 。

► 10. [HM28] 正文中的论断证明, $c_m = \log_{10} r - 1 + \epsilon_m$, 其中 ϵ_m 当 $m \rightarrow \infty$ 时趋于 0, 试求出 c_m 的渐近展开式的下一项。

► 11. [M15] 证明, 如果 U 是按照对数定律分布的一个随机变量, 则 $1/U$ 也是。

12. [HM25] (R. W. Hamming) 本题的目的在于证明浮点乘法的结果趋于比操作数更精确地遵守对数定律。设 U 和 V 是规格化的随机正浮点数, 它们的小数部分分别以密度函数 $f(x)$ 和 $g(x)$ 独立地分布。于是, $f_u \leq r$ 和 $f_v \leq s$ 的概率是 $\int_{1/b}^r \int_{1/b}^s f(x)g(y)dy dx$, 其中 $1/b \leq r, s \leq 1$ 。设 $h(x)$ 是 $U \times V$ (未舍入) 的小数部分的密度函数。定义一个密度函数 f 的异常性 $A(f)$ 为极大相对误差

$$A(f) = \max_{1/b \leq x \leq 1} \left| \frac{f(x) - l(x)}{l(x)} \right|$$

其中 $l(x) = 1/(x \ln b)$ 是对数分布的密度函数。

证明 $A(h) \leq \min(A(f), A(g))$ 。(特别是, 如果两个小数有对数分布, 则乘积也是。)

► 13. [M20] 浮点乘法程序算法 4.2.1M 在规格化时, 依赖于是否有 $f_u f_v \geq 1/b$, 而要求右移 0

次或1次。假定输入操作数按照对数定律独立地分布,问结果规格化时不需左移的概率是多少?

►14. [HM30] 设 U 和 V 是随机规格化的正浮点数,它们的小数部分是按照对数定律独立分布的。且设 p_k 是它们指数的差为 k 的概率。假定指数的分布与小数部分的分布无关,试给出在进行浮点加法 $U \oplus V$ 期间“小数溢出”的概率公式,用进制 b 和量 p_0, p_1, p_2, \dots 表示,把这个结果同习题1做比较(忽略舍入)。

15. [HM28] 设 U, V, p_0, p_1, \dots 如习题14所示,并假定使用进制10的算术。证明不管 p_0, p_1, p_2, \dots 的值为何,和数 $U \oplus V$ 不会精确地遵守对数定律,而且事实上 $U \oplus V$ 有前导数字1的概率总是严格地小于 $\log_{10} 2$ 。

16. [HM28] (P. Diaconis) 设对于每个 $n, P_0(n)$ 是0或1,并像在(9)中那样,通过重复平均来定义“概率” $P_{m+1}(n)$,证明如果 $\lim_{n \rightarrow \infty} P_1(n)$ 不存在,则对于任何 $m, \lim_{n \rightarrow \infty} P_m(n)$ 也不存在。[提示:证明对于某个固定常数 $M > 0$,当我们有 $(a_1 + \dots + a_m)/n \rightarrow 0$ 和 $a_{n+1} \leq a_n + M/n$ 时, $a_n \rightarrow 0$]。

►17. [HM25] (M. Tsuji) 定义 $\Pr(S(n))$ 的值的另一个方法是计算量 $\lim_{n \rightarrow \infty} (H_n^{-1} \sum_{k=1}^n [S(k)]/k)$;可以证明只要按照定义3.5A后者存在,这个调和概率就存在而且等于 $\Pr(S(n))$ 。试证明命题“($\log_{10} n \bmod 1 < r$)”的调和概率存在而且等于 r 。(因此在这个意义下整数的初始数字精确地满足对数定律。)

►18. [HM30] 设 $P(S)$ 是任何在正整数集合 S 上定义的真值函数,但不必是在所有这样的集合上定义,并满足以下较弱的公理:

i) 如果 $P(S)$ 与 $P(T)$ 有定义且 $S \cap T = \emptyset$, 则 $P(S \cup T) = P(S) + P(T)$ 。

ii) 如果 $P(S)$ 有定义,则 $P(S+1) = P(S)$, 其中 $S+1 = \{n+1 \mid n \in S\}$ 。

iii) 如果 $P(S)$ 有定义,则 $P(2S) = \frac{1}{2}P(S)$, 其中 $2S = \{2n \mid n \in S\}$ 。

iv) 如果 S 是所有正整数的集合,则 $P(S) = 1$ 。

v) 如果 $P(S)$ 有定义,则 $P(S) \geq 0$ 。

另外,假定对于所有正整数 $a, P(L_a)$ 有定义,其中 L_a 是所有其十进制表示以 a 开始的整数的集合:

$$L_a = \{n \mid 10^m a \leq n < 10^m(a+1), \text{对某个整数 } m\}$$

(在这个定义中, m 可以为负;例如,1是 L_{10} 的一个元素,但不是 L_{11} 的元素。)证明对于所有整数 $a \geq 1, P(L_a) = \log_{10}(1 + 1/a)$ 。

19. [HM25] (R. L. Duncan) 证明斐波那契数的前导数字遵守小数部分的对数定律: $\Pr(10^{f_{F_n}} < r) = \log_{10} r$ 。

20. [HM40] 通过求出当 $n \rightarrow \infty$ 时 $P_m(10^s) - S_m(s)$ 的渐近特性,使(16)更精确。

4.3 多精度算术

现在我们考虑有任意高精度的数的运算。为考察的简便性,假定考虑的对象是整数而不是一个嵌入了小数点的数。

4.3.1 经典算法

在这一小节中,我们将讨论下列运算的一些算法。

- a) n 位整数的加法或减法, 给出一个 n 位的答案和一个进位。
- b) n 位整数与 m 位整数的乘法, 给出 $(m+n)$ 位的答案。
- c) $m+n$ 位整数除以 n 位整数的除法, 给出一个 $(m+1)$ 位的商和 n 位的余数。

这些可以叫做“经典算法”, 因为在若干世纪中, 仅当同这些过程相联系时才使用“算法”一词。 n 位整数一词, 指的是小于 b^n 的任何整数, 其中 b 是表达数的常规位置记法的进制, 在这个记法中这样的数至多需要 n 个位置。

把整数的经典算法应用到含有小数点的数, 或者扩充精度的浮点数, 这些都是直截了当的事, 其方式就像把在 MIX 中对于整数定义的算术运算, 应用到这些更为一般的问题上那样。

在这一节中, 我们将研究对于 b 进制记法下表达的整数, 进行上述 a), b) 和 c) 运算的算法, 其中 b 是任意的大于或等于 2 的整数。这样, 这些算法是算术过程的十分一般的定义, 因而它们是与任何具体的计算机无关的。但这一节中的讨论也是有一点面向机器的, 因为我们主要关心的是由计算机进行高精度计算的有效方法。尽管我们的例子是以假想的 MIX 计算机为基础的, 实际上同样的考虑几乎适用于每台其它的机器。

为了了解扩充精度的数, 最重要的事实是它们都可看成以 w 进制记号写成的数, 其中 w 是计算机的字长。例如, 在一台字长为 $w = 10^{10}$ 的计算机上, 占 10 个字的整数有 100 个十进制数字; 但是我们把它当做一个 10^{10} 进制的 10 位的数。这个观点是正确的, 正如我们可以简单地把一些位合在一起来实现从二进制记号转换成八进制记号一样(见等式 4.1-(5))。

在这些约定之下, 我们有下列基本运算:

- a_0) 一位整数的加法和减法, 给出一位的答案和进位。
- b_0) 一位的整数乘以另外一位整数的乘法, 给出一个两位的答案。
- c_0) 一个两位的整数除以一个一位整数的除法, 假定商是一位的整数, 而且产生的余数也是一位的。

如果必要的话, 通过调整字长, 则几乎所有的计算机都有这三种运算可用, 因而我们就能借助于基本的运算 a_0), b_0) 和 c_0) 构造上述算法 a), b) 和 c)。

由于把扩充精度整数具体化为进制 b 的数, 因此设想 $b = 10$, 并设想用手进行算术运算的情况有时是有帮助的。这样就使运算 a_0) 类似于记忆加法表, b_0) 类似于记忆乘法表, 而 c_0) 实际上是反过来记忆乘法表。因此高精度数的更为复杂的运算 a), b), c), 就可用在初等学校所学的简单加法、减法、乘法和长除过程来进行。事实上, 我们在这一小节里所要讨论的大多数算法, 都不过是熟知的纸笔运算的机械化。当然, 必须指出, 这些算法比起五年级时所学的要精确得多, 而且我们还打算把计算机存储空间和运行时间尽可能缩小。

为避免繁琐的讨论和麻烦的记号, 假定我们处理的所有数据都是非负的。计算符号等附加工作也是十分直截了当的, 尽管当在不使用带符号量表示的计算机上处

理补数时需要小心。这样一些问题会在本小节接近末尾处讨论。

首先来看加法,它当然非常简单,但它值得仔细研究,因为同样的思想也出现在其它算法中。

算法 A(非负整数的加法) 给定非负的 n 位整数 $(u_{n-1}u_{n-2}\cdots u_0)_b$ 和 $(v_{n-1}v_{n-2}\cdots v_0)_b$ 。本算法形成它们的 b 进制和 $(w_nw_{n-1}\cdots w_1w_0)_b$ 。这里 w_n 是“进位”,而且总是等于 0 或 1。

A1. [初始化] 置 $j \leftarrow n, k \leftarrow 0$ 。(变量 j 将跑遍各个数字位置,而变量 k 记住在每个步骤的进位。)

A2. [加数字] 置 $w_j \leftarrow (u_j + v_j + k) \bmod b$, 以及 $k \leftarrow \lfloor (u_j + v_j + k)/b \rfloor$ 。(由关于计算的归纳法,我们将总有

$$u_j + v_j + k \leq (b-1) + (b-1) + 1 < 2b$$

因此依赖于有无进位出现, k 被置成 0 或 1, 等价地, $k \leftarrow \lceil u_j + v_j + k \geq b \rceil$ 。)

A3. [对 j 进行循环] j 减 1。现在若 $j < n$, 则返回步骤 A2; 否则置 $w_n \leftarrow k$, 并终止这一算法。■

关于算法 A 成立的形式证明, 见习题 4。

这个加法过程的 MIX 程序, 可以采取如下形式:

程序 A(非负整数的加法) 设 $\text{LOC}(u_j) \equiv U + j, \text{LOC}(v_j) \equiv V + j, \text{LOC}(w_j) \equiv W + j$, $rII \equiv j - n, rA \equiv k$, 字的大小 $\equiv b, N \equiv n$ 。

01		ENN1	N	1	<u>A1. 初始化。</u> $j \leftarrow 0$ 。
02		JOV	OFLO	1	确保溢出开关断开
03	1H	ENTA	0	$N + 1 - K$	$k \leftarrow 0$
04		JIZ	3F	$N + 1 - K$	如果 $j = n$, 则转 A3。
05	2H	ADD	$U + N, 1$	N	<u>A2. 加数字</u>
06		ADD	$V + N, 1$	N	
07		STA	$W + N, 1$	N	
08		INCI	1	N	<u>A3. 对 j 进行循环。</u> $j \leftarrow j + 1$
09		JNOV	1B	N	如果无溢出, 则置 $k \leftarrow 0$
10		ENTA	1	K	否则置 $k \leftarrow 1$
11		JIF	2B	K	如果 $j < n$, 则转 A2
12	3H	STA	$W + N$	1	在 w_n 中存最后进位 ■

这个程序的运行时间是 $10N + 6$ 个周期, 同进位的次数 K 无关。在这小节的末尾对量 K 进行了详细分析。

对算法 A 可以进行许多修改, 而在下边的一些习题中仅提到其中一些修改。关

于这个算法的推广,可用“如何为一台数字计算机设计加法线路”这样的标题。

减法的问题类似于加法,但其差别是值得注意的:

算法 S(非负整数的减法) 给定非负 n 位整数 $(u_{n-1} \cdots u_1 u_0)_b \geq (v_{n-1} \cdots v_1 v_0)_b$, 这个算法形成它们非负 b 进制的差 $(w_{n-1} \cdots w_1 w_0)_b$ 。

S1. [初始化] 置 $j \leftarrow 0, k \leftarrow 0$ 。

S2. [减数字] 置 $w_j \leftarrow (u_j - v_j + k) \bmod b, k \leftarrow \lfloor (u_j - v_j + d)/b \rfloor$ 。(换言之,依据是否出现“借位”,即是否 $u_j - v_j + k < 0$, 把 k 置成 -1 或 0 。在 w_j 的计算中,注意我们必然有 $-b = 0 - (b-1) + (-1) \leq u_j - v_j + k \leq (b-1) - 0 + 0 < b$; 因此 $0 \leq u_j - v_j + k + b < 2b$, 而这提示了下面要说明的计算机实现的方法。)

S3. [对 j 进行循环] j 加 1。现在如果 $j < n$, 则返回步骤 S2; 否则这个算法终止。(当算法终止时,我们应有 $k = 0$; 当且仅当 $(v_{n-1} \cdots v_1 v_0)_b > (u_{n-1} \cdots u_1 u_0)_b$ 时,条件 $k = -1$ 出现,这同给定的假定矛盾。见习题 12。) ▮

在实现减法的 MIX 程序中,最方便的是在整个算法中保留值 $1+k$, 而不是 k , 从而可以在步骤 S2 中计算 $u_j - v_j + (1+k) + (b-1)$ 。(记住 b 是字长。)下面的程序将说明这一点。

程序 S(非负整数的减法) 这一程序同程序 A 类似,但 $rA \equiv 1+k$ 。这里,和这节的其它程序一样,单元 WM1 包含常数 $b-1$, 即可以存于 MIX 字中的最大值;参见程序 4.2.3D, 行 38~39。

01	ENT1	N	1	<u>S1. 初始化。</u> $j \leftarrow 0$
02	JOV	OFL0	1	确保溢出开关断开
03	1H	J1Z	DONE	K+1 如果 $j=0$, 则终止
04	ENTA	1	K	置 $k \leftarrow 0$
05	2H	ADD	U+N,1	<u>S2. 减数字</u>
06	SUB	V+N,1	N	计算 $u_j - v_j + k + b$
07	ADD	WM1	N	
08	STA	W,1	N	(可以是负 0)
09	INCI	1	N	<u>S3. 对 j 进行循环。</u> $j \leftarrow j+1$
10	JOV	1B	N	如果溢出则置 $k \leftarrow 0$
11	ENTA	0	N-K	否则置 $k \leftarrow -1$
12	J1P	2B	N-K	如果 $j < n$ 返回 S2
13	HLT	5		(出错, $v > u$) ▮

这个程序的运行时间是 $12N+3$ 个周期,它比程序 A 的运行时间稍长些。

读者可能会问,为什么不值得搞一个综合的加减法程序以代替算法 A 和算法 S。但对计算机程序的考察表明,一般以使用两个不同的程序为好,它使得计算的内循环尽可能快地进行,因为这些程序很短。

下一个问题是乘法,这里我们要把用于算法 A 的思想稍稍向前推进一步。

算法 M(非负整数的乘法) 给定非负整数 $(u_{m-1} \cdots u_1 u_0)_b$ 和 $(v_{n-1} \cdots v_1 v_0)_b$, 这个算法形成它们的 b 进制的乘积 $(w_{m+n-1} \cdots w_1 w_0)_b$ 。(这个过程通常的纸和笔方法是对于 $0 \leq j < n$, 首先形成部分乘积 $(u_{m-1} \cdots u_1 u_0) \times v_j$, 而后以适当的比例因子把这些乘积加在一起;但在计算机中最好是使乘法和加法并发地进行,像在本算法中所描述的一样。)

M1.[初始化] 把 $w_{m-1}, w_{m-2}, \cdots, w_0$ 全部置为 0。置 $j \leftarrow 0$ 。(如果在这一步 w_{m-1}, \cdots, w_0 不全清 0, 则以下的一些步骤将置

$$(w_{m+n-1} \cdots w_0)_b \leftarrow (u_{m-1} \cdots u_0)_b \times (v_{n-1} \cdots v_0)_b + (w_{m-1} \cdots w_0)_b$$

这个更一般的乘和加运算通常是有用的。)

M2.[乘数为 0?] 如果 $v_j = 0$, 则置 $w_{j+m} \leftarrow 0$, 并返回步骤 M6。(如果 v_j 为 0 的机会相当大, 则这个检验可节省大量时间, 但它也可以省去而不影响算法的正确性。)

M3.[初始化 i] 置 $i \leftarrow 0, k \leftarrow 0$ 。

M4.[乘和加] 置 $t \leftarrow u_i \times v_j + w_{i+j} + k$, 然后置 $w_{i+j} \leftarrow t \bmod b$ 且 $k \leftarrow \lfloor t/b \rfloor$ (这里“进位” k 总是在范围 $0 \leq k < b$ 中, 见下面。)

M5.[对 i 进行循环] i 加 1。现在如果 $i < m$, 则返回步骤 M4, 否则置 $w_{j+m} \leftarrow k$ 。

M6.[对 j 进行循环] j 加 1。现在如果 $j < n$, 则返回步骤 M2, 否则这算法终止。 |

表 1 中, 假定 $b = 10$, 通过说明在步骤 M5 和 M6 开始时的计算状态, 解释了算法 M。关于算法 M 的证明, 见习题 14 的答案。

表 1 914 乘以 84 的乘法

步骤	i	j	u_i	v_j	t	w_4	w_3	w_2	w_1	w_0
M5	0	0	4	4	16	.	.	0	0	6
M5	1	0	1	4	05	.	.	0	5	6
M5	2	0	9	4	36	.	.	6	5	6
M6	3	0	.	4	36	.	3	6	5	6
M5	0	1	4	8	37	.	3	6	7	6
M5	1	1	1	8	17	.	3	7	7	6
M5	2	1	9	8	76	.	6	7	7	6
M6	3	1	.	8	76	7	6	7	7	6

两个不等式

$$0 \leq t < b^2, \quad 0 \leq k < b \quad (1)$$

对于这个算法的有效实现是关键性的,因为它们指出这些计算需要多大的寄存器。随着这个算法的进行,这些不等式可以通过归纳法加以证明,因为如果在步骤 M4 的开始时有 $k < b$,则我们有

$$u_i \times u_j + w_{i+j} + k \leq (b-1) \times (b-1) + (b-1) + (b-1) = b^2 - 1 < b^2.$$

下列的 MIX 程序说明了在一台计算机上实施算法 M 时一些必要的考虑。如果我们的计算机有一条“乘和加”指令,或者有用来求和的一个双倍长的累加器,则对步骤 M4 的编码会稍微简单些。

程序 M(非负整数的乘法) 这个程序类似于程序 A, $r11 \equiv i - m$, $r12 \equiv j - n$, $r13 \equiv i + j$, $\text{CONTENTS}(\text{CARRY}) \equiv k$ 。

01	ENT1	M-1	1	<u>M1. 初始化</u>
02	JOV	OFLO	1	确保溢出开关断开
03	STZ	W,1	M	$w_{r11} \leftarrow 0$
04	DEC1	1	M	
05	J1NN	*-2	M	对 $m > r11 \geq 0$ 重复
06	ENN2	N	1	$j \leftarrow 0$
07	1H LDX	V+N,2	N	<u>M2. 乘数为 0 吗?</u>
08	JXZ	8F	N	若 $v_j = 0$, 则置 $w_{j+m} \leftarrow 0$ 并转 M6
09	ENN1	M	N-Z	<u>M3. 初始化 i。</u> $i \leftarrow 0$
10	ENT3	N,2	N-Z	$(i+j) \leftarrow j$
11	ENTX	0	N-Z	$k \leftarrow 0$
12	2H STX	CARRY	(N-Z)M	<u>M4. 乘和加</u>
13	LDA	U+M,1	(N-Z)M	
14	MUL	V+N,2	(N-Z)M	$rAX \leftarrow u_i \times v_j$
15	SLC	5	(N-Z)M	交换 $rA \leftrightarrow rX$
16	ADD	W,3	(N-Z)M	加 w_{i+j} 到低位一半
17	JNOV	*+2	(N-Z)M	出现上溢?
18	INCX	1	K	若然, 则进位 1 到高位那半
19	ADD	CARRY	(N-Z)M	加 k 到低位那半
20	JNOV	*+2	(N-Z)M	出现上溢?
21	INCX	1	K'	若然, 则进位 1 到高位那半
22	STA	W,3	(N-Z)M	$w_{i+j} \leftarrow t \bmod b$
23	INC1	1	(N-Z)M	<u>M5. 对 i 进行循环。</u> $i \leftarrow i+1$
24	INC3	1	(N-Z)M	$(i+j) \leftarrow (i+j)+1$
25	JIN	2B	(N-Z)M	若 $i < m$, 则以 $rX = \lfloor t/b \rfloor$ 返回 M4

26	8H	STX	$W+M+N, 2$	N	置 $w_{j+m} \leftarrow k$
27		INC2	1	N	M6. 对 j 进行循环。 $j \leftarrow j+1$
28		J2N	1B	N	重复直到 $j = n$ I

程序 M 的执行时间依赖于被乘数 u 中的位数 M ; 乘数 v 中的位数 N ; 乘数中 0 的个数 Z ; 以及在 t 的计算中, 在对乘积的低位那半做加法期间, 出现的进位次数 K 和 K' 。如果对 K 和 K' 取合理的(尽管是稍微悲观的)近似值 $\frac{1}{2}(N-Z)M$, 则我们求得总的运行时间为 $28MN + 4M + 10N + 3 - Z(28M + 3)$ 个周期。如果删去步骤 M2, 则运行时间将是 $28MN + 4M + 7N + 3$ 个周期, 所以仅当乘数中 0 的位置的密度为 $Z/N > 3/(28M + 3)$ 时这个步骤才是有利的。如果乘数完全随机地选择, 则预料比例 Z/N 约仅为 $1/b$, 它是极小的。所以我们结论, 步骤 M2 通常是不值得的, 除非 b 很小。

当 m 和 n 很大时, 算法 M 不是进行乘法最快的方式, 尽管它有简便的优点。4.3.3 小节讨论了一些较快的方法, 甚至当 $m = n = 4$ 时, 也有可能比算法 M 乘得更快。

在这一小节所考虑的最后的算法是长除法, 其中我们要进行 $(m+n)$ 位的整数除以 n 位整数的除法。这里通常的纸和笔的方法包含有人在进行除法时的某些推测和技巧; 我们必须或者从算法中消除这些猜测, 或者要发展某种理论来更仔细地说明它们。

对于通常的长除法过程, 人们立即会想到将一般的问题分成一些更简单的步骤, 每一步骤是 $(n+1)$ 位被除数 u 除以 n 位除数 v 的除法, 其中 $0 \leq u/v < b$; 在每一步骤之后的余数 r 小于 v , 所以在随后的步骤中我们使用 $rb + (\text{被除数的下一位})$ 作为新的 u 。例如, 如果要求 3142 除以 53, 首先把 314 除以 53 得 5 和余数 49; 然后以 53 除 492, 得到 9 和余数 15; 于是我们有商数 59 和余数 15。显然, 这种想法普遍有效, 因而对一个适当的除法算法的探索就归纳为下列问题(图 6):

设 $u = (u_n u_{n-1} \cdots u_1 u_0)_b$ 和 $v = (v_{n-1} \cdots v_1 v_0)_b$ 是 b 进制下的非负整数, 其中 $u/v < b$ 。试求一个算法以确定 $q = \lfloor u/v \rfloor$ 。

$$\begin{array}{r}
 \phantom{u_n u_{n-1} \cdots u_1 u_0} q \\
 v_{n-1} \cdots v_1 v_0 \overline{) u_n u_{n-1} \cdots u_1 u_0} \\
 \underline{\phantom{u_n u_{n-1} \cdots u_1 u_0} \longleftarrow qv \longrightarrow} \\
 \phantom{u_n u_{n-1} \cdots u_1 u_0} \longleftarrow r \longrightarrow
 \end{array}$$

图 6 要求: 一个快速地确定 q 的方式

我们可能注意到, 条件 $u/v < b$ 等价于条件 $u/b < v$, 它和 $\lfloor u/b \rfloor < v$ 相同; 这就是条件 $(u_n u_{n-1} \cdots u_1 u_0)_b < (v_{n-1} v_{n-2} \cdots v_0)_b$ 。其次, 如果我们写 $r = u - qv$, 则 q

是使得 $0 \leq r < v$ 的惟一整数。

这个问题的最明显的解决方法,是根据 u 和 v 的最高位数字,来对 q 做推测。这一方法是否可靠不够明显,但这种算法是值得研究的。因此我们置

$$\hat{q} = \min \left(\left\lfloor \frac{u_n b + u_{n-1}}{v_{n-1}} \right\rfloor, b-1 \right) \quad (2)$$

这个公式说, \hat{q} 是通过把 u 的两位前导数字除以 v 的前导数字得到的;而且若结果是 b 或更大,则以 $b-1$ 代替之。

现在我们研究一个值得注意的事实,即只要 v_{n-1} 适当大,这个值 \hat{q} 总是非常近似于所求的答案 q 。为了分析 \hat{q} 怎样接近于 q ,我们首先要说明 \hat{q} 绝不会太小。

定理 A 在上述记号下 $\hat{q} \geq q$ 。

证明 由于 $q \leq b-1$, 如果 $\hat{q} = b-1$, 则定理肯定是真的。否则我们有 $q = \lfloor (u_n b + u_{n-1}) / v_{n-1} \rfloor$, 因此 $\hat{q} v_{n-1} \geq u_n b + u_{n-1} - v_{n-1} + 1$ 。由此得出

$$\begin{aligned} u - \hat{q}v &\leq u - \hat{q}v_{n-1}b^{n-1} \leq \\ &u_n b^n + \cdots + u_0 - (u_n b^n + u_{n-1} b^{n-1} - v_{n-1} b^{n-1} + b^{n-1}) = \\ &u_{n-2} b^{n-2} + \cdots + u_0 - b^{n-1} + v_{n-1} b^{n-1} < v_{n-1} b^{n-1} \leq v \end{aligned}$$

因为 $u - \hat{q}v < v$, 我们必有 $\hat{q} \geq q$ 。 ■

现在再证明,在实际情况下 \hat{q} 不可能太大于 q , 假设 $\hat{q} \geq q+3$ 。我们有

$$\hat{q} \leq \frac{u_n b + u_{n-1}}{v_{n-1}} = \frac{u_n b^n + u_{n-1} b^{n-1}}{v_{n-1} b^{n-1}} \leq \frac{u}{v_{n-1} b^{n-1}} < \frac{u}{v - b^{n-1}}$$

(情况 $v = b^{n-1}$ 是不可能的, 因为如果 $v = (100 \cdots 0)_b$, 则 $q = q_0$) 而且, 关系 $q > (u/v) - 1$ 意味着

$$3 \leq \hat{q} - q < \frac{u}{v - b^{n-1}} - \frac{u}{v} + 1 = \frac{u}{v} \left(\frac{b^{n-1}}{v - b^{n-1}} \right) + 1$$

因此

$$\frac{u}{v} > 2 \left(\frac{v - b^{n-1}}{b^{n-1}} \right) \geq 2(v_{n-1} - 1)$$

最后, 由于 $b-4 \geq \hat{q}-3 \geq q = \lfloor u/v \rfloor \geq 2(v_{n-1}-1)$, 我们有 $v_{n-1} < \lfloor b/2 \rfloor$ 。这就证明了定理 B:

定理 B 如果 $v_{n-1} \geq \lfloor b/2 \rfloor$, 则 $\hat{q} - 2 \leq q \leq \hat{q}$ 。 ■

这个定理最重要的部分是: 其结论与 b 无关; 不管 b 怎样大, 试验的商 \hat{q} 的误差绝不大于 2。

条件 $v_{n-1} \geq \lfloor b/2 \rfloor$ 非常像一个规格化要求; 事实上, 它恰是在一台二进制计算机上的浮点二进制规格化条件。确保 v_{n-1} 充分大的一个简单方法, 是以 $\lfloor b/(v_{n-1}+1) \rfloor$ 乘 u 和 v ; 这不会改变 u/v 的值, 也不增加 v 中的位数, 而习题 23 证明, 这总是使得 v_{n-1} 的新值充分大。(关于规格化除数的另一个方法, 在习题 28 中讨论。)

根据上述事实,我们已有能力来写求长除法的算法。这个算法在步骤 D3 中对 q 的选择稍微做了改进,它保证 $q = \bar{q}$ 或 $\bar{q} - 1$;事实上,这里对 \bar{q} 所做的改进了的选择几乎总是精确的。

算法 D(非负整数的除法) 给定非负整数 $u = (u_{m+n-1} \cdots u_1 u_0)_b$ 和 $v = (v_{n-1} \cdots v_1 v_0)_b$, 其中 $v_{n-1} \neq 0$ 且 $n > 1$, 我们构造 b 进制的商 $\lfloor u/v \rfloor = (q_m q_{m-1} \cdots q_0)_b$ 和余数 $u \bmod v = (r_{n-1} \cdots r_1 r_0)_b$ 。(当 $n = 1$ 时,应该使用习题 16 的较简单的算法。)

D1. [规格化] 置 $d \leftarrow \lfloor b/(v_{n-1} + 1) \rfloor$, 然后置 $(u_{m+n} u_{m+n-1} \cdots u_1 u_0)_b$ 等于 $(u_{m+n-1} \cdots u_1 u_0)_b$ 乘以 d ; 类似地置 $(v_n v_{n-1} \cdots v_1 v_0)_b$ 等于 $(v_{n-1} \cdots v_1 v_0)_b$ 乘以 d 。(注意,在 u_{n+m-1} 左边引进的新数字位置 u_{n+m} ; 如果 $d = 1$, 则在这一步所要做的一切就是置 $u_{n+m} \leftarrow 0$ 。在一台二进制计算机上,更可取的是选择 d 为 2 的一个乘方,而不用这里所建议的值;任何使 $v_{n-1} \geq \lfloor b/2 \rfloor$ 的 d 值都行。见习题 37c。)

D2. [初始化 j] 置 $j \leftarrow m$ 。(对 j 的循环,即步骤 D2 到 D7,实际上将是 $(u_{j+n} \cdots u_{j+1} u_j)_b$ 除以 $(v_{n-1} \cdots v_1 v_0)_b$ 的除法,以得到商的单个数字 q_j ; 参考图 6。)

D3. [计算 q] 置 $q \leftarrow \lfloor (u_{j+n} b + u_{j+n-1})/v_{n-1} \rfloor$ 并令 r 是余数 $(u_{j+n} b + u_{j+n-1}) \bmod v_{n-1}$ 。现在测试是否 $q = b$ 或 $q v_{n-2} > b r + u_{j+n-2}$; 如果是,则 q 减 1, r 加上 v_{n-1} , 如果 $r < b$ 则重复此测试。(对 v_{n-2} 的这一测试高速度地确定试验值 q 为 1 太大的大多数情况,且消除 q 为 2 太大的所有情况;见习题 19, 20, 21。)

D4. [乘和减] 以 $(u_{j+n} u_{j+n-1} \cdots u_j)_b - q (v_{n-1} \cdots v_1 v_0)_b$ 代替 $(u_{j-n} u_{j+n-1} \cdots u_j)_b$ 。这个计算(类似于算法 M 的步骤 M3, M4 和 M5)由一位数的简单乘法和减法组合而成。数字 $(u_{j+n}, u_{j+n-1}, \cdots, u_j)$ 应保持为正; 如果这个步骤的结果实际上是负的,则 $(u_{j+n}, u_{j+n-1}, \cdots, u_j)_b$ 应保留作为真正值加上 b^{j+1} 。即作为真正值的 b 的补码,并应记住对左边的一个“借位”。

D5. [测试余数] 置 $q_j \leftarrow q$ 。如果步骤 D4 的结果为负,则转向步骤 D6; 否则转向步骤 D7。

D6. [往回加] (如习题 21 所示,需要这一步的概率很小,仅有 $2/b$ 的阶;因此当进行排错时,促使这一步实施的测试数据应当专门设计。) q_j 减 1, 并把 $(0 v_{n-1} \cdots v_1 v_0)_b$ 加到 $(u_{j+n} u_{j+n-1} \cdots u_{j+1} u_j)_b$ 上。(一个进位将出现于 u_{j+n} 的左边,而它应予忽略,因为它同出现于 D4 中的“借位”相抵消。)

D7. [对 j 进行循环] j 减 1。现在,如果 $j \geq 0$, 则返回到 D3。

D8. [不规格化] 现在 $(q_m \cdots q_1 q_0)_b$ 是所求的商,而所求的余数可通过 $(u_{n-1} \cdots u_1 u_0)_b$ 除以 d 得到。■

把算法 D 表示成一个 MIX 程序有若干有趣之点:

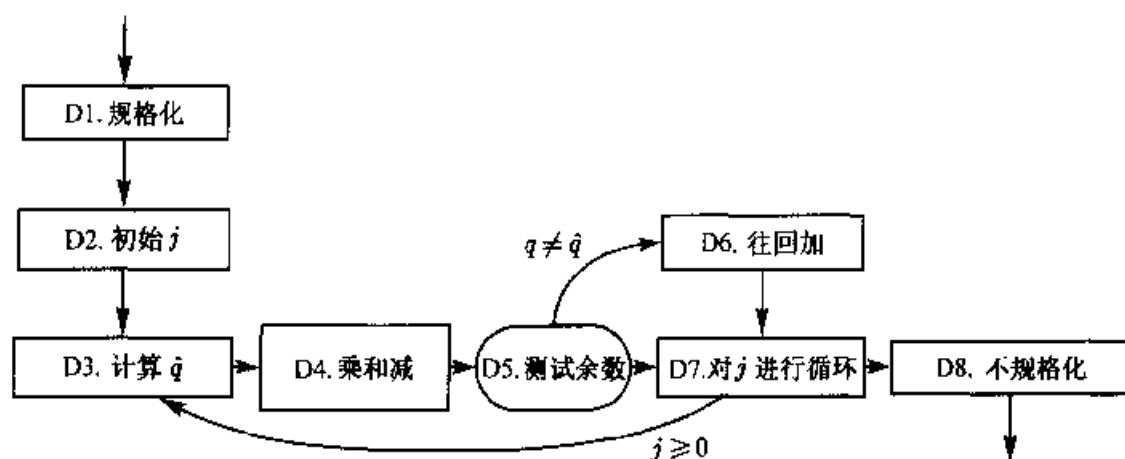


图7 长除法

程序 D(非负整数的除法) 对这个程序的约定和程序 A 的类似: $r11 \equiv i - n$; $r12 \equiv j$, $r13 \equiv i + j$ 。

001	D1	JOV	OFL0	1
...				
039	D2	ENT2	M	1
040		STZ	V + N	1
041	D3	LDA	U + N, 2(1:5)	M + 1
042		LDX	U + N - 1, 2	M + 1
043		DIV	V + N - 1	M + 1
044		JOV	1F	M + 1
045		STA	QHAT	M + 1
046		STX	RHAT	M + 1
047		JMP	2F	M + 1
048	1H	LDX	WM1	
049		LDA	U + N - 1, 2	
050		JMP	4F	
051	3H	LDX	QHAT	E
052		DECX	1	E
053		LDA	RHAT	E
054	4H	STX	QHAT	E
055		ADD	V + N - 1	E

D1. 规格化

(见习题 25)

D2. 初始化。 $j \leftarrow m$ 为方便 D4, 置 $v_n \leftarrow 0$ D3. 计算 q $rAX \leftarrow u_{j+n}b + u_{j+n-1}$ $rA \leftarrow \lfloor rAX / v_{n-1} \rfloor$ 如果商 = b 则跳转 $q \leftarrow rA$ $r \leftarrow u_{j+n}b + u_{j+n-1} -$ $qv_{n-1} = (u_{j+n}b +$
 $u_{j+n-1}) \bmod v_{n-1}$ $rX \leftarrow b - 1$ $rA \leftarrow u_{j+n-1}$ (这里 $u_{j+n} = v_{n-1}$) q 减 1相应地调整 r : $q \leftarrow rX$ $rA \leftarrow r + v_{n-1}$

056		JOV	D4	E	(若 $r \geq b$, 则 $qv_{n-2} < rb$)
057		STA	RHAT	E	$r \leftarrow rA$
058		LDA	QHAT	E	
059	2H	MUL	$V + N - 2$	$M + E + 1$	
060		CMPA	RHAT	$M + E + 1$	测试是否 $qv_{n-2} \leq$
061		JL	D4	$M + E + 1$	$rb + u_{j+n-2}$
062		JG	3B	E	
063		CMPX	$U + N - 2, 2$		
064		JG	3B		否则, q 太大了
065	D4	ENTX	1	$M + 1$	<u>D4. 乘和减</u>
066		ENN1	N	$M + 1$	$i \leftarrow n$
067		ENT3	0, 2	$M + 1$	$(i + j) \leftarrow j$
068	2H	STX	CARRY	$(M + 1)(N + 1)$	(这里 $1 - b < rX \leq +1$)
069		LDAN	$V + N, 1$	$(M + 1)(N + 1)$	
070		MUL	QHAT	$(M + 1)(N + 1)$	$rAX \leftarrow -qv_i$
071		SLC	5	$(M + 1)(N + 1)$	交换 $rA \leftrightarrow rX$
072		ADD	CARRY	$(M + 1)(N + 1)$	加上来自右边数字
073		JNOV	$* + 2$	$(M + 1)(N + 1)$	的贡献, 加 1
074		DECX	1	K	若和 $\leq -b$, 进位 -1
075		ADD	U, 3	$(M + 1)(N + 1)$	加 u_{i+j}
076		ADD	WM1	$(M + 1)(N + 1)$	加 $b - 1$ 以产生 + 号
077		JNOV	$* + 2$	$(M + 1)(N + 1)$	如无溢出, 则进位 -1
078		INCX	1	K'	$rX \equiv$ 进位 +1
079		STA	U, 3	$(M + 1)(N + 1)$	$u_{i+j} \leftarrow rA$ (可能是负 0)
080		INC1	1	$(M + 1)(N + 1)$	
081		INC3	1	$(M + 1)(N + 1)$	
082		J1NP	2B	$(M + 1)(N + 1)$	对 $0 \leq i \leq n$ 重复
083	D5	LDA	QHAT	$M + 1$	<u>D5. 测试余数</u>
084		STA	Q, 2	$M + 1$	置 $q_j \leftarrow q$
085		JXP	D7	$M + 1$	(这里 $rX = 0$ 或 1, 因 $v_n = 0$)
086	D6	DECA	1		<u>D6. 往回加</u>

087	STA	Q,2		置 $q_j \leftarrow q - 1$
088	ENN1	N		$i \leftarrow 0$
089	ENT3	0,2		$(i+j) \leftarrow j$
090	1H	ENTA	0	(这实际上是程序 A)
091	2H	ADD	U,3	
092		ADD	V+N,1	
093		STA	U,3	
094		INC1	1	
095		INC3	1	
096		JNOV	1B	
097		ENTA	1	
098		JINP	2B	
099	D7	DEC2	1	$M+1$ D7. 对 j 进行循环
100		J2NN	D3	$M+1$ 对 $m \geq j \geq 0$ 重复
101	D8	...		(见习题 26) ■

注意步骤 D3 看起来相当复杂的计算和判定,在机器内部是如何容易地处理的。也请注意步骤 D4 的程序类似于程序 M,区别仅在于这里吸收了程序 S 的思想。

程序 D 的运行时间可以通过考察程序中所示的量 M, N, E, K 和 K' 来加以估计(这些量忽略了仅仅以非常小的概率出现的若干情况;例如,我们可以假定行 048~050,063~064 以及步骤 D6 绝不执行)。这里 $M+1$ 是商所占的字的个数; N 是除数占的字的个数; E 是步骤 D3 中向下调整 q 的次数; K 和 K' 是在进行乘法和减法循环中所做的某些“进位”调整的次数。如果我们假定 $K+K'$ 近似于 $(N+1)(M+1)$, 而且 E 近似于 $\frac{1}{2}M$, 则得到近似于

$$30MN + 30N + 89M + 111$$

个周期的总运行时间,如果 $d > 1$,那就再加上 $67N + 235M + 4$ (习题 25 和 26 的程序段已经包括在这些总数中)。当 M 和 N 很大时,这仅仅大约比用程序 M 进行商与除数相乘的时间多出大约 7%。

当进制 b 相对地是小的,因此 b^2 小于计算机的字的的大小时,则多精度除法可以通过不把中间结果的个别数字减少到 $[0, b)$ 的范围中而加速;请见 D. M. Smith, *Math. Comp.* **65** (1996), 157~163。关于对算法 D 的评述请见本小节结尾处的一些习题。

有可能通过使用乘法和加法程序校验除法程序等等的结果,来对多精度算术运

算的程序进行排错。下列类型的测试数据有时是有用的：

$$(t^m - 1)(t^n - 1) = t^{m+n} - t^n - t^m + 1$$

如果 $m < n$, 则这个数有 t 进制展开式

$$\underbrace{(t-1) \cdots (t-1)}_{m-1 \text{ 位}} \quad (t-2) \quad \underbrace{(t-1) \cdots (t-1)}_{n-m \text{ 位}} \quad \underbrace{0 \cdots 0}_{m-1 \text{ 位}} \quad 1$$

例如, $(10^3 - 1)(10^8 - 1) = 99899999001$ 。在程序 D 的情况下, 也有必要找出某些测试例题, 它们使程序的那些很少执行过的部分能得到使用; 因为程序的某些部分, 即使已进行过一百万次随机的测试, 也可能总是得不到测试。(见习题 22。)

现在我们已经看到怎样以带符号量的数来进行运算。现在我们来考虑, 在使用补码记法的计算机上, 对于同样的问题应该采用什么方法。对于 2 的补码和 1 的补码记法, 最好是让进制 b 是字长的一半; 于是对于一台字长是 32 位的计算机, 在上述算法中我们将使用 $b = 2^{31}$ 。一个多精度数, 除了最高位字之外, 所有字的符号位全都是 0, 所以在进行计算机的乘法和除法运算期间不需要进行异常符号校正。事实上, 补码记法的基本意思是要求我们把除最高位以外所有字认为是非负的。例如, 采用 1 个 8 位的字, 则 2 的补码数

$$11011111 \quad 11111110 \quad 1101011$$

(其中符号仅仅对最高位字给出) 应当想成是

$$-2^{21} + (1011111)_2 \cdot 2^{14} + (1111110)_2 \cdot 2^7 + (1101011)_2$$

另一方面, 对 2 的补码记法进行运算的某些二进计算机也提供真正不带符号的算术。例如, 设 x 和 y 是 32 位操作数。一台计算机可能把它们当做在 $-2^{31} < x, y < 2^{31}$ 范围中的 2 的编码的数。如果我们忽略溢出, 则 32 位的和 $(x + y) \bmod 2^{32}$ 在任何一个解释下都是相同的。但当我们改变假定的范围时, 溢出在不同情况下出现。如果计算机允许在不带符号的解释下对进位 $\lfloor (x + y)/2^{32} \rfloor$ 的容易计算, 而且它还提供对于不带符号的 32 位整数的一个完全的 64 位乘积, 则在我们的高精度算法中可以使用 $b = 2^{32}$ 代替 $b = 2^{31}$ 。

当使用补码记法时, 带符号数的加法是比较容易的, 因为 n 位非负整数的加法程序可以用于任意 n 位整数; 正负号仅出现于头一个字中, 所以低位字可以加到一起, 而不论真正的符号是什么(然而当使用 1 的补码记法时, 对于最左的进位必须予以特殊注意; 它必须被加到最低位字上, 而且可能进一步向左传播)。类似地, 我们发现, 带符号数的减法对于补码记法是比较简单的。另一方面, 实现乘和除的最简单办法看来应通过非负量进行, 并且预先做适当的求补操作以确保两个操作数都是非负的。通过使用某些在补码记法下直接处理负数的技巧, 有可能避免这个求补操作, 而且也不难看出在双精度乘法中这可以怎样进行; 但当要求高精度时, 就应该小心, 不要减慢了子程序的内循环。

现在我们回过头来分析程序 A 中出现的量 K , 也就是当把两个 n 位数相加到一起时出现的进位次数。尽管 K 对程序 A 的总运行时间没有影响, 但它确实影响到与程序 A 相对应的处理补码记法的程序的运行时间, 而且对它的分析本身, 作为生成函数的一个重要应用也是有趣的。

假设 u 和 v 是一致分布于范围 $0 \leq u, v < b^n$ 中的 n 位独立随机整数。设 p_{nk} 是在 u 和 v 的加法中恰有 k 个进位出现, 且这些进位之一出现在最高位 (使得 $u + v \geq b^n$) 的概率。类似地, 设 q_{nk} 是恰有 k 个进位出现, 但是在最高位处没有进位的概率。于是不难看出, 对于所有 k 和 n ,

$$\begin{aligned} p_{0k} &= 0, & p_{(n+1)(k-1)} &= \frac{b+1}{2b}p_{nk} + \frac{b-1}{2b}q_{nk} \\ q_{0k} &= \delta_{0k}, & q_{(n+1)k} &= \frac{b-1}{2b}p_{nk} + \frac{b+1}{2b}q_{nk} \end{aligned} \quad (3)$$

当 u_{n-1} 和 v_{n-1} 独立且在 $0 \leq u_{n-1}, v_{n-1} < b$ 的范围内是一致分布的整数时, 由于 $(b-1)/2b$ 是 $u_{n-1} + v_{n-1} \geq b$ 的概率, 而 $(b+1)/2b$ 是 $u_{n-1} + v_{n-1} + 1 \geq b$ 的概率, 故得到上面的结果。

为了得到关于 p_{nk} 和 q_{nk} 的进一步信息, 我们可以建立生成函数

$$P(z, t) = \sum_{k, n} p_{nk} z^k t^n, \quad Q(z, t) = \sum_{k, n} q_{nk} z^k t^n \quad (4)$$

由(3)我们有基本的关系式

$$\begin{aligned} P(z, t) &= zt \left(\frac{b+1}{2b} P(z, t) + \frac{b-1}{2b} Q(z, t) \right) \\ Q(z, t) &= 1 + t \left(\frac{b-1}{2b} P(z, t) + \frac{b+1}{2b} Q(z, t) \right) \end{aligned}$$

这两个方程式很容易对 $P(z, t)$ 和 $Q(z, t)$ 求解; 而且如果命

$$G(z, t) = P(z, t) - Q(z, t) = \sum_n G_n(z) t^n$$

其中 $G_n(z)$ 是当 n 位数相加时总进位数的生成函数, 则我们求得

$$G(z, t) = (b - zt)/p(z, t) \quad (5)$$

其中

$$p(z, t) = b - \frac{1}{2}(1+b)(1+z)t + zt^2$$

注意, $G(1, t) = 1/(1-t)$, 而且这可以用 $G_n(1)$ 必须等于 1 (它是所有可能的概率之和) 的事实进行校验。求(5)对 z 的偏导数, 我们得到

$$\begin{aligned} \frac{\partial G}{\partial z} &= \sum_n G'_n(z) t^n = \frac{-t}{p(z, t)} + \frac{t(b-zt)(b+1-zt)}{2p(z, t)^2} \\ \frac{\partial^2 G}{\partial z^2} &= \sum_n G''_n(z) t^n = \frac{-t^2(b+1-2t)}{p(z, t)^2} + \frac{t^2(b-zt)(b+1-2t)^2}{2p(z, t)^3} \end{aligned}$$

现在置 $z=1$ 并展成部分分式

$$\begin{aligned} \sum_n G'_n(1) t^n &= \frac{t}{2} \left(\frac{1}{(1-t)^2} - \frac{1}{(b-1)(1-t)} + \frac{1}{(b-1)(b-t)} \right) \\ \sum_n G''_n(1) t^n &= \frac{t^2}{2} \left(\frac{1}{(1-t)^3} - \frac{1}{(b-1)^2(1-t)} + \frac{1}{(b-1)^2(b-t)} + \right. \\ &\quad \left. \frac{1}{(b-1)(b-t)^2} \right) \end{aligned}$$

由此得出进位的平均数,即 K 的平均值是

$$G'_n(1) = \frac{1}{2} \left(n - \frac{1}{b-1} \left(1 - \left(\frac{1}{b} \right)^n \right) \right) \quad (6)$$

方差是

$$\begin{aligned} G''_n(1) + G'_n(1) - G'_n(1)^2 = \\ \frac{1}{4} \left(n + \frac{2n}{b-1} - \frac{2b+1}{(b-1)^2} + \frac{2b+2}{(b-1)^2} \left(\frac{1}{b} \right)^n - \right. \\ \left. \frac{1}{(b-1)^2} \left(\frac{1}{b} \right)^{2n} \right) \end{aligned} \quad (7)$$

所以在这些假定之下进位的数目仅仅稍微小于 $\frac{1}{2}n$ 。

历史和文献 这小节所述经典算法的早期历史,做为一个有趣的课题留给读者,这里只追溯它们在计算机上实现的历史。

D. N. Lehmer 和 J. P. Ballantine 讨论了在台式计算器上乘很大的数时以 10^k 作为给定进制的用法,见 AMM 30 (1923), 67~69。

J. von Neumann 和 H. H. Goldstine 在他们的程序设计导论(1947 年发表)中首先讨论了计算机上的双精度算术[J. von Neumann, *Collected Works* 5, 142~151]。上边的定理 A 和 B 是由 D. A. Pope 和 M. L. Stein 给出的[CACM 3 (1960), 652~654],他们的论文也包含有关于双精度程序早期工作的文献目录。A. G. Cox 和 H. A. Luther 讨论了关于选择试验商 q 的其它方法,见 CACM 4 (1961), 353[以 $v_{n-1} + 1$ 除而不以 v_{n-1} 除],另外有 M. L. Stein 也进行了这方面的讨论,见 CACM 7 (1964), 472~474[根据 v_{n-2} 的大小,或者除以 v_{n-1} 或者除以 $v_{n-1} + 1$]; E. V. Krishnamurthy[CACM 8 (1965), 179~181]指出,在后一方法中单精度余数的考查可导致对定理 B 的改进。Krishnamurthy 和 Nandi[CACM 10 (1967), 809~813]提出了一种方法,根据操作数的若干前导数字计算 q ,从而取代算法 D 的规格化和非规格化操作。G. E. Collins 和 D. R. Musser 对 Pope 和 Stein 的算法进行了有趣的分析[*Information Processing Letters* 6 (1977), 151~155]。

已经提出关于除法的若干其它方法:

1)“傅里叶除法”[J. Fourier, *Analyse des Équations Déterminées* (Paris: 1831), § 2.21]。这个方法通常在台式计算器上使用,实质上是通过在每一步中增加除数和被除数的精度,以得到商的每个新的数字。作者所做的某些颇为广泛的检验表明,这个方法肯定是比上述的“除和校正”的技术要差的,但在某些应用中,傅里叶除法也可能是实用的。见 D. H. Lehmer, AMM 33 (1926), 198~206; J. V. Uspensky, *Theory of Equations* (New York: McGraw-Hill, 1948), 159~164。

2)为计算一个数的倒数的“牛顿方法”,广泛用于早期那些还没有单精度除法指令的计算机中。其思想是先找出数 $1/v$ 的某个初始近似值 x_0 ,然后令 $x_{n+1} = 2x_n - vx_n^2$ 。这个方法迅速收敛到 $1/v$,因为 $x_n = (1-\epsilon)/v$ 意味着 $x_{n+1} = (1-\epsilon^2)/v$ 。利

用公式

$$x_{n+1} = x_n + x_n(1 - vx_n) + x_n(1 - vx_n)^2 = \\ x_n(1 + (1 - vx_n)(1 + (1 - vx_n)))$$

能够得到三阶的收敛性,即在每一步以 $O(\epsilon^3)$ 代替 ϵ ,而且类似的公式对第四阶的收敛性成立,等等;见 P. Rabinowitz, CACM 4 (1961), 98。对于非常大的数的计算,如果我们在每一步增加 x_n 的精度,而且使用 4.3.3 小节的快速乘法程序,则牛顿的二阶方法及随后的乘以 u 实际上可以比算法 D 快得多(其细节见算法 4.3.3R)。E. V. Krishnamurthy 讨论了某些有关的迭代方案,见 IEEE Trans. C-19 (1970), 227~231。

3) 有些除法方法也以计算

$$\frac{u}{u+\epsilon} = \frac{u}{v} \left(1 - \left(\frac{\epsilon}{v} \right) + \left(\frac{\epsilon}{v} \right)^2 - \left(\frac{\epsilon}{v} \right)^3 + \cdots \right)$$

为基础。见 H. H. Laughlin, AMM 37 (1930), 287~293。我们在双精度情况(等式 4.2.3-(2))已使用了这一思想。

除了刚才所引用的文献外,下列关于多精度算术的早期论文也是有趣的:A. H. Stroud 和 D. Secrest 描述了利用 1 的补码算术的高精度浮点计算程序 [Comp. J. 6 (1963), 62~66]。B. I. Blum 描述了用于 FORTRAN 程序的扩充精度子程序 [CACM 8 (1965) 318~320]; M. Tienari 和 V. Suokonautio 给出了用于 ALGOL 的扩充精度子程序 [BIT 6 (1966), 332~338]。G. E. Collins 利用链接存储分配技术,漂亮地描述了不限精度的整数算术 [CACM 9 (1966), 578~589]。关于更大得多的运算种类,包括对数和三角函数在内,见 R. P. Brent, ACM Trans. Math. Software 4 (1978), 57~81; D. M. Smith, ACM Trans. Math. Software 17 (1991), 273~283。

人类在计算方面的进步传统地以在历史上一个给定的时间里已经知道的 π 的十进数字的个数来作为度量。4.1 节提到了某些早期的发展;大约在 1719 年, Thomas Fantet de Lagny 已经把 π 计算到 127 位 [Mémoires Acad. Sci. Paris (1719), 135~145; 一个印刷错误影响了第 113 位数字]。在发现了一些更好的公式之后, 1844 年来自汉堡的一位名叫 Zacharias Dase 的著名心算家只用了少于两个月的时间就正确地计算到 200 位 [Crelle 27 (1844), 198]。后来 William Shanks 于 1853 年发表了 π 的 607 位数字,而且继续扩充他的计算直到在 1873 年他得到 707 位数字。[参见 W. Shanks, Contributions to Mathematics (London: 1853); Proc. Royal Soc. London 21 (1873), 318~319; 22 (1873), 45~46; J. C. V. Hoffmann, Zeit. für math. und naturwiss. Unterricht 26 (1895), 261~264。] Shanks 的 707 位的值在很多年里广泛为数学文献所引用。但是在 1945 年 D. F. Ferguson 注意到它从第 528 位开始,包含好些个错误 [Math. Gazette 30 (1946), 89~90]。1949 年 G. Reitwiesner 和他的同事们利用劳动节周末期间在 ENIAC 上花费了 70 小时的计算时间得到 2 037 个正确的数字 [Math. Tables and Other Aids to Comp. 4 (1950), 11~15]。1958 年,在 IBM 704 上运行 100 分钟之后, F. Genuys 达到了 10 000 位数字 [Chiffres 1 (1958), 17~22]; 不久之后, D. Shanks (和 William Shanks 无关) 和 J. W. Wrench, Jr. 在一台

IBM 7090 上计算 8 小时,又花费了 4.5 小时进行检查之后,发表了头 100 000 个数字[Math. Comp. 16 (1962), 76~99]。他们的检查实际上揭露了一个硬件瞬态错误,当计算被重复时它就跑掉了。1973 年法国原子能委员会的 Jean Guilloud 和 Martine Bouyer 在一台 CDC 7600 上花费了接近 24 小时的计算机时间之后,计算出 π 的 100 万位数字[见柴田昭彦(A. Shibata), *Surikagaku* 20 (1982), 65~73]。令人欣喜的是 I. J. Matrix 博士在此 7 年之前就正确地预测,第 100 万位数字将是“5”[Martin Gardner, *New Mathematical Diversions* (Simon and Schuster, 1966), 第 8 章补遗]。Gregory V. Chudnovsky 和 David V. Chudnovsky, 并且也由金田康正(Yasumasa Kanada)和田村良明(Yoshiaki Tamura)独立地在 1989 年越过了 10 亿位数的障碍。1991 年,在其国产并行计算机上运行 250 小时之后,Chudnovsky 们把他们的计算扩充到 20 亿位。[见 Richard Preston, *The New Yorker* 68, 2 (1992 年 3 月 2 日), 36~37。Proc. Nat. Acad. Sci. 86 (1989), 8178~8182 描述了 Chudnovsky 们所使用的新公式。]金田康正和高桥大介(Daisuke Takahashi)使用了两个独立的方法,它们分别要求在有 1 024 个处理单元的日立 SR2201 计算机上运行 29.0 小时和 37.1 小时,在 1997 年 7 月,得到了超过 515 亿位数字,它是我们进入新千年的新记录。

在这一小节里,我们已经把我们的讨论限制于用于计算机程序设计的算术技术。用于算术运算的硬件实现的许多算法也是很有趣的,但是看来它们不能应用于高精度的软件程序;例如,请见 G. W. Reitwiesner, “Binary Arithmetic”, *Advances in Computers* 1 (New York, Academic Press, 1960), 231~308; O. L. MacSorley, *Proc. IRE* 49 (1961), 67~91; G. Metze, *IRE Trans. EC* 11 (1962), 761~764; H. L. Garner, “Number Systems and Arithmetic”, *Advances in Computers* 6 (New York: Academic Press, 1965), 131~194。A. Edelman 在 *SIAM Review* 39 (1997), 54~67 上讨论了在 1994 年的奔腾芯片的除法程序中发现的不出名但很有教益的错误。已经有人研究了对于硬件的加法和乘法运算所能达到的极大执行时间[S. Winograd, *JACM* 12 (1965), 277~285, 14 (1967), 793~802; R. P. Brent, *IEEE Trans. C* 19 (1970), 758~759; R. W. Floyd, *FOCS* 16 (1975), 3~5]。也请参见 4.3.3E 小节。

习 题

1. [42] 通过查阅,比如说,孙子, al-Khwārizmī, al-Uqlidisi, 斐波那契和 Robert Recorde 的著作,研究对于算术的经典算法的早期历史,并尽可能忠实地把他们的方法翻译成更精确的算法记号。

2. [15] 推广算法 A 使它进行“列的加法”,即求 m 个非负的 n 位整数的和(假定 $m \leq b$)。

3. [21] 写出习题 2 中算法的一个 MIX 程序,并估计作为 m 和 n 的函数的运行时间。

4. [M21] 利用 1.2.1 小节所说明的归纳论断的方法,给出算法 A 的正确性的一个形式证明。

5. [21] 算法 A 从右到左地把两个输入相加,但有时数据更易于从左到右存取。试设计一个算法,它产生和算法 A 同样的答案,但它从左到右生成答案的数字,而且如果出现一个进位使得以前的一个值不正确,就回过头改变以前的某些值。(注意:古代印度和阿拉伯的手稿是以这个方

式即从左到右来进行加法的,大概是因为在算盘上习惯于从左到右工作;自右到左的加法算法归功于 al-Uqlidisi 的改进,也许是因为阿拉伯文是从右到左写的。)

►6.[22] 设计一个算法,它从左到右进行加法(像在习题5中那样),但在答案的数字可能受到将来进位影响之前它不存储这个数字,一旦存入,任何答案数字就不再变动了。[提示:记住未曾存入答案中的相继的 $b-1$ 的个数。]例如,在从磁带上由左到右读和写输入和输出数的情况下,或者如果它们出现在一个直接的线性表中时,这类算法是适当的。

7.[M26] 习题5的算法中一个进位会迫使回过头改变部分答案的 k 个数字($k=1,2,\dots,n$)。试给出该算法发现这种进位的平均次数。(假定两个输入数都是独立的,而且一致分布于 0 与 b^n-1 之间。)

8.[M26] 写出习题5中算法的一个 MIX 程序,并且如在上文中的计算那样,以预期的进位数为基础来确定它的平均运行时间。

►9.[21] 推广算法 A 以得到一个算法,这个算法把进制为 b_0, b_1, \dots (从右到左)的混合进制数系中的两个 n 位数相加。因此最低位数字在 0 与 b_0-1 之间,下一位数字在 0 和 b_1-1 之间,等等;参考等式 4.1-(9)。

10.[18] 如果交换行 06 和 07 的指令,则程序 S 是否能适当地工作? 如果交换行 05 和 06 的指令呢?

11.[10] 试设计一个算法,它比较进制 b 的两个非负的 n 位整数 $u = (u_{n-1} \dots u_1 u_0)_b$ 和 $v = (v_{n-1} \dots v_1 v_0)_b$ 以确定是否 $u < v$, $u = v$ 或 $u > v$ 。

12.[16] 算法 S 假定了我们知道两个输入操作数中哪个是更大的;如果这个信息未知,我们可以不管三七二十一地照样实施减法,我们将发现在这个算法的结尾还存在一个额外的“借位”。试设计另一个算法,它将用来(如果在算法 S 的结尾存在一个“借位”)对 $(w_{n-1} \dots w_1 w_0)_b$ 进行求补并从而得到 u 与 v 之差的绝对值。

13.[21] 写出一个 MIX 程序,它以 v 乘 $(u_{n-1} \dots u_1 u_0)_b$, 其中 v 是单精度的数(即 $0 \leq v < b$),产生出答案 $(w_{n-1} \dots w_1 w_0)_b$ 。问需要多少运行时间?

►14.[M22] 利用 1.2.1 小节所说明的归纳论断的方法,给出算法 M 的正确性的形式证明(见习题 4)。

15.[M20] 如果我们希望构造两个 n 位小数的乘积 $(.u_1 u_2 \dots u_n)_b \times (.v_1 v_2 \dots v_n)_b$ 并且仅需得到结果的一个 n 位近似值 $(.w_1 w_2 \dots w_n)_b$, 则可用算法 M 得到一个 $2n$ 位的答案,然后舍入成所希望的近似值。但这包含了两倍于合理精度所需要的工作,因为对于 $i+j > n+2$, 乘积 $u_i v_j$ 对答案的贡献非常小。

如果对于 $i+j > n+2$, 在进行乘法时不计算这些乘积 $u_i v_j$, 而假定它们为 0, 试给出可能出现的极大误差估计。

►16.[20] (短除法) 试设计一个算法,它把一个非负 n 位整数 $(u_{n-1} \dots u_1 u_0)_b$ 除以 v , 其中 v 是单精度数(即 $0 < v < b$), 并产生商 $(w_{n-1} \dots w_1 w_0)_b$ 和余数 r 。

17.[M20] 在图 6 的记号下,假定 $v_{n-1} \geq \lfloor b/2 \rfloor$; 证明如果 $u_n = v_{n-1}$, 则我们必有 $q = b-1$ 或 $b-2$ 。

18.[M20] 在图 6 的记号下,证明如果 $q' = \lfloor (u_n b + u_{n-1}) / (v_{n-1} + 1) \rfloor$, 则 $q' \leq q$ 。

►19.[M21] 在图 6 的记号下,设 q 是对 q 的一个近似值,并命 $r = u_n b + u_{n-1} - q v_{n-1}$ 。假设 $v_{n-1} > 0$, 试证明如果 $q v_{n-2} > b r + u_{n-2}$, 则 $q < q$ 。[提示:通过考察 v_{n-2} 的影响来加强定理 A 的证明。]

20. [M22] 利用习题 19 的记号和假定, 证明如果 $qv_{n-2} \leq br + u_{n-2}$, 则 $\bar{q} = q$ 或 $q = \bar{q} - 1$ 。

► 21. [M23] 证明如果 $v_{n-1} \geq \lfloor b/2 \rfloor$, 而且如果在习题 19 和习题 20 的记号下, $qv_{n-2} \leq br + u_{n-2}$ 但 $\bar{q} \neq q$, 则 $u \bmod v \geq (1 - 2/b)v$ 。(后一事件以近似的概率 $2/b$ 出现, 所以当 b 是一个计算机的字长时, 我们在算法 D 中必须有 $q_i = \bar{q}$, 除非在非常稀少的情况下有所例外。)

► 22. [24] 利用进制 b 为 10 的算法 D, 试找出一个四位数字除以一个三位数字的例子, 对于它来说算法 D 中的步骤 D6 是必要的。

23. [M23] 给定 v 和 b 是整数, 而且 $1 \leq v < b$, 证明总有 $\lfloor b/2 \rfloor \leq v \lfloor b/(v+1) \rfloor < (v+1) \lfloor b/(v+1) \rfloor \leq b$ 。

24. [M20] 利用 4.2.4 小节中说明的前导数字分布定律, 给出算法 D 中 $d-1$ 的概率的一个近似公式(当 $d=1$ 时, 我们可以省略步骤 D1 和 D8 中的大多数计算)。

25. [26] 写出对于步骤 D1 的一个 MIX 程序, 它对完成程序 D 来说是必需的。

26. [21] 写出对于步骤 D8 的一个 MIX 程序, 它对完成程序 D 来说是必需的。

27. [M20] 证明算法 D 的步骤 D8 的开始, 未规格化的数 $(u_{n-1} \cdots u_1 u_0)_b$ 总是 d 的精确的倍数。

28. [M30] (A. Svoboda, *Stroje na Zpracování Informací* 9 (1963), 25 ~ 32) 设 $v = (v_{n-1} \cdots v_1 v_0)_b$ 是任意进制 b 的整数, 其中 $v_{n-1} \neq 0$, 实施下列操作:

N1. 如果 $v_{n-1} < b/2$, 则以 $\lfloor (b+1)/(v_{n-1}+1) \rfloor$ 乘 v 。命此步的结果是 $(v_n v_{n-1} \cdots v_1 v_0)_b$ 。

N2. 如果 $v_{n-1} = 0$, 则置 $v \leftarrow v + (1/b) \lfloor b(b - v_{n-1})/(v_{n-1} + 1) \rfloor v$; 命这个步骤的结果是 $(v_n v_{n-1} \cdots v_0, v_{-1} \cdots)_b$ 。重复步骤 N2 直到 $v_{n-1} \neq 0$ 。

试证步骤 N2 至多被实施三次, 而且我们在这个计算的末尾总有 $v_n = 1, v_{n-1} = 0$ 。

[注: 如果 u 和 v 两者都乘以上述常数, 则我们并不改变商 u/v 的值, 而且这个除数已经转换成形式 $(10v_{n-2} \cdots v_0, v_{-1} v_{-2} v_{-3})_b$ 。这种除数形式是非常方便的, 因为在算法 D 的记法下, 我们可以简单地在步骤 D3 开始处取 $\bar{q} = u_{j+n}$ 作为一个试验的除数, 或当 $(u_{j+n+1}, u_{j+n}) = (1, 0)$ 时取 $\bar{q} = b - 1$ 。]

29. [15] 证明或否定: 在算法 D 的步骤 D7 的开始, 我们总有 $u_{j+n} = 0$ 。

► 30. [22] 如果存储空间很有限, 则在实施这小节的某些算法时, 就可能希望对于输入和输出两者使用相同的存储单元。在执行算法 A 或 S 时, 是否有可能把 $w_0, w_1, \cdots, u_{n-1}$ 分别存于和 u_0, \cdots, u_{n-1} 或 v_0, \cdots, v_{n-1} 相同的单元? 是否有可能让 q_0, \cdots, q_m 同算法 D 中的 u_n, \cdots, u_{n+m} 占有相同的单元? 在算法 M 中的输入和输出之间, 是否有任何允许的存储单元的重叠?

31. [28] 假设 $b=3$, 而且 $u = (u_{m+n-1} \cdots u_1 u_0)_3, v = (v_{n-1} \cdots v_1 v_0)_3$ 是在平衡的三进制记号下的整数(参考 4.1 节), $v_{n-1} \neq 0$ 。试设计一个长除法的算法, 它以 v 除 u , 得到一个绝对值不超过 $\frac{1}{2}|v|$ 的余数。试找出一个算法, 如果把它加到一台平衡的三进制计算机的算术线路中去, 则它将是有效的。

32. [M40] 假设 $b=2i, u$ 和 v 是以虚 4 数系表达的复数。试设计一种算法, 它以 v 除 u , 也许得到一个某种适当的余数, 并比较它们的效能。

33. [M40] 设计一个取平方根的算法, 它类似于算法 D, 也类似于求平方根的手算方法。

34. [40] 编出用来对任意整数做四种算术运算的一组计算机程序, 对于整数的大小不附加任何限制, 只是假定不应超过计算机总的存储容量。(利用链接存储分配, 以便在寻找存放结果的单元时不浪费时间。)

35. [40] 编制出“双精度浮点”算术的一组计算机子程序, 利用余量 0, 进制 $b, 9$ 位浮点数表

示,其中 b 是计算机字长,而且允许用一个全字来作为指数(于是每个浮点数表示成内存中的 10 个字,而且所有的调整都通过移动全字长而不是在这些字之内的移位来进行)。

36. [M25] 只使用多精度加法,减法以及除以小数的除法,给定 ϕ 的一个适当精确的近似值,试说明怎样把 $\ln \phi$ 计算到高精度。

► 37. [20] (E. Salamin) 说明当 d 是一台二进计算机上的 2 的次幂时,如何避免算法 D 的规格化和非规格化步骤,而无须改变由该算法所计算的试验商数字的序列。(如果步骤 D1 的规格化没有实行,在步骤 D3 中如何能计算 \bar{q} ?)

38. [M35] 假设 u 和 v 是在范围 $0 \leq u, v < 2^n$ 中的整数。试设计通过进行 $O(n)$ 次加法,减法和 $(n+2)$ 个二进位的数的比较这些运算来计算几何均值 $\left[\sqrt{uv} + \frac{1}{2} \right]$ 的一个方法。[提示:利用一个“流水线”把乘法和求平方根的经典方法组合在一起。]

39. [25] (D. Bailey, P. Borwein 和 S. Pluffe, 1996) 通过利用等式

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)$$

和对于 $O(\log n)$ 个二进位的整数进行 $O(n \log n)$ 个算术运算。试说明怎样无须知道前 $n-1$ 个二进位就可计算出 π 的二进表示的第 n 位。(假设 π 的二进数字没有令人惊奇地长的连续的全 0 或全 1 的长串。)

40. [M24] 当我们知道余数将为零时,我们有时要以 v 来除 u 。说明如果 u 是一个 $2n$ 位数而 v 是一个 n 位数且 $u \bmod v = 0$ 时,如果从左到右地计算一半的商,又从右到左地计算另外一半,则我们可以节省算法 D 的大约 75% 的工作。

► 41. [M26] 高精度算术的许多应用要求重复计算对一个固定 n 位数 w 的取模,其中 w 和基数 b 互素。通过使用由 Peter L. Montgomery 给出的一个技巧 [Math. Comp. 44 (1985), 519 ~ 521], 我们可以加速这样的计算。这个技巧通过实质上从右到左地工作而不是从左到右,而使求余数的过程流水线化。

a) 给定 $u = \pm(u_{m+n-1} \cdots u_1 u_0)_b$, $w = (w_{n-1} \cdots w_1 w_0)_b$ 和一个数 w' 使得 $w_0 w' \bmod b = 1$, 说明如何计算 $v = \pm(v_{n-1} \cdots v_1 v_0)_b$ 使得 $b^n v \bmod w = u \bmod w$ 。

b) 给定 n 位带符号的整数 u, v, w 且 $|u|, |v| < w$, 并且像在 a) 中那样给定 w' , 说明如何计算一个 n 位整数 t 使得 $|t| < w$ 且 $b^n t \equiv uv \pmod{w}$ 。

c) a) 和 b) 的算法如何方便模 w 算术的实现?

42. [HM35] 给定 m 和 b , 令 P_{nk} 是当 u_1, \dots, u_m 是 b 进制下随机 n 位整数时 $\lfloor (u_1 + \cdots + u_m)/b^n \rfloor = k$ 的概率。(这是习题 2 的列加法算法中的 w_n 分布。)证明 $P_{nk} = \frac{1}{m!} \langle \binom{m}{k} \rangle + O(b^{-n})$, 其中 $\langle \binom{m}{k} \rangle$ 是一个欧拉数(参见 5.1.3 小节)。

► 43. [22] 在数字化图像中灰度的阴影或颜色组成通常被表示为在 $[0, 255]$ 范围中的 8 个二进制的数 u , 表示分数 $u/255$ 。给定两个这样的分数 $u/255$ 和 $v/255$, 图形算法通常需要计算它们近似的乘积 $w/255$, 其中 w 是最接近于 $uv/255$ 的整数。证明 w 可以从有效的公式

$$t = uv + 128, \quad w = \lfloor (t/256 + t)/256 \rfloor$$

得到。

*4.3.2 模算术

对于很大的数进行算术运算,可以利用另一个有趣的方案,它是以数论上某些简单原理为基础的。这一思想是假定有若干个不含公因子的模 m_1, m_2, \dots, m_r , 并且通过“余数” $u \bmod m_1, u \bmod m_2, \dots, u \bmod m_r$ 间接地计算 u , 而不是直接地对数 u 进行计算。

为了记法上的方便,这一小节自始至终都设

$$u_1 = u \bmod m_1, \quad u_2 = u \bmod m_2, \quad \dots, \quad u_r = u \bmod m_r \quad (1)$$

借助于除法容易从一个整数 u 计算 (u_1, u_2, \dots, u_r) , 而且——更为重要的是注意到——在这个过程中不损失信息, 因为假若我们知道 u 不太大, 我们总是可以从 (u_1, u_2, \dots, u_r) 重新计算 u 。例如, 如果 $0 \leq u < v \leq 1000$, 则不可能有 $(u \bmod 7, u \bmod 11, u \bmod 13)$ 等于 $(v \bmod 7, v \bmod 11, v \bmod 13)$ 。这是下面所述的“中国剩余定理”^{*}的一个推论。

因此我们可以认为 (u_1, u_2, \dots, u_r) 是整数 u 的一种新型的计算机内部表示, 即整数 u 的一个“模表示”。

模表示的优点是, 加法、减法和乘法都非常简单:

$$(u_1, \dots, u_r) + (v_1, \dots, v_r) = ((u_1 + v_1) \bmod m_1, \dots, (u_r + v_r) \bmod m_r) \quad (2)$$

$$(u_1, \dots, u_r) - (v_1, \dots, v_r) = ((u_1 - v_1) \bmod m_1, \dots, (u_r - v_r) \bmod m_r) \quad (3)$$

$$(u_1, \dots, u_r) \times (v_1, \dots, v_r) = ((u_1 \times v_1) \bmod m_1, \dots, (u_r \times v_r) \bmod m_r) \quad (4)$$

例如, 为导出(4)我们需要对于每个模 m_j 证明

$$uv \bmod m_j = (u \bmod m_j)(v \bmod m_j) \bmod m_j$$

但这是初等数论的一个基本事实: 当且仅当 $x \equiv y \pmod{m_j}$ 时, $x \bmod m_j = y \bmod m_j$; 而且, 如果 $x \equiv x'$ 且 $y \equiv y'$, 则 $xy \equiv x'y' \pmod{m_j}$; 因此 $(u \bmod m_j)(v \bmod m_j) \equiv uv \pmod{m_j}$ 。

模表示的缺点, 是我们不能容易地测定 (u_1, \dots, u_r) 是否大于 (v_1, \dots, v_r) 。它也难于测定一个加法、减法或乘法的结果是否出现溢出, 而且更难于实施除法。当经常需要把这样的操作同加法、减法和乘法相联系时, 仅当有模表示的快速来回转换手段可资利用时, 使用模算术才是正当的。因此, 模与位置记数法之间的转换是这一节中我们所感兴趣的主要课题之一。

利用(2), (3)和(4)进行加法、减法和乘法的过程称为剩余算术或模算术。可以由模算术处理的数的范围等于 $m = m_1 m_2 \dots m_r$, 即诸模之乘积; 而且当 $r \approx n$ 时如果每个 m_j 都接近于我们的计算机字的大小, 我们可以处理 n 位的数。因此我们看到, 利用模算术进行加、减或乘 n 位数字的数所需要的时间, 实质上与 n 成正比(不

* 即秦九韶定理。——译者注

计模表示来回转换的时间)。当考虑加法和减法时,这全然没有好处。但是对于乘法,却可能有相当大的好处,因为 4.3.1 小节的通常的方法要求与 n^2 成正比的执行时间。

而且,在一台允许许多运算同时进行的计算机上,模算术甚至对于加法和减法都可以有相当重要的优点;相对于不同模数的运算可以同时完成,所以我们在速度方面就得到了实质性的提高。而通过上一节中所讨论的通常技术,却不可能实现这类减少执行时间的效果,因为必须考虑到进位的传播。也许在某一天高度并行的计算机将使得同时操作非常普遍,以致当需要对一个要求高精度的问题迅速给出答案时,模算术在“实时”计算方面将具有巨大的重要性。(对于高度并行的计算机,通常更喜欢同时运行 k 个独立的程序,而不是以 k 倍的速度运行一单个程序,因为后一种情况是更复杂的,也不能更有效地利用机器。“实时”计算却属例外,这使得模算术固有的并行性更有意义。)

现在我们来考察奠定数的模表示基础的基本事实:

定理 C(中国剩余定理) 设 m_1, m_2, \dots, m_r 是两两互素的正整数,即

$$m_j \perp m_k \quad \text{当 } j \neq k \text{ 时} \quad (5)$$

设 $m = m_1 m_2 \cdots m_r$, 并设 a, u_1, u_2, \dots, u_r 是整数。于是恰有一个整数 u , 它满足条件

$$a \leq u < a + m, \quad \text{及} \quad u \equiv u_j \pmod{m_j}, \quad \text{对于 } 1 \leq j \leq r \quad (6)$$

证明 如果对于 $1 \leq j \leq r, u \equiv v \pmod{m_j}$, 则对于所有 $j, u - v$ 是 m_j 的倍数, 所以(5)意味着 $u - v$ 是 $m = m_1 m_2 \cdots m_r$ 的倍数。这证明了(6)至多有一个解, 为完成证明, 我们还须证明至少有一个解, 而这可以用两个简单方法得出:

方法 1(“非构造性”证明) 当 u 跑遍 m 个不同值 $a \leq u < a + m$ 时, r 元组 $(u \bmod m_1, \dots, u \bmod m_r)$ 必然也跑遍 m 个不同的值, 因为(6)至多有一个解。但恰巧有 $m_1 m_2 \cdots m_r$ 种可能的 r 元组 (v_1, \dots, v_r) 使得 $0 \leq v_j < m_j$ 。因此每个 r 元组必定恰出现一次, 而且必然有 n 的某值使 $(u \bmod m_1, \dots, u \bmod m_r) = (u_1, \dots, u_r)$ 。

方法 2(“构造性”的证明) 我们可以找数 $M_j, 1 \leq j \leq r$, 使得

$$M_j \equiv 1 \pmod{m_j} \quad \text{和} \quad M_j \equiv 0 \pmod{m_k} \quad \text{对于 } k \neq j \quad (7)$$

这是由于(5)意味着 m_j 与 m/m_j 互素而得出的, 所以通过欧拉定理(习题 1.2.4-28)可以取

$$M_j = (m/m_j)^{\varphi(m_j)} \quad (8)$$

现在数

$$u = a + ((u_1 M_1 + u_2 M_2 + \cdots + u_r M_r - a) \bmod m) \quad (9)$$

满足(6)的所有条件。 ■

中国数学家孙子指出了这一定理的一个非常特殊的情况, 他给出了一个称为“太元”(“大大推广”)的规则, 他写作的日期很难确定, 一般认为是在公元 280 和 473 年之间。中世纪印度的数学家通过他们的古塔卡(Kuttaka)方法(请见 4.5.2 小

节),进一步发展了这些技术。但是是秦九韶在他的《九章算术》(1247)中第一个表述了该定理并以适当的一般性证明了它。后一工作也考虑了像在习题3那样诸模有公因子的情况。[参见 J. Needham, *Science and Civilization in China* 3 (Cambridge University Press, 1959), 33 ~ 34, 119 ~ 120; 李俨(Y. Li)和杜石然(S. Du), *Chinese Mathematics* (Oxford: Clarendon, 1987), 92 ~ 94, 105, 161 ~ 166; 沈康身(K. Shen), *Archive for History of Exact Sciences* 38 (1988), 285 ~ 305。]关于这个理论的许多早期的贡献已经由 L. E. Dickson 做了概述 [*History of the Theory of Numbers* 2 (Carnegie Inst. of Washington, 1920, 57 ~ 64)]。

作为定理 C 的一个推论,我们可以在任何相继的 $m = m_1 m_2 \cdots m_r$ 个整数的区间中,使用数的模表示。例如,我们可以在(6)中取 $a = 0$,而且仅仅处理小于 m 的非负整数。另一方面,当做加法和减法时,和做乘法一样,通常最方便的是假定所有模数 m_1, m_2, \dots, m_r 都是奇数,使得 $m = m_1 m_2 \cdots m_r$ 也是奇数,这样可以处理范围

$$-\frac{m}{2} < u < \frac{m}{2} \quad (10)$$

中的整数,这一范围关于 0 是完全对称的。

为了实施在(2),(3)和(4)中引出的基本操作,当 $0 \leq u_j, v_j < m_j$ 时,我们需要计算 $(u_j + v_j) \bmod m_j$, $(u_j - v_j) \bmod m_j$ 以及 $u_j v_j \bmod m_j$ 。如果 m_j 是一个单精度的数,则通过先做一个乘法再做一个除法运算,来形成 $u_j v_j \bmod m_j$ 是最方便的。对于加法和减法,情况更简单些,因为不需要做除法;可以方便地使用下列公式:

$$(u_j + v_j) \bmod m_j = u_j + v_j - m_j [u_j + v_j \geq m_j] \quad (11)$$

$$(u_j - v_j) \bmod m_j = u_j - v_j + m_j [u_j < v_j] \quad (12)$$

(参考 3.2.1.1 小节)。由于我们要 m 尽可能大,最容易的是命 m_1 为一个计算机字所能容纳的最大奇数,命 m_2 为与 m_1 互素的小于 m_1 的最大奇数,命 m_3 是与 m_1 和 m_2 两者皆互素的小于 m_2 的最大奇数,如此等等,直至找到足够的 m_j 来给出所希望的范围 m 。4.5.2 小节讨论了为确定两个整数是否互素的一些有效的方法。

作为一个简单的例子,假设我们有字长仅为两位数的十进计算机,所以字的大小为 100。则上一段所述过程将给出

$$m_1 = 99, m_2 = 97, m_3 = 95, m_4 = 91, m_5 = 89, m_6 = 83 \quad (13)$$

等等。

在二进计算机上,有时希望以一种不同的方式来选择 m_j ,即选择

$$m_j = 2^{e_j} - 1 \quad (14)$$

换言之,每个模数比 2 的一个乘方小 1。 m_j 的这样一种选择,通常使得基本算术运算更为简单,因为像在 1 的补码算术中那样,对模 $2^{e_j} - 1$ 进行运算是比较容易的。当按照这个策略来选定模数时,稍微放松条件 $0 \leq u_j < m_j$ 是有帮助的,这样我们仅要求

$$0 \leq u_j < 2^{e_j}, \quad u_j \equiv u \pmod{2^{e_j} - 1} \quad (15)$$

于是,也允许值 $u_j = m_j = 2^{e_j} - 1$ 作为代替 $u_j = 0$ 的另一个可供选用的方案,因为这不影响定理 C 的正确性,而且它意味着 u_j 可以是任意的 e_j 位二进制数。在这一假定之下,模 m_j 的加法和乘法运算就变成:

$$u_j \oplus v_j = ((u_j + v_j) \bmod 2^{e_j}) + [u_j + v_j \geq 2^{e_j}] \quad (16)$$

$$u_j \otimes v_j = (u_j v_j \bmod 2^{e_j}) \oplus \lfloor u_j v_j / 2^{e_j} \rfloor \quad (17)$$

(这里的 \oplus 和 \otimes 指的是,当利用(15)的约定进行加法和乘法时,对于 (u_1, \dots, u_r) 和 (v_1, \dots, v_r) 的个别分量分别要做的运算。)等式(12)对于减法仍然有效,或者我们可以用

$$u_j \ominus v_j = ((u_j - v_j) \bmod 2^{e_j}) - [u_j < v_j] \quad (18)$$

即使当 m_j 大于计算机字长时,这些运算仍可有效地进行;因为计算一个正数模 2 的某次幂的余数,或者除一个数的 2 的次幂,是一件容易的事情。在(17)中,像在习题 3.2.1.1-8 中所讨论的那样,我们有乘积的“高半部”和“低半部”的和。

如果使用形如 $2^e - 1$ 的模,则我们必须知道在什么条件下数 $2^e - 1$ 与 $2^f - 1$ 互素。幸而,有一个非常简单的规则

$$\gcd(2^e - 1, 2^f - 1) = 2^{\gcd(e, f)} - 1 \quad (19)$$

它特别指出,当且仅当 e 和 f 互素时, $2^e - 1$ 与 $2^f - 1$ 互素。等式(19)是由欧几里得算法和等式

$$(2^e - 1) \bmod (2^f - 1) = 2^{e \bmod f} - 1 \quad (20)$$

得出的(见习题 6)。在字长为 2^{32} 的一台计算机上,我们因而就可以选择比如说 $m_1 = 2^{32} - 1$, $m_2 = 2^{31} - 1$, $m_3 = 2^{29} - 1$, $m_4 = 2^{27} - 1$, $m_5 = 2^{25} - 1$;这将允许我们有效地进行在大小为 $m_1 m_2 m_3 m_4 m_5 > 2^{143}$ 范围内整数的加法、减法和乘法。

我们已经发现,模表示的来回转换运算是非常重要的。如果我们给出一数 u , 则简单地通过以 m_1, \dots, m_r 来除 u 并保存余数,就可以得到它的模表示 (u_1, \dots, u_r) 。如果 $u = (v_m v_{m-1} \dots v_0)_b$, 则一个可能更有吸引力的过程是利用模算术计算多项式

$$(\dots(v_m b + v_{m-1})b + \dots)b + v_0$$

的值。当 $b=2$ 且当模 m_j 有特殊形式 $2^{e_j} - 1$ 时,这两个方法都归结为一个十分简单的过程:考虑把有 e_j 位的块组合在一起形成的 u 的二进表示

$$u = a_t A^t + a_{t-1} A^{t-1} + \dots + a_1 A + a_0 \quad (21)$$

其中 $A = 2^{e_j}$, 而且对于 $0 \leq k \leq t, 0 \leq a_k \leq 2^{e_j}$ 。于是

$$u \equiv a_t + a_{t-1} + \dots + a_1 + a_0 \pmod{2^{e_j} - 1} \quad (22)$$

因为 $A \equiv 1$, 因此我们可以利用(16)把诸 e_j 位的数相加: $a_t \oplus \dots \oplus a_1 \oplus a_0$, 得到 u_j 。这一过程类似于熟知的“舍 9 法”操作,该法在以十进数表示 u 时用来确定 $u \bmod 9$ 。

从模表示转换成位置记数表示要稍微困难些。在这方面,注意一下计算的研究如何改变我们对于数学证明的观点,是有趣的:定理 C 告诉我们,从 (u_1, \dots, u_r) 转

换成 u 是可能的,而且给出了两个证明。我们所考虑的头一个证明是经典的,它仅仅依赖于一些非常简单的概念,即以下的事实:

i) 任何是 m_1, m_2, \dots 和 m_r 的倍数的数,当诸 m_j 两两互素时,必是 $m_1 m_2 \cdots m_r$ 的倍数;

ii) 如果 m 只鸽子放入 m 个鸽巢当中而且没有两只鸽子同在一个鸽巢中,则在每个鸽巢中必然有一只鸽子。

按照数学的美学传统思想,这毫无疑问是定理 C 的最漂亮的证明;但从计算观点看,它是毫无价值的!这等于说,“试验 $u = a, a + 1, \dots$ 直到你发现一个值,使得 $u \equiv u_1 \pmod{m_1}, \dots, u \equiv u_r \pmod{m_r}$ 。”

定理 C 的第二个证明是更为明显的;它说明怎样计算 r 个新的常数 M_1, \dots, M_r , 并且借助于这些常数通过公式(9)而得到解。这个证明使用了更复杂的一些概念(例如,欧拉定理),但从计算的观点来看,它是更为令人满意的,因为常数 M_1, \dots, M_r 仅仅需要确定一次。另一方面,通过等式(8)确定 M_j 肯定不是轻而易举的,因为一般说来,欧拉 φ 函数的计算要求把 M_j 分解成素因子的乘方。比起使用(8),有好得多的方法来计算 M_j 。在这方面我们可以再次看到数学的优美性和计算的有效性之间的区别。但即使我们通过最好的方法来求 M_j , 我们仍然受阻于这样一个事实,即 M_j 是巨大的数 m/m_j 的一个倍数。因此,(9)迫使我们进行大量高精度的计算,而这样的计算恰恰是我们首先希望通过模算术来加以避免的。

所以,如果打算有一个从 (u_1, \dots, u_r) 变换成 u 的真正能用的方法,我们需要有定理 C 的甚至更好的证明。这样的方法是 H. L. Garner 于 1958 年提出的;它可以利用对于 $1 \leq i < j \leq r$ 的 $\binom{r}{2}$ 个常数 c_{ij} 进行,其中

$$c_{ij} m_i \equiv 1 \pmod{m_j} \quad (23)$$

用欧几里得算法容易计算这些常数,因为对于任何给定的 i 和 j , 算法 4.5.2X 确定 a 和 b , 使得 $am_i + bm_j = \gcd(m_i, m_j) = 1$, 而且我们可以取 $c_{ij} = a$ 。当模有特殊形式 $2^e - 1$ 时,习题 6 中给出了确定 c_{ij} 的一个简单方法。

一旦确定了满足(23)的 c_{ij} , 我们即可置

$$\begin{aligned} v_1 &\leftarrow u_1 \bmod m_1 \\ v_2 &\leftarrow (u_2 - v_1) c_{12} \bmod m_2 \\ v_3 &\leftarrow ((u_3 - v_1) c_{13} - v_2) c_{23} \bmod m_3 \\ &\vdots \\ v_r &\leftarrow (\cdots ((u_r - v_1) c_{1r} - v_2) c_{2r} - \cdots - v_{r-1}) c_{(r-1)r} \bmod m_r \end{aligned} \quad (24)$$

于是

$$u = v_r m_{r-1} \cdots m_2 m_1 + \cdots + v_3 m_2 m_1 + v_2 m_1 + v_1 \quad (25)$$

是满足条件

$$0 \leq u < m, \quad u \equiv u_j \pmod{m_j} \quad \text{对于 } 1 \leq j \leq r \quad (26)$$

的数。(见习题8;习题7给出了不包含这样多的辅助常数的重写(24)的另一个方法。)等式(25)是 u 的一个混合进制表示。利用4.4节的方法它可以换成二进或十进的记号。如果 $0 \leq u < m$ 不是所希望的范围,则在转换过程之后,可以增加或减去 m 的一个适当倍数。

(24)中所示的计算的优点是 v_j 的计算可以仅使用算术模 m_j 来完成,它已经构造到模算术的算法之中。而且,(24)允许并行的计算:我们可以由 $(v_1, \dots, v_r) \leftarrow (u_1 \bmod m_1, \dots, u_r \bmod m_r)$ 开始,然后在 $1 \leq j < r$ 的时间 j , 对于 $j < k \leq r$ 同时置 $v_k \leftarrow (v_k - v_j) c_{jk} \bmod m_k$ 。A. S. Fraenkel 讨论了计算混合进制表示,并允许类似并行性可能性的另一种方法,见 *Proc. ACM Nat. Conf.* **19** (Philadelphia:1964), E1.4。

重要的是注意混合进制表示(25)足以用来比较两个模数的量。因为如果我们知道 $0 \leq u < m$ 和 $0 \leq u' < m$, 则通过首先转换成 (v_1, \dots, v_r) 和 (v'_1, \dots, v'_r) , 然后按照字典序测试是否 $v_r < v'_r$, 或者是否 $v_r = v'_r$ 且 $v_{r-1} < v'_{r-1}$ 等等, 就可以知道是否 $u < u'$ 。如果我们仅仅想要知道 (u_1, \dots, u_r) 是否小于 (u'_1, \dots, u'_r) , 那就不必把所有的数都转换成二进或十进记号。

比较两个数,或者判定一个模数是否为负的操作,在直观上是非常简单的,所以我们希望有比转换成混合进制更为容易的方法来进行测试。但是下列定理表明,找到一个好得多的方法希望甚小,因为一个模数的范围实质上还依赖于所有剩余 (u_1, \dots, u_r) 的所有位。

定理 S (Nicholas Szabó, 1961) 借助于上面的记号,假定 $m_1 < \sqrt{m}$, 并设 L 是范围

$$m_1 \leq L \leq m - m_1 \quad (27)$$

中的任何值,设 g 是使得集合 $\{g(0), g(1), \dots, g(m_1 - 1)\}$ 包含少于 m_1 个值的任何函数,则有数 u 和 v 使得

$$g(u \bmod m_1) = g(v \bmod m_1), u \bmod m_j = v \bmod m_j, \text{ 对于 } 2 \leq j \leq r \quad (28)$$

$$0 \leq u < L \leq v < m \quad (29)$$

证明 由假定,必然存在 $u \neq v$ 满足(28),因为 g 必须对两个不同的剩余取相同的值。设 (u, v) 是 $0 \leq u < v < m$ 的满足(28)的一对值,其中 u 是一个极小值。由于 $u' = u - m_1$ 和 $v' = v - m_1$ 也满足(28),所以由 u 的极小性必有 $u' < 0$ 。因此 $u < m_1 \leq L$; 而且如果(29)不成立,则我们必有 $v < L$ 。但是 $v > u$, 而且 $v - u$ 是 $m_2 \cdots m_r = m/m_1$ 的倍数,所以 $v \geq v - u \geq m/m_1 > m_1$ 。因此如果(29)对于 (u, v) 不成立,则对于数偶 $(u'', v'') = (v - m_1, u + m - m_1)$ 将成立。 ■

当然,代替 m_1 , 对任何 m_i 都可以证明一个类似的结果;而且也可以以条件 “ $a \leq u < a + L \leq v < a + m$ ” 代替(29),只须在证明过程中做一些小的改变即可。因此,定理 S 表明,许多简单的函数不可能用来确定一个模数的范围。

现在重复这一小节中所讨论的主要论点:模算术对于一些应用可能有相当大的优点,在这样一些应用中,居支配地位的计算涉及同加法和减法相结合的大整数的精确乘法(或乘方),但很少需要除法或数的比较,或者测试中间结果是否“溢出”范围。(重要的是不要忘记后一个限制;对于测试溢出,有些方法可以利用,如同在习题 12 中那样,但是一般说来它们都如此之复杂,以致把模算术的优点化为乌有。)H. Takahas 和 Y. Ishibashi 讨论了关于模计算的若干应用,见 *Information Proc. in Japan* **1** (1961), 28~42。

这种应用的一个例子是有理系数线性方程组的精确解。由于种种原因,在这种情况下最好假定模数 m_1, m_2, \dots, m_r 都是素数;这些线性方程可以独立地模每个 m_j 来求解。I. Borosh 和 A. S. Fraenkel 给出了这个过程的详细讨论 [*Math. Comp.* **20** (1966), 107~112], 而且由 A. S. Fraenkel 和 D. Loewenthal 做了进一步的改进 [*J. Res. National Bureau of Standards* **75B** (1971), 67~75]。借助于他们的方法,在一台 CDC 1604 计算机上,用了不到 20 分钟的运行时间就精确地求得了有 120 个未知数的 111 个联立线性方程组的 9 个独立解。当系数矩阵是病态时,同样的过程对于解浮点系数的联立线性方程组也不无价值。模技术(把给定的浮点系数当做精确的有理数)给出了求得真正答案的方法,而这个方法所需要的时间比其它已知能产生可靠近似答案的方法都要少! [关于这个方法的进一步发展,见 M. T. McClellan, *JACM* **20** (1973), 563~588; 关于它的局限性讨论,亦见 E. H. Bareiss, *J. Inst. Math. and Appl.* **10** (1972), 68~104。]

涉及模算术的公开的著作大多数是面向硬件设计的,因为模算术的无进位性质从高速运算的观点来看是有吸引力的。这一思想首先是 A. Svoboda 和 M. Valach 在捷克斯洛伐克杂志 *Stroje na Zpracování Informací* (*Information Processing Machines*) **3** (1955), 247~295 上发表的;而后 H. L. Garner 独立地提出了它 [*IRE Trans. EC-8* (1959), 140~147]。A. S. Fraenkel 提议使用形如 $2^k - 1$ 的模数 [*JACM* **8** (1961), 87~96], A. Schönhage 揭示了使用这样的模数的若干优点 [*Computing* **1** (1966), 182~196]。关于这个课题的另外的信息和一个广泛的文献目录,见 N. S. Szabó 和 R. I. Tanaka 的著作 *Residue Arithmetic and its Applications to Computer Technology* (New York: McGraw-Hill, 1967)。1968 年由 I. Y. Akushsky 和 D. I. Yuditsky 所著的一本俄文书中,包括有关于复数模的一章 [参考 *Rev. Roumaine de Math. Pures et Appl.* **15** (1970), 159~160]。

关于模算术的进一步讨论可以在 4.3.3B 小节中找到。

The notice-board had said he was in Room 423,
but the numbering system, nominally consecutive,
seemed to have been applied on a plan that could only
have been the work of a lunatic or a mathematician

通知栏上说他在 423 号房间,
但是这个名义上连续的计数系统,

似乎一直用于只可能是一个疯子或数学家所做的一个计划上。

—ROBERT BARNARD, *The Case of the Missing Brontë* (1983)

习 题

1. [20] 求满足条件 $u \bmod 7 = 1, u \bmod 11 = 6, u \bmod 13 = 5$, 且 $0 \leq u < 1000$ 的所有整数 u 。

2. [M20] 如果允许 a, u_1, u_2, \dots, u_r 和 m 是任意实数 (不仅仅是整数), 则定理 C 是否仍成立?

► 3. [M26] (广义中国剩余定理) 设 m_1, m_2, \dots, m_r 是整数, m 是 m_1, m_2, \dots, m_r 的最小公倍数, 并设 a, u_1, u_2, \dots, u_r 是任意整数, 假定

$$u_i \equiv u_j \pmod{\gcd(m_i, m_j)}, \quad 1 \leq i < j \leq r$$

证明恰有一个整数 u 满足条件

$$a \leq u < a + m, \quad u \equiv u_j \pmod{m_j}, \quad 1 \leq j \leq r$$

而且当前边的条件不成立时, 不可能有这样的整数 u 。

4. [20] 继续(13)中所示的过程, m_7, m_8, m_9, \dots 将是什么?

► 5. [M23] 假设(13)的方法已继续进行到不能选择更多的 m_i 了, 这个“贪婪的”方法是否给出最大可达到的值 $m_1 m_2 \dots m_r$, 使得 m_i 都是小于 100 的而且都是两两互素的奇正整数?

6. [M22] 设 e, f, g 是非负整数。

a) 证明当且仅当 $e \equiv f \pmod{g}$ 时, $2^e \equiv 2^f \pmod{2^g - 1}$ 。

b) 给定 $e \bmod f = d$ 和 $ce \bmod f = 1$, 证明恒等式

$$((1 + 2^d + \dots + 2^{(c-1)d}) \cdot (2^f - 1)) \bmod (2^f - 1) = 1$$

(于是, 对于如同(23)中所要求的 $2^e - 1$ 的模 $2^f - 1$ 逆, 我们有了一个比较简单的公式。)

► 7. [M21] 证明(24)可重写如下:

$$v_1 \leftarrow u_1 \bmod m_1$$

$$v_2 \leftarrow (u_2 - v_1)c_{12} \bmod m_2$$

$$v_3 \leftarrow (u_3 - (v_1 + m_1 v_2))c_{13}c_{23} \bmod m_3$$

⋮

$$u_r \leftarrow (u_r - (v_1 + m_1(v_2 + m_2(v_3 + \dots + m_{r-2}v_{r-1})\dots)))c_{1r}\dots c_{(r-1)r} \bmod m_r$$

如果以这种方式重写这些公式, 则仅仅需要 $r-1$ 个常数 $C_j = c_{1j}\dots c_{(j-1)j} \bmod m_j$, 而不是像在(24)中那样需要 $r(r-1)/2$ 个常数 c_{ij} 。试从计算机计算的观点着眼, 讨论这种形式的公式相对于(24)的优点。

8. [M21] 证明由(24)和(25)所定义的数 u 满足(26)。

9. [M20] 说明如何只使用模 m_i 算术计算 u_i 的值, 就从混合进制记法(25)的值 v_1, \dots, v_r 返回到原来的剩余 u_1, \dots, u_r 。

10. [M25] 处于对称范围(10)中的一个整数 u , 可以通过求出数 u_1, \dots, u_r 来表示, 其中 $u \equiv u_j \pmod{m_j}$ 而且 $-m_j/2 < u_j < m_j/2$, 而不是像在正文中那样坚持 $0 \leq u_j < m_j$ 。试讨论这样的模算术过程, 在同这样的对称表示相关联时, 它们将是适用的(包括转换过程(24))。

11. [M23] 假设所有的 m_i 都是奇数, 而且已知 $u = (u_1, \dots, u_r)$ 是偶数, 其中 $0 \leq u < m$ 。试求一个使用模算术来计算 $u/2$ 的相当快的方法。

12. [M10] 证明, 如果 $0 \leq u, v < m$, 则当且仅当和数小于 u 时, u 和 v 的模加法引起溢出

(即超出由模的表示所允许的范围之外)。(这样,溢出的检测问题就等价于比较问题。)

►13. [M25](自守的数) 一个 n 位十进数 $x > 1$ 被“娱乐”数学家称为“自守的”,如果 x^2 的最后 n 个数字等于 x 。例如,9376 是一个 4 位的自守数,因为 $9376^2 = 87909376$ 。[见 *Scientific American* 218 (1968 年 1 月),125.]

a) 证明当且仅当分别地有 $x \bmod 5^n = 0$ 或 1, 以及 $x \bmod 2^n = 1$ 或 0 时,一个 n 位数 $x > 1$ 是自守的(于是,如果 $m_1 = 2^n$ 和 $m_2 = 5^n$, 则仅有的两个 n 位自守数是 (7) 中的数 M_1 和 M_2)。

b) 证明如果 x 是 n 位自守的, 则 $(3x^2 - 2x^3) \bmod 10^{2n}$ 是 $2n$ 位自守的。

c) 给定 $cx \equiv 1 \pmod{y}$, 试求依赖于 c 和 x 但不依赖于 y , 且满足 $c'x^2 \equiv 1 \pmod{y^2}$ 的 c' 的简单公式。

►14. [M30](梅森乘法) $(x_0, x_1, \dots, x_{n-1})$ 和 $(y_0, y_1, \dots, y_{n-1})$ 的循环卷积定义为 $(z_0, z_1, \dots, z_{n-1})$, 其中

$$z_k = \sum_{i+j=k \pmod{n}} x_i y_j, \quad \text{对于 } 0 \leq k < n$$

在 4.4.3 和 4.6.4 小节中我们将研究实现循环卷积的有效算法。

考虑以下列形式表示的 q 个二进位的整数 u 和 v :

$$u = \sum_{k=0}^{n-1} u_k 2^{\lfloor kq/n \rfloor}, \quad v = \sum_{k=0}^{n-1} v_k 2^{-kq/n}.$$

其中 $0 \leq u_k, v_k < 2^{-(k+1)q/n + \lfloor kq/n \rfloor}$ 。(这个表示是进制 $2^{\lfloor q/n \rfloor}$ 和 $2^{-q/n}$ 的混合。)使用一个适当的循环卷积, 提出一个求

$$w = (uv) \bmod (2^q - 1)$$

的表示的好方法[提示: 不要怕浮点算术]。

*4.3.3 乘法能有多快?

在定位计数系统中, 乘法的通常方法(即算法 4.3.1M), 把一个 m 位数字的数乘以一个 n 位数字的数需要近似于 cmn 个运算, 其中 c 是一个常数。在这一小节中, 为方便起见, 我们假定 $m = n$, 并且考虑下列问题: 每个通用的做两个 n 位数字乘法的计算机算法, 当 n 增大时, 是否总要求同 n^2 成正比的执行时间?

(在这个问题中, “通用的”算法指的是这样一个算法, 它接受定位计数法下的数 n 和两个任意的 n 位数作为输入, 而它输出定位计数法下两个数的乘积。肯定地说, 如果允许对 n 的每个值选择不同的算法, 则这个问题就没有什么兴趣了, 因为乘法可对任何确定的 n 值以对某张大表的“查表操作”来进行。“计算机算法”这一术语指的是一个这样的算法, 它宜于在一台数字计算机(例如 MIX)上实现, 而且执行时间是在这样一台计算机上来实施该算法所需的时间。)

A. 数字方法 令人有些惊异的是对于上述问题的答案竟是“否”。事实上, 不难看出这究竟是因为什么。为方便起见, 在这一节内始终假定, 我们处理的是表示成二进制记号的整数。如果有两个 $2n$ 位数字 $u = (u_{2n-1} \dots u_1 u_0)_2$ 和 $v = (v_{2n-1} \dots v_1 v_0)_2$, 则我们可以写

$$u = 2^n U_1 + U_0, \quad v = 2^n V_1 + V_0 \quad (1)$$

其中 $U_1 = (u_{2n-1} \cdots u_n)_2$ 是 u 的“高位那一半”，而 $U_0 = (u_{n-1} \cdots u_1 u_0)_2$ 是“低位那一半”；而且类似地， $V_1 = (v_{2n-1} \cdots v_n)_2$ ， $V_0 = (v_{n-1} \cdots v_1 v_0)_2$ 。现在我们有

$$uv = (2^{2n} + 2^n)U_1V_1 + 2^n(U_1 - U_0)(V_0 - V_1) + (2^n + 1)U_0V_0 \quad (2)$$

这个公式把 2^n 位数乘法的问题归结为 n 位数的三个乘法，即 U_1V_1 ， $(U_1 - U_0) \cdot (V_0 - V_1)$ 和 U_0V_0 ，加上某些简单的移位和加法运算。

当想要得到四倍精度的结果时，公式(2)可用于双精度乘法，而且它只比在许多机器上用的传统方法稍快一点。但是公式(2)的主要优点在于我们可以使用它来定义乘法的递归过程，当 n 很大时，这个过程要比熟知的 n^2 阶方法快得多：如果 $T(n)$ 是为实施 n 位数的乘法所需要的时间，则对于某个常数 c 我们有

$$T(2n) \leq 3T(n) + cn \quad (3)$$

因为(2)的右边仅仅使用三个乘法加上某些加法和移位。按归纳法，关系式(3)意味着，如果我们把 c 选择得充分大，使得如下不等式当 $k=1$ 时成立，则

$$T(2^k) \leq c(3^k - 2^k), \quad k \geq 1 \quad (4)$$

由此我们有

$$T(n) \leq T(2^{\lceil \lg n \rceil}) \leq c(3^{\lceil \lg n \rceil} - 2^{\lceil \lg n \rceil}) < 3c \cdot 3^{\lg n} = 3cn^{\lg 3} \quad (5)$$

关系式(5)说明，乘法的运算时间可以从 n^2 阶减少到 $n^{\lg 3} \approx n^{1.585}$ 的阶，所以当 n 很大时，递归方法要比传统方法快得多。习题 18 讨论这个方法的一个实现。

(以阶 $n^{\lg 3}$ 的运行时间来进行乘法的一种类似的但更为复杂的方法，看来首先是由 A. Karatsuba 提出的[Doklady Akad. Nauk SSSR 145 (1962), 293~294; 英译见 Soviet Physics-Doklady 7 (1963), 595~596]。奇怪的是，这个思想在 1962 年以前似乎未被发现；还没有报道过任何一位用心算乘很大的数而闻名于世的“计算天才”曾使用过这类方法，尽管把公式(2)改为十进记号似乎可导致用心算乘 8 位数字数的一个相当容易的方法。

如果我们注意到刚才所使用的方法实际上是一个更为一般的方法在 $r=1$ 时的特例，则对于任何固定的 r ，用这个一般的方法可推出

$$T((r+1)n) \leq (2r+1)T(n) + cn \quad (6)$$

在 n 趋于无穷时，运行时间还能进一步减少。这个更为一般的方法是：设

$$u = (u_{(r+1)n-1} \cdots u_1 u_0)_2 \quad \text{和} \quad v = (v_{(r+1)n-1} \cdots v_1 v_0)_2$$

被分成 $r+1$ 个片段

$$u = U_r 2^{rn} + \cdots + U_1 2^n + U_0, \quad v = V_r 2^{rn} + \cdots + V_1 2^n + V_0 \quad (7)$$

其中每个 U_j 和每个 V_j 都是一个 n 位数。考虑多项式

$$U(x) = U_r x^r + \cdots + U_1 x + U_0, \quad V(x) = V_r x^r + \cdots + V_1 x + V_0 \quad (8)$$

并命

$$W(x) = U(x)V(x) = W_{2r} x^{2r} + \cdots + W_1 x + W_0 \quad (9)$$

由于 $u = U(2^n)$ 和 $v = V(2^n)$ ，我们有 $uv = W(2^n)$ ，所以如果我们已知 $W(x)$ 的系数，则就能容易地计算 uv 。问题是要找一个仅需 $2r+1$ 个 n 位数的乘法，加上某些

执行时间只同 n 成比例的其它运算,即可计算 $W(x)$ 的系数的好方法。这个方法可以通过计算

$$U(0)V(0) = W(0), U(1)V(1) = W(1), \dots, U(2r)V(2r) = W(2r) \quad (10)$$

来完成。一个 $2r$ 次多项式的系数可以写成该多项式在 $2r+1$ 个不同点处的值的线性组合;这样一个线性组合至多要求同 n 成比例的执行时间。(实际上,乘积 $U(j) \cdot V(j)$ 严格说并不是 n 位数的乘积,但它们都是至多 $n+t$ 位数的乘积,其中 t 是依赖于 r 的一个固定值。容易设计 $n+t$ 位数的一个乘法程序,它仅要求 $T(n) + c_1 n$ 次运算,其中 $T(n)$ 是 n 位乘法需要的运算次数,因为 t 位乘以 n 位数的两个乘积,当 t 固定时,可以在 $c_2 n$ 个运算中完成。)因此我们得到一个满足(6)的乘法方法。

如果像在推导(5)中那样论证,关系式(6)意味着 $T(n) \leq c_3 n^{\log_{r+1}(2r+1)} < c_3 n^{1+\log_{r+1} 2}$;所以我们现在已经证明了下面的结果:

定理 A 给定 $\epsilon > 0$, 对于一个与 n 无关的常数 $c(\epsilon)$, 存在一个乘法算法,使得为乘两个 n 位数所需要的初等运算的次数 $T(n)$ 满足

$$T(n) < c(\epsilon) n^{1+\epsilon} \quad \blacksquare \quad (11)$$

这一定理仍然不是我们寻求的最终结果。它对于实用说来,是不能令人满意的,这是因为当 $\epsilon \rightarrow 0$ 时(因此,当 $r \rightarrow \infty$ 时),这个方法变得极其复杂,它引起 $c(\epsilon)$ 如此急剧地增长,以致在对我们(5)有任何实质性的改进以前,就需要 n 的非常大的值。而且,它对于理论的目的说来,也不是令人满意的,因为它不能充分发挥多项式方法的全部威力,而这个方法正是它的基础。如果让 r 同 n 一起变动,即随 n 的增加而选择越来越大的 r 值,则我们就可以得到一个更好的结果。这个思想是由 A. L. Toom 提出的 [Doklady Akad. Nauk SSSR 50 (1963), 496~498, 英译见 Soviet Mathematics 3 (1963), 714~716], 他用这个思想证明:当 n 增长时,可用相当少的部件来构造 n 位数乘法的计算机线路。后来, S. A. Cook [On the Minimum Computation Time of Functions (哈佛大学论文, 1960), 51~77] 说明了如何采用 Toom 的方法加快计算机程序。

在进一步讨论 Toom-Cook 的算法之前,让我们研究从 $U(x)$ 和 $V(x)$ 转换到 $W(x)$ 的系数的一个小例子。这个例子并不说明这个方法的有效性,因为数太小了;但它指出了在一般情况下可以做的某些有用的简化。假设我们要进行 $u = 1234$ 和 $v = 2341$ 的乘法,在二进记法下,这是

$$u = (0100 \ 1101 \ 0010)_2 \quad \text{乘以} \quad v = (1001 \ 0010 \ 0101)_2 \quad (12)$$

设 $r=2$; (8) 中的多项式 $U(x), V(x)$ 是

$$U(x) = 4x^2 + 13x + 2, \quad V(x) = 9x^2 + 2x + 5$$

因此我们发现,对于 $W(x) = U(x)V(x)$,

$$\begin{aligned} U(0) &= 2, & U(1) &= 19, & U(2) &= 44, & U(3) &= 77, & U(4) &= 118; \\ V(0) &= 5, & V(1) &= 16, & V(2) &= 45, & V(3) &= 92, & V(4) &= 157; \\ W(0) &= 10, & W(1) &= 304, & W(2) &= 1980, & W(3) &= 7084, & W(4) &= 18526 \end{aligned} \quad (13)$$

我们当前的工作是从后边五个值来计算 $W(x)$ 的五个系数。

有一个吸引人的小算法,当给定值 $W(0), W(1), \dots, W(m-1)$ 时它能用来计算多项式 $W(x) = W_{m-1}x^{m-1} + \dots + W_1x + W_0$ 的系数。让我们首先写

$$W(x) = a_m x^{m-1} + a_{m-2} x^{m-2} + \cdots + a_1 x^1 + a_0 \quad (14)$$

其中 $x^{(k)} = x(x-1)\cdots(x-k+1)$, 而且其中诸系数 a_j 是未知的。降阶乘幂有重要的性质, 即

$$W(x+1) - W(x) = (m-1)a_{m-1}x^{m-2} + (m-2)a_{m-2}x^{m-3} + \cdots + a_1$$

因此由归纳法,我们发现对于所有 $k \geq 0$,

$$\begin{aligned} \frac{1}{k!} (W(x+k) - \binom{k}{1} W(x+k-1) + \binom{k}{2} W(x+k-2) - \cdots + \\ (-1)^k W(x)) = \binom{m-1}{k} a_{m-1} x^{\frac{m-1-k}{2}} + \\ \binom{m-2}{k} a_{m-2} x^{\frac{m-2-k}{2}} + \cdots + \binom{k}{k} a_k \end{aligned} \quad (15)$$

以 $(1/k!) \Delta^k W(x)$ 表示 (15) 的左边, 我们看到

$$\frac{1}{k!} \Delta^k W(x) = \frac{1}{k} \left(\frac{1}{(k-1)!} \Delta^{k-1} W(x+1) - \frac{1}{(k-1)!} \Delta^{k-1} W(x) \right)$$

而且 $(1/k!) \Delta^k W(0) = a_k$ 。所以,利用一个非常简单的方法就可以计算系数 a_j ,这里以(13)中的多项式 $W(x)$ 来说明:

10				
	294			
304		$1382/2 = 691$		
	1676		$1023/3 = 341$	
1980		$3428/2 = 1714$		$144/4 = 36$
	5104		$1455/3 = 485$	(16)
7084		$6338/2 = 3169$		
	11442			
8256				

这个表最左边一列是 $W(0), W(1), \dots, W(4)$ 的给定值的一份清单; 通过计算前列逐个值之间的差并除以 k , 就得到第 k 列。系数 a_j 出现于这些列的顶部, 于是 $a_0 = 10, a_1 = 294, \dots, a_4 = 36$, 因此我们有

$$W(x) = 36x^4 + 341x^3 + 691x^2 + 294x + 10 = ((36(x-3) + 341)(x-2) + 691)(x-1) + 294)x + 10 \quad (17)$$

一般说来,我们可以写

$$W(x) = (\cdots((a_{m-1}(x-m+2) + a_{m-2})(x-m+3) + a_{m-3})(x-m+4) + \cdots + a_1)x + a_0$$

而且这个公式说明了如何能从诸 a 得到系数 $W_{m-1}, \cdots, W_1, W_0$:

$$\begin{array}{r}
 \begin{array}{|l} 36 \end{array} \quad \begin{array}{|l} 341 \\ -3 \cdot 36 \end{array} \\
 \hline
 \begin{array}{|l} 36 \end{array} \quad \begin{array}{|l} 233 \\ -2 \cdot 36 \end{array} \quad \begin{array}{|l} 691 \\ -2 \cdot 233 \end{array} \\
 \hline
 \begin{array}{|l} 36 \end{array} \quad \begin{array}{|l} 161 \\ -1 \cdot 36 \end{array} \quad \begin{array}{|l} 255 \\ -1 \cdot 161 \end{array} \quad \begin{array}{|l} 294 \\ -1 \cdot 225 \end{array} \\
 \hline
 \begin{array}{|l} 36 \end{array} \quad \begin{array}{|l} 125 \end{array} \quad \begin{array}{|l} 64 \end{array} \quad \begin{array}{|l} 69 \end{array} \quad \left. \begin{array}{|l} 10 \end{array} \right\}
 \end{array} \quad (18)$$

这里水平线下的数逐次地表示多项式

$$\begin{aligned}
 & a_{m-1} \\
 & a_{m-1}(x-m+2) + a_{m-2} \\
 & (a_{m-1}(x-m+2) + a_{m-2})(x-m+3) + a_{m-3}
 \end{aligned}$$

等的系数。

由这个表,我们有

$$W(x) = 36x^4 + 125x^3 + 64x^2 + 69x + 10 \quad (19)$$

所以原来问题的答案是 $1234 \times 2341 = W(16) = 2888794$, 其中 $W(16)$ 通过加法和移位得到。4.6.4 小节讨论了这个求系数方法的一个推广。

等式 1.2.6-(45) 的基本的斯特林恒等式

$$x^n = \begin{Bmatrix} n \\ n \end{Bmatrix} x^n + \cdots + \begin{Bmatrix} n \\ 1 \end{Bmatrix} x^1 + \begin{Bmatrix} n \\ 0 \end{Bmatrix}$$

说明, 如果 $W(x)$ 的系数非负, 则数 a_j 也非负, 而且在这种情况下, 上述计算中的所有中间结果非负。这就进一步简化了 Toom-Cook 的乘法算法。我们现在详细讨论这个算法(而心急的读者应该跳到以下的子小节 C)。

算法 T(二进制数的高精度乘法) 给定一正整数 n 和两个非负的 n 位数 u 和 v , 这一算法给出它们的 $2n$ 位乘积 w 。使用四个辅助栈来存放进行这一过程期间所处理的数字:

栈 U, V : 步骤 T4 中的 $U(j)$ 和 $V(j)$ 的临时存储单元

栈 C : 要做乘法的数以及控制代码

栈 W : $W(j)$ 的存储单元

这些栈可以包含二进数或者称为代码 1、代码 2 和代码 3 的特殊控制符号。本算法也构造数 q_k, r_k 的一份辅助表;维持该表的方式是,它可以存作一个线性表,其中遍历这个表(前后移动)的一个指针可用来访问当前感兴趣的表项。

(栈 C 和 W 以一种相当直截了当的方式来控制乘法算法的递归机制,它是第 8 章讨论的一般过程的一个特殊情况。)

T1. [计算 q, r 表] 置栈 U, V, C 和 W 为空。置

$$k \leftarrow 1, q_0 \leftarrow q_1 \leftarrow 16, r_0 \leftarrow r_1 \leftarrow 4, Q \leftarrow 4, R \leftarrow 2$$

现在如果 $q_{k-1} + q_k < n$, 则置

$$k \leftarrow k + 1, Q \leftarrow Q + R, R \leftarrow \lfloor \sqrt{Q} \rfloor, q_k \leftarrow 2^Q, r_k \leftarrow 2^R$$

并重复这一操作直到 $q_{k-1} + q_k \geq n$ 。(注: $R \leftarrow \lfloor \sqrt{Q} \rfloor$ 的计算不要求开平方,因为,如果 $(R+1)^2 \leq Q$ 则我们只需置 $R \leftarrow R+1$,而如果 $(R+1)^2 > Q$,则保持 R 不变;见习题 2。在这一步骤里我们构造序列

$$\begin{array}{cccccccc} k & = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & \cdots \\ q_k & = & 2^4 & 2^4 & 2^6 & 2^8 & 2^{10} & 2^{13} & 2^{16} & \cdots \\ r_k & = & 2^2 & 2^2 & 2^2 & 2^2 & 2^3 & 2^3 & 2^4 & \cdots \end{array}$$

70000 位数的乘法,将使这步以 $k=6$ 终止,因为 $70000 < 2^{13} + 2^{16}$ 。)

T2. [把 u, v 压入栈] 置代码 1 于栈 C ,然后把 u 和 v 作为两个恰有 $q_{k-1} + q_k$ 位的数压入栈 C 。

T3. [校验递归层次] k 减 1。如果 $k=0$,则堆栈 C 顶部含有两个 32 位数 u 和 v ;把它们去掉,并利用一个乘 32 位数的内部程序置 $w \leftarrow uv$,转到步骤 T10。如果 $k > 0$,则置 $r \leftarrow r_k, q \leftarrow q_k, p \leftarrow q_{k-1} + q_k$,并继续进行到步骤 T4。

T4. [分成 $r+1$ 部分] 把栈 C 顶部的数看成是 $r+1$ 个 q 位数的一份表($U_r \cdots U_1 U_0$)_{2^q}(栈 C 顶部现在包含一个 $(r+1)q = (q_k + q_{k+1})$ 位的数)。对于 $j=0, 1, \cdots, 2r$ 计算 p 位数

$$(\cdots (U_{r,j} + U_{r-1,j})j + \cdots + U_{1,j})j + U_{0,j} = U(j)$$

并且逐次地把这些值压入栈 U (栈底现在包含 $U(0)$,然后是 $U(1)$,等等,而 $U(2r)$ 在顶上。由习题 3 有

$$U(j) \leq U(2r) < 2^q((2r)^r + (2r)^{r-1} + \cdots + 1) < 2^{q+1}(2r)^r \leq 2^p$$

然后从栈 C 消去 $U_r \cdots U_1 U_0$ 。

现在栈 C 顶上包含另外 $r+1$ 个 q 位数的表 $V_r \cdots V_1 V_0$,而且 p 位数

$$(\cdots (V_{r,j} + V_{r-1,j})j + \cdots + V_{1,j})j + V_{0,j} = V(j)$$

应以同样方式压入栈。在这完成之后,从栈 C 消去 $V_r \cdots V_1 V_0$ 。

T5. [递归] 逐次地把下列各项压入栈 C ,同时消除栈 U 和 V 中的内容:

代码 2, $V(2r), U(2r)$, 代码 3, $V(2r-1), U(2r-1), \cdots$,

代码 3, $V(1), U(1)$, 代码 3, $V(0), U(0)$

返回步骤 T3。

T6. [保存一个乘积] (这时乘法算法已经置 w 成为诸乘积 $W(j) = U(j)V(j)$ 之一。)把 w 压入栈 W (这数 w 包含 $2(q_k + q_{k-1})$ 位)。返回步骤 T3。

T7. [求诸 a] 置 $r \leftarrow r_k, q \leftarrow q_k, p \leftarrow q_{k-1} + q_k$ 。(这时栈 W 从底到顶包含一系列数,以 $W(0), W(1), \dots, W(2r)$ 结束,每个 $W(j)$ 是一个 $2p$ 位数。)

现在对 $j = 1, 2, 3, \dots, 2r$ 实施下列循环:对于 $t = 2r, 2r-1, 2r-2, \dots, j$ 置 $W(t) \leftarrow (W(t) - W(t-1))/j$ 。(这里 j 必须增值,而 t 必须减值。量 $(W(t) - W(t-1))/j$ 将总是一个可用 $2p$ 位表示的非负整数,参考(16)。)

T8. [求诸 W] 对 $j = 2r-1, 2r-2, \dots, 1$ 实施下列循环:对于 $t = j, j+1, \dots, 2r-1$, 置 $W(t) \leftarrow W(t) - jW(t+1)$ 。(这里 j 必须减值且 t 必须增值。这一运算的结果将再次是一个非负的 $2p$ 位整数;参考(18)。)

T9. [置答案] 置 w 成为 $2(q_k + q_{k+1})$ 位整数

$$(\dots(W(2r)2^q + W(2r-1))2^q + \dots + W(1))2^q + W(0)$$

从栈 W 消去 $W(2r), \dots, W(0)$ 。

T10. [返回] 置 $k \leftarrow k+1$, 消去栈 C 的顶部。如果它是代码 3, 则转到 T6。如果它是代码 2, 则把它压入栈 W 并转到 T7。如果它是代码 1, 则终止这一算法(w 是答案)。

现在借助于我们所说的“周期”的某些东西亦即基本的机器操作时间,来估计算法 T 的运行时间。步骤 T1 花费 $O(q_k)$ 个周期时间,即使我们在内部把数 q_k 表示成一个在后边加上某个限定符的 q_k 位的长串也一样,因为 $q_k + q_{k-1} + \dots + q_0$ 仍将是 $O(q_k)$ 。显然,步骤 T2 花费 $O(q_k)$ 个单位时间。

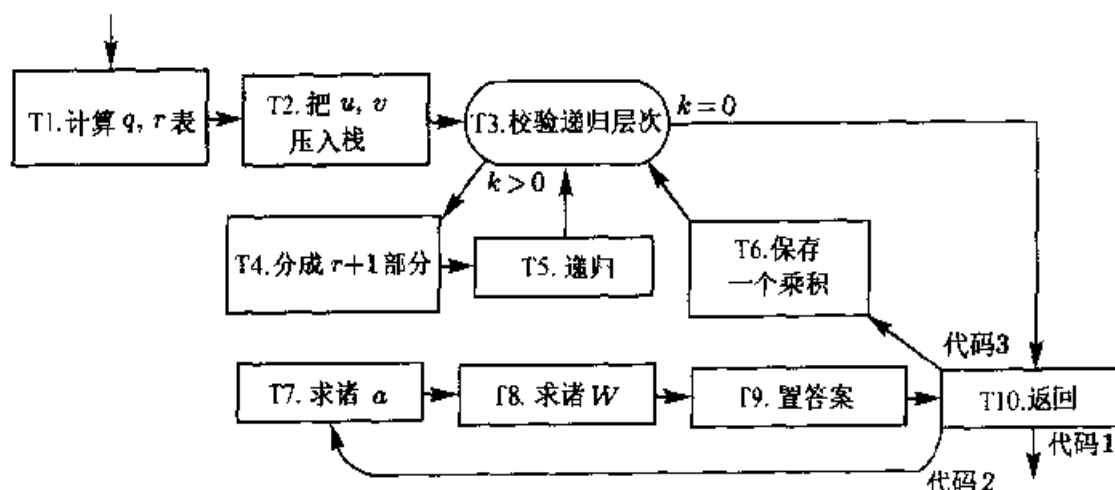


图 8 高精度乘法的 Toom-Cook 算法

现在命 t_k 表示对于一个特殊的 k 值(在 k 的值于步骤 T3 开始时减小了之后)为从步骤 T3 达到 T10 所需的计算数量。步骤 T3 至多要求 $O(q)$ 个周期。步骤 T4 包含 r 个 $(\lg 2r)$ 位数乘以一个 p 位数的乘法,以及 r 个 p 位数的加法,全部重复 $4r$

+2次。于是,我们需要总共 $O(r^2q \log r)$ 个周期。步骤 T5 要求移动 $4r+2$ 个 p 位数,所以它包含 $O(rq)$ 个周期。步骤 T6 要求 $O(q)$ 个周期,而且每次迭代它都做 $2r+1$ 次。当算法实际上调用它自身时(通过返回步骤 T3),涉及的递归每次要求 t_{k-1} 个周期, $2r+1$ 次。步骤 T7 要求 $O(r^2)$ 个 p 位数的减法和 $2p$ 位数除以 $(\lg 2r)$ 位数的除法,所以它要求 $O(r^2q \log r)$ 个周期。类似地,步骤 T8 要求 $O(r^2q \log r)$ 个周期。步骤 T9 包含 $O(rq)$ 个周期,而且 T10 几乎全然不花费任何时间。

总计起来,我们有 $T(n) = O(q_k) + O(q_k) + t_{k-1}$, 其中(如果 $q = q_k$ 和 $r = r_k$) 对运行时间的主要贡献满足

$$\begin{aligned} t_k &= O(q) + O(r^2q \log r) + O(rq) + (2r+1)O(q) + O(r^2q \log r) + \\ &\quad O(r^2q \log r) + O(rq) + O(q) + (2r+1)t_{k-1} = \\ &\quad O(r^2q \log r) + (2r+1)t_{k-1} \end{aligned}$$

于是有一常数 c , 使

$$t_k \leq cr_k^2 q_k \lg r_k + (2r_k + 1)t_{k-1}$$

为了完成对 t_k 的估计,我们通过硬算可以证明对于某个常数 C ,

$$t_k \leq Cq_{k+1}2^{2.5\sqrt{\lg q_{k+1}}} \quad (20)$$

我们选择 $C > 20c$, 并把 C 也取得足够大,使(20)对于 $k \leq k_0$ 成立,其中 k_0 将在下边加以确定。于是当 $k > k_0$ 时,令 $Q_k = \lg q_k$, $R_k = \lg r_k$; 由归纳法有

$$t_k \leq cq_k r_k^2 \lg r_k + (2r_k + 1)Cq_k 2^{2.5\sqrt{Q_k}} = Cq_{k+1} 2^{2.5\sqrt{\lg q_{k+1}}} (\eta_1 + \eta_2)$$

其中

$$\eta_1 = \frac{c}{C} R_k 2^{R_k} 2^{2.5\sqrt{Q_{k+1}}} < \frac{1}{20} R_k 2^{-R_k} < 0.05$$

$$\eta_2 = \left(2 + \frac{1}{r_k}\right) 2^{2.5(\sqrt{Q_k} - \sqrt{Q_{k+1}})} \rightarrow 2^{-1/4} < 0.85$$

因为当 $k \rightarrow \infty$ 时,

$$\sqrt{Q_{k+1}} - \sqrt{Q_k} = \sqrt{Q_k + \lfloor \sqrt{Q_k} \rfloor} - \sqrt{Q_k} \rightarrow \frac{1}{2}$$

由此得出,我们可以求出 k_0 ,使得对于所有 $k > k_0$, $\eta_2 < 0.95$,按归纳法这就完成了(20)的证明。

因此,最后我们可以估计 $T(n)$ 。由于 $n > q_{k-1} + q_{k-2}$,我们有 $q_{k-1} < n$; 因此

$$r_{k-1} = 2^{\lfloor \sqrt{\lg q_{k-1}} \rfloor} < 2^{\sqrt{\lg n}}, \quad \text{且 } q_k = r_{k-1} q_{k-1} < n 2^{\sqrt{\lg n}}$$

于是

$$t_{k-1} \leq Cq_k 2^{2.5\sqrt{\lg q_k}} < Cn 2^{\sqrt{\lg n} + 2.5(\sqrt{\lg n} + 1)}$$

而且由于 $T(n) = O(q_k) + t_{k-1}$, 我们最后导出了以下定理:

定理 B 存在一个常数 c_0 ,使得算法 T 的执行时间小于 $c_0 n 2^{3.5\sqrt{\lg n}}$ 个周期。 ■

由于 $n2^{3.5\sqrt{\lg n}} = n^{1+3.5/\sqrt{\lg n}}$, 这一结果比定理 A 强得多。对这个算法稍加复杂化, 把上述思想施加于明显的极限上(见习题 5), 我们就可以把估计的执行时间改进为

$$T(n) = O(n2^{\sqrt{2 \lg n}} \log n) \quad (21)$$

***B. 一个模方法** 有另一个非常快地乘大数的方法, 它是以 4.3.2 小节提出的模算术的思想为基础的。开头很难相信这个算法能有什么好处, 因为基于模算术的乘法算法除了乘法运算本身之外, 必须包括模的选择以及这些数到模表示的来回转换。不过, 尽管有这些难以对付的困难, A. Schönhage 还是发现, 所有这些运算都可以非常迅速地完成。

为了了解 Schönhage 方法的实质性机理, 我们将考虑一个非常特殊的情况。考虑由规则

$$q_0 = 1, \quad q_{k+1} = 3q_k - 1 \quad (22)$$

所定义的序列, 使得 $q_k = 3^k - 3^{k-1} - \dots - 1 = \frac{1}{2}(3^k + 1)$ 。借助于乘 p_{k-1} 位数的乘法, 我们将研究乘 p_k 个二进制数的过程, 其中 $p_k = (18q_k + 8)$ 。于是, 如果我们知道怎样乘有 $p_0 = 26$ 位的数, 则要描述的这一过程将向我们显示怎样来乘 $p_1 = 44$ 位的数, 然后是 92 位的数, 再后是 260 位, 等等, 最后每步使位数几乎以 3 的因子增长。

当乘 p_k 位数时, 其思想是使用 6 个模

$$\begin{aligned} m_1 &= 2^{6q_k-1} - 1, & m_2 &= 2^{6q_k+1} - 1, & m_3 &= 2^{6q_k+2} - 1 \\ m_4 &= 2^{6q_k+3} - 1, & m_5 &= 2^{6q_k+5} - 1, & m_6 &= 2^{6q_k+7} - 1 \end{aligned} \quad (23)$$

由等式 4.3.2-(19), 这些模是互素的, 因为指数

$$6q_k - 1, \quad 6q_k + 1, \quad 6q_k + 2, \quad 6q_k + 3, \quad 6q_k + 5, \quad 6q_k + 7 \quad (24)$$

总是互素的(见习题 6)。 (23) 中的 6 个模有能力表示直到 $m = m_1 m_2 m_3 m_4 m_5 m_6 > 2^{36q_k+16} = 2^{2p_k}$ 的数, 所以在 p_k 位数 u 和 v 的乘法中没有机会溢出。这样我们可以使用下列方法: 当 $k > 0$ 时,

a) 计算 $u_1 = u \bmod m_1, \dots, u_6 = u \bmod m_6$; 以及 $v_1 = v \bmod m_1, \dots, v_6 = v \bmod m_6$ 。

b) u_1 乘以 v_1, u_2 乘以 v_2, \dots, u_6 乘以 v_6 。这些至多都是 $6q_k + 7 = 18q_{k-1} + 1 < p_{k-1}$ 位的数, 所以通过使用假设的 p_{k-1} 位乘法过程即可实施乘法。

c) 计算 $w_1 = u_1 v_1 \bmod m_1, w_2 = u_2 v_2 \bmod m_2, \dots, w_6 = u_6 v_6 \bmod m_6$ 。

d) 计算 w , 使得 $0 \leq w < m, w \bmod m_1 = w_1, \dots, w \bmod m_6 = w_6$ 。

设 t_k 是这一过程所需要的时间数量。不难看到, 运算 a) 花费 $O(p_k)$ 个周期, 因为如 4.3.2 小节所示, $u \bmod (2^n - 1)$ 的确定是十分简单的(像“舍九算法”)。类似地, 运算 c) 花费 $O(p_k)$ 个周期。运算 b) 实际上要求 $6t_{k-1}$ 个周期。剩下运算 d), 它

似乎是一个十分困难的计算,但 Schönage 已经找到在 $O(p_k \log p_k)$ 个周期中实现步骤 d) 的一个巧妙的方法,而且这是这个方法的关键。结果,我们有

$$t_k = 6t_{k-1} + O(p_k \log p_k)$$

因为 $p_k = 3^{k+2} + 17$, 我们可以证明进行 n 位乘法的时间是

$$T(n) = O(N^{\log_3 6}) = O(N^{1.631}) \quad (25)$$

(见习题 17。)

所以尽管模方法比在这节开始所讨论的 $O(n^{\log 3})$ 过程更为复杂,但事实上,等式(25)表明,它确实导致比 n 位数乘法的 $O(n^2)$ 实质上更好的执行时间。于是我们看到如何使用两种完全不同的方法中的任一种来改进经典的方法。

现在我们来分析以上的运算 d), 假定给了我们两两互素的正整数 $e_1 < e_2 < \dots < e_r$; 设

$$m_1 = 2^{e_1} - 1, \quad m_2 = 2^{e_2} - 1, \quad \dots, \quad m_r = 2^{e_r} - 1 \quad (26)$$

而且也给了我们数 w_1, \dots, w_r , 使得 $0 \leq w_j \leq m_j$ 。我们的任务是来确定数 w 的二进表示, 它满足条件

$$0 \leq w < m_1 m_2 \dots m_r \quad (27)$$

$$w \equiv w_1 \pmod{m_1}, \quad \dots, \quad w \equiv w_r \pmod{m_r}$$

这个方法基于 4.3.2 小节的(24)和(25)。首先对于 $j=2, \dots, r$ 计算

$$w'_j = (\dots((w_j - w'_1)c_{1j} - w'_2)c_{2j} - \dots - w'_{j-1})c_{(j-1)j} \pmod{m_j} \quad (28)$$

其中 $w'_1 = w_1 \pmod{m_1}$; 然后计算

$$w = (\dots(w'_r m_{r-1} + w'_{r-1})m_{r-2} + \dots + w'_2)m_1 + w'_1 \quad (29)$$

这里 c_{ij} 是使得 $c_{ij}m_i \equiv 1 \pmod{m_j}$ 的一个数; 这些数 c_{ij} 均未给出, 它们必须由诸 e_j 确定。

对于所有 j , 计算(28)包含 $\binom{r}{2}$ 个模 m_j 加法(它们中的每一个花费 $O(e_r)$ 个周期), 加上 $\binom{r}{2}$ 个乘以 c_{ij} 的模 m_i 乘法。通过公式(29)来计算 w 需要 r 个加法和 r 个乘以 m_j 的乘法; 以 m_j 来乘是容易的, 因为这仅仅是相加、移位和减法, 所以显然等式(29)的计算花费 $O(r^2 e_r)$ 个周期。我们马上就将看到, 乘以 c_{ij} 模 m_j 的乘法, 仅仅要求 $O(e_r \log e_r)$ 个周期, 因此, 整个转换工作可以在 $O(r^2 e_r \log e_r)$ 个周期中完成。

上述的考察留下了下列问题需要解决: 给定正数 e 和 f 且 $e < f$, 和一个非负整数 $u < 2^f$, 计算 $(cu) \pmod{2^f - 1}$, 其中 c 是使得 $(2^e - 1)c \equiv 1 \pmod{2^f - 1}$ 的数; 而且这整个计算必须在 $O(f \log f)$ 个周期中完成。习题 4.3.2-6 的结果给出了关于 c 的一个公式, 它提出了一个合适的过程。首先我们求最小整数 b 使得

$$be \equiv 1 \pmod{f} \quad (30)$$

利用欧几里得算法将在 $O((\log f)^3)$ 个周期中发现 b , 因为当应用于 e 和 f 时它要

求 $O(\log f)$ 次迭代, 而且每次迭代要求 $O((\log f)^2)$ 个周期; 另一种方法, 我们在这里通过简单地试验 $b=1, 2, \dots$, 可以非常粗略地进行, 而又不违背总的时间限制, 直到 (30) 被满足为止; 这样的过程全部将花费 $O(f \log f)$ 个周期。一旦已经求得 b , 则习题 4.3.2-6 告诉我们

$$c = c[b] = \left(\sum_{0 \leq i < b} 2^i \right) \bmod (2^f - 1) \quad (31)$$

硬算 $(cu) \bmod (2^f - 1)$ 的乘法, 对于解决这个问题来说是不够好的, 因为我们还不知道怎样在 $O(f \log f)$ 个周期中乘一般的 f 位数。但是 c 的特殊形式提供了一条线索: c 的二进表示是由排成正规模式的一些二进位组成, 而且等式 (31) 说明, 数 $c[2b]$ 可以以简单的方式从 $c[b]$ 得到。这提示, 如果我们以适当灵活的, 比如说如下的方法, 在 $\lg b$ 步中构造 $c[b]u$, 则我们可以迅速地进行数 u 和 $c[b]$ 的乘法: 设 b 的二进记号是

$$b = (b_s \cdots b_2 b_1 b_0)_2$$

我们可以计算由如下一些规则定义的序列 a_k, d_k, u_k 和 v_k :

$$\begin{aligned} a_0 &= e, & a_k &= 2a_{k-1} \bmod f \\ d_0 &= b_0 e, & d_k &= (d_{k-1} + b_k a_k) \bmod f \\ u_0 &= u, & u_k &= (u_{k-1} + 2^{a_{k-1}} u_{k-1}) \bmod (2^f - 1) \\ v_0 &= b_0 u, & v_k &= (v_{k-1} + b_k 2^{d_{k-1}} u_k) \bmod (2^f - 1) \end{aligned} \quad (32)$$

对 k 用归纳法容易证明

$$\begin{aligned} a_k &= (2^k e) \bmod f; & u_k &= (c[2^k]u) \bmod (2^f - 1) \\ d_k &= ((b_k \cdots b_1 b_0)_2 e) \bmod f; & v_k &= (c[(b_k \cdots b_1 b_0)_2]u) \bmod (2^f - 1) \end{aligned} \quad (33)$$

因此所希望的结果, $(c[b]u) \bmod (2^f - 1)$, 是 v_s 。由 $a_{k-1}, d_{k-1}, u_{k-1}, v_{k-1}$ 计算 a_k, d_k, u_k, v_k , 花费 $O(\log f) + O(\log f) + O(f) + O(f) = O(f)$ 个周期; 因此整个计算可以像所希望的那样在 $sO(f) = O(f \log f)$ 个周期中完成。

读者将发现, 非常仔细地研究由 (32) 和 (33) 表示的灵巧的方法是有教益的。4.6.3 小节讨论了类似的技术。

Schönhage 的论文 [Computing 1 (1966), 182~196] 证明, 利用 $r \approx 2^{\sqrt{2 \lg n}}$ 个模, 这些思想可被扩充到 n 位数的乘法, 并得到类似于算法 T 的一个方法。我们在这里将不详述细节, 因为算法 T 总是优越的; 事实上, 一个甚至更好的方法是我们的下一个议题。

C. 离散傅里叶变换 在高精度乘法中关键的问题是“卷积”, 例如

$$u_r v_0 + u_{r-1} v_1 + \cdots + u_0 v_r \quad (34)$$

的确定, 而且在卷积和称为傅里叶变换的一个重要数学概念之间有着密切的关系。如果 $\omega = \exp(2\pi i/K)$ 是 K 次单位根, 则复数序列 $(u_0, u_1, \dots, u_{K-1})$ 的一维傅里叶变换可以定义为 $(\hat{u}_0, \hat{u}_1, \dots, \hat{u}_{K-1})$, 其中

$$\hat{a}_s = \sum_{0 \leq r < K} \omega^s u_r, \quad 0 \leq s < K \quad (35)$$

设 $(\hat{v}_0, \hat{v}_1, \dots, \hat{v}_{K-1})$ 以同样方式定义, 和 $(v_0, v_1, \dots, v_{K-1})$ 的傅里叶变换一样, 不难看出, $(\hat{a}_0 \hat{v}_0, \hat{a}_1 \hat{v}_1, \dots, \hat{a}_{K-1} \hat{v}_{K-1})$ 是 $(w_0, w_1, \dots, w_{K-1})$ 的变换, 其中

$$w_r = u_r v_0 + u_{r-1} v_1 + \dots + u_0 v_r + u_{K-1} v_{r+1} + \dots + u_{r+1} v_{K-1} = \sum_{i+j=r \pmod{K}} u_i v_j \quad (36)$$

当 $K \geq 2n-1$ 和 $u_n = u_{n+1} = \dots = u_{K-1} = v_n = v_{r+1} = \dots = v_{K-1} = 0$ 时, 诸 w 恰是我们所需的乘积, 因为当 $0 \leq r \leq 2n-2$ 时, 项 $u_{K-1} v_{r+1} + \dots + u_{r+1} v_{K-1}$ 消失。换句话说, 一个卷积的变换是这些变换的普通乘积。这一思想实际上是 Toom 使用的多项式的一个特殊情况(参考(10)), 其中以单位根代替 x 。

如果 K 是 2 的幂, 则当以某种方式安排计算时, 可以十分迅速地得到离散傅里叶变换(35), 而且逆变换也这样(即由诸 \hat{w} 确定诸 w)。傅里叶变换的这个性质是由 V. Strassen 于 1968 年剖析的, 他发现了比过去所有已知的方案都快的大数乘法规则。他和 A. Schönhage 后来改进了这个方法, 并在 *Computing* 7 (1971), 281~292 上发表了改进的一些过程。类似的思想, 但是用于所有都是整数的方法, 是由 J. M. Pollard 独立地给出的 [*Math. Comp.* 25 (1971), 365~374]。为了了解他们解这个问题的方法, 我们首先看看快速傅里叶变换的机制。

给定 $K = 2^k$ 个复数 (u_0, \dots, u_{K-1}) 的一个序列, 并给定复数

$$\omega = \exp(2\pi i/K) \quad (37)$$

通过执行下列方案可以快速地计算在(35)中定义的序列 $(\hat{a}_0, \dots, \hat{a}_{K-1})$ 。(在这些公式中, 参数 s_j 和 t_j 或者是 0, 或者是 1, 使得每一“趟”表示 2^k 个初等的计算。)

趟 0 设 $A^{[0]}(t_{k-1}, \dots, t_0) = u_t$, 其中 $t = (t_{k-1} \dots t_0)_2$

趟 1 置 $A^{[1]}(s_{k-1}, t_{k-2}, \dots, t_0) \leftarrow$

$$A^{[0]}(0, t_{k-2}, \dots, t_0) + \omega^{2^{k-1}s_{k-1}} A^{[0]}(1, t_{k-2}, \dots, t_0)$$

趟 2 置 $A^{[2]}(s_{k-1}, s_{k-2}, t_{k-3}, \dots, t_0) \leftarrow$

$$A^{[1]}(s_{k-1}, 0, t_{k-3}, \dots, t_0) + \omega^{2^{k-2}(s_{k-2}s_{k-1})_2} A^{[1]}(s_{k-1}, 1, t_{k-3}, \dots, t_0)$$

...

趟 k 置 $A^{[k]}(s_{k-1}, \dots, s_1, s_0) \leftarrow$

$$A^{[k-1]}(s_{k-1}, \dots, s_1, 0) + \omega^{(s_0 s_1 \dots s_{k-1})_2} A^{[k-1]}(s_{k-1}, \dots, s_1, 1)$$

由归纳法相当容易证明, 我们有

$$A^{[j]}(s_{k-1}, \dots, s_{k-j}, t_{k-j-1}, \dots, t_0) = \sum_{0 \leq t_{k-1}, \dots, t_{k-j} \leq 1} \omega^{2^{k-j}(s_{k-j} \dots s_{k-1})_2 (t_{k-1} \dots t_{k-j})_2} u_t \quad (38)$$

其中 $t = (t_{k-1} \dots t_1 t_0)_2$

使得

$$A^{[k]}(s_{k-1}, \dots, s_1, s_0) = \hat{a}_s, \quad \text{其中 } s = (s_0 s_1 \dots s_{k-1})_2 \quad (39)$$

(重要的是注意 s 中二进数字在最后的結果(39)中被颠倒了, 4.6.4 小节包含关于这样的变换的进一步讨论。)

为了由 $(\hat{u}_0, \dots, \hat{u}_{K-1})$ 的值得到逆傅里叶变换 (u_0, \dots, u_{K-1}) , 请注意“双重变换”是

$$\begin{aligned} \hat{\hat{u}}_r &= \sum_{0 \leq s < K} \omega^{rs} \hat{u}_s = \sum_{0 \leq s, t < K} \omega^{rs} \omega^{st} u_t = \\ &= \sum_{0 \leq t < K} u_t \left(\sum_{0 \leq s < K} \omega^{s(t+r)} \right) = Ku_{(-r) \bmod K} \end{aligned} \quad (40)$$

因为除非 j 是 K 的倍数, 否则几何级数 $\sum_{0 \leq s < K} \omega^{sj}$ 的和为 0。因此逆变换可以和变换本身同样的方式来进行计算, 只是最后的结果必须除以 K 和稍微“洗”一下。

回到整数乘法的问题, 假设我们希望计算两个 n 位整数 u 和 v 的乘积。和在算法 T 中一样, 我们的处理对象是位组; 设

$$2n \leq 2^k l < 4n, \quad K = 2^k, \quad L = 2^l \quad (41)$$

而且写

$$u = (U_{K-1} \dots U_1 U_0)_L, \quad v = (V_{K-1} \dots V_1 V_0)_L \quad (42)$$

把 u 和 v 看做是 L 进制下的 K 位数, 因此每个数字 U_j 或 V_j 是一个 l 位整数。实际上, 对于所有 $j \geq K/2$, 前导数字 U_j 和 V_j 为 0, 因为 $2^{k-1}l \geq n$ 。以后我们将选择对于 k 和 l 的适当值; 我们暂时的目标是一般地看看发生什么情况, 使得当所有事实出现在我们面前时能明智地选择 k 和 l 。

乘法过程的下一步是计算序列 (u_0, \dots, u_{K-1}) 和 (v_0, \dots, v_{K-1}) 的傅里叶变换 $(\hat{u}_0, \dots, \hat{u}_{K-1})$ 和 $(\hat{v}_0, \dots, \hat{v}_{K-1})$, 其中我们定义

$$u_i = U_i / 2^{k+i}, \quad v_i = V_i / 2^{k+i} \quad (43)$$

这一调整是为了方便而进行的, 使得 u_i 和 v_i 小于 2^{-k} , 确保对于所有 s , 绝对值 $|a_s|$ 和 $|\hat{v}_s|$ 都将小于 1。

这里出现一个明显的问题, 因为复数 ω 不能精确地以二进制记号表示。那我们怎样计算一个可靠的傅里叶变换呢? 真走运, 如果我们仅仅以适当的精度进行计算, 则将一切顺利。我们暂时回避这个问题并假定用的是无限精度的计算, 后边我们将分析真正需要多少精度。

一旦已经找到 \hat{u}_s 和 \hat{v}_s , 我们就对于 $0 \leq s < K$ 令 $w_s = \hat{u}_s \hat{v}_s$, 并确定逆傅里叶变换 (w_0, \dots, w_{K-1}) 。如同上边说明的那样, 现在有

$$w_r = \sum_{i+j=r} u_i v_j = \sum_{i+j=r} U_i V_j / 2^{2k+2l}$$

使得整数 $W_r = 2^{2k+2l} w_r$ 是所求乘积

$$u \cdot v = W_{K-2} L^{K-2} + \dots + W_1 L_1 + W_0 \quad (44)$$

中的系数。由于 $0 \leq W_r < (r+1)L^2 < KL^2$, 每个 W_r 至多有 $k+2l$ 位, 所以当诸 W 已知时, 计算二进表示将是不难的, 除非 k 相对于 l 很大。

例如, 假设当参数是 $k=3$ 和 $l=4$ 时我们要来把 $u=1234$ 和 $v=2341$ 相乘。

从 u 开始对 (a_0, \dots, a_7) 的计算进行如下(请见(12)):

$$\begin{array}{l}
 (r, s, t) = (0, 0, 0) \quad (0, 0, 1) \quad (0, 1, 0) \quad (0, 1, 1) \quad (1, 0, 0) \quad (1, 0, 1) \quad (1, 1, 0) \quad (1, 1, 1) \\
 2^7 A^{[0]}(r, s, t) = \quad 2 \quad 13 \quad 4 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \\
 2^7 A^{[1]}(r, s, t) = \quad 2 \quad 13 \quad 4 \quad 0 \quad 2 \quad 13 \quad 4 \quad 0 \\
 2^7 A^{[2]}(r, s, t) = \quad 6 \quad 13 \quad 2 \quad 13 \quad 2+4i \quad 13 \quad 2-4i \quad 13 \\
 2^7 A^{[3]}(r, s, t) = \quad 19 \quad -7 \quad -2+13i \quad -2-13i \quad \alpha+\beta \quad \alpha-\beta \quad \alpha-\beta \quad \bar{\alpha}+\bar{\beta}
 \end{array}$$

这里 $\alpha = 2 + 4i$, $\beta = 13\omega$ 和 $\omega = (1+i)/\sqrt{2}$; 这给了我们在表 1 中的以 \hat{u}_s 打头的列。 \hat{v}_s 列以相同的方式从 v 得出; 然后我们把 \hat{u}_s 乘以 \hat{v}_s 得到 \hat{w}_s 。利用关系(40)转换再次给了我们 w_s 和 W_s 。再次地我们得到(19)中的卷积, 这次使用的是复数而不是坚持所有整数的方法。

现在让我们尝试估计一下, 如果在计算傅里叶变换中使用 m 个二进位的定点算术, 对于很大的数这个方法花费多少时间。习题 10 说明在转换计算的所有趟期间由于(43)的比例, 所有的量 $A^{[j]}$ 的数值都将小于 1, 因此对于所有中间的量的实部和虚部处理 m 个二进位的小数 $(.a_{-1} \dots a_{-m})_2$ 就足够了。由于输入 u_i 和 v_i 是实数值的, 因此简化是可能的; 在每步骤中只要进行 K 个实数值而不是 $2K$ 个实数值(参见习题 4.6.4-14)。我们将忽略这样的改进以便使复杂性保持极小。

表 1 通过离散傅里叶变换的乘法

s	$2^7 \hat{u}_s$	$2^7 \hat{v}_s$	$2^{14} \hat{w}_s$	$2^{14} w_s$	$2^{14} W_s$
0	19	16	304	80	10
1	$2+4i+13\omega$	$5+9i+2\omega$	$-26+64i+69\omega-125\bar{\omega}$	0	69
2	$-2+13i$	$-4+2i$	$-18-56i$	0	64
3	$2-4i-13\bar{\omega}$	$5-9i-2\bar{\omega}$	$-26-64i+125\omega-69\bar{\omega}$	0	125
4	-7	12	-84	288	36
5	$2+4i-13\omega$	$5+9i-2\omega$	$-26+64i-69\omega+125\bar{\omega}$	1000	0
6	$-2-13i$	$-4-2i$	$-18+56i$	512	0
7	$2-4i+13\bar{\omega}$	$5-9i+2\bar{\omega}$	$-26-64i-125\omega+69\bar{\omega}$	552	0

头一个工作是计算 ω 和它的幂, 为简便起见, 我们将构造值为 $\omega^0, \dots, \omega^{K-1}$ 的一张表。命

$$\omega_r = \exp(2\pi i/2^r) \quad (45)$$

使得 $\omega_1 = -1, \omega_2 = i, \omega_3 = (1+i)/\sqrt{2}, \dots, \omega_k = \omega$ 。若 $\omega_r = x_r + iy_r$, 则我们有 $\omega_{r+1} = x_{r+1} + iy_{r+1}$, 其中

$$x_{r+1} = \sqrt{\frac{1+x_r}{2}}, \quad y_{r+1} = \frac{y_r}{2x_{r+1}} \quad (46)$$

[参见 S. R. Tate, *IEEE Transactions* SP-43 (1995), 1709~1711。]相对于我们所需

要的其它计算说来, $\omega_1, \omega_2, \dots, \omega_k$ 的计算花费的时间可以忽略, 所以可以使用任何直截了当地计算平方根的算法。一旦已经计算了 ω_r , 我们就可以通过注意到

$$\omega' = \omega_1^{j_1-1} \cdots \omega_{k-1}^{j_{k-1}} \omega_k^{j_k} \quad \text{如果 } j = (j_{k-1} \cdots j_1 j_0)_2 \quad (47)$$

来计算所有乘幂 ω' 。这个计算方法防止误差传播, 因为每个 ω' 是顶多 k 个 ω_r 的乘积。计算所有 ω' 的全部时间是 $O(KM)$, 其中 M 是进行 m 个二进位复数乘法的时间, 因为从一个以前计算的值得到每个 ω' 只需要一个乘法就够了。随后的步骤将需要多于 $O(KM)$ 个周期, 所以 ω 的乘幂在可忽略的花费下就已可计算出来。

三个傅里叶变换的每一个都由 k 趟组成, 其中的每一趟包含形如 $a \leftarrow b + \omega^j c$ 的 K 个操作, 所以为计算傅里叶变换所需要的全部时间是

$$O(kKM) = O(Mnk/l)$$

最后, 使用(44)计算 uv 的诸二进数字所需工作量是 $O(K(k+l)) = O(n + nk/l)$ 。把所有这些操作加起来, 我们发现, 为乘 n 位数 u 和 v 的全部时间将是 $O(n) + O(Mnk/l)$ 。

现在看看中间精度 m 需要多大, 以便我们能知道 M 需要多大。为简便起见, 我们将满足于对精度的安全估计, 而不去寻找最好的界。以这样的方式来计算所有 ω' 是方便的, 即使得我们的近似 $(\omega')'$ 将满足 $|(\omega')'| \leq 1$; 如果我们向零的方向截断而不是舍入, 这个条件就容易保证, 因为在(46)中 $x_{r+1}^2 + y_{r+1}^2 = (1 + x_r^2 + y_r^2 + 2x_r)/(2 + 2x_r)$ 。对于 m 位定点复算术, 我们需要实施的操作都由以近似计算

$$a' \leftarrow \text{truncate}(b' + (\omega')'c') \quad (48)$$

代替形如 $a \leftarrow b + \omega^j c$ 的精确计算而得到, 其中 $b', (\omega')'$ 和 c' 是以前算出的近似值。所有这些复数和它们的近似绝对值都以 1 为界。如果 $|b' - b| \leq \delta_1, |(\omega')' - \omega| \leq \delta_2$ 和 $|c' - c| \leq \delta_3$, 不难看出将有 $|a' - a| < \delta + \delta_1 + \delta_2 + \delta_3$, 其中

$$\delta = |2^{-m} + 2^{-m}i| = 2^{1/2-m} \quad (49)$$

因为我们有 $|(\omega')'c' - \omega^j c| = |((\omega')' - \omega^j)c' + \omega^j(c' - c)| \leq \delta_2 + \delta_3$, 而且 δ 超过极大的截断误差。由(46)中定义的数的近似值 ω_r' 开始, 可得近似 $(\omega^j)'$, 而且我们可以假定(46)是以充分的精确性实施的, 使得 $|\omega_r' - \omega_r| < \delta$ 。于是(47)意味着对于所有的 j , $|(\omega^j)' - \omega^j| < (2k-1)\delta$, 因为这个误差是由于至多 k 个近似和 $k-1$ 个截断所致。

如果在任何一趟快速傅里叶变换之前有至多为 ϵ 的误差, 则该趟操作应有(48)的形式, 其中 $\delta_1 = \delta_3 = \epsilon$, 且 $\delta_2 = (2k-1)\delta$; 而且在这趟之后的误差至多将为 $2\epsilon + 2k\delta$ 。在第 0 趟没有误差, 所以通过对 j 用归纳法我们发现在第 j 趟之后的极大误差以 $(2^j - 1) \cdot 2k\delta$ 为界, 而且所计算的 a_j 值将满足 $|(a_j)' - a_j| < (2^k - 1) \cdot 2k\delta$ 。对于 $(\hat{w}_j)'$ 类似的公式成立; 而且我们将有

$$|(\hat{w}_j)' - \hat{w}_j| < 2(2^k - 1) \cdot 2k\delta + \delta < (4k2^k - 2k)\delta$$

在逆变换期间有一个附加的误差累计, 但除以 $K = 2^k$ 的除法会改进其中的大多数误差; 按照同样的论证, 我们发现所计算的值 w_r' 将满足

$$|(\hat{w}_r)' - \hat{w}_r| < 2^k(4k2^k - 2^k)\delta + (2^k - 1)2k\delta; |w_r' - w_r| < 4k2^k\delta \quad (50)$$

我们需要足够的精度使 $2^{2k+2l}w_r'$ 舍入到正确的整数 W_r , 因此我们需要

$$2^{2k+2l+2(1+\lg k)+k+1/2} m \leq \frac{1}{2} \quad (51)$$

即 $m \geq 3k + 2l + \lg k + 7/2$ 。如果我们只是要求

$$k \geq 7 \quad \text{和} \quad m \geq 4k + 2l \quad (52)$$

则这将成立。关系(41)和(52)可用来确定参数 k, l, m 使得乘法花费 $O(n) + O(Mnk/l)$ 个时间单位, 其中 M 是乘 m 位小数所需的时间。

比如说, 如果我们使用的是 MIX, 假设我们要来乘每个有 $n = 2^{13} = 8192$ 位数的二进制数。我们可以选择 $k = 11, l = 8, m = 60$, 使得必要的 m 位操作不过是双精度算术。因此, 进行定点 m 位复乘法的运行时间将相对地小。对于三倍精度运算我们可以增加到比如说 $k = l = 15, n \leq 15 \times 2^{14}$, 这超出了 MIX 的内存容量。在一个更大的机器上, 如果我们取 $k = l = 27$ 和 $m = 144$, 我们可以对一对吉比特的数做乘法。

关于 k, l 和 m 的选择的进一步研究导致令人惊奇的结论: 对于所有实用的目的来说, 我们可以假定 M 是常数, 而且 Schönhage-Strassen 乘法技术将有与 n 成线性比的运行时间。原因在于我们可以选择 $k = l$ 和 $m = 6k$; k 的这个选择总是小于 $\lg n$, 所以我们将绝不需要使用 6 倍以上的精度, 除非 n 大于我们的计算机字长(特别是 n 将要大于一个变址寄存器的容量, 所以我们大概不能把数 u 和 v 放入主存中)。

因此, 除开常数因式的改进之外, 快速乘法的实际问题已经解决了。事实上, 对于实用的高精度乘法来说, 习题 4.6.4-59 的全整数卷积算法大概是一个更好的选择。然而, 我们对于乘很大的数的兴趣有一部分是在理论上的, 因为探讨计算复杂性的最终极限是有趣的。所以让我们忘记实用的考虑并假设 n 极大, 也许比宇宙中原子的个数还大得多。可以令 m 近似于 $6 \lg n$, 而且递归地使用同一算法来做 m 个二进位的乘法。运行时间将满足 $T(n) = O(nT(\lg n))$; 因此

$$T(n) \leq Cn(C \lg n)(C \lg \lg n)(C \lg \lg \lg n) \cdots \quad (53)$$

其中乘积继续进行到满足 $\lg \cdots \lg n \leq 2$ 的一个因子为止。

Schönhage 和 Strassen 在他们的论文中指出如何通过使用整数 ω 来做模 $2^e + 1$ 的整数傅里叶变换, 以便把这个理论上界改进为 $O(n \log n \log \log n)$ 。这个上界可应用于图灵机, 即应用于具有有限内存的有限个任意长特的计算机。

如果允许用一个更强有力的计算机, 并且随机访问有限长的任意个字, Schönhage 已经指出上界降低到 $O(n \log n)$ 。因为我们可以选择 $k = l$ 和 $m = 6k$, 而且对于 $0 \leq x, y < 2^{\lceil m/12 \rceil}$ 我们有时间来构造所有可能的乘积 xy 的一个完备的乘法表(这种乘积的个数是 2^k 或 2^{k+1} , 而且我们可以在 $O(k)$ 步内通过由每个表项的前组项之一的加法计算出这个表项, 因此对于这个计算来说 $O(k2^k) = O(n)$ 步就足够了)。在这种情况下, M 是在 $2^{\lceil m/12 \rceil}$ 进制下进行 12 位算术所需要的时间, 而且

由此得出 $M = O(k) = O(\log n)$, 因为 1 位乘法可以通过查表完成。(假定访问内存的一个字的时间同该字的地址中的二进位数成正比例。)

而且, Schönage 在 1979 年发现, 一个指针机器可以在 $O(n)$ 步之内进行 n 位乘法; 请见习题 12。如同在 2.6 节末尾所讨论的, 当 $n \rightarrow \infty$ 时这样的装置(它也叫做“存储修改机器”和“链接自动机”)似乎提供了计算的最好模型。因此我们可以结论, 无论是对于理论的目的而言还是对实用而言, 在 $O(n)$ 步内进行乘法都是可能的。

1986 年, 由 D. V. Chudnovsky, G. V. Chudnovsky, M. M. Denneau 和 S. G. Younis 设计了一台称为小费马(Little Fermat)的不同寻常的通用计算机, 它具有快速地乘很大的整数的专门能力。它的硬件以快速的对 257 个二进位字的模 $2^{256} + 1$ 算术为特征; 使用 256 单字乘法, 连同只要求加法、减法和移位的三个离散变换一起, 可以完成 256 字的数组的一个卷积。基于近似 60ns 的一个流水线周期时间, 这使得有可能在少于 0.1s 的时间内来把两个 10^6 个二进位的整数相乘 [Proc. Third Int. Conf. on Supercomputing 2 (1988), 498 ~ 499; Contemporary Math. 143 (1993), 136]。

D. 除法 既然我们已经有了乘法的有效子程序, 现在考虑逆问题。结果是, 除一个常数因子外, 除法可以像乘法那样快地实现。

为了进行一个 n 位数 u 除以一个 n 位数 v 的除法, 我们可以首先找 $1/v$ 的一个 n 位近似值, 然后乘以 u 以得到对于 u/v 的一个近似值 q 。最后, 我们可以利用另一个乘法对 q 做一点必要的校正, 以确保 $0 \leq u - qv < v$ 。由此推理, 我们看到, 只要有一个近似一个 n 位数的倒数的算法就够了。下边的算法就是这样的, 它使用如 4.3.1 小节末尾所说明的“牛顿方法”。

算法 R(高精度倒数) 设 v 有二进表示 $v = (0.v_1v_2v_3\cdots)_2$, 其中 $v_1 = 1$ 。这个算法计算 $1/v$ 的近似值 z , 使得

$$|z - 1/v| \leq 2^{-n} \quad (54)$$

R1. [初始近似] 置 $z \leftarrow \frac{1}{4} \lfloor 32/(4v_1 + 2v_2 + v_3) \rfloor$, 和 $k \leftarrow 0$ 。

R2. [牛顿迭代] (这时有一个二进形式 $(xx.xx\cdots x)_2$ 的数 z , 它在小数点后有 $2^k + 1$ 位, 而且 $z \leq 2$ 。)利用高速乘法程序, 精确计算 $z^2 = (xxx.xx\cdots x)_2$ 。然后精确计算 $V_k z^2$, 其中 $V_k = (0.v_1v_2\cdots v_{2^{k+1}+3})_2$ 。然后置 $z \leftarrow 2z - V_k z^2 + r$, 其中必要时加上的 $0 \leq r < 2^{-2^{k+1}-1}$ 是为了舍入 z , 使它是 $2^{-2^{k+1}-1}$ 的倍数。最后置 $k \leftarrow k + 1$ 。

R3. [测试结束否] 如果 $2^k < n$, 则返回步骤 R2; 否则, 本算法终止。 ▮

这个算法是根据 S. A. Cook 的一个提议给出的。一项类似的技术已经用到计算机的硬件上 [见 Anderson, Earle, Goldschmidt 和 Powers, IBM J. Res. Dev. 11 (1967), 48 ~ 52]。当然, 有必要十分小心地校验算法 R 的精确性, 因为它可能非常

接近于不精确。我们在步骤 R2 开始和结束处,将用归纳法证明

$$z \leq 2 \quad \text{和} \quad |z - 1/v| \leq 2^{-2^k} \quad (55)$$

为了这个目的,设 $\delta_k = 1/v - z_k$, 其中 z_k 是步骤 R2 进行第 k 次迭代之后 z 的值。为了开始对 k 的归纳法,我们有

$$\delta_0 = 1/v - 8/v' + (32/v' - \lfloor 32/v' \rfloor)/4 = \eta_1 + \eta_2$$

其中 $v' = (v_1 v_2 v_3)_2$, $\eta_1 = (v' - 8v)/vv'$, 使得 $-\frac{1}{2} < \eta_1 \leq 0$, 以及 $0 \leq \eta_2 < \frac{1}{4}$ 。因此

$|\delta_0| < \frac{1}{2}$ 。现在假设(55)已经对 k 验证了;则

$$\begin{aligned} \delta_{k+1} &= 1/v - z_{k+1} = 1/v - z_k - z_k(1 - z_k V_k) - r = \\ &\delta_k - z_k(1 - z_k v) - z_k^2(v - V_k) - r = \\ &\delta_k - (1/v - \delta_k)v\delta_k - z_k^2(v - V_k) - r = \\ &v\delta_k^2 - z_k^2(v - V_k) - r \end{aligned}$$

现在

$$0 \leq v\delta_k^2 < \delta_k^2 \leq (2^{-2^k})^2 = 2^{-2^{k+1}}$$

而且

$$0 \leq z^2(v - V_k) + r < 4(2^{-2^{k+1}-3}) + 2^{-2^{k+1}-1} = 2^{-2^{k+1}}$$

所以 $|\delta_{k+1}| \leq 2^{-2^{k+1}}$ 。我们仍然必须验证(55)的头一个不等式;为证明 $z_{k+1} \leq 2$, 有三种情况:

a) $V_k = \frac{1}{2}$; 则 $z_{k+1} = 2$ 。

b) $V_k \neq \frac{1}{2} = V_{k-1}$; 则 $z_k = 2$, 所以 $2z_k - z_k^2 V_k \leq 2 - 2^{-2^{k+1}-1}$ 。

c) $V_{k-1} \neq \frac{1}{2}$; 则 $z_{k+1} = 1/v - \delta_{k+1} < 2 - 2^{-2^{k+1}} \leq 2$, 因为 $k > 0$ 。

算法 R 的运行时间以

$$2T(4n) + 2T(2n) + 2T(n) + 2T\left(\frac{1}{2}n\right) + \cdots + O(n)$$

步为界, 其中 $T(n)$ 是为进行 n 位数乘法所需时间的上界。如果对于某个单调非递减函数 $f(n)$, $T(n)$ 有 $nf(n)$ 的形式, 则我们有

$$T(4n) + T(2n) + T(n) + \cdots < T(8n) \quad (56)$$

所以除法可以用与乘法相当的速度进行, 只差一个常数因子。

R. P. Brent 已经证明, 如果乘 n 位数花费 $M(n)$ 个时间单位, 则像 $\log x$, $\exp x$ 和 $\arctan x$ 这样的函数可以在 $O(M(n) \log n)$ 步内求到 n 个有效位 [JACM 23 (1976), 242~251]。

E. 实时乘法 如果 n 位数的乘法可以仅仅在 n 步中实现, 那自然是很妙的。

我们已经从 n^2 阶降低到 n 阶,所以也许我们能把时间压缩到绝对极小值。事实上,如果我们离开通常的计算机程序设计领域,并且允许构造有无限多个全部一起动作的部件的计算机,则确实有可能如我们输入数字那样快地输出答案。

自动机的线性迭代阵列是一组设备 M_1, M_2, M_3, \dots , 在计算的每一步,它们每一个都可处于一个有限状态的集合中。诸机器 M_2, M_3, \dots 全都有相同的线路,而且它们在时间 $t+1$ 的状态是它们自己在时间 t 的状态以及它们的左邻和右邻在时间 t 的状态的函数。但头一台机器 M_1 稍微不同:它在时间 $t+1$ 时的状态是它自己和 M_2 在时间 t 的状态,以及在时间 t 时的输入的函数。一个线性迭代阵列的输出是一个定义在 M_1 的状态上的函数。

设 $u = (u_{n-1} \cdots u_1 u_0)_2, v = (v_{n-1} \cdots v_1 v_0)_2$ 和 $q = (q_{n-1} \cdots q_1 q_0)_2$ 是二进制数,并设 $uv + q = w = (w_{2n-1} \cdots w_1 w_0)_2$ 。一个值得注意的事实是,可以独立于 n , 构造一个线性迭代阵列,如果在时间 $0, 1, 2, \dots$ 时给定输入 $(u_0, v_0, q_0), (u_1, v_1, q_1), (u_2, v_2, q_2), \dots$, 它将在时间 $1, 2, 3, \dots$ 输出 w_0, w_1, w_2, \dots 。

我们可以用计算机硬件语言来叙述这个现象。为此只需指出有可能设计具有如下性质的一个集成线路模块:如果我们在一条直线上把充分多的集成片绕接在一起,让每一个模块仅与它的左邻和右邻通信,则得到的线路将恰在 $2n$ 个时钟脉冲内产生 n 位数的 $2n$ 位乘积。

其基本思想可如此理解:在时间 0, 机器 M_1 读出 (u_0, v_0, q_0) , 因而它能够在时间 1 输出 $(u_0 v_0 + q_0) \bmod 2$ 。然后在时间 2 它看到 (u_1, v_1, q_1) 而且它能够输出 $(u_0 v_1 + u_1 v_0 + q_1 + k) \bmod 2$, 其中 k 是由上一步留下的进位。其次它看到 (u_2, v_2, q_2) 并输出 $(u_0 v_2 + u_1 v_1 + u_2 v_0 + q_2 + k_2) \bmod 2$; 而且,它的状态保持 u_2 和 v_2 的值,使机器 M_2 能够在时间 3 读这些值,并且能够计算 $u_2 v_2$, 以便 M_1 在时间 4 获益。机器 M_1 实质上安排来启动 M_2 乘序列 $(u_2, v_2), (u_3, v_3), \dots$, 而 M_2 最终将给 M_3 以乘 $(u_4, v_4), (u_5, v_5)$ 等等的工作。幸而,诸项事务是在丝毫不损失时间之下完成的。读者将发现,从下面简略的描述中进一步推导它的细节是有趣的。

每个自动机有 2^{11} 个状态

$$(c, x_0, y_0, x_1, y_1, x, y, z_2, z_1, z_0)$$

其中 $0 \leq c < 4$, 而且诸 x 、诸 y 和诸 z 的每个或是 0 或是 1。开始时,所有设备都处于状态 $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ 。假设一台机器 $M_j, j > 1$, 在时间 t 处于状态 $(c, x_0, y_0, x_1, y_1, x, y, z_2, z_1, z_0)$, 而此时它的左邻 M_{j-1} 处于状态 $(c', x'_0, y'_0, x'_1, y'_1, x', y', z'_2, z'_1, z'_0)$, 它的右邻 M_{j+1} 处于状态 $(c'', x''_0, y''_0, x''_1, y''_1, x'', y'', z''_2, z''_1, z''_0)$, 则机器 M_j 将在时间 $t+1$ 进入状态 $(c', x'_0, y'_0, x'_1, y'_1, x', y', z'_2, z'_1, z'_0)$, 其中

$$\begin{aligned} c' &= \min(c+1, 3), \text{ 如果 } c' = 3, \text{ 否则 } 0 \\ (x'_0, y'_0) &= (x', y'), \text{ 如果 } c = 0, \text{ 否则 } (x_0, y_0) \\ (x'_1, y'_1) &= (x', y'), \text{ 如果 } c = 1, \text{ 否则 } (x_1, y_1) \end{aligned} \quad (57)$$

$$(x', y') = (x^l, y^l), \text{ 如果 } c \geq 2, \text{ 否则 } (x, y)$$

而且 $(z'_2 z'_1 z'_0)_2$ 是对于

$$z'_0 + z_1 + z'_2 + \begin{cases} x^l y^l & \text{如果 } c = 0 \\ x_0 y^l + x^l y_0 & \text{如果 } c = 1 \\ x_0 y^l + x_1 y_1 + x^l y_0 & \text{如果 } c = 2 \\ x_0 y^l + x_1 y + x y_1 + x^l y_0 & \text{如果 } c = 3 \end{cases} \quad (58)$$

的二进记号。最左边的机器 M_1 的动作几乎和其它机器相同;当它接受输入 (u, v, q) 时,它的动作就好像在它左边有一机器处于状态 $(3, 0, 0, 0, 0, u, v, q, 0, 0)$ 一样。阵列的输出是 M_1 的 z_0 部件。

表 2 示出了这个阵列在输入

$$u = v = (\cdots 00010111)_2, \quad q = (\cdots 00001011)_2$$

时的动作的一个例子。输出序列出现在 M_1 的状态的右下部分:

$$0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, \cdots$$

从右到左表示数 $(\cdots 01000011100)_2$ 。

这个构造是以和 A. J. Atrubin 首先发表的论文相类似的思想为基础的,见 *IEEE Trans. EC-14* (1965), 394~399。

这确实很快,但仅当输入的二进位在同一个时间抵达时这个迭代阵列才是最优的。如果输入的二进位全都同时出现,我们喜欢并行的线路,它在延迟 $O(\log n)$ 层次之后将得到两个 n 个二进位数的乘积。这类有效的线路已经由比如说 C. S. Wallace, *IEEE Trans. EC-13* (1964), 14~17; D. E. Knuth, *The Stanford GraphBase* (New York: ACM Press, 1994), 270~279 所描述。

S. Winograd [*JACM* 14 (1967), 793~802] 考察了当给定 n 且可同时一齐利用任意代码形式的输入时,在一条逻辑线路中可达到的极小乘法时间。当必须同时支持乘法和加法时,关于类似的问题,也见姚期智 (A. C. Yao), *STOC* 13 (1981), 308~311; Mansour, Nisan 和 Tiwari, *STOC* 22 (1990), 235~243。

表 2 在线性迭代阵列中的乘法

时间	输入	模块 M_1	模块 M_2	模块 M_3
	u, q v, q	$c \quad x_0 \quad x_1 \quad x \quad z_2$ $y_0 \quad y_1 \quad y \quad z_1$ z_0	$c \quad x_0 \quad x_1 \quad x \quad z_2$ $y_0 \quad y_1 \quad y \quad z_1$ z_0	$c \quad x_0 \quad x_1 \quad x \quad z_2$ $y_0 \quad y_1 \quad y \quad z_1$ z_0
0	$1 \quad 1$ $1 \quad 1$	$0 \quad 0 \quad 0 \quad 0 \quad 0$ $0 \quad 0 \quad 0 \quad 0 \quad 0$ 0	$0 \quad 0 \quad 0 \quad 0 \quad 0$ $0 \quad 0 \quad 0 \quad 0 \quad 0$ 0	$0 \quad 0 \quad 0 \quad 0 \quad 0$ $0 \quad 0 \quad 0 \quad 0 \quad 0$ 0
1	$1 \quad 1$ $1 \quad 1$	$1 \quad 1 \quad 0 \quad 0 \quad 0$ $1 \quad 1 \quad 0 \quad 0 \quad 1$ 0	$0 \quad 0 \quad 0 \quad 0 \quad 0$ $0 \quad 0 \quad 0 \quad 0 \quad 0$ 0	$0 \quad 0 \quad 0 \quad 0 \quad 0$ $0 \quad 0 \quad 0 \quad 0 \quad 0$ 0

黄金规则是我的绊脚石，
而实践使我发疯。

—MANUSCRIPT COLLECTED BY J. O. HALLIWELL (c. 1570)

习 题

1. [22] 如果以十进制代替二进制, (2) 中表达的思想可以推广成十进系统。试利用这个推广, 计算 1234 乘以 2341 (把这个四位数字的乘积归结成两位数字的数的三个乘积, 再把每个两位数乘积归结成一位数字的乘积)。

2. [M22] 证明在算法 T 的步骤 T1 中, 当我们置 $R \leftarrow \lfloor \sqrt{Q} \rfloor$ 时, R 的值或者保持相同, 或者增加 1 (因此, 像在该步骤所看到的一样, 我们不必计算平方根)。

3. [M22] 证明当 $k > 0$ 时, 算法 T 中定义的序列 q_k, r_k 满足不等式 $2^{q_k+1}(2r_k)^k \leq 2^{q_k-1+q_1}$ 。

► 4. [28] (K. Baker) 证明, 在点 $x = -r, \dots, 0, \dots, r$ 处 (而不是像算法 T 中那样在点 $x = 0, 1, \dots, 2r$ 处) 求多项式 $W(x)$ 的值是有利的。多项式 $U(x)$ 可以改写成

$$U(x) = U_e(x^2) + xU_o(x^2),$$

类似地, $V(x)$ 和 $W(x)$ 也可以这种方式展开。说明怎样利用这一思想, 以便在步骤 T7 和 T8 得到更快的计算。

► 5. [35] 证明如果在算法 T 的步骤 T1 处, 我们置 $R \leftarrow \lfloor \sqrt{2Q} \rfloor + 1$ 以代替 $R \leftarrow \lfloor \sqrt{Q} \rfloor$, 并给 q_0, q_1, r_0 和 r_1 以适当初始值, 则 (21) 可被改进成 $t_k \leq q_{k+1} 2^{\sqrt{2} \lg q_{k+1}} (\lg q_{k+1})$ 。

6. [M23] 证明 (24) 中的六个数两两互素。

7. [M23] 证明 (25)。

8. [M20] 正确或错误: 在 (39) 中我们可以忽略二进位的反序 $(s_{k-1}, \dots, s_0) \rightarrow (s_0, \dots, s_{k-1})$, 因为无论如何逆傅里叶变换将再次把这些二进位逆过来。

9. [M15] 假设以 ω^q 代替 ω 的所有出现, 应用正文的傅里叶变换的方法, 其中 q 是某个固定的整数。求由这个一般过程得到的数 $(\bar{u}_0, \bar{u}_1, \dots, \bar{u}_{K-1})$ 和当 $q=1$ 时得到的数 $(u_0, u_1, \dots, u_{K-1})$ 之间的一个简单关系。

10. [M26] (43) 中的调整使我们清楚地看到, 在 Schönhage-Strassen 乘法算法计算 a_i 和 \hat{v}_i 期间, 由变换子程序的趟 j 计算的所有复数 $A^{[j]}$ 的绝对值将小于 2^{j-k} 。证明在第三次傅里叶变换 (\hat{a}_i 的计算) 期间所有 $A^{[j]}$ 的绝对值将小于 1。

► 11. [M26] 如果 n 为定值, 在由 (57) 和 (58) 定义的线性迭代阵列之下为计算 n 位二进数的乘积将需要多少个自动机? (注意, 自动机 M_i 只受它右边的机器的分量 z_0^i 的影响, 所以每当输入是 n 位二进数时, 我们可以去掉其 z_0 分量总为 0 的所有自动机。)

► 12. [M41] (A. Schönhage) 本题的目的是证明指针机器的一个简单形式可以在 $O(n)$ 步内乘 n 位二进位数。这个机器没有用于算术的内部设备; 它的所有工作都是处理节点和指针。每个节点有相同的有限个数的链接字段, 并有有限多个链接寄存器, 这个机器可以做的仅有操作是:

i) 输入一个二进位, 如果该位为 0 则跳转;

ii) 输出 0 或 1;

iii) 以另一个寄存器的内容或者以一个由寄存器指向的节点中的链接字段的内容装入一个寄存器;

iv) 把一个寄存器的内容存入到由一个寄存器指向的节点的一个链接字段中;

- v) 如果两个寄存器相等, 则跳转;
- vi) 建立一个新节点并使一个寄存器指向它;
- vii) 停止。

试在这样一台机器上有效地实现傅里叶变换的乘法方法。[提示: 首先证明, 如果 N 是任何正整数, 则有可能建立 N 个节点表示整数 $\{0, 1, \dots, N-1\}$, 其中表示 p 的节点有指向表示 $p+1, \lfloor p/2 \rfloor$ 和 $2p$ 的节点的指针。这些节点可以在 $O(N)$ 步内建立。说明现在模拟 N 进制的算术毫无困难: 例如, 给定指向 p 和 q 的指针, 则可花费 $O(\log N)$ 步来找 $(p+q) \bmod N$ 的节点并且确定是否 $p+q \geq N$; 而且乘法可以在 $O(\log N)^2$ 步内模拟。现在考虑正文中的算法, 其中 $k=l$ 和 $m=6k$ 及 $N=2^{\lceil m/13 \rceil}$, 使得在定点算术计算中的所有量都是 N 进制的 13 位整数。最后, 利用下边的思想, 说明快速傅里叶变换的每一趟可以在 $O(K + (N \log N)^2) = O(K)$ 步内完成; K 个必要的赋值的每一个可以被“编译”成字长为 N 的类似 MIX 的模拟计算机的有限指令表, 而且如果 K 个这样机器的指令首先被排序, 使得所有相同的指令可被一起实现, 则并行运行的这些机器的指令可以在 $O(K + (N \log N)^2)$ 步内被模拟。(如果两个指令有相同的操作码, 相同的寄存器内容, 以及相同的内存操作数, 则这两个指令相同。)注意, $N^2 = O(n^{12/13})$, 所以 $(N \log N)^2 = O(K)$ 。]

13. [M25] (A. Schönhage) 当 m 和 n 都非常大但 n 比 m 大得多, 根据本小节对 $m=n$ 所证明的结果, 试说明什么是一个 m 位数乘以一个 n 位数所需要时间的好上界?

14. [M42] 写出算法 T 的一个程序, 并加入习题 4 的改进。试把它同算法 4.3.1M 的一个程序及以 (2) 为基础的一个程序做比较, 看看 n 必须多大才能使算法 T 真的成为一个改进。

15. [M49] (S. A. Cook) 如果第 k 位输出产生之前不从右到左地读入诸操作数的第 $k+1$ 位输入, 则说一个乘法算法是联机的。问什么是在各类自动机上可以实现的最快联机乘法算法?

► 16. [25] 证明, 即使当 K 不是 2 的一个乘幂时, 为计算离散傅里叶变换 (35), 只须花费 $O(K \log K)$ 个算术运算。[提示: 把 (35) 重写成形式

$$\hat{u}_j = \omega^{-j^2/2} \sum_{0 \leq r < k} \omega^{(r+1)j/2} \omega^{-r^2/2} u_r$$

并把这个和表达成一个卷积。]

17. [M26] Karatsuba 的乘法方案 (2) 做 K_n 个一位的乘法, 当它形成 n 位数的乘积时, 其中 $K_1=1, K_{2n}=3K_n$, 以及 $K_{2n+1}=2K_{n+1}+K_n, n \geq 1$ 。当 $n=2^{e_1}+2^{e_2}+\dots+2^{e_r}, e_1 > e_2 > \dots > e_r \geq 0$ 时, 通过求对于 K_n 的一个显式来“解”这个递推式。

► 18. [M30] 当通过基于 (2) 的一个递归算法实现乘法时, 试设计对中间结果分配内存的一个方案; 给定两个 N 位整数 u 和 v , 每一个都在内存的 N 个连续的位置中, 说明怎样来安排这个计算使得乘积出现在工作存储器的一个 $(3N + O(\log N))$ 位置的最低有效的 $2N$ 个位置中。

► 19. [M23] 如果允许你来测试一个操作数是否小于另一个, 说明如何以一个有界的运算个数来计算 $uv \bmod m$, 而这些运算须满足习题 3.2.1.1-11 的基础规则。 u 和 v 都是变量, 但 m 是常数。提示: 考虑 (2) 中的分解。

4.4 进制转换

如果我们的祖先已经发明了用他们的两个拳头或八个手指计数进行算术运算, 而不是用 10 个“数字”, 则我们就不必为写二进制转换程序而操心了 (而且也许我们根本不会去学那么多数系的知识了)。在本节, 我们将讨论从一种进制的位置记法到另一进制的位置记法的数的转换; 当然, 在二进计算机上把十进制的输入数据转

换成二进形式,以及把二进答案转换成十进形式的时候,这一过程最为重要。

A. 四个基本的方法 二进-十进转换是所有操作中同机器最有关的操作之一,因为计算机设计师们一直在想不同的办法用计算机硬件来实现它。因此,我们将仅仅讨论有关的一般原理,从这些原理出发,程序员可以选择最适合于他的机器的过程。

我们假定待变换的数都是非负的,因为符号的处理很容易解决。

假定我们正进行从 b 进制到 B 进制的转换(习题 1 和 2 中考虑混合进制的推广)。大多数进制转换子程序都是以乘法和除法为基础的,并利用下列四个方案之一。头两种方法适用于整数(即小数点在右边),其它方法适用于小数(即小数点在左边)。通常不可能精确地用一个有尽的 B 进制小数 $(0.U_{-1}U_{-2}\cdots U_{-M})_B$ 表达一个有尽的 b 进制小数 $(0.u_{-1}u_{-2}\cdots u_{-m})_b$ 。例如,分数 $\frac{1}{10}$ 有无穷的二进表示 $(0.0001100110011\cdots)_2$ 。因此有时需要把结果舍入到 M 位的一些方法。

• **方法 1a**(利用 b 进制算术除以 B) 给定一个整数 u ,我们可以得到它的 B 进制表示 $(\cdots U_2U_1U_0)_B$ 如下:

$$U_0 = u \bmod B, U_1 = \lfloor u/B \rfloor \bmod B, U_2 = \lfloor \lfloor u/B \rfloor / B \rfloor \bmod B, \cdots,$$

当 $\lfloor \cdots \lfloor \lfloor u/B \rfloor / B \rfloor \cdots / B \rfloor = 0$ 时停止。

• **方法 1b**(利用 B 进制的算术乘以 b) 如果 u 有 b 进制的表示 $(u_m \cdots u_1 u_0)_b$, 则我们可以利用 B 进制算术来求形如

$$((\cdots (u_m b^{m-1} + u_{m-1})b + \cdots)b + u_1)b + u_0$$

的多项式 $u_m b^m + \cdots + u_1 b + u_0 = u$ 的值。

• **方法 2a**(利用 b 进制的算术乘以 B) 给定一个小数 u ,我们可以逐个得到它的 B 进制表示 $(.U_{-1}U_{-2}\cdots)_B$ 的诸数字如下:

$$U_{-1} = \lfloor uB \rfloor, U_{-2} = \lfloor \{uB\}B \rfloor, U_{-3} = \lfloor \{\{uB\}B\}B \rfloor, \cdots$$

其中 $\{x\}$ 表示 $x \bmod 1 = x - \lfloor x \rfloor$ 。如果希望把结果舍入成 M 位,则在计算了 U_{-M} 之后,这个计算就可以停止,此时如果 $\{\cdots \{\{uB\}B\} \cdots B\}$ 大于 $\frac{1}{2}$,则 U_{-M} 应当加 1。(但要注意,这可能引起进位的传播,这些进位必须加到 B 进算术形式的答案中来。因此在计算开始之前,加一常数 $\frac{1}{2}B^{-M}$ 到原来的数 u 更为简单。但当 $\frac{1}{2}B^{-M}$ 不能像 b 进制数那样在计算机内精确地表示时,这可能导致一个不正确的答案。进一步要注意的是,如果 $b^m \geq 2B^M$,答案有可能舍入成 $(1.00\cdots 0)_B$ 。

习题 3 表明如何推广这个方法使得 M 可变,恰好大到足以把原来的数表示到指定的精度;在这种情况下不出现进位的问题。

·方法 2b(利用 B 进算术除以 b) 如果 u 有 b 进表示 $(0, u_{-1}u_{-2}\cdots u_{-m})_b$, 我们可以使用 B 进算术以下列形式

$$((\cdots(u_{-m}/b + u_{1-m})/b + \cdots + u_{-2})/b + u_{-1})/b$$

来计算 $u_{-1}b^{-1} + u_{-2}b^{-2} + \cdots + u_{-m}b^{-m}$ 。应该小心控制在除以 b 的过程中的截断或舍入误差, 这些通常是可以忽略的, 但不总是这样。

总之, 方法 1a, 1b, 2a 和 2b 给了我们转换整数的两个方法和转换小数的两个方法。而且, 通过乘以或除以 b 或 B 的适当次幂, 肯定可以在整数和小数之间进行转换, 因此, 当试图进行转换时至少有四种方法可供选择。

B. 单精度转换 为了说明这四种方法, 假定 MIX 是一台二进制计算机, 并且假定我们要把一个二进制整数 u 转换成为一个十进制整数, 于是 $b=2$ 和 $B=10$ 。可把方法 1a 编成程序如下:

ENT1	0	置 $j \leftarrow 0$	
LDX	U		
ENTA	0	置 $rAX \leftarrow u$	
1H	DIV	$= 10 =$	$(rA, rX) \leftarrow (\lfloor rAX/10 \rfloor, rAX \bmod 10)$
			(1)
STX	ANSWER, 1	$U_j \leftarrow rX$	
INCL	1	$j \leftarrow j + 1$	
SRAX	5	$rAX \leftarrow rA$	
JXP	1B	重复直到结果为 0	■

为得到 M 个数字, 这需要 $18M + 4$ 个周期。

方法 1a 使用除以 10 的除法; 方法 2a 使用乘以 10 的乘法, 所以它可能稍微快一点。但为了使用方法 2a, 我们必须处理小数, 而这导致了一种有趣的情况。设 w 是计算机字长, 并假定 $u < 10^n < w$ 。通过做一次除法, 我们可以求出 q 和 r , 其中

$$wu = 10^n q + r, \quad 0 \leq r < 10^n \quad (2)$$

现在如果把方法 2a 应用于小数 $(q+1)/w$, 则我们将会在 n 步中得到 u 从左到右的诸数字, 因为

$$\left\lfloor 10^n \frac{q+1}{w} \right\rfloor = \left\lfloor u + \frac{10^n - r}{w} \right\rfloor = u. \quad (3)$$

(这个思想是 P. A. Samet 给出的, 见 *Software Practice & Experience* 1 (1971), 93~96。)

下面是对应的 MIX 程序:

	JOV	OFLO	确保溢出开关断开
	LDA	U	
	LDX	$= 10^n -$	$rAX \leftarrow wu + 10^n$
	DIV	$= 10^n =$	$rA \leftarrow q \cdot 1, rX \leftarrow r$
	JOV	ERROR	如果 $u \geq 10^n$ 则跳转
	ENT1	$n - 1$	置 $j \leftarrow n - 1$
2H	MUL	$= 10 =$	现在想像小数点在左边, $rA = x$
	STA	ANSWER, 1	置 $U_j \leftarrow \lfloor 10x \rfloor$
	SLAX	5	$x \leftarrow \{10x\}$
	DEC1	1	$j \leftarrow j - 1$
	JLNN	2B	对 $n > j \geq 0$ 重复执行

(4)

这个程序稍长一点,需要 $16n + 19$ 个周期,所以如果 $n = M \geq 8$,它比程序(1)稍快些;当存在前导0时,(1)将更快些。

当 $10^m < w < 10^{m+1}$ 时,目前的程序(4)不能用来转换整数 $u \geq 10^m$,因为我们需要取 $n = m + 1$ 。在这种情况下,通过计算 $\lfloor u/10^m \rfloor$ 我们可以得到 u 的前导数字;然后当 $n = m$ 时, $u \bmod 10^m$ 即可如同上边那样进行转换。

从左到右地得到答案数字这一事实在某些应用中可能是一个优点(例如,当一次打出答案的一位数字时)。于是我们看见,尽管使用不精确除法时要附带做一点数值分析,但它可以使分数方法能用来进行整数的转换。

对方法 1a 做修改可以避免除以 10,办法是用两个乘法来代替它。这个修改是重要的,因为进制转换通常是通过没有除法能力的小的“卫星”计算机完成的。如果 x 是 $1/10$ 的一个近似,使得

$$\frac{1}{10} < x < \frac{1}{10} + \frac{1}{w}$$

那么容易证明(见习题 7) $\lfloor ux \rfloor = \lfloor u/10 \rfloor$ 或 $\lfloor u/10 \rfloor + 1$, 只要 $0 \leq u < w$ 。因此,如果计算 $u - 10\lfloor ux \rfloor$,我们将有能力确定 $\lfloor u/10 \rfloor$ 的值:

$$\lfloor u/10 \rfloor = \lfloor ux \rfloor - \lfloor u < 10\lfloor ux \rfloor \rfloor \quad (5)$$

同时我们还将确定 $u \bmod 10$ 。利用(5)进行转换的一个 MIX 程序出现在习题 8 中,对每位数字它要求大约 33 个周期。

如果计算机在它内部的指令系统中既无除法也无乘法,如同习题 9 中所说的那样,通过审慎地进行移位和加法,我们仍然能使用方法 1a 来进行转换。

从二进制转换到十进制的另一个方法是使用方法 1b,但为做到这一点我们需要在一个十进数系中模拟加倍。这个方法一般来说最适合于加到计算机硬件中去;然而如表 1 所示,利用二进加法、二进移位及二进抽取或屏蔽(在寄存器中对每一位进行“逻辑与”),即可把十进数的加倍过程编成程序,这是由 Peter L. Montgomery 提出的。

表 1 加倍一个二进编码的十进数

运算	一般形式	例子
1. 给定的数	$u_{11} u_{10} u_9 u_8 u_7 u_6 u_5 u_4 u_3 u_2 u_1 u_0$	0011 0110 1001 = 369
2. 每个数字加 3	$v_{11} v_{10} v_9 v_8 v_7 v_6 v_5 v_4 v_3 v_2 v_1 v_0$	0110 1001 1100
3. 抽取每个高二进位	$v_{11} \quad 0 \quad 0 \quad 0 \quad v_7 \quad 0 \quad 0 \quad 0 \quad v_3 \quad 0 \quad 0 \quad 0$	00001 0001 000
4. 右移两位和减	$0 \quad v_{11} \quad v_{10} \quad 0 \quad v_7 \quad v_6 \quad 0 \quad 0 \quad v_3 \quad v_2 \quad 0$	0000 0110 0110
5. 加上原来的数	$w_{11} w_{10} w_9 w_8 w_7 w_6 w_5 w_4 w_3 w_2 w_1 w_0$	0011 1100 1111
6. 加上原来的数	$x_{12} x_{11} x_{10} x_9 x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0$	0 0111 0011 1000 = 738

当 $0 \leq d \leq 4$ 时这个方法把每个数字 d 改变成为 $2d$, 而当 $5 \leq d \leq 9$ 时, 改变成为 $6 + 2d = (2d - 10) + 2^4$; 对于用 4 个二进位编一个数字的十进数来说, 这刚好是加倍所需要的。

另外一个有关的思想, 是记住十进形式下 2 的乘方的一张表, 并且通过模拟十进加法来把适当的乘方加到一起, 在 7.1 节中有关于位处理技术的综述。

最后, 甚至方法 2b 也能用于二进整数到十进整数的转换。我们可以像在 (2) 中那样求 q , 而后模拟 $q + 1$ 除以 w 的十进除法, 办法是利用一个类似于刚才描述的加倍过程的“取半”过程(习题 10), 且仅仅保留答案中小数点右边的头几位。在这种情况下, 方法 2b 似乎并不比已经讨论过的其它三种方法有什么优点, 但我们已经确认了早先所做的说明, 即对于从一个进制到另一个进制的整数的转换, 至少有四种不同的方法可以利用。

现在我们考虑十进-二进转换($b = 10, B = 2$)。方法 1a 模拟除以 2 的十进除法; 这是能行的(见习题 10), 但它主要适合于做成硬件, 而不是编成程序。

在绝大多数情况下, 方法 1b 是进行十进-二进转换最实用的方法。下面是它的 MIX 代码, 其中假定在被转换的数 $(u_m \cdots u_1 u_0)_{10}$ 中至少有两位数字, 并假定 $10^{m+1} < w$, 使得溢出不成为问题:

```

ENT1  M - 1          置  $j \leftarrow m - 1$ 
LDA   INPUT + M      置  $U \leftarrow u_m$ 
1H   MUL    = 10 =
SLAX  5
ADD   INPUT, 1         $U \leftarrow 10U + u_j$ 
DEC1  1
JINN  1B              对于  $m > j \geq 0$  重复

```

(6)

乘以 10 可由加法和移位代替。

习题 19 中描述了一个更复杂但或许更快的方法, 它使用了大约 $\lg m$ 个乘法、抽取和加法, 而不是 $m - 1$ 个乘法和加法。

对于十进小数 $(0.u_{-1}u_{-2}\cdots u_{-m})_{10}$ 到二进制的转换, 我们可以使用方法 2b; 或者更一般地, 我们可以通过方法 1b 转换整数 $(u_{-1}u_{-2}\cdots u_{-m})_{10}$, 然后除以 10^m 。

C. 手算 计算机程序员偶尔也有必要用手算来转换数, 由于这是在小学未曾讲过的一个课题, 因此可能值得在这里简单地考查一下。有一些非常简单的手算方法可进行十进制和八进制记号之间的转换, 这很容易学会, 所以应当广为普及。

八进制整数转换为十进制 最简单的转换是从八进制到十进制; 这个技术看来首先是由 Walter Soden 发表的, 见 *Math. Comp.* 7 (1953), 273~274。为了进行这种转换, 写下给定的八进整数, 然后在第 k 步利用十进算术加倍前 k 个前导数字, 并利用十进算术从 $k+1$ 个前导数字减去这数。如果给定的数有 $m+1$ 位数字, 则这个过程终止于 m 步。如同这里给出的例子那样, 为了防止产生使人烦恼的错误, 插入一个小数点来说明加倍哪些数字乃是一个好的想法。

例 1 把 $(5325121)_8$ 转换成十进制的数。

$$\begin{array}{r}
 5.3\ 2\ 5\ 1\ 2\ 1 \\
 -1\ 0 \\
 \hline
 4\ 3.2\ 5\ 1\ 2\ 1 \\
 -\ 8\ 6 \\
 \hline
 3\ 4\ 6.5\ 1\ 2\ 1 \\
 -\ 6\ 9\ 2 \\
 \hline
 2\ 7\ 7\ 3.1\ 2\ 1 \\
 -\ 5\ 5\ 4\ 6 \\
 \hline
 2\ 2\ 1\ 8\ 5.2\ 1 \\
 -\ 4\ 4\ 3\ 7\ 0 \\
 \hline
 1\ 7\ 7\ 4\ 8\ 2.1 \\
 -\ 3\ 5\ 4\ 9\ 6\ 4 \\
 \hline
 1\ 4\ 1\ 9\ 8\ 5\ 7
 \end{array}$$

答案: $(1419857)_{10}$

对于这些计算的一项相当好的检验, 可以通过“去 9”方法给出。十进数数字的和必须模 9 同余于八进数数字的交替的和与差, 且后者的最右边数字应给成正号。

在上述例子中, 我们有 $1+4+1+9+8+5+7=35$, 以及 $1-2+1-5+2-3+5=-1$, 因此差是 36 (9 的倍数)。如果这项测试失败, 则它还可以应用到第 k 步之后的 $k+1$ 个前导数字, 而且利用一个“二进检索”程序, 可以确定出错的位置; 换句话说, 通过首先检验中间结果, 然后依据中间结果是正确还是不正确, 把同一过程使用于这个计算的头一半或第二半去, 可以找出错误的位置。

当然, “去 9”过程仅仅有 89% 的可靠性, 因为有几分之一的可能出现两个随机整数相差 9 的一个倍数的情况。一个更好的检验是使用求逆方法, 把答案转换回来成为八进制数。我们现在就来考虑这个方法。

把十进制整数转换成八进制 类似的过程可用来进行相反的转换: 写下给定的十进整数, 然后在第 k 步利用八进算术加倍 k 个前导数字, 并且利用八进算术把这些数字加到 $k+1$ 个前导数字上。如果给定的数有 $m+1$ 个数字, 这一过程将终止于 m 步。

例 2 把 $(1419857)_{10}$ 转换成八进制的数。

$$\begin{array}{r}
 1\ 4\ 1\ 9\ 8\ 5\ 7 \\
 +\ \underline{2} \\
 1\ 6\ 1\ 9\ 8\ 5\ 7 \\
 +\ \underline{3\ 4} \\
 2\ 1\ 5\ 9\ 8\ 5\ 7 \\
 +\ \underline{4\ 3\ 2} \\
 2\ 6\ 1\ 3\ 8\ 5\ 7 \\
 +\ \underline{5\ 4\ 2\ 6} \\
 3\ 3\ 5\ 6\ 6\ 5\ 7 \\
 +\ \underline{6\ 7\ 3\ 5\ 4} \\
 4\ 2\ 5\ 2\ 4\ 1\ 7 \\
 +\ \underline{1\ 0\ 5\ 2\ 5\ 0\ 2} \\
 5\ 3\ 2\ 5\ 1\ 2\ 1
 \end{array}$$

答案: $(5325121)_8$ 。

(注意非八进制数字 8 和 9 进入了这个八进制计算中。)答案可如同上述的讨论那样进行校验。这个方法是 Charles P. Rozier 发表的, 见 *IEEE Trans. CE-11*(1962), 708 ~ 709。

刚才给出的两个过程实质上是一般进制转换过程的方法 1b。十进记号下的加倍和减法就好像乘以 $10 - 2 = 8$; 八进记号下的加倍和加法就好像乘以 $8 + 2 = 10$ 。十六进制/十进制的转换有类似的一个方法, 但它稍微麻烦些, 因为它包含乘以 6 的乘法而不是乘以 2。

为了把这两个方法保留在我们的脑海中, 不难记住, 当从八进制转换成十进制时, 我们必须做减法, 因为一个数的十进制表示是更小的; 类似地, 当从十进制转换成八进制时, 我们必须做加法。计算利用的是答案的进制, 而不是利用给定数的进制, 不然我们就不能得到所求的答案。

转换小数 还没有发现用人工同样快速地转换小数的方法。最好的方法似乎是方法 2a, 并且用加倍和加法或减法来简化乘以 10 或乘以 8。在这种情况下, 我们把加法-减法的准则反过来, 当转换成十进制时做加法, 而当转换成八进制时做减法。在这个计算中, 我们使用给定的输入数的进制, 而不是答案的进制(见例 3 和 4)。这个过程的难度大约是上述对于整数的方法的两倍。

例 3 把 $(.14159)_{10}$ 转换成八进制的数。

$$\begin{array}{r}
 .1\ 4\ 1\ 5\ 9 \\
 \underline{2\ 8\ 3\ 1\ 8-} \\
 1.1\ 3\ 2\ 7\ 2 \\
 \underline{2\ 6\ 5\ 4\ 4-} \\
 1.0\ 6\ 1\ 7\ 6 \\
 \underline{1\ 2\ 3\ 5\ 2-}
 \end{array}$$

$$\begin{array}{r}
 0.49408 \\
 \underline{98816} \cdot \\
 3.95246 \\
 \underline{190528} - \\
 7.62112 \\
 \underline{124224} - \\
 4.96896
 \end{array}$$

答案: $(.110374\cdots)_8$ 例4 将 $(.110374)_8$ 转换成十进制的数。

$$\begin{array}{r}
 .110374 \\
 \underline{220770} + \\
 1.324730 \\
 \underline{651660} + \\
 4.121160 \\
 \underline{242340} + \\
 1.454140 \\
 \underline{1130300} + \\
 5.671700 \\
 \underline{1563600} + \\
 8.502600 \\
 \underline{1205400} + \\
 6.233400
 \end{array}$$

答案: $(.141586\cdots)_{10}$

D. 浮点转换 当要转换的是浮点值时,有必要同时处理指数和小数部分,因为指数的转换将影响小数部分。设要转换成十进制的数是 $f \cdot 2^e$, 我们可以用 $F \cdot 10^E$ 的形式来表达 2^e (通常借助于辅助表), 而后把 Ff 转换成为十进制。或者, 我们可以用 $\log_{10} 2$ 来乘 e , 并把这结果舍入成最接近的整数 E , 然后 $f \cdot 2^e$ 除以 10^E , 并转换这个结果。反之, 假定要转换成二进制的数是 $F \cdot 10^E$, 我们可以转换 F 而后以浮点数 10^E 来乘它 (再次使用辅助表)。有一些现成的方法, 通过利用若干乘法和 (或) 除法来减少辅助表的最大长度, 尽管这可能引起舍入误差的传播。习题 17 考虑误差的极小化。

E. 多精度转换 当转换极长的数时, 最方便的是由转换数字块开始, 这些数字段可以用单精度技术来处理, 而后用简单的多精度技术把这些块组合在一起。例如, 假设 10^n 是小于计算机字长的 10 的最高乘幂, 则

a) 为把一个多精度整数从二进制转换成十进制, 应反复地以 10^n 除它 (于是用方法 1a 从二进制转换成 10^n 进制)。对于 10^n 进制表示的每一位单精度操作给出 n 个十进数字。

b) 为把一个多精度小数从二进制转换成十进制, 方法是类似的, 乘以 10^n 即可

(即对于 $B = 10^n$ 使用方法 2a)。

c) 为把一个多精度整数从十进制转换成二进制, 首先转换每块 n 个数字的诸字块, 然后使用方法 1b 从 10^n 进制转换成二进制。

d) 为把一个多精度小数从十进制转换成二进制, 如在 c) 中那样, 应首先转换成 10^n 进制, 然后使用方法 2b。

F. 历史和文献 进制转换技术起源于古代处理重量、测量和货币的一些问题中, 其中一般都涉及混合进制系统; 通常都要编制辅助表来帮助进行转换。17 世纪时, 当十六进制小数正为十进制小数所取代时, 为了能使用天文数字表的现成书籍, 有必要在两个系统之间进行转换; 把小数从六十进制转换成为十进制及反过来的系统方法是在 William Oughtred 的 *Clavis Mathematicæ* 的 1667 年版, 第 6 章, 第 18 节中给出的(这个材料在 1631 年出的 Oughtred 的原版书中还未出现)。塔吉克斯坦撒马尔罕(Samarkand)的 al-Kāshī 在他的 *Key to Arithmetic* (1427) 中已经给出一些转换规则, 其中方法 1a, 1b 和 2a 已被清楚地说明 [Istoriko-Mat. Issled. 7 (1954), 126~135], 但他的工作尚不为欧洲人所知。18 世纪美国数学家 Hugh Jones 使用 “octavation” 和 “decimation” 两词来描述八进/十进转换, 但他的方法不像他的术语那样聪明。A. M. Legendre [*Théorie des Nombres* (Paris: 1798), 229] 说明, 如果反复地除以 64, 则正整数可以方便地转换成二进形式。

1946 年, H. H. Goldstine 和 J. von Neumann 在他们的经典的学术论文 *Planning and Coding Problems for an Electronic Computing Instrument* 中着重讨论了进制转换问题, 因为有必要来论证使用二进制算术的正当性; 见 John von Neumann, *Collected Works* 5 (New York: Macmillan, 1963), 127~142。关于在二进制计算机上进行转换的另一项早期的讨论是由 F. Koons 和 S. Lubkin 发表的, 见 *Math. Comp.* 3 (1949), 427~431, 其中他们提出了一个不太寻常的方法。关于浮点转换的第一次讨论是由 F. L. Bauer 和 K. Samelson 在稍后一些时间给出的 [*Zeit. für angewandte Math. und Physik* 4 (1953), 312~316]。

类似地, 下列论文有着历史意义: G. T. Lake 的一篇笔记 [*CACM* 5 (1962), 468~469] 提到了用于进行转换的一些硬件技术并且给出了一些清楚的例子。A. H. Stroud 和 D. Secrest [*Comp. J.* 6 (1963), 62~66] 讨论了多精度浮点数的转换问题。能保留表示中所蕴涵的“有效位”个数的非规格化浮点数的转换问题, 已由 H. Kanner [*JACM* 12 (1965), 242~246] 与 N. Metropolis 和 R. L. Ashenurst [*Math. Comp.* 19 (1965), 435~441] 做了讨论。也见 K. Sikdar, *Sankhyā* B30 (1968), 315~334 及他的文章中引用的参考文献。

P. J. Plauger 在 *The Standard C Library* (Prentice Hall, 1992), 301~331 上给出了 C 程序设计语言中对于整数和浮点数格式化输入输出的详细的子程序。

习 题

►1.[25] 推广方法 1b, 使它对混合进制记法有效, 转换 $a_m b_{m-1} \cdots b_1 b_0 + \cdots + a_1 b_0 + a_0$ 成为 $A_M B_{M-1} \cdots B_1 B_0 + \cdots + A_1 B_0 + A_0$, 其中对于 $0 \leq j < m, 0 \leq J < M$, 有 $0 \leq a_j < b_j, 0 \leq A_J < B_J$ 。

给出你的推广的一个例子, 通过手算来把量“3 天, 9 小时, 12 分, 37 秒”转换成为吨, 担, 石, 磅和盎司。(设 1 秒等于 1 盎司, 英制的重量单位是 1 石 = 14 磅, 1 担 = 8 石, 1 吨 = 20 担。)换言之, 设 $b_0 = 60, b_1 = 60, b_2 = 24, m = 3, B_0 = 16, B_1 = 14, B_2 = 8, B_3 = 20, M = 4$; 问题是要使用推广方法 1b 的一个系统方法, 求出适当范围里的 A_4, \cdots, A_0 , 使得 $3b_2 b_1 b_0 + 9b_1 b_0 + 12b_0 + 37 = A_4 B_3 B_2 B_1 B_0 + A_3 B_2 B_1 B_0 + A_2 B_1 B_0 + A_1 B_0 + A_0$ 。(所有算术都要在混合进制系统下完成。)

2.[25] 如在习题 1 中那样推广方法 1a, 使它对于混合进制记法有效。作为你的推广的举例, 请通过手算解决如习题 1 中所述的同样的转换问题。

►3.[25](D. Taranto) 当对小数进行转换时, 一般没有明显的方法来判定在答案中应给出多少位。试设计方法 2a 的一个简单的推广, 使得: 给定 0 和 1 之间的两个正 b 进制小数 u 和 ϵ , 把 u 转换成舍入后的 B 进制等价数 U , 这个 U 应在小数点右边有足够多的位数 M 位, 以保证 $|U - u| < \epsilon$ 。(特别是, 如果 u 是 b^{-m} 的倍数且 $\epsilon = b^{-m}/2$, 则 U 的值将正好有足够的数字, 使得当给定 U 和 m 时, 可以精确地把 u 重新计算出来。注意, M 可以是零。例如, 若 $\epsilon \leq \frac{1}{2}$ 且 $u > 1 - \epsilon$, 则正确的答案应是 $U = 1$ 。)

4.[M21] (a) 证明每一个具有有尽的二进表示的实数必有一个有尽的十进表示。(b) 试求关于正整数 b 和 B 的一个简单条件, 使得当且仅当每一个有尽的 b 进制表示的实数, 也有有尽的 B 进制表示时, 它被满足。

5.[M20] 证明若指令“LDX = 10”被代之以对于某个其它常数 c 的“LDX = c ”, 则程序 (4) 仍将有效。

6.[30] 当 b 或 B 是 -2 时, 讨论方法 1a, 1b, 2a 和 2b 的使用。

7.[M18] 假定 $0 < a \leq x \leq a + 1/w$ 以及 $0 \leq u \leq w$, 其中 u 是一个整数, 证明 $\lfloor ux \rfloor$ 等于 $\lfloor au \rfloor$ 或 $\lfloor au \rfloor + 1$ 。而且, 如果 $u < aw$ 和 a^{-1} 是一个整数, 则精确地有 $\lfloor ux \rfloor = \lfloor au \rfloor$ 。

8.[24] 写出一个类似于 (1) 的 MIX 程序, 它使用 (5) 而且不包含任何除法指令。

►9.[M29] 本题的目的是当 u 是一个非负整数时, 仅仅通过二进制移位、屏蔽和加法运算来计算 $\lfloor u/10 \rfloor$ 和 $u \bmod 10$ 。设 $k \geq 2$ 是一个固定整数, 并考虑计算

$$v \leftarrow u + 1, v \leftarrow v + \left\lfloor \frac{v}{2} \right\rfloor, v \leftarrow v + \left\lfloor \frac{v}{16} \right\rfloor, v \leftarrow v + \left\lfloor \frac{v}{256} \right\rfloor, v \leftarrow v + \left\lfloor \frac{v}{2^{2^k}} \right\rfloor$$

$$q \leftarrow \left\lfloor \frac{v}{16} \right\rfloor, r \leftarrow v \bmod 16, r \leftarrow r + \left\lfloor \frac{r}{4} \right\rfloor, r \leftarrow \left\lfloor \frac{r}{2} \right\rfloor$$

使得 $q \neq \lfloor u/10 \rfloor$ 或 $r \neq u \bmod 10$ 的最小正整数 u 是多少?

10.[22] 表 1 示出如何通过在一台二进计算机上使用各种移位、抽取和加法操作来把一个二进编码的十进数加倍。试给出一个类似的方法, 它计算一个二进编码的十进数之半。(如果该数是奇数, 则抛掉其剩余。)

11.[16] 把 $(57721)_8$ 转换成十进制。

►12.[22] 想出把整数从三进制记法转换成十进制记法的一个快速的手算方法, 并通过把 $(1212011210210)_3$ 转换成十进制来说明你的方法。你将怎样从十进制转换成三进制?

►13. [25] 假设单元 $U+1, U+2, \dots, U+m$ 包含一个多精度小数 $(.u_{-1}u_{-2}\dots u_{-m})_b$, 其中 b 是 MIX 的字长。试写出一个 MIX 程序, 它把这个小数转换成十进记法, 并把它截取成为 180 位十进数字。答案应打印成两行, 而且数字 9 位一组, 分成 20 块, 每块用空白分隔开来(用 CHAR 指令)。

►14. [M27](A. Schönhage) 当 n 很大时, 正文中对多精度整数进行转换的方法要用 n^2 阶的执行时间转换一个 n 位整数。试证有可能在 $O(M(n) \log n)$ 步内把 n 位十进制整数转换成二进制记法, 其中 $M(n)$ 是为乘 n 位二进制数所需要的步数, 它满足“光滑性条件” $M(2n) \geq 2M(n)$ 。

15. [M47] 在习题 14 中给出的转换大整数的时间上界, 能否大大降低(参考习题 4.3.3-12)?

16. [41] 试构造一个从十进制转换成二进制的进制转换的快速线性迭代阵列(参考 4.3.3F 小节)。

17. [M40] 试设计一个“理想的”浮点转换子程序, 它把 p 位十进制数转换成 P 位二进制数, 而且反过来也行, 在两种情况下, 都产生 4.2.2 小节意义下的真正的舍入结果。

18. [HM34](David W. Matula) 设 $\text{round}_b(u, p)$ 是在 4.2.2 小节的意义下 b, u 和 p 的函数, 它表示 u 的最好 p 位 b 进制浮点近似。在 $\log_b b$ 是无理数及指数范围无限的假定下证明: 当且仅当 $B^{p-1} \geq b^p$ 时, 对于所有 p 位 b 进制的浮点数 u ,

$$u = \text{round}_b(\text{round}_B(u, P), p)$$

成立。(换言之, 如同上面这个公式所描述的那样, 对 u 做到一个无关的 B 进制的“理想的”输入转换, 接着再对这个结果做“理想的”输出转换, 将总是再产生出 u 来, 当且仅当中间的精确性 P 是适当地大时。)

19. [M23] 设十进制数 $u = (u_7 \dots u_1 u_0)_{10}$ 被表示成二进编码的十进制数 $U = (u_7 \dots u_1 u_0)_{16}$ 。求适当的常数 c_i 和屏蔽码 m_i , 使得对 $i = 1, 2, 3$, 重复的操作 $U \leftarrow U - c_i (U \wedge m_i)$ 将把 U 转换成为 u 的二进制表示, 其中“ \wedge ”表示抽取(即按位的“逻辑与”)。

4.5 有理算术

知道某个数值问题的答案恰是 $1/3$, 而不是打印成“0.333333574”的一个浮点数, 通常是重要的。如果以分数而不是以分数的近似来进行算术运算, 则许多计算可以完全没有任何累计舍入误差。这就得到令人快慰的安全感。而通常在进行浮点计算时缺乏这种感觉, 这意味着计算的精度不能再有所改进。

4.5.1 分数

当要进行分数算术时, 可以把数表示成整数偶 (u/u') , 其中 u 和 u' 互素, 而且 $u' > 0$ 。数 0 表示为 $(0/1)$ 。在这种形式下, 当且仅当 $u = v$ 和 $u' = v'$ 时, $(u/u') = (v/v')$ 。

当然, 分数的乘法较为简单; 为形成 $(u/u') \times (v/v') = (w/w')$, 我们可以简单地计算 uv 和 $u'v'$ 。两个乘积 uv 和 $u'v'$ 可以不是互素的, 但如果 $d = \text{gcd}(uv, u'v')$, 则所求的答案就是 $w = uv/d, w' = u'v'/d$ (见习题 2)。4.5.2 小节讨论了计算最大公因子的有效方法。

实施乘法的另一个方法是求 $d_1 = \text{gcd}(u, v')$ 和 $d_2 = \text{gcd}(u', v)$; 于是答案是 w

$= (u/d_1)(v/d_2)$, $w' = (u'/d_2)(v'/d_1)$ (见习题3)。这个方法需要计算两次 gcd, 但它并不真比以前的方法慢; 求 gcd 的过程包含一些实际上同它的输入的对数成比例的迭代, 所以为计算 d_1 和 d_2 所需要的迭代总数, 实际上和进行单个 d 的计算期间所需要的迭代总数相同。而且, 计算 d_1 和 d_2 的每个迭代可能更快, 因为要考察的数相对地比较小。如果 u, u', v, v' 都是单精度的量, 则这个方法就有这样一个优点, 即除非两个答案 w 和 w' 全都不可能以单精度形式表示, 否则在计算中不出现双精度数。

除法也可以以类似的方式进行; 见习题4。

加法和减法稍微复杂些。一个显而易见的方法是, 置 $(u/u') + (v/v') = ((uv' \pm u'v)/u'v')$, 然后如同在第一个乘法方法中那样通过计算 $d = \gcd(uv' \pm u'v, u'v')$, 就可以把这个分数约简成最小的项。但是, 如果我们从计算 $d_1 = \gcd(u', v')$ 开始, 则仍有可能避免处理这样大的数。如果 $d_1 = 1$, 则所求的分子和分母是 $w = uv' \pm u'v$ 和 $w' = u'v'$ (按照定理 4.5.2D, 如果分母 u' 和 v' 随机分布, 则 d_1 大约有 61% 的时间为 1, 所以把这一情况单独出来是明智的)。如果 $d_1 > 1$, 则设 $t = u(v'/d_1) \pm v(u'/d_1)$ 并计算 $d_2 = \gcd(t, d_1)$; 最后答案是 $w = t/d_2$, $w' = (u'/d_1)(v'/d_2)$ 。(习题6证明 w 和 w' 的这些值是互素的。)如果使用单精度数, 则这个方法只需要单精度运算, 除非 t 可能是一个双精度数或稍大一些(见习题7); 由于 $\gcd(t, d_1) = \gcd(t \bmod d_1, d_1)$, 故 d_2 的计算不需要双倍精度。

例如, 为计算 $(7/66) + (17/12)$, 我们构造 $d_1 = \gcd(66, 12) = 6$; 然后 $t = 7 \cdot 2 + 17 \cdot 11 = 201$ 和 $d_2 = \gcd(201, 6) = 3$, 所以答案为 $(201/3)/(66/6 \times 12/3) = 67/44$ 。

为帮助校验有理算术的子程序, 建议使用具有已知逆的矩阵(例如, 柯西矩阵, 习题 1.2.3-41)的求逆。

对于分数计算的说明, 在许多情况下, 数增长得十分快; 所以如果打算使每个分数 (u/u') 中的 u 和 u' 是单精度数, 则重要的是要在每一个加法、减法、乘法以及除法子程序中包括溢出测试。对于某些数值问题, 完善的精度很重要, 因此在分子和分母中允许有任意精度的一组分数算术运算子程序是非常有用的。

这一小节的方法也可扩充到除了有理数之外的其它数域中去; 例如, 我们可以对形如 $(u + u'\sqrt{5})/u''$ 的量进行算术运算, 其中 u, u' 和 u'' 都是整数, $\gcd(u, u', u'') = 1$, 且 $u'' > 0$; 也可以对形如 $(u + u'\sqrt[3]{2} + u''\sqrt[3]{4})/u'''$ 的量进行算术运算, 等等。

如果不坚持对分数进行精确计算, 则考虑“开缝定点”(fixed slash)和“开缝浮点”(floating slash)数是有趣的。它们类似于浮点数, 但是以有理分数而不是以面向进制的分数为基础。在一个二进的开缝定点方案中, 对某个给定的 p , 一个可表示的分数的分子和分母都至多由 p 位组成。在一个开缝浮点方案中, 对某个给定的 q , 分子位和分母位的和至多为 q , 而且这个表示的另一个字段用来指出这些 q 位中有多少属于分子。无穷可表示为 $(1/0)$ 。为了对这样的数进行算术运算, 我们定义 $x \oplus y = \text{round}(x + y)$, $x \ominus y = \text{round}(x - y)$ 等等, 其中如果 x 可表示, 则

$\text{round}(x) = x$, 否则它是围绕 x 的两个可表示的数之一。

乍一看, $\text{round}(x)$ 的最好定义似乎是选择最接近于 x 的可表示数, 就好像我们在浮点算术中进行舍入那样。但经验已经证明, 最好使舍入偏倚于“简单”的数, 因为小的分子和分母的数比复杂的分数出现得更为经常。我们要求舍入成 $1/2$ 的数比舍入成 $127/255$ 的数更多。在实践中证明最成功的舍入规则称做“中间舍入”: 如果 (u/u') 和 (v/v') 是相邻的可表示数, 使得每当 $u/u' \leq x \leq v/v'$ 时我们必然有 $\text{round}(x)$ 等于 (u/u') 或 (v/v') , 则中间舍入规则指出:

$$\text{当 } x < \frac{u+v}{u'+v'} \text{ 时, } \text{round}(x) = \frac{u}{u'}, \quad \text{当 } x > \frac{u+v}{u'+v'} \text{ 时, } \text{round}(x) = \frac{v}{v'} \quad (1)$$

如果恰有 $x = (u+v)/(u'+v')$, 则令 $\text{round}(x)$ 是具有最小分母的邻界分数(或者, 如果 $u' = v'$, 则为具有最小分子的邻界分数)。习题 4.5.3-43 说明, 实现有效的中间舍入并不困难。

例如, 假设我们正在进行 $p=8$ 的开缝定点算术运算, 使得可表示的数 (u/u') 有 $-128 < u < 128$ 和 $0 \leq u' < 256$, 且 $u \perp u'$ 。这虽然不够精确, 但足以给我们开缝算术运算的感觉了。相邻于 $0 = (0/1)$ 的数是 $(-1/255)$ 和 $(1/255)$; 因此, 按照中间舍入规则, 当且仅当 $|x| \leq 1/256$ 时我们将有 $\text{round}(x) = 0$ 。假设我们正在进行一项取整个形式为 $\frac{22}{7} = \frac{314}{159} + \frac{1300}{1113}$ 的计算, 如果我们正使用精确的有理算术, 但中间的量必须舍入成可表示的数。在这种情况下, $\frac{314}{159}$ 将舍入到 $(79/40)$, 而 $\frac{1300}{1113}$ 将舍入成 $(7/6)$, 被舍入的项相加成为 $\frac{79}{40} + \frac{7}{6} = \frac{377}{120}$, 它舍入成 $(22/7)$; 所以尽管要求三次舍入, 我们还是得到正确的答案。这个例子不是特别设计的。当问题的答案是一个简单的分数时, 开缝算术趋于使中间的舍入误差消失。

在 P. Henrici 的著作中, 首先讨论了一台计算机内分数的精确表示, 见 JACM 3 (1956), 6~9。David W. Matula 在 S. K. Zaremba 编的 *Application of Number Theory to Numerical Analysis* (New York: Academic Press, 1972), 486~489 中, 提出了开缝定点和浮点运算的思想。Matula 和 Kornerup 在 *Proc. IEEE Symp. Computer Arith.* 4 (1978), 29~47; *Lecture Notes in Comp. Sci.* 72 (1979), 383~397; *Computing, Suppl.* 2 (1980), 85~111; *IEEE Trans. C-32* (1983), 378~388; *IEEE Trans. C-34* (1985), 3~18; *IEEE Trans. C-39* (1990), 1106~1115 中讨论了这一思想的发展。

习 题

1. [I5] 提出一个用于比较两个分数的合理的计算方法, 以测试是否 $(u/u') < (v/v')$ 。
2. [M15] 证明: 如果 $d = \gcd(u, v)$, 则 u/d 和 v/d 互素。
3. [M20] 证明: $u \perp u'$ 和 $v \perp v'$ 意味着

$$\gcd(uv, u'v') = \gcd(u, v')\gcd(u', v)。$$

4. [11] 试设计对于分数的一个除法算法,它类似于正文中的第二个乘法算法(注意,必须考虑 v 的符号)

5. [10] 试用正文中建议的方法计算 $(17/120) + (-27/70)$ 。

►6. [M23] 证明 $u \perp u'$ 和 $v \perp v'$ 意味着 $\gcd(uv' + u'v, u'v') = d_1 d_2$, 其中 $d_1 = \gcd(u', v')$, $d_2 = \gcd(d_1, u(v'/d_1) + v(u'/d_1))$ 。(因此,如果 $d_1 = 1$, 则 $uv' + u'v \perp u'v'$ 。)

7. [M22] 在正文中建议的加法-减法方法下,如果输入的分子和分母的绝对值都小于 N , 量 t 的绝对值可变成多大?

►8. [22] 试讨论利用 $(1/0)$ 和 $(-1/0)$ 作为 ∞ 和 $-\infty$ 的表示和(或)作为溢出的表示。

9. [M23] 如果 $1 \leq u', v' < 2^n$, 证明 $\lfloor 2^{2n} u/u' \rfloor - \lfloor 2^{2n} v/v' \rfloor$ 意味着 $u/u' - v/v'$ 。

10. [41] 推广习题 4.3.1-34 中提出的子程序,使它们能处理“任意”有理数。

11. [M23] 考虑形如 $(u + u'\sqrt{5})/u''$ 的分数,其中 u, u', u'' 是整数, $\gcd(u, u', u'') = 1$, 且 $u'' > 0$ 。说明如何进行这样两个分数的除法并且得到有相同形式的商。

12. [M16] 给定分子的长度加分母的长度的界 q , 问最大的有限开缝浮点数是多少? 什么数舍入成 $(0/1)$?

13. [20] (Matula 和 Kornerup) 试讨论在一个 32 位字中的开缝浮点数的表示。

14. [M23] 说明怎样计算使得 $M_1 \leq u \leq M_2$ 和 $N_1 \leq u' \leq N_2$ 且 $u \perp u'$ 的整数偶 (u, u') 的精确个数。(这可以用来确定在开缝算术中有多少数是可表示的。按照定理 4.5.2D, 这个数将近似为 $(6/\pi^2)(M_2 - M_1)(N_2 - N_1)$ 。)

15. [42] 修改你的计算机上的编译程序之一,使得它以开缝浮点计算代替所有浮点计算。通过运行现成程序来对开缝算术的使用进行实验,这些程序是由实际上已牢记浮点算术的程序员写成的。(当调用像开方或对数这样的特殊子程序时,你的系统在调用子程序之前,应当自动地把开缝数转换成为浮点形式,以后再回到开缝形式。应当有新的选择以分数格式打印开缝的数;然而,如果对于用户的源程序不做修改的话,那也应该像通常一样以十进记法打印开缝的数。当开缝浮点数被替换时,结果是更好还是更糟?

16. [40] 试对开缝数的区间算术做试验。

4.5.2 最大公因子

如果 u 和 v 是不全为 0 的整数,则我们就说它们的最大公因子 $\gcd(u, v)$ 是同时整除 u 和 v 的最大整数。这个定义是有意义的,因为如果 $u \neq 0$, 则没有大于 $|u|$ 的整数可以整除 u , 但整数 1 整除 u 和 v 两者;因此必定有同时整除两者的最大整数。当 u 和 v 都为零时,每个整数都整除,所以上述定义不能应用;这时置

$$\gcd(0, 0) = 0 \quad (1)$$

是方便的。

刚才给出的定义明显地意味着

$$\gcd(u, v) = \gcd(v, u) \quad (2)$$

$$\gcd(u, v) = \gcd(-u, v) \quad (3)$$

$$\gcd(u, 0) = |u| \quad (4)$$

在前一节,我们把以“最低项”表达一个有理数的问题归结为求它的分子和分母的最大公因子的问题。在 3.2.1.2, 3.3.3, 4.3.2, 4.3.3 这些小节里,已经作为例子

提出了最大公因子的其它应用。所以最大公因子是重要的而且值得认真研究。

两个整数 u 和 v 的最小公倍数写成 $\text{lcm}(u, v)$, 也是一个很重要的相关概念。它被定义为 u 和 v 的倍数中最小的正整数, 而且 $\text{lcm}(u, 0) = \text{lcm}(0, v) = 0$ 。向孩子们讲授如何做分数加法 $u/u' + v/v'$ 的经典方法是训练他们求最小公分母, 这就是 $\text{lcm}(u', v')$ 。

按照“算术基本定理”(已在习题 1.2.4-21 中证明了), 每个正整数 u 均可表示成形式

$$u = 2^{u_2} 3^{u_3} 5^{u_5} 7^{u_7} 11^{u_{11}} \cdots = \prod_{p \text{ 为素数}} p^{u_p} \quad (5)$$

其中指数 u_2, u_3, \dots 是惟一确定的非负整数, 而且其中除有限个指数外, 其余皆为零。从一个正整数的这个“规范因子分解式”出发, 立即得到计算 u 和 v 的最大公因子的一个方法: 由(2), (3)和(4), 我们可以假定 u 和 v 都是正整数, 而且如果两者已经规范地分解成素数因子, 则我们有

$$\gcd(u, v) = \prod_{p \text{ 为素数}} p^{\min(u_p, v_p)} \quad (6)$$

$$\text{lcm}(u, v) = \prod_{p \text{ 为素数}} p^{\max(u_p, v_p)} \quad (7)$$

于是, 例如, $u = 7000 = 2^3 \cdot 5^3 \cdot 7$ 和 $v = 4400 = 2^4 \cdot 5^2 \cdot 11$ 的最大公因子是 $2^{\min(3, 4)} \times 5^{\min(3, 2)} \times 7^{\min(1, 0)} \times 11^{\min(0, 1)} = 2^3 \times 5^2 = 200$ 。两数的最小公倍数是 $2^4 \times 5^3 \times 7 \times 11 = 154000$ 。

从公式(6)和(7), 容易证明有关 \gcd 和 lcm 的一些基本的恒等式:

$$\gcd(u, v)w = \gcd(uw, vw), \quad \text{如果 } w \geq 0 \quad (8)$$

$$\text{lcm}(u, v)w = \text{lcm}(uw, vw), \quad \text{如果 } w \geq 0 \quad (9)$$

$$u \cdot v = \gcd(u, v) \cdot \text{lcm}(u, v), \quad \text{如果 } u, v \geq 0 \quad (10)$$

$$\gcd(\text{lcm}(u, v), \text{lcm}(u, w)) = \text{lcm}(u, \gcd(v, w)) \quad (11)$$

$$\text{lcm}(\gcd(u, v), \gcd(u, w)) = \gcd(u, \text{lcm}(v, w)) \quad (12)$$

后两个公式是类似于熟知的恒等式 $uv + uw = u(v + w)$ 的分配律。等式(10)把 $\gcd(u, v)$ 的计算归结成为 $\text{lcm}(u, v)$ 的计算, 而且反过来又把 $\text{lcm}(u, v)$ 的计算归结为 $\gcd(u, v)$ 的计算。

欧几里得算法 尽管等式(6)从理论上说是有用的, 但一般说来它对于实际计算最大公因子并没有什么帮助, 因为它要求我们首先确定 u 和 v 的规范因子分解。还没有已知的方法非常快地来求一个整数的诸素因子(见 4.5.4 小节)。幸而有一个不需要分解两个整数就能计算它们的最大公因子的有效方法。事实上, 这样一个方法在 2250 多年以前就发现了; 这就是“欧几里得”算法, 在 1.1 节和 1.2.1 小节我们就已经考察过了。

欧几里得算法见于欧几里得的 *Elements* 一书(大约公元前 300 年)的卷 7, 命题 1 和 2, 但这大概不是他自己的发明。学者们相信这个方法大约在那之前 200 年就已经知道了, 至少以它的减法形式, 而且几乎肯定为 Eudoxus(约公元前 375 年)所

知;见 K. von Fritz, *Ann. Math.* (2) **46** (1945), 242 ~ 246. Aristotle 在他的 *Topics*, 158b, 29 ~ 35 上也提到了它。然而关于这样早期历史的过硬证据幸存下来的极少 [请见 W. R. Knorr, *The Evolution of the Euclidean Elements* (Dordrecht: 1975)]。

我们可以把欧几里得方法称做所有算法的祖先,因为它是今日幸存的最老的不寻常的算法,主要能与这一荣耀相抗衡的也许是古老的埃及的乘法方法,它是以加倍和加法为基础的,而且如同 4.6.3 节所说明的,它形成有效计算第 n 次乘方的基础。但是埃及的原稿仅给出不是完全系统的一些例子,而且这些例子肯定未加系统阐述;因此埃及的算法肯定配不上算法的名字。我们也知道用于诸如解两个变量的特殊二次方程组的若干古代巴比伦人的方法,其中包含了一些真正的算法,而不仅仅是对于某些输入参数的方程特殊解;尽管巴比伦人总是联系对具体的输入数据进行工作的例子来介绍每个方法,但他们总是有规律地伴随着正文来说明一般过程。[见 D. E. Knuth, *CACM* **15** (1972), 671 ~ 677; **19** (1976), 108。]这些巴比伦算法中有许多比欧几里得算法早大约 1500 年,而且它们是已知最早的关于数学的书面过程的例证。但它们都没有欧几里得算法的气质,因为它们都不含迭代,而且还因为它们已为现代代数方法所代替。

由于欧几里得算法从实用上和从历史上说都是重要的,现在就让我们考虑欧几里得本人是怎样处理它的。把他的话语译成现代术语,欧几里得所说的就是:

命题 给定两个正整数,求它们的最大公因子。

设 A 和 C 是两个给定的正整数;求它们的最大公因子。如果 C 整除 A ,则 C 就是 C 和 A 的一个公因子,因为它也整除本身。而且事实上显然它是最大的,因为再无比 C 大的数能整除 C 了。

但如果 C 不整除 A ,则不断地从 A, C 两数中之大者减去小者,直到得出整除前一数的某个数。这种情况最终一定要发生,因为如果得到的是 1,则它就整除前一个数。

现在设 E 是 A 除以 C 的正余数;设 F 是 C 除以 E 的正余数,而且设 F 是 E 的一个因子。由于 F 整除 E 和 E 整除 $C - F$,故 F 也整除 $C - F$;但它也整除本身,所以它整除 C 。而且 C 整除 $A - E$;因此 F 也整除 $A - E$,但它也整除 E ;因此它整除 A 。于是它是 A 和 C 的一个公因子。

现在论证它也是最大的。如果 F 不是 A 和 C 最大的公因子,则将有某个更大的数整除它们。设这样一个数是 G 。

现在由于 G 整除 C 而 C 整除 $A - E$,故 G 整除 $A - E$,但也整除整个 A ,所以它整除余数 E ,但 E 整除 $C - F$;因此 G 也整除 $C - F$,但 G 也整除整个 C ,所以它整除余数 F ;即,一个较大的数整除一个较小的数,这是不可能的。

因此,没有大于 F 的数能整除 A 和 C ,所以 F 是它们的最大公因子。

推论 这一论证使得下列结论成为显然:整除两个数的任何数必整除它们的最大公因子。证毕。

这里在一个重要方面简化了欧几里得的叙述:希腊数学家不认为 1 是另外的正整数的“因子”。两个正整数或者都为 1,或者互素,或者有一个最大的公因子。事实上,1 甚至不被认为是一个“数”,而且 0 当然是不存在的。这些相当笨拙的约定使得欧几里得有必要重复他的许多讨论,而且他已经给出了两个分开的命题,其中每一个实际上都同现在给出的这个相类似。

在欧几里得的讨论中,他首先提出重复地从两个现有数的大者减去小者,直到我们得到这样两个数——其中一个是另一个的倍数为止。但在证明中,他实际上依赖于一个数除以另一个数的余数;由于他没有0的简单概念,故他不能说一个数整除另一个数时的余数是什么。所以可以合理地认为,他想像每个除法(不是个别的减法)都是算法的一个单一的步骤,因此对他的算法的“真正的”意译也许可以说是下面的算法。

算法 E(原始欧几里得算法) 给定两个大于1的整数 A 和 C ,本算法求它们的最大公因子。

E1. [A 可被 C 整除?] 如果 C 整除 A ,则这算法以 C 作为答案终止。

E2. [以余数代替 A] 如果 $A \bmod C$ 等于1,则给定的数互素,所以算法终止。

否则,以 $(C, A \bmod C)$ 代替 (A, C) 并返回步骤 E1。 ─

上面摘录的欧几里得给出的证明,是特别有趣的,因为它实质上全然不是证明!他只是实施步骤 E1 一次或三次验证了这个算法的结果。诚然,他必然认识到步骤 E1 可进行三次以上,尽管他没有提出这样的可能性。由于没有用数学归纳法给出一个证明的思想,他仅仅对于有限种情况给出了一个证明(事实上,对于要求对一般的 n 建立的过程,他通常仅仅证明 $n=3$ 的情况)。尽管由于欧几里得在逻辑推导的技巧上所作出的巨大推进而应得地闻名于世,但通过归纳法给出正确证明的技术直到许多世纪之后还迟迟未被发现;而且证明算法正确性的严格思想,实际上直到现在才真正搞清楚(关于欧几里得算法的一个完备的证明,以及对于算法的一般证明过程的讨论,见 1.2.1 小节)。

值得注意的是,用于求最大公因子的这个算法被欧几里得选来作为他发展数论的第一步。当今的许多现代教科书仍然沿用相同的讲述顺序。欧几里得也给出了一个方法(命题 34)来求两个整数 u 和 v 的最小公倍数,即以 $\gcd(u, v)$ 除 u 并且以 v 来乘其结果,这等价于等式(10)。

如果避免欧几里得对于0和1的偏见,则我们就可以用下列方式重新叙述算法 E。

算法 A(现代欧几里得算法) 给定非负整数 u 和 v ,本算法求出它们的最大公因子(注意:由于等式(2)和(3),任意整数 u 和 v 的最大公因子可以通过把这个算法应用于 $|u|$ 和 $|v|$ 得到)。

A1. [$v=0$?] 如果 $v=0$,则本算法以 u 作为答案而终止。

A2. [取 $u \bmod v$] 置 $r \leftarrow u \bmod v, u \leftarrow v, v \leftarrow r$,并返回 A1。(这一步的操作减小 v 的值,但保持 $\gcd(u, v)$ 不变。) ─

例如,我们可以计算 $(40902, 24140)$ 如下:

$$\begin{aligned}\gcd(40902, 24140) &= \gcd(24140, 16762) = \gcd(16762, 7378) = \\ &= \gcd(7378, 2006) = \gcd(2006, 1360) = \\ &= \gcd(1360, 646) = \gcd(646, 68) = \gcd(68, 34) =\end{aligned}$$

$$\gcd(34, 0) = 34$$

如果 q 是任意整数, 算法 A 的正确性的证明容易由等式(4)和

$$\gcd(u, v) = \gcd(v, u - qv) \quad (13)$$

得出。等式(13)成立, 因为 u 和 v 的任何公因子都是 u 和 $u - qv$ 的一个因子, 反过来, v 和 $u - qv$ 的任何公因子也必整除 u 和 v 。

下列的 MIX 程序说明了算法 A 可容易地在一台计算机上实现。

程序 A(欧几里得算法) 假设 u 和 v 是单精度的非负整数, 分别存放在单元 U 和 V 中; 这一算法把 $\gcd(u, v)$ 置于 rA 中。

	LDX	U	1	$rX \leftarrow u$
	JMP	2F	1	
1H	STX	V	T	$v \leftarrow rX$
	SRAX	5	T	$rAX \leftarrow rA$
	DIV	V	T	$rX \leftarrow rAX \bmod v$
2H	LDA	V	$1 + T$	$rA \leftarrow v$
	JXNZ	1B	$1 - T$	如果 $rX = 0$ 则完成

这一程序的运行时间是 $19T + 6$ 个周期, 其中 T 是实施除法的次数。4.5.3 小节的讨论说明, 当 u 和 v 独立地一致分布于 $1 \leq u, v \leq N$ 的范围内时, 我们可以取 $T = 0.842766 \ln N + 0.06$ 作为一个近似的平均值。

一个二进制方法 尽管欧几里得手工算法已经沿用了这么多世纪, 但它却不是求最大公因子的最好方法, 这说来颇为令人惊讶。一个十分不同的主要适合于二进算术的 \gcd 算法是由 Josef Stein 于 1961 年发现的[见 *J. Comp. Phys.* 1 (1967), 397 ~ 405]。这个新算法不要求除法指令。它仅仅依赖于下列运算: (i) 减法; (ii) 奇偶校验; (iii) 偶数折半(它对应于在二进制表示中的右移)。

二进制 \gcd 算法基于正整数 u 和 v 的四项简单事实:

- 如果 u 和 v 都是偶数, 则 $\gcd(u, v) = 2\gcd(u/2, v/2)$ [见等式(8)]。
- 如果 u 是偶数而 v 是奇数, 则 $\gcd(u, v) = \gcd(u/2, v)$ [见等式(6)]。
- 如同在欧几里得算法中一样, $\gcd(u, v) = \gcd(u - v, v)$ [见等式(13), (2)]。
- 如果 u 和 v 都是奇数, 则 $u - v$ 是偶数, 而且 $|u - v| < \max(u, v)$ 。

算法 B(二进制的 \gcd 算法) 给定正整数 u 和 v 。本算法求它们的最大公因子。

- B1.** [求 2 的乘方] 置 $k \leftarrow 0$, 而后重复地置 $k \leftarrow k + 1, u \leftarrow u/2, v \leftarrow v/2$ 共 0 次或多次, 直到 u 和 v 不都是偶数为止。
- B2.** [初始化] (现在 u 和 v 的值已经被除以 2^k , 而且它们现有值中至少有一个是奇数。)如果 u 是奇数, 则置 $t \leftarrow v$ 并转到 B4。否则 $t \leftarrow u$ 。
- B3.** [t 折半] (这时 t 为偶数, 而且非零。)置 $t \leftarrow t/2$ 。
- B4.** [t 是偶数吗?] 如果 t 为偶数, 则转回 B3。

- B5.** [重置 $\max(u, v)$] 如果 $t > 0$, 则置 $u \leftarrow t$; 否则置 $v \leftarrow -t$ (u 和 v 之较大者已为 $|t|$ 所代替, 除了或许在头一次实施这一步骤期间之外。)
- B6.** [做减法] 置 $t \leftarrow u - v$, 如果 $t \neq 0$, 则转回 B3。否则本算法以 $u \cdot 2^k$ 作为输出而终止。 ■

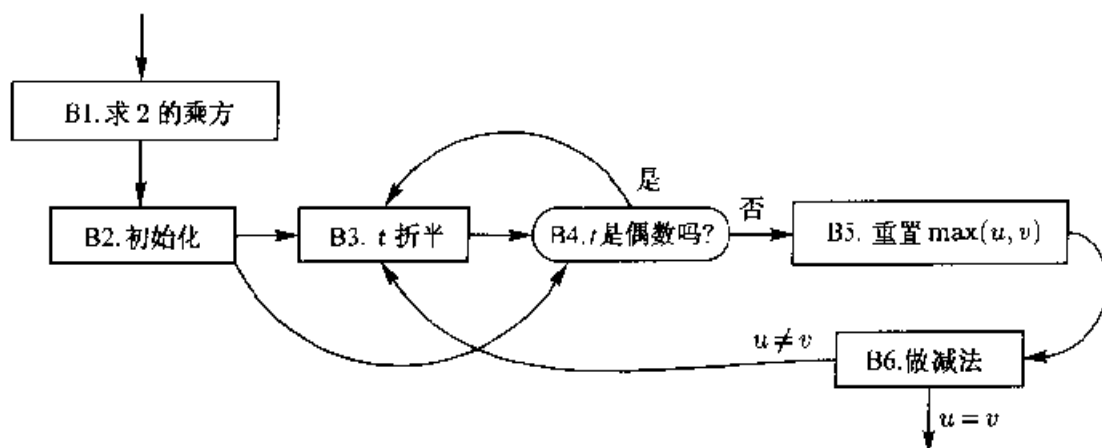


图 9 求最大公因子的二进算法

作为算法 B 的一个例子, 我们来考虑 $u = 40902, v = 24140$, 我们已经使用同样的数试验过欧几里得算法。步骤 B1 置 $k \leftarrow 1, u \leftarrow 20451, v \leftarrow 12070$, 然后 t 被置为 -12070 , 继又被 -6035 所代替; 然后 v 被 6035 所代替, 这一计算的过程如下:

u	v	t
20451	6035	$+14416, +7208, +3604, +1802, +901;$
901	6035	$-5134, -2567;$
901	2567	$-1666, -833;$
901	833	$+68, +34, +17;$
17	833	$-816, -408, -204, -102, -51;$
17	51	$-34, -17;$
17	17	0

答案是 $17 \times 2^1 = 34$ 。这里有必要比算法 A 做稍多一点的迭代, 但由于不使用除法, 故每次迭代都比较简单。

算法 B 的一个 MIX 程序仅仅比算法 A 多出一段代码。为使这样一个程序成为算法 B 的相当典型的一个二进计算机表示, 我们假定 MIX 被扩充成包括如下一些运算符:

- SLB (AX 二进左移) $C = 6; F = 6$ 。寄存器 A 和 X 的内容被“左移”M 个二进位; 即 $|rAX| \leftarrow |2^M rAX| \bmod B^{10}$, 其中 B 是字节长 (如同所有的 MIX 移位指令一样, rA 和 rX 的符号均不受影响)。
- SRB (AX 二进右移) $C = 6, F = 7$ 。A 和 X 的内容被“右移”M 个二进位; 即 $|rAX| \leftarrow \lfloor |rAX| / 2^M \rfloor$ 。

·JAE, JAO (A 偶转移, A 奇转移) C=40; F 分别等于 6, 7。如果 rA 分别为偶或奇, 则出现 JMP。

·JXE, JXO (X 偶转移, X 奇转移) C=47; F 分别等于 6, 7。类似于 JAE, JAO。

程序 B(二进制 gcd 算法) 假设 u 和 v 是单精度的正整数, 分别存放于单元 U 和 V 中; 这一程序使用算法 B 把 $\gcd(u, v)$ 放进 rA 中。寄存器分配: $rA \equiv t, r11 \equiv k$ 。

01	ABS	EQU	1:5		
02	B1	ENT1	0	1	<u>B1. 求 2 的乘方</u>
03		LDX	U	1	$rX \leftarrow u$
04		LDAN	V	1	$rA \leftarrow -v$
05		JMP	1F	1	
06	2H	SRB	1	A	rA, rX 折半
07		INCL	1	A	$k \leftarrow k + 1$
08		STX	U	A	$u \leftarrow u/2$
09		STA	V(ABS)	A	$v \leftarrow v/2$
10	1H	JXO	B4	$1 + A$	若 u 是奇数则置 $t \leftarrow -v$, 转 B4
11	B2	JAE	2B	$B + A$	<u>B2. 初始化</u>
12		LDA	U	B	$t \leftarrow u$
13	B3	SRB	1	D	<u>B3. t 折半</u>
14	B4	JAE	B3	$1 - B + D$	<u>B4. t 为偶?</u>
15	B5	JAN	1F	C	<u>B5. 重置 $\max(u, v)$</u>
16		STA	U	E	若 $t > 0$, 则置 $u \leftarrow t$
17		SUB	V	E	$t \leftarrow u - v$
18		JMP	2F	E	
19	1H	STA	V(ABS)	$C - E$	若 $t < 0$, 则置 $v \leftarrow -t$
20	B6	ADD	U	$C - E$	<u>B6. 做减法</u>
21	2H	JANZ	B3	C	若 $t \neq 0$, 则转 B3
22		LDA	U	1	$rA \leftarrow u$
23		ENTX	0	1	$rX \leftarrow 0$
24		SLB	0, 1	1	$rA \leftarrow 2^k \cdot rA$

这个程序的运行时间为

$$9A + 2B + 6C + 3D + E + 13$$

个单位, 其中 $A = k$, 如果在步骤 B2 中 $t \leftarrow u$, 则 $B = 1$ (否则 $B = 0$), C 是做减法的次数, D 是在步骤 B3 中折半的次数, 而 E 是在步骤 B5 中 $t > 0$ 的次数。本节后边讨论的计算意味着, 假定随机输入的 u 和 v 是在 $1 \leq u, v < 2^N$ 的范围内, 我们可以取 $A = 1/3, B = 1/3, C = 0.71N - 0.5, D = 1.41N - 2.7, E = 0.35N - 0.4$ 作为对于这些量的平均值。因此总的运行时间大约为 $8.8N + 5.2$ 个周期, 而在同样假定下

程序 A 大约需要 $11.1N + 7.1$ 个周期。当 $A = 0, B = 1, C = N, D = 2N - 2, E = N - 1$ 时, 在这个范围内的 u 和 v 出现最坏的运行时间; 总计为 $13N + 8$ 个周期(而对于程序 A 对应的值为 $26.8N + 19$)。

因而, 由于运算的简化, 程序 B 中较快的迭代弥补了所需要的较大的迭代次数。我们已经发现, 在 MIX 计算机上, 二进算法要比欧几里得算法快大约 20%。当然, 在其它计算机上, 情况可能有所不同, 无论在何种情况下, 这两个程序都是十分有效的。但是看起来, 即使像欧几里得算法这样古老的方法, 也经不起进一步的考验。

二进最大公因子算法本身可能有一个著名起源, 因为它在古老中国可能已广为人知。称做《九章算术》(约公元 1 世纪) 的一本经典课本的第一章第六节*, 给出把一个分数约简成最低项的下列方法:

可半者半之。

不可半者, 副置分母、子之数, 以少减多,

更相减损, 求其等也。

以等数约之。

如果重复指令意味着回到折半步骤而不是重复减法步骤——这点不清楚——这个方法实质上就是算法 B。[请见 Y. Mikami, *The Development of Mathematics in China and Japan* (Leipzig: 1913), 11; K. Vogel, *Neun Bücher arithmetischer Technik* (Braunschweig: Vieweg, 1968), 8.]

V.C. Harris [*Fibonacci Quarterly* 8 (1970), 102 ~ 103; 也见 V. A. Lebesgue, *J. Math. Pures Appl.* 12 (1847), 497 ~ 520] 提出了欧几里得算法和二进算法之间的有趣交叉。如果 u 和 v 是奇数, 并有 $u \geq v > 0$, 则我们总可以令

$$u = qv \pm r$$

其中 $0 \leq r < v$ 且 r 为偶数; 如果 $r \neq 0$, 则我们置 $r \leftarrow r/2$ 直到 r 为奇数为止, 然后置 $u \leftarrow v, v \leftarrow r$ 且重复这一个过程。在随后的迭代中, $q \geq 3$ 。

推广 我们可以把计算 $\gcd(u, v)$ 的一些方法推广, 以便解决稍微困难些的问题。例如, 假定我们要计算 n 个整数 u_1, u_2, \dots, u_n 的最大公因子。

假定诸 u 是非负的, 计算 $\gcd(u_1, u_2, \dots, u_n)$ 的一个办法, 就是以下列方式推广欧几里得算法: 如果所有的 u_j 均为 0, 则最大公因子取为 0; 否则如果仅有一个 u_j 非 0, 则它就是最大公因子; 否则对于所有 $k \neq j$, 以 $u_k \bmod u_j$ 代替 u_k , 其中 u_j 是非 0 的诸 u 当中的最小者。

在上一段里描述的算法, 是欧几里得算法的一个自然的推广, 而且可以以类似方式论证其正确性。但是, 还有一个更简单的方法可以利用, 它以容易验证的恒等式

$$\gcd(u_1, u_2, \dots, u_n) = \gcd(u_1, \gcd(u_2, \dots, u_n)) \quad (14)$$

为基础。为了计算 $\gcd(u_1, u_2, \dots, u_n)$, 我们可以按如下步骤进行:

* 即《九章算术》方田章约分术。——本书责任编辑

算法 C(n 个整数的最大公因子) 给定整数 u_1, u_2, \dots, u_n , 其中 $n \geq 1$, 本算法利用 $n=2$ 时的一个算法作为子程序, 计算它们的最大公因子。

C1. 置 $d \leftarrow u_n, j \leftarrow n-1$ 。

C2. 如果 $d \neq 1$ 且 $k > 0$, 则置 $d \leftarrow \gcd(u_k, d)$ 和 $k \leftarrow k-1$ 并重复这一步骤。否则 $d = \gcd(u_1, \dots, u_n)$ 。 **I**

这个方法把 $\gcd(u_1, \dots, u_n)$ 的计算归结为重复地一次计算两个数的最大公因子。它利用了 $\gcd(u_1, \dots, u_j, 1) = 1$ 这个事实; 这样做很有好处, 因为, 如果 u_{n-1} 和 u_n 是随机地选择的, 我们将有 60% 以上的机会 $\gcd(u_{n-1}, u_n) = 1$ 。在大多数情况下, 在前头几步的计算中, d 的值将急剧地减小, 因而使得剩下的计算十分快。这里欧几里得算法相对于算法 B 有一个优点, 因为它的运行时间主要地由 $\min(u, v)$ 所支配, 算法 B 的运行时间则主要是由 $\max(u, v)$ 所支配; 合理的做法是, 如果 u 比 v 大很多, 则宜于先实施欧几里得算法的一次迭代, 以 $u \bmod v$ 代替 u , 然后再以算法 B 继续之。

断定对于随机输入 $\gcd(u_{n-1}, u_n)$ 将有 60% 以上时间等于 1, 是数论中下列著名结果的一个推论。

定理 D (G. Lejeune Dirichlet, *Abhandlungen Königlich Preuß. Akad. Wiss.* (1849), 69~83) 如果 u 和 v 是随机地选择的整数, 则 $\gcd(u, v) = 1$ 的概率为 $6/\pi^2 \approx .60793$ 。

习题 10 给出了这个定理的精确公式及严格证明, 该公式仔细定义了“随机地选择”的含义。眼下我们给出一个启发性的论证, 它说明为什么这个定理是讲得通的。

如果我们不加证明地假定 $u \perp v$ 的适当定义的概率 p 是存在的, 则我们就可以确定对于任意正整数 d , $\gcd(u, v) = d$ 的概率, 因为当且仅当 u 是 d 的倍数, 而且 $(u/d \perp v/d)$ 时, $\gcd(u, v) = d$ 。于是 $\gcd(u, v) = d$ 的概率等于 $1/d$ 乘 $1/d$ 乘 p , 即 p/d^2 。现在我们对于所有可能的 d 值把这些概率加起来, 就得到

$$1 = \sum_{d \geq 1} p/d^2 = p \left(1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \dots \right)$$

由于由式 1.2.7-(7), 和数 $1 + \frac{1}{4} + \frac{1}{9} + \dots = H_{\infty}^{(2)}$ 等于 $\pi^2/6$, 故为使这个等式成立, 我们需要有 $p = 6/\pi^2$ 。 **I**

欧几里得算法可以另一种重要的方式加以推广: 我们可以在计算 $\gcd(u, v)$ 的同时, 计算整数 u' 和 v' , 使得

$$uu' + vv' = \gcd(u, v) \quad (15)$$

欧几里得算法的这一推广, 用向量记法描述比较方便:

算法 X(扩充的欧几里得算法) 给定非负整数 u 和 v , 本算法确定一个向量 (u_1, u_2, u_3) 使得 $uu_1 + vu_2 = u_3 = \gcd(u, v)$ 。这个计算利用辅助向量 $(v_1, v_2, v_3), (t_1, t_2, t_3)$; 所有的向量都以这样一种方式操作, 即在整个计算过程中, 如下关

系式

$$ut_1 + vt_2 = t_3, \quad uu_1 + vv_2 = u_3, \quad uv_1 + vu_2 = v_3 \quad (16)$$

成立。

X1. [初始化] 置 $(u_1, u_2, u_3) \leftarrow (1, 0, u), (v_1, v_2, v_3) \leftarrow (0, 1, v)$ 。

X2. [$v_3 = 0$ 吗?] 如果 $v_3 = 0$, 则此算法终止。

X3. [除, 减] 置 $q \leftarrow \lfloor u_3/v_3 \rfloor$, 然后置

$$\begin{aligned} (t_1, t_2, t_3) &\leftarrow (u_1, u_2, u_3) - (v_1, v_2, v_3)q, \\ (u_1, u_2, u_3) &\leftarrow (v_1, v_2, v_3), \quad (v_1, v_2, v_3) \leftarrow (t_1, t_2, t_3) \end{aligned}$$

返回步骤 X2。 |

例如, 设 $u = 40902, v = 24140$ 。在步骤 X2 我们有

q	u_1	u_2	u_3	v_1	v_2	v_3
—	1	0	40902	0	1	24140
1	0	1	24140	1	-1	16762
1	1	-1	16762	-1	2	7378
2	-1	2	7378	3	-5	2006
3	3	-5	2006	-10	17	1360
1	-10	17	1360	13	-22	646
2	13	-22	646	-36	61	68
9	-36	61	68	337	-517	34
2	337	-571	34	-710	1203	0

因此解答是 $337 \times 40902 - 571 \times 24140 = 34 = \gcd(40902, 24140)$ 。

算法 X 可追溯到由印度北方 Aryabhata 的 *Aryabhatiya* (公元 499 年)。他的描述是相当神秘的, 但以后的诠释者如 6 世纪的 Bhāscara I 阐明了这个规则。它被称做 *kuttaka* (“粉碎器”)。[请见 B. Datta 和 A. N. Singh, *History of Hindu Mathematics* 2 (Lahore: Motilal Banarsi Das, 1938), 89 ~ 116。] 它的正确性由 (16) 和下述事实得出: 就 u_3 和 v_3 的操作而言, 这个算法同算法 A 是相同的; 1.2.1 小节讨论了算法 X 的详细证明。Gordon H. Bradley 已经发现, 我们可以去掉 u_2, v_2 和 t_2 来避免算法 X 中的大量计算; 然后再利用关系式 $uu_1 + vv_2 = u_3$ 来确定 u_2 。

习题 15 表明, $|u_1|, |u_2|, |v_1|, |v_2|$ 的值被输入 u 和 v 的大小所界限。算法 B 利用二进制记法的性质计算最大公因子, 它可以以类似的方式被推广; 见习题 39。关于算法 X 的某些有教益的推广, 见 4.6.1 小节中的习题 18 和习题 19。

奠定欧几里得算法之基础的一些思想, 也可应用于求带有整系数的任意线性方程组的整数通解。例如, 假设我们要求满足两个方程

$$10w + 3x + 3y + 8z = 1 \quad (17)$$

$$6w - 7x - 5z = 2 \quad (18)$$

的所有整数 w, x, y, z , 我们可以引进一个新变量

$$\begin{aligned} \lfloor 10/3 \rfloor w + \lfloor 3/3 \rfloor x + \lfloor 3/3 \rfloor y + \lfloor 8/3 \rfloor z = \\ 3w + x + y + 2z = t_1 \end{aligned}$$

用它来消去 y ; 则等式(17)变为

$$(10 \bmod 3)w + (3 \bmod 3)x + 3t_1 + (8 \bmod 3)z = w + 3t_1 + 2z = 1 \quad (19)$$

等式(18)保持不变, 新的方程可以用来消去 w , 于是(18)变成

$$6(1 - 3t_1 - 2z) - 7x - 5z = 2$$

即

$$7x + 18t_1 + 17z = 4 \quad (20)$$

现在我们和前边一样引进一个新变量

$$x + 2t_1 + 2z = t_2$$

并从(20)消去 x :

$$7t_2 + 4t_1 + 3z = 4 \quad (21)$$

为了消去变量 z , 可以用同样的方式引进另外一个新变量, 它有最小的系数:

$$2t_2 + t_1 + z = t_3$$

从(21)消去 z 得到

$$t_2 + t_1 + 3t_3 = 4 \quad (22)$$

最终这个方程可用来消去 t_2 。现在只剩下两个独立的变量 t_1 和 t_3 ; 我们把它代回原来的变量就得到通解

$$\begin{aligned} w &= 17 - 5t_1 - 14t_3 \\ x &= 20 - 5t_1 - 17t_3 \\ y &= -55 + 19t_1 + 45t_3 \\ z &= -8 + t_1 + 7t_3 \end{aligned} \quad (23)$$

换言之, 原来方程(17)、(18)的所有整数解 (w, x, y, z) 均可从式(23)通过 t_1 和 t_3 独立地取遍所有整数而得到。

刚才说明的这个一般方法, 是以下列过程为基础的: 求出方程组中绝对值最小的一个非零系数 c 。假设这个系数出现于具有形式

$$cx_0 + c_1x_1 + \cdots + c_kx_k = d \quad (24)$$

的一个方程中; 为简便计, 假设 $c > 0$ 。如果 $c = 1$, 则利用这个方程来从方程组的其它方程中消去变量 x_0 ; 然后对剩下的方程重复此过程(如果再无剩下的方程, 则计算就此停止, 我们实际上已经得到了用未被消去的变量表示的一般解)。如果 $c > 1$, 则若 $c_1 \bmod c = \cdots = c_k \bmod c = 0$, 检查是否有 $d \bmod c = 0$, 否则就没有整数解; 然后以 c 来除(24)的两边并像 $c = 1$ 的情况那样消去 x_0 。最后, 如果 $c > 1$, 而且 $c_1 \bmod c, \cdots, c_k \bmod c$ 不全为零, 则引进一个新变量

$$\lfloor c/c \rfloor x_0 + \lfloor c_1/c \rfloor x_1 + \cdots + \lfloor c_k/c \rfloor x_k = t \quad (25)$$

从其它方程消去变量 x_0 , 采用 t , 并以

$$ct + (c_1 \bmod c)x_1 + \cdots + (c_k \bmod c)x_k = d \quad (26)$$

代替原来的方程(24)。(参见上例中的(19)和(21)。

这个过程必然会终止,因为每一步骤或减少方程个数,或者减少方程组中最小非零系数的大小。对于确定的整数 u 和 v ,当这个过程被应用于方程 $ux + vy = 1$ 时,它实质上跑遍算法 X 的步骤。

当只允许变量取整数值时,刚才说明的变量变换的方法是求解线性方程组的一个简单和直截了当方法,但它不是对于这个问题可利用的最好方法。有可能有实质性的改进,但它超出本书的范围[请见 Henri Cohen, *A Course in Computational Algebraic Number Theory* (New York: Springer, 1993), 第 2 章]。

对于高斯整数 $u + iv$ 和在某些其它二次域中也可以使用欧几里得算法的变形。例如,请见 A. Hurwitz, *Acta Math.* **11** (1987), 187~200; E. L. Kaltofen 和 H. Rolletschek, *Math. Comp.* **53** (1989), 697~720; A. Knopfmacher 和 J. Knopfmacher, *BIT* **31** (1991), 286~292。

高精度计算 如果 u 和 v 是非常大的整数,并要求有高精度的表示,则二进制方法(算法 B)乃是计算它们的最大公因子的简单而又相当有效的方法,因为它仅含减法和移位。

相比之下,欧几里得算法似乎就没有多少吸引力了,因为步骤 A2 要求 u 除以 v 的多精度除法,但是这个困难实际上并不像它表面上看到的那么大,因为我们将 4.5.3 小节证明,商 $\lfloor u/v \rfloor$ 几乎总是非常小的;例如,采取随机输入,商 $\lfloor u/v \rfloor$ 将在近乎 99.856% 的情况下小于 1000。因此,利用单精度计算,连同比较简单的计算 $u - qv$ 的操作,其中 q 是单精度数,几乎总是有可能求出 $\lfloor u/v \rfloor$ 和 $(u \bmod v)$ 。而且,如果原来 u 比 v 大得多(例如,初始数据可能有这种形式),我们对于有很大的商 q 并不介意,因为在这种情况下当以 $u \bmod v$ 代替 u 时,欧几里得算法向前进了一大步。

通过使用 D. H. Lehmer [AMM **45** (1938), 227~233] 给出的一个方法,当涉及高精度的数时,可实现欧几里得算法在速度方面的一个重大改进。当仅处理很大数的前导数字时,有可能对大多数计算用单精度算术,从而大大减少了所涉及的多精度运算的个数。通过进行“虚拟的”计算代替真正的计算,我们能节省大量的时间。

例如,假设我们正在使用仅有四位数字的字的一个机器,考虑一对八位数 $u = 27182818$, $v = 10000000$ 。命 $u' = 2718$, $v' = 1001$, $u'' = 2719$, $v'' = 1000$; 则 u'/v' 和 u''/v'' 都是 u/v 的近似,且

$$u'/v' < u/v < u''/v'' \quad (27)$$

比值 u/v 确定欧几里得算法中所得到的商的序列。如果同时对单精度值 (u'/v') 和 (u''/v'') 执行欧几里得算法,直到得到一个不同的商为止,则不难看出,如果我们采用的是多精度数 (u, v) ,到此为止的商序列应是相同的。于是考虑当应用欧几里得算法于 (u', v') 和 (u'', v'') 时会发生什么情况:

u'	v'	q'	u''	v''	q''
2718	1001	2	2719	1000	2
1001	716	1	1000	719	1
716	285	2	719	281	2
285	146	1	281	157	1
146	139	1	157	124	1
139	7	19	124	33	3

在两种情况下头五个商相同,所以它们必定是真正的一些商。但在第六步以后,我们发现 $q' \neq q''$,所以停止单精度计算。我们已经获得了这样的知识,即如果我们使用的是原来的多精度数过程的话,则计算过程本应如下所列

$$\begin{array}{ccc}
 u & v & q \\
 u_0 & v_0 & 2 \\
 v_0 & u_0 - 2v_0 & 1 \\
 u_0 - 2v_0 & -u_0 + 3v_0 & 2 \\
 -u_0 + 3v_0 & 3u_0 - 8v_0 & 1 \\
 3u_0 - 8v_0 & -4u_0 + 11v_0 & 1 \\
 -4u_0 + 11v_0 & 7u_0 - 19v_0 & ?
 \end{array} \quad (28)$$

(下个商位于3和9之间的某种。)不管在 u 和 v 当中有多少位数字,只要(27)成立,则欧几里得算法的头五步总和(28)相同。因此我们可以避免头五步的多精度运算,而用 $-4u_0 + 11v_0$ 和 $7u_0 - 19v_0$ 的一个多精度计算来代替它们。在这种情况下,我们将得到 $u = 1268728$, $v = 279726$;这个计算现在将以类似的方式以 $u' = 1268$, $v' = 280$, $u'' = 1269$, $v'' = 279$, 等等进行。如果有一个更大的累加器,则单精度计算还可以做更多步;我们的例子表明,一个复合步骤当中仅仅组合了欧几里得算法的5个循环,但对于(比如说)10位数字的字长,我们一次大约能做12个循环。4.5.3小节中证明的结果意味着,在每次迭代时可被代替的多精度循环的数目,实际上同在单精度计算中所用的数字个数成正比。

Lehmer 的方法可系统阐述如下:

算法 L(对大数的欧几里得算法) 设 u, v 是以多精度表示的非负整数,且 $u \geq v$ 。本算法利用辅助的单精度 p 位数字的变量 a, v, A, B, C, D, T, q 和辅助的多精度变量 t 和 w , 计算 u 和 v 的最大公因子。

L1. [初始化] 如果 v 小到足以表示成单精度的值,则通过算法 A 计算 $\gcd(u, v)$ 并终止这个计算。否则,设 a 是 u 的 p 个前导数字并设 v 是 v 的 p 个对应数字;换言之,如果采用 b 进制记法,则 $a \leftarrow \lfloor u/b^k \rfloor$, $v \leftarrow \lfloor v/b^k \rfloor$, 其中 k 在条件 $a < b^p$ 的前提下尽可能地小。

置 $A \leftarrow 1, B \leftarrow 0, C \leftarrow 0, D \leftarrow 1$ 。在算法 A 对多精度数的等价动作下,这些变

量表示(28)中的系数,其中

$$u = Au_0 + Bv_0, \quad v = Cu_0 + Dv_0 \quad (29)$$

借助于上边例子中的记法,我们也有

$$u' = a + B, v' = v + D, u'' = a + A, v'' = v + C \quad (30)$$

- L2. [测试商]** 置 $q \leftarrow \lfloor (a + A)/(v + C) \rfloor$ 。如果 $q \neq \lfloor (a + B)/(v + D) \rfloor$, 则转到步骤 L4。(这一步骤测试在上述例子的记法下, 是否 $q' \neq q''$ 。当进行这一步计算时, 在一些特殊情况下, 可能出现单精度溢出, 但仅当 $a = b^p - 1$, 且 $A = 1$ 或当 $v = b^p - 1$ 和 $D = 1$ 时才会出现; 由于(30), 条件

$$\begin{aligned} 0 \leq a + A \leq b^p, \quad 0 \leq v + C < b^p \\ 0 \leq a + B \leq b^p, \quad 0 \leq v + D \leq b^p \end{aligned} \quad (31)$$

将总是成立。可能有 $v + C = 0$ 或 $v + D = 0$, 但不能两者同时成立; 因此, 在这一步骤除以 0 就意味着“直接转到 L4”。)

- L3. [模拟欧几里得算法]** 置 $T \leftarrow A - qC, A \leftarrow C, C \leftarrow T, T \leftarrow B - qD, B \leftarrow D, D \leftarrow T, T \leftarrow a - qv, a \leftarrow v, v \leftarrow T$, 并转回步骤 L2。(在(29)的约定下, 这些单精度运算如同在(28)中那样都等价于多精度运算。)

- L4. [多精度步骤]** 如果 $B = 0$, 则利用多精度除法置 $t \leftarrow u \bmod v, u \leftarrow v, v \leftarrow t$ 。(这仅当单精度运算不能模拟任何多精度运算时才发生。它意味着欧几里得算法要求一个非常大的商, 而这是极端稀少的情况。)否则, 置 $t \leftarrow Au, t \leftarrow t + Bv, w \leftarrow Cu, w \leftarrow w + Dv, u \leftarrow t, v \leftarrow w$ (直接利用多精度运算)。转回到步骤 L1。 ─

由于(31), 贯穿于这个计算过程中 A, B, C, D 的值始终是单精度的数。

算法 L 要求比算法 B 有更复杂的程序, 但对于很大的数, 在许多计算机上它将是更快的。然而, 算法 B 的二进技术可以类似的方式加快(见习题 38)到它继续获胜的程度。算法 L 有这样一个优点, 它确定在欧几里得算法中得到的商的序列, 而这有许多应用(例如, 请见 4.5.3 小节中的习题 43, 47, 49 和 51)。也可参见习题 4.5.3-46。

*** 对二进算法的分析** 现在我们通过研究算法 B 的运算时间来结束这一节, 以便论证早先指出的一些公式。

算法 B 的特性的精确确定看起来极难推导, 但我们可以借助它的特性的一个近似模拟来开始研究它。假设 u 和 v 都是奇数且 $u > v$, 以及

$$\lfloor \lg u \rfloor = m, \quad \lfloor \lg v \rfloor = n \quad (32)$$

(于是, u 是一个 $(m+1)$ 位二进位数, 而 v 是一个 $(n+1)$ 位二进位数。)考虑算法 B 的一个减和移位循环, 即在步骤 B6 开始而后在步骤 B5 结束之后停止的一个操作。对于 $u > v$ 的每个减和移位循环, 形成 $u - v$ 并且向右移动这个数量直到得到代替 u 的一个奇数为止。在随机条件下, 我们将期望有大约一半的时间有 $u' = (u - v)/2$, 有大约四分之一的时间有 $u' = (u - v)/4$, 有大约八分之一的时间 $u' = (u - v)/8$, 等等。我们有

$$\lfloor \lg u' \rfloor = m - k - r \quad (33)$$

其中 k 是 $u - v$ 右移的位数, r 是 $\lfloor \lg u \rfloor - \lfloor \lg(u - v) \rfloor$, 即从 u 减 v 期间左边失去的位数。注意, 当 $m \geq n + 2$ 时 $r \leq 1$, 而当 $m = n$ 时 $r \geq 1$ 。

k 和 r 之间的交互是十分混乱的(请见习题 20), 但 Richard Brent 通过假定 u 和 v 都足够大以致一个连续的分布可描述比例 v/u , 同时 k 离散地变化, 发现了分析近似特性的一个很好的方法。[请见 J. F. Traub 所编 *Algorithms and Complexity* (New York: Academic Press, 1976), 321–355.] 我们假定 u 和 v 是实质上随机的很大整数, 除非它们是奇数和它们的比有一个确定的概率分布。于是除非 t 将是偶数, 否则在步骤 B6 中的量 $t = u \cdot v$ 的最低有效位实质上是随机的。因此 t 将以 2^{-k} 的概率是 2^k 的奇数倍。这是在减和移位循环中需要的 k 个右移的近似概率。换言之, 如果我们假定步骤 B4 总是以 $1/2$ 的概率转移到 B3, 则我们得到算法 B 的性质的一个合理的近似。

令 $G_n(x)$ 是在这个假定下在 n 个减法 and 移位循环被实施之后 $\min(u, v)/\max(u, v) \geq x$ 的概率。如果 $u \geq v$ 而且如果恰好实施了 k 个右移, 则比 $X = v/u$ 被改变成 $X' = \min(2^k v/(u - v), (u - v)/2^k v) = \min(2^k X/(1 - X), (1 - X)/2^k X)$ 。于是我们将有 $X' \geq x$ 当且仅当 $2^k X/(1 - X) \geq x$ 和 $(1 - X)/2^k X \geq x$; 而这和

$$\frac{1}{1 + 2^k/x} \leq X \leq \frac{1}{1 + 2^k x} \quad (34)$$

是相同的。因此对于 $0 \leq x \leq 1$, $G_n(x)$ 满足有趣的递推式

$$G_{n+1}(x) = \sum_{k \geq 1} 2^{-k} \left(G_n \left(\frac{1}{1 + 2^k/x} \right) - G_n \left(\frac{1}{1 + 2^k x} \right) \right) \quad (35)$$

其中 $G_0(x) = 1 - x$ 。计算的实验指出, $G_n(x)$ 快速地收敛到一个极限分布 $G_\infty(x) = G(x)$, 尽管收敛性的形式证明看来是困难的。我们将假定 $G(x)$ 存在; 因此它满足, 对于 $0 < x \leq 1$,

$$G(x) = \sum_{k \geq 1} 2^{-k} \left(G \left(\frac{1}{1 + 2^k/x} \right) - G \left(\frac{1}{1 + 2^k x} \right) \right) \quad (36)$$

$$G(0) = 1; \quad G(1) = 0 \quad (37)$$

令

$$\begin{aligned} S(x) &= \frac{1}{2} G \left(\frac{1}{1 + 2x} \right) + \frac{1}{4} G \left(\frac{1}{1 + 4x} \right) - \frac{1}{8} G \left(\frac{1}{1 + 8x} \right) + \cdots \\ &= \sum_{k \geq 1} 2^{-k} G \left(\frac{1}{1 + 2^k x} \right) \end{aligned} \quad (38)$$

于是我们有

$$G(x) = S(1/x) - S(x) \quad (39)$$

定义

$$G(1/x) = -G(x) \quad (40)$$

使得对于所有 $x > 0$, (39) 成立是方便的。当 x 从 0 跑到 ∞ 时, $S(x)$ 从 0 增到 1, 因此 $G(x)$ 从 +1 减小成 -1。当然当 $x > 1$ 时 $G(x)$ 不再是一个概率; 但尽管如此它

仍然是有意义的(请见习题 23)。

我们将假定有幂级数 $\alpha(x), \beta(x), \gamma_m(x), \delta_m(x), \lambda(x), \mu(x), \sigma_m(x), \tau_m(x)$ 和 $\rho(x)$ 使得

$$G(x) = \alpha(x) \lg x + \beta(x) + \sum_{m=1}^{\infty} (\gamma_m(x) \cos 2\pi m \lg x + \delta_m(x) \sin 2\pi m \lg x) \quad (41)$$

$$S(x) = \lambda(x) \lg x + \mu(x) + \sum_{m=1}^{\infty} (\sigma_m(x) \cos 2\pi m \lg x + \tau_m(x) \sin 2\pi m \lg x) \quad (42)$$

$$\rho(x) = G(1+x) = \rho_1 x + \rho_2 x^2 + \rho_3 x^3 + \rho_4 x^4 + \rho_5 x^5 + \rho_6 x^6 + \cdots \quad (43)$$

因为可以证明(35)的解 $G_n(x)$ 对于 $n \geq 1$ 有这个性质。(例如, 见习题 30。)对于 $|x| < 1$ 这个幂级数收敛。

由等式(36)~(43), 关于 $\alpha(x), \dots, \rho(x)$ 我们可以推导出什么呢? 首先从(38), (40)和(43), 我们有

$$2S(x) = G(1/(1+2x)) + S(2x) = S(2x) - \rho(2x) \quad (44)$$

因此等式(42)成立当且仅当

$$2\lambda(x) = \lambda(2x) \quad (45)$$

$$2\mu(x) = \mu(2x) + \lambda(2x) - \rho(2x) \quad (46)$$

$$2\sigma_m(x) = \sigma_m(2x) \quad 2\tau_m(x) = \tau_m(2x), \text{ 对于 } m \geq 1 \quad (47)$$

关系式(45)告诉我们 $\lambda(x)$ 只不过是 x 的一个常数倍; 由于这个常数是负的, 所以我们将写

$$\lambda(x) = -\lambda x \quad (48)$$

(相关的系数结果是

$$\lambda = 0.39792\ 26811\ 88316\ 64407\ 67071\ 61142\ 65498\ 23098 + \quad (49)$$

但尚不知计算它的容易的方法。)关系式(46)告诉我们当 $k > 1$ 时 $p_1 = -\lambda$, 而且 $2\mu_k = 2^k u_k - 2^k \rho_k$; 换言之, 对于 $k \geq 2$,

$$u_k = \rho_k / (1 - 2^{1-k}) \quad (50)$$

从(47)我们还知道两个幂级数系

$$\sigma_m(x) = \sigma_m x, \quad \tau_m(x) = \tau_m x \quad (51)$$

都不过是线性函数(对于 $\gamma_m(x)$ 和 $\delta_m(x)$ 这不成立)。

在(44)中以 $1/(2x)$ 代替 x 得出

$$2S(1/(2x)) = S(1/x) + G(x/(1+x)) \quad (52)$$

而且当 x 接近于 0 时(39)把这个等式转换成 G 和 S 之间的一个关系

$$2G(2x) + 2S(2x) = G(x) + S(x) + G(x/(1+x)) \quad (53)$$

当把这个等式两边展开成幂级数时 $\lg x$ 的系数必须一致, 因此

$$2\alpha(2x) - 4\lambda x = \alpha(x) - \lambda x + \alpha(x/(1+x)) \quad (54)$$

等式(54)是定义 $\alpha(x)$ 的一个递推式,事实上,考虑满足

$$\psi(z) = \frac{1}{2} \left(z + \psi\left(\frac{z}{2}\right) + \psi\left(\frac{z}{2+z}\right) \right), \quad \psi(0) = 0, \psi'(0) = 1 \quad (55)$$

的函数 $\psi(z)$, 于是(54)指出

$$\alpha(x) = \frac{3}{2} \lambda \psi(x) \quad (56)$$

而且,(55)的迭代产生

$$\begin{aligned} \psi(z) &= \frac{z}{2} \left(\frac{1}{1} + \frac{1}{2} \left(\frac{1}{2} + \frac{1}{2+z} \right) + \frac{1}{4} \left(\frac{1}{4} + \frac{1}{4+z} + \frac{1}{4+2z} + \frac{1}{4+3z} \right) + \cdots \right) \\ &= \frac{z}{2} \sum_{k \geq 0} \frac{1}{2^k} \sum_{0 \leq j < 2^k} \frac{1}{2^k + jz} \end{aligned} \quad (57)$$

由此得出, $\psi(z)$ 的幂级数展开是

$$\psi(z) = \sum_{n \geq 1} (-1)^{n-1} \psi_n z^n, \quad \psi_n = \frac{1}{2n} \sum_{k=0}^{n-1} \frac{B_k}{2^{k+1}-1} \binom{n}{k} + \frac{\delta_{n,1}}{2} \quad (58)$$

请见习题 27。对于 ψ_n 的这一公式令人惊异地类似于在同数字查找树算法即等式 6.3-(18) 相关联中出现的一个表达式。习题 28 证明 $\psi_n = \Theta(n^{-2})$ 。

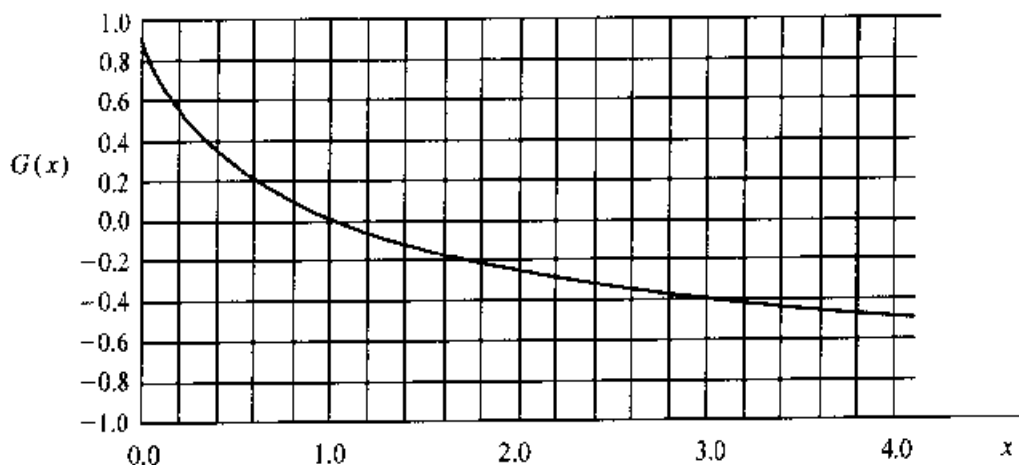


图 10 在二进制 gcd 算法中的极限比分布

除了常数 $\lambda = -\rho_1$, 我们现在知道 $\alpha(x)$, 而除了系数 μ_1 外, (50) 把 $\mu(x)$ 和 $\rho(x)$ 关联起来。习题 25 的答案表明借助于 $\rho_1, \rho_3, \rho_5, \dots$ 可以表达 $\rho(x)$ 的所有系数; 而且, 常数 σ_m 和 τ_m 可以通过用解习题 29 的方法来计算, 而且在函数 $\gamma_m(x)$ 和 $\delta_m(x)$ 的系数之间的复杂关系也成立。然而, 看起来没有办法来计算进入 $G(x)$ 的各种函数的所有系数, 除非通过精巧的数值方法来迭代递推式(36)。

一旦我们已经计算了 $G(x)$ 的好的近似, 我们可以估计算法 B 的渐近平均运行时间如下: 如果 $u \geq v$ 且实施 k 个右移, 则数量 $Y = uv$ 被改变成 $Y' = (u-v)v/2^k$; 因此比例 Y/Y' 是 $2^k/(1-X)$, 其中 $X = v/u$ 以 $G(x)$ 的概率大于等于 x 。因此平均说来 uv 中的二进制数以常数

$$b = E \lg(Y/Y') = \sum_{k \geq 1} 2^{-k} (f_k(0) + \int_0^1 G(x) f'_k(x) dx)$$

递减,其中 $f_k(x) = \lg(2^k/(1-x))$;我们有

$$b = \sum_{k=1}^{\infty} 2^{-k} \left(k + \int_0^1 \frac{G(x)dx}{(1-x)\ln 2} \right) = 2 + \int_0^1 \frac{G(x)dx}{(1-x)\ln 2} \quad (59)$$

当最终 $u=v$ 时, $\lg uv$ 的预期值将近似于 0.9779(请见习题 14);因此算法 B 的减和移位循环的总数将近似于 $1/b$ 乘以 $\lg uv$ 的初始值。由对称性,这大约是 $2/b$ 乘以 $\lg u$ 的初始值。1997 年由 Richard Brent 特所进行的数值计算给出对于这个基础常数的值

$$2/b = 0.70597\ 12461\ 01916\ 39152\ 93141\ 35852\ 88176\ 66677 + \quad (60)$$

Brigitte Vallée 对这些函数的深入研究令她怀疑常数 λ 和 b 可能通过值得注意的公式

$$\frac{\lambda}{b} = \frac{2 \ln 2}{\pi^2} \quad (61)$$

来相关联。确实,由 Brent 所计算的值得同这逗人的猜测完全一致。Vallée 已经使用非常重要的严格的“动态”方法成功地分析了算法 B [Algorithmica 22 (1998), 660~685]。

让我们回到我们在 (32) 中的假定,即 u 和 v 是在 $2^m \leq u < 2^{m+1}$ 和 $2^n \leq v < 2^{n+1}$ 范围中的奇数。通过好几百万个随机输入和在 $29 \leq m, n \leq 37$ 范围中的 m 和 n 的各种值对于算法 B 的经验检验指出,这个算法实际的平均特性是由下列公式给出的 ($m \geq n$):

$$C \approx \frac{1}{2}m + 0.203n + 1.9 - 0.4(0.6)^{m-n} \quad (62)$$

$$D \approx m + 0.41n - 0.5 - 0.7(0.6)^{m-n}$$

而且这些观察到的平均值有一个相当小的标准差。(62)中 m 的系数 $1/2$ 和 1 可以严格地加以验证(请见习题 21)。

如果我们转而假定, u 和 v 是在范围

$$1 \leq u \leq 2^N, \quad 1 \leq v < 2^N \quad (63)$$

的范围内独立和一致地分布的任何整数,则我们可以从已经给出的数据计算 C 和 D 的平均值(请见习题 22):

$$C \approx 0.70N + O(1), \quad D \approx 1.41N + O(1) \quad (64)$$

这同对于 $N \leq 30$ 对好几百万个随机输入所做的进一步的经验检验的结果完全一致;后者表明,给定输入 u 和 v 的这个分布,我们可以取

$$C = 0.70N - 0.5, \quad D = 1.41N - 2.7 \quad (65)$$

作为这些值的适当估计。

在 (63) 的假定之下,在 Brent 的算法 B 的连续模型中的理论分析预测 C 和 D 将渐近地等于 $2N/b$ 和 $4N/b$, 其中 $2/b \approx 0.70597$ 是 (60) 中的常数。同实验的一致性是如此之好,因此 Brent 常数 $2/b$ 必定是 (65) 中的数“0.70”的值,因此我们应该以 (62) 中的 0.206 代替 0.203。

这就完成了我们对于 C 和 D 的平均值的研究,出现在算法 B 的运行时间中的

其它三个量十分容易分析;请见习题 6.7 和 8。

既然我们知道平均说来算法 B 近似的特性如何,让我们来考虑一个“最坏情况”的场景:在某种意义下 u 和 v 最难处理的值是什么? 如果我们像在以前一样假定

$$\lfloor \lg u \rfloor = m \text{ 和 } \lfloor \lg v \rfloor = n$$

我们要来找出使算法最慢地运行的 u 和 v 。当考虑辅助的簿记时,减法比起移位稍微花更长些的时间。因此这个问题也可以通过求需要最多减法的输入 u 和 v 来重新表述。答案是令人惊讶的;尽管一个幼稚的分析将预测有可能有高得多的 C 值(请见习题 35),但 C 的极大值恰是

$$\max(m, n) + 1 \quad (66)$$

对于最坏情况(66)的推导是十分有趣的,因此把它留给读者们自己求解(请见习题 36 和 37)。

习 题

1. [M21] 怎样从(6)和(7)容易地导出(8),(9),(10),(11)和(12)?

2. [M22] 假定 u 整除 $v_1 v_2 \cdots v_n$, 证明 u 整除

$$\gcd(u, v_1) \gcd(u, v_2) \cdots \gcd(u, v_n)$$

3. [M23] 证明使得 $\text{lcm}(u, v) = n$ 的有序正整数偶 (u, v) 的个数是 n^2 的因子个数。

4. [M21] 假定 u 和 v 是正整数, 证明存在 u 的因子 u' 和 v 的因子 v' , 使得 $u' \perp v'$ 和 $u'v' = \text{lcm}(u, v)$ 。

► 5. [M26] 设计一个用于计算两个整数的最大公因子的算法(类似于算法 B), 它以两数的平衡的三进制表示为基础。把它应用于计算 $\gcd(40902, 24140)$ 。

6. [M22] 假定 u 和 v 是随机正整数, 试求对程序 B 的计时起作用的量 A (它是在准备阶段对 u 和 v 两者的右移次数)的平均值和标准偏差。

7. [M20] 分析对程序 B 的计时起作用的量 B 。

► 8. [M25] 证明在程序 B 中, E 的平均值近似地等于 $\frac{1}{2} C_{\text{ave}}$, 其中 C_{ave} 是 C 的平均值。

9. [18] 利用算法 B 和手算, 求 $\gcd(31408, 2718)$ 。利用算法 X, 也求整数 m 和 n 使得 $31408m + 2718n = \gcd(31408, 2718)$ 。

► 10. [HM24] 设 q_n 是在 $1 \leq u, v \leq n$ 的范围内使得 $u \perp v$ 的整数有序数偶 (u, v) 的个数。本题的任务是证明: $\lim_{n \rightarrow \infty} q_n / n^2 = 6/\pi^2$, 由此建立定理 D。

a) 使用容斥原理(1.3.3 小节)证明

$$q_n = n^2 \sum_{p_1} \lfloor n/p_1 \rfloor^2 + \sum_{p_1 < p_2} \lfloor n/p_1 p_2 \rfloor^2 - \cdots$$

其中求和取遍所有素数 p_i 。

b) 莫比乌斯(Mobius)函数 $\mu(n)$ 由下列规则定义: $\mu(1) = 1$, 如果 p_1, p_2, \dots, p_r 是不同素数, $\mu(p_1 p_2 \cdots p_r) = (-1)^r$, 而且如果 n 可由一个素数的平方整除, $\mu(n) = 0$ 。证明 $q_n = \sum_{k \geq 1} \mu(k) \cdot \lfloor n/k \rfloor^2$ 。

c) 作为 b) 的一个推论, 证明 $\lim_{n \rightarrow \infty} q_n / n^2 = \sum_{k \geq 1} \mu(k) / k^2$ 。

d) 证明 $(\sum_{k=1}^{\infty} \mu(k)/k^2)(\sum_{m=1}^{\infty} 1/m^2) = 1$. 提示: 当这些级数绝对收敛时, 我们有

$$\left(\sum_{k=1}^{\infty} a_k/k^2\right)\left(\sum_{m=1}^{\infty} b_m/m^2\right) = \sum_{n=1}^{\infty} \left(\sum_{d|n} a_d b_{n/d}\right)/n^2$$

11. [M22] $\gcd(u, v) \leq 3$ 的概率是多少? (见定理 D₀) $\gcd(u, v)$ 的平均值是多少?

12. [M24] (E. Cesàro) 如果 u 和 v 是随机正整数, 它们可能的公共(正)因子的平均个数是多少? (提示: 见习题 10(d) 中的等式, 取 $a_k = b_m = 1$.)

13. [HM23] 假定 u 和 v 都是随机的奇正整数, 证明它们以 $8/\pi^2$ 的概率互素。

► 14. [HM25] 当 u 和 v 是 (a) 随机整数, (b) 随机正奇整数时, $\ln \gcd(u, v)$ 的期望值是多少?

15. [M21] 当算法 X 终止时, v_1 和 v_2 的值是多少?

► 16. [M22] 给定正整数 u, v 和 m , 且 v 与 m 互素, 试设计一个算法做 u 除以 v 模 m 的除法。换言之, 你的算法应该求在范围 $0 \leq w < m$ 内的 w , 使得 $u = vw \pmod{m}$ 。

► 17. [M20] 给定两个整数 u 和 v 使得 $uv \equiv 1 \pmod{2^e}$, 说明如何来计算一个整数 u' , 使得 $u'v \equiv 1 \pmod{2^{2e}}$ 。[这导致计算一个奇数模 2 的幂的倒数的一个快速的算法, 因为我们可以对于 $e=8$ 或 $e=16$, 以所有这样的倒数的一个表开始。]

► 18. [M24] 说明当 u 和 v 很大时, 如何能推广算法 L (像把算法 A 推广成算法 X 那样) 以得到 (15) 的解?

19. [21] 利用正文中的方法求下列方程组的整数通解:

$$(a) \begin{cases} 3x + 7y + 11z = 1 \\ 5x - 7y - 5z = 3 \end{cases} \quad (b) \begin{cases} 3x + 7y + 11z = 1 \\ 5x + 7y - 5z = -3 \end{cases}$$

20. [M37] 设 u 和 v 是奇整数, 独立和一致地分布于范围 $2^m \leq u < 2^{m+1}$, $2^n \leq v < 2^{n+1}$ 内。作为 m, n, m' 和 n' 的函数, 算法 B 中的一个减和移位循环把 u 和 v 减小到范围 $2^{m'} \leq u < 2^{m'+1}$, $2^{n'} \leq v < 2^{n'+1}$ 内的精确概率是多少?

21. [HM26] 设 C_{mn}, D_{mn} 分别是在算法 B 中减和移位循环的平均次数, 当 u 和 v 是奇数时, $\lfloor \lg u \rfloor = m, \lfloor \lg v \rfloor = n$ 。证明对于固定的 n , 当 $m \rightarrow \infty$ 时, $C_{mn} = \frac{1}{2}m + O(1)$ 和 $D_{mn} = m + O(1)$ 。

22. [M28] 继续上题, 证明: 如果对于某些常数 α, β 和 γ , $C_{mn} = \alpha m + \beta n + \gamma$, 则

$$\sum_{1 \leq n < m \leq N} (N-m)(N-n)2^{m+n-2}C_{mn} = 2^{2N} \left(\frac{11}{27}(\alpha + \beta)N + O(1) \right)$$

$$\sum_{1 \leq m \leq n \leq N} (N-n)^2 2^{2n-2}C_{mn} = 2^{2N} \left(\frac{5}{27}(\alpha + \beta)N + O(1) \right)$$

► 23. [M20] 当算法 B 以大的随机整数开始时, 在算法的 n 次减法和移位循环之后, $v/u \leq x$ 的概率是多少? (这里 x 是任何大于或等于 0 的实数; 我们不假定 $u \geq v$.)

24. [M20] 假定在步骤 B6 中 $u > v$, 并假定比 v/u 有 Brent 极限分布 G , 下次遇到步骤 B6 时 $u < v$ 的概率是多少?

25. [M21] 等式 (46) 意味着 $\rho_1 = -\lambda$; 证明 $\rho_2 = \lambda/2$ 。

26. [M22] 证明当 $G(x)$ 满足 (36) ~ (40) 时, 我们有

$$2G(x) - 5G(2x) + 2G(4x) = G(1+2x) - 2G(1+4x) \\ + 2G(1+1/x) - G(1+1/2x)$$

27. [M22] 证明 (58), 它借助于伯努利数表达 ϕ_n 。

28. [HM36] 研究 ϕ_n 的渐近特性。提示: 请见习题 6.3-34。

► 29. [HM26] (R. P. Brent) 求 $G_1(x)$, 即像在 (35) 中定义的那样, 它是算法 B 头一个减和移

位循环之后 $\min(u, v)/\max(u, v)$ 的分布。提示:令 $S_{n+1}(x) = \sum_{k=1}^{\infty} 2^{-k} G_n(1/(1+2^k x))$, 并使用调和数的 Mellin 变换方法[请见 P. Flajolet, X. Gourdon 和 P. Damas, *Theor. Comp. Sci.* **144** (1995), 3 ~ 58]。

30. [HM39] 继续上一题, 确定 $G_2(x)$ 。

31. [HM46] 证明或反驳 Vallée 的猜测(61)。

32. [M47] 是否有满足(36)和(37)的唯一的连续函数 $G(x)$?

33. [M46] 试分析在程序 B 之后论述的 Harris 的“二进制欧几里得算法”。

34. [HM49] 找出 Brent 的模型描述算法 B 的渐近特性的一个严格证明。

35. [M23] 考虑对于所有非负整数 $m, n \geq 0$, 都有顶点 (m, n) 的一个有向图, 而且每当算法 B 的一个减和移位循环有可能把满足 $\lfloor \lg u \rfloor = m$ 和 $\lfloor \lg v \rfloor = n$ 的整数 u 和 v 转换成满足 $\lfloor \lg u' \rfloor = m'$ 和 $\lfloor \lg v' \rfloor = n'$ 的整数 u' 和 v' 时, 从 (m, n) 和 (m', n') 有一条有向边; 对于所有 $n \geq 0$ 也还有一个特殊的“停止”顶点, 以及从 (n, n) 到“停止”的有向边。从 (n, n) 到停止顶点最长的通路长度是多少? (这给出算法 B 的极长运行时间的上限。)

▶ 36. [M28] 给定 $m \geq n \geq 1$, 求使得 $\lfloor \lg u \rfloor = m$ 和 $\lfloor \lg v \rfloor = n$ 的 u 和 v 的值, 使算法 B 需要 $m+1$ 个减法步。

37. [M32] 试证明算法 B 的减法步骤 B6 不会执行 $1 + \lfloor \lg \max(u, v) \rfloor$ 次以上。

▶ 38. [M32] (R. W. Gosper) 说明, 使用类似于在算法 L 中的那些思想, 如何修改对于大数的算法 B。

▶ 39. [M28] (V. R. Pratt) 把算法 B 推广成为类似于算法 X 的一个算法 Y。

▶ 40. [M25] (R. P. Brent 和孔祥重(H. T. Kung)) 从硬件实现的观点看, 下列二进制 gcd 算法的变形要比算法 B 更好些, 因为它不要求对 $u \cdot v$ 符号的测试。假设 u 是奇数; u 和 v 可以为正或为负。

K1. [初始化] 置 $c \leftarrow 0$ 。(这个计数器估计 $\lg |u|$ 和 $\lg |v|$ 之间的差。)

K2. [完成了吗?] 如果 $v = 0$, 则以 $|u|$ 作为答案结束。

K3. [使 v 为奇数] 置 $v \leftarrow v/2$ 和 $c \leftarrow c+1$ 零次或多次, 直到 v 为奇数为止。

K4. [使 $c \leq 0$] 如果 $c > 0$, 则交换 $u \leftrightarrow v$ 并且置 $c \leftarrow -c$ 。

K5. [减少] 置 $w \leftarrow (u+v)/2$ 。如果 w 为偶数, 置 $v \leftarrow w$; 否则置 $v \leftarrow w - v$ 。返回到步骤 K2。

试证步骤 K2 顶多执行 $2 + 2 \lg \max(|u|, |v|)$ 次。

41. [M22] 当 m 和 n 是非负整数时, 使用欧几里得算法求 $\gcd(10^m - 1, 10^n - 1)$ 的一个简单公式。

42. [M30] 计算行列式

$$\begin{vmatrix} \gcd(1,1) & \gcd(1,2) & \cdots & \gcd(1,n) \\ \gcd(2,1) & \gcd(2,2) & \cdots & \gcd(2,n) \\ \vdots & \vdots & & \vdots \\ \gcd(n,1) & \gcd(n,2) & \cdots & \gcd(n,n) \end{vmatrix}$$

*4.5.3 欧几里得算法的分析

欧几里得算法的执行时间依赖于 T , 即实施除法步骤 A2 的次数(见算法 4.5.2A 和程序 4.5.2A)。量 T 在其它算法的运行时间中, 例如, 在满足互反公式的

函数求值算法(见 3.3.3 小节)的执行时间中,也是一个重要的因数。在这一节中,我们将看到对于这个量 T 的数学上的分析是有趣的和有教益的。

与连分数的关系 欧几里得算法同连分数有着密切的关系,连分数形如

$$\cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{\ddots + \cfrac{b_n}{a_{n-1} + \cfrac{b_n}{a_n}}}}} = b_1 / (a_1 + b_2 / (a_2 + b_3 / (\cdots / (a_{n-1} + b_n / a_n) \cdots))) \quad (1)$$

连分数有一套美妙的理论,它是许多书的主题,例如, O. Perron, *Die Lehre von den Kettenbrüchen*, 第 3 版 (Stuttgart: Teubner, 1954), 2 卷; A. Khinchin, *Continued Fractions*, Peter Wynn 译 (Groningen: P. Noordhoff, 1963); H. S. Wall, *Analytic Theory of Continued Fractions* (New York: Van Nostrand, 1948)。关于这个课题的早期历史,也见 Claude Brezinski, *History of Continued Fractions and Padé Approximants* (Springer, 1991)。在这里有必要把我们自己限定在比较简略的对于这个理论的讨论,仅仅研究那些帮助我们对欧几里得算法的特性有更多了解的方面。

我们主要感兴趣的连分数是(1)中所有 b 都等于 1 的情形。为了记法的方便,定义

$$\|x_1, x_2, \cdots, x_n\| = 1 / (x_1 + 1 / (x_2 + 1 / (\cdots (x_{n-1} + 1 / x_n) \cdots))) \quad (2)$$

于是,例如

$$\|x_1\| = \frac{1}{x_1}, \quad \|x_1, x_2\| = \frac{1}{x_1 + 1/x_2} = \frac{x_2}{x_1 x_2 + 1} \quad (3)$$

如果 $n=0$, 则符号 $\|x_1, \cdots, x_n\|$ 表示 0。对于 $n \geq 0$, 我们也定义 n 个变量的所谓连续多项式 $K_n(x_1, x_2, \cdots, x_n)$, 其规则为

$$K_n(x_1, x_2, \cdots, x_n) = \begin{cases} 1 & \text{如果 } n = 0 \\ x_1 & \text{如果 } n = 1 \\ x_1 K_{n-1}(x_2, \cdots, x_n) + K_{n-2}(x_3, \cdots, x_n) & \text{如果 } n > 1 \end{cases} \quad (4)$$

于是, $K_2(x_1, x_2) = x_1 x_2 + 1$, $K_3(x_1, x_2, x_3) = x_1 x_2 x_3 + x_1 + x_3$, 等等。一般说来,如同欧拉在 18 世纪时所指出的, $K_n(x_1, x_2, \cdots, x_n)$ 是由 $x_1 x_2 \cdots x_n$ 开始, 并删除 0 个或多个非重叠的相继变量对 $x_j x_{j+1}$ 而得到的所有项之和; 共有 F_{n+1} 个这样的项。

连续多项式的基本性质是显公式

$$\|x_1, x_2, \cdots, x_n\| = K_{n-1}(x_2, \cdots, x_n) / K_n(x_1, x_2, \cdots, x_n) \quad n \geq 1 \quad (5)$$

这可以由归纳法证明, 因为它意味着

$$x_0 + \|x_1, \cdots, x_n\| = K_{n+1}(x_0, x_1, \cdots, x_n) / K_n(x_1, \cdots, x_n)$$

因此 $\|x_0, x_1, \cdots, x_n\|$ 是式右的倒数。

在

$$K_n(x_1, x_2, \dots, x_n) = K_n(x_n, \dots, x_2, x_1) \quad (6)$$

的意义下, K 多项式是对称的。这由上面欧拉的发现推出, 而且作为一个推论我们有, 对 $n > 1$,

$$K_n(x_1, \dots, x_n) = x_n K_{n-1}(x_1, \dots, x_{n-1}) + K_{n-2}(x_1, \dots, x_{n-2}) \quad (7)$$

K 多项式也满足重要的恒等式(见习题 4)

$$K_n(x_1, \dots, x_n) K_n(x_2, \dots, x_{n+1}) - K_{n+1}(x_1, \dots, x_{n+1}) K_{n-1}(x_2, \dots, x_n) = (-1)^n, n \geq 1 \quad (8)$$

后一等式同(5)在一起意味着

$$\|x_1, \dots, x_n\| = \frac{1}{q_0 q_1} - \frac{1}{q_1 q_2} + \frac{1}{q_2 q_3} - \dots + \frac{(-1)^{n-1}}{q_{n-1} q_n} \quad (9)$$

其中 $q_k = K_k(x_1, \dots, x_k)$ 。于是 K 多项式与连分数密切相关。

在范围 $0 \leq X < 1$ 中的每一实数 X 都有一个如下定义的正则连分数: 设 $X_0 = X$ 而且对于所有使 $X_n \neq 0$ 的 $n \geq 0$, 设

$$A_{n+1} = \lfloor 1/X_n \rfloor, \quad X_{n+1} = 1/X_n - A_{n+1} \quad (10)$$

如果 $X_n = 0$, 则量 A_{n+1} 和 X_{n+1} 无定义, 而且对于 X 的正则连分数是 $\|A_1, \dots, A_n\|$ 。如果 $X_n \neq 0$, 则这个定义保证了 $0 \leq X_{n+1} < 1$, 所以每个 A 全都是正整数。(10)的定义也意味着

$$X = X_0 - \frac{1}{A_1 + X_1} = \frac{1}{A_1 + 1/(A_2 + X_2)} = \dots$$

因此, 每当 X_n 有定义时, 对所有 $n \geq 1$,

$$X = \|A_1, \dots, A_{n-1}, A_n + X_n\| \quad (11)$$

特别是, 当 $X_n = 0$ 时, $X = \|A_1, \dots, A_n\|$ 。如果 $X_n \neq 0$, 则 X 总位于 $\|A_1, \dots, A_n\|$ 和 $\|A_1, \dots, A_n + 1\|$ 之间, 因为由(7), 量 $q_n = K_n(A_1, \dots, A_n + X_n)$ 随着 X_n 从 0 增到 1 而单调地从 $K_n(A_1, \dots, A_n)$ 递增至 $K_n(A_1, \dots, A_n + 1)$, 而且, 由(9), 当 q_n 增加时, 根据 n 是偶数或奇数, 连分数增加或减小。由(5), (7), (8)和(10),

$$\begin{aligned} |X - \|A_1, \dots, A_n\|| &= |\|A_1, \dots, A_n + X_n\| - \|A_1, \dots, A_n\|| = \\ &= |\|A_1, \dots, A_n, 1/X_n\| - \|A_1, \dots, A_n\|| = \\ &= \left| \frac{K_n(A_2, \dots, A_n, 1/X_n)}{K_{n+1}(A_1, \dots, A_n, 1/X_n)} - \frac{K_{n-1}(A_2, \dots, A_n)}{K_n(A_1, \dots, A_n)} \right| = \\ &= \frac{1}{1/(K_n(A_1, \dots, A_n) K_{n+1}(A_1, \dots, A_n, 1/X_n))} \leq \\ &= \frac{1}{1/(K_n(A_1, \dots, A_n) K_{n+1}(A_1, \dots, A_n, A_{n+1}))} \quad (12) \end{aligned}$$

因此, 除非 n 很小, 不然 $\|A_1, \dots, A_n\|$ 是 X 的极其接近的近似值。如果 X_n 对于所有的 n 都是非零的, 在这种情况下我们得到一个无穷的连分数 $\|A_1, A_2, A_3, \dots\|$, 这样的连分数的值定义为 $\lim_{n \rightarrow \infty} \|A_1, A_2, \dots, A_n\|$; 而且由不等式(12), 显然这个极

限等于 X 。

实数的正则连分数展开有若干类似于十进制数系中数的表示性质。如果用上述公式计算某些熟悉的实数的正则连分数展开,则我们发现,例如

$$\frac{8}{29} = // 3, 1, 1, 1, 2 //$$

$$\sqrt{\frac{8}{29}} = // 1, 1, 9, 2, 2, 3, 2, 2, 9, 1, 2, 1, 9, 2, 2, 3, 2, 2, 9, 1, 2, 1, 9, 2, 2, 3, 2, 2, 9, 1, \dots //$$

$$\sqrt[3]{2} = 1 + // 3, 1, 5, 1, 1, 4, 1, 1, 8, 1, 14, 1, 10, 2, 1, 4, 12, 2, 3, 2, 1, 3, 4, 1, 1, 2, 14, 3, \dots //$$

$$\pi = 3 + // 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, 2, 2, 1, 84, 2, 1, 1, 15, 3, 13, \dots //$$

$$e = 2 + // 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1, 1, 10, 1, 1, 12, 1, 1, 14, 1, 1, 16, 1, 1, 18, 1, \dots //$$

$$\gamma = // 1, 1, 2, 1, 2, 1, 4, 3, 13, 5, 1, 1, 8, 1, 2, 4, 1, 1, 40, 1, 11, 3, 7, 1, 7, 1, 1, 5, 1, 49, \dots //$$

$$\phi = 1 + // 1, \dots // \quad (13)$$

数 A_1, A_2, \dots 称做 X 的部分商。注意在 $\sqrt{8/29}$, ϕ 和 e 的部分商中出现的数的正则模式;习题 12 和 16 中讨论了关于这一特性的原因,在 $\sqrt[3]{2}$, π 或 γ 的部分商中没有明显的模式。

说明这一点是有趣的,古希腊人在发现了无理数的存在性之后,对实数的头一个定义实质上是以无穷连分数叙述的(后来他们采取了 Eudoxus 的建议,即 $x = y$ 应定义为“对于所有有理数 r ,当且仅当 $y < r$ 时 $x < r$)。见 O. Becker, *Quellen und Studien zur Geschichte Math., Astron., Physik B2* (1933), 311~333。

当 X 是有理数时,正则连分数自然地对应于欧几里得算法。我们假定 $X = v/u$, 其中 $u > v \geq 0$ 。正则连分数算法从 $X_0 = X$ 开始;我们定义 $U_0 = u, V_0 = v$ 。假定 $X_n = V_n/U_n \neq 0$, (10) 就变成

$$A_{n+1} = \lfloor U_n/V_n \rfloor, \quad X_{n+1} = U_n/V_n - A_{n+1} = (U_n \bmod V_n)/V_n \quad (14)$$

因此,如果我们定义

$$U_{n+1} = V_n, \quad V_{n+1} = U_n \bmod V_n \quad (15)$$

则在算法执行全过程中,条件 $X_n = V_n/U_n$ 都成立。而且, (15) 恰巧是在欧几里得算法中对变量 u, v 所做的变换(见算法 4.5.2A, 步骤 A2)。例如,由于 $\frac{8}{29} = // 3, 1, 1, 1, 2 //$, 我们知道应用于 $u = 29, v = 8$ 的欧几里得算法恰好要求 5 个除法步骤,而且在步骤 A2 中的商 $\lfloor u/v \rfloor$ 将逐次为 3, 1, 1, 1 和 2。当 $X_n = 0, n \geq 1$ 时,最后的部分商 A_n 必然总是 2 或更大,因为 X_{n-1} 小于 1。

从同欧几里得算法的这个对应关系,我们可以看出,当且仅当 X 是有理数时,对于 X 的正则连分数展开以 $X_n = 0$ 终止在某一步;因为很显然,如果 X 是无理数,则 X_n 就不能为 0,反过来我们知道,欧几里得算法总是有穷的。如果在实施欧几里得算法期间得到的部分商是 A_1, A_2, \dots, A_n , 则由 (5), 有

$$\frac{v}{u} = \frac{K_{n-1}(A_2, \dots, A_n)}{K_n(A_1, A_2, \dots, A_n)} \quad (16)$$

当 $A_1 = 0$ 时, 如果对 $u < v$ 应用欧几里得算法, 这个公式也成立。而且, 由于关系 (8), 连分数 $K_{n-1}(A_2, \dots, A_n)$ 和 $K_n(A_1, A_2, \dots, A_n)$ 互素, 而 (16) 右边的分数是在最低项中, 因此

$$u = K_n(A_1, A_2, \dots, A_n)d, \quad v = K_{n-1}(A_2, \dots, A_n)d \quad (17)$$

其中 $d = \gcd(u, v)$ 。

最坏情况 我们现在可应用这些观察来确定在最坏情况下欧几里得算法的特性, 或者换言之, 来给出除法步数的上界。当输入是连续的斐波那契数时出现最坏的情况:

定理 F 对于 $n \geq 1$, 设 u 和 v 是满足 $u > v > 0$ 的整数, 并且使得应用于 u 和 v 的欧几里得算法恰好需要 n 个除法步骤, 而且使得 u 是满足这些条件的尽可能小的数。则 $u = F_{n+2}$, $v = F_{n+1}$ 。

证明 由 (17), 我们必然有 $u = K_n(A_1, A_2, \dots, A_n)d$, 其中 A_1, A_2, \dots, A_n, d 是正整数, 而且 $A_i \geq 2$ 。由于 K_n 是具有非负系数的一个多项式, 包含着所有的变量, 故仅当 $A_1 = 1, \dots, A_{n-1} = 1, A_n = 2, d = 1$ 时才达到极小值。在 (17) 中置这些值, 就得到所求的结果。 ■

这一定理在历史上被说成是斐波那契序列的头一个实际应用; 此后就逐步发现了斐波那契数在算法和算法研究中的许多其它应用。这个结果实质上是由 T. F. de Lagny 给出的 [*Mém. Acad. Sci. Paris* 11 (1733), 363~364], 他把好多个连分数造了表并且发现对于一个给定的长度的连分数来说, 斐波那契数给出最小的分子和分母。然而, 他没有明确提到 \gcd 计算; 斐波那契数和欧几里得算法之间的联系是由 É. Lèger 首先指出的 [*Correspondance Math. et Physique* 9 (1837), 483~485]。

不久之后, P. J. E. Finck [*Traité Élémentaire d'Arithmétique* (Strasbourg: 1841), 44] 通过其它方法证明, 当 $u > v > 0$ 时, $\gcd(u, v)$ 至多花费 $2 \lg v + 1$ 步; 而且 G. Lamé [*Comptes Rendus Acad. Sci. Paris* 19 (1844), 867~870] 把这改进成 $5 \lceil \log_{10}(v+1) \rceil$ 。关于算法分析先驱研究的完整细节出现在 J. O. Shallit 的一篇有趣评论中 [*Historia Mathematica* 21 (1994), 401~419]。然而, 对于最坏情况的一个更精确的估计, 是定理 F 的一个直接推论:

推论 L 如果 $0 \leq v < N$, 则当算法 4.5.2A 应用于 u, v 时, 所需要的除法步骤数至多是 $\lfloor \log_\phi(3 - \phi)N \rfloor$ 。

证明 在步骤 A1 之后我们有 $v > u \bmod v$ 。因此由定理 F, 当 $v = F_{n+1}$ 和 $u \bmod v = F_n$ 时出现极大步骤数 n 。由于 $F_{n+1} < N$, 我们有 $\phi^{n+1}/\sqrt{5} < N$ (见等式 (1.2.8-(15))), 所以 $\phi^n < (\sqrt{5}/\phi)N = (3 - \phi)N$ 。 ■

$\log_\phi(3 - \phi)N$ 近似于 $2.078 \ln N + .6723 \approx 4.785 + \log_{10} N + .6723$ 。关于定理

F 的推广见习题 31, 36 和 38。

一个近似的模型 既然已经知道了可能出现的极大除法步骤数, 我们即可求出平均数。设 $T(m, n)$ 是当欧几里得算法的输入为 $u = m, v = n$ 时所进行的除法步骤数。于是

$$T(m, 0) = 0; \quad T(m, n) = 1 + T(n, m \bmod n), \quad \text{如果 } n \geq 1 \quad (18)$$

设 T_n 是当 $v = n$ 和当 u 随机选择时的平均除法步骤数; 由于在第一次除法步骤之后, 只有 $u \bmod v$ 的值影响这一算法, 故我们有

$$T_n = \frac{1}{n} \sum_{0 \leq k < n} T(k, n) \quad (19)$$

例如, $T(0, 5) = 1, T(1, 5) = 2, T(2, 5) = 3, T(3, 5) = 4, T(4, 5) = 3$, 所以

$$T_5 = \frac{1}{5}(1 + 2 + 3 + 4 + 3) = 2\frac{3}{5}$$

我们的目的是估计当 n 很大时的 T_n 。一个想法是试验由 R. W. Floyd 提出的一个近似方法: 我们可以假定, 对于 $0 \leq k < n$, n 的值模 k 实质上是“随机”的, 于是我们置

$$T_n \approx 1 + \frac{1}{n}(T_0 + T_1 + \cdots + T_{n-1})$$

因此 $T_n \approx S_n$, 其中序列 $\langle S_n \rangle$ 是递推关系

$$S_0 = 0, \quad S_n = 1 + \frac{1}{n}(S_0 + S_1 + \cdots + S_{n-1}), \quad n \geq 1 \quad (20)$$

的解。注意到

$$\begin{aligned} S_{n+1} &= 1 + \frac{1}{n+1}(S_0 + S_1 + \cdots + S_{n-1} + S_n) = \\ &= 1 + \frac{1}{n+1}(n(S_n - 1) + S_n) = S_n + \frac{1}{n+1} \end{aligned}$$

这个递推式容易求解; 因此 S_n 是 $1 + \frac{1}{2} + \cdots + \frac{1}{n} = H_n$, 即是一个调和数。现在近似 $T_n \approx S_n$ 提示, 我们可以有 $T_n \approx \ln n + O(1)$ 。

然而, 把这个近似值同 T_n 的真值表作比较表明 $\ln n$ 太大了; T_n 并不这样快地增长。我们关于 n 模 k 是随机的尝试性假定因此太悲观了。确实, 更仔细的观察表明在 $1 \leq k \leq n$ 的范围内, $n \bmod k$ 的平均值小于 $\frac{1}{2}k$ 的平均值:

$$\begin{aligned} \frac{1}{n} \sum_{1 \leq k \leq n} (n \bmod k) &= \frac{1}{n} \sum_{1 \leq k, q \leq n} (n - qk) [\lfloor n/(q+1) \rfloor < k \leq \lfloor n/q \rfloor] = \\ &= n - \frac{1}{n} \sum_{1 \leq q \leq n} q \left(\binom{\lfloor n/q \rfloor + 1}{2} - \binom{\lfloor n/(q+1) \rfloor + 1}{2} \right) = \\ &= n - \frac{1}{n} \sum_{1 \leq q \leq n} \left(\binom{\lfloor n/q \rfloor + 1}{2} \right) = \\ &= \left(1 - \frac{\pi^2}{12} \right) n + O(\log n) \end{aligned} \quad (21)$$

(参考习题 4.5.2-10(c))。这大约只是 $.1775n$, 而不是 $.25n$; 所以 $n \bmod k$ 的值势必小于 Floyd 模型的预测, 因此欧几里得算法比起我们预期的更快。

一个连续模型 当 $v = N$ 时, 欧几里得算法的特性实质上决定于当 $X = 0/N, 1/N, \dots, (N-1)/N$ 时, 正则连分数过程的特性。因此我们要研究当 N 很大和当 X 是在 $[0, 1)$ 中均匀分布的随机实数时正则连分数的特性。给定 $X = X_0$ 的一个均匀分布, 考虑分布函数

$$F_n(x) = \Pr(X_n \leq x), \quad \text{对于 } 0 \leq x \leq 1 \quad (22)$$

由正则连分数的定义, 我们有 $F_0(x) = x$, 而且

$$\begin{aligned} F_{n+1}(x) &= \sum_{k \geq 1} \Pr(k \leq 1/X_n \leq k+x) = \\ &= \sum_{k \geq 1} \Pr(1/(k+x) \leq X_n \leq 1/k) = \\ &= \sum_{k \geq 1} (F_n(1/k) - F_n(1/(k+x))) \end{aligned} \quad (23)$$

如果由这些公式定义的分佈 $F_0(x), F_1(x), \dots$ 趋向于一个极限分佈 $F_\infty(x) = F(x)$, 则我们将有

$$F(x) = \sum_{k \geq 1} (F(1/k) - F(1/(k+x))) \quad (24)$$

(一个类似的关系式 4.5.2-(36) 在我们关于二进制 gcd 算法的研究中出现过。) 对任何底 $b > 1$, 满足 (24) 的一个函数是 $F(x) = \log_b(1+x)$; 见习题 19。进一步的条件 $F(1) = 1$ 意味着我们应取 $b = 2$ 。因此, 有理由猜测 $F(x) = \lg(1+x)$, 并且 $F_n(x)$ 趋于这一特性。

例如, 我们可以猜测, $F\left(\frac{1}{2}\right) = \lg\left(\frac{3}{2}\right) \approx 0.58496$; 让我们看看对于小的 n , $F_n\left(\frac{1}{2}\right)$ 如何接近于这个值。我们有 $F_0\left(\frac{1}{2}\right) = 0.50000$, 而且

$$\begin{aligned} F_1(x) &= \sum_{k \geq 1} \left(\frac{1}{k} - \frac{1}{k+x} \right) = H_x \\ F_1\left(\frac{1}{2}\right) &= H_{1/2} = 2 - 2 \ln 2 \approx 0.61371 \\ F_2\left(\frac{1}{2}\right) &= H_{2/2} - H_{2/3} + H_{2/4} - H_{2/5} + H_{2/6} - H_{2/7} + \dots \end{aligned} \quad (25)$$

(参见附录 A 的表 3。) 幂级数展开

$$H_x = \zeta(2)x - \zeta(3)x^2 + \zeta(4)x^3 - \zeta(5)x^5 + \dots$$

使得计算下列数值成为可能:

$$F_2\left(\frac{1}{2}\right) = 0.57655\ 93276\ 99914\ 08418\ 82618\ 72122\ 27055\ 92452 - \quad (26)$$

我们更接近于 0.58496; 但是并非立即就清楚, 对于 $n = 3$ 如何得到 $F_n\left(\frac{1}{2}\right)$ 的一个

好的估计,而对真正大的 n 值则更不清楚。

C. F. Gauss(高斯)最先研究了分布 $F_n(x)$ 。1799年2月5日,他第一次考虑了这个问题。他于1800年在笔记本上列出了各种递推关系并且给出了一份简单的值表,包括(不精确的)近似 $F_2\left(\frac{1}{2}\right) \approx 0.5748$ 。在进行了这些计算之后,高斯写到:“它们变得如此复杂,以致看来没有什么希望了。”12年后,高斯写了一封信给拉普拉斯,信中把这作为他不能满意地解决的问题提出来。他写到:“通过非常简单的推理,我发现,对于无穷的 n , $F_n(x) = \log(1+x)/\log 2$ 。但是自那以后,我对于很大的,但不是无穷的 n 值,为确定 $F_n(x) - \log(1+x)/\log 2$ 所做的努力,都毫无结果。”他根本没有发表他的“非常简单的推理”,而且也不完全清楚他是否果真找到了一个严格的证明[见其 *Werke*, vol. 10¹, 552~556]。自那以后过了一百多年,才由 R. O. Kuz'min 发表了一个证明[*Atti del Congresso Internazionale dei Matematici* 6 (Bologna, 1928), 83~89], 他证明,对于某个正常数 Λ ,

$$F_n(x) = \lg(1+x) - O(e^{-\Lambda/n})$$

不久, Paul Lévy 把误差项改进成 $O(e^{-\Lambda n})$ [*Bull. Soc. Math. de France* 57 (1929), 178~194]*; 但是高斯的问题,即求 $F_n(x) - \lg(1+x)$ 的渐近特性问题,一直到1974年当 Eduard Wirsing 发表了关于这一情况的漂亮的分析时,才真正地解决了[*Acta Arithmetica* 24 (1974), 507~528]。这里我们将研究 Wirsing 方法的最简单方面,因为他的方法是线性算子的一个有教益的用法。

如果 G 是对于 $0 \leq x \leq 1$ 定义的 x 的任何函数, 命 SG 是由

$$SG(x) = \sum_{k \geq 1} \left(G\left(\frac{1}{k}\right) - G\left(\frac{1}{k+x}\right) \right) \quad (27)$$

定义的函数。于是, S 是把一个函数改变成另一个函数的算子。特别是, 由(23)我们有 $F_{n+1}(x) = SF_n(x)$, 因此

$$F_n = S^n F_0 \quad (28)$$

(在这个讨论中, F_n 代表一个分布函数, 而不是一个斐波那契数。) 注意, S 是一个“线性算子”; 即对于所有常数 c , $S(cG) = c(SG)$, 而且 $S(G_1 + G_2) = SG_1 + SG_2$ 。

现在如果 G 有一个有界的一阶导数, 我们可以逐项地求(27)的微分以证明

$$(SG)'(x) = \sum_{k \geq 1} \frac{1}{(k+x)^2} G'\left(\frac{1}{k+x}\right) \quad (29)$$

因此 SG 也有一个有界的一阶导数(当导数的级数一致收敛时, 一个收敛级数的逐项微分是正当的; 参考 K. Knopp, *Theory and Application of Infinite Series* (Glasgow: Blackie, 1951), § 47)。

命 $H = SG$, 并设 $g(x) = (1+x)G'(x)$, $h(x) = (1+x)H'(x)$ 。由此得出

* 关于 Lévy 的有趣证明的说明见本书第1版。——原注

$$h(x) = \sum_{k \geq 1} \frac{1+x}{(k+x)^2} \left(1 + \frac{1}{k+x}\right)^{-1} g\left(\frac{1}{k+x}\right) = \\ \sum_{k \geq 1} \left(\frac{k}{k+1+x} - \frac{k-1}{k+x}\right) g\left(\frac{1}{k+x}\right)$$

换句话说, $h = Tg$, 其中 T 是由

$$Tg(x) = \sum_{k \geq 1} \left(\frac{k}{k+1+x} - \frac{k-1}{k+x}\right) g\left(\frac{1}{k+x}\right) \quad (30)$$

定义的。

继续下去我们看到, 如果 g 有一个有界的一阶导数, 我们可以逐项微分以证明 Tg 也有:

$$(Tg)'(x) = - \sum_{k \geq 1} \left(\left(\frac{k}{(k+1+x)^2} - \frac{k-1}{(k+x)^2} \right) g\left(\frac{1}{k+x}\right) + \right. \\ \left. \left(\frac{k}{k+1+x} - \frac{k-1}{k+x} \right) \frac{1}{(k+x)^2} g'\left(\frac{1}{k+x}\right) \right) = \\ = - \sum_{k \geq 1} \left(\frac{k}{(k+1+x)^2} \left(g\left(\frac{1}{k+x}\right) - g\left(\frac{1}{k+1+x}\right) \right) + \right. \\ \left. \frac{1+x}{(k+x)^3(k+1+x)} g'\left(\frac{1}{k+x}\right) \right)$$

结果就有第三个线性算子 U , 使得 $(Tg)' = -U(g')$, 即

$$U\varphi(x) = \sum_{k \geq 1} \left(\frac{k}{(k+1+x)^2} \int_{1/(k+1+x)}^{1/(k+x)} \varphi(t) dt + \right. \\ \left. \frac{1+x}{(k+x)^3(k+1+x)} \varphi\left(\frac{1}{k+x}\right) \right) \quad (31)$$

所有这一切同我们的问题有什么关系? 好, 如果我们置

$$F_n(x) = \lg(1+x) + R_n \lg(1+x) \quad (32)$$

$$f_n(x) = (1+x)F_n'(x) = \frac{1}{\ln 2}(1 + R_n'(\lg(1+x))) \quad (33)$$

我们有

$$f_n'(x) = R_n''(\lg(1+x))/((\ln 2)^2(1+x)) \quad (34)$$

在这些变换之后, $\lg(1+x)$ 项的作用就消失了。而且因为 $F_n = S^n F_0$, 我们有 $f_n = T^n f_0$ 且 $f_n' = (-1)^n U^n f_0'$ 。对于 n 用归纳法, 可证 F_n 与 f_n 都有有界导数。于是 (34) 变成

$$(-1)^n R_n''(\lg(1+x)) = (1+x)(\ln 2)^2 U^n f_0'(x) \quad (35)$$

现在 $F_0(x) = x$, $f_0(x) = 1+x$, 且 $f_0'(x)$ 是常数函数 1。下面我们来证明算子 U^n 使常数函数成为一个具有非常小的值的函数, 因此对于 $0 \leq x \leq 1$, $|R_n''(x)|$ 必定非常小。最后, 我们可以证明 $R_n(x)$ 本身很小以使这个论证确定无疑: 由于我们有 $R_n(0) = R_n(1) = 1$, 由熟知的内插公式 (以 $x_0 = 0, x_1 = x, x_2 = 1$ 参考习题 4.6.4-15)

得出,对于某个函数 $\xi_n(x)$,

$$R_n(x) = -\frac{x(1-x)}{2} R_n''(\xi_n(x)) \quad (36)$$

其中,当 $0 \leq x \leq 1$ 时, $0 \leq \xi_n(x) \leq 1$ 。

现在一切取决于我们是否有能力证明 U^n 产生小的函数值,其中 U 是(31)中定义的线性算子。注意到对所有 x ,若 $\varphi(x) \geq 0$,则对所有 x , $U\varphi(x) \geq 0$ 。在这个意义下, U 是一个正算子。由此得出, U 是保序的:如果对所有 x , $\varphi_1(x) \leq \varphi_2(x)$,则对所有 x 我们有 $U\varphi_1(x) \leq U\varphi_2(x)$ 。

剖析这一性质的一个办法是求一个这样的函数 φ ,使得我们能精确地计算 $U\varphi$,并且使用这一函数的常数倍来界限我们真正感兴趣的那些函数。首先,我们找这样的函数 g ,使得 Tg 是容易计算的。如果我们考虑对于所有 $x \geq 0$,而不是仅仅对 $[0, 1]$ 定义的函数,注意到当 G 连续时,

$$\begin{aligned} SG(x+1) - SG(x) &= G\left(\frac{1}{1+x}\right) - \lim_{k \rightarrow \infty} G\left(\frac{1}{k+x}\right) = \\ &= G\left(\frac{1}{1+x}\right) - G(0) \end{aligned} \quad (37)$$

因此容易从(27)去掉求和。由于 $T((1+x)G') = (1+x)(SG)'$,由此得出(见习题20)

$$\frac{Tg(x)}{1+x} - \frac{Tg(1+x)}{2+x} = \left(\frac{1}{1+x} - \frac{1}{2+x}\right) g\left(\frac{1}{1+x}\right) \quad (38)$$

如果置 $Tg(x) = 1/(1+x)$,我们会发现 $g(x)$ 对应的值是 $1+x - 1/(1+x)$ 。设 $\varphi(x) = g'(x) = 1 + 1/(1+x)^2$,则 $U\varphi(x) = -(Tg)'(x) = 1/(1+x)^2$;这就是我们寻找的函数 φ 。

对于 φ 的这个选择,对于 $0 \leq x \leq 1$,我们有 $2 \leq \varphi(x)/U\varphi(x) = (1+x)^2 + 1 \leq 5$,因此

$$\frac{1}{5} \varphi \leq U\varphi \leq \frac{1}{2} \varphi$$

由 U 和 φ 的正定性,我们可以再次把 U 应用于这个不等式,得到 $\frac{1}{25} \varphi \leq \frac{1}{5} U\varphi \leq$

$U^2 \varphi \leq \frac{1}{2} U\varphi \leq \frac{1}{4} \varphi$;在应用 $n-1$ 次后,我们对于这个具体的 φ 有

$$5^{-n} \varphi \leq U^n \varphi \leq 2^{-n} \varphi \quad (39)$$

命 $\chi(x) = f_0'(x) = 1$ 是常数函数,则对 $0 \leq x \leq 1$,我们有 $\frac{5}{4} \chi \leq \varphi \leq 2\chi$,因此

$$\frac{5}{8} 5^{-n} \chi \leq \frac{1}{2} 5^{-n} \varphi \leq \frac{1}{2} U^n \varphi \leq U^n \chi \leq \frac{4}{5} U^n \varphi \leq \frac{4}{5} 2^{-n} \varphi \leq \frac{8}{5} 2^{-n} \chi$$

由(35)得出,对 $0 \leq x \leq 1$,

$$\frac{5}{8} (\ln 2)^2 5^{-n} \leq (-1)^n R_n''(x) \leq \frac{16}{5} (\ln 2)^2 2^{-n}$$

因此由(32)和(36),我们证明出下列结果:

定理 W 当 $n \rightarrow \infty$ 时,分布 $F_n(x)$ 等于 $\lg(1+x) + O(2^{-n})$ 。事实上,对于 $0 \leq x \leq 1$, $F_n(x) - \lg(1+x)$ 位于 $\frac{5}{16}(-1)^{n+1}5^{-n}(\ln(1+x))(\ln 2/(1+x))$ 和 $\frac{8}{5}(-1)^{n+2}2^{-n}(\ln(1+x))(\ln 2/(1+x))$ 之间。■

通过选择一个稍微不同的 φ ,我们可以得到更严密的界(见习题 21)。事实上,Wirsing在他的论文中走得更远,他证明

$$F_n(x) = \lg(1+x) + (-\lambda)^n \Psi(x) + O(x(1-x)(\lambda + 0.031)^n) \quad (40)$$

其中

$$\lambda = 0.30366\ 30028\ 98732\ 65859\ 74481\ 21901\ 55623\ 31109 = \\ //\ 3, 3, 2, 2, 3, 13, 1, 174, 1, 1, 1, 2, 2, 2, 1, 1, 1, 2, 2, 1, \dots // \quad (41)$$

是一个基本常数(显然同更熟知的常数无关),而且其中 Ψ 是一个有趣的函数,除由 -1 到 $-\infty$ 的负实轴外,它在整个复平面上是解析的。Wirsing 的函数满足 $\Psi(0) = \Psi(1) = 0$, $\Psi'(0) < 0$ 以及 $S\Psi = -\lambda\Psi$;因此由(37),它满足恒等式

$$\Psi(z) - \Psi(z+1) = \frac{1}{\lambda} \Psi\left(\frac{1}{1+z}\right) \quad (42)$$

而且 Wirsing 演示了,当 $N \rightarrow \infty$ 时

$$\Psi\left(-\frac{u}{v} + \frac{1}{N}\right) = c\lambda^{-n} \log N + O(1) \quad (43)$$

其中 c 是一个常数,而且当把欧几里得算法应用于整数 $u > v > 0$ 时, $n = T(u, v)$ 是迭代的次数。

过了一些年后,才由 K. I. Babenko [Doklady Akad. Nauk SSSR **238** (1978), 1021~1024] 找到了高斯问题的完全的解。他使用强有力的泛函分析技术证明对所有 $0 \leq x \leq 1, n \geq 1$,

$$F_n(x) = \lg(1+x) + \sum_{j \geq 2} \lambda_j^n \Psi_j(x) \quad (44)$$

这里 $|\lambda_2| > |\lambda_3| \geq |\lambda_4| \geq \dots$, 而且每个 $\Psi_j(z)$ 是除了割线 $[-\infty, -1]$ 之外的复平面上的解析函数。函数 Ψ_2 是 Wirsing 的 Ψ , $\lambda_2 = -\lambda$, 而 $\lambda_3 \approx 0.10088$, $\lambda_4 \approx -0.03550$, $\lambda_5 \approx 0.01284$, $\lambda_6 \approx 0.00472$, $\lambda_7 \approx 0.00175$ 。Babenko 也求出了特征值 λ_j 的进一步性质,特别是证明当 $j \rightarrow \infty$ 时,它们按指数速度变小,而且在(44)中对于 $j \geq k$ 的和以 $(\pi^2/6)|\lambda_k|^{n-1} \min(x, 1-x)$ 为界。[关于进一步的介绍见 Babenko 和 Yuriev, Doklady Akad. Nauk SSSR **240** (1978), 1273~1276; Mayer 和 Roepstorff, J. Statistical Physics **47** (1987), 149~171; **50** (1988), 331~344; D. Hensley, J. Number Theory **49** (1994), 142~182; Daudé, Flajolet 和 Vallée, Combinatorics, Probability and Computing **6** (1997), 397~433; Flajolet 和 Vallée, Theoretical Comp. Sci. **194** (1998), 1~34。](41)中 λ 的 40 位值是由 John Hershberger 计算的。

由连续到离散 我们现在已经推导出了当 x 是在区间 $[0, 1]$ 上均匀分布的一个

实数时,关于连分数的概率分布的一些结果。但是一个实数是有理的概率为 0——几乎所有数都是无理的——所以这些结果不能直接应用于欧几里得算法。在我们可以把定理 W 应用于我们的问题之前,必须克服某些技术性的困难。考虑以初等测度论为基础的下列观察:

引理 M 设 $I_1, I_2, \dots, J_1, J_2, \dots$ 是包含在区间 $[0, 1]$ 中的互不相交的区间, 并设

$$\mathcal{I} = \bigcup_{k \geq 1} I_k, \quad \mathcal{J} = \bigcup_{k \geq 1} J_k, \quad \mathcal{K} = [0, 1] \setminus (\mathcal{I} \cup \mathcal{J})$$

假设 \mathcal{K} 的测度为 0, 又设 P_n 是集合 $\{0/n, 1/n, \dots, (n-1)/n\}$, 则

$$\lim_{n \rightarrow \infty} \frac{|\mathcal{I} \cap P_n|}{n} = \mu(\mathcal{I}) \quad (45)$$

这里 $\mu(\mathcal{I})$ 是 \mathcal{I} 的勒贝格测度, 即 $\sum_{k \geq 1} \text{length}(I_k)$, 而且 $|\mathcal{I} \cap P_n|$ 表示在集合 $\mathcal{I} \cap P_n$ 中的元素个数。

证明 设 $\mathcal{I}_N = \bigcup_{1 \leq k \leq N} I_k$ 及 $\mathcal{J}_N = \bigcup_{1 \leq k \leq N} J_k$ 。给定 $\epsilon > 0$, 找一个足够大的 N 使得 $\mu(\mathcal{I}_N) + \mu(\mathcal{J}_N) \geq 1 - \epsilon$, 并设

$$\mathcal{K}_N = \mathcal{K} \cup \bigcup_{k > N} I_k \cup \bigcup_{k > N} J_k$$

如果 I 是具有 (a, b) 或 $[a, b)$ 或 $(a, b]$ 或 $[a, b]$ 中任何一个形式的区间, 显然 $\mu(I) = b - a$, 且

$$n\mu(I) - 1 \leq |I \cap P_n| \leq n\mu(I) + 1$$

现在设 $r_n = |\mathcal{I}_N \cap P_n|$, $s_n = |\mathcal{J}_N \cap P_n|$, $t_n = |\mathcal{K}_N \cap P_n|$; 我们有

$$r_n + s_n + t_n = n$$

$$n\mu(\mathcal{I}_N) - N \leq r_n \leq n\mu(\mathcal{I}_N) + N$$

$$n\mu(\mathcal{J}_N) - N \leq s_n \leq n\mu(\mathcal{J}_N) + N$$

因此,

$$\mu(\mathcal{I}) - \frac{N}{n} - \epsilon \leq \mu(\mathcal{I}_N) - \frac{N}{n} \leq \frac{r_n}{n} \leq \frac{r_n + t_n}{n} =$$

$$1 - \frac{s_n}{n} \leq 1 - \mu(\mathcal{J}_N) + \frac{N}{n} \leq \mu(\mathcal{I}) + \frac{N}{n} + \epsilon.$$

这对所有的 n 和所有的 ϵ 成立; 因此 $\lim_{n \rightarrow \infty} r_n/n = \mu(\mathcal{I})$ 。 ■

习题 25 表明, 为证明 (45) 需要某些稍加限制的假设。在这个意义下, 引理 M 是不平凡的。

部分商的分布 现在我们可以把定理 W 和引理 M 合在一起推导关于欧几里得算法某些实实在在的事实。

定理 E 设 n 和 k 是正整数, 并设 $p_k(a, n)$ 是当 $v = n$ 和 u 随机选择时, 欧几里得算法中第 $(k+1)$ 个商 A_{k+1} 等于 a 的概率, 则

$$\lim_{n \rightarrow \infty} p_k(a, n) = F_k\left(\frac{1}{a}\right) - F_k\left(\frac{1}{a+1}\right)$$

其中, $F_k(x)$ 是分布函数(22)。

证明 $[0, 1)$ 中使得 $A_{k+1} = a$ 的所有 X 的集合 \mathcal{J} 是一些不相交区间的并, 因此使得 $A_{k+1} \neq a$ 的所有 X 的集合 \mathcal{J} 也是这样。因此, 对于使得 A_{k+1} 没有定义的所有 X 的集合 \mathcal{K} 可应用引理 M。而且, $F_k(1/a) - F_k(1/(a+1))$ 是 $1/(a+1) < X_k \leq 1/a$ 的概率 $\mu(\mathcal{J})$, 即 $A_{k+1} = a$ 的概率。 ■

作为定理 E 和 W 的一个推论, 我们可以说, 一个等于 a 的商出现的近似概率为

$$\lg(1 + 1/a) - \lg(1 + 1/(a+1)) = \lg((a+1)^2/((a+1)^2 - 1))$$

于是

$$\text{商 } 1 \text{ 出现的概率是 } \lg\left(\frac{4}{3}\right) = 41.504\%;$$

$$\text{商 } 2 \text{ 出现的概率是 } \lg\left(\frac{9}{8}\right) = 16.993\%;$$

$$\text{商 } 3 \text{ 出现的概率是 } \lg\left(\frac{16}{15}\right) = 9.311\%;$$

$$\text{商 } 4 \text{ 出现的概率是 } \lg\left(\frac{25}{24}\right) = 5.889\%。$$

实际上, 如果欧几里得算法产生商 A_1, A_2, \dots, A_t , 则仅当 k 相对于 t 来说相当小时, 上边一些证明才能保证 A_k 的这一特性; 值 A_{t-1}, A_{t-2}, \dots 不为这个证明所包括。但是事实上我们可以证明最后的商 A_{t-1}, A_{t-2}, \dots 的分布实质上 and 最初的是一样的。

例如, 考虑其分母为 29 的所有真分数集合的正则连分数展开:

$$\begin{array}{llll} \frac{1}{29} = // 29 // & \frac{8}{29} = // 3, 1, 1, 1, 2 // & \frac{15}{29} = // 1, 1, 14 // & \frac{22}{29} = // 1, 3, 7 // \\ \frac{2}{29} = // 14, 2 // & \frac{9}{29} = // 3, 4, 2 // & \frac{16}{29} = // 1, 1, 4, 3 // & \frac{23}{29} = // 1, 3, 1, 5 // \\ \frac{3}{29} = // 9, 1, 2 // & \frac{10}{29} = // 2, 1, 9 // & \frac{17}{29} = // 1, 1, 2, 2, 2 // & \frac{24}{29} = // 1, 4, 1, 4 // \\ \frac{4}{29} = // 7, 4 // & \frac{11}{29} = // 2, 1, 1, 1, 3, // & \frac{18}{29} = // 1, 1, 1, 1, 1, 3 // & \frac{25}{29} = // 1, 6, 4 // \\ \frac{5}{29} = // 5, 1, 4 // & \frac{12}{29} = // 2, 2, 2, 2 // & \frac{19}{29} = // 1, 1, 1, 9 // & \frac{26}{29} = // 1, 8, 1, 2 // \\ \frac{6}{29} = // 4, 1, 5 // & \frac{13}{29} = // 2, 4, 3 // & \frac{20}{29} = // 1, 2, 4, 2 // & \frac{27}{29} = // 1, 13, 2 // \\ \frac{7}{29} = // 4, 7 // & \frac{14}{29} = // 2, 14 // & \frac{21}{29} = // 1, 2, 1, 1, 1, 2 // & \frac{28}{29} = // 1, 28 // \end{array}$$

通过这个表可以得到以下事实:

a) 如同早先提出过的, 最后的商总是 2 或更多。而且, 我们有明显的恒等式

$$// x_1, \dots, x_{n-1}, x_n + 1 // = // x_1, \dots, x_{n-1}, x_n, 1 // \quad (46)$$

而这表明其最后一个商为 1 的部分分式是怎样同正则连分数相关的。

b) 右边一列的值和左边那些列的值有一个简单的关系; 在往下读之前读者能发现这个对应关系吗? 有关的恒等式是:

$$1 // x_1, x_2, \dots, x_n // - // 1, x_1 - 1, x_2, \dots, x_n // \quad (47)$$

见习题 9。

c) 在头两列中, 左边和右边之间有对称性: 如果 $// A_1, A_2, \dots, A_i //$ 出现, 则 $// A_i, \dots, A_2, A_1 //$ 也出现。情况将总是这样(见习题 26)。

d) 如果我们考虑表中所有的商, 我们发现总共有 96 个商, 其中 $\frac{39}{96} = 40.6\%$ 等于 1, $\frac{21}{96} = 21.9\%$ 等于 2, $\frac{8}{96} = 8.3\%$ 等于 3; 这同上边列出的概率相当吻合。

除法步骤数 现在回到我们原来的问题并观察 T_n , 即当 $v = n$ 时除法步骤的平均数(见等式(19))。下面是 T_n 的某些值的抽样。

$n =$	95	96	97	98	99	100	101	102	103	104	105
$T_n =$	5.0	4.4	5.3	4.8	4.7	4.6	5.3	4.6	5.3	4.7	4.6
$n =$	996	997	998	999	1000	1001	...	9999	10000	10001	
$T_n =$	6.5	7.3	7.0	6.8	6.4	6.7	...	8.6	8.3	9.1	
$n =$	49998	49999	50000	50001	...	99999	100000	100001			
$T_n =$	9.8	10.6	9.7	10.0	...	10.7	10.3	11.0			

请注意一个奇特的特性: 当 n 为素数时 T_n 趋向于比它的邻值更高, 而当 n 有许多因子时, 则比邻值为低(在此表中, 97, 101, 103, 997 和 49999 是素数; $10001 = 73 \cdot 137$, $49998 = 2 \cdot 3 \cdot 13 \cdot 641$; $50001 = 3 \cdot 7 \cdot 2381$; $99999 = 3 \cdot 3 \cdot 41 \cdot 271$, $100001 = 11 \cdot 9091$)。不难理解这为什么出现: 如果 $\gcd(u, v) = d$, 则应用于 u 和 v 的欧几里得算法的特性实质上就好像它应用于 u/d 和 v/d 那样。因此, 当 $v = n$ 有若干个因子时, 有好几种 u 的选择, 使得 n 的行为好像是它是较小的那样。

由此我们考虑另一个量 τ_n , 它是当 $v = n$ 和当 u 与 n 互素时的平均除法次数。于是

$$\tau_n = \frac{1}{\varphi(n)} \sum_{\substack{0 \leq m < n \\ m \perp n}} T(m, n) \quad (48)$$

由此得出

$$T_n = \frac{1}{n} \sum_{d \mid n} \varphi(d) \tau_d \quad (49)$$

下面是对于和上边所考虑的相同的 n 值的 τ_n 的一张表。

$n =$	95	96	97	98	99	100	101	102	103	104	105
$\tau_n =$	5.4	5.3	5.3	5.6	5.2	5.2	5.4	5.3	5.4	5.3	5.6
$n =$	996	997	998	999	1000	1001	...	9999	10000	10001	
$\tau_n =$	7.2	7.3	7.3	7.3	7.3	7.4	...	9.21	9.21	9.22	

$$\begin{array}{cccccccc} n = & 49998 & 49999 & 50000 & 50001 & \cdots & 99999 & 100000 & 100001 \\ \tau_n = & 10.59 & 10.58 & 10.57 & 10.59 & \cdots & 11.170 & 11.172 & 11.172 \end{array}$$

显然, τ_n 的特性比 T_n 好得多, 因此它应当更易于分析。观察对于小的 n 的 τ_n 表揭示了某些奇怪的异常性, 例如 $\tau_{50} = \tau_{100}$ 及 $\tau_{60} = \tau_{120}$ 。但随着 n 增大, τ_n 的特性确实如上表所表示的那样十分规则, 而且它们没有显出同 n 的因子分解特性有任何重要的关系。如果对于上边给出的 τ_n 值, 在图纸上画作 $\ln n$ 的函数, 则将发现这些值非常靠近以下的直线

$$\tau_n \approx 0.843 \ln n + 1.47 \quad (50)$$

如果我们再进一步研究连分数过程, 就可以解释这一特性。如同在(15)中所表述的那样, 在欧几里得算法中我们有

$$\frac{V_0}{U_0} - \frac{V_1}{U_1} \cdots - \frac{V_{t-1}}{U_{t-1}} = \frac{V_t}{U_t}$$

因为 $U_{k+1} = V_k$; 因此如果 $U = U_0$ 和 $V = V_0$ 互素, 而且如果有 t 个除法步骤, 我们有

$$X_0 X_1 \cdots X_{t-1} = 1/U$$

置 $U = N$ 和 $V = m < N$, 我们求得

$$\ln X_0 + \ln X_1 + \cdots + \ln X_{t-1} = -\ln N \quad (51)$$

我们知道 X_0, X_1, X_2, \dots 的近似分布, 所以可以利用这个等式来估计

$$t = T(N, m) = T(m, N) - 1$$

回到定理 W 前边的公式, 当 X_0 是均匀分布于 $[0, 1)$ 的实数时, 我们求得 $\ln X_n$ 的平均值为

$$\int_0^1 \ln x \cdot F'_n(x) dx = \int_0^1 \ln x \cdot f_n(x) dx / (1+x) \quad (52)$$

其中 $f_n(x)$ 在(33)中定义。现在利用我们早先已经导出的事实(见习题 23)可知:

$$f_n(x) = \frac{1}{\ln 2} + O(2^{-n}) \quad (53)$$

因此 $\ln X_n$ 的平均值非常好地由

$$\begin{aligned} \frac{1}{\ln 2} \int_0^1 \frac{\ln x}{1+x} dx &= \frac{1}{\ln 2} \int_0^\infty \frac{u e^{-u}}{1+e^{-u}} du = \\ &= -\frac{1}{\ln 2} \sum_{k \geq 1} (-1)^{k-1} \int_0^\infty u e^{-ku} du = \\ &= -\frac{1}{\ln 2} \left(1 - \frac{1}{4} + \frac{1}{9} - \frac{1}{16} + \frac{1}{25} - \cdots \right) = \\ &= -\frac{1}{\ln 2} \left(1 + \frac{1}{4} + \frac{1}{9} + \cdots - 2 \left(\frac{1}{4} + \frac{1}{16} + \frac{1}{36} + \cdots \right) \right) = \\ &= -\frac{1}{2 \ln 2} \left(1 + \frac{1}{4} + \frac{1}{9} + \cdots \right) = \end{aligned}$$

$$- \pi^2 / (12 \ln 2)$$

所近似。因此由(51)我们期望有近似公式

$$- t \pi^2 / (12 \ln 2) \approx - \ln N$$

即 t 应当近似地等于 $((12 \ln 2) / \pi^2) \ln N$ 。这个常数 $(12 \ln 2) / \pi^2 = 0.842765913 \cdots$ 同早先所得到的经验公式(50)非常一致,所以我们有理由相信

$$\tau_n \approx \frac{12 \ln 2}{\pi^2} \ln n + 1.47 \quad (54)$$

当 $n \rightarrow \infty$ 时表示 τ_n 真正的渐近特性。

如果假定(54)是正确的,则我们得到公式

$$T_n \approx \frac{12 \ln 2}{\pi^2} \left(\ln n - \sum_{d|n} \frac{\Lambda(d)}{d} \right) + 1.47 \quad (55)$$

其中 $\Lambda(d)$ 是由规则

$$\Lambda(n) = \begin{cases} \ln p & \text{如果对于素数的 } p \text{ 和 } r \geq 1, n = p^r \\ 0 & \text{否则} \end{cases} \quad (56)$$

定义的 von Mangoldt 函数(见习题 27)。例如,

$$\begin{aligned} T_{100} &\approx \frac{12 \ln 2}{\pi^2} \left(\ln 100 - \frac{\ln 2}{2} - \frac{\ln 2}{4} - \frac{\ln 5}{5} - \frac{\ln 5}{25} \right) + 1.47 \approx \\ &(0.843)(4.605 - 0.347 - 0.173 - 0.322 - 0.064) + 1.47 \approx 4.59 \end{aligned}$$

T_{100} 的精确值是 4.56。

通过计算

$$\frac{1}{N} \sum_{n=1}^N T_n \quad (57)$$

我们也可以估计当 u 和 v 两者都均匀分布于 1 和 N 之间时除法步骤的平均次数。假定公式(55),习题 29 证明这个和有

$$\frac{12 \ln 2}{\pi^2} \ln N + O(1) \quad (58)$$

的形式,而且对用来推导等式 4.5.2-(65)的同样的数进行的经验计算表现了同公式

$$\frac{12 \ln 2}{\pi^2} \ln N + 0.06 \quad (59)$$

很好的一致。当然一般来说我们关于 T_n 和 τ_n 还未曾证明任何东西;至今我们仅仅考虑了为什么上边的公式应当成立的可能的原因。幸而,根据好几位数学家的仔细分析,我们现在有可能提供严格的证明。

在上边的一些公式中,前导系数 $(12 \ln 2) / \pi^2$ 在 John D. Dixon 和 Hans A. Heilbronn 独立研究下首先被确定了。Dixon [*J. Number Theory* 2 (1970), 414~422] 建立了 $F_n(x)$ 分布的理论,证明了在适当的意义下,各部分商基本上是彼此无关的,而且证明了对于所有正的 ϵ , 除在范围 $1 \leq m < n \leq N$ 中的 m 和 n 的 $\exp(-c(\epsilon) \cdot (\log N)^{d/2}) N^2$ 个值外,我们有 $|T(m, n) - ((12 \ln 2) / \pi^2) \ln n| < (\ln n)^{(1/2)+\epsilon}$, 其中

$c(\epsilon) > 0$ 。Heilbronn 的方法十分不同,他完全利用整数而不是连续变量。他的思想在习题 33 和 34 中以稍微不同的形式给出,它是以 τ_n 可以在某种方式下同表示 n 的方式数有关这样一个事实为基础的。而且,他的文章[*Number Theory and Analysis*, Paul Turán 编辑(New York:Plenum,1969),87~96]表明,我们上边已经讨论的各部分商 $1, 2, \dots$ 的分布实际上可应用于有给定分母的诸分式所属的部分商的集合;这是定理 E 的一个更明确的形式。若干年后, J. W. Porter 得到了更为深刻的结果[*Mathematika* 22 (1975), 20~28], 他确定

$$\tau_n = \frac{12 \ln 2}{\pi^2} \ln n + C + O(n^{-1/6+\epsilon}) \quad (60)$$

其中 $C \approx 1.46707\ 80794$ 是常数

$$\frac{6 \ln 2}{\pi^2} (3 \ln 2 + 4\gamma - 24\pi^{-2} \zeta'(2) - 2) - \frac{1}{2} \quad (61)$$

见 D. E. Knuth, *Computers and Math. with Applic.* 2 (1976), 137~139。于是猜测(50)就完全证明了。利用(60), Graham H. Norton [*J. Symbolic Computation* 10 (1990), 53~58]推广了习题 29 的计算, 证明在(59)中的经验常数 0.06 实际上是

$$\frac{6 \ln 2}{\pi^2} (3 \ln 2 + 4\gamma - 12\pi^{-2} \zeta'(2) - 3) - 1 \approx 0.0653514259 \dots \quad (62)$$

G. E. Collins 在 *SICOMP* 3 (1974), 1~10 中证明了使用算术运算的经典算法, 多精度整数的欧几里得算法的平均运行时间的阶为

$$(1 + \log(\max(u, v)/\gcd(u, v))) \log \min(u, v) \quad (63)$$

小结 我们已经发现了当输入 u 和 v 是连续的斐波那契数时出现欧几里得算法的最坏情况(定理 F); 当 $0 \leq v < N$ 时除法步骤数绝不超过 $\lceil 4.8 \log_{10} N - 0.32 \rceil$ 。我们已经确定各种部分商的值的频率, 例如, 证明了大约在 41% 的情况下除法步骤会遇到 $\lfloor u/v \rfloor = 1$ (定理 E)。最后, Heilbronn 和 Porter 的定理证明, 当 $v = n$ 时, 平均除法步骤数 T_n 近似地等于

$$((12 \ln 2)/\pi^2) \ln n \approx 1.9405 \log_{10} n$$

减去如等式(55)中所示的以 n 的因子为基础的一个校正项。

习 题

►1.[20] 由于在算法 4.5.2A 中商 $\lfloor u/v \rfloor$ 等于 1 的机会超过 40%, 所以在某些计算机上对于这种情况进行测试并在商为 1 时避免做除法可能是有利的。下面的欧几里得算法的 MIX 程序比程序 4.5.2A 更有效吗?

LDX	U	$rX \leftarrow u$	SRAX	5	$rAX \leftarrow rA$
JMP	2F		JL	2F	$u - v < v?$
1H STX	V	$v \leftarrow rX$	DIV	V	$rX \leftarrow rAX \bmod v$
SUB	V	$rA \leftarrow u - v$	2H LDA	V	$rA \leftarrow v$
CMPL	V		JXNZ	1B	如果 $rX = 0$, 则完成

2. [M21] 计算矩阵乘积

$$\begin{pmatrix} x_1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_2 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} x_n & 1 \\ 1 & 0 \end{pmatrix}$$

3. [M21] 行列式

$$\det \begin{pmatrix} x_1 & 1 & 0 & \cdots & 0 \\ -1 & x_2 & 1 & & 0 \\ 0 & -1 & x_3 & 1 & \vdots \\ \vdots & & & -1 & 1 \\ 0 & 0 & \cdots & -1 & x_n \end{pmatrix}$$

的值等于多少?

4. [M20] 证明等式(8)。

5. [FM25] 设 x_1, x_2, \cdots 是一实数序列, 它的每一个数都大于某一正实数 ϵ 。证明无穷连分数 $\|x_1, x_2, \cdots\| = \lim_{n \rightarrow \infty} \|x_1, \cdots, x_n\|$ 存在。并且证明如果我们仅仅假定对于所有 $j, x_j > 0$, $\|x_1, x_2, \cdots\|$ 未必存在。

6. [M23] 如果 B_1, B_2, \cdots 是正整数, 则无穷连分数 $\|B_1, B_2, \cdots\|$ 是 0 和 1 之间的一个无理数 X , 对于所有 $n \geq 1$, 它的连分数有 $A_n = B_n$ 。证明在这个意义下, 一个数的正则连分数展开是惟一的; 而且如果 B_1, B_2, \cdots, B_m 是正整数且 $B_m > 1$, 则对于 $1 \leq n \leq m$, 对于 $X = \|B_1, \cdots, B_m\|$ 的连分数有 $A_n = B_n$ 。

7. [M26] 求整数 $\{1, 2, \cdots, n\}$ 的所有排列 $p(1)p(2)\cdots p(n)$, 使得对所有 x_1, x_2, \cdots, x_n , $K_n(x_1, x_2, \cdots, x_n) = K_n(x_{p(1)}, x_{p(2)}, \cdots, x_{p(n)})$ 成立。

8. [M20] 试证: 在正则连分数过程中, 每当 X_n 有定义时,

$$-1/X_n = \|A_n, \cdots, A_1, -X\|$$

9. [M21] 证明连分数满足下列恒等式:

$$a) \|x_1, \cdots, x_n\| = \|x_1, \cdots, x_k + \|x_{k+1}, \cdots, x_n\|\|, \quad 1 \leq k \leq n;$$

$$b) \|0, x_1, x_2, \cdots, x_n\| = x_1 + \|x_2, \cdots, x_n\|, \quad n \geq 1;$$

$$c) \|x_1, \cdots, x_{k-1}, x_k, 0, x_{k+1}, x_{k+2}, \cdots, x_n\| = \|x_1, \cdots, x_{k-1}, x_k + \|x_{k+1}, x_{k+2}, \cdots, x_n\|\|, \quad 1 \leq k < n;$$

$$d) 1 - \|x_1, x_2, \cdots, x_n\| = \|1, x_1 - 1, x_2, \cdots, x_n\|, \quad n \geq 1.$$

10. [M28] 由习题 6 的结果, 每个无理实数 X 有形如

$$X = A_0 + \|A_1, A_2, A_3, \cdots\|$$

的惟一正则连分数表示, 其中 A_0 是一个整数, 并且 A_1, A_2, A_3, \cdots 是正整数。证明如果 X 有这个表示, 则对于适当的整数 B_0, B_1, \cdots, B_m , $1/X$ 的正则连分数是

$$1/X = B_0 + \|B_1, \cdots, B_m, A_5, A_6, \cdots\|$$

($A_0 < 0$ 的情况当然是最有趣的。)说明怎样借助于 A_0, A_1, A_2, A_3 和 A_4 来确定诸 B 。

11. [M30] (J.-A. Serret, 1850) 设 $X = A_0 + \|A_1, A_2, A_3, A_4, \cdots\|$ 和 $Y = B_0 + \|B_1, B_2, B_3, B_4, \cdots\|$ 是在习题 10 的意义下, 两个实数 X 和 Y 的正则连分数表示。当且仅当对于某些使 $|qt - rs| = 1$ 的整数 q, r, s, t , 我们有 $X = (qY + r)/(sY + t)$ 时, 有对于某个 m 和 n 及所有 $k \geq 0$,

$A_{m+k} = B_{n+k}$ 。证明在这个意义下,这些表示“最终一致”。(这个定理是下列简单结果对于连分数表示的类似物,即当且仅当对于某些整数 q, r 和 $s, X = (10^s Y + r)/10^s$ 时,在十进系统中 X 和 Y 的表示才最终一致。)

►12. [M30] 一个二次无理数是形如 $(\sqrt{D} - U)/V$ 的数,其中 D, U 和 V 是整数, $D > 0, V \neq 0$, 且 D 不是一个完全平方。不失一般性,我们可以假定 V 是 $D - U^2$ 的一个因子,因为否则这个数可以改写为 $(\sqrt{DV^2} - U|V|)/(V|V|)$ 。

a) 证明一个二次无理数 $X = (\sqrt{D} - U)/V$ 的正则连分数展开(在习题 10 的意义下)由下列公式得到:

$$V_0 = V, \quad A_0 = \lfloor X \rfloor, \quad U_0 = U + A_0 V;$$

$$V_{n+1} = (D - U_n^2)/V_n, \quad A_{n+1} = \lfloor (\sqrt{D} + U_n)/V_{n+1} \rfloor, \quad U_{n+1} = A_{n+1}V_{n+1} - U_n$$

b) 证明对所有 $n > N, 0 < U_n < \sqrt{D}, 0 < V_n < 2\sqrt{D}$, 其中 N 是依赖于 X 的某个整数;因此每个二次无理数的正则连分数表示最终是周期的。[提示:证明

$$(-\sqrt{D} - U)/V = A_0 + // A_1, \dots, A_n, -V_n/(\sqrt{D} + U_n) //$$

并使用等式(5)证明,当 n 很大时, $(\sqrt{D} + U_n)/V_n$ 是正的。]

c) 设 $p_n = K_{n+1}(A_0, A_1, \dots, A_n)$ 和 $q_n = K_n(A_1, \dots, A_n)$, 证明恒等式 $Vp_n^2 + 2Up_nq_n + ((U^2 - D)/V)q_n^2 = (-1)^{n+1}V_{n+1}$ 。

d) 证明当且仅当无理数 X 是一个二次无理数时 X 的正则连分数表示最终是周期的。(当且仅当实数 X 是有理数时, X 的十进展开最终是周期的,本题是上述事实连分数中的对应定理。)

13. [M40] (J. Lagrange, 1767) 设 $f(x) = a_n x^n + \dots + a_0, a_n > 0$, 是有整系数的多项式,且没有有理根,恰好有一个实数根 $\xi > 1$ 。利用下列算法(它实质上仅涉及加法),用一个计算机程序做实验,来求 ξ 的头一千个左右的部分商:

L1. 置 $A \leftarrow 1$ 。

L2. 对于 $k = 0, 1, \dots, n-1$ (在此顺序下) 以及 $j = n-1, \dots, k$ (在此顺序下), 置 $a_j \leftarrow a_{j+1} + a_j$ 。
(此步以 $g(x) = f(x+1)$ 代替 $f(x)$, 该 $g(x)$ 是其根比 $f(x)$ 的根小 1 的多项式。)

L3. 如果 $a_n + a_{n-1} + \dots + a_0 < 0$, 置 $A \leftarrow A + 1$ 并返回 L2。

L4. 输出 A (它是下一个部分商的值)。以 $(-a_0, -a_1, \dots, -a_n)$ 代替系数 $(a_n, a_{n-1}, \dots, a_0)$ 并返回 L1。(这一步以其根是 f 的那些根的倒数之多项式来代替 $f(x)$ 。)

例如,以 $f(x) = x^3 - 2$ 开始,这个算法将输出“1”(把 $f(x)$ 改变成 $x^3 - 3x^2 - 3x - 1$); 然后“3”(把 $f(x)$ 改变成 $10x^3 - 6x^2 - 6x - 1$) 等等。

14. [M22] (A. Hurwitz, 1891) 证明当给定 X 的部分商时,下列规则使我们有可能来求 $2X$ 的正则连分数展开:

$$\begin{aligned} 2 // 2a, b, c, \dots // &= // a, 2b + 2 // c, \dots // \\ 2 // 2a + 1, b, c, \dots // &= // a, 1, 1 + 2 // b - 1, c, \dots // \end{aligned}$$

给定(13)中 e 的展开,使用这一思想来求 $\frac{1}{2}e$ 的正则连分数。

15. [M31] (R. W. Gosper) 推广习题 14, 给定 x 的连分数 $x_0 + // x_1, x_2, \dots //$, 并给定满足 $ad \neq bc$ 的整数 a, b, c, d , 试设计一个算法,它计算 $(ax + b)/(cx + d)$ 的连分数 $X_0 + // X_1, X_2, \dots //$ 。使你的算法成为“联机共行子程序”,在输入每个 x_j 之前它输出尽可能多的 X_k 。说明当 $x = -1 + // 5, 1, 1, 1, 2, 1, 2 //$ 时,你的程序如何计算 $(97x + 39)/(-62x - 25)$ 。

16. [HM30] (L. Euler, 1731) 设 $f_0(z) = (e^z - e^{-z})/(e^z + e^{-z}) = \tanh z$, 并设 $f_{n+1}(z) = 1/f_n(z) - (2n+1)/z$. 证明对于所有 n , $f_n(z)$ 是在原点的一个邻域中复变量 z 的一个解析函数, 而且它满足微分方程 $f'_n(z) = 1 - f_n(z)^2 - 2nf_n(z)/z$. 试用这一事实证明

$$\tanh z = // z^{-1}, 3z^{-1}, 5z^{-1}, 7z^{-1}, \dots //$$

然后应用 Hurwitz 规则(习题 14)证明

$$e^{-1/n} = // 1, (2m+1)n-1, 1 //, \quad m \geq 0$$

(这个记法表示无穷连分数 $// 1, n-1, 1, 1, 3n-1, 1, 1, 5n-1, 1, \dots //$.) 并求当 $n > 0$ 是奇数时 $e^{-2/n}$ 的正则连分数展开.

► 17. [M23] (a) 证明 $// x_1, -x_2 // = // x_1 - 1, 1, x_2 - 1 //$. (b) 推广这个恒等式, 得到对于 $// x_1, -x_2, x_3, -x_4, x_5, -x_6, \dots, x_{2n-1}, -x_{2n} //$ 的一个公式, 其中当诸 x 为大正整数时, 所有部分商都是正整数. (c) 习题 16 的结果意味着 $\tan 1 = // 1, -3, 5, -7, \dots //$. 试求 $\tan 1$ 的正则连分数展开.

18. [M25] 证明 $// a_1, a_2, \dots, a_m, x_1, a_1, a_2, \dots, a_m, x_2, a_1, a_2, \dots, a_m, x_3, \dots // = // a_m, \dots, a_2, a_1, x_1, a_m, \dots, a_2, a_1, x_2, a_m, \dots, a_2, a_1, x_3, \dots //$ 不依赖于 x_1, x_2, x_3, \dots . 提示: 以 $K_m(a_1, a_2, \dots, a_m)$ 乘两个连分数.

19. [M20] 证明 $F(x) = \log_b(1+x)$ 满足等式(24).

20. [HM20] 由(37)推导(38).

21. [HM29] (E. Wirsing) 我们已经对于以 $Tg(x) = 1/(x+1)$ 对应于 g 的一个函数 φ 得到界(39). 证明当 $c > 0$ 是一个适当的常数时, 对应于 $Tg(x) = 1/(x+c)$ 的函数产生较好的界.

22. [HM46] (K. I. Babenko) 对于小的 $j \geq 3$ 和对于 $0 \leq x \leq 1$, 建立有效的手段来计算(44)中的量 λ_j 和 $\Psi_j(x)$ 的精确近似.

23. [HM23] 利用定理 W 的证明中的一些结果证明(53).

24. [M22] 在一个随机实数的正则连分数展开中, 一个部分商 A_n 的平均值等于多少?

25. [HM25] 找一个集合 $\mathcal{I} = I_1 \cup I_2 \cup I_3 \cup \dots \subseteq [0, 1]$ 的一个例子, 使得(45)对它不成立, 其中诸 I 是不相交区间.

26. [M23] 如果把数 $\{1/n, 2/n, \dots, \lfloor n/2 \rfloor/n\}$ 表达成为正则连分数, 则每当 $// A_1, A_2, \dots, A_i //$ 出现时 $// A_i, \dots, A_2, A_1 //$ 也出现. 证明在这种意义下, 结果的左右边是对称的.

27. [M21] 由(49)和(54)推导(55).

28. [M23] 证明下列涉及三个数论函数 $\varphi(n), \mu(n), \Lambda(n)$ 的恒等式:

$$a) \sum_{d|n} \mu(d) = \delta_{n1}, \quad b) \ln n = \sum_{d|n} \Lambda(d), \quad n = \sum_{d|n} \varphi(d)$$

$$c) \Lambda(n) = \sum_{d|n} \mu\left(\frac{n}{d}\right) \ln d, \quad \varphi(n) = \sum_{d|n} \mu\left(\frac{n}{d}\right) d$$

29. [M23] 假定 T_n 由(55)给出, 证明(57) 等于(58).

► 30. [HM32] 经常有人提议欧几里得算法的下列“贪婪”形式: 若 $u \bmod v > \frac{1}{2}v$, 则在进行除法步骤期间不是以 $u \bmod v$ 代替 v 而是以 $|(u \bmod v) - v|$ 代替 v . 于是, 如果 $u = 26$ 和 $v = 7$, 则我们有 $\gcd(26, 7) = \gcd(-2, 7) = \gcd(7, 2)$; 当从 26 减去 7 的倍数时, -2 是绝对值最小的余数. 把这个过程同欧几里得算法进行比较; 试估计这一方法平均节省的除法步骤数.

► 31. [M35] 找出习题 30 中提议的对欧几里得算法所做修改的最坏情况; 要求 n 个除法步骤的最小输入 $u > v > 0$ 是什么?

32. [20] (a) 长度为 n 的一个莫尔斯代码序列是 r 个点和 s 根连字线的一个串, 其中 $r+2s=n$ 。例如, 长度为 4 的莫尔斯代码序列是

...., ..—, .—., —., — —

注意连分数 $K_4(x_1, x_2, x_3, x_4)$ 是 $x_1x_2x_3x_4 + x_1x_2 + x_1x_4 + x_3x_4 + 1$, 找出并证明 $K_n(x_1, \dots, x_n)$ 与长度为 n 的莫尔斯代码序列之间的简单关系。(b) (L. Euler, *Novi Comm. Acad. Sci. Pet.* 9 (1762), 53 ~ 69) 证明

$$K_{m+n}(x_1, \dots, x_{m+n}) = K_m(x_1, \dots, x_m)K_n(x_{m+1}, \dots, x_{m+n}) + K_{m-1}(x_1, \dots, x_{m-1})K_{n-1}(x_{m+2}, \dots, x_{m+n})$$

33. [M32] 设 $h(n)$ 是 n 的形式为 $n = xx' + yy'$, $x > y > 0$, $x' > y' > 0$, $x \perp y$ (x, x', y, y' 为整数) 的表示的个数。

a) 证明如果把条件放松成为允许 $x' = y'$, 则表示的个数为 $h(n) + \lfloor (n-1)/2 \rfloor$ 。

b) 证明对于固定的 $y > 0$ 和 $0 < t \leq y$, 其中 $t \perp y$, 且对于满足 $x't \equiv n \pmod{y}$ 的在 $0 < x' < n/(y+t)$ 范围内的每个固定的 x' , 恰有一个 n 的表示满足 a) 的限制和条件 $x \equiv t \pmod{y}$ 。

c) 因此, $h(n) = \sum \lfloor (n/(y+t) - t'/y) \rfloor - \lfloor (n-1)/2 \rfloor$, 其中求和对所有使得 $t \perp y$, $t \leq y$, $t' \leq y$, $tt' \equiv n \pmod{y}$ 的正整数 y, t, t' 进行。

d) 证明 $h(n)$ 的每一个表示都可惟一地表示成形式

$$x = K_m(x_1, \dots, x_m), \quad y = K_{m-1}(x_1, \dots, x_{m-1}) \\ x' = K_k(x_{m+1}, \dots, x_{m+k})d, \quad y' = K_{k-1}(x_{m+2}, \dots, x_{m+k})d$$

其中 m, k, d 和 x_i 是正整数, $x_1 \geq 2, x_{m+k} \geq 2$, 且 d 是 n 的一个因子。习题 32 的恒等式现在意味着 $n/d = K_{m+k}(x_1, \dots, x_{m+k})$ 。反之, 使得 $x_1 \geq 2, x_{m+k} \geq 2$ 及 $K_{m+k}(x_1, \dots, x_{m+k})$ 整除 n 的任何给定的正整数序列 x_1, \dots, x_{m+k} , 都以这种方式对应于 $m+k-1$ 个 n 的表示。

e) 因此, $nT_n = \lfloor (5n-3)/2 \rfloor + 2h(n)$ 。

34. [HM40] (H. Heilbronn) 设 $h_d(n)$ 是如题 33 所述的使得 $xd < x'$ 的 n 的表示个数加上满足 $xd = x'$ 的表示个数的一半。

a) 设 $g(n)$ 是不要求 $x \perp y$ 的表示数。证明

$$h(n) = \sum_{d \mid n} \mu(d) g\left(\frac{n}{d}\right), \quad g(n) = 2 \sum_{d \mid n} h_d\left(\frac{n}{d}\right)$$

b) 推广习题 33b), 证明对于 $d \geq 1$, $h_d(n) = \sum (n/(y(y+t))) + O(n)$, 其中求和对所有使得 $t \perp y$, $0 < t \leq y < \sqrt{n/d}$ 的整数 y 和 t 进行。

c) 证明 $\sum_{y=1}^n (y/(y+t)) = \varphi(y) \ln 2 + O(\sigma_{-1}(y))$, 其中求和在范围 $0 < t \leq y$, $t \perp y$ 上进行, 而且 $\sigma_{-1}(y) = \sum_{d \mid y} (1/d)$ 。

d) 证明 $\sum_{y=1}^n \varphi(y)/y^2 = \sum_{d=1}^n \mu(d) H_{\lfloor n/d \rfloor} / d^2$ 。

e) 因此我们有渐近公式

$$T_n = ((12 \ln 2)/\pi^2)(\ln n - \sum_{d \mid n} \Lambda(d)/d) + O(\sigma_{-1}(n)^2)$$

35. [HM41] (A. C. Yao 和 D. E. Knuth) 证明对于 $1 \leq m < n$, 分数 m/n 的所有部分商之和等于 $2(\sum \lfloor x/y \rfloor + \lfloor n/2 \rfloor)$, 其中的求和是对满足习题 33a) 的条件的所有表示 $n = xx' + yy'$ 进行。证明 $\sum \lfloor x/y \rfloor = 3\pi^{-2}n(\ln n)^2 + O(n \log n (\log \log n)^2)$, 并把这应用于欧几里得算法的“古代”形式 (它只使用减法而不用除法)。

36. [M25] (G. H. Bradley) 如果始终使用欧几里得算法, 则使得由算法 4.5.2C 计算 $\gcd(u_1, \dots, u_n)$ 要求 N 个除法的最小的 u_n 值是多少? 假定 $N \geq n \geq 3$ 。

37. [M38] (T. S. Motzkin 和 E. G. Straus) 设 a_1, \dots, a_n 是正整数。证明当 $a_{p(1)} \geq a_{p(n)} \geq a_{p(2)} \geq a_{p(n-1)} \geq \dots$ 时, 出现对于 $\{1, 2, \dots, n\}$ 的所有排列 $p(1) \dots p(n)$ 的 $\max K_n(a_{p(1)}, \dots, a_{p(n)})$; 当 $a_{p(1)} \leq a_{p(n)} \leq a_{p(3)} \leq a_{p(n-2)} \leq a_{p(5)} \leq \dots \leq a_{p(6)} \leq a_{p(n-3)} \leq a_{p(4)} \leq a_{p(n-1)} \leq a_{p(2)}$ 时出现极小。

38. [M25] (J. Mikusiński) 设 $L(n) = \max_{m \geq 0} T(m, n)$ 。证 F 证明了 $L(n) \leq \log_2(\sqrt{5}n + 1) - 2$; 证明 $2L(n) \geq \log_2(\sqrt{5}n + 1) - 2$ 。

► 39. [M25] (R. W. Gosper) 如果一个垒球运动员击球的平均次数是 .334, 他已经经历过的击球状态的最小次数是多少? [对非垒球爱好者的说明: 击球平均次数 = (击球数)/(所处的击球状态次数)舍入到三位十进制数。]

► 40. [M28] (Stern-Brocot 树) 考虑一个无穷二叉树, 其中每个节点都以分数 $(p_l + p_r)/(q_l + q_r)$ 来标号, p_l/q_l 是节点最接近的左祖宗的标号, p_r/q_r 是节点最接近的右祖宗的标号。(左祖宗是在对称序下在节点之前的一个节点, 而右祖宗在这节点之后。关于对称序的定义见 2.3.1 小节。)如果节点没有左祖宗, 则 $p_l/q_l = 0/1$; 如果它没有右祖宗, 则 $p_r/q_r = 1/0$ 。于是根的标号为 $1/1$; 它的两个儿子的标号是 $1/2$ 和 $2/1$; 在第 2 级上, 自左至右的四个节点的标号是 $1/3, 2/3, 3/2$ 和 $3/1$; 在第 3 级上, 8 个节点的标号是 $1/4, 2/5, 3/5, 3/4, 4/3, 5/3, 5/2, 4/1$; 等等。

证明在每个标号 p/q 中 p 与 q 互素; 而且在对称序下以 p/q 标号的节点在 p'/q' 标号的节点之前当且仅当这些标号满足 $p/q < p'/q'$ 。试求出一个节点的标号的连分数和连接该节点的通路之间的联系, 由此说明每个正有理数恰作为这树中一个节点的标号出现。

41. [M40] (J. Shallit, 1979) 证明

$$\frac{1}{2^1} + \frac{1}{2^3} + \frac{1}{2^7} + \dots = \sum_{n \geq 1} \frac{1}{2^{2^n - 1}}$$

的正则连分数展开仅含 1 和 2 而且有相当简单的模式。证明当 l 是任何大于或等于 2 的整数时, Liouville 数 $\sum_{n \geq 1} l^{-n!}$ 的部分商也有一个正则模式 [由 J. Liouville 在 J. de Math. Pures et Appl. 16 (1851), 133~142 引进的这些数, 是首先被证明为超越数的显式定义的数, 前一个数和类似的常数最先是 A. J. Kempner 证明为超越数的, 见 Trans. Amer. Math. Soc 17 (1916), 476~482]。

42. [M30] (J. Lagrange, 1798) 设 X 有正则连分数展开 $// A_1, A_2, \dots //$, 并设 $q_n = K_n(A_1, \dots, A_n)$ 。设 $\|x\|$ 表示从 x 到最接近的整数的距离, 即 $\min_p |x - p|$ 。证明对于 $1 \leq q < q_n$, $\|qX\| \geq \|q_{n-1}X\|$ (于是所谓收敛数 $p_n/q_n = // A_1, \dots, A_n //$ 的分母 q_n 是“破记录”的整数, 它们使 $\|qX\|$ 达到新的低阶)。

43. [M30] (D. W. Matula) 证明当数 $x > 0$ 不是可表示的时, 对于开缝定点或开缝浮点数, “中间舍入”规则, 即等式 4.5.1-(1) 可以简单实现如下: 设 x 的正则连分数展开是 $a_0 + // a_1, a_2, \dots //$, 且设 $p_n = K_{n+1}(a_0, \dots, a_n)$, $q_n = K_n(a_1, \dots, a_n)$ 。则 $\text{round}(x) = (p_i/q_i)$, 其中 (p_i/q_i) 是可表示的, 但 (p_{i+1}/q_{i+1}) 不是 [提示: 见习题 40]。

44. [M25] 假设我们正在用中间舍入做开缝定点数的算术运算, 其中当且仅当 $|u| < M$ 和 $0 \leq u' < N$ 和 $u \perp u'$ 时, 分式 (u/u') 是可表示的。证明或否定: 假定 $u' < \sqrt{N}$ 且无溢出出现, 则对于所有可表示的 (u/u') 和 (v/v') 有恒等式 $((u/u') \oplus (v/v')) \ominus (v/v') = (u/u')$ 。

45. [M25] 证明当 $n \rightarrow \infty$ 时, 应用到两个 n 个二进位数的欧几里得算法 (算法 (4.5.2A) 要求 $O(n^2)$ 个时间单位 (对于算法 4.5.2B 同样的上限明显地成立))。

46. [M43] 如果使用计算最大公因子的另一个算法, 习题 45 中的上限 $O(n^2)$ 能否减少?

47. [M40] 当 x 是以高精度给出的一个实数时, 试编制一个计算机程序来找出尽可能多的 x 的部分商。使用你的程序来计算欧拉常数 γ 的头数千位部分商, 这可以如同 D. W. Sweeney 在

Math. Comp. **17** (1963), 170~178 中所说明的那样来计算。(如果 γ 是一个有理数, 你可以发现它的分子和分母, 由此解决一个有名的数学问题。按照正文中的理论, 当给定的数是随机时, 我们预测对于每个十进数字, 我们得到大约 0.97 个部分商。多精度并不必要; 请见算法 4.5.2L 以及 J. W. Wrench, Jr. 和 D. Shanks 的文章, *Math. Comp.* **20** (1966), 444~447。)

48. [M21] 令 $T_0 = (1, 0, u)$, $T_1 = (0, 1, v)$, \dots , $T_{n+1} = ((-1)^{n+1}v/d, (-1)^n u/d, 0)$ 是由算法 4.5.2X (扩充的欧几里得算法) 计算出来的向量序列, 并设 $//u_1, \dots, a_n //$ 是对于 v/u 的正则连分数, 对于 $1 \leq j \leq n$, 借助于涉及 a_1, \dots, a_n 的连式表达 T_j 。

49. [M33] 通过调整算法 4.5.2X 最后的迭代使得 a_n 任选地由两个部分商 $(a_n - 1, 1)$ 所代替, 我们可以假定迭代次数 n 有一个给定的奇偶性。继续上道题, 令 λ 和 μ 是任意正实数并令 $\theta = \sqrt{\lambda\mu v/d}$, 其中 $d = \gcd(u, v)$ 。证明, 如果 n 是偶数, 而且如果 $T_j = (x_j, y_j, z_j)$, 我们有

$$\min_{j=1}^{n+1} \left| \lambda x_j + \mu z_j - [j \text{ 偶}] \theta \right| \leq \theta。$$

► 50. [M25] 给定一个无理数 $\alpha \in (0, 1)$ 及实数 β 和 γ 且 $0 \leq \beta < \gamma < 1$, 设 $f(\alpha, \beta, \gamma)$ 是使得 $\beta \leq \alpha n \bmod 1 < \gamma$ 的最小非负整数 n 。(由于习题 3.5-22 的 Weyl 定理, 这样一个整数存在。)试设计计算 $f(\alpha, \beta, \gamma)$ 的一个算法。

► 51. [M30] (有理数构造) 在 $316 \cdot 28481 \equiv 41$ 的意义下, 数 28481 将等于 $41/316$ (modulo 199999)。人们怎样发现它? 给定整数 a 和 m 且 $m > a > 1$, 说明怎样求整数 x 和 y 使得 $ax \equiv y$ (modulo m), $x \perp y$, $0 < x \leq \sqrt{m/2}$ 且 $|y| \leq \sqrt{m/2}$, 或者确定没有这样的 x 和 y 存在。能否有多于一个的解?

4.5.4 分解素因子

在本书中我们已经遇到的好多个计算方法都以如下事实为根据, 即每个正整数 n 都可以用惟一的方式表达成

$$n = p_1 p_2 \cdots p_t, \quad p_1 \leq p_2 \leq \cdots \leq p_t \quad (1)$$

的形式, 其中每个 p_k 为素数 (当 $n=1$ 时, 等式对 $t=0$ 成立)。但不幸的是, 找 n 的这个素因子分解式, 或者确定 n 是否素数, 却不是一件简单的事情。现在已人人皆知, 对一个很大的数 n 进行因子分解, 比计算两个很大的数 m 和 n 的最大公因子要困难得多; 因此只要可能, 我们就应该避免对很大的数进行因子分解。但是已经发现了某些用来简化分解因子问题的巧妙方法, 我们现在就来研究其中的一些方法。[关于 1950 年以前因子分解的丰富历史已由 H. C. Williams 和 J. O. Shallit 编述, *Proc. Symp. Applied Math.* **48** (1993), 481~531。]

除法和因子 首先我们考虑最明显的分解因子算法: 如果 $n > 1$, 则我们可以用素数 $p=2, 3, 5, \dots$ 来逐个地除 n , 直到发现使得 $n \bmod p = 0$ 的最小的 p , 则这个 p 就是 n 的最小素因子, 而且同一过程可被应用于 $n \leftarrow n/p$, 以图用 p 和更高的素数来除 n 的这个新值。如果在任何阶段我们求得 $n \bmod p \neq 0$ 但 $\lfloor n/p \rfloor \leq p$, 则我们即可得出 n 是素数的结论; 因为如果 n 不是素数, 则由 (1), 我们必有 $n \geq p_1^2$, 但 $p_1 > p$ 意味着 $p_1^2 \geq (p+1)^2 > p(p+1) > p^2 + (n \bmod p) \geq \lfloor n/p \rfloor p + (n \bmod p) = n$ 。

这把我们引到下列过程:

算法 A (通过除法进行因子分解) 给定一个正整数 N , 这个算法找出等式(1)中那样的 N 的素因子 $p_1 \leq p_2 \leq \dots \leq p_t$ 。这个方法利用了一个辅助的“试验因子”序列

$$2 = d_0 < d_1 < d_2 < d_3 < \dots \quad (2)$$

它包括小于或等于 \sqrt{N} 的所有素数(而且它也可以包括不是素数的值, 如果这样做适宜的话)。这个 d 的序列至少必须包括一个使得 $d_k \geq \sqrt{N}$ 的值。

A1. [初始化] 置 $t \leftarrow 0, k \leftarrow 0, n \leftarrow N$ 。(在算法执行期间, 变量 t, k, n 通过条件“ $n = N/p_1 \dots p_t$ 和 n 没有小于 d_k 的素因子”相关。)

A2. [$n=1$?] 如果 $n=1$, 则算法终止。

A3. [除] 置 $q \leftarrow \lfloor n/d_k \rfloor, r \leftarrow n \bmod d_k$ (这里 q 和 r 是当 n 除以 d_k 时得到的商和余数)。

A4. [余数为 0?] 如果 $r \neq 0$, 则转向步骤 A6。

A5. [已找到因子] t 增加 1, 置 $p_t \leftarrow d_k$, 置 $n \leftarrow q$, 返回步骤 A2。

A6. [低的商?] 如果 $q > d_k$, 则 k 增加 1 并返回步骤 A3。

A7 [n 为素数] t 增加 1, 置 $p_t \leftarrow n$, 并终止此算法。 ▮

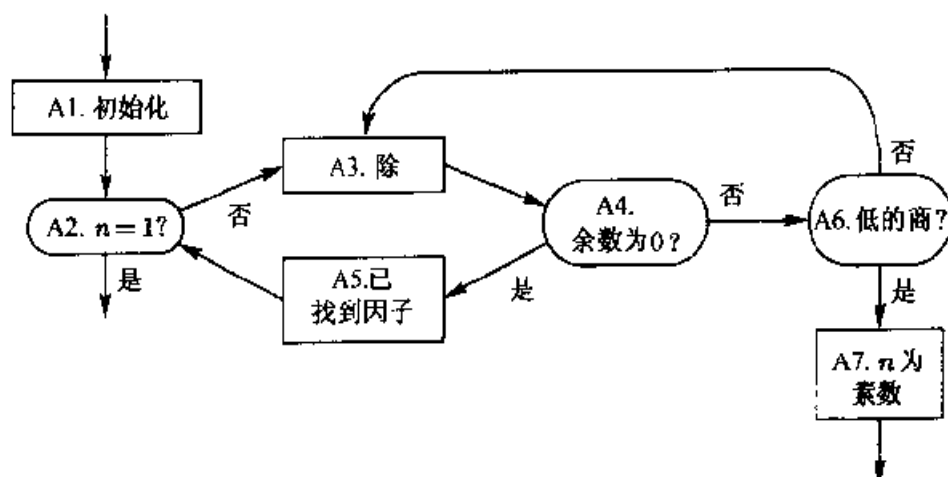


图 11 简单的分解因子算法

作为算法 A 的一个例子, 考虑数 $N=25852$ 的因子分解。我们立即求得 $n = 2 \times 12926$; 因此 $p_1 = 2$ 。而且, $12926 = 2 \times 6463$, 所以 $p_2 = 2$ 。但现在 6463 不为 2, 3, 5, \dots , 19 所整除; 我们发现 $n = 23 \times 281$, 因此 $p_3 = 23$ 。最后 $281 = 12 \times 23 + 5$ 且 $12 \leq 23$; 因此 $p_4 = 281$ 。这个确定 25852 的因子的过程总共含 12 个除法运算; 另一方面, 如果我们试图对稍小的数 25849 (它是素数) 进行因子分解, 至少要执行 38 个除法运算。这说明算法 A 要求大略同 $\max(p_{t-1}, \sqrt{p_t})$ 成比例的运行时间。(如果 $t=1$, 而且我们采取 $p_0=1$ 的约定, 则这个公式是正确的。)

在算法 A 中所用的试验因子序列 d_0, d_1, d_2, \dots 可以简单地取作 2, 3, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35, \dots , 其中在头三项过后, 我们交替地加 2 和 4。这个序列包含了不是 2 和 3 的倍数的所有数, 也包含诸如 25, 35, 49 等等不是素数的数, 但这个算法仍将给出正确的答案。通过从这个表中删去数 $30m \pm 5$ ($m \geq 1$) 可以消去 5 的所有多余倍数。这就可进一步节省 20% 的计算时间。排除 7 的倍数可使这个表缩短 14% 以上, 等等。可以用一份紧凑的二进位小表来控制试验因子的选择。

如果已知 N 很小, 则具有一份所有必要的素数的表作为程序的组成部分是合理的。例如, N 小于一百万, 则我们只需要小于 1000 的 168 个素数就够了 (在 N 是大于 997^2 的素数的情况下, 可以加上值 $d_{168} = 1000$ 来终止这个表)。可以借助一个简短的辅助程序建立这样一个表。例如, 请见算法 1.3.2P, 或见习题 8。

算法 A 中需要多少试验因子呢? 设 $\pi(x)$ 是小于或等于 x 的素数的个数, 则 $\pi(2) = 1, \pi(10) = 4$; 1798 年自 Legendre (勒让德) 开始, 世界上许多伟大的数学家已广泛研究了 this 函数的渐近特性。在 19 世纪期间取得了许多进展而以 1899 年达到顶峰, 这一年 Charles de La Vallée Poussin 证明, 对某个 $A > 0$,

$$\pi(x) = \int_2^x \frac{dt}{\ln t} + O(xe^{-A\sqrt{\log x}}) \quad (3)$$

[*Mém. Couronnés Acad. Roy. Belgique* 59 (1899), 1~74; 也见 J. Hadamard, *Bull. Soc. Math. France* 24 (1896), 199~220.] 由分部积分得到, 对所有固定的 $r \geq 0$,

$$\pi(x) = \frac{x}{\ln x} + \frac{x}{(\ln x)^2} + \frac{2!x}{(\ln x)^3} + \dots + \frac{r!x}{(\ln x)^{r+1}} + O\left(\frac{x}{(\log x)^{r+2}}\right) \quad (4)$$

(3) 中的误差项随后已被改进; 例如, 它可以用 $O(x \exp(-A(\log x)^{3/5}/(\log \log x)^{1/5}))$ 所代替 [见 A. Walfisz, *Weyl'sche Exponentialsummen in der neueren Zahlentheorie* (Berlin: 1963), 第 5 章。] Bernhard Riemann 在 1859 年猜测

$$\pi(x) = \sum_{k=1}^{\lg x} \frac{\mu(k)}{k} L(\sqrt[k]{x}) + O(1) = L(x) - \frac{1}{2}L(\sqrt{x}) - \frac{1}{3}L(\sqrt[3]{x}) + \dots + O(1) \quad (5)$$

其中 $L(x) = \int_2^x \frac{dt}{\ln t}$, 当 x 大小适当时他的公式同实际的计算很吻合。例如, 我们有下列的表 (请见习题 41):

x	$\pi(x)$	$L(x)$	黎曼公式
10^3	168	176.6	168.3
10^6	78498	78626.5	78527.4
10^9	50847534	50849233.9	50847455.4
10^{12}	37607912018	37607950279.8	37607910542.2
10^{15}	29844570422669	29844571475286.5	29844570495886.9
10^{18}	24739954287740860	24739954309690414.0	24739954284239494.4

但是,大素数的分布不是这样简单,1914年 J. E. Littlewood 否定了黎曼的猜想(5);见 Hardy 和 Littlewood, *Acta Math.* **41** (1918), 119~196, 其中证明了对于无穷多的 x 有一个正常数 C 使得 $\pi(x) > L(x) + C\sqrt{x} \log \log \log x / \log x$ 。Littlewood 的结果表明,素数在本质上是有些神秘的,因而在真正地了它们的分布之前,有必要发掘出深刻的数学特性。黎曼作出了另一个更为可信的猜想,即著名的“黎曼假设”。他指出,除了在 z 是一个负偶整数的平凡情况外,仅当 z 的实部等于 $1/2$ 时,复函数 $\zeta(z)$ 为零。如果这个假设为真,则意味着 $\pi(x) = L(x) + O(\sqrt{x} \log x)$; 见习题 25。Richard Brent 使用了 D. H. Lehmer 的一个方法,对于所有“小”的 z 值从计算上验证黎曼的假设,他的方法是证明当虚部的范围是 $0 < \Im z < 32585736.4$ 时 $\zeta(z)$ 正好有 75 000 000 个 0; 所有这些 0 都有 $\Re z = \frac{1}{2}$ 而 $\zeta'(z) \neq 0$ [*Math. Comp.* **33** (1979), 1361~1372]。

为了分析算法 A 的平均特性,我们希望知道最大的素因子 p_i 将趋于多大。这个问题首先是由 Karl Dickman 研究的 [*Arkiv för Mat., Astron. och Fys.* **22A**, 10 (1930), 1~14], 他研究了在 1 和 x 之间的随机整数的最大素因子小于或等于 x^α 的概率。Dickman 利用启发式方法论证,当 $x \rightarrow \infty$ 时这个概率趋向于极限值 $F(\alpha)$, 其中 F 可以从函数方程

$$F(\alpha) = \int_0^\alpha F\left(\frac{t}{1-t}\right) \frac{dt}{t}, \quad \text{当 } 0 \leq \alpha \leq 1 \text{ 时}; \quad F(\alpha) = 1, \text{ 当 } \alpha \geq 1 \text{ 时} \quad (6)$$

计算出来。他的论证实质上是这样: 给定 $0 < t < 1$, 最大素因子处于 x^t 和 t^{t+dt} 之间的小于 x 的整数之个数是 $xF'(t)dt$ 。在该范围内素数 p 的个数是 $\pi(x^{t+dt}) - \pi(x^t) = \pi(x^t + (\ln x)x^t dt) - \pi(x^t) = x^t dt/t$ 。对于每一个这样的 p , 使得“ $np \leq x$ 且 n 的最大素因子小于或等于 p ”的整数 n 的个数是小于或等于 x^{1-t} 的 n 的个数, 这些数的最大素因子小于或等于 $(x^{1-t})^{t/(1-t)}$, 即 $x^{1-t}F(t/(1-t))$ 。因此, $xF'(t)dt = (x^t dt/t)(x^{1-t}F(t/(1-t)))$, 通过积分可得(6)。这种启发性论证可以严格化; V. Ramaswami [*Bull. Amer. Math. Soc.* **55** (1949), 1122~1127] 证明, 当 $x \rightarrow \infty$ 时, 对于固定的 α , 上述问题中的概率渐近地为 $F(\alpha) + O(1/\log x)$, 许多其他作者已经推广了这一分析 [见 Karl K. Norton 的综述, *Memoirs Amer. Math. Soc.* **106** (1971), 9~27]。

如果 $\frac{1}{2} \leq \alpha \leq 1$, 公式(6)简化成为

$$F(\alpha) = 1 - \int_\alpha^1 F\left(\frac{t}{1-t}\right) \frac{dt}{t} = 1 - \int_\alpha^1 \frac{dt}{t} = 1 + \ln \alpha$$

因此, 比如说, 小于或等于 x 的随机整数有大于 \sqrt{x} 的一个素因子的概率是 $1 - F\left(\frac{1}{2}\right) = \ln 2$, 即大约 69%。在所有这样的情况下, 算法 A 必须费尽气力来工作。

这一讨论的纯结果是,如果我们想要分解一个六位数,算法 A 将较快地给出答案;但对于很大的 N ,除非我们意外地走运,不然用试验除法进行因子分解的计算机时间将迅速地超过实用的极限。

在本节的下面一部分我们将看到,有相当好的方法来确定一个相当大的数 n 是否素数,而不必试验直到 \sqrt{n} 之前的所有因子。因此如果我们在步骤 A2 和 A3 之间插入一个素性测试,算法 A 通常将运行得更快;这个改进了的算法的运行时间大体上同 p_{i-1} 成比例,这就是 N 的第二大的素因子,而不是 $\max(p_{i-1}, \sqrt{p_i})$ 。通过类似于 Dickman 的论证(见习题 18),我们可以证明,一个小于或等于 x 的随机整数的第二大素因子小于或等于 x^β 的近似概率为 $G(\beta)$,其中

$$G(\beta) = \int_0^\beta \left(G\left(\frac{t}{1-t}\right) - F\left(\frac{t}{1-t}\right) \right) \frac{dt}{t}, \quad 0 \leq \beta \leq 1/2 \quad (7)$$

显然,对于 $\beta \geq 1/2$, $G(\beta) = 1$ (见图 12)。(6)和(7)的数值计算产生下列的“百分点”:

$F(\alpha), G(\beta) =$.01	.05	.10	.20	.35	.50	.65	.80	.90	.95	.99
$\alpha =$.2697	.3348	.3785	.4430	.5220	.6065	.7047	.8187	.9048	.9512	.9900
$\beta =$.0056	.0273	.0531	.1003	.1611	.2117	.2582	.3104	.3590	.3967	.4517

因此,第二大素因子将有大约一半的时间小于或等于 $x^{.2117}$,等等。

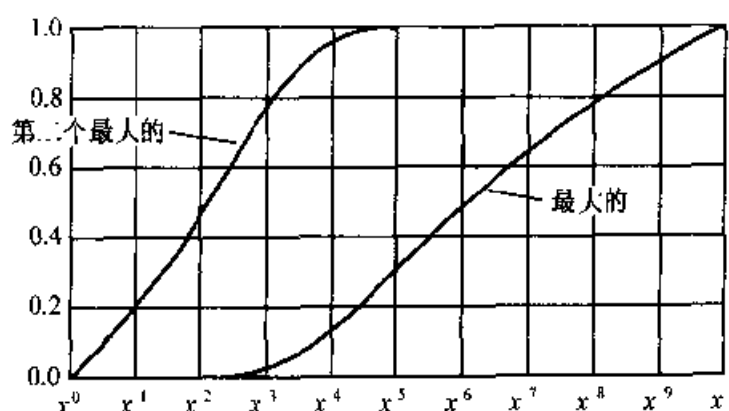


图 12 小于或等于 x 的一个随机整数的两个最大素因子的概率分布函数

素因子的总数 t 也已被广泛地分析了。显然 $1 \leq t \leq \lg N$,但是这些下限和上限很少达到。有可能证明,如果 N 在 1 和 x 之间随机地选择,则当 $x \rightarrow \infty$ 时,对任何固定的 c , $t \leq \ln \ln x + c \sqrt{\ln \ln x}$ 的概率趋向于

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^c e^{-u^2/2} du \quad (8)$$

换句话说, t 的分布实质上是正态的,且均值和方差为 $\ln \ln x$;所有小于或等于 x 的大整数中大约有 99.73% 有 $|t - \ln \ln x| \leq 3 \sqrt{\ln \ln x}$ 。而且,已知对于 $1 \leq N \leq x$, $t - \ln \ln x$ 的平均值趋于

$$\gamma + \sum_{p \text{ 素数}} (\ln(1 - 1/p) + 1/(p-1)) = \gamma + \sum_{n=2}^{\infty} \frac{\varphi(n) \ln \zeta(n)}{n} =$$

$$1.03465\ 38818\ 97437\ 91161\ 97942\ 98464\ 63825\ 47603 + \quad (9)$$

[参考 G. H. Hardy 和 E. M. Wright, *An Introduction to the Theory of Numbers*, 第 5 版 (Oxford, 1979), § 22.11; 也见 P. Erdős 和 M. Kac, *Amer. J. Math.* **26** (1940), 738 ~ 742.]

素因子大小同排列有显著的联系: 当 $n \rightarrow \infty$ 时, 一个随机的 n 个二进位整数的第 k 个最大素因子中平均的二进位数渐近地和一个随机 n 元排列的第 k 个最大循环的平均长度相同 [见 D. E. Knuth 和 L. Trapp Pardo, *Theoretical Comp. Sci.* **3** (1976), 321 ~ 348; A. M. Vershik, *Soviet Math. Doklady* **34** (1987), 57 ~ 61]。由此得出, 算法 A 通常先得到一些小因子而后开始对剩下的一些大因子进行漫长的查找。

Patrick Billingsley 对一个随机整数的素因子的概率分布给出了精彩说明, 见 *AMM* **80** (1973), 1099 ~ 1115; 也参见他在 *Annals of Probability* **2** (1974), 749 ~ 791 的文章。

通过拟随机周期的因子分解 在靠近第 3 章的开始处, 我们发现“随机选择的一个随机数生成程序并不很随机”。在这一章里再次面对我们的这一原理有赎罪式的性质。经 J. M. Pollard 发现 [*BIT* **15** (1975), 331 ~ 334], 它导致一个令人惊讶的有效的因子分解方法。在 Pollard 的方法中的计算步骤数有 $\sqrt{p_{t-1}}$ 的阶, 所以当 N 很大时, 它比算法 A 要快得多, 按照 (7) 和图 12, 运行时间通常将是在 $N^{1/4}$ 之下。

设 $f(x)$ 是带有整系数的任何多项式, 并考虑由

$$x_0 = y_0 = A; \quad x_{m+1} = f(x_m) \bmod N, \quad y_{m+1} = f(y_m) \bmod p \quad (10)$$

定义的两个序列, 其中 p 是 N 的任何素因子, 由此得出, 对于 $m \geq 1$,

$$y_m = x_m \bmod p \quad (11)$$

现在习题 3.1-7 表明对于某个 $m \geq 1$, 我们将有 $y_m = y_{\ell(m)-1}$, 其中 $\ell(m)$ 是小于或等于 m 的 2 的最大次幂。于是 $x_m - x_{\ell(m)-1}$ 将是 p 的一个倍数。其次, 如果 $f(y) \bmod p$ 的特性是从集合 $\{0, 1, \dots, p-1\}$ 到它自身的一个随机映像, 则习题 3.1-12 表明, 最小的这样的 m 的平均值将有 \sqrt{p} 的阶。事实上, 下边的习题 4 表明, 对于随机映像, 这个平均值小于 $1.625 Q(p)$, 其中函数 $Q(p) \approx \sqrt{\pi p/2}$ 是在 1.2.11.3 小节中定义的。如果 N 的不同素因子对应于 m 的不同的值 (因为当 N 很大时, 它们几乎肯定如此), 则我们可以对于 $m = 1, 2, 3, \dots$ 来计算 $\gcd(x_m - x_{\ell(m)-1}, N)$, 直到未分解的余数为素数时为止。Pollard 把他的技术称做“ ρ 方法”, 因为像 y_0, y_1, \dots 这样一个最终周期性的序列让人想起希腊字母 ρ 。

由第 3 章的理论, 我们知道, 就我们的目的而言, 一个线性多项式 $f(x) = ax + c$ 将不是充分随机的。次最简单的情况是二次的, 比如说, $f(x) = x^2 + 1$ 。尽管我们不知道这两函数是充分随机的, 我们的无知势必支持随机性的假设, 而且经验测试

表明,这个 f 实质上是如所期望的那样有效。事实上, f 大概比随机稍好一些,因为 $x^2 + 1$ 只取 $\frac{1}{2}(p+1)$ 个不同的模 p 值。请见 Arney 和 Bender, *Pacific J. Math.* **103** (1982), 269~294。因此下列过程是合理的。

算法 B (通过 ρ 方法的因子分解) 此算法以很高的概率输出一个给定整数 $N \geq 2$ 的素因子, 尽管它有时会失败。

B1. [初始化] 置 $x \leftarrow 5, x' \leftarrow 2, k \leftarrow 1, l \leftarrow 1, n \leftarrow N$ (在此算法执行期间, n 是 N 的未分解部分, 而变量 x 和 x' 表示 (10) 中的量 $x_m \bmod n$ 和 $x_{\ell(m)-1} \bmod n$, 其中 $f(x) = x^2 + 1, A = 2, l = \ell(m), k = 2l - m$)。

B2. [素性检验] 如果 n 是素数 (见以下讨论), 则输出 n ; 算法结束。

B3. [找到因子?] 置 $g \leftarrow \gcd(x' - x, n)$ 。如果 $g = 1$, 则转向步骤 B4; 否则输出 g 。现在如果 $g = n$, 则算法结束 (而且它已经失败, 因为我们知道 n 不是素数), 否则置 $n \leftarrow n/g, x \leftarrow x \bmod n, x' \leftarrow x' \bmod n$, 并返回步骤 B2。 (注意, g 可能不是素数; 这应当加以检验。万一 g 不是素数, 它的素因子大概不能用本算法确定。)

B4. [前进] 置 $k \leftarrow k - 1$ 。如果 $k = 0$, 置 $x' \leftarrow x, l \leftarrow 2l, k \leftarrow l$ 。置 $x \leftarrow (x^2 + 1) \bmod n$ 并返回 B3。 **|**

作为算法 B 的一个例子, 我们再次试验分解 $N = 25852$ 。第三次执行步骤 B3 将输出 $g = 4$ (它不是素数)。继续迭代 6 次之后本算法求得因子 $g = 23$ 。在这个例子中, 算法 B 的效果并不突出, 当然它是为分解大的数而设计的。算法 A 花费更长得多的时间来求大的素因子, 但当它消除小的素因子时, 它是最佳的。实用上, 在转到运行算法 B 之前, 我们应当先运行算法 A 一段时间。

考虑 10 个最大的 6 位素数, 我们能对算法 B 的杰出本领有更好的了解。在下面的表中给出了算法 B 为求素因子 p 所需要做的迭代次数 $m(p)$:

$p =$	999863	999883	999907	999917	999931	999953	999959	999961	999979	999983
$m(p) =$	276	409	2106	1561	1593	1091	474	1819	395	814

实验指出, 当 $p < 1000000$ 时, $m(p)$ 的平均值大约为 $2\sqrt{p}$, 而绝不超出 $12\sqrt{p}$ 。对于 $p < 10^6$, 极大的 $m(p)$ 是 $m(874771) = 7685$; 而且当 $p = 290047$, $m(p) = 6251$ 时, $m(p)/\sqrt{p}$ 的极大值出现。按照这些实验结果, 几乎所有 12 位数的因子分解都可以用算法 B 以少于 2000 次迭代来完成 (同算法 A 的大约 75 000 次除法相比)。

在算法 B 的每次迭代当中, 最花费时间的运算是步骤 B4 中的多精度乘法和除法, 以及步骤 B3 中的 \gcd 。“Montgomery 乘法”技术 (习题 4.3.1-41) 将使这加快。如果 \gcd 运算较慢, Pollard 提议在取每个 \gcd 之前, 累计比如说 10 个相继的 $(x' - x)$ 值的乘积模 n , 以提高速度。这样做能以一个乘法模 N 代替 90% 的 \gcd 运算, 而仅仅稍稍增加一点失败的机会。他也提议在步骤 B1 中以 $m = q$ 而不是 $m = 1$ 开始, 其中 q 比如说是你打算使用的迭代次数的 $\frac{1}{10}$ 。

在对于很大的 N 出现失误的稀少情况下,我们可以对某个 $c \neq 0$ 或 1 , 用 $f(x) = x^2 + c$ 进行试验。也应避免值 $c = -2$, 因为递推式 $x_{m+1} = x_m^2 - 2$ 有形如 $x_m = r^{2^m} + r^{-2^m}$ 的解。 c 的其它值似乎不导致简单的关系模 p , 而且当对适当的开始值使用它们时,应当都是令人满意的。

Richard Brent 使用算法 B 的一个修改发现 $2^{256} + 1$ 的素因子 1238926361552897。[请参见 *Math. Comp.* **36** (1981), 627 ~ 630; **38** (1982), 253 ~ 255。]

费马方法 分解因子问题的另一个方法是由 Pierre de Fermat(费马)于 1643 年使用的。相对而言,它更适合于求大的因子,而不是求小因子。[费马关于他的方法的最初描述已译成英文,可以在 L. E. Dickson 的不朽之作 *History of the Theory of Numbers* 1 (Carnegie Inst. of Washington 1919), 357 中找到。]

假定 $N = uv$, 其中 $u \leq v$ 。为实用起见,我们可以假定 N 是奇数 e ; 这意味着 u 和 v 都是奇数,我们令

$$x = (u + v)/2, \quad y = (v - u)/2 \quad (12)$$

$$N = x^2 - y^2, \quad 0 \leq y < x \leq N \quad (13)$$

费马的方法的要点是系统地寻找满足等式(13)的 x 和 y 的值。因此下边的算法说明怎样可以不使用任何除法来进行分解。

算法 C (用加法和减法进行分解) 给定奇数 N , 本算法确定小于等于 \sqrt{N} 的 N 的最大素因子。

C1. [初始化] 置 $x \leftarrow 2\lfloor\sqrt{N}\rfloor + 1, y \leftarrow 1, r \leftarrow \lfloor\sqrt{N}\rfloor^2 - N$ 。(在此算法执行期间,当我们寻找(13)的一个解时, x, y, r 分别对应于 $2x + 1, 2y + 1, x^2 - y^2 - N$; 我们将有 $|r| < x$ 和 $y < x$ 。)

C2. [完成了吗?] 如果 $r = 0$, 则算法结束; 我们有

$$N = ((x - y)/2)((x + y - 2)/2)$$

因此 $(x - y)/2$ 是小于或等于 \sqrt{N} 的 N 的最大因子。

C3. [步进 x] 置 $r \leftarrow r + x$ 和 $x \leftarrow x + 2$ 。

C4. [步进 y] 置 $r \leftarrow r - y$ 和 $y \leftarrow y + 2$ 。

C5. [测试 r] 如果 $r > 0$ 则返回步骤 C4, 否则转回 C2。 **|**

用手来做这个算法求 377 的因子, 读者可能会感到有兴趣。为了求 $N = uv$ 的因子 u 和 v , 所需步骤实际上与 $(x + y - 2)/2 - \lfloor\sqrt{N}\rfloor = v - \lfloor\sqrt{N}\rfloor$ 成比例, 这当然可能是一个很大的数, 尽管在大多数计算机上每一步都可以非常快地完成。R. S. Lehman 作出了一项改进, 它在最坏的情况下只需要 $O(N^{1/3})$ 次运算 [*Math. Comp.* **28** (1974), 637 ~ 646]。

把算法 C 称做“费马方法”是不十分正确的, 因为费马使用的是一个更精简了的方法。算法 C 的主循环在计算机上是十分快的, 但用手算却不很合适。费马实质上

并不保持 y 的运行值;他的做法是观察 $x^2 - N$ 并且通过观察它的最低有效数字来猜测这个量是否完全平方(一个完全平方的最后两位数字必须是 00, $e1$, $e4$, 25, $o6$ 或 $e9$, 其中 e 为偶数而 o 是奇数)。因此他避免执行步骤 C4 和 C5, 而代之以有时确定某个数不是一个完全平方。

当然观察最右边的数字的费马方法可以用取其它模来加以推广。为明确起见, 假设 $N = 8616460799$, 并考虑下表:

m	如果 $x \bmod m$ 为	则 $x^2 \bmod m$ 为	且 $(x^2 - N) \bmod m$ 为
3	0, 1, 2	0, 1, 1	1, 2, 2
5	0, 1, 2, 3, 4	0, 1, 4, 4, 1,	1, 2, 0, 0, 2
7	0, 1, 2, 3, 4, 5, 6	0, 1, 4, 2, 2, 4, 1	5, 6, 2, 0, 0, 2, 6
8	0, 1, 2, 3, 4, 5, 6, 7	0, 1, 4, 1, 0, 1, 4, 1	1, 2, 5, 2, 1, 2, 5, 2
11	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10	0, 1, 4, 9, 5, 3, 3, 5, 9, 4, 1	10, 0, 3, 8, 4, 2, 2, 4, 8, 3, 0

如果 $x^2 - N$ 是一个完全平方 y^2 , 则对于所有 m , 它必定有同这个事实一致的模 m 剩余。例如, 如果 $N = 8616460799$ 和 $x \bmod 3 \neq 0$, 则 $(x^2 - N) \bmod 3 = 2$, 所以 $x^2 - N$ 不能是一个完全平方; 因此每当 $N = x^2 - y^2$ 时 x 必定是 3 的倍数。事实上, 这个表告诉我们:

$$\begin{aligned}
 x \bmod 3 &= 0 \\
 x \bmod 5 &= 0, 2 \text{ 或 } 3 \\
 x \bmod 7 &= 2, 3, 4 \text{ 或 } 5 \\
 x \bmod 8 &= 0 \text{ 或 } 4 \text{ (因此 } x \bmod 4 = 0) \\
 x \bmod 11 &= 1, 2, 4, 7, 9 \text{ 或 } 10
 \end{aligned} \tag{14}$$

这就相当地缩小了寻找 x 的范围。例如, x 必须是 12 的倍数。我们必须有 $x \geq \lfloor \sqrt{N} \rfloor = 92825$, 因此 12 这样的最小倍数是 92832。这个值模 (5, 7, 11) 分别有剩余 (2, 5, 3), 所以相对于模 11 它不满足 (14)。 x 增加 12 使模 5 剩余改变 2, 模 7 剩余改变 5, 而模 11 剩余改变 1; 所以容易看到满足 (14) 中所有条件的大于等于 92825 的头一个 x 值是 $x = 92880$ 。现在 $92880^2 - N = 10233601$, 而求平方根的纸笔方法告诉我们 $10233601 = 3199^2$ 确实是一个完全平方。因此我们找到了要求的解 $x = 92880, g = 3199$, 因而因子分解是

$$8616460799 = (x - y)(x + y) = 89681 \cdot 96079$$

N 的这个值是有趣的, 因为英国经济学家和逻辑学家 W. S. Jevons 在一本非常有名的书中对它作了如下的介绍: “给定任意两个数, 通过一个简单和确实可靠的过程我们可以得到它们的乘积, 但是当给定一个很大的数来确定它的因子时, 那就完全是另一回事了。读者能够说出哪两个数相乘将产生出数 8 616 460 799 吗? 我想, 除了我之外不大可能有谁竟会知道它。” [The Principles of Science (Macmillan, 1874), 第 7 章。] 然而, 我们已经看到, 费马在一个信封的背面, 用了不到 10 分钟, 就把 N 给分解了。Jevons 关于因子分解相对于乘法的困难的主要论点是切中要害

的,但是那仅仅在形成彼此不相靠近的数的乘积时才如此。

不用(14)中所考虑的模数,我们可以使用不同素数的任何乘幂。例如,如果用25代替5,我们将发现, $x \bmod 25$ 的值只能为0,5,7,10,15,18和20。这就比(14)给出更多的信息。一般来说,对于奇素数 p ,每当 $x^2 - N \equiv 0 \pmod{p}$ 有一个解 x 时,比起模 p 来,模 p^2 将得到更多的信息。

刚才所用的模数方法称为筛选过程,因为我们可以想像所有的整数都通过一个“筛”。对于这个筛来说,只有满足 $x \bmod 3 = 0$ 的那些值能出来,然后把把这些数移到只允许满足 $x \bmod 5 = 0, 2$ 或3的数通过的另一个筛上去,等等。每个筛本身都将删去大约余下的数的一半(见习题6);而且当我们相对于两两互素的模数来筛时,由中国剩余定理(定理4.3.2C),每个筛是同其它筛无关的。所以如果我们相对于(比如说30个)不同的素数进行筛选,在每 2^{30} 个数当中将仅仅大约有一个值需要加以考察,以看看 $x^2 - N$ 是否为完全平方 y^2 。

算法D(用筛进行分解) 给定奇数 N ,本算法确定小于等于 \sqrt{N} 的 N 的最大因子。本过程使用两两互素的且与 N 互素的模 m_1, m_2, \dots, m_r 。对于 $0 \leq j < m_i, 1 \leq i \leq r$,假定我们可以访问 r 个筛表 $S[i, j]$,其中

$$S[i, j] = [j^2 - N \equiv y^2 \pmod{m_i} \text{ 有一个解 } y]$$

D1. [初始化] 置 $x \leftarrow \lceil \sqrt{N} \rceil$,而且对于 $1 \leq i \leq r$ 置 $k_i \leftarrow (-x) \bmod m_i$ (在此算法中下标变量 k_1, k_2, \dots, k_r 将被置成 $k_i = (-x) \bmod m_i$)。

D2. [筛] 如果对于 $1 \leq i \leq r, S[i, k_i] = 1$,转向步骤D4。

D3. [步进 x] 置 $x \leftarrow x + 1$,而且对于 $1 \leq i \leq r$,置 $k_i \leftarrow (k_i + 1) \bmod m_i$,返回步骤D2。

D4. [检验 $x^2 - N$] 置 $y \leftarrow \lfloor \sqrt{x^2 - N} \rfloor$ 或 $\lceil \sqrt{x^2 - N} \rceil$ 。如果 $y^2 = x^2 - N$,则 $(x - y)$ 是所需因子,算法结束;否则返回步骤D3。 ▮

有若干种方法使这个过程运行得更快。例如,我们已经看到,如果 $N \bmod 3 = 2$,则 x 必定是3的倍数;我们可以置 $x = 3x'$,并使用对应于 x' 的不同的筛,把速度增加三倍。如果 $N \bmod 9 = 1, 4$ 或7,则 x 必须分别同余于 $\pm 1, \pm 2$ 或 ± 4 (模9);所以我们使用两个筛(一个是对 x' 的,而另一个是对 x'' 的,其中 $x = 9x' + a$ 和 $x = 9x'' - a$),以把速度增加 $4\frac{1}{2}$ 倍。如果 $N \bmod 4 = 3$,则 $x \bmod 4$ 已知而且速度进一步增加4倍;在其它情况下,当 $N \bmod 4 = 1$ 时, x 必须是奇数,以便速度可以加倍。使本算法的速度加倍的另一方法(以存储空间为代价)是组合成对的模数,并且对于 $1 \leq k < \frac{1}{2}r$ 使用 $m_1 \cdot m_k$ 代替 m_k 。

加速算法D的一个甚至更为重要的方法是使用在大多数二进计算机上使用的“布尔运算”。例如,我们假定MIX是每个字有30位的一台二进计算机,表 $S[i, k_i]$ 可以保存于内存中;每项只用一位;于是30个值可以存入一个字中。对于 $1 \leq k \leq$

30, 如果内存中一个特定字的第 k 个为 0, 则 AND 运算用 0 代替累加器的第 k 位, 于是 AND 可以一次用来处理 x 的 30 个值! 为方便起见, 我们可以复制若干个 $S[i, j]$ 表, 使得 m_i 的表项含 $\text{lcm}(m_i, 30)$ 位; 然后每个模的筛表填满整数个字。在这些假定之下, 30 次执行算法 D 中的主循环等价于下列形式的代码:

D2	LD1	K1	$r11 \leftarrow k'_1$
	LDA	S1, 1	$rA \leftarrow S'[1, r11]$
	DEC1	1	$r11 \leftarrow r11 - 1$
	J1NN	$* + 2$	
	INC1	M1	如果 $r11 < 0$, 则置 $r11 \leftarrow r11 + \text{lcm}(m_1, 30)$
	ST1	K1	$k'_1 \leftarrow r11$
	LD1	K2	$r11' \leftarrow k'_2$
	AND	S2, 1	$rA \leftarrow rA \wedge S'[2, r11]$
	DEC1	1	$r11 \leftarrow r11 - 1$
	J1NN	$* + 2$	
	INC1	M2	如果 $r11 < 0$, 则置 $r11 \leftarrow r11 + \text{lcm}(m_2, 30)$
	ST1	K2	$k'_2 \leftarrow r11$
	LD1	K3	$r11 \leftarrow k'_3$
	...		(m_3 到 m_r 与 m_2 类似)
	ST1	Kr	$k'_r \leftarrow r11$
	INCX	30	$x \leftarrow x + 30$
	JAZ	D2	如果全都筛完, 则重复 I

30 次迭代的循环次数实质上是 $2 + 8r$; 如果 $r = 11$, 这意味着在每次迭代中使用三次循环, 恰如在算法 C 中那样, 而且算法 C 还多包含另外 $y = \frac{1}{2}(v - u)$ 次迭代。

如果 m_i 的表项未充满整数个字, 就要在每次迭代时进一步移动表项, 以便适当地调整二进位。这将往主循环附加上大量的代码, 而且它大概将使这个程序太慢, 因而比不上算法 C, 除非 $v/u \leq 100$ (见习题 7)。

筛过程可以应用于大量其它的问题, 而不必要做许多算术运算。Marvin C. Wunderlich 编写了关于这些技术的一个评述, 见 JACM 14 (1967), 10~19。

F. W. Lawrence 在 19 世纪提出了用于因子分解的特殊筛机器的构造 [Quart. J. of Pure and Applied Math. 28 (1896), 285~311], 而 E. O. Carissan 于 1919 年使用 14 个模完成了这样一个装置 [关于 Carissan 遗失了很久的筛如何被重新发现并流传后世的有趣故事, 请参见 Shallit, Williams 和 Morain, Math. Intelligencer 17, 3 (1995), 41~47]。D. H. Lehmer 和他的助手们在 1926 年—1989 年期间构造和使用了许多不同的筛装置, 他们从使用自行车链开始, 而后使用光电单元和其它类型的技术;

比如,参见AMM 40 (1933), 401~406。Lehmer的电子延迟线筛1965年开始操作,每秒处理100万个数。到1995年,有可能构造每秒筛61.44亿个数的机器,在大约5.2 ns中执行256次步骤D2和D3的迭代[参见Lukes, Patterson和Williams, *Nieuw Archief voor Wiskunde* (4) 13 (1995), 113~119]。D. H. Lehmer和Emma Lehmer在 Math. Comp. 28 (1974), 625~635上描述了用筛进行因子分解的另外的方法。

素性检验 至今我们所讨论的算法,没有一个是确定一个很大的数 n 是否素数的有效方法。幸而解决这个问题还有其它方法可用,É. Lucas和其他人,主要是D. H. Lehmer已经给出了一些有效的方法[见 $\text{Bull. Amer. Math. Soc.}$ 33 (1927), 327~340]。

按照费马定理(定理1.2.4F),每当 p 为素数和 x 不是 p 的倍数时,我们有 $x^{p-1} \bmod p = 1$ 。而且有有效的方法计算 $x^{n-1} \bmod n$,且只需要 $O(\log n)$ 个模 n 乘法运算(我们将在下边的4.6.3小节研究它们)。因此我们通常可以确定,当这个关系不成立时, n 不是素数。

例如,费马曾经证明数 $2^1+1, 2^2+1, 2^4+1, 2^8+1$ 及 $2^{16}+1$ 为素数。在1640年写给梅森的一封信中,费马猜测 $2^{2^n}+1$ 总是素数,但他说他没有能力确定数 $4294967297 = 2^{32}+1$ 是否素数。费马和梅森都没有解决这个问题,尽管他们可以用下列方法做到这一点:数 $3^{2^{32}} \bmod (2^{32}+1)$ 可以通过做32次模 $2^{32}+1$ 平方运算来计算,且答案是3029026160;因此(根据费马自己的定理,他在同一年即1640年,发现了它!),数 $2^{32}+1$ 不是素数。这个论证绝对没有提示我们哪些是因子,但它回答了费马的问题。

对于证明一个给定数的非素性来说,费马定理是一个强有力的检验。当 n 不是素数时,总是有可能来求 $x < n$ 的一个值,使得 $x^{n-1} \bmod n \neq 1$;事实上,经验证明,这样一个值几乎总能非常快地求出。有某些稀少的 n 值,它们经常使得 $x^{n-1} \bmod n$ 等于1,但在此情况下 n 有小于 $\sqrt[3]{n}$ 的因子;见习题9。

使用下面的思想可以推广这个方法来证明一个很大的素数 n 真是素数:如果有一个数 x ,对于它来说 x 的模 n 阶等于 $n-1$,则 n 是素数(x 的模 n 阶是使得 $x^k \bmod n = 1$ 的最小正整数 k ;见3.2.1.2小节。)因为这个条件意味着数 $x^1 \bmod n, \dots, x^{n-1} \bmod n$ 是不同的,且与 n 互素,所以它们必定是在某个顺序下的数 $1, 2, \dots, n-1$;于是 n 没有真因子。如果 n 是素数,那么这样一个数 x (称做 n 的一个原根)将总是存在;见习题3.2.1.2-16。事实上,原根相当多,它们有 $\varphi(n-1)$ 个,而这是相当大的数,因为 $n/\varphi(n-1) = O(\log \log n)$ 。

没有必要对所有小于或等于 $n-1$ 的 k 来计算 $x^k \bmod n$,以确定 x 的阶是否 $n-1$; x 的阶为 $n-1$ 的充要条件是:

i) $x^{n-1} \bmod n = 1$;

ii) 对于整除 $n-1$ 的所有素数 p , $x^{(n-1)/p} \bmod n \neq 1$ 。

因为当且仅当 s 是 x 的模 n 阶的倍数时 $x^s \bmod n = 1$ 。如果两个条件成立,而且如果 k 是 x 的模 n 阶,则我们由此知道 k 是 $n-1$ 的一个因子,但对于 $n-1$ 的任何素因子 p ,不是 $(n-1)/p$ 的一个因子;惟一剩下的可能性是 $k = n-1$ 。这就完成了证明,即条件 i) 和 ii) 足以保证 n 的素性。

习题 10 表明,对于每个素数 p ,我们事实上可以使用不同的 x 值,且 n 将仍然是素数;我们可以把考虑限于 x 的素数值,因为由习题 3.2.1.2-15, uv 模 n 的阶整除 u 和 v 的阶的最小公倍数。使用在 4.6.3 节中所讨论的求数的乘幂的快速方法,可以有效地检验条件 i) 和 ii)。但是有必要知道 $n-1$ 的素因子,所以我们有一个有趣的情况,在这个情况下, n 的因子分解依赖于 $n-1$ 的因子分解。

一个例子 对于相当典型的大数因子分解的研究有助于牢记我们已经讨论过的一些思想。我们来求一下 $2^{214} + 1$, 一个 65 位数的素因子。如果我们注意到

$$2^{214} + 1 = (2^{107} - 2^{54} + 1)(2^{107} + 2^{54} + 1) \quad (15)$$

则稍有一点洞察力就可开始这个因子分解:它是因子分解 $4x^4 + 1 = (2x^2 + 2x + 1) \cdot (2x^2 - 2x + 1)$ 的一个特殊情况。1742 年欧拉曾问哥德巴赫写信谈起它 [P. H. Fuss, *Correspondance Math. et Physique* 1 (1843), 145]。这个问题现在归结成为考察 (15) 中的每一个 33 位因子。

一个计算机程序很容易就发现 $2^{107} - 2^{54} + 1 = 5 \cdot 857 \cdot n_0$, 其中

$$n_0 = 37866809061660057264219253397 \quad (16)$$

是没有小于 1000 的素因子的 29 位数。

使用算法 4.6.3A 进行多精度计算表明

$$3^{n_0-1} \bmod n_0 = 1$$

所以我们怀疑 n_0 是素数。通过试验十万亿左右可能的因子来证明 n_0 是素数肯定是行不通的,但是上边讨论的方法给出了关于素性的一个能行的检验:我们的下一个目标是来分解 $n_0 - 1$ 。只经过一点点的困难,计算机就告诉我们

$$n_0 - 1 = 2 \cdot 2 \cdot 19 \cdot 107 \cdot 353 \cdot n_1, \quad n_1 = 13191270754108226049301$$

这里 $3^{n_1-1} \bmod n_1 \neq 1$, 所以 n_1 非素数;继续进行算法 A 或算法 B, 我们求得

$$n_1 = 91813 \cdot n_2, \quad n_2 = 143675413657196977$$

这次 $3^{n_2-1} \bmod n_2 = 1$, 所以我们将试图证明 n_2 是素数。求出小于 1000 的因子产生出 $n_2 - 1 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 3 \cdot 3 \cdot 547 \cdot n_3$, 其中 $n_3 = 1824032775457$ 。由于 $3^{n_3-1} \bmod n_3 \neq 1$, 我们知道 n_3 是合数,而且算法 A 求出 $n_3 = 1103 \cdot n_4$, 其中 $n_4 = 1653701519$ 。数 n_4 表现得好像是一个素数(即 $3^{n_4-1} \bmod n_4 = 1$), 所以我们计算

$$n_4 - 1 = 2 \cdot 7 \cdot 19 \cdot 23 \cdot 137 \cdot 1973$$

好;这就是我们头一次完全的因子分解。我们现在可以返回去解释原先的一个子问题,即证明 n_4 是素数。使用习题 10 所建议的过程,我们计算下列的值:

x	p	$x^{(n_4-1)/p} \bmod n_4$	$x^{n_4-1} \bmod n_4$	
2	2	1	(1)	
2	7	766408626	(1)	
2	19	332952683	(1)	
2	23	1154237810	(1)	
2	137	373782186	(1)	(17)
2	1973	490790919	(1)	
3	2	1	(1)	
5	2	1	(1)	
7	2	1653701518	1	

(这里“(1)”表示无需计算的结果 1,因为它可以从以前的计算推导出来。)于是 n_4 是素数,而且 n_2-1 已完全分解。类似的计算表明 n_2 是素数,而且 n_0-1 的这一完全因子分解最终证明,在另一个类似于(17)的计算之后, n_0 是素数。

(17)的最后三行表示查找满足 $x^{(n_4-1)/2} \not\equiv x^{n_4-1} \equiv 1 \pmod{n_4}$ 的整数 x 。如果 n_4 是素数,我们只有 50% 的成功机会,所以情况 $p=2$ 典型地是最难验证的一个。我们使用二次倒数法则(参考习题 23)把这部分计算流水线化,它告诉我们,例如,每当 q 是一个同余于 $\pm 1 \pmod{5}$ 的素数时必有 $5^{(q-1)/2} \equiv 1 \pmod{q}$ 。只要计算 $n_4 \bmod 5$ 就会明白, $x=5$ 大概不可能有助于证明 n_4 是素数。但是事实上,习题 26 的结果意味着当检验 n 的素性时, $p=2$ 的情况实际上全然不必加以考虑,除非 $n-1$ 可以为 n 的一个高次幂所整除,所以我们可能要整个地删除(17)的最后三行。

要分解的下一个量是(15)的另一半,即

$$n_5 = 2^{107} + 2^{54} + 1$$

由于 $3^{n_5-1} \bmod n_5 \neq 1$,我们知道 n_5 不是素数,而且算法 B 表明 $n_5 = 843589 \cdot n_6$,其中 $n_6 = 192343993140277293096491917$ 。不幸的是, $3^{n_6-1} \bmod n_6 \neq 1$,所以留给我们的一个 27 位的非素数。继续执行算法 B 会把我们的耐心消耗完(不是我们的预算——因为我们正在使用的是周末的闲暇时间,而不是“主要时间”(这里主要一词即素数——译者))。但是算法 D 的筛方法将能把 n_6 分解成它的两个因子

$$n_6 = 8174912477117 \cdot 23528569104401$$

(结果是在 6 432 966 次迭代之后,算法 B 也成功了。)在一个相当长的时间里,算法 A 不可能发现 n_6 的因子。

现在计算完成了: $2^{214} + 1$ 有素因子分解

$$5 \cdot 857 \cdot 843589 \cdot 8174912477117 \cdot 23528569104401 \cdot n_0$$

其中 n_0 是(16)中的 29 位素数。在这些计算当中包含一些好运气,因为如果我们不以已知的因子分解(15)开始,十分可能的是,我们首先将找出小的因子,把 n 归结为

$n_5 n_0$ 。分解这 55 位数字更难得多了——算法 D 将是无用的,而且算法 B 使用的时间将太多,因为在这里高精度是必须的。

在 John Brillhart 和 J. L. Selfridge 的一论篇文 [*Math. Comp.* **21** (1967), 87 ~ 96)] 中,可以找到数十个进一步的数值例子。

改进的素性检验 刚才说明的过程要求,在我们能证明 n 是素数之前,要对 $n-1$ 进行完全的因子分解,所以对于很大的 n ,它将陷于困境。在习题 15 中描述了另一个技术,它代而使用对 $n+1$ 的因子分解,如果 $n-1$ 证明是太难了,则 $n+1$ 也许会容易些。

有一些重大改进可利用来处理很大的 n 。例如,不难证明费马定理的一个更强的逆定理,它只要求 $n-1$ 的部分因子分解。习题 26 表明,我们可以避免(17)中的大多数计算;三个条件 $2^{n_4-1} \bmod n_4 = \gcd(2^{(n_4-1)/23} - 1, n_4) = \gcd(2^{(n_4-1)/1973} - 1, n_4) = 1$ 足以证明 n_4 是素数,Brillhart、Lehmer 和 Selfridge 事实上已经建立了一种方法,该法只需要数 $n-1$ 和 $n+1$ 的部分分解 [*Math. Comp.* **29** (1975), 620 ~ 647], 推论 11]: 假设 $n-1 = f^- r^-$ 且 $n+1 = f^+ r^+$, 其中我们知道 f 和 f^+ 的完全因子分解,而且还知道, r^- 和 r^+ 的所有因子大于等于 b 。如果乘积 $(b^3 f^- f^+ \cdot \max(f^-, f^+))$ 大于 $2n$, 则只需少量额外计算(在他们的文章中有介绍)即可确定 n 是否为素数。因此,只须通过找出 $n \pm 1$ 的所有小于 30030 的素因子,通常在 1 s 之内就可以检验多达 35 位数字的素性[见 J. L. Selfridge 和 M. C. Wunderlich, *Congressus Numerantium*, **12** (1974), 109 ~ 120]。像 $n^2 \pm n + 1$ 和 $n^2 + 1$ 这样其它量的部分因子分解,可以进一步用来改进这个方法。[见 H. C. Williams 和 J. S. Judd, *Math. Comp.* **30** (1976), 157 ~ 172, 867 ~ 886]。

实际上,当 n 没有小的素因子且 $3^{n-1} \bmod n = 1$ 时,进一步的计算几乎总是表明 n 是素数(在作者的经验中,一个稀有的例外是 $n = \frac{1}{7}(2^{28} - 9) = 2341 \cdot 16381$)。

另一方面, n 的某些非素数值对于我们已经讨论的素性检验来说肯定是坏消息,因为对子所有与 n 互素的 x , $x^{n-1} \bmod n = 1$ 可能成立(见习题 9)。一个最小的这样的数是 $n = 3 \cdot 11 \cdot 17 = 561$; 这里,在等式 3.2.1.2-(9)的记号下, $\lambda(n) = \text{lcm}(2, 10, 16) = 80$, 所以每当 x 与 561 互素时 $x^{80} \bmod 561 = 1 = x^{560} \bmod 561$ 。在偶然碰见它的因子以前,我们的过程将不断失败。为了改进这个方法,我们需要一个快速的方法来确定非素数 n 的非素性,甚至在这种病态情况下亦然。

下面的令人吃惊的简单过程能以很高的概率来完成这一工作。

算法 P (概率素性检验) 给定奇整数 n , 本算法判定 n 是否素数。如同在下边的解释中所说明的,重复执行此算法若干次,在一个精确的意义下,有可能使 n 的素性极为可信,但是没有严格的证明。设 $n = 1 + 2^k q$, 其中 q 为奇数。

P1. [生成 x] 设 x 是在 $1 < x < n$ 范围内的一个随机数。

P2. [乘幂化] 置 $j = 0$ 和 $y \leftarrow x^q \bmod n$ (如同在以前的素性检验中一样, $x^q \bmod$

n 应当在 $O(\log q)$ 步内计算出, 参考 4.6.3 小节)。

P3. [完成了?] (现在 $y = x^{2^j} \bmod n$) 如果 $j = 0$ 和 $y = 1$, 或者如果 $y = n - 1$, 则结束这个算法并说“ n 可能是素数”。如果 $j > 0$ 和 $y = 1$, 则转向步骤 P5。

P4. [j 增值] j 增加 1。如果 $j < k$, 则置 $y \leftarrow y^2 \bmod n$ 并返回步骤 P3。

P5. [不是素数] 结束算法, 并说“ n 肯定不是素数”。 ■

构成算法 P 的思想是, 如果 $x^q \bmod n \neq 1$ 和 $n = 1 + 2^k q$ 是素数, 则值序列

$$x^q \bmod n, x^{2^q} \bmod n, x^{4^q} \bmod n, \dots, x^{2^{k-1}q} \bmod n$$

将以 1 结束, 而且在头一个 1 的前边的值将是 $n - 1$ (当 p 是素数时, 对于 $y^2 \equiv 1 \pmod{p}$, 仅有的解是 $y \equiv \pm 1$, 因为 $(y - 1)(y + 1)$ 必须是 p 的倍数)。

习题 22 证明了对于所有 n , 算法 P 出错的可能顶多是 $\frac{1}{4}$ 这一基本事实。实际上, 对于大多数 n 它很少失灵; 但关键之点是, 不管 n 的值如何, 失误的概率有界。

假设我们重复调用算法 P, 而且每当我们到达步骤 P1 时, 独立地和随机地选择 x 。一旦此算法报告 n 是非素数, 我们就可以确信地说, n 肯定不是素数。但如果此算法在一行中 25 次报告说 n 可能是素数, 则我们可以说 n “几乎肯定是素数”。因为这样一个一行 25 次过程给出关于它的输入的错误信息的概率小于 $(1/4)^{25}$ 。这种机会小于 10^{15} 分之一。即使我们以这样一个过程验证了 10 亿个不同的素数, 预料出错的个数仍将小于 $\frac{1}{1000000}$ 。因此如果真出了错, 与其说算法 P 重复地猜测错, 倒不如说由于硬件的失灵或宇宙射线的原因, 我们的计算机在它的计算中丢了一位。

这样的概率性算法使我们对传统的可靠性标准提出一个问号: 我们是否真正需要素性的严格证明。由于人们并不想放弃传统的证明思想, Gary L. Miller 已经证明 (以稍弱的形式), 如果数论中称做广义黎曼假设的某一著名猜测可被证明, 则或者 n 是素数, 或者存在一个 $x < 2(\ln n)^2$ 使得算法 P 将发现 n 的非素性 [见 J. Comp. System Sci. **13** (1976), 300~317]。在这个上界中的常数 2 是由 Eric Bach 给出的, Math. Comp. **55** (1990), 355~380。关于黎曼假设的各种推广的说明, 请见 E. Bach 和 J. O. Shallit 所著 *Algorithmic Number Theory 1* (MIT Press, 1996), 第 8 章。于是, 相对于运行时间为 $O(\log n)^3$ 的概率性方法, 我们将有在 $O(\log n)^5$ 个基本操作下检验素性的严格方法。但人们也可能问起, 是否广义黎曼假设的任何所谓的证明有和对随机的 x 重复应用算法 P 同样的可靠性。

关于素性的概率性检验首先是由 R. Solovay 和 V. Strassen 于 1974 年提出的, 他们发现了习题 23b) 描述的有趣但更复杂的检验 [见 SICOMP **6** (1977), 84~85; **7** (1978), 118], 算法 P 是 M. O. Rabin 给出的一个过程的简化版本, 它部分是以 Gary L. Miller 的思想为基础的 [参考由 J. F. Traub 编的 *Algorithms and Complexity* (Academic Press, 1976), 35~36]。B. Arazzi [Comp. J. **37** (1994), 219~222] 已经发现通过对于剩余的数使用 Montgomery 的快速方法, 对于很大的 n 可使算法 P 大大加速

(习题 4.3.1-41)。

素性检验的一个完全不同的方法是 1980 年由 Leonard M. Adleman 发现的。他的极为有趣的方法是以代数整数论为基础的, 所以它超出了本书的范围; 但它导致了一个非概率的过程, 此过程将在至多几个小时内, 判定多达比如说 250 位数字的任何数的素性。[一般来说, 运行时间是 $(\log n)^{O(\log \log \log n)}$; 请见 L. M. Adleman, C. Pomerance 以及 R. S. Rumely, *Annals of Math.* **117** (1983), 173 ~ 206.] 但 H. W. Lenstra, Jr. 所做修改在实践中是更快的, 它已由 H. Cohen 和 A. K. Lenstra 所实现的 [*Math. Comp.* **42** (1984), 297 ~ 330; **48** (1987), 103 ~ 121]。

Adleman 和黄铭德 (Ming-Deh A. Huang) 后来发现一个过程, 它对于所有素数 n 寻找素性的严格证明, 而且以很高的概率具有 $\log n$ 的多项式的运行时间 [*Lecture Notes in Math.* **1512** (1992)]。然而, 他们的方法似乎只有纯粹的理论价值。

通过连分数分解因子 至今我们已经讨论的分解因子过程通常在 30 位数或更多位数时就受挫了。因而, 如果要前进一步就需要新的思想。幸而确有这样的思想; 事实上, 有分别由 A. M. Legendre 和 M. Kraitchik 给出的两种思想, 它们导致许多年前 D. H. Lehmer 和 R. E. Powers 发明出一种新技术 [*Bull. Amer. Math. Soc.* **37** (1931), 770 ~ 776]。但是, 这个方法在当时未被使用, 因为它不大适合于台式计算机。这种否定的判断一直流行到 20 世纪 60 年代末, 那时 John Brillhart 发现 Lehmer-Powers 的方法值得重新引起注意, 因为它非常适合于计算机程序设计。事实上, 后来他和 Michael A. Morrison 把它发展成为在 70 年代有名的所有多精度因子分解方法的翘楚。他们的程序在一台 IBM 360/91 计算机上大约 30 s 之内处理典型的 25 位数字, 在约 50 min 之内处理 40 位数字 [见 *Math. Comp.* **29** (1975), 183 ~ 205]。1970 年这个方法获得了它的头一次辉煌的成功, 发现 $2^{128} + 1 = 59649589127497217 \cdot 5704689200685129054721$ 。

其基本思想是寻找数 x 和 y 使得

$$x^2 \equiv y^2 \pmod{N}, \quad 0 < x, y < N, x \neq y, x + y \neq N \quad (18)$$

费马的方法提出了较强的要求 $x^2 - y^2 = N$ 。但实际上同余式 (18) 足以把 N 分成因子: 它意味着 N 是 $x^2 - y^2 = (x - y)(x + y)$ 的因子, 但 N 既不整除 $x - y$ 也不整除 $x + y$; 因此 $\gcd(N, x - y)$ 和 $\gcd(N, x + y)$ 是 N 的真因子, 它们能用 4.5.2 小节中的有效方法找到。

发现 (18) 的解的一个方法是对于小的值 $|a|$, 寻找 x 的值使得 $x^2 \equiv a \pmod{N}$ 。我们将会看到, 把这个同余式的一些解归拢在一起来得到 (18) 的解通常是一件简单的事。现在对于某个 k 和 d , 以及小的 $|a|$, 如果 $x^2 = a + kNd^2$, 则分式 x/d 是对于 \sqrt{kN} 的一个好的近似; 反之, 如果 x/d 是 \sqrt{kN} 的一个特别好的近似, 则差 $|x^2 - kNd^2|$ 将很小。这个发现提示我们寻找 \sqrt{kN} 的连分数展开, 因为我们在等式 4.5.3-(12) 和习题 4.5.3-42 中已经看到, 连分数产生好的有理近似。

在习题 4.5.3-12 中证明了二次无理式的连分数的许多使人高兴的性质。以下的算法利用这些性质导出对同余式

$$x^2 \equiv (-1)^{e_0} p_1^{e_1} p_2^{e_2} \cdots p_m^{e_m} \pmod{N} \quad (19)$$

的解。这里我们使用小素数 $p_1=2, p_2=3, \dots$, 直到 p_m 的固定集合; 在这个表中应出现的 p 都满足 $p=2$ 或 $(kN)^{(p-1)/2} \bmod p \leq 1$, 因为其它的素数绝不会是由此算法生成的数的因子(见习题 14)。如果 $(x_1, e_{01}, e_{11}, \dots, e_{m1}), \dots, (x_r, e_{0r}, e_{1r}, \dots, e_{mr})$ 是(19)的解, 它们使得向量和

$$(e_{01}, e_{11}, \dots, e_{m1}) + \dots + (e_{0r}, e_{1r}, \dots, e_{mr}) = (2e'_0, 2e'_1, \dots, 2e'_m) \quad (20)$$

的每个分量都是偶数, 则除了可能有 $x \equiv \pm y$ 外,

$$x = (x_1 \cdots x_r) \bmod N, \quad y = ((-1)^{e'_0} p_1^{e'_1} \cdots p_m^{e'_m}) \bmod N \quad (21)$$

产生(18)的一个解。条件(20)实质上指出这些向量模 2 线性相关, 所以如果已经找到(19)的至少 $m+2$ 个解, 则我们必定有(20)的一个解。

算法 E(通过连分数进行因子分解) 给定一个正整数 N 和正整数 k , 使得 kN 不是一个完全平方, 本算法通过分析 \sqrt{kN} 的连分数的收敛式, 试图发现对于固定的 m , 同余式(19)的解。(另一个算法使用这些输出来发现 N 的因子, 这是习题 12 的主题。)

- E1.** [初始化] 置 $D \leftarrow kN, R \leftarrow \lfloor \sqrt{D} \rfloor, R' \leftarrow 2R, U \leftarrow U' \leftarrow R', V \leftarrow 1, V' \leftarrow D - R^2, P \leftarrow R, P' \leftarrow 1, A \leftarrow 0, S \leftarrow 0$ 。(此算法遵循习题 4.5.3-12 的一般过程, 寻找 \sqrt{kN} 的连分数展开。变量 U, U', V, V', P, P', A 和 S 分别表示该习题称为 $R + U_n, R + U_{n-1}, V_n, V_{n-1}, p_n \bmod N, p_{n-1} \bmod N, A_n$ 及 $n \bmod 2$ 的那些量。我们将总有 $0 < V < U \leq R'$, 所以仅仅 P 和 P' 需要最高精度。)
- E2.** [推进 U, V, S] 置 $T \leftarrow V, V \leftarrow A(U' - U) + V', V' \leftarrow T, A \leftarrow \lfloor U/V \rfloor, U' \leftarrow U, U \leftarrow R' - (U \bmod V), S \leftarrow 1 - S$ 。
- E3.** [分解 V] (由习题 4.5.3-12c) 对于与 P 互素的某个 Q , 我们有 $P^2 - kNQ^2 = (-1)^S V$ 。置 $(e_0, e_1, \dots, e_m) \leftarrow (S, 0, \dots, 0), T \leftarrow V$ 。现在对 $1 \leq j \leq m$ 做下列工作: 如果 $T \bmod p_j = 0$, 则置 $T \leftarrow T/p_j$; 且 $e_j \leftarrow e_j + 1$, 并重复这个过程直到 $T \bmod p_j \neq 0$ 为止。
- E4.** [得到解了?] 如果 $T = 1$, 则输出值 $(P, e_0, e_1, \dots, e_m)$, 它组成(19)的一个解。(如果已经生成足够的解, 则我们现在可以结束这个算法。)
- E5.** [推进 P, P'] 如果 $V \neq 1$, 则置 $T \leftarrow P, P \leftarrow (AP + P') \bmod N, P' \leftarrow T$, 并返回步骤 E2。否则连分数过程已开始重复它的循环, 也许对 S 除外, 所以算法结束。(循环通常很长, 以致这种情况不会发生。) ■

考虑一个例子: $N = 197209, k = 1, m = 3, p_1 = 2, p_2 = 3, p_3 = 5$, 我们可以说明算法 E 对相对小的数的应用。这个计算过程如表 1 所示。

表1 算法E的一个示例

 $N = 197209, k = 1, m = 3, p_1 = 2, p_2 = 3, p_3 = 5$

	L	V	A	P	S	T	输出
E1 之后:	888	1	0	444	0	-	
E4 之后:	876	73	12	444	1	73	
E4 之后:	882	145	6	5329	0	29	
E4 之后:	857	37	23	32418	1	37	
E4 之后:	751	720	1	159316	0	1	$159316^2 \equiv +2^2 \cdot 3^2 \cdot 5^1$
E4 之后:	852	143	5	191734	1	143	
E4 之后:	681	215	3	131941	0	43	
E4 之后:	863	656	1	193139	1	41	
E4 之后:	883	33	26	127871	0	11	
E4 之后:	821	136	6	165232	1	17	
E4 之后:	877	405	2	133218	0	1	$133218^2 \equiv +2^0 \cdot 3^4 \cdot 5^1$
E4 之后:	875	24	36	37250	1	1	$37250^2 \equiv -2^3 \cdot 3^1 \cdot 5^0$
E4 之后:	490	477	1	93755	0	53	

继续计算,在头100次迭代中给出25个输出;换句话说,这个算法正在十分迅速地求解。但是某些解是平凡的,例如,如果上述的计算再多继续14次,我们将得到输出 $197197^2 \equiv 2^4 \cdot 3^2 \cdot 5^0$,它不值得注意,因为 $197197 \equiv -12$ 。上边的头两个解已足以完成这个因子分解:我们已经求得

$$(159316 \cdot 133218)^2 \equiv (2^2 \cdot 3^3 \cdot 5^1)^2 \pmod{197209}$$

于是(18)对于 $x = (159316 \cdot 133218) \bmod 197209 = 126308, y = 540$ 成立。由欧几里得算法, $\gcd(126308 - 540, 197209) = 199$;因此我们得到很好的因子分解

$$197209 = 199 \cdot 991$$

按照 R. Schroepel 于1975年告诉作者的未发表的思路,对算法E的运行时间作一启发式的分析,可以理解该算法为什么能如此成功地分解大的数。为方便起见,我们假定 $k = 1$,为产生 N 的因子分解所需的输出数大约同正被分解出的小素数的个数 m 成正比。步骤E3的每次执行花费的时间的阶大约为 $m \log N$ 个时间单位,所以总共的运行时间大约同 $m^2 \log N / P$ 成比例,其中 P 是每次迭代有一成功输出的概率。如果我们做保守的假设,即 V 随机地分布于0和 $2\sqrt{N}$ 之间,则概率 P 是 $(2\sqrt{N})^{-1}$ 乘以小于 $2\sqrt{N}$ 的整数个数,而这些整数的素因子全都在集合 $\{p_1, \dots, p_m\}$ 中。习题29给出 P 的一个下限,由之我们可得出结论:运行时间的阶至多为

$$\frac{2\sqrt{N}m^2 \log N}{m^r / r!}, \quad \text{其中 } r = \left\lfloor \frac{\log 2\sqrt{N}}{\log p_m} \right\rfloor \quad (22)$$

如果命 $\ln m$ 近似于 $\frac{1}{2} \sqrt{\ln N \ln \ln N}$, 则我们有 $r \approx \sqrt{\ln N / \ln \ln N} - 1$, 假定 $p_m = O(m \log m)$, 则公式(22)归结为

$$\exp(2\sqrt{(\ln N)(\ln \ln N)} + O((\log N)^{1/2}(\log \log N)^{-1/2}(\log \log \log N)))$$

换句话说,在相当可信的假定下,预期算法的运行时间至多为 $N^{\epsilon(N)}$,其中当 $N \rightarrow \infty$ 时指数 $\epsilon(N) \approx 2\sqrt{\ln \ln N / \ln N}$ 趋向 0。

当 N 是在实用范围中时,我们当然不能对这种渐近估计太过认真。例如,如果 $N = 10^{50}$,当 $\alpha \approx 4.75$ 时我们有 $N^{1/\alpha} = (\lg N)^\alpha$,而且当 $N = 10^{200}$ 时,对于 $\alpha \approx 8.42$ 同样的关系成立。函数 $N^{\epsilon(N)}$ 的增长阶大约在 $N^{1/\alpha}$ 和 $(\lg N)^\alpha$ 之间;但除非 N 大得出奇,所有这三种形式差不多是相同的。M. C. Wunderlich 所做的大量计算实验表明,算法 E 的一个改进型的执行要比我们估计的执行好得多[参考 *Lecture Notes in Math.* **751** (1979), 328~342];尽管当 $N = 10^{50}$ 时 $2\sqrt{\ln \ln N / \ln N} \approx .41$,当分解 $10^{13} \leq N \leq 10^{42}$ 范围内的几千个数时,他花费的运行时间约为 $N^{0.15}$ 。

算法 E 实质上是通过以 kN 代替 N 来开始它的分解的努力的,这种方法有一点怪(如果不是彻底的愚蠢)。“请原谅,你是否介意,我在尝试对你的数进行因子分解之前,先用 3 乘它。”尽管这样,它确是好的想法,因为 k 的某些值将使诸数 V 可能为更多的小数所整除,因此它们将更有可能在步骤 E3 中被完全分解;另一方面, k 的一个大值将使 V 个数更大,因此它们将较不可能完全分解。我们要通过明智地选择 k 来平衡这些趋势。例如考虑 V 被 5 的乘幂整除的可能性。在步骤 E3 中,我们有 $p^2 - kNQ^2 = (-1)^S V$,所以如果 5 整除 V ,我们有 $P^2 \equiv kNQ^2 \pmod{5}$ 。在这个同余式中, Q 不能是 5 的一个倍数,因为它与 P 互素,所以我们可以写 $(P/Q)^2 \equiv kN \pmod{5}$ 。如果假定 P 和 Q 是随机的互素整数,使得 24 种可能的数偶 $(P \bmod 5, Q \bmod 5) \neq (0, 0)$ 是同样可能的,则依赖于 $kN \bmod 5$ 是 0, 1, 2, 3 或 4, 5 整除 V 的概率是 $\frac{4}{24}, \frac{8}{24}, 0, 0$ 或 $\frac{8}{24}$ 。类似地,除非 kN 是 25 的倍数, 25 整除 V 的概率分别是 $0, \frac{40}{600}, 0, 0, \frac{40}{600}$ 。一般来说,给定使 $(kN)^{(p-1)/2} \bmod p = 1$ 的奇素数 p ,我们求得 V 为 p^e 的倍数的概率是 $2/(p^{e-1}(p+1))$;而且 p 整除 V 的平均次数是 $2p/(p^2 - 1)$ 。由 R. Schroepel 提议的这个分析提示, k 的最好选择是使

$$\sum_{j=1}^m f(p_j, kN) \log p_j - \frac{1}{2} \log k \quad (23)$$

取极大值,其中 f 是在习题 28 中定义的函数,因为这实质上是当我们达到步骤 E4 时 $\ln(\sqrt{N}/T)$ 的期望值。

当 k 和 m 都已选择好时,用算法 E 将得到最好的结果。 m 的适当选择只能通过实验检验做到,因为我们所做的渐近分析太粗糙了,尚不足以给出充分精确的信息,而且对于这个算法的各种各样的改进可能会产生预料不到的效果。例如,通过比较步骤 E3 同算法 A 我们可以作出一个重要的改进:每当我们发现 $T \bmod p_j \neq 0$ 和 $\lfloor T/p_j \rfloor \leq p_j$ 时, V 的因子分解即可停止,因为 T 将成为 1 或素数。如果 T 是大于 p_m 的素数(在此情况下它最多是 $p_m^2 + p_m - 1$),我们仍然可以输出 $(P, e_0, \dots,$

e_m, T), 因为已经得到一个完全的因子分解。算法的第二阶段只使用诸素数 T 至少已经出现两次的那些输出。这一修改的效果是产生更长的素数表, 而不增加因子分解的时间。Wunderlich 的实验指出, 当 N 处于 10^{40} 附近时, 利用这一改进, $m \approx 150$ 就工作得很好。

因为步骤 E3 是这个算法中最消耗时间的部分, Morrison, Brillhart 及 Schroeppel 已经提出了好几种方法, 以便在成功希望不大时中断这个步骤: (a) 每当 T 变为单精度值时, 仅当 $\lfloor T/p_j \rfloor > p_j$, 及 $3^{T-1} \bmod T \neq 1$ 时, 才继续进行。(b) 在找出小于 $\frac{1}{10} p_m$ 的因子之后, 如果 T 仍然大于 p_m^2 , 则放弃之。(c) 对于比如说 100 批左右连续的 V , 仅仅找出直到, 比如说 p_5 的因子, 以后才继续因子分解, 但只对每批中已经产生最小剩余的 T 的那些 V 进行。(在找出直到 p_5 的因子之前, 计算 $V \bmod p_1^{f_1} p_2^{f_2} p_3^{f_3} p_4^{f_4} p_5^{f_5}$ 是明智的, 其中诸 f 足够小使得 $p_1^{f_1} p_2^{f_2} p_3^{f_3} p_4^{f_4} p_5^{f_5}$ 能用单精度处理, 但又要足够大, 使得不可能出现 $V \bmod p_i^{f_i+1} = 0$ 。因此单精度剩余是 V 模五个小素数的值的特征。

关于算法 E 的输出中循环长度的估计, 见 H. C. Williams, *Math. Comp.* **36** (1981), 593~601。

*** 一个理论的上限** 从计算复杂性的观点看, 我们希望知道是否有任何因子分解的方法, 它的预期运行时间可以证明是 $O(N^{\epsilon(N)})$, 其中当 $N \rightarrow \infty$ 时, $\epsilon(N) \rightarrow 0$ 。我们已经看到算法 E 大概有这样一种特性, 但是, 似乎没有希望找到一个严格的证明, 因为连分数不是充分地有规律的。在这种意义下, 存在一个好的因子分解算法的头一个证明是由 John Dixon 于 1978 年发现的; 事实上, Dixon 证明, 考虑算法 E 的一个简化形式就足够了。在算法 E 的简化形式中去掉了连分数的工具, 但保留了 (18) 的基本思想。

Dixon 的方法 [*Math. Comp.* **36** (1981), 255~260] 简单地说是, 假设已知 N 至少有两个不同的素因子, 而且 N 不为头 m 个素数 p_1, p_2, \dots, p_m 所整除: 在 $0 < X < N$ 范围内选择一个随机整数 X , 并令 $V = X^2 \bmod N$ 。如果 $V = 0$, 则数 $\gcd(X, N)$ 是 N 的一个适当因子。否则, 和在步骤 E3 中一样, 找出 V 的所有小的素因子; 换句话说, 把 V 表达成形式

$$V = p_1^{e_1} \cdots p_m^{e_m} T \quad (24)$$

其中 T 不为头 m 个素数的任何一个所整除。如果 $T = 1$, 则这个算法如同在步骤 E4 中那样进行, 输出 (X, e_1, \dots, e_m) , 它表示 (19) 的一个解且 $e_0 = 0$ 。这个过程以 X 的新随机值继续下去, 直到有充分多输出使我们能通过习题 12 的方法发现 N 的一个因子。

为了分析这个算法, 我们要找出下列概率的界: (a) 一个随机的 X 产生一个输出的概率; (b) 在找出一个因子之前要求的大量输出的概率。设 $P(m, N)$ 是 (a) 的

概率,即当 X 随机地选择时 $T=1$ 的概率。在试验了 X 的 M 个值之后,平均说来,我们将得到 $MP(m, N)$ 个输出;而且输出的数目有一个二项式分布,所以标准差小于均值的平方根。概率(b)相当容易处理,因为习题 13 证明,这个算法需要多于 $m+k$ 个输出的概率小于 2^{-k} 。

习题 30 证明,当 $r = 2 \lfloor \log N / (2 \log p_m) \rfloor$ 时, $P(m, N) \geq m^r / (r! N)$, 所以如同在(22)中做过的那样,我们可以估计运行时间,但用 N 代替量 $2\sqrt{N}$ 。这一次我们选择

$$r = \sqrt{2 \ln N / \ln \ln N} + \theta$$

其中 $|\theta| \leq 1$ 且 r 是偶数,因此我们选择 m 使得

$$r = \ln N / \ln p_m + O(1/\log \log N)$$

这意味着

$$\begin{aligned} \ln p_m &= \sqrt{\frac{\ln N \ln \ln N}{2}} - \frac{\theta}{2} \ln \ln N + O(1) \\ \ln m = \ln \pi(p_m) &= \ln p_m - \ln \ln p_m + O(1/\log p_m) = \\ &= \sqrt{\frac{\ln N \ln \ln N}{2}} - \frac{\theta+1}{2} \ln \ln N + O(\log \log \log N) \\ \frac{m^r}{r! N} &= \exp(-\sqrt{2 \ln N \ln \ln N} + O(r \log \log \log N)) \end{aligned}$$

我们将选择 M 使得 $Mm^r/(r! N) \geq 4m$; 因此预期的输出个数 $MP(m, N)$ 至少将是 $4m$ 。算法运行时间的阶是 $Mm \log N$, 加上对于习题 12 的 $O(m^3)$ 步; 结果表明, $O(m^3)$ 小于 $Mm \log N$, 即

$$\exp(\sqrt{8(\ln N)(\ln \ln N)} + O((\log N)^{1/2}(\log \log N)^{-1/2}(\log \log \log N)))$$

这个方法找不到一个因子的概率小得可以忽略不计, 因为得到少于 $2m$ 个输出的概率至多是 $e^{-m/2}$ (见习题 31), 而由头 $2m$ 个输出 ($m \gg \ln N$) 中找不到因子的概率至多是 2^{-m} 。我们已经证明了下面 Dixon 原来定理的稍微增强的形式。

定理 D 有一个算法, 其运行时间是 $O(N^{c(N)})$, 其中 $c(N) = c \sqrt{\ln \ln N / \ln N}$ 且 c 是任何大于 $\sqrt{8}$ 的常数, 每当 N 至多有两个不同的素因子时, 这个算法以 $1 - O(1/N)$ 的概率找出 N 的一个非平凡的因子。 ■

其它方法 John M. Pollard 还曾提出了另一个因子分解的技术 [Proc. Cambridge Phil. Soc. 76 (1974), 521~528], 当 $p-1$ 没有大的素因子时, 他给出了一个实用的方法来发现 N 的素因子 p 。在算法 A 和 B 已经在很大的 N 运行太久时, 后一个算法 (见习题 19) 大概应是尝试的头一件事。

由 R. K. Guy 和 J. H. Conway 写的综述论文, *Congressus Numerantium* 16 (1976), 49~89, 给出了直到那时为止关于这方面的发展前景。Guy 指出: “在本世纪里, 如果不采用特殊形式, 某个人能够正规地分解大小为 10^{80} 的数, 那我会甚感惊讶。”在此后 20 年间他确实要大感惊讶许多次了。

从 Carl Pomerance 1981 年的二次筛方法 [参见 Lecture Notes in Comp. Sci. 209

(1985), 169~182]开始, 在 20 世纪 80 年代期间对于大数的因子分解取得了巨大的发展。后来, Hendrick Lenstra 设计了椭圆曲线方法 [Annals of Math. **126** (1987), 649~673], 它预期花费约 $\exp(\sqrt{(2+\epsilon)(\ln p)(\ln \ln p)})$ 个乘法来求出一个素因子 p 。这近似于当 $p \approx \sqrt{N}$ 时我们对算法 E 估计的运行时间的平方根, 而且当 N 有相对地小的素因子时, 它甚至变得更好些。关于这个方法的一个精彩解释已由 Joseph H. Silverman 和 John Tate 给出, *Rational Points on Elliptic Curves* (New York: Springer, 1992), 第 4 章。

John Pollard 于 1988 年又以另一个新技术回到这个领域, 它的这个技术已被称做数域筛, 有关这个方法的一系列的论文, 请见 *Lecture Notes in Math.* **1554** (1993), 它是当前分解极大整数的最佳方法。当 $N \rightarrow \infty$ 时预期它的运行时间的阶是

$$\exp((64/9 + \epsilon)^{1/3} (\ln N)^{1/3} (\ln \ln N)^{2/3}) \quad (25)$$

按照 A. K. Lenstra 的说法, 数域筛的调节好了的版本开始超过二次筛的调节好的版本的转折点出现于 $N \approx 10^{112}$ 。

关于这些新方法的细节超出本书的范围, 但是通过注意到对于形如 $2^{2^k} + 1$ 形式的未被分解的费马数被攻克的某些早期成功的故事, 我们对于它们的有效性可以获得一些概念。比如, 在大约花掉了 700 个工作站的空闲时间进行 4 个月的计算之后, 用数域筛求出了因子分解

$$2^{512} + 1 = 2424833 \cdot$$

$$7455602825647884208337395736200454918783366342657 \cdot p_{99}$$

[Lenstra, Lenstra, Manasse 和 Pollard, *Math. Comp.* **61** (1993), 319~349; **64** (1995), 1357]; 这里 p_{99} 表示一个 99 位的素数。下一个费马数有两倍于此的数字, 但在 1995 年 10 月 20 日它屈服于椭圆曲线方法:

$$2^{1024} + 1 = 45592577 \cdot 6487031809 \cdot$$

$$4659775785220018543264560743076778192897 \cdot p_{252}$$

[Richard Brent, *Math. Comp.* **68** (1999), 429~451。]事实上, 早在 1988 年, Brent 已经碰巧使用椭圆曲线方法解决了下一个情况:

$$2^{2048} + 1 = 319489 \cdot 974849 \cdot$$

$$167988556341760475137 \cdot 3560841906445833920513 \cdot p_{564}$$

除了一个之外所有素因子都小于 10^{22} , 因此椭圆曲线方法是赢家。

那 $2^{4096} + 1$ 如何呢? 现时, 这个数似乎还完全无法达到。它有 5 个小于 10^{16} 的因子, 但未被分解的余数有 1187 位十进数字。下一个情况 $2^{8192} + 1$ 有小于 10^{27} 的 4 个已知的因子 [Crandall 和 Fagin, *Math. Comp.* **62** (1994), 321; Brent Crandall, Dilcher 和 van Halewyn, 待发表] 以及一个巨大的未被分解的余数。

秘密因子 当 R. L. Rivest, A. Shamir 以及 L. Adleman 发现了对信息进行编码的一个方法时, 全球范围内对于因子分解问题的兴趣骤增。该方法可以只须知道一个很大的数 N 的因子就可明显地译码, 即使编码方法人人皆知。由于有相当数量

的世界最伟大的数学家都未能找到因子分解的有效方法,因此这个方案[CACM 21 (1978), 120~126]几乎肯定提供了在计算机网络中保护机密数据及进行通信的安全方法。

让我们想像称做 RSA 箱的一个小的电子设备,它的内存中有两个大的素数 p 和 q 。我们将假定 $p-1$ 和 $q-1$ 不为 3 所整除。RSA 箱以某种方式同一台计算机相连,而且它已告知计算机乘积 $N = pq$;然而,除非通过对 N 进行因子分解,没有人能发现 p 和 q 的值,因为 RSA 箱的设计很特殊。一旦有人企图撬开它,它就自动毁掉。换句话说,如果它被冲撞,或者如果它受到任何辐射(这辐射有可能改变或读出存在里边的数据),它就将抹去它的内存。而且 RSA 箱是充分可靠的,因而它绝不需要维护;如果出现一个紧急情况,或者如果它破旧了,我们把它扔掉再买一个即可。素因子 p 和 q 是由 RSA 箱根据本质上像宇宙射线这样的真正随机的现象的某个方案自动生成的。重要之点在于没有人知道 p 和 q ,甚至没有人或组织拥有或有可能访问这个 RSA 箱;没有办法贿赂或敲诈任何人或扣压任何人质以便发现 N 的因子。

为了传送一个秘密信息给 RSA 箱的主人,如果这个箱的乘积为 N ,那么你要把信息分成数 (x_1, x_2, \dots, x_k) 的序列,其中每个 x_i 是在 0 和 N 之间,然后传送数

$$(x_1^3 \bmod N, \dots, x_k^3 \bmod N)$$

如果知道 p 和 q , RSA 箱可以破译信息,因为它先计算一个数 $d < N$,使得 $3d \equiv 1 \pmod{(p-1)(q-1)}$;利用 4.6.3 小节的方法,它现在能在合理的时间内计算 $(x^3)^d \bmod N = x$ 。自然, RSA 箱的这个魔数 d 只有它自己知道;事实上, RSA 箱可以仅仅记住 d 而不是 p 和 q ,因为在计算了 N 之后,它惟一的责任是保护它的秘密,并取立方根模 N 。

如果 $x < \sqrt[3]{N}$,则上述的编码方案是低效的,因为 $x^3 \bmod N = x^3$,而且立方根容易找到。4.2.4 小节中前导数字的对数定律意味着 k 位信息 (x_1, \dots, x_k) 的前导位 x_1 将有 $1/3$ 的机会小于 $\sqrt[3]{N}$,所以这是一个要加以解决的问题。习题 32 给出了一个解决办法。

RSA 编码方案的安全性依赖于:如果不知道 N 的因子,则没有人能发现怎样来取立方根模 N 。也许这样的方法找不到,但是我们不能绝对肯定。至今能明确的是,所有通常发现立方根的方法都将失灵。例如,实质上,没有办法尝试把数 d 作为 N 的一个函数来计算;原因是,如果 d 已知或者事实上如果已知适当大小的任何数 m ,使得对于足够数量的 x , $x^m \bmod N = 1$ 成立,则我们可以在另外一些步之内找到 N 的因子(见习题 34)。于是,明显地或含蓄地以寻找这样一个 m 为基础的任何解法都不能比分解因子更好。

然而,某些预防措施是必要的,如果把同一个消息发送给在一个计算机网络上的三个不同的人处,则通过中国剩余定理某个知道 x^3 模 N_1, N_2 和 N_3 的人可以重新构造 $x^3 \bmod N_1 N_2 N_3 = x^3$,所以 x 将不再是一个秘密了。事实上,即使把一个带

有“时间标签”的消息 $(2^{\lfloor \lg t \rfloor} (x + t_i))^3 \bmod N_i$ 发送给 7 个不同的人,通过已知或可猜测的 t_i ,也可推导出 x 的值来(见习题 44)。因此某些密码专家建议用 $2^{16} + 1 = 65537$ 作为指数而不是用 3 来进行编码;这个指数是素数,而且 $x^{65537} \bmod N$ 的计算时间是 $x^3 \bmod N$ 的计算时间的 8.5 倍。[CCITT Recommendations Blue Book, (Geneva: International Telecommunication Union, 1989), Fascicle VIII. 8, Recommendation X.509, Annex C, 74 ~ 76。] Rivest, Shamir 和 Adleman 原来的建议是用 $x^a \bmod N$ 来对 x 进行编码,其中 a 是和 $\varphi(N)$ 互素的任何指数,而不就是 3;然而,实际上,我们喜欢的是编码比译码更快的指数。

为使 RSA 方案能行,数 p 和 q 不应当仅仅是“随机”素数。我们已经提及, $p-1$ 和 $q-1$ 不应为 3 所整除,因为要确保模 N 惟一立方根存在。另一个条件是, $p-1$ 应当有至少一个非常大的素因子, $q-1$ 也应如此;否则 N 就可以使用习题 19 的算法来进行因子分解。事实上,该算法实质上依赖于寻找具有下列性质的相当小的数 m : 对于它 $x^m \bmod N$ 经常等于 1,而且我们刚刚见到,这样一个 m 是危险的。当 $p-1$ 和 $q-1$ 有很大的素因子 p_1 和 q_1 时,习题 34 中的理论意味着, m 或者是 $p_1 q_1$ 的一个倍数(因此 m 将很难被发现)或者是 $x^m \equiv 1$ 小于 $1/p_1 q_1$ 的概率(因此 $x^m \bmod N$ 几乎永不为 1)。除了这个条件之外,我们不希望 p 和 q 彼此很接近,以免算法 D 能发现它们:事实上,我们不要比例 p/q 接近于一个简单的分数,否则算法 C 的 Lehman 推广就能发现它们。

生成 p 和 q 的下列过程几乎肯定是不可破的:由比如说在 10^{80} 和 10^{81} 之间的一个真正的随机数 p_0 开始,寻找大于 p_0 的头一个素数 p_1 ;这将要求检验大约 $\frac{1}{2} \ln p_0 \approx 90$ 个奇数,而在做了算法 P 的 50 次尝试之后,以大于 $1 - 2^{-100}$ 的概率来使 p_1 成为一个“可能的素数”将是充分的。然后,在比如说 10^{39} 和 10^{40} 之间选择另一个真正随机的数 p_2 ,寻找形如 $kp_1 + 1$ 的头一个素数 p ,其中 $k \geq p_2$, k 是偶数,而且 $k \equiv p_1 \pmod{3}$ 。在找到一个素数 p 之前,这将要求检验大约 $\frac{1}{3} \ln p_1 p_2 \approx 90$ 个数。这个素数 p 将有大约 120 位数字那么长,可以使用一个类似的构造来找出大约 130 位数长的素数 q 。为增加安全性,可能还值得建议,应检查确认 $p+1$ 和 $q+1$ 都不完全由相当小的因子组成(请见习题 20)。其数量级将是大约 10^{250} 的乘积 $N = pq$,现在满足我们所有的要求。现时这样一个 N 可以被分解因子是不可想像的。

例如,假设我们知道有一个方法可以在 $N^{0.1}$ ms 之内分解一个 250 位数字的数 N ,这相当于 10^{25} ms,而一年中仅有 31 556 952 000 000 ms,所以我们将需要多于 3×10^{11} 年的中央处理器时间来完成这个因子分解。即使一个政府部门购买 100 亿台计算机而且把它们全都用来解决这个问题,在它们当中的一台可以把 N 分解成因子之前,那也将花费 31 年时间。况且政府购买这么多专门的机器总会走漏风声,那样人们将开始使用 300 位数字的 N 。

由于人人都知道 $x \mapsto x^3 \bmod N$ 的编码方法,因此这个码除仅可由 RSA 箱分解开之外,还有另外的优点。这样的“公钥”系统首先是由 W. Diffie 和 M. E. Hellman 在 *IEEE Trans. IT-22* (1976), 644 ~ 654 上发表的。下面举例说明当编码方法是公开的时可以做什么。假设 Alice 要用电子邮件同 Bob 通信,而且假设他们俩都想要信件是签了名的,使得接收者都能确信没有其他人伪造任何消息。设 $E_A(M)$ 是传送给 Alice 的消息 M 的编码函数,设 $D_A(M)$ 是由 Alice 的 RSA 箱完成的译码,并设 $E_B(M), D_B(M)$ 是 Bob 的 RSA 箱的相应的编码和译码函数,则 Alice 可以发送一个签了名的消息,办法是先签上她自己的名字和对某些机密的消息填上日期。然后用她的机器计算 $D_B(M)$,并把 $E_B(D_A(M))$ 传送给 Bob。当 Bob 得到这个消息后,他的 RSA 箱把它转换成 $D_A(M)$,而且他知道 E_A ,所以他可以计算 $M = E_A(D_A(M))$ 。这将使他相信,消息确实来自 Alice;其他人不可能发送消息 $D_A(M)$ 。(好,Bob 本人现在知道 $D_A(M)$,所以他可以通过把 $E_X(D_A(M))$ 传送给 Xavier 来假冒 Alice,为了挫败任何这样有意的假冒, M 的内容应明确地指出它仅仅供 Bob 观阅。)

我们可能要问,Alice 和 Bob 怎么知道彼此的编码函数 E_A 和 E_B 呢?这是不能简单地存在一个公开的文件中的,因为 Charlie 可能破译这个文件,用他自己计算过的另一个 N 代替它;然后 Charlie 可能在 Alice 和 Bob 发现丢失了什么之前就偷偷摸摸地截获住秘密的消息并予译码。解决方法是在一个特殊的公开目录中保持乘积数 N_A 和 N_B ,这个目录中有它自己的 RSA 箱和它自己广泛公开化的乘积数 N_D 。当 Alice 要知道如何同 Bob 通信时,她向目录问 Bob 的乘积数;目录计算机向她传送一个签了名的消息,该消息给出 N_B 的值。没有人能伪造这样一个消息,所以它必定是真实的。

Michael Rabin 已经提出了 RSA 箱方案的有趣的替代办法[MIT Lab. for Comp. Sci., Report TR-212 (1979)],他提出用函数 $x^2 \bmod N$ 而不是 $x^3 \bmod N$ 进行编码。在这种情况下编码机制可以称为一个 SQRT 箱,它送回 4 个不同的消息;原因是 4 个不同的数有相同的平方模 N ,即 $x, -x, fx \bmod N$ 及 $(-fx) \bmod N$,其中

$$f = (p^{q-1} - q^{p-1}) \bmod N。$$

如果我们预先约定 x 是偶数,或者 $x < \frac{1}{2}N$,则二义性落到两个消息当中,而假定两个当中仅有一个有意义。如同在习题 35 中所示,二义性事实上可以完全消除。Rabin 的方案有重要的性质,它可以证明下列结论:求平方根模 N 和因子分解 $N = pq$ 一样难,因为当 x 随机地选择时通过取 $x^2 \bmod N$ 的平方根,我们有 1/2 的概率找到一个 y 值,使得

$$x^2 \equiv y^2, \quad \text{且 } x \not\equiv \pm y$$

然后 $\gcd(x - y, N) = p$ 或 q 。然而,这个系统有一个致命的缺点,而它在 RSA 方案中似乎不出现(见习题 33):任何访问 SQRT 箱的人都能容易地确定它的 N 的因子。这不仅使不忠实的雇员可以作弊或进行敲诈,它也允许人们发现他们的 p 和 q ,在这之后他们可能断言,他们在某个传送的文件上的“签字”是伪造的。显然,秘密通

信的目标导致了微妙的问题,它十分不同于我们在算法设计和分析中通常所面临的那些问题。

已知的最大素数 在本书的其它地方,我们已经讨论了若干计算方法,它们都要求使用很大的素数,而刚刚介绍的一些技术可比较容易地用来发现比如说多达25位或小一点的素数。表2示出了小于典型计算机字长的10个最大素数(一些其它有用的素数见习题3.2.1.2-22和4.6.4-57的答案)。

表2 有用的素数表

N	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
2^{15}	19	49	51	55	61	75	81	115	121	135
2^{16}	15	17	39	57	87	89	99	113	117	123
2^{17}	1	9	13	31	49	61	63	85	91	99
2^{18}	5	11	17	23	33	35	41	65	75	93
2^{19}	1	19	27	31	45	57	67	69	85	87
2^{20}	3	5	17	27	59	69	129	143	153	185
2^{21}	9	19	21	55	61	69	105	111	121	129
2^{22}	3	17	27	33	57	87	105	113	117	123
2^{23}	15	21	27	37	61	69	135	147	157	159
2^{24}	3	17	33	63	75	77	89	95	117	167
2^{25}	39	49	61	85	91	115	141	159	165	183
2^{26}	5	27	45	87	101	107	111	117	125	135
2^{27}	39	79	111	115	135	187	199	219	231	235
2^{28}	57	89	95	119	125	143	165	183	213	273
2^{29}	3	33	43	63	73	75	93	99	121	133
2^{30}	35	41	83	101	105	107	135	153	161	173
2^{31}	1	19	61	69	85	99	105	151	159	171
2^{32}	5	17	65	99	107	135	153	185	209	267
2^{33}	9	25	49	79	105	285	301	303	321	355
2^{34}	41	77	113	131	143	165	185	207	227	281
2^{35}	31	49	61	69	79	121	141	247	309	325
2^{36}	5	17	23	65	117	137	159	173	189	233
2^{37}	25	31	45	69	123	141	199	201	351	375
2^{38}	45	87	107	131	153	185	191	227	231	257
2^{39}	7	19	67	91	135	165	219	231	241	301
2^{40}	87	167	195	203	213	285	293	299	389	437
2^{41}	21	31	55	63	73	75	91	111	133	139
2^{42}	11	17	33	53	65	143	161	165	215	227
2^{43}	57	67	117	175	255	267	291	309	319	369
2^{44}	17	117	119	129	143	149	287	327	359	377
2^{45}	55	69	81	93	121	133	139	159	193	229
2^{46}	21	57	63	77	167	197	237	287	305	311
2^{47}	115	127	147	279	297	339	435	541	619	649
2^{48}	59	65	89	93	147	165	189	233	243	257

(续)

N	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
2^{40}	55	99	225	427	517	607	649	687	861	871
2^{60}	93	107	173	179	257	279	369	395	399	453
2^{63}	25	165	259	301	375	387	391	409	457	471
2^{64}	59	83	95	179	189	257	279	323	353	363
10^6	17	21	39	41	47	69	83	93	117	137
10^7	9	27	29	57	63	69	71	93	99	111
10^8	11	29	41	59	69	153	161	173	179	213
10^9	63	71	107	117	203	239	243	249	261	267
10^{10}	33	57	71	119	149	167	183	213	219	231
10^{11}	23	53	57	93	129	149	167	171	179	231
10^{12}	11	39	41	63	101	123	137	143	153	233
10^{16}	63	83	113	149	183	191	329	357	359	369
小于 N 的 10 个大的素数是 $N - a_1, \dots, N - a_{10}$										

实际上已经知道一些大得多的特殊形式的素数,而且有时候求尽可能大的素数是重要的。因此,在本节的末尾让我们来研究一下发现那些已知最大素数的有趣方式。这样的素数都有 $2^n - 1$ 的形式,其中 n 是各种特殊的值,因此它们特别适合于二进制计算机的某些应用。

除非 n 为素数,否则,形如 $2^n - 1$ 的数不可能是素数,因为 $2^{uv} - 1$ 可为 $2^u - 1$ 整除。1644 年,Marin Mersenne(梅森)指出,实际上对于 $p = 2, 3, 5, 7, 13, 17, 19, 31, 67, 127, 257$, 数 $2^p - 1$ 都是素数,而对于其它小于 257 的 p , 则都不是素数。这使得他的同时代的人大为惊奇。(这个命题是在他的 *Cogitata Physico-Mathematica* 的前言中联系完全数的讨论而出现的。奇怪的是,他还作了如下的注释:“要问一个给定的 15 或 20 位数字是否素数,无论用什么已知的办法,即使用尽全部时间,也不足以测定。”)梅森早年经常同费马、笛卡尔以及其他人就类似的课题进行通信联系,但他对于这一论断没有给出证明,而且过了 200 多年之后也没有人知道他正确与否。欧拉在用了几年时间尝试证明这一事实屡遭失败之后,终于在 1772 年证明了 $2^{31} - 1$ 是素数。大约 100 年后,É. Lucas 发现 $2^{127} - 1$ 是素数,但 $2^{67} - 1$ 是有问题的;因此,梅森的论述不完全准确。然后 1883 年, I. M. Pervushin 证明了 $2^{61} - 1$ 是素数[参考 *Istoriko-Mat. Issledovaniya* 6 (1953), 559], 这增加了人们的推测,以为梅森只是出了抄写错误,即把 61 写成了 67。最终梅森命题的其它错误被发现了;如同某些早期的学者们所曾经猜测的那样, R. E. Powers [AMM 18(1911), 195] 发现 $2^{89} - 1$ 是素数;而且过了 3 年之后,他证明 $2^{107} - 1$ 也是素数。M. Kraitchik 于 1922 年证明, $2^{257} - 1$ 不是素数[参见他的 *Recherches sur la Théorie des Nombres* (Paris: 1924), 21]; 计算的错误可能已经潜入他的计算当中,但他的结论已证实是正确的。

形如 $2^p - 1$ 的数现在被称为梅森数,而且已经知道,对于 p 等于

2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281,
3217, 4253, 4423, 9689, 9941, 11213, 19937, 21701, 23209, 44497, 86243,

$$110503, 132049, 216091, 756839, 859433, 1257787, 1398269, 2976221 \quad (26)$$

的梅森素数。在底部一行的大多数数是由 David Slowinski 和他的助手们在调试新的超级计算机时发现的[请见 *J. Recreational Math.* **11** (1979)258~261]。在 20 世纪 90 年代他和 Paul Gage 合作发现了 756839, 859433 和 1257787。然而, 两个当前最大的指数 1398269 和 2976221 分别是由 Joel Armengaud 和 Gordon Spence 使用不再流行的个人计算机找到的; 他们使用由 George Woltman 所编制的一个程序。此人于 1996 年发起了伟大的互联网梅森素数寻找工程(GIMPS), 注意在(26)中未出现素数 $8191 = 2^{13} - 1$; 梅森已经指出过, $2^{8191} - 1$ 是素数, 而且其他人也已猜测, 任何梅森素数也许还可用在指数中。

对于大的素数的查找还未被系统化, 因为人们一般地都试图建立一个难以打破的世界记录而不愿对较小的指数花费时间; 例如, 在 1983 年证实了 $2^{132049} - 1$ 是素数, 在 1984 年证实了 $2^{216091} - 1$, 但是在 1988 年之前还未发现 $2^{110503} - 1$ 。因此小于 $2^{2976221} - 1$ 的一个或多个未知的梅森数仍然可能存在。(根据 Woltman, 到 1997 年 5 月 26 日为止, 直到 1 000 000 的所有指数都已被检验, 而他的志愿者们正在填补剩下的空隙。)

由于 $2^{2976221} - 1$ 接近有 900 000 位十进数字, 显然已经使用了某些特殊的技术来证明这样的数是素数。(事实上, 1996 年 4 月 12 日, 对 $2^{1257787} - 1$ 所做的初始验证, 在 Cray T94 上花费了少于 8.3 个小时的时间。1997 年 8 月对于 $2^{2976221} - 1$ 的初始验证, 在一台 100MHz 的奔腾计算机上花费了 15 天。)关于检验一个给定的梅森数 $2^p - 1$ 的素性的有效方法, 首先是由 É. Lucas 给出的 [*Amer. J. Math.* **1** (1878), 184~239, 289~321, 特别是 316 页], 并由 D. H. Lehmer 作了改进 [*Annals of Math.* **31** (1930), 419~448, 特别是 443 页]。Lucas-Lehmer 的检验, 是当 $n+1$ 的因子已知时, 检验 n 的素性现在所用方法的一个特殊情况, 其内容如下:

定理 I 设 q 是奇素数, 并通过规则

$$L_0 = 4, \quad L_{n+1} = (L_n^2 - 2) \bmod (2^q - 1) \quad (27)$$

定义序列 $\langle L_n \rangle$ 。于是 $2^q - 1$ 是素数, 当且仅当 $L_{q-2} = 0$

例如, $2^3 - 1$ 是素数, 因为 $L_1 = (4^2 - 2) \bmod 7 = 0$ 。这个测试特别适合于二进制计算机, 因为计算 $\bmod (2^q - 1)$ 是非常方便的; 参考 4.3.2 小节。习题 4.3.2-14 说明当 q 极端地大时如何节省时间。

证明 通过考察一些本身有独立价值的递推序列的若干特性, 只须使用非常简单的数论原理就能证明定理。考虑由规则

$$\begin{aligned} U_0 &= 0, & U_1 &= 1, & U_{n+1} &= 4U_n - U_{n-1} \\ V_0 &= 2, & V_1 &= 4, & V_{n+1} &= 4V_n - V_{n-1} \end{aligned} \quad (28)$$

定义的序列 $\langle U_n \rangle$ 和 $\langle V_n \rangle$ 。用归纳法, 容易证明下列等式

$$V_n = U_{n+1} - U_{n-1} \quad (29)$$

$$U_n = ((2 + \sqrt{3})^n - (2 - \sqrt{3})^n) / \sqrt{12} \quad (30)$$

$$V_n = (2 + \sqrt{3})^n + (2 - \sqrt{3})^n \quad (31)$$

$$U_{m+n} = U_m U_{n+1} - U_{m-1} U_n \quad (32)$$

当 p 是素数且 $e \geq 1$ 时, 我们现在证明一个辅助性结果:

$$\text{如果 } U_n \equiv 0 \pmod{p^e}, \text{ 则 } U_{np} \equiv 0 \pmod{p^{e+1}} \quad (33)$$

这可以从习题 3.2.2-11 的更一般的考虑得出, 但也可以给出序列 (28) 的直接证明。假设 $U_n = bp^e$, $U_{n+1} = a$ 。由 (32) 和 (28), $U_{2n} = bp^e(2a - 4bp^e) \equiv 2aU_n \pmod{p^{e+1}}$, 同时我们有 $U_{2n+1} = U_{n+1}^2 - U_n^2 \equiv a^2$ 。类似地, $U_{3n} = U_{2n+1}U_n - U_{2n}U_{n-1} \equiv 3a^2U_n$ 且 $U_{3n+1} = U_{2n+1}U_{n+1} - U_{2n}U_n \equiv a^3$ 。一般地,

$$U_{kn} \equiv ka^{k-1}U_n, \quad U_{kn+1} \equiv a^k \pmod{p^{e+1}}$$

所以如果我们取 $k = p$, 则 (33) 得证。

从公式 (30) 和 (31), 用二项式定理展开 $(2 \pm \sqrt{3})^n$, 我们可以得到 U_n 和 V_n 的其它表达式:

$$U_n = \sum_k \binom{n}{2k+1} 2^{n-2k-1} 3^k, \quad V_n = \sum_k \binom{n}{2k} 2^{n-2k+1} 3^k \quad (34)$$

现在如果置 $n = p$, 其中 p 为一奇素数, 而且如果使用这样一个事实, 即除当 $k = 0$ 或 $k = p$ 外, $\binom{p}{k}$ 是 p 的倍数, 则我们求得

$$U_p \equiv 3^{(p-1)/2}, \quad V_p \equiv 4 \pmod{p} \quad (35)$$

如果 $p \neq 3$, 则费马定理告诉我们 $3^{p-1} \equiv 1$; 因此 $(3^{(p-1)/2} - 1) \times (3^{(p-1)/2} + 1) \equiv 0$, 且 $3^{(p-1)/2} \equiv \pm 1$ 。当 $U_p \equiv -1$ 时, 我们有 $U_{p-1} = 4U_p - U_{p+1} = 4U_p + V_p - U_{p+1} \equiv -U_{p+1}$; 因此 $U_{p-1} \pmod{p} = 0$ 。当 $U_p \equiv +1$ 时, 我们有 $U_{p-1} = 4U_p - U_{p+1} = 4U_p - V_p - U_{p+1} \equiv -U_{p+1}$; 因此 $U_{p-1} \pmod{p} = 0$ 。我们由此证明了对于所有的素数 p , 有一整数 $\epsilon(p)$, 使得

$$U_{p+\epsilon(p)} \pmod{p} = 0, \quad |\epsilon(p)| \leq 1 \quad (36)$$

现在如果 N 是任意正整数, 而且如果 $m = m(N)$ 是使得 $U_{m(N)} \pmod{N} = 0$ 的最小正整数, 则有

$$U_n \pmod{N} = 0 \quad \text{当且仅当} \quad n \text{ 是 } m(N) \text{ 的倍数} \quad (37)$$

(这个数 $m(N)$ 称做 N 在序列中出现的秩。) 为证明 (37), 注意到序列 $U_m, U_{m+1}, U_{m+2}, \dots \pmod{N}$ 同余于 aU_0, aU_1, aU_2, \dots , 其中 $a = U_{m+1} \pmod{N}$ 与 N 互素, 因为 $\gcd(U_n, U_{n+1}) = 1$ 。

通过这些预备知识, 我们做好了证明定理 L 的准备。由 (27) 和归纳法

$$L_n = V_{2^n} \pmod{2^q - 1} \quad (38)$$

而且, 恒等式 $2U_{n+1} = 4U_n + V_n$ 意味着 $\gcd(U_n, V_n) \leq 2$, 因为 U_n 和 V_n 的任何公因子必整除 U_n 和 $2U_{n+1}$, 而 $U_n \perp U_{n+1}$ 。所以 U_n 和 V_n 没有公共的奇因子, 而且

如果 $L_{q-2} = 0$, 则必有

$$U_{2^{q-1}} = U_{2^{q-2}} V_{2^{q-2}} \equiv 0 \pmod{2^q - 1}$$

$$U_{2^{q-2}} \not\equiv 0 \pmod{2^q - 1}$$

现在如果 $m = m(2^q - 1)$ 是 $2^q - 1$ 出现的秩, 则 m 必然是 2^{q-1} 的, 但非 2^{q-2} 的一个因子; 于是 $m = 2^{q-1}$ 。我们还将证明 $n = 2^q - 1$ 因此是素数: 设 n 的因子分解是 $p_1^{e_1} \cdots p_r^{e_r}$ 。所有素数 p_i 皆大于 3, 因为 n 是奇数且同余于 $(-1)^q - 1 = -2 \pmod{3}$ 。由 (33), (36) 和 (37), 我们知道 $U_i \equiv 0 \pmod{2^q - 1}$, 其中

$$t = \text{lcm}(p_1^{e_1-1}(p_1 + \epsilon_1), \dots, p_r^{e_r-1}(p_r + \epsilon_r))$$

而且每个 ϵ_j 是 ± 1 。因此 t 是 $m = 2^{q-1}$ 的倍数。设 $n_0 = \prod_{j=1}^r p_j^{e_j-1}(p_j + \epsilon_j)$; 我们有 $n_0 \leq \prod_{j=1}^r p_j^{e_j-1} \left(p_j + \frac{1}{5} p_j \right) = \left(\frac{6}{5} \right)^r n$ 。还由于 $p_j + \epsilon_j$ 是偶数, 所以 $t \leq n_0 / 2^{r-1}$, 这是因为, 每次取两个偶数的最小公倍数时, 就失去了一个因子 2。把这些结果结合在一起, 我们就有 $m \leq t \leq 2 \left(\frac{3}{5} \right)^r n < 4 \left(\frac{3}{5} \right)^r m < 3m$; 因此 $r \leq 2$ 且 $t = m$ 或 $t = 2m$, 即 2 的乘幂。因此 $e_1 = 1, e_r = 1$, 而且如果 n 不是素数, 则必定有 $n = 2^q - 1 = (2^k + 1)(2^l - 1)$, 其中 $2^k + 1$ 和 $2^l - 1$ 是素数。但当 q 是奇数时, 后者显然不可能, 所以 n 为素数。

反之, 假设 $n = 2^q - 1$ 为素数, 我们必须证明 $V_{2^{q-2}} \equiv 0 \pmod{n}$ 。为此目的, 只须证明 $V_{2^{q-1}} \equiv -2 \pmod{n}$ 就行了, 因为 $V_{2^{q-1}} = (V_{2^{q-2}})^2 - 2$ 。现在

$$\begin{aligned} V_{2^{q-1}} &= ((\sqrt{2} + \sqrt{6})/2)^{n+1} + ((\sqrt{2} - \sqrt{6})/2)^{n+1} = \\ &= 2^{-n} \sum_k \binom{n+1}{2k} \sqrt{2}^{n+1-2k} \sqrt{6}^{2k} = 2^{(1-n)/2} \sum_k \binom{n+1}{2k} 3^k \end{aligned}$$

由于 n 是奇素数, 故除了当 $2k = 0$ 和 $2k = n+1$ 外,

$$\binom{n+1}{2k} = \binom{n}{2k} + \binom{n}{2k-1}$$

可为 n 整除; 因此

$$2^{(n-1)/2} V_{2^{q-1}} \equiv 1 + 3^{(n+1)/2} \pmod{n}$$

这里 $2 \equiv (2^{(q+1)/2})^2$, 所以由费马定理 $2^{(n-1)/2} \equiv (2^{(q+1)/2})^{(n-1)} \equiv 1$ 。最后, 由于二次互反律 (参考习题 23) 的简单情况, $3^{(n-1)/2} \equiv -1$, 因为 $n \bmod 3 = 1$ 和 $n \bmod 4 = 3$ 。这意味着 $V_{2^{q-1}} \equiv -2$, 所以如所希那样必有 $V_{2^{q-2}} \equiv 0$ 。 ■

一位其著作现在保存在意大利图书馆的佚名作者在 1460 年发现了 $2^{17} - 1$ 和 $2^{19} - 1$ 是素数 [请见 E. Picutti, *Historia Math.* **16** (1989), 123-136]。自那以后, 世界上明确知道的最大素数几乎一直是梅森素数。不过情况大概要起变化, 因为梅森素数越来越难找, 还因为习题 27 给出了对其它形式的素数的有效检验方法。

习 题

1. [10] 如果算法 A 中试验因子的序列 d_0, d_1, d_2, \dots 包含一个非素数的数, 试问它为什么不可能出现于输出中?

2. [15] 如果已知对算法 A 的输入 N 等于 3 或更大, 问步骤 A2 可否消去?

3. [M20] 证明有一个具有下列性质的数 P : 如果 $1000 \leq n \leq 1000000$, 则当且仅当 $\gcd(n, P) = 1$ 时 n 为素数。

4. [M29] 在习题 3.1-7 和 1.2.11.3 小节的记法下, 证明使得 $X_n = X_{\ell(n)-1}$ 的最小的 n 的平均值介于 $1.5Q(m) - 0.5$ 和 $1.625Q(m) - 0.5$ 之间。

5. [21] 当取模数 3, 5, 7, 8 和 11 时, 用费马方法(算法 D)并用手算来求 11111 的因子。

6. [M24] 如果 p 为一奇素数, 且 N 不是 p 的倍数, 证明使得 $0 \leq x < p$ 和 $x^2 - N \equiv y^2 \pmod{p}$ 有一个解 y 的整数 x 的个数等于 $(p \pm 1)/2$ 。

7. [25] 当模 m_i 的表项不能恰好填满整数个存储字时, 讨论在一台二进计算机上对算法 D 的筛进行编程的问题。

► 8. [23] (Eratosthenes 筛, 公元前 3 世纪) 下列过程显然能发现小于一个给定整数 N 的所有奇素数, 因为它筛去所有的非素数; 从所有 1 和 N 之间的奇数开始; 然后对于 $k = 2, 3, 4, \dots$ 直到达到使 $p_k^2 > N$ 的一个素数 p_k , 逐个地去掉第 k 个素数 p_k 的倍数 $p_k^2, p_k(p_k + 2), p_k(p_k + 4), \dots$

试说明如何把刚才描述的过程编成一个能直接在计算机上进行有效计算且不使用乘法的算法。

9. [M25] 设 n 是一个奇数, $n \geq 3$ 。证明如果定理 3.2.1.2B 的数 $\lambda(n)$ 是 $n-1$ 的因子但不等于 $n-1$, 则 n 必有 $p_1 p_2 \cdots p_t$ 的形式, 其中诸 p 是不同的素数而且 $t \geq 3$ 。

► 10. [M26] (John Selfridge) 证明, 如果对于 $n-1$ 的每个素因子 p , 都有一个数 x_p 使得 $x_p^{(n-1)/p} \bmod n \neq 1$ 但 $x_p^{n-1} \bmod n = 1$, 则 n 为素数。

11. [M20] 当 $n = 197209, k = 5, m = 1$ 时, 算法 E 给出什么输出? [提示: $\sqrt{5 \cdot 197209} = 992 + //1,495,2,495,1,1984//$ 。]

► 12. [M28] 试设计一个算法, 它使用算法 E 的输出求 N 的一个真因子, 假定算法 E 已经产生足够的输出以推导 (18) 的一个解。

13. [HM25] (J. D. Dixon) 证明每当习题 12 的算法给出一个解 (x, e_0, \dots, e_m) , 这个解的指数同以前的解的指数模 2 线性相关, 且当 n 有 d 个不同的素因子, x 随机地选择时, 找不到一个因子分解的概率是 2^{1-d} 。

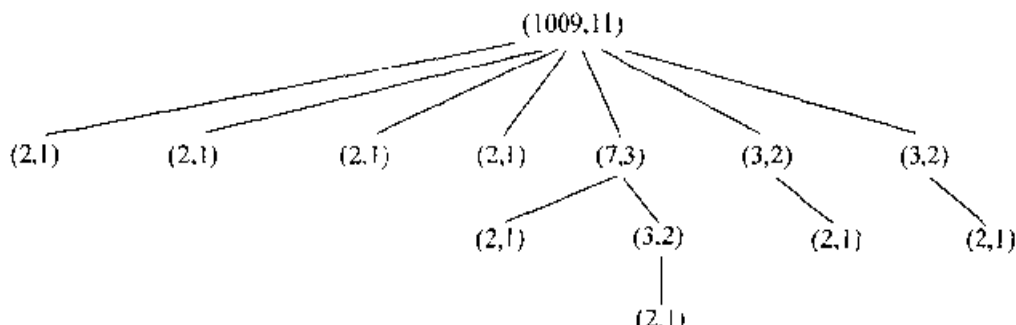
14. [M20] 证明算法 E 的步骤 E3 中的数 T 不可能是奇素数 p 的倍数, 其中 $(kN)^{(p-1)/2} \bmod p > 1$ 。

► 15. [M34] (Lucas 和 Lehmer) 设 P, Q 是互素的整数, 而且设 $U_0 = 0, U_1 = 1$, 且对于 $n \geq 1$, $U_{n+1} = PU_n - QU_{n-1}$ 。证明: 如果 N 是与 $2P^2 - 8Q$ 互素的正整数, 而且如果 $U_{N+1} \bmod N = 0$, 同时对于每个整除 $N+1$ 的素数 p , $U_{(N+1)/p} \bmod N \neq 0$, 则 N 是素数(当已知 $N+1$ 的因子而不是 $N-1$ 的因子时, 这给出了对素性的一个测试。如同在习题 4.6.3-26 中一样, U_m 的值可在 $O(\log m)$ 个步骤内计算出来。[提示: 见定理 L 的证明。]

16. [M50] 有无限多个梅森素数吗?

17. [M25] (V. R. Pratt) 用费马定理的逆来进行素性的一个完备的证明, 采取树的形式。这树的节点有 (q, x) 的形式, 其中 q 和 x 是满足下列算术条件的正整数。(i) 如果 $(q_1, x_1), \dots$,

(q_i, x_i) 是 (q, x) 的儿子, 则 $q = q_1 \cdots q_i + 1$ 。[特别是如果 (q, x) 没有儿子, 则 $q = 2$ 。] (ii) 如果 (r, y) 是 (q, x) 的儿子, 则 $x^{q-1} \not\equiv 1 \pmod{q}$ 。 (iii) 对于每个节点 (q, x) 有 $x^{q-1} \equiv 1 \pmod{q}$ 。由这些条件得出, 对于所有节点 (q, x) , q 是素数且 x 是一个模 p 原根。[例如, 树



显示, 1009 是素数。]证明具有根 (q, x) 的这样一棵树至多有 $f(q)$ 个节点, 其中 f 是一个增长很慢的函数。

►18. [HM23] 仿照正文中对 (6) 的推导给出一个 (7) 的启发式证明。什么是 $p_{i-1} \leq \sqrt{p_i}$ 的近似概率?

►19. [M25] (J. M. Pollard) 说明怎样来计算一个数 M , 它可被所有素数 p 所整除, 且使 $p-1$ 是某个给定的数 D [提示: 考虑形如 $a^n - 1$ 的数] 的一个因子。这样一个 M 在因子分解中是有用的, 因为通过计算 $\gcd(M, N)$, 我们可以发现 N 的一个因子。试把这个思想推广为一个有效的方法, 当 $p-1$ 的所有素因子中, 除至多一个素因子小于 10^5 外, 其它都小于 10^3 时, 这个方法能以很高的概率发现一个给定的大数 N 的诸素因子 p 。[例如, 通过这个方法将发现整除 (15) 的第二个最大的素数, 因为它是 $1 + 2^4 \cdot 5^3 \cdot 67 \cdot 107 \cdot 199 \cdot 41231$ 。]

20. [M40] 在以 $p+1$ 代替 $p-1$ 的情况下, 考虑习题 19。

21. [M49] (R. K. Guy) 设 $m(p)$ 是算法 B 为找出素因子 p 所需的迭代次数。当 $p \rightarrow \infty$ 时, $m(p) = O(\sqrt{p \log p})$ 吗?

►22. [M30] (M. O. Rabin) 给定 n , 设 p_n 是算法 P 猜错的概率。证明对所有 n , $p_n < \frac{1}{4}$ 。

23. [M35] 对所有整数 $p \geq 0$, 及所有奇整数 $q > 1$, 当 q 为素数时, 雅可比符号 $\left(\frac{p}{q}\right)$ 通过规则 $\left(\frac{p}{q}\right) \equiv p^{(q-1)/2} \pmod{q}$ 定义为 $-1, 0$ 或 $+1$; 当 q 是 t 个素数的乘积 $q_1 \cdots q_t$ (不必不同) 时, $\left(\frac{p}{q}\right) = \left(\frac{p}{q_1}\right) \cdots \left(\frac{p}{q_t}\right)$ 。这样它推广了习题 1.2.4-47 的符号。

a) 证明 $\left(\frac{p}{q}\right)$ 满足下列关系, 因此它可被有效地计算: $\left(\frac{0}{q}\right) = 0$; $\left(\frac{1}{q}\right) = 1$; $\left(\frac{p}{q}\right) = \left(\frac{p \bmod q}{q}\right)$; $\left(\frac{2}{q}\right) = (-1)^{(q^2-1)/8}$; $\left(\frac{pp'}{q}\right) = \left(\frac{p}{q}\right) \left(\frac{p'}{q}\right)$; 如果 p 和 q 都是奇数, 则 $\left(\frac{p}{q}\right) = (-1)^{(p-1)(q-1)/4} \left(\frac{q}{p}\right)$ 。[后一定律是把 $\left(\frac{p}{q}\right)$ 的计算归结为 $\left(\frac{q}{p}\right)$ 的计算的互反关系, 当 p 和 q 皆为素数时, 它已在习题 1.2.4-47(d) 中被证明, 所以在该特殊情况下你可以假定它的正确性。]

b) (Solovay 和 Strassen) 证明如果 n 是奇数但不是素数, 使得 $1 \leq x < n$ 和 $0 \neq \left(\frac{x}{n}\right) \equiv x^{(n-1)/2} \pmod{n}$ 的整数 x 的个数至多是 $\frac{1}{2} \varphi(n)$ 。(因此, 对所有固定的 n , 下列的检验过程以

至少 $\frac{1}{2}$ 的概率正确地确定 n 是否素数:“随机地生成满足 $1 \leq x < n$ 的 x , 如果 $0 \neq \left(\frac{x}{n}\right) \equiv x^{(n-1)/2} \pmod{n}$, 则说 n 可能为素数, 否则说 n 肯定不是素数。”

c) (L. Monier) 证明如果 n 和 x 是使得算法 P 作出结论“ n 可能为素数”的两个数, 则 $0 \neq \left(\frac{x}{n}\right) \equiv x^{(n-1)/2} \pmod{n}$ 。[因此算法 P 总是优越于 b) 中的检验。]

►24. [M25] (L. Adleman) 当 $n > 1$ 和 $r > 1$ 是整数, n 为素数时, 我们说 n “通过算法 P 的 x 检验”, 如果 $x \bmod n = 0$ 或者如果步骤 P2~P5 导致 n 可能是素数的结论的话。证明对于任何 n , 存在一组正整数 $x_1, \dots, x_m \leq N$ 及 $m \leq \lg N$, 使得在 $1 < n \leq N$ 范围内的一个奇正整数是素数, 当且仅当对于 $x = x_1 \bmod n, \dots, x = x_m \bmod n$, 它通过算法 P 的 x 检验。因此关于素性的概率检验原则上可以转换成一个不靠概率的有效检验(你不需要说明怎样有效地计算 x_i ; 只须证明它们存在)。

25. [HM41] (B. Riemann) 证明

$$\pi(x) + \frac{\pi(x^{1/2})}{2} + \frac{\pi(x^{1/3})}{3} + \dots = \int_2^x \frac{dt}{\ln t} - 2 \sum' \int_{-\infty}^{\sigma} \frac{e^{(t+i\tau)\ln x} dt}{t+i\tau} + O(1)$$

其中求和对所有使得 $\tau > 0$ 和 $\zeta(\sigma+i\tau) = 0$ 的复数 $\sigma+i\tau$ 进行。

►26. [M25] (H. C. Pocklington, 1914) 设 $N = fr + 1$, 其中 $0 < r \leq f + 1$ 。证明: 如果对于 f 的每个素因子 p 存在一个整数 x_p 使得 $x_p^{N-1} \bmod N = \gcd(x_p^{(N-1)/p} - 1, N) = 1$, 则 N 是素数。

►27. [M30] 证明有一个方法来检验形如 $N = 5 \cdot 2^n + 1$ 的数的素性, 该方法使用和 Lucas-Lehmer 在定理 L 中检验梅森素数近似地相同的取平方模 N 的个数。[提示: 见上题。]

28. [M27] 给定一个素数 p 和一个正整数 d , p 整除 $A^2 - dB^2$ (计算重数) 的平均次数 $f(p, d)$ 的值是多少? 其中 A 和 B 是随机整数, 它们彼此独立, 但必须满足 $A \perp B$ 的条件。

29. [M25] 证明小于等于 n , 且其素因子全都包含于一个给定的素数集合 $\{p_1, \dots, p_m\}$ 的正整数个数至少是 $m^r/r!$, 其中 $r = \lfloor \log n / \log p_m \rfloor$ 且 $p_1 < \dots < p_m$ 。

30. [HM35] (J. D. Dixon 和 Claus-Peter Schnorr) 设 $p_1 < \dots < p_m$ 是素数, 它们不整除奇数 N , 并设 $r \leq \log N / \log p_m$ 是一个偶整数。证明在 $0 \leq X < N$ 范围内使得 $X^2 \bmod N = p_1^{e_1} \dots p_m^{e_m}$ 的整数 X 的个数至少是 $m^r/r!$ 提示: 设 N 的因子分解为 $q_1^{f_1} \dots q_d^{f_d}$, 证明每当我们有 $e_1 + \dots + e_m \leq r$ 且 $p_1^{e_1} \dots p_m^{e_m}$ 对于 $1 \leq i \leq d$ 是一个二次余模 q_i 时, 指数 (e_1, \dots, e_m) 的序列导致 2^d 个解 X 。这样的指数序列可以作为有序偶 $(e'_1, \dots, e'_m; e''_1, \dots, e''_m)$ 而得到, 其中 $e'_1 + \dots + e'_m \leq \frac{1}{2}r$ 且 $e''_1 + \dots + e''_m \leq \frac{1}{2}r$ 且对于 $1 \leq i \leq d$

$$(p_1^{e'_1} \dots p_m^{e'_m})^{(q_i-1)/2} \equiv (p_1^{e''_1} \dots p_m^{e''_m})^{(q_i-1)/2} \pmod{q_i}$$

31. [M20] 使用习题 1.2.10-21 来估计 Dixon 的因子分解算法(在定理 D 前边已有介绍)获得少于 $2m$ 个输出的概率。

►32. [M32] 说明如何修改 RSA 编码方案, 使得对于小于 $\sqrt[3]{N}$ 的消息不存在任何问题, 即消息的长度不会大量地增加。

33. [M50] 证明或否定: 给定一个数 $N = pq$, 它的素因子满足 $p \equiv q \equiv 2 \pmod{3}$, 而且给定 $x^3 \bmod N$ 的值, 则有一个相当有效的算法, 它有能力以不可忽略的概率找出 N 的因子。[如果这可以证明, 它将不仅说明立方根的问题和因子分解一样困难, 而且它还将说明, RSA 方案和 SQRT 方案一样有致命的缺陷。]

34. [M30] (Peter Weinberger) 假设在 RSA 方案中 $N = pq$, 并假设你知道一个数 m , 对于至少 10^{-12} 的所有正整数 x 有 $x^m \bmod N = 1$ 。如果 m 不太大 (比如说 $m < N^{10}$), 说明你将不太困难地对 N 进行因子分解。

► 35. [M25] (H. C. Williams, 1979) 设 N 是两个素数 p 和 q 的乘积, 其中 $p \bmod 8 = 3$ 和 $q \bmod 8 = 7$, 证明雅可比符号满足 $\left(\frac{-x}{N}\right) = \left(\frac{x}{N}\right) - \left(\frac{2x}{N}\right)$, 并使用它来设计一个类似于 Rabin 的 SQRT 箱的一个无消息二义性的编码/译码方案。

36. [HM24] 除非 N 极大, (22) 后边的渐近分析对于给出有意义的值来说太粗糙了, 因为当 N 在一个实用的范围中时, $\ln \ln N$ 总是比较小的。试进行更精确的分析, 以便对合理的 N 值能进一步了解 (22) 的特性; 并说明如何选择 $\ln m$ 的一个值, 除了顶多 $\exp(O(\log \log N))$ 大小的因子外, 它极小化 (22)。

37. [M27] 证明每个正整数 D 的平方根有形如

$\sqrt{D} = R + // a_1, \dots, a_n, 2R, a_1, \dots, a_n, 2R, a_1, \dots, a_n, 2R, \dots //$ 的周期连分数, 除非 D 是一个完全的平方根, 其中 $R = \lfloor \sqrt{D} \rfloor$ 且 (a_1, \dots, a_n) 是一个回文 (即对于 $1 \leq i \leq n, a_i = a_{n+1-i}$)。

38. [25] (无用的素数) 对于 $0 \leq d \leq 9$, 试求 P_d , 即最大的 50 位十进数字的素数, 它有等于 d 的极大的十进数字的个数。(首先把 d 的数极大化, 然后求最大的这样的素数。)

39. [40] 许多素数 p 都有这样一个性质, 即 $2p+1$ 也是一个素数。例如, $5 \rightarrow 11 \rightarrow 23 \rightarrow 47$ 。更一般地说, 对于某个 $k \geq 0$, 如果 p 和 q 都是素数, 而 $q = 2^k p + 1$, 就说 q 是 p 的一个后继。例如 $2 \rightarrow 3 \rightarrow 7 \rightarrow 29 \rightarrow 59 \rightarrow 1889 \rightarrow 3779 \rightarrow 7559 \rightarrow 4058207223809 \rightarrow 32465657790473 \rightarrow 4462046030502692971872257 \rightarrow 95$ (30 位被省略的数字) $37 \rightarrow \dots; 95 \dots 37$ 的最小后继有 103 位十进数字。试求出你能求的最长的逐次的素数链。

► 40. [M36] (A. Shamir) 考虑一台抽象计算机, 它能在仅仅一个单位的时间内对任意长度的整数 x 和 y 实现 $x+y, x-y, x \cdot y$ 和 $\lfloor x/y \rfloor$ 的运算, 不管这些整数有多大。这个机器把整数存入一个随机存取的内存中, 而且给定 x 和 y , 它能取决于是否 $x=y$ 来选择不同的程序步骤。本题的目的是说明, 在这样一台计算机上存在令人惊异地快速的方法对数进行因子分解。(因此大概很难证明在实际的机器上进行因子分解是固有地复杂的, 尽管我们猜测它是复杂的。)

a) 给定一个整数值 $n \geq 2$, 试找出在这样一台计算机上以 $O(\log n)$ 的步骤计算 $n!$ 的方法。

[提示: 如果 A 是一个充分大的整数, 则由 $(A+1)^m$ 的值可以容易地计算二项式系数 $\binom{m}{k} = m! / (m-k)! k!$ 。]

b) 给定一个整数值 $n \geq 2$, 说明如何在这样一台计算机上以 $O(\log n)$ 的步骤计算有下列这样性质的数 $f(n)$: 如果 n 是素数, 则 $f(n) = n$; 否则 $f(n)$ 是 n 的一个真因子 (但不必是素数)。[提示: 如果 $n \neq 4$, 一个这样的函数是 $\gcd(m(n), n)$, 其中 $m(n) = \min \{m \mid m! \bmod n = 0\}$ 。]

[作为 b) 的一个推论, 我们通过对任意大的整数只做 $O(\log n)^2$ 个算术运算, 就可对一个给定的数 n 进行完全的因子分解。给定一个部分的因子分解 $n = n_1 \cdots n_r$, 每一个非素数 n_i 都可以在 $\sum O(\log n_i) = O(\log n)$ 的步骤内被 $f(n_i) \cdot (n_i / f(n_i))$ 所代替, 而这个求精运算可以重复进行直到所有 n_i 都是素数为止。]

► 41. [M28] (Lagarias, Miller 和 Odlyzko) 本题的目的是说明通过只观察小于 N^2 的素数就可计算小于 N^3 的素数个数, 因此可在 $O(N^{2+\epsilon})$ 步内计算 $\pi(N^3)$ 。

我们说一个“ m 幸存者”指的是其素因子全都超过 m 的一个正整数; 因此, 一个 m 幸存者是在小于等于 m 的素数的所有倍数都被筛出之后的 Eratosthenes 筛 (习题 8) 中剩下来的。令 $f(x,$

m) 是小于等于 x 的 m 幸存者的个数, 而令 $f_k(x, m)$ 是恰好有 k 个素因子 (计算重数) 的这样的幸存者的个数。

a) 证明 $\pi(N^3) = \pi(N) + f_2(N^3, N) - 1 - f_2(N^3, N)$ 。

b) 说明如何对于 $x \leq N^2$ 从 $\pi(x)$ 的值来计算 $f_2(N^3, N)$ 。用你的方法手算 $f_2(1000, 10)$ 。

c) 和问题 b) 相同, 但不是算 $f_2(N^3, N)$, 而是计算 $f(N^3, N)$ 。[提示: 使用恒等式 $f(x, p_j) = f(x, p_{j-1}) - f(x/p_j, p_{j-1})$, 其中 p_j 是第 j 个素数且 $p_0 = 1$ 。]

d) 试讨论用于有效计算 b) 和 c) 中的量的数据结构。

42. [M35] (H. W. Lenstra, Jr.) 给定 $0 < r < s < N$ 且 $r \perp s$ 和 $N \perp s$ 。试证通过对 $(\lg N)$ 个二进位的数实施 $O([N/s^3]^{1/2} \log s)$ 次明确选定的算术运算, 有可能找出同余于 r (modulo s) 的 N 的所有因子。[提示: 应用习题 4.5.3~49。]

► 43. [M43] 令 $m = pq$ 是定理 3.5P 中那样的 r 个二进位的 Blum 整数, 并设 $Q_m = \{y \mid y = x^2 \pmod m, \text{ 对于某个 } x\}$ 。于是 Q_m 有 $(p+1)(q+1)/4$ 个元素, 而且每一个元素 $y \in Q_m$ 有惟一的根 $x = \sqrt{y}$, 使得 $x \in Q_m$ 。假设 $G(y)$ 是当 y 是 Q_m 的一个随机元素时, 以大于等于 $\frac{1}{2} - \epsilon$ 的概率正确地猜测 $\sqrt{y} \pmod 2$ 的一个算法。本题的目的是证明由 G 解决的问题几乎和对 m 因子分解的问题一样难。

a) 试构造一个算法 $A(G, m, \epsilon, y, \delta)$, 它使用随机数和算法 G 来猜测一个给定整数 y 是否在 Q_m 中, 而无需计算 \sqrt{y} 。你的算法应该以大于等于 $1 - \delta$ 的概率正确地猜测, 而且若假定 $T(G) \geq r^2$, 则它的运行时间 $T(A)$ 至多应是 $O(\epsilon^{-2} (\log \delta^{-1}) T(G))$ 。(如果 $T(G) < r^2$, 则在这个公式中以 $(T(G) + r^2)$ 来代替 $T(G)$ 。)

b) 试构造一个算法 $F(G, m, \epsilon)$, 它以预期的运行时间 $T(F) = O(r^2 (\epsilon^{-6} + \epsilon^{-4} \cdot (\log \epsilon^{-1}) T(G)))$ 来求 m 的因子。

提示: 对于固定的 $y \in Q_m$, 以及对于 $0 \leq v < m$, 设 $tv = v\sqrt{y} \pmod m$ 和 $\lambda v = tv \pmod 2$ 。注意, $\lambda(-v) + \lambda v = 1$ 且 $\lambda(v_1 + \dots + v_n) = (\lambda v_1 + \dots + \lambda v_n + \lfloor (tv_1 + \dots + tv_n)/m \rfloor) \pmod 2$ 。其次我们有 $\tau(\frac{1}{2}v) = \frac{1}{2}(\tau v + m\lambda v)$; 这里 $\frac{1}{2}v$ 代表 $(\frac{m+1}{2}v) \pmod m$ 。如果 $\pm v \in Q_m$, 我们有 $\tau(\pm v) = \sqrt{v^2 y}$; 因此算法 G 给了我们以大约所有 v 的一半来猜测 λv 的一个方法。

44. [M35] (J. Håstad) 证明, 如果对于 $1 \leq i < j \leq 7$, $m_i \perp m_j$, 则当对于 $1 \leq i \leq 7$, $a_{i0} + a_{i1}x + a_{i2}x^2 + a_{i3}x^3 \equiv 0 \pmod{m_i}$, $0 < x < m_i$, $\gcd(a_{i0}, a_{i1}, a_{i2}, a_{i3}, m_i) = 1$ 且 $m_i > 10^{27}$ 时, 不难求 x 。(所有变量都是整数, 除 x 之外全都已知。) 提示: 当 L 是实数的任何非奇异矩阵时, Lenstra, Lenstra 和 Lovász 的算法 [Mathematische Annalen 261 (1982), 515~534] 有效地求一个非零整向量 $v = (v_1, \dots, v_n)$, 使得长度 $\|vL\| \leq \sqrt{n} 2^n |\det L|^{1/n}$ 。

► 45. [M41] (J. M. Pollard 和 Claus-Peter Schnorr) 证明: 给定整数 a, b 和 n 且 $ab \perp n$ 且 n 为奇数, 即使 n 的因子分解是不知道的, 但仍存在一个有效的方法来求解对于整数 x 和 y 的同余式

$$x^2 - ay^2 \equiv b \pmod n$$

[提示: 使用恒等式 $(x_1^2 - ay_1^2)(x_2^2 - ay_2^2) = x^2 - ay^2$, 其中 $x = x_1x_2 - ay_1y_2$ 且 $y = x_1y_2 + x_2y_1$ 。]

46. [HM30] (L. Adleman) 设 p 是一个相当大的素数且设 a 是一模 p 原根; 于是, 在范围 $1 \leq b < p$ 中的所有整数 b , 对于满足 $1 \leq n < p$ 的某个惟一的 n , 可写成 $b = a^n \pmod p$ 。

试设计一个算法, 它使用类似于 Dixon 的因子分解算法的思想, 对于所有的 $\epsilon > 0$, 对于给定的 b , 在 $O(p^\epsilon)$ 步骤内, 几乎总能找到 n 。[提示: 通过构造数 n_i 的一个所有组成开始, 使得

$a^n \bmod p$ 只有小的素因子。]

47. [M50] 以 ASCII 码表示的某个文字摘录 $x = x_1x_2$, 有一个加密了的值 $(x_1^3 \bmod N, x_2^3 \bmod N) =$

(14E97EF5C531D92591B89CDBAB48444A04612C01AA29C2A8FA10FA804EF7AC3CE03D7D3667C4D3E132A24A68
E6797FE28650DC3ADF327474B86B0CBD5387A49872CE012269A59B3E4B3ED83B74681A78AD7B6D1772A7451E,
15B025E2AEE095A9542590184CF62F72B2EE8DD794AEF8511F2591E6BC2C8B8A8E48AF1FE04FF2FD933E730
9205A3418DBB9BB8C6A7665DA309531735FE86C741D1261B34CB2668FA34D0C0C28575A2454E3DB00E438AC7)

这是在十六进制下, 其中 N 是

17B2353B9595ECA69FEF80940160C4084286D1255EFE49D114F2E633F82C8D5224FC4AA6F9104CED2BCA810
BEA76157FFDC78F9656A0ED9B3F6CCAB99001B8B2571F4EBD095925F07F9BEE5111E8375DEF71593628AD8D1

问 x 是什么?

*The problem of distinguishing prime numbers from composites,
and of resolving composite numbers into their prime factors,
is one of the most important and useful in all of arithmetic.
... The dignity of science seems to demand that every aid to the solution
of such an elegant and celebrated problem be zealously cultivated.*

区分素数同合数, 以及把合数分解成它们的素因子,
在所有算术运算中是最重要和最有用的问题之一。
……科学的尊严似乎要求, 对于解决这样一个精彩驰名的问题
的每一个帮助都应予以积极热情的扶植。

—— C. F. GAUSS, *Disquisitiones Arithmeticae*, Article 329 (1801)

4.6 多项式算术

我们已经研究的一些技术可以自然地应用于许多不同类型的数学量上, 而不只是数。这节我们将讨论多项式, 它是数的下一步。形式地说, S 上的一个多项式是形如

$$u(x) = u_n x^n + \cdots + u_1 x + u_0 \quad (1)$$

的表达式, 其中系数 u_n, \cdots, u_1, u_0 是某个代数系统 S 的元素, 变量 x 可以看做带有不确定意义的形式符号。我们将假定, 代数系统 S 是含 1 的可交换环; 这意味着 S 允许进行加法、减法和乘法运算, 并满足传统性质: 加法和乘法是定义在 S 上的二元运算; 它们是可结合和可交换的, 而且其中乘法对加法可分配。有一个加法的单位元素 0 和乘法的单位元素 1, 使得对于 S 中的所有的 a , $a + 0 = a$ 和 $a \cdot 1 = a$ 。减法是加法的逆。但关于除法作为乘法的逆的可能性, 我们不作任何假定。多项式 $0x^{n+m} + \cdots + 0x^{n+1} + u_n x^n + \cdots + u_1 x + u_0$ 被认为是和 (1) 相同的多项式, 尽管它的表达式在形式上是不同的。

如果 $u_n \neq 0$, 我们说 (1) 是一个 n 次多项式且前导系数为 u_n ; 在这一情况下, 我们写

$$\deg(u) = n, \ell(u) = u_n \quad (2)$$

按照约定,我们也置

$$\deg(0) = -\infty, \ell(0) = 0 \quad (3)$$

其中 0 表示其系数全部为 0 的 0 多项式。如果前导系数 $\ell(u) = 1$, 则我们说 $u(x)$ 是一首一多项式。

多项式算术主要由多项式加法、减法和乘法组成;在某些情况下,进一步的运算例如除法,指数运算,分解因子以及求最大公因子是重要的。加法、减法和乘法是以一种自然方式定义的,就如同变量 x 是 S 的一个元素那样:多项式的加法和减法通过把同类的 x 幂的系数相加或相减来完成。乘法则通过规则

$$(u_r x^r + \cdots + u_0)(v_s x^s + \cdots + v_0) = w_{r+s} x^{r+s} + \cdots + w_0$$

来完成,其中

$$w_k = u_0 v_k + u_1 v_{k-1} + \cdots + u_{k-1} v_1 + u_k v_0 \quad (4)$$

在后一公式中,如果 $i > r$ 或 $j > s$, 则 u_i 或 v_j 被当做 0。

代数系统 S 通常是整数的或有理数的集合;或者它本身可以是一个(不同于 x 的变量的)多项式集合;在后一情况下(1)是一个多变量多项式,即若干个变量的多项式。当代数系统 S 由整数 $0, 1, \dots, m-1$ 组成,而且加法、减法和乘法是按模 m 实现的(参考 4.3.2-(11))时,出现另一个重要的情况;这就是所谓模 m 的多项式算术。当每个系数是 0 或 1 时,即是模 2 的多项式算术,这个特殊情况是特别重要的。

读者应当注意多项式算术和多精度算术(4.3.1 小节)之间的类似性,在多精度算术中的进制 b 被换成为 x 。主要差别是,多项式算术中 x^k 的系数 u_k 同它的相邻系数 $u_{k \pm 1}$ 没有实质上的关系,所以不存在从一个位置到另一个位置“进位”的概念。事实上,除了不考虑进位外,模 b 的多项式算术实质上就等于进制为 b 的多精度算术。例如,比较在二进制数系下的 $(1101)_2$ 乘以 $(1011)_2$ 的乘法同模 2 的 $x^3 + x^2 + 1$ 和 $x^3 + x + 1$ 的类似乘法:

二进系统	模 2 多项式
$\begin{array}{r} 1101 \\ \times 1011 \\ \hline 1101 \\ 1101 \\ 1101 \\ \hline 10001111 \end{array}$	$\begin{array}{r} 1101 \\ \times 1011 \\ \hline 1101 \\ 1101 \\ 1101 \\ \hline 1111111 \end{array}$

这些多项式的模 2 乘积通过免去所有进位而得到,因此它是 $x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ 。如果我们把同样的多项式当做整系数多项式相乘,而不取模 2 的剩余,则结果将是 $x^6 + x^5 + x^4 + 3x^3 + x^2 + x + 1$;进位再次免去,但在这种情况下,系数可以变得任意大。

鉴于同多精度算术的这种强烈的类似性,在这节里已没有必要进一步讨论多项式的加法、减法和乘法。然而,应当指出通常多项式算术在实际应用中颇不同于多

精度算术的某些方面:由于多项式通常都有很多0系数的趋势,而且多项式次数通常很高,所以希望能有特殊形式的表示;见2.2.4小节。而且,多变量多项式算术的程序,用递归的结构最易于理解;第8章中要讨论这一情况。

尽管多项式加法、减法和乘法的技术是比较直截了当的,但是多项式算术有某些其它重要的方面,值得特殊考虑。因此,在以下的小节内将讨论多项式的除法及其相关联的技术,诸如求最大公因子和因子分解。我们还将讨论有效求值的问题,即当 x 是 S 的给定元素时,要用尽可能少的运算求出 $u(x)$ 的值。当 n 很大时,非常快地计算 x^n 这一特殊情况十分重要,所以要在4.6.3小节中详细讨论。

用来进行多项式算术的头一个重要的计算机子程序集是ALPAK系统[W. S. Brown, J. P. Hyde 和 B. A. Tague, *Bell System Tech. J.* **42** (1963), 2081~2119; **43** (1964), 785~804, 1547~1562]。在这方面的另一个里程碑是George Collins的PM系统[CACM **9** (1966), 578~589];也见C. L. Hamblin, *Comp. J.* **10** (1967), 168~171。

习 题

1. [10] 如果做模10多项式算术,则 $7x+2$ 减 x^2+5 等于多少? $6x^2+x+3$ 乘以 $5x^2+2$ 等于多少?

2. [17] 真还是假:(a)首一多项式的乘积是首一的。(b)次数分别为 m 和 n 的多项式的乘积有次数 $m+n$ 。(c)次数分别为 m 和 n 的多项式之和有次数 $\max(m, n)$ 。

3. [M20] 如果(4)中的每个系数 $u_r, \dots, u_0, v_s, \dots, v_0$ 是满足条件 $|u_i| \leq |m_1|, |v_j| \leq m_2$ 的整数,则乘积系数 w_k 的极大绝对值是多少?

►4. [21] 如果诸系数已被装入计算机的字内,通过使用在一台二进制计算机上的通常的算术运算,能否便利地进行模2的多项式乘法?

►5. [M24] 当 n 很大时,说明如何修改Karatsuba的方法(见4.3.3小节),以使用正比于 $O(n^{\lg 3})$ 的执行时间来进行模2的次数小于等于 n 的两个多项式的乘法。

4.6.1 多项式除法

当算术是对于一个域上的多项式进行时,以一个多项式除另一个多项式,有可能像以一个多精度的整数除另一个多精度的整数那样进行。一个域 S 是带有幺元的可交换环,其中加法、减法和乘法及精确的除法都是可能的;通常这意味着当 u 和 v 是 S 的元素且 $v \neq 0$ 时,有 S 中的一个元素 w ,使得 $u = vw$ 。在应用中出现的最重要的系数域是:

a) 有理数(表示成分数,见4.5.1小节);

b) 实数或复数(在一台计算机内借助于浮点近似值表示,见4.2节);

c) 模 p 的整数,其中 p 是素数(其中除法可如习题4.5.2-16中所建议的那样实现);

d) 一个域上的有理函数,即两个多项式的商的系数在该域中,分母是首一的。具有特殊重要性的是模2的整数域,其仅有的元素是0和1。在这个域上的多项式

(即模 2 的多项式)同二进记法下表达的整数有许多类似之点;而且这个域上的有理函数同其分子和分母以二进记法表示的有理数非常类似。

给定一个域上的两个多项式 $u(x)$ 和 $v(x)$, $v(x) \neq 0$, 我们可以用 $v(x)$ 除 $u(x)$ 得到一个商多项式 $q(x)$ 和一个余多项式 $r(x)$, 满足条件

$$u(x) = q(x) \cdot v(x) + r(x), \quad \deg(r) < \deg(v) \quad (1)$$

容易看出,至多有一对多项式 $(q(x), r(x))$ 满足这些关系;因为如果相对于相同的多项式 $u(x)$ 和 $v(x)$, $(q_1(x), r_1(x))$ 和 $(q_2(x), r_2(x))$ 都满足 (1), 则有 $q_1(x)v(x) + r_1(x) = q_2(x)v(x) + r_2(x)$, 所以 $(q_1(x) - q_2(x))v(x) = r_2(x) - r_1(x)$ 。现在如果 $q_1(x) - q_2(x)$ 非零, 则我们有 $\deg((q_1 - q_2) \cdot v) = \deg(q_1 - q_2) + \deg(v) \geq \deg(v) > \deg(r_2 - r_1)$, 矛盾;因此 $q_1(x) - q_2(x) = 0$ 且 $r_1(x) = r_2(x)$ 。

下列算法实际上同多精度除法算法 4.3.1D 相同,但不涉及任何进位。它可以用来确定 $q(x)$ 和 $r(x)$:

算法 D (一个域上的多项式除法) 给定一个域 S 上的多项式

$$u(x) = u_m x^m + \cdots + u_1 x + u_0, \quad v(x) = v_n x^n + \cdots + v_1 x + v_0$$

其中 $v_n \neq 0$, 且 $m \geq n \geq 0$, 这一算法求域 S 上的满足 (1) 的多项式

$$q(x) = q_{m-n} x^{m-n} + \cdots + q_0, \quad r(x) = r_{n-1} x^{n-1} + \cdots + r_0$$

D1. [对 k 进行迭代] 对 $k = m - n, m - n - 1, \cdots, 0$, 执行步骤 D2; 然后此算法以 $(r_{n-1}, \cdots, r_0) \leftarrow (u_{n-1}, \cdots, u_0)$ 终止。

D2. [除法循环] 对于 $j = n + k - 1, n + k - 2, \cdots, k$, 置 $q_k \leftarrow u_{n+k} / v_n$, 而后置

$$u_j \leftarrow u_j - q_k v_{j-k} \quad (\text{后一操作相当于用次数小于 } n + k \text{ 的一个多项式 } u(x) - q_k x^k v(x) \text{ 代替 } u(x)). \quad \blacksquare$$

算法 D 的一个例子见 (5) 的下边。算术运算的次数实际上同 $n(m - n + 1)$ 成比例。注意,系数的显式除法仅仅在步骤 D2 的开始处进行,而且除数总是 v_n ; 所以,如果 $v(x)$ 是首一多项式 ($v_n = 1$), 则全然没有真正的除法。如果乘法比除法易于进行,则宜于在本算法开始时,计算 $1/v_n$, 并在步骤 D2 中乘以这个量。

我们通常以 $u(x) \bmod v(x)$ 代替 (1) 中的余式 $r(x)$ 。

惟一因子分解整环 如果我们限于考虑一个域上的多项式,则将不能直接处理许多重要的情况,例如整系数的多项式,或若干变量的多项式。因此现在让我们考虑更一般的情况,即系数的代数系统 S 是惟一因子分解整环,这个整环不必是一个域。这意味着 S 是有么元的可交换环,而且

i) 每当 u 和 v 都是 S 的非零元素时, $uv \neq 0$;

ii) S 的每一非零元素 u 或者是一个单位,或者有作为素元 p_1, \cdots, p_t 的乘积的“惟一”表示

$$u = p_1 \cdots p_t, \quad t \geq 1 \quad (2)$$

这里,单位是有一个倒数的元素,即对于 S 中的某个 v 使得 $uv=1$ 的一个元素 u ; 而一个素元 p 则是一个非单位的元素,它使得仅当 q 或 r 之一是单位时, $p=qr$ 可为真。如果 $p_1 \cdots p_t = q_1 \cdots q_s$, 其中所有 p 和 q 都是素元,则 $s=t$, 而且有 $\{1, \cdots, t\}$ 的一个排列 $\pi_1 \cdots \pi_t$, 使得对某些单位 a_1, \cdots, a_t , $p_1 = a_1 q_{\pi_1}, \cdots, p_t = a_t q_{\pi_t}$ 。在这个意义下,表示(2)是惟一的。换言之,除了有一些单位和因子的次序不同外,分解成素元是惟一的。

任何域都是惟一因子分解整环,其中每个非零元素都是一个单位,而且没有素元。整数构成一个惟一因子分解整环,其中单位是 $+1$ 和 -1 , 素元是 $\pm 2, \pm 3, \pm 5, \pm 7, \cdots$ 。 S 是所有整数的集合这一情况具有决定性的重要意义,因为以整数来进行工作通常比用任何有理系数都更为可取。

关于多项式的关键事实之一(见习题 10)是:一个惟一因子分解整环上的多项式形成一个惟一的因子分解整环。在这个整环中是素元的一个多项式通常称做不可约多项式。通过反复使用惟一因子分解定理,我们可以证明,整数环上或者在任何域上的任意个变量的多项式,都可以惟一地分解成不可约多项式。例如,在整数环上的多变量多项式 $90x^3 - 120x^2y + 18x^2yz - 24xy^2z$ 是五个不可约多项式的乘积 $2 \cdot 3 \cdot x \cdot (3x - 4y) \cdot (5x + yz)$ 。同样的多项式,作为有理数域上的多项式,是三个不可约多项式的乘积 $(6x) \cdot (3x - 4y) \cdot (5x + yz)$; 这个因子分解也可以写成 $x \cdot (90x - 120y) \cdot \left(x + \frac{1}{5}yz\right)$, 而且还有无穷的其它形式,尽管这个因子分解实质上是惟一的。

和通常一样,如果 $u(x) = v(x)q(x)$, 其中 $q(x)$ 是某个多项式,则我们说, $u(x)$ 是 $v(x)$ 的一个倍式,同时 $v(x)$ 是 $u(x)$ 的一个除式。如果我们有一个算法, 对于一个惟一因子分解整环 S 的任意非零元素 u 和 v , 能告知 u 是否 v 的倍元素, 而且如果 $u = vw$, 则确定 w , 那么算法 D 就给了我们一个方法, 对于 S 上的任意多项式 $u(x)$ 和 $v(x)$, 它告知我们 $u(x)$ 是否为 $v(x)$ 的倍式。因为如果 $u(x)$ 是 $v(x)$ 的一个倍式, 容易看出, 每次我们达到步骤 D2 时, u_{n+k} 必定是 v_n 的一个倍式, 因此将找到商 $u(x)/v(x)$ 。递归地应用这一发现, 我们就得到一个算法, 它判定 S 上的一个任意多个变量的给定多项式, 是否是 S 上另一个给定的多项式的倍式, 而且当存在时就把商找出来。

在一个惟一因子分解整环中, 如果没有素元能同时整除其中的一组元素, 就称这组元素是互素的。如果一个惟一因子分解整环上的一个多项式的系数互素, 就称该多项式为本原的。(这个概念不应同 3.2.2 小节中讨论的十分不同的“模 p 本原多项式”的概念相混淆。) 下列事实具有基本的重要性, 它是由 C. F. Gauss 在他的名作 *Disquisitiones Arithmeticae* (Leipzig: 1801) 的论文 42 中为了研究整数的多项式而引进的:

引理 G(高斯引理) 一个惟一因子分解整环上的本原多项式的乘积是本原的。

证明 设 $u(x) = u_mx^m + \cdots + u_0$ 和 $v(x) = v_nx^n + \cdots + v_0$ 是本原多项式。如

果 p 是此整环的任意素元,我们必须证明 p 不能整除乘积 $u(x)v(x)$ 的所有系数。由假定,有一个下标 j 使得 u_j 不为 p 所整除,还有一个下标 k 使得 v_k 不为 p 所整除。取 j 和 k 尽可能小,则在 $u(x)v(x)$ 中 x^{j+k} 的系数是

$$u_j v_k + u_{j+1} v_{k-1} + \cdots + u_{j+k} v_0 + u_{j-1} v_{k+1} + \cdots + u_0 v_{k+j}$$

并且容易看出这不是 p 的倍数(因为它的首项不是,而所有其它项都是)。■

如果在惟一因子分解整环 S 上的一个非零多项式不是本原的,则我们可以写出 $u(x) = p_1 \cdot u_1(x)$, 其中 p_1 是整除 $u(x)$ 的所有系数的 S 的素元, $u_1(x)$ 是 S 上的另一个非零多项式。 $u_1(x)$ 的所有系数比 $u(x)$ 的相应系数少一个素因子。现在如果 $u_1(x)$ 不是本原的,则我们可以写 $u_1(x) = p_2 \cdot u_2(x)$, 等等,而且这一过程必定以一个表示 $u(x) = c \cdot u_k(x)$ 而最后终止,其中 c 是 S 的一个元素,而且 $u_k(x)$ 是本原的。事实上,我们有下列引理作为引理 G 的补充。

引理 H 一个惟一因子分解整环 S 上的任何非零多项式 $u(x)$ 均可分解成形式 $u(x) = c \cdot v(x)$, 其中 c 在 S 中而且 $v(x)$ 是本原的。而且,在下列意义下,这个表示是惟一的,即如果 $u = c_1 v_1(x) = c_2 v_2(x)$, 则 $c_1 = a c_2$, $v_2(x) = a v_1(x)$, 其中 a 是 S 的一个单位元素。

证明 我们已经证明了这样一个表示存在,因此这里只须证它的惟一性。假定 $c_1 \cdot v_1(x) = c_2 \cdot v_2(x)$, 其中 $v_1(x)$ 和 $v_2(x)$ 是本原的。设 p 是 S 的任何素元,如果 p^k 整除 c_1 , 则 p^k 也整除 c_2 ; 否则 p^k 将整除 $c_2 \cdot v_2(x)$ 的所有系数,所以 p 将整除 $v_2(x)$ 的所有系数,矛盾。类似地, p^k 整除 c_2 仅当 p^k 整除 c_1 。因此,由惟一因子分解, $c_1 = a c_2$, 其中 a 是一个单位元; 而且 $0 = a c_2 v_1(x) - c_2 v_2(x) = c_2 (a v_1(x) - v_2(x))$, 所以 $a v_1(x) - v_2(x) = 0$ 。■

因此我们可以把任何非零多项式 $u(x)$ 写成

$$u(x) = \text{cont}(u) \cdot \text{pp}(u(x)) \quad (3)$$

其中 $\text{cont}(u)$, 即 u 的容度, 是 S 的一个元素, 而 $\text{pp}(u(x))$, 即 $u(x)$ 的本原部分, 是 S 上的一个本原多项式。当 $u(x) = 0$ 时, 定义 $\text{cont}(u) = \text{pp}(u(x)) = 0$ 是很方便的。把引理 G 和 H 结合在一起, 就给出了关系式

$$\begin{aligned} \text{cont}(u \cdot v) &= a \text{cont}(u) \text{cont}(v) \\ \text{pp}(u(x) \cdot v(x)) &= b \text{pp}(u(x)) \text{pp}(v(x)) \end{aligned} \quad (4)$$

其中 a 和 b 是单位, 它们取决于容度被计算的方法而且 $ab = 1$ 。当我们对整数上的多项式进行处理时, 仅有的单位元素是 $+1$ 和 -1 , 因此定义 $\text{pp}(u(x))$ 使得它的前导系数为正是方便的。于是 (4) 对于 $a = b = 1$ 为真。当处理一个域上的多项式时, 我们可以取 $\text{cont}(u) = \ell(u)$, 使得 $\text{pp}(u(x))$ 是首一的; 在这种情况下, 对于所有的 $u(x)$ 和 $v(x)$, (4) 再次对于 $a = b = 1$ 成立。

例如, 如果我们处理整数上的多项式, 设 $u(x) = 26x^2 + 39$, $v(x) = 21x + 14$, 则

最大公因子 当有惟一的因子分解时,就可以来谈论两个元素的最大公因子。为尽可能多的素元整除的公因子(参考等式 4.5.2-(6))。然而,由于一个因子分解整环可以有許多单位元素,故在这个最大公因子的定义中有二义性;是 u 和 v 的最大公因子,则当 a 是单位元素时 $a \cdot w$ 也是。反之,惟一因子分解意味着,如果 w_1 和 w_2 两者都是 u 和 v 的最大公因子,则对于某一单位元有 $w_1 = aw_2$ 。换句话说,一般说来,谈论 u 和 v 的“某个”最大公因子是没有意义的;有一个最大公因子的集合,每一个是其它的单位倍元。

如果 $v(x) = 0$, 则 $\gcd(u(x), v(x)) = u(x)$
 否则 $\gcd(u(x), v(x)) = \gcd(v(x), r(x))$

例如,用关于整数模 13 上的多项式的欧几里得算法,我们来确定 $x^8 + x^6 + 10x^4 + 10x^3 + 8x^2 + 2x + 8$ 和 $3x^6 + 5x^4 + 9x^2 + 4x + 8$ 模 13 的最大公因子。首先,仅写出系数说明算法 D 的步骤,我们有

因此

类似地

(这里的等号指的是模 13 同余, 因为对于系数的所有算术都是按模 13 处理的。) 这

个计算表明, 12 是两个最初的多项式的最大公因子。现在一个域中的任何非零元素都是这个域上的多项式整环中的一个单位, 因此, 在域的情况下, 习惯上是以前导系数来除这个算法的结果而产生一个首一多项式, 将这个多项式称为两个给定多项式的最大公因子。因此(6)中计算的最大公因子相应地取为 1, 而不是 12。(6)中最后的步骤已可省略, 因为如果 $\deg(v) = 0$, 则 $\gcd(u(x), v(x)) = 1$, 而不管你对 $u(x)$ 选择的是什么多项式。习题 4 确定对于模 p 随机多项式的欧几里得算法的平均运行时间。

现在我们转到更为一般的情况, 其中的多项式在不是一个域的惟一因子分解整环上给出。从等式(4), 我们可以推导出重要的关系

$$\begin{aligned}\text{cont}(\gcd(u, v)) &= a \cdot \gcd(\text{cont}(u), \text{cont}(v)) \\ \text{pp}(\gcd(u(x), v(x))) &= b \cdot \gcd(\text{pp}(u(x)), \text{pp}(v(x)))\end{aligned}\quad (7)$$

其中 a 和 b 是单位元素。这里 $\gcd(u(x), v(x))$ 是 x 的任何特定的多项式, 是 $u(x)$ 和 $v(x)$ 的最大公因子。等式(7)把求任意多项式的最大公因子的问题归结为求本原多项式的最大公因子的问题。

用于一个域上的多项式除法的算法 D 可以推广成为任何代数系统上多项式的伪除法, 这个代数系统是有么元的可变换环。我们注意到算法 D 要求的只是显式地除以 $\ell(v)$, 即 $v(x)$ 的前导系数, 而步骤 D2 要进行恰好 $m - n + 1$ 次; 于是如果 $u(x)$ 和 $v(x)$ 以整系数开始, 而且如果我们在有理数上进行工作, 则仅仅是出现在 $q(x)$ 和 $r(x)$ 中的系数的分母是 $\ell(v)^{m-n+1}$ 的因子, 这提示我们, 总是可以求得多项式 $q(x)$ 和 $r(x)$, 使得对于任何多项式 $u(x)$ 和 $v(x) \neq 0$, 假定 $m \geq n$,

$$\ell(v)^{m-n+1}u(x) = q(x)v(x) + r(x), \quad \deg(r) < n \quad (8)$$

其中 $m = \deg(u)$ 和 $n = \deg(v)$ 。

算法 R(多项式的伪除法) 给定多项式

$$u(x) = u_mx^m + \cdots + u_1x + u_0, \quad v(x) = v_nx^n + \cdots + v_1x + v_0$$

其中 $v_n \neq 0, m \geq n \geq 0$, 这个算法求满足(8)的多项式 $q(x) = q_{m-n}x^{m-n} + \cdots + q_0$ 和 $r(x) = r_{n-1}x^{n-1} + \cdots + r_0$ 。

R1. [对 k 进行迭代] 对 $k = m - n, m - n - 1, \cdots, 0$, 执行步骤 R2, 然后算法以 $(r_{n-1}, \cdots, r_0) = (u_{n-1}, \cdots, u_0)$ 终止。

R2. [乘法循环] 置 $q_k \leftarrow u_{n+k}v_n^k$, 并且对于 $j = n + k - 1, n + k - 2, \cdots, 0$, 置 $u_j \leftarrow v_n u_j - u_{n-k}v_{j-k}$ 。(当 $j < k$ 时, 这意味着 $u_j \leftarrow v_n u_j$, 因为我们把 v_{-1}, v_{-2}, \cdots 当做零。如果对于 $0 \leq t < m - n$, 以 $v_n^{m-n-t}u_t$ 代替 u_t 开始这个算法, 则这些乘法可以避免。) **■**

一个计算例子见于下面的(10)。通过对 $m - n$ 用归纳法, 容易证明算法 R 的

正确性。因为每次执行步骤 R2 实质上是以 $\ell(v)u(x) - \ell(u)x^k v(x)$ 代替 $u(x)$, 其中 $k = \deg(u) - \deg(v)$ 。注意, 在这个算法中根本不用任何除法; $q(x)$ 和 $r(x)$ 的系数本身又都是 $u(x)$ 和 $v(x)$ 的系数的某个多项式函数。如果 $v_n = 1$, 则这个算法等同于算法 D。如果 $u(x)$ 和 $v(x)$ 都是一个惟一因子分解整环上的多项式, 和以前一样我们可以证明多项式 $q(x)$ 和 $r(x)$ 是惟一的; 因此在惟一因子分解整环上进行伪除法的另一个方法是以 v_n^{m-n+1} 来乘 $u(x)$ 并应用算法 D, 且知道在步骤 D2 中的所有商都将存在。

用以下方法可以把算法 R 推广成为在惟一因子分解整环上本原多项式的一个“广义欧几里得算法”: 设 $u(x)$ 和 $v(x)$ 是本原多项式且 $\deg(u) \geq \deg(v)$, 并借助算法 R 确定满足 (8) 的多项式 $r(x)$ 。现在我们可以证明 $\gcd(u(x), v(x)) = \gcd(v(x), r(x))$; $u(x)$ 和 $v(x)$ 的任何公因子都整除 $u(x)$ 和 $v(x)$; 反之, $v(x)$ 和 $r(x)$ 的任何公因子都整除 $\ell(v)^{m-n+1}u(x)$, 而且它必定是本原的 (因为 $v(x)$ 是本原的), 所以它整除 $u(x)$ 。如果 $r(x) = 0$, 我们因此有 $\gcd(u(x), v(x)) = v(x)$; 另一方面, 如果 $r(x) \neq 0$, 我们有 $\gcd(v(x), r(x)) = \gcd(v(x), \text{pp}(r(x)))$, 因为 $v(x)$ 是本原的, 所以这个过程可以被迭代。

算法 E (广义欧几里得算法) 给定在一个惟一因子分解整环 S 上的非零多项式 $u(x)$ 和 $v(x)$, 本算法计算 $u(x)$ 和 $v(x)$ 的最大公因子。我们假定存在一些辅助算法, 以计算 S 的元素的非零最大公因子, 而且当 $b \neq 0$ 及 a 是 b 的一个倍数时, 在 S 中以 b 除 a 。

- E1. [归结为本原的]** 置 $d \leftarrow \gcd(\text{cont}(u), \text{cont}(v))$, 使用假定的计算 S 中的最大公因子的算法 (由定义 $\text{cont}(u)$ 是 $u(x)$ 的系数的最大公因子)。以多项式 $u(x)/\text{cont}(u) = \text{pp}(u(x))$ 代替 $u(x)$; 类似地, 以 $\text{pp}(v(x))$ 来代替 $v(x)$ 。
- E2. [伪除法]** 利用算法 R 计算 $r(x)$ (没有必要计算商多项式 $q(x)$)。如果 $r(x) = 0$, 则转到 E4。如果 $\deg(r) = 0$, 则以常数多项式“1”代替 $v(x)$ 并转到 E4。
- E3. [使得余式成为本原的]** 以 $v(x)$ 代替 $u(x)$, 并以 $\text{pp}(r(x))$ 代替 $v(x)$ 。转回步骤 E2。 (这是“欧几里得步骤”, 类似于我们已经看到的其它情况下的欧几里得算法。)
- E4. [配上容度]** 此算法终止, 且以 $d \cdot v(x)$ 作为所求答案。 ▮

作为算法 E 的一个例子, 我们来计算整系数多项式

$$\begin{aligned} u(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5 \\ v(x) &= 3x^6 + 5x^4 - 4x^2 - 9x + 21 \end{aligned} \quad (9)$$

的最大公因子。这些多项式是本原的, 所以步骤 E1 置 $d \leftarrow 1$ 。在步骤 E2 中, 我们有伪除法

$$\begin{array}{r}
 3 \ 0 \ 5 \ 0 \ -4 \ -9 \ 21 \) \ 1 \ 0 \ 1 \ 0 \ -3 \ -3 \ 8 \ 2 \ -5 \\
 \underline{3 \ 0 \ 3 \ 0 \ -9 \ -9 \ 24 \ 6 \ -15} \\
 3 \ 0 \ 5 \ 0 \ 4 \ -9 \ 21 \\
 \underline{0 \ -2 \ 0 \ -5 \ 0 \ 3 \ 6 \ -15} \\
 0 \ -6 \ 0 \ -15 \ 0 \ 9 \ 18 \ -45 \\
 \underline{0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0} \\
 -6 \ 0 \ -15 \ 0 \ 9 \ 18 \ -45 \\
 \underline{-18 \ 0 \ -45 \ 0 \ 27 \ 54 \ -135} \\
 -18 \ 0 \ -30 \ 0 \ 24 \ 54 \ -126 \\
 \underline{-15 \ 0 \ 3 \ 0 \ -9}
 \end{array} \quad (10)$$

这里商 $q(x)$ 是 $1 \cdot 3^2 x^2 + 0 \cdot 3^1 x + (-6) \cdot 3^0$; 我们有

$$27u(x) = v(x)(9x^2 - 6) + (-15x^4 + 3x^2 - 9) \quad (11)$$

现在步骤 E3 以 $v(x)$ 代替 $u(x)$ 和以 $\text{pp}(r(x)) = 5x^4 - x^2 + 3$ 代替 $v(x)$ 。下表概述了后续的计算, 其中仅示出系数

$$\begin{array}{rrr}
 u(x) & v(x) & r(x) \\
 1, 0, 1, 0, -3, -3, 8, 2, -5 & 3, 0, 5, 0, -4, -9, 21 & -15, 0, 3, 0, -9 \\
 3, 0, 5, 0, -4, -9, 21 & 5, 0, -1, 0, 3 & -585, -1125, 2205 \\
 5, 0, -1, 0, 3 & 13, 25, -49 & -233150, 307500 \\
 13, 25, -49 & 4663, -6150 & 143193869
 \end{array} \quad (12)$$

通过使用本小节早先所述的域上多项式的欧几里得算法, 把这个计算同有理数上而不是整数上的同一个最大公因子的计算加以比较是有教益的。出现了下边的惊人复杂的序列:

$$\begin{array}{rr}
 u(x) & v(x) \\
 1, 0, 1, 0, -3, -3, 8, 2, -5 & 3, 0, 5, 0, -4, -9, 21 \\
 3, 0, 5, 0, -4, -9, 21 & -\frac{5}{9}, 0, \frac{1}{9}, 0, -\frac{1}{3} \\
 -\frac{5}{9}, 0, \frac{1}{9}, 0, -\frac{1}{3} & -\frac{117}{25}, -9, \frac{441}{25} \\
 -\frac{117}{25}, -9, \frac{441}{25} & \frac{233150}{19773}, -\frac{102500}{6591} \\
 \frac{233150}{19773}, -\frac{102500}{6591} & -\frac{1288744821}{543589225}
 \end{array} \quad (13)$$

为了改进该算法,我们可以在每步把 $u(x)$ 和 $v(x)$ 约化成首一多项式,因为这就消去了使得系数不必要地复杂化的“单位”因子;这实际上就是有理数上的算法 E:

$$\begin{array}{ll}
 u(x) & v(x) \\
 1, 0, 1, 0, -3, -3, 8, 2, -5 & 1, 0, \frac{5}{3}, 0, -\frac{4}{3}, -3, 7 \\
 1, 0, \frac{5}{3}, 0, -\frac{4}{3}, -3, 7 & 1, 0, -\frac{1}{5}, 0, \frac{3}{5} \\
 1, 0, -\frac{1}{5}, 0, \frac{3}{5} & 1, \frac{25}{13}, -\frac{49}{13} \quad (14) \\
 1, \frac{25}{13}, -\frac{49}{13} & 1, -\frac{6150}{4663} \\
 1, -\frac{6150}{4663} & 1
 \end{array}$$

在(13)和(14)两者当中,多项式序列实际上和(12)一样,它是通过整数上的算法 E 得到的;惟一的差别是这些多项式已经乘以某些有理数。不论我们有 $5x^4 - x^2 + 3$, 还是 $-\frac{5}{9}x^4 + \frac{1}{9}x^2 - \frac{1}{3}$, 还是 $x^4 - \frac{1}{5}x^2 + \frac{3}{5}$, 实质上计算都是相同的。但使用有理数算术的任一个算法,倾向于比全整数的算法 E 要慢一些,因为有理算术在多项式有很大的次数时,要求在每一步内做更多的整数 gcd 计算。

把(12), (13)和(14)同上边的(6)加以比较也是有教益的,在那里我们以相当少的工作量就确定了同样的模 13 多项式 $u(x)$ 和 $v(x)$ 的 gcd。由于 $\ell(u)$ 和 $\ell(v)$ 都不是 13 的倍数,故 $\gcd(u(x), v(x)) = 1$ (模 13) 这一事实已足以证明 $u(x)$ 和 $v(x)$ 在整数上互素(因而在有理数上也如此);在 4.6.2 小节的结尾,我们将回过头来考虑节省时间的问题。

子结式算法 George E. Collins [JACM 14 (1967), 128~142] 发现了一般地说比算法 E 更为优越,并提供了关于算法 E 的特性的更进一步信息的一个巧妙算法。随后又由 W. S. Brown 和 J. F. Traub [JACM 18 (1971), 505~514; 也见 W. S. Brown, ACM Trans. Math. Software 4 (1978), 237~249] 对它做了改进。这一算法避免了步骤 E3 中本原部分的计算,代之以用 S 的一个元素来除,而这个元素已知为 $r(x)$ 的一个因子。

算法 C (一个惟一因子分解整环上的最大公因子) 这个算法有和算法 E 一样的输入和输出假定,而且有一个优点,就是所需要的关于系数最大公因子的计算较少。

C1. [归结为本原的] 如同算法 E 的步骤 E1 中那样,置 $d \leftarrow \gcd(\text{cont}(u), \text{cont}(v))$, 并以 $(\text{pp}(u(x)), \text{pp}(v(x)))$ 代替 $(u(x), v(x))$ 。置 $g \leftarrow h \leftarrow 1$ 。

C2.[伪除法] 置 $\delta \leftarrow \deg(u) - \deg(v)$ 。用算法 R 计算 $r(x)$ 。如果 $r(x) = 0$, 则转到 C4。如果 $\deg(r) = 0$, 则以常数“1”代替 $v(x)$, 并转到 C4。

C3.[调整余式] 以 $v(x)$ 代替多项式 $u(x)$, 以 $r(x)/gh^\delta$ 代替 $v(x)$ 。(这时 $r(x)$ 的所有系数都是 gh^δ 的倍数。)然后置 $g \leftarrow \ell(u)$, $h \leftarrow h^{1-\delta}g^\delta$ 并返回 C2。(即使 $\delta > 1$, h 的新值仍将在整环 S 内。)

C4.[配上容度] 返回 $d \cdot \text{pp}(v(x))$ 作为答案。 ▮

如果我们把这一算法应用于早先考虑的多项式(9), 则在步骤 C2 的开始处得到下列结果序列:

$u(x)$	$v(x)$	g	h	
1, 0, 1, 0, -3, 3, 8, 2, -5	3, 0, 5, 0, -4, -9, 21	1	1	
3, 0, 5, 0, -4, -9, 21	-15, 0, 3, 0, -9	3	9	(15)
-15, 0, 3, 0, -9	65, 125, -245	-15	25	
65, 125, -245	-9326, 12300	65	169	

在此算法结束后, $r(x)/gh^\delta = 260708$ 。

这个多项式序列是由算法 E 产生的序列中多项式的整数倍组成的。尽管多项式未约化成本原形式, 但这些系数总保持适当的大小, 这是因为步骤 C3 中有一个约化因子所致。

为了分析算法 C, 并证明它是正确的, 让我们把它产生的多项式序列叫做 $u_1(x), u_2(x), u_3(x), \dots$, 其中 $u_1(x) = u(x)$ 和 $u_2(x) = v(x)$ 。设对于 $j \geq 1$, $\delta_j = n_j - n_{j+1}$, 其中 $n_j = \deg(u_j)$; 并设对于 $j \geq 2$, $g_1 = h_1 = 1$, $g_j = \ell(u_j)$, $h_j = h_{j-1}^{1-\delta_{j-1}} \cdot g_j^{\delta_{j-1}}$, 于是我们有

$$\begin{aligned}
 g_2^{\delta_1+1} u_1(x) &= u_2(x) q_1(x) + g_1 h_1^{\delta_1} u_3(x), & n_3 < n_2 \\
 g_3^{\delta_2+1} u_2(x) &= u_3(x) q_2(x) + g_2 h_2^{\delta_2} u_4(x), & n_4 < n_3 \\
 g_4^{\delta_3+1} u_3(x) &= u_4(x) q_3(x) + g_3 h_3^{\delta_3} u_5(x), & n_5 < n_4
 \end{aligned} \tag{16}$$

等等。当 $n_{k+1} = \deg(u_{k+1}) \leq 0$ 时此过程终止。我们必须证明 $u_3(x), u_4(x), \dots$ 的系数都在 S 中, 即诸因子 $g_j h_j^{\delta_j}$ 完全整除诸余式的系数, 而且我们还必须证明诸 h_j 的值全都属于 S 。证明相当棘手, 但通过一个例子很容易理解。

如同在(15)中那样, 假设 $n_1 = 8, n_2 = 6, n_3 = 4, n_4 = 2, n_5 = 1, n_6 = 0$, 使得 $\delta_1 = \delta_2 = \delta_3 = 2, \delta_4 = \delta_5 = 1$ 。让我们写 $u_1(x) = a_8 x^8 + a_7 x^7 + \dots + a_0, u_2(x) = b_6 x^6 + b_5 x^5 + \dots + b_0, \dots, u_5(x) = e_1 x + e_0, u_6(x) = f_0$, 使得 $h_1 = 1, h_2 = b_6^2, h_3 = c_4^2/b_6^2$,

$h_4 = d_2^2 b_6^2 / c_4^2$ 。在这些项之下,考虑表1中所示数组是有帮助的。为具体起见,我们假设多项式的系数都是整数。我们有 $b_6^3 u_1(x) = u_2(x) q_1(x) + u_3(x)$; 所以如果我们以 b_6^3 乘行 A_5 , 并且减去行 B_7, B_6 和 B_5 的适当倍(对应于 $q_1(x)$ 的系数), 则我们将得到行 C_5 。如果我们以 b_6^3 乘行 A_4 , 并减去行 B_6, B_5 和 B_4 的倍数, 就得到行 C_4 。类似地, 我们有 $C_4^3 u_2(x) = u_3(x) q_2(x) + b_6^5 u_4(x)$; 所以我们可以 c_4^3 乘行 B_3 , 减去行 C_5, C_4 和 C_3 的整数倍再除以 b_6^5 而得到行 D_3 。

表1 算法C中的系数

行名	行														乘以	代之以行
A_5	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	0	0	0	0	0	b_6^3	C_5
A_4	0	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	0	0	0	0	b_6^3	C_4
A_3	0	0	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	0	0	0	b_6^3	C_3
A_2	0	0	0	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	0	0	b_6^3	C_2
A_1	0	0	0	0	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	0	b_6^3	C_1
A_0	0	0	0	0	0	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	b_6^3	C_0
B_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	0	0	0	0	0	0	0		
B_6	0	b_6	b_5	b_4	b_3	b_2	b_1	b_0	0	0	0	0	0	0		
B_5	0	0	b_6	b_5	b_4	b_3	b_2	b_1	b_0	0	0	0	0	0		
B_4	0	0	0	b_6	b_5	b_4	b_3	b_2	b_1	b_0	0	0	0	0		
B_3	0	0	0	0	b_6	b_5	b_4	b_3	b_2	b_1	b_0	0	0	0	c_4^3/b_6^5	D_3
B_2	0	0	0	0	0	b_6	b_5	b_4	b_3	b_2	b_1	b_0	0	0	c_4^3/b_6^5	D_2
B_1	0	0	0	0	0	0	b_6	b_5	b_4	b_3	b_2	b_1	b_0	0	c_4^3/b_6^5	D_1
B_0	0	0	0	0	0	0	0	b_6	b_5	b_4	b_3	b_2	b_1	b_0	c_4^3/b_6^5	D_0
C_5	0	0	0	0	c_4	c_3	c_2	c_1	c_0	0	0	0	0	0		
C_4	0	0	0	0	0	c_4	c_3	c_2	c_1	c_0	0	0	0	0		
C_3	0	0	0	0	0	0	c_4	c_3	c_2	c_1	c_0	0	0	0		
C_2	0	0	0	0	0	0	0	c_4	c_3	c_2	c_1	c_0	0	0		
C_1	0	0	0	0	0	0	0	0	c_4	c_3	c_2	c_1	c_0	0	$d_2^2 b_6^4 / c_4^5$	E_1
C_0	0	0	0	0	0	0	0	0	0	c_4	c_3	c_2	c_1	c_0	$d_2^2 b_6^4 / c_4^5$	E_0
D_3	0	0	0	0	0	0	0	0	d_2	d_1	d_0	0	0	0		
D_2	0	0	0	0	0	0	0	0	0	d_2	d_1	d_0	0	0		
D_1	0	0	0	0	0	0	0	0	0	0	d_2	d_1	d_0	0		
D_0	0	0	0	0	0	0	0	0	0	0	0	d_2	d_1	d_0	$e_2^2 c_4^2 / d_2^3 b_6^2$	F_0
E_1	0	0	0	0	0	0	0	0	0	0	0	0	e_1	e_0	0	
E_0	0	0	0	0	0	0	0	0	0	0	0	0	0	e_1	e_0	
F_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f_0	

为了证明 $u_4(x)$ 有整系数, 让我们考虑矩阵

$$\begin{matrix} A_2 \\ A_1 \\ A_0 \\ B_4 \\ B_3 \\ B_2 \\ B_1 \\ B_0 \end{matrix} \begin{pmatrix} a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & 0 & 0 \\ 0 & a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & 0 \\ 0 & 0 & a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 & 0 & 0 & 0 \\ 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 & 0 & 0 \\ 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 & 0 \\ 0 & 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 \\ 0 & 0 & 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \end{pmatrix} = M \quad (17)$$

所指出的行操作和各行的一个重新排列, 将把 M 变换成

$$\begin{matrix} B_4 \\ B_3 \\ B_2 \\ B_1 \\ C_2 \\ C_1 \\ C_0 \\ D_0 \end{matrix} \begin{pmatrix} b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 & 0 & 0 & 0 \\ 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 & 0 & 0 \\ 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 & 0 \\ 0 & 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 \\ 0 & 0 & 0 & 0 & c_4 & c_3 & c_2 & c_1 & c_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_4 & c_3 & c_2 & c_1 & c_0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_4 & c_3 & c_2 & c_1 & c_0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_2 & d_1 & d_0 \end{pmatrix} = M' \quad (18)$$

由于从 M 导出 M' 的方式, 如果 M_0 和 M'_0 表示通过选择 M 和 M' 中的 8 个相应的列而得到的任何正方矩阵, 我们必有

$$b_5^3 \cdot b_6^3 \cdot b_6^3 \cdot (c_4^3/b_6^5) \cdot \det M_0 = \pm \det M'_0$$

例如, 我们选择头 7 列以及包含 d_1 的一列, 则

$$b_6^3 \cdot b_6^3 \cdot b_6^3 \cdot (c_4^3/b_6^5) \cdot \det \begin{pmatrix} a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & 0 \\ 0 & a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_0 \\ 0 & 0 & a_8 & a_7 & a_6 & a_5 & a_4 & a_1 \\ b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 \\ 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & 0 \\ 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_2 & 0 \\ 0 & 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_0 \\ 0 & 0 & 0 & 0 & b_6 & b_5 & b_4 & b_1 \end{pmatrix} = \pm b_6^4 \cdot c_4^3 \cdot d_1$$

由于 $b_6 c_4 \neq 0$, 这证明了 d_1 是整数; 类似地, d_2 和 d_0 是整数。

一般地说, 我们可以类似方式证明 $u_{j+1}(x)$ 有整系数。如果从由行 $A_{n_2-n_j}$ 到 A_0 和由 $B_{n_1-n_j}$ 到 B_0 所组成的矩阵 M 开始, 而且实施表 1 中指出的行操作, 则我们将得到一个由行 $B_{n_1-n_j}$ 到 $B_{n_3-n_j+1}$, $C_{n_2-n_j}$ 到 $C_{n_4-n_j+1}$, \dots , $P_{n_j-2-n_j}$ 到 P_1 , $Q_{n_j-1-n_j}$ 到 Q_0 , 最后是 R_0 (包含 $u_{j+1}(x)$ 的系数的一行) 以某种顺序组成的矩阵 M' 。抽取适当

的列示出

$$\begin{aligned} & (g_2^{\delta_2+1}/g_1 h_1^{\delta_1})^{n_2-n_j+1} (g_3^{\delta_3+1}/g_2 h_2^{\delta_2})^{n_3-n_j+1} \cdots (g_j^{\delta_j+1}/g_{j-1} h_{j-1}^{\delta_{j-1}})^{n_j-n_j+1} \det M_0 = \\ & \pm g_2^{n_1-n_3} g_3^{n_2-n_4} \cdots g_{j-1}^{n_{j-2}-n_j} g_j^{n_{j-1}-n_j+1} r_i \end{aligned} \quad (19)$$

其中 r_i 是 $u_{j+1}(x)$ 的一个给定系数, 而 M_0 是 M 的一个子矩阵。诸 h 已经非常巧妙地选定使得这个等式简化为

$$\det M_0 = \pm r_i \quad (20)$$

(见习题 24)。因此 $u_{j+1}(r)$ 的每一系数都可表达为其元素为 $u(x)$ 和 $v(x)$ 的系数的一个 $(n_1+n_2-2n_j+2) \times (n_1+n_2-2n_j+2)$ 矩阵的行列式。

剩下要证明的是巧妙地选择的诸 h 也是整数。可应用类似的技术: 例如, 观察矩阵

$$\begin{matrix} A_1 \\ A_0 \\ B_3 \\ B_2 \\ B_1 \\ B_0 \end{matrix} \begin{pmatrix} a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & 0 \\ 0 & a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 & 0 & 0 \\ 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 & 0 \\ 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 \\ 0 & 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \end{pmatrix} = M \quad (21)$$

如表 1 中所确定的那样的行操作以及行的重新排列导致

$$\begin{matrix} B_3 \\ B_2 \\ B_1 \\ B_0 \\ C_1 \\ C_0 \end{matrix} \begin{pmatrix} b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 & 0 & 0 \\ 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 & 0 \\ 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 \\ 0 & 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \\ 0 & 0 & 0 & 0 & c_4 & c_3 & c_2 & c_1 & c_0 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_4 & c_3 & c_2 & c_1 & c_0 \end{pmatrix} = M' \quad (22)$$

因此, 如果考虑通过选择 M 和 M' 的 6 个对应列所得到的任何子矩阵 M_0 和 M'_0 , 则我们有 $b_6^3 \cdot b_6^3 \cdot \det M_0 = \pm \det M'_0$ 。当把 M_0 选择成为 M 的头 6 个列时, 我们发现 $\det M_0 = \pm c_4^2/b_6^2 = \pm h_3$, 所以 h_3 是一个整数。

一般地说, 为证明对于 $j \geq 3$, h_j 是一个整数, 我们从由行 $A_{n_2-n_j-1}$ 到 A_0 , 及 $B_{n_1-n_j-1}$ 到 B_0 所组成的矩阵 M 开始; 然后实施适当的行操作直到得到一个由行 $B_{n_1-n_j-1}$ 到 $B_{n_3-n_j}$, $C_{n_2+n_j-1}$ 到 $C_{n_4-n_j}$, \cdots , $P_{n_1-2-n_j-1}$ 到 P_0 , $Q_{n_1-1-n_j-1}$ 到 Q_0 的矩阵 M' 。命 M_0 是 M 的头 $n_1+n_2-2n_j$ 列, 我们得到

$$\begin{aligned} & (g_2^{\delta_2+1}/g_1 h_1^{\delta_1})^{n_2-n_j} (g_3^{\delta_3+1}/g_2 h_2^{\delta_2})^{n_3-n_j} \cdots (g_j^{\delta_j+1}/g_{j-1} h_{j-1}^{\delta_{j-1}})^{n_j-n_j} \det M_0 = \\ & \pm g_2^{n_1-n_3} g_3^{n_2-n_4} \cdots g_{j-1}^{n_{j-2}-n_j} g_j^{n_{j-1}-n_j} \end{aligned} \quad (23)$$

这个等式可干净利落地简化为

$$\det M_0 = \pm h_j \quad (24)$$

(这个证明尽管是对于整数环论述的,但明显地可应用于任何惟一因子分解整环。)

在验证算法 C 的过程中,我们也已经知道,由此算法所涉及的 S 的每个元素都可表达成为一个行列式,这行列式的元素是原来多项式的本原部分的系数。Hadamard(阿达马)的著名定理(见习题 15)指出

$$|\det(a_{ij})| \leq \prod_{1 \leq j \leq n} \left(\sum_{1 \leq i \leq n} a_{ij}^2 \right)^{1/2} \quad (25)$$

因此如果给定的多项式 $u(x)$ 和 $v(x)$ 的所有系数在绝对值上都以 N 为界,则在由算法 C 计算的多项式中出现的每个系数至多是

$$N^{m+n} (m+1)^{n/2} (n+1)^{m/2} \quad (26)$$

这同一上界可应用于在执行算法 E 期间计算的所有多项式 $u(x)$ 和 $v(x)$ 的系数上,因为在算法 E 中得到的诸多项式总是算法 C 中得到的多项式的因子。

关于系数的这一上界是极其令人满意的,因为它比起我们所能期望的要好得多。例如,设想如果我们在步骤 E3 和 C3 中不作校正,仅用 $r(x)$ 代替 $v(x)$ 会发生什么情况。这是最简单的 gcd 算法,而且它是传统地出现于代数教科书(用于理论目的,而不打算用于实际计算)中的一个算法。如果假设 $\delta_1 = \delta_2 = \dots = 1$,我们发现 $u_3(x)$ 的系数以 N^3 为界, $u_4(x)$ 的系数以 N^7 为界, $u_5(x)$ 的系数以 N^{17} 为界, \dots ; $u_k(x)$ 的系数以 N^{a_k} 为界,其中 $a_k = 2a_{k-1} + a_{k-2}$ 。于是,当 $m = n+1$ 时,上界将不是(26),而近似地是

$$N^{0.5(2.414)^n} \quad (27)$$

而且经验证明,这个简单的算法事实上确有这个特性;在每步中系数中的位数都按指数增长!相反,在算法 E 中,位数的增长顶多只比线性增长多一点。

我们对算法 C 进行的证明的另一个副产品是,多项式的次数在每步将几乎总是减 1,使得如果给定的多项式是“随机的”,步骤 C2(或 E2)的迭代次数将总是 $\deg(v)$ 。为了看看这为什么会发生,例如,注意在(17)和(18)中我们可选择 M 和 M' 的头 8 列,然后我们发现当且仅当 $d_3 = 0$,即当且仅当

$$\det \begin{pmatrix} a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 \\ 0 & a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 \\ 0 & 0 & a_8 & a_7 & a_6 & a_5 & a_4 & a_3 \\ b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 & 0 \\ 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \\ 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 \\ 0 & 0 & 0 & b_6 & b_5 & b_4 & b_3 & b_2 \\ 0 & 0 & 0 & 0 & b_6 & b_5 & b_4 & b_3 \end{pmatrix} = 0$$

时, $u_4(x)$ 的次数小于 3。一般来说,对于 $j > 1$,当且仅当在 $u(x)$ 和 $v(x)$ 的系数中一个类似的行列式为 0 时, δ_j 将大于 1。由于这样一个行列式是在这些系数中的非

零的多变量多项式,因此它“几乎总是”或“概率为1”地非零的(关于这个命题的更精确的表述,见习题16,关于有关的证明见习题4)。在(15)的多项式例子中, δ_2 和 δ_3 都等于2,所以它们确实是例外的。

上述的考虑可用来推导有名的事实,即两个多项式互素当且仅当它们的结式非零。结式是有表1中的行 A_5 到 A_0 及 B_7 到 B_0 的形式的行列式。(这是“Sylvester行列式”;见习题12。B. L. van der Waerden讨论了结式进一步的性质,见*Modern Algebra*, Fred Blum译(New York: Ungar, 1949),第27~28节。)从上边讨论的观点,我们可以说,gcd“几乎总是”零次的,因为Sylvester结式几乎从不为零。但是如果没有相当的机会使gcd是一个正次数多项式的话,许多有实用意义的计算就不会进行了。

通过考虑 $u(x) = w(x)u_1(x)$ 和 $v(x) = w(x)u_2(x)$,其中 $u_1(x)$ 和 $u_2(x)$ 互素且 $w(x)$ 是本原的,我们可以精确地看到,在算法E和C执行期间,当gcd不是1时会发生什么情况。当算法E对 $u(x) = u_1(x)$ 和 $v(x) = u_2(x)$ 进行工作时,如果得到多项式 $u_1(x), u_2(x), u_3(x), \dots$,则容易证明,对于 $u(x) = w(x)u_1(x)$ 和 $v(x) = w(x)u_2(x)$,所得到的序列只不过是 $w(x)u_1(x), w(x)u_2(x), w(x)u_3(x), w(x)u_4(x), \dots$ 等等。对于算法C,特性与此不同:如果当算法C应用于 $u(x) = u_1(x)$ 和 $v(x) = u_2(x)$ 时,得到多项式 $u_1(x), u_2(x), u_3(x), \dots$,而且如果我们假定 $\deg(u_{j+1}) = \deg(u_j) - 1$ (当 $j > 1$ 时它几乎总是真的),则当算法C应用于 $u(x) = w(x)u_1(x)$ 及 $v(x) = w(x)u_2(x)$ 时,得到序列

$$w(x)u_1(x), w(x)u_2(x), \ell^2 w(x)u_3(x), \ell^4 w(x)u_4(x), \ell^6 w(x)u_5(x), \dots \quad (28)$$

其中 $\ell = \ell(w)$ (见习题13)。尽管存在另外的 ℓ 因子,算法C还将是优越于算法E的,因为比起重复地计算本原部分来,倒不如处理稍微大一些的多项式更容易些。


诸如在算法C和E中那样的多项式余式序列不仅仅对求最大公因子有用;另一个重要的应用是按照J. Sturm的著名定理[*Mém. Présentées par Divers Savants* 6 (Paris: 1835), 271~318],对于一个给定的区间内的一个给定的多项式枚举实根的个数。设 $u(x)$ 是实数上的一个多项式,具有不同的复根。我们将在下节看到,根是不同的当且仅当 $\gcd(u(x), u'(x)) = 1$,其中 $u'(x)$ 是 $u(x)$ 的导数;因此,有一个多项式余式序列证明 $u(x)$ 与 $u'(x)$ 互素。置 $u_0(x) = u(x), u_1(x) = u'(x)$,而且(遵照Sturm定理)我们把所有余式的符号取反,对某些正常数 c_j 和 d_j 得到

$$\begin{aligned} c_1 u_0(x) &= u_1(x)q_1(x) - d_1 u_2(x) \\ c_2 u_1(x) &= u_2(x)q_2(x) - d_2 u_3(x) \\ &\vdots \\ c_k u_{k-1}(x) &= u_k(x)q_k(x) - d_k u_{k+1}(x) \end{aligned} \quad (29)$$

其中 $\deg(u_{k+1}) = 0$ 。我们说在 a 处 $u(x)$ 的变差 $V(u, a)$ 是在序列 $u_0(a), u_1(a), \dots, u_{k+1}(a)$ 中符号变化的个数,不计0。例如,如果符号序列是 $0, +, -, -, 0, +,$

$+$, $-$, 我们有 $V(u, a) = 3$ 。Sturm 定理断言, 在区间 $a < x \leq b$ 中, $u(x)$ 的根的个数为 $V(u, a) - V(u, b)$; 其证明惊人地短(见习题 22)。

尽管算法 C 和 E 是有趣的, 但它们不是惟一的方法。4.6.2 小节末尾讨论了在整数上计算多项式 gcd 另一些重要的方法。也有一个一般的计算行列式的算法, 它可以说是把算法 C 作为一个特殊情况; 见 E. H. Bareiss, *Math. Comp.* **22**(1968), 565~578。

 在本书的第 4 版中我计划重写本小节的剖析, 把 19 世纪关于行列式的研究, 以及 W. Habicht 的著作, *Comm. Math. Helvetici* **21**(1948), 99~116 考虑在内。对于后者的一个精彩的讨论已经由 R. Loos 在 *Computing*, Supplement 4 (1982), 115~137 中给出。C. L. Dodgson (aka Lewis Carroll) 从雅可比的一个定理出发导出的用于计算行列式的一个有趣的方法, 也同这些方法高度有关。有关子矩阵的行列式之间的恒等式早期历史的概述, 也请参见 D. E. Knuth, *Electronic J. Combinatorics* **3**, 2 (1996), 文章 R5, § 3。

习 题

1. [10] 当 $u(x) = x^6 + x^5 - x^4 + 2x^3 + 3x^2 - x + 2$ 和 $v(x) = 2x^3 + 2x^2 - x + 3$ 在整数上时, 计算伪商 $q(x)$ 和伪余式 $r(x)$, 即满足 (8) 的多项式。

2. [15] 在模 7 下, $3x^6 + x^5 + 4x^4 + 4x^3 + 3x^2 + 4x + 2$ 和它的“反序” $2x^6 + 4x^5 + 3x^4 + 4x^3 + 4x^2 + x + 3$ 的最大公因子是什么?

► 3. [M25] 证明一个域 S 上的多项式的欧几里得算法可被扩充来求 S 上的多项式 $U(x)$, $V(x)$, 使得(参考算法 4.5.2X)

$$u(x)V(x) + U(x)v(x) = \gcd(u(x), v(x))$$

由这个扩充的算法计算的多项式 $U(x)$ 和 $V(x)$ 的次数是多少? 证明如果 S 是有理数域, 而且 $u(x) = x^m - 1$ 和 $v(x) = x^n - 1$, 则扩充的算法产生整系数的多项式 $U(x)$ 和 $V(x)$ 。当 $u(x) = x^{21} - 1$, $v(x) = x^{13} - 1$ 时, 求 $U(x)$ 和 $V(x)$ 。

► 4. [M30] 设 p 为素数, 并假设把欧几里得算法应用于模 p 多项式 $u(x)$ 和 $v(x)$, 产生次数分别为 $m, n, n_1, \dots, n_t, -\infty$ 的多项式序列, 其中 $m = \deg(u)$, $n = \deg(v)$, 且 $n_i \geq 0$ 。假设 $m \geq n$ 。如果 $u(x)$ 和 $v(x)$ 是首一多项式, 且在次数分别为 m 和 n 的所有 p^{m+n} 对首一多项式上独立并一致地分布, 则作为 m, n 和 p 的函数, 三个量 $t, n_1 + \dots + n_t, (n - n_1)n_1 + \dots + (n_{t-1} - n_t)n_t$ 的平均值是多少? (这三个量是欧几里得算法应用于模 p 多项式时的运行时间中的基本要素, 其中假定除法是通过算法 D 完成的。)[提示: 证明 $u(x) \bmod v(x)$ 一致分布且与 $v(x)$ 无关。]

5. [M22] 如果 $u(x)$ 和 $v(x)$ 都是独立地一致分布的 n 次首一多项式, 则 $u(x)$ 和 $v(x)$ 模 p 互素的概率为何?

6. [M23] 我们已经看到, 整数的欧几里得算法可直接改编成求多项式的最大公因子的一个算法。能否以类似的方式把“二进 gcd 算法”(即算法 4.5.2B)改编为多项式算法?

7. [M10] 在一个惟一因子分解整环 S 上, 多项式整环中的所有单位元素是哪些?

► 8. [M22] 证明如果整系数的一个多项式在整数环上不可约, 则把它当做有理数域上的一个多项式时也是不可约的。

9. [M25] 设 $u(x)$ 和 $v(x)$ 是一个惟一因子分解整环上的本原多项式。证明当且仅当有 S 上的多项式 $u(x)$ 和 $v(x)$ 使得 $u(x)V(x) + U(x)v(x)$ 是一个次数为 0 的多项式时, $U(x)$ 和 $V(x)$ 互素。[提示:如同习题 3 中扩充算法 4.5.2A 那样扩充算法 E。]

10. [M28] 证明, 一个惟一因子分解整环上的多项式形成一个惟一因子分解整环。[提示:使用习题 9 的结果来帮助证明,至多可能有一种类型的因子分解。]

11. [M22] 如果次数序列已经是 $9, 6, 5, 2, -\infty$, 而不是 $8, 6, 4, 2, 1, 0$, 则表 1 中出现什么行名称?

► 12. [M24] 设 $u_1(x), u_2(x), u_3(x), \dots$ 是在算法 C 的一个运行期间得到的一个多项式序列。“Sylvester 矩阵”是由行 A_{n_2-1} 直到 A_0, B_{n_1-1} 直到 B_0 形成的一个方阵(在类似于表 1 的那种记法下)。证明如果 $u_1(x)$ 和 $u_2(x)$ 有一个正次数的公因子, 则 Sylvester 矩阵的行列式为 0; 反之, 若对某个 $k, \deg(u_k) = 0$, 试借助于 $\ell(u_j)$ 和 $\deg(u_j), 1 \leq j \leq k$, 推导它的绝对值的一个公式, 以证明 Sylvester 矩阵的行列式非零。

13. [M22] 证明, 当 $\delta_1 = \delta_2 = \dots = \delta_{k-1} = 1$ 时, 如同 (28) 中所示, $\gcd(u(x), v(x))$ 的本原部分的前导系数 ℓ 进入算法 C 的多项式序列。什么是一般的 δ_j 的特性?

14. [M29] 设 $r(x)$ 是当 $u(x)$ 伪除以 $v(x)$ 时的伪余式。如果 $\deg(u) \geq \deg(v) + 2$, 且 $\deg(v) \geq \deg(r) + 2$, 证明 $r(x)$ 是 $\ell(v)$ 的倍数。

15. [M26] 证明阿达马不等式 (25)。[提示:考虑矩阵 AA^t 。]

► 16. [M22] 设 $f(x_1, \dots, x_n)$ 是不恒等于零的一个多变量多项式, 且设 $r(S_1, \dots, S_n)$ 是 $f(x_1, \dots, x_n) = 0$ 的根 (x_1, \dots, x_n) 的集合, 使得 $x_1 \in S_1, \dots, x_n \in S_n$ 。如果 f 中变量 x_j 的次数至多是 $d_j, \leq |S_j|$, 证明

$$|r(S_1, \dots, S_n)| \leq |S_1| \cdots |S_n| - (|S_1| - d_1) \cdots (|S_n| - d_n)$$

因此, 当集合 S_j 变得更大时, 随机地求一个根的概率 $|r(S_1, \dots, S_n)| / |S_1| \cdots |S_n|$ 趋于零。[这个不等式在设计随机化算法时有许多应用, 因为它提供了检验和的积的复杂和是否为零而无须展开所有项的一个好方法。]

17. [M32] (对于串多项式除法的 P.M. Cohn 算法) 设 A 是一个字母表, 即符号的集合。 A 上的一个串 α 是 $n \geq 0$ 个符号的一个序列 $\alpha = a_1 \cdots a_n$, 其中每个 a_j 都在 A 中。 α 的长度以 $|\alpha|$ 表示, 是符号的个数 n 。 A 上的串多项式是有限和 $U = \sum_{j \geq 0} r_j \alpha_j$, 其中每个 r_j 是一非零有理数而且每个 α_j 是 A 上的一个串; 假定当 $j \neq k$ 时, $\alpha_j \neq \alpha_k$ 。如果 $U = 0$ (即, 如果和为空), 则 U 的次数 $\deg(U)$ 定义为 $-\infty$, 否则 $\deg(U) = \max |\alpha_k|$ 。串多项式的和与积以显然的方式定义; 因此 $(\sum_j r_j \alpha_j) \cdot (\sum_k s_k \beta_k) = \sum_{j,k} r_j s_k \alpha_j \beta_k$, 其中两个串的乘积通过简单地并列它们而得到, 然后我们收集同类项。例如, 如果 $A = \{a, b\}$, $U = ab + ba - 2a - 2b$, $V = a + b - 1$, 则 $\deg(U) = 2, \deg(V) = 1, V^2 = aa + ab + ba + bb - 2a - 2b + 1, V^2 - U = aa + bb + 1$ 。显然, $\deg(UV) = \deg(U) + \deg(V), \deg(U + V) \leq \max(\deg(U), \deg(V))$, 在后一公式中, 如果 $\deg(U) \neq \deg(V)$, 则等号成立。(除了在乘法之下变量不可交换外, 串多项式可以认为是有理数域上通常的多变量多项式。用通常的纯粹数学的语言来说, 具有如这里所定义的运算的串多项式集合, 是有理数上由 A 生成的“自由结合代数”。)

a) 设 Q_1, Q_2, U, V 是具有 $\deg(U) \geq \deg(V)$ 和 $\deg(Q_1 U - Q_2 V) < \deg(Q_1 U)$ 的串多项式。试给出一个算法来求一串多项式 Q , 使 $\deg(U - QV) < \deg(U)$ 。(于是如果给了我们 U 和 V , 使得对某个 Q_1 和 $Q_2, Q_1 U = Q_2 V + R$ 和 $\deg(R) < \deg(Q_1 U)$, 则这些条件有一个 $Q_1 = 1$ 解。)

b) 给定 U 和 V 是对于某个 Q_1 和 Q_2 具有 $\deg(V) > \deg(Q_1 U - Q_2 V)$ 的串多项式。证明 a)

的结果可被改进来求一个商 Q , 使得 $U = QV + R$, $\deg(R) < \deg(V)$. (对于串多项式, 这类似于 (1); a) 说明, 在较弱的假定下, 我们可以使得 $\deg(R) < \deg(U)$.)

c) 一个齐次多项式是这样多项式, 其中所有项有相同的次数(长度)。如果 U_1, U_2, V_1, V_2 是满足 $U_1 V_1 = U_2 V_2$ 和 $\deg(V_1) \geq \deg(V_2)$ 的齐次串多项式, 证明有一个齐次串多项式 U , 使得 $U_2 = U_1 U$ 和 $V_1 = UV_2$ 。

d) 给定 U 和 V 是满足 $UV = VU$ 的齐次串多项式, 证明对某些整数 m, n 和有理数 r, s , 有一个齐次串多项式 W 使得 $U = rW^m, V = sW^n$ 。试给出一个算法来计算有最大次数的这样一个 W 。(这个算法是有趣的, 例如, 当 $U = \alpha$ 和 $V = \beta$ 是满足 $\alpha\beta = \beta\alpha$ 的串时, 则 W 只不过是一个串 γ 。当 $U = x^m$ 和 $V = x^n$ 时, 最大次数的解是串 $W = x^{\gcd(m, n)}$, 所以这个算法包括了整数的 gcd 算法, 作为一个特殊情况。)

► 18. [M24] (串多项式的欧几里得算法) 设 V_1 和 V_2 是不全为 0 的串多项式, 且具有一个公共的左倍串。(这意味着存在不全为零的串多项式 U_1 和 U_2 , 使得 $U_1 V_1 = U_2 V_2$ 。) 本题的目的是求一个算法来计算它们的最大公共右因串 $\gcd(V_1, V_2)$, 以及最小公共左倍串 $\text{lcm}(V_1, V_2)$ 。后面的量定义如下: $\gcd(V_1, V_2)$ 是 V_1 和 V_2 的公共右因串(即对某个 W_1 和 $W_2, V_1 = W_1 \gcd(V_1, V_2)$ 和 $V_2 = W_2 \gcd(V_1, V_2)$), 而且 V_1 和 V_2 的任何公共右因串是 $\gcd(V_1, V_2)$ 的一个右因串; 对某个 Z_1 和 $Z_2, \text{lcm}(V_1, V_2) = Z_1 V_1 = Z_2 V_2$, 而且 V_1 和 V_2 的任何公共左倍串是 $\text{lcm}(V_1, V_2)$ 的一个左倍串。

例如, 设 $U_1 = abbbab + abbab + bbab + ab - 1, V_1 = babab + abab + ab - b; U_2 = abb + ab - b, V_2 = babbabab + bababab + babab + abab - babb - 1$ 。则我们有 $U_1 V_1 = U_2 V_2 = abbbabbabab + abbabbabab + abbbababab + abbababab - bbabbabab + abbbubab - bbababab + 2abbabab - abbbubb + ababab - abbab - bbabab - babab + bbabb - abb - ab + b$ 。对于这些串多项式, 可以证明 $\gcd(V_1, V_2) = ab + 1$, 且 $\text{lcm}(V_1, V_2) = U_1 V_1$ 。

习题 17 的除法算法可改述如下: 如果 V_1 和 V_2 是串多项式, $V_2 \neq 0$, 而且如果 $U_1 \neq 0$ 和 U_2 满足等式 $U_1 V_1 = U_2 V_2$, 则存在串多项式 Q 和 R 使得

$$V_1 = QV_2 + R, \quad \text{其中 } \deg(R) < \deg(V_2)$$

容易推出, Q 和 R 是惟一确定的; 它们不依赖于给定的 U_1 和 U_2 。而且在下列意义下, 结果是左右对称的:

$$U_2 = U_1 Q + R', \quad \text{其中 } \deg(R') = \deg(U_1) - \deg(V_2) + \deg(R) < \deg(U_1)$$

证明这一除法算法可被推广成一个这样的算法, 它计算 $\text{lcm}(V_1, V_2)$ 及 $\gcd(V_1, V_2)$; 事实上, 推广的算法求串多项式 Z_1, Z_2 , 使得 $Z_1 V_1 + Z_2 V_2 = \gcd(V_1, V_2)$ 。[提示: 使用辅助变量 $u_1, u_2, v_1, v_2, w_1, w_2, w'_1, w'_2, z_1, z_2, z'_1, z'_2$, 它们的值是串多项式; 从置 $u_1 \leftarrow U_1, u_2 \leftarrow U_2, v_1 \leftarrow V_1, v_2 \leftarrow V_2$ 开始, 并且在整个算法中每当进行第 n 次迭代时, 维持条件

$$\begin{aligned} U_1 w_1 + U_2 w_2 &= u_1, & z_1 V_1 + z_2 V_2 &= v_1 \\ U_1 w'_1 + U_2 w'_2 &= u_2, & z'_1 V_1 + z'_2 V_2 &= v_2 \\ u_1 z_1 - u_2 z'_1 &= (-1)^n U_1, & w_1 v_1 - w'_1 v_2 &= (-1)^n V_1 \\ -u_1 z_2 + u_2 z'_2 &= (-1)^n U_2, & -w_2 v_1 + w'_2 v_2 &= (-1)^n V_2 \end{aligned}$$

这可以认为是欧几里得算法的“最终的”扩充。]

19. [M39] (方阵的公共因子) 习题 18 表明, 当乘法不可交换时, 最大公共右因子的概念可以是 有意义的。证明任何两个 $n \times n$ 阶整数矩阵 A 和 B 有最大公共右矩阵因子 D 。[建议: 设计一

个其输入为 A 和 B , 且其输出为整数矩阵 D, P, Q, X, Y 的算法, 其中 $A = PD, B = QD, D = XA + YB$ 。]求矩阵 $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ 和 $\begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix}$ 的最大公共右因子。

20. [M40] 研究欧几里得算法的精度: 关于计算其系数是浮点数的多项式的最大公因子, 你能说些什么?

21. [M25] 如果诸给定多项式系数的绝对值以 N 为界, 证明为计算整数上两个 n 次多项式的 \gcd , 算法 C 所要求的计算时间是 $O(n^4(\log Nn)^3)$ 。

22. [M23] 证明 Sturm 定理。[提示: 某些符号序列是不可能的。]

23. [M22] 证明如果 (29) 中的 $u(x)$ 有 $\deg(u)$ 个实根, 则对于 $0 \leq j \leq k$, 我们有 $\deg(u_{j+1}) = \deg(u_j) - 1$ 。

24. [M21] 证明 (19) 可简化为 (20), (23) 可简化为 (24)。

25. [M24] (W. S. Brown) 证明对于 $j \geq 3$, (16) 中的所有多项式 $u_j(x)$ 是 $\gcd(\ell(u), \ell(v))$ 的倍数, 并且说明由此如何改进算法 C。

► 26. [M26] 本题的目的是: 给出关于下列事实在多项式中的类似结果, 即, 正整数元素的连分数给出对实数的最好近似 (习题 4.5.3-42)。

设 $u(x)$ 和 $v(x)$ 是一个域上的多项式, $\deg(u) > \deg(v)$, 并设当欧几里得算法应用于 $u(x)$ 和 $v(x)$ 时, $a_1(x), a_2(x), \dots$ 为商多项式。例如, (5) 和 (6) 中的商序列是 $9x^2 + 7, 5x^2 + 5, 6x^3 + 5x^2 + 6x + 5, 9x + 12$ 。我们希望证明连分数 $// a_1(x), a_2(x), \dots //$ 的收敛项 $p_n(x)/q_n(x)$ 是有理函数 $v(x)/u(x)$ 的低次“最好近似”, 其中借助于等式 4.5.3-(4) 的连续多项式, 我们有 $p_n(x) = K_{n-1}(a_2(x), \dots, a_n(x))$, 且 $q_n(x) = K_n(a_1(x), \dots, a_n(x))$ 。根据约定, 我们命 $p_0(x) = q_{-1}(x) = 0, p_{-1}(x) = q_0(x) = 1$ 。

证明: 如果 $p(x)$ 和 $q(x)$ 都是使得对某个 $n \geq 1, \deg(q) < \deg(q_n)$ 及 $\deg(pu - qv) \leq \deg(p_{n-1}u - q_{n-1}v)$ 的多项式, 则对某个常数 $c, p(x) = cp_{n-1}(x)$ 和 $q(x) = cq_{n-1}(x)$ 。特别是, 每个 $q_n(x)$ 在下列定义下是一个“破记录”的多项式, 就是对于任何多项式 $p(x)$, 没有较小次数的非零多项式 $q(x)$ 能使 $p(x)u(x) - q(x)v(x)$ 达到像 $p_n(x)u(x) - q_n(x)v(x)$ 那样小的次数。

27. [M23] 当我们预先知道余式将为零时, 提出一个加速 $u(x)$ 除以 $v(x)$ 的方法。

* 4.6.2 多项式的因子分解

现在让我们考虑多项式的因子分解问题, 而不仅仅求两个或多个多项式的最大公因子。

以模 p 进行因子分解 如同在整数情况下那样 (4.5.2, 4.5.4 小节), 因子分解的问题似乎比求最大公因子更为困难。但是在模一个素整数 p 之下进行多项式的因子分解, 并不像我们所意料的那样困难。比起使用任何已知的方法求一任意 n 位二进数值的因子来, 求任意 n 次多项式在模 2 之下的因子, 要容易得多。这令人惊奇的情况是 1967 年 Elwyn R. Berlekamp 所发现的一个启发性的因子分解算法的推论 [Bell System Technical J. 46 (1967), 1853~1859]。

设 p 是一个素数; 以下讨论中对于多项式的所有算术都将在模 p 之下进行。假设有人已经给了我们一个多项式 $u(x)$, 其系数是从集合 $\{0, 1, \dots, p-1\}$ 中选择的; 我们可以假定 $u(x)$ 是首一的。我们的目标是把 $u(x)$ 表达成形式

$$u(x) = p_1(x)^{e_1} \cdots p_r(x)^{e_r} \quad (1)$$

其中 $p_1(x), \dots, p_r(x)$ 是不同的首一不可约多项式。

作为头一步,我们可以使用一个标准技术来确定 e_1, \dots, e_r 中是否有大于 1 者。如果

$$u(x) = u_n x^n + \cdots + u_0 = v(x)^2 w(x) \quad (2)$$

则它(以通常方式,但在模 p 下)形成的导数是

$$u'(x) = nu_n x^{n-1} + \cdots + u_1 = 2v(x)v'(x)w(x) + v(x)^2 w'(x) \quad (3)$$

而且这是平方后的因子 $v(x)$ 的倍数,因此我们分解 $u(x)$ 的头一步是构造

$$\gcd(u(x), u'(x)) = d(x) \quad (4)$$

如果 $d(x)$ 等于 1,则我们知道 $u(x)$ 是无平方的,是不同素因子 $p_1(x), \dots, p_r(x)$ 的乘积。如果 $d(x)$ 不等于 1,而且 $d(x) \neq u(x)$,则 $d(x)$ 是 $u(x)$ 的一个真因子;在 $d(x)$ 的因子和 $u(x)/d(x)$ 的因子之间的关系在这种情况下极好地加速了因子分解过程(见习题 34 和 36)。最后,如果 $d(x) = u(x)$,则必然有 $u'(x) = 0$;因此 x^k 的系数 u_k 仅当 k 是 p 的倍数时为非零。这意味着 $u(x)$ 可以写成形如 $v(x^p)$ 的一个多项式,而且在这样的情况下,我们有

$$u(x) = v(x^p) = (v(x))^p \quad (5)$$

这一因子分解的过程可以通过求 $v(x)$ 的不可约因子并求它们的 p 次方来完成。

恒等式 (5) 对于读者可能显得有些陌生;它是一个重要的事实,它对于 Berlekamp 的算法和我们所要讨论的好些个其它方法说来是基本的。可以证明它如下:如果 $v_1(x)$ 和 $v_2(x)$ 是任意模 p 多项式,则

$$\begin{aligned} (v_1(x) + v_2(x))^p &= v_1(x)^p + \binom{p}{1} v_1(x)^{p-1} v_2(x) + \cdots + \\ &\quad \binom{p}{p-1} v_1(x) v_2(x)^{p-1} + v_2(x)^p = v_1(x)^p + v_2(x)^p \end{aligned}$$

因为二项式系数 $\binom{p}{1}, \dots, \binom{p}{p-1}$ 都是 p 的倍数。而且,如果 a 是任意整数,则由费马定理 $a^p \equiv a \pmod{p}$ 。因此,当 $v(x) = v_m x^m + v_{m-1} x^{m-1} + \cdots + v_0$ 时,我们求得

$$\begin{aligned} v(x)^p &= (v_m x^m)^p + (v_{m-1} x^{m-1})^p + \cdots + (v_0)^p = \\ &= v_m x^{mp} + v_{m-1} x^{(m-1)p} + \cdots + v_0 = v(x^p) \end{aligned}$$

上述注释说明了,分解一个多项式的问题归结为分解一个无平方的多项式的问题,因此我们假定

$$u(x) = p_1(x) p_2(x) \cdots p_r(x) \quad (6)$$

是不同素元的乘积。当仅给出 $u(x)$ 时,我们怎么能巧妙地发现诸 $p_j(x)$ 呢? Berlekamp 的思想是利用中国剩余定理,如同这个定理对于整数都是正确的一样,它对多项式也正确(见习题 3)。如果 (s_1, s_2, \dots, s_r) 是模 p 的整数 r 元组,则中国剩余

定理意味着有一惟一的多项式 $v(x)$ 使得

$$\begin{aligned} v(x) &\equiv s_1 \pmod{p_1(x)}, \dots, v(x) \equiv s_r \pmod{p_r(x)} \\ \deg(v) &< \deg(p_1) + \deg(p_2) + \dots + \deg(p_r) = \deg(u) \end{aligned} \quad (7)$$

这里出现的记法“ $g(x) \equiv h(x) \pmod{f(x)}$ ”和在习题 3.2.2-11 中的“ $g(x) \equiv h(x) \pmod{f(x) \text{ 和 } p}$ ”有相同的定义,因为我们正在考虑模 p 之下的多项式算术。(7)中的多项式 $v(x)$ 给了我们一个得到 $u(x)$ 的因子的办法,因为如果 $r \geq 2$, 且 $s_1 \neq s_2$, 则 $\gcd(u(x), v(x) - s_1)$ 可为 $p_1(x)$, 但不为 $p_2(x)$ 所整除。

由于这一发现表明可以从(7)的适当的解得到关于 $u(x)$ 的因子的信息,让我们更仔细地分析(7)。首先,我们可以发现对于 $1 \leq j \leq r$, 多项式 $v(x)$ 满足条件 $v(x)^p \equiv s_j^p = s_j \equiv v(x) \pmod{p_j(x)}$, 因此

$$v(x)^p \equiv v(x) \pmod{u(x)}, \quad \deg(v) < \deg(u) \quad (8)$$

其次,有基本的恒等式

$$x^p - x \equiv (x - 0)(x - 1) \cdots (x - (p - 1)) \pmod{p} \quad (9)$$

(见习题 6); 因此在模 p 之下进行工作时

$$v(x)^p - v(x) = (v(x) - 0)(v(x) - 1) \cdots (v(x) - (p - 1)) \quad (10)$$

是对于任何多项式 $v(x)$ 的一个恒等式。如果 $v(x)$ 满足(8), 则 $u(x)$ 应能整除(10)的左边, 所以 $u(x)$ 的每个不可约因子必然整除(10)右边的 p 个互素的因子之一。换句话说, 对于某些 s_1, s_2, \dots, s_r , (8)的所有解必定有(7)的形式; 恰好有 p^r 个(8)的解。

因此, 同余式(8)的解 $v(x)$ 提供了对 $u(x)$ 进行因子分解的一把钥匙。首先求(8)的所有解似乎比分解 $u(x)$ 要更困难, 但事实上并非如此, 因为(8)的解的集合在加法之下是封闭的。设 $\deg(u) = n$; 我们可以构造 $n \times n$ 矩阵

$$Q = \begin{pmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,n-1} \\ \vdots & \vdots & & \vdots \\ q_{n-1,0} & q_{n-1,1} & \cdots & q_{n-1,n-1} \end{pmatrix} \quad (11)$$

其中

$$x^{pk} \equiv q_{k,n-1}x^{n-1} + \cdots + q_{k,1}x + q_{k,0} \pmod{u(x)} \quad (12)$$

则 $v(x) = v_{n-1}x^{n-1} + \cdots + v_1x + v_0$ 是(8)的解当且仅当

$$(v_0, v_1, \dots, v_{n-1})Q = (v_0, v_1, \dots, v_{n-1}) \quad (13)$$

因为后一等式成立, 当且仅当

$$\begin{aligned} v(x) &= \sum_j v_j x^j = \sum_j \sum_k v_k q_{k,j} x^j = \\ &= \sum_k v_k x^{pk} = v(x^p) \equiv v(x)^p \pmod{u(x)} \end{aligned}$$

因此, Berlekamp 分解算法如下:

B1. 确保 $u(x)$ 是无平方的; 换句话说, 如果 $\gcd(u(x), u'(x)) \neq 1$, 则按照这小节早先所指出的方法简化分解 $u(x)$ 的问题。

B2. 构造(11)和(12)所定义的矩阵。这可以如同下边所说明的那样,依赖于 p 是否很大,以两个方式之一来完成。

B3. “三角化”矩阵 $Q - I$, 其中 $I = (\delta_{ij})$ 是 $n \times n$ 阶单位矩阵, 求它的秩数 $n - r$ 和线性无关向量 $v^{[1]}, \dots, v^{[r]}$, 使得对于 $1 \leq j \leq r$, $v^{[j]}(Q - I) = (0, 0, \dots, 0)$ 。(第一个向量 $v^{[1]}$ 总可以取作 $(1, 0, \dots, 0)$, 表示(8)的平凡解 $v^{[1]}(x) = 1$ 。计算可以利用适当的列操作来进行, 如同在下边的算法 N 中所说明的那样。)这时, r 是 $u(x)$ 的不可约因子的个数, 因为(8)的解是对于所有满足 $0 \leq t_1, \dots, t_r < p$ 的整数选择对应于向量 $t_1 v^{[1]} + \dots + t_r v^{[r]}$ 的 p^r 个多项式。因此, 如果 $r = 1$, 则 $u(x)$ 是不可约的, 且过程终止。

B4. 对于 $0 \leq s < p$, 计算 $\gcd(u(x), v^{[2]}(x) - s)$, 其中 $v^{[2]}(x)$ 是由向量 $v^{[2]}$ 表示的多项式。这个结果将是 $u(x)$ 的一个非平凡的因子分解, 因为 $v^{[2]}(x) - s$ 是非 0 的, 而且次数小于 $\deg(u)$, 由习题 7, 每当 $v(x)$ 满足(8)时, 我们有

$$u(x) = \prod_{0 \leq s < p} \gcd(v(x) - s, u(x)) \quad (14)$$

如果未能成功地使用 $v^{[2]}(x)$ 把 $u(x)$ 分解成 r 个因子, 则通过对于 $0 \leq s < p$ 和迄今为止找到的所有因子 $w(x)$, 及 $k = 3, 4, \dots$, 计算 $\gcd(v^{[k]}(x) - s, w(x))$, 能够得到进一步的因子, 直至得到 r 个因子为止。(如果在(7)中选择 $s_i \neq s_j$, 则我们得到(8)的一个解 $v(x)$, 它把 $p_i(x)$ 与 $p_j(x)$ 区别开来; 某个 $v^{[k]}(x) - s$ 将可为 $p_i(x)$ 而不为 $p_j(x)$ 所整除, 所以这一过程最终将找到所有的因子。)

如果 p 是 2 或 3, 则步骤 B4 的计算十分有效; 但如果 p 很大, 比如说大于 25, 则如同我们后边将看到的那样, 有好得多的方法来做。■

历史注记: M. C. R. Butler [Quart. J. Math. 5 (1954), 102~107] 发现对应于有 r 个不可约因子的一个无平方多项式的矩阵 $Q - I$, 有秩 $n - r$, 模 p 。实际上, 这一事实蕴涵在 K. Petr [Časopis pro Pěstování Matematiky a Fysiky 66 (1937), 85~94] 的一个更一般的结果当中, Petr 确定了 Q 的特征多项式, 也参见 Š. Schwarz, Quart. J. Math. 7 (1956), 110~124。

作为算法 B 的例子, 现在让我们确定在模 13 下

$$u(x) = x^8 + x^6 + 10x^4 + 10x^3 + 8x^2 + 2x + 8 \quad (15)$$

的因子分解(这一多项式在 4.6.1 小节的若干例子中出现)。使用算法 4.6.1E 的快速计算表明, $\gcd(u(x), u'(x)) = 1$; 因此, $u(x)$ 是无平方的, 我们再转到步骤 B2。步骤 B2 包含计算矩阵 Q , 在现在情况下它是 8×8 阶的阵列。 Q 的头一行总是 $(1, 0, 0, \dots, 0)$, 表示多项式 $x^0 \bmod u(x) = 1$ 。第二行表示 $x^{13} \bmod u(x)$, 而且, (对于较小的 k 值)一般地, $x^k \bmod u(x)$ 可以容易地确定如下: 如果

$$u(x) = x^n + u_{n-1}x^{n-1} + \dots + u_1x + u_0$$

而且如果

$$x^k \equiv a_{k,n-1}x^{n-1} + \cdots + a_{k,1}x + a_{k,0} \pmod{u(x)}$$

则

$$\begin{aligned} x^{k+1} &\equiv a_{k,n-1}x^n + \cdots + a_{k,1}x^2 + a_{k,0}x \equiv \\ &a_{k,n-1}(-u_{n-1}x^{n-1} - \cdots - u_1x - u_0) + a_{k,n-2}x^{n-1} + \cdots + a_{k,0}x \equiv \\ &a_{k+1,n-1}x^{n-1} + \cdots + a_{k+1,1}x + a_{k+1,0} \end{aligned}$$

其中

$$a_{k+1,j} = a_{k,j-1} - a_{k,n-1}u_j \quad (16)$$

在这个公式中把 $a_{k,-1}$ 当做 0 来处理, 所以 $a_{k+1,0} = -a_{k,n-1}u_0$ 。简单的“移位寄存器”递推式(16)使得对于 $k=1, 2, 3, \dots, (n-1)p$ 容易计算 $x^k \bmod u(x)$ 。当然, 在一台计算机里边, 这一计算一般是通过维持一个一维数组 $(a_{n-1}, \dots, a_1, a_0)$ 并反复地置

$$t \leftarrow a_{n-1}, a_{n-1} \leftarrow (a_{n-2} - tu_{n-1}) \bmod p, \dots, a_1 \leftarrow (a_0 - tu_1) \bmod p$$

以及 $a_0 \leftarrow (-tu_0) \bmod p$ 来进行。(我们已经看见过同随机数生成相联系的类似过程, 3.2.2-(10)。)对于(15)中所举的多项式例子 $u(x)$, 利用模 13 算术, 可得到下列 $x^k \bmod u(x)$ 的系数序列:

k	$a_{k,7}$	$a_{k,6}$	$a_{k,5}$	$a_{k,4}$	$a_{k,3}$	$a_{k,2}$	$a_{k,1}$	$a_{k,0}$
0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	0
2	0	0	0	0	0	1	0	0
3	0	0	0	0	1	0	0	0
4	0	0	0	1	0	0	0	0
5	0	0	1	0	0	0	0	0
6	0	1	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0
8	0	12	0	3	3	5	11	5
9	12	0	3	3	5	11	5	0
10	0	4	3	2	8	0	2	8
11	4	3	2	8	0	2	8	0
12	3	11	8	12	1	2	5	7
13	11	5	12	10	11	7	1	2

因此 Q 的第二行是 $(2, 1, 7, 11, 10, 12, 5, 11)$ 。类似地, 可以确定 $x^{26} \bmod u(x), \dots, x^{91} \bmod u(x)$, 而且求得

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 7 & 11 & 10 & 12 & 5 & 11 \\ 3 & 6 & 4 & 3 & 0 & 4 & 7 & 2 \\ 4 & 3 & 6 & 5 & 1 & 6 & 2 & 3 \\ 2 & 11 & 8 & 8 & 3 & 1 & 3 & 11 \\ 6 & 11 & 8 & 6 & 2 & 7 & 10 & 9 \\ 5 & 11 & 7 & 10 & 0 & 11 & 7 & 12 \\ 3 & 3 & 12 & 5 & 0 & 11 & 9 & 12 \end{bmatrix} \quad (17)$$

$$Q - I = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 7 & 11 & 10 & 12 & 5 & 11 \\ 3 & 6 & 3 & 3 & 0 & 4 & 7 & 2 \\ 4 & 3 & 6 & 4 & 1 & 6 & 2 & 3 \\ 2 & 11 & 8 & 8 & 2 & 1 & 3 & 11 \\ 6 & 11 & 8 & 6 & 2 & 6 & 10 & 9 \\ 5 & 11 & 7 & 10 & 0 & 11 & 6 & 12 \\ 3 & 3 & 12 & 5 & 0 & 11 & 9 & 11 \end{bmatrix}$$

这就结束了步骤 B2; Berlekamp 过程的下一步要求 $Q - I$ 的“零空间”。一般地说, 假设 A 是一个域上的 $n \times n$ 阶矩阵, 它的秩数 $n - r$ 有待确定; 进一步假设我们希望确定线性无关向量 $v^{[1]}, v^{[2]}, \dots, v^{[r]}$, 使得 $v^{[1]}A = v^{[2]}A = \dots = v^{[r]}A = (0, \dots, 0)$ 。对于这项计算的一个算法, 可以这样的观察为基础, 即 A 的任何列可以乘以一个非 0 的量, 而且它的一个列的任何倍数都可加到不同的列上, 而不改变向量 $v^{[1]}, \dots, v^{[r]}$ 的秩数 (这些变换相当于以 AB 代替 A , 其中 B 是一个非奇异矩阵)。因此可以使用下列熟知的“三角化”过程:

算法 N (零空间算法) 设 A 是一个 $n \times n$ 阶矩阵, 其元素 a_{ij} 属于一个域, 下标在 $0 \leq i, j < n$ 范围内。这个算法输出 r 个向量 $v^{[1]}, \dots, v^{[r]}$, 它们是在域上线性无关的, 而且满足 $v^{[j]}A = (0, \dots, 0)$, 其中 $n - r$ 是 A 的秩数。

N1. [初始化] 置 $c_0 \leftarrow c_1 \leftarrow \dots \leftarrow c_{n-1} \leftarrow -1, r \leftarrow 0$ 。(在此计算期间, 仅当 $a_{ij} = -1$ 及行 c_j 的所有其它元素都为 0 时, 才有 $c_j \geq 0$ 。)

N2. [对 k 进行循环] 对于 $k = 0, 1, \dots, n - 1$ 进行步骤 N3, 而后终止此算法。

N3. [就相关性扫描行] 如果在 $0 \leq j < n$ 的范围内, 有某个 j , 使得 $a_{kj} \neq 0$ 和 $c_j < 0$, 则执行下列过程: 以 $-1/a_{kj}$ 乘 A 的第 j 列 (使得 a_{kj} 变成 -1); 然后对所有 $i \neq j$, 把第 j 列的 a_{ki} 倍加到第 i 列上去; 最后, 置 $c_j \leftarrow k$ 。(因为不难证明, 对所有 $s < k, a_{sj} = 0$, 这些运算不影响 A 的 $0, 1, \dots, k - 1$ 行。)

另一方面, 如果在 $0 \leq j < n$ 的范围内没有 j 使得 $a_{kj} \neq 0$ 和 $c_j < 0$, 则置 $r \leftarrow r + 1$, 而且输出向量

$$v^{(k)} = (v_0, v_1, \dots, v_{n-1})$$

它由规则

$$v_j = \begin{cases} a_{kj}, & \text{如果 } c_s = j \geq 0 \\ 1, & \text{如果 } j = k \\ 0, & \text{否则} \end{cases} \quad (18)$$

所定义。

有一个例子可以揭示这一算法的机理。设 A 是整数模 13 域上 (17) 的矩阵 $Q - I$ 。当 $k=0$ 时, 我们输出向量 $v^{[1]} = (1, 0, 0, 0, 0, 0, 0, 0)$ 。当 $k=1$ 时, 我们可以在步骤 N3 中取 j 等于 0, 2, 3, 4, 5, 6 或 7 之一; 这里的选择完全是任意的, 尽管它影响由本算法输出的具体向量。对于手算, 最方便的是取 $j=5$, 因为已有 $a_{15} = 12 = -1$; 步骤 N3 的列操作则把 A 变成为矩阵

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \textcircled{12} & 0 & 0 \\ 11 & 6 & 5 & 8 & 1 & 4 & 1 & 7 \\ 3 & 3 & 9 & 5 & 9 & 6 & 6 & 4 \\ 4 & 11 & 2 & 6 & 12 & 1 & 8 & 9 \\ 5 & 11 & 11 & 7 & 10 & 6 & 1 & 10 \\ 1 & 11 & 6 & 1 & 6 & 11 & 9 & 3 \\ 12 & 3 & 11 & 9 & 6 & 11 & 12 & 2 \end{pmatrix}$$

(在“5”列“1”行上画了圈的元素用来指示 $c_5 = 1$ 。记住, 算法 N 对于矩阵的行和列的编号, 是从 0 而不是从 1 开始的。) 当 $k=2$ 时, 我们可以选择 $j=4$ 并以类似的方式进行, 而得到下列各矩阵, 它们均有和 $Q - I$ 相同的零空间:

$$\begin{array}{cc} k=2 & k=3 \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \textcircled{12} & 0 & 0 \\ 0 & 0 & 0 & 0 & \textcircled{12} & 0 & 0 & 0 \\ 8 & 1 & 3 & 11 & 4 & 9 & 10 & 6 \\ 2 & 4 & 7 & 1 & 1 & 5 & 9 & 3 \\ 12 & 3 & 0 & 5 & 3 & 5 & 4 & 5 \\ 0 & 1 & 2 & 5 & 7 & 0 & 3 & 0 \\ 11 & 6 & 7 & 0 & 7 & 0 & 6 & 12 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \textcircled{12} & 0 & 0 \\ 0 & 0 & 0 & 0 & \textcircled{12} & 0 & 0 & 0 \\ 0 & \textcircled{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 9 & 8 & 9 & 11 & 8 & 8 & 5 \\ 1 & 10 & 4 & 11 & 4 & 4 & 0 & 0 \\ 5 & 12 & 12 & 7 & 3 & 4 & 6 & 7 \\ 2 & 7 & 2 & 12 & 9 & 11 & 11 & 2 \end{pmatrix} \end{array}$$

$$\begin{array}{c}
 k = 4 \qquad \qquad \qquad k = 5 \\
 \left(\begin{array}{cccccccc}
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & \textcircled{12} & 0 & 0 \\
 0 & 0 & 0 & 0 & \textcircled{12} & 0 & 0 & 0 \\
 0 & \textcircled{12} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & \textcircled{12} \\
 1 & 10 & 4 & 11 & 4 & 4 & 0 & 0 \\
 8 & 2 & 6 & 10 & 11 & 11 & 0 & 9 \\
 1 & 6 & 4 & 11 & 2 & 0 & 0 & 10
 \end{array} \right) \quad \left(\begin{array}{cccccccc}
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & \textcircled{12} & 0 & 0 \\
 0 & 0 & 0 & 0 & \textcircled{12} & 0 & 0 & 0 \\
 0 & \textcircled{12} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & \textcircled{12} \\
 \textcircled{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 5 & 0 & 0 & 0 & 5 & 5 & 0 & 9 \\
 12 & 9 & 0 & 0 & 11 & 9 & 0 & 10
 \end{array} \right)
 \end{array}$$

现在没有画圈元素的每一列全为 0;所以当 $k=6$ 和 $k=7$ 时,此算法另外输出两个向量,即

$$v^{[2]} = (0, 5, 5, 0, 9, 5, 1, 0), \quad v^{[3]} = (0, 9, 11, 9, 10, 12, 0, 1)$$

在 $k=5$ 之后,由矩阵 A 的形式可知,这些向量显然满足方程 $vA = (0, \dots, 0)$ 。因为这个计算产生了三个线性无关的向量,故 $u(x)$ 必然恰有三个不可约的因子。

最后,我们可以转到分解过程的步骤 B4。对于 $0 \leq s < 13$ 计算 $\gcd(u(x), v^{[2]}(x) - s)$, 其中 $v^{[2]}(x) = x^6 + 5x^5 + 9x^4 + 5x^2 + 5x$, 当 $s=0$ 时给出答案 $x^5 + 5x^4 + 9x^3 + 5x + 5$, 当 $s=2$ 时给出 $x^3 + 8x^2 + 4x + 12$;而对于 s 的其它值 \gcd 是 1。因此 $v^{[2]}(x)$ 仅仅给了三个因子中的两个。转到 $\gcd(v^{[3]}(x) - s, x^5 + 5x^4 + 9x^3 + 5x + 5)$, 其中 $v^{[3]}(x) = x^7 + 12x^5 + 10x^4 + 9x^3 + 11x^2 + 9x$, 当 $s=6$ 时,我们得到值 $x^4 + 2x^3 + 3x^2 + 4x + 6$, 当 $s=8$ 时,得到 $x+3$, 当 s 为其它值时得到 1。于是完全的因子分解是

$$u(x) = (x^4 + 2x^3 + 3x^2 + 4x + 6)(x^3 + 8x^2 + 4x + 12)(x + 3) \quad (19)$$

现在让我们来估计当在模 p 下分解 n 次多项式时的 Berlekamp 方法的运行时间。首先假定 p 比较小,使得四种算术运算在模 p 下可以在实际上固定长度的时间内完成。(通过如习题 9 中所提议的那样存储一份倒数表,模 p 除法可以转换成乘法;例如当以模 13 进行工作时,我们有 $\frac{1}{2} = 7, \frac{1}{3} = 9$ 等等。)步骤 B1 中的计算花费 $O(n^2)$ 个时间单位;步骤 B2 花费 $O(pn^2)$ 个。对于步骤 B3 我们使用算法 N,它顶多要求 $O(n^3)$ 个时间单位。最后,在步骤 B4 中,我们可以发现,用欧几里得算法计算 $\gcd(f(x), g(x))$ 花费 $O(\deg(f)\deg(g))$ 个时间单位;因此对于固定的 j 和 s 以及对于迄今业已找到的 $u(x)$ 的所有因子 $w(x)$, 计算 $\gcd(v^{[j]}(x) - s, w(x))$ 花费 $O(n^2)$ 个时间单位。因而步骤 B4 至多要求 $O(prn^2)$ 个时间单位。当 p 是小的素数时, Berlekamp 过程在 $O(n^3 + prn^2)$ 步内分解在模 p 之下的任意 n 次多项式;而习题 5 表明,平均的因子数 r 近似于 $\ln n$ 。因此这个算法要比分解 p 进系统中 n 位数的任何已知方法都要快得多。

当然,当 n 和 p 很小时,一个类似于算法 4.5.4A 的试错法因子分解过程甚至比 Berlekamp 的方法还要快。习题 1 意味着,即使当 n 很大时,在进行任何更复杂的过程之前,先把小次数的因子剔出来是一个好主意。

当 p 很大时,将使用 Berlekamp 过程的另一种不同的实现办法进行计算。模 p 除法不再使用任何辅助的倒数表来进行,大概将代之以使用习题 4.5.2-16 的方法,它花费 $O((\log p)^2)$ 步。然后步骤 B1 将花费 $O(n^2(\log p)^2)$ 个单位时间;类似地,步骤 B3 花费 $O(n^3(\log p)^2)$ 个单位时间。在步骤 B2 中,当 p 很大时,我们可用比 (16) 更有效的方式构造 $x^p \bmod u(x)$; 4.6.3 小节说明,这个值实际上可以通过使用 $O(\log p)$ 个平方模 $u(x)$ 的操作,从 $x^k \bmod u(x)$ 变成 $x^{2^k} \bmod u(x)$,并且同乘以 x 的操作一起而得到。如果我们对于 $m = n, n+1, \dots, 2n-2$ 首先造一份 $x^m \bmod u(x)$ 的辅助表,则这一平方操作相对说来是容易实施的;如果 $x^k \bmod u(x) = c_{n-1}x^{n-1} + \dots + c_1x + c_0$,则有

$$x^{2^k} \bmod u(x) = (c_{n-1}^2x^{2n-2} + \dots + (c_1c_0 + c_1c_0)x + c_0^2) \bmod u(x)$$

其中 x^{2n-2}, \dots, x^n 可用辅助表中的多项式代替。计算 $x^p \bmod u(x)$ 的全部时间需要 $O(n^2(\log p)^3)$ 个单位,这样我们就得到了 Q 的第二行。为了得到 Q 的更多行,只须以类似于平方模 $u(x)$ 的方式重复地乘以 $x^p \bmod u(x)$,我们就可计算 $x^{2^p} \bmod u(x), x^{3^p} \bmod u(x), \dots$; 步骤 B2 在另外的 $O(n^3(\log p)^2)$ 个时间单位内完成。于是步骤 B1, B2, B3 总共花费 $O(n^2(\log p)^3 + n^3(\log p)^2)$ 个时间单位;这三个步骤能告诉我们 $u(x)$ 的因子个数。

但当 p 很大且我们达到步骤 B4 时,要求我们对 p 个不同的 s 值计算最大公因子,甚至当 p 仅仅是中等规模时这也办不到。这一障碍首先被 Hans Zassenhaus 所排除[J. Number Theory 1 (1969), 291~311],他说明了如何确定所有“有用的” s 值(见习题 14); 而一个甚至更好的处理办法是由 Zassenhaus 和 Cantor 于 1980 年发现的。如果 $v(x)$ 是 (8) 的任意一个解,那么我们知道 $u(x)$ 整除 $v(x)^p - v(x) = v(x) \cdot (v(x)^{(p-1)/2} + 1) \cdot (v(x)^{(p-1)/2} - 1)$ 。这就提示我们计算

$$\gcd(u(x), v(x)^{(p-1)/2} - 1) \quad (20)$$

很幸运, (20) 将是 $u(x)$ 的一个非平凡的因子。事实上,通过考虑 (7), 可以精确地确定我们运气的大小。设对于 $1 \leq j \leq r, v(x) \equiv s_j \pmod{p_j(x)}$, 则 $p_j(x)$ 整除 $v(x)^{(p-1)/2} - 1$ 当且仅当 $s_j^{(p-1)/2} \equiv 1 \pmod{p}$ 。我们知道, 在 $0 \leq s < p$ 范围内恰有 $(p-1)/2$ 个整数 s 满足 $s^{(p-1)/2} \equiv 1 \pmod{p}$, 因此大约有一半的 $p_j(x)$ 将出现于 (20) 的 gcd 中。更确切地说, 如果 $v(x)$ 是 (8) 的一个随机的解, 其中所有的 p^r 解都是同样可能的, 则 (20) 的 gcd 等于 $u(x)$ 的概率恰是 $((p-1)/2p)^r$, 而它等于 1 的概率是 $((p+1)/2p)^r$ 。因此对于所有的 $r \geq 2$ 和 $p \geq 3$, 得到一个非平凡的因子的概率是

$$1 - \left(\frac{p-1}{2p}\right)^r - \left(\frac{p+1}{2p}\right)^r = 1 - \frac{1}{2^{r-1}} \left(1 + \binom{r}{2} p^{-2} + \binom{r}{4} p^{-4} + \dots\right) \geq \frac{4}{9}$$

因此, 除非 p 十分小, 用下边的过程来代替步骤 B4 是一个好主意: 置 $v(x) \leftarrow a_1 v^{[1]}(x) + a_2 v^{[2]}(x) + \dots + a_r v^{[r]}(x)$, 其中诸系数 a_j 是在 $0 \leq a_j < p$ 的范围内随机选择的。设 $u(x)$ 的当前的部分因子分解是 $u_1(x) \cdots u_t(x)$, 其中 t 开始时为 1, 对所有使得 $\deg(u_i) > 1$ 的 i 计算

$$g_t(x) = \gcd(u_t(x), v(x)^{(p-1)/2 - 1})$$

每当找到一个非平凡的 \gcd 时,就用 $g_t(x)(u_t(x)/g_t(x))$ 代替 $u_t(x)$ 且增加 t 的值。对于不同的 $v(x)$ 重复这一过程直到 $t=r$ 为止。

如果假定(因为我们可以这样做)只需要(8)的 $O(\log r)$ 个随机解 $v(x)$,就可以给出为实施 B4 这一替换步骤所要求的时间之上界,它花费 $O(m(\log p)^2)$ 步计算 $v(x)$;而且如果 $\deg(u_i) = d$,则它花费 $O(d^2(\log p)^3)$ 步计算 $v(x)^{(p-1)/2} \bmod u_i(x)$ 及另外的 $O(d^2(\log p)^2)$ 步计算 $\gcd(u_i(x), v(x)^{(p-1)/2 - 1})$ 。因此总共的时间是 $O(n^2(\log p)^3 \log r)$ 。

不同次数的因子分解 我们现在将回过头来寻求模 p 下的因子的一个稍许更简单的方法。在本小节我们已经研究过的一些想法涉及许多对子计算代数的有教益的见解,所以作者不惜笔墨对它们作了介绍;但是,我们也发现,在模 p 之下进行因子分解的问题实际上无须依赖于这么多概念就能解决。

首先我们可以利用这样一个事实,即一个 d 次不可约多项式 $q(x)$ 是 $x^{p^d} - x$ 的一个因子,而且对于 $1 \leq c < d$ 它不是 $x^{p^c} - x$ 的一个因式,见习题 16。因此通过采取下列策略,我们可以分别剔除各次的不可约的因子。

D1. 如同在 Berlekamp 方法中一样,排除平方因子。也置 $v(x) \leftarrow u(x)$, $w(x) \leftarrow "x"$, 以及 $d \leftarrow 0$ 。(这里 $v(x)$ 和 $w(x)$ 是以多项式作为值的变量。)

D2. (此时 $w(x) = x^{p^d} \bmod v(x)$; $v(x)$ 的所有不可约因子都不同并有大于 d 的次数。)如果 $d+1 > \frac{1}{2} \deg(v)$, 这一过程即终止,因为我们或者有 $v(x) = 1$, 或者 $v(x)$ 不可约。否则 d 增加 1 且用 $w(x)^p \bmod v(x)$ 代替 $w(x)$ 。

D3. 求 $g_d(x) = \gcd(w(x) - x, v(x))$ 。(这是其次数为 d 的 $u(x)$ 的所有不可约因子的乘积。)如果 $g_d(x) \neq 1$, 则用 $v(x)/g_d(x)$ 代替 $v(x)$ 以及用 $w(x) \bmod v(x)$ 代替 $w(x)$; 而如果 $g_d(x)$ 的次数大于 d , 则用以下的算法来求它的因子。返回步骤 D2。 ■

这一过程确定每个次数 d 的诸不可约因子的乘积,因此它告知我们各个次数有多少个因子。由于在我们举例的多项式(19)中三个因子有不同的次数,因此无须分解多项式 $g_d(x)$ 就可全部发现它们。

为了完成这一方法,当 $\deg(g_d) > d$ 时,我们需要有一种方法来把多项式 $g_d(x)$ 分解成为它的不可约因子,Michael Rabin 于 1976 年指出,这可以通过在有 p^d 个元素的域中进行算术运算来实现。David G. Cantor 和 Hans Zassenhaus 于 1979 年发现,以下列恒等式为基础,甚至有更为简单的处理方法:如果 p 是任何奇素数,则对于所有多项式 $t(x)$,我们有

$$g_d(x) = \gcd(g_d(x), t(x)) \cdot \gcd(g_d(x), t(x)^{(p^d-1)/2} + 1) \cdot \gcd(g_d(x), t(x)^{(p^d-1)/2} - 1) \quad (21)$$

因为 $t(x)^{p^d} - t(x)$ 是 d 次的所有不可约多项式的倍数。(我们可以认为 $t(x)$ 是大小为 p^d 的域的一个元素, 其中该域由如习题 16 中那样的模一个不可约多项式 $f(x)$ 的所有多项式组成。)现在习题 29 表明, 当 $t(x)$ 是小于等于 $2d-1$ 次的一个随机多项式时, $\gcd(g_d(x), t(x)^{(p^d-1)/2} - 1)$ 有大约 50% 的机会将是 $g_d(x)$ 的一个非平凡因子; 因此发现所有这些因式将不需要许多随机的试验。不失一般性, 我们可以假定 $t(x)$ 是首一的, 因为 $t(x)$ 的整数倍没有造成区别, 顶多把 $t(x)^{(p^d-1)/2}$ 改变成它的负值。于是在 $d=1$ 的情况下, 我们可以取 $t(x) = x + s$, 其中 s 是随机选定的。

当只使用线性多项式 $t(x)$ 时, 对于 $d > 1$, 有时这一过程可能成功。例如, 在模 3 下有 8 个三次不可约多项式 $f(x)$, 而且通过对于 $0 \leq s < 3$ 计算 $\gcd(f(x), (x+s)^3 - 1)$, 它们都将被区分开:

$f(x)$	$s=0$	$s=1$	$s=2$
$x^3 + 2x + 1$	1	1	1
$x^3 + 2x + 2$	$f(x)$	$f(x)$	$f(x)$
$x^3 + x^2 + 2$	$f(x)$	$f(x)$	1
$x^3 + x^2 + x + 2$	$f(x)$	1	$f(x)$
$x^3 + x^2 + 2x + 1$	1	$f(x)$	$f(x)$
$x^3 + 2x^2 + 1$	1	$f(x)$	1
$x^3 + 2x^2 + x + 1$	1	1	$f(x)$
$x^3 + 2x^2 + 2x + 2$	$f(x)$	1	1

习题 31 包含了为什么线性多项式可能是有效的部分说明; 然而, 当次数为 d 的不可约多项式的个数超过 2^d 时, 显然将存在不能由 $t(x)$ 的线性选择来进行区分的不可约多项式。

习题 30 讨论了当 $p=2$ 时代替式 (21) 的一个有效方法。E. Kaltofen 和 V. Shoup 发现了当 p 很大时对于不同次数的因子分解的一个更快的方法; 对于实用大小的数运行时间是 $O(n^{2.5} + n^{1+\epsilon} \log p)$ 个模 p 下的算术运算, 而且当 ω 是习题 4.6.4-66 中的“快速”矩阵乘法的指数和当 $n \rightarrow \infty$ 时, 是 $O(n^{(5+\omega+\epsilon)/4} \log p)$ 个这样的运算。[请见 *J. Symbolic Comp.* **20** (1995), 363~397; *Math. Comp.* **67** (1998), 1179~1197。]

历史注记:通过首先计算 $g(x) = \gcd(x^{p-1} - 1, f(x))$ 而后对于任意的 s , 计算 $\gcd(g(x), (x+s)^{(p-1)/2} \pm 1)$ 来求一个无平方多项式 $f(x)$ 模 p 的所有线性因子的思想, 是由 A. M. Legendre 给出的, *Mémoires Acad. Sci. Paris* (1785), 484~490; 他的动机是求形如 $f(x) = py$ 的 Diophantine 方程的所有整数解, 即 $f(x) \equiv 0 \pmod{p}$ 。体现在算法 D 的更一般的次数分开技术是在公元 1800 年之前由高斯发现的, 但未发表[参见他的 *Werke* 2 (1876), 237], 而后由 Évariste Galois (伽罗瓦) 在创立有限域理论的现在成为经典的论文中发现 [*Bulletin des Sciences, Mathématiques, Physiques et Chimiques* 13 (1830), 428~435; 重印于 *J. de Math. Pures et Appliquées* 11 (1846), 398~407]。然而, 高斯和伽罗瓦的这一成果超越了当时的时代, 直到 J. A. Serret 稍后给出详细的解释 [*Mémoires Acad. Sci. Paris, Series 2*, 35 (1866), 617~688; 算法 D 在 § 7 中] 之前, 它还未被很好地理解。关于把 $g_d(x)$ 分成为不可约因子的一些专门的过程随后由许多作者设计出来, 但是在计算机的发明使它们成为合乎要求之前, 显然还没有发现出对于很大的 p 有效地工作的具有完全一般性的方法。带有严格地分析了的运行时间的头一个这样的随机算法是由 E. Berlekamp 发表的 [*Math. Comp.* 24 (1970), 713~735]; 它由 Robert T. Moenck [*Math. Comp.* 31 (1977), 235~250], M. O. Rabin [*SICOMP* 9 (1980), 273~280], D. G. Cantor 和 H. J. Zassenhaus [*Math. Comp.* 36 (1981), 587~592] 所改进和简化。Paul Camion 独立地发现了对于多变量多项式的一些特殊情况的推广 [*Comptes Rendus Acad. Sci. Paris* A291 (1980), 479~482. *IEEE Trans.* IT-29 (1983), 378~385]。

P. Flajolet, X. Gourdon 和 D. Panario 分析了在模 p 之下分解一个随机多项式所需要的平均运算次数, *Lecture Notes in Comp. Sci.* 1099 (1996), 232~243。

整数上的因子分解 当我们不是在模 p 之下进行工作时, 求整系数多项式的完全因子分解要稍微困难些, 但是仍有相当有效的方法可用于此目的。

Isaac Newton 在他的 *Arithmetica Universalis* (1707) 一书中给出了求整系数多项式的线性和二次因子的方法。这个方法在 1793 年被一个名叫 Friedrich von Schubert 的天文学家所推广, 他说明怎样在有限步之内求出次数为 n 的所有因子; 见 M. Cantor, *Geschichte der Mathematik* 4 (Leipzig: Teubner, 1908), 136~137。大约 90 年后, L. Kronecker 重新独立地发现了 von Schubert 的方法。不幸的是当 n 是 5 或更大时, 这个方法效率非常之低。借助于上面介绍的“模 p ”因子分解方法, 可以得到好得多的结果。

假设我们要求整数上一个给定多项式

$$u(x) = u_n x^n + u_{n-1} x^{n-1} + \cdots + u_0, \quad u_n \neq 0$$

的不可约因子。作为头一步, 我们可以除以系数的最大公因子; 这使我们得到一个本原多项式。通过像在习题 34 中那样, 剔出 $\gcd(u(x), u'(x))$, 我们还可以假定 $u(x)$ 是无平方的。

现在, 如果 $u(x) = v(x)w(x)$, 其中每一个多项式都有整系数, 显然, 对于所有素数 p , 我们有 $u(x) \equiv v(x)w(x) \pmod{p}$, 所以除非 p 整除 $\ell(u)$, 否则有

一个非平凡的模 p 因子分解。因此可以试用分解 $u(x)$ 模 p 的一个有效方法来重新构造整数上的 $u(x)$ 的可能的因子分解。

例如设

$$u(x) = x^8 + x^6 - 3x^5 - 3x^3 + 8x^2 + 2x - 5 \quad (22)$$

我们在上面的(19)中已经看到

$$u(x) \equiv (x^4 + 2x^3 + 3x^2 + 4x + 6)(x^3 + 8x^2 + 4x + 12)(x + 3) \pmod{13} \quad (23)$$

而且 $u(x)$ 模 2 的完全因子分解显出一个 6 次的因子和另一个 2 次的因子(见习题 10)。从(23)我们可以看出, $u(x)$ 没有 6 次的因子, 所以它必然是在整数上不可约的。

这个具体的例子也许太简单了; 经验表明, 通过考察它们模一些素数的因子, 大多数不可约的多项式都可以用这种方式来加以识别, 但是确定不可约性并非总是这样容易。例如, 有一些多项式对于所有素数 p 来说, 它们在模 p 之下可以适当地分解, 而且诸因子具有一致的次数, 但在整数上它们却不可约(见习题 12)。

习题 38 中显示了一大类不可约的多项式, 而习题 27 中证明, 几乎所有的多项式在整数上不可约。但是我们通常不试图去分解一个随机的多项式; 大概总有某种原因来预期一个非平凡的因子, 否则我们就不会首先有因子分解的企图了。我们需要一个方法, 当因子存在时它能找到它们。

一般来说, 如果通过考虑在模不同的素数之下 $u(x)$ 的特性, 以求 $u(x)$ 的诸因子, 则这些结果将不易于结合在一起。例如如果 $u(x)$ 实际上是 4 个二次多项式的乘积, 则将很难把它在不同的素数模之下的映像凑合在一起。因此最好只用一个素数, 并且看看一旦觉得模这个素数将使诸因子有正确的次数时, 我们可以从它得到多少好处。

一个主意是在模一个很大的素数 p 之下进行工作, 它足够大, 使得在整数上的任何真正的因子分解 $u(x) = v(x)w(x)$ 中的系数实际上必然处于 $-p/2$ 和 $p/2$ 之间。于是所有可能的整数因子都可以从模 p 因子读出, 而模 p 因子的计算我们是知道的。

习题 20 说明如何看到关于多项式的系数的相当好的上界。例如如果式(22)是可约的, 则它将有小于等于 4 次的一个因式 $v(x)$, 而且由该题的结果可知, v 的系数的绝对值顶多为 34。所以如果我们在模任何大于 68 的素数 p 之下工作, 则 $u(x)$ 的所有可能的因子将是相当明显的。其实, 在模 71 之下的完全因子分解是

$$(x + 12)(x + 25)(x^2 - 13x - 7)(x^4 - 24x^3 - 16x^2 + 31x - 12)$$

而且我们立即看到, 在整数上这些多项式中没有一个式(22)的因子, 因为它们的常数项不能整除 5; 而且也不能把两个因子组合在一起而得到(22)的一个因子, 因为可想见的常数项 $12 \times 25, 12 \times (-7), 12 \times (-12)$ 当中没有一个模 71 同余于 ± 1 或 ± 5 。

顺便说一句, 得到关于多项式因子的系数的好上界并不是轻而易举的, 因为当多项式相乘时会有大量的项互消。例如, 看上去极为普通的多项式 $x^n - 1$ 有不可约

的因子,对无穷多的 n ,这些因子的系数超过 $\exp(n^{1/\lg n})$ 。[见 R. C. Vaughan, *Michigan Math. J.* **21**(1974), 289~295.] 习题 32 中讨论了 $x^n - 1$ 的因子分解。

如果 $u(x)$ 有很大的次数或很大的系数,则素数 p 可能需要极其大,假定 $u(x)$ 在模 p 之下是无平方的,代替使用一个很大的素数 p ,我们也可利用小的 p 。因为在这种情况下,称为 Hensel 引理的一个重要的构造可以以独特的方式用来把模 p 的一个因子分解扩充成为对于任意高的指数 e 在模 p^e 下的一个因子分解(见习题 22)。如果对于 $p=13$ 和 $e=2$,我们把 Hensel 引理应用于(23),就得到惟一的因子分解(模 169):

$$u(x) \equiv (x-36)(x^3-18x^2+82x-66)(x^4+54x^3-10x^2+69x+84)$$

把这些因子叫做 $v_1(x)v_3(x)v_4(x)$,我们看到 $v_1(x)$ 和 $v_3(x)$ 在整数上都不是 $u(x)$ 的因子,当系数已由模 169 归结到 $\left(-\frac{169}{2}, \frac{169}{2}\right)$ 的范围内时,它们的乘积 $v_1(x)v_3(x)$ 也不是 $u(x)$ 的因子。因此我们已经穷尽所有的可能性,并且再次证明 $u(x)$ 在整数上是不可约的——这次只使用它在模 13 下的因子分解。

我们已经考虑的例子在一个重要的方面是不典型的:我们一直在对(22)中的首一多项式 $u(x)$ 进行因子分解,所以我们可以假定,它的所有因子都是首一的。但如果 $u_n > 1$,我们应当做什么呢?在这样一种情况下,除一个之外,所有多项式因子的前导系数都可以在几乎任意的模 p^e 之下变化;我们肯定不希望试验所有可能性。也许读者已经注意到这个问题了。幸而有一个简单的方法:因子分解 $u(x) = v(x) \cdot w(x)$ 意味着因子分解 $u_n u(x) = v_1(x) w_1(x)$,其中 $\ell(v_1) = \ell(w_1) = u_n = \ell(u)$ 。(对不起,在我对你的多项式进行因子分解之前,你是否介意我用它的前导系数来乘多项式?)我们实际上可以如上边那样进行,但使用 $p^e > 2B$,其中 B 现在是 $u_n u(x)$ 的而不是 $u(x)$ 的因子的极大系数的上界。

把所有这些发现合在一起就得到以下的过程:

F1. 求惟一的无平方因子分解

$$u(x) \equiv \ell(u) v_1(x) \cdots v_r(x) \pmod{p^e}$$

其中 p^e 像上边说明的那样充分大,且 $v_i(x)$ 是首一的。(对于大多数素数这是可能的,见习题 23。)并且置 $d \leftarrow 1$ 。

F2. 对于因子 $v(x) = v_{i_1}(x) \cdots v_{i_d}(x)$,如果 $d = \frac{1}{2}r$ 则 $i_1 = 1$ 的每一个组合,构造惟一的多项式 $\bar{v}(x) \equiv \ell(u) v(x) \pmod{p^e}$,它的系数全都在区间 $\left[-\frac{1}{2}p^e, \frac{1}{2}p^e\right)$ 内。如果 $\bar{v}(x)$ 整除 $\ell(u)u(x)$,则输出因子 $\text{pp}(\bar{v}(x))$,用这个因子除 $u(x)$,并从模 p^e 的因子表撤销对应的 $v_i(x)$;从 r 中减去撤销的因子个数,如果 $d > \frac{1}{2}r$ 则终止此算法。

F3. d 加 1,如果 $d \leq \frac{1}{2}r$ 则返回 F2。 **I**

在此过程的结尾, $u(x)$ 的当前值将是原来给出的多项式的最后不可约因子。注意, 如果 $|u_0| < |u_n|$, 则用反序多项式 $u_0x^n + \cdots + u_n$ (其因子是 $u(x)$ 的因子的反序) 来做所有工作是更可取的。

所述的这一过程要求 $p^r > 2B$, 其中 B 是 $u_q u(x)$ 的任何因子的系数的界, 但如果我们仅保证它对次数 $\leq \frac{1}{2} \deg(u)$ 的因子正确, 则可以使用小得多的 B 值。在这种情况下, 每当 $\deg(v) > \frac{1}{2} \deg(u)$ 时, 步骤 F2 中的可除性检验应当应用于 $w(x) = v_1(x) \cdots v_r(x)/v(x)$ 而不是 $v(x)$ 。

如果我们决定仅仅保证 B 应该限定 $u(x)$ 的至少一个真因子的系数, 则我们仍可以更进一步减少 B 。(例如, 当我们分解一个非素数的 N 而不是一个多项式时, 某些因子可能是非常大的, 但至少有一个将小于等于 \sqrt{N} 。) 这个思想是由 B. Beauzamy, V. Trevisan 和 P. S. Wang 提出的 [J. Symbolic Comp. 15 (1993), 393 ~ 413], 习题 21 对它作了讨论。在步骤 F2 中的可除性检验因此必须既应用于 $v(x)$ 也应用于 $w(x)$, 但是由于通常 p^r 是小得多的, 因此计算会更快些。

上边的算法包含一个明显的瓶颈: 我们可能要检验 $2^r - 1$ 个那么多的可能的因子 $v(x)$ 。在随机的情况下 2^r 的平均值大约是 n , 或者也许是 $n^{1.5}$ (见习题 5), 但在非随机情况下我们将需要尽可能加速这部分程序。一个快速抛弃虚假因子的方法是首先计算尾部的系数 $\bar{v}(0)$, 仅当其整除 $\ell(u)u(0)$ 时才继续进行; 除非此整除性条件被满足, 否则前一段说明的复杂性可不必考虑, 因为甚至当 $\deg(v) > \frac{1}{2} \deg(u)$ 时, 这样一个检验也是正确的。

加速这个过程的另一个重要的方法是把 r 减小, 使得它趋于反映因子的真实个数。对于各种小的素数 p_j , 上边的不同次数因子分解算法可以应用, 这样就对每个素数得到了在模 p_j 下诸因子的可能次数之集合 D_j ; 见习题 26。我们可以把 D_j 表示成一个 n 个二进位的串。现在我们计算交 $\cap D_j$, 即这些二进位串的按位“与”, 而且仅对

$$\deg(i_1) + \cdots + \deg(i_d) \in \cap D_j$$

才实施步骤 F2。其次把 p 选择成有最小 r 值的 p_j 。这技术是 David R. Musser 给出的, 他的经验提示尝试大约 5 个素数 p_j [见 JACM 25 (1978), 271 ~ 282]。当然如果当前的 $\cap D_j$ 表明 $u(x)$ 不可约, 则我们应当立即停止。

在 JACM 22 (1975), 291 ~ 308 中, Musser 已经给出类似于上述步骤的一个因子分解方法的完整的讨论。步骤 F1 ~ F3 结合了由 G. E. Collins 于 1978 年提出的改进, 即在寻找试验的因子时一次取 d 个因子的组合, 而不是全部次数 d 的组合。这一改进是重要的, 因为在有理数上是不可约的多项式的模 p 的因子具有统计特性 (参考习题 37)。

A. K. Lenstra, H. W. Lenstra, Jr. 和 L. Lovász 介绍了他们著名的“LLL 算法”以

便得到关于为分解整数上的一个多项式所需计算量的严格最坏情况的上限[Math. Annalen 261 (1982), 515~534]。他们的方法不要求随机数,而且对于次数为 n 的 $u(x)$ 它的运行时间是 $O(n^{12} + n^9(\log \|u\|)^3)$ 二进位运算,其中 $\|u\|$ 在习题 20 中定义。这个估计包括查找一个适当的素数 p 的时间以及用算法 B 求所有在模 p 之下的因子的时间。当然,在实用中使用随机化的启发式方法运行得显著地更快些。

最大公因子 类似的技术可用来计算多项式的最大公因子:如果在整数上 $\gcd(u(x), v(x)) = d(x)$, 而且 $\gcd(u(x), v(x)) = q(x) \pmod{p}$, 其中 $q(x)$ 是首一的, 则 $d(x)$ 是 $u(x)$ 和 $v(x)$ 模 p 的一个公因子, 因此

$$d(x) \text{ 整除 } q(x) \pmod{p} \quad (24)$$

如果 p 不整除 u 和 v 两者的前导系数, 则它就不整除 d 的前导系数; 在这种情况下, $\deg(d) \leq \deg(q)$ 。当对这样一个素数 p , $q(x) = 1$ 时, 我们必定因此有 $\deg(d) = 0$, 及 $d(x) = \gcd(\text{cont}(u), \text{cont}(v))$ 。这就论证了 4.6.1 小节中所作注释, 即 4.6.1-(6) 中 $\gcd(u(x), v(x))$ 模 13 的简单计算足以证明 $u(x)$ 和 $v(x)$ 在整数上是互素的; 算法 4.6.1E 或算法 4.6.1C 比较费力的计算是不必要的。由于两个随机的本原多项式在整数上几乎总是互素的, 而且由习题 4.6.1-5, 因为它们以 $1 - 1/p$ 的概率在模 p 下互素, 因此, 通常来做模 p 的计算是一个好的想法。

如以前所解释的那样, 对于实践中出现的非随机多项式我们也需要有好的方法。因此希望改进我们的技术并且完全以在模素数 p 之下工作所得到的信息为基础, 发现在整数上一般地如何求 $\gcd(u(x), v(x))$ 。我们可以假定 $u(x)$ 和 $v(x)$ 是本原的。

代替直接地计算 $\gcd(u(x), v(x))$, 去寻找多项式

$$\bar{d}(x) = c \cdot \gcd(u(x), v(x)) \quad (25)$$

是更方便的, 其中常数 c 被选择成使

$$\ell(\bar{d}) = \gcd(\ell(u), \ell(v)) \quad (26)$$

对于适当的 c , 这个条件将总是成立, 因为 $u(x)$ 和 $v(x)$ 的任何公因子的前导系数必定是 $\gcd(\ell(u), \ell(v))$ 的一个因子。一旦已求得满足这些条件的 $\bar{d}(x)$, 我们就可以容易地来计算 $\text{pp}(\bar{d}(x))$, 它是 $u(x)$ 和 $v(x)$ 的真正的最大公因子。条件(26)是方便的, 因为它避免了 \gcd 的单位倍数的不确定性; 我们实际上已经使用同样的思想来控制在分解因子的程序中的前导系数。

如果 p 是一个足够大的素数, 基于应用于 $\ell(\bar{d})u(x)$ 或 $\ell(\bar{d})v(x)$ 的习题 20 中系数的上限, 我们计算所有系数都在 $[-\frac{1}{2}p, \frac{1}{2}p)$ 中的惟一多项式 $\bar{q}(x) \equiv \ell(\bar{d})q(x) \pmod{p}$ 。当 $\text{pp}(\bar{q}(x))$ 整除 $u(x)$ 与 $v(x)$ 两者时, 由于(24)它必然等子 $\gcd(u(x), v(x))$ 。另一方面, 如果它不整除 $u(x)$ 与 $v(x)$ 两者, 必然有 $\deg(q) > \deg(d)$ 。对算法 4.6.1E 的研究揭示, 仅当 p 整除以精确的整数算术通过此算法计算的非零余式之一的前导系数时, 才有这种情况; 否则模 p 欧几里得算法和算法

4.6.1E 一样精确地处理同样的多项式序列,顶多只差一个非零常数倍数(模 p)。所以只有少量“倒霉”的素数可以引起我们丢失 gcd,但如果我们继续试验就将立即找到一个幸运的素数。

如果系数的上界是如此之大,以致单精度的素数 p 不够用,那么我们可以计算 $\bar{d}(x)$ 模若干个素数 p ,直到它被 4.3.2 小节的中国剩余算法确定为止。这个方法已由 W. S. Brown 在 JACM 18 (1971), 478~504 中详细作了介绍,它是由 Brown 与 G. E. Collins 给出的。或者如同 J. Moses 与 D. Y. Y. Yun [Proc. ACM Conf. 28 (1973), 159~166] 所建议的,我们可以使用 Hensel 方法确定对于充分大 e 的模 p 的 $\bar{d}(x)$ 。Hensel 的构造在计算上显得比中国剩余方法要优越;但是仅当

$$d(x) \perp u(x)/d(x) \quad \text{或} \quad d(x) \perp v(x)/d(x) \quad (27)$$

时它才直接地生效,因为这个想法是把习题 22 的技术应用于因子分解 $\ell(\bar{d})u(x) \equiv \bar{q}(x)u_1(x)$ 或 $\ell(\bar{d})v(x) \equiv \bar{q}(x)v_1(x) \pmod{p}$ 两者之一。习题 34 和 35 表明有可能把事情安排成每当需要时(27)便成立。(类似于对互素整数所用的符号, (2)中的

$$u(x) \perp v(x) \quad (28)$$

表示 $u(x)$ 和 $v(x)$ 互素。)

这里所概述的 gcd 算法要比 4.6.1 节中介绍的那些算法快得多,仅当多项式剩余序列非常短时例外。也许最好的一般过程是从模一个相当小的素数 p 下计算 $\text{gcd}(u(x), v(x))$ 开始,而不是从 $\ell(u)$ 和 $\ell(v)$ 的一个因子开始。如果结果 $q(x)$ 为 1,任务就完成了。如果它有高的次数,我们就用算法 4.6.1C;否则使用上述方法之一,首先以 $u(x)$ 和 $v(x)$ 的系数以及以 $q(x)$ (小)的次数为基础计算 $\bar{d}(x)$ 的系数的上界。像在因子分解问题中那样,如果尾部系数比前导系数更简单,我们应当应用这一过程到 $u(x), v(x)$ 的反序上而后再反序这个结果。

多变量多项式 类似的技术导致了对于整系数多变量多项式的因子分解或 gcd 计算的有用算法。通过模不可约多项式 $x_2 - a_2, \dots, x_t - a_t$ 处理多项式 $u(x_1, \dots, x_t)$ 是方便的,这些多项式起着上边讨论中的 p 的作用。由于 $v(x) \bmod (x - a) = v(a)$, 因此

$$u(x_1, \dots, x_t) \bmod \{x_2 - a_2, \dots, x_t - a_t\}$$

的值是单变量多项式 $u(x_1, a_2, \dots, a_t)$ 。当选定了整数 a_2, \dots, a_t 使得 $u(x_1, a_2, \dots, a_t)$ 在 x_1 上有和原来的多项式 $u(x_1, x_2, \dots, x_t)$ 相同的次数时, Hensel 构造的一个适当推广将把这个单变量多项式的无平方因子分解“升高”成模 $|(x_2 - a_2)^{n_2}, \dots, (x_t - a_t)^{n_t}|$ 下的因子分解,其中 n_j 是 u 中 x_j 的次数;同时,我们也可以在模一个适当的素数之下进行工作。应当有尽可能多的 a_j 为 0, 以便保持中间结果的稀疏性。关于细节,除了早先引用的 Musser 和 Moses 与 Yun 的文章外,还请见 P. S. Wang, Math. Comp. 32(1978), 1215~1231。

从上面引证的前驱性的论文发表以来,已经积累了相当多的计算经验。关于更

新情况的综述,请见 R. E. Zippel, *Effective Polynomial Computation* (Boston: Kluwer, 1993)。而且,现在有可能分解通过一个“黑箱子”计算程序隐式给出的多项式,即便若把输入和输出的多项式明确写出,它们将填满整个宇宙[见 E. Kaltofen 和 B. M. Trager, *J. Symbolic Comp.* **9** (1990), 301 ~ 320; Y. N. Lakshman 和 B. David Saunders, *SICOMP* **24** (1995), 387 ~ 397]。

*The asymptotically best algorithms frequently turn out
to be worst on all problems for which they are used.*

近似地最好的算法经常对于使用它们的所有问题而言是最坏的。

——D. G. CANTOR and H. ZASSENHAUS (1981)

习 题

►1. [M24] 设 p 是素数,且设 $u(x)$ 是一个 n 次的随机多项式,并假定 p^n 个首一多项式的每一个都是同等可能的。试证明,如果 $n \geq 2$, $u(x)$ 有一个在模 p 之下的线性因子的概率介于 $(1 + p^{-1})/2$ 和 $(2 + p^{-2})/3$ 之间,含这两个值在内。当 $n \geq p$ 时,给出这一概率的一个闭区间形式。线性因子的平均个数是多少?

►2. [M25] (a) 证明在一个唯一因子分解整环上,任何首一多项式 $u(x)$ 均可以唯一地表示成形式

$$u(x) = v(x)^2 w(x)$$

其中 $w(x)$ 是无平方的(没有形如 $d(x)^2$ 的正次数的因子),且 $v(x)$ 和 $w(x)$ 都是首一的。(b) (E. R. Berlekamp) 当 p 为素数时,有多少 n 次的首一多项式是模 p 无平方的?

3. [M25] (对于多项式的中国剩余定理) 设 $u_1(x), \dots, u_r(x)$ 是域 S 上的多项式,且对于所有 $j \neq k$, $u_j(x) \nmid u_k(x)$ 。对于 S 上任何给定的多项式 $w_1(x), \dots, w_r(x)$, 证明 S 上有唯一的多项式 $v(x)$, 使得 $\deg(v) < \deg(u_1) + \dots + \deg(u_r)$, 且对于 $1 \leq j \leq r$, $v(x) \equiv w_j(x) \pmod{u_j(x)}$ 。当 S 是所有整数的集合时,这个结果是否还成立?

4. [HM28] 设 a_{np} 是模素数 p 的,次数为 n 的首一不可约多项式的个数。求出生成函数 $G_p(z) = \sum_n a_{np} z^n$ 的一个公式。[提示:证明下列幂级数恒等式: $f(z) = \sum_{j \geq 1} g(z^j)/j^j$ 当且仅当 $g(z) = \sum_{n \geq 1} \mu(n) f(z^n)/n^j$ 。] $\lim_{p \rightarrow \infty} a_{np}/p^n$ 是多少?

5. [HM30] 设 A_{np} 是模素数 p 下,随机地选择的 n 次多项式的因子的平均个数。证明 $\lim_{p \rightarrow \infty} A_{np} = H_n$ 。当 r 是不可约因子的个数时, 2^r 的平均值的极限是多少?

6. [M21] (J. L. Lagrange, 1771) 证明同余式(9)。[提示:在 p 个元素的域中分解 $x^p - x$]

7. [M22] 证明等式(14)。

8. [HM20] 怎样才能确保由算法 N 输出的向量是线性无关的?

9. [20] 给定 101 的一个原根 2,说明如何用一种简单的方式来构造模 101 的倒数表。

►10. [21] 利用 Berlekamp 过程,求在模 2 下(22)中多项式 $u(x)$ 的完全因子分解。

11 [22] 求在模 5 下,(21)中多项式 $u(x)$ 的完全因子分解。

►12. [M22] 对所有素数 p ,利用 Berlekamp 算法来确定 $u(x) = x^4 + 1$ 模 p 的因子个数。[提示:分别考虑 $p=2$, $p=8k+1$, $p=8k+3$, $p=8k+5$, $p=8k+7$ 的情况;什么是矩阵 Q ? 你不必发现因子;仅须确定有多少因子。]

13. [M25] 继续上一题,借助于量 $\sqrt{-1}, \sqrt{2}, \sqrt{-2}$ ——当这样的平方根在模 p 之下存在时,

对所有奇素数 p , 试给出在模 p 之下 $x^4 + 1$ 的因子的一个明确的公式。

14. [M25] (II. Zassenhaus) 设 $v(x)$ 是 (8) 的一个解, 并设 $w(x) = \prod (x - s)$, 其中乘积是对于使得 $\gcd(u(x), v(x) - s) \neq 1$ 的所有 $0 \leq s < p$ 取的。说明给定 $u(x)$ 和 $v(x)$ 后, 如何计算 $w(x)$ 。[提示: 等式 (14) 意味着 $w(x)$ 是使得 $u(x)$ 整除 $w(v(x))$ 的最低多项式。]

► 15. [M27] (模素数平方根) 试设计一个算法, 计算模一个给定素数 p 之下的一个给定整数 u 的“平方根”, 即求一个整数 v , 只要它存在, 就有 $v^2 \equiv u \pmod{p}$ 。你的算法应当对于非常大的素数 p 也能保证效率。(对于 $p \neq 2$, 这个问题的一个解导致了以通常方式利用二次形式解任何给定的模 p 二次方程的一个过程。) 提示: 考虑当把本小节的因子分解方法应用于多项式 $x^2 - u$ 时将发生什么情况。

16. [M30] (有限域) 本题的目的是证明 E. Galois 在 1830 年引进的域的基本性质。

a) 给定 $f(x)$ 是模素数 p 下的不可约 n 次多项式, 证明次数小于 n 的 p^n 个多项式在算术模 $f(x)$ 和 p 下形成一个域。[注: 习题 4 中证明了每一次数的不可约多项式的存在性; 因此对于所有素数 p 和所有 $n \geq 1$, 具有 p^n 个元素的域存在。]

b) 证明具有 p^n 个元素的任何域都有一个原根 ξ , 使得这个域的元素是 $\{0, 1, \xi, \xi^2, \dots, \xi^{p^n-2}\}$ 。[提示: 习题 3.2.1.2-16 提供了在 $n=1$ 的特殊情况下的证明。]

c) 如果 $f(x)$ 是模 p 的 n 次不可约多项式, 证明当且仅当 m 是 n 的倍数时, $x^{p^m} - x$ 可被 $f(x)$ 整除 (由此得出, 我们可以稍快地检验不可约性: 一个给定的 n 次多项式 $f(x)$ 是模 p 不可约的, 当且仅当 $x^{p^n} - x$ 可被 $f(x)$ 整除而且对于所有整除 n 的素数 q , $x^{p^{n/q}} - x \nmid f(x)$ 。

17. [M23] 设 F 是有 13^2 个元素的一个域, 对于每个满足 $1 \leq f < 13^2$ 的整数 f , F 中阶数为 f 的元素有多少? (一个元素 a 的阶数就是使得 $a^m = 1$ 的最小正整数 m 。)

► 18. [M25] 设 $u(x) = u_n x^n + \dots + u_0$, $u_n \neq 0$ 是具有整系数的本原多项式, 并设 $v(x)$ 是由

$$v(x) = u_n^{-1} \cdot u(x/u_n) = x^n + u_{n-1} x^{n-1} + u_{n-2} u_n x^{n-2} + \dots + u_0 u_n^{n-1}$$

定义的首一多项式。(a) 若 $v(x)$ 在整数上有完全的因子分解 $p_1(x) \cdots p_r(x)$, 其中每个 $p_j(x)$ 是首一的, 问在整数上 $u(x)$ 的完全因子分解是什么? (b) 如果 $w(x) = x^m + w_{m-1} x^{m-1} + \dots + w_0$ 是 $v(x)$ 的一个因子, 证明对于 $0 \leq k < m$, w_k 是 u_n^{m-1-k} 的倍数。

19. [M20] (Eisenstein 准则) 也许最有名的整数上不可约多项式类是由 T. Schönemann 在 Crelle 32 (1846), 100 介绍而后由 G. Eisenstein 在 Crelle 38 (1850), 166~169 上推广的: 设 p 是素数且设 $u(x) = u_n x^n + \dots + u_0$ 有下列性质: (i) u_n 不被 p 整除; (ii) u_{n-1}, \dots, u_0 被 p 整除; (iii) u_0 不被 p^2 整除。证明 $u(x)$ 在整数上不可约。

20. [HM33] 如果 $u(x) = u_n x^n + \dots + u_0$ 是复数上的任一多项式, 设 $\|u\| = (|u_n|^2 + \dots + |u_0|^2)^{1/2}$ 。

a) 设 $u(x) = (x - \alpha)w(x)$ 和 $\bar{u}(x) = (\bar{\alpha}x - 1)w(x)$, 其中 α 是任何复数且 $\bar{\alpha}$ 是它的复共轭。证明 $\|u\| = \|v\|$ 。

b) 设在复数上 $u(x)$ 的完全因子分解是 $u_n(x - \alpha_1) \cdots (x - \alpha_n)$, 令

$$M(u) = |u_n| \prod_{j=1}^n \max(1, |\alpha_j|)$$

证明 $M(u) \leq \|u\|$ 。

c) 证明对于 $0 \leq j \leq n$, $|u_j| \leq \binom{n-1}{j} M(u) + \binom{n-1}{j-1} |u_{j-1}|$ 。

d) 把这些结果合在一起以证明如果 $u(x) = v(x)w(x)$ 及 $v(x) = v_n x^n + \dots + v_0$, 其中 u ,

v, w 都有整系数, 则 v 的系数以

$$|v_j| \leq \binom{m-1}{j} \|u\| + \binom{m-1}{j-1} |u_n|$$

为界。

21. [HM32] 继续习题 20, 我们将推导在整数上的多变量多项式因子的系数的有用的上界。为方便起见, 我们将用黑体字母表示 t 个整数的序列。因此, 代替

$$u(x_1, \dots, x_t) = \sum_{j_1, \dots, j_t} u_{j_1, \dots, j_t} x_1^{j_1} \cdots x_t^{j_t}$$

我们将简单地写 $u(\mathbf{x}) = \sum_{\mathbf{j}} u_{\mathbf{j}} x^{\mathbf{j}}$ 。注意对于 $\mathbf{x}^{\mathbf{j}}$ 的约定; 我们也写 $\mathbf{j}! = j_1! \cdots j_t!$ 和 $\sum \mathbf{j} = j_1 + \cdots + j_t$ 。

a) 证明恒等式

$$\sum_{\mathbf{j}, \mathbf{k}} \frac{1}{\mathbf{j}! \mathbf{k}!} \sum_{\mathbf{p}, \mathbf{q}, \mathbf{s} \geq 0} [\mathbf{p} - \mathbf{j} = \mathbf{q} - \mathbf{k}] a_{\mathbf{p}} b_{\mathbf{q}} \frac{\mathbf{p}! \mathbf{q}!}{(\mathbf{p} - \mathbf{j})!} \sum_{\mathbf{r}, \mathbf{s} \geq 0} [\mathbf{r} - \mathbf{j} = \mathbf{s} - \mathbf{k}] c_{\mathbf{r}} d_{\mathbf{s}} \frac{\mathbf{r}! \mathbf{s}!}{(\mathbf{r} - \mathbf{j})!} =$$

$$\sum_{\mathbf{i} \geq 0} \mathbf{i}! \sum_{\mathbf{p}, \mathbf{s} \geq 0} [\mathbf{p} + \mathbf{s} = \mathbf{i}] a_{\mathbf{p}} d_{\mathbf{s}} \sum_{\mathbf{q}, \mathbf{r} \geq 0} [\mathbf{q} + \mathbf{r} = \mathbf{i}] b_{\mathbf{q}} c_{\mathbf{r}}$$

b) 多项式 $u(\mathbf{x}) = \sum_{\mathbf{j}} u_{\mathbf{j}} x^{\mathbf{j}}$ 称为 n 次齐次的, 如果每项有 n 的总次数; 于是每当 $u_{\mathbf{j}} \neq 0$ 时 $\sum \mathbf{j} = n$ 。考虑系数的加权和 $B(u) = \sum_{\mathbf{j}} \mathbf{j}! |u_{\mathbf{j}}|^2$ 。使用 a) 证明, 每当 $u(\mathbf{x}) = v(\mathbf{x})w(\mathbf{x})$ 是齐次的时, $B(u) \geq B(v)B(w)$ 。

c) 当 u 是 n 次齐次时, 一个多项式 $u(\mathbf{x})$ 的 Bombieri 范数 $[u]$ 定义为 $\sqrt{B(u)/n!}$ 。对于非齐次多项式也可定义它, 办法是增加一个新变量 x_{t+1} 并且以 x_{t+1} 的幂来乘每一项, 使得 u 变成齐次的而不增加它的极大次数。例如, 设 $u(x) = 4x^3 + x - 2$; 对应的齐次多项式是 $4x^3 + xy^2 - 2y^3$, 因而我们有 $[u]^2 = (3! \cdot 0! \cdot 4^2 + 1! \cdot 2! \cdot 1^2 + 0! \cdot 3! \cdot 2^2)/3! = 16 + \frac{1}{3} + 4$ 。如果 $u(x, y, z) = 3xy^3 - z^2$, 类似地我们有 $[u]^2 = (1! \cdot 3! \cdot 0! \cdot 0! \cdot 3^2 + 0! \cdot 0! \cdot 2! \cdot 2! \cdot 1^2)/4! = \frac{9}{4} + \frac{1}{6}$ 。当 $u(\mathbf{x}) = v(\mathbf{x}) \cdot w(\mathbf{x})$ 时, 关于 $[u], [v]$ 和 $[w]$ 之间的关系, b) 告诉我们什么?

d) 证明如果 $u(x)$ 是一个变量的 n 次可约多项式, 它有其系数的绝对值顶多是 $n!^{1/4} [u]^{1/2} (n/4)!$ 的一个因子。问 t 个变量的齐次多项式对应的结果是什么?

e) 当 $u(x) = (x^2 - 1)^n$ 时, 明确地和近似地计算 $[u]$ 。

f) 证明 $[u][v] \geq [uv]$ 。

g) 证明: 当 $u(x)$ 是一个 n 次多项式而 $M(u)$ 是习题 20 中定义的量时,

$$2^{-n/2} M(u) \leq [u] \leq 2^{n/2} M(u)$$

(因此 d) 的上限粗略地是在该题中我们得到的上限的平方根。)

► 22. [M24] (Hensel 引理) 设 $u(x), v_e(x), w_e(x), a(x), b(x)$ 是具有整系数的多项式, 并满足关系式

$$u(x) \equiv v_e(x)w_e(x) \pmod{p^e}$$

$$a(x)v_e(x) + b(x)w_e(x) \equiv 1 \pmod{p}$$

其中 p 是素数, $e \geq 1$, $v_e(x)$ 是首一的, $\deg(a) < \deg(w_e)$, $\deg(b) < \deg(v_e)$, 且 $\deg(u) = \deg(v_e) + \deg(w_e)$ 。说明当 e 增 1 时怎样计算满足同样条件的多项式 $v_{e+1}(x) \equiv v_e(x)$ 和 $w_{e+1}(x) \equiv w_e(x) \pmod{p^e}$ 。而且证明在模 p^{e+1} 下, $v_{e+1}(x)$ 和 $w_{e+1}(x)$ 是惟一的。

由在习题 10 中求出的在模 2 下 (22) 的因子分解开始, 对 $p=2$ 使用你的方法证明 (22) 是整

数上不可约的(注意欧几里得扩充算法,习题4.6.1-3将从 $e=1$ 开始这个过程)。

23. [HM23] 设 $u(x)$ 是整系数的无平方多项式。证明仅有有限多个素数使得 $u(x)$ 不是模 p 无平方的。

24. [M20] 正文仅谈及整数上的因子分解,而没有谈及有理数上的因子分解。请说明怎样在有理数域上求有理系数的一个多项式的完全因子分解。

25. [M25] 在有理数域上,什么是 $x^5 + x^4 + x^3 + x + 2$ 的完全因子分解?

26. [20] 设 d_1, \dots, d_r 是模 p 之下 $u(x)$ 的不可约因子的次数,各以适当的重数出现,使得 $d_1 + \dots + d_r = n = \deg(u)$ 。说明怎样通过对长度为 n 的二进位串执行 $O(r)$ 个操作,计算集合 $\{\deg(v) \mid \text{对于某个 } v(x), w(x), u(x) \equiv v(x)w(x) \pmod{p}\}$ 。

27. [HM30] 证明在某种适当的意义下,整数上的一个随机本原多项式“几乎总是”不可约的。

28. [M25] 当对于每个次数 d 至多有一个不可约多项式时不同次数的因子分解过程是“幸运的”;此时完全不需要把 $g_d(x)$ 分解因子。当 $p \rightarrow \infty$ 时,对于固定的 n ,在模 p 下分解一个 n 次随机多项式时,这样一种走运的情况的概率是多少?

29. [M22] 设 $g(x)$ 是在模一个奇素数 p 下,两个或多个次数为 d 的不同不可约多项式的乘积。证明:当 $t(x)$ 从模 p 下次数小于 $2d$ 的 p^{2d} 个多项式中随机地选择时,对任何固定的 $g(x)$, $\gcd(g(x), t(x)^{(p^d-1)/2} - 1)$ 将是 $g(x)$ 的真因子的概率大于等于 $1/2 - 1/(2p^d)$ 。

30. [M25] 证明如果 $q(x)$ 是模 p 下次数为 d 的一个不可约多项式,而且 $t(x)$ 是任意的多项式,则 $(t(x) + t(x)^p + t(x)^{p^2} + \dots + t(x)^{p^{d-1}}) \bmod q(x)$ 的值是一个整数(即次数 ≤ 0 的一个多项式)。利用这一事实,设计在 $p=2$ 的情况下,分解类似于(21)的 d 次不可约多项式的乘积 $g_d(x)$ 的一个随机化算法。

31. [HM30] 设 p 是奇素数且设 $d \geq 1$, 证明存在一个数 $n(p, d)$, 它有以下两个性质: (i) 对所有整数 t , 在模 p 下恰有 $n(p, d)$ 个 d 次不可约多项式 $q(x)$ 满足 $(x+t)^{(p^d-1)/2} \bmod q(x) = 1$ 。 (ii) 对所有整数 $0 \leq t_1 < t_2 < p$, 在模 p 下恰有 $n(p, d)$ 个 d 次不可约多项式 $q(x)$ 满足 $(x+t_1)^{(p^d-1)/2} \bmod q(x) = (x+t_2)^{(p^d-1)/2} \bmod q(x)$ 。

► 32. [M30] (循环多项式) 设 $\Psi_n(x) = \prod_{1 \leq k \leq n, k \perp n} (x - \omega^k)$, 其中 $\omega = e^{2\pi i/n}$; 于是, $\Psi_n(x)$ 的根是单位的复 n 次根, 对于 $m < n$ 它不是 m 次根,

a) 证明 $\Psi_n(x)$ 是一个整系数多项式, 而且

$$x^n - 1 = \prod_{d \mid n} \Psi_d(x); \quad \Psi_n(x) = \prod_{d \mid n} (x^d - 1)^{\mu(n/d)}$$

(参考习题4.5.2-10b)及4.5.3-28c)。

b) 证明 $\Psi_n(x)$ 是整数上的不可约多项式, 因此上述公式是整数上 $x^n - 1$ 的完全因子分解。
[提示: 如果 $f(x)$ 是整数上 $\Psi_n(x)$ 的一个不可约因子, 且 ζ 是使 $f(\zeta) = 0$ 的一个复数。证明对所有不整除 n 的素数 p , $f(\zeta^p) = 0$ 。使用下列事实可能是有帮助的, 对所有这样的素数, 在模 p 下, $x^n - 1$ 是无平方的。]

c) 讨论 $\Psi_n(x)$ 的计算, 而且对于 $n \leq 15$ 把这些值造成表。

33. [M18] 真或假: 如果 $u(x) \neq 0$, 且 $u(x)$ 模 p 的完全因子分解是 $p_1(x)^{e_1} \cdots p_r(x)^{e_r}$, 则 $u(x)/\gcd(u(x), u'(x)) = p_1(x) \cdots p_r(x)$ 。

► 34. [M25] (无平方因子分解) 显然, 一个惟一因子分解整环的任何本原多项式都可表达成形式 $u(x) = u_1(x)u_2(x)^2u_3(x)^3 \cdots$, 其中多项式 $u_i(x)$ 是无平方的且彼此互素。这个表示除单位

的倍数外是惟一的,其中 $u_j(x)$ 是恰好 j 次整除 $u(x)$ 的所有不可约多项式的乘积;而且,它是表示那些参与乘法、除法及 gcd 运算的多项式的有用的方法。

设 $\text{GCD}(u(x), v(x))$ 是生成三个答案

$$\text{GCD}(u(x), v(x)) = (d(x), u(x)/d(x), v(x)/d(x))$$

的一个过程,其中 $d(x) = \text{gcd}(u(x), v(x))$ 。等式(25)下边的正文中所描述的取模方法总是以 $u(x)/d(x)$ 和 $v(x)/d(x)$ 的试除结束,以确保没有使用“倒霉的素数”,所以量 $u(x)/d(x)$ 和 $v(x)/d(x)$ 是 gcd 计算的副产品。于是当我们使用取模法时,实质上可以像计算 $\text{gcd}(u(x), v(x))$ 那样快地计算 $\text{GCD}(u(x), v(x))$ 。

试设计一个过程,它得到整数上一个给定本原多项式 $u(x)$ 的无平方表示 $(u_1(x), u_2(x), \dots)$ 。你的算法应当恰恰执行 e 次 GCD 的计算,其中 e 是使 $u_e(x) \neq 1$ 的最大下标,而且每个 GCD 计算都应满足(27),以便能使用 Hensel 构造。

35. [M22] (D. Y. Y. Yun) 试设计一个算法,给定 $u(x)$ 和 $v(x)$ 的无平方表示 $(u_1(x), u_2(x), \dots)$ 及 $(v_1(x), v_2(x), \dots)$, 它计算整数上 $w(x) = \text{gcd}(u(x), v(x))$ 的无平方表示 $(w_1(x), w_2(x), \dots)$ 。

36. [M27] 推广习题 34 的过程,使得当系数算术是在模 p 下进行时,它将得到一个给定多项式 $u(x)$ 的无平方表示 $(u_1(x), u_2(x), \dots)$ 。

37. [HM24] (George E. Collins) 设 d_1, \dots, d_r 是和为 n 的正整数,并设 p 为素数。当在模 p 之下,一个 n 次随机整多项式 $u(x)$ 完全分解时,它的不可约因子有次数 d_1, \dots, d_r 的概率是多少? 证明这个概率渐近地和 n 个元素的一个随机排列有长度为 d_1, \dots, d_r 的循环之概率相同。

38. [HM27] (Perron 准则) 设 $u(x) = x^n + u_{n-1}x^{n-1} + \dots + u_0$ 是具有整系数且使得 $u_0 \neq 0$ 以及或者 $|u_{n-1}| > 1 + |u_{n-2}| + \dots + |u_0|$ 或者 $(u_{n-1} = 0$ 且 $u_{n-2} > 1 + |u_{n-3}| + \dots + |u_0|)$ 的一个多项式。证明 $u(x)$ 在整数上是不可约的。[提示:证明几乎所有 u 的根的绝对值都小于 1。]

39. [HM42] (David G. Cantor) 证明如果多项式 $u(x)$ 是在整数上不可约的,在其不可约性证明中二进位的个数至多是 $\deg(u)$ 和系数的长度的多项式,则在此意义下它有一个不可约性的“简明的”证明。(如同在习题 4.5.4-17 一样,这里要求的仅仅是证明的长度的上限,而不是为求这样一个证明所需要的时间的上限。)提示:如果 $v(x)$ 是不可约的而且 t 是整数上的任何多项式,则 $v(t(x))$ 的所有因子都有大于等于 $\deg(v)$ 的次数。Perron 准则提供了大量的不可约多项式 $v(x)$ 。

▶ 40. [M20] (P. S. Wang) 如果 u_n 是 $u(x)$ 的前导系数和 B 是 u 的某个因子的系数的上限,正文的因子分解算法要求我们在模 p^e 之下分解一个因子,其中 $p^e > 2|u_n|B$, 但当 B 是由习题 21 的方法选定的时, $|u_n|$ 可能比 B 还大。证明如果 $u(x)$ 是可约的,则通过使用习题 4.5.3-51 的算法,每当 $p^e \geq 2B^2$ 时,存在一个方法从在模 p^e 之下的一个因子恢复它真正的因子之一。

41. [M47] (Beauzamy, Trevisan 和 Wang) 证明或反驳:有一个常数 c ,使得如果 $f(x)$ 是任意的整数多项式且其所有系数的绝对值小于等于 B ,则它的不可约因子之一有以 cB 为界的系数。

4.6.3 求幂值

在这一小节中我们将研究有效地计算 x^n 的有趣问题, x 和 n 是给定的,其中 n 是正整数。例如,假定要计算 x^{16} ;我们可以简单地由 x 开始并且乘以 x 十五次。但是如果反复地取每一部分结果的平方,逐次形成 x^2, x^4, x^8, x^{16} ,则有可能仅

使用四次乘法就得到相同的答案。

一般来说,同样的思想可以如下的方式应用于任何 n 值:以二进制数系来写 n (去掉左边的零)。然后以字母对 SX 代替每个“1”,以 S 代替每个“0”,而且划掉现在在左边出现的“SX”。如果“S”解释为平方运算,“X”解释为乘以 x 的运算,则这个结果就成为计算 x^n 的一个规则。例如,如果 $n=23$,则它的二进制表示是 10111;所以我们构造 SX S SX SX SX,并删去前导的 SX 而得到规则 SSXSXSX。这个规则指出了我们应该“平方,平方,乘以 x ,平方,乘以 x ,平方,乘以 x ”;换言之,我们将逐次计算 $x^2, x^4, x^5, x^{10}, x^{11}, x^{22}, x^{23}$ 。

通过考虑计算中的指数序列,不难论证这一“二进方法”:如果我们重新把“S”解释为乘以 2 的运算,并把“X”解释为加 1 的运算,而且如果我们以 1 而不是以 x 开始,由于二进制数系的性质,这一规则将导出 n 的计算。这个方法是十分古老的,公元前 200 年它就出现于 Pingala 的印度经典著作 *Chandaś-sūtra* 中[见 B. Datta 和 A. N. Singh, *History of Hindu Mathematics* 2 (Lahore: Motilal Banarsi Das, 1935), 76]。在其后的一千多年间,似乎在印度之外没有人提及这个方法,但在公元 952 年大马上革的 al-Uqlidisi 给出了对于任意 n 怎样有效地计算 2^n 的清晰的讨论;见 A. S. Saidan, *The Arithmetic of al-Uqlidisi* (Dordrecht: D. Reidel, 1975), 341~342, 其中对 $n=51$ 说明了一般的思想。也见 al-Uqlidisi, *Chronology of Ancient Nations*, E. Sachau 编译 (London: 1879), 132~136; 这部 11 世纪的阿拉伯著作曾产生巨大的影响。

用于得到 x^n 的这个 S 和 X 的二进方法,除了 x 和当前的部分结果外并不需要临时存储,所以它很适合于放入一台二进制计算机的硬件中。这个方法很容易编成程序;但它要求自左至右地扫描 n 的二进表示。计算机程序一般喜欢以另一个方向进行,因为可用的除以 2 和模 2 剩余的运算都将推出从右至左的二进表示。因此,基于自右至左地对数进行扫描为基础的下列算法,通常是更为方便的:

算法 A (指数的自右至左的二进方法) 这一算法计算 x^n , 其中 n 是正整数。(这里 x 属于任何代数系统,其中已经定义了一个可结合乘法,并有幺元 1。)

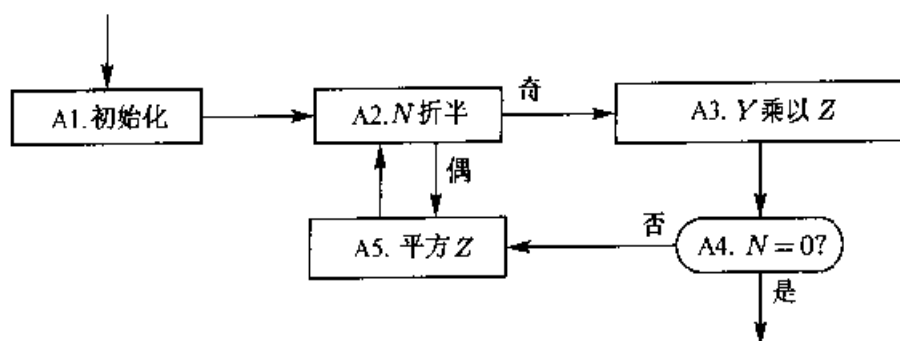


图 13 x^n 的求值,以对 n 的二进制表示自右至左地进行扫描为基础

A1. [初始化] 置 $N \leftarrow n, Y \leftarrow 1, Z \leftarrow x$ 。

A2. [N 折半] (这时,我们有 $x^n = YZ^N$ 。)置 $N \leftarrow \lfloor N/2 \rfloor$, 同时确定 N 是偶数还

是奇数。若 N 是偶,则跳到 A5。

A3. [Y 乘以 Z] 置 $Y \leftarrow Z$ 乘 Y 。

A4. [$N=0?$] 如果 $N=0$ 则算法终止,且 Y 是答案。

A5. [平方 Z] 置 $Z \leftarrow Z$ 乘 Z ,并返回步骤 A2。 ─

作为算法 A 的一个例子,考虑在计算 x^{23} 中的步骤:

	N	Y	Z
步骤 A1 后	23	1	x
步骤 A5 后	11	x	x^2
步骤 A5 后	5	x^3	x^4
步骤 A5 后	2	x^7	x^8
步骤 A5 后	1	x^7	x^{16}
步骤 A4 后	0	x^{23}	x^{16}

对应于算法 A 的一个 MIX 程序见习题 2。

伟大的计算家 al-Kāshī 在公元 1427 年就指出了算法 A [Istoriko-Mat. Issledovaniia 7 (1954), 256~257]。这个算法同埃及数学家早在公元前 2000 年实际使用的乘法过程有密切关系;因为如果我们把步骤 A3 改成“ $Y \leftarrow Y + Z$ ”,步骤 A5 改成“ $Z \leftarrow Z + Z$ ”,而且我们把 Y 置成 0 以代替步骤 A1 中的置成 1,则此算法就以 $Y = nx$ 终止。[见 A. B. Chace, *The Rhind Mathematical Papyrus* (1927); W. W. Struve, *Quellen und Studien zur Geschichte der Mathematik A1* (1930)。]这是手算乘法的一个实用方法,因为它仅仅包含简单的加法、折半和相加的运算。通常把它叫做乘法的“俄国农民方法”,因为 19 世纪时西方访问者在俄国发现俄国农民广泛使用这个方法。

算法 A 所需要的乘法次数为

$$\lfloor \lg n \rfloor + \nu(n)$$

其中 $\nu(n)$ 是 n 的二进表示中 1 的个数。这比在本节开始时提出的自左到右的二进方法只多做一次乘法,因为这里第一次执行步骤 A3 时只不过是做乘以 1 的乘法。

由于这个算法要求簿记时间,故二进方法对于小的 n 值,比如说, $n \leq 10$,通常并没有价值,除非做一次乘法需要相当长的时间。如果预先知道 n 的值,则自左至右的二进方法是可取的。在某些情况下,例如 4.6.2 小节中讨论的 $x^n \bmod u(x)$ 的计算,乘以 x 要比进行一般的乘法或者对一个值求平方容易得多,故在这种情况下对于指数的二进方法主要适合于十分大的 n 。当 n 大于计算机字长时,如果我们希望计算 x^n 的精确的多精度的值,除非 n 是如此之大,以致要使用 4.3.3 小节的高速乘法程序,否则二进方法毫无帮助;而这样的应用是很少的。类似地,二进方法通常也不适宜于对一个多项式作乘方,关于多项式指数的广泛著作的讨论,见 R. J.

Fateman, *SICOMP* 3 (1974), 196~213。

这些评述的要点在于,二进方法是很好的,但不是万灵药。当 $x^j \cdot x^k$ 相乘的时间实质上与 j 和 k 无关时(例如,对于浮点乘法,或模 m 乘法)它们最适用;在这种情况下,运行时间从阶 n 减少成阶 $\log n$ 。

更少的乘法 若上作者断言(没有证明)二进方法实际上给出了可能的最小的乘法数,但这是不对的。最小的反例是 $n = 15$,这时二进方法做6次乘法,然而我们可以以两次乘法计算 $y = x^3$,并且以另外的三次计算 $x^{15} = y^5$,即仅仅用五次乘法就达到所求结果。现在让我们讨论用于计算 x^n 的其它某些过程,并假定 n 是预先知道的。这样的过程是有意义的,例如,当一个优化的编译程序生成机器代码时。

因子方法是以 n 的因子分解为基础的。如果 $n = pq$,其中 p 是 n 的最小素因子,而且 $q > 1$,则我们可以首先计算 x^p 而后对这个量作乘方到 q 次幂来计算 x^n 。如果 n 为素数,则我们可以计算 x^{n-1} 并乘以 x 。当然,如果 $n = 1$,我们全然无须进行计算。对于任何给定的 n ,反复应用这些规则,就可完成计算 x^n 的过程。例如,如果我们要计算 x^{55} ,我们首先计算 $y = x^5 = x^4 x = (x^2)^2 x$;然后我们构造 $y^{11} = y^{10} y = (y^2)^5 y$ 。整个过程需要8个乘法,而二进方法需要9个。平均说来,因子方法比二进方法更好些,但是也有一些情况($n = 33$ 是最小的例子),其中二进方法优于因子方法。

二进方法可以推广到 m 进方法如下:设 $n = d_0 m^t + d_1 m^{t-1} + \cdots + d_t$,其中对于 $0 \leq j \leq t, 0 \leq d_j < m$,这个计算以构造 $x, x^2, x^3, \cdots, x^{m-1}$ 开始。(实际上,仅仅需要使 d_j 出现在 n 的表示中的幂 x^{d_j} ,注意到这一点通常可节省一些工作量。)然后对 x^{d_0} 作乘方到 m 次幂,并乘以 x^{d_1} ;我们已经计算出 $y_1 = x^{d_0 m + d_1}$ 。其次,对 y_1 作乘方到 m 次幂,并乘以 x^{d_2} ,得到 $y_2 = x^{d_0 m^2 + d_1 m + d_2}$ 。这样继续进行这一过程直到算出 $y_t = x^n$ 为止。每当 $d_j = 0$ 时,它当然不需要乘以 x^{d_j} 。注意,当 $m = 2$ 时,这个方法归纳为早先讨论的自左至右的二进方法。还有一个不太明显的自右至左的 m 进方法,它花费更多的内存但只增加了不多的步数(见习题9)。如果 m 是一个小素数,当诸系数作模 m 处理时, m 进方法对于计算一个多项式在模另一个多项式之下的乘方特别有效(见等式4.6.2-(5))。

图14中示出了对于所有比较小的 n 值(特别是对于实际应用中出现的大多数 n),只需要最少的乘法次数的一个系统的方法。为了计算 x^n ,先在这棵树中找到 n ,于是从根到 n 的通路指出在 x^n 的有效计算中出现的指数序列。生成这棵“幂树”的规则见习题5。计算机检验已经表明,对于图中列出的所有 n 值,这棵幂树都给出最优的结果。但对于充分大的 n 值,幂树方法并不总是最优的;最小的例子是 $n = 77, 154, 233$ 。幂树既优于二进方法也优于因子方法的第一个情况是 $n = 23$ 。因子方法超过幂树方法的第一种情况是 $n = 19879 = 103 \cdot 193$;这样的情况是十分罕见的。(对于 $n \leq 100\,000$,幂树方法好于因子方法88 803次;打平11 191次;失利仅6次。)

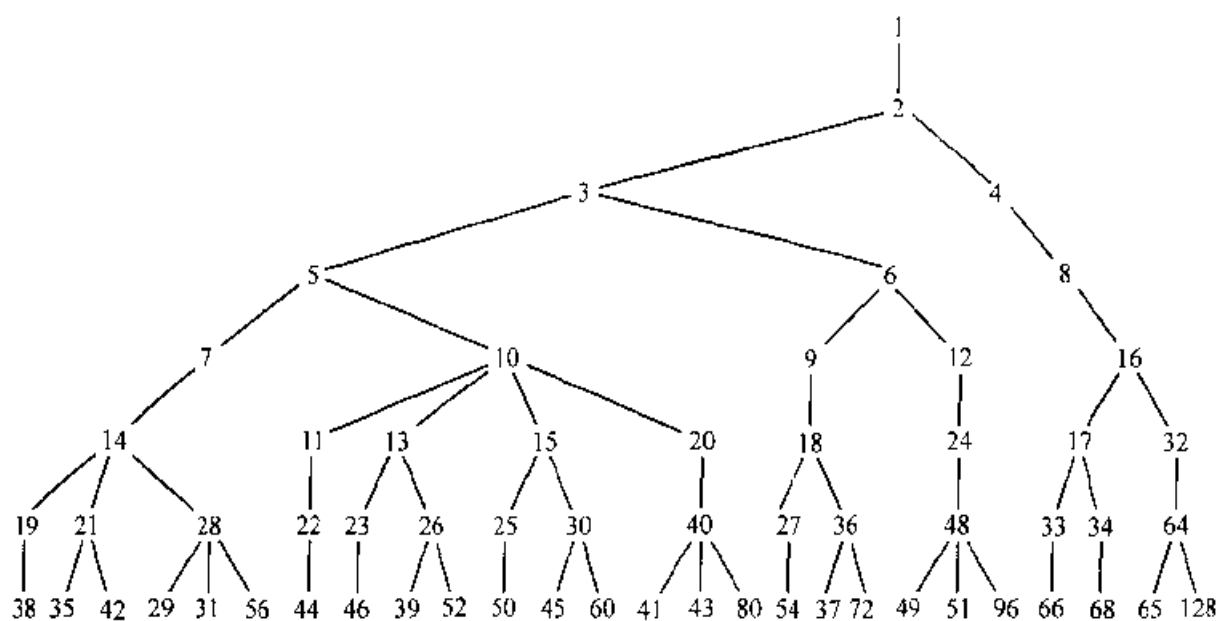


图 14 “幂树”

加法链 通过乘法计算 x^n 的最经济的方式,是一个具有有趣历史的数学问题。我们现在将详细地考察它,这不仅因为它本身是有趣的,而且还因为它是研究“最优计算方法”时遇到的理论问题的一个突出例子。

尽管我们涉及的是 x 的幂的乘法,但是这个问题可以很容易地归结为加法,因为指数可以相加。这导致下列的抽象表述: n 的一个加法链是整数序列

$$1 = a_0, a_1, a_2, \dots, a_r = n \quad (1)$$

它具有下列性质:对于所有 $i = 1, 2, \dots, r$, 对于某个 $k \leq j < i$,

$$a_i = a_j + a_k \quad (2)$$

考察这个定义的一个途径是考虑一台简单的计算机,它有一个累加器,并有三个运算符 LDA, STA 和 ADD; 机器开工时在它的累加器中有数 1,它通过把以前的结果加在一起来计算数 n 。注意, a_1 必须等于 2,而 a_2 是 2, 3 或 4。

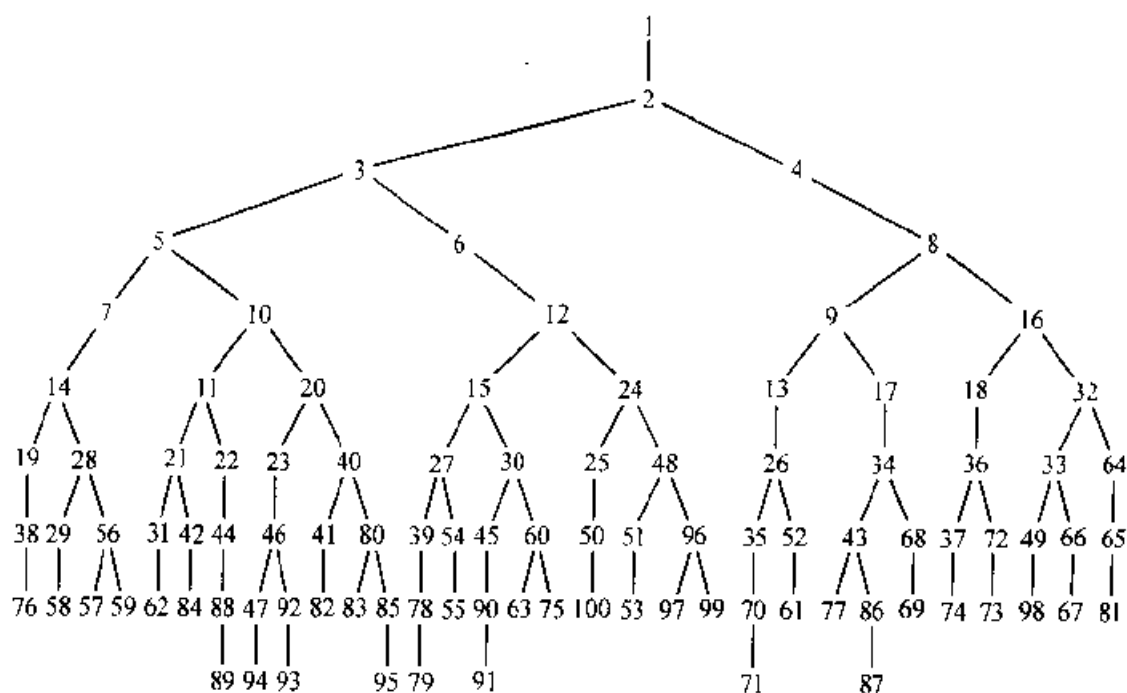
以 $l(n)$ 表示存在的 n 的加法链的最小长度 r , 则 $l(1) = 0, l(2) = 1, l(3) = l(4) = 2$, 等等。本小节的剩下部分就是要尽可能地多发现关于这个函数的性质。图 15 以树形显示了对于小 n 的 $l(n)$ 值,它表明对于所有 $n \leq 100$, 怎样以最少的乘法计算 x^n 。

确定 $l(n)$ 的问题看来首先是由 H. Dhall 于 1894 年提出的, 而由 E. de Jonquières 给出的一个部分解提到了因子方法[见 *L'Intermédiaire des Mathématiciens* 1(1894), 20, 162~164]。在 de Jonquières 的解中,他列出了对于所有素数 $p < 200$ 他所认为 $l(p)$ 的值,但是对于 $p = 107, 149, 163, 179$, 他的表值高出 1。

因子方法立即告诉我们

$$l(mn) \leq l(m) + l(n) \quad (3)$$

因为我们可以取链 $1, a_1, \dots, a_r = m$ 和 $1, b_1, \dots, b_s = n$, 并构造链 $1, a_1, \dots, a_r,$

图 15 对 $n \leq 100$, 最小化乘法次数的树

$a, b_1, \dots, a, b_s = mn$ 。

我们也可以把 m 进方法加到加法链的术语中去。考虑 $m = 2^k$ 的情况, 并以 m 进数系写 $n = d_0 m^t + d_1 m^{t-1} + \dots + d_t$; 对应的加法链具有形式

$$\begin{aligned}
 &1, 2, 3, \dots, m-2, m-1 \\
 &2d_0, 4d_0, \dots, md_0, md_0 + d_1 \\
 &2(md_0 + d_1), 4(md_0 + d_1), \dots, m(md_0 + d_1), m^2 d_0 + md_1 + d_2, \\
 &\dots, m^t d_0 + m^{t-1} d_1 + \dots + d_t
 \end{aligned} \quad (4)$$

这个链的长度是 $m-2-(k+1)t$; 而且通过消去未出现诸系数 d_i 的头一行的某些元素, 加上头一行中已经出现的 $2d_0, 4d_0, \dots$ 当中的一些元素, 通常它还可以缩短。每当数字 d_i 为 0 时, 相应行右端的步骤当然可以中止。而且, 正如 E. G. Thurber 已经发现的 [Duke Math. J. 40 (1973), 907~913], 如果我们提前 e 步把形如 $d_i/2^e$ 的值投入计算, 我们就可省略头一行中所有的偶数 (除 2 以外)。

当一般方案 (4) 简化成本小节开始处提到的“S”和“X”规则时, m 进方法最简单的情况是二进方法 ($m=2$): 对于 $2n$ 来说, 二进加法链是对于 n 的二进链后边跟以 $2n$; 对于 $2n+1$ 来说, 它是 $2n$ 的二进链后边跟以 $2n+1$ 。由二进方法我们得出如下结论:

$$l(2^{e_0} + 2^{e_1} + \dots + 2^{e_t}) \leq e_0 + t, \text{ 如果 } e_0 > e_1 > \dots > e_t \geq 0 \quad (5)$$

为了下边讨论方便, 现在我们定义两个辅助函数:

$$\lambda(n) = \lfloor \lg n \rfloor \quad (6)$$

$$\nu(n) = \text{在 } n \text{ 的二进表示中 } 1 \text{ 的个数} \quad (7)$$

于是 $\lambda(17) = 4, \nu(17) = 2$; 这些函数可以由递推关系定义:

$$\lambda(1) = 0; \quad \lambda(2n) = \lambda(2n+1) = \lambda(n) + 1 \quad (8)$$

$$\nu(1) = 1, \quad \nu(2n) = \nu(n), \quad \nu(2n+1) = \nu(n) + 1 \quad (9)$$

用这些函数来表示, n 的二进加法链恰要求 $\lambda(n) + \nu(n) - 1$ 步, 而且(5)变成

$$l(n) \leq \lambda(n) + \nu(n) - 1 \quad (10)$$

特殊的链类 不失任何一般性, 我们可以假设, 一个加法链是递升的:

$$1 = a_0 < a_1 < a_2 < \cdots < a_r = n \quad (11)$$

因为如果任何两个 a 相等, 则它们之一就可以丢掉; 而且我们也可以把序列(1)重新安排成递升顺序并删去大于 n 的项而不破坏加法链的性质(2)。从现在起我们将只考虑递升链而不明确地提出这个假定。

这时定义一些关于加法链的特别术语是方便的。自定义, 对于 $1 \leq i \leq r$, 对于某个 j 和 $k, 0 \leq k \leq j < i$, 我们有

$$a_i = a_j + a_k \quad (12)$$

如果这个关系对于多于一对的 (j, k) 成立, 我们就令 j 尽可能地大。如果 $j = k = i-1$, 我们就说(11)的步骤 i 是加倍; 于是 a_i 有可以跟在递增的链 $1, a_1, \dots, a_{i-1}$ 之后的极大值 $2a_{i-1}$ 。如果 j (但不必是 k) 等于 $i-1$, 我们就说步骤 i 是一个星步。下面要说明星步的重要性。最后如果 $\lambda(a_i) = \lambda(a_{i-1})$, 我们就说步骤 i 是一个小步。由于 $a_{i-1} < a_i \leq 2a_{i-1}$, 量 $\lambda(a_i)$ 总是等于 $\lambda(a_{i-1})$ 或 $\lambda(a_{i-1}) + 1$; 由此得出, 在任何链(11)中, 长度 r 等于 $\lambda(n)$ 加上小步的个数。

在这些步型之间有若干初等关系成立: 步骤 1 总是一个加倍, 一个加倍显然是一个星步, 但绝不是一个步。一个加倍后边必须跟着一个星步。而且如果步 i 不是一个小步, 则步 $i+1$ 或者是一个小步或者是一个星步, 或者同为两者; 换句话说, 如果步 $i+1$ 既不是小步也不是星步, 则步 i 必然是小步。

一个星链是仅含有星步的一个加法链。这意味着每项 a_i 是 a_{i-1} 和以前的一个 a_k 之和; 上边等式(2)之后讨论的简单“计算机”在一个星链中只利用两个操作 STA 和 ADD (而不是 LDA), 因为这序列的每个新的项利用累加器中以前的结果。至今我们已经讨论的大多数加法链都是星链。 n 的一个星链的极小长度以 $l^*(n)$ 表示; 显然

$$l(n) \leq l^*(n) \quad (13)$$

我们现在准备来推导关于加法链的一些不平凡的事实。首先我们可以证明, 如果 r 离 $\lambda(n)$ 不远, 则必定存在相当多的加倍。

定理 A 如果加法链(11)包括 d 个加倍和 $f = r - d$ 个非加倍, 则

$$n \leq 2^{d-1} F_{f+3} \quad (14)$$

证明 对 $r = d + f$ 用归纳法, 当 $r = 1$ 时我们看到(14)肯定是真的。当 $r > 1$ 时, 有三种情况: 如果步 r 是一个加倍, 则 $\frac{1}{2}n = a_{r-1} \leq 2^{d-2} F_{f+3}$; 因此得出(14)。

如果步 r 和 $r-1$ 都非加倍, 则 $a_{r-1} \leq 2^{d-1} F_{f+2}$ 且 $a_{r-2} \leq 2^{d-1} F_{f+1}$; 因此由斐波那契序列的定义, $n = a_r \leq a_{r-1} + a_{r-2} \leq 2^{d-1} (F_{f+2} + F_{f+1}) = 2^{d-1} F_{f+3}$ 。最后, 如果步 r 不是一个加倍但步 $r-1$ 是一个加倍, 则 $a_{r-2} \leq 2^{d-2} F_{f+2}$ 且 $n = a_r \leq a_{r-1} + a_{r-2} = 3a_{r-2}$ 。现在 $2F_{f+3} - 3F_{f+2} = F_{f+1} - F_f \geq 0$; 因此在所有情况下 $n \leq 2^{d-1} F_{f+3}$ 。■

我们使用的证明方法表明不等式(14)在所述假定下是“最好的”; 加法链

$$1, 2, \dots, 2^{d-1}, 2^{d-1} F_3, 2^{d-1} F_4, \dots, 2^{d-1} F_{f+3} \quad (15)$$

有 d 个加倍和 f 个非加倍。

推论 如果加法链(11)包括 f 个非加倍和 s 个小步, 则

$$s \leq f \leq 3.271s \quad (16)$$

证明 显然, $s \leq f$ 。我们有 $2^{\lambda(n)} \leq n \leq 2^{d-1} F_{f+3} \leq 2^d \phi^f = 2^{\lambda(n)+s} (\phi/2)^f$, 因为 $d+f = \lambda(n)+s$, 而且因为当 $f \geq 0$ 时, $F_{f+3} \leq 2\phi^f$ 。因此 $0 \leq s \ln 2 + f \ln(\phi/2)$, 由事实 $\ln 2 / \ln(2/\phi) \approx 3.2706$ 得出(16)。■

对于特殊 n 的 $l(n)$ 值 用归纳法容易证明 $a_i \leq 2^i$, 因此在任何加法链(11)中 $\lg n \leq r$ 。因此

$$l(n) \geq \lceil \lg n \rceil \quad (17)$$

这一下界, 连同由二进方法给出的上界(10)一起, 给了我们值

$$l(2^A) = A \quad (18)$$

$$l(2^A + 2^B) = A + 1, \quad \text{如果 } A > B \quad (19)$$

换句话说, 当 $v(n) \leq 2$ 时, 二进方法是最优的。通过某些进一步的计算, 我们可以把这些公式推广到 $v(n) = 3$ 的情况:

定理 B 如果 $A > B > C$, 则

$$l(2^A + 2^B + 2^C) = A + 2 \quad (20)$$

证明 事实上, 我们可以证明一个更强的结果, 我们在本节后面还要用到它: 所有恰恰含有一个小步的加法链必有下列六种类型之一 (其中所有以“...”表示的步都表示加倍):

类型 1 $1, \dots, 2^A, 2^A + 2^B, \dots, 2^{A+C} + 2^{B+C}; A > B \geq 0, C \geq 0$

类型 2 $1, \dots, 2^A, 2^A + 2^B, 2^{A+1} + 2^B, \dots, 2^{A+C+1} + 2^{B+C}; A > B \geq 0, C \geq 0$

类型 3 $1, \dots, 2^A, 2^A + 2^{A-1}, 2^{A+1} + 2^{A-1}, 2^{A+2}, \dots, 2^{A+C}; A > 0, C \geq 2$

类型 4 $1, \dots, 2^A, 2^A + 2^{A-1}, 2^{A+1} + 2^A, 2^{A+2}, \dots, 2^{A+C}; A > 0, C \geq 2$

类型 5 $1, \dots, 2^A, 2^A + 2^{A-1}, \dots, 2^{A+C} + 2^{A+C-1}, 2^{A+C+1} + 2^{A+C-2}, \dots, 2^{A+C+D+1} + 2^{A+C+D-2}; A > 0, C > 0, D \geq 0$

类型 6 $1, \dots, 2^A, 2^A + 2^B, 2^{A+1}, \dots, 2^{A+C}; A > B \geq 0, C \geq 1$

直截了当的手算表明, 这六种类型已包括了所有的可能性。由定理 A 的推论可知, 当有一个小步时, 至多有三个非加倍; 这极大值仅在类型 3 的序列中才会出现。

上边所有类型都是星链,只有当 $B < A - 1$ 时的类型 6 除外。

现在从下列观察

$$l(2^A + 2^B + 2^C) \leq A + 2$$

和 $l(2^A + 2^B + 2^C)$ 必定大于 $A + 1$, 即可得到定理, 因为六个可能的类型没有一个有 $\nu(n) > 2$ 成立。 ■

(E. de Jonquières 于 1894 年未加证明指出, 当 $\nu(n) > 2$ 时 $l(n) \geq \lambda(n) + 2$ 。定理 B 的头一个公开的证明是由 A. A. Gioia, M. V. Subbarao 和 M. Sugunamma 给出的, 见 *Duke Math. J.* 29 (1962), 481 ~ 487。)

当 $A > B > C > D$ 时, 就要更多地涉及 $l(2^A + 2^B + 2^C + 2^D)$ 的计算。通过二进方法, 它至多是 $A + 3$, 而由定理 B 的证明, 它至少是 $A + 2$ 。 $A + 2$ 的值是可能的, 因为我们知道当 $n = 15$ 或 $n = 23$ 时, 二进方法不是最优的。现在我们将看到, 当 $\nu(n) = 4$ 时, 它的完整特性是可确定的。

定理 C 如果 $\nu(n) \geq 4$, 则 $l(n) \geq \lambda(n) + 3$, 只有当 $A > B > C > D$ 和 $l(2^A + 2^B + 2^C + 2^D)$ 等于 $A + 2$ 时的下列情况例外:

情况 1 $A - B = C - D$ (例子: $n = 15$)

情况 2 $A - B = C - D + 1$ (例子: $n = 23$)

情况 3 $A - B = 3, C - D = 1$ (例子: $n = 39$)

情况 4 $A - B = 5, B - C = C - D = 1$ (例子: $n = 135$)

证明 当 $l(n) = \lambda(n) + 2$ 时, 有一个恰有两个小步的 n 的加法链。这样一个加法链在定理 B 的证明中从六个加法链之一着手, 后跟一个小步, 再后跟一系列非小步。如果对于定理 B 中所列的四种情况之一, $n = 2^A + 2^B + 2^C + 2^D$, 则我们说 n 是“特殊的”。如同习题 13 中所示, 对于每个特定的 n , 我们可以得到所要求形式的加法链; 因此, 剩下的只是证明, 除非 a_r 是特殊的, 否则在恰有两个小步的链中, 不包含满足 $\nu(a_r) \geq 4$ 的任何元素。

设“一个反例链”是具有两个小步的一个加法链, 使得 $\nu(a_r) \geq 4$, 但 a_r 是非特殊的。如果反例链存在, 则设 $1 = a_0 < a_1 < \cdots < a_r = n$ 是有最短长度的一个反例链。于是步 r 不是一个小步, 因为除了特殊的 n 以外, 定理 B 的证明中提出的六个类型中没有一个是以后跟具有 $\nu(n) \geq 4$ 的小步。此外, 步骤 r 不是加倍, 否则 a_0, \cdots, a_{r-1} 将是一个更短的反例链; 而且步骤 r 是一个星步, 否则 $a_0, \cdots, a_{r-2}, a_r$ 将是一个更短的反例链。于是

$$a_r = a_{r-1} + a_{r-k}, \quad k \geq 2; \quad \text{而且 } \lambda(a_r) = \lambda(a_{r-1}) + 1 \quad (21)$$

设 c 是由算法 4.3.1A 在二进数系中把 a_{r-1} 加到 a_{r-k} 上去时出现的进位个数, 利用基本的关系式

$$\nu(a_r) = \nu(a_{r-1}) + \nu(a_{r-k}) - c \quad (22)$$

我们现在可以证明步 $r-1$ 不是一个小步 (见习题 14)。

设 $m = \lambda(a_{r-1})$, 由于 r 或 $r-1$ 都不是一个小步, 因此 $c \geq 2$; 而且仅当

$a_{r-1} \geq 2^m + 2^{m-1}$ 时, 才有 $c=2$ 。

现在我们假设 $r-1$ 不是一个星步。则 $r-2$ 是一个小步, $a_0, \dots, a_{r-3}, a_{r-1}$ 是一个仅有一个小步的链; 因此 $\nu(a_{r-1}) \leq 2$ 且 $\nu(a_{r-2}) \leq 4$ 。现在关系 (22) 仅当 $\nu(a_r) = 4, \nu(a_{r-1}) = 2, k=2, c=2, \nu(a_{r-2}) = 4$ 时才成立。从 $c=2$ 我们得到结论, $a_{r-1} = 2^m + 2^{m-1}$; 因此 $a_0, a_1, \dots, a_{r-3} = 2^{m-1} + 2^{m-2}$ 是仅有一个小步的加法链, 而且它必是类型 1 的, 所以 a_r 属于情况 3。于是, $r-1$ 是一个星步。

现在假设对于某个 $t, a_{r-1} = 2^t a_{r-k}$ 。如果 $\nu(a_{r-1}) \leq 3$, 则由 (22), $c=2, k=2$, 而且我们看到, a_r 必属于情况 3。另一方面, 如果 $\nu(a_{r-1}) = 4$, 则 a_{r-1} 是特殊的, 而且通过考虑每种情况容易看到, a_r 也属于四种情况之一。(例如, 当 $a_{r-1} = 90, a_{r-k} = 45$ 时; 或当 $a_{r-1} = 120, a_{r-k} = 15$ 时出现情况 4。) 因此, 我们可以得出结论: 对任何 $t, a_{r-1} \neq 2^t a_{r-k}$ 。

我们已经证明, 对某个 $q \geq 2, a_{r-1} = a_{r-2} + a_{r-q}$ 。如果 $k=2$, 则 $q > 2$ 且 $a_0, a_1, \dots, a_{r-2}, 2a_{r-2}, 2a_{r-2} + a_{r-q} = a_r$ 是一个反例序列, 其中 $k > 2$; 因此我们可以假定 $k > 2$ 。

现在让我们假设 $\lambda(a_{r-k}) = m-1$; 如同习题 14 中所说明的, 通过类似的论证, $\lambda(a_{r-k}) < m-1$ 的情况可以排除。如果 $k=4$, 则 $r-2$ 和 $r-3$ 两者都是小步; 因此 $a_{r-4} = 2^{m-1}$, 而且 (22) 是不可能的。因此 $k=3$; 步 $r-2$ 是小步, $\nu(a_{r-3}) = 2, c=2, a_{r-1} \geq 2^m + 2^{m-1}$, 以及 $\nu(a_{r-1}) = 4$ 。当 a_{r-2} 加到 $a_{r-1} - a_{r-2}$ 上时, 至少应有两个进位; 因此 $\nu(a_{r-2}) = 4$, 且 a_{r-2} (是特殊的而且大于等于 $\frac{1}{2}a_{r-1}$) 对某个 d 有形式 $2^{m-1} + 2^{m-2} + 2^{d+1} + 2^d$ 。现在 a_{r-1} 或者是 $2^m + 2^{m+1} + 2^{d+1} + 2^d$, 或者是 $2^m + 2^{m-1} + 2^{d+2} + 2^{d+1}$, 而且在两种情况下 a_r 都必定是 $2^{m-1} + 2^{m-2}$, 所以 a_r 属于情况 3。 ■

E. G. Thurber [Pacific J. Math. 49 (1973), 229~242] 已经推广定理 C, 以证明当 $\nu(n) > 8$ 时 $l(n) \geq \lambda(n) + 4$ 。一般来说, 猜测 $l(n) \geq \lambda(n) + \lg \nu(n)$ 似乎是合理的, 因为 A. Schönhage 已经非常接近于证明了这一事实 (见习题 28)。

* 渐近值 定理 C 表明, 对于很大的 n , 当 $\nu(n) > 4$ 时, 要得到 $l(n)$ 的精确值大概是十分困难的; 然而, 我们可以确定当 $n \rightarrow \infty$ 时的极限的渐近特性。

定理 D (A. Brauer, Bull. Amer. Math. Soc. 45 (1939), 736~739)

$$\lim_{n \rightarrow \infty} l^*(n)/\lambda(n) = \lim_{n \rightarrow \infty} l(n)/\lambda(n) = 1 \quad (23)$$

证明 对于 2^k 进方法的加法链 (4), 如果删去链中出现两次的任何元素的第二次出现, 它就成为一个星链; 因为如果 a_i 是未在第一行出现但在第二行出现的 $2d_0, 4d_0, \dots$ 诸元素当中的头一个元素, 则我们有 $a_i \leq 2(m-1)$; 因此, 对于头一行中的某个 $a_j, a_i = (m-1) + a_j$ 。总计链的长度, 对于所有 $k \geq 1$, 我们有

$$\lambda(n) \leq l(n) \leq l^*(n) < (1 + 1/k) \lg n + 2^k \quad (24)$$

如果我们选择,比方, $k = \lceil \frac{1}{2} \lg \lambda(n) \rceil$, 则定理得证。 ■

如果我们对于很大的 n , 在(24)中命 $k = \lambda\lambda(n) - 2\lambda\lambda\lambda(n)$, 其中 $\lambda\lambda(n)$ 表示 $\lambda(\lambda(n))$, 则我们得到更强的渐近上限

$$l(n) \leq l^*(n) \leq \lambda(n) + \lambda(n)/\lambda\lambda(n) + O(\lambda(n)\lambda\lambda\lambda(n)/\lambda\lambda(n)^2) \quad (25)$$

第二项 $\lambda(n)/\lambda\lambda(n)$ 实质上是从(24)能得到的最佳项。对于下限的更深入的分析表明 $\lambda(n)/\lambda\lambda(n)$ 这个项实际上在(25)中是必不可少的。为了看出为什么是这样, 让我们考虑以下事实。

定理 E (Paul Erdos, *Acta Arithmetica* (1960), 77~81) 设 ϵ 是一个正实数, 则对于某个小于2的 α 和所有适当大的 m , 使得

$$\lambda(n) = m, \quad r \leq m + (1 - \epsilon)m/\lambda(m) \quad (26)$$

的加法链(11)的数目小于 α^m 。(换言之, 当 m 很大时短到使(26)成立的加法链的数目大大小于使得 $\lambda(n) = m$ 的 n 值的个数。)

证明 我们要估计可能的加法链数, 为此, 我们的第一个目标是对定理 A 进行改进, 以便能更加满意地处理非加倍。

引理 P 设 $\delta < \sqrt{2} - 1$ 是一个固定的正实数。如果加法链的步 i 不是一个加倍, 且对于某个 $j, 0 \leq j < i, a_i < a_j(1 + \delta)^{i-j}$, 则称步 i 为一个“微步”。如果加法链包含 s 个小步和 t 个微步, 则

$$t \leq s/(1 - \theta), \quad \text{其中 } (1 + \delta)^2 = 2^\theta \quad (27)$$

证明 对于每个微步 $i_k, 1 \leq k \leq t$, 及某个 $j_k < i_k$, 我们有 $a_{i_k} < a_{j_k}(1 + \delta)^{i_k - j_k}$ 。设 I_1, \dots, I_t 是区间 $(j_1, i_1], \dots, (j_t, i_t]$, 其中记号 $(j, i]$ 代表使得 $j < k \leq i$ 的所有整数 k 的集合。有可能(见习题 17)找出非重叠的区间 $J_1, \dots, J_h = (j'_1, i'_1], \dots, (j'_h, i'_h]$, 使得

$$\begin{aligned} I_1 \cup \dots \cup I_t &= J_1 \cup \dots \cup J_h \\ a_{i'_k} &< a_{j'_k}(1 + \delta)^{2(i'_k - j'_k)}, \quad \text{对于 } 1 \leq k \leq h \end{aligned} \quad (28)$$

现在对于区间 J_1, \dots, J_h 之外的所有步 i , 我们有 $a_i \leq 2a_{i-1}$; 因此如果令

$$q = (i'_1 - j'_1) + \dots + (i'_h - j'_h)$$

我们有

$$2^{\lambda(n)} \leq n \leq 2^{r-q}(1 + \delta)^{2q} = 2^{\lambda(n)+s-(1-\theta)q} \leq 2^{\lambda(n)+s-(1-\theta)t} \quad \blacksquare$$

转回定理 E 的证明, 让我们选择 $\delta = 2^{1/4} - 1$, 并把每个加法链的 r 个步骤分成三类:

$$t \text{ 个微步, } u \text{ 个加倍, } v \text{ 个其它步, } t + u + v = r \quad (29)$$

以另一方式计数, 我们有 s 个小步, 其中 $s + m = r$ 。由假设, 定理 A, 以及引理 P, 我

们得到关系式

$$s \leq (1 - \epsilon)m/\lambda(m), \quad t + v \leq 3.271s, \quad t \leq s/(1 - \epsilon/2) \quad (30)$$

若给定满足这些条件的 s, t, u, v , 则有

$$\binom{r}{t, u, v} = \binom{r}{t + v} \binom{t + v}{v} \quad (31)$$

种方式来分派这些步到特定的类中。若给定这些步的这样一个分布, 我们来考虑如何能选择非微步: 如果步 i 是(29)中“其它”步之一, 则 $a_i \geq (1 + \delta)a_{i-1}$, 所以 $a_i - a_j + a_k$, 其中 $\delta a_{i-1} \leq a_k \leq a_j \leq a_{i-1}$ 。而且 $a_j \leq a_i/(1 + \delta)^{i-j} \leq 2a_{i-1}/(1 + \delta)^{i-1}$, 所以 $\delta \leq 2/(1 + \delta)^{i-1}$ 。这至多给出对于 j 的 β 种选择, 其中 β 是仅依赖于 δ 的一个常数。对 k 也至多有 β 种选择, 所以对每个非微步指定 j 和 k 的方式数至多是

$$\beta^2 \quad (32)$$

最后, 一旦对于每个非微步已经选择了“ j ”和“ k ”, 则对微步选择 j 和 k 的方式将少于

$$\binom{r^2}{t} \quad (33)$$

我们以少于(33)的方式, 在 $0 \leq k_h \leq j_h < r$ 的范围内选择 t 个不同的下标对 $(j_1, k_1), \dots, (j_t, k_t)$ 。然后对于每个微步 i , 依次地使用一对下标 (j_h, k_h) , 使得

- $j_h < i$;
- 在尚未对较小的诸微步 i 使用过的下标对当中使 $a_{j_h} + a_{k_h}$ 尽量小。
- $a_i = a_{j_h} + a_{k_h}$ 满足微步的定义。

如果这样的 (j_h, k_h) 对不存在, 则我们就得不到加法链; 另一方面, 在指定位置具有微步的任何加法链, 都必须以这些方式之一来进行选择, 所以(33)是可能性的数目的上限。

于是, 满足(26)的可能的加法链的总数是以(31)乘(32)乘(33), 并对于所有相关的 s, t, u 和 v 求和为上限。现在借助于这些函数的一个颇为标准的估计, 即可完成定理 E 的证明(习题 18)。 ■

推论 几乎对于所有的 n , $l(n)$ 的值近似于 $\lambda(n) + \lambda(n)/\lambda\lambda(n)$ 。更精确地说, 有一函数 $f(n)$, 使得当 $n \rightarrow \infty$ 时 $f(n) \rightarrow 0$, 而且

$$\Pr(|l(n) - \lambda(n) - \lambda(n)/\lambda\lambda(n)| \geq f(n)\lambda(n)/\lambda\lambda(n)) = 0 \quad (34)$$

(关于这个概率“Pr”的定义见 3.5 节。)

证明 上限(25)表明, 无绝对值符号时(34)成立。下限由定理 E 得来, 如果我们让 $f(n)$ 足够缓慢地减少成 0, 使得当 $f(n) \leq \epsilon$ 时, 值 N 是如此地大, 以致至多有 ϵN 个值 $n \leq N$ 使得 $l(n) \leq \lambda(n) + (1 - \epsilon)\lambda(n)/\lambda\lambda(n)$ 成立。 ■

*** 星链** 乐观的人们感到假定 $l(n) = l^*(n)$ 是有道理的; 给定具有极小长度 $l(n)$ 的一个加法链, 似乎难以相信我们会找不到满足(外观上适度的)星条件的同样长度的一个加法链。但在 1958 年 Walter Hansen 却证明了一个引人注目的定理,

即对于某些很大的 n 值, $l(n)$ 确定地小于 $l^*(n)$, 他还证明了好几个有关的定理, 我们现在就来研究这几个定理。

Hansen 的定理从研究一个星链的详细结构开始。设 $n = 2^{e_0} + 2^{e_1} + \cdots + 2^{e_r}$, 其中 $e_0 > e_1 > \cdots > e_r \geq 0$, 并设 $1 = a_0 < a_1 < \cdots < a_r = n$ 是 n 的一个星链。如果在这个链中有 d 个加倍, 我们定义辅助序列

$$0 = d_0 \leq d_1 \leq d_2 \leq \cdots \leq d_r = d \quad (35)$$

其中 d_i 是在步 $1, 2, \dots, i$ 当中加倍的数目。我们也定义一个“多重集合”序列 S_0, S_1, \dots, S_r , 它记住这个链中出现的 2 的乘方。(一个多重集合是一个数学实体, 它类似于一个集合, 但可以包含重复的元素; 一个对象可以若干次地作为一个多重集合的元素, 而且它出现的重复次数是有意义的。关于多重集合的一些熟知的例子, 见习题 19。)多重集合 S_i 由下列规则定义:

- a) $S_0 = \{0\}$;
- b) 如果 $a_{i+1} = 2a_i$, 则 $S_{i+1} = S_i + 1 = \{x+1 \mid x \in S_i\}$;
- c) 如果 $a_{i+1} = a_i + a_k, k < i$, 则 $S_{i+1} = S_i \uplus S_k$ 。

(符号 \uplus 意味着把多重集合结合在一起, 增加多重性。)从这个定义, 得出

$$a_i = \sum_{x \in S_i} 2^x \quad (36)$$

这里求和当中的项不必是不同的。特别是

$$n = 2^{e_0} + 2^{e_1} + \cdots + 2^{e_r} = \sum_{x \in S_r} 2^x \quad (37)$$

后一求和中的元素个数至多是 2^f , 其中 $f = r - d$ 是非加倍的个数。

由于(37)中 n 有两个不同的二进表示, 我们可以把多重集合 S_r 划分成多重集合 M_0, M_1, \dots, M_t , 使得

$$2^{e_j} = \sum_{x \in M_j} 2^x, \quad 0 \leq j \leq t \quad (38)$$

这可以通过把 S_r 的元素重新排列成非递降的顺序 $x_1 \leq x_2 \leq \cdots$, 并取 $M_t = \{x_1, x_2, \dots, x_k\}$ 来完成, 其中 $2^{x_1} + \cdots + 2^{x_k} = 2^{e_t}$ 。这必然是可能的, 因为 e_t 是诸 e 当中最小的。类似地, $M_{t-1} = \{x_{k+1}, x_{k+2}, \dots, x_{k'}\}$, 等等, 这一过程在二进记法下极易显现。下面是一个例子。

设 M_j 包含 m_j 个元素(考虑多重性在内), 则 $m_j \leq 2^f - t$, 因为 S_r 至多有 2^f 个元素, 而且它已经划分成 $t+1$ 个非空的多重集合。由等式(38), 我们可以看到

$$e_j \geq x > e_j - m_j, \quad \text{对所有 } x \in M_j \quad (39)$$

通过构造多重集合 M_{ij} (它记录了 M_j 原型的历史), 我们完成了对星链结构的考察。把多重集合 S_i 分划成 $t+1$ 个多重集合如下:

- a) $M_{ij} = M_j$;
- b) 如果 $a_{i+1} = 2a_i$, 则 $M_{ij} = M_{(i+1)j} - 1 = \{x-1 \mid x \in M_{(i+1)j}\}$;

c) 如果 $a_{i+1} = a_i + a_k, k < i$, 则(因为 $S_{i+1} = S_i \cup S_k$)令 $M_{ij} = M_{(i+1)j}$ 减 S_k , 即从 $M_{(i+1)j}$ 删去 S_k 的元素。如果 S_k 的某个元素出现于两个或多个不同的多重集合 $M_{(i+1)j}$ 中, 则我们就从具有最大的 j 值的集合中删去它; 当 i 固定时, 这个规则对每个 j 惟一地确定 M_{ij} 。

从这个定义得出, 对所有 $x \in M_{ij}$,

$$e_j + d_i - d \geq x > e_j + d_i - d - m_j \quad (40)$$

作为这一详细构造的例子, 我们考虑星链 1, 2, 3, 5, 10, 20, 23, 对于它们有 $t = 3, r = 6, d = 3, f = 3$ 。我们得到多重集合的下列阵列:

$(d_0, d_1, \dots, d_6):$	0	1	1	1	2	3	3
$(a_0, a_1, \dots, a_6):$	1	2	3	5	10	20	23
$(M_{03}, M_{13}, \dots, M_{63}):$							0
$(M_{02}, M_{12}, \dots, M_{62}):$							1
$(M_{01}, M_{11}, \dots, M_{61}):$			0	0	1	2	2
$(M_{00}, M_{10}, \dots, M_{60}):$	0	1	1	1	2	3	3
				1	2	3	3
	S_0	S_1	S_2	S_3	S_4	S_5	S_6

$M_3 \quad e_3 = 0, m_3 = 1$
 $M_2 \quad e_2 = 1, m_2 = 1$
 $M_1 \quad e_1 = 2, m_1 = 1$
 $M_0 \quad e_0 = 4, m_0 = 2$

于是 $M_{40} = \{2, 2\}$, 等等。从这个构造, 我们可看出 d_i 是 S_i 的最大元素; 因此

$$d_i \in M_{i0} \quad (41)$$

这个结构的最重要的部分来自等式(40); 它的直接结论之一是

引理 K 如果 M_{ij} 和 M_{uv} 都含有一个公共的整数 x , 则

$$-m_v < (e_j - e_u) - (d_u - d_i) < m_j \quad | \quad (42)$$

尽管引理 K 可能看起来不是极端强有力的, 但它指出, (当 m_j 和 m_v 都相当小以及当 M_{ij} 和 M_{uv} 包含一个公共的元素时) 步 u 与步 i 之间加倍的数目近似地等于指数 e_v 与 e_j 之差。这使加法链具有某种程度的规则性; 而且它提示, 假定诸指数 e_i 离得足够远, 我们有可能证明一个类似于上而定理 B 的结果, 即 $l^*(n) = e_0 + t$ 。下而的定理说明了事实上这一点是如何实现的。

定理 H (W. Hansen, Crelle **202** (1959), 129~136) 设 $n = 2^{e_0} + 2^{e_1} + \dots + 2^{e_t}, e_0 > e_1 > \dots > e_t \geq 0$ 。如果

$$e_0 > 2e_1 + 2.271(t-1) \quad \text{且对于 } 1 \leq i \leq t, \quad e_{i-1} \geq e_i + 2m \quad (43)$$

其中 $m = 2^{\lfloor 3.271(t-1) \rfloor} - t$, 则 $l^*(n) = e_0 + t$ 。

证明 我们可以假定 $t > 2$, 因为当 $t \leq 2$ 时, 如果没有对诸 e 的限制, 这个定理的结果是真的。假设我们有 n 的一个星链 $1 = a_0 < a_1 < \dots < a_r = n$, 且 $r \leq e_0 + t -$

1. 设整数 d, f, d_0, \dots, d_r 和多重集合 M_j 及 S_j 反映了这个链的结构, 如先前定义的那样。由定理 A 的推论, 我们知道 $f \leq \lfloor 3.271(t-1) \rfloor$; 因此 m 的值是关于每个多重集合 M_j 中元素个数 m_j 的真正上限。

在求和

$$a_i = \left(\sum_{x \in M_{i,0}} 2^x \right) + \left(\sum_{x \in M_{i,1}} 2^x \right) + \dots + \left(\sum_{x \in M_{i,r}} 2^x \right)$$

中, 如果我们把这个和式想像成是在二进制系统中进行的, 则从对应于 $M_{i,j}$ 的项到对应于 $M_{i,j-1}$ 的项没有进位传播, 因为诸 e 相距很远 (参考 (40))。特别是, 对于 $j \neq 0$ 的所有项之和将不会进位而影响 $j=0$ 的那些项, 所以我们必定有

$$a_i \geq \sum_{x \in M_{i,0}} 2^x \geq 2^{\lambda(a_i)}, \quad 0 \leq i \leq r \quad (44)$$

为了证明定理 H, 我们将乐于证明, 在某种意义上, n 的 t 个额外的乘方, 必须“逐个置入”, 所以我们要找出一个方法来搞清楚这些项的每一个实际上是在哪一步进入加法链的。

设 j 是 1 和 t 之间的一个数。由于 $M_{0,j}$ 为空而 $M_{r,j} = M_j$ 非空, 我们可以求使得 $M_{i,j}$ 为非空的头一步 i 。

从定义 $M_{i,j}$ 的方式, 我们知道步 i 是一个非加倍: 对某个 $u < i-1$, $a_i = a_{i-1} + a_u$ 。我们还知道 $M_{i,j}$ 的所有元素都是 S_u 的元素。我们将证明同 a_i 相比, a_u 必定是比较小的。

设 x_j 是 $M_{i,j}$ 的一个元素, 则因为 $x_j \in S_u$, 故有某个 v , 使得 $x_j \in M_{uv}$ 。因此得出

$$d_i - d_u > m \quad (45)$$

即, 至少有 $m+1$ 个加倍出现于步 u 与步 i 之间。因为如果 $d_i - d_u \leq m$, 则引理 K 告诉我们 $|e_j - e_v| < 2m$; 因此 $v = j$ 。但这是不可能的, 因为由我们对于步 i 的选择, M_{uj} 为空。

S_u 的所有元素都小于等于 $e_1 + d_i - d$ 。因为如果 $x \in S_u \subseteq S_i$ 和 $x > e_1 + d_i - d$, 则由 (40), $x \in M_{u,0}$ 且 $x \in M_{i,0}$; 所以引理 K 意味着 $|d_i - d_u| < m$, 同 (45) 矛盾。事实上, 这个论证证明了, $M_{i,0}$ 同 S_u 没有公共元素, 所以 $M_{(i-1),0} = M_{i,0}$ 。从 (44) 我们有 $a_{i-1} \geq 2^{\lambda(a_i)}$, 而且因此步 i 是一个小步。

我们现在可以推导出在这整个证明中什么大概是关键的事实: S_u 的所有元素都在 $M_{u,0}$ 中。因为如果不然, 设 x 是 S_u 的一个元素且 $x \notin M_{u,0}$ 。由于 $x \geq 0$, (40) 意味着 $e_1 \geq d - d_u$, 因此

$$e_0 = f + d - s \leq 2.271s + d \leq 2.271(t-1) + e_1 + d_u$$

由假设 (43), 这意味着 $d_u > e_1$ 。但由 (41), $d_u \in S_u$, 而且它不可能在 $M_{i,0}$ 中, 因此 $d_u \leq e_1 + d_i - d \leq e_1$, 矛盾。

回到 $M_{i,j}$ 中的元素 x_j , 我们有 $x_j \in M_{uv}$; 而且我们已经证明 $v=0$ 。因此再次通过方程 (40),

$$e_0 + d_u - d \geq x_j > e_0 + d_u - d - m_0 \quad (46)$$

对于所有 $j = 1, 2, \dots, t$, 我们已经确定满足(46)的一个数 x_j , 以及一个小步 i , 在这步里项 2^e 可以说已经进入加法链中。如果 $j \neq j'$, 则出现这种情况的步 i 不可能对于 j 和 j' 两者都相同; 因为(46)将告诉我们 $|x_j - x_{j'}| < m$, 而 M_{ij} 和 $M_{ij'}$ 的元素之间的差距必然大于 m , 因为 e_j 和 $e_{j'}$ 相距很远。我们被迫作出结论, 这个链至少包含 t 个小步, 但这是一个矛盾。 ▮

定理 F (W. Hansen)

$$l(2^A + xy) \leq A + \nu(x) + \nu(y) - 1, \quad \text{如果 } \lambda(x) + \lambda(y) \leq A \quad (47)$$

证明 把二进方法和因子方法结合起来, 就可以构造一个加法链(一般地它不是星链)。设 $x = 2^{x_1} + \dots + 2^{x_u}$, $y = 2^{y_1} + \dots + 2^{y_v}$, 其中 $x_1 > \dots > x_u \geq 0$, $y_1 > \dots > y_v \geq 0$ 。

这个链的最初一些步形成 2 的逐次乘方, 直至达到 2^{A-y_1} 为止; 在这些步之间, 另外的一些值 $2^{x_u-1} + 2^{x_u}$, $2^{x_u-2} + 2^{x_u-1} + 2^{x_u}$, \dots 以及 x 被插入在适当位置上。在已经形成的直到 $2^{A-y_1} + x(2^{y_1-y_1} + \dots + 2^{y_{i-1}-y_1})$ 的一条链之后, 继续加 x 并把得到的和加倍 $y_i - y_{i+1}$ 次; 这就得出

$$2^{A-y_{i+1}} + x(2^{y_1-y_{i+1}} + \dots + 2^{y_i-y_{i+1}})$$

如果对于 $i = 1, 2, \dots, v$ 已经完成了这个构造, 同时为方便起见假定 $y_{v+1} = 0$, 则如所希望的那样, 我们有 $2^A + xy$ 的一个加法链。 ▮

定理 F 使我们能够求出使 $l(n) < l^*(n)$ 的诸值, 因为定理 H 给出了在某些情况下 $l^*(n)$ 的一个明显的值。例如, 设 $x = 2^{1016} + 1$, $y = 2^{2032} + 1$, 并设

$$n = 2^{6103} + xy = 2^{6103} + 2^{3048} + 2^{2032} - 2^{1016} + 1$$

按照定理 F, 我们有 $l(n) \leq 6106$ 。但对于 $m = 508$, 定理 H 也适用, 而这证明 $l^*(n) = 6107$ 。

大量的计算机计算已经证明, $n = 12509$ 是使得 $l(n) < l^*(n)$ 的最小的值。对于这个 n 值, 没有像序列 $1, 2, 4, 8, 16, 17, 32, 64, 128, 256, 512, 1024, 1041, 2082, 4164, 8328, 8345, 12509$ 这样短的星链。对于 $\nu(n) = 5$ 和 $l(n) \neq l^*(n)$ 最小的 n 是 $16537 = 2^{14} + 9 \cdot 17$ (见习题 15)。

Jan van Leeuwen 推广了定理 H 来证明, 如果 $e_0 > \dots > e_t$ 相距足够远, 则对于所有固定的 $k \geq 1$,

$$l^*(k2^{e_0}) + t \leq l^*(kn) \leq l^*(k2^{e_t}) + e_0 - e_t + t$$

[Crelle 295 (1977), 202~207.]

某些猜测 尽管推测 $l(n) = l^*(n)$ 看起来是合乎道理的, 但我们现在看到这是不成立的。另一个似乎正确的推测[首先由 A. Goulard 提出并由 E. de Jonquières 在 *L'Intermed. des Math.* 2 (1895), 125~126 中声称“证明”]是 $l(2n) = l(n) + 1$; 一个加倍步是如此有效, 因而似乎不可能有比加一个加倍步到 n 的最短链还要短的

任何 $2n$ 的链。但是计算机的计算表明,这一推测也是不成立的,因为 $l(191) = l(382) = 11$ 。(382 的长度为 11 的一个星链不难找到;例如 1, 2, 4, 5, 9, 14, 23, 46, 92, 184, 198, 382。数 191 是使得 $l(n) = 11$ 的极小值,用手工证明 $l(191) > 10$ 似乎非常困难。作者用计算机生成的关于这一事实的证明,使用了将在 7.2.2 小节中概述的一个后溯法,详细考察了 948 种情况。)使得 $l(2n) = l(n)$ 成立的最小 4 个 n 值是 $n = 191, 701, 743, 1111$; E. G. Thurber 在 *Pacific J. Math.* **49** (1973), 229~242 中证明,这些数的第 3 个是这样的 n 的一个无穷系列的一员,即对于所有 $k \geq 5, 23 \cdot 2^k + 7$ 的无穷多个数之一。推测 $l(2n) \geq l(n)$ 似乎是合理的,但甚至这一推测也不成立。Kevin R. Hebb 已经证明,对于所有固定的不是 2 的幂的整数 m , $l(n) - l(mn)$ 可以变成任意大 [Notices Amer. Math. Soc. **21** (1974), A ~ 294]。 $l(mn) < l(n)$ 的最小的情况是 $l((2^{13} + 1)/3) = 15$ 。

设 $c(r)$ 是使得 $l(n) = r$ 的 n 的最小值。对于这些 n 的序列计算 $l(n)$ 似乎是最难的。我们有下列的表:

r	$c(r)$	r	$c(r)$	r	$c(r)$
1	2	10	127	19	18287
2	3	11	191	20	34303
3	5	12	379	21	65131
4	7	13	607	22	110591
5	11	14	1087	23	196591
6	19	15	1903	24	357887
7	29	16	3583	25	685951
8	47	17	6271	26	1176431
9	71	18	11231	27	2211837

对于 $r \leq 11, c(r)$ 的值近似地等于 $c(r-1) + c(r-2)$, 而且这个事实导致若干人推测 $c(r)$ 好像函数 ϕ^r 一样增长;但定理 D 的结果(对于 $n = c(r)$)意味着当 $r \rightarrow \infty$ 时, $r / \lg c(r) \rightarrow 1$ 。这里列出的对于大于 18 的 r 这些值,除了 $c(24)$ 首先是由 Daniel Bleichenbacher 计算的以外,都是由 Achim Flammenkamp 计算的。Flammenkamp 注意到对于 $10 \leq r \leq 27, c(r)$ 可以由公式 $2^r \exp(-\theta r / \lg r)$ 很好地近似,其中 θ 接近 $\ln 2$;这同上限(25)非常一致。鉴于因子方法,很多人一度猜测, $c(r)$ 将总是一个素数;但是 $c(15), c(18)$ 和 $c(21)$ 都可为 11 所整除。或许没有一个关于加法链的猜测是保险的!

列出的 $l(n)$ 值表明,这个函数令人吃惊地光滑;例如,对于在 $1125 \leq n \leq 1148$ 的范围内的所有 $n, l(n) = 13$ 。计算机计算表明,使用公式

$$l(n) = \min(l(n-1) + 1, l_n) - \delta_n \quad (48)$$

可以对所有 $2 \leq n \leq 1000$ 编制 $l(n)$ 表,其中如果 n 是素数,则 $l_n = \infty$, 否则如果 p 是整除 n 的最小素数,则 $l_n = l(p) + l(n/p)$;而且若 n 在表 1 中则 $\delta_n = 1$, 否则

$\delta_n = 0$ 。

设 $d(r)$ 是方程 $l(n) = r$ 的解 n 的个数, 根据 Flammenkamp, 下表列出了这一函数的最初一些值:

r	$d(r)$	r	$d(r)$	r	$d(r)$	r	$d(r)$	r	$d(r)$
1	1	6	15	11	246	16	4490	21	90371
2	2	7	26	12	432	17	8170	22	165432
3	3	8	44	13	772	18	14866	23	303475
4	5	9	78	14	1382	19	27128	24	558275
5	9	10	136	15	2481	20	49544	25	1028508

$d(r)$ 确实必须是 r 的一个递增函数, 但没有明显的方法证明这一似乎简单的断言, 更不必说对于很大的 r 确定 $d(r)$ 的渐近增长了。

关于加法链, 仍然未解决的最有名的问题是 Scholz-Brouer 推测, 它指出

$$l(2^n - 1) \leq n - 1 + l(n) \quad (49)$$

事实上, 计算机的计算表明, 对于 $1 \leq n \leq 24$, (49) 中的等式成立; E. G. Thurber 用手算 [Discrete Math. 16 (1976), 279~289] 证明, 对于 $n = 32$ 等式也成立。关于加法链的许多研究试图证明 (49)。数 $2^n - 1$ 的二进表示中有许多的 1, 其加法链具有特别的重要性, 因为这是对于二进方法最糟的情况。Arnold Scholz 于 1937 年起名“加法链”(用德文) 并且把 (49) 作为一个问题提出来 [Jahresbericht der deutschen Mathematiker-Vereinigung, Abteilung II, 47 (1937), 41~42]; Alfred Brauer 于 1939 年证明

$$l^*(2^n - 1) \leq n - 1 + l^*(n) \quad (50)$$

表 1 特殊加法链的 n 值

23	163	229	319	371	413	453	553	599	645	707	741	813	849	903
43	165	233	323	373	419	455	557	611	659	709	749	825	863	905
59	179	281	347	377	421	457	561	619	667	711	759	835	869	923
77	203	283	349	381	423	479	569	623	669	713	779	837	887	941
83	211	293	355	382	429	503	571	631	677	715	787	839	893	947
107	213	311	359	395	457	509	573	637	683	717	803	841	899	955
149	227	317	367	403	451	551	581	643	691	739	809	845	901	983

Hansen 定理证明, $l(n)$ 可以小于 $l^*(n)$, 所以为了证明或否定 (49) 肯定需要更多的工作。作为这个努力的一步, Hansen 定义了 l^0 链(处于 l 链和 l^* 链“之间”)的概念。在一个 l^0 链中, 某些元素是划底线的; 条件是 $a_i = a_j + a_k$, 其中 a_j 是小于 a_i 的最大划底线元素。

作为一条 l^0 链的例子(肯定不是极小的 l^0 链), 考虑

$$\underline{1}, \underline{2}, \underline{4}, \underline{5}, \underline{8}, \underline{10}, \underline{12}, \underline{18} \quad (51)$$

容易验证每个元素和前边划底线的元素之间的差就在链中。我们命 $l^0(n)$ 表示 n 的 l^0 链的极小长度。显然, $l(n) \leq l^0(n) \leq l^*(n)$ 。

定理 F 中构造的链是一个 l^0 链(见习题 22); 因此对于某个 n , 我们有 $l^0(n) < l^*(n)$ 。不知道是否在所有情况下都有 $l(n) = l^0(n)$; 如果这个等式为真, 则 Scholz-Brauer 推测将被证实, 因为有 Hansen 的另一定理:

定理 G $l^0(2^n - 1) \leq n - 1 + l^0(n)$

证明 设 $1 = a_0, a_1, \dots, a_r = n$ 是 n 的极小长度的 l^0 链, 并设 $1 = b_0, b_1, \dots, b_t = n$ 是划有底线元素的子序列(可以假定 n 是划底线的)。则我们能得到 $2^n - 1$ 的 l^0 链如下:

a) 对于 $0 \leq i \leq r$, 包括 $l^0(n) - 1$ 个数 $2^{a_i} - 1$ 划底线当且仅当 a_i 是划底线的。

b) 对于 $0 \leq j < t$ 和对于 $0 < i \leq b_{j+1} - b_j$, 包括数 $2^i(2^{b_j} - 1)$ 都划底线(这总共是 $b_1 - b_0 + \dots + b_t - b_{t-1} = n - 1$ 个数)。

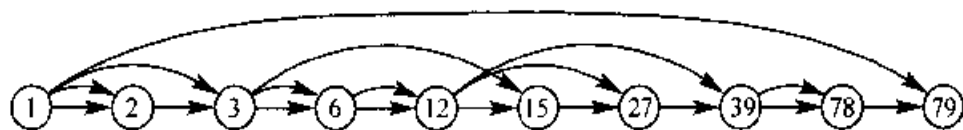
c) 把 a) 和 b) 中的数排序成为递升序列。

我们容易验证, 这给出一条 l^0 链: b) 中的数都等于 a) 或 b) 的某个其它元素的两倍; 而且这个元素是前边的划底线的元素。如果 $a_i = b_j + a_k$, 其中 b_j 是小于 a_i 的划底线元素之最大者, 则 $a_k = a_i - b_j \leq b_{j+1} - b_j$, 所以 $2^{a_k}(2^{b_j} - 1) = 2^{a_i} - 2^{a_k}$ 是链中的划底线元素, 并恰恰在 $2^{a_i} - 1$ 之前。由于 $2^{a_i} - 1 = (2^{a_i} - 2^{a_k}) + (2^{a_k} - 1)$, 其中这两个值都出现于链中, 因而我们得到一条具有 l^0 性质的加法链。 ■

对应于(51)的在证明定理 G 中构造的链是:

1, 2, 3, 6, 12, 15, 30, 31, 60, 120, 240, 255, 510, 1020, 1023, 2040,
4080, 4095, 8160, 16320, 32640, 65280, 130560, 261120, 262143

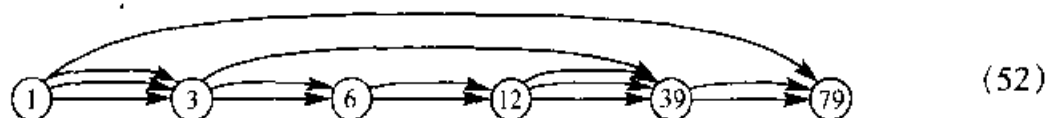
图表示 一个加法链(1)以自然的方式对应于一个有向图, 其中对于 $0 \leq i \leq r$ 顶点被标号为 a_i , 而且我们画从 a_j 到 a_i 和从 a_k 到 a_i 的有向边作为(2)中每步 $a_i = a_j + a_k$ 的表示。例如, 出现于图 15 中的加法链 1, 2, 3, 6, 12, 15, 27, 39, 78, 79 对应于有向图



如果对于一个以上的下标对 (j, k) 有 $a_i = a_j + a_k$, 则在构造有向图时, 我们选择一个确定的 j 和 k 。

一般来说, 除了这样一个有向图的头一个顶点之外, 所有顶点都将恰在两个有向边的头部; 然而这并不是这个图的真正重要的性质, 因为它掩盖了许多不同的加法链实质上可以等价这一事实。如果一个顶点的出口次数是 1, 则它仅用于后边的某一步中, 因此, 后边这一步实质上是三个输入 $a_j + a_k + a_m$ 之和, 它可以计算为 $(a_j + a_k) + a_m$, 或者为 $a_j + (a_k + a_m)$, 或者为 $a_k + (a_j + a_m)$ 。这三种选择是无所谓的, 但是加法链的约定迫使我们要对它们进行区别。通过删去其出口次数为 1 的任何顶点, 并把来自它的前驱的有向边附加到它的后继, 我们就可以避免这样的冗余

性。例如,上边的图将变成



我们也可以删去其出口次数为0的任何顶点,当然最后的顶点 a_r 除外,因为这样一个顶点对应于加法链中无用的步。

这样一来,每个加法链都导致一个约简的有向图,它含有一个“源”顶点(标号1)及一个“槽”顶点(标号 n);除源顶点外,每个顶点的入口次数 ≥ 2 ,除槽顶点外,每个顶点的出口次数 ≥ 2 。反过来,没有有向回路的任何这样的有向图至少对应于一个加法链,因为我们可以对顶点实行拓扑排序并且对于每个入口次数 $d > 0$ 的顶点写下 $d - 1$ 个加法步。通过考察约简的图可以重新构造加法链的长度,无用步除外;它是

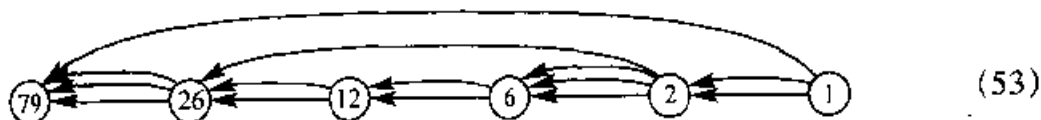
$$(\text{有向边数}) - (\text{顶点数}) + 1 \quad (53)$$

因为删去出口次数为1的一个顶点,也就删去了一条有向边。

我们说两个加法链等价,如果它们有相同的约简有向图。例如,加法链1,2,3,6,12,15,24,39,40,79等价于我们作为出发点的链,因为它也导致(52)。这一例子表明一个非星链可以等价于一个星链。一个加法链等价于一个星链当且仅当它的约简有向图只能以一种方式进行拓扑排序。

这个图表示的一个重要性质已由 N. Pippenger 提出:每个顶点的标号恰好等于从源到该顶点的有向通路的个数。于是,寻找 n 的最优加法链的问题等价于对于一个源顶点和一个槽顶点及从源到槽恰有 n 个有向通路的所有有向图极小化量(53)的问题。

由于有向图的对称性,这个特征有一个惊人的推论。如果我们把所有有向边的方向颠倒过来,则源和槽的作用互换,而且我们得到对应于同一个 n 的加法链集合的另一个有向图。这些加法链和我们作为出发点的链有相同的长度(53)。例如,如果在(52)中把箭头改成从右到左运行,而且按照从右边顶点过来的通路数重新标号这些顶点,我们就得到



对应于这一约简的有向图的星链之一是

$$1, 2, 4, 6, 12, 24, 26, 52, 78, 79$$

我们可以称这个链为原来加法链的对偶。

习题 39 和 40 讨论了这个图表示及对偶性原理的重要推论。

习 题

1. [15] 当算法 A 终止时, Z 的值是什么?

2. [24] 给定整数 n 和 x , 写出算法 A 的一个 MIX 程序, 以计算 $x^n \bmod w$, 这里 w 是字长。假定 MIX 有二进操作 SRB, JAB, 等等, 它们在 4.5.2 小节已作介绍。写出另一个以串行方式(反复地乘以 x)计算 $x^n \bmod w$ 的程序, 并比较这些程序的运行时间。

► 3. [22] 怎样通过(a) 二进方法, (b) 三进方法, (c) 四进方法, (d) 因子方法来计算 x^{975} ?

4. [M20] 找一个数 n , 对于它, 八进(2^3 进)方法要比二进方法少做 10 个乘法。

► 5. [24] 图 14 示出了“幂树”的头八层。这株树的第 $k+1$ 层定义如下, 假设已经构造了头 k 层: 从左到右依次取第 k 层的每个节点 n , 并在它之下(依次)附加节点

$$n+1, n+a_1, n+a_2, \dots, n+a_{k-1} - 2n$$

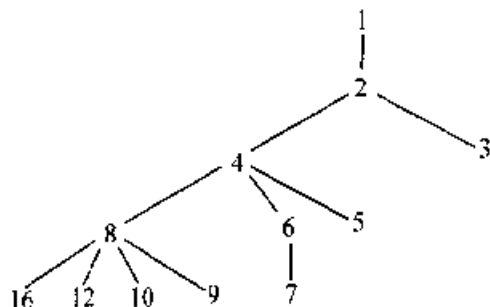
其中 $1, a_1, a_2, \dots, a_{k-1}$ 是从树的根到 n 的通路; 但抛弃任何一个重复已经在树中出现的数的节点。

试设计一个有效的算法, 它构造幂树的头 $r+1$ 层。[提示: 对于 $0 \leq j \leq 2^r$, 利用两组变量 $LINKU[j]$, $LINKR[j]$; 如果 j 是树中的一个数, 则这些点分别地指向上和指向右。]

6. [M26] 如果对习题 5 中给出的幂树的定义稍作改变, 使得 n 下边的节点以递减的顺序

$$n+a_{k-1}, \dots, n+a_2, n+a_1, n+1$$

而不是以递升的顺序附加上来, 则我们得到一株其头 5 层为



的树。试证明这个树给出了计算 x^n 的一个方法, 它要求和二进方法恰巧同样多的乘法; 因此它不像幂树那样好, 尽管它是用几乎相同的方式构造出来的。

7. [M21] 证明有无穷多个 n 的值,

a) 对于它用因子方法比二进方法更好;

b) 对于它用二进方法比因子方法更好;

c) 对于它用幂树方法比二进方法和因子方法都好。

(这里“更好”的方法是指用较少的乘法来计算 x^n 。)

8. [M21] 证明幂树(习题 5)对 x^n 的计算绝不比二进方法用更多的乘法。

► 9. [25] 试设计一个乘幂过程, 它类似于算法 A, 但以进制 $m=2^r$ 为基础。你的方法应当执行近似于 $\lg n + \nu + m$ 次乘法, 其中 ν 是在 n 的 m 进表示中非零数字的个数。

10. [10] 图 15 示出一株树, 它指出对所有 $n \leq 100$, 以最少的乘法次数来计算 x^n 的一个方法。这株树怎样能在仅有 100 个存储单元的一台计算机内方便地表示出来?

► 11. [M26] 图 15 的树描述了对链中所有 i , 有 $l(a_i) = i$ 的加法链 a_0, a_1, \dots, a_r 。当 $n=43$ 和 $n=77$ 时, 试求对于 n 的所有具有这个性质的加法链。试说明图 15 这样的任意树必然包括通

路 1, 2, 4, 8, 9, 17, 34, 43, 77 或通路 1, 2, 4, 8, 9, 17, 34, 68, 77 之一。

12. [M10] 对于所有正整数 n , 是否有可能推广图 15 中示出的树成为一株对于计算 x^n 来说提供极少乘法规则的无穷树?

13. [M21] 对于定理 C 中所列的四种情况的每一种, 找出长度为 $A+2$ 的一个星链(因此定理 C 对于以 l^* 代替 l 也成立)。

14. [M29] 通过揭示(a)步 $r-1$ 不是一个小步以及(b) $\lambda(a_{r-k})$ 不能小于 $m-1$, 来完成定理 C 的证明。

15. [M43] 写出一个计算机程序以推广定理 C, 表征所有使得 $l(n) = \lambda(n) + 3$ 的 n 和所有使得 $l^*(n) = \lambda(n) + 3$ 的 n 。

16. [HM15] 证明定理 D 不是仅仅由于二进方法而显然成立的; 如果 $l^B(n)$ 表示由二进的 S 和 X 方法产生的 n 的加法链长度, 则当 $n \rightarrow \infty$ 时, $l^B(n)/\lambda(n)$ 不趋于一个极限。

17. [M25] 说明怎样找出在引理 P 的证明中所要求的区间 J_1, \dots, J_k 。

18. [HM24] 设 β 是一个正常数, 证明有一个常数 $\alpha < 2$ 使得对所有的大数 m ,

$$\sum \binom{m+1}{s} \binom{t+v}{t} \beta^{2v} \binom{m+1}{t}^2 < \alpha^m$$

其中求和是对满足(30)的所有 s, t, v 进行。

19. [M23] 一个“多重集合”类似于一个集合, 但同一元素在其中可以重复有限次。如果 A 和 B 是多重集合, 我们定义新的多重集合 $A \uplus B, A \cup B, A \cap B$ 如下: 一个在 A 中恰巧出现 a 次和在 B 中恰巧出现 b 次的元素, 在 $A \uplus B$ 中恰巧出现 $a+b$ 次, 在 $A \cup B$ 中恰巧出现 $\max(a, b)$ 次, 在 $A \cap B$ 中恰巧出现 $\min(a, b)$ 次。(一个“集合”是含元素不多于一次的多重集合; 如果 A 和 B 是集合, 则 $A \cup B$ 和 $A \cap B$ 也是, 而且本题中给出的定义同通常集合的并和交的定义一致。)

a) 一个正整数 n 的素因子分解是一个其元素为素数的多重集合 N , 这里 $\prod_{p \in N} p = n$ 。每一个正整数都可以惟一地分解成素数这一事实给了我们一个在正整数和素数的有限多重集合之间的一一对应关系; 例如如果 $n = 2^2 \cdot 3^3 \cdot 17$, 则对应的多重集合是 $N = \{2, 2, 3, 3, 3, 17\}$ 。如果 M 和 N 是对应于 m 和 n 的多重集合, 则什么是对应于 $\gcd(m, n)$, $\text{lcm}(m, n)$ 和 mn 的多重集合?

b) 复数上的每一个首一多项式 $f(z)$ 以自然方式对应于它的“根”的多重集合 F ; 我们有 $f(z) = \prod_{\zeta \in F} (z - \zeta)$ 。如果 $f(z)$ 和 $g(z)$ 是对应于复数的有限多重集合 F 和 G 的多项式, 则对应于 $F \uplus G, F \cup G$ 和 $F \cap G$ 的多项式是什么?

c) 尽你之所能, 找出多重集合之间相对于三个运算 \uplus, \cup 和 \cap 成立的有趣的恒等式。

20. [M20] 在(a)类型 3 的, (b)类型 5 的星链的 Hansen 结构分解中出现的序列 S_i 和 M_{ij} ($0 \leq i \leq r, 0 \leq j \leq t$) 是什么? (6 种“类型”在定理 B 的证明中定义。)

► 21. [M26] (W. Hansen) 设 q 是任何正整数, 试求使得 $l(n) \leq l^*(n) - q$ 的一个 n 值。

22. [M20] 证明在定理 F 的证明中构造的加法链是一个 l^0 链。

23. [M20] 证明 Brauer 不等式(50)。

► 24. [M22] 推广定理 G 的证明, 以证明对于任何整数 $B > 1, l^0((B^n - 1)/(B - 1)) \leq (n - 1)l^0(B) + l^0(n)$; 并且证明 $l(2^{mn} - 1) \leq l(2^m - 1) + mn - m + l^0(n)$ 。

25. [20] 设 y 是一个小数, $0 < y < 1$, 在二进制数系中表示成 $y = (.d_1 \cdots d_k)_2$ 。试设计一个算法, 利用乘法和开平方根的运算来计算 x^y 。

► 26. [M25] 给定很大的整数 n 和 m , 试设计一个有效的算法, 它计算模 m 下的第 n 个斐波那契数 F_n 。

27. [M21] (A. Flammenkamp) 使每个加法链至少包含 6 个小步的最小的 n 是多少?

28. [HM33] (A. Schönhage) 本题的目的是给出 $l(n) \geq \lambda(n) + \lg v(n) -$

$O(\log \log(v(n) + 1))$ 的一个短证明

a) 当 $x = (x_k \cdots x_0, x_{-1} \cdots)_2$ 和 $y = (y_k \cdots y_0, y_{-1} \cdots)_2$ 都是以二进制写出的实数时, 如果对所有 $j, x_j \leq y_j$, 我们就写 $x \sqsubseteq y$. 给出构造具有下列性质的最小 z 的一个简单规则: $x' \sqsubseteq x$ 和 $y' \sqsubseteq y$ 意味着 $x' + y' \sqsubseteq z$. 用 $x \nabla y$ 表示这个数, 证明 $v(x \nabla y) \leq v(x) + v(y)$.

b) 给定使 $r = l(n)$ 的任何加法链 (11), 设序列 d_0, d_1, \dots, d_r 如 (35) 中所定义的那样, 用下列规则定义序列 A_0, A_1, \dots, A_r : $A_0 = 1$; 如果 $a_i = 2a_{i-1}$, 则 $A_i = 2A_{i-1}$; 否则如果对于某些 $0 \leq k \leq j < i, a_i = a_j + a_k$, 则 $A_i = A_{j-1} \nabla (A_{i-1}/2^{d_j - d_k})$. 对于 $0 \leq i \leq r$, 有 $a_i \in A_i$, 在此意义下证明, 这一序列“覆盖”给定的链.

c) 设 δ 是一个正整数 (下面再选择). 如果 $d_j - d_k \geq \delta$, 则称非加倍步 $a_i = a_j + a_k$ 为“婴步”, 否则称它为“闭步”. 设 $B_0 = 1$; 如果 $a_i = 2a_{i-1}$, 则 $B_i = 2B_{i-1}$; 如果 $a_i = a_j + a_k$ 是婴步, 则 $B_i = B_{j-1} \nabla (B_{i-1}/2^{d_j - d_k})$; 否则 $B_i = \rho(2B_{i-1})$, 其中 $\rho(x)$ 是对 $0 \leq e \leq \delta$, 使得 $x/2^e \sqsubseteq y$ 的最小数. 证明对于 $0 \leq i \leq r, A_i \subset B_i$ 且 $v(B_i) \leq (1 + \delta c_i) 2^{b_i}$, 其中 b_i 和 c_i 分别表示小于等于 i 的婴步和闭步的个数. [提示: 证明 B_i 中的诸 1 出现在大小 $\geq 1 + \delta c_i$ 的连续块中.]

d) 我们现在有 $l(n) = r - b_r + c_r + d_r$, 且 $v(n) \leq v(B_r) \leq (1 + \delta c_r) 2^{b_r}$. 说明怎样选择 δ 以便得到本习题开始时指出的不等式. [提示: 见 (16), 并注意对于某个依赖于 δ 的 $\alpha < 1, n \leq 2^{\alpha b_r - c_r}$.]

29. [M49] 对所有正整数 n , 是否有 $v(n) \leq 2^{l(n) - \lambda(n)}$? (如果是, 我们有下限 $l(2^n - 1) \geq n - 1 + \lceil \lg n \rceil$; 参考 (17) 和 (49).)

30. [20] 一个加减法链以规则 $a_i = a_j \pm a_k$ 代替 (2); 正文中介绍的想像的计算机有一个新的操作码 SUB (这在实际中对应于用乘和除法来计算 x^n). 对于某个 n , 找出一个加减法链, 它少于 $l(n)$ 步.

31. [M46] (D. H. Lehmer) 给定一个小的正“权”, 试剖折加法链 (1) 中极小化 $q + (r - q)$ 的问题, 这里 q 是“简单”步的个数, 在这些步当中 $a_i = a_{i-1} + 1$. (如果乘以 x 的乘法比一般乘法更为简单, 则这个问题就更接近于计算 x^n 的许多实际应用. 参见 4.6.2 小节中的应用.)

32. [M30] (A. C. Yao, F. F. Yao 和 R. L. Graham) 把“开销” a_i, a_k 同加法链 (1) 中的每一步 $a_i = a_j + a_k$ 关联起来. 证明自左至右的二进方法对于所有正整数 n 产生极小总开销的一个链.

33. [15] 有多少长度为 9 的加法链以 (52) 作为它们的约简有向图?

34. [M23] 当 $e_0 > \dots > e_r \geq 0$ 时, $n = 2^{e_0} + \dots + 2^{e_r}$ 的加法链是 $1, 2, \dots, 2^{e_0 - e_1}, 2^{e_0 - e_1} + 1, \dots, 2^{e_0 - e_2} + 2^{e_1 - e_2}, 2^{e_0 - e_2} + 2^{e_1 - e_2} + 1, \dots, n$. 这对应于本节开始处介绍的 S 和 X 方法, 而算法 A 对应于把两个序列 $(1, 2, 4, \dots, 2^{e_0})$ 和 $(2^{e_1 - 1} + 2^{e_1}, 2^{e_1 - 2} + 2^{e_1 - 1} + 2^{e_1}, \dots, n)$ 排序成递升序列所得到的加法链. 证明或否定: 这些加法链每一个都是另一个的对偶.

35. [M27] 当 $e_0 > e_1 + 1$ 时, 没有无用步的加法链有多少等价于习题 34 中所讨论的加法链?

▶ 36. [25] (E. G. Straus) 找出至多用 $2\lambda(\max(n_1, n_2, \dots, n_m)) + 2^m - m - 1$ 次乘法来计算一般单项式 $x_1^{n_1} x_2^{n_2} \cdots x_m^{n_m}$ 的一个方法.

37. [HM30] (A. C. Yao) 设 $l(n_1, \dots, n_m)$ 是含有 m 个给定的数 $n_1 < \dots < n_m$ 的最短加法链的长度. 证明 $l(n_1, \dots, n_m) \leq \lambda(n_m) + m\lambda(n_m)/\lambda\lambda(n_m) + O(\lambda(n_m)\lambda\lambda\lambda(n_m)/\lambda\lambda(n_m)^2)$, 由此推广 (25).

38. [M47] 在习题 37 的记法下, 当 $m \rightarrow \infty$ 时, $l(1, 4, 9, \dots, m^2) - m$ 的渐近值是多少?

▶ 39. [M25] (J. Olivos, 1979) 设 $l([n_1, n_2, \dots, n_m])$ 是在习题 36 的意义下计算单项式 $x_1^{n_1} x_2^{n_2} \cdots x_m^{n_m}$ 所需要的极小乘法次数, 其中每个 n_i 是正整数. 通过证明 $l([n_1, n_2, \dots, n_m]) = l(n_1,$

$n_2, \dots, n_m) + m - 1$ 来证明这个问题等价于习题 37 中的问题。[提示:通过考虑有一个以上源顶点的图来推广有向图的构造。]

► 40. [M21] (J. Olivos) 推广因子方法和定理 F, 证明

$$l(m_1 n_1 + \dots + m_t n_t) \leq l(m_1, \dots, m_t) + l(n_1, \dots, n_t) + t - 1$$

其中 $l(n_1, \dots, n_t)$ 在习题 37 中定义。

41. [M40] (P. Downey, B. Leong, R. Sethi) 设 G 是具有 n 个顶点 $|1, \dots, n|$ 和 m 个边的一个连通图, 其中对于 $1 \leq j \leq m$ 这些边连接 u_j 到 v_j 。证明对于所有充分大的 $A, l(1, 2, \dots, 2^{A u_1}, 2^{A u_1} + 2^{A v_1} + 1, \dots, 2^{A u_m} + 2^{A v_m} + 1) = An + m + k$, 其中 k 是 G 的一个顶点覆盖 (即对于 $1 \leq j \leq m$ 或包含 u_j 或包含 v_j 的一个集合) 中的极小顶点数。

42. [M50] 对于所有正整数 n , 是否 $l(2^n - 1) \leq n - 1 + l(n)$? 等式总是成立吗? $l(n) - l^0(n)$ 吗?

4.6.4 多项式求值

我们已经知道了计算特殊多项式 x^n 的有效方法, 现在让我们考虑对于给定的 x 值, 计算一个 n 次多项式

$$u(x) = u_n x^n + u_{n-1} x^{n-1} + \dots + u_1 x + u_0, \quad u_n \neq 0 \quad (1)$$

的问题。这个问题在实践中经常出现。

在下列讨论中, 我们将集中研究把用计算机计算多项式所需要的操作次数极小化的问题, 并大胆地假定所有的算术运算都是精确的。多项式通常是用浮点算术求值的, 浮点运算是不精确的, 而且对于不同的求值方案一般地将给出不同的答案。所达到的精确度的数值分析依赖于所考虑的具体多项式的系数, 而这超出了本书的范围; 读者应该注意观察以浮点算术进行的任何计算的精确度。从数值的观点看, 我们将介绍的方法在大多数情况下已被证明是相当令人满意的, 但也可以给出许多坏的例子。[关于多项式快速计算的稳定性的著作的综述, 以及关于对某些高速算法来说不能保证某种数值稳定性的说明, 见 Webb Miller, *SICOMP* 4(1975), 105 ~ 107.]

在一小节, 我们将认为变量 x 就是一个数。但是重要的是要牢记, 当变量是多精度数, 多项式或矩阵这样的大对象时, 我们将讨论的大多数方法也将正确。在这样的情况下, 有效的公式导致甚至更大的报偿, 特别是当我们能减少乘法的次数时。

程序员新手通常直接以教科书上给出的形式计算多项式 (1): 首先计算 $u_n x^n$, 然后计算 $u_{n-1} x^{n-1}, \dots, u_1 x$, 最后把 (1) 的所有项加在一起。但即使使用 4.6.3 小节的有效办法来计算这种方法下的 x 的乘方, 所得到的计算不用说也是缓慢的, 除非几乎所有的系数 u_k 都为零。如果这些系数都非零, 另一种显然的办法是从右到左地计算 (1), 对于 $k = 1, \dots, n$ 计算 x^k 和 $u_k x^k + \dots + u_0$ 的值。这样一个过程含 $2n - 1$ 个乘法和 n 个加法, 它还需要一些指令来存储和从内存中检索中间结果。

霍纳规则 通常教给程序员新手的最初的技术之一是重新安排这个计算的一

个巧妙方法,即用下边的方法计算 $u(x)$:

$$u(x) = (\cdots(u_n x + u_{n-1})x + \cdots)x + u_0 \quad (2)$$

从 u_n 开始乘以 x , 加 u_{n-1} , 乘以 x , \cdots , 乘以 x , 加 u_0 。这个计算形式通常称做“霍纳规则”;我们已经看到它用于 4.4 节的进制转换中。整个过程要求 n 个乘法和 n 个加法,对于每个为零的系数减少一个加法。而且,没有必要存储中间结果,因为在计算期间出现的每个结果一旦被算出就立即投入使用。

W. G. Horner (霍纳) 在 19 世纪初为计算多项式的根而给出了这一规则 [*Philosophical Transactions, Royal Society of London* **109** (1819), 308~335], 这一方法如此命名[见 J. L. Coolidge, *Mathematics of Great Amateurs* (Oxford, 1949), 第 15 章]的原因是霍纳的名字已与(2)联系在一起;但是实际上 Isaac Newton (牛顿)早在 150 年前就利用了同一思想。例如,在最初写于 1669 年,题为 *De Analysi per Aequationes Infinitas* 的一篇有名的著作中,牛顿把多项式 $y^4 - 4y^3 + 5y^2 - 12y + 17$ 写为

$$\overline{y - 4 \times y: + 5 \times y: - 12 \times y: + 17}$$

同时说明后来被称为牛顿求根方法的思想。这显然说明(2)的思想,因为他通常使用水平线和冒号代替括有向边来表示分组。在未发表的笔记中牛顿已经使用这个思想许多年了。[见 *The Mathematical Papers of Isaac Newton*, D. T. Whiteside 编, **1** (1967), 490, 531; **2** (1968), 222.]独立地,一个等价于霍纳方法的方法事实上在 13 世纪时在中国就被秦九韶使用了[见 Y. Mikami, *The Development of Mathematics in China and Japan* (1913), 73~77]。

已经提出了对霍纳规则的好几个推广。我们首先考虑 z 是一个复数而系数 u_k 是实数时计算 $u(z)$ 的方法。特别是,当 $z = e^{i\theta} = \cos \theta + i \sin \theta$ 时,多项式 $u(z)$ 实质上是两个傅里叶级数

$$(u_0 + u_1 \cos \theta + \cdots + u_n \cos n\theta) + i(u_1 \sin \theta + \cdots + u_n \sin n\theta)$$

显然,复数加法和乘法可以归结为对实数的一系列运算:

实 + 复 要求 1 个加法
 复 + 复 要求 2 个加法
 实 \times 复 要求 2 个乘法
 复 \times 复 要求 4 个乘法, 2 个加法
 或 3 个乘法, 5 个加法

(见习题 41。这里减法被认为和加法等价。)因此当 $z = x + iy$ 为复数时,霍纳规则(2)使用 $4n - 2$ 个乘法和 $3n - 2$ 个加法,或者 $3n - 1$ 个乘法和 $6n - 5$ 个加法来计算 $u(z)$ 。实际上这些加法中的 $2n - 4$ 个可以节省,因为我们每次乘的是同一个数 z 。计算 $u(x + iy)$ 的另一个过程是设

$$\begin{aligned} a_1 &= u_n, & b_1 &= u_{n-1}, & r &= x + x, & s &= x^2 + y^2 \\ a_j &= b_{j-1} + ra_{j-1}, & b_j &= u_{n-j} - sa_{j-1}, & & & & 1 < j \leq n \end{aligned} \quad (3)$$

然后用归纳法容易证明 $u(z) = za_n + b$ 。这个方案[BIT 5(1965), 142; 也见 G. Goertzel, AMM 65 (1958), 34~35]只要求 $2n+2$ 个乘法和 $2n+1$ 个加法, 所以当 $n \geq 3$ 时它是对于霍纳规则的改进。在傅里叶级数的情况下, 当 $z = e^{i\theta}$ 时, 我们有 $s=1$, 所以乘法的次数降到 $n+1$ 。这个事实的寓意在于一名好的程序员不应无选择地使用高级程序设计语言内部的“复数算术”特性。

考虑用 $x - x_0$ 除多项式 $u(x)$ 的过程, 并使用算法 4.6.1D 来得到 $u(x) = (x - x_0) \cdot q(x) + r(x)$; 这里 $\deg(r) < 1$, 所以 $r(x)$ 是一个与 x 无关的常数, 而且 $u(x_0) = 0 \cdot q(x_0) + r = r$ 。对这个加法过程的考察揭示, 这个计算实质上与计算 $u(x_0)$ 的霍纳规则相同。类似地, 如果我们以多项式 $(z - z_0)(z - \bar{z}_0) = z^2 - 2x_0z + x_0^2 + y_0^2$ 除 $u(z)$, 则得到的计算结果等价于(3); 我们得到 $u(z) = (z - z_0)(z - \bar{z}_0)q(z) + a_nz + b_n$, 因此 $u(z_0) = a_nz_0 + b_n$ 。

一般地说, 如果以 $f(x)$ 除 $u(x)$, 以得到 $u(x) = f(x)q(x) + r(x)$, 则只要 $f(x_0) = 0$, 就有 $u(x_0) = r(x_0)$; 这个发现导致了霍纳规则的进一步推广。例如, 我们可以设 $f(x) = x^2 - x_0^2$; 这就产生了“第二阶”霍纳规则

$$u(x) = (\cdots(u_{2\lfloor n/2 \rfloor}x^2 + u_{2\lfloor n/2 \rfloor - 2})x^2 + \cdots)x^2 + u_0 + ((\cdots(u_{2\lfloor n/2 \rfloor - 1}x^2 + u_{2\lfloor n/2 \rfloor - 3})x^2 + \cdots)x^2 + u_1)x \quad (4)$$

第二阶规则使用 $n+1$ 个乘法和 n 个加法(见习题 5); 所以从这个观点来看它不是对霍纳规则的改进。但至少在两种情况下(4)是有用的: 如果我们要计算 $u(x)$ 和 $u(-x)$ 两者, 则这个方法只须多做一次加法运算就产生出 $u(-x)$; 两个值几乎可以像一个值一样方便地得到。而且, 如果我们有允许并行计算的一台计算机, 则(4)的两行即可独立地求值, 所以我们节省大约一半的运行时间。

当计算机允许一次在 k 个运算器上并行计算时, 可以使用一个“第 k 阶”霍纳规则(由 $f(x) = x^k - x_0^k$ 以类似方式得到)。进行并行计算的另一项有吸引力的方法已由 G. Estrin 提出[Proc. Western Joint Computing Conf. 17 (1960), 33~40]; 当 $n=7$ 时, Estrin 的方法是:

处理器 1	处理器 2	处理器 3	处理器 4	处理器 5
$a_1 = u_7x + u_6$	$b_1 = u_5x + u_4$	$c_1 = u_3x + u_2$	$d_1 = u_1x + u_0$	x^2
$a_2 = a_1x^2 + b_1$		$c_2 = c_1x^2 + d_1$		x^4
$a_3 = a_2x^4 + c_2$				

这里 $a_3 = u(x)$ 。然而, W. S. Dorn 的一个有趣的分析[IBM J. Res. and Devel. 6 (1962), 239~245]表明, 如果每个算术单元都必须访问一个一次只同一个处理器进行通信的存储器的话, 则这些方法通常并不是对第二阶规则的改进。

多项式值造表 如果我们希望在一个算术级数的多处地方计算 n 次多项式(即如果我们要计算 $u(x_0), u(x_0+h), u(x_0+2h), \cdots$), 在开头少数几步之后, 这个过程可以归结为仅做加法。因为如果我们以任何数列 $(\alpha_0, \alpha_1, \cdots, \alpha_n)$ 开始并应用变换

$$\alpha_0 \leftarrow \alpha_0 + \alpha_1, \alpha_1 \leftarrow \alpha_1 + \alpha_2, \dots, \alpha_{n-1} \leftarrow \alpha_{n-1} + \alpha_n \quad (5)$$

我们发现 k 次应用(5)产生

$$\alpha_j^{(k)} = \binom{k}{0} \beta_j + \binom{k}{1} \beta_{j+1} + \binom{k}{2} \beta_{j+2} + \dots, \quad 0 \leq j \leq n$$

其中 β_j 表示 α_j 的初始值且对于 $j > n, \beta_j = 0$ 。特别是

$$\alpha_0^{(k)} = \binom{k}{0} \beta_0 + \binom{k}{1} \beta_1 + \dots + \binom{k}{n} \beta_n \quad (6)$$

是 k 的 n 次多项式。如习题 7 中所示,适当地选择诸 β ,我们可以把事情安排成,对所有的 k ,量 $\alpha_0^{(k)}$ 是所需的值 $u(x_0 + kh)$ 。换句话说,(5)中 n 个加法的每次执行将产生给定多项式的下一个值。

警告:在多次重复(5)之后舍入误差会累积,因而 α_j 的一个误差产生了在所计算多项式中系数 x^0, \dots, x^j 的对应误差。因此,在大量迭代之后诸 α 的值应当“刷新”。

变量的导数和变化 有时我们希望在给定一个常数 x_0 和 $u(x)$ 的系数的情况下求 $u(x - x_0)$ 的系数。例如,如果 $u(x) = 3x^2 + 2x - 1$,则 $u(x - 2) = 3x^2 - 10x + 7$ 。这类似于把基数 x 转换成基数 $x + 2$ 的进制转换问题。由泰勒定理,通过在 $x = x_0$ 对 $u(x)$ 求导数,即得所求的系数,这就是

$$u(x + x_0) = u(x_0) + u'(x_0)x + (u''(x_0)/2!)x^2 + \dots + (u^{(n)}(x_0)/n!)x^n \quad (7)$$

所以问题就等价于求 $u(x)$ 和它的所有导数。

如果我们写 $u(x) = q(x)(x - x_0) + r$,则 $u(x + x_0) = q(x + x_0)x + r$;所以 r 是 $u(x + x_0)$ 的常系数,而且问题归结为求 $q(x + x_0)$ 的系数,其中 $q(x)$ 是已知的 $n - 1$ 多项次式于是有下列算法:

H1. 对 $0 \leq j \leq n$ 置 $v_j \leftarrow u_j$ 。

H2. 对于 $k = 0, 1, \dots, n - 1$ (在此顺序下),对于 $j = n - 1, \dots, k + 1, k$ (在此顺序下)置 $v_j \leftarrow v_j + x_0 v_{j+1}$ 。 ▮

在 H2 步的结尾我们有 $u(x + x_0) = v_n x^n + \dots + v_1 x + v_0$ 。这个过程是霍纳求根方法的主要部分,而且当 $k = 0$ 时它恰是计算 $u(x_0)$ 的规则(2)。

霍纳方法要求 $(n^2 + n)/2$ 次乘法和 $(n^2 + n)/2$ 次加法;但请注意,如果 $x_0 = 1$,就可以避免所有的乘法。幸运的是,我们通过引入比较少的乘法和除法就可以把一般问题归结为 $x_0 = 1$ 的情况。

S1. 计算并存储值 x_0^2, \dots, x_0^n 。

S2. 对于 $0 \leq j \leq n$ 置 $v_j \leftarrow u_j x_0^j$ 。(现在 $v(x) = u(x_0 x)$)

S3. 执行 H2 步,但取 $x_0 = 1$ 。(现在 $v(x) = u(x_0(x + 1)) = u(x_0 x + x_0)$)

S4. 对于 $0 < j \leq n$ 置 $v_j \leftarrow v_j / x_0^j$ 。(现在 $v(x) = u(x + x_0)$ 如所求。) ▮

这个思想是 M. Shaw 和 J. F. Traub 给出的 [JACM 21(1974), 161~167], 和霍纳方法有同样多次的加法及同样的数值稳定性, 但它仅仅需要 $2n-1$ 次乘法和 $n-1$ 次除法, 因为 $v_n = u_n$ 。这些乘法中大约有 $\frac{1}{2}n$ 个又可以避免 (见习题 6)。

如果我们只要最初一些或最后一些导数, Shaw 和 Traub 已经发现有进一步的方法来节省时间。例如, 如果只要计算 $u(x)$ 和 $u'(x)$, 我们可以用 $2n-1$ 次加法和大约 $n + \sqrt{2n}$ 次乘除法做这一工作如下:

- D1. 计算和存储值 $x^2, x^3, \dots, x^t, x^{2t}$, 其中 $t = \lceil \sqrt{n/2} \rceil$ 。
 D2. 对于 $0 \leq j \leq n$ 置 $v_j \leftarrow u_j x^{f(j)}$, 其中对于 $0 \leq j < n$, $f(j) = t-1 - ((n-1-j) \bmod 2t)$ 且 $f(n) = t$ 。
 D3. 对于 $j = n-1, \dots, 1, 0$, 置 $v_j \leftarrow v_j - v_{j+1} x^{g(j)}$; 这里当 $n-1-j$ 是 $2t$ 的倍数时 $g(j) = 2t$, 否则 $g(j) = 0$ 且不需要乘以 $x^{g(j)}$ 。
 D4. 对 $j = n-1, \dots, 2, 1$, 置 $v_j \leftarrow v_j + v_{j+1} x^{g(j)}$ 。现在 $v_0/x^{f(0)} = u(x)$ 且 $v_1/x^{f(1)} = u'(x)$ 。 ▮

系数的改写 现在转到我们原来的问题, 即对于“随机”的 x 值, 尽可能快速地计算一个给定的多项式 $u(x)$ 。这个问题的重要性部分地是由于像 $\sin x, \cos x, e^x$ 等等标准函数通常是用子程序计算的, 而这些子程序依赖于某些多项式的计算。这种多项式求值是如此经常地出现, 因而我们希望找出最快的方法来进行这一计算。

如果我们首先对系数 u_0, u_1, \dots, u_n 进行“改写”或“加些条件”, 则五次或五次以上的任意多项式可以用少于霍纳规则所要求的操作次数来求值。如以下所说明的那样, 这个改写过程可能包含大量的工作; 但是它不会白做, 因为它只须做一次就够了, 而多项式的求值要做多次。关于标准函数的“改写后的”多项式的例子, 见 V. Y. Pan, USSR Computational Math. and Math. Physics 2 (1963), 137~146。

系数的改写有助于多项式计算的最简单的情形是一个四次多项式

$$u(x) = u_4 x^4 + u_3 x^3 + u_2 x^2 + u_1 x + u_0, \quad u_4 \neq 0 \quad (8)$$

这个等式可以改写成为最先由 T. S. Motzkin 提议的一种形式

$$y = (x + \alpha_0)x + \alpha_1, \quad u(x) = ((y + x + \alpha_2)y + \alpha_3)\alpha_4 \quad (9)$$

其中 $\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4$ 是适当地“改写”的系数。这一方案中的计算包含三次乘法, 五次加法, 及 (在如 MIX 这样的单累加器机器上的) 一条存中间结果 y 到临时存储器中去的指令。同霍纳规则作比较, 我们已经用一个加法及一个可能的存储指令换下了一个乘法。如果这个多项式要经常求值, 则上述比较小的节省也是值得的。(当然, 如果乘法的时间同加法时间相等, (9) 就没有什么改进; 我们能看到一般的四次多项式总是至少要求八次算术运算来求它的值。)

等置 (8) 与 (9) 的系数, 我们得到借助于诸 u_k 计算诸 α_j 的公式:

$$\alpha_0 = \frac{1}{2}(u_3/u_4 - 1), \quad \beta = u_2/u_4 - \alpha_0(\alpha_0 + 1), \quad \alpha_1 = u_1/u_4 - \alpha_0\beta$$

$$\alpha_2 = \beta - 2\alpha_1, \quad \alpha_3 = u_0/u_4 - \alpha_1(\alpha_1 + \alpha_2), \quad \alpha_4 = u_4 \quad (10)$$

一个类似的方案(见于习题 18)在和(9)同样多步内计算一个四次多项式;该法在某些情况下将给出比(9)更大的数值精度,尽管在其它情况下它产生较差的精度。

在实践中通常出现的多项式的前导系数都比较小,使得在(10)中除以 u_4 导致不稳定性。在这种情况下,首先用 $u_4|^{1/4}x$ 代替 x 通常是可取的,这样做可把(8)归结为其前导系数为 ± 1 的一个多项式。一个类似的变换可以应用于更高次的多项式。这一思想是 C. T. Fike 给出的[CACM 10 (1967), 175~178],他给出了好些个有趣的例子。

使用规则 $u(x) = U(x)x + u_0$, 其中利用(9)中的方法计算 $U(x) = u_5x^4 + u_4x^3 + u_3x^2 + u_2x + u_1$, 则任何五次多项式都可以用四次乘法,六次加法及一次存储来求值。或者,如果计算的形式是

$$y = (x + \alpha_0)^2, \quad u(x) = (((y + \alpha_1)y + \alpha_2)(x + \alpha_3) + \alpha_4)\alpha_5 \quad (11)$$

则我们可以用四次乘法,五次加法及三次存储来进行计算。为了确定这些 α , 需要解一个三次方程(见习题 19)。

在许多计算机上,(11)所要求的存储操作的次数小于 3。例如,我们能够计算 $(x + \alpha_0)^2$ 而无需存储 $x + \alpha_0$ 。事实上,现今大多数计算机有一个以上的算术寄存器用于浮点计算,所以我们可以完全避免存储。由于在不同的计算机上算术运算广泛多样的特性,因此在本小节后半部我们将仅仅计算算术运算的次数,而不涉及由累加器到存储器及由存储器到累加器的操作。计算方案通常可以以简捷的方式加以改编,以适应任何具体的计算机,并只需要很少的辅助操作;另一方面,必须记住,这种额外的开销也可能使得我们节省一两个乘法这一件事变得没有多少意义,特别是如果机器代码是由未予以优化的编译程序产生的时。

使用

$$\begin{aligned} z &= (x + \alpha_0)x + \alpha_1, & w &= (x + \alpha_2)z + \alpha_3 \\ u(x) &= ((w + z + \alpha_4)w + \alpha_5)\alpha_6 \end{aligned} \quad (12)$$

的方案,用四次乘法和七次加法,通常就能计算一个六次多项式 $u(x) = u_6x^6 + \dots + u_1x + u_0$ 。[见 D. E. Knuth, CACM 5 (1962), 595~599。]这就节省了霍纳规则所要求的六次乘法中的两次。这里我们又要解一个三次方程:由于 $\alpha_6 = u_6$, 我们可以假定 $u_6 = 1$ 。在此假定下,令 $\beta_1 = (u_5 - 1)/2$, $\beta_2 = u_4 - \beta_1(\beta_1 + 1)$, $\beta_3 = u_3 - \beta_1\beta_2$, $\beta_4 = \beta_1 - \beta_2$, $\beta_5 = u_2 - \beta_1\beta_3$ 。令 β_6 是三次方程

$$2y^3 + (2\beta_4 - \beta_2 + 1)y^2 + (2\beta_5 - \beta_2\beta_4 - \beta_3)y + (u_1 - \beta_2\beta_5) = 0 \quad (13)$$

的一个实根(这个方程总有一个实根,因为对于很大的正 y , 左边的多项式趋于 $+\infty$, 而对于很大的负 y 它趋于 $-\infty$; 因此它必定在这之间某处取 0 值)。现在如果定义

$$\beta_7 = \beta_6^2 + \beta_4\beta_6 + \beta_5, \quad \beta_8 = \beta_3 - \beta_6 - \beta_7$$

则我们最后有

$$\begin{aligned} \alpha_0 &= \beta_2 - 2\beta_6, & \alpha_2 &= \beta_1 - \alpha_0, & \alpha_1 &= \beta_6 - \alpha_0\alpha_2 \\ \alpha_3 &= \beta_7 - \alpha_1\alpha_2, & \alpha_4 &= \beta_8 - \beta_7 - \alpha_1, & \alpha_5 &= u_0 - \beta_7\beta_8 \end{aligned} \quad (14)$$

我们可以编造一个例子来说明这一过程:假设要计算 $x^6 + 13x^5 + 49x^4 + 33x^3 - 61x^2 - 37x + 3$, 我们得到 $\alpha_6 = 1, \beta_1 = 6, \beta_2 = 7, \beta_3 = -9, \beta_4 = -1, \beta_5 = -7$, 因此有三次方程

$$2y^3 - 8y^2 + 2y + 12 = 0 \quad (15)$$

这个方程有 $\beta_6 = 2$ 作为一个根, 我们接着求得 $\beta_7 = -5, \beta_8 = -6, \alpha_0 = 3, \alpha_2 = 3, \alpha_1 = -7, \alpha_3 = 16, \alpha_4 = 6, \alpha_5 = -27$ 。因此得到的方案是

$$z = (x + 3)x - 7, \quad w = (x + 3)z + 16, \quad u(x) = (w + z + 6)w - 27$$

正巧, 量 $x + 3$ 在这里出现两次, 所以我们应该找到了一个方法, 它使用三次乘法和六次加法。

V. Y. Pan 提出了处理六次方程的另一个方法 [Problemy Kibernetiki 5(1961), 17~29]。他的方法要多做一次加法运算, 但在预备阶段它只包含有理运算, 且不需要解三次方程。可以如下进行:

$$\begin{aligned} z &= (x + \alpha_0)x + \alpha_1, & w &= z + x + \alpha_2 \\ u(x) &= (((z - x + \alpha_3)w + \alpha_4)z + \alpha_5)\alpha_6 \end{aligned} \quad (16)$$

为确定诸 α , 我们再次用 $u_6 = \alpha_6$ 除多项式, 使得 $u(x)$ 成为首一的。于是可以验证 $\alpha_0 = u_5/3$ 和

$$\alpha_1 = (u_1 - \alpha_0 u_2 + \alpha_0^2 u_3 - \alpha_0^3 u_4 + 2\alpha_0^5)/(u_3 - 2\alpha_0 u_4 + 5\alpha_0^3) \quad (17)$$

注意, Pan 的方法要求 (17) 中的分母不为 0。换句话说, 仅当

$$27u_3u_6^2 - 18u_6u_5u_4 + 5u_3^3 \neq 0 \quad (18)$$

时 (16) 才可使用。事实上, 这个量不能太小, 否则 α_1 就会变得太大。一旦确定了 α_1 , 剩下的诸 α 就可以从下列方程确定:

$$\begin{aligned} \beta_1 &= 2\alpha_0, & \beta_2 &= u_4 - \alpha_0\beta_1 - \alpha_1 \\ \beta_3 &= u_3 - \alpha_0\beta_2 - \alpha_1\beta_1, & \beta_4 &= u_2 - \alpha_0\beta_3 - \alpha_1\beta_2 \\ \alpha_3 &= \frac{1}{2}(\beta_3 - (\alpha_0 - 1)\beta_2 + (\alpha_0 - 1)(\alpha_0^2 - 1)) - \alpha_1 \\ \alpha_2 &= \beta_2 - (\alpha_0^2 - 1) - \alpha_3 - 2\alpha_1, & \alpha_4 &= \beta_4 - (\alpha_2 + \alpha_1)(\alpha_3 + \alpha_1) \\ \alpha_5 &= u_0 - \alpha_1\beta_4 \end{aligned} \quad (19)$$

我们已经详细地讨论了次数 $n = 4, 5, 6$ 的情况, 因为 n 的这些值在应用中最常出现。现在我们考虑 n 次多项式的一般计算方案, 这是包含至多 $\lfloor n/2 \rfloor + 2$ 次乘法和 n 次加法的一个方法。

定理 E 具有实系数的每一 n 次多项式, $n \geq 3$, 可用下列方案进行计算:

$$\begin{aligned} y &= x + c, \quad w = y^2; \quad z = \begin{cases} (u_n y + \alpha_0)y + \beta_0, & n \text{ 为偶数} \\ u_n y + \beta_0, & n \text{ 为奇数} \end{cases} \\ u(x) &= (\cdots((z(w - \alpha_1) + \beta_1)(w - \alpha_2) + \beta_2)\cdots)(w - \alpha_m) + \beta_m \end{aligned} \quad (20)$$

c, α_k 和 β_k 为适当选择的实参数, $m = \lceil n/2 \rceil - 1$ 。事实上, 有可能选择这些常数使得 $\beta_m \rightarrow 0$ 。

证明 我们首先考虑当 c 固定时, 可以在(20)中选定诸 α 和诸 β 的情况, 设

$$p(x) = u(x - c) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \quad (21)$$

我们要来证明对于某个多项式 $p_1(x)$ 及某常数 α_m, β_m , $p(x)$ 有 $p_1(x)(x^2 - \alpha_m) + \beta_m$ 的形式。如果以 $x^2 - \alpha_m$ 来除 $p(x)$, 我们可以看到仅当由 $p(x)$ 的每个奇数号系数形成的辅助多项式

$$q(x) = a_{2m+1}x^m + a_{2m-1}x^{m-1} + \cdots + a_1 \quad (22)$$

是 $x - \alpha_m$ 的倍数时, 剩余 β_m 才是一个常数。反之, 如果 $q(x)$ 有 $x - \alpha_m$ 作为一个因子, 则对于可以由除法确定的某个常数 β_m , $p(x) = p_1(x)(x^2 - \alpha_m) + \beta_m$ 。

类似地, 我们要 $p_1(x)$ 具有 $p_2(x)(x^2 - \alpha_{m-1}) + \beta_{m-1}$ 的形式, 而这等于说 $q(x)/(x - \alpha_m)$ 是 $x - \alpha_{m-1}$ 的倍数; 这是因为, 如果 $q_1(x)$ 是对应于 $p_1(x)$ 的多项式, 如同 $q(x)$ 对应于 $p(x)$ 那样, 我们就有 $q_1(x) = q(x)/(x - \alpha_m)$ 。依此类推, 我们发现当且仅当

$$q(x) = a_{2m+1}(x - \alpha_1) \cdots (x - \alpha_m) \quad (23)$$

时参数 $\alpha_1, \beta_1, \cdots, \alpha_m, \beta_m$ 才存在。换句话说, 或者 $q(x)$ 恒等于 0 (这仅当 n 是偶数时才会发生), 或者 $q(x)$ 是所有根为实根的 m 次多项式。

现在有一个令人惊奇的由 J. Eve 发现的事实 [Numer. Math. 6 (1964), 17~21]: 如果 $p(x)$ 至少有 $n-1$ 个其实部全为非负或全为非正的复根, 则相应的多项式 $q(x)$ 恒等于零或根全为实根 (见习题 23)。由于当且仅当 $p(x+c)=0$ 时 $u(x)=0$, 故我们只需要把参数 c 选择得足够大以使 $u(x)=0$ 的至少 $n-1$ 个根有大于等于 $-c$ 的实部即可, 此时只要 $a_{n-1} = u_{n-1} - nc u_n \neq 0$, (20) 将必定成立。

我们也可以把 c 确定成使得这些条件都满足而且使 $\beta_m = 0$ 。首先确定 $u(x) = 0$ 的 n 个根。如果 $a + bi$ 是有最大或最小实部的一个根, 且 $b \neq 0$, 设 $c = -a$ 和 $\alpha_m = -b^2$; 则 $x^2 - \alpha_m$ 是 $u(x - c)$ 的一个因子。如果有最小或最大实部的根是实数, 但有第二个最小 (或第二个最大) 实部的根是非实数, 则可应用同样的变换。如果具有最小 (或最大) 实部的两个根都是实数, 则它们可分别表达成 $a - b$ 和 $a + b$ 的形式。设 $c = -a$ 及 $\alpha_m = b^2$ 。再一次 $x^2 - \alpha_m$ 是 $u(x - c)$ 的一个因子。(c 的其它值通常也是可能的, 见习题 24。) 除非 $q(x)$ 恒等于 0, 否则系数 a_{n-1} 将至少对于这些情况之一为非 0。 ■

注意, 这个证明方法通常至少给出两个 c 值, 而且我们也有机会以 $(m-1)!$ 种方式排列 $\alpha_1, \cdots, \alpha_{m-1}$ 。在各种可能的方式中, 某些方式可能比其它一些给出更合要求的数值精度。

当然, 当我们对整数模 m 而不是对实数进行处理时, 不出现数值精度的问题。当 m 同 $2u_4$ 互素时方案 (9) 对于 $n=4$ 有效, 而当 m 同 $6u_6$ 及 (17) 的分母互素时,

方案(16)对 $n=6$ 有效。习题 44 表明对于任何首一 n 次多项式模任何的 $m, n/2 + O(\log n)$ 个乘法和 $O(n)$ 个加法就足够了。

*** 多项式链** 现在我们考虑最优性的问题。就最少的算术运算次数来说,计算各次多项式可能的最好方案是什么呢?在不允许对系数预先改写的情况下,这个问题首先是由 A. M. Ostrowski 分析的[*Studies in Mathematics and Mechanics Presented to R. von Mises* (New York: Academic Press, 1954), 40~48];在改写系数的情况下,由 T. S. Motzkin 作了分析[见 *Bull. Amer. Math. Soc.* **61** (1955), 163]。

为了研究这个问题,我们可以推广 4.6.3 小节的加法链概念成为多项式链。令多项式链是形如

$$x = \lambda_0, \quad \lambda_1, \quad \dots, \quad \lambda_r = u(x) \quad (24)$$

的序列,其中 $u(x)$ 是 x 的某个多项式,而且对于 $1 \leq i \leq r$,

$$\begin{aligned} \text{或者 } \lambda_i &= (\pm \lambda_j) \circ \lambda_k, & 0 \leq j, k < i \\ \text{或者 } \lambda_i &= \alpha_j \circ \lambda_k, & 0 \leq k < i \end{aligned} \quad (25)$$

这里“ \circ ”表示三个运算“+”,“-”或“ \times ”之一,而 α_j 表示一个所谓参数。头一种类型的步称为链步,而第二种类型的步称为参数步。我们假定在每种参数步中使用不同的参数 α_j ;如果有 s 个参数步,则它们应当依次包含 $\alpha_1, \alpha_2, \dots, \alpha_s$ 。

由此得出,在链的末端的多项式 $u(x)$ 有形式

$$u(x) = q_n x^n + \dots + q_1 x + q_0 \quad (26)$$

其中 q_n, \dots, q_1, q_0 是具有整系数的 $\alpha_1, \alpha_2, \dots, \alpha_s$ 的多项式。我们将把数 $\alpha_1, \alpha_2, \dots, \alpha_s$ 解释为实数,因此我们限于考虑具有实系数的多项式的计算。一个多项式链的结果集合 R 定义为当 $\alpha_1, \alpha_2, \dots, \alpha_s$ 独立地取所有可能的实数值时出现的所有实向量 (q_n, \dots, q_1, q_0) 的集合。

如果对于 $t+1$ 个不同整数 $j_0, \dots, j_t \in \{0, 1, \dots, n\}$ 的每种选择,有一个整系数非 0 多变量多项式 $f_{j_0 \dots j_t}$,使得对于 R 中所有的 (q_n, \dots, q_1, q_0) , $f_{j_0 \dots j_t}(q_{j_0}, \dots, q_{j_t}) = 0$,我们就说结果集合 R 至多有 t 个自由度,而且链(24)至多有 t 个自由度。我们也说,如果 (u_n, \dots, u_1, u_0) 在 R 中,则链(24)计算一个给定的多项式 $u(x) = u_n x^n + \dots + u_1 x + u_0$ 。由此得出,至多有 n 个自由度的一个多项式链不能计算所有的 n 次多项式(见习题 27)。

作为多项式链的一个例子,考虑当 n 为奇数时,对应于定理 E 的链

$$\begin{aligned} \lambda_0 &= x \\ \lambda_1 &= \alpha_1 + \lambda_0 \\ \lambda_2 &= \lambda_1 \times \lambda_1 \\ \lambda_3 &= \alpha_2 \times \lambda_1 \\ \left. \begin{aligned} \lambda_{1+3i} &= \alpha_{1+2i} + \lambda_{3i} \\ \lambda_{2+3i} &= \alpha_{2+2i} + \lambda_2 \\ \lambda_{3+3i} &= \lambda_{1+3i} \times \lambda_{2+3i} \end{aligned} \right\} & 1 \leq i < n/2 \end{aligned} \quad (27)$$

有 $\lfloor n/2 \rfloor + 2$ 次乘法和 n 次加法, $\lfloor n/2 \rfloor + 1$ 个链步和 $n + 1$ 个参数步。由定理 E, 结果集合 R 包括所有 $u_n \neq 0$ 的 (u_n, \dots, u_1, u_0) 的集合, 所以 (27) 计算所有 n 次多项式。我们不能证明 R 顶多有 n 个自由度, 因为结果集合有 $n + 1$ 个独立的分量。

有 s 个参数步的多项式链顶多有 s 个自由度。在某种意义上, 这是显然的: 我们不能用少于 t 个任意参数计算有 t 个自由度的一个函数。但这个直观的事实不容易形式地证明; 例如, 有把实线映射到一个平面的连续函数 (“填满空间的曲线”), 这样的函数把一个参数映射成两个独立的参数。为了我们的目的, 我们需要验证任何整系数的多项式函数都不能有这样一个性质; 一个证明见习题 28。

给定这一事实, 即可着手证明我们寻求的结果。

定理 M (T. S. Motzkin, 1954) 具有 $m > 0$ 个乘法的一个多项式链顶多有 $2m$ 个自由度。

证明 设 $\mu_1, \mu_2, \dots, \mu_m$ 是链中代表乘法运算的诸 λ_i , 则

$$\mu_i = S_{2i-1} \times S_{2i}, \quad 1 \leq i \leq m \quad (28)$$

$$\text{且} \quad u(x) = S_{2m+1}$$

其中, 每个 S_j 是诸 μ , 诸 x 及诸 α 的某个和。令 $S_j = T_j + \beta_j$, 其中 T_j 是诸 μ 和诸 x 的一个和, 而 β_j 是诸 α 的一个和。

现在 $u(x)$ 可表达为具有整系数的 $x, \beta_1, \dots, \beta_{2m+1}$ 的一个多项式。由于诸 β 可表达为 $\alpha_1, \dots, \alpha_s$ 的线性函数, 故由所有实数值 $\beta_1, \dots, \beta_{2m+1}$ 表示的值的集合包含链的结果集合。因此至多有 $2m + 1$ 个自由度; 习题 30 表明, 当 $m > 0$ 时这可改进成 $2m$ 。 ■

定理 M 的证明中提到的构造的一个例子见习题 25。对于加法可证明类似的结果:

定理 A (E. G. Belaga, 1958) 一个含 q 个加法和减法的多项式链至多有 $q + 1$ 个自由度。

证明 [Problemy Kibernetiki 5 (1961), 7~15] 设 $\kappa_1, \dots, \kappa_q$ 是链中对应于加法或减法运算的诸 λ_i , 则

$$\kappa_i = \pm T_{2i-1} \pm T_{2i}, \quad 1 \leq i \leq q \quad (29)$$

$$\text{且} \quad u(x) = T_{2q+1}$$

其中每个 T_j 是诸 κ , x , 及诸 α 的乘积。我们可以写 $T_j = A_j B_j$, 其中 A_j 是诸 α 的乘积, B_j 是诸 κ 和诸 x 的乘积。现在对 $i = 1, 2, \dots, q$ 可以对链作下列变换: 设 $\beta_i = A_{2i}/A_{2i-1}$, 使得 $\kappa_i = A_{2i-1}(\pm B_{2i-1} \pm \beta_i B_{2i})$ 。然后把 κ_i 改变成为 $\pm B_{2i-1} \pm \beta_i B_{2i}$, 并用 $A_{2i-1}\kappa_i$ 代入将来的公式 $T_{2i+1}, T_{2i+2}, \dots, T_{2q+1}$ 中出现的每个 κ_i (这个替换可以改变 $A_{2i+1}, A_{2i+2}, \dots, A_{2q+1}$ 的值)。

在对于所有 i 完成了上述变换后, 设 $\beta_{q+1} = A_{2q+1}$; 则 $u(x)$ 可以表达成具有整

系数的 $\beta_1, \dots, \beta_{q+1}$ 和 x 的多项式。我们几乎已经完成了大部分的证明,但我们必须小心,因为当 $\beta_1, \dots, \beta_{q+1}$ 跑遍所有实数值时,所得到的多项式不可能包括由原始链所表示的所有多项式(见习题 26);对于诸 α 的某些值,有可能有 $A_{2i-1} = 0$,而这使 β_i 无定义。

为了完成证明,我们注意到原来的链的结果集合 R 可以写成 $R = R_1 \cup R_2 \cup \dots \cup R_q \cup R'$, 其中 R_i 是当 $A_{2i-1} = 0$ 时可能的结果向量的集合,而 R' 是当诸 α 非 0 时可能的结果向量的集合。上边的讨论证明, R' 至多有 $q+1$ 个自由度。如果 $A_{2i-1} = 0$, 则 $T_{2i-1} = 0$, 所以加法步 κ_i 可以撤销,以得到计算结果集合 R_i 的另一个链。由归纳法我们看到,每个 R_i 至多有 q 个自由度。因此由习题 29, R 至多有 $q+1$ 个自由度。 ─

定理 C 如果对于某个 $n \geq 2$, 一个多项式链 (24) 计算所有 n 次多项式 $u(x) = u_n x^n + \dots + u_0$, 则它至少包括 $\lfloor n/2 \rfloor + 1$ 个乘法和 n 个加减法。

证明 设有 m 个乘法步。由定理 M, 这个链至多有 $2m$ 个自由度, 所以 $2m \geq n+1$ 。类似地, 由定理 A, 有大于等于 n 个加减法。 ─

这个定理指出, 没有单个方法能用少于 $\lfloor n/2 \rfloor + 1$ 次乘法或少于 n 个加法而计算出所有 n 次多项式。习题 29 的结果使我们能加强这个结论, 并指出, 没有有穷的这种多项式链的汇集能满足给定次数的所有多项式。当然, 有些特殊的多项式可以更有效地计算。我们实际上已经证明的是, 具有代数无关的系数(在它们不满足非平凡多项式方程这一意义下)的多项式, 需要有 $\lfloor n/2 \rfloor + 1$ 次乘法和 n 次加法。可惜的是, 在计算机中, 我们处理的系数总是有理数, 所以上述定理实际上不适用; 事实上, 习题 42 表明, $O(\sqrt{n})$ 次乘法(以及可能还有大量加法)总是够用了。从实用的观点看, 定理 C 的上限可应用于“几乎所有”的系数, 而且似乎它们可应用于所有合理的计算方案。此外, 甚至在有理数的情况下, 也有可能得到对应于定理 C 的那些下界: 通过加强上述证明, V. Strassen 指出, 比如说, 多项式

$$u(x) = \sum_{k=0}^n 2^{2^{kn^3}} x^k \quad (30)$$

不能用任何长度 $< n^2 / \lg n$ 的多项式链来计算, 除非这个链至少有 $\frac{1}{2}n - 2$ 个乘法和 $n - 4$ 个除法 [SICOMP 3 (1974), 128~149]。(30) 中的系数非常大; 但也有可能找到其系数仅为 0 和 1 的多项式, 使得计算它们的每一多项式链对所有充分大的 n 至少含 $\sqrt{n}/(4 \lg n)$ 个链乘法, 甚至当允许参数 α_j 是任意复数时也是如此。[见 R. J. Lipton, SICOMP 7 (1978), 61~69; C.-P. Schnorr, Lecture Notes in Comp. Sci. 53 (1977), 135~147。] Jean-Paul van de Wiele 已经证明, 对于某个 $c > 0$, 某个 0-1 多项式的计算总共要求至少 $cn / \log n$ 次算术运算 [FOCS 19 (1978), 159~165]。

除了平凡的 $n-2$ 的情况外, 在定理 C 的下限和已知可达到的实际运算次数之间仍然存在着距离。定理 E 给出 $\lfloor n/2 \rfloor + 2$ 次乘法, 而不是 $\lfloor n/2 \rfloor + 1$ 次, 尽管它确

实达到了极小加法次数。对于 $n=4$ 和 $n=6$, 我们的特殊方法有极小的乘法次数, 但多了一次加法。当 n 是奇数时, 不难证明, 对于乘法和加法, 定理 C 的下限不能同时达到; 见习题 33。对于 $n=3, 5$ 和 7 , 有可能证明至少有 $\lfloor n/2 \rfloor + 2$ 次乘法是必需的。习题 35 和习题 36 证明, 当 $n=4$ 或 $n=6$ 时, 定理 C 的下限都不能实现; 因此当 $n < 8$ 时, 我们已经讨论的方法是最好的。当 n 为偶数时, Motzkin 证明 $\lfloor n/2 \rfloor + 1$ 次乘法是充分的, 但他的构造包含不确定的加法个数 (见习题 39)。V. Y. Pan 建立了对于 $n=8$ 的一个最优方案, 他证明了对于这一情况, 当有 $\lfloor n/2 \rfloor + 1$ 个乘法时, $n+1$ 个加法是必要和充分的。他也证明了对于所有偶数 $n \geq 10$, $\lfloor n/2 \rfloor + 1$ 次乘法和 $n+2$ 次加法是充分的。Pan 的文章 [STOC 10 (1978), 162~172] 也确定了对于所有次数 n , 当计算整个地是对复数而不是对实数进行时, 所需要的乘法和加法的精确极小次数。习题 40 讨论了当 $n \geq 9$ 的奇数值出现时的有趣情况。

显然, 我们对单变量多项式的链已经得到的结果可以毫不困难地扩充到多变量多项式。例如, 如果在不改写系数的情况下要寻找多项式计算的最优方案, 则我们可以把 $u(x)$ 看做 $n+2$ 个变量 x, u_n, \dots, u_1, u_0 的一个多项式; 习题 38 证明, 在这种情况下 n 次乘法和 n 次加法是必要的。其实, A. Borodin [Theory of Machines and Computations, Z. Kohavi 和 A. Paz 编 (New York: Academic Press, 1971), 45~58] 已经证明, 霍纳规则 (2) 实际上是在 $2n$ 次运算中计算 $u(x)$ 而不需要预先处理的惟一方式。

稍作修改, 上述方法即可推广到含有除法的链, 即推广到多项式以及有理函数。奇怪的是, 如果乘法和除法速度相等, 甚至当允许预先处理时, 从计算运算次数的观点看, 霍纳规则的连分数模拟现在看来也是最优的 (见习题 37)。

有时, 在多项式的计算期间除法是有帮助的, 尽管多项式仅仅借助于乘法和加法来定义。我们已经在多项式导数的 Shaw-Traub 算法中见到了一些例子。另一个例子是

$$x^n + \dots + x + 1$$

由于这个多项式可以写成 $(x^{n+1} - 1)/(x - 1)$, 故我们可以通过 $l(n+1)$ 次乘法 (见 4.6.3 小节), 两个减法及一个除法来计算它, 而避免做除法的技术似乎要求大约三倍的运算 (见习题 43)。

特殊的多变量多项式 一个 $n \times n$ 矩阵的行列式可以认为是 n^2 个变量 x_{ij} 的一个多项式, $1 \leq i, j \leq n$ 。如果 $x_{11} \neq 0$, 则有

$$\det \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ x_{31} & x_{32} & \cdots & x_{3n} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{pmatrix} = x_{11} \det \begin{pmatrix} x_{22} - (x_{21}/x_{11})x_{12} \cdots x_{2n} - (x_{21}/x_{11})x_{1n} \\ x_{32} - (x_{31}/x_{11})x_{12} \cdots x_{3n} - (x_{31}/x_{11})x_{1n} \\ \vdots \\ x_{n2} - (x_{n1}/x_{11})x_{12} \cdots x_{nn} - (x_{n1}/x_{11})x_{1n} \end{pmatrix} \quad (31)$$

因此, 一个 $n \times n$ 矩阵的行列式可以通过计算一个 $(n-1) \times (n-1)$ 矩阵的行列式

再加上 $(n-1)^2 + 1$ 次乘法, $(n-1)^2$ 次加法, 及 $n-1$ 次除法来计算。由于一个 2×2 的行列式可以用两次乘法和一次加法来计算, 我们看到, 几乎所有矩阵 (即不需要除以 0 的除法的那些矩阵) 的行列式可以用顶多 $(2n^3 - 3n^2 + 7n - 6)/6$ 次乘法, $(2n^3 - 3n^2 + n)/6$ 次加法, 以及 $(n^2 - n - 2)/2$ 次除法来计算。

当出现 0 时, 行列式甚至更易于计算。例如, 如果 $x_{11} = 0$ 但 $x_{21} \neq 0$, 则有

$$\det \begin{bmatrix} 0 & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ x_{31} & x_{32} & \cdots & x_{3n} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & & x_{nn} \end{bmatrix} = x_{21} \det \begin{bmatrix} x_{12} & \cdots & x_{1n} \\ x_{32} - (x_{31}/x_{21})x_{22} & \cdots & x_{3n} - (x_{31}/x_{21})x_{2n} \\ \vdots & & \vdots \\ x_{n2} - (x_{n1}/x_{21})x_{22} & \cdots & x_{nn} - (x_{n1}/x_{21})x_{2n} \end{bmatrix} \quad (32)$$

这里约简为 $(n-1) \times (n-1)$ 行列式节省了在 (31) 中所用的 $n-1$ 次乘法和 $n-1$ 次加法, 补偿了为认记这个情况所需要的额外的簿记操作。因此任何行列式都可以用大约 $\frac{2}{3}n^3$ 次算术运算 (包括除法) 计算。这是值得注意的, 因为它是有 $n!$ 个项的一个多项式而且每项有 n 个变量。

如果我们要计算一个整元素矩阵行列式, (31) 和 (32) 的过程显然是没有吸引力的, 因为它要求有理算术。然而, 我们可以使用这个方法来计算行列式模 p (对任何素数 p), 因为除法模 p 是可能的 (习题 4.5.2-16)。如果对于充分多的素数进行了这一计算, 则这个行列式的精确值可以如同在 4.3.2 小节中所说明的那样求得, 因为阿达马不等式 4.6.1 - (25) 给出了这个量的上限。

一个 $n \times n$ 矩阵 X 的特征多项式 $\det(xI - X)$ 的系数也可在 $O(n^3)$ 步内计算; 见 J. H. Wilkinson, *The Algebraic Eigenvalue Problem* (Oxford: Clarendon Press, 1965), 353~355, 410~411。习题 70 讨论了一个涉及 $O(n^4)$ 步的有趣的不用除法的方法。

一个矩阵的积和式是非常类似于行列式的一个多项式, 惟一的差别是它的所有非零系数是 +1。于是我们有

$$\text{per} \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & & \vdots \\ x_{n1} & \cdots & x_{nn} \end{bmatrix} = \sum x_{1j_1} x_{2j_2} \cdots x_{nj_n} \quad (33)$$

其中求和是对 $\{1, 2, \dots, n\}$ 的所有排列 $j_1 j_2 \cdots j_n$ 进行的。这个函数似乎应当比看来更复杂的行列式要容易计算, 但却没有已知的办法来像行列式那样有效地计算它。习题 9 和习题 10 证明, 对于很大的 n , 大大地少于 $n!$ 次运算就足够了, 但是所有已知的方法的执行时间仍然随矩阵的大小以指数增长。事实上, Leslie G. Valiant 已经证明, 如果我们忽略计算的运行时间的多项式因素, 则计算一个给定的 0-1 矩阵的积和式, 就同计算一个不确定的多项式时间图灵机所接受计算的个数一样困难。因此, 对于积和式的一个多项式时间的求值算法将意味着, 大量不能用有效解法求

解的其它著名的问题将可在多项式时间内求解。另一方面, Valiant 证明, 对所有 $k \geq 2$, 一个 $n \times n$ 的整数矩阵的积和式可以在 $O(n^{4k-3})$ 步内在模 2^k 下进行计算 [见 *Theoretical Comp. Sci.* 8 (1979), 189~201]。

涉及矩阵的另一个基本运算当然是矩阵乘法; 如果 $X = (x_{ij})$ 是一个 $m \times n$ 矩阵, $Y = (y_{jk})$ 是一个 $n \times s$ 矩阵, 且 $Z = (z_{ik})$ 是一个 $m \times s$ 矩阵, 则公式 $Z = XY$ 意味着

$$z_{ik} = \sum_{j=1}^n x_{ij} y_{jk}, \quad 1 \leq i \leq m, \quad 1 \leq k \leq s \quad (34)$$

这个等式可以认为是 $mn + ns$ 个变量的 ms 个联立多项式的计算, 每个多项式是两个 n 位向量的“内积”。一个直接的计算将含 mns 个乘法和 $ms(n-1)$ 个加法; 但 S. Winograd 在 1967 年发现, 有一个方法来把大约一半的乘法换成加法:

$$\begin{aligned} z_{ik} &= \sum_{1 \leq j \leq n/2} (x_{i,2j} + y_{2j-1,k})(x_{i,2j-1} + y_{2j,k}) - a_i - b_k + x_{in} y_{nk} \quad [n \text{ 为奇数}] \\ a_i &= \sum_{1 \leq j \leq n/2} x_{i,2j} x_{i,2j-1}; \quad b_k = \sum_{1 \leq j \leq n/2} y_{2j-1,k} y_{2j,k} \end{aligned} \quad (35)$$

这个方案使用 $\lceil n/2 \rceil ms + \lfloor n/2 \rfloor (m+s)$ 次乘法和 $(n+2)ms + (\lfloor n/2 \rfloor - 1)(ms + m + s)$ 次加法或减法; 总的运算次数稍微增加了, 但乘法的次数大约已经折半 [见 *IEEE Trans. C-17* (1968), 693~694]。Winograd 的令人惊奇的构造导致许多人更仔细地考虑矩阵乘法的问题, 而且激起广泛的猜测, 即为了进行 $n \times n$ 矩阵乘法, $n^3/2$ 次乘法将是必需的, 因为对于一个变量的多项式已知有类似的下限成立。

Volker Strassen 于 1968 年发现了对于很大的 n 的一个甚至更好的方案。他找到一个方法, 可以只用 7 次乘法而不必像在 (35) 中那样依赖于乘法的可交换性, 来计算 2×2 矩阵的乘积。由于 $2n \times 2n$ 矩阵可以划分成四个 $n \times n$ 矩阵, 利用他的思想可以只用 7^k 次乘法而不是 $(2^k)^3 = 8^k$ 次乘法来递归地得到 $2^k \times 2^k$ 矩阵的积。加法的次数也以 7^k 的阶增长。Strassen 原来的 2×2 恒等式 [Numer. Math. 13 (1969), 354~356] 使用 7 次乘法和 18 次加法; S. Winograd 后来发现了下列更经济的公式:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix} = \begin{pmatrix} aA + bB & w + v + (a + b - c - d)D \\ w + u + d(B + C - A - D) & w + u + v \end{pmatrix} \quad (36)$$

其中 $u = (c - a)(C - D)$, $v = (c + d)(C - A)$, $w = aA + (c + d - a)(A + D - C)$ 。如果适当地保留中间结果, 则 (36) 含 7 次乘法和仅 15 次加法; 对 k 用归纳法, 我们可以用 7^k 次乘法和 $5(7^k - 4^k)$ 次加法来乘 $2^k \times 2^k$ 矩阵。因此为乘 $n \times n$ 矩阵所需要的运算总数已经从 n^3 阶减少成 $O(n^{\lg 7}) = O(n^{2.8074})$ 。类似的减少也可应用于计算行列式和矩阵的逆; 见 J. R. Bunch 和 J. E. Hopcroft, *Math. Comp.* 28 (1974), 231~236。

Strassen 的指数 $\lg 7$ 在 1978 年之前一直得不到改进, 直到 Viktor Pan 发现它可以降低到 $\log_{70} 143640 \approx 2.795$ (见习题 60) 为止。这个新的突破导致了对这个问题

的进一步深入的分析,而且 D. Bini, M. Capovani, D. Coppersmith, G. Lotti, F. Romani, A. Schönhage, V. Pan 以及 S. Winograd 共同的努力最终得到了渐近运行时间方面的显著减少。习题 60~67 讨论了用来确立这些上限的一些有趣的技术。特别是习题 66 包含了 $O(n^{2.55})$ 个运算就足够的一个相当简单的证明。到 1997 年已知的最好上限是 $O(n^{2.376})$, 它是由 Coppersmith 和 Winograd 给出的 [J. Symbolic Comp. 9 (1990), 251~280]。与此相对照,当前知道的最好下限是 $2n^2 - 1$ (见习题 12)。

这些理论的结果是十分引人注目的,但是从一种实用的观点看,它们的用途有限,因为 n 必须很大才能抵消额外的簿记代价。Richard Brent [斯坦福计算机科学报告 CS157 (1970 年 3 月), 也见 Numer. Math. 16 (1970), 145~156] 发现,仅当 $n \geq 40$ 时, Winograd 的方案 (35) 的精细实现,连同为保持数值稳定性的适当比例因子,才变得比通常的方法更好,而且当 $n = 100$ 时,它只节省大约 7% 的运行时间。对于复数算术情况稍有不同,当 $n > 20$ 时 (35) 就变得有利了,而且当 $n = 100$ 时节省 18% 的运行时间。他估计,在 $n \approx 250$ 之前, Strassen 的方案还不会超过 (35); 而且这样庞大的矩阵,在实践中很少出现,除非它们非常稀疏,而这时可使用其它技术。而且,阶为 n^ω , 其中 $\omega < 2.7$ 的已知方法有相当大的比例常数,使得在它们战胜方案 (36) 之前,需要 10^{23} 次以上的乘法。

相反,我们下边将讨论的方法非常实际并已发现有广泛的用途。在分别具有 m_1, \dots, m_n 个元素的区域上, n 个变量的复值函数 F 的一个离散傅里叶变换 f , 由下列等式定义: 对于 $0 \leq s_1 < m_1, \dots, 0 \leq s_n < m_n$,

$$f(s_1, \dots, s_n) = \sum_{\substack{0 \leq t_1 < m_1 \\ \vdots \\ 0 \leq t_n < m_n}} \exp\left(2\pi i \left(\frac{s_1 t_1}{m_1} + \dots + \frac{s_n t_n}{m_n}\right)\right) F(t_1, \dots, t_n) \quad (37)$$

“变换”这个名字是正当的,因为如同习题 13 中所示,我们可以由值 $f(s_1, \dots, s_n)$ 恢复值 $F(t_1, \dots, t_n)$ 。在所有 $m_j = 2$ 的重要特殊情况下,对于 $0 \leq s_1, \dots, s_n \leq 1$, 我们有

$$f(s_1, \dots, s_n) = \sum_{0 \leq t_1, \dots, t_n \leq 1} (-1)^{s_1 t_1 + \dots + s_n t_n} F(t_1, \dots, t_n) \quad (38)$$

而这可以认为是 2^n 个变量 $F(t_1, \dots, t_n)$ 的 2^n 个线性多项式的联立求值。F. Yates 给出的一个著名技术 [The Design and Analysis of Factorial Experiments (Harpenden: Imperial Bureau of Soil Sciences, 1937)] 可以用来把隐含于 (38) 中的加法数从 $2^n(2^n - 1)$ 减少成 $n2^n$ 。考虑 $n = 3$ 的情况可以理解 Yates 的方法: 设 $x_{t_1 t_2 t_3} = F(t_1, t_2, t_3)$ 。

给定	第一步	第二步	第三步
x_{000}	$x_{000} + x_{001}$	$x_{000} + x_{001} + x_{010} + x_{011}$	$x_{000} + x_{001} + x_{010} + x_{011} + x_{100} + x_{101} + x_{110} + x_{111}$

x_{001}	$x_{010} + x_{011}$	$x_{100} + x_{101} + x_{110} + x_{111}$	$x_{000} - x_{001} + x_{010} - x_{011} + x_{100} - x_{101} + x_{110} - x_{111}$
x_{010}	$x_{100} + x_{101}$	$x_{000} - x_{001} + x_{010} - x_{011}$	$x_{000} + x_{001} - x_{010} - x_{011} + x_{100} + x_{101} - x_{110} - x_{111}$
x_{011}	$x_{110} + x_{111}$	$x_{100} - x_{101} + x_{110} - x_{111}$	$x_{000} - x_{001} - x_{010} + x_{011} + x_{100} - x_{101} - x_{110} + x_{111}$
x_{100}	$x_{000} - x_{001}$	$x_{000} + x_{001} - x_{010} - x_{011}$	$x_{000} + x_{001} + x_{010} + x_{011} - x_{100} - x_{101} - x_{110} - x_{111}$
x_{101}	$x_{010} - x_{011}$	$x_{100} + x_{101} - x_{110} - x_{111}$	$x_{000} - x_{001} + x_{010} - x_{011} - x_{100} + x_{101} - x_{110} + x_{111}$
x_{110}	$x_{100} - x_{101}$	$x_{000} - x_{001} - x_{010} + x_{011}$	$x_{000} + x_{001} - x_{010} - x_{011} - x_{100} - x_{101} + x_{110} + x_{111}$
x_{111}	$x_{110} - x_{111}$	$x_{100} - x_{101} - x_{110} + x_{111}$	$x_{000} - x_{001} - x_{010} + x_{011} - x_{100} + x_{101} + x_{110} - x_{111}$

为了从“给定”达到“第一步”，需要四次加法和四次减法。Yates 方法的有趣特征是使我们从“给定”到达“第一步”所进行的变换同样使我们从“第一步”到达“第二步”和从“第二步”到达“第三步”。在每种情况下，我们做四次加法，然后四次减法；而且在三步之后，我们在原来由 $F(s_1, s_2, s_3)$ 所占据的位置上幻术般地有了所希望的傅里叶变换 $f(s_1, s_2, s_3)$ 。

这个特殊情况通常称做 2^n 个数据元素的阿达马变换或沃尔什变换，因为对应的符号模式是由 J. Hadamard [*Bull. Sci. Math.* (2) 17 (1893), 240 ~ 246] 和 J. L. Walsh [*Amer. J. Math.* 45 (1923), 5 ~ 24] 研究的。注意，在上边的“第三步”从左到右符号变化的个数分别为

$$0, 7, 3, 4, 1, 6, 2, 5$$

这是数 $\{0, 1, 2, 3, 4, 5, 6, 7\}$ 的一个排列。沃尔什发现，如果我们适当地排列变换的元素，则在一般情况下将恰有 $0, 1, \dots, 2^n - 1$ 个符号变化，所以这些系数提供了对于具有各种频率的正弦曲线的离散近似。（关于阿达马-沃尔什系数的进一步讨论见 7.2.1.1 小节。）

Yates 的方法可以推广到任何离散傅里叶变换的求值，而且事实上可以推广到能写成一般形式

$$f(s_1, s_2, \dots, s_n) = \sum_{\substack{0 \leq t_1 < m_1 \\ \vdots \\ 0 \leq t_n < m_n}} g_1(s_1, s_2, \dots, s_n, t_1) g_2(s_2, \dots, s_n, t_2) \cdots g_n(s_n, t_n) F(t_1, t_2, \dots, t_n) \quad (39)$$

的任何和集求值，其中 $0 \leq s_j < m_j$ ，函数 $g_j(s_j, \dots, s_n, t_j)$ 是给定的。具体进行如下：

$$\begin{aligned} f_0(t_1, t_2, t_3, \dots, t_n) &= F(t_1, t_2, t_3, \dots, t_n) \\ f_1(s_n, t_1, t_2, \dots, t_{n-1}) &= \sum_{0 \leq t_n < m_n} g_n(s_n, t_n) f_0(t_1, t_2, \dots, t_n) \\ f_2(s_{n-1}, s_n, t_1, \dots, t_{n-2}) &= \sum_{0 \leq t_{n-1} < m_{n-1}} g_{n-1}(s_{n-1}, s_n, t_{n-1}) f_1(s_n, t_1, \dots, t_{n-1}) \\ &\vdots \\ f_n(s_1, s_2, s_3, \dots, s_n) &= \sum_{0 \leq t_1 < m_1} g_1(s_1, \dots, s_n, t_1) f_{n-1}(s_2, s_3, \dots, s_n, t_1) \end{aligned}$$

$$f(s_1, s_2, s_3, \dots, s_n) = f_n(s_1, s_2, s_3, \dots, s_n) \quad (40)$$

对于如上所示的 Yates 方法, $g_j(s_j, \dots, s_n, t_j) = (-1)^{t_j}$; $f_0(t_1, t_2, t_3)$ 表示“给定”; $f_1(s_3, t_1, t_2)$ 表示“第一步”, 等等。每当对于相当简单的函数 $g_j(s_j, \dots, s_n, t_j)$, 一个所求的和集可以写成(39)的形式时, 方案(40)将把计算的数量的阶从 N^2 减少到 $N \log N$ 左右, 其中 $N = m_1 \cdots m_n$ 是数据点的个数; 而且这个方案对于并行计算也很理想。习题 14 和 53 中讨论了一维傅里叶变换的重要情况; 我们在 4.3.3C 小节中也考虑了一维的情况。

我们考虑多项式计算的一种更特殊情况。 n 阶拉格朗日内插多项式可写成

$$\begin{aligned} u_{[n]}(x) = & y_0 \frac{(x-x_1)(x-x_2)\cdots(x-x_n)}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)} + \\ & y_1 \frac{(x-x_0)(x-x_2)\cdots(x-x_n)}{(x_1-x_0)(x_1-x_2)\cdots(x_1-x_n)} + \cdots + \\ & y_n \frac{(x-x_0)(x-x_1)\cdots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\cdots(x_n-x_{n-1})} \end{aligned} \quad (41)$$

它是 x 的次数 $\leq n$ 的惟一多项式, 它在 $n+1$ 个不同点 $x = x_0, x_1, \dots, x_n$ 处分别取值 y_0, y_1, \dots, y_n 。 (因为由(41)显然可知, 对于 $0 \leq k \leq n$, 有 $u_{[n]}(x_k) = y_k$ 。如果 $f(x)$ 是次数 $\leq n$ 的任何这样的多项式, 则 $g(x) = f(x) - u_{[n]}(x)$ 的次数 $\leq n$, 而且对于 $x = x_0, x_1, \dots, x_n$, $g(x)$ 为 0; 因此 $g(x)$ 必定是多项式 $(x-x_0)(x-x_1)\cdots(x-x_n)$ 的倍数。后一多项式的次数大于 n , 所以 $g(x) = 0$ 。) 如果我们假定在某个表中一个函数的值可以用一个多项式很好地近似, 则拉格朗日公式(41)可用来在表中未出现的点 x 处“内插”函数的值。拉格朗日于 1795 年向他在巴黎师范学校的学生介绍了(41)[见他的 *Œuvres* 7 (Paris: 1877), 286]; 但是剑桥大学的 Edward Waring 实际上应得这个荣誉, 因为他在 *Philosophical Transactions* 69 (1779), 59~67 中就已经十分清楚明晰地给出相同的公式。

看样子在 Waring 和拉格朗日的公式中有不少的加法、减法、乘法和除法。事实上, 恰有 n 次加法, $2n^2 + 2n$ 次减法, $2n^2 + n - 1$ 次乘法及 $n + 1$ 次除法。但幸而(如同我们现在有条件猜想的那样)有可能进行改进。

简化(41)的基本思想是利用以下的事实: 对于 $x = x_0, \dots, x_{n-1}$,

$$u_{[n]}(x) - u_{[n-1]}(x) = 0$$

于是 $u_{[n]}(x) - u_{[n-1]}(x)$ 是 n 次或更少次的一个多项式, 而且是 $(x-x_0)\cdots(x-x_{n-1})$ 的一个倍数。我们结论 $u_{[n]}(x) = \alpha_n(x-x_0)\cdots(x-x_{n-1}) + u_{[n-1]}(x)$, 其中 α_n 是一个常数。这就导致牛顿内插公式

$$\begin{aligned} u_{[n]}(x) = & \alpha_n(x-x_0)(x-x_1)\cdots(x-x_{n-1}) + \cdots + \\ & \alpha_2(x-x_0)(x-x_1) + \alpha_1(x-x_0) + \alpha_0 \end{aligned} \quad (42)$$

其中诸 α 是我们从给定的 $x_0, x_1, \dots, x_n, y_0, y_1, \dots, y_n$ 确定的某些系数。注意, 这个公式对于所有 n 成立; 系数 α_k 不依赖于 x_{k+1}, \dots, x_n 或 y_{k+1}, \dots, y_n 。一旦诸 α 已

知,牛顿内插公式就便于计算了,因为我们可以再次推广霍纳规则并令

$$u_{[n]}(x) = ((\cdots(a_n(x - x_{n-1}) + a_{n-1})(x - x_{n-2}) + \cdots)(x - x_0) + a_0) \quad (43)$$

这要求 n 次乘法和 $2n$ 次加法。或者,我们可以从右到左地计算(42)的每个个别项;这样用 $2n-1$ 次乘法和 $2n$ 次加法就可以计算所有值 $u_{[0]}(x), u_{[1]}(x), \cdots, u_{[n]}(x)$, 而这表明一个内插过程是否收敛。

通过在下表中(这里示出 $n=3$ 的情况)计算均差,可以求出牛顿公式中的系数 α_k :

$$\begin{array}{l} y_0 \\ y_1 \\ y_2 \\ y_3 \end{array} \quad \begin{array}{l} (y_1 - y_0)/(x_1 - x_0) = y'_1 \\ (y_2 - y_1)/(x_2 - x_1) = y'_2 \\ (y_3 - y_2)/(x_3 - x_2) = y'_3 \end{array} \quad \begin{array}{l} (y'_2 - y'_1)/(x_2 - x_0) = y''_2 \\ (y'_3 - y'_2)/(x_3 - x_1) = y''_3 \end{array} \quad \begin{array}{l} (y''_3 - y''_2)/(x_3 - x_0) = y'''_3 \end{array} \quad (44)$$

有可能证明 $\alpha_0 = y_0, \alpha_1 = y'_1, \alpha_2 = y''_2$, 等等,并证明均差同被内插函数的导数有重要关系;见习题 15。因此下列计算(对应于(44))可用来得到诸 α :

由 $(\alpha_0, \alpha_1, \cdots, \alpha_n) \leftarrow (y_0, y_1, \cdots, y_n)$ 开始;

然后对于 $k=1, 2, \cdots, n$ (按这个顺序),

对于 $j=n, n-1, \cdots, k$ (按这个顺序),置 $\alpha_j \leftarrow (\alpha_j - \alpha_{j-1})/(x_j - x_{j-k})$ 。

这个过程需要 $\frac{1}{2}(n^2+n)$ 次除法和 n^2+n 次减法,所以大约节省了(41)的工作量的四分之三。

例如,假设我们要用一个三次多项式从 $0!, 1!, 2!$ 和 $3!$ 的值估计 $1.5!$ 。均差是

x	y	y'	y''	y'''
0	1			
1	1	0	$\frac{1}{2}$	
2	2	1	$\frac{3}{2}$	$\frac{1}{3}$
3	6	4		

所以 $u_{[0]}(x) = u_{[1]}(x) = 1, u_{[2]}(x) = \frac{1}{2}x(x-1) + 1, u_{[3]}(x) = \frac{1}{3}x(x-1)(x-2) + \frac{1}{2}x(x-1) + 1$ 。在 $u_{[3]}(x)$ 中置 $x=1.5$ 给出 $-.125 + .375 + 1 = 1.25$;可以推测“正确”的值是 $\Gamma(2.5) = \frac{3}{4}\sqrt{\pi} \approx 1.33$ 。(但是当然有许多其它的序列,它们也以数 $1, 1, 2$ 和 6 开始。)

如果我们要对若干个多项式进行内插,这些多项式有相同的内插点 x_0, x_1, \cdots, x_n , 但有不同的值 y_0, y_1, \cdots, y_n , 则以 W. J. Taylor 所建议的形式 [J. Research Nat. Bur. Standards 35 (1945), 151~155] 重写(41)是合乎要求的:

$$u_{[n]}(x) = \left(\frac{y_0 w_0}{x - x_0} + \cdots + \frac{y_n w_n}{x - x_n} \right) \bigg/ \left(\frac{w_0}{x - x_0} + \cdots + \frac{w_n}{x - x_n} \right) \quad (45)$$

当 $x \in |x_0, x_1, \dots, x_n|$ 时, 其中

$$w_k = 1/(x_k - x_0) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n) \quad (46)$$

为了其数值稳定性, 也建议采用这个形式[请见 P. Henrici, *Essentials of Numerical Analysis* (New York: Wiley 1982), 237 ~ 243]。(45)的分母是 $1/(x - x_0)(x - x_1) \cdots (x - x_n)$ 的部分分式展开。

Adi. Shamir [CACM 22 (1979), 612 ~ 613] 发现了多项式内插的一个重要的和令人惊奇的应用, 他发现多项式模 p 可用来“分享一个秘密”。这意味着我们可以设计一个密钥或口令系统, 只要掌握这个系统中的任何 $n+1$ 把钥匙就能有效地计算能打开(比方说)一个门的魔数 N , 但是只掌握任何 n 把钥匙却不能得到任何关于 N 的信息。Shamir 给出了对这问题的简单得令人惊讶的解, 他选择一个随机多项式 $u(x) = u_n x^n + \cdots + u_1 x + u_0$, 其中 $0 \leq u_i < p$ 且 p 是一个很大的素数。秘密的每一部分是在 $0 < x < p$ 范围中的整数 p , 连同 $u(x) \bmod p$ 的值; 而且超秘密的数 N 是常数项 u_0 。给定 $n+1$ 个值 $u(x_i)$, 我们可以通过内插导出 N 。而如果只给了 n 个 $u(x_i)$ 值, 则存在惟一的具有给定常数项的多项式 $u(x)$, 但它在 x_1, \dots, x_n 处有同样的值。因此这 n 个值并不能使某个特定的 N 比任何其它值更有可能性。

注意, 计算内插多项式恰好是 4.3.2 小节和习题 4.6.2-3 的中国剩余算法的一个特殊情况, 因为我们知道 $u_{[n]}(x)$ 模诸互素多项式 $x - x_0, \dots, x - x_n$ 的值。(如同我们在 4.6.2 小节中看到的, $f(x) \bmod (x - x_0) = f(x_0)$ 。)在这个解释之下, 牛顿的公式(42)正好是等式 4.3.2-(25)的“混合进制表示”; 而且 4.3.2-(24)提供了用和(44)相同的运算次数来计算 a_0, \dots, a_n 的另一个方法。

应用快速傅里叶变换, 有可能把内插的运行时间减少成为 $O(n(\log n)^2)$, 对于有关的算法, 例如解中国剩余问题和计算 n 个不同的点处 n 次多项式的值, 也可相应地缩减运行时间。[见 E. Horowitz, *Inf. Proc. Letters* 1 (1972), 157 ~ 163; A. Borodin 和 R. Moenck, *J. Comp. Syst. Sci.* 8 (1974), 336 ~ 385; A. Borodin, *Complexity of Sequential and Parallel Numerical Algorithms*, J. F. Traub 编 (New York: Academic Press, 1973), 149 ~ 180; D. Bini 和 V. Pan, *Polynomial and Matrix Computations* 1 (Boston: Birkhäuser, 1994), 第 1 章。]然而这些发现主要是有理论意义, 因为这些已知的算法都有相当大的开销因子, 使得它们没有吸引力除非当 n 很大时。

由 T. N. Thiele 于 1909 年给出的对于均差方法的一个重要推广, 既可应用于多项式也可应用于多项式的商。在 L. M. Milne-Thompson 的 *Calculus of Finite Differences* (London: MacMillan, 1933), 第 5 章中讨论了 Thiele 的“倒数差”方法; 也见 R. W. Floyd, *CACM* 3(1960), 508。

*** 双线性形式** 我们在本小节中已经讨论的好多个问题是计算一组双线性形式的一般问题的特殊情况。双线性形式是

$$z_k = \sum_{i=1}^n \sum_{j=1}^n t_{ijk} x_i y_j, \quad 1 \leq k \leq s \quad (47)$$

其中 t_{ijk} 是属于某一给定域的特定系数。三维数组 (t_{ijk}) 称为 $m \times n \times s$ 张量, 我们可以把它写成 s 个 $m \times n$ 矩阵, 即对每个 k 值写一个矩阵。例如, 复数乘法问题, 即计算

$$z_1 + iz_2 = (x_1 + ix_2)(y_1 + iy_2) = (x_1y_1 - x_2y_2) + i(x_1y_2 + x_2y_1) \quad (48)$$

的问题, 是计算由 $2 \times 2 \times 2$ 张量

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

确定的双线性形式的问题。在(34)中定义的矩阵乘法是计算对应于一个具体的 $mn \times ns \times ms$ 张量的一组双线性形式的问题。如果我们令诸 x 是常数而不是变量, 那么傅里叶变换也可铸入这个模型中, 尽管它们是线性的而不是双线性的。

如果我们限定于可称做规范(normal)的计算方案, 则双线性形式的计算最容易研究。在这种方案下, 所有链的乘法在诸 x 的一个线性组合和诸 y 的线性组合之间发生。于是, 我们构造 r 个乘积

$$w_l = (a_{1l}x_1 + \cdots + a_{ml}x_m)(b_{1l}y_1 + \cdots + b_{nl}y_n), \quad 1 \leq l \leq r \quad (49)$$

并得到诸 z 作为这些乘积的线性组合

$$z_k = c_{k1}w_1 + \cdots + c_{kr}w_r, \quad 1 \leq k \leq s \quad (50)$$

这里所有的 a, b 和 c 都属于一个给定的系数域。比较(50)和(47), 我们看到一个规范的计算方案对于张量 (t_{ijk}) 是正确的当且仅当对于 $1 \leq i \leq m, 1 \leq j \leq n$ 及 $1 \leq k \leq s$,

$$t_{ijk} = a_{i1}b_{j1}c_{k1} + \cdots + a_{ir}b_{jr}c_{kr} \quad (51)$$

我们说一个非 0 张量 (t_{ijk}) 的秩(rank)为 1, 如果有三个向量 (a_1, \cdots, a_m) , (b_1, \cdots, b_n) , (c_1, \cdots, c_s) 使得对于所有 i, j, k , $t_{ijk} = a_i b_j c_k$ 。通过说 (t_{ijk}) 的秩是使得 (t_{ijk}) 可表达为给定域中 r 个秩为 1 的张量之和的极小数 r , 我们即可把这个定义推广到所有张量上。把这个定义同等式(51)作一下比较, 即表明一个张量的秩是在对应的双线性形式的规范计算中链乘法的极小个数。附带指出, 当 $s=1$ 时张量 (t_{ijk}) 仅仅是一个通常的矩阵, 而 (t_{ijl}) 作为一个张量的秩和作为一个矩阵的秩相同(见习题 49)。张量的秩的概念是由 F. L. Hitchcock 在 *J. Math. and Physics* 6 (1927), 164~189 中引进的; 它对多项式计算的复杂性的应用是由 V. Strassen 在一篇重要的文章中指出的, *Crelle* 264 (1973), 184~202。

关于矩阵乘法的 Winograd 方案(35)是“异常的”, 因为在做乘法之前, 它把诸 x 和诸 y 混合起来。另一方面, Strassen-Winograd 方案(36)并不依赖于乘法的可交换性, 所以它是规范的。事实上, (36)相当于对于 2×2 矩阵乘法用下列方式来把 4×4 张量表示为 7 个秩为 1 的张量之和:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} +$$

$$\text{(这里 } \bar{1} \text{ 表示 } -1_c\text{)} \quad (52)$$

当张量 (t_{ijk}) 可以表示为 r 个秩为 1 的张量的和(51)时, 命 A, B, C 是大小分别为 $m \times r, n \times r, s \times r$ 的矩阵 $(a_{ij}), (b_{jl}), (c_{kl})$; 我们将说 A, B, C 是张量 (t_{ijk}) 的一个实现。例如, (52) 中 2×2 矩阵乘法的实现可以由矩阵

确定。

引理 T 设 (A, B, C) 是一个 $m \times n \times s$ 张量 (t_{ijk}) 的实现, 则 $\text{rank}(A) \geq$

$\text{rank}(t_{i(jk)}), \text{rank}(B) \geq \text{rank}(t_{j(ik)}),$ 且 $\text{rank}(C) \geq \text{rank}(t_{k(ij)});$ 因此

$$\text{rank}(t_{ijk}) \geq \max(\text{rank}(t_{i(jk)}), \text{rank}(t_{j(ik)}), \text{rank}(t_{k(ij)}))$$

证明 由对称性只须证明 $r \geq \text{rank}(A) \geq \text{rank}(t_{i(jk)})$ 。由于 A 是一个 $m \times r$ 矩阵, 显然, A 不能有大于 r 的秩。而且, 按照(51), 矩阵 $(t_{i(jk)})$ 等于 AQ , 其中 Q 是由 $Q_{l(j,k)} = b_{jl}c_{kl}$ 定义的 $r \times ns$ 矩阵。如果 x 是任何使得 $xA = 0$ 的行向量, 则 $xAQ = 0$, 因此在 A 中的所有线性相关性在 AQ 中也出现。由此得出 $\text{rank}(AQ) \leq \text{rank}(A)$ 。■

作为使用引理 T 的一个例子, 我们考虑多项式乘法的问题。假设我们要把一个一般的二次多项式乘以一个一般的三次多项式, 得到乘积的系数

$$(x_0 + x_1u + x_2u^2)(y_0 + y_1u + y_2u^2 + y_3u^3) = z_0 + z_1u + z_2u^2 + z_3u^3 + z_4u^4 + z_5u^5 \quad (54)$$

这是计算对应于 $3 \times 4 \times 6$ 张量

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (55)$$

的六个双线性形式的问题。为了简便, 我们可以把(54)写成 $x(u)y(u) = z(u)$, 令 $x(u)$ 表示多项式 $x_0 + x_1u + x_2u^2$, 等等。(我们又回到了在这一小节开始时的老地方, 因为等式(1)指的是 $u(x)$ 而不是 $x(u)$; 我们换了一种表示方法, 因为多项式的系数现在是我们所感兴趣的变量。)

如果(55)中的六个矩阵的每一个被认为是以 $\langle i, j \rangle$ 作为下标的长度为 12 的一个向量, 显然这些向量线性无关, 因为它们在不同位置中非零。因此由引理 T, (55)的秩至少是 6。相反, 通过仅仅做六个链乘法, 例如通过计算

$$x(0)y(0), x(1)y(1), \dots, x(5)y(5) \quad (56)$$

有可能得到系数 z_0, z_1, \dots, z_5 , 这给出 $z(0), z(1), \dots, z(5)$ 的值, 而且上边为了内插而建立的公式将产生 $z(u)$ 的系数。 $x(j)$ 和 $y(j)$ 的计算可以完全借助加法和/或参数乘法来进行, 而且内插公式只采取这些值的线性组合的形式。因此, 所有链乘法如(56)所示, 而且(55)的秩是 6。(当在算法 4.3.3T 中乘高精度的数时, 我们实质上使用同样的技术。)

结果, 上一段中概述的(55)的实现 (A, B, C) 是

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 4 & 9 & 16 & 25 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 4 & 9 & 16 & 25 \\ 0 & 1 & 8 & 27 & 64 & 125 \end{pmatrix} \cdot \begin{pmatrix} 120 & 0 & 0 & 0 & 0 & 0 \\ -274 & 600 & -600 & 400 & -150 & 24 \\ 225 & -770 & 1070 & -780 & 305 & -50 \\ -85 & 355 & -590 & 490 & -205 & 35 \\ 15 & -70 & 130 & -120 & 55 & -10 \\ -1 & 5 & -10 & 10 & -5 & 1 \end{pmatrix} \times \frac{1}{120} \quad (57)$$

于是,这个方案确实实现了极小的链乘法个数,但它完全不实用,因为它包含了这么多加法和参数乘法。我们现在将研究由 S. Winograd 提出的生成更有效方案的一个实用方法。

首先,当 $\deg(x) = m$ 和 $\deg(y) = n$, 且当 $p(u)$ 是任意 $m+n$ 次首一多项式时,为计算 $x(u)y(u)$ 的系数,可以使用恒等式

$$x(u)y(u) = (x(u)y(u) \bmod p(u)) + x_m y_n p(u) \quad (58)$$

多项式 $p(u)$ 应当选择成使得 $x(u)y(u) \bmod p(u)$ 的系数易于计算。

其次,当多项式 $p(u)$ 可以分解成 $q(u)r(u)$, 其中 $\gcd(q(u), r(u)) = 1$ 时,为计算 $x(u)y(u) \bmod p(u)$ 的系数,可以使用恒等式

$$\begin{aligned} x(u)y(u) \bmod q(u)r(u) &= (a(u)r(u)(x(u)y(u) \bmod q(u)) + \\ &\quad b(u)q(u)(x(u)y(u) \bmod r(u))) \bmod q(u)r(u) \end{aligned} \quad (59)$$

其中 $a(u)r(u) + b(u)q(u) = 1$; 这实质上是中国剩余定理应用于多项式的情况。

第三,通过使用平凡的恒等式

$$x(u)y(u) \bmod p(u) = (x(u) \bmod p(u))(y(u) \bmod p(u)) \bmod p(u) \quad (60)$$

我们总能计算多项式 $x(u)y(u) \bmod p(u)$ 的系数。如同我们将看到的那样,重复应用(58),(59)和(60)会产生有效的方案。

对于例子(54),我们选择 $p(u) = u^5 - u$ 并应用(58);作这一选择的原因看上去就会明白。令 $p(u) = u(u^4 - 1)$, 则规则(59)归结为

$$\begin{aligned} x(u)y(u) \bmod u(u^4 - 1) &= (- (u^4 - 1)x_0y_0 + \\ &\quad u^4(x(u)y(u) \bmod (u^4 - 1))) \bmod (u^5 - u) \end{aligned} \quad (61)$$

这里我们使用了 $x(u)y(u) \bmod u = x_0y_0$ 这一事实;一般来说,以使 $p(0) = 0$ 这样的方式来选择 $p(u)$ 是一个好的想法,这使使用上述简化成为可能。如果我们现在能确定多项式 $x(u)y(u) \bmod (u^4 - 1) = w_0 + w_1u + w_2u^2 + w_3u^3$ 的系数 w_0, w_1, w_2, w_3 , 我们的问题就将获解,因为

$$u^4(x(u)y(u) \bmod (u^4 - 1)) \bmod (u^5 - u) = w_0u^4 + w_1u + w_2u^2 + w_3u^3$$

而且(58)和(61)的组合将归结为

$$\begin{aligned} x(u)y(u) &= x_0y_0 + (w_1 - x_2y_3)u + w_2u^2 + w_3u^3 + \\ &\quad (w_0 - x_0y_0)u^4 + x_2y_3u^5 \end{aligned} \quad (62)$$

(当然这个等式可直接验证。)

剩下需要解决的问题是计算 $x(u)y(u) \bmod (u^4 - 1)$; 而这个子问题本身就很有趣。我们暂且允许 $x(u)$ 是三次的而不是二次的, 则 $x(u)y(u) \bmod (u^4 - 1)$ 的系数分别为

$$\begin{aligned} x_0y_0 + x_1y_3 + x_2y_2 + x_3y_1, & \quad x_0y_1 + x_1y_0 + x_2y_3 + x_3y_2, \\ x_0y_2 + x_1y_1 + x_2y_0 + x_3y_3, & \quad x_0y_3 + x_1y_2 + x_2y_1 + x_3y_0 \end{aligned}$$

而对应的张量为

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (63)$$

一般来说, 当 $\deg(x) = \deg(y) = n - 1$ 时, $x(u)y(u) \bmod (u^n - 1)$ 的系数称为 $(x_0, x_1, \dots, x_{n-1})$ 和 $(y_0, y_1, \dots, y_{n-1})$ 的循环卷积。第 k 个系数 w_k 是双线性形式 $\sum x_i y_j$, 它对满足 $i + j \equiv k \pmod{n}$ 的所有 i 和 j 求和。

应用规则(59)可以得到 4 次循环卷积。头一步是求 $u^4 - 1$ 的因子, 即 $(u - 1) \cdot (u + 1)(u^2 + 1)$ 。我们可以把这写成为 $(u^2 - 1)(u^2 + 1)$, 然后应用规则(59), 然后再次使用(59)到模 $(u^2 - 1) = (u - 1)(u + 1)$ 那部分; 但更容易的方法是直接推广中国剩余规则(59)到几个互素因子的情况。例如, 我们有

$$\begin{aligned} x(u)y(u) \bmod q_1(u)q_2(u)q_3(u) &= (a_1(u)q_2(u)q_3(u)(x(u)y(u) \bmod q_1(u)) + \\ &\quad a_2(u)q_1(u)q_3(u)(x(u)y(u) \bmod q_2(u)) + \\ &\quad a_3(u)q_1(u)q_2(u)(x(u)y(u) \bmod q_3(u))) \bmod q_1(u)q_2(u)q_3(u) \end{aligned} \quad (64)$$

其中 $a_1(u)q_2(u)q_3(u) + a_2(u)q_1(u)q_3(u) + a_3(u)q_1(u)q_2(u) = 1$ 。(通过注意到 $1/q_1(u)q_2(u)q_3(u)$ 的部分分式展开是 $a_1(u)/q_1(u) + a_2(u)/q_2(u) + a_3(u)/q_3(u)$, 也可以以另一种方式来理解这个等式。)由(64)我们得到

$$\begin{aligned} x(u)y(u) \bmod (u^4 - 1) &= \left(\frac{1}{4}(u^3 + u^2 + u + 1)x(1)y(1) - \frac{1}{4}(u^3 - u^2 + u - 1) \cdot \right. \\ &\quad \left. x(-1)y(-1) - \frac{1}{2}(u^2 - 1)(x(u)y(u) \bmod (u^2 + 1)) \right) \bmod (u^4 - 1) \end{aligned} \quad (65)$$

余下的问题是计算 $x(u)y(u) \bmod (u^2 + 1)$, 而这正好是调用规则(60)的时候。首先我们简化 $x(u)$ 和 $y(u) \bmod (u^2 + 1)$, 得到 $X(u) = (x_0 - x_2) + (x_1 - x_3)u$, $Y(u) = (y_0 - y_2) + (y_1 - y_3)u$ 。然后由(60)可计算 $X(u)Y(u) = Z_0 + Z_1u + Z_2u^2$, 进而在模 $(u^2 + 1)$ 下简化这个式子, 得到 $(Z_0 - Z_2) + Z_1u$ 。计算 $X(u)Y(u)$ 的工作是简单的; 我们可以以 $p(u) = u(u + 1)$ 来使用规则(58), 得到

$$Z_0 = X_0Y_0, \quad Z_1 = X_0Y_1 - (X_0 - X_1)(Y_0 - Y_1) + X_1Y_1, \quad Z_2 = X_1Y_1$$

(我们由此以更加系统的方式重新发现等式 4.3.3-(2)中的窍门。)综上所述, 就得到了 4 次循环卷积的下列实现 (A, B, C) :

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 1 & \bar{1} & 0 & 1 & \bar{1} \\ 1 & 1 & \bar{1} & 0 & \bar{1} \\ 1 & \bar{1} & 0 & \bar{1} & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 1 & \bar{1} & 0 & 1 & 1 \\ 1 & 1 & \bar{1} & 0 & \bar{1} \\ 1 & \bar{1} & 0 & \bar{1} & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 2 & \bar{2} & 0 \\ 1 & \bar{1} & 2 & 2 & \bar{2} \\ 1 & 1 & \bar{2} & 2 & 0 \\ 1 & \bar{1} & \bar{2} & \bar{2} & 2 \end{pmatrix} \times \frac{1}{4} \quad (66)$$

这里 $\bar{1}$ 代表 -1 , $\bar{2}$ 代表 -2 。

n 次循环卷积的张量满足

$$t_{i,j,k} = t_{k,-j,-i} \quad (67)$$

在模 n 之下处理下标, 因为当且仅当 $i + j \equiv k \pmod{n}$ 时, $t_{ijk} = 1$ 。于是如果 $(a_{il}), (b_{jl}), (c_{kl})$ 是循环卷积的一个实现则 $(c_{kl}), (b_{-j,l}), (a_{il})$ 也是; 特别是, 我们可以通过把(66)变换成为

$$\begin{pmatrix} 1 & 1 & 2 & \bar{2} & 0 \\ 1 & \bar{1} & 2 & 2 & \bar{2} \\ 1 & 1 & \bar{2} & 2 & 0 \\ 1 & \bar{1} & \bar{2} & \bar{2} & 2 \end{pmatrix} \times \frac{1}{4}, \begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 1 & \bar{1} & 0 & \bar{1} & 1 \\ 1 & 1 & \bar{1} & 0 & \bar{1} \\ 1 & \bar{1} & 0 & 1 & \bar{1} \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 1 & \bar{1} & 0 & 1 & \bar{1} \\ 1 & 1 & \bar{1} & 0 & \bar{1} \\ 1 & \bar{1} & 0 & \bar{1} & 1 \end{pmatrix} \quad (68)$$

而实现(63)。现在所有复杂的标量见于矩阵 A 中。这在应用中是很重要的, 因为我们通常要对许多不同的 y_0, y_1, y_2, y_3 和固定的 x_0, x_1, x_2, x_3 来计算卷积。在这种情况下, 对于 x 的算术运算可以一劳永逸地完成, 因此我们不必把它算在内。于是当事先知道 x_1, x_2, x_3, x_4 时, (68) 导致计算循环卷积 w_0, w_1, w_2, w_3 的下列方案:

$$\begin{aligned} s_1 &= y_0 + y_2, & s_2 &= y_1 + y_3, & s_3 &= s_1 + s_2, & s_4 &= s_1 - s_2, \\ s_5 &= y_0 - y_2, & s_6 &= y_3 - y_1, & s_7 &= s_5 - s_6 \\ m_1 &= \frac{1}{4}(x_0 + x_1 + x_2 + x_3) \cdot s_3, & m_2 &= \frac{1}{4}(x_0 - x_1 + x_2 - x_3) \cdot s_4, \\ m_3 &= \frac{1}{2}(x_0 + x_1 - x_2 - x_3) \cdot s_5 \\ m_4 &= \frac{1}{2}(-x_0 + x_1 + x_2 - x_3) \cdot s_6, & m_5 &= \frac{1}{2}(x_3 - x_1) \cdot s_7 \\ t_1 &= m_1 + m_2, & t_2 &= m_3 + m_5, & t_3 &= m_1 - m_2, & t_4 &= m_4 - m_5 \\ w_0 &= t_1 + t_2, & w_1 &= t_3 + t_4, & w_2 &= t_1 - t_2, & w_3 &= t_3 - t_4 \end{aligned} \quad (69)$$

它有 5 个乘法和 15 个加法, 而循环卷积的定义含 16 个乘法和 12 个加法。我们后边将证明, 5 个乘法是必要的。

利用(62)回到原来的乘法问题(54), 我们已经推导出实现

$$\begin{bmatrix} 4 & 0 & 1 & 1 & 2 & \bar{2} & 0 \\ 0 & 0 & 1 & \bar{1} & 2 & 2 & \bar{2} \\ 0 & 4 & 1 & 1 & \bar{2} & 2 & 0 \end{bmatrix} \times \frac{1}{4} \cdot \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & \bar{1} & 0 & \bar{1} & 1 \\ 0 & 0 & 1 & 1 & \bar{1} & 0 & \bar{1} \\ 0 & 1 & 1 & \bar{1} & 0 & 1 & \bar{1} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & \bar{1} \\ 0 & 0 & 1 & 1 & \bar{1} & 0 & \bar{1} \\ 0 & 0 & 1 & 1 & 0 & \bar{1} & 1 \\ \bar{1} & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (70)$$

这个方案比链乘法的极小次数多用一次乘法,但它比(57)要求少得多的参数乘法。当然,必须承认这个方案仍然颇为复杂:如果我们的目标只不过是计算两个给定的多项式的乘积 $(x_0 + x_1u + x_2u^2)(y_0 + y_1u + y_2u^2 + y_3u^3)$ 的系数 z_0, z_1, \dots, z_5 ,作为一个碰运气的问题,我们最好的赌注显然仍是做12次乘法和6次加法的那个方法——除非(比如说),诸 x 和诸 y 是矩阵。另外一个要求8次乘法和18次加法的相当吸引人的方案,出现于习题58(b)中。注意如果诸 x 是固定的而诸 y 是变化的,则(70)以7次乘法和17次加法进行计算。尽管这个方案本身不像它宣称的那样特别有用,但我们的推导已经显示了在各种各样其它情况下的一些重要技术。例如,Winograd已经利用这个方法,利用比快速傅里叶变换算法所需要的要少得多的乘法来计算傅里叶变换(见习题53)。

让我们通过确定对应于两个多项式在模第三个多项式之下的乘法的 $n \times n \times n$ 张量的精确秩来结束这小节。

$$z_0 + z_1u + \dots + z_{n-1}u^{n-1} = (x_0 + x_1u + \dots + x_{n-1}u^{n-1})(y_0 + y_1u + \dots + y_{n-1}u^{n-1}) \bmod p(u) \quad (71)$$

这里 $p(u)$ 代表任何给定的 n 次首一多项式;特别是, $p(u)$ 可以是 $u^n - 1$,所以我们研究的结果之一将是推导对应于 n 次循环卷积的张量的秩。以

$$p(u) = u^n - p_{n-1}u^{n-1} - \dots - p_1u - p_0 \quad (72)$$

的形式来写 $p(u)$ 是方便的,我们有 $u^n \equiv p_0 + p_1u + \dots + p_{n-1}u^{n-1} \pmod{p(u)}$ 。

张量元素 t_{ijk} 是在 $u^{i+j} \bmod p(u)$ 中 u^k 的系数;而这是矩阵 P' 的 i 行 k 列的元素,其中

$$P = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ p_0 & p_1 & p_2 & \cdots & p_{n-1} \end{bmatrix} \quad (73)$$

称为 $p(u)$ 的伴随矩阵。(在我们的讨论中下标 i, j, k 为从0到 $n-1$ 而不是从1到 n 。)把这个张量转置一下是方便的,因为如果 $T_{ijk} = t_{ijk}$,则对于 $k = 0, 1, 2, \dots, n-1$, (T_{ijk}) 的各层次可简单地由矩阵

$$I \quad P \quad P^2 \quad \cdots \quad P^{n-1} \quad (74)$$

给出。

(74)中矩阵的头一行分别是单位向量 $(1, 0, 0, \dots, 0)$, $(0, 1, 0, \dots, 0)$, $(0, 0, 1, \dots, 0)$, \dots , $(0, 0, 0, \dots, 1)$, 因此, 当且仅当系数 v_k 全为 0 时, 线性组合 $\sum_{k=0}^{n-1} v_k P^k$ 是零矩阵。而且, 大多数这些线性组合实质上是非奇异矩阵, 因为当且仅当 $v(u)w(u) \equiv 0 \pmod{p(u)}$ 时, 我们有

$$(w_0, w_1, \dots, w_{n-1}) \sum_{k=0}^{n-1} v_k P^k = (0, 0, \dots, 0)$$

其中 $v(u) = v_0 + v_1 u + \dots + v_{n-1} u^{n-1}$ 和 $w(u) = w_0 + w_1 u + \dots + w_{n-1} u^{n-1}$ 。因此, 当且仅当多项式 $v(u)$ 是 $p(u)$ 的某个因子的倍数时, $\sum_{k=0}^{n-1} v_k P^k$ 是奇异矩阵。我们现在已做好准备来证明所需的结果。

定理 W(S. Winograd, 1975) 设 $p(u)$ 是 n 次首一多项式, 它在一个给定无穷域上的完全分解是

$$p(u) = p_1(u)^{e_1} \cdots p_q(u)^{e_q} \quad (75)$$

则在这个域上对应于双线性形式(71)的张量(74)的秩为 $2n - q$ 。

证明 通过以适当方式使用规则(58), (59)和(68), 可以只用 $2n - q$ 个链乘法计算双线性形式, 所以我们只须证明秩 $r \geq 2n - q$ 。上边的讨论确定了 $\text{rank}(T_{(ij)k}) = n$; 因此由引理 T, (T_{ijk}) 的任何 $n \times r$ 的实现 (A, B, C) 都有 $\text{rank}(C) = n$ 。我们的策略是通过找出具有下列两个性质的一个向量 $(v_0, v_1, \dots, v_{n-1})$, 再次使用引理 T:

- i) 向量 $(v_0, v_1, \dots, v_{n-1})C$ 至多有 $q + r - n$ 个非零系数。
- ii) 矩阵 $v(P) = \sum_{k=0}^{n-1} v_k P^k$ 非奇异。

上述事实加上引理 T 将证明 $q + r - n \geq n$, 因为恒等式

$$\sum_{i=1}^r a_{il} b_{il} \left(\sum_{k=0}^{n-1} v_k c_{kl} \right) = v(P)_{ij}$$

表明如何用 $q + r - n$ 个链乘法实现秩为 n 的 $n \times n \times 1$ 张量 $v(P)$ 。

为方便起见, 我们可以假定 C 的头 n 列是线性无关的。设 D 是使得 DC 的头 n 列等于单位矩阵的 $n \times n$ 矩阵。如果有 D 的顶多 q 行的线性组合 $(v_0, v_1, \dots, v_{n-1})$ 使得 $v(P)$ 非奇异, 则我们的目标就实现了; 这样一个向量将满足条件 i) 和 ii)。

由于 D 的行是线性无关的, 所以没有不可约因子 $p_\lambda(u)$ 能整除对应于每个行的多项式。给定一个向量 $w = (w_0, w_1, \dots, w_{n-1})$, 令 $\text{covered}(w)$ 是使得 $w(u)$ 不是 $p_\lambda(u)$ 的倍数的所有 λ 的集合。由两个向量 v 和 w , 我们可以找到一个线性组合

$v + \alpha w$, 使得对于这域中的某个 α

$$\text{covered}(v + \alpha w) = \text{covered}(v) \cup \text{covered}(w) \quad (76)$$

原因是如果 λ 为 v 或 w 覆盖但不是为两者所覆盖, 则对于所有非 0 的 α , λ 为 $v + \alpha w$ 所覆盖; 如果 λ 为 v 和 w 两者所覆盖但不为 $v + \alpha w$ 所覆盖, 则对于所有 $\beta \neq \alpha$, λ 为 $v + \beta w$ 所覆盖。通过试验 $q + 1$ 个不同的 α 值, 至少有一个必然产生 (76)。这样一来, 我们可以系统地构造 D 的至多 q 行的线性组合, 并且对 $1 \leq \lambda \leq q$ 覆盖所有 λ 。 ■

定理 W 的最重要推论之一是, 一个张量的秩可以依赖于构成 (A, B, C) 的实现元素所属的域。例如, 考虑对应于 5 次循环卷积的张量; 这等价于多项式模 $p(u) = u^5 - 1$ 的乘法。由习题 4.6.2 - 32 知, 在有理数域上, $p(u)$ 的完全因子分解是 $(u - 1)(u^4 + u^3 + u^2 + u + 1)$, 所以张量的秩是 $10 - 2 = 8$ 。另一方面, 在实数上的完全分解, 借助于数 $\phi = \frac{1}{2}(1 + \sqrt{5})$, 是 $(u - 1)(u^2 + \phi u + 1)(u^2 - \phi^{-1}u - 1)$; 于是, 如果我们允许任意实数在 A, B, C 中出现, 则秩仅为 7。在复数上秩为 5。在二维张量 (即矩阵) 中, 秩可以通过计算子矩阵的行列式和测试 0 来确定, 就不出现这个现象。当含有矩阵的元素的域被嵌在一个更大的域中时, 一个矩阵的秩不变, 但当域变得更大时一个张量的秩可以变小。

在给出定理 W 的文章 [*Math. Systems Theory* 10 (1977), 169 ~ 180] 中, Winograd 证明, 当 q 大于 1 时, 在 $2n - q$ 个链乘法下 (71) 的所有实现对应于 (59) 的使用。此外他还证明, 在 $\deg(x) + \deg(y) + 1$ 个链乘法下计算 $x(u)y(u)$ 的系数的惟一方法是使用内插, 或对于在这个域中分解成不同线性因子的一个多项式使用 (58)。最后他证明了, 当 $q = 1$ 时, 在 $2n - 1$ 个链乘法中计算 $x(u)y(u) \bmod p(u)$ 的惟一方式实质上是使用 (60)。这些结果对所有多项式链都成立, 而不仅仅是对“规范”的那些。在 SICOMP 9 (1980), 225 ~ 229 中, 他把这些结果推广到多变量多项式。

在一个适当大的域中, 一个任意的 $m \times n \times 2$ 张量的张量秩已经由 Joseph Ja' Ja' 确定, SICOMP 8 (1979), 443 ~ 462; JACM 27 (1980), 822 ~ 830。也可见他在 SICOMP 9 (1980), 713 ~ 728 中关于可交换双线性形式的有趣讨论。然而, 计算在任意的有限域上一个任意的 $n \times n \times n$ 张量的张量秩的问题是 NP 完全的 [J. Hastad, *Journal of Algorithms* 11 (1990), 644 ~ 654]。

关于进一步的阅读资料 在这一小节里, 我们只是肤浅地接触了非常广泛的课题的表面; 在这个课题中正在出现许多优美的理论。关于更加广泛的讨论见 A. Borodin 和 I. Munro, *Computational Complexity of Algebraic and Numeric Problems* (New York: American Elsevier, 1975); D. Bini 和 V. Pan, *Polynomial and Matrix Computations 1* (Boston: Birkhäuser, 1994); P. Bürgisser, M. Clausen 和 M. Amin Shokrollahi, *Algebraic Complexity Theory* (Heidelberg: Springer, 1997)。

习 题

1. [15] 什么是计算一个“奇”多项式

$$u(x) = u_{2n+1}x^{2n+1} + u_{2n-1}x^{2n-1} + \cdots + u_1x$$

的好办法?

► 2. [M20] 如果不像在正文中那样用步骤 H1 和 H2 计算 $u(x+x_0)$, 试讨论当使用多项式乘法和加法而不用系数区域中的算术运算时如何应用霍纳规则(2)。

3. [20] 为求两个变量的多项式 $\sum_{i+j \leq n} u_{ij}x^i y^j$ 之值, 给出类似于霍纳规则的一个方法(这个多项式有 $(n+1)(n+2)/2$ 个系数, 且“总次数”为 n)。计算你所用的加法和乘法的次数。

4. [M20] 当我们在一个复数点 z 处对一个实系数多项式求值时, 正文说明方案(3)比霍纳规则优越。当系数和变量 z 两者都是复数时, 试比较(3)和霍纳规则; 对于每个方法需要多少次(实的)乘法和加减法?

5. [M15] 计算第二阶规则(4)所需要的乘法和加法的次数。

6. [22] (L. de Jong 和 J. van Leeuwen) 说明怎样只通过计算 x_0 的大约 $\frac{1}{2}n$ 次幂, 来改进 Shaw-Traub 算法的步骤 S1, ..., S4。

7. [M25] 如何计算 β_0, \dots, β_n , 使得对于所有整数 k , (6) 有值 $u(x_0 + kh)$?

8. [M20] 阶乘乘方 $x^{\frac{x}{k}}$ 定义为 $k! \binom{x}{k} = x(x-1)\cdots(x-k+1)$, 说明怎样以至多 n 次乘法和 $2n-1$ 次加法, 由 x 和 $n+3$ 个常数 $u_n, \dots, u_0, 1, n-1$ 开始, 来求 $u_n x^{\frac{x}{2}} + \cdots + u_1 x^{\frac{1}{2}} + u_0$ 的值。

9. [M25] (H. J. Ryser) 证明如果 $X = (x_{ij})$ 是一个 $n \times n$ 阶矩阵, 则

$$\text{per}(X) = \sum (-1)^{\epsilon_1 + \cdots + \epsilon_n} \prod_{1 \leq i \leq n} \sum_{1 \leq j \leq n} \epsilon_j x_{ij}$$

这里求和对所有独立地等于 0 和 1 的 2^n 种 $\epsilon_1, \dots, \epsilon_n$ 的选择进行。计算通过这个公式求 $\text{per}(X)$ 的值所需要的加法和乘法运算的次数。

10. [M21] 一个 $n \times n$ 阶矩阵 $X = (x_{ij})$ 的积和式可计算如下: 由 n 个量 $x_{11}, x_{12}, \dots, x_{1n}$ 开始, 对于 $1 \leq k < n$, 假定对于 $\{1, 2, \dots, n\}$ 的所有 k 元子集 S 已经计算了 $\binom{n}{k}$ 个量 A_{kS} , 其中求和 $A_{kS} = \sum x_{1j_1} \cdots x_{kj_k}$ 是对 S 的元素的所有 $k!$ 个排列 $j_1 \cdots j_k$ 进行的; 然后形成所有的和式

$$A_{(k+1)S} = \sum_{j \in S} A_{k(S \setminus \{j\})} x_{(k+1)j}$$

我们有 $\text{per}(X) = A_{n, \{1, \dots, n\}}$ 。这个方法要求多少次加法和乘法? 需要多少个临时存储?

11. [M46] 有没有任何方法使用少于 2^n 次算术运算, 来计算一个一般的 $n \times n$ 矩阵的积和式?

12. [M50] 为形成两个 $n \times n$ 矩阵的乘积要求的极小乘法次数是多少? 什么是最小的指数 ω 使得对所有 $\epsilon > 0$, 只要 $O(n^{\omega+\epsilon})$ 次乘法就够? (对小的和大的 n 求好的上限和下限。)

13. [M23] 借助于 $f(s_1, \dots, s_n)$ 的值表达 $F(t_1, \dots, t_n)$, 求出一般的离散傅里叶变换(37)的逆。[提示: 见等式 1.2.9-(13)。]

► 14. [HM28] (快速傅里叶变换) 证明: 利用对复数的算术运算, 方案(40)可用来计算一维离散傅里叶变换

$$f(s) = \sum_{0 \leq r < 2^n} F(r) \omega^{rs}, \quad \omega = e^{2\pi i/2^n}, \quad 0 \leq s < 2^n$$

估计所执行的算术运算的次数。

►15. [HM28] 在 $n+1$ 个不同的点 x_0, x_1, \dots, x_n 处函数 $f(x)$ 的 n 次均差 $f(x_0, x_1, \dots, x_n)$ 由公式

$$f(x_0, x_1, \dots, x_n) = (f(x_0, x_1, \dots, x_{n-1}) - f(x_1, \dots, x_{n-1}, x_n)) / (x_0 - x_n)$$

来定义, 其中 $n > 0$ 。于是, $f(x_0, x_1, \dots, x_n) = \sum_{k=0}^n f(x_k) / \prod_{0 \leq j < n, j \neq k} (x_k - x_j)$ 是它的 $n+1$ 个自变量的对称函数。(a) 证明: 对于在 $\min(x_0, \dots, x_n)$ 和 $\max(x_0, \dots, x_n)$ 之间的某个 θ , 如果 n 阶导数 $f^{(n)}(x)$ 存在且连续, 则 $f(x_0, \dots, x_n) = f^{(n)}(\theta) / n!$ 。[提示: 证明恒等式

$$f(x_0, x_1, \dots, x_n) = \int_0^1 dt_1 \int_0^{t_1} dt_2 \cdots \int_0^{t_{n-1}} dt_n f^{(n)}(x_0(1-t_1) + x_1(t_1-t_2) + \cdots + x_{n-1}(t_{n-1}-t_n) + x_n(t_n-0))$$

当 x_j 不是不同的时, 这个公式也以同一个有用的方式定义 $f(x_0, x_1, \dots, x_n)$ 。] (b) 如果 $y_j = f(x_j)$, 证明在牛顿内插公式(42)中 $a_j = f(x_0, \dots, x_j)$ 。

16. [M22] 如果给定牛顿内插多项式(42)中 $x_0, x_1, \dots, x_{n-1}, a_0, a_1, \dots, a_n$ 的值, 我们怎样能容易地计算 $u_{[n]}(x) = u_n x^n + \cdots + u_0$ 的系数?

17. [M20] 证明对于 $0 \leq k \leq n$, 当 $x_k = x_0 + kh$ 时, 内插公式(45)归结为涉及二项式系数的一个非常简单的表达式。[提示: 见习题 1.2.6-48。]

18. [M20] 如果把四次方案(9)变成

$$y = (x + a_0)x + a_1, \quad u(x) = ((y - x + a_2)y + a_3)a_4$$

为借助于诸 u_k 来计算诸 a_j , 应以什么公式取代(10)?

►19. [M24] 说明怎样从 $u(x)$ 的系数 u_5, \dots, u_1, u_0 来确定(11)中改写的系数 a_0, a_1, \dots, a_5 , 并且对具体的多项式 $u(x) = x^5 + 5x^4 - 10x^3 - 50x^2 + 13x + 60$ 求出诸 a 。

►20. [21] 写出一个 MIX 程序, 它按照方案(11)计算一个五次多项式; 试通过对(11)稍作修改, 使得这个程序尽可能地有效。使用 MIX 的浮点算术运算符 FADD 和 FMUL, 有关它们的介绍见 4.2.1 小节。

21. [20] 利用正文中未加考虑的(15)的两个根, 通过方案(12)找出计算多项式 $x^6 + 13x^5 + 49x^4 + 33x^3 - 61x^2 - 37x + 3$ 值的两个新方法。

22. [18] 怎样利用 Pan 的方法(16)计算 $x^6 - 3x^5 + x^4 - 2x^3 + x^2 - 3x - 1$ 的值?

23. [HM30] (J. Eve) 设 $f(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_0$ 是具有实系数的 n 次多项式, 它至少有 $n-1$ 个有非负实部的根。设

$$g(z) = a_n z^n + a_{n-2} z^{n-2} + \cdots + a_{n \bmod 2} z^{n \bmod 2}$$

$$h(z) = a_{n-1} z^{n-1} + a_{n-3} z^{n-3} + \cdots + a_{(n-1) \bmod 2} z^{(n-1) \bmod 2}$$

假定 $h(z)$ 不恒等于 0。

a) 证明 $g(z)$ 至少有 $n-2$ 个虚根(即实部为 0 的根), 而 $h(z)$ 至少有 $n-3$ 个虚根。[提示: 对于充分大的半径 R , 当 z 围绕图 16 中所示的通路行进时, 考虑通路 $f(z)$ 围绕原点循环的次数。]

b) 证明 $g(z) = 0, h(z) = 0$ 的根的平方都是实数。

►24. [M24] 对于多项式 $u(x) = (x+7)(x^2+6x+10)(x^2+4x+5)(x+1)$, 求满足定理 E 的条件的 c 和 α_k, β_k 的值。选择这些值得使得 $\beta_2 = 0$ 。请对这个问题给出两个不同的解。

25. [M20] 当把定理 M 证明中的构造应用于(低效的)多项式链

$$\lambda_1 = \alpha_1 + \lambda_0, \lambda_2 = -\lambda_0 - \lambda_0, \lambda_3 = \lambda_1 + \lambda_1, \lambda_4 = \alpha_2 \times \lambda_1,$$

$$\lambda_5 = \lambda_0 - \lambda_0, \lambda_6 = \alpha_5 - \lambda_5, \lambda_7 = \alpha_7 \times \lambda_6, \lambda_8 = \lambda_7 \times \lambda_1,$$

$$\lambda_9 = \lambda_1 \times \lambda_4, \lambda_{10} = \alpha_8 - \lambda_9, \lambda_{11} = \lambda_3 - \lambda_{10}$$

时,怎样借助 $\alpha_1, \dots, \alpha_8$ 表达 $\beta_1, \beta_2, \dots, \beta_6$?

►26. [M21] (a) 给出对应于计算 3 次多项式的霍纳规则的多项式链。(b) 使用出现于正文中的定理 A 证明中的构造,借助于 $\beta_1, \beta_2, \beta_3, \beta_4$ 和 x , 表达 $\kappa_1, \kappa_2, \kappa_3$ 和结果多项式 $u(x)$ 。(c) 证明,当 $\beta_1, \beta_2, \beta_3, \beta_4$ 独立地取所有实数值时,(b)中得到的结果集合省略了(a)的结果集合中的某些向量。

27. [M22] 设 R 是这样一个集合,它包括使 $q_n \neq 0$ 的所有实数 $n+1$ 元组 (q_n, \dots, q_1, q_0) ;证明 R 不会至多有 n 个自由度。

28. [HM20] 证明如果 $f_0(a_1, \dots, a_s), \dots, f_s(a_1, \dots, a_s)$ 是整系数的多变量多项式,则对于所有实数 a_1, \dots, a_s ,有一非 0 的整系数多项式 $g(x_0, \dots, x_s)$ 使得 $g(f_0(a_1, \dots, a_s), \dots, f_s(a_1, \dots, a_s)) = 0$ 。(因此有 s 个参数的任何多项式链至多有 s 个自由度。)[提示:利用关于“代数相关性”的定理,它们可以在(比如说)B. L. van der Waerden 著的 *Modern Algebra* 一书中找到, Fred Blum 译 (New York: Ungar, 1949), 第 64 节。]

►29. [M20] 设 R_1, R_2, \dots, R_m 都是至多有 t 个自由度的实数 $n+1$ 元组的集合。证明并集 $R_1 \cup R_2 \cup \dots \cup R_m$ 也至多有 t 个自由度。

►30. [M28] 证明有 m_c 个链乘法和 m_p 个参数乘法的一条多项式链至多有 $2m_c + m_p + \delta_{0m_c}$ 个自由度。[提示:推广定理 M,说明头一个链乘法和每个参数乘法实质上都能引进一个新的参数到结果集合中。]

31. [M23] 证明有能力计算所有 n 次首一多项式的一条多项式链,至少有 $\lfloor n/2 \rfloor$ 次乘法和至少有 n 次加减法。

32. [M24] 找出长度为极小的多项式链,它能计算形如 $u_4 x^4 + u_2 x^2 + u_0$ 的所有多项式;并证明它的长度是极小的。

►33. [M25] 设 $n \geq 3$ 是奇数。证明有 $\lfloor n/2 \rfloor + 1$ 个乘法步的一条多项式链不可能计算所有 n 次多项式,除非它至少有 $n+2$ 个加减法步。[提示:见习题 30。]

34. [M26] 设 $\lambda_0, \lambda_1, \dots, \lambda_r$ 是一多项式链,其中所有加法和减法步都是参数步而且它至少含有一个参数乘法。假设这个方案有 m 次乘法和 $k = r - m$ 次加减法,而且由这条链计算的多项式最大次数为 n 。试证明所有可由这条链计算且其 x^n 项的系数不是 0 的多项式,都可通过另外一条链来计算,这条链至多有 m 个乘法和至多 k 个加法,并没有减法;而且新的链的最后一步应仅仅是参数乘法。

►35. [M25] 证明,使用 3 次乘法计算一般 4 次多项式的任何多项式链,必然至少有 5 次加减法。[提示:假定只有 4 次加减法,证明可用习题 34;因此这个方案必然有一种特殊形式,它不可能表示所有的 4 次多项式。]

36. [M27] 继续上一道题,证明仅仅利用 4 次乘法计算一般 6 次多项式的任何多项式链,必然至少有 7 次加减法。

37. [M21] (T. S. Motzkin) 证明系数在域 S 中的形如

$$(u_n x^n + u_{n-1} x^{n-1} + \dots + u_1 x + u_0) / (x^n + v_{n-1} x^{n-1} + \dots + v_1 x + v_0)$$

的“几乎所有”有理函数,均可以对 S 中适当的 α_j, β_j 利用方案

$$\alpha_1 + \beta_1 / (x + \alpha_2 + \beta_2 / (x + \dots + \beta_n / (x + \alpha_{n+1}) \dots))$$

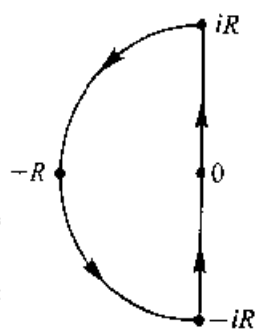


图 16 Eve 定理的证明

来求值(这连分数方案有 n 个除法和 $2n$ 个加法;所谓“几乎所有”有理函数,我们指的是除了那些其系数满足某个非平凡的多项式方程之外的所有有理函数)。试对于有理函数 $(x^2 + 10x + 29)/(x^2 + 8x + 19)$ 确定诸 α 和诸 β 。

►38. [HM32] (V. Y. Pan, 1962) 本题的目的是证明,如果不做系数的预先改写,则霍纳规则真正是最优的;如果给定变量 u_n, \dots, u_1, u_0, x 和任意常数,则计算 $u_n x^n + \dots + u_1 x + u_0$ 需要 n 次乘法和 n 次加法。考虑如以前那样的链,只是把 u_n, \dots, u_1, u_0, x 看做变量;我们可以说,例如 $\lambda_{j-1} = u_j, \lambda_0 = x$ 。为了证明霍纳规则是最好的,方便的做法是首先证明一个稍微更一般的定理:设 $A = (a_{ij}), 0 \leq i \leq m, 0 \leq j \leq n$ 是秩为 $n+1$ 的 $(m+1) \times (n+1)$ 实矩阵;并设 $B = (b_0, \dots, b_m)$ 是实向量。证明计算

$$P(x; u_0, \dots, u_n) = \sum_{i=0}^m (a_{i0} u_0 + \dots + a_{in} u_n + b_i) x^i$$

的任何多项式链至少包含 n 个链乘法。(注意,这并不意味着我们考虑的是某个固定的链,其中诸参数 α_i 被赋以依赖于 A 和 B 的值;它还意味着链和诸 α 的值都可以依赖于给定的矩阵 A 和向量 B 。不论 A, B 和诸 α_i 的值如何选定,如果不做 n 个“链步”乘法就不可能计算 $P(x; u_1, \dots, u_n)$ 。) A 的秩为 $n+1$ 的假定意味着 $m \geq n$ 。[提示:证明从任何这样的方案我们可以导出另一个方案,它有较少的链乘法而且 n 的值少了 1。]

39. [M29] (T. S. Motzkin, 1954) 证明对于 $1 < k \leq m$ 的形如

$$w_1 = x(x + \alpha_1) + \beta_1, \quad w_k = w_{k-1}(w_1 + \gamma_k x + \alpha_k) + \varepsilon_k x + \beta_k$$

的方案可用来计算实数上的所有 $2m$ 次的首一多项式,其中 α_k, β_k 是实数,而 γ_k, δ_k 是整数。(对于不同的多项式我们可能要选择不同的 $\alpha_k, \beta_k, \gamma_k$ 和 δ_k 。)只要可能就尝试令 $\delta_k = 0$ 。

40. [M41] 定理 C 中乘法次数的下限能否从 $\lfloor n/2 \rfloor + 1$ 升高到 $\lceil n/2 \rceil + 1$ (参考习题 33)?

41. [22] 证明 $(a + bi)(c + di)$ 的实部和虚部都可通过做 3 次实数乘法和 5 次实数加法得到,其中 2 次加法仅含 a 和 b 。

42. [36] (M. Paterson 和 L. Stockmeyer) (a) 证明有 $m \geq 2$ 个链乘法的一个多项式链至多有 $m^2 + 1$ 个自由度。(b) 证明对所有 $n \geq 2$ 存在 n 次多项式,其所有系数皆为 0 或 1,如果我们要求所有参数 α_j 为整数,则它不能由有少于 $\lfloor \sqrt{n} \rfloor$ 次乘法的任何多项式链来计算。(c) 证明任何 n 次整系数多项式都可由一个全整数算法来计算,这个算法至多执行 $2\lfloor \sqrt{n} \rfloor$ 次乘法,如果我们不在乎做多少次加法的话。

43. [22] 说明怎样用 $2l(n+1) - 2$ 次乘法和 $l(n+1)$ 次加法(没有除法或减法)计算 $x^n + \dots + x + 1$, 其中 $l(n)$ 是在 4.6.3 小节中研究的函数。

►44. [M25] 证明,使用 u_{n-1}, u_{n-2}, \dots 的整系数多项式参数 $\alpha_1, \alpha_2, \dots$, 可以通过 $\frac{1}{2}n + O(\log n)$ 次乘法和小于等于 $\frac{5}{4}n$ 次加法, 计算任何首一多项式 $u(x) = x^n + u_{n-1}x^{n-1} + \dots + u_0$ 。[提示:考虑 $n = 2^l$ 的情况]。

►45. [HM22] 令 (t_{ijk}) 是 $m \times n \times s$ 张量,并令 F, G, H 分别为大小为 $m \times m, n \times n, s \times s$ 的非奇异矩阵。如果对于所有的 i, j, k ,

$$T_{ijk} = \sum_{i'=1}^m \sum_{j'=1}^n \sum_{k'=1}^s F_{ii'} G_{jj'} H_{kk'} t_{i'j'k'}$$

试证明张量 (T_{ijk}) 有和 (t_{ijk}) 相同的秩。[提示:考虑当以相同的方式把 F^{-1}, G^{-1}, H^{-1} 应用于 (T_{ijk}) 时,会发生什么情况。]

46. [M28] 证明由 (x_1, x_2) 和 (y_1, y_2) 构成的双线性形式的所有 (z_1, z_2) 对可以用至多 3 个链乘法来计算。换句话说, 证明每个 $2 \times 2 \times 2$ 张量有小于等于 3 的秩。

47. [M25] 证明对所有 m, n 和 s , 存在一个 $m \times n \times s$ 的张量, 其秩至少是 $\lceil mns/(m+n+s) \rceil$ 。反之, 证明每个 $m \times n \times s$ 的张量至多有 $mns/\max(m, n, s)$ 的秩。

48. [M21] 如果 (t_{ijk}) 和 (t'_{ijk}) 分别是大小为 $m \times n \times s$ 和 $m' \times n' \times s'$ 的张量, 它们的直接和 $(t_{ijk}) \oplus (t'_{ijk}) = (t''_{ijk})$ 是如下定义的 $(m+m') \times (n+n') \times (s+s')$ 张量: 如果 $i \leq m, j \leq n, k \leq s$, 则 $t''_{ijk} = t_{ijk}$; 如果 $i > m, j > n, k > s$, 则 $t''_{ijk} = t'_{i-m, j-n, k-s}$; 否则 $t''_{ijk} = 0$ 。它们的直接积 $(t_{ijk}) \otimes (t'_{ijk}) = (t'''_{ijk})$ 是由 $t'''_{(i_1, j_1, k_1), (i_2, j_2, k_2)} = t_{i_1, j_1, k_1} t'_{i_2, j_2, k_2}$ 定义的 $mm' \times nn' \times ss'$ 张量。试推导下列上限: $\text{rank}(t'''_{ijk}) \leq \text{rank}(t_{ijk}) + \text{rank}(t'_{ijk})$, $\text{rank}(t'''_{ijk}) \leq \text{rank}(t_{ijk}) \cdot \text{rank}(t'_{ijk})$ 。

► 49. [HM25] 按照矩阵的秩为线性无关的行的极大个数这一传统定义, 证明一个 $m \times n \times 1$ 张量 (t_{jk}) 的秩和它作为一 $m \times n$ 矩阵 (t_{ij1}) 的秩相同。

50. [HM20] (S. Winograd) 设 (t_{ijk}) 是对应于 $m \times n$ 矩阵乘以 $n \times 1$ 列向量的 $mn \times n \times m$ 张量, 试证明 (t_{ijk}) 的秩是 mn 。

► 51. [M24] (S. Winograd) 试设计 2 次循环卷积的一个算法, 它使用 2 次乘法和 4 次加法, 不计对 x_i 的运算。类似地, 试设计 3 次循环卷积的算法, 用 4 次乘法和 11 次加法 (见 (69), 它在 4 次的情况下解决了类似的问题)。

52. [M25] (S. Winograd) 令 $n = n' n''$, 其中 $n' \perp n''$ 。给定 n' 和 n'' 次的循环卷积的规范方案, 其中分别用 (m', m'') 次链乘法, (p', p'') 次参数乘法及 (a', a'') 次加法。说明如何使用 $m' m''$ 次链乘法, $p' n'' + m' p''$ 次参数乘法及 $a' n'' + m' a''$ 次加法, 来构造 n 次循环卷积的规范方案。

53. [HM40] (S. Winograd) 设 ω 是单位的复数 m 次根, 并考虑一维离散傅里叶变换

$$f(s) = \sum_{t=1}^m F(t) \omega^{st}, \quad 1 \leq s \leq m$$

(a) 当 $m = p^e$ 是一奇素数的幂时, 证明对于 $0 \leq k < e$, 计算 $(p-1)p^k$ 次循环卷积的有效规范方案将导致计算 m 个复数的傅里叶变换的有效算法。试给出对于 $p=2$ 的情况的类似构造。

(b) 当 $m = m' m''$ 和 $m' \perp m''$ 时, 证明对于 m' 和 m'' 的傅里叶变换算法可结合起来以产生对于 m 个元素的一个傅里叶变换算法。

54. [M23] 定理 W 针对的是一个无限域。为了使定理 W 的证明有效, 一个有限域必须有多少个元素?

55. [HM22] 当 P 是任意一个 $n \times n$ 矩阵时, 确定张量 (74) 的秩。

56. [M32] (V. Strassen) 证明, 对于 $1 \leq k \leq s$, 计算一组二次型 $\sum_{i=1}^n \sum_{j=1}^n \tau_{ijk} x_i x_j$ 的任何多项式链必定至少使用 $\frac{1}{2} \text{rank}(\tau_{ijk} + \tau_{jik})$ 个链乘法。[提示: 证明链乘法的极小个数是在所有张量 (t_{ijk}) 上取的 (t_{ijk}) 的极小秩, 使得对所有 $i, j, k, t_{ijk} + t_{jik} = \tau_{ijk} + \tau_{jik}$ 。] 利用这个事实证明, 计算对应于一个张量 (t_{ijk}) 的一组双线性形式 (47) 的任何多项式链, 无论规范的或异常的, 必定至少使用 $\frac{1}{2} \text{rank}(t_{ijk})$ 个链乘法。

57. [M20] 证明快速傅里叶变换可以用来计算两个给定的 n 次多项式乘积 $x(u)y(u)$ 的系数, 并使用 $O(n \log n)$ 次复数 (精确) 加法和乘法运算。[提示: 考虑系数的傅里叶变换的乘积。]

58. [HM28] (a) 证明多项式乘法张量 (55) 的任何实现 (A, B, C) 必定有下列任何性质: A 的三行的任何非零线性组合必定是至少具有四个非零元素的一个向量, B 的四行的任何非零线性组合必定至少有两个非零元素。(b) 只使用 0, +1 和 -1 作为元素, 求 (55) 的一个实现 $(A, B,$

C), 其中 $r=8$ 。试使用尽可能多的 0。

►59. [M40] (H. J. Nussbaumer, 1980) 正文定义两个序列 $(x_0, x_1, \dots, x_{n-1})$ 和 $(y_0, y_1, \dots, y_{n-1})$ 的循环卷积为序列 $(z_0, z_1, \dots, z_{n-1})$, 其中 $z_k = x_0 y_k + \dots + x_k y_0 + x_{k+1} y_{n-1} + \dots + x_{n-1} y_{k+1}$ 。我们类似地定义反循环卷积, 但用

$$z_k = x_0 y_k + \dots + x_k y_0 - (x_{k+1} y_{n-1} + \dots + x_{n-1} y_{k+1})$$

当 n 是 2 的幂时, 构造在整数上的循环和反循环卷积的有效算法。你的算法应当全部使用整数, 而且它们应当至多执行 $O(n \log n)$ 次乘法 and 至多 $O(n \log n \log \log n)$ 次加法或减法或偶数除以 2 的除法。[提示: 使用 (59), 阶为 $2n$ 的一个循环卷积可归结为阶为 n 的循环和反循环卷积。]

60. [M27] (V. Y. Pan) $(m \times n)$ 乘以 $(n \times s)$ 的矩阵乘法问题对应于一个 $mn \times ns \times sm$ 的张量 $(t_{i,j,k})_{(i,j,k) \in S}$, 其中 $t_{i,j,k} = 1$ 当且仅当 $i' = i, j' = j$ 及 $k' = k$ 。这个张量 $T(m, n, s)$ 的秩是使得 $a_{ij'l}, b_{jk'l}, c_{kl'l}$ 存在并满足

$$\sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n \\ 1 \leq k \leq s}} x_{ij} y_{jk} z_{ki} = \sum_{1 \leq l \leq r} \left(\sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} a_{ij'l} x_{ij} \right) \left(\sum_{\substack{1 \leq j \leq n \\ 1 \leq k \leq s}} b_{jk'l} y_{jk} \right) \left(\sum_{\substack{1 \leq k \leq s \\ 1 \leq i \leq m}} c_{kl'l} z_{ki} \right)$$

的最小的 r 。设 $M(n)$ 是 $T(n, n, n)$ 的秩。本题的目的是剖析这样一个三线性表示的对称性, 并得到当 $m = n = s = 2^\nu$ 时整数上的矩阵乘法的有效实现。为方便起见, 我们把下标 $\{1, \dots, n\}$ 分成两个子集 $O = \{1, 3, \dots, n-1\}$ 和 $E = \{2, 4, \dots, n\}$, 每个子集含有 ν 个元素。我们用规则如果 $i \in O$, 则 $\bar{i} = i+1$; 如果 $i \in E$, 则 $\bar{i} = i-1$ 来建立 O 与 E 之间的一一对应。于是对所有下标 i , 我们有 $\bar{\bar{i}} = i$ 。

a) 恒等式

$$abc + ABC = (a+A)(b+B)(c+C) - (a+A)bC - A(b+B)c - aB(c+C)$$

意味着

$$\sum_{1 \leq i, j, k \leq n} x_{ij} y_{jk} z_{ki} = \sum_{(i, j, k) \in S} (x_{ij} + x_{\bar{i}\bar{j}})(y_{jk} + y_{\bar{j}\bar{k}})(z_{ki} + z_{\bar{k}\bar{i}}) = \Sigma_1 - \Sigma_2 - \Sigma_3$$

其中 $S = E \times E \times E \cup E \times E \times O \cup E \times O \times E \cup O \times E \times E$ 是至多含一个奇下标的所有三元组的集合, Σ_1 是对于 $(i, j, k) \in S$ 的形如 $(x_{ij} + x_{\bar{i}\bar{j}})y_{jk}z_{ki}$ 的所有项之和; 而 Σ_2 和 Σ_3 类似地是项 $x_{\bar{i}\bar{j}}(y_{jk} + y_{\bar{j}\bar{k}})z_{ki}$, $x_{ij}y_{\bar{j}\bar{k}}(z_{ki} + z_{\bar{k}\bar{i}})$ 之和。显然 S 有 $4\nu^3 = \frac{1}{2}n^3$ 项。证明 $\Sigma_1, \Sigma_2, \Sigma_3$ 中的每一个都可以实现为 $3\nu^2$ 个三线性项之和; 而且如果把形如 (i, i, \bar{i}) 和 (i, \bar{i}, i) 及 (\bar{i}, i, i) 的 3ν 个三元组从 S 中删去, 则我们可以以这样一个方式修改 $\Sigma_1, \Sigma_2, \Sigma_3$, 使恒等式仍然有效, 而无须增加任何新的三线性项。于是

当 n 为偶数时, $M(n) \leq \frac{1}{2}n^3 + \frac{9}{4}n^2 - \frac{3}{2}n$ 。

b) 应用 a) 的方法证明, 两个大小为 $m \times n \times s$ 的独立矩阵的乘法问题可以用 $mns + mn + ns + sm$ 次非交换乘法来实现。

61 [M26] 令 (t_{ijk}) 是一个任意域上的张量, 我们把 $\text{rank}_d(t_{ijk})$ 定义为使得有形如

$$\sum_{l=1}^r a_{il}(u) b_{jl}(u) c_{kl}(u) = t_{ijk} u^d + O(u^{d+1})$$

的一个实现的 r 的极小值, 其中 $a_{il}(u), b_{jl}(u), c_{kl}(u)$ 是在这个域上 u 的多项式, 于是 rank_0 是一个张量的通常的秩。证明

a) $\text{rank}_{d+1}(t_{ijk}) \leq \text{rank}_d(t_{ijk})$;

b) $\text{rank}(t_{ijk}) \leq \binom{d+2}{2} \text{rank}_d(t_{ijk})$;

c) 在习题 48 的意义下, $\text{rank}_d((t_{ijk}) \odot (t'_{ijk})) \leq \text{rank}_d(t_{ijk}) + \text{rank}_d(t'_{ijk})$;

d) $\text{rank}_{d+d'}((t_{ijk}) \odot (t'_{ijk})) \leq \text{rank}_d(t_{ijk}) + \text{rank}_{d'}(t'_{ijk})$;

e) $\text{rank}_{d+d'}((t_{ijk}) \otimes (t'_{ijk})) \leq \text{rank}_d(r(t'_{ijk}))$, 其中 $r = \text{rank}_d(t_{ijk})$, rT 表示 T 的 r 个副本的直接和 $T \oplus \cdots \oplus T$ 。

62. [M24] 用 $\text{rank}(t_{ijk})$ 表示的 (t_{ijk}) 的边界秩, 是 $\min_{d \geq 0} \text{rank}_d(t_{ijk})$, 其中 rank_d 在习题 61 中定义。

证明张量 $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ 在每个域上有秩 3 但边界秩为 2。

63. [HM30] 设 $T(m, n, s)$ 是如习题 60 那样的矩阵乘法的张量, 并设 $M(N)$ 是 $T(N, N, N)$ 的秩。

a) 证明 $T(m, n, s) \otimes T(M, N, S) = T(mM, nN, sS)$ 。

b) 证明 $\text{rank}_d(T(mN, nN, sN)) \leq \text{rank}_d(M(N)T(m, n, s))$ (见习题 61e))。

c) 如果 $T(m, n, s)$ 有秩小于等于 r , 证明当 $N \rightarrow \infty$ 时, $M(N) = O(N^{w(m, n, s, r)})$, 其中

$$\omega(m, n, s, r) = 3 \log r / \log mns$$

d) 如果 $T(m, n, s)$ 有边界秩小于等于 r , 证明 $M(N) = O(N^{w(m, n, s, r)}(\log N)^2)$ 。

64. [M30] (A. Schönhage) 证明 $\text{rank}_2(T(3, 3, 3)) \leq 21$, 所以 $M(N) = O(N^{2.78})$ 。

► 65. [M27] (A. Schönhage) 证明 $\text{rank}_2(T(m, 1, n) \oplus T(1, (m-1)(n-1), 1)) = mn + 1$ 。提示: 考虑三线性形式, 当 $\sum_{i=1}^m X_i = \sum_{j=1}^n Y_j = 0$ 时,

$$\sum_{i=1}^m \sum_{j=1}^n (x_i + uX_{ij})(y_j + uY_{ij})(Z + u^2z_{ij}) - (x_1 + \cdots + x_m)(y_1 + \cdots + y_n)Z$$

66. [HM33] 我们现在能够使用习题 65 的结果来加强习题 63 的渐近上限。

a) 证明极限 $\omega = \lim_{n \rightarrow \infty} \log M(n) / \log n$ 存在。

b) 证明 $(mns)^{\omega/3} \leq \text{rank}(T(m, n, s))$ 。

c) 令 t 是张量 $T(m, n, s) \oplus T(M, N, S)$ 。证明 $(mns)^{\omega/3} + (MNS)^{\omega/3} \leq \text{rank}(t)$ 。提示: 考虑 t 同它自己的直接积。

d) 因此 $16^{\omega/3} + 9^{\omega/3} \leq 17$, 而且有 $\omega < 2.55$ 。

67. [HM40] (D. Coppersmith 和 S. Winograd) 通过推广习题 65 和 66, 我们可以得到对 ω 的甚至更好的上限。

a) 在引理 T 的记号之下, 如果 $\text{rank}(t_{i(jk)}) = m$, $\text{rank}(t_{j(ki)}) = n$ 和 $\text{rank}(t_{k(ij)}) = s$, 则说张量 (t_{ijk}) 是非退化的。证明对于 $m \times n \times s$ 矩阵乘法的张量 $T(m, n, s)$ 是非退化的。

b) 证明非退化张量的直接和是非退化的。

c) 一个长度为 r 具有实现 (A, B, C) 的 $m \times n \times s$ 的张量 t 说是可改进的, 如果对于 $1 \leq i \leq m$ 和 $1 \leq j \leq n$, 它是非退化的而且有非零元素 d_1, \dots, d_r 使得 $\sum_{l=1}^r a_{il} b_{jl} d_l = 0$ 。证明在这样的情况下 $t \oplus T(1, q, 1)$ 有小于等于 r 的边界秩, 其中 $q = r - m - n$ 。提示: 有 $q \times r$ 的矩阵 V 和 W 使得对于所有有关的 i 和 j , $\sum_{l=1}^r v_{il} b_{jl} d_l = \sum_{l=1}^r a_{il} w_{jl} d_l = 0$ 和 $\sum_{l=1}^r v_{il} w_{jl} d_l = \delta_{ij}$ 。

d) 说明习题 65 的结果为什么是 c) 的一个特殊情况。

e) 证明 $\text{rank}(T(m, n, s)) \leq r$ 意味着

$$\text{rank}_2(T(m, n, s) \oplus T(1, r - n(m + s - 1), 1)) \leq r + n$$

f) 因此, 对于所有 $n > 1$, ω 严格地小于 $\log M(n) / \log n$ 。

g) 把 c) 推广到 (A, B, C) 仅在习题 61 的较弱的意义下实现 t 的情况。

h) 由 d) 我们有 $\text{rank}(T(3, 1, 3) \oplus T(1, 4, 1)) \leq 10$; 因此由习题 61d) 我们也有 $\text{rank}(T(9, 1, 9) \oplus 2T(3, 4, 3) \oplus T(1, 16, 1)) \leq 100$ 。证明如果我们简单地删除对应于 $T(1, 16, 1)$ 的 $16 + 16$ 变量的 A

和 B 的行, 我们得到可改进的 $T(9, 1, 9) \oplus 2T(3, 4, 3)$ 的一个实现。因此事实上我们有 $\text{rank}(T(9, 1, 9) \oplus 2T(3, 4, 3) \oplus T(1, 34, 1)) \leq 100$ 。

i) 推广习题 66c), 证明

$$\sum_{p=1}^r (m_p n_p s_p)^{\omega/3} \leq \text{rank}(\bigoplus_{p=1}^r T(m_p, n_p, s_p))$$

j) 因此 $\omega < 2.5$ 。

68. [M45] 是否有一个方法通过少于 $n-1$ 个乘法和 $2n-4$ 个加法来计算多项式

$$\sum_{1 \leq i < j \leq n} x_i x_j = x_1 x_2 + \cdots + x_{n-1} x_n$$

(有 $\binom{n}{2}$ 项)?

► 69. [HM27] (V. Strassen, 1973) 证明一个 $n \times n$ 矩阵的行列式 (31) 可以通过做 $O(n^5)$ 次乘法和 $O(n^5)$ 次加法或减法, 不做除法来求值。[提示: 考虑 $\det(I + Y)$, 其中 $Y = X - I$ 。]

► 70. [HM25] 一个矩阵 X 的特征多项式 $f_X(\lambda)$ 定义为 $\det(\lambda I - X)$ 。证明, 如果 $X = \begin{pmatrix} x & u \\ v & Y \end{pmatrix}$, 其中 X, u, v 和 Y 分别有 $n \times n, 1 \times (n-1), (n-1) \times 1$ 和 $(n-1) \times (n-1)$ 的大小, 我们有

$$f_X(\lambda) = f_Y(\lambda) \left(\lambda - x - \frac{uv}{\lambda} - \frac{uYv}{\lambda^2} - \frac{uY^2v}{\lambda^3} - \cdots \right)$$

证明这个关系允许我们以大约 $\frac{1}{4}n^4$ 次乘法, $\frac{1}{4}n^4$ 次加减法, 不做除法来计算 f_X 的系数。提示: 使用恒等式

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & D \end{pmatrix} \begin{pmatrix} A - BD^{-1}C & B \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ D^{-1}C & I \end{pmatrix}$$

当 D 非奇异时, 它对大小为 $l \times l, l \times m, m \times l$ 和 $m \times m$ 的任何矩阵 A, B, C 和 D 成立。

► 71. [HM30] 四项链和多项式链一样, 只是它除了允许加法, 减法和乘法之外还允许除法。证明如果 $f(x_1, \dots, x_n)$ 可以通过有 m 次链乘法和 d 次除法的一个四项链计算, 则对于 $1 \leq k \leq n$, $f(x_1, \dots, x_n)$ 和所有 n 个它的偏微分 $\partial f(x_1, \dots, x_n) / \partial x_k$ 都可通过至多有 $3m + d$ 次链乘法和 $2d$ 次除法的单个四项链来计算。(结果, 例如, 为计算一个矩阵的行列式的任何有效算法都导致计算它的所有余子式的有效方法, 因此也导致计算逆矩阵的一个有效方法。)

72. [M48] 是否有可能在有限步之内确定在 (比如说) 有理数域上任何给定张量 (t_{ijk}) 的秩?

73. [HM25] (J. Morgenstern, 1973) 证明: 对于离散傅里叶变换 (37) 的任何多项式链, 至少有 $\frac{1}{2}m_1 \cdots m_n \lg m_1 \cdots m_n$ 次加减法, 如果没有链乘法且每个参数乘法都是对于有 $|a_j| \leq 1$ 的一个复值常数进行的。提示: 考虑由头 k 步计算的线性变换的矩阵。

74. [HM33] (A. Nozaki, 1978) 多项式求值理论的大部分涉及链乘法的限, 但是同非整常数的乘法也是必不可少的。本题的目的是建立关于常数的适当理论。如果有整数 (k_1, \dots, k_r) 使得 $\gcd(k_1, \dots, k_r) = 1$ 和 $k_1 v_1 + \cdots + k_r v_r$ 是全整数的向量, 我们就说实向量 v_1, \dots, v_r 是 Z 有关的。如果没有这样的 (k_1, \dots, k_r) 存在, 则向量 v_1, \dots, v_r 是 Z 无关的。

a) 证明如果 $r \times s$ 矩阵 V 的诸列是 Z 无关的, 则当 U 是任何 $s \times s$ 单模矩阵 (其行列式为 ± 1 的整数矩阵) 时, VU 的诸列也是 Z 无关的。

b) 令 V 是有 Z 无关列的一个 $r \times s$ 矩阵。证明由输入 x_1, \dots, x_r 来计算 Vx 的元素的一个多项式链至少需要 s 次乘法, 其中 $x = (x_1, \dots, x_r)^T$ 。

c) 令 V 是有 Z 无关的 s 个列的 $r \times t$ 矩阵, 证明从输入 x_1, \dots, x_t 来计算 V_i 的元素的一个多项链至少需要 s 次乘法, 其中 $x = (x_1, \dots, x_t)^T$.

d) 说明如何只使用一次乘法, 来从 x 和 y 计算值偶 $\lfloor x/2 + y, x + y/3 \rfloor$. 尽管为计算值偶 $\lfloor x/2 + y, x + y/2 \rfloor$ 需要两次乘法。

* 4.7 幂级数的操作

如果给定两个幂级数

$$U(z) = U_0 + U_1 z + U_2 z^2 + \dots, \quad V(z) = V_0 + V_1 z + V_2 z^2 + \dots \quad (1)$$

其系数属于一个域, 则我们可以形成它们的和, 它们的积, 以及有时它们的商, 等等, 以得到新的幂级数。多项式显然是幂级数的特殊情况, 它仅有有限个项。

当然, 在一台计算机内仅仅能够表示和存储有限个项, 所以提出幂级数算术在计算机上是否可能的问题是有意义的; 而如果可能的话, 它是怎样不同于多项式算术的? 答案是, 我们仅仅处理幂级数的头 N 个系数, 其中 N 是一个参数, 原则上它可能任意地大; 我们实质上做的不是通常的多项式算术, 而是模 z^N 下的多项式算术, 而且这通常导致一种略有不同的观点。此外, 像“反演”这样一些特殊运算, 可以对幂级数实施但却不能对多项式实施, 因为多项式在这些运算之下不是封闭的。

幂级数的处理对数值分析有好多应用, 但也许它最大的用途是确定渐近展开(如同我们在 1.2.11.3 小节中已经看到的那样), 或者用来计算由某些生成函数定义的量。后一种应用需要精确地(而不是通过浮点算术)来计算系数。这一节的所有算法, 除去明显的例外, 可以仅用有理运算进行, 所以在需要时可以使用 4.5.1 小节的一些技术来得出精确的结果。

当然, $W(z) = U(z) \pm V(z)$ 的计算是平凡的, 因为对于 $n = 0, 1, 2, \dots$, 我们有 $W_n = [z^n] W(z) = U_n \pm V_n$ 。利用熟知的卷积规则也容易计算 $W(z) = U(z)V(z)$:

$$W_n = \sum_{k=0}^n U_k V_{n-k} = U_0 V_n + U_1 V_{n-1} + \dots + U_n V_0 \quad (2)$$

当 $V_0 \neq 0$ 时, 通过交换(2)中的 U 和 W , 可以得到商 $W(z) = U(z)/V(z)$; 我们得到规则

$$W_n = \left(U_n - \sum_{k=0}^{n-1} W_k V_{n-k} \right) / V_0 = \\ (U_n - W_0 V_n - W_1 V_{n-1} - \dots - W_{n-1} V_1) / V_0 \quad (3)$$

对于诸 W 的这一递推关系, 使我们容易逐次地确定 W_0, W_1, W_2, \dots , 而在计算出 W_{n-1} 之前无须输入 U_n 和 V_n 。我们说具有这种性质的幂级数操作算法是 在线的 。通过一个在线的算法可确定结果的 N 个系数 W_0, W_1, \dots, W_{N-1} 而无须预先知道 N , 所以在原则上有可能无限地运行这个算法并计算出整个幂级数; 或者运行它直到满足某个条件为止。(“在线”的反面是“离线”。)

如果系数 U_k 和 V_k 都是整数但 W_k 不是, 则递推关系(3)就包含了分数计算。通

过习题2中介绍的全整数方法可以避免这一点。

现在我们考虑如何计算 $W(z) = V(z)^\alpha$, 其中 α 是一个“任意”的次幂。例如, 我们可以取 $\alpha = \frac{1}{2}$ 来计算 $V(z)$ 的平方根, 或者我们可以求 $V(z)^{-10}$ 或者甚至 $V(z)^\pi$ 。

如果 V_m 是 $V(z)$ 的头一个非0系数, 则我们有

$$\begin{aligned} V(z) &= V_m z^m (1 + (V_{m+1}/V_m)z + (V_{m+2}/V_m)z^2 + \cdots) \\ V(z)^\alpha &= V_m^\alpha z^{\alpha m} (1 + (V_{m+1}/V_m)z + (V_{m+2}/V_m)z^2 + \cdots)^\alpha \end{aligned} \quad (4)$$

当且仅当 αm 是一个非负整数时, 这是一个幂级数。从(4)我们可以看出, 计算一般乘方的问题可归结为 $V_0 = 1$ 的情况; 然后问题就成为计算

$$W(z) = (1 + V_1 z + V_2 z^2 + V_3 z^3 + \cdots)^\alpha \quad (5)$$

的系数。显然, $W_0 = 1^\alpha = 1$ 。

求(5)的系数的明显方法是使用二项式定理, 即等式 1.2.9-(19), 或者(如果 α 是一个正整数)可像 4.6.3 小节中那样重复地进行平方。但是欧拉发现了得到幂级数的乘方的一个更简单和更有效的方法 [*Introductio in Analysin Infinitorum* 1(1748), § 76]: 如果 $W(z) = V(z)^\alpha$, 则通过微分法我们有

$$W_1 + 2W_2 z + 3W_3 z^2 + \cdots = W'(z) = \alpha V(z)^{\alpha-1} V'(z) \quad (6)$$

因此

$$W'(z)V(z) = \alpha W(z)V'(z) \quad (7)$$

如果现在等置(7)中 z^{n-1} 的系数, 则我们求得

$$\sum_{k=0}^n k W_k V_{n-k} = \alpha \sum_{k=0}^n (n-k) W_k V_{n-k} \quad (8)$$

这给了我们所有 $n \geq 1$ 都成立的计算规则

$$\begin{aligned} W_n &= \sum_{k=1}^n \left(\left(\frac{\alpha+1}{n} \right) k - 1 \right) V_k W_{n-k} = \\ &= ((\alpha+1-n)V_1 W_{n-1} + (2\alpha+2-n)V_2 W_{n-2} + \cdots + n\alpha V_n W_0)/n \end{aligned} \quad (9)$$

式(9)导出了一个简单在线算法, 它使用大约 $2n$ 次乘法计算第 n 个系数, 因而可以逐次地确定 W_1, W_2, \dots 。注意特殊情况 $\alpha = -1$, 这时(9)变成(3)的特殊情况 $U(z) = V_0 = 1$ 。

当 f 是满足一个简单微分方程的任意函数时, 一个类似的技术可用于构造 $f(V(z))$ (例如见习题4)。一个比较直截了当的“幂级数方法”通常被用于获得微分方程的解; 这种技术在几乎所有微分方程的课本中都有讲述。

级数的反演 也许最值得关注的幂级数变换, 要推“级数的反演”了。这个问题是对 t 解方程

$$z = t + V_2 t^2 + V_3 t^3 + V_4 t^4 + \cdots \quad (10)$$

得到幂级数

$$t = z + W_2 z^2 + W_3 z^3 + W_4 z^4 + \cdots \quad (11)$$

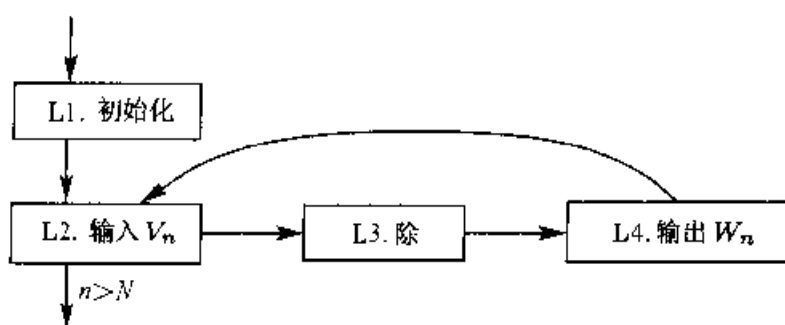


图 17 用算法 L 进行幂级数反演

的系数。

已经知道有好多个有趣的方法来实现这样的反演。我们可以说“经典的”方法是以拉格朗日的著名的求逆公式 [*Mémoires Acad. Royale des Sciences et Belles-Lettres de Berlin* 24 (1768), 251~326] 为基础, 它指出

$$W_n = \frac{1}{n} [t^{n-1}] (1 + V_2 t + V_3 t^2 + \cdots)^{-n} \quad (12)$$

例如, 我们有 $(1-t)^{-5} = \binom{4}{4} + \binom{5}{4}t + \binom{6}{4}t^2 + \cdots$; 因此在 $z = t - t^2$ 的反演中第 5 项系数 W_5 等于 $\binom{8}{4}/5 = 14$, 这同 2.3.4.4 小节中枚举二叉树的公式相符。

关系式(12)有一个简单的算法证明(见习题 16), 它表明如果对 $n = 1, 2, 3, \cdots$ 逐次计算负次幂 $(1 + V_2 t + V_3 t^2 + \cdots)^{-n}$, 则我们可以反演级数(10)。这一思想的直截了当的应用将导致一个在线反演算法, 它使用近似于 $N^3/2$ 次乘法来求 N 个系数, 但等式(9)使得有可能只处理 $(1 + V_2 t + V_3 t^2 + \cdots)^{-n}$ 的头 n 个系数, 得到只要求大约 $N^3/6$ 次乘法的一个在线算法。

算法 L (拉格朗日幂级数反演) 这个在线算法对于 $n = 2, 3, 4, \cdots, N$, 输入(10)中 V_n 的值并输出(11)中 W_n 的值。(数 N 不必事先确定; 可以使用任何需要的终止准则。)

L1. [初始化] 置 $n \leftarrow 1, U_0 \leftarrow 1$ 。(关系式

$$(1 + V_2 t + V_3 t^2 + \cdots)^{-n} = U_0 + U_1 t + \cdots + U_{n-1} t^{n-1} + O(t^n) \quad (13)$$

将在整个算法的过程中保持成立。)

L2. [输入 V_n] n 加 1。如果 $n > N$, 此算法终止; 否则输入下个系数 V_n 。

L3. [除] 对于 $k = 1, 2, \cdots, n-2$ (按此顺序), 置 $U_k \leftarrow U_k - U_{k-1} V_2 - \cdots - U_1 V_k - U_0 V_{k+1}$; 然后置 $U_{n-1} \leftarrow -2U_{n-2} V_2 - 3U_{n-3} V_3 - \cdots - (n-1)U_1 V_n - nU_0 V_n$ 。(由此我们已经以 $V(z)/z$ 来除 $U(z)$; 参见(3)和(9)。)

L4. [输出 W_n] 输出 U_{n-1}/n (它就是 W_n) 并返回 L2。 ■

当应用于例子 $z = t - t^2$ 时, 算法 L 计算

n	V_n	U_0	U_1	U_2	U_3	U_4	W_n
1	1	1					1
2	-1	1	2				1
3	0	1	3	6			2
4	0	1	4	10	20		5
5	0	1	5	15	35	70	14

习题 8 表明,对算法 L 稍加修改后,只须稍加努力就可解决更为一般的问题。

我们考虑解关于 t 的方程

$$U_1 z + U_2 z^2 + U_3 z^3 + \cdots = t + V_2 t^2 + V_3 t^3 + \cdots \quad (14)$$

得到幂级数

$$t = W_1 z + W_2 z^2 + W_3 z^3 + W_4 z^4 + \cdots \quad (15)$$

的系数。方程(10)是 $U_1=1, U_2=U_3=\cdots=0$ 的特殊情况。如果 $U_1 \neq 0$,我们可以假定 $U_1=1$,如果以 $(U_1 z)$ 代替 z 的话;但我们将考虑一般方程(14),因为 U_1 可能等于 0。

算法 T(一般幂级数的反演) 这个在线算法对于 $n=1, 2, 3, \cdots, N$ 输入(14)中 U_n 和 V_n 的值,并输出(15)中 W_n 的值。在这些计算中,使用一个辅助矩阵 $T_{mn}, 1 \leq m \leq n \leq N$ 。

T1. [初始化] 置 $n \leftarrow 1$, 设头两个输入(即 U_1 和 V_1)分别存于 T_{11} 和 V_1 中(我们必定有 $V_1=1$)。

T2. [输出 W_n] 输出 T_{1n} 的值(它是 W_n)。

T3. [输入 U_n, V_n] n 增加 1。如果 $n > N$, 算法终止;否则把其次的两个输入值(即 U_n 和 V_n)存入 T_{1n} 和 V_n 中。

T4. [乘] 置

$$T_{mn} \leftarrow T_{11} T_{m-1, n-1} + T_{12} T_{m-1, n-2} + \cdots + T_{1, n-m+1} T_{m-1, m-1}$$

且对于 $2 \leq m \leq n$, 置 $T_{1n} \leftarrow T_{1n} - V_m T_{mn}$ 。(在这步之后,对于 $1 \leq m \leq n$, 我们有

$$t^m = T_{mm} z^m + T_{m, m+1} z^{m+1} + \cdots + T_{mn} z^n + O(z^{n+1}) \quad (16)$$

对于 $m \geq 2$ 用归纳法,容易验证(16),而且当 $m=1$ 时,我们由(14)和(16)有 $U_n = T_{1n} + V_2 T_{2n} + \cdots + V_n T_{nn}$ 。)返回 T2。 **I**

等式(16)说明了这个算法的机制,它是由 Henry C. Thacher, Jr. 给出的[CACM 9 (1966), 10~11]。运行时间实质上 and 算法 L 一样但却需要颇多的存储空间。习题 9 讨论了这个算法的一个例子。

幂级数反演的另一个方法由 R. P. Brent 和 H. T. Kung(孔祥重)提出[JACM 25 (1978), 581~595],它所依据的事实是:在实数上用来求方程的根的标准迭代过程,也可用于幂级数方程。特别是,给定在接近 t 处有良好特性的函数 $f(x)$,我们即可考虑

近似地计算使得 $f(t)=0$ 的实数 t 的牛顿方法:如果 x 是 t 的一个好的近似,则 $\phi(x) = x - f(x)/f'(x)$ 将甚至更好,因为如果写 $x = t + \epsilon$,我们就有 $f(x) = f(t) + \epsilon f'(t) + O(\epsilon^2)$, $f'(x) = f'(t) + O(\epsilon)$; 因此 $\phi(x) = t + \epsilon - (0 + \epsilon f'(t) + O(\epsilon^2))/(f'(t) + O(\epsilon)) = t + O(\epsilon^2)$ 。对于幂级数应用这一思想,令 $f(x) = V(x) \cdot U(z)$, 其中 U 和 V 都是方程(14)中的幂级数。我们希望求 z 的幂级数 t , 使得 $f(t)=0$ 。令 $x = W_1 z + \cdots + W_{n-1} z^{n-1} = t + O(z^n)$ 是 t 的 n 阶近似; 则 $\phi(x) = x - f(x)/f'(x)$ 将是 $2n$ 阶近似, 因为对于这个 f 和 t , 牛顿方法的假定成立。

换句话说,我们可以使用下述过程:

算法 N(用牛顿方法求一般幂级数的反演) 这一“半在线”算法对于 $2^k \leq n < 2^{k+1}$ 输入(14)中 U_n 和 V_n 的值, 然后对于 $2^k \leq n < 2^{k+1}$ 输出(15)中 W_n 的值。由此对于 $k = 0, 1, 2, \dots, K$, 它一次产生成批的答案, 每批 2^k 个。

N1. [初始化] 置 $N \leftarrow 1$ (我们将有 $N = 2^k$)。输入头两个系数 U_1 和 V_1 (其中 $V_1 = 1$), 并置 $W_1 \leftarrow U_1$ 。

N2. [输出] 对 $N \leq n < 2N$ 输出 W_n 。

N3. [输入] 置 $N \leftarrow 2N$ 。如果 $N > 2^K$, 则算法终止; 否则对于 $N \leq n < 2N$ 输入值 U_n 和 V_n 。

N4. [牛顿步] 在幂级数

$$U_1 z + \cdots + U_{2N-1} z^{2N-1} - V(W_1 z + \cdots + W_{N-1} z^{N-1}) = R_0 z^N + R_1 z^{N+1} + \cdots + R_{N-1} z^{2N-1} + O(z^{2N})$$

$$V'(W_1 z + \cdots + W_{N-1} z^{N-1}) = Q_0 + Q_1 z + \cdots + Q_{N-1} z^{N-1} + O(z^N)$$

中(其中 $V(z) = x + V_2 z^2 + \cdots$, $V'(x) = 1 + 2V_2 x + \cdots$), 使用幂级数复合算法(见习题 11), 计算系数 Q_j 和 R_j ($0 \leq j < N$)。然后把 W_N, \dots, W_{2N-1} 置入下列幂级数的系数中:

$$\frac{R_0 + R_1 z + \cdots + R_{N-1} z^{N-1}}{Q_0 + Q_1 z + \cdots + Q_{N-1} z^{N-1}} = W_N + \cdots + W_{2N-1} z^{N-1} + O(z^N)$$

并返回 N2。 ■

这个算法为得到多至 $N = 2^K$ 个系数所需运行时间是 $T(N)$, 其中

$$T(2N) = T(N) + (\text{做 N4 步的时间}) + O(N) \quad (17)$$

用于复合幂级数的简捷算法和 N4 步中的除法将花费阶为 N^3 的步骤数, 所以算法 N 要比算法 T 运行得慢些。然而, Brent 和 Kung 已经找到一个方法, 用 $O(N \log N)^{3/2}$ 次算术运算来进行所要求的幂级数的复合, 习题 6 给出一个甚至更快的进行除法的方法; 因此(17)表明, 当 $N \rightarrow \infty$ 时幂级数反演可以只须做 $O(N \log N)^{3/2}$ 次运算来实现。(另一方面, 比例性常数要求在算法 L 和 T 输给这种“高速”的方法之前 N 必须很大。)

历史记录: J. N. Bramhall 和 M. A. Chapple 在 CACM 4 (1961), 317~318, 503 中发表了头一个进行幂级数反演的 $O(N^3)$ 方法, 它是一个离线算法, 与习题 16 中的方法基本相同, 它的运行时间与算法 L 和 T 几乎相同。

级数的迭代 如果要研究一个迭代过程 $x_n \leftarrow f(x_{n-1})$ 的特性,首先要研究一个给定的函数 f 同它本身的 n 重组合,即 $x_n = f(f(\cdots f(x_0)\cdots))$ 。我们定义 $f^{[0]}(x) = x$ 和 $f^{[n]}(x) = f(f^{[n-1]}(x))$,使得对所有整数 $m, n \geq 0$,

$$f^{[m+n]}(x) = f^{[m]}(f^{[n]}(x)) \quad (18)$$

在许多场合它也使得当 n 是负整数时,记号 $f^{[n]}(x)$ 有意义,即,如果 $f^{[n]}$ 和 $f^{[-n]}$ 是逆函数,则 $x = f^{[n]}(f^{[-n]}(x))$;如果逆函数是惟一的,则(18)对所有整数 m 和 n 成立。幂级数的反演实质上是寻找逆幂级数 $f^{[-1]}(x)$ 的运算;例如,方程(10)和(11)实质上指出 $z = V(W(z))$ 和 $t = W(V(t))$,所以 $W = V^{[-1]}$ 。

假设给定两个幂级数 $V(z) = z + V_2 z^2 + \cdots$ 和 $W(z) = z + W_2 z^2 + \cdots$ 使得 $W = V^{[-1]}$ 。设 u 是任何非零的常数,并考虑函数

$$U(z) = W(uV(z)) \quad (19)$$

容易看出 $U(U(z)) = W(u^2 V(z))$,而且一般地对所有整数 n ,

$$U^{[n]}(z) = W(u^n V(z)) \quad (20)$$

因此对于 n 次迭代量 $U^{[n]}$,我们有一个简单的表达式,对所有 n ,它可以用大约同样的工作量来计算。此外,我们甚至可以用(20)对非整数的 n 值定义 $U^{[n]}$;例如“半迭代量” $U^{[1/2]}$ 是使得 $U^{[1/2]}(U^{[1/2]}(z)) = U(z)$ 的一个函数(通过用 \sqrt{u} 和 $-\sqrt{u}$ 作为(20)中 $u^{1/2}$ 的值,可以得到两个这样的函数 $U^{[1/2]}$)。

从 V 和 u 开始,然后定义 U ,我们即得到了(20)中的简单式子。但是在实践中,一般要走另一条路:从某个给定的函数 U 开始,我们要找 V 和 u 使得(19)成立,即使得

$$V(U(z)) = uV(z) \quad (21)$$

这样一个函数 V 叫做 U 的 Schröder 函数,因为它是由 Ernst Schröder 在 *Math. Annalen* 3 (1871), 296~322 中给出的。我们现在来考察对一个给定的幂级数 $U(z) = U_1 z + U_2 z^2 + \cdots$ 求 Schröder 函数 $V(z) = z + V_2 z^2 + \cdots$ 的问题。显然,如果要使(21)成立则 $u = U_1$ 。

以 $u = U_1$ 展开(21)并等置 z 的系数导致一等式序列,这些等式从下列式子开始:

$$U_1^2 V_2 + U_2 = U_1 V_2$$

$$U_1^3 V_3 + 2U_1 U_2 V_2 + U_3 = U_1 V_3$$

$$U_1^4 V_4 + 3U_1^2 U_2 V_3 + 2U_1 U_3 V_2 + U_2^2 V_2 + U_3 = U_1 V_4$$

等等。显然,当 $U_1 = 0$ 时没有解(除非平凡地 $U_2 = U_3 = \cdots = 0$);否则就有惟一解,除非 U_1 是一个单位根。我们还可能预料到,当 $U_1 = 1$ 时有某种趣事出现,因为等式(20)告诉我们,如果在这种情况下 Schröder 函数存在,则 $U^{[n]}(z) = z$ 。眼下,我们假定 U_1 非零而且不是单位根,则 Schröder 函数确实存在,因而下一个问题是怎样计算它而又不用做太多工作。

R.P.Brent 和 J.F.Traub 已经提出了下列过程。方程(21)导致了一个类似的但形式更为复杂的子问题,所以我们给自己提出一个更为一般的任务,而它的子任务有同

样的形式:我们现在来找 $V(z) = V_0 + V_1 z + \cdots + V_{n-1} z^{n-1}$ 使得

$$V(U(z)) = W(z)V(z) + S(z) + O(z^n) \quad (22)$$

给定 $U(z)$, $W(z)$, $S(z)$ 和 n , 其中 n 是 2 的幂, 且 $U(0) \neq 0$ 。如果 $n=1$, 我们简单地令 $V_0 = S(0)/(1-W(0))$, 而如果 $S(0) \neq 0$ 和 $W(0)=1$, 则 $V_0=1$ 。而且有可能从 n 进行到 $2n$: 首先我们求 $R(z)$ 使得

$$V(U(z)) = W(z)V(z) + S(z) - z^n R(z) + O(z^{2n}) \quad (23)$$

然后计算

$$\hat{W}(z) = W(z)(z/U(z))^n + O(z^{2n}), \hat{S}(z) = R(z)(z/U(z))^n + O(z^n) \quad (24)$$

并求 $\hat{V}(z) = \hat{V}_0 + \hat{V}_1 z + \cdots + \hat{V}_{2n-1} z^{2n-1}$ 使得

$$\hat{V}(U(z)) = \hat{W}(z)\hat{V}(z) + \hat{S}(z) + O(z^n) \quad (25)$$

由此得出, 如同所希望的那样, $V^*(z) = V(z) + z^n \hat{V}(z)$ 满足

$$V^*(U(z)) = W(z)V^*(z) + S(z) + O(z^{2n})$$

这个过程的运行时间 $T(n)$ 满足

$$T(2n) = 2T(n) + C(n) \quad (26)$$

其中 $C(n)$ 是计算 $R(z)$, $\hat{W}(z)$ 和 $\hat{S}(z)$ 的时间。函数 $C(n)$ 由计算 $V(U(z))$ 模 z^{2n} 的时间所支配, 而且 $C(n)$ 大概比 $n^{1+\epsilon}$ 的阶增长更快; 因此 (26) 的解的 $T(n)$ 阶将为 $C(n)$ 。例如, 如果 $C(n) = cn^3$, 我们有 $T(n) \approx \frac{4}{3} cn^3$; 或如果用“快速”组合, 则 $C(n)$ 是 $O(n \log n)^{3/2}$, 我们有 $T(n) = O(n \log n)^{3/2}$ 。

当 $W(0)=1$ 和 $S(0) \neq 0$ 时, 这一过程失败, 所以我们需要研究什么时候会发生这一情况。对 n 用归纳法容易证明, 用 Brent-Traub 方法得到 (22) 的解必需考虑精确的 n 个子问题。其中右边的 $V(z)$ 的系数按某种顺序分别取值 $W(z)(z/U(z))^j + O(z^n)$, 其中 $0 \leq j < n$ 。因此如果 $W(0) = U_1$ 且 U_1 不是单位根, 则仅当 $j=1$ 时才有 $W(0)=1$; 在这种情况下, 仅当对于 $n=2$ 式 (22) 没有解时, 这一过程才失败。

因此每当 U_1 非零和不是单位根时, 通过对于 $n=2, 4, 8, 16, \dots$ 和 $W(z) = U_1$ 及 $S(z)=0$ 解 (22), 可以求出 U 的 Schröder 函数。

如果 $U_1=1$, 则除非 $U(z)=z$, 否则没有 Schröder 函数。但 Brent 和 Traub 已经找到了计算 $U^{[n]}(z)$ 的一个快速方法, 甚至当 $U_1=1$ 时也适用, 它利用函数 $V(z)$ 使得

$$V(U(z)) = U'(z)V(z) \quad (27)$$

成立。如果对于同一个 V , 两个函数 $U(z)$ 和 $\hat{U}(z)$ 都满足 (27), 容易验证它们的复合 $U(\hat{U}(z))$ 也满足 (27); 因此 $U(z)$ 的所有迭代都是 (27) 的解。假设 $U(z) = z + U_k z^k + U_{k+1} z^{k+1} + \dots$, 其中 $k \geq 2$ 和 $U_k \neq 0$ 。于是可以证明, 有惟一的满足 (27) 的形如 $V(z) = z^k + V_{k+1} z^{k+1} + V_{k+2} z^{k+2} + \dots$ 的幂级数。反之, 如果给定这样的函数 $V(z)$, 而且如果给定 $k \geq 2$ 和 U_k , 则有惟一的满足 (27) 的形如 $U(z) = z + U_k z^k + U_{k+1} z^{k+1} + \dots$ 的幂级数。所求的迭代 $U^{[n]}(z)$ 是满足

$$V(P(z)) = P'(z)V(z) \quad (28)$$

的惟一幂级数 $P(z)$, 使得 $P(z) = z + nU_k z^k + \dots$ 。通过适当的算法可以求出 $V(z)$ 和

$P(z)$ 两者(见习题 14)。

如果 U_1 是 k 次单位根,但不等于 1,则同样的方法可应用于函数 $U^{[k]}(z) = z + \dots$,而且通过做 $l(k)$ 次复合操作可由 $U(z)$ 求得 $U^{[k^l]}(z)$ (参考 4.6.3 小节)。我们也可以处理 $U_1 = 0$ 的情况:如果 $U(z) = U_k z^k + U_{k+1} z^{k+1} + \dots$,其中 $k \geq 2$ 且 $U_k \neq 0$,则先求方程 $V(U(z)) = U_k V(z)^k$ 的一个解;于是

$$U^{[n]}(z) = V^{-1}(U_k^{[k^n]}(V(z)^k)) \quad (29)$$

最后,如果 $U(z) = U_0 + U_1 z + \dots$,其中 $U_0 \neq 0$,设 α 是使得 $U(\alpha) = \alpha$ 的一个“固定点”,令

$$\tilde{U}(z) = U(\alpha + z) - \alpha = zU'(\alpha) + z^2 U''(\alpha)/2! + \dots \quad (30)$$

则 $U^{[n]}(z) = \tilde{U}^{[n]}(z - \alpha) + \alpha$ 。在 Brent 和 Traub 的文章中可以找到进一步的细节 [SICOMP 9 (1980), 54~66]。(27)的 V 函数以前已被 M. Kuczma 考虑过, *Functional Equations in a Single Variable* (Warsaw: PWN-Polish Scientific, 1968), 引理 9.4,也曾由 E. Jabotinsky 在更早些年暗示过(见习题 23)。

代数函数 满足形如

$$A_n(z)W(z)^n + \dots + A_1(z)W(z) + A_0(z) = 0 \quad (31)$$

的一般方程(其中每个 $A_i(z)$ 都是一个多项式)的每个幂级数 $W(z)$ 的系数,都可以用 H. T. Kung 和 J. F. Traub 的方法有效地加以计算,见 JACM 25 (1978), 245~260。也见 D. V. Chudnovsky 和 G. V. Chudnovsky, *J. Complexity* 2 (1986), 271~294; 3 (1987), 1~25。

习 题

1. [M10] 正文说明当 $V_0 \neq 0$ 时怎样以 $V(z)$ 来除 $U(z)$; 当 $V_0 = 0$ 时除法又应该怎样进行?

2. [20] 如果 $U(z)$ 和 $V(z)$ 的系数是整数且 $V_1 \neq 0$, 求对于整数 $V_0^{-1} W_n$ 的一个递推关系, 其中 W_n 由 (3) 定义。你将如何使用这个结果于幂级数除法?

3. [M15] 当 $\alpha = 0$ 时, 公式 (9) 给出正确的结果吗? 当 $\alpha = 1$ 呢?

► 4. [HM23] 证明当 $V_0 = 0$ 时对 (9) 的简单修改可用来计算 $c^{V(z)}$, 而当 $V_0 = 1$ 时可用来计算 $\ln V(z)$ 。

5. [M00] 当一个幂级数被反演两次时, 即, 如果算法 L 或 T 的输出再次被反演时, 将发生什么?

► 6. [M21] (H. T. Kung) 当 $V(0) \neq 0$ 时, 通过求出方程 $f(x) = 0$ (其中 $f(x) = x^{-1} \cdot V(z)$) 的幂级数根, 把牛顿方法应用到 $W(z) = 1/V(z)$ 的计算中。

7. [M23] 使用拉格朗日的求逆公式 (12) 求出在 $z = t - t^n$ 的反演中系数 W_n 的一个简单表达式。

► 8. [M25] 如果 $W(z) = W_1 z + W_2 z^2 + W_3 z^3 + \dots = G_1 t + G_2 t^2 + G_3 t^3 + \dots = G(t)$, 其中 $z = V_1 t + V_2 t^2 + V_3 t^3 + \dots$ 且 $V_1 \neq 0$ 。拉格朗日证明

$$W_n = \frac{1}{n} [t^{n-1}] G'(t) / (V_1 + V_2 t + V_3 t^2 + \dots)^n$$

(等式(12)是 $G_1 = V_1 = 1, G_2 = G_3 = \cdots = 0$ 的特殊情况.) 试推广算法 L, 使得它得到在这个更一般情况下的系数 W_1, W_2, \cdots , 而不相当地增加它的运行时间。

9. [11] 当算法 T 确定在 $x = t - t^2$ 的反演中的头五个系数时, 试求出由算法 T 计算出的 T_{mn} 的值。

10. [M20] 给定 $y = x^a + a_1 x^{a-1} + a_2 x^{a-2} + \cdots, a \neq 0$, 说明怎样计算展开式 $x = y^{1/a} + b_2 y^{2/a} + b_3 y^{3/a} + \cdots$ 中的系数。

► 11. [M25] (幂级数的复合) 设

$U(z) = U_0 + U_1 z + U_2 z^2 + \cdots$ 且 $V(z) = V_1 z + V_2 z^2 + V_3 z^3 + \cdots$, 试设计一个算法, 以计算 $U(V(z))$ 的头 N 个系数。

12. [M20] 找出多项式除法和幂级数除法之间的联系: 给定在一个域上次数分别为 m 和 n 的多项式 $u(x)$ 和 $v(x)$, 说明怎样只利用对幂级数的运算来求多项式 $q(x), r(x)$, 使得 $u(x) = q(x) \cdot v(x) + r(x)$ 且 $\deg(r) < n$ 。

13. [M27] (有理函数近似) 有时希望求这样的多项式, 它们的商和某个给定的幂级数有相同的初始项。例如, 如果 $W(z) = 1 + z + 3z^2 + 7z^3 + \cdots$, 实质上有四种不同方法把 $W(z)$ 表达为 $w_1(z)/w_2(z) + O(z^4)$, 其中 $w_1(z)$ 和 $w_2(z)$ 是有 $\deg(w_1) + \deg(w_2) < 4$ 的多项式

$$(1 + z + 3z^2 + 7z^3)/1 = 1 + z + 3z^2 + 7z^3 + 0z^4 + \cdots$$

$$(3 - 4z + 2z^2)/(3 - 7z) = 1 + z + 3z^2 + 7z^3 + \frac{49}{3}z^4 + \cdots$$

$$(1 - z)/(1 - 2z - z^2) = 1 + z + 3z^2 + 7z^3 + 17z^4 + \cdots$$

$$1/(1 - z - 2z^2 - 2z^3) = 1 + z + 3z^2 + 7z^3 + 15z^4 + \cdots$$

这类有理函数普通称为 Padé 近似, 因为它们为 H. E. Padé 广泛地研究 [Annales Scient. de l'École Normale Supérieure (3) 9 (1892), S1~S93; (3) 16 (1899), 395~426]。

证明使 $\deg(w_1) + \deg(w_2) < N$ 的所有 Padé 近似 $W(z) = w_1(z)/w_2(z) + O(z^N)$ 都可以通过把扩充的欧几里得算法应用于多项式 z^N 和 $W_0 + W_1 z + \cdots + W_{N-1} z^{N-1}$ 而得到; 并设计在每个 W_i 都是整数的情况下的一个全整数算法。[提示: 见习题 4.6.1-26c]

► 14. [HM30] 当 $U(z) = z + U_k z^k + \cdots$ 时, 利用(27)和(28)补足用于计算 $U^{[n]}(z)$ 的 Brent 和 Traub 方法的细节。

15. [HM20] 对于什么样的函数 $U(z), V(z)$ 有(27)中的简单形式 z^k ; 关于 $U(z)$ 的迭代你推导出什么?

16. [HM21] 设 $W(z) = G(t)$ 如同在习题 8 那样, 求系数 W_1, W_2, W_3, \cdots 的“明显”方法进行如下: 置 $n \leftarrow 1$ 和 $R_1(t) \leftarrow G(t)$, 然后通过重复地置 $W_n \leftarrow [t] R_n(t)/V_1, R_{n+1}(t) \leftarrow R_n(t)/V(t) - W_n, n \leftarrow n+1$ 保持关式 $W_n V(t) + W_{n+1} V(t)^2 + \cdots = R_n(t)$ 。

通过证明, 对所有 $n \geq 1$ 和 $k \geq 1$,

$$\frac{1}{n} [t^{n-1}] R'_{k+1}(t) t^n / V(t)^n = \frac{1}{n+1} [t^n] R'_k(t) t^{n+1} / V(t)^{n+1}$$

来证明习题 8 的拉格朗日公式。

► 17. [M20] 给定幂级数 $V(z) = V_1 z + V_2 z^2 + V_3 z^3 + \cdots$, 我们把 V 的幂矩阵定义为系数 $v_{nk} = \frac{n!}{k!} [z^n] V(z)^k$ 的无穷数组; V 的第 n 个似次幂(poweroid)定义为 $V_n(x) = v_{n0} + v_{n1}x + \cdots + v_{nn}x^n$ 。证明诸似次幂满足一般的卷积律

$$V_n(x+y) = \sum_k \binom{n}{k} V_k(x) V_{n-k}(y)$$

(例如当 $V(z) = z$ 时我们有 $V_n(x) = x^n$, 而且这是二项式定理。当 $V(z) = \ln(1/(1-z))$ 时, 由等式 1.2.9-(26), 我们有 $v_{nk} = \begin{bmatrix} n \\ k \end{bmatrix}$; 因此似次幂 $V_n(x)$ 是 x^n , 而且恒等式是习题 1.2.6-33 中证明的结果。当 $V(z) = e^z - 1$ 时我们有 $V_n(x) = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} x^k$ 而且这个公式等价于

$$\begin{bmatrix} l+m \\ m \end{bmatrix} \begin{bmatrix} n \\ l+m \end{bmatrix} = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} \begin{bmatrix} k \\ l \end{bmatrix} \begin{bmatrix} n-k \\ m \end{bmatrix}.$$

这是以前我们未曾见过的一个恒等式。在组合数学和算法分析中出现的系数的若干其它的三角数组也被证实是幂级数的幂矩阵。)

18. [HM22] 继续习题 17, 证明似次幂也满足

$$xV_n(x+y) = (x+y) \sum_k \begin{bmatrix} n-1 \\ k-1 \end{bmatrix} V_k(x) V_{n-k}(y)$$

[提示: 考虑 $e^{xV(z)}$ 的导数。]

19. [M25] 继续习题 17, 借助于头一列的数 $v_n = v_{n1} = n! V_n$ 来表达所有数 v_{nk} , 并求出所有列都可由序列 v_1, v_2, \dots 加以计算出来的简单递推关系。特别证明, 如果所有 v_n 是整数, 则所有 v_{nk} 都是整数。

20. [HM20] 继续习题 17, 假设我们有 $W(z) = U(V(z))$ 和 $U_0 = 0$ 。证明 W 的幂矩阵是 U 的幂矩阵的乘积: $w_{nk} = \sum_j v_{nj} u_{jk}$ 。

► 21. [HM27] 继续上题, 假设 $V_1 \neq 0$ 并设 $W(z) = -V_1^{-1}(1-z)$ 。本题的目的是证明, V 和 W 的幂矩阵是彼此“对偶”的; 例如, 当 $V(z) = \ln(1/(1-z))$ 时, 我们有 $V_1^{-1}(z) = 1 - e^{-z}$, $W(z) = e^z - 1$, 而对应的幂矩阵是著名的斯特林三角 $v_{nk} = \begin{bmatrix} n \\ k \end{bmatrix}$, $w_{nk} = \begin{bmatrix} n \\ k \end{bmatrix}$ 。

a) 证明对于斯特林数的逆公式 1.2.6-(47) 一般都成立:

$$\sum_k v_{nk} w_{km} (-1)^{n-k} = \sum_k w_{nk} v_{km} (-1)^{n-k} = \delta_{nm}$$

b) 关系式 $v_{n(n-k)} = n^k [z^k] (V(z)/z)^n$ 表明, 对于固定的 k , 量 $v_{n(n-k)}/V_1^n$ 是次数小于等于 $2k$ 的 n 的多项式。因此我们可以定义, 如同在 1.2.6 小节我们对于斯特林数所做的那样, 当 k 是一个非负整数时, 对于任意的 α ,

$$v_{\alpha(\alpha-k)} = \alpha^k [z^k] (V(z)/z)^{\alpha-k}$$

证明 $v_{(-k)(-n)} = w_{nk}$ (这推广了等式 1.2.6-(58))。

► 22. [HM27] 给定 $U(z) = U_0 + U_1 z + U_2 z^2 + \dots$ 且 $U_0 \neq 0$, 第 α 个导出的函数 $U^{|\alpha|}(z)$ 是由等式

$$V(z) = U(zV(z)^\alpha)$$

隐含定义的幂级数 $V(z)$ 。

a) 证明 $U^{|\alpha|}(z) = U(z)$ 和 $U^{|\alpha|+|\beta|}(z) = U^{|\alpha+\beta|}(z)$ 。

b) 设 $B(z)$ 是简单的二项式序列 $1+z$ 。以前我们在哪里见过 $B^{|\alpha|}(z)$?

c) 证明 $[z^n] U^{|\alpha|}(z)^x = \frac{x}{x+n\alpha} [z^n] U(z)^{x+n\alpha}$ 。提示: 如果 $W(z) = z/U(z)^\alpha$, 则我们有

$$U^{|\alpha|}(z) = (W^{|\alpha|-1}(z)/z)^{1/\alpha}.$$

d) 因此任何似次幂 $V_n(x)$ 不仅满足习题 17 和 18 的恒等式, 而且满足

$$\frac{(x+y)V_n(x+y+n\alpha)}{x-y+n\alpha} = \sum_i \binom{n}{i} \frac{xV_i(x+ka)}{x+ka} \frac{yV_{n-k}(y+(n-k)\alpha)}{y+(n-k)\alpha}$$

$$\frac{V_l(x-y)}{y-na} = (x+y) \sum_1 \binom{n-1}{k-1} \frac{V_l(x+ka)}{x+ka} \frac{V_{n-k}(y-ka)}{y-ka}$$

[特殊情况包括阿贝尔二项式定理,即等式 1.2.6-(16);Rothe 恒等式 1.2.6-(26)与 1.2.6-(30);Torelli 和,习题 1.2.6-34。]

23. [HM35] (E. Jabotinsky) 以同样的脉络继续进行,假设 $U = (u_{nk})$ 是 $U(z) = z + U_2 z^2 + \cdots$ 的幂矩阵,令 $u_n = u_{nn} = n! U_n$ 。

a) 说明如何来计算一个矩阵 $\ln U$ 使得 $U^{[a]}(z)$ 的幂矩阵是 $\exp(\alpha \ln U) = I + \alpha \ln U + (\alpha \ln U)^2/2! + \cdots$ 。

b) 设 l_{nk} 是 $\ln U$ 的 n 行 k 列的元素,并设

$$l_n = l_{n1} \cdot L(z) = l_2 \frac{z^2}{2!} + l_3 \frac{z^3}{3!} + l_4 \frac{z^4}{4!} + \cdots$$

证明对于 $1 \leq k \leq n$, $l_{nk} = \binom{n}{k-1} l_{n+1-k}$ [提示: $U^{[a]}(z) = z + aL(z) + O(a^2)$ 。]

c) 把 $U^{[a]}(z)$ 当做 a 和 z 两者的函数,证明

$$\frac{\partial}{\partial \alpha} U^{[a]}(z) = L(z) \frac{\partial}{\partial z} U^{[a]}(z) = L(U^{[a]}(z))$$

(因此 $L(z) = (l_k/k!) V(z)$, 其中 $V(z)$ 是 (27) 和 (28) 中的函数。)

d) 证明如果 $u_2 \neq 0$, 则数 l_n 可从递推式

$$l_2 = u_2, \quad \sum_{k=2}^n \binom{n}{k} l_k u_{n+1-k} = \sum_{k=2}^n l_k u_{nk}$$

计算出来。当 $u_2 = 0$ 时,你将怎样使用这个递推式?

e) 证明恒等式

$$u_n = \sum_{m=0}^{n-1} \frac{n!}{m!} \sum_{\substack{j_1 + \cdots + j_m = n-m-1 \\ k_1, \dots, k_m \geq 2}} \frac{n_0!}{k_1!} \frac{n_1!}{k_2!} \cdots \frac{n_{m-1}!}{k_m!} l_{k_1} l_{k_2} \cdots l_{k_m}$$

其中 $n_j = 1 + k_1 + \cdots + k_j$ 。

24. [HM25] 给定幂级数 $U(z) = U_1 z + U_2 z^2 + \cdots$, 其中 U_1 不是单位根, 令 $U = (u_{nk})$ 是 $U(z)$ 的幂矩阵。

a) 说明怎样计算一个矩阵 $\ln U$, 使得 $U^{[a]}(z)$ 的幂矩阵是 $\exp(\alpha \ln U) = I + \alpha \ln U + (\alpha \ln U)^2/2! + \cdots$ 。

b) 证明如果 $W(z)$ 不恒等于零, 而且如果 $U(W(z)) = W(U(z))$, 则对于某个复数 α , $W(z) = U^{[a]}(z)$ 。

25. [M24] 如果 $U(z) = z + U_k z^k + U_{k+1} z^{k+1} + \cdots$ 和 $V(z) = z + V_l z^l + V_{l+1} z^{l+1} + \cdots$, 其中 $k \geq 2$, $l \geq 2$, $U_k \neq 0$, $V_l \neq 0$ 且 $U(V(z)) = V(U(z))$ 。证明, 我们必定有 $k = l$ 而且对于 $a = V_k/U_k$, $V(z) = U^{[a]}(z)$ 。

26. [M22] 证明如果 $U(z) = U_0 + U_1 z + U_2 z^2 + \cdots$ 和 $V(z) = V_1 z + V_2 z^2 + \cdots$ 是所有系数为 0 或 1 的幂级数, 对于任何 $\epsilon > 0$, 我们可以在 $O(N^{1+\epsilon})$ 步内得到 $U(V(z)) \bmod 2$ 的前 N 个系数。

27. [M22] (D. Zeilberger) 给定 q , m 和 $V(z) = 1 + V_1 z + V_2 z^2 + \cdots$ 的系数, 求用于计算 $W(z) = V(z) V(qz) \cdots V(q^{m-1}z)$ 的系数的类似于 (9) 的递推式, 假定 q 不是一个单位根。

► 28. [HM26] 一个 Dirichlet 级数是形如 $V(z) = V_1/1^z + V_2/2^z + V_3/3^z + \cdots$ 的和; 两个这样的

级数的乘积 $U(z)V(z)$ 是 Dirichlet 级数 $W(z)$, 其中

$$W_n = \sum_{d \mid n} U_d V_{n/d}$$

通常的幂级数是 Dirichlet 级数的特殊情况, 因为当 $z = 2^{-s}$ 时我们有 $V_0 + V_1 z + V_2 z^2 + V_3 z^3 + \cdots = V_0/1^s + V_1/2^s + V_2/4^s + V_3/8^s + \cdots$ 。事实上, Dirichlet 级数必等价于任意多个变量的幂级数 $V(z_1, z_2, \cdots)$, 其中 $z_k = p_k^{-s}$, p_k 是第 k 个素数。

假定给出一个 Dirichlet 级数 $V(z)$, 而且我们要来计算: (a) 当 $V_1 = 1$ 时 $W(z) = V(z)^a$; (b) 当 $V_1 = 0$ 时 $W(z) = \exp V(z)$; (c) 当 $V_1 = 1$ 时 $W(z) = \ln V(z)$ 。试求推论 (9) 和习题 4 的公式的递推关系。[提示: 令 $t(n)$ 是 n 的素因子, 包括倍数的全部个数, 并令 $\delta \sum_n V_n/n^s = \sum_n t(n) V_n/n^s$ 。证明 δ 类似于导数; 例如 $\delta e^{V(z)} = e^{V(z)} \delta V(z)$ 。]

*It seems impossible that any thing
should really alter the series of things,
without the same power which first produced them.*

任何事情似乎都不可能
真正地改变事物的序列,
如果没有最初产生出这些事物的相同的力量。的话。

——EDWARD STILLINGFLEET, *Origines Sacrae*, 2:3:2 (1662)

习 题 答 案

*This branch of mathematics is the only one, I believe,
in which good writers frequently get results entirely erroneous.*

*... It may be doubted if there is a single
extensive treatise on probabilities in existence
which does not contain solutions absolutely indefensible.*

我相信,数学的这个分支是惟一这样的分支,
在这个分支中,就连好的作者也经常得到完全错误的结果。

……是否存在关于概率的一篇详尽的论文,
它不包含绝对站不住脚的解,可能是令人怀疑的。

C. S. Peirce, in *Popular Science Monthly* (1878)

关于习题的说明

1. 对于一个有数学素养的读者说来,这是一个一般难度的问题。

3. (由 Roger Frye 于 1987 年在一台 Connection Machine 上经过大约 110 小时的计算之后给出的解。) $95800^4 + 217519^4 + 414560^4 = 422481^4$ 。

4. (本书初稿的一位读者报告说,他已经发现了一个真正值得注意的证明,但遗憾的是他并没有把这一证明告诉我。)

3.1 节

1. a) 这通常不会成功,因为电话用户总是尽可能选择尾数为零的电话号码。在某些地方,电话号码也许是随机地指定的。但是在任何情况下,你都不会从同一页上得到若干相继的随机数,因为常有同一个电话号码在一行中多次出现的情形。

b) 但是你使用的是左面的页还是右面的页? 比如说,用左页号,除以 2,并取单个数字,总的页数应当是 20 的倍数;即使如此,这个方法仍会有某些偏倚。

c) 面上的标志将使骰子有某些偏倚,但是对于实际应用来说,这个方法十分令人满意(而且,作者在准备这一套书中的好些例子时已经用了它)。关于这些骰子的进一步的讨论见 *Math. Comp.* **15** (1961), 94~95。

d) (这是有意为引起惊奇而抛出的难题。)这个数不是很均匀随机的。如果每分钟放射的平均数为 m , 则计数器记下 k 的概率是 $e^{-m} m^k / k!$ (泊松分布); 所以选择数字 0 的概率是 $e^{-m} \sum_{k \geq 0} m^{10k} / (10k)!$, 等等。所得数字是偶数的概率是 $e^{-m} \cosh m = \frac{1}{2} + \frac{1}{2} e^{-2m}$, 而这绝不等于 $\frac{1}{2}$ (尽管当 m 非常大时,误差小到可以忽略)。

然而,如果对于所有 $i \neq j, m_i$ 严格地小于 m_j , 则取 10 个输入 (m_0, \dots, m_9) 而后输出 j 是合理的; 如果极小值出现一次以上, 则再试一遍。(见 h)。

e) 是的, 假定自从最后一次用这种方式选择数字以来, 时间是随机的。但是, 在边界线情况下可能有偏倚。

f), g) 否, 人们常常设想某些数字(比如说 7)具有更高的概率。

h) 是的; 你对马赋予号码时, 把给定的数字赋予优胜马的概率是 $\frac{1}{10}$ 。

2. 这种序列的数目是多项式系数 $1000000! / (100000!)^{10}$; 概率是这个数除以 $10^{1000000}$, 即百万位数字的序列总数, 由斯特林近似公式我们求得这个概率接近于

$$1/(16\pi^4 10^{22} \sqrt{2\pi}) \approx 2.56 \times 10^{-26}$$

大约在 4×10^{25} 次中有一次机会。

3. 3040504030。

4. (a) 步骤 K11 仅可从步骤 K10 或步骤 K2 进入, 而且在每种情况下, 我们通过简单的论证即知 X 不可能为 0。如果 X 在这一步可以为 0, 则这个算法将不会终止。

(b) 如果开始时 X 是 3830951656, 则除了我们是以 $Y=3$ 而不是以 $Y=5$ 到达步骤 K11 之外, 这个计算和出现于表 1 中的许多步类似; 因此 $3830951656 \rightarrow 5870802097$ 。类似地, $5870802097 \rightarrow 1226919902 \rightarrow 3172562687 \rightarrow 3319967479 \rightarrow 6065038420 \rightarrow 6065038420 \rightarrow \dots$ 。

5. 因为只可能有 10^{10} 个 10 位数字的数, 因此在头 $10^{10} - 1$ 个步骤期间, X 的某个值必然重复, 而只要一个值重复了, 这个序列就继续重复它前面的特性。

6. a) 如同上题那样, 可以论证这个序列最终必然会重复一个值; 假设这个重复在步骤 $\mu + \lambda$ 时头一次出现, 其中 $X_{\mu+\lambda} = X_\mu$ (这个条件确定 μ 和 λ)。我们有 $0 \leq \mu < m, 0 < \lambda \leq m, \mu + \lambda \leq m$ 。当且仅当 f 是一个循环排列时, 才达到 $\mu = 0, \lambda = m$; 而且出现 $\mu = m - 1, \lambda = 1$, 例如, 如果 $X_0 = 0$, 则 $x < m - 1$ 时 $f(x) = x + 1$, 而 $f(m - 1) = m - 1$ 。

b) 若 $r > n$, 则当且仅当 $r - n$ 是 λ 的一个倍数且 $n \geq \mu$ 时我们有 $X_r = X_n$ 。因此当且仅当 n 是 λ 的一个倍数且 $n \geq \mu$ 时, $X_{2n} = X_n$, 由此即得所求的结果。[注: 这实质上是一个熟知数学结果的证明, 即在一个有限半群中一个元素的幂包括惟一的幂等元: 取 $X_1 = a, f(x) = ax$ 。]

c) 一旦找到了 n , 对 $i \geq 0$ 生成 X_i 和 X_{n+i} , 直到头一次发现 $X_i = X_{n+i}$ 为止; 于是 $\mu = i$ 。如果对于 $0 < i < \mu, X_{n+i}$ 的值没有等于 X_n 的, 则由此得出 $\lambda = n$, 否则 λ 是这样的 i 中之最小者。

7. a) 使得 $n - (\ell(n) - 1)$ 是 λ 的倍数, 而且 $\ell(n) - 1 \geq \mu$ 的最小的 $n > 0$ 是 $n = 2^{\lceil \lg \max(n+1, \lambda) \rceil} - 1 + \lambda$ 。[可以把它和使得 $X_{2n} = X_n$ 最小的 $n > 0$ 比较, 此即 $\lambda(\lceil \mu/\lambda \rceil + \delta_{\mu 0})$ 。]

b) 从 $X = Y = X_0, k = m = 1$ 开始。(在这个算法的关键处我们将有 $X =$

X_{2m-k-1} , $Y = X_{m-1}$ 和 $m = \ell(2m-k)$ 。)为了生成下一个随机数,执行下列步骤:置 $X \leftarrow f(X)$ 和 $k \leftarrow k-1$ 。如果 $X = Y$,就停止(周期长度 λ 等于 $m-k$)。否则如果 $k = 0$,则置 $Y \leftarrow X$, $m \leftarrow 2m$, $k \leftarrow m$ 。输出 X 。

注:Brent也曾考虑过更为一般的方法,其中 $Y = X_{n_i}$ 的相继的值满足 $n_1 = 0$, $n_{i+1} = 1 + \lfloor pn_i \rfloor$, 其中 p 是任何大于 1 的数。他证明, p 的最好选择,近似于 2.4771,同 $p = 2$ 比较,大约节省了 3% 的迭代(见习题 4.5.4-4)。

但是 b) 部分中的算法有严重的缺陷,因为在停止之前它可能产生大量的非随机数。例如,我们可能有像 $\lambda = 1, \mu = 2^k$ 这样特别坏的情况。习题 6b) 中以 Floyd 的思想为基础的方法,即对于 $n = 0, 1, 2, \dots$, 保持 $Y = X_{2n}$ 和 $X = X_n$ 的方法,要求的函数计算比 Brent 方法多一些,但只要有一个数输出两次,它就停止执行。

另一方面,如果 f 未知(例如,如果我们从外部源在线接受值 X_0, X_1, \dots) 或者如果 f 难以应用,则下面由 R. W. Gosper 给出的循环探测算法将是可取的:当接受 X_n 时保持一张辅助表 T_0, T_1, \dots, T_m , 其中 $m = \lfloor \lg n \rfloor$ 。开始时,置 $T_0 \leftarrow X_0$ 。对于 $n = 1, 2, \dots$, 把 X_n 同 $T_0, \dots, T_{\lfloor \lg n \rfloor}$ 中的每一个进行比较;如果找不到相符的,则置 $T_{e(n)} \leftarrow X_n$, 其中 $e(n) = \max\{e \mid 2^e \text{ 整除 } n+1\}$ 。但是如果找到了一个相符 $X_n = T_k$, 则 $\lambda = n - \max\{l \mid l < n \text{ 且 } e(l) = k\}$ 。在把 X_n 存入 $T_{e(n)}$ 之后,它随后同 $X_{n+1}, X_{n+2}, \dots, X_{n+2^{e(n)+1}}$ 作比较。因此在生成 $x_{\mu+\lambda-j}$ (其中 $j \geq 0$ 是满足 $e(\mu+j) \geq \lceil \lg \lambda \rceil - 1$ 的极小值)之后,这个过程立即停止。通过这个方法,不会生成多于两次的 X 值,而且至多有 $\max(1, 2^{\lceil \lg \lambda \rceil - 1})$ 个值生成一次以上。(MIT AI Laboratory Memo 239 (1972 年 2 月 29 日), Hack 132。]

R. Sedgewick, T. G. Szymanski 和 A. C. Yao 已经分析了基于参数 $m \geq 2$ 和 $g \geq 1$ 的一个更复杂的算法:在计算 X_n 的时刻大小为 m 的一个辅助表包含 X_0, X_b, \dots, X_{gb} , 其中 $b = 2^{\lceil \lg n/m \rceil}$ 且 $q = \lceil n/b \rceil - 1$ 。如果 $n \bmod gb < b$, 就把 X_n 同这个表的元素作比较;最终将出现相等,因此在对 f 至多进行 $(g+1)2^{\lceil \lg(\mu+\lambda) \rceil + 1}$ 进一步的计算之后,我们可以重新构造 μ 和 λ 。如果对 f 的计算花费 τ 个时间单位,而且如果对于 X_n 是否表中元素的检测花费 σ 个时间单位,则 g 被选中使得总的运行时间是 $(\mu + \lambda)(\tau + O(\frac{\sigma\tau}{m})^{1/2})$; 如果 $\sigma/\tau = O(m)$, 则这是最优的。而且,除非 $\mu + \lambda > mn/(m+4g+2)$, 否则 X_n 不被计算,所以我们可以使用这个“在线”方法,来输出保证不同的元素,而对于每个输出只做 $2 + O(m^{-1/2})$ 个函数计算。[SICOMP 11 (1982), 376~390。]

8. a), b), 00, 00, \dots [62 个开始值]; 10, 10, \dots [19]; 60, 60, \dots [15]; 50, 50, \dots [1]; 24, 57, 24, 57, \dots [3]。c) 42 或 69; 这两者都导致一个含有 15 个不同值的集合, 即 (42 或 69), 76, 77, 92, 46, 11, 12, 14, 19, 36, 29, 84, 05, 02, 00。

9. 由于 $X < b^n$, 我们有 $X^2 < b^{2n}$, 所以平方取中是 $\lfloor X^2/b^n \rfloor \leq X^2/b^n$ 。如果 $X > 0$, 则 $X^2/b^n < Xb^n/b^n = X$ 。

10. 如果 $X = ab^n$, 则这个序列的下一个数有相同的形式, 它等于 $(a^2 \bmod b^n) \cdot b^n$ 。如果 a 是 b 的所有素因子的倍数, 则这个序列将立即退化成 0; 如果不然, 则这个序列将退化成为有和 X 相同的一般形式的一些数的循环。

B. Jansson 已经发现了有关平方取中法的进一步的事实, *Random Number Generators* (Stockholm: Almqvist & Wiksell, 1966), Section 3A。数值学家们将有兴趣获悉下列事实: 数 3792 在四位平方取中方法下是自再生的, 因为 $3792^2 = 14379264$; 而且 (如 Jansson 已经发现的), 在另一种意义下它也是“自再生”的, 因为它的素因子分解为 $3 \times 79 \times 2^4$!

11. $\lambda = 1, \mu = 0$ 的概率是 $X_1 = X_0$ 的概率, 即 $1/m$ 。 $\lambda = 1, \mu = 1$ 或 $\lambda = 2, \mu = 0$ 的概率是 $X_1 \neq X_2$ 和 X_2 有某一个值的概率, 所以它是 $(1 - 1/m)(1/m)$ 。类似地, 这个序列有任意给定的 μ 和 λ 的概率仅是 $\mu + \lambda$ 的一个函数, 即

$$P(\mu, \lambda) = \frac{1}{m} \prod_{1 \leq k \leq \mu + \lambda} \left(1 - \frac{k}{m}\right)$$

对于 $\lambda = 1$ 的概率, 我们有

$$\sum_{\mu \geq 0} \frac{1}{m} \prod_{k=1}^{\mu} \left(1 - \frac{k}{m}\right) = \frac{1}{m} Q(m)$$

其中 $Q(m)$ 是在 1.2.11.3 小节等式 (2) 中定义的。由该小节的等式 (25), 这个概率近似为 $\sqrt{\pi/2m} \approx 1.25/\sqrt{m}$ 。事实上, 算法 K 收敛的机会仅仅大约为八万分之一; 作者的运气肯定是不好的。但也存在“巧合”现象, 见习题 15。

$$12. \sum_{\substack{1 \leq \lambda \leq m \\ 0 \leq \mu < m}} \lambda P(\mu, \lambda) = \frac{1}{m} \left(1 + 3 \left(1 - \frac{1}{m}\right) + 6 \left(1 - \frac{1}{m}\right) \left(1 - \frac{2}{m}\right) + \cdots\right) = \frac{1 + Q(m)}{2}$$

(见前面的答案。一般来说, 如果 $f(a_0, a_1, \cdots) = \sum_{n \geq 0} a_n \prod_{k=1}^n (1 - k/m)$, 则 $f(a_0, a_1, \cdots) = a_0 + f(a_1, a_2, \cdots) - f(a_1, 2a_2, \cdots)/m$; 对 $a_n = (n+1)/2$ 应用这个恒等式。) 因此, λ (而且, 由 $P(\mu, \lambda)$ 的对称性, 也是 $\mu + 1$) 的平均值近似于 $\sqrt{\pi m/8} + \frac{1}{3}$ 。

$\mu + \lambda$ 的平均值恰为 $Q(m)$, 近似于 $\sqrt{\pi m/2} - \frac{1}{3}$ 。[关于另一个推导和进一步的结果, 包括关于当时的渐近值, 见 A. Rapoport, *Bull. Math. Biophysics* **10** (1948), 145~157, 以及 B. Harris, *Annals Math. Stat.* **31** (1960), 1045~1062; 也见 I. M. Sobol, *Theory of Probability and its Applications* **9** (1964), 333~338。Sobol 讨论了对于更一般的序列的渐近周期长度——这些一般的序列是 $X_{n+1} = f(X_n)$, 若 $n \not\equiv 0 \pmod{m}$; $X_{n+1} = g(X_n)$, 若 $n \equiv 0 \pmod{m}$; f 和 g 两者都是随机的。]

13. Paul Purdom 和 John Williams, *Trans. Amer. Math. Soc.* **133** (1968), 547~551。] 设 T_{mn} 是有 n 个一循环和没有长度大于 1 的循环的函数个数, 则

$$T_{mn} = \binom{m-1}{n-1} m^{m-n}$$

(在习题 2.3.4.4-25 中这是 $\binom{m}{n} r(m, m-n)$), 任何函数是这样: 一个函数后边接上那些构成一循环的 n 个元素的一个排列。因此 $\sum_{n \geq 1} T_{mn} n! = m^m$ 。

设 P_{nk} 是最长循环长度为 k 的那些 n 个元素排列的个数, 则最大循环长度为 k 的函数个数是 $\sum_{n \geq 1} T_{mn} P_{nk}$ 。为得到 k 的平均值, 我们计算 $\sum_{k \geq 1} \sum_{n \geq 1} k T_{mn} P_{nk}$, 由习题 1.3.3-23 的结果, 它是 $\sum_{n \geq 1} T_{mn} n! (cn + \frac{1}{2}c + O(n^{-1}))$, 其中 $c \approx .62433$ 。求和后, 我们得到平均值 $cQ(m) + \frac{1}{2}c + O(m^{-1/2})$ 。(当 X_0 随机地选择时, 这不比平均值大很多。 $\max \mu$ 的平均值渐近于 $Q(m) \ln 4$, 而 $\max(\mu + \lambda)$ 的平均值渐近于 $1.9268Q(m)$; 请见 Flajolet 和 Odlyzko, *Lecture Notes In Comp. Sci.* **434** (1990), 329~354。)

14. 设 $c_r(m)$ 是恰有 r 个不同的最终循环的函数的个数。从递推式 $c_1(m) = (m-1)! - \sum_{k \geq 0} \binom{m}{k} (-1)^k (m-k)^k c_1(m-k)$, 我们找出解 $c_1(m) = m^{m-1} Q(m)$, 这个递推式是通过计算其映像至多含有 $m-k$ 个元素的函数个数得出的(见习题 1.2.11.3-16)。得到 $c_1(m)$ 的另一种方式, 也许是更优雅和更有启示的方式, 在习题 2.3.4.4-17 中给出。 $c_r(m)$ 的值可以像在习题 13 中那样确定:

$$c_r(m) = \sum_{n \geq 1} T_{mn} \left[\begin{matrix} n \\ r \end{matrix} \right] = m^{m-1} \left(\frac{1}{0!} \left[\begin{matrix} 1 \\ r \end{matrix} \right] + \frac{1}{1!} \left[\begin{matrix} 2 \\ r \end{matrix} \right] \frac{m-1}{m} + \frac{1}{2!} \left[\begin{matrix} 3 \\ r \end{matrix} \right] \frac{m-1}{m} \frac{m-2}{m} + \dots \right)$$

现在可以计算出所求的平均值; 它是(见习题 12)

$$E_m = \frac{1}{m} \left(H_1 + 2H_2 \frac{m-1}{m} + 3H_3 \frac{m-1}{m} \frac{m-2}{m} + \dots \right) = 1 + \frac{1}{2} \frac{m-1}{m} + \frac{1}{3} \frac{m-1}{m} \frac{m-2}{m} + \dots$$

这后一公式是用 Martin D. Kruskal 给出的十分不同的手段得到的, 见 *AMM* **61** (1954), 392~397。利用积分表示

$$E_m = \int_0^1 \left(\left(1 + \frac{x}{m} \right)^m - 1 \right) e^{-x} \frac{dx}{x}$$

他证明了渐近关系 $\lim_{m \rightarrow \infty} \left(E_m - \frac{1}{2} \ln m \right) = \frac{1}{2}(\gamma + \ln 2)$ 。关于进一步的结果和参考文献, 见 John Riordan, *Annals Math. Stat.* **33** (1962), 178~185。

15. 对于所有的 x , $f(x) \neq x$ 的概率是 $(m-1)^m / m^m$, 它近似于 $1/e$ 。因此在一个像算法 K 这样的算法中, 存在一个自重复的值, 这一点也不奇怪——它以 $1 - 1/e \approx .63212$ 的概率出现。惟一“巧合”的事情是当 X_0 随机地选择时, 作者碰巧地遇到了这样一个值(见习题 11)。

16. 当一对相继的元素第二次出现时, 这个序列将重复。极大周期是 m^2 (参照

下一个习题)。

17. 在任意选择 X_0, \dots, X_{k-1} 之后, 令 $X_{n+1} = f(X_n, \dots, X_{n-k+1})$, 其中 $0 \leq x_1, \dots, x_k < m$ 意味着 $0 \leq f(x_1, \dots, x_k) < m$ 。极大周期为 m^k 。这是一个明显的上限, 但上限可以达到并不明显; 对于适当的 f , 它总是可以被达到的, 有关这一点的构造性证明, 请看习题 3.2.2-17 和习题 3.2.2-21, 关于达到它的方式数, 见习题 2.3.4.2-23。

18. 和习题 7 一样, 但是使用 k 元组 (X_n, \dots, X_{n-k+1}) 代替单个元素 X_n 。

20. 只须考虑由步骤 K2 ~ K13 所定义的较简单的映射 $g(X)$ 就足够了。由 6065038420 往后工作, 我们得到总共有 597 个解; 最小的是 0009612809, 最大的是 9995371004。

21. 我们可以像上道题一样通过 $g(X)$ 来工作, 但现在我们要向前运行函数而不是向后。在时间和空间之间有一个有趣的折衷。注意步骤 K1 的机制趋向于使周期长度变小。对于大的入度 X 的存在性也是这样; 例如, 在步骤 K2 中 $X = *6*****$ 的 512 种选择也将以 $X \leftarrow 0500000000$ 转到 K10。

Scott Fluhrer 发现了算法 K 的另一个不动点, 即值 5008502835(!)。他也发现了周期为 3 的 $0225923640 \rightarrow 2811514413 \rightarrow 0590051662 \rightarrow 0225923640$, 并生成全部 7 个周期。恰有 128 个起始值导致重复的值 5008502835; 其中最小的是 0008502835, 最大的是 9944390948。算法 K 是一个糟糕的随机数生成程序。

22. 如果 f 真正随机的话, 这将是理想的; 但我们怎样构造这样的 f 呢? 在这样的方案之下由算法 K 定义的函数要工作得好得多, 尽管它确实有肯定非随机的性质 (参见上一个答案)。

23. 函数 f 排列它的循环元素: 令 (x_0, \dots, x_{k-1}) 是该排列的逆的“非寻常的”表示。然后像在习题 2.3.4.4-18 那样来定义 x_k, \dots, x_{m-1} 。[见 *J. Combinatorial Theory* 8 (1970), 361~375。]

例如, 如果 $m = 10$ 且 $(f(0), \dots, f(9)) = (3, 1, 4, 1, 5, 9, 2, 6, 5, 4)$, 我们有 $(x_0, \dots, x_9) = (4, 9, 5, 1, 1, 3, 4, 2, 6, 5)$; 如果 $(x_0, \dots, x_9) = (3, 1, 4, 1, 5, 9, 2, 6, 5, 4)$, 我们有 $(f(0), \dots, f(9)) = (6, 4, 9, 3, 1, 1, 2, 5, 4, 5)$ 。

3.2.1 小节

1. 取 X_0 为偶数, a 为偶数, c 为奇数, 则对于 $n > 0$, X_n 为奇数。

2. 设 X_r 是这个序列中头一个重复的值。如果对于 $0 < k < r$ 中的某个 k 有 X_r 等于 X_k , 那我们就可以证明 $X_{r-1} = X_{k-1}$, 因为当 a 与 m 互素时, X_n 惟一地确定 X_{n-1} 。因此 $k = 0$ 。

3. 如果 d 是 a 和 m 的最大公因子, 则 aX_n 至多可以取 m/d 个值。情况甚至可以更坏; 例如, 如果 $m = 2^r$ 且 a 为偶数, 则等式 (6) 说明这个序列最终是常数。

4. 对 k 用归纳法。

5. 如果 a 和 m 互素, 则有一个数 a' , 使得 $aa' \equiv 1 \pmod{m}$ 。于是 $X_{n-1} =$

$(a'X_n - a'c) \bmod m$, 一般来说, 当 $k > 0, n - k \geq 0$ 时,

$$X_{n-k} = ((a')^k X_n - c(a' + \cdots + (a')^k)) \bmod m = \\ ((a')^k X_n + ((a')^k - 1)c/b) \bmod m$$

如果 a 不与 m 互素, 则当给定 X_n 时, 不可能确定 X_{n-1} 。 $m/\gcd(a, m)$ 的倍数可以加到 X_{n-1} 上而无需改变 X_n 的值(也见习题 3.2.1.3-7)。

3.2.1.1 小节

1. 设 c' 是同余式 $ac' \equiv c \pmod{m}$ 的解(于是, 如果 a' 是习题 3.2.1-5 的答案中的数, 则 $c' = a'c \bmod m$), 我们有

LDA X; ADD CPRIME; MUL A

这个加法运算有可能导致溢出。(由本章后边导出的一些结果, 可知最好的办法大概是取 $c = a$ 并以“INCA 1”代替 ADD, 以节省一个时间单位。这样, 如果 $X_0 = 0$, 则在周期结束之前将不出现溢出, 所以实际上溢出不会出现。)

2. RANDM STJ 1F

LDA XRAND

MUL 2F

SLAX 5

ADD 3F (或者, 如果 c 很小, 则代之以 INCA c)

STA XRAND

1H JNOV *

JMP *-1

XRAND CON X_0

2H CON a

3H CON c |

3. 令 $a' = aw \bmod m$, 并令 m' 使得 $mm' \equiv 1 \pmod{w}$ 。置 $y \leftarrow \text{lomult}(a', x)$, $z \leftarrow \text{himult}(a', x)$, $t \leftarrow \text{lomult}(m', y)$, $u \leftarrow \text{himult}(m, t)$ 。于是我们有 $mt \equiv a'x \pmod{w}$, 因此 $a'x - mt = (z - u)w$, 因此 $ax \equiv z - u \pmod{m}$; 由此得出 $ax \bmod m = z - u + [z < u]m$ 。

4. 当且仅当 $x \equiv y \pmod{2^e}$ 和 $-2^{e-1} \leq y < 2^{e-1}$ 时定义运算 $x \bmod 2^e = y$ 。由

$$Y_0 = X_0 \bmod 2^{32}, \quad Y_{n+1} = (aY_n + c) \bmod 2^{32}$$

定义的同余序列 $\langle Y_n \rangle$ 在 370 型机器上很容易计算, 因为对所有 2 的补码数 y 和 z , y 和 z 的乘积的低半是 $(yz) \bmod 2^{32}$, 而且因为忽略溢出的加法所提供的结果也是 $\bmod 2^{32}$ 的。这个序列有标准线性同余序列 $\langle X_n \rangle$ 的全部随机性, 因为 $Y_n \equiv X_n \pmod{2^{32}}$ 。其实, Y_n 的 2 的补码表示, 对所有 n 都等同于 X_n 的二进表示 [G. Marsaglia 和 T.A. Bray 在 CACM 11(1968), 757~759 中首先指出这一点]。

5. (a) 减法: LDA X; SUB Y; JANN * + 2; ADD M。 (b) 加法: LDA X; SUB M; ADD Y; JANN * + 2; ADD M。(注意, 如果 m 超过字大小的一半, 则指令“SUB M”必须在指令“ADD Y”之前。)

6. 这些序列实质上并无二致, 因为加常数 $(m - c)$ 和减常数 c 有相同的效果。由于这个操作必须和乘法相结合, 减法过程几乎不比加法过程更优(至少在 MIX 的情况下), 除非有必要避免影响溢出开关时。

7. 在 $z^{kr} - 1$ 的因子分解中出现有 $z^k - 1$ 的诸素因子。如果 r 是奇数, 则 $z^k + 1$ 的素因子出现于 $z^{kr} + 1$ 的因式分子中。而且 $z^{2k} - 1$ 等于 $(z^k - 1)(z^k + 1)$ 。

```

8. JOV      * + 1      (确保溢出是关闭的)
   LDA      X
   MUL      A
   STX      TEMP
   ADD      TEMP      加低半部到高半部
   JNOV     * + 2      如果大于等于 w 则减 w - 1
   INCA     1          在这一步不可能产生溢出

```

注: 由于在一台 e 个二进位 1 的补码计算机上加法是在 $\text{mod}(2^e - 1)$ 下进行的, 因此有可能把习题 4 和习题 8 的技术结合起来, 对于所有 1 的补码数 y 和 z (不管符号), 通过把乘积 yz 的两半 (各有 e 位) 加在一起产生出 $(yz) \bmod (2^e - 1)$ 。

9. a) 两边都等于 $aq \lfloor r/q \rfloor$ 。

b) 置 $t \leftarrow a(x \bmod q) - r \lfloor x/q \rfloor$, 其中 $r = m \bmod a$; 可以预先计算出常数 q 和 r 。然后 $ax \bmod m = t + \lfloor t < 0 \rfloor m$, 因为我们可以证明 $t > -m$: 显然 $a(x \bmod q) \leq a(q-1) < m$ 。如果 $0 < r \leq q$, 也有 $r \lfloor x/q \rfloor \leq r \lfloor (m-1)/q \rfloor = r \lfloor a + (r-1)/q \rfloor = ra \leq qa < m$; 而且 $a^2 \leq m$ 意味着 $r < a \leq q$ 。[B. A. Wichmann 和 I. D. Hill 发表的程序中隐含了这一技术, *Applied Stat.* **31** (1982), 190。]

10. 如果 $r > q$ 且 $x = m - 1$, 我们有 $r \lfloor x/q \rfloor \geq (q+1)(a+1) > m$ 。所以对于方法 9 b) 的有效性来说, $r \leq q$ 的条件是必要和充分的; 这意味着 $\frac{m}{q} - 1 \leq a \leq \frac{m}{q}$ 。

令 $t = \lfloor \sqrt{m} \rfloor$, 对于 $1 \leq q \leq t$ 来说, 区间 $\left[\frac{m}{q} - 1, \frac{m}{q} \right]$ 是不相交的, 而且依赖于 q 是否 m 的一个因子, 它们精确地包括 1 或 2 个整数。这些区间用做对于 $a > \sqrt{m}$ 的所有解; 它们也包括, 如果 $(\sqrt{m} \bmod 1) < 1/2$ 则 $a = t$ 以及如果 $m = t^2$, 则 $a = t - 1$ 的情况。因此“幸运乘数”的总的个数精确地为 $2 \lfloor \sqrt{m} \rfloor + \lfloor d(m)/2 \rfloor - \left[(\sqrt{m} \bmod 1) < \frac{1}{2} \right] - 1$, 其中 $d(m)$ 是 m 的因子的个数。

11. 我们可以假定 $a \leq \frac{1}{2}m$; 否则由 $(m-a)x \bmod m$ 我们可以得到 $ax \bmod m$ 。于是我们可以表示 $a = a'a'' - a'''$, 其中 a', a'' 和 a''' 都小于 \sqrt{m} ; 例如, 我们可以取 $a' = \sqrt{m} - 1$ 和 $a'' = \lceil a/a' \rceil$ 。由此得出, $ax \bmod m$ 是 $(a'(a''x \bmod m) \bmod m - (a'''x \bmod m)) \bmod m$, 而且内部的三个运算都可通过习题 9 来处理。

当 $m = 2^{31} - 1$ 时, 我们可利用 $m - 1$ 有 192 个因子这个事实来找 $m = q'a' + 1$

的那些情况,以简化一般的方法,因为 $r' = 1$ 。结果是,当 $a = 62089911$ 时,在这些因子中有 86 个导致幸运的 a'' 和 a''' 。这样的最好情况大概是 $a' = 3641, a'' = 17053, a''' = 62$, 因为 3641 和 62 两者都整除 $m - 1$ 。这个分解产生了下列方案:

$$t \leftarrow 17053(x \bmod 125929) - 16410 \lfloor x/125929 \rfloor$$

$$t \leftarrow 3641(t \bmod 589806) - \lfloor t/589806 \rfloor$$

$$t \leftarrow t - (62(x \bmod 34636833) - \lfloor x/34636833 \rfloor)$$

其中“ $-$ ”表示模 m 减法。模运算用做一个乘法和 一个减法, 因为 $x \bmod q = x - q \lfloor x/q \rfloor$, 而且运算 $\lfloor x/q \rfloor$ 已经完成了; 因此, 我们已经实施了 7 次乘法, 3 次除法和 7 次减法。然而甚至更好的是注意到 62089911 本身有 24 个因子; 它们导致 5 个具有 $a''' = 0$ 的适合的因子分解。例如, 当 $a' = 863$ 和 $a'' = 70317$ 时, 我们只须 6 次乘法, 2 次除法, 4 次减法:

$$t \leftarrow 883(x \bmod 2432031) - 274 \lfloor x/2432031 \rfloor$$

$$t \leftarrow 70317(t \bmod 30540) - 2467 \lfloor t/30540 \rfloor$$

[对于所有的 a 和 m , 乘法和除法的最坏次数能否减少到至多 11, 或者 12 是否最好的上限? 另外一个实现 12 次的方法请见习题 4.3.3-19。]

12. a) 令 $m = 9999998999 = 10^{10} - 10^3 - 1$ 。为了以 10 乘 $(x_9 x_8 \cdots x_0)_{10}$ 模 m , 使用 $10^{10} x_9 \equiv 10^3 x_9 + x_9$ 这一事实: 即加 $(x_9 000)_{10}$ 到 $(x_8 x_7 \cdots x_0 x_9)_{10}$ 。并且为避免循环移位, 想像这些数字被安排在轮子上: 就是把高阶数字 x_9 加到向左移三个位置的数字 x_2 上, 并指向 x_8 作为新的高阶数字。如果 $x_9 + x_2 \geq 10$, 则一个进位向左传播, 而如果这个进位一直前进到 x_8 的左边, 它不仅向 x_9 传播, 而且也向 x_2 的位置传播; 在最后安置停当之前, 它可能继续既从 x_9 也从 x_2 传播。(这些数也可能变成稍大于 m 。例如, 0999999900 变成 $9999999000 = m + 1$, 它又变成 $9999999009 = m + 10$ 。但是一个冗余的表示未必是有害的。)

b) 这是除以 10 的运算, 所以我们做的是 a) 的对立面: 循环地向左移动高阶数字的指针, 并从数字左边三位减去新的高阶数字。如果减的结果是负, 则以通常方式“借位”(算法 4.3.1S); 即前边一位数字减 1。借位也可能像在 a) 中那样传播, 但是绝不会超过高阶数字。这个运算保持诸数字为非负, 而且小于 m 。(因此, 除以 10 被证实是比乘以 10 容易些的。)

c) 我们能记住借位而不是传播它, 因为它可以加入到下一步的减法中。因此, 如果我们定义数字 x_n 和借位 b_n , 通过递推式

$$x_n = (x_{n-10} - x_{n-3} - b_n) \bmod 10 = x_{n-10} - x_{n-3} - b_n + 10b_{n+1}$$

我们通过对 n 的归纳法有 $9999999000^n \bmod 9999998999 = X_n$, 其中

$$X_n = (x_{n-1} x_{n-2} x_{n-3} x_{n-4} x_{n-5} x_{n-6} x_{n-7} x_{n+2} x_{n+1} x_n)_{10} - 1000b_{n+3} = \\ (x_{n-1} x_{n-2} \cdots x_{n-10})_{10} - (x_{n-1} x_{n-2} x_{n-3})_{10} - b_n$$

假定初始条件被建立使得 $X_0 = 1$ 。注意

$$10X_{n+1} = (x_n x_{n-1} x_{n-2} x_{n-3} x_{n-4} x_{n-5} x_{n-6} x_{n+3} x_{n+2} x_{n+1} 0)_{10} - 10000b_{n+4} =$$

$$mx_n + X_n$$

由此得出,对于所有的 $n \geq 0, 0 \leq X_n < m$ 。

d) 如果 $0 \leq U < m$, U/m 的十进表示的头一位数字是 $\lfloor 10U/m \rfloor$, 而且随继的数字是 $(10U \bmod m)/m$ 的十进表示; 例如, 参看 4.4 节中的方法 2a。因此 $U/m = (.u_1 u_2 \dots)_{10}$, 如果我们置 $U_0 = U$ 和 $U_n = 10U_{n-1} \bmod m = 10U_{n-1} - mu_n$ 的话。简略地说, $1/m$ 的数字是对于 $n = 1, 2, \dots, 10^n \bmod m$ 的前导数字, 即最终成为周期的一个序列。这些是在相反次序下 $10^{-n} \bmod m$ 的前导数字, 所以我们在 c) 已经计算过它们了。

一个严格的证明当然比仅仅接受它更为可取。设 λ 是使 $10^\lambda \equiv 1 \pmod{m}$ 的最小正整数, 并且对于所有 $n < 0$, 定义 $x_n = x_{n \bmod \lambda}$, $b_n = b_{n \bmod \lambda}$, $X_n = X_{n \bmod \lambda}$ 。于是 c) 中对于 x_n, b_n 和 X_n 的递推式对于所有整数 n 成立。如果 $U_0 = 1$, 由此得出 $U_n = X_{-n}$, 和 $u_n = x_{-n}$; 因此

$$\frac{999999900^n \bmod 9999998999}{9999998999} = (.x_{n-1} x_{n-2} x_{n-3} \dots)_{10}$$

e) 设 w 是计算机的字长, 使用递推式

$$x_n = (x_{n-k} - x_{n-l} - b_n) \bmod w = x_{n-k} - x_{n-l} - b_n + wb_{n+1}$$

其中 $0 < l < k$ 且 k 很大。于是 $(.x_{n-1} x_{n-2} x_{n-3} \dots)_w = X_n/m$, 其中 $m = w^k - w^l - 1$ 且 $X_{n+1} = (w^{k-1} - w^{l-1})X_n \bmod m$ 。对于 $n \geq 0$, 关系

$$X_n = (x_{n-1} \dots x_{n-k})_w - (x_{n-1} \dots x_{n-l})_w + b_n$$

成立; x_{-1}, \dots, x_{-k} 和 b_0 的值应该使得 $0 \leq X_0 < m$ 。

这样的随机数生成程序, 以及在下列习题中的类似程序, 是由 G. Marsaglia 和 A. Zaman 引进的 [Annals of Applied Probability 1 (1991), 462 ~ 480], 他们把它叫做带借位减法方法。他们的出发点是分母为 m 的分数的 w 进制表示。与线性同余序列的关系是由 Shu Tezuka 注意到, 而由 Tezuka, L'Ecuyer 和 Couture 详细分析的 [ACM Trans. Modeling and Computer Simulation 3 (1993), 315 ~ 331]。习题 3.2.1.2-22 中讨论了周期的长度。

13. 乘以 10 现在要求把被加的数字取负。为了这一目的, 把一个数表示成最后三位数字取负是方便的; 例如, $9876543210 = (9876544 \bar{7}9\bar{0})_{10}$ 。然后 $10 \text{ 乘 } (x_9 \dots x_3 \bar{x}_2 \bar{x}_1 \bar{x}_0)_{10}$ 是 $(x_8 \dots x_3 x' \bar{x}_1 \bar{x}_0 \bar{x}_9)_{10}$, 其中 $x' = x_9 - x_2$ 。类似地 $(x_9 \dots x_3 \bar{x}_2 \bar{x}_1 \bar{x}_0)_{10}$ 除以 10 是 $(x_0 x_9 \dots x_4 \bar{x}'' \bar{x}_2 \bar{x}_1)_{10}$, 其中 $x'' = x_0 - x_3$ 。递推式

$$x_n = (x_{n-3} - x_{n-10} - b_{n-1}) \bmod 10 = x_{n-3} - x_{n-10} - b_{n-1} + 10b_n$$

产生 $8999999101^n \bmod 9999999001 = X_n$, 其中

$$X_n = (x_{n-1} x_{n-2} x_{n-3} x_{n-4} x_{n-5} x_{n-6} x_{n-7} x_{n+2} x_{n+1} \bar{x}_n)_{10} + 1000b_{n+3} = (x_{n-1} x_{n-2} \dots x_{n-10})_{10} - (x_{n-1} x_{n-2} x_{n-3})_{10} + b_n$$

当把进制从 10 推广成 w 时, 我们发现, w 的逆幂模 $w^k - w^l + 1$ 由

$$x_n = (x_{n-l} - x_{n-k} - b_n) \bmod w = x_{n-l} - x_{n-k} - b_n + wb_{n+1}$$

生成(和习题 12 相同,但 k 和 l 交换)。

14. 温和的推广:对于任何小于或等于字长 w 的 b ,我们可以有效地来做除以 b 模 $b^k - b^l \pm 1$,因为当 $b < w$ 时对于 x_n 的递推几乎和当 $b = w$ 时的递推一样有效。

强的推广:如果我们定义

$$m = a_k b^k + \cdots + a_1 b - 1 \quad \text{和} \quad X_n = \left(\sum_{j=1}^k a_j (x_{n-1} \cdots x_{n-k})_b + c_n \right) (\text{sign } m)$$

则在 $X_n / |m| = (.x_{n-1} x_{n-2} \cdots)_b$ 的意义下,递推式

$$x_n = (a_1 x_{n-1} + \cdots + a_k x_{n-k} + c_n) \bmod b, \quad c_{n+1} = \left\lfloor \frac{a_1 x_{n-1} + \cdots + a_k x_{n-k} + c_n}{b} \right\rfloor$$

等价于 $X_n = b^{-1} X_{n-1} \bmod |m|$ 。应该把初始值 $x_{-1} \cdots x_{-k}$ 和 c_0 选择成使得 $0 \leq X_0 < |m|$;那我们将有对于 $n \geq 0$, $x_n = (bX_{n+1} - X_n) / |m|$ 。对于 $j < 0$ 出现于公式 $X_n / |m| = (.x_{n-1} x_{n-2} \cdots)_b$ 中的 x_j 的值被适当地当做 $x_{j \bmod k}$, 其中 $b^k \equiv 1 \pmod{m}$;这些值可能不同于开始时被提供的数 x_{-1}, \cdots, x_{-k} 。进位数字 c_n 将满足

$$\sum_{j=1}^k \min(0, a_j) \leq c_n < \sum_{j=1}^k \max(0, a_j)$$

如果初始的进位是在这个范围的话。

特殊情况 $m = b^k + b^l - 1$, 对于它 $a_j = \delta_{jl} + \delta_{jk}$, 是有特别意义的,因为它可被容易地计算; Marsaglia 和 Zaman 把它叫做带进位加法生成程序:

$$x_n = (x_{n-l} + x_{n-k} + c_n) \bmod b = x_{n-l} + x_{n-k} + c_n - bc_{n+1}$$

另一个潜在的有吸引力的可能性是在对于比如说 $b = 2^{31}$ 和 $m = 65430b^2 + b - 1$ 的生成程序中使用 $k = 2$ 。这个模数 m 是素数,而且周期长度结果是 $(m-1)/2$ 。3.3.4 小节的谱检验指出,平面之间的间隔是好的(很大的 ν 值),尽管与对于这个特定的模数 m 的其它乘数作比较,乘数 b^{-1} 当然是差的。

习题 3.2.1.2-22 包含关于带借位减法和带进位加法的模数的另外信息,这些模数导致极长的周期。

3.2.1.2 小节

1. 由定理 A, 周期长度是 m (参见习题 3)。

2. 是的,这些条件意味着定理 A 中的条件,因为 2^e 仅有的素因子是 2,而且任何奇数同 2^e 互素。(事实上,本习题的条件是必要和充分的。)

3. 由定理 A, 我们需要 $a \equiv 1 \pmod{4}$ 和 $a \equiv 1 \pmod{5}$ 。由 1.2.4 小节的定律 D, 这等价于 $a \equiv 1 \pmod{20}$ 。

4. 通过在 $m = 2^e$ 的情况下使用定理 A, 我们知道 $X_{2^{e-1}} \equiv 0 \pmod{2^{e-1}}$ 。对于 $m = 2^e$ 也使用定理 A, 我们知道 $X_{2^{e-1}} \not\equiv 0 \pmod{2^e}$ 。由此得出, $X_{2^{e-1}} = 2^{e-1}$ 。更一般地,我们可以使用等式 3.2.1-(6) 来证明,这个周期的第二半实质上 and 头一半

是相类似的, 因为 $X_{n+2^{e-1}} = (X_n + 2^{e-1}) \bmod 2^e$ 。(四分之一也是类似的, 见习题 21。)

5. 对 $p=3, 11, 43, 281, 86171$, 我们需要 $a \equiv 1 \pmod{p}$ 。由 1.2.4 小节的定律 D。这等价于 $a \equiv 1 \pmod{3 \cdot 11 \cdot 43 \cdot 281 \cdot 86171}$, 所以惟一的解是糟糕的乘数 $a=1$ 。

6. (参见上一习题。)同余式 $a \equiv 1 \pmod{3 \cdot 7 \cdot 11 \cdot 13 \cdot 37}$ 意味着, 对于 $0 \leq k \leq 8$, 解是 $a = 1 + 111111k$ 。

7. 使用引理 Q 的证明的符号, μ 是使得 $X_{\mu+\lambda} = X_\mu$ 的最小值; 所以它是使得 $Y_{\mu+\lambda} = Y_\mu$ 和 $Z_{\mu+\lambda} = Z_\mu$ 的最小值。这表明 $\mu = \max(\mu_1, \dots, \mu_t)$ 。可实现的最高的 μ 是 $\max(e_1, \dots, e_t)$, 但没有人会真正要实现它。

8. 我们有 $a^2 \equiv 1 \pmod{8}$; 所以 $a^4 \equiv 1 \pmod{16}$, $a^8 \equiv 1 \pmod{32}$, 等等。如果 $a \bmod 4 = 3$, 则 $a-1$ 是一个奇数的两倍; 所以 $(a^{2^{e-1}} - 1)/(a-1) \equiv 0 \pmod{2^e}$ 当且仅当 $(a^{2^{e-1}} - 1)/2 \equiv 0 \pmod{2^{e-1}/2}$, 它为真。

9. 借助于 Y_n 替换 X_n , 并作简化。如果 $X_0 \bmod 4 = 3$, 这个习题的公式不适用; 但它们确实适用于序列 $Z_n = (-X_n) \bmod 2^e$, 它实质上有相同的特性。

10. 对于奇素数 p , 仅当 $m=1, 2, 4, p^e$ 和 $2p^e$ 时。在所有其它情况下, 定理 B 的结果是对于欧拉定理(习题 1.2.4-28)的改进。

11. (a) $x+1$ 或 $x-1$ (但非二者) 将是 4 的倍数, 所以 $x \mp 1 = q2^f$, 其中 q 是奇数而 f 大于 1。(b) 在给定的情况下, $f < e$, 所以 $e \geq 3$ 。我们有 $\pm x \equiv 1 \pmod{2^f}$ 和 $\pm x \not\equiv 1 \pmod{2^{f+1}}$ 以及 $f > 1$ 。因此通过应用引理 P, 我们求得 $(\pm x)^{2^{e-f}-1} \not\equiv 1 \pmod{2^e}$, 而 $x^{2^{e-f}} = (\pm x)^{2^{e-f}} \equiv 1 \pmod{2^e}$ 。所以阶是 2^{e-f} 的一个因子, 而不是 2^{e-f-1} 的一个因子。(c) i 有阶 1; $2^e - 1$ 有阶 2; 因此当 $e \geq 3$ 时极大周期是 2^{e-2} , 而对于 $e \geq 4$, 就需要有 $f=2$, 即 $x \equiv 4 \pm 1 \pmod{8}$ 。

12. 如果 k 是 $p-1$ 的真因子, 而且如果 $a^k \equiv 1 \pmod{p}$, 则由引理 P, 我们有 $a^{kp^{e-1}} \equiv 1 \pmod{p^e}$ 。类似地, 如果 $a^{p-1} \equiv 1 \pmod{p^2}$, 则我们发现 $a^{(p-1)p^{e-2}} \equiv 1 \pmod{p^e}$ 。所以在这些情况下, a 不是本原的。反之, 如果 $a^{p-1} \not\equiv 1 \pmod{p^2}$, 则定理 1.2.4F 和引理 P 告诉我们, $a^{(p-1)p^{e-2}} \not\equiv 1 \pmod{p^e}$, 但是 $a^{(p-1)p^{e-1}} \equiv 1 \pmod{p^e}$ 。所以阶是 $(p-1)p^{e-1}$ 的因子而不是 $(p-1)p^{e-2}$ 的因子; 因此它有 kp^{e-1} 的形式, 其中 k 整除 $p-1$ 。但如果 a 在模 p 下是本原的, 则同余式 $a^{kp^{e-1}} \equiv a^k \equiv 1 \pmod{p}$ 意味着 $k = p-1$ 。

13. 设 $a \bmod p \neq 0$ 且设 λ 是 a 的模 p 阶。由定理 1.2.4F, λ 是 $p-1$ 的因子。如果 $\lambda < p-1$, 则 $(p-1)/\lambda$ 有一个素因子 q 。

14. 设 $0 < k < p$ 。如果 $a^{p-1} \equiv 1 \pmod{p^2}$, 则 $(a+kp)^{p-1} \equiv a^{p-1} + (p-1) \cdot a^{p-2}kp \pmod{p^2}$; 而这是 $\not\equiv 1$ 的, 因为 $(p-1)a^{p-2}k$ 不是 p 的倍数。由习题 12, $a+kp$ 是模 p^e 本原的。

15. (a) 如果 $\lambda_1 = p_1^{e_1} \cdots p_{t_1}^{e_{t_1}}$, $\lambda_2 = p_1^{f_1} \cdots p_{t_2}^{f_{t_2}}$, 设 $\kappa_1 = p_1^{g_1} \cdots p_{t_1}^{g_{t_1}}$, $\kappa_2 = p_1^{h_1} \cdots p_{t_2}^{h_{t_2}}$, 其中

$$g_j = e_j \text{ 且 } h_j = 0, \quad \text{对于 } e_j < f_j$$

$$g_j = 0 \text{ 且 } h_j = f_j, \quad \text{对于 } e_j \geq f_j$$

现在, $a_1^{e_1}$ 和 $a_2^{e_2}$ 有周期 λ_1/κ_1 和 λ_2/κ_2 , 而且后者互素。其次, $(\lambda_1/\kappa_1)(\lambda_2/\kappa_2) = \lambda$, 所以只须考虑 λ_1 与 λ_2 互素的情况, 也就是当 $\lambda = \lambda_1\lambda_2$ 时的情况。现在由于 $(a_1a_2)^\lambda \equiv 1$, 我们有 $1 \equiv (a_1a_2)^{\lambda\lambda_1} \equiv a_2^{\lambda\lambda_1}$; 因此 $\lambda\lambda_1$ 是 λ_2 的倍数。这意味着 λ 是 λ_2 的倍数, 因为 λ_1 与 λ_2 互素。类似地, λ 是 λ_1 的倍数; 因此 λ 是 $\lambda_1\lambda_2$ 的倍数。但显然 $(a_1a_2)^{\lambda_1\lambda_2} \equiv 1$, 所以 $\lambda = \lambda_1\lambda_2$ 。

(b) 如果 a_1 有阶 $\lambda(m)$, a_2 有阶 λ , 则由 (a) 可知, $\lambda(m)$ 必须是 λ 的倍数, 否则我们可以找出一个更高阶的元素, 即 $\text{lcm}(\lambda, \lambda(m))$ 阶的元素。

$$16. a) f(x) = (x-a)(x^{n-1} + (a+c_1)x^{n-2} + \cdots + (a^{n-1} + \cdots + c_{n-1})) + f(a).$$

b) 当 $n=0$ 时这个命题是显然的。如果 a 是一个根, 则 $f(x) \equiv (x-a)q(x)$; 因此如果 a' 是任意其它的根, 则

$$0 \equiv f(a') \equiv (a' - a)q(a')$$

而且由于 $a' - a$ 不是 p 的倍数, a' 必是 $q(x)$ 的根。所以如果 $f(x)$ 有多于 n 个不同的根, 则 $q(x)$ 就有多于 $n-1$ 个不同的根。c) $\lambda(p) \geq p-1$, 因为为了具有这样多的根, $f(x)$ 必须有大于等于 $p-1$ 的次数。但由定理 1.2.4F, $\lambda(p) \leq p-1$ 。

17. 由引理 P, $11^5 \equiv 1 \pmod{25}$, $11^5 \not\equiv 1 \pmod{125}$, 等等; 所以 11 的阶是 $5^{e-1} \pmod{5^e}$, 不是极大值 $\lambda(5^e) = 4 \cdot 5^{e-1}$ 。但由引理 Q, 总的周期长度是周期模 2^e (即 2^{e-2}) 和周期模 5^e (即 5^{e-1}) 的最小公倍数, 而这就是 $2^{e-2}5^{e-1} = \lambda(10^e)$ 。周期模 5^e 可以是 5^{e-1} 或 $2 \cdot 5^{e-1}$ 或 $4 \cdot 5^{e-1}$, 而不影响周期模 10^e 的长度, 因为取了最小公倍数。在模 5^e 下为本原的一些值, 是同余于 2, 3, 8, 12, 13, 17, 22, 23 模 25 的那些值 (参考习题 12), 即 3, 13, 27, 37, 53, 67, 77, 83, 117, 123, 133, 147, 163, 173, 187, 197。

18. 根据定理 C, $a \pmod{8}$ 必然是 3 或 5。知道了 a 模 5 和模 25 的周期, 我们就可以应用引理 P 以确定 $a \pmod{25}$ 的允许的值。周期 = $4 \cdot 5^{e-1}$: 2, 3, 8, 12, 13, 17, 22, 23; 周期 = $2 \cdot 5^{e-1}$: 4, 9, 14, 19; 周期 = 5^{e-1} : 6, 11, 16, 21。这 16 个值的每一个产生出 a 的一个值, $0 \leq a < 200$, 满足 $a \pmod{8} = 3$, 以及另一个 a 的值, 满足 $a \pmod{8} = 5$ 。

19. 表 3.3.4-1 的行 17~20 中有若干个例子。

20. a) 当且仅当 $Y_n \equiv Y_{n+k} \pmod{m'}$ 时我们有 $AY_n + X_0 \equiv AY_{n+k} + X_0 \pmod{m}$ 。b) (i) 显然。(ii) 定理 A。(iii) 当且仅当 $a^n \equiv 1 \pmod{2^{e+1}}$ 时 $(a^n - 1)/(a - 1) \equiv 0 \pmod{2^e}$; 如果 $a \not\equiv -1$, a 模 2^{e+1} 的阶是它模 2^e 的阶的两倍。(iv) 当且仅当 $a^n \equiv 1$ 时, $(a^n - 1)/(a - 1) \equiv 0 \pmod{p^e}$ 。

21. 由等式 3.2.1-(6) 知 $X_{n+s} \equiv X_n + X_s$; 而且 s 是 m 的因子, 因为当 m 是 p 的幂时 s 是 p 的幂。因此一个给定的整数 q 是 m/s 的倍数当且仅当 $X_q \equiv 0$, 当且仅当 q 是 $m/\gcd(X_s, m)$ 的倍数。

22. 算法 4.5.4P 当比如说 $b \approx 2^{32}$ 和 $l < k \approx 100$ 时有能力在相当的时间内测试形如 $m = b^k \pm b^l \pm 1$ 的数是否为素数; 这些计算应当在 b 进制之下进行使得 m 的特

殊形式加速平方模 m 的运算。(例如,考虑在十进制下平方模 9999998999。)当然,算法 4.5.4P 仅当已知 m 没有小的因子时才应被使用。

Marsaglia 和 Zaman[*Annals of Applied Probability* 1 (1991), 474~475]证明,当 b 是素数 $2^{32}-5$ 时, $m = b^{43} - b^{22} + 1$ 是有原根 b 的素数。这要求进行因子分解 $m-1 = b^{22}(b-1)(b^6 + b^5 + b^4 + b^3 + b^2 + b + 1)(b^{14} + b^7 + 1)$ 以便确定 b 的本原性。 $m-1$ 的 17 个素因子之一有 99 位十进数字。结果,我们可以确保序列 $x_n = (x_{n-22} - x_{n-43} - c_n) \bmod b = x_{n-22} - x_{n-43} - c_n + bc_{n+1}$ 当 $c_0 = 0$ 时对于种子值 $0 \leq x_{-1}, \dots, x_{-43} < b$ 的每一个非零的选择有长度 $m-1 \approx 10^{414}$ 。

然而,从生日间隔检验的观点(见 3.3.2J 小节)来看,对于 k 来说 43 是相当小的值,而且 22 相当接近于 $43/2$ 。“混合”的考虑表明,我们喜欢在 l/k 的连分数中头一些部分商是小的那些 k 和 l 的值。为了避免这个生成程序的潜在问题,好的想法是如 Lüscher 所建议的那样,抛弃某些数(参见 3.2.2 小节)。

这里是形如 $b^k \pm b^l \pm 1$ 的一些素数,当 $b = 2^{32}$ 和 $50 < k \leq 100$ 时满足混合的限制:对于带借位减法, $b^{57} - b^{17} - 1, b^{73} - b^{17} - 1, b^{86} - b^{62} - 1, b^{88} - b^{52} - 1, b^{95} - b^{61} - 1; b^{58} - b^{33} + 1, b^{62} - b^{17} + 1, b^{69} - b^{24} + 1, b^{70} - b^{57} + 1, b^{87} - b^{24} + 1$ 。对于带进位加法, $b^{56} + b^{22} - 1, b^{61} + b^{44} - 1, b^{74} + b^{27} - 1, b^{90} + b^{65} - 1$ 。(从混合的观点看不大合乎要求的是素数 $b^{56} - b^5 - 1, b^{56} - b^{32} - 1, b^{66} - b^{57} - 1, b^{76} - b^{15} - 1, b^{84} - b^{26} - 1, b^{90} - b^{42} - 1, b^{93} - b^{18} - 1; b^{52} - b^8 + 1, b^{60} - b^{12} + 1, b^{67} - b^8 + 1, b^{67} - b^{63} + 1, b^{83} - b^{14} + 1; b^{65} + b^2 - 1, b^{76} + b^{11} - 1, b^{88} + b^{30} - 1, b^{92} + b^{48} - 1$ 。)

为了计算得到的序列的周期,我们需要知道 $m-1$ 的因子;但对于这样大的数这是不可行的,除非我们极端幸运。假设我们在寻找素因子 q_1, \dots, q_t 中确实成功,则 $b^{(m-1)/q} \bmod m = 1$ 的概率极小,仅为 $1/q$,除非 q 是非常小的素数。因此,我们可以十分确信,即使不能对 $m-1$ 进行因子分解, $b^n \bmod m$ 周期也是极长的。

确实,即使 m 不是素数,这个周期几乎肯定地是非常长的。例如,考虑 $k=10, l=3, b=10$ 的情况(对于随机数生成来说它是太小了,但是足以令我们能容易地计算出精确结果)。在这种情况下, $\langle 10^n \bmod m \rangle$ 当 $m = 9999998999 = 439 \cdot 22779041$ 时有周期长度 $\text{lcm}(219, 11389520) = 2494304880$; 而当 $m = 9999999001$ 时为 4999999500 ; 当 $m = 10000000999$ 时为 5000000499 ; 以及当 $m = 10000001001 = 3 \cdot 17 \cdot 2687 \cdot 72973$ 时,为 $\text{lcm}(1, 16, 2686, 12162) = 130668528$ 。当 m 不是素数时,种子的稀少选择可能缩短周期。但如果我们选择比如说 $k=1000, l=619$ 和 $b=2^{16}$, 我们几乎不出错。

3.2.1.3 小节

1. $c=1$ 总是与 B^5 互素;而且每一个整除 $m = B^5$ 的素数是 B 的一个因子,所以它整除 $b = B^2$ 至少到第二次幂。

2. 只有 3, 所以不管它的周期如何长,也不推荐这个生成程序。

3. 在两种情况下,效能都是 18(见下题)。

4. 由于 $a \bmod 4 = 1$, 我们必然有 $a \bmod 8 = 1$ 或 5 , 所以 $b \bmod 8 = 0$ 或 4 。如果 b 是 4 的奇数倍, 而且如果 b_1 是 8 的倍数, 则显然 $b^s \equiv 0 \pmod{2^e}$ 意味着 $b_1^s \equiv 0 \pmod{2^e}$, 所以 b_1 不能有高于 b 的效能。

5. 效能是对于所有的 j , 使得 $f_j s \geq e_j$ 的最小 s 值。

6. 为了有像 4 这样高的效能, 模必须能被 2^7 或 p^4 (p 为奇素数) 整除。仅有的值是 $m = 2^{27} + 1$ 和 $10^9 - 1$ 。

7. $a' = (1 - b + b^2 - \cdots) \bmod m$, 其中 b^s, b^{s+1} 等中的项都被去掉了 (如果 s 是效能的话)。

8. 由于 X_n 总是奇数, $X_{n+2} = (2^{34} + 3 \cdot 2^{18} + 9) X_n \bmod 2^{35} = (2^{34} + 6X_{n+1} - 9X_n) \bmod 2^{35}$ 。给定 Y_n 和 Y_{n+1} , $Y_{n+2} \approx (5 + 6(Y_{n+1} + \epsilon_1) - 9(Y_n + \epsilon_2)) \bmod 10$ 的可能性是有限而且是非随机的, 其中 $0 \leq \epsilon_1 < 1, 0 \leq \epsilon_2 < 1$ 。

注: 如果在习题 3 中所提出的乘数, 比如说是 $2^{33} + 2^{18} + 2^2 + 1$, 而不是 $2^{23} + 2^{13} + 2^2 + 1$, 则我们将类似地求出 $X_{n+2} \approx 10X_{n+1} + 25X_n \equiv \text{常数} \pmod{2^{35}}$ 。一般来说, 当 δ 很小时, 我们不要求 $a \pm \delta$ 为 2 的高次幂所整除, 否则我们就得到“二阶低效能”。关于更详细的讨论, 见 3.3.4 小节。

在 MacLaren 和 Marsaglia 的论文中可找到有关本习题中的生成程序的讨论, 见 JACM 12(1965), 83~89。M. Greenberger 首先揭示了这种生成程序的缺陷, CACM 8(1965), 177~179。但是这样的生成程序后来仍然广泛使用达 10 年以上 (参看 3.3.4 小节中关于 RANDU 的讨论)。

3.2.2 小节

1. 使用这个方法时必需非常小心。首先, aU_n 可能是如此之大, 使得后面的对 c/m 的加法几乎将失去所有有效位, 而且“mod 1”操作将近乎破坏可能剩下的任何有效位的痕迹。我们的结论是, 双精度浮点算术是必要的。即使用了双精度, 人们必须确保无论如何不会出现舍入等等事情, 来影响序列的数, 因为这将破坏这个序列好的特性的理论基础 (但见习题 23)。

2. X_{n+1} 或者等于 $X_{n-1} + X_n$ 或者等于 $X_{n-1} + X_n - m$ 。如果 $X_{n+1} < X_n$, 我们必定有 $X_{n+1} = X_{n-1} + X_n - m$; 因此 $X_{n+1} < X_{n-1}$ 。

3. (a) 在步骤 M3 之后有下划线的数是 $V[j]$ 。所以效能已经减小成 $1!$ (见习题 15 的答案中的进一步的注解。)

输出: 初始的	0	4	5	6	2	0	3	(2	7	4	1	6	3	0	5)	并	重	复
V[0]:	0	<u>4</u>	<u>7</u>	7	7	7	7	7	<u>4</u>	<u>7</u>	7	7	7	7	7	7	<u>4</u>	<u>7</u> ...
V[1]:	3	3	3	3	3	3	<u>2</u>	<u>5</u>	5	5	5	5	5	5	<u>2</u>	<u>5</u>	5	5 ...
V[2]:	2	2	2	2	<u>0</u>	<u>3</u>	3	3	3	3	3	<u>0</u>	<u>3</u>	3	3	3	3	3 ...
V[3]:	5	5	<u>6</u>	<u>1</u>	1	1	1	1	1	<u>6</u>	<u>1</u>	1	1	1	1	1	1	1 ...
X:		4	7	6	1	0	3	2	5	4	7	6	1	0	3	2	5	4 7 ...
Y:		0	1	6	7	4	5	2	3	0	1	6	7	4	5	2	3	0 1 ...

(b) 在步骤 B2 之后, 有下划线的数是 $V[j]$ 。在这种情况下输出比输入好得多:

输出:	初始的	2	3	6	5	7	0	0	5	3	...	4	6	(3 0 ... 4 7) ...
$V[0]:$	0	0	0	0	0	0	<u>5</u>	<u>4</u>	4	...	1	1	1 1	...
$V[1]:$	3	3	<u>6</u>	<u>1</u>	1	1	1	1	1	...	0	0	0 4	...
$V[2]:$	2	<u>7</u>	7	7	<u>3</u>	3	3	3	<u>7</u>	...	6	<u>2</u>	2 2	...
$V[3]:$	5	5	5	5	<u>0</u>	0	<u>2</u>	2	2	...	<u>3</u>	3	<u>5</u> 5	...
$X:$	4	7	6	1	0	3	2	5	4	7	...	3	2	5 4

在 46 步之后它进入长度为 40 的一个重复循环: 236570 05314 72632 40110 37564 76025 12541 73625 03746 (30175 24061 52317 46203 74531 60425 16753 02647)。通过把习题 3.1-7 的方法应用到上边的数组直到有一个列重复出现为止, 可以容易地找到这个循环。

4. 许多随机序列的低位字节(例如, 对于 $m =$ 字长的线性同余序列)比起高位字节的随机性要差。见 3.2.1.1 小节。

5. 随机化的效果将被极小化, 因为 $V[j]$ 总是包含某个范围内的一个数, 实质上有 $j/k \leq V[j]/m < (j+1)/k$ 。然而, 可以使用某些类似的方法: 我们可以取 $Y_n = X_{n-1}$, 或者可以通过从大约中间处而不是在极左端处抽取数字来从 X_n 选择 j 。这些建议中没有一个是将产生类似于算法 B 那样的周期增长。(然而, 习题 27 表明, 算法 B 不必增加周期长度。)

6. 例如, 如果 $x_n/m < \frac{1}{2}$, 则 $x_{n+1} = 2x_n$ 。

7. [W. Mantel, *Nieuw Archief voor Wiskunde* (2) 1 (1897), 172~184.]

	00...01		00...01
	00...10		00...10
x 值子序列:	...	变成	...
	10...00		10...00
	CONTENTS(A)		00...00
			CONTENTS(A)

8. 如同在定理 3.2.1.2A 的证明中那样, 我们可以假设 $X_0 = 0, m = p^e$ 。首先假定这个序列有周期长度 p^e ; 由此得出这个序列的周期模 p^f 有长度 $p^f, 1 \leq f \leq e$, 否则模 p^f 的某些余数将不会出现。显然, c 不是 p 的倍数, 否则每个 X_n 都将是 p 的倍数。如果 $p \leq 3$, 则通过反复试验, 容易确定条件 iii) 和 iv) 的必要性, 所以我们可以假定 $p \geq 5$ 。如果 $d \not\equiv 0 \pmod{p}$, 则对于某些整数 a_1 和 c_1 及对所有整数 x , $dx^2 + ax + c \equiv d(x + a_1)^2 + c_1 \pmod{p^e}$; 这个二次式在点 x 和 $-x - 2a_1$ 取相同的值, 所以它不可能取模 p^e 的所有值。因此 $d \equiv 0 \pmod{p}$; 而且如果 $a \not\equiv 1$, 则对某个 x 我们将有 $dx^2 + ax + c \equiv x \pmod{p}$, 这与这个序列模 p 有周期长度 p 的事实矛盾。

为证明这个条件的充分性, 由定理 3.2.1.2A 和某些平凡情况的考虑, 我们可以

假定 $m = p^e$, 其中 $e \geq 2$ 。如果 $p = 2$, 则通过试验我们有 $X_{n+2} \equiv X_n + 2 \pmod{4}$; 而如果 $p = 3$, 使用 i) 和 ii), 我们有 $X_{n+3} \equiv X_n - d + 3c \pmod{9}$ 。对于 $p \geq 5$, 可以证明 $X_{n+p} \equiv X_n + pc \pmod{p^2}$: 令 $d = pr, a = 1 + ps$ 。于是如果 $X_n \equiv cn + pY_n \pmod{p^2}$, 我们必定有 $Y_{n+1} \equiv n^2c^2r + ncs + Y_n \pmod{p}$; 因此, $Y_n \equiv \binom{n}{3}2c^2r + \binom{n}{2}(c^2r + cs) \pmod{p}$ 。于是, $Y_p \pmod{p} = 0$, 而且证明了所要求的关系。

现在我们可以证明, 对于某些满足 $t \pmod{p} \neq 0$ 的 t , 以及对于所有的 $f \geq 1$, “提示”中定义的整数序列 $\langle X_n \rangle$ 满足关系式

$$X_{n+p^f} \equiv X_n + tp^f \pmod{p^{f+1}}, \quad n \geq 0$$

这只要证明序列 $\langle X_n \pmod{p^e} \rangle$ 有周期长度 p^e 就足够了, 因为周期的长度是 p^e 的因子而不是 p^{e-1} 的因子。上边的关系式对于 $f=1$ 已经建立, 而对于 $f > 1$, 可以用下列方式通过归纳法证明: 设

$$X_{n+p^f} \equiv X_n + tp^f + Z_np^{f+1} \pmod{p^{f+2}}$$

则用于生成这个序列的二次律, 连同 $d = pr, a = 1 + ps$ 一起, 产生 $Z_{n+1} = 2rtnc + st + Z_n \pmod{p}$ 。由此得出 $Z_{n+p} \equiv Z_n \pmod{p}$; 因此对于 $k = 1, 2, 3, \dots$,

$$X_{n+kp^f} \equiv X_n + k(tp^f + Z_np^{f+1}) \pmod{p^{f+2}}$$

置 $k = p$ 就完成了证明。

注: 如果 $f(x)$ 是次数高于 2 的多项式, 而且 $X_{n+1} = f(X_n)$, 则分析将更复杂, 尽管我们可以使用 $f(m + p^k) = f(m) + p^kf'(m) + p^{2k}f''(m)/2! + \dots$ 的事实来证明许多多项式递推式给出极大周期。例如, Coveyou 已经证明, 如果 $f(0)$ 为奇数, 对于 $j = 0, 1, 2, 3, f'(j) \equiv 1, f''(j) \equiv 0$, 以及 $f(j+1) \equiv f(j) + 1 \pmod{4}$, 则周期是 $m = 2^e$ [Studies in Applied Math. 3 (Philadelphia: SIAM, 1969), 70~111]。

9. 设 $X_n = 4Y_n + 2$; 则序列 Y_n 满足二次递推式 $Y_{n+1} = (4Y_n^2 + 5Y_n + 1) \pmod{2^{e-2}}$ 。

10. 情况 1: $X_0 = 0, X_1 = 1$; 因此 $X_n \equiv F_n$ 。我们探求使得 $F_n \equiv 0$ 及 $F_{n+1} \equiv 1 \pmod{2^e}$ 的最小的 n 。因为 $F_{2n} = F_n(F_{n-1} + F_{n+1}), F_{2n-1} = F_n^2 + F_{n+1}^2$, 我们对 e 用归纳法发现, 对于 $e > 1, F_{3 \cdot 2^{e-1}} \equiv 0$ 和 $F_{3 \cdot 2^{e-1}+1} \equiv 2^e + 1 \pmod{2^{e+1}}$ 。这意味着周期是 $3 \cdot 2^{e-1}$ 的因子, 但不是 $3 \cdot 2^{e-2}$ 的因子, 所以它或者是 $3 \cdot 2^{e-1}$, 或者是 2^{e-1} 。但 $F_{2^{e-1}}$ 总是奇数 (因为仅仅 F_{3n} 是偶数)。

情况 2: $X_0 = a, X_1 = b$, 则 $X_n \equiv aF_{n-1} + bF_n$; 我们需要找出满足 $a(F_{n+1} - F_n) + bF_n \equiv a$ 及 $aF_n + bF_{n+1} \equiv b$ 的最小正数 n 。这意味着 $(b^2 - ab - a^2)F_n \equiv 0, (b^2 - ab - a^2)(F_{n+1} - 1) \equiv 0$ 。且 $b^2 - ab - a^2$ 是奇数 (即与 m 互素), 所以这个条件等价于 $F_n \equiv 0, F_{n+1} \equiv 1$ 。

确定对于任意模, F_n 的周期的一些方法见 D. D. Wall 的文章, AMM 67 (1960),

525~532。关于斐波那契序列模 2^e 进一步的事实已由 B. Jansson 作了推导 [Random Number Generators (Stockholm: Almqvist & Wiksell, 1966), Section 3C1]。

11. a) 对于某个 $u(z)$ 和 $v(z)$, 其中 $v(z) \not\equiv 0 \pmod{f(z) \text{ 和 } p}$, 我们有 $z^\lambda = 1 + f(z)u(z) + p^e v(z)$ 。由二项式定理可知

$$z^{\lambda p} = 1 + p^{e+1}v(z) + p^{2e+1}v(z)^2(p-1)/2$$

加上同余于零 (modulo $f(z)$ 和 p^{e+2}) 的另一些项。由于 $p^e > 2$, 我们有 $z^{\lambda p} = 1 + p^{e+1} \cdot v(z) \pmod{f(z) \text{ 和 } p^{e+2}}$ 。如果 $p^{e+1}v(z) \equiv 0 \pmod{f(z) \text{ 和 } p^{e+2}}$, 则必存在多项式 $a(z)$ 和 $b(z)$, 使得 $p^{e+1}(v(z) + pa(z)) = f(z)b(z)$ 。因为 $f(0) = 1$, 这意味着 $b(z)$ 是 p^{e+1} 的倍数 (由高斯引理 4.6.1G); 因此 $v(z) \equiv 0 \pmod{f(z) \text{ 和 } p}$, 矛盾。

b) 如果 $z^\lambda - 1 = f(z)u(z) + p^e v(z)$, 则

$$G(z) = u(z)/(z^\lambda - 1) + p^e v(z)/f(z)(z^\lambda - 1)$$

因此对于很大的 n , $A_{n+\lambda} \equiv A_n \pmod{p^e}$ 。反之, 如果 $\langle A_n \rangle$ 有后一性质, 则对于某整系数多项式 $u(z)$ 和 $v(z)$ 以及某个整系数幂级数 $H(z)$ 有 $G(z) = u(z) + v(z)/(1 - z^\lambda) + p^e H(z)$ 。这意味着有恒等式 $1 - z^\lambda = u(z)f(z)(1 - z^\lambda) + v(z)f(z) + p^e H(z)f(z)(1 - z^\lambda)$; 而且 $H(z)f(z)(1 - z^\lambda)$ 是一个多项式, 因为等式的其它项都是多项式。

c) 我们只需证明由 $\lambda(p^e) \neq \lambda(p^{e+1})$ 可推出 $\lambda(p^{e+1}) = p\lambda(p^e) \neq \lambda(p^{e+2})$ 就够了。应用 a) 和 b), 我们知道 $\lambda(p^{e+2}) \neq p\lambda(p^e)$, 而且 $\lambda(p^{e+1})$ 是 $p^\lambda(p^e)$ 的因子, 但不是 $\lambda(p^e)$ 的因子。因此, 如果 $\lambda(p^e) = p^f q$, 其中 $q \bmod p \neq 0$, 则 $\lambda(p^{e+1})$ 必然是 $p^{f+1}d$, 其中 d 是 q 的一个因子。但现在 $X_{n+p^{e+1}} \equiv X_n \pmod{p^e}$; 因此 $p^{f+1}d$ 是 $p^f q$ 的倍数, 因此 $d = q$ 。[注: 假设 $p^e > 2$ 是必需的; 例如, 设 $a_1 = 4, a_2 = -1, k = 2$; 则 $\langle A_n \rangle = 1, 4, 15, 56, 209, 780, \dots; \lambda(2) = 2, \lambda(4) = 4, \lambda(8) = 4。$]

d) $g(z) = X_0 + (X_1 - a_1 X_0)z + \dots + (X_{k-1} - a_1 X_{k-2} - a_2 X_{k-3} - \dots - a_{k-1} X_0)z^{k-1}$ 。

e) 可以把 b) 中的推导推广成为 $G(z) = g(z)/f(z)$ 的情况, 则周期长度 λ 的假定意味着 $g(z)(1 - z^\lambda) \equiv 0 \pmod{f(z) \text{ 和 } p^e}$ 。我们仅仅讨论上面 $g(z) = 1$ 的特殊情况。但这同余式的两边可乘以 Hensel 的 $b(z)$, 而且我们得到 $1 - z^\lambda \equiv 0 \pmod{f(z) \text{ 和 } p^e}$ 。

注: 无须使用生成函数, 而使用类似于习题 8 答案中的那些方法, 可以给出 c) 中的结果的更为“初等”的证明: 如果对于 $n = r, r+1, \dots, r+k-1$ 及某些整数 B_n , $A_{\lambda+n} - A_n + p^e B_n$, 则若通过给定的递推关系来定义 $B_{r+k}, B_{r+k+1}, \dots$, 那么对于所有 $n \geq r$, 这同一关系成立。由于得到的 B 序列是 A 序列移位的某些线性组合, 我们将有, 对于所有足够大的 n 值, $B_{\lambda+n} \equiv B_n \pmod{p^e}$ 。现在 $\lambda(p^{e+1})$ 必然是 $\lambda = \lambda(p^e)$ 的某个倍数; 对于所有充分大的 n , 对于 $j = 1, 2, 3, \dots$, 我们有 $A_{n+j\lambda} = A_n + p^e (B_n + B_{n+\lambda} + B_{n+2\lambda} + \dots + B_{n+(j-1)\lambda}) \equiv A_n + jp^e B_n \pmod{p^{2e}}$ 。没有 k 个连续的

B 是 p 的倍数; 因此当 $e \geq 2$ 时可直接得出 $\lambda(p^{e+1}) = p\lambda(p^e) \neq \lambda(p^{e+2})$ 。我们还必须证明, 当 p 是奇数和 $e = 1$ 时, $\lambda(p^{e+2}) \neq p\lambda(p^e)$ 。这里我们假设 $B_{\lambda+n} = B_n + pC_n$, 并注意当 n 足够大时 $C_{n+\lambda} \equiv C_n \pmod{p}$ 。于是 $A_{n+p} \equiv A_n + p^2 \left(B_n + \binom{p}{2} C_n \right) \pmod{p^3}$, 因而这个证明可容易地完成。

关于这个问题的历史, 见 Morgan Ward, *Trans. Amer. Math. Soc.* **35** (1933), 600~628; 也见 D. W. Robinson, *AMM* **73** (1966), 619~621。

12. 周期长度模 2 顶多是 4; 而由上题的考虑周期长度模 2^{e+1} 顶多是极大长度模 2^e 的两倍。所以极大的可以想像的周期长度是 2^{e+1} ; 这是可以达到的, 比如说, 在 $a=0, b=c=1$ 的平凡情况下。

13, 14. 显然, $Z_{n+\lambda} = Z_n$, 所以 λ' 肯定是 λ 的一个因子。设 λ' 和 λ_1 的最小公倍数是 λ'_1 , 并类似地定义 λ'_2 , 我们有 $X_n + Y_n \equiv Z_n \equiv Z_{n+\lambda'_1} \equiv X_n + Y_{n+\lambda'_1}$, 所以 λ'_1 是 λ_2 的倍数。类似地, λ'_2 是 λ_1 的倍数。这就得出了所求的结果(在能构造 $\lambda' = \lambda_0$ 的序列以及构造 $\lambda' = \lambda$ 的序列这一意义下, 这个结果是“最好的”)。

15. 对所有充分大的 n , 算法 M 在步骤 M1 生成 (X_{n+k}, Y_n) , 并在步骤 M3 输出 $Z_n = X_{n+k-q_n}$ 。于是 $\langle Z_n \rangle$ 有长度为 λ' 的周期, 其中 λ' 是对于所有很大的 n , 使得 $X_{n+k-q_n} = X_{n+\lambda'+k-q_{n+\lambda'}}$ 的最小正整数。由于 λ 是 λ_1 和 λ_2 的倍数, 由此得出 λ' 是 λ 的一个因子(这些发现来自 Alon G. Waterman)。

对所有很大的 n , 由诸 X 的不同性我们也有 $n+k-q_n \equiv n+\lambda'+k-q_{n+\lambda'} \pmod{\lambda_1}$ 。 $\langle q_n \rangle$ 的界意味着对所有很大的 n , $q_{n+\lambda'} = q_n + c$, 其中 $c \equiv \lambda' \pmod{\lambda_1}$ 且 $|c| < \frac{1}{2}\lambda_1$ 。但由于 $\langle q_n \rangle$ 有界, c 必须为 0。因此 $\lambda' \equiv 0 \pmod{\lambda_1}$, 而且对于所有很大的 n , $q_{n+\lambda'} = q_n$; 由此得出, λ' 是 λ_2 和 λ_1 的倍数, 所以 $\lambda' = \lambda$ 。

注: 习题 3.2.1.2-4 的答案意味着当 $\langle Y_n \rangle$ 是极大周期模 $m = 2^e$ 的一个线性同余序列, 且 k 是 2 的幂时, 周期长度 λ_2 将至多是 2^{e-2} 。

16. 有好几个证明方法。

(1) 利用有限域理论。在具有 2^k 个元素的域中设 ξ 满足 $\xi^k = a_1 \xi^{k-1} + \cdots + a_k$ 。设 $f(b_1 \xi^{k-1} + \cdots + b_k) = b_k$, 其中每一个 b_j 是 0 或者是 1; 这是一个线性函数。在执行 (10) 之前如果在生成算法中字 x 是 $(b_1 b_2 \cdots b_k)_2$, 而且如果 $b_1 \xi^{k-1} + \cdots + b_k \xi^0 = \xi^n$, 则在执行 (10) 之后字 x 表示 ξ^{n+1} 。因此这个序列是 $f(\xi^n), f(\xi^{n+1}), f(\xi^{n+2}), \dots$; 且 $f(\xi^{n+k}) = f(\xi^n \xi^k) = f(a_1 \xi^{n+k-1} + \cdots + a_k \xi^n) = a_1 f(\xi^{n+k-1}) + \cdots + a_k f(\xi^n)$ 。

(2) 利用硬算, 或者初等的技巧。我们得到一个序列 $X_{nj}, n \geq 0, 1 \leq j \leq k$, 满足

$$X_{(n+1)j} \equiv X_{n(j+1)} + a_j X_{n1}, \quad 1 \leq j < k; \quad X_{(n+1)k} \equiv a_k X_{n1} \pmod{2}$$

我们必须证明, 这意味着对 $n \geq k, X_{nk} \equiv a_1 X_{(n-1)k} + \cdots + a_k X_{(n-k)k}$ 。其实, 它意味

着当 $1 \leq j \leq k \leq n$ 时, $X_{nj} \equiv a_1 X_{(n-1)j} + \cdots + a_k X_{(n-k)j}$ 。对于 $j=1$, 这是显然的, 因为 $X_{n1} \equiv a_1 X_{(n-1)1} + X_{(n-1)2} \equiv a_1 X_{(n-1)1} + a_2 X_{(n-2)1} + X_{(n-2)3}$, 等等。对于 $j > 1$, 通过归纳法我们有

$$X_{nj} \equiv X_{(n+1)(j-1)} - a_{j-1} X_{n1} \equiv \sum_{1 \leq i \leq k} a_i X_{(n+1-i)(j-1)} - a_{j-1} \sum_{1 \leq i \leq k} a_i X_{(n-i)1} \equiv \sum_{1 \leq i \leq k} a_i (X_{(n+1-i)(j-1)} - a_{j-1} X_{(n-i)1}) \equiv a_1 X_{(n-1)j} + \cdots + a_k X_{(n-k)j}$$

这个证明不与这些操作是在模 2, 还是在模任何素数下进行的有关。

17. (a) 当这个序列终止时, $k-1$ 元组 $(X_{n+1}, \cdots, X_{n+k-1})$ 第 $m+1$ 次出现。一个给定的 $k-1$ 元组 $(X_{r+1}, \cdots, X_{r+k-1})$ 仅能有 m 个不同的前驱 X_r , 所以这些出现之一必定是对于 $r=0$ 的。(b) 由于 $k-1$ 元组 $(0, \cdots, 0)$ 出现 $m+1$ 次, 因此每个可能的前驱都出现, 所以对于所有 $a_1, 0 \leq a_1 < m$, k 元组 $(a_1, 0, \cdots, 0)$ 出现。设 $1 \leq s < k$ 并假定我们已经证明当 $a_s \neq 0$ 时, 所有 k 元组 $(a_1, \cdots, a_s, 0, \cdots, 0)$ 出现于这个序列中。由构造方法可知, 这个 k 元组将不出现于这序列中, 除非对于 $1 \leq y < m$, $(a_1, \cdots, a_s, 0, \cdots, 0, y)$ 早先已经出现了。因此 $k-1$ 元组 $(a_1, \cdots, a_s, 0, \cdots, 0)$ 已经出现 m 次, 而且所有 m 个可能的前驱都出现; 这意味着对于 $0 \leq a < m$, $(a, a_1, \cdots, a_s, 0, \cdots, 0)$ 出现。由归纳法, 现在完成了证明。

利用习题 2.3.4.2-23 的有向图, 这个结果也可以从定理 2.3.4.2D 得出; 由 $(x_1, \cdots, x_j, 0, \cdots, 0)$ 到 $(x_2, \cdots, x_j, 0, 0, \cdots, 0)$ 的有向边的集合 (其中 $x_j \neq 0$ 且 $1 \leq j \leq k$), 形成同杜威十进记数法巧妙地相关的一个有向子树。

18. 由习题 16, U_{n+1} 的最高有效位完全由 U_n 的头一位和第三位确定, 所以 64 个可能的数偶 $(\lfloor 8U_n \rfloor, \lfloor 8U_{n+1} \rfloor)$ 中仅 32 个出现。(注: 如果我们用的是, 比如说, 11 位数 $U_n = (.X_{11n}X_{11n+1}\cdots X_{11n-10})_2$, 则这个序列对于许多应用来说就会令人满意了。如果出现于 A 中的另一个常数有更多的“1”的位, 则广义谱检验也许会给出关于它的适当性的某个指示。见习题 3.3.4-24; 我们可以在维数 $t = 36, 37, 38, \cdots$ 下, 考察 ν_t 。

21. [J. London Math. Soc. 21 (1946), 169~172.] 像习题 7 中那样, 在适当位置插入 0, 将使任何没有 k 个连续的 0 的周期长度为 $m^k - 1$ 的序列, 变成一个周期长度 m^k 的序列; 反之, 我们可以从周期长度为 m^k 的一个序列开始, 由这个周期删去适当的 0, 以形成其它类型的序列。我们称这些为类型 A 和 B 的“ (m, k) 序列”。对于所有的素数 p 和所有 $k \geq 1$, 这些假设使我们确信 (p, k) 序列的存在性; 因此对于所有这样的 p 和 k , 我们有类型 B 的 (p, k) 序列。

为得到类型 B 的一个 (p^r, k) 序列, 设 $e = qr$, 其中 q 是 p 的幂而 r 不是 p 的倍数。由类型 A 的一个 (p, qrk) 序列即 X_0, X_1, X_2, \cdots 开始; 然后 (利用 p 进数系统) 分组的数字 $(X_0, \cdots, X_{q-1})_p, (X_q, \cdots, X_{2q-1})_p, \cdots$ 形成类型 A 的一个 (p^q, rk) 序列, 因为 q 与 $p^{qrk} - 1$ 互素, 这个序列有 $p^{qrk} - 1$ 的周期长度。这就导致了类型 B 的一个

(p^q, rk) 序列 $\langle Y_n \rangle$; 而且通过类似的论证, $(Y_0 Y_1 \cdots Y_{r-1})_{p^q}, (Y_r Y_{r+1} \cdots Y_{2r-1})_{p^q}, \cdots$ 是类型 B 的 (p^q, k) 序列, 因为 r 与 p^q 互素。

为得到对于任意 m 的类型 B 的 (m, k) 序列, 利用中国剩余定理, 对于 m 的每个素数幂的因子, 我们可以把 (p^e, k) 序列结合在一起。但是还可利用一个更简单的方法。设 $\langle X_n \rangle$ 是类型 B 的 (r, k) 序列, $\langle Y_n \rangle$ 是类型 B 的 (s, k) 序列, 其中 r 与 s 互素, 则 $\langle (X_n + Y_n) \bmod rs \rangle$ 是类型 B 的 (rs, k) 序列。

A. Lempel 发现了对于任意 k 产生出 $(2, k)$ 序列的一个简单一致的构造方法 [IEEE Trans. C-19 (1970), 1204 ~ 1209]。

22. 由中国剩余定理, 我们可以求出常数 a_1, \cdots, a_k , 它们模 m 的每个素因子后, 即是我们所想要的余数。如果 $m = p_1 p_2 \cdots p_t$, 则周期长度将是 $\text{lcm}(p_1^k - 1, \cdots, p_t^k - 1)$ 。事实上, 如习题 11 所示, 对于任意 m (不必是无平方的), 我们可以达到相当长的周期。

23. 减法可能比加法更快, 参见习题 3.2.1.1-5。由习题 30, 周期长度仍然是 $2^{e-1}(2^{55} - 1)$ 。R. Brent 已经指出, 这些计算可精确地对 $[0, 1)$ 中的浮点数进行; 参见习题 3.6-11。

24. 向后运行这个序列。换句话说, 如果 $Z_n = Y_{-n}$, 我们有 $Z_n = (Z_{n-k+l} - Z_{n-k}) \bmod 2 = (Z_{n-k+l} + Z_{n-k}) \bmod 2$ 。

25. 这个想法可以节省子程序调用的大部分开销。例如, 假设通过 JMP RANDM 来调用程序 A, 其中我们有

```
RANDOM STJ 1F
      LDA Y, 6
      :
      ENT6 55 } 程序 A
1H    JMP *    |
```

每个随机数的开销因而是 $14 + \frac{2}{55}$ 个时间单位。但假设我们代而使用, 比如说 “DEC6

1; J6Z RNGEN; LDA V, 6” 来生成随机数, 且子程序如下:

```
RNGEN  STJ    1F          ENT6   31
      ENT6   24          LDA    Y, 6
      LDA    Y+31, 6      ADD    Y+24, 6
      ADD    Y, 6         STA    Y, 6
      STA    Y+31, 6      DEC6   1
      DEC6   1            J6P    *-4
      J6P    *-4          ENT6   55
                        1H    JMP    *    |
```

现在开销仅是 $\left(12 + \frac{6}{55}\right)u$ 。[以 C 语言表达的一个类似的实现, 被用于 The Stanford GraphBase (New York: ACM Press, 1994), CB_FLIP 中。] 其实, 许多应用发现一次

就生成随机数的一个数组是可取的,而且,当我们使用 Lüscher 的方法增强随机性后,后一方法实质上是强制性的;参见 3.6 节中的 C 和 FORTRAN 程序。

27. 设 $J_n = \lfloor kX_n/m \rfloor$ 。引理:在 $(k^2 + 7k - 2)/2$ 个连续的值

$$0^{k+2} \quad 1 \quad 0^{k+1} \quad 2 \quad 0^k \quad \cdots \quad (k-1) \quad 0^3$$

出现于 $\langle J_n \rangle$ 序列中之后,对于 $0 \leq j < k$, 算法 B 将有 $V[i] < m/k$, 而且还有 $Y < m/k$ 。

证明:设 S_n 是使得在刚好生成 X_n 之前 $V[i] < m/k$ 的位置 i 的集合,并设 j_n 是使得 $V[j_n] \leftarrow X_n$ 的下标。如果 $j_n \in S_n$ 且 $J_n = 0$, 则 $S_{n+1} = S_n \cup \{j_n\}$ 且 $j_{n+1} > 0$; 如果 $j_n \in S_n$, 且 $J_n = 0$, 则 $S_{n+1} = S_n$, $j_{n+1} = 0$ 。因此在 $k+2$ 个相继的 0 之后,我们必定有 $0 \in S_n$ 和 $j_{n+1} = 0$ 。然后,在“1 0^{k+1}”之后必定有 $\{0, 1\} \subseteq S_n$ 而且 $j_{n+1} = 0$; 在“2 0^k”之后我们必定有 $\{0, 1, 2\} \subseteq S_n$ 和 $j_{n+1} = 0$; 等等。

推论:令 $l = (k^2 + 7k - 2)/2$, 如果 $\lambda \geq lk'$, 则或者算法 B 产生长度为 λ 的周期, 或者序列 $\langle X_n \rangle$ 很差地分布。证明: J 的任何给定长度 l 的模式不出现在长度 λ 的一个随机序列中的概率小于 $(1 - k^{-l})^{\lambda/l} < \exp(-k^{-l}\lambda/l) \leq e^{-1}$; 因此所述模式将出现。在它出现之后,算法 B 随继的特性在每次它达到周期的这部分时将是相同的。(当 $k > 4$ 时,我们要求 $\lambda > 10^{21}$, 所以这个结果纯粹是学术性的,但较小的界是可能的。)

29. 下列算法在最坏情况下执行大约 k^2 个操作,但它的平均运行时间要快得多,或许是 $O(\log k)$ 或甚至 $O(1)$:

X1. 置 $(a_0, a_1, \dots, a_k) \leftarrow (x_1, \dots, x_k, m-1)$ 。

X2. 设 i 是使 $a_i > 0$ 和 $i > 0$ 的极小值。当 $a_k > 0$ 时,对于 $j = j+1, \dots, k$, 执行子程序 Y。

X3. 如果 $a_0 > a_k$, 则 $f(x_1, \dots, x_k) = a_0$; 否则如果 $a_0 > 0$, 则 $f(x_1, \dots, x_k) = a_0 - 1$, 否则 $f(x_1, \dots, x_k) = a_k$ 。 ─

Y1. 置 $l \leftarrow 0$ 。(在步骤 Y1~Y3 中的子程序实质上是检验字典序关系 $(a_i, \dots, a_{i+k-1}) \geq (a_j, \dots, a_{j+k-1})$, 如果必要时减少 a_k 使这个不等式成立。我们假定 $a_{k+1} = a_1, a_{k+2} = a_2$, 等等。)

Y2. 如果 $a_{i+l} > a_{j+l}$, 则退出这个子程序。否则如果 $j+l = k$, 则置 $a_k \leftarrow a_{i-l}$ 。否则如果 $a_{i+l} = a_{j+l}$, 则转到步骤 Y3。否则如果 $j+l > k$, 则 a_k 减 1 并退出。否则置 $a_k \leftarrow 0$ 并退出。

Y3. l 加 1, 而且如果 $l < k$ 则返回步骤 Y2。 ─

当 $m = 2$ 时这个问题首先是由 H. Fredricksen 解决的 [*J. Combinatorial Theory* 9 (1970), 1~5; *A12* (1972), 153~154]; 在特殊情况下,这个算法较为简单而且它可以通过 k 个二进位的寄存器来进行。也可参见 H. Fredricksen 和 J. Maiorana, *Discrete Math.* 23 (1978), 207~210。

30. a) 由习题 11, 只须证明周期长度模 8 是 $4(2^k - 1)$; 这将为真当且仅当

$x^{2(2^k-1)} \not\equiv 1 \pmod{8 \text{ 和 } f(x)}$, 当且仅当 $x^{2^k-1} \not\equiv 1 \pmod{4 \text{ 和 } f(x)}$ 。写 $f(x) = f_e(x^2) + xf_o(x^2)$, 其中 $f_e(x^2) = \frac{1}{2}(f(x) + f(-x))$ 。于是 $f(x)^2 + f(-x)^2 = 2f(x^2) \pmod{8}$ 当且仅当 $f_e(x)^2 + xf_o(x)^2 \equiv f(x) \pmod{4}$; 而且后一条件成立当且仅当 $f_e(x)^2 \equiv -xf_o(x)^2 \pmod{4 \text{ 和 } f(x)}$, 因为 $f_e(x)^2 + xf_o(x)^2 = f(x) + O(x^{k-1})$ 。而且, 对模 2 和 $f(x)$ 进行工作, 我们有 $f_e(x)^2 \equiv f_e(x^2) \equiv xf_o(x^2) \equiv x^2 f_o(x)^2$, 因此 $f_e(x) \equiv x^{2^{k-1}} f_o(x)$ 。所以 $f_e(x)^2 \equiv x^2 f_o(x)^2 \pmod{4 \text{ 和 } f(x)}$, 得到提示。一个类似的论述证明 $x^{2^k} \equiv x \pmod{4 \text{ 和 } f(x)}$, 当且仅当 $f(x)^2 + f(-x)^2 \equiv 2(-1)^k \cdot f(-x^2) \pmod{8}$ 。

b) 仅当 l 是奇数和 $k=2l$ 时, 这个条件才能成立。但仅当 $k=2$ 时 $f(x)$ 才是模 2 本原的 [Math. Comp. 63 (1994), 389~401。]

31. 对于某些 Y_n 和 Z_n , 通过应用定理 3.2.1.2C, 我们有 $X_n \equiv (-1)^{Y_n} 3^{Z_n} \pmod{2^e}$; 因此 $Y_n = (Y_{n-24} + Y_{n-55}) \pmod{2}$ 和 $Z_n = (Z_{n-24} + Z_{n-55}) \pmod{2^{e-2}}$ 。由于 Z_k 是奇数当且仅当 $X_k \pmod{8} = 3$ 或 5 , 由上题周期长度是 $2^{e-3}(2^{55}-1)$ 。

32. 我们可以忽略“mod m ”而在以后再把它放回来。生成函数 $g(z) = \sum_n X_n z^n$ 是 $1/(1-z^{24}-z^{55})$ 的多项式倍数; 因此 $\sum_n X_{2n} z^{2n} = \frac{1}{2}(g(z) + g(-z))$ 是可由 $(1-z^{24}-z^{55})(1-z^{24}+z^{55}) = 1-2z^{24}+z^{48}-z^{110}$ 整除的多项式。因此头一个要求的递推是 $X_{2n} = (2X_{2(n-12)} - X_{2(n-24)} + X_{2(n-55)}) \pmod{m}$ 。类似地, $\sum_n X_{3n} z^{3n} = \frac{1}{3}(g(z) + g(\omega z) + g(\omega^2 z))$, 其中 $\omega = e^{2\pi i/3}$, 因此我们求得 $X_{3n} = (3X_{3(n-8)} - 3X_{3(n-16)} + X_{3(n-24)} + X_{3(n-55)}) \pmod{m}$ 。

33. (a) 由对 t 作数学归纳法, 有 $g_{n+t}(z) \equiv z^t g_n(z) \pmod{m \text{ 和 } 1+z^{31}-z^{55}}$ 。(b) 由于 $z^{500} \pmod{1+z^{31}-z^{55}} = 792z^2 + z^5 + 17z^6 + 715z^9 + 36z^{12} + z^{13} + 364z^{16} + 210z^{19} + 105z^{23} + 462z^{26} + 16z^{30} + 1287z^{33} + 9z^{36} + 18z^{37} + 1001z^{40} + 120z^{43} + z^{44} + 455z^{47} + 462z^{50} + 120z^{54}$ (参见算法 4.6.1D), 我们有 $X_{500} = (792X_2 + X_5 + \cdots + 120X_{54}) \pmod{m}$ 。

[比较类似的公式 $X_{165} = (X_0 + 3X_7 + X_{14} + 3X_{31} + 4X_{38} + X_{45}) \pmod{m}$ 和上题中关于 $\langle X_{3n} \rangle$ 更稀疏的递推是有趣的。Lüscher 的生成 165 个数和仅使用头 55 个数的方法显然优越于生成 165 个数和仅使用 X_3, X_6, \dots, X_{165} 的想法。]

34. 令 $q_0 = 0, q_1 = 1, q_{n+1} = cq_n + aq_{n-1}$ 。然后我们有 $\begin{pmatrix} 0 & 1 \\ a & c \end{pmatrix}^n = \begin{pmatrix} aq_{n-1} & q_n \\ aq_n & q_{n+1} \end{pmatrix}$,

$X_n = (q_{n+1}X_0 + aq_n)/(q_nX_0 + aq_{n-1})$, 而且对于 $n \geq 1, x^n \pmod{f(x)} \equiv q_n x + aq_{n-1}$ 。因此如果 $X_0 = 0$, 我们有 $X_n = 0$ 当且仅当 $x^n \pmod{f(x)}$ 是一个非零常数。

35. 条件(i)和(ii)意味着 $f(x)$ 是不可约的。因为如果 $f(x) = (x - r_1)(x - r_2)$ 且 $r_1 r_2 \neq 0$, 我们有: 如果 $r_1 \neq r_2$, 则 $x^{p-1} \equiv 1$, 而如果 $r_1 = r_2$, 则 $x^p \equiv r_1$ 。

令 ξ 是有 p^2 个元素的一个域的原根, 且设 $\xi^{2k} = c_k \xi^k + a_k$ 。我们所求的二次多项式即是多项式 $f_k(x) = x^2 - c_k x - a_k$, 其中 $1 \leq k < p^2 - 1$ 且 $k \perp p + 1$ (参见习题 4.6.2-16)。每个多项式对于两个 k 值出现; 因此解的个数是 $\frac{1}{2}(p^2 - 1) \prod_{q \mid p+1, q \text{ 素数}} (1 - 1/q)$ 。

36. 在这种情况下, X_n 总是为奇数, 所以 $X_n^{-1} \bmod 2^e$ 存在。在答案 34 中定义的序列 $\langle q_n \rangle$ 是 $0, 1, 2, 1, 0, 1, 2, 1, \dots$ 模 4。我们也有 $q_{2n} = q_n(q_{n+1} + aq_{n-1})$ 和 $q_{2n-1} = aq_{n-1}^2 + q_n^2$; 因此 $q_{2n+1} - aq_{2n-1} = (q_{n+1} - aq_{n-1})(q_{n+1} + aq_{n+1})$ 。由于当 n 是偶数时, $q_{n+1} + aq_{n+1} \equiv 2 \pmod{4}$, 我们导出, 对于所有的 $e \geq 0$, q_{2^e} 是 2^e 的奇数倍, 而 $q_{2^e+1} - aq_{2^e-1}$ 是 2^{e+1} 的奇数倍。因此

$$q_{2^e} + aq_{2^e-1} \equiv q_{2^e+1} - aq_{2^e} + 2^{e+1} \pmod{2^{e+2}}$$

而且 $X_{2^{e+2}} \equiv (q_{2^e+1} + aq_{2^e}) / (q_{2^e} + aq_{2^e-1}) \not\equiv 1 \pmod{2^e}$, 同时 $X_{2^{e+1}} \equiv 1$ 。反之, 我们需要 $a \bmod 4 = 1$ 和 $c \bmod 4 = 2$; 否则 $X_{2n} \equiv 1 \pmod{8}$ 。[Eichenauer, Lehn 和 Topuzoğlu, *Math. Comp.* **51** (1988), 757~759。]这个序列的低阶二进位有一个短的周期, 所以带有素模数的反演生成程序是可取的。

37. 我们可以假定 $b_1 = 0$ 。由习题 34, V 中的一个典型的向量是

$$(x, (s'_2 x + as_2) / (s_2 x + as''_2), \dots, (s'_d x + as_d) / (s_d x + as''_d))$$

其中 $s_i = q_{b_i}$, $s'_i = q_{b_i+1}$, $s''_i = q_{b_i-1}$ 。这个向量属于超平面 H 当且仅当

$$r_1 x + \frac{r_2 t_2}{x - u_2} + \dots + \frac{r_d t_d}{x - u_d} \equiv r_0 - r_2 s'_2 s_2^{-1} - \dots - r_d s'_d s_d^{-1} \pmod{p}$$

其中 $t_j = a - as'_j s''_j^{-2} = -(-a)^{b_j} s_j^{-2}$ 和 $u_j = as''_j s_j^{-1}$ 。但是这个关系等价于次数小于等于 d 的多项式同余; 所以对于 x 的 $d+1$ 个值它不能成立, 除非对于所有的 x , 包括不同的点 $x = u_2, \dots, x = u_d$, 它都成立。因此 $r_2 = \dots = r_d \equiv 0$, 而且 $r_1 \equiv 0$ 。[参见 J. Eichenauer-Herrmann, *Math. Comp.* **56** (1991), 297~301。]

注: 如果我们考虑有行 $\{(1, v_1, \dots, v_d) \mid (v_1, \dots, v_d) \in V\}$ 的 $(p+1-d) \times (d+1)$ 矩阵 M , 这道习题等价于断言, M 的任何 $d+1$ 行在模 p 之下线性无关, 有趣的是对于 $p \approx 1000$ 和 $0 \leq n \leq p$ 画出点 (X_n, X_{n+1}) , 看到的是圆圈而不是直线。

3.3.1 小节

1. 有 $k = 11$ 个范畴, 所以应使用行 $v = 10$ 。

2. $\frac{2}{49}, \frac{3}{49}, \frac{4}{49}, \frac{5}{49}, \frac{6}{49}, \frac{9}{49}, \frac{6}{49}, \frac{5}{49}, \frac{4}{49}, \frac{3}{49}, \frac{2}{49}$ 。

3. $V = 7 \frac{173}{240}$, 仅仅稍微高于由好骰子得到的! 为什么我们不检测加重, 有两个原因: (a) 新的概率(参考习题 2)并不真正远离等式(1)中旧的概率。两个骰子之和使概率趋于平滑。如果考虑的是 36 个可能对偶值的每一对, 并加以计值, 则我们大概很快地就会发现这一差别(假定两个骰子是可以区别的)。(b) 一个更重要的原因是, 对于发现重大差别说来, n 太小了, 如果同样的实验对充分大的 n 进行, 就将发现有差错的骰子(见习题 12)。

4. 对于 $2 \leq s \leq 12$ 和 $s \neq 7$, $p_s = \frac{1}{12}$, $p_7 = \frac{1}{6}$ 。 V 的值是 $16 \frac{1}{2}$, 它落入表 1 中 75% 和 95% 的项之间; 所以尽管实际上并没有太多的 7 出现, 但它是合理的。

5. $K_{20}^+ = 1.15$, $K_{20}^- = 0.215$ 。这些同随机特性相差不太大(大约处于 94% 和 86% 的水平), 但它们非常接近(这个习题的数据值由附录 A 表 1 得出)。

6. $X_j \leq x$ 的概率是 $F(x)$, 所以我们有在 1.2.10 小节中讨论的二项式分布: $F_n(x) = s/n$ 具有概率

$$\binom{n}{s} F(x)^s (1 - F(x))^{n-s}$$

均值是 $F(x)$; 标准差是 $\sqrt{F(x)(1 - F(x))/n}$ 。 [参考等式 1.2.10-(19)。这提示, 较好的方法是把统计公式定义为

$$K_n^+ = \sqrt{n} \max_{-\infty < x < \infty} (F_n(x) - F(x)) / \sqrt{F(x)(1 - F(x))}$$

见习题 22。对于 $x < y$, 我们可以计算 $F_n(y) - F_n(x)$ 的平均值和标准差, 并且得到 $F_n(x)$ 和 $F_n(y)$ 的协方差。利用这些事实, 可以证明, 对于很大的 n 值, 函数 $F_n(x)$ 的特性像“布朗运动”那样, 而且取自这个概率论分支的技术可以用来研究它。J. L. Doob 和 M. D. Donsker 的论文说明了这一情况, 见 *Annals Math. Stat.* **20** (1949), 393 ~ 403 和 **23** (1952), 277 ~ 281; 一般认为这是研究 KS 检验最明智的方法。]

7. 在等式(13)中置 $j = n$ 即知 K_n^+ 绝不为负, 而且它能达到 \sqrt{n} 那么高。类似地, 取 $j = 1$ 来对 K_n^- 作同样的观察。

8. 对 20 次观察计算了新的 KS 统计量。当计算统计量 KS 时, K_{10}^+ 的分布用作 $F(x)$ 。

9. 这个思想是错误的, 因为所有的观察都必须是独立的。对于相同数据, 统计量 K_n^+ 和 K_n^- 之间有一个关系, 所以每个检验应当独立地判定。(一个检验的高数值势必使另一个检验有低的值。)类似地, 图 2 和图 5 中的项(它对于每个生成程序示出 15 个检验)并不给出 15 个独立的观察, 因为 5 的极大值检验并非同 4 的极大值检验无关。每一水平行的三种检验是独立的(因为它们是对这个序列的不同部分进行的), 但在一系列中的 5 个检验是有一定关系的。其纯效果是: 对于一个检验适用的 95% 的概率水平不能想当然地适用于相同数据的整组检验。教训: 当检验一个随机数生成程序时, 我们可以预期它“通过”好几个检验的每一个, 例如, 频率检验,

极大检验, 运行检验, 等等; 但是由若干不同的检验得到的数据数组不应该当成是一个单位, 因为这些检验本身可能是不独立的。\$K_n^+\$ 和 \$K_n^-\$ 统计应该认为是两个单独的检验; 一个好的随机数来源将通过两个检验。

10. 每个 \$Y_s\$ 是双重的, 而且 \$np_s\$ 是双重的, 所以 (6) 的分子是四重的, 虽然分母仅仅是双重的。因此 \$V\$ 的新值恰是旧值的两倍。

11. 经验分布函数保持相同; \$K_n^+\$ 和 \$K_n^-\$ 乘以 \$\sqrt{2}\$。

12. 设 \$Z_s = (Y_s - np_s)/\sqrt{np_s}\$。\$V\$ 的值是 \$n\$ 乘

$$\sum_{s=1}^k (q_s - p_s + \sqrt{q_s/n} Z_s)^2 / p_s$$

而且当 \$n\$ 增加时, 后一个量保持非零有界 (因为 \$Z_s n^{-1/4}\$ 有界的概率为 1)。因此在 \$p_s\$ 的假定下, \$V\$ 的值将增加到一个极不可能的值。

对于 KS 检验, 设 \$F(x)\$ 是假设的分布, \$G(x)\$ 是实际的分布, 并设 \$h = \max |G(x) - F(x)|\$。取 \$n\$ 充分大, 使得 \$|F_n(x) - G(x)| > h/2\$ 以非常小的概率出现; 于是 \$|F_n(x) - F(x)|\$ 在假定的分布 \$F(x)\$ 之下将高达一个极不可能的值。

13. (由于我们所指的是最小的上界, 所以 “max” 的符号实际上应以 “sup” 来代替; 但是为避免很多读者由于不大熟悉 “sup” 符号而产生混乱, 正文中便使用了 “max”。) 为方便起见, 设 \$X_0 = -\infty, X_{n+1} = +\infty\$。当 \$X_j \leq x < X_{j+1}\$ 时, 我们有 \$F_n(x) = j/n\$; 因此 \$\max(F_n(x) - F(x)) = j/n - F(X_j)\$, 而且在这个区间中 \$\max(F(x) - F_n(x)) = F(X_{j+1}) - j/n\$。当 \$j\$ 从 0 变到 \$n\$ 时, \$x\$ 的所有实值都被考虑了; 这就证明了

$$K_n^+ = \sqrt{n} \max_{0 \leq j \leq n} \left(\frac{j}{n} - F(X_j) \right)$$

$$K_n^- = \sqrt{n} \max_{1 \leq j \leq n+1} \left(F(X_j) - \frac{j-1}{n} \right)$$

这些等式等价于 (13), 因为在极大符号下的额外项是非正的, 而且由习题 7 它必定是多余的。

14. 左边的对数简化为

$$- \sum_{s=1}^k Y_s \ln \left(1 + \frac{Z_s}{\sqrt{np_s}} \right) + \frac{1-k}{2} \ln(2\pi n) - \frac{1}{2} \sum_{s=1}^k \ln p_s -$$

$$\frac{1}{2} \sum_{s=1}^k \ln \left(1 + \frac{Z_s}{\sqrt{np_s}} \right) + O\left(\frac{1}{n}\right)$$

而这个量进一步简化 (通过展开 \$\ln(1 + Z_s/\sqrt{np_s})\$ 并知 \$\sum_{s=1}^k Z_s/\sqrt{np_s} = 0\$) 成

$$- \frac{1}{2} \sum_{s=1}^k Z_s^2 + \frac{1-k}{2} \ln(2\pi n) - \frac{1}{2} \ln(p_1 \cdots p_k) + O\left(\frac{1}{\sqrt{n}}\right)$$

15. (i) 从行列式消去因子 \$r^{n-1}\$, (ii) 用包含 “\$\cos \theta_1 - \sin \theta_1 0 \cdots 0\$” 的行的诸余子

式(每个余子式行列式都可用归纳法加以计算)展开得到的行列式,以及(iii) 利用 $\sin^2 \theta_1 + \cos^2 \theta_1 = 1$, 不难计算出对应的雅可比行列式。

$$16. \int_0^{z\sqrt{2x}+y} \exp\left(-\frac{u^2}{2x} + \dots\right) du = y e^{-z^2} + O\left(\frac{1}{\sqrt{x}}\right) + \int_0^{z\sqrt{2x}} \exp\left(-\frac{u^2}{2x} + \dots\right) du$$

后一积分是

$$\int_0^{z\sqrt{2x}} e^{-u^2/2} du + \frac{1}{3x^2} \int_0^{z\sqrt{2x}} e^{-u^2/2} u^3 du + O\left(\frac{1}{\sqrt{x}}\right)$$

当把它们全都放在一起时,最后的结果是

$$\frac{\gamma(x+1, x+z\sqrt{2x}+y)}{\Gamma(x+1)} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z\sqrt{2}} e^{-u^2/2} du + \frac{e^{-z^2}}{\sqrt{2\pi x}} \left(y - \frac{2}{3} - \frac{2}{3}z^2\right) + O\left(\frac{1}{x}\right)$$

如果我们置 $z\sqrt{2} = x_p$, 并令

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z\sqrt{2}} e^{-u^2/2} du = p, \quad x+1 = \frac{\nu}{2}, \quad \gamma\left(\frac{\nu}{2}, \frac{t}{2}\right) / \Gamma\left(\frac{\nu}{2}\right) = p$$

其中 $t/2 = x + z\sqrt{2x} + y$, 我们可以对 y 求解并得到 $y = \frac{2}{3}(1+z^2) + O(1/\sqrt{x})$, 它

同上边的分析是一致的。因此解是 $t = \nu + 2\sqrt{\nu}z + \frac{4}{3}z^2 - \frac{2}{3} + O(1/\sqrt{\nu})$ 。

17. a) 换变量, $x_j \leftarrow x_j + t$ 。

b) 对 n 用归纳法; 由定义,

$$P_{n0}(x+t) = \int_n^x P_{(n-1)0}(x_n-t) dx_n$$

c) 左边是

$$\int_n^{x+t} dx_n \cdots \int_{k+1}^{x_{k+2}} dx_{k+1} \text{ 乘 } \int_t^k dx_k \int_t^{x_k} dx_{k-1} \cdots \int_t^{x_2} dx_1$$

d) 从 b) 与 c) 我们有

$$P_{nk}(x) = \sum_{r=0}^k \frac{(r-t)^r}{r!} \frac{(x+t-r)^{n-r-1}}{(n-r)!} (x+t-n)$$

(24)中的分子是 $P_{n|t|}(n)$ 。

18. 如同在正文中关于(24)的推导中所注释的那样, 对于 $0 \leq x \leq 1$ 我们可以假设 $F(x) = x$ 。如果 $0 \leq X_1 \leq \cdots \leq X_n \leq 1$, 则令 $Z_j = 1 - X_{n+1-j}$ 。我们有 $0 \leq Z_1 \leq \cdots \leq Z_n \leq 1$; 而且对于 X_1, \dots, X_n 计算的 K_n^+ 等于对 Z_1, \dots, Z_n 计算的 K_n^- 。这个对称关系在相等体积的集合之间建立了一对一的对应, 对于它们, K_n^+ 和 K_n^- 都落入一个给定的范围中。

20. 例如, 项 $O(1/n)$ 是 $-\left(\frac{4}{9}s^4 - \frac{2}{3}s^2\right)/n + O(n^{-3/2})$ 。H. A. Lauwerier 已经得到一个完整的展开, *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete* 2 (1963), 61~68。

23. 设 m 是大于等于 n 的任意数。(a) 如果 $\lfloor mF(X_i) \rfloor := \lfloor mF(X_j) \rfloor$ 且 $i > j$, 则 $i/n - F(X_i) > j/n - F(X_j)$ 。(b) 对 $0 \leq k < m$ 以 $a_k = 1.0, b_k = 0.0$ 和 $c_k = 0$ 开始, 然后对于每个观察量 X_j 做下列操作: 置 $Y \leftarrow F(X_j), k \leftarrow \lfloor mY \rfloor, a_k \leftarrow \min(a_k, Y), b_k \leftarrow \max(b_k, Y), c_k \leftarrow c_k + 1$ 。(假设 $F(X_j) < 1$, 使得 $k < m$ 。)然后置 $j \leftarrow 0, r^+ \leftarrow r^- \leftarrow 0$, 而且对 $k = 0, 1, \dots, m-1$ (按此顺序), 每当 $c_k > 0$ 时, 做下列操作: 置 $r^- \leftarrow \max(r^-, a_k - j/n), j \leftarrow j + c_k, r^+ \leftarrow \max(r^+, j/n - b_k)$ 。最后置 $K_n^+ \leftarrow \sqrt{n}r^+, K_n^- \leftarrow \sqrt{n}r^-$ 。所需时间是 $O(m+n)$, 而且不必预先知道 n 的精确值。(如果估计值 $\left(k + \frac{1}{2}\right)/m$ 用做 a_k 和 b_k , 使得对于每个 k 实际上仅仅计算 c_k , 我们就得到好到 $\frac{1}{2}\sqrt{n}/m$ 之内的 K_n^+ 和 K_n^- 的估计, 甚至当 $m < n$ 时亦然。)[ACM Trans. Math. Software 3 (1977), 60~64。]

25. a) 由于 $c_{ij} = E(\sum_{k=1}^n a_{ik}X_k \sum_{l=1}^n a_{jl}X_l) = \sum_{k=1}^n a_{ik}a_{jk}$, 我们有 $C = AA^T$ 。

b) 考虑奇异值的分解 $A = UDV^T$, 其中 U 和 V 是大小 $m \times m$ 和 $n \times n$ 正交的, 而 D 是 $m \times n$ 的并有元素 $d_{ij} = [i=j]\sigma_j$; 奇异值 σ_j 总为正。[例如, Golub 和 Van Loan, *Matrix Computations* (1996), § 2.5.3。]如果 $\bar{C}\bar{C} = C$, 我们有 $SBS = S$, 其中 $S = DD^T$ 而 $B = U^T\bar{C}U$ 。因此 $s_{ij} = [i=j]\sigma_j^2$, 其中我们令 $\sigma_{n+1} = \dots = \sigma_m = 0$, 并令 $s_{ij} = \sum_{k=1}^n d_{ik}b_{kj}s_{ij} = \sigma_i^2\sigma_j^2b_{ij}$ 。因此如果 $i, j \leq n$, 则 $b_{ij} = [i=j]/\sigma_j^2$, 而且我们导出 D^TBD 是 $n \times n$ 单位矩阵。令 $Y = (Y_1 - \mu_1, \dots, Y_m - \mu_m)^T$ 和 $X = (X_1, \dots, X_n)^T$; 由此得出 $W = Y^T\bar{C}Y = X^T A^T \bar{C} A X = X^T V D^T B D V^T X = X^T X$ 。

3.3.2 小节

1. 对于 χ^2 检验的观察量必须是独立的。在第二个序列中相继的观察量显然是相关的, 因为一个的第二分量等于下一个的第一分量。

2. 对于 $0 \leq j < n$, 构造 t 元组 (Y_j, \dots, Y_{j+t-1}) , 并且计算这当中有多少个等于每个可能的值。在每个范畴中以 $k = d^t$ 和以 $1/d^t$ 的概率来应用 χ^2 检验, 观察的数目至少应该是 $5d^t$ 。

3. 通过枚举其余 $n-1$ 个发生可以出现的可能位置, 以及通过计算这样一个模式的概率, 恰好有 j 个值被考察的概率, 即 U_{j-1} 是位于范围 $\alpha \leq U_{j-1} < \beta$ 之内的第 n 个元素的概率, 容易看出是

$$\binom{j-1}{n-1} p^n (1-p)^{j-n}$$

生成函数是 $G(z) = (pz/(1 - (1-p)z))^n$, 由于当 $n=1$ 时给定的分布是同一事物的 n 重卷积, 因而它是有意义的。因此均值和方差同 n 成比例; 现在容易发现被考察的诸 U 的数目有特征 $(\min n, \text{ave } n/p, \max \infty, \text{dev } \sqrt{n(1-p)/p})$ 。当 $n=1$ 时, 这个概率分布的更详细的讨论可以在习题 3.4.1-17 的答案中找到; 也可见习题 2.3.4.2-26 的更一般的结果。

4. 长度大于等于 r 的一个间隔的概率, 是 r 个连续的 U 位于给定范围之外的概率, 即 $(1-p)^r$ 。长度精确地等于 r 的一个间隔的概率是长度大于等于 r 的概率减去长度大于等于 $r+1$ 的概率。

5. 当 N 趋于无穷时, n 也这样 (具有概率 1), 因此除了最后的间隔长度外, 这个检验恰和正文中介绍的间隔检验一样。而且正文的间隔检验肯定渐近于所述的 χ^2 分布, 因为每个间隔的长度显然同其它长度无关。[注: 这个结果的十分复杂的证明是由 E. Bofinger 和 V. J. Bofinger 在 *Annals. Math. Stat.* **32** (1961), 524~534 中给出的。他们的文章是值得注意的, 因为它讨论间隔检验的好多个有趣的变形; 例如, 他们还证明, 量

$$\sum_{0 \leq r \leq t} \frac{(Y_r - (Np)p_r)^2}{(Np)p_r}$$

不趋于 χ^2 分布, 尽管由于 Np 是 n 的期望值, 已有人提议把这个统计作为一个“更强”的检验。]

7. 5, 3, 5, 6, 5, 5, 4。

8. 见习题 10 且有 $w=d$ 。

9. (在步骤 C1 和 C4 中把 d 改成 w 。) 我们有

$$p_r = \frac{d(d-1)\cdots(d-w+1)}{d^r} \left\{ \begin{matrix} r-1 \\ w-1 \end{matrix} \right\}, \text{ 对于 } w \leq r < t$$

$$p_t = 1 - \frac{d!}{d^{t-1}} \left(\frac{1}{0!} \left\{ \begin{matrix} t-1 \\ d \end{matrix} \right\} + \cdots + \frac{1}{(d-w)!} \left\{ \begin{matrix} t-1 \\ w \end{matrix} \right\} \right)$$

10. 如同习题 3 中那样, 我们实际上仅仅需要考虑 $n=1$ 的情况。一个集券集合的长度为 r 的概率的生成函数, 由前边的习题和等式 1.2.9-(28), 是

$$G(z) = \frac{d!}{(d-w)!} \sum_{r \geq 0} \left\{ \begin{matrix} r-1 \\ w-1 \end{matrix} \right\} \left(\frac{z}{d} \right)^r = z^w \left(\frac{d-1}{d-z} \right) \cdots \left(\frac{d-w+1}{d-(w-1)z} \right)$$

利用定理 1.2.10A 和习题 3.4.1-17 容易计算均值和方差。我们发现

$$\text{mean}(G) = w + \left(\frac{d}{d-1} - 1 \right) + \cdots + \left(\frac{d}{d-w+1} - 1 \right) = d(H_d - H_{d-w}) = \mu$$

$$\text{var}(G) = d^2(H_d^{(2)} - H_{d-w}^{(2)}) - d(H_d - H_{d-w}) = \sigma^2$$

由于对于一个集券集合的检索被重复 n 次, 所考察的诸 U 的个数有特征 $(\min wn, \text{ave } \mu n, \max \infty, \text{dev } \sigma \sqrt{n})$ 。

11. |1|2|9 8 5 3|6| 7 0 |4|。

12. 算法 R (用于运行检验的数据)

- R1.** [初始化] 置 $j \leftarrow -1$, 并置 $\text{COUNT}[1] \leftarrow \text{COUNT}[2] \leftarrow \cdots \leftarrow \text{COUNT}[6] \leftarrow 0$ 。还置 $U_n \leftarrow U_{n-1}$, 以便于终止算法。
- R2.** [置 r 为 0] 置 $r \leftarrow 0$ 。
- R3.** [是否 $U_j < U_{j+1}$?] r 和 j 增 1。如果 $U_j < U_{j+1}$, 则重复这一步。
- R4.** [记录长度] 如果 $r \geq 6$, 则 $\text{COUNT}[6]$ 加 1, 否则 $\text{COUNT}[r]$ 加 1。
- R5.** [完成了?] 如果 $j < n-1$, 则返回 R2。 ■

13. 有 $(p+q+1) \binom{p+q}{p}$ 种方式得到 $U_{i-1} \geq U_i < \cdots < U_{i+p-1} \geq U_{i+p} < \cdots < U_{i+p+q-1}$; 减去其中 $U_{i-1} < U_i$ 的 $\binom{p+q+1}{p+1}$ 种, 并减去其中 $U_{i+p-1} < U_{i+p}$ 的 $\binom{p+q+1}{1}$ 种, 然后对于同时满足 $U_{i-1} < U_i$ 和 $U_{i+p-1} < U_{i+p}$ 的情况加上 1, 因为这种情况已经减去两次。(这是容斥原理的一个特殊情况, 在 1.3.3 节中有进一步的说明。)

14. 长度为 r 的一个运行以概率 $1/r! - 1/(r+1)!$ 出现, 这里假定诸 U 都不同。因此我们对于 $r < t$ 使用 $p_r = 1/r! - 1/(r+1)!$, 而对于长度大于等于 t 的运行, 使用 $p_t = 1/t!$ 。

15. 当 F 连续且 X 有分布 F 时, 这对 $F(X)$ 总为真; 见等式 3.3.1-(23) 之后的注释。

16. a) $Z_{jt} = \max(Z_{j(t-1)}, Z_{(j+1)(t-1)})$ 。因此如果把 $Z_{j(t-1)}$ 存入存储器中, 则很容易把这个数组变换成 Z_{jt} 的集合, 而且不需要辅助的存储器。b) 由于他的“改进”, 每个 V 确实应有所指出的分布, 但诸观察已不再是独立的。事实上, 当 U_j 是相当大的值时, 所有 $Z_{jt}, Z_{(j-1)t}, \cdots, Z_{(j-t+1)t}$ 都将等于 U_j ; 所以我们几乎得到重复同样数据 t 次的效果(而且这将乘 V 以 t , 如习题 3.3.1-10 中那样)。

17. b) 由 Binet 的恒等式, 差是 $\sum_{0 \leq k < j < n} (U'_k V'_j - U'_j V'_k)^2$, 而这肯定是非负的。c) 因此如果 $D^2 = N^2$, 则对于每一对 j, k , 我们必有 $U'_k V'_j - U'_j V'_k = 0$ 。这意味着矩阵

$$\begin{bmatrix} U'_0 & U'_1 & \cdots & U'_{n-1} \\ V'_0 & V'_1 & \cdots & V'_{n-1} \end{bmatrix}$$

有小于 2 的秩, 所以它的行是线性相关的。(假定 U'_0 和 V'_0 不全为零, 则对于 $1 \leq j < n$, $U'_0 V'_j - U'_j V'_0 = 0$ 意味着存在常数 α, β 使得对所有 j , 有 $\alpha U'_j + \beta V'_j = 0$, 利用这一事实可以给出更为初等的证明; U'_0 和 V'_0 全为 0 的情况, 可以通过适当的重新编号予以避免。)

18. (a) 分子是 $-(U_0 - U_1)^2$, 分母是 $(U_0 - U_1)^2$ 。(b) 在这种情况下, 分子是 $-(U_0^2 + U_1^2 + U_2^2 - U_0 U_1 - U_1 U_2 - U_2 U_0)$; 分母是 $2(U_0^2 + \cdots + U_2 U_0)$ 。(c) 由

习题 1.2.3-30 或 1.2.3-31, 分母总等于 $\sum_{0 \leq j < k < n} (U_j - U_k)^2$ 。

19. 事实上, 每当 U_0, U_1, \dots, U_{n-1} 的联合分布是对称的(在排列之下不变)时, 所述结果成立。令 $S_1 = U_0 + \dots + U_{n-1}$, $S_2 = U_0^2 + \dots + U_{n-1}^2$, $X = U_0 U_1 + \dots + U_{n-2} U_{n-1} = U_{n-1} U_0$, 而且 $D = nS_2 - S_1^2$ 。还令 $Ef(U_0, \dots, U_{n-1})$ 表示在 $D \neq 0$ 的条件支配下 $f(U_0, \dots, U_{n-1})$ 的期望值。由于 D 是一个对称函数, 对于 $\{0, \dots, n-1\}$ 的所有排列 p , 我们有 $Ef(U_0, \dots, U_{n-1}) = Ef(U_{p(0)}, \dots, U_{p(n-1)})$ 。因此 $ES_2/D = nEU_0^2/D$, $ES_1^2/D = n(n-1)E(U_0 U_1/D) + nEU_0^2/D$, 以及 $EX/D = nE(U_0 U_1/D)$ 。由此得出 $1 = E(nS_2 - S_1^2)/D = -(n-1)E(nX - S_1^2)/D$ 。(严格地说, ES_2/D 和 ES_1^2/D 可能是无穷的, 所以必须小心地仅对已知存在的期望值的线性组合进行工作。)

20. 令 $E_{1111}, E_{211}, E_{22}, E_{31}$ 和 E_4 分别表示 $E(U_0 U_1 U_2 U_3/D^2)$, $E(U_0^2 U_1 U_2/D^2)$, $E(U_0^2 U_1^2/D^2)$, $E(U_0^3 U_1/D^2)$, $E(U_0^4/D^2)$ 的值。于是我们有 $ES_2^2/D^2 = n(n-1)E_{22} + nE_4$, $E(S_2 S_1^2/D^2) = n(n-1)(n-2)E_{211} + n(n-1)E_{22} + 2n(n-1)E_{31} + nE_4$, $ES_1^4/D^2 = n(n-1)(n-2)(n-3)E_{1111} + 6n(n-1)(n-2)E_{211} + 3n(n-1)E_{22} + 4n(n-1)E_{31} + nE_4$, $EX^2/D^2 = n(n-3)E_{1111} + 2nE_{211} + nE_{22}$, $E(XS_1^2/D^2) = n(n-2)(n-3)E_{1111} + 5n(n-2)E_{211} + 2nE_{22} + 2nE_{31}$, $E((U_0 - U_1)^4/D^2) = 6E_{22} - 8E_{31} + 2E_4$, 因此头一个结果为真。

令 $\delta = \alpha((\ln n)/n)^{1/3}$, $M = \alpha^3/2 + 1/3$, 而且 $m = \lceil 1/\delta \rceil$ 。如果我们把分布的范围划分成 m 个等概率的部分, 我们可以使用尾部不等式 1.2.10-(24) 和 (25), 证明每一部分以大于等于 $1 - O(n^{-M})$ 的概率, 包含在 $n\delta(1 - \delta)$ 和 $n\delta(1 + \delta)$ 点之间。

因此, 如果分布是一致的, 则至少以这个概率 $D = \frac{1}{12}n^2(1 + O(\delta))$ 。如果 D 不在此

范围内, 我们有 $0 \leq (U_0 - U_1)^4/D^2 \leq 1$ 。由于 $E((U_0 - U_1)^4) = \int_0^1 \int_0^1 (x - y)^4 dx dy$

$= \frac{1}{15}$, 我们可以得出结论, $E((U_0 - U_1)^4/D^2) = \frac{48}{5}n^{-4}(1 + O(\delta)) + O(n^{-M})$ 。

注: 令 N 是 (23) 的分子。当所有变量都有正态分布时, W. J. Dixon 证明 $e^{(wN + zD)/n}$ 的期望值是

$$(1 - 2z - 2zw)^{1/2} (1 - 2z + \sqrt{(1 - 2z)^2 - 4w^2})^{-n/2} + O(w^n)$$

相对于 w 求微分和相对于 z 求积分, 他发现当 $n > 2k$ 时, 矩量 $E(N/D)^{2k-1} = \left(-\frac{1}{2}\right)^k / \left(n - \frac{1}{2}\right)^k$, $E(N/D)^{2k} = \left(+\frac{1}{2}\right)^k / \left(n + \frac{1}{2}\right)^k$ 。特别是, 在这种情况下的方差恰是 $1/(n+1) - 1/(n-1)^2$ 。[Annals of Math. Stat. 15 (1944), 119~144。]

21. 在执行到步骤 P2 时 $c_{r-1} = s - 1$ 的值依次为 2, 3, 7, 6, 4, 2, 2, 1, 0; 因此 $f = 886862$ 。

22. $1024 = 6! + 2 \cdot 5! + 2 \cdot 4! + 2 \cdot 3! + 2 \cdot 2! + 0 \cdot 1!$, 所以我们要求执行到步骤 P2 时

$s-1$ 的连续的值依次为 $0, 0, 0, 1, 2, 2, 2, 2, 0$; 若向后运行, 则排列是 $(9, 6, 5, 2, 3, 4, 0, 1, 7, 8)$ 。

23. 令 $P'(x_1, \dots, x_t) = \frac{1}{\lambda} \sum_{n=0}^{\lambda-1} [(Y'_n, \dots, Y'_{n+t-1}) = (x_1, \dots, x_t)]$ 。于是, 我们有

$$Q(x_1, \dots, x_t) = \sum_{(y_1, \dots, y_t)} P'(y_1, \dots, y_t) P((x_1 - y_1) \bmod d, \dots, (x_t - y_t) \bmod d)$$

更紧凑地, $Q(x) = \sum_y P'(y) P(x - y)$ 。因此, 使用一般的不等式 $(EX)^2 \leq EX^2$, 我们有 $\sum_x (Q(x) - d^{-t})^2 = \sum_x (\sum_y P'(y) (P(x - y) - d^{-t}))^2 \leq \sum_x \sum_y P'(y) (P(x - y) - d^{-t})^2 = \sum_y P'(y) \sum_x (P(x) - d^{-t})^2 = \sum_x (P(x) - d^{-t})^2$ 。[参见 G. Marsaglia, *Comp. Sci. and Statistics: Symp. on the Interface* 16 (1984), 5~6。这个结果仅当 $d^t \leq 2\lambda$ 时才是有意义的, 因为每个 $P(x)$ 都是 $1/\lambda$ 的倍数。]

24. 对于串 α 的头 k 个和最后 k 个元素写 $k:\alpha$ 和 $\alpha:k$ 。令 $K(\alpha, \beta) = [\alpha = \beta] / P(\alpha)$; 并令 \bar{C} 是具有元素 $\bar{c}_{\alpha\beta} = K(\alpha, \beta) - K(t-1:\alpha, t-1:\beta)$ 的 $d^t \times d^t$ 的矩阵。设 C 是对于 $|\alpha| = t$, 除以 n 的随机变量 $N(\alpha)$ 的协方差矩阵。对于 d^{t-1} 个串 α 的每一个这些变量受到 $\sum_{a=0}^{d-1} N(a\alpha) = \sum_{a=0}^{d-1} N(\alpha a)$ 的约束, 但是所有其它的线性约束都可从这些约束导出(请见定理 2.3.4.2G)。因此 C 有 $d^t - d^{t-1}$ 的秩, 而且由习题 3.3.1-25, 只须证明 $C\bar{C}C = C$ 即可。

不难验证 $c_{\alpha\beta} = P(\alpha\beta) \sum_{k+l \leq t} T_k(\alpha, \beta)$, 其中 $T_k(\alpha, \beta)$ 是当我们在 α 上附加 β 并且向右滑动 k 个位置时对应于可能出现的重叠的一个项:

$$T_k(\alpha, \beta) = \begin{cases} K(t+k:\alpha, \beta:t+k) - 1, & \text{如果 } k \leq 0 \\ K(\alpha:t-k, t-k:\beta) - 1, & \text{如果 } k \geq 0 \end{cases}$$

例如, 如果 $d=2, t=5, \alpha=01101$ 和 $\beta=10101$, 我们有 $c_{\alpha\beta} = P(0)^4 P(1)^6 (P(01))^{-1} - P(101)^{-1} + P(1)^{-1} = 9$ 。因此 $\bar{C}C$ 的元素 $\alpha\beta$ 是 $P(\alpha\beta)$ 乘以

$$\sum_{|\gamma|=t-1} \sum_{a,b=0}^{d-1} P(\gamma ab) \sum_{k+l \leq t} \sum_{|l| \leq t} T_k(\alpha, \gamma a) (K(a, b) - 1) T_l(\gamma b, \beta)$$

给定 k 和 l , 把乘积 $T_k(\alpha, \gamma a) (K(a, b) - 1) T_l(\gamma b, \beta)$ 扩展到 8 项, 当乘以 $P(\gamma ab)$ 和对于所有 γab 求和时它们的每一个通常求和成 ± 1 。例如, 当 $\alpha = a_1 \dots a_t, \beta = b_1 \dots b_t, \gamma = c_1 \dots c_{t-1}$ 且 $t \geq 5$ 时, $P(\gamma ab) K(2:\alpha, \gamma a:2) K(a, b) K(3:\gamma b, \beta:3)$ 的和是 $P(c_4 \dots c_{t-2})$ 之和, 它为 1。如果 $t=4$, 同样的和将是 $K(a_1, b_4)$, 但它将同 $P(\gamma ab) K(2:\alpha, \gamma a:2) (-1) K(3:\gamma b, \beta:3)$ 之和相抵消。除非 $k \leq 0 \leq l$, 否则纯效果是 0; 否则结果是 $K(i:(j:\alpha), (\beta:l):i) - K(i-1:(j:\alpha), (\beta:l+1):i-1)$, 其中 $i = \min(t+k, t-1)$ 和 $j = \max(0, k+l)$ 。对 k 和 l 求和缩短成 $c_{\alpha\beta}$ 。

25. 事实上, 经验检验证明, 当把 (22) 推广成任意 t 时, C_1^{-1} 和 $C_1^{-1} C_2 C_1^{-1}$ 的对应元素之比在 $t \geq 5$ 时非常接近于 $-t$ 。例如, 当 $t=6$ 时, 它们全都居于 -6.039 和 -6.111 之间; 当 $t=20$ 时, 它们全都居于 -20.039 和 -20.045 之间。这种现象需要解释。

26. a) 向量 (S_1, \dots, S_n) 在超平面 $S_1 + \dots + S_n = 1$ 中, 在由不等式 $S_1 \geq 0, \dots, S_n$

≥ 0 所定义的 $n-1$ 维多边形中是一致地分布的点。一个容易的归纳法证明

$$\int_{s_1}^{\infty} dt_1 \int_{s_2}^{\infty} dt_2 \cdots \int_{s_{n-1}}^{\infty} dt_{n-1} [1 - t_1 - \cdots - t_{n-1} \geq s_n] = \frac{(1 - s_1 - s_2 - \cdots - s_n)_+^{n-1}}{(n-1)!}$$

为了获得其概率,以在特殊情况 $s_1 = \cdots = s_n = 0$ 下它的值来除这个积分。[Brunode Finetti, *Giornale Istituto Italiano degli Attuari* 27 (1964), 151~173。]

b) $S_{(1)} \geq s$ 的概率是 $S_1 \geq s_1, \cdots, S_n \geq s$ 的概率。

c) $S_{(k)} \geq s$ 的概率是 S_j 中至多 $k-1$ 个小于 s 的概率;因此 $1 - F_k(s) = G_1(s) + \cdots + G_{k-1}(s)$, 其中 $G_j(s)$ 是精确地有 j 个间距小于 s 的概率。由对称性, $G_j(s)$ 是 $\binom{n}{j}$ 乘以 $S_1 < s_1, \cdots, S_j < s, S_{j+1} \geq s, \cdots, S_n \geq s$ 的概率;而且后者是 $\Pr(S_1 < s, \cdots, S_{j-1} < s, S_j \geq 0, S_{j+1} \geq s, \cdots, S_n \geq s) - \Pr(S_1 < s, \cdots, S_{j-1} < s, S_j \geq s, \cdots, S_n \geq s)$ 。重复应用 a) 表明 $G_j(s) = \binom{n}{j} \sum_l \binom{j}{l} (-1)^{j-l} (1 - (n-l)s)_+^{n-1}$; 因此

$$1 - F_k(s) = \sum_l \binom{n}{l} \binom{n-l-1}{k-l-1} (-1)^{k-l-1} (1 - (n-l)s)_+^{n-1}$$

特别是,最大的间距 $S_{(n)}$ 有分布

$$F_n(s) = 1 - \sum_l \binom{n}{l} \binom{n-l-1}{n-l-1} (-1)^{n-l-1} (1 - (n-l)s)_+^{n-1} = \sum_l \binom{n}{l} (-1)^l (1 - ls)_+^{n-1}$$

[顺便指出,类似的量 $x^{n-1}(n-1)!^{-1}F_n(x^{-1})$ 结果是一致离差的和 $U_1 + \cdots + U_n$ 的密度函数。]

d) 由公式 $Es^r = r \int_0^1 (1 - F(s)) s^{r-1} ds$ 和 $\int_0^1 s^r (1 - ks)_+^{n-1} ds = k^{-r-1} n^{-1} \binom{n+r}{r}^{-1}$, 我们求得 $ES_{(k)} = n^{-1}(H_n - H_{n-k})$, 而且,通过一点代数知识, $ES_{(k)}^2 = n^{-1}(n+1)^{-1}(H_n^{(2)} - H_{n-k}^{(2)} + (H_n - H_{n-k})^2)$ 。因此 $S_{(k)}$ 的方差等于 $n^{-1} \cdot (n+1)^{-1}(H_n^{(2)} - H_{n-k}^{(2)} - (H_n - H_{n-k})^2/n)$ 。

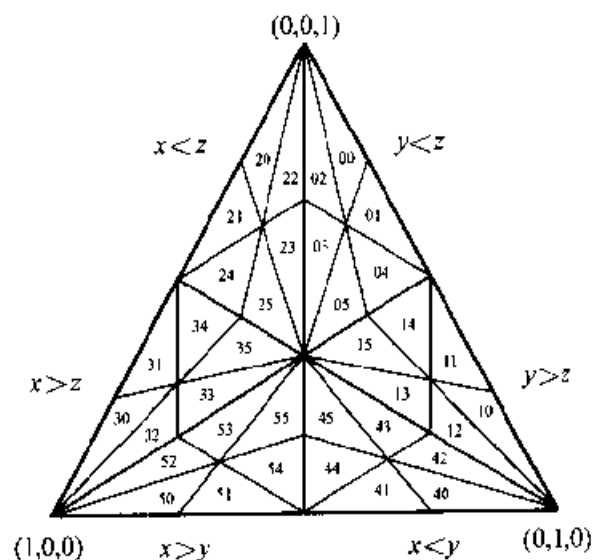
[分布 $F_k^{(s)}$ 首先是由 W. A. Whitworth 发现的。他把这个问题当做 *Choice and Chance* (Cambridge, 1867) 中的第 667 个问题并在 *DDC Exercises in Choice and Chance* (Cambridge, 1897) 中给出了解。Whitworth 也发现了计算函数 $G_k(s) = F_k(s) - F_{k+1}(s)$ 中任何多项式的期望值的一个优雅的方法,发表在标题为 *The Expectation of Parts* (Cambridge, 1898) 的小册子中,并被归入到 *Choice and Chance* 的第 5 版(1901)中。对于均值和方差及各种更一般的间隔统计的简化表达式是由 Barton 和 David 发现的, *J. Royal Stat. Soc* B18 (1956), 79~94。关于统计学家把对间隔的分析作为数据潜在偏倚的线索的传统方法的综述,参见 R. Pyke, *J. Royal*

Stat. Soc. B27 (1965), 395~449.]

27. 考虑由不等式 $S_1 \geq 0, \dots, S_n \geq 0$ 所定义的在超平面 $S_1 + \dots + S_n = 1$ 中的多边形。这个多边形由诸 S 的次序定义的(假定诸 S 是不相同的) $n!$ 同余子多边形组成, 而且排序的操作是大的多边形到其中 $S_1 \leq \dots \leq S_n$ 的子多边形 $n!$ 对 1 的折叠。把 $(S_{(1)}, \dots, S_{(n)})$ 变成 (S'_1, \dots, S'_n) 的变换是一对一的映射, 它通过 $n!$ 的因子来扩展微分的体积。它把子多边形的顶点 $\left(\frac{1}{n}, \dots, \frac{1}{n}\right), \left(0, \frac{1}{n-1}, \dots, \frac{1}{n-1}\right), \dots, (0, \dots, 0, 1)$ 变成分别的顶点 $(1, 0, \dots, 0), (0, 1, \dots, 0), \dots, (0, \dots, 0, 1)$, 它线性地拉长和扭曲过程中的整个形状。(在子多边形中顶点 $\left(0, \dots, 0, \frac{1}{j}, \dots, \frac{1}{j}\right)$ 和 $\left(0, \dots, 0, \frac{1}{k}, \dots, \frac{1}{k}\right)$ 之间的欧几里得距离是 $|j^{-1} - k^{-1}|^{1/2}$; 这个变换产生一个正规的单纯形, 其中所有的 n 个顶点都相距 $\sqrt{2}$ 。)

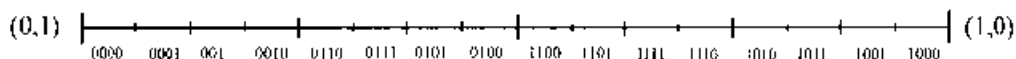
如果我们从图形上考察当 $n=3$ 时的细节, 最容易理解迭代的间距的特性。在这种情况下, 多边形只不过是等边三角形, 它的点由重心坐标 $(x, y, z), x + y + z = 1$ 表示。伴随的图式说明了这个三角形的一个递归分解的头两级。 6^2 个子三角形的每一个都已经以两位数代码 pq 来加以标号, 其中 p 表示当 $(x, y, z) = (S_1, S_2, S_3)$ 被排序成为 $(S_{(1)}, S_{(2)}, S_{(3)})$ 时可应用的排列, 而 q 表示按照代码

$$\begin{aligned} 0: x < y < z, & 1: x < z < y, & 2: y < x < z, \\ 3: y < z < x, & 4: z < x < y, & 5: z < y < x \end{aligned}$$



当 S'_1, S'_2, S'_3 被排序时在下一个阶段的排列。例如, 子三角形 34 的点有 $S_2 < S_3 < S_1$ 和 $S'_3 < S'_1 < S'_2$ 。我们可以继续这一过程直到无穷多级; 有无理数的重心坐标的三角形的所有点因此获得作为一个无穷的 6 进制展开的惟一表示。一个四边形可以类似地被划分成为 $24, 24^2, 24^3, \dots$ 个子四边形, 而且一般地说, 对于任何 $n-1$ 维的单纯形这个过程构造一个 $n!$ 进制的展开。

当 $n=2$ 时,这一过程是特别简单的:如果 $x \in [0, \frac{1}{2}, 1]$, 变换就把间距 $(x, 1-x) = (x, y)$ 转换成等于 $(2x \bmod 1, 2y \bmod 1)$ 或者 $(2y \bmod 1, 2x \bmod 1)$, 取决于 $x < y$ 还是 $x > y$ 而定。因此重复的检验实质上是把二进表示左移一位,很可能把结果取补。在对于 e 个二进位的数做了至多 $e+1$ 次迭代之后,这个过程必定收敛到固定点 $(0, 1)$ 。在 $n=2$ 的情况下的排列代码简单地对应于折叠和延长一条直线;划分的头四级有下列四个二进位的代码:



这个序列精确地是在 7.2.1 小节中研究的 Gray 二进制码。一般地说,对于一个 n 单纯形的 $n!$ 进制排列码有这样一个性质,即除了在一个数位位置外,相邻的区域有相等的代码。间距变换的每个迭代把每个点的表示的最左数字移出。注意相等的生日间隔是靠近头一级分解的边界的那些点。

从 (S_1, \dots, S_n) 到 (S'_1, \dots, S'_n) 的这个基本变换隐含在 *Choice and Chance* 第 5 版中 Whitworth 对命题 LVI 给出的证明中(参见答案 26 中的参考文献)。它首先是由 J. Durbin 明确地研究的 [*Biometrika* 48 (1961), 41~55], 他是受到 P. V. Sukhatme 的类似构造的启示而进行这一研究的 [*Annals of Eugenics* 8 (1937), 52~56]。迭代间隔的排列代码是由 H. E. Daniels 引进的 [*Biometrika* 49 (1962), 139~149]。

28. (a) 由习题 5.1.1-16, 把 m 分成 n 个不同的正部分的分划数是 $p_n\left(m - \binom{n+1}{2}\right)$ 。这些分划可以以 $n!$ 种方法来排列产生具有 $0 = y_1 < y_2 < \dots < y_n < m$ 的 n 元组 (y_1, \dots, y_n) ; 而且这些 n 元组的每个都导致有 $y_1 = 0$ 和 $0 < y_2, \dots, y_n < m$ 的 $(n-1)!$ 个 n 元组。现在对于每个 y_j 加上一个常数模 m ; 这保持间隔。因此 $b_{n00}(m) = mn!(n-1)!p_n\left(m - \binom{n+1}{2}\right)$ 。

(b) 零间隔对应于同一个瓮中的球, 而且它们对相同间隔的计数贡献 $s-1$ 。因此 $b_{nrs}(m) = \left\{ \begin{matrix} n \\ n-s \end{matrix} \right\} b_{(n-s)(r+1-s)0}(m)$ 。

(c) 由于 $\left\{ \begin{matrix} n \\ n-1 \end{matrix} \right\} = \binom{n}{2}$, 因此概率为

$$n!(n-1)!m^{-n} \left(p_n\left(m - \binom{n+1}{2}\right) - \frac{1}{2} p_{n-1}\left(m - \binom{n}{2}\right) \right)$$

29. 由上一个答案和习题 5.1.1-15, 我们有 $b_{n0}(z) = n!(n-1)!z^{\binom{n+1}{2}}/(1-z) \cdots (1-z^n)$ 。当 $r=1$ 时, 在我们前边的推导中的 $n!$ 变成 $n!/2$, 而且满足 $s_1 + \dots + s_n = m$ 的对于 $0 < s_1 < \dots < s_k \leq s_{k+1} < \dots < s_n$ 的解的数目是满足 $(s_1-1) + \dots + (s_k-k) + (s_{k+1}-k) + \dots + (s_n-n+1) = m - \binom{n}{2} - k$ 的对于 $0 \leq s_1-1 \leq \dots \leq$

$s_k - k \leq s_{k+1} - k \leq \dots \leq s_n - n + 1$ 的解的个数。因此 $b_{n1}(z) = \frac{1}{2} n! (n-1)! \sum_{k=1}^n$

$(z^k - 2^n) z^{\binom{n}{2}} / (1-z) \cdots (1-z^n)$ 。类似的一个论证表明

$$\frac{b_{n2}(z)}{n!(n-1)!} = \left(\frac{1}{2!2!} \sum_{1 \leq k < k' \leq n} (z^k - z^n)(z^{k'} - z^{n-1}) + \frac{1}{3!} \sum_{1 \leq k < k' < k'' \leq n} (z^k - z^n)(z^{k'} - z^{n-1})(z^{k''} - z^{n-2}) \right) \times \frac{z^{\binom{n-1}{2}}}{(1-z) \cdots (1-z^n)}$$

由公式

$$\frac{\sum_r b_{nr}(z) w^r}{n!(n-1)! z^n} = \sum_{0 \leq b_1, \dots, b_{n-1} \leq 1} \frac{(z - b_1 z^n) \cdots (z^{n-1} - b_{n-1} z^n)}{c_1 \cdots c_{n-1} (1-z) \cdots (1-z^n)} \left(\frac{w}{z^{n-1}} \right)^{b_1} \cdots \left(\frac{w}{z^1} \right)^{b_{n-1}}$$

我们可以得到对于一般的 r 的 $b_{nr}(z)$, 其中 $c_k = 1 + b_k + b_k b_{k-1} + \dots + b_k \cdots b_2 b_1 = 1 + b_k c_{k-1}$ (特殊情况 $w = 1$ 是很有趣的, 因为在这种情况下左边加成 $(1-z)^{-n}/n!$)。

30. 对于鞍点方法, 这是一个好问题 [N. G. de Bruijn, *Asymptotic Methods in Analysis* (North-Holland, 1961), 第 5 章]。我们有 $p_n(m) = \frac{1}{2\pi i} \oint e^{f(z)} \frac{dz}{z}$, 其中 $f(z) = -m \ln z - \sum_{k=1}^n \ln(1 - z^k)$ 。令 $\rho = n/m$ 和 $\delta = \sqrt{n}/m$; 在通路 $z = e^{-\rho + i t \delta}$ 上积分给出 $p_n(m) = \frac{\delta}{2\pi} \int_{-\pi/\delta}^{\pi/\delta} \exp(f(e^{-\rho + i t \delta})) dt$ 。使用等式

$$g(se^t) = \sum_{j=0}^n \frac{t^j}{j!} \theta^j g(s) + \int_0^t \frac{u^n}{n!} \theta^{n+1} g(se^{t-u}) du$$

是方便的, 其中 $g = g(z)$ 是任何解析函数而且 θ 是算子 $z \frac{d}{dz}$ 。当函数 $\theta^j g$ 在 e^z 处被计算时, 其结果和当 $g(e^z)$ 相对于 z 被微分 j 次是相同的。这个原理导致了下列公式:

$$\theta^j f(e^{-\rho}) = -m[j=1] + \frac{j!n}{\rho^j} + (-1)^j \sum_{k=1}^n \sum_{l \geq j} \frac{l! \beta_l}{l! \cdot l!} k^l \rho^{l-j}$$

这是由于另一个方便的恒等式所致, 即

$$\ln\left(\frac{1 - e^{-z}}{z}\right) = \sum_{n \geq 1} \frac{B_n z^n}{n \cdot n!}$$

因此我们得到积分项的一个渐近展开

$$\exp f(e^{-\rho + i t \delta}) = \exp\left(\sum_{j \geq 0} \frac{t^j \delta^j}{j!} \theta^j f(e^{-\rho})\right) = e^{t^2/2 + f(e^{-\rho})} \exp(ic_1 t - c_2 t^2 - ic_3 t^3 + \dots)$$

其中 $c_1 = \left(\frac{n(n+1)}{2} B_1 + \frac{n(n+1/2)(n+1)}{6} B_2 \rho\right) \delta + O(n^{-3})$, 等等; 而且结果是对 $j \geq 8, c_j = O(n^{-3})$ 。分解出常数项

$$\frac{\delta}{2\pi} e^{i(\dots)} = \frac{\delta}{2\pi n! \rho^r e^{-\frac{a}{m}}} \exp\left(-\sum_{k=1}^n \sum_{l=1}^{\infty} \frac{B_l}{l \cdot l!} k^l \rho^l\right) =$$

$$\frac{\sqrt{n} m^{n-1} e^{n-a/4}}{2\pi n! n^n} \left(1 + \frac{18\alpha - a^2}{72n} + \frac{108\alpha^2 - 36\alpha^3 + a^4}{10368n^2} + O(n^{-3})\right)$$

给我们留下当 $|t| \geq n$ 时其积分项指数地很小的一个积分。我们可以忽略较大的 t 值, 因为部分分数展开表明积分项是 $O((m/n)^{n/2})$; 单位的其它根作为分母的一个极点没有一个出现多于 $n/2$ 次。因此我们被允许“卖掉尾部”[CMath, § 9.4] 并对所有 t 进行积分。公式 $\int_{-\infty}^{\infty} e^{-t^2/2} t^j dt = (j-1)(j-3)\cdots(1) \sqrt{2\pi}$ [j 偶] 和 $n! =$

$(n/e)^n \sqrt{2\pi n} \exp\left(\frac{1}{12}n^{-1} + O(n^{-3})\right)$ 足以来完成这个计算。

以 $q_n(m) = p_n\left(m - \binom{n+1}{2}\right)$ 来代替 $p_n(m)$ 计算以同样的方式进行, 但 c_1 增加 $\frac{1}{2}\alpha(n^{1/2} - n^{-1/2})$ 和增加另一个因子 $\exp\left(-\rho\binom{n+1}{2}\right)$ 。我们得到

$$q_n(m) = \frac{m^{n-1} e^{-a/4}}{n!(n-1)!} \left(1 - \frac{13\alpha^2}{288n} + \frac{169\alpha^4 - 2016\alpha^3 - 1728\alpha^2 + 41472\alpha}{165888n^2} + O(n^{-3})\right)$$

这和对于 $p_n(m)$ 的公式相匹配, 但 α 已被改变成 $-\alpha$ 。(事实上, 如果我们定义

$$p_n(m) = r_n\left(2m + \binom{n+1}{2}\right) \text{ 和 } q_n(m) = r_n\left(2m - \binom{n+1}{2}\right),$$

生成函数 $R_n(z) = \sum_m r_n(z^m) = \prod_{k=1}^n (z^{-k} - z^k)^{-1}$ 满足 $R_n(1/z) = (-1)^n R_n(z)$ 。在下列意义下这意味

着一个对偶公式 $r_n(-m) = (-1)^{n-1} r_n(m)$, 即当我们把 $r_n(m)$ 表达作为 m 和

单位根的一个多项式时这个等式是恒等地为真的。因此我们可以说 $q_n(m) =$

$p_n(-m)$ 。关于这样的对偶性的一般的讨论可以在 G. Pólya, *Math. Zeitschrift* **29**

(1928), 549~640, § 44 中找到。)关于进一步的信息, 请见 G. Szekeres, *Quarterly J.*

Math. Oxford **2** (1951), 85~108; **4** (1953), 96~111。

当 $m = 2^{25}$ 和 $n = 512$ 时, $q_n(m)$ 的精确值是 $7.08069\ 34695\ 90264\ 094 \cdots \times 10^{1514}$; 我们的近似值给出 $7.08069\ 3501 \times 10^{1514}$ 的估计。

由习题 28, 生日检验求出 $R=0$ 的间隔的概率, 是 $b_{n00}(m)/m^n = n!(n-1)!m^{1-n}$ 。

$q_n(m) = e^{-a/4} + O(n^{-1})$, 因为来自 $b_{n01}(m)$ 的贡献 $\approx \frac{\alpha}{2n} e^{-a/4} = O(n^{-1})$ 。把因子

$g_n(z) = \sum_{k=1}^n (z^{-k} - 1)$ 插入对于 $q_n(m)$ 的积分项有以 $\frac{\alpha}{2} + O(n^{-1})$ 来乘结果的效果, 因为

$$g_n(e^{-\rho + i\delta}) = \binom{n}{2} \rho + O(n^3 \rho^2) + i t O(n^2 \delta) - \frac{1}{2} t^2 O(n^3 \delta^2) + \cdots$$

类似地, 额外的因子 $\sum_{1 \leq j < k < n} (z^{-j} - 1)(z^{-k} - 1)$ 实质上乘以 $\frac{1}{8} n^4 \rho^2 = \frac{1}{8} \alpha^2$, 加上

$O(n^{-1})$; 对于 $R=2$ 的概率的其它贡献是 $O(n^{-1})$ 。这样一来, 我们求得 r 个等间

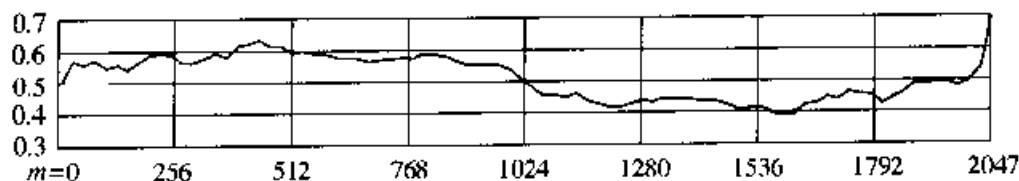
隔的概率是 $e^{-\alpha/4} (\alpha/4)^r / r! + O(n^{-1})$, 即一个泊松分布; 如果我们进行展开到 $O(n^{-2})$, 则会出现更复杂的项。

31. 79 个二进位由 24 个 3 位的集合, $\{Y_n, Y_{n+31}, Y_{n+55}\}, \{Y_{n+1}, Y_{n+32}, Y_{n+56}\}, \dots, \{Y_{n+23}, Y_{n+54}, Y_{n+78}\}$, 加上 7 个另外的二进位 $Y_{n+24}, \dots, Y_{n+30}$ 组成。后边这些位同等可能地为 0 或 1, 但是三位的每一组, 它们将是 0, 0, 0 的概率是 $\frac{1}{4}$, 它们将是 0, 1, 1 的概率是 $3/4$ 。因此, 对于二进位和的概率生成函数是 $f(z) = \left(\frac{1+z}{2}\right)^7 \left(\frac{1+3z^2}{4}\right)^{24}$, 即为一个 55 次多项式。(好, 但是还不十分好; 严格地说, 它是 $(2^{55}f(z) - 1)/(2^{55} - 1)$, 因为排除全为 0 的情况。) $2^{55}f(z)$ 的系数很容易由机器计算, 而且我们求得, 有比 0 更多的 1 的概率是 $18509401282464000/(2^{55} - 1) \approx 0.51374$ 。

注: 本习题是以 Vattulainen, Ala-Nissila 和 Kankaala 的发现 [Physical Review Letters 73 (1994), 2513~2516] 为基础的, 即一个滞后的斐波拉契生成程序通不过更复杂的二维随机步行检验。注意序列 Y_{2n}, Y_{2n+2}, \dots 也将通不过这个检验, 因为它满足同样的递推。对于 1 的偏倚也进入由 $X_n = (X_{n-55} \pm X_{n-24}) \bmod 2^r$ 生成的偶数值元素组成的子序列; 在二进记法之下, 我们趋向于有比 $(\dots 00)_2$ 更多的 $(\dots 10)_2$ 的出现。

在这个检验中对于数 79 没有什么神秘可言; 经验表明, 在长度为 101 或 1001 或 10001 的随机步行中也存在有面向 1 居多数的重要的偏倚。但是形式证明似乎是困难的。在 86 步之后生成函数是 $\left(\frac{1+3z^2}{4}\right)^{17} \left(\frac{1+2z^2+4z^3+z^4}{8}\right)^7$; 然后我们得到因于 $(1+2z^2+5z^3+5z^4+10z^5+8z^6+z^7)/32$; 然后 $(1+2z^2+7z^3+7z^4+15z^5+25z^6+29z^7+28z^8+13z^9+z^{10})/128$ 等等。当步行得更远些时, 分析变得越来越复杂。

直观上说, 出现在头 79 步中的 1 的数量优越, 只要随后的数在 0 和 1 之间相当平衡, 就会坚持, 下图示出一个小得多的情况的一些结果, 即生成程序 $Y_n = (Y_{n-2} + Y_{n-11}) \bmod 2$, 对它容易进行穷尽分析。在这种情况下, 长度为 445 的随机步行有 64% 的机会在起始点的右边结束; 仅当步行的长度增加到周期长度的一半时这种偏倚才消失 (当然, 在这之后, 有可能有更多的 0, 尽管完全的周期仍然缺一个 0)。



$Y_n = Y_{n-2} \oplus Y_{n-11}$ 时在随机 m 个元组中 1 的个数超过 0 的个数的概率

Lüscher 的抛弃技术可用来避免向 1 的偏倚(参见 3.2.2 小节的结尾)。例如,对于步 55 和 24,当在 165 的批中生成数时,如果对于每一批仅仅使用头 55 个数的话。对于长度为 1001 的随机步行未发现随机性的偏倚。

32. 相对于概率 $\left(\frac{1}{2} - \epsilon, \epsilon, \frac{1}{2}\right)$, 如果我们取值 $(-1 - 2\epsilon^2, \epsilon - 2\epsilon^2, 1 - 2\epsilon)$, 则不然。如果 ϵ 充分小, 则以 $\left(\frac{1}{2} + \epsilon\right)^2 < \frac{1}{2}$ 的概率, $X + Y > 0$ 。[因此, 基于他们的平均得分, 两个高尔夫球手可能水平相当, 但是一个有更大的可能在一轮淘汰赛中取胜, 而另一个人则将更经常地在两轮中取胜。关于对于类似现象的讨论, 参见 T. M. Cover, *Amer. Statistician* **43** (1989), 277~278。]

33. 我们实质上要 $[z^{(k+l-1)/2}] \left(\frac{1+z}{2}\right)^{k-2l} \left(\frac{1+3z^2}{4}\right)^l / (1-z)$ 。令 $m = k - 2l$ 和 $n = l$; 要求的系数是 $\frac{1}{2\pi i} \oint e^{g(z)} \frac{dz}{z(1-z)}$, 其中 $g(z) = m \ln\left(\frac{1+z}{2}\right) + n \ln\left(\frac{1+3z^2}{4}\right) - \left(\frac{m+3n-1}{2}\right) \ln z$ 。沿着 $z = e^u$ 来积分是方便的(和见机行事的), 其中 $\epsilon^2 = 4/(m+3n)$, 而对于 $-\infty < t < \infty, u = -1 + it$ 。我们有 $g(e^u) = -u/2 + u^2/2 + c_3 \epsilon u^3 + c_4 \epsilon^2 u^4 + \dots$, 其中 $c_k = \epsilon^2 \theta^k g(1)/k! = O(1)$ 。还有 $1/(1-e^u) = \frac{-1}{\epsilon u} + \frac{1}{2} - B_2 \epsilon u/2! - \dots$ 。乘出积分项和使用 $\frac{1}{2\pi i} \int_{-i\infty}^{i\infty} e^{u^2/2} \frac{du}{u} = \frac{1}{2}$ 和 $\frac{1}{2\pi i} \int_{-i\infty}^{a+i\infty} e^{u^2/2} u^{2k} du = (-1)^k (2k-1)(2k-3)\dots(1) \sqrt{2\pi}$ 产生渐近公式 $\frac{1}{2} + (2\pi)^{-1/2} n(m+3n)^{-3/2} + O((m+3n)^{-3/2})$ 。如果 $m+3n$ 是偶数, 假定我们给出 $z^{(m+3n)/2}$ 的一半系数为 1 和一半系数为 0, 则相同的渐近公式成立。(这个系数是 $\left(\frac{2}{\pi(m+3n)}\right)^{1/2} + O((m+3n)^{-3/2})$ 。)

34. 排除一个给定的两个字母的子串或子串对, 长度为 n 的串的个数为在一个适当的生成函数中 z^n 的系数, 而且它可以被写为 $ce^{\tau n} m^n + O(1)$, 其中 c 和 τ 有在 $\epsilon = 1/m$ 的幂之下的级数展开:

情况	被排除的	生成函数	c	τ
1	aa	$(1+z)/p(z)$	$1 + \epsilon^2 - 2\epsilon^3 + \dots$	$-\epsilon^2 + \epsilon^3 - \frac{5}{2}\epsilon^4 + \dots$
2	ab	$1/(1-mz+z^2)$	$1 + \epsilon^2 + 3\epsilon^4 + \dots$	$\epsilon^2 - \frac{3}{2}\epsilon^4 + \dots$
3	aa, bb	$(1+z)/(p(z)+z^2)$	$1 + 2\epsilon^2 - 4\epsilon^3 + \dots$	$-2\epsilon^2 + 2\epsilon^3 - 8\epsilon^4 + \dots$
4	aa, bc	$(1+z)/(p(z)+z^2+z^3)$	$1 + 2\epsilon^2 - 2\epsilon^3 + \dots$	$2\epsilon^2 + \epsilon^3 - 7\epsilon^4 + \dots$
5	ab, bc	$(1+z)/(1-mz+2z^2-z^3)$	$1 + 2\epsilon^2 - 2\epsilon^3 + \dots$	$-2\epsilon^2 + \epsilon^3 - 6\epsilon^4 + \dots$
6	ab, cd	$1/(1-mz+2z^2)$	$1 + 2\epsilon^2 + 12\epsilon^4 + \dots$	$-2\epsilon^2 - 6\epsilon^4 + \dots$

(这里, a, b, c, d 表示不同的字母, $p(z) = 1 - (m-1)(z+z^2)$ 。结果, 排除 $\{ab, ba\}$ 或 $\{aa, ab\}$ 的效果等价于排除 $\{aa, bb\}$; 排除 $\{ab, ac\}$ 等价于排除 $\{ab, cd\}$ 。)令 $S_n^{(j)}$ 是在情况 j 中 z^n 的系数, 并设 X 是不出现的两个字母组合的总数, 于是 $EX = (mS_n^{(1)} + m^2 S_n^{(2)})/m^n$, 而且

$$EX^2 = (mS_n^{(1)} + m^2(S_n^{(2)} + 6S_n^{(3)}) + 2m^3(S_n^{(4)} + S_n^{(5)} + S_n^{(6)}) + m^4S_n^{(6)})/m^n$$

35. a) $ES_m = N^{-1} \sum_{n=0}^{N-1} \sum_{j=0}^{m-1} Z_{n+j} = N^{-1} \sum_{j=0}^{m-1} \sum_{n=0}^{N-1} Z_{n+j} = m/N$, 因为 $\sum_{n=0}^{N-1} Z_{n+j} = 2^{k-1} - (2^{k-1} - 1) = 1$ 。

b) 令 $\xi^k = a_1 \xi^{k-1} + \dots + a_k$, 并且把线性函数 f 定义为习题 3.2.2-16 的头一个解。于是 $Y_n = f(\xi^n)$, 而且由此得出, $Y_{n+i} + Y_{n+j} = f(\xi^{n+i}) + f(\xi^{n+j}) \equiv f(\xi^{n+i} + \xi^{n+j}) - f(\xi^n \alpha) \pmod{2}$, 其中当 $i \neq j \pmod{N}$ 时 α 非零。因此 $ES_m^2 = N^{-1} \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \sum_{n=0}^{N-1} Z_{n+i} Z_{n+j} = N^{-1} (\sum_{i=0}^{m-1} \sum_{n=0}^{N-1} Z_{n+i}^2 + 2 \sum_{0 \leq i < j < m} \sum_{n=0}^{N-1} Z_n) = m - m(m-1)/N$ 。

c) 当每个 Z_n 都是真正随机的时, $E \sum_{j=0}^{m-1} Z_{n+j} = \sum_{j=0}^{m-1} EZ_{n+j} = 0$ 和 $E(\sum_{j=0}^{m-1} Z_{n+j})^2 = \sum_{j=0}^{m-1} EZ_{n+j}^2 + \sum_{0 \leq i < j < m} (EZ_{n+i})(EZ_{n+j}) = m$ 。因此当 $m \ll N$ 时 S_m 的均值和方差非常接近于正确的值。

d) $ES_m^3 = N^{-1} \sum_{h=0}^{m-1} \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \sum_{n=0}^{N-1} Z_{n+h} Z_{n+i} Z_{n+j}$ 。如果 h, i 或 j 的任何一个都相等, 则对 n 的求和为 1; 因此

$$ES_m^3 = \frac{1}{N} (m^3 - m^3 + 6 \sum_{0 \leq h < i < j < m} \sum_{n=0}^{N-1} Z_{n+h} Z_{n+i} Z_{n+j})$$

像在 b) 中那样进行论证, 我们发现若果 $\xi^h + \xi^i + \xi^j \neq 0$, 对 n 求和将为 1; 否则它将是 $-N$ 。因此 $ES_m^3 = m^3 - 6B(N+1)/N$, 其中 $B = \sum_{0 \leq h < i < j < m} [\xi^h + \xi^i + \xi^j = 0] = \sum_{0 \leq i < j < m} [1 + \xi^i + \xi^j = 0](m-j)$ 。最后发现这个域中 $1 + \xi^i = \xi^j$ 当且仅当对于 $0 < l < k$, $f(\xi^{i+l}) = f(\xi^{j+l})$, 假定 $0 < i < j < N$ 。

e) 对 $i=31$ 和 $j=55$ 有惟一非零项出现; 因此 $B=79-55=24$ 。(当 $i=62$ 和 $j=110$ 时下一个非零项出现。) 在真正随机的情况下, ES_m^3 应当为零, 所以这个值 $ES_{79}^3 \approx -144$ 是显著地非随机的。奇怪的是它是负的, 尽管习题 31 证明, S_{79} 通常是正的。当它确实骤降到零之下时, S_{79} 的值趋向于更严重地为负。

参考文献: IEEE Trans IT-14(1968), 569~576。通过 M. Matsumoto 和 Y. Kurita 的实验 [ACM Trans Modeling and Comp. Simul. 2 (1992), 179~194; 4 (1994), 254~266] 确认基于三项式的生成程序不能通过这样的分布检验, 即使当步子很大时。也见 ACM Trans. Modeling and Comp. Simul. 6 (1996), 99~106, 其中他们揭示了低密度的指数级长的子序列。

3.3.3 小节

$$1. y((x/y)) + \frac{1}{2}y - \frac{1}{2}y\delta(x/y)。$$

$$2. ((x)) = - \sum_{n \geq 1} \frac{1}{n\pi} \sin 2\pi nx, \text{ 它对于所有的 } x \text{ 都收敛。 (在等式(24)中的表示}$$

可被认为是对 x 为有理数的情况的“有限”的傅里叶级数。)

3. 和是 $((2^n x)) - ((x))_c$ [见 *Trans. Amer. Math. Soc.* **65** (1949), 401.]

4. $d_{\max} = 2^{10} \cdot 5$ 。注意 $X_{n+1} < X_n$ 的概率是 $\frac{1}{2} + \epsilon$, 其中

$$|\epsilon| < d/(2 \cdot 10^{10}) \leq 1/(2 \cdot 5^9)$$

所以从定理 P 的观点看来, 每个效能为 10 的生成程序都是很不错的。

5. 一个中间结果:

$$\sum_{0 \leq x < m} \frac{x}{m} \frac{s(x)}{m} = \frac{1}{12} \sigma(a, m, c) + \frac{m}{4} - \frac{c}{2m} - \frac{x'}{2m}$$

6. a) 用归纳法和公式

$$\left(\left(\frac{hj+c}{k} \right) \right) - \left(\left(\frac{hj+c-1}{k} \right) \right) = \frac{1}{k} - \frac{1}{2} \delta \left(\frac{hj+c}{k} \right) - \frac{1}{2} \delta \left(\frac{hj+c-1}{k} \right)$$

b) 利用事实 $-\left(\left(\frac{h'j}{k} \right) \right) = -\left(\left(\frac{j}{hk} - \frac{k'j}{h} \right) \right) = \left(\left(\frac{k'j}{h} \right) \right) - \frac{j}{hk} + \frac{1}{2} \delta \left(\frac{k'j}{h} \right)$ 。

7. 在习题 1.2.4-45 的第二个公式中取 $m=h, n=k, k=2$:

$$\sum_{0 \leq j < k} \left(\frac{hj}{k} - \left(\left(\frac{hj}{k} \right) \right) + \frac{1}{2} \right) \left(\frac{hj}{k} - \left(\left(\frac{hj}{k} \right) \right) - \frac{1}{2} \right) +$$

$$2 \sum_{0 \leq j < h} \left(\frac{kj}{h} - \left(\left(\frac{kj}{h} \right) \right) + \frac{1}{2} \right) j = kh(h-1)$$

简化左边的和数, 并由标准操作得到

$$h^2k - hk - \frac{h}{2} + \frac{h^2}{6k} + \frac{k}{12} + \frac{1}{4} - \frac{h}{6} \sigma(h, k, 0) - \frac{h}{6} \sigma(k, h, 0) +$$

$$\frac{1}{12} \sigma(1, k, 0) = h^2k - hk$$

由于 $\sigma(1, k, 0) = (k-1)(k-2)/k$, 这约化为互反律。

8. 见 *Duke Math. J.* **21** (1954), 391~397。

9. 由有趣的恒等式 $\sum_{k=0}^{p-1} \lfloor kp/r \rfloor \lfloor kq/r \rfloor + \sum_{k=0}^{p-1} \lfloor kq/p \rfloor \lfloor kr/p \rfloor + \sum_{k=0}^{q-1} \lfloor kr/q \rfloor \lfloor kp/q \rfloor = (p-1)(q-1)(r-1)$ 开始, 假定 $p \perp q, q \perp r$ 和 $r \perp p$, 对于它可能有一个简单的几何证明。[U. Dieter, *Abh. math. Sem. Univ. Hamburg* **21** (1957), 109~125.]

10. 由[8], 显然 $\sigma(k-h, k, c) = -\sigma(h, k, -c)$ 。在定义(16)中用 $k-j$ 代替 j 来导出 $\sigma(h, k, c) = \sigma(h, k, -c)$

11. a) $\sum_{0 \leq j < dk} \left(\left(\frac{j}{dk} \right) \right) \left(\left(\frac{hj+c}{k} \right) \right) = \sum_{\substack{0 \leq i < d \\ 0 \leq j < k}} \left(\left(\frac{ik+j}{dk} \right) \right) \left(\left(\frac{hj+c}{k} \right) \right)$; 利用(10)来对 i

求和。

b) $\left(\left(\frac{hj+c+\theta}{k} \right) \right) = \left(\left(\frac{hj+c}{k} \right) \right) + \frac{\theta}{k} - \frac{1}{2} \delta \left(\frac{hj+c}{k} \right)$; 现在求和。

12. 由于 $\left(\left(\frac{hj+c}{k} \right) \right)$ 以某个次序取遍同 $\left(\left(\frac{j}{k} \right) \right)$ 一样的值, 柯西不等式意味着

$\sigma(h, k, c)^2 \leq \sigma(h, k, 0)^2$; 而且 $\sigma(1, k, 0)$ 可以直接地求和, 参考习题 7。

13. 如果 $hh' \equiv 1 \pmod{k}$, 则

$$\sigma(h, k, c) + \frac{3(k-1)}{k} = \frac{12}{k} \sum_{0 \leq j < k} \frac{\omega^{-hj}}{(\omega^{-hj} - 1)(\omega^j - 1)} + \frac{6}{k}(c \bmod k) - 6 \left(\left(\frac{h'c}{k} \right) \right)$$

14. $(2^{38} - 3 \cdot 2^{20} + 5)/(2^{70} - 1) \approx 2^{-32}$ 。尽管有局部的非随机性, 但是整体值令人极端满意。

15. 用 $\lfloor c \rfloor$ 和 $\lceil c \rceil$ 来代替 (19) 中出现的 c^2 。

16. 对于 $1 \leq r \leq t$, 所提示的恒等式等价于 $m_1 = p_r m_{r+1} + p_{r-1} m_{r+2}$, 这由归纳法得出。(也可参考 4.5.3-32。)现在用 $\sum_{j \leq r \leq t} b_r m_{r+1}$ 代替 c_j , 并比较有待证明的恒等式两边 b_j 的系数。

注: 对于所有指数 $e \geq 1$, 我们利用类似的论证可得

$$\sum_{1 \leq j \leq t} (-1)^{j+1} \frac{c_j^e}{m_j m_{j+1}} = \frac{1}{m_1} \sum_{1 \leq r \leq t} (-1)^{r+1} b_r \frac{(c_j^e - c_{j+1}^e)}{c_j - c_{j+1}} p_{j-1}$$

17. 在这个算法运行期间, 对于 $j = 1, 2, \dots, t+1$, 我们将有 $k = m_j$, $h = m_{j+1}$, $c = c_j$, $p = p_{j-1}$, $p' = p_{j-2}$, $s = (-1)^{j+1}$ 。

D1. [初始化] 置 $A \leftarrow 0$, $B \leftarrow h$, $p \leftarrow 1$, $p' \leftarrow 0$, $s \leftarrow 1$ 。

D2. [除] 置 $a \leftarrow \lfloor k/h \rfloor$, $b \leftarrow \lfloor c/h \rfloor$, $r \leftarrow c \bmod h$ (现在 $a = a_j$, $b = b_j$, 且 $r = c_{j+1}$)。

D3. [累加] 置 $A \leftarrow A + (a - 6b)s$, $B \leftarrow B + 6bp(c + r)s$ 。如果 $r \neq 0$, 或 $c = 0$, 置 $A \leftarrow A - 3s$ 。如果 $h = 1$, 置 $B \leftarrow B + ps$ 。(这减去 $3e(m_{j+1}, c_j)$ 并注意 $\sum (-1)^{j+1}/m_j m_{j+1}$ 个项。)

D4. [准备下一次迭代] 置 $c \leftarrow r$, $s \leftarrow -s$; 置 $r \leftarrow k - ah$, $k \leftarrow h$, $h \leftarrow r$; 置 $r \leftarrow ap + p'$, $p' \leftarrow p$, $p \leftarrow r$ 。如果 $h > 0$, 就返回 D2。 ■

在这个算法结束时, p 将等于 k 原来的值 k_0 , 所以所求的答案将是 $A + B/p$ 。如果 $s < 0$, 则 p' 最后的值将是 h' , 否则 p' 将是 $k_0 - h'$ 。通过对 A 作适当的调整, 有可能把 B 保持在 $0 \leq B < k_0$ 的范围内, 因此如果 k_0 是单精度的数, 就只要求单精度的操作(以及双精度的乘积和被除数)。

18. 稍经思索即知对于所有的 $z \geq 0$ (而不仅仅当 $k \geq z$ 时) 公式

$$S(h, k, c, z) = \sum_{0 \leq j < k} (\lfloor j/k \rfloor - \lfloor (j-z)/k \rfloor) ((hj + c)/k)$$

事实上是正确的。令 $\lfloor j/k \rfloor - \lfloor (j-z)/k \rfloor = \frac{z}{k} + \left(\left(\frac{j-z}{k} \right) \right) - \left(\left(\frac{j}{k} \right) \right) + \frac{1}{2} \delta_{j,0} - \frac{1}{2} \delta \left(\frac{j-z}{k} \right)$ 并求和即得 $S(h, k, c, z) = \frac{zd}{k} \left(\left(\frac{c}{d} \right) \right) + \frac{1}{12} \sigma(h, k, hz + c) - \frac{1}{12} \sigma(h, k, c) + \frac{1}{2} \left(\left(\frac{c}{k} \right) \right) - \frac{1}{2} \left(\left(\frac{hz + c}{k} \right) \right)$, 其中 $d = \gcd(h, k)$ 。[给定 α , 这个公式允许我们借助于广义戴德金和表达 $X_{n+1} < X_n < \alpha$ 的概率。]

19. 所求概率是

$$\begin{aligned}
& m^{-1} \sum_{x=0}^{m-1} \left(\left\lfloor \frac{x-\alpha}{m} \right\rfloor - \left\lfloor \frac{x-\beta}{m} \right\rfloor \right) \left(\left\lfloor \frac{s(x)-\alpha'}{m} \right\rfloor - \left\lfloor \frac{s(x)-\beta'}{m} \right\rfloor \right) = \\
& m^{-1} \sum_{x=0}^{m-1} \left(\frac{\beta-\alpha}{m} + \left(\left\lfloor \frac{x-\beta}{m} \right\rfloor - \left\lfloor \frac{x-\alpha}{m} \right\rfloor \right) + \frac{1}{2} \delta \left(\frac{x-\alpha}{m} \right) - \frac{1}{2} \delta \left(\frac{x-\beta}{m} \right) \right) \times \\
& \left(\frac{\beta'-\alpha'}{m} + \left(\left\lfloor \frac{s(x)-\beta'}{m} \right\rfloor - \left\lfloor \frac{s(x)-\alpha'}{m} \right\rfloor \right) + \frac{1}{2} \delta \left(\frac{s(x)-\alpha'}{m} \right) - \frac{1}{2} \delta \left(\frac{s(x)-\beta'}{m} \right) \right) = \\
& \frac{\beta-\alpha}{m} \frac{\beta'-\alpha'}{m} + \frac{1}{12m} (\sigma(a, m, c + a\alpha - \alpha') - \sigma(a, m, c + a\alpha - \beta') + \\
& \sigma(a, m, c + a\beta - \beta') - \sigma(a, m, c + a\beta - \alpha')) + \epsilon
\end{aligned}$$

其中 $|\epsilon| \leq 2.5/m$ 。

[这个方法是由 U. Dieter 给出的。根据定理 K, 真正的概率与理想的值 $\frac{\beta-\alpha}{m}$ 的差异不超过 $\sum_{j=1}^r a_j/4m$; 反之, 通过适当地选择 $\alpha, \beta, \alpha', \beta'$, 由定理 K 是“最好的”这一事实, 当有很大的部分商时, 我们至少能把差异的界缩小一半。注意, 当 $a \approx \sqrt{m}$ 时, 这个差异不能超过 $O(1/\sqrt{m})$, 所以甚至习题 14 中那个局部非随机的生成程序也能很好地通过全周期上的序列检验; 看来我们应当坚持极小的差异。]

20. $\sum_{0 \leq x < m} \left\lfloor (x - s(x))/m \right\rfloor \left\lfloor (s(x) - s(s(x)))/m \right\rfloor / m = \sum_{0 \leq x < m} \left((x - s(x))/m + (((bx + c)/m)) + \frac{1}{2} \right) \left((s(x) - s(s(x)))/m + ((a(bx + c)/m)) + \frac{1}{2} \right) / m$; 而且 $x/m = ((x/m)) + \frac{1}{2} - \frac{1}{2} \delta(x/m)$, $s(x)/m = (((ax + c)/m)) + \frac{1}{2} - \frac{1}{2} \delta((ax + c)/m)$, $s(s(x))/m = (((a^2x + ac + c)/m)) + \frac{1}{2} - \frac{1}{2} \delta(a^2x + ac + c)/m$ 。设 $s(x') = s(s(x'')) = 0$ 且 $d = \gcd(b, m)$ 。如果 $a'a \equiv 1 \pmod{m}$, 则现在这个和约简为

$$\begin{aligned}
& \frac{1}{4} + \frac{1}{12m} (S_1 - S_2 + S_3 - S_4 + S_5 - S_6 + S_7 - S_8 + S_9) + \frac{d}{m} \left(\left(\frac{c}{d} \right) \right) + \\
& \frac{1}{2m} \left(\left(\left(\frac{x' - x''}{m} \right) \right) - \left(\left(\frac{x'}{m} \right) \right) + \left(\left(\frac{x''}{m} \right) \right) + \right. \\
& \left. \left(\left(\frac{ax' + c}{m} \right) \right) - \left(\left(\frac{ac}{m} \right) \right) - \left(\left(\frac{c}{m} \right) \right) - \frac{1}{2} \right)
\end{aligned}$$

其中 $S_1 = \sigma(a, m, c)$, $S_2 = \sigma(a^2, m, ac + c)$, $S_3 = \sigma(ab, m, ac)$, $S_4 = \sigma(1, m, 0) = (m-1)(m-2)/m$, $S_5 = \sigma(a, m, c)$, $S_6 = \sigma(b, m, c)$, $S_7 = -\sigma(a'-1, m, a'c)$ 及 $S_8 = -\sigma(a'(a'-1), m, (a')^2c)$; 而且最后

$$S_9 = 12 \sum_{0 \leq x < m} \left(\left(\frac{bx + c}{m} \right) \right) \left(\left(\frac{a(bx + c)}{m} \right) \right) =$$

$$\begin{aligned}
& 12d \sum_{0 \leq x < m/d} \left(\left(\frac{x + c_0/d}{m/d} \right) \right) \left(\left(\frac{a(x + c_0/d)}{m/d} \right) \right) = \\
& 12d \sum_{0 \leq x < m/d} \left(\left(\frac{x}{m/d} \right) + \frac{c_0}{m} - \frac{1}{2} \delta_{x0} \right) \left(\left(\frac{a(x + c_0/d)}{m/d} \right) \right) = \\
& d \left(\sigma(ad, m, ac_0) + 12 \frac{c_0}{m} \left(\left(\frac{ac_0}{d} \right) \right) - 6 \left(\left(\frac{ac_0}{m} \right) \right) \right)
\end{aligned}$$

其中 $c_0 = c \bmod d$ 。当 d 很小且分数 $a/m, (a^2 \bmod m)/m, (ab \bmod m)/m, b/m, (a'-1)/m, (a'(a'-1) \bmod m)/m, ((ad) \bmod m)/m$ 的部分商全都很小时, 总和将接近于 $\frac{1}{6}$ 。(注意, $a'-1 \equiv -b + b^2 - \dots$, 如同在习题 3.2.1.3-7 中那样。)

21. 首先注意, 主要的积分可很好地分解:

$$s_n = \int_{x_n}^{x_{n+1}} x |ax + \theta| dx = \frac{1}{a^2} \left(\frac{1}{3} - \frac{\theta}{2} + \frac{n}{2} \right), \text{ 如果 } x_n = \frac{n-\theta}{a};$$

$$s = \int_0^1 x |ax + \theta| dx = s_0 + s_1 + \dots + s_{a-1} + \int_{\theta/a}^0 (ax + \theta) dx = \frac{1}{3a} - \frac{\theta}{2a} + \frac{a-1}{4a} + \frac{\theta^2}{2a}.$$

$$\text{因此 } C = \left(s - \left(\frac{1}{2} \right)^2 \right) / \left(\frac{1}{3} - \left(\frac{1}{2} \right)^2 \right) = (1 - 6\theta + 6\theta^2)/a.$$

22. 在不相交的区间 $\left[\frac{1-\theta}{a}, \frac{1-\theta}{a-1} \right), \left[\frac{2-\theta}{a}, \frac{2-\theta}{a-1} \right), \dots, \left[\frac{a-\theta}{a}, 1 \right)$ 中, 我们有 $s(x) < x$, 这些区间有长度

$$1 + \sum_{0 < j \leq a-1} \left(\frac{j-\theta}{a-1} \right) - \sum_{0 < j \leq a} \left(\frac{j-\theta}{a} \right) = 1 + \frac{a}{2} - \theta - \frac{a+1}{2} + \theta = \frac{1}{2}$$

23. 当对于 $0 < j \leq k < a$, x 在 $\left[\frac{k-\theta}{a}, \frac{k-\theta}{a-1} \right)$ 之内和 $ax + \theta - k$ 在 $\left[\frac{j-\theta}{a}, \frac{j-\theta}{a-1} \right)$ 之内时; 或者当 x 在 $\left[\frac{a-\theta}{a}, 1 \right)$ 内且 $ax + \theta - a$ 在 $\left[\frac{j-\theta}{a}, \frac{j-\theta}{a-1} \right)$ 之内 (对于 $0 < j \leq \lfloor a\theta \rfloor$) 或 $\left[\frac{\lfloor a\theta \rfloor + 1 - \theta}{a}, \theta \right)$ 之内时, 我们有 $s(s(x)) < s(x) < x$, 所求的概率是

$$\begin{aligned}
& \sum_{0 < j \leq k < a} \frac{j-\theta}{a^2(a-1)} + \sum_{0 < j \leq \lfloor a\theta \rfloor} \frac{j-\theta}{a^2(a-1)} + \frac{1}{a^2} \max(0, \lfloor a\theta \rfloor + \theta - 1) = \\
& \frac{1}{6} + \frac{1}{6a} - \frac{\theta}{2a} + \frac{1}{a^2} \left(\frac{\lfloor a\theta \rfloor (\lfloor a\theta \rfloor + 1 - 2\theta)}{2(a-1)} + \max(0, \lfloor a\theta \rfloor + \theta - 1) \right)
\end{aligned}$$

对于很大的 a , 它等于 $\frac{1}{6} + (1 - 3\theta + 3\theta^2)/6a + O(1/a^2)$ 。注意, $1 - 3\theta + 3\theta^2 \geq \frac{1}{4}$,

所以 θ 不能选择成使这个概率为真。

24. 沿用上题的解法, 区间长度的和是

$$\sum_{0 < j_1 < \dots < j_{t-1} < a} \frac{j_1}{a^{t-1}(a-1)} = \frac{1}{a^{t-1}(a-1)} \binom{a+t-2}{t}$$

为计算平均长度, 设 p_k 为长度大于等于 k 的一个运行的概率; 平均值是

$$\sum_{k=1}^{\infty} p_k = \sum_{k=1}^{\infty} \binom{a+k-2}{k} \frac{1}{a^{k-1}(a-1)} = \left(\frac{a}{a-1}\right)^a = \frac{a}{a-1}$$

一个真正的随机序列的值将是 $e-1$; 而我们的值是 $e-1+(e/2-1)/a+O(1/a^2)$ 。

[注: 对于一个递增的运行同样的结果成立, 因为我们有 $U_n > U_{n+1}$ 当且仅当 $1-U_n < 1-U_{n+1}$ 。这将使我们怀疑, 线性同余序列中的运行可能比通常的运行更长些, 所以应当把运行检验应用到这样的生成程序上。]

25. 对于某一个 k , x 必在区间 $[(k+\alpha'-\theta)/a, (k+\beta'-\theta)/a)$ 中, 并也在区间 $[\alpha, \beta)$ 中。设 $k_0 = \lceil a\alpha + \theta - \beta' \rceil$, $k_1 = \lceil a\beta + \theta - \beta' \rceil$ 。由于边界条件, 我们得到概率 $(k_1 - k_0)(\beta' - \alpha')/a + \max(0, \beta - (k_1 + \alpha' - \theta)/a) - \max(0, \alpha - (k_0 + \alpha' - \theta)/a)$ 。这是 $(\beta - \alpha)(\beta' - \alpha') + \epsilon$, 其中 $|\epsilon| < 2(\beta' - \alpha')/a$ 。

26. 见图 A-1, 次序 $U_1 < U_3 < U_2$ 和 $U_2 < U_3 < U_1$ 是不可能的; 其它四种次序的概率均为 $\frac{1}{4}$ 。

27. $U_k = \{F_{k-1}U_0 + F_kU_1\}$ 。我们需要有 $F_{k-1}U_0 + F_kU_1 < 1$ 和 $F_kU_0 + F_{k+1}U_1 > 1$ 。与 $U_0 > U_1$ 相对应的半单位正方形按 k 的不同值, 被分割成如图 A-2 所示。如果 $k=1$, 则对于长度为 k 的一个运行的概率为 $\frac{1}{2}$; 如果 $k>1$, 则概率为 $1/F_{k-1}F_{k+1} - 1/F_kF_{k+2}$ 。对于一个随机序列, 相应的概率是 $2k/(k+1)! - 2(k+1)/(k+2)!$; 下表比较了开头的一些值。

k :	1	2	3	4	5
斐波那契情况下的概率:	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{10}$	$\frac{1}{24}$	$\frac{1}{65}$
随机情况的概率:	$\frac{1}{3}$	$\frac{5}{12}$	$\frac{11}{60}$	$\frac{19}{360}$	$\frac{29}{2520}$

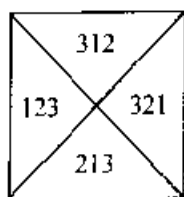


图 A-1 斐波那契生成程序的排列区域

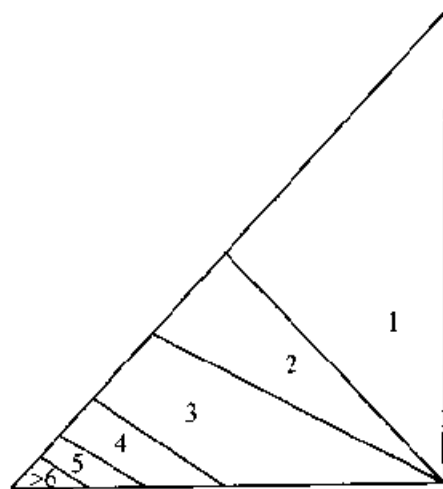


图 A-2 斐波那契生成程序的运行长度区域

28. 图 A-3 示出一般情况下的各种区域。如果 U_1 和 U_2 是随机地选择的, “213”区域意味着 $U_2 < U_1 < U_3$; 区域“321”意味着 $U_3 < U_2 < U_1$, 等等。123 和 321 的概率都是 $\frac{1}{4} - \alpha/2 + \alpha^2/2$; 所有其余情况的概率都是 $\frac{1}{8} + \alpha/4 - \alpha^2/4$ 。为使所有这些概率都等于 $\frac{1}{6}$, 我们必须有 $1 - 6\alpha + 6\alpha^2 = 0$ 。[这一习题建立了 J. N. Franklin 给出的一个定理, 见 *Math. Comp.* 17 (1963), 28 - 59, 定理 13; Franklin 的论文的其他结果同习题 22 和 23 有关。]

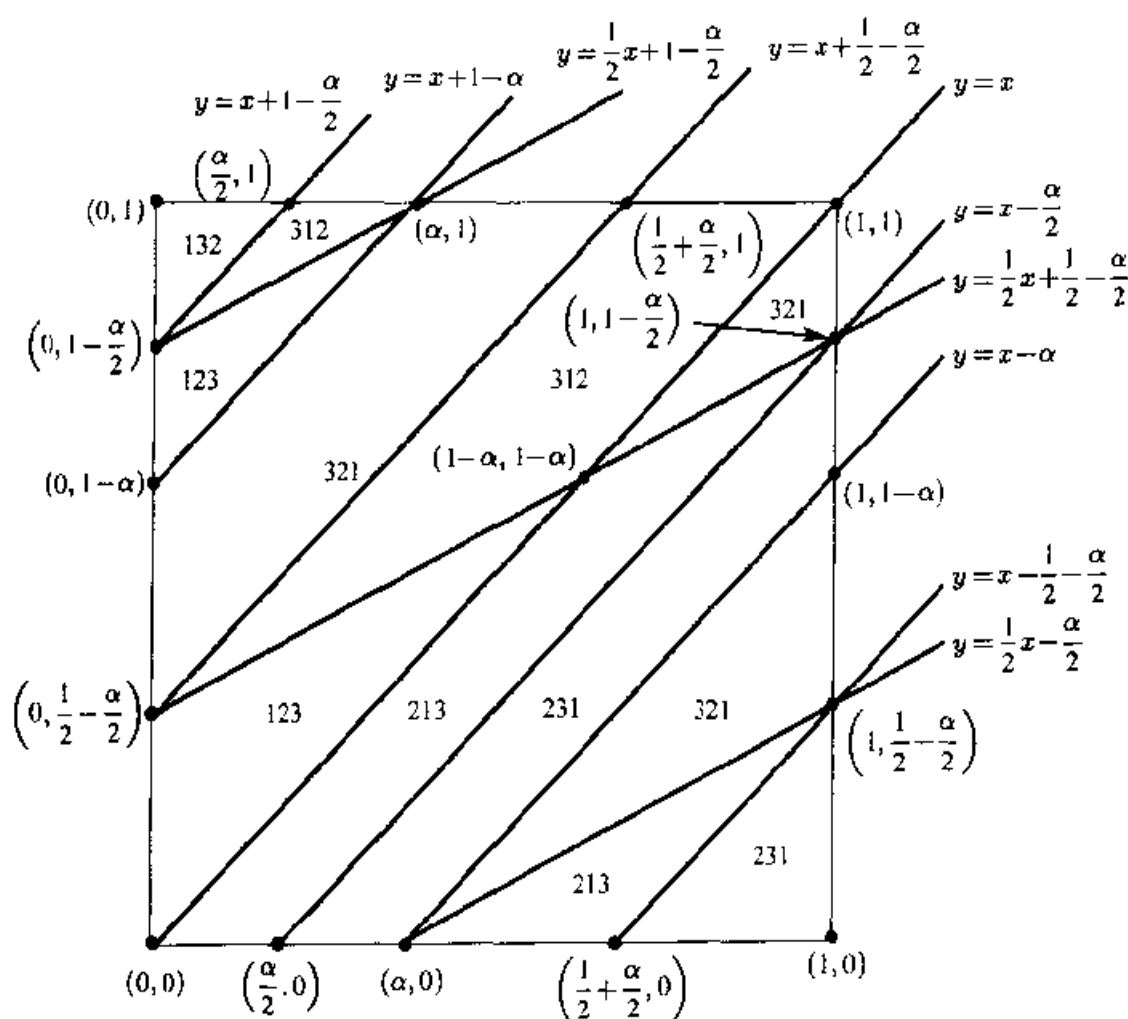


图 A-3 具有效能 2 的生成程序的排列区域; $\alpha = (a-1)c/m$

3.3.4 小节

1. 对于极大周期的生成程序, 一维的精度 ν_1 总是 m 且 $\mu_1 = 2$ 。

2. 设 V 是行为 V_1, \dots, V_t 的矩阵。在 $Y \neq (0, \dots, 0)$ 和 VY 是整列向量 X 这两个条件下使 $Y \cdot Y$ 极小化, 等价于在 X 是非零整列向量的条件下使 $(V^{-1}X) \cdot (V^{-1}X)$ 极小化。

X)极小化。 V^{-1} 的列是 U_1, \dots, U_l 。

3. $a^2 \equiv 2a - 1$, 且 $a^3 \equiv 3a - 2 \pmod{m}$ 。通过考察(15)的所有短的解,除了下列情况外,对于向量 $(1, -2, 1)$ 和 $(1, -1, -1, 1)$ 我们分别求得 $v_3^2 = 6$ 和 $v_4^2 = 4$ 。这些情况是: $m = 2^e q$, q 为奇数, $e \geq 3$, $a \equiv 2^{e-1} \pmod{2^e}$, $a \equiv 1 \pmod{q}$, $v_3^2 = v_4^2 = 2$; $m = 3^e q$, $3 \nmid q$, $e \geq 2$, $a \equiv 1 \pmod{3^e}$, $a \equiv 1 \pmod{q}$, $v_4^2 = 2$; $m = 9$, $a = 4$ 或 7 , $v_2^2 = v_3^2 = 5$ 。

4. a) 对于 (x_1, x_2) 的惟一选择是 $\frac{1}{m}(y_1 u_{22} - y_2 u_{21}, -y_1 u_{12} + y_2 u_{11})$, 而且这 $\equiv \frac{1}{m}(y_1 u_{22} + y_2 a u_{22}, -y_1 u_{12} - y_2 a u_{12}) \equiv (0, 0) \pmod{1}$; 即 x_1 和 x_2 都是整数。

b) 当 $(x_1, x_2) \neq (0, 0)$ 时, 我们有 $(x_1 u_{11} + x_2 u_{21})^2 + (x_1 u_{12} + x_2 u_{22})^2 = x_1^2(u_{11}^2 + u_{12}^2) + x_2^2(u_{21}^2 + u_{22}^2) + 2x_1 x_2(u_{11} u_{21} + u_{12} u_{22})$, 而由假设, 这大于等于 $(x_1^2 + x_2^2 - |x_1 x_2|)(u_{11}^2 + u_{12}^2) \geq u_{11}^2 + u_{12}^2$ 。

[注意, 这是比引理 A 更强的结果, 引理 A 仅告诉我们 $x_1^2 \leq (u_{11}^2 + u_{12}^2)(u_{21}^2 + u_{22}^2)/m^2$ 和 $x_2^2 \leq (u_{11}^2 + u_{12}^2)^2/m^2$, 其中后者可以大于等于 1。这一思想实质上是高斯的一个约简的二元二次型的概念, 见 *Disquisitiones Arithmeticae* (Leipzig: 1801), § 171.]

5. 条件(30)保持不变, 因此当 a 与 m 互素时 h 在步骤 S2 中不能为零。由于在该步中 h 总是减小的, S2 最终以 $u^2 + v^2 \geq s$ 结束。注意, 在整个计算过程中 $pp' \leq 0$ 。

所提示的不等式在头一次执行 S2 时确实成立。由等式(24)可知, 极小化 $(h' - q'h)^2 + (p' - q'p)^2$ 的整数 q' 是 $q' = \text{round}((h'h + p'p)/(h^2 + p^2))$ 。如果 $(h' - q'h)^2 + (p' - q'p)^2 < h^2 + p^2$, 我们必定有 $q' \neq 0$, $q' \neq -1$, 因此 $(p' - q'p)^2 \geq p^2$, 因此 $(h' - q'h)^2 < h^2$, 即 $|h' - q'h| < h$, 即 q' 是 q 或 $q+1$ 。我们有 $hu + pv \geq h(h' - q'h) + p(p' - q'p) \geq -\frac{1}{2}(h^2 + p^2)$, 所以如果 $u^2 + v^2 < s$, 步骤 S2 的下一迭代将保持提示中的假定。如果 $u^2 + v^2 \geq s > (u-h)^2 + (v-p)^2$, 我们有 $2|h(u-h) + p(v-p)| = 2(h(h-u) + p(p-v)) = (u-h)^2 + (v-p)^2 + h^2 + p^2 - (u^2 + v^2) \leq (u-h)^2 + (v-p)^2 \leq h^2 + p^2$, 因此由习题 4, $(u-h)^2 + (v-p)^2$ 是极小的。最后如果 $u^2 + v^2$ 和 $(u-h)^2 + (v-p)^2$ 都大于等于 s , 令 $u' = h' - q'h$, $v' = p' - q'p$; 则 $2|hu' + pv'| \leq h^2 + p^2 \leq u'^2 + v'^2$, 且由习题 4, $h^2 + p^2$ 是极小的。

[Kaib 和 Schnorr 讨论了相对于其它的度量求最短的 2 维向量的推广, *J. Algorithms* 21 (1996), 565~578]。

6. 在上一答案中, 如果 $u^2 + v^2 \geq s > (u-h)^2 + (v-p)^2$ 我们有 $(v-p)^2 > v^2$, 因此 $(u-h)^2 < u^2$; 而且如果 $q = a_j$, 使得 $h' = a_j h + u$, 我们必定有 $a_{j+1} = 1$ 。由此得出在习题 3.3.3-16 的记号下, $v_2^2 = \min_{0 \leq j < t} (m_j^2 + p_{j-1}^2)$ 。

现在我们有 $m_0 = m_j p_j + m_{j+1} p_{j-1} = a_j m_j p_{j-1} + m_j p_{j-2} + m_{j+1} p_{j-1} < (a_j + 1 +$

$1/a_j)m_j p_{j-1} \leq (A+1+1/A)m_j p_{j-1}$, 而且 $m_j^2 + p_{j-1}^2 \geq 2m_j p_{j-1}$, 因此就有所求结果。

7. 利用条件(19), 我们将证明, 对于所有 $k \neq j$, $U_j \cdot U_k = 0$ 当且仅当对于所有 $k \neq j$, $V_j \cdot V_k = 0$ 。假定对于所有 $k \neq j$, $U_j \cdot U_k = 0$, 并设 $U_j = \alpha_1 V_1 + \cdots + \alpha_t V_t$ 。于是对于所有 k , $U_j \cdot U_k = \alpha_k$, 因此对于所有 $k \neq j$, $U_j = \alpha_j V_j$, 而且 $V_j \cdot V_k = \alpha_j^{-1}(U_j \cdot V_k) = 0$ 。一个对称的论证即可证明其反面。

8. 显然, $\nu_{t+1} \leq \nu_t$ (这是算法 S 中隐含的一个事实, 因为当 t 增加 s 不变)。对于 $t=2$, 这等价于 $(m\mu_2/\pi)^{1/2} \geq \left(\frac{3}{4}m\mu_3/\pi\right)^{1/3}$, 即 $\mu_3 \leq \frac{4}{3}\sqrt{m/\pi}\mu_2^{3/2}$, 这个上限归结为 $\frac{4}{3}10^{-4}/\sqrt{\pi}$ 连同给定的参数, 但对于很大的 m 和固定的 μ_2 , 上限(40)是更好的。

9. 设 $f(y_1, \dots, y_t) = \theta$; 则 $\gcd(y_1, \dots, y_t) = 1$, 所以有行列式为 1 的整矩阵 W 以 (y_1, \dots, y_t) 为它的第一行。(对于这行中最小非零项的量用归纳法来证明这一事实。)现在如果 $X = (x_1, \dots, x_t)$ 是一个行向量, 我们有 $XW = X'$ 当且仅当 $X = X'W^{-1}$, 且 W^{-1} 是行列式为 1 的整矩阵, 因此由 WU 定义的 g 型满足 $g(x_1, \dots, x_t) = f(x'_1, \dots, x'_t)$; 而且 $g(1, 0, \dots, 0) = \theta$ 。

不失一般性, 假定 $f = g$ 。如果现在 S 是任意正交矩阵, 则矩阵 US 定义和 U 相同的型, 这是由于 $(XUS)(XUS)^T = (XU)(XU)^T$ 。选择 S 使得它的头一列是 U_1^T 的一个倍数, 而它的其它列是任何适当的向量, 对于某些 $\alpha_1, \alpha_2, \dots, \alpha_t$ 和 $(t-1) \times (t-1)$ 矩阵 U' , 我们有

$$US = \begin{bmatrix} \alpha_1 & 0 & \cdots & 0 \\ \alpha_2 & & & \\ \vdots & & U' & \\ \alpha_t & & & \end{bmatrix}$$

因此, $f(x_1, \dots, x_t) = (\alpha_1 x_1 + \cdots + \alpha_t x_t)^2 + h(x_2, \dots, x_t)$ 。由此得出 $\alpha_1 = \sqrt{\theta}$ [事实上, 对于 $1 \leq j \leq t$, $\alpha_j = (U_1 \cdot U_j)/\sqrt{\theta}$], 而且 h 是由 U' 定义的正定二次型, 其中 $\det U' = (\det U)/\sqrt{\theta}$ 。对 t 用归纳法, 有整数 (x_2, \dots, x_t) 使

$$h(x_2, \dots, x_t) \leq \left(\frac{4}{3}\right)^{(t-2)/2} |\det U|^{2/(t-1)} \theta^{1/(t-1)}$$

而且对这些整数值我们可以选择 x_1 使得 $|x_1 + (\alpha_2 x_2 + \cdots + \alpha_t x_t)/\alpha_1| \leq \frac{1}{2}$, 即

$(\alpha_1 x_1 + \cdots + \alpha_t x_t)^2 \leq \frac{1}{4}\theta$ 。因此

$$\theta \leq f(x_1, \dots, x_t) \leq \frac{1}{4}\theta + \left(\frac{4}{3}\right)^{(t-2)/2} |\det U|^{2/(t-1)} \theta^{1/(t-1)}$$

因此立即得出所求的不等式。

[注:对于 $t=2$ 这个结果是最好的。对于一般的 t , Hermite 的定理意味着 $\mu_t \leq \pi^{t/2} (4/3)^{t(t-1)/4} / (t/2)!$ 。Minkowski 给出的一个基本定理(“每个 t 维的关于原点对称的且体积大于等于 2^t 的凸集包含一个非零整数点”)告诉我们 $\mu_t \leq 2^t$; 对于 $t \geq 9$ 这是比 Hermite 的定理更强的。已知还有更强的结果, 参考(41)。]

10. 由于 y_1 和 y_2 是互素的, 我们可以解 $u_1 y_2 - u_2 y_1 = m$; 而且对于所有 q , $(u_1 + q y_1) y_2 - (u_2 + q y_2) y_1 = m$, 所以通过选择一个适当的整数 q , 我们可以确定 $2 \mid u_1 y_1 + u_2 y_2 \leq y_1^2 + y_2^2$ 。现在 $y_2(u_1 + a u_2) \equiv y_2 u_1 - y_1 u_2 \equiv 0 \pmod{m}$, 而且 y_2 必定与 m 互素, 因此 $u_1 + a u_2 \equiv 0$ 。最后命 $|u_1 y_1 + u_2 y_2| = \alpha m$, $u_1^2 + u_2^2 = \beta m$, $y_1^2 + y_2^2 = \gamma m$; 我们有 $0 \leq \alpha \leq \frac{1}{2} \gamma$, 剩下的是要证明 $\alpha \leq \frac{1}{2} \beta$ 和 $\beta \gamma \geq 1$ 。恒等式 $(u_1 y_2 - u_2 y_1)^2 + (u_1 y_1 + u_2 y_2)^2 = (u_1^2 + u_2^2)(y_1^2 + y_2^2)$ 意味着 $1 + \alpha^2 = \beta \gamma$ 。如果 $\alpha > \frac{1}{2} \beta$, 我们有 $2\alpha\gamma > 1 + \alpha^2$, 即 $\gamma - \sqrt{\gamma^2 - 1} < \alpha \leq \frac{1}{2} \gamma$ 。但是 $\frac{1}{2} \gamma < \sqrt{\gamma^2 - 1}$ 意味着 $\gamma^2 > \frac{4}{3}$, 矛盾。

11. 由于 a 是奇数, $y_1 + y_2$ 必为偶。为避免 y_1 和 y_2 都是偶数的解, 命 $y_1 = x_1 + x_2$, $y_2 = x_1 - x_2$, 并对 $x_1 \perp x_2$ 和 x_1 为偶数解 $x_1^2 + x_2^2 = m/\sqrt{3} - \epsilon$; 对应的乘数 a 将是 $(x_2 - x_1)a \equiv x_2 + x_1 \pmod{2^e}$ 的解。不难证明 $a \equiv 1 \pmod{2^{k-1}}$ 当且仅当 $x_1 \equiv 0 \pmod{2^k}$, 所以当 $x_1 \bmod 4 = 2$ 时我们得到最好的效能。问题归结为求 $x_1^2 + x_2^2 = N$ 的互素的解, 其中 N 是形如 $4k+1$ 的很大整数。通过在高斯整数上分解 N , 我们可以看到, 当且仅当 N 的每个素因子(在通常的整数上)有 $4k+1$ 的形式时这些解存在。

根据费马的著名定理, 形如 $4k+1$ 的每个素数 p 可以以惟一的方式(除 u 和 v 的符号外)写成为 $p = u^2 + v^2 = (u + iv)(u - iv)$, v 为偶数。通过解 $x^2 \equiv -1 \pmod{p}$, 可以有效地计算数 u 和 v , 然后通过高斯整数上的欧几里得算法, 计算 $u + iv = \gcd(x + i, p)$ 。[我们可以对几乎一半的整数 n 取 $x = n^{(p-1)/4} \bmod p$, 欧几里得算法的这一应用实质上等价于求使得 $u \pm xv \equiv 0 \pmod{p}$ 的最小非零 $u^2 + v^2$ 。当以通常的方式应用对于整数的欧几里得算法于 p 和 x 时, 也出现 u 和 v 的值; 参见 J. A. Serret 和 C. Hermite, *J. de Math. Pures et Appl.* **5** (1848), 12~15。] 如果 N 的素因子分解是 $p_1^{e_1} \cdots p_r^{e_r} = (u_1 + iv_1)^{e_1} (u_1 - iv_1)^{e_1} \cdots (u_r + iv_r)^{e_r} (u_r - iv_r)^{e_r}$, 通过命 $|x_2| + i|x_1| = (u_1 + iv_1)^{e_1} (u_2 + iv_2)^{e_2} \cdots (u_r + iv_r)^{e_r}$, 我们得到 $x_1^2 + x_2^2 = N$, $x_1 \perp x_2$, x_1 为偶数的 2^{r-1} 个不同的解, 而且所有这样的解都是以这种方式得到的。

注: 当 $m = 10^e$ 时, 可以使用一个类似的过程, 但由于我们必须继续试验, 直到求出满足 $x_1 \equiv 0 \pmod{10}$ 的解, 所以它有五倍的工作量。例如, 当 $m = 10^{10}$ 时, 我们有 $\lfloor m/\sqrt{3} \rfloor = 5773502691$, 且 $5773502689 = 53 \cdot 108934013 = (7 + 2i)(7 - 2i)(2203 +$

10202i)(2203 - 10202i)。在两个解 $|x_2| + i|x_1| = (7 + 2i)(2203 + 10202i)$ 或 $(7 + 2i)(2203 - 10202i)$ 中,前者给出 $|x_1| = 67008$ (不好),而后者给出 $|x_1| = 75820$, $|x_2| = 4983$ (它是可用的)。通过取 $x_1 = 75820, x_2 = -4983$, 得到表 1 的行 9。

表的第 14 行是这样得到的: $\lfloor 2^{32}/\sqrt{3} \rfloor = 2479700524$; 我们降到 $N = 2479700521$, 它等于 $37 \cdot 797 \cdot 84089$ 并有四个解 $N = 4364^2 + 49605^2 = 26364^2 + 42245^2 = 38640^2 + 31411^2 = 11960^2 + 48339^2$ 。对应的乘数是 2974037721, 2254986297, 4246248609 以及 956772177。我们还试验 $N - 4$, 但它不合法, 因为它可由 3 整除。另一方面, 素数 $N - 8 = 45088^2 + 21137^2$ 导致乘数 3825140801。类似地, 从 $N - 20, N - 44, N - 48$ 等等我们得到另外的乘数。行 14 上的乘数是由这个过程找到的头 16 个乘数中最好的; 它是从 $N - 68$ 得到的四个中的一个。

12. $U'_j \cdot U'_j = U_j \cdot U_j + 2 \sum_{i \neq j} q_i (U_i \cdot U_j) + \sum_{i \neq j} \sum_{k \neq j} q_i q_k (U_i \cdot U_k)$ 。关于 q_k 的偏导数是 (26) 左边的两倍。如果可以达到极小值, 则这些偏导数必须全部消失。

13. $u_{11} = 1, u_{21} = \text{无理数}, u_{12} = u_{22} = 0$ 。

14. 在执行了欧几里得算法三步之后, 我们求得 $v_2^2 = 5^2 + 5^2$, 则 S4 产生

$$U = \begin{pmatrix} -5 & 5 & 0 \\ -18 & -2 & 0 \\ 1 & -2 & 1 \end{pmatrix}, V = \begin{pmatrix} -2 & 18 & 38 \\ -5 & -5 & -5 \\ 0 & 0 & 100 \end{pmatrix}$$

通过变换 $(j, q_1, q_2, q_3) = (1, *, 0, 2), (2, -4, *, 1), (3, 0, 0, *), (1, *, 0, 0)$ 得到

$$U = \begin{pmatrix} -3 & 1 & 2 \\ -5 & -8 & -7 \\ 1 & -2 & 1 \end{pmatrix}, V = \begin{pmatrix} -22 & -2 & 18 \\ -5 & -5 & -5 \\ 9 & -31 & 29 \end{pmatrix}, Z = (0, 0, 1)$$

于是 $v_3 = \sqrt{6}$, 这我们由习题 3 已经知道。

15. (11) 中可达到的最大的 q , 减去可达到的最小者, 加 1, 是 $|u_1| + \cdots + |u_t| - \delta$, 其中, 对于某个 i 和 j , 若 $u_i u_j < 0$, 则 $\delta = 1$, 否则 $\delta = 0$ 。例如如果 $t = 5, u_1 > 0, u_2 > 0, u_3 > 0, u_4 = 0$ 和 $u_5 < 0$, 则可达到的最大值是 $q = u_1 + u_2 + u_3 - 1$ 且最小值是 $q = u_5 + 1 = -|u_5| + 1$ 。

[注意, 当 c 改变时, 超平面的个数不变, 因此同样的答案可应用于覆盖 L 而不是覆盖 L_0 的问题。但是, 所述公式对于覆盖 L_0 不总是精确的, 因为同单位超立方体相交的超平面不可能都包含 L_0 的点。在上边的例子中, 如果 $u_1 + u_2 + u_3 > m$, 我们在 L_0 中总不能达到值 $q = u_1 + u_2 + u_3 - 1$; 当且仅当在非负整数 (x_1, x_2, x_3, x_4) 中 $m - u_1 - u_2 - u_3 = x_1 u_1 + x_2 u_2 + x_3 u_3 + x_4 |u_5|$ 有一个解时它才是可达到的。当 $|u_1| + \cdots + |u_t|$ 为极小时, 所述极限总是可达到的, 这一点可能是真的, 但并不显然。]

16. 只须确定 (15) 的所有解有极小值 $|u_1| + \cdots + |u_t|$, 如果这些解的任何一个有相反符号的分量, 就减 1。

我们不用正定二次型, 而用颇为类似的函数 $f(x_1, \dots, x_t) = |x_1 U_1 + \dots + x_t U_t|$, 并定义 $|Y| = |y_1| + \dots + |y_t|$. 不等式 (21) 可以用 $|x_k| \leq f(y_1, \dots, y_t) \cdot (\max_{1 \leq j \leq t} |v_{kj}|)$ 代替。

于是可以得到一个可用的算法如下。用“置 $U \leftarrow (m), V \leftarrow (1), r \leftarrow 1, s \leftarrow m, t \leftarrow 1$ ”代替步骤 S1 到 S3。(这里 U 和 V 是 1×1 矩阵; 于是二维的情况将用一般的方法来处理。当然可以使用 $t=2$ 的一个特殊过程; 参见习题 5 的答案后面的参考文献。) 在步骤 S4 和 S7 中, 置 $s \leftarrow \min(s, |U_k|)$ 。在步骤 S7 中, 置 $z_k \leftarrow \lfloor \max_{1 \leq j \leq t} |v_{kj}| s/m \rfloor$ 。在步骤 S9 中, 置 $s \leftarrow \min(s, |Y| - \delta)$; 而在步骤 S10, 输出 $s = N_t$ 。否则如它表示的那样, 离开此算法, 因为它已经产生了适当的短向量。[见 *Math. Comp.* **29** (1975), 827~833.]

17. 在 S9 中当 $k > t$ 时, 如果 $Y \cdot Y \leq s$, 就输出 Y 和 $-Y$; 而且如果 $Y \cdot Y < s$, 就收回对于这个 t 以前的向量输出。[就作者编制表 1 的经验而言, 对于每个 v_t 恰有一个向量(及它的负), 除非当 $y_1 = 0$ 或 $y_t = 0$ 。]

18. a) 设 $x = m, y = (1 - m)/3, v_{ij} = y + x\delta_{ij}, u_{ij} = -y + \delta_{ij}$ 。于是对于 $j \neq k$, $V_i \cdot V_k = \frac{1}{3}(m^2 - 1), V_k \cdot V_k = \frac{2}{3}\left(m^2 + \frac{1}{2}\right), U_j \cdot U_j = \frac{1}{3}(m^2 + 2), z_k \approx \sqrt{\frac{2}{9}} m$ 。(这个例子对于 $a=1$ 满足 (28) 而且对所有 $m \equiv 1 \pmod{3}$ 有效。)

b) 交换步骤 S5 中 U 和 V 。对所有改变了值的 U_i , 置 $s \leftarrow \min(s, U_i \cdot U_i)$ 。例如, 当 $m = 64$, 对于 $j = 1$ 的这个变换可应用于 a) 的矩阵, 它把

$$V = \begin{bmatrix} 43 & -21 & -21 \\ -21 & 43 & -21 \\ -21 & -21 & 43 \end{bmatrix}, U = \begin{bmatrix} 22 & 21 & 21 \\ 21 & 22 & 21 \\ 21 & 21 & 22 \end{bmatrix}$$

约简为

$$V = \begin{bmatrix} 1 & 1 & 1 \\ -21 & 43 & -21 \\ -21 & -21 & 43 \end{bmatrix}, U = \begin{bmatrix} 22 & 21 & 21 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

[由于这个变换可以增加 V_j 的长度, 包含这两个变换的算法必须小心避免无穷循环, 也见习题 23。]

19. 否。因为所有对角线外元素非负和所有对角线元素为 1 的非单位矩阵的积不可能是单位矩阵。

[然而, 如果在 $-2V_i \cdot V_j = V_j \cdot V_j$ 时接着执行有 $q = -1$ 的变换, 循环将是可能的; 如果允许非缩短的变换, 舍入规则必须是对符号非对称的。]

20. 当 $a \bmod 8 = 5$ 时, 对于周期中 x 的点 $2^{-e}(x, s(x), \dots, s^{[t-1]}(x))$ 和对于 $0 \leq y < 2^{e-2}$ 的点 $2^{2^{-e}}(y, \sigma(y), \dots, \sigma^{t-1}(y))$ 加上 $(X_0 \bmod 4)/2^e$ 相同, 其中 $\sigma(y) =$

$(ay + \lfloor a/4 \rfloor) \bmod 2^{e-2}$ 。所以在这种情况下,我们可以以 $m = 2^{e-2}$ 使用算法 S。

当 $a \bmod 8 = 3$ 时,覆盖点 $2^{-e}(x, s(x), \dots, s^{(t-1)}(x)) \bmod 1$ 的平行超平面之间的极小距离和覆盖点 $2^{-e}(x, -s(x), \dots, (-1)^{t-1}s^{(t-1)}(x))$ 的极大距离相同,因为坐标的负值不改变距离。后面的点是 $2^{-e}(y, \sigma(y), \dots, \sigma^{t-1}(y))$, 其中 $\sigma(y) = (-ay - \lfloor a/4 \rfloor) \bmod 2^{e-2}$, 加上一个常数偏离。再次对于 $m = 2^{e-2}$ 应用算法 S, 把 a 变成 $m - a$ 对结果没有影响。

21. $X_{4n+4} \equiv X_{4n} \pmod{4}$, 所以现在命 $V_1 = (4, 4a^2, 4a^3)/m$, $V_2 = (0, 1, 0)$, $V_3 = (0, 0, 1)$ 定义对应的格 L_0 是适当的。

24. 设 $m = p$, 可以给出类似于正文的分析。例如, 当 $t = 4$ 时我们有 $X_{n+3} = ((a^2 + b)X_{n+1} + abX_n) \bmod m$, 而且我们要极小化 $u_1^2 + u_2^2 + u_3^2 + u_4^2 \neq 0$ 使得 $u_1 + bu_3 + abu_4 \equiv u_2 + au_3 + (a^2 + b)u_4 \equiv 0 \pmod{m}$ 。

用置

$$U \leftarrow \begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix}, V \leftarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, R \leftarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, s \leftarrow m^2, t \leftarrow 2$$

和输出 $v_2 = m$ 的操作代替步骤 S1 到 S3, 用以下的 S4' 代替步骤 S4:

S4'. [推进 t] 如果 $t = T$, 则算法结束。否则置 $t \leftarrow t + 1$ 和 $R \leftarrow R \begin{pmatrix} 0 & b \\ 1 & a \end{pmatrix} \bmod m$ 。置 U_t 成为 t 个元素的新行 $(-r_{12}, -r_{22}, 0, \dots, 0, 1)$, 而且对于 $1 \leq i < t$ 置 $u_{it} \leftarrow 0$ 。置 V_t 成为新行 $(0, \dots, 0, m)$ 。对于 $1 \leq i < t$, 置 $q \leftarrow \text{round}((v_{i1}r_{12} + v_{i2}r_{22})/m)$, $v_{it} \leftarrow v_{i1}r_{12} + v_{i2}r_{22} - qm$ 和 $U_t \leftarrow U_t + qU_i$ 。最后置 $s \leftarrow \min(s, U_t \cdot U_t)$, $k \leftarrow t, j \leftarrow 1$ 。

[一个类似的推广可应用于满足线性递推式 3.2.2-(8) 的长度为 $p^k - 1$ 的所有序列。另外的数值例子已经由 A. Grube, *Zeitschrift für angewandte Math. und Mechanik* **53** (1973), T223~T225; L'Ecuyer, Blouin 以及 Couture, *ACM Trans. Modeling and Comp. Simul.* **3** (1993), 87~98 给出。]

25. 给定的和至多是量 $\sum_{0 \leq k \leq m/2d} r(dk) = 1 + \frac{1}{d}f(m/d)$ 的两倍, 其中

$$f(m) = \frac{1}{m} \sum_{1 \leq k \leq m/2} \csc(\pi k/m) = \frac{1}{m} \int_1^{m/2} \csc(\pi x/m) dx + O\left(\frac{1}{m}\right) = \frac{1}{\pi} \ln \tan\left(\frac{\pi}{2m}x\right) \Big|_1^{m/2} + O\left(\frac{1}{m}\right)$$

[当 $d=1$ 时, 我们有 $\sum_{0 \leq k \leq m} r(k) = (2/\pi) \ln m + 1 + (2/\pi) \ln(2e/\pi) + O(1/m)$ 。]

26. 当 $m=1$ 时, 我们不能用(52), 因为 k 将是 0。如果 $\gcd(q, m) = d$, 以 m/d 代替 m 即可做同样的推导。假定我们有 $m = p_1^{e_1} \cdots p_r^{e_r}$ 和 $\gcd(a-1, m) = p_1^{f_1} \cdots p_r^{f_r}$ 及 $d = p_1^{d_1} \cdots p_r^{d_r}$ 。如果以 m/d 代替 m , 则 s 用 $p_1^{\max(0, e_1 - f_1 - d_1)} \cdots p_r^{\max(0, e_r - f_r - d_r)}$ 代替。

27. 使用下列函数是方便的: 如果 $x=0$, $\rho(x)=1$, 如果 $0 < x \leq m/2$, $\rho(x)=x$, 如果 $m/2 < x < m$, $\rho(x)=m-x$; 如果 $0 \leq x \leq m/2$, $\text{trunc}(x)=\lfloor x/2 \rfloor$, 如果 $m/2 < x < m$, $\text{trunc}(x)=m-\lfloor (m-x)/2 \rfloor$; 如果 $x=0$, $L(x)=0$, 如果 $0 < x \leq m/2$, $L(x)=\lfloor \lg x \rfloor + 1$, 如果 $m/2 < x < m$, $L(x)=-(\lfloor \lg(m-x) \rfloor + 1)$; 且 $l(x)=\max(1, 2^{l(x)-1})$ 。注意对于 $0 < x < m$, $l(L(x)) \leq \rho(x) < 2l(L(x))$ 且 $2\rho(x) \leq 1/r(x) = m \sin(\pi x/m) < \pi \rho(x)$ 。

如果向量 (u_1, \dots, u_t) 非零且满足 (15), 就说它是坏的, 并设 ρ_{\min} 是在所有坏的 (u_1, \dots, u_t) 上 $\rho(u_1) \cdots \rho(u_t)$ 的极小值。称向量 (u_1, \dots, u_t) 是在类 $(L(u_1), \dots, L(u_t))$ 中。因此至多有 $(2\lg m + 1)^t$ 个类, 而且类 (L_1, \dots, L_t) 至多含 $l(L_1) \cdots l(L_t)$ 个向量。我们的证明是基于说明在每个固定的类中的坏向量仅对 $\sum r(u_1, \dots, u_t)$ 贡献至多 $2/\rho_{\min}$; 这就证明了所要求的上限, 因为 $1/\rho_{\min} \leq \pi^t r_{\max}$ 。

命 $\mu = \lfloor \lg \rho_{\min} \rfloor$ 。一个向量上的 μ 重截断运算符定义为下列运算重复执行 μ 次: “命 j 是使得 $\rho(u_j) > 1$ 的极小值, 并以 $\text{trunc}(u_j)$ 代替 u_j ; 但如果对于所有 j , $\rho(u_j) = 1$, 就什么也不做。” (此运算实质上是抛掉与 (u_1, \dots, u_t) 有关的一位信息。) 如果 (u'_1, \dots, u'_t) 和 (u''_1, \dots, u''_t) 是有相同的 μ 重截断的同一类的两个向量, 我们说它们是相似的; 在这种情况下得出 $\rho(u'_1 - u''_1) \cdots \rho(u'_t - u''_t) < 2^\mu \leq \rho_{\min}$ 。例如当 m 很大和 $\mu=5$ 时, 形如 $((1x_2x_1)_2, 0, m - (1x_3)_2, (101x_5x_4)_2, (1101)_2)$ 的任何两个向量是相似的; μ 重截断运算符逐个地删除 x_1, x_2, x_3, x_4, x_5 。由于两个坏向量的差满足 (15), 因此两个不相等的坏向量不可能是相似的。因此类 (L_1, \dots, L_t) 至多可包含 $\max(1, l(L_1) \cdots l(L_t)/2^\mu)$ 个坏向量。如果类 (L_1, \dots, L_t) 恰含一个坏向量 (u_1, \dots, u_t) , 我们有 $r(u_1, \dots, u_t) \leq r_{\max} \leq 1/\rho_{\min}$; 如果它包含 $\leq l(L_1) \cdots l(L_t)/2^\mu$ 个坏向量, 它们每一个都有 $r(u_1, \dots, u_t) \leq 1/\rho(u_1) \cdots \rho(u_t) \leq 1/l(L_1) \cdots l(L_t)$, 因此我们有 $1/2^\mu < 2/\rho_{\min}$ 。

28. 设 $\zeta = e^{2\pi i/(m-1)}$ 并设 $S_{kl} = \sum_{0 \leq t < m-1} \omega^t \zeta^{kt}$ 。(51) 的类似是 $|S_{k0}| = \sqrt{m}$, 因此与 (53) 类似的式子是 $|N^{-1} \sum_{0 \leq t < N} \omega^t| = O((\sqrt{m} \log m)/N)$ 。类似的定理则表述为: $D_N^{(t)} = O\left(\frac{\sqrt{m}(\log m)^{t+1}}{N}\right) = O((\log m)^t r_{\max})$, $D_m^{(t)} = O((\log m)^t r_{\max})$ 。

事实上, $D_m^{(t)} \leq \frac{m-2}{m-1} \sum r(u_1, \dots, u_t)$ [对 (15) 的非零解求和] + $\frac{1}{m-1} \sum r(u_1, \dots, u_t)$ [对所有非零的 (u_1, \dots, u_t) 求和]。由习题 25 和 $d=1$, 后边的和是 $O(\log m)^t$, 而前一个和与 27 题同样处理。

我们现在考虑量 $R(a) = \sum r(u_1, \dots, u_t)$, 这里是对 (15) 的非零解求和。由于 m 是素数, 对每个 (u_1, \dots, u_t) , 至多有 $t-1$ 个 a 的值可以使它是 (15) 的一个解, 因此 $\sum_{0 < a < m} R(a) \leq (t-1) \sum r(u_1, \dots, u_t) = O(t(\log m)^t)$ 。由此得出, 取遍所有 $\varphi(m-1)$ 个原根的 $R(a)$ 的平均值是 $O(t(\log m)^t/\varphi(m-1))$ 。

注:一般来说, $1/\varphi(n) = O(\log \log n/n)$; 我们因此证明了对于所有素数 m 和所有 T , 存在一个模 m 下的原根 a 使得线性同余序列 $(1, a, 0, m)$ 有差异 $D_{m-1}^{(t)} = O(m^{-1} T (\log m)^7 \log \log m)$, $1 \leq t \leq T$ 。这个证明的方法不能推广到周期为 2^e 模 2^e 的线性同余生成程序的类似结果上, 因为例如向量 $(1, -3, 3, -1)$ 对几乎 $2^{2e/3}$ 个 a 的值是 (15) 的解。

29. 为了获得一个上限, 允许 $u = (u_1, \dots, u_t)$ 的非零分量是 $1 \leq |u_j| \leq \frac{1}{2}m$ 的任何实值。如果 k 个分量为非零, 利用习题 27 的答案的符号, 我们有 $r(u) \leq 1/(2^k \rho(u))$ 。而且如果 $u_1^2 + \dots + u_k^2$ 有一个给定的值 v^2 , 通过取 $u_1 = \dots = u_{k-1} = 1$ 和 $u_k^2 = v^2 - k + 1$ 我们极小化 $\rho(u)$ 。因此 $r(u) \leq 1/2^k \sqrt{v^2 - k + 1}$ 。但是由于 $v \geq k \geq 2$, 因此有 $2^k \sqrt{v^2 - k + 1} \geq \sqrt{8}v$ 。

30. 对 $1 \leq q < m$ 和 $0 \leq p < a$ 极小化 $q |aq - mp|$ 。利用习题 4.5.3-42 的记法, 对于 $0 \leq n \leq s$ 我们有 $aq_n - mp_n = (-1)^n K_{s-n-1}(a_{n+2}, \dots, a_s)$ 。在 $q_{n-1} \leq q < q_n$ 的范围内, 我们有 $|aq - mp| \geq |aq_{n-1} - mp_{n-1}|$; 因此 $q |aq - mp| \geq q_{n-1} \cdot |aq_{n-1} - mp_{n-1}|$, 且极小值是 $\min_{0 \leq n < q_n} |aq_n - mp_n| = \min_{0 \leq n < s} K_n(a_1, \dots, a_n) \cdot K_{s-n-1}(a_{n+2}, \dots, a_s)$ 。由习题 4.5.3-32, 我们有 $m = K_n(a_1, \dots, a_n) a_{n+1} K_{s-n-1}(a_{n+2}, \dots, a_s) + K_n(a_1, \dots, a_n) K_{s-n-2}(a_{n+3}, \dots, a_s) + K_{n-1}(a_1, \dots, a_{n-1}) K_{s-n-1}(a_{n+2}, \dots, a_s)$; 而且我们的问题实质上是使量 $m/K_n(a_1, \dots, a_n) K_{s-n-1}(a_{n+2}, \dots, a_s)$ 达到极大; 它位于 a_{n+1} 和 $a_{n+1} + 2$ 之间。

现在设 $A = \max(a_1, \dots, a_s)$ 。由于 $r(m-u) = r(u)$, 对于 $1 \leq u \leq \frac{1}{2}m$ 的某个 u , 我们可以假定 $r_{\max} = r(u)r(au \bmod m)$ 。置 $u' = \min(au \bmod m, (-au) \bmod m)$, 我们有 $r_{\max} = r(u)r(u')$ 。从上一段我们知道 $uu' \geq qq'$, 其中 $A/m \leq 1/qq' \leq (A+2)/m$ 。其次对于 $0 < u \leq \frac{1}{2}m$, $2u \leq r(u)^{-1} \leq \pi u$, 所以 $r_{\max} \leq 1/(4uu')$ 。因此我们有 $r_{\max} \leq (A+2)/(4m)$ 。(有一个类似的下限, 即 $r_{\max} > A/(\pi^2 m)$ 。)

31. 等价地, 猜测如下: 对于某个 n 和某个 $a_i \in \{1, 2, 3\}$, 所有很大的 m 都可写成 $m = K_n(a_1, \dots, a_n)$ 。对于固定的 n , 3^n 个数 $K_n(a_1, \dots, a_n)$ 有阶为 $(1+\sqrt{2})^n$ 的平均值, 而且它们的标准差的阶为 $(2.51527)^n$; 所以这个猜测几乎肯定是对的。S. K. Zaremba 于 1972 年猜测所有的 m 都可用 $a_i \leq 5$ 来表示; T. W. Cusick 在 *Mathematika* 24 (1977), 166~172 中关于这个问题做了某些推进。看来只有 $m = 54$ 和 $m = 150$ 的情况要求 $a_i = 5$, 而且要求 $a_i = 4$ 的最大的 m 是 2052, 2370, 5052 和 6234; 至少作者已经发现了对于小于 2000000 的所有其它整数的 $a_i \leq 3$ 的表示。当我们要求 $a_i \leq 2$ 时, $K_n(a_1, \dots, a_n)$ 的平均值是 $\frac{4}{5}2^n + \frac{1}{5}(-2)^n$, 同时标准差随

$(2.04033)^n$ 而增长。依作者的经验这样的数的密度(对于 $m \leq 2^{20}$, 它考虑每块有 2^{14} 个数的 2^6 块), 看来是在 .50 和 .65 之间变化。

[关于求具有小的部分商的乘数的一个计算方法, 请见 I. Borosh 及 H. Niederreiter, *BIT* 25 (1980), 193~208。他们已经求出对于 $25 \leq e \leq 35$ 的且有 $m = 2^e$ 的 2 有界的解。]

32. a) $U_n = Z_n/m_1 \equiv (m_2 - m_1) Y_n/m_1 m_2 \pmod{1}$, 以及 $(m_1 - m_2)/m_1 m_2 \approx 2^{-54}$ 。(因此我们通过分析 U_n 可以分析 Z_n 的高阶二进位。低阶二进位大概也是随机的, 但是这个论证对它们不适用。) b) 对于所有的 n 我们有 $U_n = W_n/m$ 。中国剩余定理告诉我们只须验证同余 $W_n \equiv X_n m_2 \pmod{m_1}$ 和 $W_n \equiv -Y_n m_1 \pmod{m_2}$, 因为 $m_1 \perp m_2$ 。[Pierre L'Ecuyer 和 Shu Tezuka, *Math. Comp.* 57 (1991), 735~746。]

3.4.1 小节

1. $a + (\beta - a)U$ 。

2. 设 $U = X/m$; 则 $\lfloor kU \rfloor = r \iff r \leq kX/m < r+1 \iff mr/k \leq X < m(r+1)/k \iff \lceil mr/k \rceil \leq X < \lceil m(r+1)/k \rceil$ 。精确的概率是由公式 $(1/m)(\lceil m(r+1)/k \rceil - \lceil mr/k \rceil) = 1/k + \epsilon$ 给出, 其中 $|\epsilon| < 1/m$ 。

3. 如果给定全字长的随机数, 则结果将如在习题 2 中那样, 至多偏离正确分布 $1/m$; 但是所有余量都给予最小的结果。因此, 如果 $k \approx m/3$, 则结果有大约 $2/3$ 的时间小于 $k/2$ 。如果 $U \geq k \lfloor m/k \rfloor$, 则通过拒绝 U 得到一个一致均匀的分布会是好得多的; 参见 D. E. Knuth, *The Stanford GraphBase* (New York: ACM Press, 1994), 221。

另一方面, 由 3.2.1.1 小节的结果, 如果使用一个线性同余序列, k 必须同模数 m 互素, 以免诸数有一个很短的周期。例如, 如果 $k=2$ 和 m 是偶数, 则这些数最好将是 0 和 1 的交替。这个方法在几乎每一种情况下都是比 (1) 慢的, 所以不予推荐。

然而不幸的是, 在许多高级语言中都不支持 (1) 中的 “himult (高位乘法)” 运算。参见习题 3.2.1.1-3。当高位乘法不可利用时, 以 m/k 来做除法可能是最好的。

4. $\max(X_1, X_2) \leq x$ 当且仅当 $X_1 \leq x$ 和 $X_2 \leq x$; $\min(X_1, X_2) \geq x$ 当且仅当 $X_1 \geq x$ 和 $X_2 \geq x$ 。两个独立的事件都发生的概率是各个概率的乘积。

5. 得到独立的一致离差 U_1 和 U_2 。置 $X \leftarrow U_2$ 。如果 $U_1 \geq p$, 则置 $X \leftarrow \max(X, U_3)$, 其中 U_3 是第三个一致离差。如果 $U_1 \geq p+q$, 则也置 $X \leftarrow \max(X, U_4)$, 其中 U_4 是第四个一致离差。这个方法显然可推广到任何多项式, 而其实甚至可推广到无穷幂级数(如同在算法 S 中作为例子说明的, 它使用极小化而不是用极大化)。

我们也可以这样做 (M. D. MacLaren 提出的): 如果 $U_1 < p$, 则置 $X \leftarrow U_1/p$; 否则如果 $U_1 < p+q$, 则置 $X \leftarrow \max((U_1 - p)/q, U_2)$; 否则置 $X \leftarrow \max((U_1 - p -$

$q)/r, U_2, U_3)$ 。用这个方法求一致离差所需的时间比其它方法少,缺点是有一些额外的算术操作而且数值上稍微不太稳定。

6. $F(x) = A_1/(A_1 + A_2)$, 其中 A_1 和 A_2 是图 A-4 中的面积; 所以

$$F(x) = \frac{\int_0^x \sqrt{1-y^2} dy}{\int_0^1 \sqrt{1-y^2} dy} = \frac{2}{\pi} \arcsin x + \frac{2}{\pi} x \sqrt{1-x^2}$$

每次遇到步骤 2 时, 在步骤 2 中终止的概率是 $p = \pi/4$, 所以执行步骤 2 的次数服从几何分布。由习题 17 知, 这个数的特征是 $(\min 1, \text{ave } 4/\pi, \max \infty, \text{dev } (4/\pi) \cdot \sqrt{1-\pi/4})$ 。

7. 如果 $k=1$, 则 $n_1 = n$ 且问题是平凡的。否则总有可能找到 $i \neq j$ 使得 $n_i \leq n \leq n_j$ 。用 n_i 个颜色 C_i 的立体和 $n - n_i$ 个颜色 C_j 的立体填 B_i , 然后 n_j 减少 $n - n_i$, 并消去颜色 C_i 。余下的问题与原来一样, 只是 k 的值减了 1; 由归纳法知它是可能的。

下列算法可以用来计算 P 和 Y 表格: 构造数偶 $(p_1, 1) \cdots (p_k, k)$ 的一个表并按头一个分量对它进行排序, 得到一个表 $(q_1, a_1) \cdots (q_k, a_k)$, 其中 $q_1 \leq \cdots \leq q_k$ 。置 $n \leftarrow k$; 重复下列操作直到 $n=0$: 置 $P[a_1-1] \leftarrow kq_1$ 和 $Y[a_1-1] \leftarrow x_{a_n}$ 。删去 (q_1, a_1) 和 (q_n, a_n) , 然后插入新的项 $(q_n - (1/k - q_1), a_n)$ 到表中的适当位置上, 并且 n 减 1。

(如果 $p_j < 1/k$, 则这个算法将不把 x_j 放进 Y 表格中; 算法 M 中隐含地用到了这一事实。这个算法总是剥夺最富裕的剩余元素并把它给予最穷的元素, 以使 (3) 中 $V < P_k$ 的概率达到极大。但是, 确定这个概率的绝对极小值却非常困难, 因为这样一个任务至少像“装箱问题”一样难; 参考 7.9 节。)

8. 对于 $0 \leq j < k$, 用 $(j + P_j)/k$ 代替 P_j 。

9. 考虑 $f''(x) = \sqrt{2/\pi}(x^2 - 1)e^{-x^2/2}$ 的符号。

10. 设对于 $1 \leq j \leq 16$, $S_j = (j-1)/5$ 和对于 $1 \leq j \leq 15$, $p_{j+15} = F(S_{j+1}) - F(S_j) - p_j$; 还设 $p_{31} = 1 - F(3)$ 和 $p_{32} = 0$ 。(等式 (15) 定义 p_1, \dots, p_{15} 。) 对于 $k=32$, 习题 7 的算法可用来计算 P_j 和 Y_j 。在这以后, 对于 $1 \leq j \leq 32$, 我们将有 $1 \leq Y_j \leq 15$ 。置 $P_0 \leftarrow P_{32}$ (它为 0), 且 $Y_0 \leftarrow Y_{32}$ 。然后对于 $0 \leq j < 32$, 置 $Z_j \leftarrow 1/(5 - 5P_j)$ 和 $Y_j \leftarrow \frac{1}{5} Y_j - Z_j$; 对于 $1 \leq j \leq 15$, 令 $Q_j \leftarrow 1/(5P_j)$ 。

设 $h = \frac{1}{5}$ 且对于 $S_j \leq x \leq S_j + h$ 设 $f_{j+15}(x) = \sqrt{2/\pi}(e^{-x^2/2} - e^{-j^2/50})/p_{j+15}$ 。然

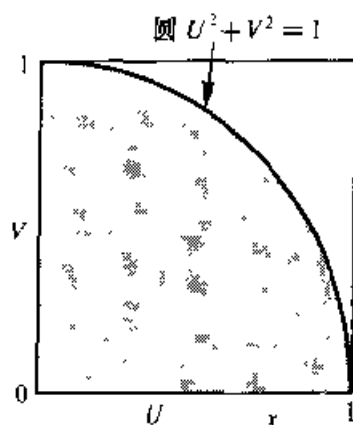


图 A-4 习题 6 的算法的“承认”区域

后对于 $1 \leq j \leq 5$, 设 $a_j = f_{j-15}(S_j)$, 对于 $6 \leq j \leq 15$ 令 $b_j = f_{j+15}(S_j)$; 以及对于 $1 \leq j \leq 5$, 令 $b_j = -hf'_{j+15}(S_j + h)$, 还有对于 $6 \leq j \leq 15$, 令 $a_j = f_{j+15}(x_j) + (x_j - S_j)b_j/h$, 其中 x_j 是方程 $f'_{j+15}(x_j) = -b_j/h$ 的根。最后对于 $1 \leq j \leq 15$, 置 $D_{j+15} \leftarrow a_j/b_j$, 和对于 $1 \leq j \leq 5$, 置 $E_{j+15} \leftarrow 25/j$, 对于 $6 \leq j \leq 15$, 置 $E_{j+15} \leftarrow 1/(e^{(2j-1)/50} - 1)$ 。

在计算表 1 时用到了下列的中间值: $(p_1, \dots, p_{31}) = (.156, .147, .133, .116, .097, .078, .060, .044, .032, .022, .014, .009, .005, .003, .002, .002, .005, .007, .009, .010, .009, .009, .008, .006, .005, .004, .002, .002, .001, .001, .003)$; $(x_6, \dots, x_{15}) = (1.115, 1.304, 1.502, 1.700, 1.899, 2.099, 2.298, 2.497, 2.697, 2.896)$; $(a_1, \dots, a_{15}) = (7.5, 9.1, 9.5, 9.8, 9.9, 10.0, 10.0, 10.1, 10.1, 10.1, 10.1, 10.2, 10.2, 10.2, 10.2)$; $(b_1, \dots, b_{15}) = (14.9, 11.7, 10.9, 10.4, 10.1, 10.1, 10.2, 10.3, 10.4, 10.5, 10.6, 10.7, 10.7, 10.8, 10.9)$ 。

11. 设对于 $t \geq 3$, $g(t) = e^{9/2}te^{-t^2/2}$ 。由于 $G(x) = \int_3^x g(t)dt = 1 - e^{-(x^2-9)/2}$,

通过置 $X \leftarrow G^{-1}(1-V) = \sqrt{9-2\ln V}$ 可以计算具有密度 g 的一个随机变量 X 。现在对于 $t \geq 3$, $e^{-t^2/2} \leq (t/3)e^{-t^2/2}$, 所以如果以概率 $f(X)/cg(X) = 3/X$ 接受 X , 我们就得到一个正确的拒绝的方法。

12. 对于 $x \geq 0$, 我们有 $f'(x) = xf(x) - 1 < 0$, 因为对于 $x > 0$, $f(x) = x^{-1} - e^{x^2/2} \int_x^\infty e^{-t^2/2} dt/t^2$ 。设 $x = a_{j-1}$ 和 $y^2 = x^2 + 2 \ln 2$; 则

$$\sqrt{2/\pi} \int_y^\infty e^{-t^2/2} dt = \frac{1}{2} \sqrt{2/\pi} e^{-x^2/2} f(y) < \frac{1}{2} \sqrt{2/\pi} e^{-x^2/2} f(x) = 2^{-1}$$

因此 $y > a_j$ 。

13. 取 $b_j = \mu_j$; 现在考虑对于每个 j , $\mu_j = 0$ 的问题。在矩阵表示下, 如果 $Y = AX$, 其中 $A = (a_{ij})$, 则我们需要 $AA^T = C = (c_{ij})$ 。(在另外的记法下, 如果 $Y_j = \sum a_{jk}X_k$, 则 Y_iY_j 的平均值是 $\sum a_{ik}a_{jk}$ 。)如果这个矩阵方程对 A 可解, 则当 A 是三角矩阵时, 它可以求解, 因为对于某个正交矩阵 U 和某个三角矩阵 B , $A = BU$, 且 $BB^T = C$ 。通过逐次地对 $a_{11}, a_{21}, a_{22}, a_{31}, a_{32}$ 等解方程 $a_{11}^2 = c_{11}, a_{11}a_{21} = c_{12}, a_{21}^2 + a_{22}^2 = c_{22}, a_{11}a_{31} = c_{13}, a_{21}a_{31} + a_{22}a_{32} = c_{23}, \dots$, 所求的三角形解可以得到。[注: 协方差矩阵必定是半正定的, 因为 $(\sum y_j Y_j)^2$ 的平均值是 $\sum c_{jj}y_j^2$, 它必定是非负的。而且当 C 是半正定时, 总有一个解, 因为 $C = U^{-1} \text{diag}(\lambda_1, \dots, \lambda_n) U$, 其中特征值 λ_j 是非负的, 而且 $U^{-1} \text{diag}(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n}) U$ 是一个解。]

14. 如果 $c > 0$, 则 $F(x/c)$; 如果 $c = 0$, 则为一个阶梯函数 $[x \geq 0]$; 而若 $c < 0$, 则为 $1 - F(x/c)$ 。

15. 分布 $\int_{-\infty}^\infty F_1(x-t)dF_2(t)$ 。密度 $\int_{-\infty}^\infty f_1(x-t)f_2(t)dt$ 。这叫做诸给定分布

的卷积。

16. 显然,如同所要求的那样,对于所有 t , $f(t) \leq cg(t)$ 。由于 $\int_0^{\infty} g(t)dt = 1$, 对于 $0 \leq t < 1$, 我们有 $g(t) = Ct^{a-1}$, 对于 $t \geq 1$ 它等于 Ce^{-t} , 其中 $C = ae/(a+e)$ 。具有密度 g 的一个随机变量容易作为两个分布的一个混合而得到, 即对于 $0 \leq x < 1$, $G_1(x) = x^a$ 和对于 $x \geq 1$, $G_2(x) = 1 - e^{1-x}$;

G1. [初始化] 置 $p \leftarrow e/(a+e)$ 。(这是 G_1 应当被使用的概率。)

G2. [生成 G 离差] 生成独立的一致离差 U 和 V , 其中 $V \neq 0$ 。如果 $U < p$, 则置 $X \leftarrow V^{1/a}$ 和 $q \leftarrow e^{-X}$; 否则置 $X \leftarrow 1 - \ln V$ 和 $q \leftarrow X^{a-1}$ 。(现在 X 有密度 g , 而且 $q = f(X)/cg(X)$ 。)

G3. [拒绝?] 生成一个新的一致离差 U 。如果 $U \geq q$, 则返回 G2。 ─

迭代的平均次数是 $c = (a+e)/(e\Gamma(a+1)) < 1.4$ 。

有好几种方法可把这个过程流水线化。首先,我们可以用比如说由算法 S 生成的均值为 1 的一个指数离差 Y 代替 V , 而后在两种情况下,我们置 $X \leftarrow e^{-Y/a}$ 或 $X \leftarrow 1 + Y$ 。而且,如果在头一种情况下我们置 $q \leftarrow pe^{-X}$, 在第二种情况下置 $q \leftarrow p + (1-p)X^{a-1}$, 我们可以使用原来的 U 而不是用步骤 G3 中新生成的那个 U 。最后,如果 $U < p/e$, 则我们可以立即接受 $V^{1/a}$, 而避免大约 30% 的对 q 的计算。

17. (a) 对 $x \geq 0$, $F(x) = 1 - (1-p)^{1+x}$ 。(b) $G(z) = pz/(1 - (1-p)z)$ 。(c) 均值 $1/p$, 标准差 $\sqrt{1-p}/p$ 。为了算出后者, 注意如果 $H(z) = q + (1-q)z$, 则 $H'(1) = 1-q$ 和 $H''(1) + H'(1) - (H'(1))^2 = q(1-q)$, 所以 $1/H(z)$ 的均值和方差分别为 $q-1$ 和 $q(q-1)$ 。(见 1.2.10 小节。)在这种情况下, $q = 1/p$; 在 $G(z)$ 的分子中额外的因子 z 使均值增加 1。

18. 置 $N \leftarrow N_1 + N_2 - 1$, 其中 N_1 和 N_2 独立地有概率为 p 的几何分布(考虑生成函数)。

19. 置 $N \leftarrow N_1 + \dots + N_t - t$, 其中 N_j 对于 p 有几何分布。(当进行一系列独立的试验时, 每个试验成功的概率为 p , 这是第 t 次成功之前失败的次数。)

对于 $t = p = \frac{1}{2}$, 而且一般地当这个分布的均值(即 $t(1-p)/p$)很小时, 我们可

以像在下列算法中那样对于 $n = 0, 1, 2, \dots$ 连续地计算概率 $p_n = \binom{t-1+n}{n} p^t (1-p)^n$:

B1. [初始化] 置 $N \leftarrow 0$, $q \leftarrow p^t$, $r \leftarrow q$, 并生成一个随机的一致离差 U 。(在此算法运行期间我们将有 $q = p_N$ 和 $r = p_0 + \dots + p_N$, 只要 $U < r$ 它就停止。)

B2. [迭代] 如果 $U \geq r$, 则置 $N \leftarrow N + 1$, $q \leftarrow q(1-p)(t-1+N)/N$, $r \leftarrow r + q$, 并重复这一步。否则返回 N 和结束。 ─

[R. Léger 已经提出对于任意大的实数值, 对于负的二项式分布的一项有趣的

技术:首先生成阶为 t 的一个随机 Γ 离差 X , 然后令 N 是均值为 $X(1-p)/p$ 的一个随机泊松离差。]

20. $R1 = 1 + (1 - A/R) \cdot R1$ 。当执行 R2 时, 算法以概率 I/R 终止; 当执行 R3 时, 它以概率 E/R 转向 R1。我们有

R1	R/A	R/A	R/A	R/A
R2	0	R/A	0	R/A
R3	0	0	R/A	$R/A - I/A$
R4	R/A	$R/A - I/A$	$R/A - E/A$	$R/A - I/A - E/A$

21. $R = \sqrt{8/e} \approx 1.71553$; $A = \sqrt{\pi/2} \approx 1.25331$ 。由于

$$\int u \sqrt{a - bu} du = (a - bu)^{3/2} \left(\frac{2}{5}(a - bu) - \frac{2}{3} \right) / b^2$$

我们有 $I = \int_0^{a/b} u \sqrt{a - bu} du = \frac{4}{15} a^{5/2} / b^2$, 其中 $a = 4(1 + \ln c)$, $b = 4c$; 当 $c = e^{1/4}$

时, I 有它的最大值 $\frac{5}{6} \sqrt{5/e} \approx 1.13020$ 。最后对于 E 需要下列的积分公式:

$$\begin{aligned} \int \sqrt{bu - au^2} du &= \frac{1}{8} b^2 a^{-3/2} \arcsin(2ua/b - 1) + \\ &\quad \frac{1}{4} ba^{-1} \sqrt{bu - au^2} (2ua/b - 1) \\ \int \sqrt{bu + au^2} du &= -\frac{1}{8} b^2 a^{-3/2} \ln(\sqrt{bu + au^2} + u\sqrt{a} + b/2\sqrt{a}) + \\ &\quad \frac{1}{4} ba^{-1} \sqrt{bu + au^2} (2ua/b + 1) \end{aligned}$$

其中 $a, b > 0$ 。设步骤 R3 中的检验是“ $X^2 \geq 4e^{x-1}/U - 4x$ ”; 则当 $u = r(x) = (e^x - \sqrt{e^{2x} - 2ex})/2ex$ 时外部区域碰到矩形的顶部。(恰好, $r(x)$ 在 $x = 1/2$ 处达到它的极大值, 这是它不可微的点!) 我们有 $E = \int_0^{r(x)} (\sqrt{2/e} - \sqrt{bu - au^2}) du$, 其中 $b = 4e^{x-1}$, $a = 4x$ 。在靠近 $x = -.35$ 处出现 E 的极大值, 在此处我们有 $E \approx .29410$ 。

22. (G. Marsaglia 给出的解) 考虑对于 $x > 0$ 时由 $G(x) = \int_{\mu}^{\infty} e^{-t} t^{x-1} dt / \Gamma(x)$

所定义的“连续泊松分布”; 如果 X 有这个分布, 则 $\lfloor X \rfloor$ 服从泊松分布, 因为 $G(x+1) - G(x) = e^{-\mu} \mu^x / x!$ 。如果 μ 很大, G 就是近似于正态的, 因此 $G^{[-1]}(F_{\mu}(x))$ 近似于线性的, 其中 $F_{\mu}(x)$ 是具有均值和方差 μ 的一个正态离差的分布函数; 即 $F_{\mu}(x) = F((x - \mu)/\sqrt{\mu})$, 其中 $F(x)$ 是正态分布函数(10)。设 $g(x)$ 是一个有效地可计算的函数, 使得对于 $-\infty < x < \infty$, $|G^{[-1]}(F_{\mu}(x)) - g(x)| < \epsilon$; 我们现在有效地生成泊松离差如下: 生成一正态离差 X , 并置 $Y \leftarrow g(\mu + \sqrt{\mu}X)$, $N \leftarrow \lfloor Y \rfloor$, $M \leftarrow \lfloor Y + \frac{1}{2} \rfloor$ 。则如果 $|Y - M| > \epsilon$, 就输出 N , 否则输出 M 。

$[G^{[-1]}(F(X)) < M]$ 。

这个方法也可应用于二项分布,且

$$G(x) = \int_p^1 u^{x-1}(1-u)^{a-x} du \frac{\Gamma(t+1)}{\Gamma(x)\Gamma(t+1-x)}$$

因为 $[G^{[-1]}(U)]$ 是具有参数 (t, p) 的二项分布, G 近似于正态的。

[还请参见 Ahrens 和 Dieter 在 *Computing* **25** (1980), 193~208 所提出的另一个方法。]

23. 是。第二个方法计算 $|\cos 2\theta|$, 其中 θ 是在 0 与 $\pi/2$ 之间一致分布的。(设 $U = r \cos \theta, V = r \sin \theta$ 。)

25. $\frac{21}{32} = (.10101)_2$ 。一般来说, 通过从左到右地用 1 代表 V , 0 代表 Λ , 最后添一个 1 , 来形成二进表示。这项技术[参考 K. D. Tocher, *J. Roy. Stat. Soc. B* **16** (1954), 49]可导致有效地生成有一个给定概率 p 的独立二进位序列, 而且它也可应用于几何分布及二项分布。

26. (a)真: $\sum_k \Pr(N_1 = k) \Pr(N_2 = n - k) = e^{-\mu_1 - \mu_2} (\mu_1 + \mu_2)^n / n!$ 。(b)假, 除非 $\mu_2 = 0$; 否则 $N_1 - N_2$ 可以是负的。

27. 设 p 的二进表示是 $(.b_1 b_2 b_3 \dots)_2$, 并按照下列规则进行计算:

B1. [初始化] 置 $m \leftarrow t, N \leftarrow 0, j \leftarrow 1$ 。(在此算法运行期间, m 表示模拟的一致离差的个数, 它们同 p 的关系仍然未知, 因为它们在其前导 $j-1$ 位同 p 匹配; 而 N 是已知的小于 p 的模拟离差个数。)

B2. [寻找下一列二进位] 生成具有二项分布 $\left(m, \frac{1}{2}\right)$ 的随机整数 M 。(现在 M 表示不能匹配 b_j 的未知离差的个数。)置 $m \leftarrow m - M$, 而且如果 $b_j = 1$, 置 $N \leftarrow N + M$ 。

B3. [完成没有?] 如果 $m = 0$ 或如果 p 的剩下的二进位 $(.b_{j+1} b_{j+2} \dots)_2$ 全为零, 则算法终止。否则, 置 $j \leftarrow j + 1$ 并返回步骤 B2。 ▮

[当对于无穷多的 $j, b_j = 1$ 时, 平均的迭代数 A_t 满足

$$A_0 = 0; \quad A_n = 1 + \frac{1}{2^n} \sum_k \binom{n}{k} A_k, \quad n \geq 1$$

设 $A(z) = \sum A_n z^n / n!$, 我们有 $A(z) = e^z - 1 + A\left(\frac{1}{2}z\right)e^{z/2}$ 。因此 $A(z)e^{-z} = 1 - e^{-z} + A\left(\frac{1}{2}z\right)e^{-z/2} = \sum_{k \geq 0} (1 - e^{-z/2^k}) = 1 - e^{-z} - \sum_{n \geq 1} (-z)^n / (n! (2^n - 1))$, 且

在习题 5.2.2-48 的记法下, $A_m = 1 + \sum_{k \geq 1} \binom{n}{k} \frac{(-1)^{k+1}}{2^k - 1} = 1 + \frac{V_{n+1}}{n+1} = \lg n + \frac{\gamma}{\ln 2} + \frac{1}{2} + f_0(n) + O(n^{-1})$ 。]

28. 在单位球上生成一随机点 (y_1, \dots, y_n) , 并设 $\rho = \sqrt{\sum a_k y_k^2}$ 。生成一独立的

一致离差 U , 而且如果 $\rho^{n+1}U < K\sqrt{\sum a_k^2 y_k^2}$, 则输出点 $(y_1/\rho, \dots, y_n/\rho)$; 否则开始跳出。这里如果 $na_n \geq a_1$ 则 $K^2 = \min\{(\sum a_k y_k^2)^{n+1}/(\sum a_k^2 y_k^2) | \sum y_k^2 = 1\} = a_n^{n+1}$, 否则为 $((n+1)/(a_1 + a_n))^{n+1}(a_1 a_n/n)^n$ 。

29. 设 $X_{n+1} = 1$, 然后对 $k = n, n-1, \dots, 1$ 置 $X_k \leftarrow X_{k+1} U_k^{1/k}$ 或 $X_k \leftarrow X_{k+1} \cdot e^{Y_k/k}$, 其中 U_k 是一致的或 Y_k 是指数的。[见 ACM Trans. Math. Software 6 (1980), 359~364。这一技术是由 David Seneschol 于 20 世纪 60 年代引进的; 请见 Amer. Statistician 26, 4 (1972 年 10 月), 56~57。生成 n 个一致的数并对它们排序的另一个方法大概更快, 用的是一个适当的排序方法。但如果只要求一些最大或最小的 X , 则这里所建议的方法是特别有价值的。注意 $(F^{[-1]}(X_1), \dots, F^{[-1]}(X_n))$ 将是被排序的有分布 F 的离差。]

30. 生成随机数 $Z_1 = -\mu^{-1} \ln U_1, Z_2 = Z_1 - \mu^{-1} \ln U_2, \dots$, 直到 $Z_{m+1} \geq 1$ 。对于 $1 \leq j \leq m$ 输出 $(X_j, Y_j) = f(Z_j)$, 其中 $f((.b_1 b_2 \dots b_{2r})_2) = ((.b_1 b_2 \dots b_r)_2, (.b_{r+1} b_{r+2} \dots b_{2r})_2)$ 。如果较低有效位是比更高有效位要非随机得多的, 则令 $f((.b_1 b_2 \dots b_{2r})_2) = ((.b_1 b_3 \dots b_{2r-1})_2, (.b_2 b_4 \dots b_{2r})_2)$ 是更安全的(但也更慢)。

31. a) 只须考虑 $k=2$ 的情况, 因为当 $X = X_1, \cos \theta = a_1$, 以及 $Y = (a_2 X_2 + \dots + a_k X_k)/\sin \theta$ 时, $a_1 X_1 + \dots + a_k X_k = X \cos \theta + Y \sin \theta$, 而且由替换 $u = s \cos \theta + t \sin \theta, v = -s \sin \theta + t \cos \theta$, 得到

$$\begin{aligned} \Pr(X \cos \theta + Y \sin \theta \leq x) &= \frac{1}{2\pi} \int_{s,t} e^{-s^2/2 - t^2/2} ds dt [s \cos \theta + t \sin \theta \leq x] = \\ &= \frac{1}{2\pi} \int_{u,v} e^{-u^2/2 - v^2/2} du dv [u \leq x] = (10) \end{aligned}$$

b) 有数 $\alpha > 1$ 和 $\beta > 1$ 使得 $(\alpha^{-24} + \alpha^{-55})/\sqrt{2} = 1$ 和 $\frac{3}{5}\beta^{-24} + \frac{4}{5}\beta^{-55} = 1$; 所以由线性递推的性质, 数 X_n 将随 n 而指数地增长。

如果我们通过比如说使用递推式 $X_n = X_{n-24} \cos \theta_n + X_{n-55} \sin \theta_n$, 其中 θ_n 是在 $[0, 2\pi)$ 中一致地选择的, 来破坏线性递推的模型, 我们大概将得到还不错的结果; 但这一方法将涉及多得多的计算。

c) 比方说, 通过 2048 个正态偏离 $X_0, \dots, X_{1023}, Y_0, \dots, Y_{1023}$ 开始, 在使用了它们中的 1/3 之后, 生成另外 2048 个如下: 在 $[0, 1024)$ 中一致地生成 a, b, c 和 d , 且 a 和 c 为奇数, 然后对于 $0 \leq j < 1024$, 置

$$\begin{aligned} X'_j &\leftarrow X_{(aj+b) \bmod 1024} \cos \theta + Y_{(cj+d) \bmod 1024} \sin \theta \\ Y'_j &\leftarrow -X_{(aj+b) \bmod 1024} \sin \theta + Y_{(cj+d) \bmod 1024} \cos \theta \end{aligned}$$

其中 $\cos \theta$ 和 $\sin \theta$ 如同在习题 23 中那样选择, 是 $(U^2 - V^2)/(U^2 + V^2)$ 和 $2UV/(U^2 + V^2)$ 的随机比。除非 $|\cos \theta| \geq \frac{1}{2}$ 和 $|\sin \theta| \geq \frac{1}{2}$, 否则我们可以拒绝 U 和 V 。2048 个新的离差现在代替旧的离差。注意对于每个新的离差只需要少量运算。

这个方法不像在 b) 中所考虑的序列那样发散, 因为平方和 $\sum (X_j^2 + Y_j^2) = \sum ((X'_j)^2 + (Y'_j)^2)$ 保留常数值 $S \approx 2048$, 除非有一点舍入误差。另一方面, S 的常数性实际上是这个方法的一个缺陷, 因为平方和实际上应有 2048 个自由度的 χ^2 分布。为了克服这个问题, 实质上分派给用户的正态离差不应是 X_j 而是 αX_j , 其中 $\alpha^2 = \frac{1}{2} (Y_{1023} + \sqrt{4095})^2 / S$ 是一个预先计算的比例因子。(量 $\frac{1}{2} (Y_{1023} + \sqrt{4095})^2$ 将是所要求的 χ^2 离差的一个合理近似。)

参考文献: C. S. Wallace [ACM Trans. on Math. Software **22** (1996), 119 ~ 127]; R. P. Brent [Lecture Notes in Comp. Sci. **1470** (1998), 1 ~ 20]。

32. a) 映射 $(X', Y') = f(X, Y)$ 是从集合 $\{x, y \geq 0\}$ 到它本身的一对一的对应, 使得 $x' + y' = x + y$ 和 $dx' dy' = dx dy$, 我们有

$$\frac{X'}{X' + Y'} = \left(\frac{X}{X + Y} - \lambda \right) \bmod 1, \frac{Y'}{X' + Y'} = \left(\frac{Y}{X + Y} + \lambda \right) \bmod 1$$

b) 这个映射是一个二对一的对应使得 $x' + y' = x + y$ 和 $dx' dy' = 2 dx dy$ 。

c) 只须考虑“ j 拍”转换: 对于一个固定的整数 j ,

$$X' = (\cdots r_{j+2} x_{j+1} x_j y_{j-1} y_j \ 2 y_j \ 3 \cdots)_2$$

$$Y' = (\cdots y_{j+2} y_{j+1} y_j x_{j-1} x_j \ 2 x_j \ 3 \cdots)_2$$

然后对于 $j = 0, 1, -1, 2, -2, \cdots$ 组成 j 拍, 注意当 $|j| \rightarrow \infty$ 时 X' 和 Y' 的联合概率分布收敛。每个 j 拍是一对一的, 且 $x' - y' = x + y$ 和 $dx' dy' = dx dy$ 。

33. 使用 U_1 作为另一个随机数生成程序(也许是带有不同的乘数的一个线性同余生成程序)的种子; 从中取 U_2, U_3, \cdots 。

3.4.2 小节

1. 从最后的 $N - t$ 个记录取出其中的 $n - m$ 个有 $\binom{N-t}{n-m}$ 种方式; 在选择了第 $t+1$ 项后, 从 $N - t - 1$ 个记录取其中的 $n - m - 1$ 个有 $\binom{N-t-1}{n-m-1}$ 种方式。

2. 当有待考察的记录数等于 $n - m$ 时, 步骤 S3 将不会转到步骤 S5。

3. 我们不应混淆“条件的”和“无条件的”概率。量 m 随机地依赖于在头 t 个元素当中发生的那些选择; 如果对于所有可能的选择(它们可能已在这些元素当中出现)取平均, 则我们将发现平均来说 $(n - m)/(N - t)$ 恰好是 n/N 。例如, 考虑第二个元素。如果在抽样中选择头一个元素(这出现的概率为 n/N), 则选择第二个元素的概率为 $(n - 1)/(N - 1)$ 。如果不选择第一个元素, 则选择第二个元素的概率是 $n/(N - 1)$ 。选择第二个元素的整个概率是 $(n/N)((n - 1)/(N - 1)) + (1 - n/N)(n/(N - 1)) = n/N$ 。

4. 由算法可知

$$p(m, t+1) = \left(1 - \frac{n-m}{N-t}\right)p(m, t) + \frac{n-(m-1)}{N-t}p(m-1, t)$$

通过对 t 用归纳法, 可以证明所求的公式。特别是 $p(n, N) = 1$ 。

5. 在习题 4 的记法下, 算法结束时 $t = k$ 的概率是 $q_k = p(n, k) - p(n, k-1) = \frac{\binom{k-1}{n-1}}{\binom{N}{n}}$ 。平均值为 $\sum_{k=0}^N k q_k = (N+1)n/(n+1)$ 。

6. 类似地, $\sum_{k=0}^N k(k+1)q_k = (N+2)(N+1)n/(n+2)$; 因此方差是 $(N+1) \cdot (N-n)n/(n+2)(n+1)^2$ 。

7. 假设选择是 $1 \leq x_1 < x_2 < \dots < x_n \leq N$ 。设 $x_0 = 0, x_{n+1} = N+1$ 。以概率 $p = \prod_{1 \leq i \leq n} p_i$ 得到此选择, 其中

$$p_i = \begin{cases} \frac{N - (t-1) - n + m}{N - (t-1)}, & \text{其中 } x_m < t < x_{m+1} \\ \frac{n-m}{N - (t-1)}, & \text{其中 } t = x_{m+1} \end{cases}$$

乘积 p 的分母为 $N!$, 对于不是 x 的那些 t , 分子包含项 $N-n, N-n-1, \dots, 1$, 对于是 x 的那些 t , 它包含项 $n, n-1, \dots, 1$ 。因此 $p = (N-n)!n!/N!$ 。例子: $n=3, N=8, (x_1, x_2, x_3) = (2, 3, 7); p = \frac{5}{8} \frac{3}{7} \frac{2}{6} \frac{4}{5} \frac{3}{4} \frac{2}{3} \frac{1}{2} \frac{1}{1} =$

8. a) $p(0, k) = \frac{\binom{N-k}{n}}{\binom{N}{n}}$ 为 $\binom{N}{n}$ 的抽样的 $\frac{\binom{N-n}{k}}{\binom{N}{k}}$ 省略头 k 个记录。

b) 置 $X \leftarrow k-1$, 其中 k 是使 $U \geq \Pr(X \geq k)$ 的极小值。因此, 以 $X \leftarrow 0, p \leftarrow N-n, q \leftarrow N, R \leftarrow p/q$ 开始, 而当 $U < R$ 时置 $X \leftarrow X+1, p \leftarrow p-1, q \leftarrow q-1, R \leftarrow R p/q$ 。(当 n/N 是, 比如说大于等于 $\frac{1}{5}$ 时这个方法好的。我们可以假设 $n/N \leq 1/2$; 否则最好选择 $N-n$ 个未抽样的项。)

c) $\Pr(\min(Y_N, \dots, Y_{N-n+1}) \geq k) = \prod_{j=0}^{n-1} \Pr(Y_{N-j} \geq k) = \prod_{j=0}^{n-1} (N-j-k)/(N-j)$ 。(如果, 比如说, $n \leq 5$, 这个方法好的。)

d) (参见习题 3.4.1-29。)值 $X \leftarrow \lfloor N(1-U^{1/n}) \rfloor$ 需要以仅仅 $O(n/N)$ 的概率被拒绝。精确的细节在 CACM 27 (1984), 703~718 中被仔细地给出。而一个实用的实现出现在 ACM Trans. Math. Software 13 (1987), 58~67 上。(当比如说 $5 < n < N$ 时, 这个方法好的。)

在跳过 X 个记录并且选择下一个之后, 我们置 $n \leftarrow n-1, N \leftarrow N-X-1$, 并重复这个过程直到 $n=0$ 。一个类似的方法加快这个水库方法; 参见 ACM Trans. Math. Software 11 (1985), 37~57。

9. 水库得到 7 个记录: 1, 2, 3, 5, 9, 13, 16。最后的抽样由记录 2, 5, 16 组成。

10. 删去步骤 R6 和变量 m 。用一个记录表代替 I 表, 把它初始化成步骤 R1 中的头 n 个记录, 并用新记录代替步骤 R4 中的第 M 个表项。

11. 像在 1.2.10 小节中那样论证(那里考虑的是 $n=1$ 的特殊情况), 我们看到生成函数为

$$G(z) = z^n \left(\frac{1}{n+1} + \frac{n}{n+1}z \right) \left(\frac{2}{n+2} + \frac{n}{n+2}z \right) \cdots \left(\frac{N-n}{N} + \frac{n}{N}z \right)$$

均值是 $n + \sum_{n < t \leq N} (n/t) = n(1 + H_N - H_n)$; 方差是 $n(H_N - H_n) - n^2(H_N^{(2)} - H_n^{(2)})$ 。

12. (注意 $\pi^{-1} = (b_1 t) \cdots (b_3 3)(b_2 2)$, 所以我们探求一个算法, 它从 π 的表示进行到 π^{-1} 的表示。)对于 $1 \leq j \leq t$, 置 $b_j \leftarrow j$ 。然后对于 $j=2, 3, \dots, t$ (按这个顺序) 交换 $b_j \leftrightarrow b_{a_j}$ 。最后对于 $j=t, \dots, 3, 2$ (按这个顺序), 置 $b_{a_j} \leftarrow b_j$ 。(这一算法是以 (a, t) $\pi_1 = \pi_1(b_1 t)$ 为基础的。)

13. 重新把这副牌编号成为 $0, 1, \dots, 2n-2$, 我们发现, s 使牌号 x 成为 $(2x) \bmod (2n-1)$ 号, 同时 c 使牌 x 成为 $(x+1) \bmod (2n-1)$ 。我们有 $(c$ 后面是 $s) = cs = sc^2$ 。因此任何 c 和 s 的乘积都可被变换成 $s'c^k$ 的形式。而且 $2^{\varphi(2n-1)} \equiv 1 \bmod (2n-1)$; 由于 $s^{\varphi(2n-1)}$ 和 c^{2n-1} 是恒等排列, 至多可能有 $(2n-1)\varphi(2n-1)$ 个排列。(不同排列的精确数是 $(2n-1)k$, 其中 k 是模 $(2n-1)$ 下 2 的阶。因为如果 $s^k = c'$, 则 c' 使牌 0 不变, 所以 $s^k = c' = \text{单位}$ 。)关于进一步的细节, 见 SIAM Review 3 (1961), 293~297。

14. a) $\frac{9}{9}$ 。不管他把它移动到哪儿, 我们都可能推导出这一点, 除非他把它放到头三个或最后两个位置之一。b) $\frac{5}{9}$ 。三个切和洗将产生至多八个周期地递增的子序列的相互混合 $a_{x_j} a_{(x_j+1) \bmod n} \cdots a_{(x_j+1) \bmod n}$; 因此子序列 $\frac{6}{9} \frac{5}{9} \frac{4}{9}$ 是显然的。[好多魔术技巧都是以下列事实为基础的, 即三次切和洗是高度地非随机的; 请参见 Martin Gardner, *Mathematical Magic Show* (Knopf, 1997), 第 7 章。]

15. 对于 $t-n < j \leq t$, 置 $Y_j \leftarrow j$ 。然后对于 $j=t, t-1, \dots, t-n+1$ 做下列运算: 置 $k \leftarrow \lfloor jU \rfloor + 1$ 。如果 $k > t-n$, 则置 $X_j \leftarrow Y_k$ 和 $Y_k \leftarrow Y_j$ 。否则如果对于某个 $i > j$, $k = X_i$ (可以使用一个符号表算法), 则置 $X_j \leftarrow Y_i$ 和 $Y_i \leftarrow Y_j$; 否则置 $X_j \leftarrow k$ 。(思路是令 Y_{t-n+1}, \dots, Y_j 表示 X_{t-n+1}, \dots, X_j , 而且如果 $i > j$ 和 $X_i \leq t-n$, 在算法 P 的执行过程中, 也令 Y_i 表示 X_{X_i} 。证明 Dahl 算法的正确性是有趣的。一个基本发现是在步骤 P2 中, 对于 $1 \leq k \leq j$, $X_k \neq k$ 意味着 $X_k > j$ 。)

16. 我们可以假定, $n \leq \frac{1}{2}N$, 否则只要找出不在抽样中的 $N-n$ 个元素就够了。使用大小为 $2n$ 的一个散列表, 想法是生成 1 和 N 之间的随机数, 把它们存入表中并且放弃重复元素, 直到 n 个不同的数都已生成为止。所生成的随机数的平均个数, 根据习题 3.2.2-10, 是 $N/N + N/(N-1) + \cdots + N/(N-n+1) < 2n$, 而且处理每个数的平均时间是 $O(1)$ 。我们要以递增次序输出这些结果, 而且这可以进行如下: 使用具有线性探查的一个有序散列表(习题 6.4-66), 这一散列表看上去将像

是, 诸值已经按递增次序插入, 而且探查的平均总数将小于 $\frac{5}{2}n$ 。因此如果我们对于键 k 使用单调的散列地址例如 $\lfloor 2n(k-1)/N \rfloor$, 则通过对这个表作至多两遍扫描, 以排好序的次序输出键, 将是一件简单的事。[参见 CACM 29 (1986), 366~367。]

17. 归纳地证明, 在步骤 j 之前, 集合 S 是从 $\{1, \dots, j-1\}$ 的 $j-N-1+n$ 个整数的一个随机抽样。[CACM 30 (1987), 754~757。Floyd 的方法可以用来加速对习题 16 的求解。它实质上是习题 15 中的 Dahl 算法的对偶, 该算法对 j 的递减值进行运算; 参见习题 12。]

3.5 节

1. 一个 b 进制序列是(参考习题 2); 一个 $[0, 1)$ 序列否(因为诸元素仅仅取有限多个值)。

2. 它是 1 分布和 2 分布的, 但不是 3 分布的(二进制数 111 不出现)。

3. 重复习题 3.2.2-17 中的序列; 且周期长度为 27。

4. 如果 $\nu_1(n), \nu_2(n), \nu_3(n), \nu_4(n)$ 是对应于 4 个概率的计数, 则对所有 n 我们有 $\nu_1(n) + \nu_2(n) = \nu_3(n) + \nu_4(n)$, 所以所求的结果由把极限相加得出。

5. 这个序列以 $\frac{1}{3}, \frac{2}{3}, \frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}$, 等等开始。当 $n=1, 3, 7, 15, \dots$ 时, 我们有 $\nu(n)=1, 1, 5, 5, \dots$ 使得 $\nu(2^{2^k-1}-1) = \nu(2^{2^k}-1) = (2^{2^k}-1)/3$; 因此 $\nu(n)/n$ 在 $\frac{1}{3}$ 和近于 $\frac{2}{3}$ 之间振荡; 而且无极限存在。所以概率是不确定的。[然而, 4.2.4 小节的方法证明, 一个数值能有意义地赋予

$$\Pr\left(U_n < \frac{1}{2}\right) = \Pr(n+1 \text{ 的 } 4 \text{ 进表示的最高位数字为 } 1)$$

即 $\log_4 2 = \frac{1}{2}$ 。]

6. 由习题 5 和归纳法,

$$\Pr(\text{对某 } j, 1 \leq j \leq k, S_j(n)) = \sum_{j=1}^k \Pr(S_j(n))$$

当 $k \rightarrow \infty$ 时后者是以 1 为界的单调序列, 所以它收敛; 而且对所有 k 有

$$\Pr(S_j(n) \text{ 对某 } j \geq 1) \geq \sum_{j=1}^k \Pr(S_j(n))$$

对于等式的一个反例, 不难通过适当的排列使得对于某个 j , $S_j(n)$ 总是真的, 然而对所有 j , $\Pr(S_j(n)) = 0$ 。

7. 设 $p_i = \sum_{j \geq 1} \Pr(S_{ij}(n))$ 。上题的结果可推广到 $\Pr(\text{对某个 } j \geq 1 \text{ 的 } S_j(n)) \geq \sum_{j \geq 1} \Pr(S_j(n))$, 其中诸 $S_j(n)$ 为任意不相交的命题。所以我们有 $1 = \Pr(\text{对某 } i, j \geq 1 \text{ 的 } S_{ij}(n)) \geq \sum_{i \geq 1} \Pr(\text{对某 } j \geq 1 \text{ 的 } S_{ij}(n)) \geq \sum_{i \geq 1} p_i = 1$, 因此 $\Pr(\text{对某个 } j$

≥ 1 的 $S_{ij}(n) = p_i$ 。给定 $\epsilon > 0$, 设 I 充分大, 使得 $\sum_{i=1}^I p_i \geq 1 - \epsilon$ 。设

$$\phi_i(N) = (\text{对于某个 } j \geq 1, \text{使 } S_{ij}(n) \text{ 为真的 } n < N \text{ 的个数})/N$$

显然, $\sum_{i=1}^I \phi_i(N) \leq 1$, 而且对于所有充分大的 N , 我们有 $\sum_{i=2}^I \phi_i(N) \geq \sum_{i=2}^I p_i - \epsilon$; 因此 $\phi_1(N) \leq 1 - \phi_2(N) - \cdots - \phi_I(N) \leq 1 - p_2 - \cdots - p_I + \epsilon \leq 1 - (1 - \epsilon - p_1) + \epsilon = p_1 + 2\epsilon$ 。这证明 $\overline{\text{Pr}}(\text{对某 } j \geq 1 \text{ 的 } S_{1j}(n)) \leq p_1 + 2\epsilon$; 因此 $\text{Pr}(\text{对某 } j \geq 1 \text{ 的 } S_{1j}(n)) \geq p_1$, 而且对于 $i = 1$, 所求结果成立。由假设的对称性可知, 它对 i 的任何值也成立。

8. 把定义 E 中对于 $j, j+d, j+2d, \dots, m+j-d$ 的概率加在一起。

9. $\limsup_{n \rightarrow \infty} (a_n + b_n) \leq \limsup_{n \rightarrow \infty} a_n + \limsup_{n \rightarrow \infty} b_n$; 因此我们求得

$$\limsup_{n \rightarrow \infty} ((y_{1n} - \alpha)^2 + \cdots + (y_{mn} - \alpha)^2) \leq m\alpha^2 - 2m\alpha^2 + m\alpha^2 = 0$$

而这仅当每个 $(y_{jn} - \alpha)$ 趋于 0 时才能发生。

10. 在等式(22)中求和。

11. 如果 $\langle U_n \rangle$ 是 $(2, 2k-1)$ 分布的, 则 $\langle U_{2n} \rangle$ 是 k 分布的。

12. 以 $f(x_1, \dots, x_k) = [u \leq \max(x_1, \dots, x_k) < v]$, 应用定理 B。

13. 设

$$p_k = \text{Pr}(U_n \text{ 开始一个长度为 } k-1 \text{ 的间隔}) =$$

$$\text{Pr}(U_{n-1} \in [\alpha, \beta), U_n \in [\alpha, \beta), \dots, U_{n+k-2} \in [\alpha, \beta), U_{n+k-1} \in [\alpha, \beta)) = p^2(1-p)^{k-1}$$

剩下的工作是把它变换成 $f(n) - f(n-1) = k$ 的概率。设 $\nu_k(n) = (\text{使得 } f(j) - f(j-1) = k \text{ 的 } j \leq n \text{ 的个数})$; 设 $\mu_k(n) = (\text{使 } U_j \text{ 是一个长度为 } k-1 \text{ 的间隔的开始的 } j \leq n \text{ 的个数})$; 且设 $\mu(n)$ 类似地统计 $U_j \in [\alpha, \beta)$ 的 $1 \leq j \leq n$ 的个数。我们有 $\mu_k(f(n)) = \nu_k(n)$, $\mu(f(n)) = n$ 。当 $n \rightarrow \infty$ 时, 我们必定有 $f(n) \rightarrow \infty$, 因此

$$\nu_k(n)/n = (\mu_k(f(n))/f(n)) \cdot (f(n)/\mu(f(n))) \rightarrow p_k/p = p(1-p)^{k-1}$$

[我们仅仅利用了这个序列是 $(k+1)$ 分布的这一事实。]

14. 设

$$p_k = \text{Pr}(U_n \text{ 开始长度为 } k \text{ 的一个运行}) =$$

$$\text{Pr}(U_{n-1} > U_n < \cdots < U_{n+k-1} > U_{n+k}) =$$

$$\frac{1}{(k+2)!} \left(\binom{k+2}{1} \binom{k+1}{1} - \binom{k+2}{1} - \binom{k+2}{1} + 1 \right) =$$

$$\frac{k}{(k+1)!} - \frac{k+1}{(k+2)!}$$

(参考习题 3.3.2-13)。现在像上题那样把它转换成 $\text{Pr}(f(n) - f(n-1) = k)$ 。[我们仅仅假定, 这个序列是 $(k+2)$ 分布的。]

15. 对于 $s, t \geq 0$, 设

$$p_{st} = \text{Pr}(X_{n-2t-3} = X_{n-2t-2} \neq X_{n-2t-1} \neq \cdots \neq X_{n-1})$$

$$X_n = \cdots = X_{n+s} \neq X_{n+s+1}) = 2^{-s-2t-3}$$

对于 $t \geq 0$ 设 $q_t = \Pr(X_{n-2t-2} = X_{n-2t-1} \neq \cdots \neq X_{n-1}) = 2^{-2t-1}$ 。由习题 7,

$$\Pr(X_n \text{ 不是一个集券集合的开始}) = \sum_{t \geq 0} q_t = \frac{2}{3}$$

$$\Pr(X_n \text{ 是长度为 } s+2 \text{ 的集券集合的开始}) = \sum_{t \geq 0} p_{st} = \frac{1}{3} \cdot 2^{-s-1}$$

然后照习题 13 的办法做。

16. (R. P. Stanley 的解法) 每当子序列 $S = (b-1), (b-2), \dots, 1, 0, 0, 1, \dots, (b-2), (b-1)$ 出现时, 一个集券集合必须在 S 的右边结束, 因为某个集券集合在 S 的头半完成。像在习题 15 中那样, 我们现在来计算一个集券集合在位置 n 处开始的概率, 其办法是首先考虑 S 的最近一项出现在位置 $n-1, n-2$ 等处结束的概率, 并在此基础上进行计算。

18. 像在定理 A 的证明中那样来计算 $\underline{\Pr}$ 和 $\overline{\Pr}$ 。

19. (T. Herzog 给出的解) 是的。例如, 当序列 $\langle U_n \rangle$ 满足 R4 (或者甚至它较弱的形式) 时, 应用习题 33 到序列 $\langle U_{\lfloor n/2 \rfloor} \rangle$ 上。

20. a) 2 和 $\frac{1}{2}$ 。(当 n 增加时, 我们把 $l_n^{(1)}$ 折半。)

b) 每一个新的点把单个区间分成两个部分。令 ρ 等于 $\max_{k=0}^{n-1} ((n+k)l_{n+k}^{(1)})$ 。于是 $1 = \sum_{k=1}^n l_n^{(k)} \leq \sum_{k=0}^{n-1} l_{n+k}^{(1)} \leq \sum_{k=0}^{n-1} \rho/(n+k) = \rho \ln 2 + O(1/n)$ 。所以无穷多个 m 有 $ml_m^{(1)} \geq 1/\ln 2 + O(1/m)$ 。

c) 为验证此提示, 令 $l_{2n}^{(k)}$ 来自于端点为 U_m 和 $U_{m'}$ 的区间, 并置 $a_k = \max(m-n, m'-n, 1)$ 。于是 $\rho = \min_{m=n+1}^{2n} ml_m^{(m)}$ 意味着 $1 = \sum_{k=1}^{2n} l_{2n}^{(k)} \geq \sum_{k=1}^{2n} \rho/(n+a_k) \geq 2\rho \sum_{k=1}^n 1/(n+k)$; 因此 $2\rho \leq 1/(H_{2n} - H_n) = 1/\ln 2 + O(1/n)$ 。

d) 我们有 $(l_n^{(1)}, \dots, l_n^{(n)}) = \left(\lg \frac{n+1}{n}, \lg \frac{n+2}{n+1}, \dots, \lg \frac{2n}{2n-1} \right)$, 因为第 $n+1$ 个点总是把最大的区间断开成长度为 $\lg \frac{2n+1}{2n}$ 和 $\lg \frac{2n+2}{2n+1}$ 的区间 [Indagationes Math.

11 (1949), 14~17]。

21. a) 否! 我们有 $\overline{\Pr}\left(W_n < \frac{1}{2}\right) \geq \limsup_{n \rightarrow \infty} \nu(\lceil 2^{n-1/2} \rceil) / \lceil 2^{n-1/2} \rceil = 2 - \sqrt{2}$,

$\underline{\Pr}\left(W_n < \frac{1}{2}\right) \leq \liminf_{n \rightarrow \infty} \nu(2^n)/2^n = \sqrt{2} - 1$, 因为 $\nu(\lceil 2^{n-1/2} \rceil) = \nu(2^n) = \frac{1}{2} \sum_{k=0}^n (2^{k+1/2} - 2^k) + O(n)$ 。

b), c) 参见 Indagationes Math. 40 (1978), 527~541。

22. 如果这个序列是 k 分布的, 则由积分和定理 B 可知, 极限为 0。反之, 注意, 如果 $f(x_1, \dots, x_k)$ 有绝对收敛的傅里叶级数

$$f(x_1, \dots, x_k) = \sum_{-\infty < c_1, \dots, c_k < \infty} a(c_1, \dots, c_k) \exp(2\pi i(c_1 x_1 + \dots + c_k x_k))$$

我们有 $\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{0 \leq n < N} f(U_n, \dots, U_{n+k-1}) = a(0, \dots, 0) + \epsilon_r$, 其中

$$|\epsilon_r| \leq \sum_{\max\{|c_1|, \dots, |c_k|\} > r} |a(c_1, \dots, c_k)|$$

所以 ϵ_r 可以弄成任意小。因此这个极限等于

$$a(0, \dots, 0) = \int_0^1 \dots \int_0^1 f(x_1, \dots, x_k) dx_1 \dots dx_k$$

而且等式(8)对于所有充分光滑的函数 f 成立。现在只需证明(9)中的函数可以通过光滑函数逼近到任意需要的精度。

23. (a) 这直接由习题 22 得出。(b) 以一个类似的方式使用一个离散傅里叶变换; 参见 D. E. Knuth, AMM 75 (1968), 260~264。

24. (a) 令 c 是任何非零整数; 由习题 22, 我们必须证明

$$\frac{1}{N} \sum_{n=0}^{N-1} e^{2\pi i c U_n} \rightarrow 0, \text{ 当 } N \rightarrow \infty \text{ 时}$$

这是因为如果 K 是任何正整数, 我们有 $\sum_{k=0}^{K-1} \sum_{n=0}^{N-1} e^{2\pi i c U_{n+k}} = K \sum_{n=0}^{N-1} e^{2\pi i c U_n} + O(K^2)$ 。因此, 由柯西不等式

$$\begin{aligned} \frac{1}{N^2} \left| \sum_{n=0}^{N-1} e^{2\pi i c U_n} \right|^2 &= \frac{1}{K^2 N^2} \left| \sum_{n=0}^{N-1} \sum_{k=0}^{K-1} e^{2\pi i c U_{n+k}} \right|^2 + O\left(\frac{K}{N}\right) \leq \\ &\frac{1}{K^2 N} \sum_{n=0}^{N-1} \left| \sum_{k=0}^{K-1} e^{2\pi i c U_{n+k}} \right|^2 + O\left(\frac{K}{N}\right) = \\ &\frac{1}{K} + \frac{2}{K^2 N} \Re \left(\sum_{0 \leq j < k < K} \sum_{n=0}^{N-1} e^{2\pi i c (U_{n+k} - U_{n+j})} \right) + O\left(\frac{K}{N}\right) \rightarrow \frac{1}{K} \end{aligned}$$

(b) 当 $d=1$ 时, 习题 22 告诉我们 $\langle (\alpha_1 n + \alpha_0) \bmod 1 \rangle$ 是等分布的当且仅当 α_1 是无理数。当 $d > 1$ 时, 我们可以使用 (a) 和对 d 的归纳法。[Acta Math. 56 (1931), 373~456。(b) 中的结果以前已经由 H. Weyl 以更复杂的形式得到, Nachr. Gesellschaft der Wiss. Göttingen, Math.-Phys. Kl. (1914), 234~244。一个类似的论证证明多项式序列是等分布的, 如果系数 $\alpha_d, \dots, \alpha_1$ 至少有一个是无理数。]

25. 如果这个序列是等分布的, 则推论 S 中的分母趋于 $\frac{1}{12}$, 而分子趋于这个习题中的量。

26. 见 Math. Comp. 17 (1963), 50~54。[也考虑 A. G. Waterman 给出的下列例子: 设 $\langle U_n \rangle$ 是一个等分布的 $[0, 1]$ 序列, $\langle X_n \rangle$ 是一个 ∞ 分布的二进序列。根据 X_n 为 0 或 1 设 $V_n = U_{\lfloor \sqrt{n} \rfloor}$ 或 $1 - U_{\lfloor \sqrt{n} \rfloor}$, 则 $\langle V_n \rangle$ 是等分布和白的, 但 $\Pr(V_n = V_{n+1}) = \frac{1}{2}$ 。设 $W_n = (V_n - \epsilon_n) \bmod 1$, 其中 $\langle \epsilon_n \rangle$ 是单调地减小成 0 的任何序列; 则 $\langle W_n \rangle$ 是

等分布和白的, 而 $\Pr(W_n < W_{n+1}) = \frac{3}{4}$ 。

28. 设 $\langle U_n \rangle$ 是 ∞ 分布的, 并考虑序列 $\left\langle \frac{1}{2}(X_n + U_n) \right\rangle$ 。利用 $\langle U_n \rangle$ 是 (16, 3) 分布的这个事实可知, 这是 3 分布的。

29. 如果 $x = x_1 x_2 \cdots x_l$ 是任意二进制数, 则我们可以考虑 $X_p \cdots X_{p+r-1} = x$ 的次数 $\nu_x^E(n)$, 其中 $1 \leq p \leq n$, 而且 p 为偶数。类似地, 设 $\nu_r^O(n)$ 计算当 p 为奇数时的次数。令 $\nu_r^E(n) + \nu_r^O(n) = \nu_x(n)$ 。现在

$$\nu_0^E(n) = \sum \nu_{0^{*} \cdots 0^{*}}^E(n) \approx \sum \nu_{0^{*} \cdots 0^{*}}^O(n) \approx \sum \nu_{* \cdots 0^{*} \cdots *}(n) \approx \cdots \approx \sum \nu_{* \cdots * \cdots 0}(n)$$

在这些和式当中的诸 ν 有 $2k$ 个下标, 下标当中有 $2k-1$ 个是星号(表示它们在被求和——每个和是对 0 和 1 的所有 2^{2k-1} 个组合进行), 而且其中“ \approx ”表示近似等式(除了由于结束条件的至多为 $2k$ 的误差外), 因此我们求得

$$\frac{1}{n} 2k \nu_0^E(n) = \frac{1}{n} \left(\sum \nu_{* \cdots 0^{*} \cdots *} + \cdots + \sum \nu_{* \cdots * \cdots 0} \right) + \frac{1}{n} \sum_x (r(x) - s(x)) \nu_x^E(n) + O\left(\frac{1}{n}\right)$$

其中 $x = x_1 \cdots x_{2k}$ 在奇位置中包含 $r(x)$ 个 0, 并在偶位置中包含 $s(x)$ 个 0。由 $(2k)$ 分布可知, 带圆括号的量趋于 $k(2^{2k-1})/2^{2k} = k/2$ 。当 $r(x) > s(x)$ 时, 如果 $\nu_x^E(n) = \nu_x(n)$, 以及当 $r(x) < s(x)$ 时, 如果 $\nu_x^E(n) = 0$, 剩下的和显然是一个极大值, 所以右边的极大值为

$$\frac{k}{2} + \sum_{0 \leq r < s \leq k} (r - s) \binom{k}{r} \binom{k}{s} / 2^{2k} = \frac{k}{2} + k \binom{2k-1}{k} / 2^{2k}$$

现在 $\Pr(X_{2n} = 0) \leq \limsup_{n \rightarrow \infty} \nu_0^E(2n)/n$, 所以证明完成。注意

$$\begin{aligned} \sum_{r,s} \binom{n}{r} \binom{n}{s} \max(r, s) &= 2n2^{2n-2} + n \binom{2n-1}{n} \\ \sum_{r,s} \binom{n}{r} \binom{n}{s} \min(r, s) &= 2n2^{2n-2} - n \binom{2n-1}{n} \end{aligned}$$

30. 构造标号为 $(Ex_1 \cdots x_{2k-1})$ 和 $(Ox_1 \cdots x_{2k-1})$ 的 2^{2k} 个节点的一个有向图, 其中每个 x_i 为 0 或 1。设从 $(Ex_1 \cdots x_{2k-1})$ 到 $(Ox_2 \cdots x_{2k})$ 有 $1 + f(x_1, x_2, \cdots, x_{2k})$ 条有向边, 而从 $(Ox_1 \cdots x_{2k-1})$ 到 $(Ex_2 \cdots x_{2k})$ 有 $1 - f(x_1, x_2, \cdots, x_{2k})$ 条有向边, 其中 $f(x_1, x_2, \cdots, x_{2k}) = \text{sign}(x_1 - x_2 + x_3 - x_4 + \cdots - x_{2k})$ 。我们发现每个节点有相同条数的有向边从它引出和引向它; 例如, $(Ex_1 \cdots x_{2k-1})$ 有 $1 - f(0, x_1, \cdots, x_{2k-1}) + 1 - f(1, x_1, \cdots, x_{2k-1})$ 条有向边引向它, 以及有 $1 + f(x_1, \cdots, x_{2k-1}, 0) + 1 + f(x_1, \cdots, x_{2k-1}, 1)$ 条由它引出, 且 $f(x, x_1, \cdots, x_{2k-1}) = -f(x_1, \cdots, x_{2k-1}, x)$ 。去掉没有引向它或由它引出的通路的所有节点, 即如果 $f(0, x_1, \cdots, x_{2k-1}) = +1$ 则删去

$(Ex_1 \cdots x_{2k-1})$, 如果 $f(1, x_1, \cdots, x_{2k-1}) = -1$ 则删去 $(Ox_1 \cdots x_{2k-1})$ 。得到的有向图可看出是连通的, 因为从任何节点我们可以达到 $(E1010 \cdots 1)$ 和从这一节点我们可以到达任何要求的节点。由定理 2.3.4.2G, 有遍历每一条有向边的一个循环通路; 这个通路有长度 2^{2k+1} , 而且我们可以假定它在节点 $(E00 \cdots 0)$ 开始。构造具有 $X_1 = \cdots = X_{2k-1} = 0$ 的一个循环序列, 而且如果这个通路的第 n 条有向边是从 $(Ex_1 \cdots x_{2k-1})$ 到 $(Ox_2 \cdots x_{2k})$, 或者从 $(Ox_1 \cdots x_{2k-1})$ 到 $(Ex_2 \cdots x_{2k})$ 的, 则 $X_{n+2k-1} = x_{2k}$ 。例如, 对于 $k=2$ 的图如图 A-5 所示, 循环通路的有向边从 1 编号到 32, 循环序列是

(0000100011001010100110111011110)(00001 \cdots)

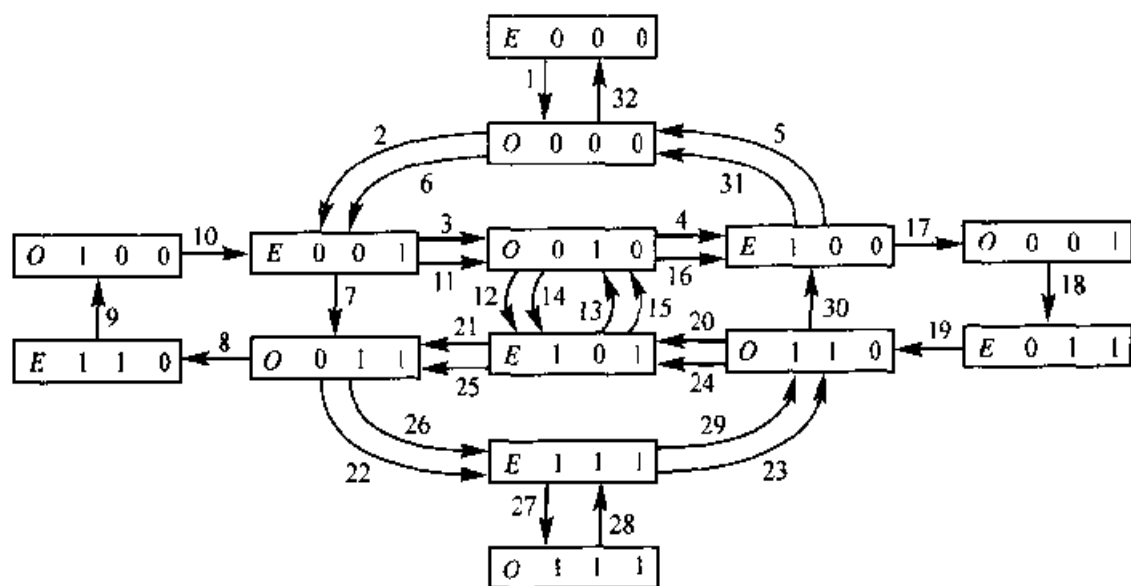


图 A-5 习题 30 中构造的有向图

注意在这个序列中 $\Pr(X_{2n} = 0) = \frac{11}{16}$ 。这个序列显然是 $(2k)$ 分布的, 因为在这个循环中每 $(2k)$ 元组 $x_1 x_2 \cdots x_{2k}$ 出现

$$1 + f(x_1, \cdots, x_{2k}) + 1 - f(x_1, \cdots, x_{2k}) = 2$$

次。 $\Pr(X_{2n} = 0)$ 有所求的值这一事实来自于在上题的证明中右边的极大值已由这个构造所实现。

31. 使用算法 W 并使用规则 \mathcal{R}_1 选择整个序列。[关于在 R5 序列这种类型的非随机行为的一个推广, 参见 Jean-Ville, *Étude Critique de la Notion de Collectif* (Paris:1939), 55~62。也许从这个观点看, R6 也太弱, 但是现在还不知道这样的反例。]

32. 如果 $\mathcal{R}, \mathcal{R}'$ 是可计算子序列规则, 则 $\mathcal{R}'' = \mathcal{R}\mathcal{R}'$ 由下列函数定义: $f_n''(x_0, \cdots, x_{n-1}) = 1$ 当且仅当 \mathcal{R} 定义 x_0, \cdots, x_{n-1} 的子序列 x_{r_1}, \cdots, x_{r_k} , 其中 $k \geq 0$ 和 $0 \leq r_1 < \cdots < r_k < n$ 和 $f_k'(x_{r_1}, \cdots, x_{r_k}) = 1$ 。

现在 $\langle X_n \rangle_{\mathcal{R}\mathcal{R}'}$ 是 $(\langle x_n \rangle_{\mathcal{R}})_{\mathcal{R}'}$, 由此立即得到结果。

33. 给定 $\epsilon > 0$, 求 N_0 使得 $N > N_0$ 意味着 $|\nu_r(N)/(N-p)| < \epsilon$ 和 $|\nu_s(N)/(N-p)| < \epsilon$ 两者都成立。然后求 N_1 使得 $N > N_1$, 意味着对于某个 $M > N_0$, t_N 是 r_M 或 s_M 。现在 $N > N_1$ 意味着

$$\left| \frac{\nu_t(N)}{N} - p \right| = \left| \frac{\nu_r(N_r) + \nu_s(N_s)}{N} - p \right| = \left| \frac{\nu_r(N_r) - pN_r + \nu_s(N_s) - pN_s}{N_r + N_s} \right| < \epsilon$$

34. 例如, 如果 t 的二进表示是 $(10^{b-2}10^{a_1}110^{a_2}1\cdots10^{a_k})_2$, 其中“0”代表 a 个连续的 0 的序列, 令规则 \mathcal{R}_t 接受 U_n 当且仅当 $\lfloor bU_{n-k} \rfloor = a_1, \dots, \lfloor bU_{n-1} \rfloor = a_k$ 。

35. 令 $a_0 = s_0$ 和 $a_{m+1} = \max\{s_k | 0 \leq k < 2^{a_m}\}$ 。构造一个这样的子序列规则, 即当 n 是在 $a_m \leq n < a_{m+1}$ 的范围内时, 这个规则选择 X_n 当且仅当对于某个 $k < 2^{a_m}$, 有 $n = s_k$ 。于是 $\lim_{m \rightarrow \infty} \nu(a_m)/a_m = \frac{1}{2}$ 。

36. 设 b 和 k 是任意的大于 1 的固定整数。设 $Y_n = \lfloor bU_n \rfloor$ 。由算法 \mathcal{S} 和 \mathcal{R} 确定的任意无穷子序列 $\langle Z_n \rangle = \langle Y_{s_n} \rangle_{\mathcal{R}}$ (像在定理 M 的证明中那样) 以直接的但从记号上看来不足取的方式对应于算法 \mathcal{S}' , \mathcal{R}' , 它们检查 $\langle X_n \rangle$ 的 $X_t, X_{t+1}, \dots, X_{t+s}$ 和 (或) 选择 $X_t, X_{t+1}, \dots, X_{t+\min(k-1, s)}$ 当且仅当 \mathcal{S} 和 \mathcal{R} 检查和 (或) 选择 Y_s , 其中 $U_s = (0, X_t, X_{t+1}, \dots, X_{t+s})_2$ 。算法 \mathcal{S}' 和 \mathcal{R}' 确定 $\langle X_n \rangle$ 的一个无穷 1 分布子序列, 而且事实上 (如在习题 32 中那样), 这个子序列是 ∞ 分布的, 所以它是 $(k, 1)$ 分布的。因此我们发现 $\Pr(Z_n = a)$ 和 $\bar{\Pr}(Z_n = a)$ 同 $1/b$ 相距小于 $1/2^k$ 。

[如果“R6”一直用“R4”或“R5”代替, 则这个习题的结果为真; 但如果使用“R1”则它为假, 因为 $X_{(2)}^{(n)}$ 可能恒等于 0。]

37. 对于 $n \geq 2$ 用 $\frac{1}{2}(U_n^2 + \delta_n)$ 代替 U_n^2 , 其中根据集合 $\{U_{(n-1)^2+1}, \dots, U_{n^2-1}\}$ 包含偶数个或奇数个小数 $\frac{1}{2}$ 的元素而令 $\delta_n = 0$ 或 1。[Advances in Math. 14 (1974), 333~334; 也见 Thomas N. Herzog 博士论文, Univ. of Maryland (1975)。]

39. 见 Acta Arithmetica 21 (1972), 45~50。c 的最好值未知。

40. 由于 F_k 仅依赖于 $B_1 \cdots B_k$, 我们有 $P(A_k^P, \mathcal{S}_N) = \frac{1}{2}$ 。设 $q(B_1 \cdots B_k) = \Pr(B_{k+1} = 1 | B_1 \cdots B_k)$, 其中概率是对于有 $B_1 \cdots B_k$ 作为头 k 个二进位的 S 的所有元素来取的。类似地, 令 $q_b(B_1 \cdots B_k) = \Pr(F_k = 1 \text{ 且 } B'_{k+1} = b | B_1 \cdots B_k)$ 。于是我们有 $\Pr(A_k^P = 1 | B_1 \cdots B_k) = \Pr((F_k + B_{k+1} + B'_{k+1}) \bmod 2 = 1 | B_1 \cdots B_k) = q \cdot \left(\frac{1}{2} - q_0 + q_1\right) + (1-q) \cdot \left(q_0 + \frac{1}{2} - q_1\right) = \frac{1}{2} - (q_0 + q_1) + 2(qq_1 + (1-q)q_0) = \frac{1}{2} - \Pr(F_k = 1 | B_1 \cdots B_k) + 2\Pr(F_k = 1 \text{ 和 } B'_{k+1} = B_{k+1} | B_1 \cdots B_k)$ 。因此 $\Pr(A_k^P = 1) = \sum_{B_1 \cdots B_k} \Pr(B_1 \cdots B_k) \Pr(A_k^P = 1 | B_1 \cdots B_k) = \frac{1}{2} - \Pr(F_k = 1) + \Pr(F_{k+1} = 1)$ 。[参见

Goldreich, Goldwasser 以及 Miceli 在 *JACM* **33** (1986), 792~807 中的定理 4。]

41. 从 $\{0, \dots, N-1\}$ 一致地选择 k 并且使用引理 P1 中的证明。于是 P1 的证明表明, A' 将以 $\sum_{k=0}^{N-1} \left(\frac{1}{2} - p_k + p_{k+1} \right) / N$ 的概率等于 1。

42. a) 设 $X = X_1 + \dots + X_n$ 。显然 $E(X) = n\mu$; 而且我们有 $E((X - n\mu)^2) = EX^2 - n^2\mu^2 = nEX_j^2 + 2 \sum_{1 \leq i < j \leq n} (EX_i)(EX_j) - n^2\mu^2 = nEX_j^2 - n\mu^2 = n\sigma^2$ 。而且 $E((X - n\mu)^2) = \sum_{x \geq 0} x \Pr((X - n\mu)^2 = x) \geq \sum_{x \geq t n \sigma^2} x \Pr((X - n\mu)^2 = x) \geq \sum_{x \geq t n \sigma^2} t n \sigma^2 \Pr((X - n\mu)^2 = x) = t n \sigma^2 \Pr((X - n\mu)^2 \geq t n \sigma^2)$ 。

b) 有一个位置 i , 其中 $c_i \neq c'_i$, 比如说, $c_i = 0$ 和 $c'_i = 1$ 。然后有另外一个位置 j , 其中 $c_j = 1$ 。对于在不同于 i 或 j 的 $k-2$ 行中 B 的任何固定设置, 我们有 $(cB, c'B) = (d, d')$ 当且仅当行 i 和 j 有特殊的值; 这以 $1/2^{2R}$ 的概率出现。

c) 在算法 L 的记号下, 取 $n = 2^k - 1$ 和 $X_i = (-1)^{G(cB + e_i)}$; 于是 $\mu = s$ 和 $\sigma^2 = 1 - s^2$ 。 $X = \sum_{c \neq 0} X_c$ 为负的概率至多是 $(X - n\mu)^2 \geq n^2 \mu^2$ 的概率。由 a) 这至多为 $\sigma^2 / (n\mu^2)$ 。

43. 对于固定的 M 的结论将是没有意义的, 因为明显地存在一个算法 (即, 知道因子的一个算法) 来对任何固定的 M 进行因子分解。这一理论适用于有短的运行时间的所有算法, 而不仅是有效地可发现的那些算法。

44. 如果对于一个随机表的每一个位的变化产生一个随机表, 则所有表是随机的 (或者全都不是), 如果我们不允许随机性的度, 则因此回答必须是“不总是”。

3.6 节

1.	RANDI	STJ	9F	存出口位置
		STA	8F	存 k 的值
		LDA	XRAND	$rA \leftarrow X$
		MUL	7F	$rAX \leftarrow aX$
		INCX	1009	$rX \leftarrow (aX + c) \bmod m$
		JOV	*+1	确保溢出开关断开
		SLAX	5	$rA \leftarrow (aX + c) \bmod m$
		STA	XRAND	存 X
		MUL	8F	$rA \leftarrow \lfloor kX/m \rfloor$
		INCA	1	加 1, 使得 $1 \leq Y \leq k$
9H	JMP	*		返回
XRAND	CON	1		X 的值; $X_0 = 1$
8H	CON	0		k 的临时存储
7H	CON	3141592621		乘数 a

2. 把一个随机数生成程序放到一个程序中使得程序的结果实际上不能为程序

员所预测。如果预先知道每个问题在机器上的运行特性,则很少会有人去写程序。然而如图灵曾经说过的,一台计算机的动作确实经常使程序员感到惊奇,特别是当他正在对他的程序进行调试时。

所以人们最好要注意。

7. 事实上,你只须 2 个二进位的值 $\lfloor X_n/2^{16} \rfloor \bmod 4$; 参见 D. E. Knuth, *IEEE Trans. IT-31* (1985), 49~52; J. Reeds, *Cryptologia* 1 (1977), 20~26; 3 (1979), 83~95, 发起了相关问题的研究。也见 L. Blum, M. Blum 以及 M. Shub, *SICOMP* 15 (1986), 364~383; J. Boyar, *J. Cryptology* 1 (1989), 177~184。在 *SICOMP* 17 (1988), 262~280 中, Frieze, Hastad, Kannan, Lagarias 和 Shamir 讨论了在像这样的问题中有一般技术。

8. 我们可以,比如说,通过做一百万次连续的调用生成 $X_{1000000}$, 并把它同正确值 $(a^{1000000} X_0 + (a^{1000000} - 1)c/(a - 1)) \bmod m$ 作比较, 它也可以表达为 $((a^{1000000} \cdot (X_0(a - 1) + c) - c) \bmod (a - 1)m)/(a - 1)$ 。后者可以通过一个独立的方法快速地进行计算(参见算法 4.6.3A)。例如, $48271^{1000000} \bmod 2147483647 = 1263606197$ 。大多数错误都将可以探测出来, 因为递推(1)不是自校正的。

9. a) X_0, X_1, \dots, X_{99} 的值不全为偶数, 多项式 $z^{100} + z^{37} + 1$ 是本原的(参见 3.2.2 小节); 因此有一个数 $h(s)$ 使得 $P_0(z) \equiv z^{h(s)} \pmod{2 \text{ 和 } z^{100} + z^{37} + 1}$ 。现在 $zP_{n+1}(z) = P_n(z) - X_n z^{37} - X_{n+63} + X_{n+63} z^{100} + X_{n+100} z^{37} \equiv P_n(z) + X_{n+63} (z^{100} + z^{37} + 1) \pmod{2}$, 所以由归纳法结果成立。

b) 在 *ran start* 中的运算“平方”和“乘以 z ”把 $p(z) = x_{99}z^{99} + \dots + x_1z + x_0$ 分别变成 $p(z)^2$ 和 $zp(z)$, 模 2 和 $z^{100} + z^{37} + 1$, 因为 $p(z)^2 \equiv p(z^2)$ 。(我们在这里仅考虑低阶二进位。其它二进位以趋向于保持和/或增强它们已有的无论什么样的无序的一种特殊方式来加以操作。)因此, 如果 $s = (1s_j \dots s_1 s_0)_2$, 我们有 $h(s) = (1s_0 s_1 \dots s_j 1)_2 \cdot 2^{69}$ 。

c) $z^{h(s)-n} \equiv z^{h(s')-n'} \pmod{2 \text{ 和 } z^{100} + z^{37} + 1}$ 意味着 $h(s) - n \equiv h(s') - n' \pmod{2^{100} - 1}$ 。由于 $2^{69} \leq h(s) < 2^{100} - 2^{69}$, 我们有 $|n - n'| \geq |h(s) - h(s')| \geq 2^{70}$ 。

[这个初始化的方法是由 R. P. Brent 的评述启发而成的, 见 *Proc. Australian Supercomputer Conf.* 5 (1972), 95~104, 尽管 Brent 的算法完全不同。一般地说, 如果这些延搁是 $k > l$, 如果 $0 < s < 2^l$, 和如果分开的参数 t 满足 $t + e \leq k$, 则这个证明的方法表明 $|n - n'| \geq 2^l - 1$, 而且仅当 $\{s, s'\} = \{0, 2^l - 1\}$ 时 $2^l - 1$ 才出现。]

10. 除了使用 *PARAMETER* 语句以增加可读性外, 下列程序属于美国国家标准局定义的简化的语言子集 *FORTRAN*。

```
SUBROUTINE RNARRY(AA, N)
  IMPLICIT INTEGER(A-Z)
  DIMENSION AA(*)
  PARAMETER (KK = 100)
```

```

    PARAMETER (LL = 37)
    PARAMETER (MM = 2 * * 30)
    COMMON /RSTATE/RANX(KK)
    SAVE /RSTATE/
    DO 1 J = 1, KK
1      AA(J) = RANX(J)
    DO 2 J = KK + 1, N
      AA(J) = AA(J - KK) - AA(J - LL)
      IF (AA(J) .LT. 0) AA(J) = AA(J) + MM
2    CONTINUE
    DO 3 J = 1, LL
      RANX(J) = AA(N + J - KK) - AA(N + J - LL)
      IF (RANX(J) .LT. 0) RANX(J) = RANX(J) + MM
3    CONTINUE
    DO 4 J = LL + 1, KK
      RANX(J) = AA(N + J - KK) - RANX(J - LL)
      IF (RANX(J) .LT. 0) RANX(J) = RANX(J) + MM
4    CONTINUE
    END
    SUBROUTINE RNSTRT(SEED)
    IMPLICIT INTEGER (A - Z)
    PARAMETER (KK = 100)
    PARAMETER (LL = 37)
    PARAMETER (MM = 2 * * 30)
    PARAMETER (TT = 70)
    PARAMETER (KKK = KK + KK - 1)
    DIMENSION X(KKK)
    COMMON/RSTATE/RANX(KK)
    SAVE /RSTATE/
    IF (SEED .LT. 0) THEN
      SSEED = MM - 1 - MOD(- 1 - SEED, MM)
    ELSE
      SSEED = MOD(SEED, MM)
    END IF
    SS = SSEED - MOD(SSEED, 2) + 2
    DO 1 J = 1, KK
      X(J) = SS
      SS = SS + SS

```

```

      IF (SS .GE. MM) SS = SS - MM + 2

1      CONTINUE
      DO 2 J = KK + 1, KKK
2          X(J) = 0
          X(2) = X(2) + 1
          SS = SSED
          T = TT - 1
10     DO 12 J = KK, 2, - 1
12         X(J + 1) = X(J)
          DO 13 J = KKK, KK - LL + 1, - 2
13         X(KKK - J + 2) = X(J) - MOD(X(J), 2)
          DO 14 J = KKK, KK + 1, - 1
              IF (MOD(X(J), 2) .EQ. 1) THEN
                  X(J - (KK - LL)) = X(J - (KK - LL)) - X(J)
                  IF (X(J - (KK - LL)) .LT. 0) X(J - (KK - LL)) = X(J - (KK
                      - LL)) + MM
                  X(J - KK) = X(J - KK) - X(J)
                  IF (X(J - KK) .LT. 0) X(J - KK) = X(J - KK) + MM
              END IF
14     CONTINUE
      IF (MOD(SS, 2) .EQ. 1) THEN
          DO 16 J = KK, 1, - 1
16         X(J + 1) = X(J)
          X(1) = X(KK + 1)
          IF (MOD(X(KK + 1), 2) .EQ. 1) THEN
              X(LL + 1) = X(LL + 1) - X(KK + 1)
              IF (X(LL + 1) .LT. 0) X(LL + 1) = X(LL + 1) + MM
          END IF
      END IF
      END IF
      IF (SS .NE. 0) THEN
          SS = SS/2
      ELSE
          T = T - 1
      END IF
      IF (T .GT. 0) GO TO 10
      DO 20 J = 1, LL

```



```

20      RANX(J + KK - LL) = X(J)
      DO 21 J = LL + 1, KK
21      RANX(J - LL) = X(J)
      END

```

11. 符合 ANSI/IEEE 标准 754 的对 64 个二进制操作数的浮点算术允许我们以对分数 U_n 的完满精度来计算 $U_n = (U_{n-100} - U_{n-37}) \bmod 1$, 这些 U_n 是 2^{-53} 的整数倍。然而, 下列程序使用的却是对 2^{-53} 的整数倍的加法递推 $U_n = (U_{n-100} + U_{n-37}) \bmod 1$, 因为与对一个中间结果的符号作条件转移相比, 流水线计算机可以更快地进行减一个整数部分的运算。习题 9 的理论同样很好地适用于这个序列。在 `ranf_start` 中主要的新思想是保持在 u 的分数的低有效位中的 ul 内的一个副本。类似于习题 10 的程序的一个 FORTRAN 翻译将和这个 C 程序一样生成完全相同的数。

```

#define KK 100                                /* the long lag */
#define LL 37                                /* the short lag */
#define mod_sum(x,y) (((x)+(y)) - (int)((x)+(y)))
                                                /* (x+y) mod 1.0 */
double ran_u[KK];                            /* the generator state */
void ranf_array(double aa[], int n) { /* aa gets n random fractions */
    register int i, j;
    for (j = 0; j < KK; j++) aa[j] = ran_u[j];
    for (; j < n; j++) aa[j] = mod_sum(aa[j - KK], aa[j - LL]);
    for (i = 0; i < LL; i++, j++) ran_u[i] = mod_sum(aa[j - KK], aa[j - LL]);
    for (; i < KK; i++, j++) ran_u[i] = mod_sum(aa[j - KK], ran_u[i - LL]);
}
#define TT 70                                /* guaranteed separation between streams */
#define is_odd(s) ((s)&1)
void ranf_start(long seed) { /* do this before using ranf_array */
    register int t, s, j;
    double u[KK + KK - 1], ul[KK + KK - 1];
    double ulp = (1.0 / (1L << 30)) / (1L << 22); /* 2 to the -52 */
    double ss = 2.0 * ulp * (seed + 2);
    for (j = 0; j < KK; j++) {
        u[j] = ss; ul[j] = 0.0;                /* bootstrap the buffer */
        ss += ss; if (ss >= 1.0) ss -= 1.0 - 2 * ulp;
                                                /* cyclic shift of 51 bits */
    }
}

```

```

for (;j<KK+KK-1;j++) u[j] = ul[j] = 0.0;
u[1] += ulp; ul[1] = ulp;          /* make u[1] (and only u[1]) "odd" */
s = seed;
t = TT-1; while (t) {
    for (j = KK-1; j>0; j--) ul[j+j] = ul[j], u[j+j] = u[j];
                                                    /* "square" */

    for (j = KK+KK-2; j>KK-LL; j-=2)
        ul[KK+KK-1-j] = 0.0, u[KK+KK-1-j] = u[j] - ul[j];
    for (j = KK+KK-2; j>=KK; j--) if (ul[j]) {
        ul[j-(KK-LL)] = ulp - ul[j-(KK-LL)],
        u[j-(KK-LL)] = mod_sum(u[j-(KK-LL)], u[j]);
        ul[j-KK] = ulp - ul[j-KK], u[j-KK] = mod_sum(u[j-KK], u[j]);
    }
    if (is_odd(s)) {                                /* "multiply by z" */
        for (j = KK; j>0; j--) ul[j] = ul[j-1], u[j] = u[j-1];
        ul[0] = ul[KK], u[0] = u[KK];
                                                    /* shift the buffer cyclically */
        if (ul[KK]) ul[LL] = ulp - ul[LL], u[LL] = mod_sum(u[LL], u[KK]);
    }
    if (s) s>>=1; else t--;
}
for (j = 0; j<LL; j++) ran_u[j+KK-LL] = u[j];
for (; j<KK; j++) ran_u[j-LL] = u[j];
}

int main() { register int m; double a[2009];
                                                    /* a rudimentary test */

    ranf_start(310952);
    for (m = 0; m<2009; m++) ranf_array(a, 1009);
    printf("%.20f\n", ran_u[0]);          /* 0.27452626307394156768 */
    ranf_start(310952);
    for (m = 0; m<1009; m++) ranf_array(a, 2009);
    printf("%.20f\n", ran_u[0]); return 0;
    /* 0.27452626307394156768 */
}

```

12. 像(1)这样的—一个简单的线性同余生成程序将失灵, 因为 m 将太小了。通过组合三个(而不是两个)这样的生成程序, 而且如同 P. L'Ecuyer 在 CACM 31 (1988), 747~748 上建议的, 使用乘数和模 (157, 32363), (146, 31727), (142,

31657), 有可能得到好的结果。然而, 最好的方法大概是使用 C 程序 *ran - array* 和 *ran - start*, 连同下列的改动以保持所有的数都在范围中: “long” 变成 “int”; “MM” 被定义为 “(10 < < 15)”; 而且变量 *ss* 的类型应是 unsigned int (无符号整数)。这生成 15 个二进位的整数, 它的所有二进位都是可用的。种子现在被限制在 [0, 32765] 的范围中。这个“基本检验程序”在给定的种子 12509 之下, 将打印 $X_{1009 \times 2009} = 9387$ 。

13. 用于带借位减法的一个程序将很类似于 *ran - array*, 但稍慢些, 因为要进行进位的维护。如同在习题 11 中那样, 浮点算术可以以完满精度被使用。通过以序列的第 $(-n)$ 个元素来初始化生成程序, 有可能来保证由不同的种子 s 所产生的序列的不相交性, 其中 $n = 2^{70s}$; 这要求计算 $b^n \bmod (b^k - b^l \pm 1)$ 。然而, 把一个 b 进制的数模 $b^k - b^l \pm 1$ 取平方要比在程序 *ran - start* 中类似的运算复杂得多, 而且对于在实用范围中的 k , 它大约要花费 $k^{1.6}$ 个运算而不是 $O(k)$ 。

当两个方法有大约相同的 k 值时, 在应用中它们大概生成相同质量的序列。两者之间惟一重要的差别是一个更好的理论保证和对于带借位减法的一个可证明的巨大周期; 对于延搁斐波那契生成程序的分析是不大完备的。经验表明, 我们不应仅仅由于这些理论的优点而减少在带借位减法中 k 的值。当所有这些都说了和做了之后, 从一个实用的观点看, 延搁的斐波那契生成程序似乎是可取的; 而带借位减法有价值主要是由于它为我们提供的对较简单方法的卓越特性的启示。

14. 我们有 $X_{n+200} \equiv (X_n + X_{n+126}) \pmod{2}$; 参见习题 3.2.2-32。因此当 $n \bmod 100 > 73$ 时, $Y_{n+100} \equiv Y_n + Y_{n+26}$ 。类似地 $X_{n+200} \equiv X_n + X_{n+26} + X_{n+89}$; 因此当 $n \bmod 100 < 11$ 时 $Y_{n+100} \equiv Y_n + Y_{n+26} + Y_{n+89}$ 。因此 Y_{n+100} 在所有情况的 $26\% + 11\%$ 中, 是 $\{Y_n, \dots, Y_{n+99}\}$ 中的仅两个或三个之和; 0 的数量优势将趋于使 $Y_{n+100} = 0$ 。

更精确地说, 考虑序列 $\langle u_1, u_2, \dots \rangle = \langle 126, 89, 152, 115, 78, \dots, 100, 63, 126, \dots \rangle$, 其中 $u_{n+1} = u_n - 37 + 100[u_n < 100]$ 。然后我们有

$$X_{n+200} = (X_n + X_{n+v_1} + \dots + X_{n+v_{k-2}} + X_{u_{k-1}}) \bmod 2$$

其中 $v_j = u_j + (-1)^{[u_j \geq 100]} 100$; 例如, $X_{n+200} \equiv X_n + X_{n+26} + X_{n+189} + X_{n+152} \equiv X_n + X_{n+26} + X_{n+189} + X_{n+52} - X_{n+115}$ 。如果下标全都小于 $n+t$ 和大于等于 $n+100+t$, 对于 $1 \leq t \leq 100$, 当 $n \bmod 100 \equiv 100-t$ 时, 我们得到对于 Y_{n-100} 的一个 k 项表达式。 $t=63$ 的情况是一个例外, 因为 $X_n + X_{n+1} + \dots + X_{n+62} + X_{n+163} + X_{n-164} + \dots + X_{n+199} \equiv 0$; 在这种情况下 Y_{n+100} 与 $\{Y_n, \dots, Y_{n+99}\}$ 无关。 $t=64$ 的情况是有趣的, 因为它给出 99 项关系 $Y_{n-100} \equiv Y_{n+1} + Y_{n+2} + \dots + Y_{n+99}$; 尽管有这么多的项但这趋向于为 0, 因为有 40 或更少的 1 的 100 元组的大多数都已经有了偶的奇偶性。

当有一个 k 项关系时, $Y_{n+100} = 1$ 的概率是

$$p_k = \sum_{l=0}^{40} \sum_{j=1}^k \binom{100-k}{l-j} \binom{k}{j} [j \text{ 奇}] / \sum_{l=0}^{40} \binom{100}{l}$$

当诸二进位被打印时,量 t 取值 $100, 99, \dots, 1, 100, 99, \dots, 1, \dots$; 所以我们求得预期被打印的 1 的个数为 $10^6(26p_2 + 11p_3 + 26p_4 + 11p_6 + 11p_9 + 4p_{12} + 4p_{20} + 3p_{28} + p_{47} + p_{74} + p_{99} + 1/2)/100 \approx 14043$ 。预期打印的十进数字的个数是 $10^6 \sum_{l=0}^{40} \binom{100}{l} / 2^{100} \approx 28444$, 所以预期的 0 个数 ≈ 14401 。

如果抛弃掉更多的元素则可检测的偏倚就可以去掉了。例如,如果仅使用 `ran_array(a, 300)` 的 100 个元素,可以证明概率是 $(26p_5 + 22p_6 + 19p_{10} + \dots)/100$; 对于 `ran_array(a, 400)` 它是很糟糕的 $(15p_3 + 37p_6 + 15p_9 + \dots)/100$, 因为 $X_{n+400} = X_n + X_{n+252}$ 。如同在正文中推荐的那样,对于 `ran_array(a, 1009)`, 我们有 $(17p_7 + 10p_{11} + 2p_{12} + \dots)/100$, 如果打印的阈值从 60 提高到比如说 75, 则它仅可由这样的实验加以探测。但那样一来,预期的输出数字仅大约为每百万个试验中的 0.28 个。

[本习题是基于 Y. Kurita, H. Leeb 和 M. Matsumoto 于 1997 年同作者通信时的思想给出的。]

15. 下列程序使得可能很快地通过表达式 `ran_arr_next()` 得到一个新的随机整数,一旦已经调用 `ran_start` 来使事情开始:

```
#define QUALITY 1009 /* recommended quality level for high-res use */
long ran_arr_buf[QUALITY];
long ran_arr_sentinel = -1;
long * ran_arr_ptr = &ran_arr_sentinel;
/* the next random number, or -1 */
#define ran_arr_next() (* ran_arr_ptr) = 0?
    * ran_arr_ptr++ : ran_arr_cycle()
long ran_arr_cycle()
{
    ran_array(ran_arr_buf, QUALITY);
    ran_arr_buf[100] = -1;
    ran_arr_ptr = ran_arr_buf + 1;
    return ran_arr_buf[0];
}
```

4.1 节

1. $(1010)_{-2}, (1011)_{-2}, (1000)_2, \dots, (11000)_{-2}, (11001)_{-2}, (11110)_{-2}$ 。
2. (a) $(110001)_2, -(11.001001001001\dots)_2, (11.00100100001111110110101\dots)_2$ 。
(b) $(11010011)_{-2}, (1101.001011001011\dots)_{-2}, (111.0110010001000000101\dots)_{-2}$ 。

(c) $(\bar{1}11\bar{1}1)_3, (\bar{1}0.0\bar{1}\bar{1}0110\bar{1}011\cdots)_3, (10.011\bar{1}111\bar{1}000\bar{1}011\bar{1}101\bar{1}111110\cdots)_3$ 。

(d) $-(9.4)_{1/10}, -(\cdots 7582417582413)_{1/10}, (\cdots 3462648323979853562951413)_{1/10}$ 。

3. $(1010113.2)_{210}$ 。

4. (a) 在 rA 和 rX 之间。(b) rX 中的余数的小数点在字节 3 和 4 之间; rA 中的商的小数点在寄存器最小有效位部分右边一个字节处。

5. 它已经从 $999\cdots 9 = 10^p - 1$ 减去, 而不是从 $1000\cdots 0 = 10^p$ 减去。

6. (a, c) $2^{p-1} - 1, -(2^{p-1} - 1)$; (b) $2^{p-1} - 1, -2^{p-1}$ 。

7. 一个负数 x 的 10 的补码表示, 可以通过考虑 $10^n + x$ (其中 n 要足够大以使这个数成为正的) 并以无限多个 9 对它向左边扩充。9 的补码表示可以通常的方式得到 (对于无尽的十进数这两个表示是相等的, 否则 9 的补码表示有 $\cdots(a)99999\cdots$ 的形式, 而 10 的补码表示有 $\cdots(a+1)0000\cdots$ 的形式)。如果我们把无限和 $N = 9 + 90 + 900 + 9000 + \cdots$ 的值当做 -1 , 则这些表示可以认为是明智的, 因为 $N - 10N = 9$ 。

还可看习题 31, 那里考虑了 p -adic 的数系。对于 p 进制表示为有穷的数, 它同这里考虑的 p 的补码记号一致, 但在 p -adic 数域和实数域之间没有简单的关系。

8. $\sum_j a_j b^j = \sum_j (a_{k_j+k-1} b^{k-1} + \cdots + a_{k_j}) b^{k_j}$ 。

9. A BAD ADOBE FACADE FADED。[注: 其它可能的“数的句子”将是 DO A DEED A DECADE; A CAD FED A BABE BEEF, COCOA, COFFEE; BOB FACED A DEAD DODO。]

10. 如果 $A_j = \begin{bmatrix} a_{k_{j+1}-1}, & a_{k_{j+1}-2}, & \cdots, & a_{k_j} \\ & b_{k_{j+1}-2}, & \cdots, & b_{k_j} \end{bmatrix}, B_j = b_{k_{j+1}-1} \cdots b_{k_j},$

则 $\begin{bmatrix} \cdots, a_3, a_2, a_1, a_0; a_{-1}, a_{-2}, \cdots \\ \cdots, b_3, b_2, b_1, b_0; b_{-1}, b_{-2}, \cdots \end{bmatrix} = \begin{bmatrix} \cdots, A_3 A_2, A_1, A_0; A_{-1}, A_{-2}, \cdots \\ \cdots, B_3, B_2, B_1, B_0; B_{-1}, B_{-2}, \cdots \end{bmatrix}$

其中 $\langle k_n \rangle$ 是任意的无限整数序列, 且 $k_{j+1} > k_j$ 。

11. (依赖于选择的是正号还是负号, 下列算法对于加法或减法有效。)

以置 $k \leftarrow a_{n+1} \leftarrow a_{n+2} \leftarrow b_{n+1} \leftarrow b_{n+2} \leftarrow 0$ 开始; 然后对于 $m = 0, 1, \cdots, n+2$ 进行如下操作: 置 $c_m \leftarrow a_m \pm b_m + k$; 然后如果 $c_m \geq 2$, 则置 $k \leftarrow -1$ 且 $c_m \leftarrow c_m - 2$; 否则如果 $c_m < 0$, 则置 $k \leftarrow 1$ 且 $c_m \leftarrow c_m + 2$; 否则 (即如果 $0 \leq c_m \leq 1$), 则置 $k \leftarrow 0$ 。

12. (a) 在负二进系统中从 $\pm(\cdots a_4 0 a_2 0 a_0)_{-2}$ 减去 $\pm(\cdots a_3 0 a_1 0)_{-2}$ (一个更巧妙的解见习题 7.1-18, 该解使用全字位运算)。(b) 在二进系统中从 $(\cdots b_4 0 b_2 0 b_0)_2$ 减去 $(\cdots b_3 0 b_1 0)_2$ 。

13. $(1.909090\cdots)_{10} = (0.090909\cdots)_{10} = \frac{1}{11}$ 。

14.

$$\begin{array}{rrrrr}
 & & 1 & 1 & 3 & 2 & 1 & [5-4i] \\
 & & \underline{1} & \underline{1} & 3 & 2 & 1 & [5-4i] \\
 & & 1 & 1 & 3 & 2 & 1 & \\
 & 1 & 1 & 2 & 0 & 2 & & \\
 & 1 & 2 & 1 & 2 & 3 & & \\
 & 1 & 1 & 3 & 2 & 1 & & \\
 1 & 1 & 3 & 2 & 1 & & & \\
 \hline
 0 & 1 & 0 & 3 & 1 & 1 & 2 & 0 & 1 & [9-40i]
 \end{array}$$

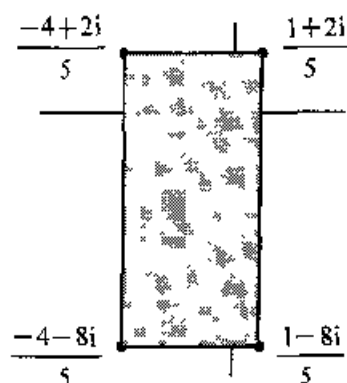


图 A-6 4 虚数的基本区域

15. $\left[-\frac{10}{11}, \frac{1}{11}\right]$, 这个矩形示于图 A-6 中。

16. 试图以非常简单的方式来完成是有吸引力的,例如,用规则 $2 = (1100)_{i-1}$ 来进位;但是如果(比如说)我们试图把 1 加到 $(11101)_{i-1} = -1$ 的话,这导致一个无尽的方法。

下列的解通过提供四种有关的算法(即加减 1 或 i)做这件事。如果 α 是 0 和 1 的串,令 α^P 是 0 和 1 的串,使得 $(\alpha^P)_{i-1} = (\alpha)_{i-1} + 1$;并令 $\alpha^{-P}, \alpha^Q, \alpha^{-Q}$ 类似地定义,且用 $-1, +i$ 和 $-i$ 分别代替 $+1$, 则

$$\begin{aligned} (\alpha 0)^P &= \alpha 1; & (\alpha x 1)^P &= \alpha^Q x 0 \\ (\alpha 0)^Q &= \alpha^P 1; & (\alpha 1)^Q &= \alpha^{-Q} 0 \\ (\alpha x 0)^{-P} &= \alpha^{-Q} x 1; & (\alpha 1)^{-P} &= \alpha 0 \\ (\alpha 0)^{-Q} &= \alpha^Q 1; & (\alpha 1)^{-Q} &= \alpha^{-P} 0 \end{aligned}$$

这里 x 代表 0 或 1, 如果需要的话, 在这些串的左边扩充 0。这个过程显然会终止的。因此形如 $a + bi$ 的每个数, 当 a 和 b 为整数时在 $i-1$ 系统中是可表示的。

17. 否(不管习题 28 如何);数 -1 不能这样表示。通过像在图 1 中那样构造集合 S 可以证明这一点。我们确实有表示 $-i = (0.1111\cdots)_{1+1}$, $i = (100.1111\cdots)_{1+1}$ 。

18. 设 S_0 是点 $(a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0)_{i-1}$ 的集合, 其中每个 a_k 是 0 或 1 (如果该图放大 16 倍, 则 S_0 由图 1 所示的 256 个内点给出)。我们首先证明 S 是封闭的: 如果 y_1, y_2, \dots 是 S 的一个无限子集, 则我们有 $y_n = \sum_{k \geq 1} a_{nk} 16^{-k}$, 其中每个 a_{nk} 在 S_0 中。构造一棵树, 其节点是 $(a_{n1}, \dots, a_{nr}), 1 \leq r \leq n$, 而且设这个树的一个节点是另一个节点的祖先, 如果它是后一节点的一个初始子序列的话。根据无限性引理 (定理 2.3.4K), 这棵树有一无限通路 (a_1, a_2, a_3, \dots) , 由此知 $\sum_{k \geq 1} a_k 16^{-k}$ 是 S 中 $\{y_1, y_2, \dots\}$ 的一个极限点。

由习题 16 的答案可知,形如 $(a + bi)/16^k$ 的所有数,当 a 和 b 为整数时,都是可表示的。因此如果 x 和 y 是任意实数且 $k \geq 1$,则对于某整数 m 和 n ,数 $z_k = ([16^k x] + [16^k y]i)/16^k$ 在 $S + m + ni$ 中。可以证明,当 $(m, n) \neq (0, 0)$ 时, $S + m + ni$ 到原点的距离有界。因此,如果 $|x|$ 和 $|y|$ 固定且 k 充分大,我们有 $z_k \in S$, 而且

$\lim_{k \rightarrow \infty} z_k = x + yi$ 在 S 中。

[B. Mandelbrot 称 S 为“双龙”, 因为他注意到它实质上是由两条“龙曲线”腹对腹地连在一起得到的: 见他的著作 *Fractals: Form, Chance, and Dimension* (San Francisco: Freeman, 1977), 313~314。在该书中他还指出界的维数为 $2 \lg x \approx 1.523627$, 其中 $x = 1 + 2r^{-2} \approx 1.69562$ 。龙曲线的其它性质在 C. Davis 和 D. E. Knuth 的文章中有介绍, 见 *J. Recr. Math.* 3 (1970), 66~81, 133~149。D. Goffinet 在 *AMM* 98 (1991), 249~255 中对数字 $\{0, 1\}$ 和其它复基的集合 S 作了说明和分析。]

I. Kátaí 和 J. Szabó 已经证明 $-d + i$ 进制产生具有数字 $\{0, 1, \dots, d^2\}$ 的一个数系; 请见 *Acta. Scient. Math.* 37 (1975), 255~260。W. J. Gilbert 考察了这样的系统进一步的性质, *Canadian J. Math.* 34 (1982), 1335~1348; *Math. Magazine* 57 (1984), 77~81。V. Norton 提出了对于数字 $\{0, 1, i, -1, -i\}$ 和 $2 + i$ 进制的另一个有趣的情况 [*Math. Magazine* 57 (1984), 250~251]。关于基于更一般的代数整数的数系的研究, 请见 I. Kátaí 和 B. Kovács, *Acta. Math. Acad. Sci. Hung.* 37 (1981), 159~164, 405~407; B. Kovács, *Acta Math. Hung.* 58 (1991), 113~120; B. Kovács 和 A. Pethő, *Studia Scient. Math. Hung.* 27 (1992), 169~172。

19. 如果 $m > u$ 或 $m < l$, 求 $a \in D$ 使得 $m \equiv a \pmod{b}$; 所求的表示将是 $m' = (m - a)/b$ 的一个表示后接 a 。注意 $m > u$ 意味着 $l < m' < m$; $m < l$ 意味着 $m < m' < u$; 所以算法终止。

[当 $b = 2$ 时无解。当且仅当 $0 \in D$ 时, 表示是惟一的; 例如当 $D = \{-3, -1, 7\}$, $b = 3$ 时出现非惟一的表示, 因为 $(a)_3 = (\overline{3773}a)_3$ 。当 $b \geq 3$ 时, 不难证明恰有 2^{b-3} 个解集合 D , 其中对所有 $a \in D$ 都有 $|a| < b$ 。而且对所有 $b \geq 3$ 和 $n \geq 1$, 当每个 ϵ_j 为 0 或 1 时, 集合 $D = \{0, 1, 2 - \epsilon_2 b^n, 3 - \epsilon_3 b^n, \dots, b - 2 - \epsilon_{b-2} b^n, b - 1 - b^n\}$ 给出惟一的表示。参考: *Proc. IEEE Symp. Comp. Arith.* 4 (1978), 1~9; *JACM* 29 (1982), 1131~1143。]

20. (a) $0.\overline{111}\dots = \overline{1}.888\dots = \overline{18}.\overline{111}\dots = \overline{18}.\overline{1\ 222\ 7.666}\dots = \dots = \overline{18}.\overline{123456\ 777\ 765432.111}\dots$ 有 9 种表示。(b) 一个“ D -分式”。 $a_1 a_2 \dots$ 总是位于 $-1/9$ 和 $+71/9$ 之间。假设 x 有 10 种以上的 D -10 进表示, 则对于充分大的 k , $10^k x$ 有 10 种表示, 它们在小数点左边不同: $10^k x = n_1 + f_1 = \dots = n_{10} + f_{10}$, 其中每个 f_j 是一个 D -分式。由整数表示的惟一性可知, n_j 都是不同的, 比如说 $n_1 < \dots < n_{10}$, 因此 $n_{10} - n_1 \geq 9$; 但这意味着 $f_1 - f_{10} \geq 9 > 71/9 - (-1/9)$, 矛盾。(c) 形如 $0.a_1 a_2 \dots$ 的任何数, 其中每个 a_j 是 -1 或 8 , 等于 $\overline{1}.a'_1 a'_2 \dots$, 其中 $a'_j = a_j + 9$ (而且它甚至还有 6 种另外的表示 $\overline{18}.a''_1 a''_2 \dots$, 等等)。

21. 通过使用类似于正文中提出的转换成平衡三叉树的方法, 我们可以转换成这样一种表示。

与习题 20 中的诸系统相反, 可以以无穷多种方法表示零, 全都由 $\frac{1}{2} + \sum_{k \geq 1} \left(-4 \frac{1}{2}\right) \cdot 10^{-k}$ (或由这个表示的负值) 得到, 方法是用 10 的一个幂乘它。1 的

表示是 $1 \frac{1}{2} - \frac{1}{2}^*, \frac{1}{2} + \frac{1}{2}^*, 5 - 3 \frac{1}{2} - \frac{1}{2}^*, 5 - 4 \frac{1}{2} + \frac{1}{2}^*, 50 - 45 - 3 \frac{1}{2} - \frac{1}{2}^*, 50 - 45 - 4 \frac{1}{2} + \frac{1}{2}^*$, 等等, 其中 $+\frac{1}{2}^* = \left(\pm 4 \frac{1}{2}\right)(10^{-1} + 10^{-2} + \cdots)$ 。[AMM 57 (1950), 90~93.]

22. 给定某个近似 $b_n \cdots b_1 b_0$, 及误差 $\sum_{k=0}^n b_k 10^k - x > 10^{-t}$, 其中 $t > 0$, 我们将说明怎样把误差减少约 10^{-t} 。(找出一个适当的 $\sum_{k=0}^n b_k 10^k > x$, 即可开始这一过程; 经过有限次数的减少以后误差就会小于 ϵ 。)只要选择 $m > n$ 充分地大使得 $-10^m \alpha$ 的十进表示中在 10^{-t} 的位置有一个 1 而在 $10^{-t-1}, 10^{-t-2}, \dots, 10^n$ 的位置没有 1, 则 $10^m \alpha + (10^m \text{ 和 } 10^n \text{ 之间 } 10 \text{ 的幂的适当和}) + \sum_{k=0}^n b_k 10^k \approx \sum_{k=0}^n b_k 10^k - 10^{-t}$ 。

23. 如同习题 18 中那样, 集合 $S = \{\sum a_k b^{-k} | a_k \in D\}$ 是封闭的, 因此它是可测的, 而且事实上它有正测度。因为 $bS = \bigcup_{a \in D} (a + S)$, 我们有 $b\mu(S) = \mu(bS) \leq \sum_{a \in D} \mu(a + S) = \sum_{a \in D} \mu(S) = b\mu(S)$, 因此当 $a \neq a' \in D$ 时, 我们必定有 $\mu((a + S) \cap (a' + S)) = 0$ 。现在如果 $0 \in D$, 则 T 有测度 0, 因为 T 是形如 $b^k(n + ((a + S) \cap (a' + S)))$, $a \neq a'$ 的可数多个集合的并, 它们每个的测度为 0。另一方面, 如同 K. A. Brakke 所指出的, 在习题 21 的数系中, 每一实数都有无穷多个表示。

[集合 T 不能是空的, 因为实数不能写成不相交的可数多个有界闭集的并; 参考 AMM 84 (1977), 827~828, 以及 Petkovšek 在 AMM 97 (1990), 408~411 中更详细的分析。如果 D 少于 b 个元素, 则用 b 进制和取自 D 的数字可表示的集合之测度为 0。如果 D 有 b 个以上的元素并表示所有实数, 则 T 的测度为无穷。]

24. 对于 $k \geq 0$, $\{2a \cdot 10^k + a' | 0 \leq a < 5, 0 \leq a' < 2\}$ 或 $\{5a' \cdot 10^k + a | 0 \leq a < 5, 0 \leq a' < 2\}$ 。[R. L. Graham 证明, 没有更多的具有这些性质的整数数字的集合, 而且 Andrew Odlyzko 已经证明, 对整数的限制是多余的。这是在下边的意义下说的: 如果 D 的最小两个元素是 0 和 1, 则所有数字必定是整数。证明: 设 $S = \{\sum_{k < 0} a_k b^k | a_k \in D\}$ 是“分数”的集合, 并设 $X = \{(a_n \cdots a_0)_b | a_k \in D\}$ 是“整数”的集合; 则 $[0, \infty) = \bigcup_{x \in X} (x + S)$, 且 $(x + S) \cap (x' + S)$ 对于 $x \neq x' \in X$ 有测度 0。我们有 $(0, 1) \subseteq S$, 而且对 m 用归纳法我们将证明, 对某个 $x_m \in X$, $(m, m+1) \subseteq x_m + S$ 。设 $x_m \in X$ 使得对于所有 $\epsilon > 0$, $(m, m+\epsilon) \cap (x_m + S)$ 有正的测度, 则 $x_m \leq m$, 而且 x_m 必定是一个整数以免 $x_{\lfloor x_m \rfloor} + S$ 重叠 $x_m + S$ 太多。如果 $x_m > 0$, 则 $(m - x_m, m - x_m + 1) \cap S$ 有正测度这一事实意味着, 由归纳法, 这个测度为 1, 而且 $(m, m+1) \subseteq x_m + S$, 因为 S 是封闭的。如果 $x_m = 0$ 和 $(m, m+1) \not\subseteq S$, 对于某个 $x'_m \in X$ 我们必定有 $m < x'_m < m+1$, 其中 $(m, x'_m) \subseteq S$; 但这样一来 $1 + S$ 就会重叠 $x'_m + S$ 。见 Proc. London Math. Soc. (3) 18 (1978), 581~595。]

注: 如果我们去掉 $0 \in D$ 的限制, 则还有许多其它情况, 它们当中有些十分有趣, 特别是 $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, $\{1, 2, 3, 4, 5, 51, 52, 53, 54, 55\}$ 和 $\{2, 3, 4, 5, 6,$

52, 53, 54, 55, 56}。或者如果允许负的数字, 则我们用习题 19 的方法得到许多其它的解, 加上像 $\{-1, 0, 1, 2, 3, 4, 5, 6, 7, 18\}$ 这样一些不寻常的数字集合。这些数字不满足那里所述的条件。因此, 不可能为带有负数字的所有解找到一个好的特征。

25. 一个 b 进表示的正数, 其小数点右边有 m 个连续的 $b-1$ 者, 必有 $c/b^n + (b^m - \theta)/b^{n+m}$ 的形式, 其中 c 和 n 是非负整数且 $0 < \theta \leq 1$ 。所以如果 u/v 有这一形式, 则 $b^{m+n}u = b^m cv + b^m v - \theta v$ 。因此 θv 是一个整数, 它是 b^m 的一个倍数。但是 $0 < \theta v \leq v < b^m$ 。[如果 $0 \leq a < b-1$, 例如在 $a/(b-1)$ 的表示中, 则可以有任意长的其它数字 $aaaaa$ 的一些运行。]

26. 通过逐步地构造所求的表示, “充分性”的证明是对于 b 进制的通常证明的一个便捷的推广。“必要性”证明分成两部分: 如果对于某个 n , β_{n+1} 大于 $\sum_{k \leq n} c_k b_k$, 则对于小的 ϵ , $\beta_{n+1} - \epsilon$ 没有表示。如果对所有 n , $\beta_{n+1} \leq \sum_{k \leq n} c_k b_k$, 但其中等式不总成立, 则我们可以证明对某个 x 有两个表示。[见 *Transactions of the Royal Society of Canada*, Series III, 46 (1952), 45~55。]

27. 对 $|n|$ 用归纳法证明: 如果 n 是偶数我们必须取 $e_0 > 0$, 由归纳法即得结果, 因为 $n/2$ 有惟一这样的表示。如果 n 是奇数, 我们必须取 $e_0 = 0$, 问题归结为表示 $-(n-1)/2$; 如果它为 0 或 1, 显然仅有一种方法进行, 否则由归纳法它有惟一的逆序表示。

[由此得出, 每个正整数恰有两种这样的表示, 且带有递减的指数 $e_0 > e_1 > \dots > e_t$; 一个 t 为偶数另一个 t 为奇数。]

28. 可以给出类似于习题 27 的一个证明。注意 $a+bi$ 等于 $1+i$ 乘以一个复整数, 当且仅当 $a+b$ 是偶数。这个表示和习题 18 的答案中讨论的龙曲线密切相关。

29. 只须证明, 满足性质 B 的任何集合 $\{T_0, T_1, T_2, \dots\}$ 均可通过消去某些集合 $\{S_0, S_1, S_2, \dots\}$ 而得到, 其中 $S_0 = \{0, 1, \dots, b-1\}$, 而且 S_1, S_2, \dots 的所有元素是 b 的倍数。

为证明后一命题, 我们可以假定 $1 \in T_0$, 而且有一最小元素 $b > 1$, 使得 $b \in T_0$ 。通过对 n 的归纳法我们将证明, 如果 $nb \in T_0$, 则 $nb+1, nb+2, \dots, nb+b-1$ 都不在任何 T_j 中; 但如果 $nb \in T_0$, 则 $nb+1, \dots, nb+b-1$ 也是如此。于是得出结果, 且 $S_1 = \{nb \mid nb \in T_0\}$, $S_2 = T_1$, $S_3 = T_2$, 等等。

如果 $nb \in T_0$, 则 $nb = t_0 + t_1 + \dots$, 其中 t_1, t_2, \dots 是 b 的倍数; 因此 $t_0 < nb$ 是 b 的一个倍数。由归纳法, $(t_0 + k) + t_1 + t_2 + \dots$ 是 $nb + k$ 的表示, 其中 $0 < k < b$; 因此对于任何 j , $nb + k \in T_j$ 。

如果 $nb \in T_0$, 且 $0 < k < b$, 设 $nb + k$ 的表示是 $t_0 + t_1 + \dots$ 。对于 $j \geq 1$, 我们不能有 $t_j = nb + k$, 以免 $nb + b$ 有两个表示 $(b-k) + \dots + (nb+k) + \dots = (nb) + \dots + b + \dots$ 。由归纳法, $t_0 \bmod b = k$; 而且表示 $nb = (t_0 - k) + t_1 + \dots$ 意味着 $t_0 = nb + k$ 。

[参考文献: *Nieuw Archief voor Wiskunde* (3)4 (1956), 15~17。P. A. MacMa

hon 推导了这一结果的有限模拟,见 *Combinatory Analysis 1* (1915), 217~233.]

30. a) 设 A_j 是数 n 的集合,其表示不含有 b_j ;则由惟一性可知, $n \in A_j$ 当且仅当 $n + b_j \notin A_j$ 。结果,我们有 $n \in A_j$ 当且仅当 $n + 2b_j \in A_j$ 。由此得出,对于 $j \neq k$, $n \in A_j \cap A_k$ 当且仅当 $n + 2b_j b_k \in A_j \cap A_k$ 。设 m 是使得 $0 \leq n < 2b_j b_k$ 的整数 $n \in A_j \cap A_k$ 的个数。则这个区间恰含有 m 个整数,它们在 A_j 中但不在 A_k 中,恰有 m 个在 A_k 中而不在 A_j 中,并且恰有 m 个既不在 A_j 也不在 A_k 中;因此 $4m = 2b_j b_k$ 。因此 b_j 和 b_k 不能皆为奇数。但是当然至少有一个 b_j 是奇数,因为奇数可以被表示。

b) 按照 a), 我们可以对诸 b 重新编号,使得 b_0 是奇数,而 b_1, b_2, \dots 是偶数;则 $\frac{1}{2}b_1, \frac{1}{2}b_2, \dots$ 必然也是一个二进基,而且这一过程可以被迭代。

c) 如果它是一个二进基,则对于任意大的 k 我们必然有正的和负的 d_k , 以便在 n 很大时表示 $\pm 2^n$ 。反之,可以使用下列算法:

S1. [初始化] 置 $k \leftarrow 0$ 。

S2. [完成没有?] 如果 $n = 0$, 则终止。

S3. [选择] 如果 n 是偶数,置 $n \leftarrow n/2$ 。否则把 $2^k d_k$ 包括在表示中,并且置 $n \leftarrow (n - d_k)/2$ 。

S4. [增加 k] k 增加 1 并且返回到 S2。 ■

在每步里强制进行选择;而且在 S3 中总是减少 $|n|$, 除非 $n = -d_k$, 因此这个算法必然终止。

d) 在上边的算法中, S2~S4 步的两个迭代将使下列值发生改变: $4m \rightarrow m, 4m+1 \rightarrow m+5, 4m+2 \rightarrow m+7, 4m+3 \rightarrow m-1$ 。沿用习题 19 中的论证方法,我们只需证明对于 $-2 \leq n \leq 8$ 算法能终止; n 的所有其它值被移向这个区间。在此范围中, $3 \rightarrow -1 \rightarrow -2 \rightarrow 6 \rightarrow 8 \rightarrow 2 \rightarrow 7 \rightarrow 0$ 且 $4 \rightarrow 1 \rightarrow 5 \rightarrow 6$ 。于是 $1 = 7 \cdot 2^0 - 13 \cdot 2^1 + 7 \cdot 2^2 - 13 \cdot 2^3 + 13 \cdot 2^5 - 13 \cdot 2^9 + 7 \cdot 2^{10}$ 。

注:选择 $d_0, d_1, d_2, \dots = 5, -3, 3, 5, -3, 3, \dots$ 也产生一个二进基数。关于进一步的细节,见 *Math. Comp.* 18 (1964), 537~546; A. D. Sands, *Acta Math. Acad. Sci. Hung* 8 (1957), 65~86。

31. (也见相关的习题 3.2.2-11, 4.3.2-13, 4.6.2-22。)

a) 通过以 2 的适当次幂乘分子和分母,我们可以假定 $u = (\dots u_2 u_1 u_0)_2$ 和 $v = (\dots v_2 v_1 v_0)_2$ 是 2-adic 整数,其中 $v_0 = 1$ 。当 $n > 0$ 时利用记号 $u^{(n)}$ 来代表整数 $(u_{n-1} \dots u_0)_2 = u \bmod 2^n$, 现在用下列计算方法确定 w :

设 $w_0 = u_0$ 和 $w^{(1)} = w_0$ 。对于 $n = 1, 2, \dots$ 假定我们已经求得一个整数 $w^{(n)} = (w_{n-1} \dots w_0)_2$ 使得 $u^{(n)} \equiv v^{(n)} w^{(n)} \pmod{2^n}$ 。于是我们有 $u^{(n+1)} \equiv v^{(n+1)} w^{(n)} \pmod{2^n}$, 因此根据量 $(u^{(n+1)} - v^{(n+1)} w^{(n)}) \bmod 2^{n+1}$ 是 0 或 2^n , $w_n = 0$ 或 1。

b) 求最小整数 k 使得 $2^k \equiv 1 \pmod{2n+1}$, 则对于某个整数 $m, 1 \leq m < 2^{k-1}$, 我们有 $1/(2n+1) = m/(2^k - 1)$ 。设 α 是 m 的 k 位二进表示, 则在二进的系

统中 $(0.aaa\cdots)_2$ 乘以 $2n+1$ 是 $(0.111\cdots)_2 = 1$, 而在 2-adic 的系统中 $(\cdots aaa)_2$ 乘以 $2n+1$ 是 $(\cdots 111)_2 = -1$ 。

c) 如果 u 是有理数, 比如说 $u = m/(2^n)$, 其中 n 是奇正数, 则 u 的 2-adic 表示是周期的, 因为带有周期展开的数的集合包括 $-1/n$ 而且在取负、除以 2 及加法运算下是封闭的。反之, 如果对所有 $N \geq \mu$, $u_{N+\lambda} = u_N$, 则 2-adic 数 $(2^\lambda - 1)2^{-\mu}u$ 是一个整数。

d) 形如 $(\cdots u_2 u_1 1)_2$ 的任何数的平方都有 $(\cdots 001)_2$ 的形式, 因此这个条件是必要的。为证明充分性, 我们可以使用下列过程来计算 $n \bmod 8 = 1$ 时的 $v = \sqrt{n}$ 。

H1. [初始化] 置 $m \leftarrow (n-1)/8$, $k \leftarrow 2$, $v_0 \leftarrow 1$, $v_1 \leftarrow 0$, $v \leftarrow 1$ (在这个算法运行期间, 我们将有 $v = (v_{k-1} \cdots v_1 v_0)_2$ 和 $v^2 = n - 2^{k+1}m$)。

H2. [变换] 如果 m 是偶数, 则置 $v_k \leftarrow 0$, $m \leftarrow m/2$ 。否则置 $v_k \leftarrow 1$, $m \leftarrow (m - v - 2^{k-1})/2$, $v \leftarrow v + 2^k$ 。

H3. [增加 k] k 加 1 并返回 H2。 ▮

32. 一个更一般的结果见 Math. Comp. **29** (1975), 84~86。

33. 设 K_n 是所有使得 $k_n = |K_n|$ 的 n 位数的集合。如果 S 和 T 是任何整数的有限集合, 如果对于某个整数 x , $S = T + x$, 我们将说 $S \sim T$, 而且我们将写 $k_n(S) = |\mathcal{K}_n(S)|$, 其中 $\mathcal{K}_n(S)$ 是 $\sim S$ 的 K_n 的所有子集的类。当 $n=0$ 时, 除非 $|S| \leq 1$ 我们有 $k_n(S) = 0$, 因为 0 是仅有的“0 位数字”数。当 $n \geq 1$ 和 $S = \{s_1, \dots, s_r\}$ 时, 我们有

$$\mathcal{K}_n(S) = \bigcup_{0 \leq j < b} \bigcup_{(a_1, \dots, a_r)} \{ \{t_1 b + a_1, \dots, t_r b + a_r\} \mid \\ \{t_1, \dots, t_r\} \in K_{n-1}(\{(s_i + j - a_i)/b \mid 1 \leq i \leq r\}) \}$$

其中内部的并是对于 $1 \leq i \leq r$ 满足条件 $a_i \equiv s_i + j \pmod{b}$ 的所有数字 (a_1, \dots, a_r) 序列进行的。在这个公式中, 我们要求对于 $1 \leq i < i' \leq r$, $t_i - t_{i'} = (s_i - a_i)/b - (s_{i'} - a_{i'})/b$, 所以下标的命名是惟一确定的。因此, 由容斥原理, 我们有 $k_n(S) = \sum_{0 \leq j < b} \sum_{m \geq 1} (-1)^{m-1} f(S, m, j)$, 其中 $f(S, m, j)$ 是整数集合的个数, 这些整数对于 m 个不同的序列 (a_1, \dots, a_r) 可以按上述方式表示为 $\{t_1 b + a_1, \dots, t_r b + a_r\}$, 求和是对 m 个不同的序列 (a_1, \dots, a_r) 的所有选择进行的。给定 m 个不同的序列 $(a_1^{(l)}, \dots, a_r^{(l)}), 1 \leq l \leq m$, 这样集合的数目是 $k_{n-1}(\{(s_i + j - a_i^{(l)})/b \mid 1 \leq i \leq r, 1 \leq l \leq m\})$ 。于是有集合的集 $\mathcal{Q}(S)$ 使得

$$k_n(S) = \sum_{T \in \mathcal{Q}(S)} c_T k_{n-1}(T)$$

其中每个 c_T 是一个整数。而且, 如果 $T \in \mathcal{Q}(S)$, 则它的元素都是接近于 S 的那些元素; 我们有 $\min T \geq (\min S - \max D)/b$ 和 $\max T \leq (\max S + b - 1 - \min D)/b$ 。于是在习题 19 的记号下我们得到对于序列 $\langle k_n(S) \rangle$ 的联立递推关系, 其中 S 跑遍 $[l, u+1]$ 的非空整子集。由于对于任何一个单元素集合 S , $k_n = k_n(S)$, 序列 $\langle k_n \rangle$ 在这

些递推式中出现。由 $k_n(S)$ 的开头几个值可以计算系数 c_T , 所以我们可以得到一个方程组, 它定义生成函数 $k_S(z) = \sum k_n(S) z^n = [|S| \leq 1] + z \sum_{T \in \square(S)} c_T k_T(z)$ 。[参考] *Algorithms* 2 (1981), 31~43。

例如, 当 $D = \{ -1, 0, 3 \}$ 和 $b = 3$ 时, 我们有 $l = -\frac{3}{2}$ 和 $u = \frac{1}{2}$, 所以有关集合 S 是 $\{0\}, \{0, 1\}, \{-1, 1\}$ 和 $\{-1, 0, 1\}$ 。对于 $n \leq 3$, 对应的序列是 $\langle 1, 3, 8, 21 \rangle, \langle 0, 1, 3, 8 \rangle, \langle 0, 0, 1, 4 \rangle$, 和 $\langle 0, 0, 0, 0 \rangle$; 所以我们得到

$$k_0(z) = 1 + z(3k_0(z) - k_{01}(z)), \quad k_{02}(z) = z(k_{01}(z) + k_{02}(z))$$

$$k_{01}(z) = zk_0(z), \quad k_{012}(z) = 0$$

而且 $k(z) = 1/(1 - 3z + z^2)$ 。在这种情况下 $k_z = F_{2n+2}$ 且 $k_n(\{0, 2\}) = F_{2n+1} - 1$ 。

34. 在符号 $\{1, 0, 1\}$ 上恰有一个串 α_n 使得 $n = (\alpha_n)_2$ 而且 α_n 没有前导的零或连续的非零: α_0 是空的, 否则 $\alpha_{2n} = \alpha_n 0, \alpha_{4n+1} = \alpha_n 01, \alpha_{4n-1} = \alpha_n 0\bar{1}$ 。通过使用变换 $1\bar{1} \rightarrow 01, \bar{1}1 \rightarrow 0\bar{1}, 01 \cdots 11 \rightarrow 10 \cdots 0\bar{1}, 0\bar{1} \cdots \bar{1}1 \rightarrow \bar{1}0 \cdots 01$, 以及插入或删除前导零, 任何表示 n 的串都可被转换成 α_n 。由于这些变换并不增加非零数字的个数, 因此 α_n 有最少数字。[*Advances in Computers* 1 (1960), 244~260。]以 $\bar{v}(n)$ 表示 α_n 中的非零数字的个数, 它是紧前边有 0 或者对于某个 $k \geq 0$, 有子串 $00(10)^k 1$ 居前的通常表示中 1 的个数。

对于 $b > 2$ 进制的一个推广已由 J. von zur Gathen 给出, *Computational Complexity* 1 (1991), 360~394。

4.2.1 小节

1. $N = (62, +.60\ 22\ 14\ 00); h = (37, +.66\ 26\ 10\ 00)$ 。注意, $10h$ 将是 $(38, +.06\ 62\ 61\ 00)$ 。

2. $b^{E-q}(1 - b^{-p}), b^{-q-p}; b^{E-q}(1 - b^{-p}), b^{-q-1}$ 。

3. 当 e 没有其最小值时, 最高的“1”位(它出现在所有这样规格化的数中)不必出现在计算机字当中。

4. $(51, +.10209877); (50, +.12346000); (53, +.99999999)$ 。如果头一个操作数曾是 $(45, -.50000000)$, 则第三个答案将是 $(54, +.10000000)$, 因为 $b/2$ 是奇数。

5. 如果 $x \sim y$ 且 m 是一个整数, 则 $mb + x \sim mb + y$, 而且通过考虑所有可能的情况, $x \sim y$ 意味着 $x/b \sim y/b$ 。另一个关键的性质是每当 $bx \sim by$ 时, x 和 y 将舍入成相同的整数。

现在如果 $b^{-p-2}F_v \neq f_v$, 我们必定有 $(b^{p+2}f_v) \bmod b \neq 0$; 因此除非 $e_u - e_v \geq 2$, 否则这个变换保持 f_v 不变。由于 u 是规格化的, 它是非零的且 $|f_u + f_v| > b^{-1} - b^{-2} \geq b^{-2}$; $f_u + f_v$ 的前导非 0 位必然至多在小数点右边两位, 而且舍入操作将把

$b^{p+j}(f_u + f_v)$ 转换为一个整数, 其中 $j \leq 1$ 。如果我们能够证明 $b^{p+j+1}(f_u + f_v) \sim b^{p+j+1}(f_u + b^{-p-2}F_v)$, 则证明将完成。由前段, 我们有 $b^{p+2}(f_u + f_v) \sim b^{p+2}f_u + F_v = b^{p+2}(f_u + b^{-p-2}F_v)$, 它意味着对所有 $j \leq 1$, 得出所求结果。类似的说明适用于算法 M 的步骤 M2。

注意, 当 $b > 2$ 是偶数时, 这样一个整数 F_v 总存在; 但当 $b = 2$ 时, 我们要求 $p + 3$ 位 (令 $2F_v$ 是一个整数)。当 b 为奇数时, 整数 F_v 总存在, 除非在除法的情况下和可能有 $\frac{1}{2}b$ 的余数时。

6. (考虑程序 A 中 $e_u = e_v, f_u = -f_v$ 的情况。) 如同在 ADD 中一样, 寄存器 A 保留它以前的符号。

7. 说一个数是规格化的当且仅当它是零或它的小数部分在 $\frac{1}{6} < |f| < \frac{1}{2}$ 的范围内。对于加法和减法来说 $(p+1)$ 位累加器就足够了; 舍入 (除在除法期间外) 等价于截取。确实是一个非常合意的系统! 我们可以用超 0 指数表示这些数, 在分数的头一个和后继数字之间插入这个指数。而且如果这分数为负, 就求补, 使得定点的顺序保持不变。

8. (a) $(06, +.12345679) \oplus (06, -.12345678), (01, +.10345678) \oplus (00, -.94000000)$; (b) $(99, +.87654321) \oplus$ 它本身, $(99, +.99999999) \oplus (91, +.50000000)$ 。

9. $a = c = (-50, +.10000000), b = (-41, +.20000000), d = (-41, +.80000000), y = (11, +.10000000)$ 。

10. $(50, +.99999000) \oplus (55, +.99999000)$ 。

11. $(50, +.10000001) \otimes (50, +.99999990)$ 。

12. 如果 $0 < |f_u| < |f_v|$, 则 $|f_u| \leq |f_v| - b^{-p}$; 因此 $1/b < |f_u/f_v| \leq 1 - b^{-p}/|f_v| < 1 - b^{-p}$ 。如果 $0 < |f_v| \leq |f_u|$, 我们有 $1/b < |f_u/f_v|/b \leq ((1 - b^{-p})/(1/b))/b = 1 - b^{-p}$ 。

13. 见 J. Michael Yohe, *IEEE Transactions C-22* (1973), 577 ~ 586; 也参考习题 4.2.2-24。

14.	FIX	STJ	9F	浮点化定点子程序
		STA	TEMP	
		LD1	TEMP(EXP)	$r11 \leftarrow e$
		SLA	1	$rA \leftarrow \pm ffff0$
		JAZ	9F	输入为 0?
		DEC1	1	
		CMFA	$= 0 = (1:1)$	如果前导字节为 0,
		JE	$* - 4$	则再进行左移
		ENN1	$-Q - 4, 1$	

	J1N	FIXOVFLO	数量太大吗?
	ENTX	0	
	SRAX	0,1	
	CMPX	= 1//2 =	
	JL	9F	
	JG	* + 2	
	JAO	9F	含混的情况变成奇数,因为 $b/2$ 是偶数
	STA	* + 1(0:0)	必要时,进行舍入
	INCA	1	加 ± 1 (不可能溢出)
9H	JMP	*	从子程序出口
15.	FP	STJ	EXITE 小数部分子程序
		JOV	OFLO 确保溢出开关断开
		STA	TEMP $TEMP \leftarrow u$
		ENTX	0
		SLA	1 $rA \leftarrow f_u$
		LD2	TEMP(EXP) $r12 \leftarrow e_u$
		DEC2	Q
		J2NP	* + 3
		SLA	0,2 消去 u 的整数部分
		ENT2	0
		JANN	1F
		ENN2	0,2 小数部分为负,求它的补
		SRAX	0,2
		ENT2	0
		JXNZ	* + 3
		JAZ	* + 2
		INCA	1
		ADD	WM1 加字长减 1
1H		INC2	Q 准备把答案规格化
		JMP	NORM 规格化,舍入并出口
8H		EQU	1(1:1)
WM1	CON	8B - 1, 8B - 1(1:4)	字长减 1

16. 如果 $|c| \geq |d|$, 则置 $r \leftarrow d \oslash c, s \leftarrow c \oplus (r \otimes d); x \leftarrow (a \oplus (b \otimes r)) \oslash s, y \leftarrow (b \ominus (a \otimes r)) \oslash s$ 。否则置 $r \leftarrow c \oslash d, s \leftarrow d \oplus (r \otimes c); x \leftarrow ((a \otimes r) \oplus b) \oslash s, y \leftarrow ((b \otimes r) \ominus a) \oslash s$ 。则 $x + iy$ 是所求的 $(a + bi)/(c + di)$ 近似。[CACM 5 (1963), 435。P. Wynn 给出了对于复数运算和函数求值的其它算法, 见 BIT 2 (1962), 232~255; 也见 Paul Friedland, CACM 10 (1967), 665。]

17. 见 Robert Morris, IEEE Transactions C-20 (1971), 1578~1579。对于这样的系统, 误差分析更为困难, 所以区间算法也更符合要求。

18. 对于正数:小数左移直到 $f_1 = 1$, 然后进行舍入, 然后如果小数为 0 (舍入上溢), 再次右移。对于负数:小数左移直到 $f_1 = 0$, 然后进行舍入, 然后如果分数为 0 (舍入下溢), 再次右移。

19. $(43 + [e_v < e_u] - [\text{分数上溢}] - 10[\text{结果为 0}] + 4[\text{量被向上舍入}] + [\text{头一个舍入数字为 } b/2] + 5[\text{舍入数字为 } \frac{b}{2} 0 \cdots 0] + 7[\text{舍入上溢}] + 7N + (11[N > 0] - 1)[rX \text{ 接收非零数字}])u$, 其中 N 是规格化期间的左移次数。例如当

$$u = +50\ 01\ 00\ 00\ 00, \quad v = -46\ 49\ 99\ 99\ 99, \quad b = 100$$

时, 出现极大时间 $73u$ 。[考虑 4.2.4 小节中的数据, 平均时间将大约是 $45.5u$ 。]

4.2.2 小节

1. $u \ominus v = u \oplus -v = -(v \oplus u) = -(v \ominus u)$ 。

2. 由 (8), (2), (6) 可知, $u \oplus x \geq u \oplus 0 = u$; 因此再次使用 (8), 得 $(u \oplus x) \oplus v \geq u \oplus v$ 。类似地, (8) 和 (6) 连同 (2) 一起, 意味着 $(u \oplus x) \oplus (v \oplus y) \geq (u \oplus x) \oplus v$ 。

3. $u = 8.0000001, v = 1.2500008, w = 8.0000008; (u \otimes v) \otimes w = 80.000064$, 而 $u \otimes (v \otimes w) = 80.000057$ 。

4. 是的; 令 $1/u \approx v = w$, 其中 v 很大。

5. 不总是; 在十进算术中取 $u = v = 9$ 。

6. (a) 是。(b) 仅对 $b + p \leq 4$ (试验 $u = 1 - b^{-p}$)。但请参见习题 27。

7. 如果 u 和 v 是相继的浮点二进数, $u \oplus v = 2u$ 或 $2v$ 。当它是 $2v$ 时, 我们通常有 $u^{\otimes 2} \oplus v^{\otimes 2} < 2v^{\otimes 2}$ 。例如, $u = (.10 \cdots 001)_2, v = (.10 \cdots 010)_2, u \oplus v = 2v$, 且 $u^{\otimes 2} + v^{\otimes 2} = (.10 \cdots 011)_2$ 。

8. a) \sim, \approx ; b) \sim, \approx ; c) \sim, \approx ; d) \sim ; e) \sim 。

9. $|u - w| \leq |u - v| + |v - w| \leq \epsilon_1 \min(b^{e_u - q}, b^{e_v - q}) + \epsilon_2 \min(b^{e_v - q}, b^{e_w - q}) \leq \epsilon_1 b^{e_u - q} + \epsilon_2 b^{e_w - q} \leq (\epsilon_1 + \epsilon_2) \max(b^{e_u - q}, b^{e_w - q})$ 。这个结果一般不能加强, 因为比如说相对于 e_v 和 e_w 两者, 我们可能有非常之小的 e_u , 而这意味着, 在这个假设之下 $u - w$ 可能相当大。

10. 如果 $a_p \geq 1$ 和 $a_1 \geq \frac{b}{2}$, 则我们有 $(.a_1 \cdots a_{p-1} a_p)_b \otimes (.9 \cdots 99)_b = (.a_1 \cdots a_{p-1} (a_p - 1))_b$; 这里“9”代表 $b - 1$ 。且 $(.a_1 \cdots a_{p-1} a_p)_b \otimes (1.0 \cdots 0)_b = (.a_1 \cdots a_{p-1} 0)_b$, 所以如果 $b > 2$ 和 $a_p \geq 1 + \left\lceil a_1 \geq \frac{b}{2} \right\rceil$, 则乘法不是单调的。但当 $b = 2$ 时, 这个论证可加推广, 以证明乘法是单调的; 显然“某台计算机”有 $b > 2$ 。

11. 不失一般性, 设 x 是一个整数, $0 \leq x < b^p$ 。如果 $e \leq 0$, 则 $t = 0$ 。如果 $0 < e \leq p$, 则 $x - t$ 至多有 $p + 1$ 个数字, 最低有效位为 0。如果 $e > p$, 则 $x - t = 0$ 。[在更弱的假设 $|t| < b^e$ 下, 这个结果也成立; 在该种情况下, 当 $e > p$ 时我们有 $x - t = b^e$ 。]

12. 假设 $e_u = p, e_v \leq 0, u > 0$. 情况 1, $u > b^{p-1}$. 情况 (1a), $w = u + 1, v \geq \frac{1}{2}$, $e_v = 0$, 则 $u' = u$ 或 $u + 1, v' = 1, u'' = u, v'' = 1$ 或 0 . 情况 (1b), $w = u, |v| \leq \frac{1}{2}$, 则 $u' = u, v' = 0, u'' = u, v'' = 0$. 如果 $|v| = \frac{1}{2}$, 而且允许更一般的舍入, 我们也可能有 $u' = u \pm 1, v'' = \mp 1$. 情况 (1c), $w = u - 1, v \leq -\frac{1}{2}, e_v = 0$, 则 $u' = u$ 或 $u - 1, v' = -1, u'' = u, v'' = 1$ 或 0 . 情况 2, $u = b^{p-1}$. 情况 (2a), $w = u + 1, v \geq \frac{1}{2}, e_v = 0$, 和 (1a) 类似. 情况 (2b), $w = u, |v| \leq \frac{1}{2}, u' \geq u$, 和 (1b) 类似. 情况 (2c), $w = u, |v| \leq \frac{1}{2}, u' < u$. 于是 $u' = u - j/b$, 其中 $v = j/b + v_1$ 和 $|v_1| \leq \frac{1}{2}b^{-1}$, 其中 $j \leq \frac{1}{2}b$ 是某个正整数. 我们有 $v' = 0, u'' = u, v'' = j/b$. 情况 (2d), $w < u$, 则 $w = u - j/b$, 其中 $v = -j/b + v_1$ 和 $|v_1| \leq \frac{1}{2}b^{-1}$, 其中 $j \leq b$ 是某个正整数; 我们有 $(v', u'') = (-j/b, u)$ 和 $(u', v'') = (u, -j/b)$ 或 $(u - 1/b, (1 - j)/b)$, 后一情况仅当 $v_1 = \frac{1}{2}b^{-1}$ 时才出现. 在所有的情况下, $u \ominus u' = u - u', v \ominus v' = v - v', u \ominus u'' = u - u'', v \ominus v'' = v - v'', \text{round}(w - u - v) = w - u - v$.

13. 由于 $\text{round}(x) = 0$ 当且仅当 $x = 0$, 因此我们要来找整数偶 (m, n) 的一个很大的集合, 它具有如下的性质: $m \oslash n$ 是整数, 当且仅当 m/n 是整数. 假定 $|m|, |n| < b^p$. 如果 m/n 是整数, 则 $m \oslash n = m/n$ 也是. 反之, 如果 m/n 不是一个整数, 但 $m \oslash n$ 是, 则我们有 $1/|n| \leq |m \oslash n - m/n| < \frac{1}{2}|m/n|b^{1-p}$, 因此 $|m| > 2b^{p-1}$. 因此我们的答案是要求 $|m| \leq 2b^{p-1}$ 和 $0 < |n| < b^p$. (稍弱些的假设也是可能的.)

14. $|(u \otimes v) \otimes w - uvw| \leq |(u \otimes v) \otimes w - (u \otimes v)w| + |w||u \otimes v - uv| \leq \delta_{(u \otimes v) \otimes w} + b^{e_w - q - l_w} \delta_{u \otimes v} \leq (1 + b)\delta_{(u \otimes v) \otimes w}$. 现在 $|e_{(u \otimes v) \otimes w} - e_{u \otimes (v \otimes w)}| \leq 2$, 所以我们可以取 $c = \frac{1}{2}(1 + b)b^{2-p}$.

15. $u \leq v$ 意味着 $(u \oplus u) \oslash 2 \leq (u \oplus v) \oslash 2 \leq (v \oplus v) \oslash 2$, 所以这个条件对所有的 u 和 v 都成立当且仅当每当 $u = v$ 时它成立. 因此, 对于进制 $b = 2$, 这个条件总是满足的 (不包括溢出); 但对于 $b > 2$, 有一些数 $v \neq w$, 使得 $v \oplus v = w \oplus w$, 因此这个条件不成立. [另一方面, 公式 $u \oplus ((v \ominus u) \oslash 2)$ 确实给出了正确范围的中间点. 证明: 只需证明 $u + (v \ominus u) \oslash 2 < v$, 即 $(v \ominus u) \oslash 2 \leq v - u$; 而且容易验证, 对所有的 $x \geq 0, \text{round}\left(\frac{1}{2}\text{round}(x)\right) \leq x$.]

16. (a) 指数变化出现在 $\Sigma_{10} = 11.111111$, $\Sigma_{91} = 101.11111$, $\Sigma_{901} = 1001.1102$, $\Sigma_{9001} = 10001.020$, $\Sigma_{90009} = 100000.91$, $\Sigma_{900819} = 1000000.0$; 因此 $\Sigma_{1000000} = 1109099.1$ 。

(b) 在计算 $\sum_{k=1}^n 1.2345679 = 1224782.1$ 之后, (14) 试图取 $- .0053187053$ 的平方根。但在这种情况下 (15) 和 (16) 是精确的。[如果 $x_k = 1 + \lfloor (k-1)/2 \rfloor 10^{-7}$, (15) 和 (16) 有 n 阶误差。对于标准差计算的进一步精确结果见 Chan 和 Lewis, CACM 22 (1979), 526~531。]

(c) 我们需要证明 $u \oplus ((v \odot u) \odot k)$ 位于 u 和 v 之间; 见习题 15。

17.	FCMP	STJ	9F	浮点比较子程序
		JOV	OFLO	确保溢出开关断开
		STA	TEMP	
		LDAN	TEMP	$v \leftarrow -v$
				(这里照抄程序 4.2.1A 的行 07~20)
		LDX	FV(0:0)	置 rX 为 0 并带 f_v 的符号
		DEC1	5	
		J1N	*+2	
		ENT1	0	以较小的差代替指数中大的差
		SRAX	5.1	
		ADD	FU	$rA \leftarrow$ 操作数的差
		JOV	7F	小数溢出: 非~
		CMPA	EPSILON(1:5)	
		JG	8F	若非~, 则转移
		JL	6F	若是~, 则转移
		JXZ	9F	若是~, 则转移
		JXP	1F	如果 $ rA = \epsilon$ 则校验 $rA \times rX$ 的符号
		JAP	9F	若是~, 则转移。(rA \neq 0)
		JMP	8F	
7H		ENTX	1	
		SRC	1	使 rA 非零并有相同符号
		JMP	8F	
1H		JAP	8F	若不是~, 则跳。(rA \neq 0)
6H		ENTA	0	
8H		CMPA	=0=	置比较指示符
9H		JMP	*	从子程序出口

19. 设对于 $k > n$, $\gamma_k = \delta_k = \eta_k = \sigma_k = 0$ 。只须求 x_1 的系数即可, 因为除所有下标都增 $k-1$ 外, x_k 的系数与它相同。设 (f_k, g_k) 分别表示 $(s_k - c_k, c_k)$ 中 x_1 的系数, 则 $f_1 = (1 + \eta_1)(1 - \gamma_1 - \gamma_1\delta_1 - \gamma_1\sigma_1 - \delta_1\sigma_1 - \gamma_1\delta_1\sigma_1)$, $g_1 = (1 + \delta_1)(1 + \eta_1)(\gamma_1 + \sigma_1 + \gamma_1\sigma_1)$, 以及 $f_k = (1 - \gamma_k\sigma_k - \delta_k\sigma_k - \gamma_k\delta_k\sigma_k)f_{k-1} + (\gamma_k - \eta_k + \gamma_k\delta_k + \gamma_k\eta_k +$

$\gamma_k \delta_k \eta_k + \gamma_k \eta_k \sigma_k + \delta_k \eta_k \sigma_k + \gamma_k \delta_k \eta_k \sigma_k) g_{k-1}, g_k = \sigma_k (1 + \gamma_k) (1 + \delta_k) f_{k-1} - (1 + \delta_k) (\gamma_k + \gamma_k \eta_k + \eta_k \sigma_k + \gamma_k \eta_k \sigma_k) g_{k-1}, 1 < k \leq n$, 于是 $f_n = 1 + \eta_1 - \gamma_1 + (\text{第二阶的 } 4n \text{ 项}) + (\text{更高阶的项}) = 1 + \eta_1 - \gamma_1 + O(n\epsilon^2)$, 这个数充分小。[Kahan 的求和公式首先发表在 CACM 8 (1965), 40; 也参考 Proc. IFIP Congress (1971), 2, 1232, 以及由 K. Ozawa 在 J. Information Proc. 6 (1983), 226~230 上论述的进一步发展。关于精确求和的另一个方法, 见 R.J. Hanson, CACM 18 (1975), 57~58。当某些 x 为负而其它为正时, 如同由 T.O. Espelid 在 SIAM Review 37 (1995), 603~607 中所说明的, 我们可以有利地匹配它们。关于给定 $|x_1, \dots, x_n|$, 精确地计算 $\text{round}(x_1 + \dots - x_n)$ 和 $\text{round}(x_1 \cdots x_n)$ 的一些算法, 也可见 G. Bohlender, IEEE Trans. C-26 (1977), 621~632。]

20. 由定理 C 的证明, 仅当 $|v| + \frac{1}{2} \geq |w - u| \geq b^{p-1} + b^{-1}$ 时, (47) 对 $e_w = p$ 才不成立; 因此 $|f_u| \geq |f_v| \geq 1 - \left(\frac{1}{2}b - 1\right)b^{-p}$ 。我们现在求出在规格化过程期间对于失灵的一个必要和充分的条件是 $|f_w|$ 实质上舍入成 2 (实际上在用于分数溢出的右调之后成为 $2/b$)——这确实是一种非常稀少的情况!

21. (G. W. Veltkamp 给出的解) 设 $c = 2^{\lceil p/2 \rceil} + 1$; 我们可以假定 $p \geq 2$, 所以 c 是可表示的。首先计算 $u' = u \otimes c, u_1 = (u \ominus u') \oplus u', u_2 = u \ominus u_1$; 类似地, $v' = v \otimes c, v_1 = (v \ominus v') \oplus v', v_2 = v \ominus v_1$ 。然后置 $w \leftarrow u \otimes v, w' \leftarrow (((u_1 \otimes v_1 \ominus w) \oplus (u_1 \otimes v_2)) \oplus (u_2 \otimes v_1)) \oplus (u_2 \otimes v_2)$ 。

只须当 $u, v > 0$ 和 $e_u = e_v = p$, 使得 u 和 v 是属于 $[2^{p-1}, 2^p)$ 的整数时, 证明这一点即可。于是 $u = u_1 + u_2$, 其中 $2^{p-1} \leq u_1 \leq 2^p, u_1 \bmod 2^{\lceil p/2 \rceil} = 0$, 且 $|u_2| \leq 2^{\lceil p/2 \rceil - 1}$; 类似地, $v = v_1 + v_2$ 。在计算 w' 期间运算是精确的, 因为 $w - u_1 v_1$ 是 2^{p-1} 的倍数, 使得 $|w - u_1 v_1| \leq |w - uv| + |u_2 v_1 + u_1 v_2 + u_2 v_2| \leq 2^{p-1} + 2^{p+\lceil p/2 \rceil} + 2^{p-1}$; 而且类似地 $|w - u_1 v_1 - u_1 v_2| \leq |w - uv| + |u_2 v| < 2^{p-1} + 2^{\lceil p/2 \rceil + 1 + p}$, 其中 $w - u_1 v_1 - u_1 v_2$ 是 $2^{\lceil p/2 \rceil}$ 的倍数。

22. 我们可以假定 $b^{p-1} \leq u, v < b^p$ 。如果 $uv \leq b^{2p-1}$, 则 $x_1 = uv - r$, 其中 $|r| \leq \frac{1}{2}b^{p-1}$, 因此 $x_2 = \text{round}(u - r/v) = x_0$ (因为 $|r/v| \leq \frac{1}{2}b^{p-1}/b^{p-1} \leq \frac{1}{2}$, 而且等式意味着 $v = b^{p-1}$, 因此 $r = 0$)。如果 $uv > b^{2p-1}$, 则 $x_1 = uv - r$, 其中 $|r| \leq \frac{1}{2}b^p$, 因此 $x_1/v = u - r/v < b^p + \frac{1}{2}b$ 且 $x_2 \leq b^p$ 。如果 $x_2 = b^p$ 则 $x_3 = x_1$ (因为条件 $(b^p - \frac{1}{2})v \leq x_1$ 意味着 x_1 是 b^p 的倍数, 而且我们有 $x_1 < b^p(v + \frac{1}{2})$)。如果 $x_2 < b^p$ 和 $x_1 > b^{2p-1}$, 则令 $x_2 = x_1/v - q$, 其中 $|q| \leq \frac{1}{2}$; 我们有 $x_3 = \text{round}(x_1 + qv) = x_1$ 。最

后如果 $x_2 < b^p$, $x_1 = b^{2p-1}$ 和 $x_3 < b^{2p-1}$, 则由上述头一种情况可知 $x_4 = x_2$ 。比如说, 当 $b = 10$, $p = 2$, $u = 19$, $v = 55$, $x_1 = 1000$, $x_2 = 18$, $x_3 = 990$ 时, 出现这种情况。

23. 如果 $u \geq 0$ 或 $u \leq -1$, 我们有 $u \text{ (mod)} 1 = u \bmod 1$, 所以恒等式成立。如果 $-1 < u < 0$, 则 $u \text{ (mod)} 1 = u \oplus 1 = u + 1 + r$, 其中 $|r| \leq \frac{1}{2}b^{-p}$; 这个恒等式成立当且仅当 $\text{round}(1+r) = 1$, 所以如果我们舍入成偶数它总成立。利用正文的舍入规则这个恒等式不成立当且仅当 b 是 4 的倍数及 $-1 < u < 0$ 且 $u \bmod 2b^{-p} = \frac{3}{2}b^{-p}$ (例如, $p = 3$, $b = 8$, $u = -(.0124)_8$)。

24. 设 $u = [u_l, u_r]$, $v = [v_l, v_r]$, 则 $u \oplus v = [u_l \nabla v_l, u_r \Delta v_r]$, 其中对所有 x , $x \Delta y = y \Delta x$, $x \Delta +0 = x$, 对所有 $x \neq +0$, $x \Delta -0 = x$, 对所有 $x \neq -\infty$, $x \Delta +\infty = +\infty$, 而且 $x \Delta -\infty$ 不必是有定义的; $x \nabla y = -((-x) \Delta (-y))$ 。如果在通常的浮点算术运算时, 由于 $x + y$ 太大, 因而 $x \oplus y$ 将溢出, 则 $x \Delta y$ 是 $+\infty$ 而且 $x \nabla y$ 是可表示的最大数。

对于减法, 设 $u \ominus v = u \oplus (-v)$, 其中 $-v = [-v_r, v_l]$ 。

乘法稍微复杂一点, 正确的过程是设

$$u \otimes v = [\min(u_l \nabla v_l, u_l \nabla v_r, u_r \nabla v_l, u_r \nabla v_r), \\ \max(u_l \Delta v_l, u_l \Delta v_r, u_r \Delta v_l, u_r \Delta v_r)]$$

其中 $x \Delta y = y \Delta x$, $x \Delta (-y) = -(x \nabla y) = (-x) \Delta y$; $x \Delta +0 = (+0, \text{对 } x > 0, -0, \text{对 } x < 0)$; $x \Delta -0 = -(x \Delta +0)$; $x \Delta +\infty = (+\infty \text{ 对 } x > +0, -\infty \text{ 对 } x < -0)$ 。(有可能只需考察 u_l, u_r, v_l 和 v_r 的符号就能确定 \min 和 \max , 由此只计算八个乘积的两个, 除非当 $u_l < 0 < u_r$ 和 $v_l < 0 < v_r$ 时; 在后边这种情况下, 我们计算四个乘积, 而且答案是 $[\min(u_l \nabla v_r, u_r \nabla v_l), \max(u_l \Delta v_l, u_r \Delta v_r)]$ 。)

最后, 如果 $v_l < 0 < v_r$, 则 $u \otimes v$ 无定义; 否则我们在乘法公式中分别以 v_r^{-1} 和 v_l^{-1} 代替 v_l 和 v_r , 其中 $x \Delta y^{-1} = x \Delta y$, $x \nabla y^{-1} = x \nabla y$, $(\pm 0)^{-1} = \pm \infty$, $(\pm \infty)^{-1} = \pm 0$ 。

[参考 E. R. Hansen, *Math. Comp.* **22** (1968), 374~384。另一个方案(其中除以 0 不给出出错信息而且区间可以是 ∞ 的邻域)已由 W. M. Kahan 提出。例如, 在 Kahan 的方案中, $[-1, +1]$ 的倒数是 $[+1, -1]$, 而且一个含 0 的区间乘以含 ∞ 的区间产生 $[-\infty, +\infty]$, 即所有数的集合。见 *Numerical Analysis*, (Univ. Michigan Engineering Summer Conf. Notes No. 6818 (1968))。]

25. 量的删除揭示了前面计算 u 和 v 时的误差。例如, 如果 ϵ 很小, 当计算 $f(x+\epsilon) \ominus f(x)$ 时, 我们通常得到很差的精度, 因为 $f(x+\epsilon)$ 的舍入计算破坏了许多关于 ϵ 的信息。最好把这样的公式改写为 $\epsilon \otimes g(x, \epsilon)$, 其中 $g(x, \epsilon) = (f(x+\epsilon) - f(x))/\epsilon$ 首先以符号方式进行计算。于是如果 $f(x) = x^2$, 则 $g(x, \epsilon) = 2x + \epsilon$; 如果 $f(x) = \sqrt{x}$, 则 $g(x, \epsilon) = 1/(\sqrt{x+\epsilon} + \sqrt{x})$ 。

26. 设 $e = \max(e_u, e_u')$, $e' = \max(e_v, e_v')$, $e'' = \max(e_{u \oplus v}, e_{u' \oplus v'})$, 并假定 $q =$

0。于是 $(u \oplus v) - (u' \oplus v') \leq u + v + \frac{1}{2}b^{e''-p} - u' - v' + \frac{1}{2}b^{e''-p} \leq eb^e + eb^{e'} + b^{e''-p}$, 以及 $e'' \geq \max(e, e')$ 。因此 $u \oplus v \sim u' \oplus v' (2e + b^{-p})$ 。

如果 $b=2$, 这个估计可以被改进成 $1.5e + b^{-p}$ 。因为如果 $u - u'$ 和 $v - v'$ 有相反的符号, $e + b^{-p}$ 是一个上限, 而且在其他情况下我们不能有 $e = e' = e''$ 。

27. 所述的恒等式是以下事实的一个结果, 即每当 $b^{-1} \leq f_v \leq b^{-1/2}$ 时, $1 \odot (1 \odot u) = u$ 。如果后一结果不成立, 则将有整数 x 和 y 使得 $b^{p-1} < x < b^{p-1/2}$, 以及或者 $y - \frac{1}{2} \leq b^{2p-1}/x < b^{2p-1}/(x - \frac{1}{2}) \leq y$ 或者 $y \leq b^{2p-1}/(x + \frac{1}{2}) < b^{2p-1}/x \leq y + \frac{1}{2}$ 。但是除非我们有 $x(x + \frac{1}{2}) > b^{2p-1}$, 否则这显然是不可能的, 这后一条件意味着 $y = \lfloor b^{p-1/2} \rfloor = x$ 。

28. 参见 Math. Comp. 32 (1978), 227~232。

29. 当 $b=2$ 和 $p=1$ 且 $x>0$ 时, 我们有 $\text{round}(x) = 2^{e(x)}$, 其中 $e(x) = \lfloor \lg \frac{4}{3}x \rfloor$ 。

令 $f(x) = x^\alpha$ 且令 $t(n) = \lfloor \lfloor \alpha n + \lg \frac{4}{3} \rfloor / \alpha + \lg \frac{4}{3} \rfloor$ 。于是 $\hat{h}(2^e) = 2^{t(e)}$ 。当 $\alpha = .99$ 时, 对于 $41 < e \leq 58$, 我们求得 $\hat{h}(2^e) = 2^{e-1}$ 。

31. 按照 4.5.3 小节的理论, 连分数 $\sqrt{3} = 1 + //1, 2, 1, 2, \dots //$ 的收敛式是 $p_n/q_n = K_{n+1}(1, 1, 2, 1, 2, \dots)/K_n(1, 2, 1, 2, \dots)$ 。这些收敛式是对 $\sqrt{3}$ 的极好近似, 因此 $3q_n^2 \approx p_n^2$; 事实上, $3q_n^2 - p_n^2 = 2 - 3(n \bmod 2)$ 。给出的例子是 $2p_{31}^2 + (3q_{31}^2 - p_{31}^2)$ ($3q_{31}^2 + p_{31}^2$) $= 2p_{31}^2 - (p_{31}^2 - 1 + p_{31}^2) = 1$ 。从 $3q_{31}^2$ 减去 p_{31}^2 的浮点减法产生零, 除非我们能几乎完美地表示 $3q_{31}^2$; 从 $9q_{31}^4$ 减去 p_{31}^4 一般地给出比 $2p_{31}^2$ 大得多的舍入误差。类似的例子可以基于近似于任何代数数的连分数给出。

4.2.3 小节

1. 首先, $(w_m, w_l) = (.573, .248)$; 其次 $w_mv_l/v_m = .290$; 所以答案是 $(.572, .958)$ 。这个结果精确到 6 位十进数字。

2. 答案不受影响, 因为规格化子程序截断到 8 位而且绝不会考察这个特定的字节位置。(规格化时左调顶多出现一次, 因为输入是规格化的。)

3. 在行 09 处上溢显然不可能出现, 因为我们正在做双字节量的加法, 也不可能行 22 处出现, 因为我们正在做四字节量的加法。在行 30 中, 我们计算的是三个四字节量之和, 这不可能溢出。最后在行 32 处, 溢出不可能, 因为乘积 $f_u f_v$ 必须小于 1。

4. 在行 03 和 04 之间插入 "JOF OFLO; ENT1 0"。用 "ADD TEMP(ABS); JNOV * + 2; INC1 1" 代替行 21~22, 并把行 28~31 改为 "SLAX 5; ADD TEMP; JNOV * + 2; INC1 1; ENTX 0, 1; SC 5"。这样做只增加 5 行代码而且仅增加 1, 2 或 3 个执行时间单位。

5. 在行 06 之后插入“JOV OFLO”。分别改变行 22,31,39 成为“SRAX 0,1”,“SLAX 5”,“ADD ACC”。在行 40 和 41 之间插入“DEC2 1;JNOV DNORM;INC2 1;INCX 1;SRC 1”。(删去“DEC2 1”而加进“STZ EXPO”是有吸引力的,但那样“INC2 1”可能使 rI2 溢出!)这样做将增加 6 行代码;运行时间减少 $3u$,除非有小数溢出,那时它增加 $7u$ 。

6.	DOUBLE	STJ	EXITDF	转换成双精度
		ENTX	0	清 rX
		STA	TEMP	
		LD2	TEMP(EXF)	$rI2 \leftarrow e$
		INC2	QQ-Q	校正余量中的差
		STZ	EXPO	$EXPO \leftarrow 0$
		SLAX	1	去掉指数
	SINGLE	JMP	DNORM	规格化并出口
		STJ	EXITF	转换成单精度
		JOV	OFLO	确保溢出开关断开
		STA	TEMP	
		LD2	TEMP(EXPD)	$rI2 \leftarrow e$
		DEC2	QQ-Q	校正余量中的差
		SLAX	2	去掉指数
		JMP	NORM	规格化,舍入并出口

7. 所有三个程序都给出 0 作为答案当且仅当精确的结果为 0,所以我们不必担忧相对误差的表达式中的 0 分母。加法程序最坏的情况是相当糟糕的:在十进制下,想像如果输入是 1.0000000 和 .9999999,则答案是 b^{-7} 而不是 b^{-8} ;因此极大相对误差 δ_1 是 b^{-1} ,其中 b 是字节大小。

对于乘法和除法,我们可以假定,两个操作数都为正而且有相同的指数 QQ。通过考察图 4,容易限定乘法中的极大误差:当 $uv \geq 1/b$ 时,我们有 $0 \leq uv - u \otimes v < 3b^{-9} + (b-1)b^{-9}$,所以相对误差以 $(b+2)b^{-8}$ 为上限。当 $1/b^2 \leq uv < 1/b$ 时,我们有 $0 \leq uv - u \otimes v < 3b^{-9}$,在这种情况下,相对误差以 $3b^{-9}/uv \leq 3b^{-7}$ 为限。我们取 δ_2 为两个估计中之较大者,即 $3b^{-7}$ 。

除法要求对程序 D 做更仔细的分析。实际上由子程序计算的量是 $\alpha - \delta - b\epsilon((\alpha - \delta'')(\beta - \delta') - \delta''') - \delta_n$,其中 $\alpha = (u_m + \epsilon u_l)/bv_n$, $\beta = v_l/bv_m$,而且非负截断误差 $(\delta, \delta', \delta'', \delta''')$ 分别小于 $(b^{-10}, b^{-5}, b^{-5}, b^{-6})$;最后 δ_n (规格化时的截断误差)是非负的,并依赖于是否出现调整而小于 b^{-9} 或 b^{-8} 。商的实际值是 $\alpha/(1 + b\epsilon\beta) = \alpha - b\epsilon\alpha\beta + b^2\epsilon\alpha\beta^2\delta''''$,其中 δ'''' 是由无穷级数(2)的截断引起的非负误差;这里 $\delta'''' < \epsilon^2 = b^{-10}$,因为它是一个交错级数。因此相对误差是 $(b\epsilon\delta' + b\epsilon\delta''\beta/\alpha + b\epsilon\delta'''/\alpha) \cdot (\delta/\alpha + b\epsilon\delta'\delta''/\alpha + b^2\epsilon\alpha\beta^2\delta'''' + \delta_n/\alpha)$ 的绝对值,乘以 $(1 + b\epsilon\beta)$ 。这个表达式中的正项以 $b^{-9} + b^{-8} + b^{-8}$ 为界,负项以 $b^{-8} + b^{-12} + b^{-8}$ 加上规格化阶段的贡献为界。规格化阶

段所贡献的数量大约为 b^{-7} 。因此显然可能出现的相对误差的最大部分来自规格化阶段,而且 $\delta_3 = (b+2)b^{-8}$ 是相对误差的可靠上限。

8. 加法:如果 $e_u \leq e_v + 1$,则在规格化阶段整个相对误差出现,所以它以 b^{-7} 为上限。如果 $e_u \geq e_v + 2$,而符号又相同,则整个误差可以再次归咎于规格化;如果符号相反,则由于把数字移出寄存器之外而产生的误差同随后在规格化时引进的误差有相反的方向。这两个误差都以 b^{-7} 为限,因此 $\delta_1 = b^{-7}$ (这实际上比习题 7 中的结果更好)。

乘法:与习题 7 类似的分析给出 $\delta_2 = (b+2)b^{-8}$ 。

4.2.4 小节

1. 由于仅当操作数有相同符号时才会出现小数溢出,因此这是小数溢出的概率除以操作数有相同符号的概率,即 $7\% / (\frac{1}{2}(91\%)) \approx 15\%$ 。

3. $\log_{10} 2.4 - \log_{10} 2.3 \approx 1.84834\%$ 。

4. 这些页将是均匀地灰白的。

5. $10f_U \leq r$ 的概率是 $(r-1)/10 + (r-1)/100 + \cdots = (r-1)/9$ 。所以在这情况下前导数字是一致地分布的。例如前导数字 1 以概率 $\frac{1}{9}$ 出现。

6. 有三个前导 0 位的概率是 $\log_{16} 2 = \frac{1}{4}$; 有两个前导 0 位的概率是 $\log_{16} 4 - \log_{16} 2 = \frac{1}{4}$; 其它两种情况与此类似。前导 0 位的“平均”数是 $1\frac{1}{2}$, 所以“有效位”的“平均”数是 $p + \frac{1}{2}$ 。然而最坏情况,即 $p-1$ 位,以较高的概率出现。实际上,通常有必要把误差估计建立在最坏情况的基础上,因为一个计算链的强度只相当于它最弱的环节的强度。在 4.2.2 小节的误差分析中,在浮点十六进制情况下,舍入的相对误差的上限为 2^{1-p} 。在二进制情况下,在所有规格化的数中我们可有 $p+1$ 位有效位(参考习题 4.2.1-3),且舍入的相对误差以 2^{1-p} 为界。广泛的计算经验确认,浮点二进制产生出比等价的浮点十六进制还精确得多的结果,即使当二进制数有 p 个二进制位的精度而不是 $p+1$ 个二进制位时亦然。

表 1 和表 2 说明,十六进制算术可以做得稍快一些,因为当右调或向左规格化时只需要较少的周期。但这个事实同 $b=2$ 相对于其它进制的实质性的优点相比是不足道的(也参见定理 4.2.2C 和习题 4.2.2-13, 15, 21),特别是因为只须通过在总的处理器开销上有一丁点的增加就可把浮点二进制数做成像浮点十六进制那样快。

7. 例如,假设 $\sum_m (F(10^{pm} \cdot 5^k) - F(10^{km})) = \log 5^k / \log 10^k$, 而且假设 $\sum_m (F(10^{km} \cdot 4^k) - F(10^{km})) = \log 4^k / \log 10^k$, 则对所有 k ,

$$\sum_m (F(10^{km} \cdot 5^k) - F(10^{km} \cdot 4^k)) = \log_{10} \frac{5}{4}$$

但现在假设 ϵ 是一个小的正数, 并且选 $\delta > 0$ 使得对 $0 < x < \delta$, $F(x) < \epsilon$, 并选 $M > 0$ 使得对于 $x > M$, $F(x) > 1 - \epsilon$. 我们可以取 k 如此之大, 使 $10^{-k} \cdot 5^k < \delta$ 和 $4^k > M$; 因此由 F 的单调性得

$$\begin{aligned} \sum_m (F(10^{km} \cdot 5^k) - F(10^{km} \cdot 4^k)) &\leq \\ \sum_{m \leq 0} (F(10^{km} \cdot 5^k) - (10^{k(m-1)} \cdot 5^k)) + \sum_{m \geq 0} (F(10^{k(m+1)} \cdot 4^k) - F(10^{km} \cdot 4^k)) &= \\ F(10^{-k} \cdot 5^k) + 1 - F(10^k \cdot 4^k) &< 2\epsilon \end{aligned}$$

8. 当 $s > r$ 时, $P_0(10^r s)$ 对小的 n 为 1, 而当 $\lfloor 10^r s \rfloor > \lfloor 10^n r \rfloor$ 时它为 0. 使得这种情况出现的最小的 n 可能是任意大的, 所以对于与 s 无关的 $N_0(\epsilon)$ 不能给出一致的界来。(一般地说, 微积分课本证明, 这样一个一致的界将意味着极限函数 $S_0(s)$ 是连续的, 但它不是。)

9. 设 q_1, q_2, \dots 使得对所有的 n , $P_0(n) = q_1 \binom{n-1}{0} + q_2 \binom{n-1}{1} + \dots$. 由此得出对所有的 m 和 n , $P_m(n) = 1^{-m} q_1 \binom{n-1}{0} + 2^{-m} q_2 \binom{n-1}{1} + \dots$.

10. 当 $1 < r < 10$ 时, 生成函数 $C(z)$ 在点 $1 + w_n$ 处有简单的极点, 其中 $w_n = 2\pi n i / \ln 10$, 因此

$$C(z) = \frac{\log_{10} r - 1}{1 - z} + \sum_{n \neq 0} \frac{1 + w_n}{w_n} \frac{e^{-w_n \ln r} - 1}{(\ln 10)(z - 1 - w_n)} + E(z)$$

其中 $E(z)$ 在整个平面是解析的。于是如果 $\theta = \arctan(2\pi / \ln 10)$, 则

$$\begin{aligned} c_m = \log_{10} r - 1 - \frac{2}{\ln 10} \sum_{n > 0} \Re \left(\frac{e^{-w_n \ln r} - 1}{w_n (1 + w_n)^m} \right) + e_m = \\ \log_{10} r - 1 - \frac{\sin(m\theta + 2\pi \log_{10} r) - \sin(m\theta)}{\pi(1 + 4\pi^2/(\ln 10)^2)^{m/2}} + O\left(\frac{1}{(1 + 16\pi^2/(\ln 10)^2)^{m/2}}\right) \end{aligned}$$

11. 当 $(\log_b U) \bmod 1$ 一致分布于 $[0, 1)$ 时, $(\log_b 1/U) \bmod 1 = 1 - (\log_b U) \bmod 1$ 亦然。

12. 我们有 $h(z) = \int_{1/b}^z f(x) dx g(z/bx) / bx + \int_z^1 f(x) dx g(z/x) / x$, 因此

$$\begin{aligned} \frac{h(z) - l(z)}{l(z)} &= \int_{1/b}^z f(x) dx \frac{g(z/bx) - l(z/bx)}{l(z/bx)} + \\ &\quad \int_z^1 f(x) dx \frac{g(z/x) - l(z/x)}{l(z/x)} \end{aligned}$$

由于 $f(x) \geq 0$, 对所有 z , $|(h(z) - l(z))/l(z)| \leq \int_{1/b}^z f(x) dx A(g) +$

$\int_z^1 f(x) dx A(g)$, 因此 $A(h) \leq A(g)$ 。由对称性, $A(h) \leq A(f)$ 。[见 Bell System

Tech. J. 49 (1970), 1609, 1625.]

13. 令 $X = (\log_b U) \bmod 1$ 和 $Y = (\log_b V) \bmod 1$, 使 X 和 Y 在 $[0, 1)$ 中独立地一致分布。当且仅当 $X + Y \geq 1$ 时不需要左移, 这种情况以 $\frac{1}{2}$ 的概率出现。

(类似地, 由算法 4.2.1M 实现的浮点除法的结果不需要规格化移位的概率是 $\frac{1}{2}$ 。这仅仅需要两个操作数独立地有相同分布这一较弱的假定就行了。)

14. 为方便起见, 这里给出 $b = 10$ 的计算。如果 $k = 0$, 则一个进位的概率是

$$\left(\frac{1}{\ln 10}\right)^2 \int_{1 \leq x, y \leq 10} \frac{dx}{x} \frac{dy}{y}$$

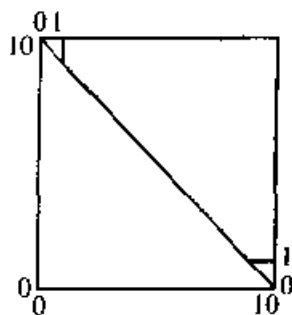


图 A-7

(见图 A-7) 积分的值是

$$\int_0^{10} \frac{dy}{y} \int_{10-y}^{10} \frac{dx}{x} - 2 \int_0^1 \frac{dy}{y} \int_{10-y}^{10} \frac{dx}{x}$$

以及

$$\int_0^1 \frac{dy}{y} \ln\left(\frac{1}{1-y/10}\right) = \int_0^1 \left(\frac{1}{10} + \frac{y}{200} + \frac{y^2}{3000} + \cdots\right) dy = \frac{t}{10} + \frac{t^2}{400} + \frac{t^3}{9000} + \cdots$$

(后一积分实际上是一个“二重对数”。) 因此, 当 $k = 0$ 时一个进位的概率是 $(1/\ln 10)^2 \cdot (\pi^2/6 - 2 \sum_{n \geq 1} 1/n^2 10^n) \approx .27154$ 。[注: 当 $b = 2$ 和 $k = 0$ 时, 小数溢出总出现; 所以这个推导证明 $\sum_{n \geq 1} 1/n^2 2^n = \pi^2/12 - (\ln 2)^2/2$ 。]

当 $k > 0$ 时, 概率是

$$\left(\frac{1}{\ln 10}\right)^2 \int_{10^{-k}}^{10^{1-k}} \frac{dy}{y} \int_{10-y}^{10} \frac{dx}{x} = \left(\frac{1}{\ln 10}\right)^2 \left(\sum_{n \geq 1} \frac{1}{n^2 10^{nk}} - \sum_{n \geq 1} \frac{1}{n^2 10^{n(k+1)}} \right)$$

于是当 $b = 10$ 时, 小数溢出将以近似概率 $.272 p_0 + .017 p_1 + .002 p_2 + \cdots$ 出现。当 $b = 2$ 时, 对应的数字是 $p_0 + .655 p_1 + .288 p_2 + .137 p_3 + .067 p_4 + .033 p_5 + .016 p_6 + .008 p_7 + .004 p_8 + .002 p_9 + .001 p_{10} + \cdots$ 。

现在如果使用来自表 1 的概率除以 .91 以消去零操作数, 并假定诸概率与操作数符号无关, 我们预测当 $b = 10$ 时, 概率大约为 14%, 而不是习题 1 中的 15%。对于 $b = 2$, 我们预计大约是 48%, 而这个表给出 44%。这些结果肯定同实验误差的极限一致。

15. 若 $k = 0$, 则当且仅当有一个进位时前导数字为 1。(当 $b \geq 4$ 时, 小数溢出和随后的舍入会产生一个前导数字 2, 但在本题中我们不考虑舍入。) 如同上题所示, 小数溢出的概率近似于 .272, 而 $.272 < \log_{10} 2$ 。

当 $k > 0$ 时, 前导数字是 1 的概率为

$$\left(\frac{1}{\ln 10}\right)^2 \left(\int_{10^{-k}}^{10^{1-k}} \frac{dy}{y} \int_{\substack{1 \leq x < 2 \\ \text{或 } 10^{-k} \leq x < 10}} \frac{dx}{x} \right) < \left(\frac{1}{\ln 10}\right)^2 \left(\int_{10^{-k}}^{10^{1-k}} \frac{dy}{y} \int_{1 \leq x \leq 2} \frac{dx}{x} \right) = \log_{10} 2$$

16. 为证明提示[这个提示见 Landau, *Prace Matematyczno-Fizyczne* 21 (1910), 103~113], 首先假定 $\limsup a_n = \lambda > 0$. 设 $\epsilon = \lambda/(\lambda + 4M)$ 并选择 N 使得对所有 $n > N$, $|a_1 + \cdots + a_n| < \frac{1}{10} \lambda n$. 设 $n > N/(1 - \epsilon)$, $n > 5/\epsilon$ 使得 $a_n > \frac{1}{2} \lambda$. 则由归纳法, 对于 $0 \leq k < n$, $a_{n-k} \geq a_n - kM/(n - n) > \frac{1}{4} \lambda$, 且 $\sum_{n-n < k \leq n} a_k \geq \frac{1}{4} \lambda (n - 1) > \frac{1}{5} \lambda n$. 但是

$$\left| \sum_{n-n < k \leq n} a_k \right| = \left| \sum_{1 \leq k \leq n} a_k - \sum_{1 \leq k \leq n-n} a_k \right| \leq \frac{1}{5} \lambda n$$

因为 $n - n > N$. 如果 $\liminf a_n < 0$, 则可推出类似的矛盾。

假定当 $n \rightarrow \infty$ 时, $P_{m+1}(n) \rightarrow \lambda$, 命 $a_k = P_m(k) - \lambda$. 如果 $m > 0$, 则 a_k 满足提示的假设(参考等式 4.2.2-(15)), 因为 $0 \leq P_m(k) \leq 1$; 因此 $P_m(n) \rightarrow \lambda$.

17. 参见 *J. Math. Soc. Japan* 4 (1952), 313~322. (从 Cesàro 的一个定理 [*Atti della Reale Accademia dei Lincei, Rendiconti* (4) 4 (1888), 452~457] 可以推出调和概率推广到通常的概率这一事实. Persi Diaconis [Ph. D. thesis Harvard University, 1974] 在其它场合, 已经证明在下列精确的意义下, 由重复地取平均得到概率的定义比之调和概率为弱, 即如果 $\lim_{m \rightarrow \infty} \liminf_{n \rightarrow \infty} P_m(n) = \lim_{m \rightarrow \infty} \limsup_{n \rightarrow \infty} P_m(n) = \lambda$, 则调和概率是 λ . 另一方面, “对于某个整数 $k > 0$, $10^k \leq n < 10^{k^2+k}$ 这一命题有调和概率 $\frac{1}{2}$, 而重复地取平均绝不会来给它以任何特别的概率。”)

18. 对于 $1 \leq a < b$ 设 $p(a) = P(L_a)$ 及 $p(a, b) = \sum_{a \leq k < b} p(k)$. 由于对所有的 a , $L_a = L_{10a} \cup L_{10a+1} \cup \cdots \cup L_{10a+9}$, 由 (i) 我们有 $p(a) = p(10a, 10(a+1))$. 而且, 因为由 (i), (ii), (iii), $P(S) = P(2S) + P(2S+1)$, 我们有 $p(a) = p(2a, 2(a+1))$. 由此得出, 对所有 $m, n \geq 0$, $p(a, b) = p(2^m 10^n a, 2^m 10^n b)$.

如果 $1 < b/a < b'/a'$, 则 $p(a, b) \leq p(a', b')$. 原因是 $\log 2 / \log 10$ 是无理数, 因此存在整数 m, n, m', n' 使得 $2^m 10^n a' \leq 2^m 10^n a < 2^m 10^n b \leq 2^{m'} 10^{n'} b'$, 因此我们可以应用 (v). (对于 $k=1$ 和 $U_n = n \log 2 / \log 10$, 参考习题 3.5-22.) 特别是, $p(a) \geq p(a+1)$, 而且由此得出 $p(a, b)/p(a, b+1) \geq (b-a)/(b+1-a)$. (参考等式 4.2.2-(15).)

现在我们可以证明每当 $b/a = b'/a'$ 时, $p(a, b) = p(a', b')$; 因为对于任意大的 n 值, $p(a, b) = p(10^n a, 10^n b) \leq c_n p(10^n a, 10^n b - 1) \leq c_n p(a', b')$, 其中 $c_n = 10^n(b-a)/(10^n(b-a)-1) = 1 + O(10^{-n})$.

对于任何正整数 n , 我们有 $p(a^n, b^n) = p(a^n, ba^{n-1}) + p(ba^{n-1}, b^2 a^{n-2}) + \cdots + p(b^{n-1} a, b^n) = np(a, b)$. 如果 $10^m \leq a^n \leq 10^{m+1}$ 且 $10^{m'} \leq b^n \leq 10^{m'+1}$, 则由

(v), $p(10m^{m-1}, 10^m) \leq p(a^n, b^n) \leq p(10^m, 10^{m'+1})$ 。但由(iv), $p(1, 10) = 1$, 因此, 对所有的 $m' \geq m$, $p(10^m, 10^{m'}) = m' - m$ 。结论: 对于所有 n , $\lfloor \log_{10} b^n \rfloor - \lfloor \log_{10} a^n \rfloor - 1 \leq np(a, b) \leq \lfloor \log_{10} b^n \rfloor + \lfloor \log_{10} a^n \rfloor + 1$, 而且 $p(a, b) = \log_{10}(b/a)$ 。

[本题受到了 D.I.A.Cohen 的启发, 他证明了一个稍弱的结果, 见 *J. Combinatorial Theory A20* (1976), 367~370。]

19. 等价地, 在定义 3.5B 的意义下 $\langle (\log_{10} F_n) \bmod 1 \rangle$ 是等分布的, 由于由 1.2.8-(14), $\log_{10} F_n = n \log_{10} \phi - \log_{10} \sqrt{5} + O(\phi^{-2n})$, 这等价于由习题 3.5-22 得出的 $\langle n \log_{10} \phi \rangle$ 的等分布。[*Fibonacci Quarterly* 5 (1967), 137~140。] 同样的证明表明序列 $\langle b^n \rangle$, 遵从不是 10 的幂的所有整数 $b > 1$ 的对数定律 [Yaglom 和 Yaglom, *Challenging Problems with Elementary Solutions* (Moscow: 1954; 英译本, 1964), Problem 91b]。

注: 许多其它的整数序列都有这个性质。例如, Persi Diaconis [*Annals of Probability* 5 (1977), 72~81] 证明了, $\langle n! \rangle$ 是这样的序列, 而且在

$$\lim_{n \rightarrow \infty} \frac{1}{n+1} \sum_{k=0}^n [10^f \binom{n}{k} < r] = \log_{10} r$$

的意义下, 二项式系数也服从对数定律。P. Schatte [*Math. Nachrichten* 148 (1990), 137~144] 证明了每当部分商如同在习题 4.5.3-16 中那样有多项式变形的一个重复模式时, 连分数的近似的分母有对数的小数部分。一个有趣的未解决的问题是序列 $\langle 2!, (2!)!, ((2!)!)!, \dots \rangle$ 是否有对数的小数部分; 参见 J.H. Conway 和 M.J.T. Guy, *Eureka* 25 (1962), 18~19。

4.3.1 小节

2. 如果有待加上的第 i 个数是 $u_i = (u_{i(m-1)} \dots u_{i1} u_{i0})_b$, 则使用算法 A, 并把步骤 A2 改动如下。

A2'. [加数字] 置 $w_j \leftarrow (u_{1j} + \dots + u_{mj} + k) \bmod b$, 且 $k \leftarrow \lfloor (u_{1j} + \dots + u_{mj} + k)/b \rfloor$ 。

(k 的极大值是 $m-1$, 所以如果 $m > b$, 步骤 A3 将不得不被改变。)

3.	ENN1	N	1	
	JOV	OFLO	1	确保溢出开关断开
	ENTX	0	1	$k \leftarrow 0$
2H	SLAX	5	N	($rX \equiv k$ 的下一个值)
	ENT3	$M * N, 1$	N	($LOC(u_{ij}) \equiv U + n(i-1) + j$)
3H	ADD	U, 3	MN	$rA \leftarrow rA + u_{ij}$
	JNOV	$* + 2$	MN	

INCX	1	K	进位 1
DEC3	N	MN	对于 $m \geq i \geq 1$ 重复
J3NN	3B	MN	$(r13 \equiv n(i-1) + j)$
STA	$W + N, 1$	N	$w_j \leftarrow rA$
INC1	1	N	
J1N	2B	N	对于 $0 \leq j < n$ 重复
STX	$W + N$	1	存最后进位于 w_n 中

假定 $K = \frac{1}{2}MN$, 则运行时间是 $5.5MN + 7N + 4$ 个周期。

4. 在 A1 之前我们可以做如下的断言: “ $n \geq 1$; 而且对于 $0 \leq i < n, 0 \leq u_i, v_i < b$ 。”在 A2 之前, 我们断言: “ $0 \leq j < n$; 对于 $0 \leq i < n, 0 \leq u_i, v_i < b$; 对于 $0 \leq i < j, 0 \leq w_i < b; 0 \leq k \leq 1$; 而且 $(u_j \cdots u_0)_b + (v_{j-1} \cdots v_0)_b = (kw_{j-1} \cdots w_0)_b$ 。”更确切地说, 后一命题意味着

$$\sum_{0 \leq i < j} u_i b^i + \sum_{0 \leq i < j} v_i b^i = kb^j + \sum_{0 \leq i < j} w_i b^i$$

在 A3 之前, 我们断言: “ $0 \leq j < n$; 对于 $0 \leq i < n, 0 \leq u_i, v_i < b$; 对于 $0 \leq i \leq j, 0 \leq w_i < b; 0 \leq k \leq 1$; 且 $(u_j \cdots u_0)_b + (v_j \cdots v_0)_b = (kw_j \cdots w_n)_b$ 。”在步骤 A3 之后, 我们断言, 对于 $0 \leq i < n, 0 \leq w_i < b; 0 \leq w_n \leq 1$; 且 $(u_{n-1} \cdots u_0)_b + (v_{n-1} \cdots v_0)_b = (w_n \cdots w_0)_b$ 。

只要验证这些断言之间必要的蕴涵关系和证明这个算法总是有穷的, 则完成此证明就只是简单的事情了。

5. B1. 置 $j \leftarrow n - 1, w_n \leftarrow 0$ 。

B2. 置 $t \leftarrow u_j + v_j, w_j \leftarrow t \bmod b, i \leftarrow j$ 。

B3. 如果 $t \geq b$, 则置 $i \leftarrow i + 1, t \leftarrow w_i + 1, w_i \leftarrow t \bmod b$, 并重复这一步骤直到 $t < b$ 。

B4. j 减 1, 而且如果 $j \geq 0$, 则返回 B2。 |

6. C1. 置 $j \leftarrow n - 1, i \leftarrow n, r \leftarrow 0$ 。

C2. 置 $t \leftarrow u_j + v_j$ 。如果 $t \geq b$, 则置 $w_i \leftarrow r + 1$, 而且对于 $i > k > j, w_k \leftarrow 0$; 然后置 $i \leftarrow j$ 和 $r \leftarrow t \bmod b$ 。否则如果 $t < b - 1$, 则置 $w_i \leftarrow r$, 而且对于 $i > k > j, w_k \leftarrow b - 1$; 然后置 $i \leftarrow j$ 和 $r \leftarrow t$ 。

C3. j 减 1。如果 $j \geq 0$, 则返回 C2; 否则 $w_i \leftarrow r$, 而且对于 $i > k \geq 0, w_k \leftarrow b - 1$ 。 |

7. 例如当 $j = n - 3$ 时, $k = 0$ 的概率为 $(b + 1)/2b$; $k = 1$ 的概率为 $((b - 1)/2b) \cdot$

$(1 - 1/b)$, 这即是出现一个进位和前一个数字不是 $b-1$ 的概率; $k=2$ 的概率是 $((b-1)/2b)(1/b)(1-1/b)$; 而且 $k=3$ 的概率是 $((b-1)/2b)(1/b)(1/b)(1)$ 。对于固定的 k , 我们可以随 j 从 $n-1$ 变到 0 而把几个概率加在一起; 这给出进位向后传播 k 个位置的平均次数

$$m_k = \frac{b-1}{2b^k} \left((n+1-k) \left(1 - \frac{1}{b} \right) + \frac{1}{b} \right)$$

作为检验, 我们求得进位的平均次数是

$$m_1 + 2m_2 + \cdots + nm_n = \frac{1}{2} \left(n - \frac{1}{b-1} \left(1 - \left(\frac{1}{b} \right)^n \right) \right)$$

同(6)一致。

8.	ENT1	N-1	1	3H	LDA	W,2	K
	JOV	OFLO	1		INCA	1	K
	STZ	W+N	1		STA	W,2	K
2H	LDA	U,1	N		INC2	1	K
	ADD	V,1	N		JOV	3B	K
	STA	W,1	N	4H	DEC1	1	N
	JNOV	4F	N		J1NN	2B	N
	ENT2	1,1	L				

运行时间依赖于 L , 即使得 $u_j + v_j \geq b$ 的位置数; 并依赖于 K , 进位的总数。不难看出, K 与出现在程序 A 中的那个量是同一个量。正文的分析表明, L 有平均值 $N((b-1)/2b)$, 而且 K 有平均值 $\frac{1}{2}(N - b^{-1} - b^{-2} - \cdots - b^{-n})$ 。所以如果我们忽略阶为 $1/b$ 的项, 则运行时间是 $9N + L + 7K + 3 \approx 13N + 3$ 个周期。

9. 在步骤 A2 中处处都以“ b_j ”代替“ b ”。

10. 如果行 06 和 07 交换, 则我们将几乎总有溢出, 但在 08 处寄存器 A 可能有一负值, 所以这样做行不通。如果交换行 05 和 06 上的指令, 则出现于此程序的溢出序列在某些情况下将稍有不同, 但这程序仍然是对的。

11. 这等价于串的字典序比较: (i) 置 $j \leftarrow n-1$; (ii) 如果 $u_j < v_j$, 则结束 $[u < v]$; 如果 $u_j = v_j$, 且 $j=0$, 则结束 $[u = v]$; 如果 $u_j = v_j$, 且 $j > 0$, 则置 $j \leftarrow j-1$ 并重复(ii); 如果 $u_j > v_j$, 则结束 $[u > v]$ 。这个算法倾向于十分快, 因为在我们遇到 $u_j \neq v_j$ 的情况之前, j 将减少很多的概率通常是最低的。

12. 以 $u_j = 0$ 和 $v_j = w_j$ 使用算法 S。在这个算法末尾将出现另一个“借位”, 对此应不予考虑。

13.	ENN1	N	1	ADD	CARRY	N
	JOV	OFLO	1	JNOV	*+2	N

ENTX	0	1	INCX	1	K
2H STX	CARRY	N	STA	W + N, 1	N
LDA	U + N, 1	N	TNC1	1	N
MUL	V	N	J1N	2B	N
SLC	5	N	STX	W + N	1

运行时间是 $23N + K + 5$ 个周期, 而且 K 大略等于 $\frac{1}{2}N$ 。

14. 关键的归纳论断在步骤 M4 开始时就应该成立; 所有其它的论断都不难由此推出, 它们是: “ $0 \leq i < m; 0 \leq j < n$; 对于 $0 \leq l < m, 0 \leq u_l < b$; 对于 $0 \leq l < n, 0 \leq v_l < b$; 对于 $0 \leq l < j + m, 0 \leq w_l < b; 0 \leq k < b$; 而且在习题 4 答案的记号下

$$(w_{j+m-1} \cdots w_0)_b + kb^{j+j} = u \times (v_{j-1} \cdots v_0)_b + (u_{i-1} \cdots u_0)_b \times v, b^j$$

15. 误差非负而且小于 $(n-2)b^{n-1}$ 。[类似地, 如果我们不考虑对于 $i+j > n+3$ 的诸乘积, 则误差将以 $(n-3)b^{n-2}$ 为界, 等等; 但是, 在某些情况下, 如果我们想要得到真正的舍入结果, 则就必须计算所有的乘积。进一步的分析表明, 通过只做计算全双字长乘积所需要的大约一半的工作, 几乎就总可以得到多精度浮点小数的正确舍入的结果; 而且, 一个简单的检验将找出需要完全精度的稀有情况。参见 W. Krandick 和 J. R. Johnson, *Proc. IEEE Symp. Computer Arithmetic* 11 (1993), 228~233。]

16. S1. 置 $r \leftarrow 0, j \leftarrow n-1$;

S2. 置 $w_j \leftarrow \lfloor (rb + u_j)/v \rfloor, r \leftarrow (rb + u_j) \bmod v$ 。

S3. j 减 1, 而且如果 $j \geq 0$ 则返回 S2。 I

$$17. u/v > u_n b^n / (v_{n-1} + 1) b^{n-1} = b(1 - 1/(v_{n-1} + 1)) > b(1 - 1(b/2)) = b - 2。$$

$$18. (u_n b + u_{n-1}) / (v_{n-1} + 1) \leq u(v_{n-1} + 1) b^{n-1} < u/v。$$

$$19. u - qv \leq u - qv_{n-1} b^{n-1} - qv_{n-2} b^{n-2} = u_{n-2} b^{n-2} + \cdots + u_0 + rb^{n-1} - qv_{n-2} \cdot b^{n-2} < b^{n-2}(u_{n-2} + 1 + rb - qv_{n-2}) \leq 0。因为 u - qv < 0, q < q。$$

20. 如果 $q \leq q - 2$, 则 $u < (q-1)v < q(v_{n-1} b^{n-1} + (v_{n-2} + 1)b^{n-2}) - v < qv_{n-1} b^{n-1} + qv_{n-2} b^{n-2} + b^{n-1} - v \leq qv_{n-1} b^{n-1} + (br + u_{n-2})b^{n-2} + b^{n-1} - v = u_n b^n + u_{n-1} b^{n-1} + u_{n-2} b^{n-2} + b^{n-1} - v \leq u_n b^n + u_{n-1} b^{n-1} + u_{n-2} b^{n-2} \leq u$ 。换言之, $u < u$, 此为矛盾。

21. (由 G. K. Goyal 给出的解) 不等式 $qv_{n-2} \leq br + u_{n-2}$ 意味着我们有 $q \leq (u_n b^2 + u_{n-1} b + u_{n-2}) / (v_{n-1} b + v_{n-2}) \leq u / ((v_{n-1} b + v_{n-2}) b^{n-2})$ 。现在 $u \bmod v = u - qv = v(1 - \alpha)$, 其中 $0 < \alpha = 1 + q - u/v \leq q - u/v \leq u(1 / ((v_{n-1} b + v_{n-2}) \cdot b^{n-2}) - 1/v) = u(v_{n-3} b^{n-3} + \cdots) / ((v_{n-1} b + v_{n-2}) b^{n-2} v) < u / (v_{n-1} b v) \leq q / (v_{n-1} \cdot b) \leq (b-1) / (v_{n-1} b)$, 而这至多是 $2/b$, 因为 $v_{n-1} \geq \frac{1}{2}(b-1)$ 。

22. 设 $u = 4100, v = 588$, 我们首先试一试 $q = \lfloor \frac{41}{5} \rfloor = 8$, 但 $8 \cdot 8 > 10(41 - 40) + 0$ 。然后置 $q = 7$, 现在发现 $7 \cdot 8 < 10(41 - 35) + 0$ 。但 7 乘 588 等于 4116, 所以真正的商是 $q = 6$ 。(顺便指出, 此例表明, 在给定的假设下, 当 $b = 10$ 时, 定理 B 已不能再作改进。)

23. 显然, $v \lfloor b/(v+1) \rfloor < (v+1) \lfloor b/(v+1) \rfloor \leq b$; 而且如果 $v \geq b/2$, 则下限肯定成立, 否则 $v \lfloor b/(v+1) \rfloor \geq v(b-v)/(v+1) \geq (b-1)/2 > \lfloor b/2 \rfloor - 1$ 。

24. 近似的概率仅是 $\log_b 2$, 不是 $\frac{1}{2}$ 。(例如, 如果 $b = 2^{32}$, 则 $v_{n-1} \geq 2^{31}$ 的概率近似于 $\frac{1}{32}$; 这对于保证步骤 D1 和 D8 中对 $d = 1$ 的特殊测试说来, 仍然是足够高的。)

25.	002	ENTA	1	1	
	003	ADD	$V + N - 1$	1	
	004	STA	TEMP	1	
	005	ENTA	1	1	
	006	JOV	1F	1	如果 $v_{n-1} = b - 1$ 则转移
	007	ENTX	0	1	
	008	DIV	$V + N - 1$	1	否则计算 $\lfloor b/(v+1) \rfloor$
	009	JOV	DIVBYZERO	1	如果 $v_{n-1} = 0$ 则转移
	010	1H STA	D	1	
	011	DECA	1	1	
	012	JANZ	$* + 3$	1	如果 $d \neq 1$ 则转移
	013	STZ	$U + M + N$	$1 - A$	置 $u_{m+n} \leftarrow 0$
	014	JMP	D2	$1 - A$	
	015	ENN1	N	A	v 乘以 d
	016	ENTX	0	A	
	017	2H STX	CARRY	AN	
	018	LDA	$V + N, 1$	AN	
	019	MUL	D	AN	
	...				(如同在习题 13 中那样)
	026	J1N	2B	AN	
	027	ENN1	$M + N$	A	(现在 $rX = 0$)
	028	2H STX	CARRY	$A(M + N)$	u 乘以 d
	029	LDA	$U + M + N, 1$	$A(M + N)$	

... (如同在习题 13 中那样)

037	J1N	2B	$A(M+N)$
038	STX	$U+M+N$	A

26. (见习题 16 的算法。)

101	D8	LDA	D	1	(余数将留在 U 到 $U+N-1$ 诸单元中)
102		DECA	1	1	
103		JAZ	DONE	1	如果 $d=1$ 则终止
104		ENT1	$N-1$	A	$r[1] \leftarrow j; j \leftarrow n-1$
105		ENTA	0	A	$r \leftarrow 0$
106	1H	LDX	$U, 1$	AN	$rAX \leftarrow rb + u_j$
107		DIV	D	AN	
108		STA	$U, 1$	AN	
109		SLAX	5	AN	$(u_j, r) \leftarrow (\lfloor rAX/d \rfloor, rAX \bmod d)$
110		DEC2	1	AN	$j \leftarrow j+1$
111		J2NN	1B	AN	对于 $n > j \geq 0$ 重复

这时,除法程序已经完成;而且由下一题, $rAX=0$ 。

27. 它是 $du \bmod dv = d(u \bmod v)$ 。

28. 为方便起见,我们假设 v 有一小数点在左边,即 $v = (v_n, v_{n-1} v_{n-2} \cdots)_b$ 。
在步骤 N1 后我们有 $1/2 \leq v < 1 + 1/b$: 因为

$$v \lfloor \frac{b+1}{v_{n-1}+1} \rfloor \leq \frac{v(b+1)}{v_{n-1}+1} = \frac{v(1+1/b)}{(1/b)(v_{n-1}+1)} < 1 + \frac{1}{b}$$

而且

$$v \lfloor \frac{b+1}{v_{n-1}+1} \rfloor \geq \frac{v(b+1-v_{n-1})}{v_{n-1}+1} \geq \frac{1}{b} \frac{v_{n-1}(b+1-v_{n-1})}{v_{n-1}+1}$$

当 $v_{n-1}=1$ 时,后一个量取它的最小值,因为它是一个凹函数,而且另一个极值要大一些。

步骤 N2 中的公式可重写成 $v \leftarrow \lfloor \frac{b(b+1)}{v_{n-1}+1} \rfloor \frac{v}{b}$, 所以如同我们上边看到的, v 绝不会变成大于等于 $1 + 1/b$ 。

在步骤 N2 迭代一次之后,如果 $t = v_{n-1} + 1$, v 的极小值大于等于

$$\begin{aligned} \left(\frac{b(b+1)-v_{n-1}}{v_{n-1}+1} \right) \frac{v}{b} &\geq \left(\frac{b(b+1)-v_{n-1}}{v_{n-1}+1} \right) \frac{v_{n-1}}{b^2} = \\ &\left(\frac{b(b+1)+1-t}{t} \right) \left(\frac{t-1}{b^2} \right) = 1 + \frac{1}{b} + \frac{2}{b^2} - \frac{1}{b^2} \left(t + \frac{b(b+1)+1}{t} \right) \end{aligned}$$

当 $t = b/2 + 1$ 时出现这个量的极小值;下限为 $1 - 3/2b$ 。因此在步骤 N2 迭代一次之后, $v_{n-1} \geq b - 2$ 。最后,当 $b \geq 5$ 时,我们有 $(1 - 3/2b)(1 + 1/b)^2 > 1$,所以至多需要增加两次迭代。当 $b < 5$ 时,这个断言是容易验证的。

29. 真的,因为 $(u_{j+n} \cdots u_j)_b < v_0$ 。

30. 在算法 A 和 S 中,如果这些算法略微改写一下,则这样的重叠是可能的;例如,在算法 A 中,我们可以这样来改写步骤 A2:“置 $t \leftarrow u_j + v_j + k$, $w_j \leftarrow t \bmod b$, $k \leftarrow \lfloor t/b \rfloor$ 。”

在算法 M 中, v_j 和 w_{j+n} 可以在相同的单元中。在算法 D 中(如同在习题 26 的程序 D 中那样),最方便的是命 $r_{n-1} \cdots r_0$ 和 $u_{n-1} \cdots u_0$ 一样;而且假设在步骤 D6 中不改变 u_{j+n} 的值,则我们也可以有 $q_m \cdots q_0$ 和 $u_{m+n} \cdots u_n$ 一样。(程序 D 的行 098 可安全地改变为“J1N 2B”,因为 u_{j+n} 在后面的计算中不使用。)

31. 如同在算法 D 中那样,对于 $u = (u_{j+n} \cdots u_{j+1} u_j)_3$ 考虑图 6 的情况。如果 u 和 v 的前导非零数字有相同的符号,则置 $r \leftarrow u - v$, $q \leftarrow 1$;否则置 $r \leftarrow u + v$, $q \leftarrow -1$ 。现在如果 $|r| > |u|$,或如果 $|r| = |u|$ 且 $u_{j-1} \cdots u_0$ 的头一个非零数字有和 r 的头一个非零数字同样的符号,则置 $q \leftarrow 0$;否则置 $u_{j+n} \cdots u_j$ 为 r 的诸数字。

32. 参见 M.Nadler, CACM 4 (1961), 192 ~ 193; Z.Pawlak 和 A.Wakulicz, Bull.del'Acad.Polonaire des Sciences, Class III, 5 (1957), 233 ~ 236 (也参见 803 ~ 804);以及习题 4.1-15。

34. 例如,参见 R.E.Maeder, The Mathematica Journal 6, 2 (1996 年春), 32 ~ 40; 6, 3 (1996 年夏), 37 ~ 43。

36. 以 $\pm 2^{-2^n}$ 的精度给定 ϕ , 我们可以通过减法逐次计算 $\phi^{-1}, \phi^{-2}, \dots$ 直到 $\phi^{-k} < 2^{-n}$; 累计误差将不超过 2^{1-n} 。然后我们使用级数 $\ln \phi = \ln((1 + \phi^{-3})/(1 - \phi^{-3})) = 2(\phi^{-3} + \frac{1}{3}\phi^{-9} + \frac{1}{5}\phi^{-15} + \dots)$ 。[参见由 C.G.Knott 编的 Napier Tercentenary Memorial (London: Longmans, 1915) 中 William Schooling 的文章, 337 ~ 344。] 1965 年由 J.W.Wrench, Jr. 提出的一个更好的过程, 是计算

$$\ln \phi = \frac{1}{2} \ln((1 + 5^{-1/2})/(1 - 5^{-1/2})) = (2\phi - 1)(5^{-1} + \frac{1}{3}5^{-2} + \frac{1}{5}5^{-3} + \dots)$$

37. 令 $d = 2^e$ 使得 $b > dv_{n-1} \geq b/2$ 。代替在步骤 D1 来对 u 和 v 进行规格化, 我们通过左移 e 个二进位来计算 $2^e(v_{n-1}v_{n-2}v_{n-3})_b$ 的两个前导数字 $v'v''$ 。在步骤 D3 中, 使用 (v', v'') 来代替 (v_{n-1}, v_{n-2}) 和使用 (u', u'', u''') 来代替 $(u_{j+n}, u_{j+n-1}, u_{j+n-2})$, 其中数字 $u'u''u'''$ 是由 $(u_{j+n} \cdots u_{j+n-3})_b$ 通过左移 e 个二进位得到的。在步骤 D8 中省略除以 d 的除法。(实质上, u 和 v 被“虚拟地”移位。当相对于 n, m 很小时, 这个方法节省计算。)

38. 置 $k \leftarrow n$, $r \leftarrow 0$, $s \leftarrow 1$, $t \leftarrow 0$, $w \leftarrow u$; 我们将保持不变量关系 $uv = 2^{2k}(r + s^2 - s) + 2^{2k-n}t + 2^{2k-2n}vw$ 连同 $0 \leq t, w < 2^n$ 以及连同 $0 < r \leq 2s$, 除非 $(r, s) = (0, 1)$ 。当 $k > 0$ 时, 设 $4w = 2^n w' + w''$ 和 $4t + w'v = 2^n t' + t''$, 其中 $0 \leq w'', t'' < 2^n$ 。

和 $0 \leq t' \leq 6$; 然后置 $t \leftarrow t'', w \leftarrow w'', s \leftarrow 2s, r \leftarrow 4r + t' - s, k \leftarrow k - 1$ 。如果 $r \leq 0$, 则置 $s \leftarrow s - 1$ 和 $r \leftarrow r + 2s$; 否则如果 $r > 2s$, 则置 $r \leftarrow r - 2s$, 以及 $s \leftarrow s + 1$ (这个校正可能要做两次)。重复直到 $k = 0$ 。然后 $uv = r + s^2 - s$, 因为 w 总是 2^{2n-2k} 的倍数。因此 $r = 0$ 当且仅当 $uv = 0$; 否则答案是 s , 因为 $uv - s \leq s^2 \leq uv + s$ 。

39. 令 $S_j = \sum_{k \geq 0} 16^{-k} / (8k + j)$ 。我们要知道是否 $2^{n-1} \pi \bmod 1 < \frac{1}{2}$ 。由于 $\pi = 4S_1 - 2S_4 - S_5 - S_6$, 因此只须有对 $2^{n-1} S_j \bmod 1$ 的好的估计就足够了。现在 $2^{n-1} S_j$ 和 $\sum_{0 \leq k < n/4} a_{njk} / (8k + j) + \sum_{k \geq n/4} 2^{n-1-4k} / (8k + j)$ 同余 (modulo 1), 其中 $a_{njk} = 2^{n-1-4k} \bmod (8k + j)$ 。头一个和中的每一项可以通过在 $O(\log n)$ 个运算之内计算 a_{njk} 而近似到 2^{-m} 的范围内 (4.6.3 小节), 而后求出调整了的商 $\lfloor 2^m a_{njk} / (8k + j) \rfloor$ 。第二个和可以通过计算 2^m 乘以它的头 $m/4$ 项而被近似到 2^{-m} 之内。如果 $m \approx 2 \lg n$, 则不确定的范围将是 $\approx 1/n$, 而且这几乎总是足够精确的。[Math. Comp. 66 (1997), 903~913。]

注: 设 $\zeta = e^{\pi i/4} = (1+i)/\sqrt{2}$ 是 1 的第 8 个根, 并考虑值 $l_j = \ln(1 - \zeta^j/\sqrt{2})$ 。于是 $l_0 = \ln(1 - 1/\sqrt{2}), l_1 = \bar{l}_7 = \frac{1}{2} \ln \frac{1}{2} - i \arctan 1, l_2 = \bar{l}_6 = \frac{1}{2} \ln \frac{3}{2} - i \arctan(1/\sqrt{2}), l_3 = \bar{l}_5 = \frac{1}{2} \ln \frac{5}{2} - i \arctan(1/3), l_4 = \ln(1 + 1/\sqrt{2})$ 。还有由 1.2.9-(13), 对于 $1 \leq j \leq 8, -S_j/2^{j/2} = \frac{1}{8}(l_0 + \zeta^{-j} l_1 + \dots + \zeta^{-7j} l_7)$ 。因此 $4S_1 - 2S_4 - S_5 - S_6 = 2l_0 - (2 - 2i)2l_1 + 2l_4 + (2 + 2i)l_7 = \pi$ 。其它有趣的恒等式是

$$\ln 2 = S_2 + \frac{1}{2} S_4 + \frac{1}{4} S_6 + \frac{1}{8} S_8$$

$$\ln 3 = 2S_2 + \frac{1}{2} S_6$$

$$\ln 5 = 2S_2 + 2S_4 + \frac{1}{2} S_6$$

$$\sqrt{2} \ln(\sqrt{2} + 1) = S_1 + \frac{1}{2} S_3 + \frac{1}{4} S_5 + \frac{1}{8} S_7$$

$$\sqrt{2} \arctan(1/\sqrt{2}) = S_1 - \frac{1}{2} S_3 + \frac{1}{4} S_5 - \frac{1}{8} S_7$$

$$\arctan(1/3) = S_1 - S_2 - \frac{1}{2} S_4 - \frac{1}{4} S_5$$

$$0 = 8S_1 - 8S_2 - 4S_3 - 8S_4 - 2S_5 - 2S_6 + S_7$$

一般说来, 我们有

$$\sum_{k \geq 0} \frac{z^{8k+1}}{8k+1} = A + B + C + D, \quad \sum_{k \geq 0} \frac{z^{8k+5}}{8k+5} = A - B + C - D$$

$$\sum_{k \geq 0} \frac{z^{8k+3}}{8k+3} = A - B - C + D, \quad \sum_{k \geq 0} \frac{z^{8k+7}}{8k+7} = A + B - C - D$$

其中

$$A = \frac{1}{8} \ln \frac{1+z}{1-z}, \quad B = \frac{1}{2^{7/2}} \ln \frac{1+\sqrt{2}z+z^2}{1-\sqrt{2}z+z^2}$$

$$C = \frac{1}{4} \arctan z, \quad D = \frac{1}{2^{5/2}} \arctan \frac{\sqrt{2}z}{1-z^2}$$

而且

$$\sum_{k \geq 0} \frac{z^{mk+a}}{mk+a} = -\frac{1}{m} (\ln(1-z) + (-1)^a [m \text{ 偶}] \ln(1+z) + f_{am}(z))$$

$$f_{am}(z) = \sum_{k=1}^{L(m)/2} \left(\cos \frac{2\pi ka}{m} \ln \left(1 - 2z \cos \frac{2\pi k}{m} + z^2 \right) - 2 \sin \frac{2\pi ka}{m} \arctan \frac{z \sin(2\pi k/m)}{1 - z \cos(2\pi k/m)} \right)$$

40. 为了获得最高有效的 $n/2$ 个单元, 我们需要大约 $\sum_{k=1}^{n/2} \approx \frac{1}{8} n^2$ 基本运算(参见习题 15)。而且当 b 是 2 的幂时, 通过使用 b -adic 的方法我们可以得到最低有效的 $n/2$ 个单元(参见习题 4.1-31); 问题可以很容易地归结到 v 是奇数的情况。令 $u = (\cdots u_2 u_1 u_0)_b$, $v = (\cdots v_2 v_1 v_0)_b$ 以及 $w = (\cdots w_2 w_1 w_0)_b$, 其中我们要来解 $u = vw \pmod{b^{n/2}}$ 。计算 v' 使得 $v'v \pmod{b} = 1$ (参见习题 4.5.2-17)。于是 $w_0 = v'u_0 \pmod{b}$, 而且我们可以计算 $u' = u - w_0 v$, $w_1 = v'u'_0 \pmod{b}$, 等等。在进行了大约 $\frac{1}{8} n^2$ 个基本运算之后, 找到最右的 $n/2$ 个单元。所以总共为 $\frac{1}{4} n^2 + O(n)$, 而算法 D 需要大约 $n^2 + O(n)$ 。对于所有 n 个数字一个纯粹从右至左的方法将需要 $\frac{1}{2} n^2 + O(n)$ 。[参见 T. Jebelean, *J. Symbolic Comp.* **15** (1993), 169 ~ 180; A. Schönhage 和 E. Vetter, *Lecture Notes in Comp. Sci.* **855** (1994), 448 ~ 459。]

41. a) 如果 $m = 0$, 令 $v = u$, 否则从 $(u_{m+n-1} \cdots u_1 u_0)_b$ 减去 xw , 其中 $x = u_0 w' \pmod{b}$; 这使 1 的数字变 0。所以我们已经有效地把 m 减小 1。(这个运算同在 b -adic 算术中的 u/w 的计算紧密相关, 因为对于某个整数 q , $u/w = q + b^m v/w$; 参见习题 4.1-31。它胜过通常的除法, 因为我们绝不需要校正一个试验因子。)

b) 把 a) 应用于乘积 uv 上。如果我们把乘法和取模交错在一起如下, 则可以保留存储空间。置 $k \leftarrow 0, t \leftarrow 0$ 。然后在 $k < n$ 时, 通过置 $t \leftarrow t + u_k v, t \leftarrow (t - xw)/b$, $k \leftarrow k + 1$, 保持不变关系 $b^k t \equiv (u_{k-1} \cdots u_0) v \pmod{w}$, 其中 $x = t_0 w' \pmod{b}$ 被选择成使 $t - xw$ 是 b 的倍数。这个解假定 t, u 和 v 有一个带符号的数量表示; 如同由 Shand 和 Vuillemin 及由 Kornerup 所讨论的那样 [IEEE Symp. Computer Arithmetic **11** (1993), 252 ~ 259, 277 ~ 283], 我们也可对小于 $2w$ 的非负数或对于补码记号进行工作。如果 n 很大, 则 4.3.3 小节的技术可加快乘法。

c) 通过一个内部值 $r(u)$ 来表示同余于 $u \pmod{w}$ 的所有数。然后加法和减法像通常那样处理, 而乘法是 $r(uv) = \text{bmult}(r(u), r(v))$, 其中 bmult 是 b 的运

算。在计算开始时,使用预先计算的常数 $a = b^{2^n} \bmod w$, 以 $r(u) = \text{bmult}(u, a)$ 代替每一操作数 u 。在结束时,以 $u = \text{bmult}(r(u), 1)$ 代替每一个 $r(u)$ 。[在应用于 4.5.4 小节的 RSA 密码中时,我们可以重新定义编码方案使得预先的计算和事后的计算都无必要。]

42. 由 J.M.Holte 在 AMM 104 (1997), 138 ~ 149 上给出的一个有趣分析确立了精确的公式

$$P_{nk} = \frac{1}{m!} \sum_j \left[\begin{matrix} m \\ m-j \end{matrix} \right] b^{-jn} \sum_{r=0}^k \binom{m+1}{r} (k+1-r)^{m-j}$$

当 $j=0$ 时,内部的求和是 $\sum_{r=0}^k (-1)^r \binom{m+1}{r} (k+1-r)^m = \langle \frac{m}{k} \rangle$ 。(习题 5.1.3 - 25 说明为什么在这个联系中会出现欧拉数。)

43. 由习题 1.2.4-35 我们有 $w = \lfloor W/2^{16} \rfloor$, 其中 $W = (2^8 + 1)t = (2^8 + 1)(uv + 2^7)$ 。因此如果 $xy/255 > c + \frac{1}{2}$, 我们有 $c < 2^8$, 因此 $w \geq \lfloor (2^{16}(c+1) + 2^8 - c)/2^{16} \rfloor \geq c+1$; 如果 $xy/255 < c + 1/2$, 我们有 $w \leq \lfloor (2^{16}(c+1) - c - 1)/2^{16} \rfloor = c$ 。[参见 J.F.Blinn, IEEE Computer Graphics and Applic. 14, 6 (1994 年 11 月), 78~82。]

4.3.2 小节

1. 解是惟一的, 因为 $7 \cdot 11 \cdot 13 = 1001$ 。定理 C 的构造性证明告诉我们, 答案是 $((11 \cdot 13)^6 + 6 \cdot (7 \cdot 13)^{10} + 5 \cdot (7 \cdot 11)^{12}) \bmod 1001$ 。但这个答案也许不够明显! 由 (24), 我们有 $v_1 = 1, v_2 = (6-1) \cdot 8 \bmod 11 = 7, v_3 = ((5-1) \cdot 2 - 7) \cdot 6 \bmod 13 = 6$, 所以 $u = 6 \cdot 7 \cdot 11 + 7 \cdot 7 + 1 = 512$ 。

2. 否。至多有一个这样的 u ; 附加条件 $u_1 \equiv \cdots \equiv u_r \pmod{1}$ 是必要和充分的, 而且由此知道这样一种推广不是非常有趣的。

3. $u \equiv u_i \pmod{m_i}$ 意味着 $u \equiv u_i \pmod{\gcd(m_i, m_j)}$, 所以如果有一个解的话, 条件 $u_i \equiv u_j \pmod{\gcd(m_i, m_j)}$ 必须确实成立。而且, 如果对所有 j 有 $u \equiv v \pmod{m_j}$, 则 $u - v$ 是 $\text{lcm}(m_1, \dots, m_r) = m$ 的倍数; 因此至多有一个解。

现在通过计算满足条件 $0 \leq u_j < m_j$ 且 $u_i \equiv u_j \pmod{\gcd(m_i, m_j)}$ 的不同 r 元组 (u_1, \dots, u_r) 的个数, 可以用一种非构造方式来完成这一证明。如果这个数是 m , 则必有一解, 因为当 u 由 a 变成 $a + m - 1$ 时, $(u \bmod m_1, \dots, u \bmod m_r)$ 取 m 个不同的值。假定已选定 u_1, \dots, u_{r-1} 满足给定的条件, 我们现在必须对于 $1 \leq j < r$ 挑选 $u_r \equiv u_j \pmod{\gcd(m_j, m_r)}$, 而且由 $r-1$ 个元素的推广的中国剩余定理, 有

$$m_r / \text{lcm}((m_1, m_r), \dots, \gcd(m_{r-1}, m_r)) = m_r / \gcd(\text{lcm}(m_1, \dots, m_{r-1}), m_r) = \text{lcm}(m_1, \dots, m_r) / \text{lcm}(m_1, \dots, m_{r-1})$$

种方式来做这件事。[这个证明是以 4.5.2 小节的恒等式 (10), (11), (12) 和 (14) 为基础的。]

推广(25)的一个构造性证明[A. S. Fraenkel, *Proc. Amer. Math. Soc.* **14** (1963), 790~791]如下: 设 $M_j = \text{lcm}(m_1, \dots, m_j)$; 我们希望求 $u = v_r M_{r-1} + \dots + v_2 M_1 + v_1$, 其中 $0 \leq v_j < M_j/M_{j-1}$ 。假定 v_1, \dots, v_{j-1} 已经确定, 则我们必须解同余式

$$v_j M_{j-1} + v_{j-1} M_{j-2} + \dots + v_1 \equiv u_j \pmod{m_j}$$

根据假设, 对于 $i < j$, 这里 $v_{j-1} M_{j-2} + \dots + v_1 \equiv u_i \pmod{\text{gcd}(m_i, m_j)}$, 所以 $c = u_j - (v_{j-1} M_{j-2} + \dots + v_1)$ 是

$$\text{lcm}(\text{gcd}(m_1, m_j), \dots, \text{gcd}(m_{j-1}, m_j)) = \text{gcd}(M_{j-1}, m_j) = d_j$$

的倍数。因此我们必须解 $v_j M_{j-1} \equiv c \pmod{m_j}$; 由欧几里得算法知有一个数 c_j , 使得 $c_j M_{j-1} = d_j \pmod{m_j}$; 因此我们可以取

$$v_j = (c_j c) / d_j \pmod{m_j / d_j}$$

注意, 如同在非构造性证明中那样, 我们有 $m_j / d_j = M_j / M_{j-1}$ 。

4. (在 $m_4 = 91 = 7 \cdot 13$ 之后, 我们已经用完了所有小于 100 的两个或多个奇素数的乘积, 所以 m_5, \dots 都必须是素数。)

$$\begin{aligned} m_7 &= 79, & m_8 &= 73, & m_9 &= 71, & m_{10} &= 67, & m_{11} &= 61 \\ m_{12} &= 59, & m_{13} &= 53, & m_{14} &= 47, & m_{15} &= 43, & m_{16} &= 41 \\ m_{17} &= 37, & m_{18} &= 31, & m_{19} &= 29, & m_{20} &= 23, & m_{21} &= 17 \end{aligned}$$

到这里就做不下去了 ($m_{22} = 1$ 不好)。

5. 否。如果我们选择 $m_1 = 3^4, m_2 = 5^2$, 等等, 则明显的上界

$$3^4 5^2 7^2 11^1 \dots = \prod_{\substack{p \text{ 奇} \\ p \text{ 素}}} p^{\lfloor \log_p 100 \rfloor}$$

就达到了。(然而, 当 r 固定时, 要极大化 $m_1 \dots m_r$; 或者当使用模 $2^e - 1$ 时, 要极大化满足 e_j 互素的 $e_1 + \dots + e_r$, 那就更为困难了。)以 256 取代 100 并允许偶数模将得出 $2^8 3^5 5^3 \dots 251^1 \approx 1.67 \cdot 10^{109}$ 。

6. a) 如果 $e = f + kg$, 则 $2^e = 2^f (2^g)^k \equiv 2^f \cdot 1^k \pmod{2^g - 1}$ 。所以如果 $2^e \equiv 2^f \pmod{2^g - 1}$, 则我们有 $2^{e \bmod g} \equiv 2^{f \bmod g} \pmod{2^g - 1}$; 而且由于后边的量处于 0 和 $2^g - 1$ 之间, 我们必有 $e \bmod g = f \bmod g$ 。b) 由 a), $(1 + 2^d + \dots + 2^{(c-1)d}) \cdot (2^e - 1) \equiv (1 + 2^d + \dots + 2^{(c-1)d}) \cdot (2^d - 1) = 2^{cd} - 1 \equiv 2^{ce} - 1 \equiv 2^1 - 1 = 1 \pmod{2^f - 1}$ 。

7. 由(23), (25)和(26)我们有 $v_j m_{j-1} \dots m_1 \equiv u_j - (v_{j-1} m_{j-2} \dots m_1 + \dots + v_1)$ 和 $C_j m_{j-1} \dots m_1 \equiv 1 \pmod{m_j}$; 参见 P. A. Pritchard, *CACM* **27** (1984), 57。

重写这些公式的这个方法使用同样数量的算术运算和较少的常数; 但是仅当我们把模数排序成 $m_1 < m_2 < \dots < m_r$ 时常数的个数才是较少的, 否则我们将需要 $m_i \bmod m_j$ 的一个表。模数的这一顺序看来可能比我们使 m_1 最大, 而使 m_2 次大等等, 要求更多的计算, 因为比起模 m_1 来, 模 m_r 有多得多的运算要做; 但由于 v_j 可以和 $m_j - 1$ 一样大, 在(24)中对于 $m_1 < m_2 < \dots < m_r$ 我们也会好些。所以尽管

4.3.3B 小节表明当这些模数有(14)的形式时正文中的公式是有利的,但是这一思想看来比正文中的公式是更可取的。

$$\begin{aligned} 8. \text{ 模 } m_j: m_{j-1} \cdots m_1 v_j &\equiv m_{j-1} \cdots m_1 (\cdots ((u_j - v_1) c_{1j} - v_2) c_{2j} - \cdots - v_{j-1}) \times \\ &c_{(j-1)j} \equiv m_{j-2} \cdots m_1 (\cdots (u_j - v_1) c_{1j} - \cdots - v_{j-2}) c_{(j-2)j} - v_{j-1} m_{j-2} \cdots m_1 \equiv \cdots \equiv u_j \\ &- v_1 - v_2 m_1 - \cdots - v_{j-1} m_{j-2} \cdots m_1. \end{aligned}$$

$$9. u_r \leftarrow ((\cdots (v_r m_{r-1} + v_{r-1}) m_{r-2} + \cdots) m_1 + v_1) \bmod m_r, \cdots,$$

$$u_2 \leftarrow (v_2 m_1 + v_1) \bmod m_2, u_1 \leftarrow v_1 \bmod m_1.$$

(如果我们要让 u_j 和 v_j 共享同一存储单元,如同在(24)中那样,则计算应按这个顺序进行。)

10. 如果我们重新定义“mod”运算符,使得它产生在对称范围中的余数,则关于算术运算的基本公式(2),(3)和(4)以及关于转换的公式(24),(25)保持不变,而(25)中的数 u 处于所希望的范围(10)中。(这里(25)是一个平衡的混合进制记法,推广了“平衡的三进制”记法。)这两个数的比较,仍以正文中所述的简单方式从左到右进行。而且,如果我们在计算机内有带符号的量的表示,即使 m_j 几乎是字大小的两倍,仍有可能以一个计算机字来保留值 u_j 。但是类似于(11)和(12)的算术运算是更困难的,所以,看起来,在大多数计算机上,按这个思想运算时速度会慢一些。

11. 乘以 $\frac{1}{2}(m+1) = \left\{ \frac{1}{2}(m_1+1), \cdots, \frac{1}{2}(m_r+1) \right\}$ 。注意, $2t \cdot \frac{m+1}{2} \equiv t \pmod{m}$ 。一般说来,如果 v 与 m 互素,则我们(通过欧几里得算法)可以找到一个数 $v' = (v'_1, \cdots, v'_r)$ 使得 $vv' \equiv 1 \pmod{m}$; 然后如果已知 u 是 v 的倍数。则我们有 $u/v = uv'$, 其中后者是通过模的乘法计算的。当 v 不与 m 互素时,除法要困难得多。

12. 在(11)中以 m 代替 m_j 。[如果 m 是奇数,则检验溢出的另一个方法是保持额外的二进位 $u_0 = u \bmod 2$ 和 $v_0 = v \bmod 2$ 。那么溢出出现当且仅当 $u_0 + v_0 \neq w_1 + \cdots + w_r \pmod{2}$, 其中 (w_1, \cdots, w_r) 是对应于 $u+v$ 的混合进制的数字。]

13. a) 对于 $p=2$ 和 $5, x^2 - x = (x-1)x \equiv 0 \pmod{10^n}$ 等价于 $(x-1)x \equiv 0 \pmod{p^n}$ 。 x 或者 $x-1$ 必须是 p 的倍数,而另一个同 p^n 互素;所以 x 或者 $x-1$ 必为 p^n 的倍数。如果 $x \bmod 2^n = x \bmod 5^n = 0$ 或 1 , 则我们必然有 $x \bmod 10^n = 0$ 或 1 ; 因此自守有 $x \bmod 2^n \neq x \bmod 5^n$ 。 b) 如果 $x = qp^n + r$, 其中 $r=0$ 或 1 , 则 $r \equiv r^2 \equiv r^3$, 所以 $3x^2 - 2x^3 \equiv (6qp^n r + 3r) - (6qp^n r + 2r) \equiv r \pmod{p^{2n}}$ 。 c) 设 c' 是 $(3(cx)^2 - 2(cx)^3)/x^2 = 3c^2 - 2c^3 x$ 。

注: 由于一个 n 位数字的自守的最后 k 位数字形成一个 k 位数字的自守, 所以就可以谈论两个 ∞ 位数 x 和 $1-x$ 的自守, 它们是 10 -adic 数(参考 4.1-31)。在模算术的情况下, 10 -adic 数的集合等价于有序偶 (u_1, u_2) 的集合, 其中 u_1 是 2 -adic 数, u_2 是 5 -adic 数。

14. 求浮点近似 $(a_0 u_0, a_1 u_1, \dots, a_{n-1} u_{n-1})$ 和 $(a_0 v_0, a_1 v_1, \dots, a_{n-1} v_{n-1})$ 的循环卷积 $(z_0, z_1, \dots, z_{n-1})$, 其中常数 $a_k = 2^{-(kq \bmod n)/n}$ 已经预先计算出来。恒等式 $u = \sum_{k=0}^{n-1} u_k a_k 2^{kq/n}$ 和 $v = \sum_{k=0}^{n-1} v_k a_k 2^{kq/n}$ 现在意味着 $w = \sum_{k=0}^{n-1} t_k a_k 2^{kq/n}$, 其中 $t_k \approx z_k/a_k$ 。如果已经维持了充分的精度, 则每个 t_k 将非常接近于一个整数。从这些整数可以容易地找到 w 的表示。[R. Crandall 和 B. Fagin, *Math. Comp.* 62 (1994), 305 ~ 324.]

4.3.3 小节

1.	$12 \times 23:$	$34 \times 41:$	$22 \times 18:$	$1234 \times 2341:$
	02	12	02	0276
	02	12	02	0276
	-01	+03	+00	-0396
	06	04	16	1394
	06	04	16	1394
	<u>0276</u>	<u>1394</u>	<u>0396</u>	<u>2888794</u>

2. $\sqrt{Q + \lfloor \sqrt{Q} \rfloor} \leq \sqrt{Q} + \sqrt{Q} < \sqrt{Q + 2\sqrt{Q} + 1} = \sqrt{Q} + 1$, 所以 $\lfloor \sqrt{Q + R} \rfloor \leq \lfloor \sqrt{Q} \rfloor + 1$.

3. 当 $k \leq 2$ 时, 结果为真, 所以假设 $k > 2$ 。设 $q_k = 2^{Q_k}$, $r_k = 2^{R_k}$, 使得 $R_k = \lfloor \sqrt{Q_k} \rfloor$ 和 $Q_k = Q_{k-1} + R_{k-1}$ 。我们必须证明 $1 + (R_k + 1)2^{R_k} \leq 2^{Q_{k-1}}$; 这个不等式一点也不相靠近。一条路子是观察当 $k > 2$ 时, $1 + (R_k + 1)2^{R_k} \leq 1 + 2^{2R_k}$ 和 $2R_k < Q_{k-1}$ 。($2R_k < Q_{k-1}$ 的事实容易用归纳法说明, 因为 $R_{k+1} - R_k \leq 1$, 而且 $Q_k - Q_{k-1} \geq 2$ 。)

4. 对于 $j = 1, \dots, r$, 计算 $U_e(j^2), jU_o(j^2), V_e(j^2), jV_o(j^2)$; 而且通过递归地调用乘法算法, 计算

$$W(j) = (U_e(j^2) + jU_o(j^2))(V_e(j^2) + jV_o(j^2))$$

$$W(-j) = (U_e(j^2) - jU_o(j^2))(V_e(j^2) - jV_o(j^2))$$

然后我们有 $W_e(j^2) = \frac{1}{2}(W(j) + W(-j))$, $W_o(j^2) = \frac{1}{2}(W(j) - W(-j))$ 。再计算 $W_e(0) = U(0)V(0)$ 。现在构造 W_e 和 W_o 的差值表, W_e 和 W_o 分别是次数为 r 和 $r-1$ 的多项式。

这个方法减少了所处理的数的大小, 也减少了加法和乘法的次数。它惟一的缺点是程序较长(因为控制略微复杂些, 而且有些计算必须以带符号的数来进行)。

另一个可能性也许是在 $1^2, 2^2, 4^2, \dots, (2^r)^2$ 处计算 W_e 和 W_o ; 尽管涉及的这些

数更大,但计算却更快,因为所有的乘法皆代之以位移,而所有除法都通过形如 $2^j/(2^k-1)$ 的二进数进行(对于以这样的数做除法,有简单的过程可利用)。

5. 以足够大的 q_0 和 q_1 开始 q 和 r 序列,使得习题3中的不等式成立。然后我们将在类似于前边定理B的一些公式中发现有 $\eta_1 \rightarrow 0$ 和 $\eta_2 = (1 + 1/(2r_k)) \cdot 2^{1+\sqrt{2Q_k}-\sqrt{2Q_{k+1}}}(Q_k/Q_{k+1})$ 。当 $k \rightarrow \infty$ 时因子 $Q_k/Q_{k+1} \rightarrow 1$, 所以如果要证明对所有很大的 k , $\eta_2 < 1 - \epsilon$, 我们可以忽略它。现在 $\sqrt{2Q_{k+1}} = \sqrt{2Q_k + 2\lceil\sqrt{2Q_k}\rceil + 2} \geq \sqrt{(2Q_k + 2\sqrt{2Q_k} + 1) + 1} \geq \sqrt{2Q_k} + 1 + 1/(3R_k)$ 。因此 $\eta_2 \leq (1 + 1/(2r_k))2^{-1/(3R_k)}$, 且对于充分大的 k , $\lg \eta_2 < 0$ 。

注:也可以修改算法T,以定义一个类似类型的序列 q_0, q_1, \dots , 它以 n 为基础,使得在步骤 T_1 之后, $n \approx q_k + q_{k+1}$ 。这个修改导致了估计(21)。

6. $6q + d_1$ 和 $6q + d_2$ 的任何公因子必须也整除它们的差 $d_2 - d_1$ 。 $\binom{6}{2}$ 个差是 2, 3, 4, 6, 8, 1, 2, 4, 6, 1, 3, 5, 2, 4, 2, 所以我们只须证明,给定的数中至多只有一个可由素数 2, 3, 5 的每一个所整除。显然仅仅 $6q + 2$ 是偶数, 仅仅 $6q + 3$ 是 3 的倍数, 而且至多只有一个 5 的倍数, 因为 $q_k \not\equiv 3 \pmod{5}$ 。

7. 设 $p_{k-1} < n \leq p_k$ 。对于某一个常数 c , 我们有 $t_k \leq 6t_{k-1} + ck3^k$; 所以 $t_k/6^k \leq t_{k-1}/6^{k-1} + ck/2^k \leq t_0 + c \sum_{j \geq 1} j/2^j = M$ 。于是 $t_k \leq M \cdot 6^k = O(p_k^{\log_3 6})$ 。

8. 错的, 为看出它是错的, 可以以 $k=2$ 来试它。

9. $\tilde{u}_s = u_{(7s) \bmod K}$ 。特别是, 如果 $q = -1$, 我们得到 $\tilde{u}_{(-r) \bmod K}$, 当计算逆变换时它避免了数据的翻转。

10. $A^{[j]}(s_{k-1}, \dots, s_{k-j}, t_{k-j-1}, \dots, t_0)$ 可以写成

$$\sum_{0 \leq i_{k-1}, \dots, i_{k-j} \leq 1} \omega^{2^{k-j}(s_{k-j} \dots s_{k-1})_2 \cdot (t_{k-1} \dots t_{k-j})_2} \left(\sum_{0 \leq p < K} \omega^{ip} u_p \right) \left(\sum_{0 \leq q < K} \omega^{iq} v_q \right)$$

而这是 $\sum_{p,q} u_p v_q S(p, q)$, 其中 $|S(p, q)| = 0$ 或 2^j 。对于 p 和 q 的恰好 $2^{2k}/2^j$ 个值, 我们有 $|S(p, q)| = 2^j$ 。

11. 一个自动机在它有 $c \geq 2$ 之前不能有 $z_2 = 1$, 这种情况在 $3j-1$ 时对 M_j 首次出现。由此得出, 在 $3(j-1)$ 的时间之前 M_j 不能有 $z_2 z_1 z_0 \neq 000$ 。而且如果 M_j 在时间 t 时有 $z_0 \neq 0$, 我们不能把这改变成为 $z_0 = 0$ 而不影响输出; 但是这个输出至少在 $t+j-1$ 的时间之前不会受 z_0 的这个值的影响, 所以我们必须有 $t+j-1 \leq 2n$ 。由于我们给出的头一个论证证明 $3(j-1) \leq t$, 故必定有 $4(j-1) \leq 2n$, 即 $j-1 \leq n/2$, 即 $j \leq \lfloor n/2 \rfloor + 1$ 。这是最好的界, 因为对于所有 $j \leq \lfloor n/2 \rfloor + 1$, 输入 $u = v = 2^{n-1}$ 要求使用 M_j 。(例如, 表2显示为在时间3时乘两个二进位的数需要 M_2 。)

12. 我们可以“扫描”MIX类指令的 K 张表, 在 $O(K + (N \log N)^2)$ 步内执行每个表上的头一条指令如下: (i) 一个基数表排序(5.2.5 小节)将在 $O(K + N)$ 的时间内把所有相同的指令集中在一起, (ii) 每个 j 条相同指令的集合可以在 $O(\log N)^2$

+ $O(j)$ 步内执行, 共有 $O(N^2)$ 个这样的集合。有限次扫描将扫完所有的表, 剩下的细节不难处理; 例如, 通过把 p 和 q 转换成二进制可以模拟算术运算。[SICOMP 9 (1980), 490~508。]

13. 如果 n 位数乘法要花费 $T(n)$ 步, 则我们可以通过把 n 位数分成为 $\lceil n/m \rceil$ 个 m 位组, 利用 $\lceil n/m \rceil T(m) + O(n + m)$ 次运算实现 m 位数和 n 位数的乘法。因此正文中引用的结果给出了在图灵机上的估计运行时间 $O(n \log m \log \log m)$, 在可随机存取有界长的字的机器上的运行时间为 $O(n \log m)$, 在指针机器上为 $O(n)$ 。

15. 已知的最好的上限是 $O(n(\log n)^2 \log \log n)$, 它是由 M.J. Fischer 和 L.J. Stockmeyer 给出的[J. Comp. and Syst. Sci. 9 (1974), 317~331]; 他们的构造对于多带图灵机有效, 而且在指针机器上是 $O(n \log n)$ 的。已知的最好的下限是 $n \log n / \log \log n$ 阶的, 它是 M.S. Paterson, M.J. Fischer 和 A.R. Meyer 给出的[SIAM/AMS Proceedings 7 (1974), 97~111]; 这可应用于多带图灵机但不能应用于指针机器。

16. 设 2^k 是超过 $2K$ 的最小的 2 的幂。置 $a_t \leftarrow \omega^{-t^2/2} u_t$ 和 $b_t \leftarrow \omega^{(2K-2-t)^2/2}$, 其中对于 $t \geq K$, $u_t = 0$ 。当 $0 \leq s < K$ 时, 对于 $r = 2K - 2 - s$ 我们要计算卷积 $c_r = \sum_{j=0}^r a_j b_{r-j}$ 。通过使用阶数为 2^k 的三个快速傅里叶变换, 如同正文中的乘法过程一样, 可以求出这个卷积。[注意这个技术有时称做“啁啾变换”, 它对于任何复数 ω , 而不必是 1 的根都有效。参见 L.I. Bluestein, Northeast Electronics Res. and Eng. Meeting Record 10 (1968), 218~219; D.H. Bailey 和 P.N. Swartztrauber, SIAM Review 33 (1991), 389~404。]

17. 量 $D_n = K_{n+1} - K_n$ 满足 $D_1 = 2$, $D_{2n} = 2D_n$, 以及 $D_{2n+1} = D_n$; 因此当 n 有前述形式时, $D_n = 2^{e_1 - t + 2}$ 。通过对 n 的归纳法, 由此得出, $K_n = 3^{e_1} + \sum_{i=2}^t 3^{e_i} \cdot 2^{e_1 - e_i - t + 3}$ 。

顺便指出, K_n 是奇数, 因此我们可以以 $(K_n + K_{n+1})/2$ 个 1 位乘法来用一个 $n+1$ 位整数乘一个 n 位整数。生成函数 $K(z) = \sum_{n \geq 1} K_n z^n$ 满足 $zK(z) + z^2 = K(z^2)(z+1)(z+2)$; 因此 $K(-1) = 1$ 和 $K(1) = \frac{1}{5}$ 。

18. 下列方案使用工作存储器的 $3N + S_N$ 个位置, 其中 $S_1 = 0$, $S_{2n} = S_n$ 以及 $S_{2n-1} = S_n + 1$, 因此在上道题的符号下 $S_n = e_1 - e_t - t + 2 - [t=1]$ 。令 $N = 2^n - \epsilon$, 其中 ϵ 是 0 或 1, 并假定 $N > 1$ 。给定 N 位的数 $u = 2^n U_1 + U_0$ 和 $v = 2^n V_1 + V_0$, 我们首先把 $|U_0 - U_1|$ 和 $|V_0 - V_1|$ 构造成在 $(3N + S_N)$ 个位置的工作区域的 0 位和 n 位开始的两个 n 位区域。然后我们把它们的乘积放在以 $3n + S_n$ 处开始的工作区域中。下一步是构造在 0 位处开始的 $2(n - \epsilon)$ 位乘积 $U_1 V_1$; 使用该乘积, 我们把 $3n + S_n$ 位置处开始的 $3n - 2\epsilon$ 个位置改成 $U_1 V_1 - (U_0 - U_1)(V_0 - V_1) + 2^n U_1 V_1$ 的值。(注意 $3n - 2\epsilon + 3n + S_n = 3N + S_N$ 。)最后, 我们构造在 0 位置处开

始的 $2n$ 位的乘积 $U_0 V_0$, 而且把它加到在 $2n + S_n$ 和 $3n - S_n$ 处开始的部分结果。我们还必须通过把它下移 $2n + S_n$ 个位置来把 $2N$ 位的答案移动到它最后的位置。

通过一个比较巧妙的变形, 即通过在指定的工作区域循环移动其输出一个给定的数量, 可以避免最后的移动。如果不允许 $2N$ 位的乘积同辅助工作区域相邻, 我们需要大约另外 N 个存储位置 (即对于输入, 输出和临时存储, 总数大约是 $6N$ 而不是 $5N$); 参见 R. Maeder, *Lecture Notes in Comp. Sci.* **722** (1993), 59~65。

19. 令 $m = s^2 + r$, 其中 $-s < r \leq s$, 我们可以以 $U_1 = \lfloor u/s \rfloor$, $U_0 = u \bmod s$, $V_1 = \lfloor v/s \rfloor$, $V_0 = v \bmod s$ 来使用 (2), 且以 s 来起 2^n 的作用。如果我们知道 $U_1 - U_0$ 和 $V_1 - V_0$ 的符号, 我们知道怎样来计算乘积 $|U_1 - U_0| |V_1 - V_0|$, 它小于 m , 以及是加上或减去它。剩下的是乘以 s 和乘以 $s^2 \equiv -r$ 。利用习题 3.2.1.1-9, 这些的每一个都可以用四个乘法/除法来完成, 但由于为计算 $sx \bmod m$ 所需要的乘法之一是乘以 r 或 $r + s$, 因此只需要 7 个。因此 14 个乘法/除法就足够了 (或者在 $u = v$ 或 u 是常数的情况下, 12 个就够了)。无须有比较操作数的能力, 通过分开计算 $U_0 V_1$ 和 $U_1 V_0$, 我们仍然能以多一个乘法来完成这件工作。

4.4 节

1. 我们通过 B_j 系统中的加法和乘法来计算 $(\cdots (a_m b_{m-1} + a_{m-1}) b_{m-2} + \cdots + a_1) b_0 + a_0$ 。

	T	= 20(cwt	= 8(st	= 14(lb	- 16(oz))
以 0 开始	0	0	0	0	0
加 3	0	0	0	0	3
乘以 24	0	0	0	4	8
加 9	0	0	0	5	1
乘以 60	0	2	5	9	12
加 12	0	2	5	10	8
乘以 60	8	3	1	0	0
加 37	8	3	1	2	5

(在一混合进制系统中的加法和乘以一个常数的乘法可以很容易地利用通常进位规则的一个简单推广完成; 参考习题 4.3.1-9。)

2. 我们计算 $\lfloor u/B_0 \rfloor, \lfloor \lfloor u/B_0 \rfloor / B_1 \rfloor$ 等等, 余数是 A_0, A_1 等等。除法在 b_j 系统中进行。

	d	= 24(h	= 60(m	= 60s))	
由 u 开始	3	9	12	37	
除以 16	0	5	4	32	余数 = 5
除以 14	0	0	21	45	余数 = 2
除以 8	0	0	2	43	余数 = 1
除以 20	0	0	0	8	余数 = 3
除以 ∞	0	0	0	0	余数 = 8

答案: 8 T 3 cwt 1 st 2 lb 5 oz。

3. 下面是由 G.L.Steele Jr. 和 Jon L. White 给出的过程, 它推广了 Taranto 对于 $B=2$ 的算法。Taranto 的算法原来发表于 CACM 2,7 (1959 年 7 月), 27。

A1. [初始化] 置 $M \leftarrow 0, U_0 \leftarrow 0$ 。

A2. [完成了?] 如果 $u < \epsilon$ 或 $u > 1 - \epsilon$, 则转向 A4 步。(否则没有 M 位小数能满足给定的条件。)

A3. [变换] 置 $M \leftarrow M + 1, U_{-M} \leftarrow \lfloor Bu \rfloor, u \leftarrow Bu \bmod 1, \epsilon \leftarrow B\epsilon$, 并返回 A2。

(这个变换实质上把我们送回到原来所处的状态; 剩下的问题是以最小的 B 进位数把 u 转换成 U , 使得 $|U - u| < \epsilon$ 。但是, 注意 ϵ 现在可以大于等于 1; 在这种情况下, 我们可以立即转到 A4 步而不是存 ϵ 的新值。)

A4. [舍入] 如果 $u \geq \frac{1}{2}$, 则 U_{-M} 加 1。(如果 u 精确地等于 $\frac{1}{2}$, 则可考虑采用别的舍入规则, 例如“仅当它为奇数时 U_{-M} 加 1”; 参考 4.2.2 小节。)

■

步骤 A4 绝不会把 U_{-M} 从 $B-1$ 增加到 B ; 因为如果 $U_{-M} = B-1$, 我们必定有 $M > 0$, 但是没有 $M-1$ 位小数是充分精确的。Steele 和 White 在他们的论文 [SIGPLAN Notices 25, 6 (1990 年 6 月), 112~126] 继续考虑了浮点的转换。也请参见由 W.H.J. Feijen 等人编辑的 *Beauty is Our Business* (New York: Springer, 1990) 中 D.E. Knuth 的文章, 233~242。

4. (a) $1/2^k = 5^k/10^k$ 。(b) b 的每一个素因子整除 B 。

5. 当且仅当 $10^n - 1 \leq c < w$; 参考 (3)。

7. $au \leq ux \leq au + u/w \leq au + 1$, 因此 $\lfloor au \rfloor \leq \lfloor ux \rfloor \leq \lfloor au + 1 \rfloor$ 。而且, 在引用的特殊情况中, 对于 $0 < \epsilon \leq \alpha$ 我们有 $ux < au + \alpha$ 且 $\lfloor au \rfloor = \lfloor au + \alpha - \epsilon \rfloor$ 。

```

8.      ENT1  0
        LDA   U
        1H    MUL    = 1//10 =
        3H    STA    TEMP

```

```

MUL    = -10 =
SLAX   5
ADD    U
JANN   2F
LDA    TEMP    (由习题 7, 仅在头一次
                  迭代时才能出现)
DECA   1
JMP    3B
2H STA  ANSWER, 1 (可能是负 0)
LDA    TEMP
INC1   1
JAP    1B  █

```

9. 令 $N = 2^{2^{k+1}}$ 。这个计算置

$$v \leftarrow \left\lfloor \frac{(2^2 - 1)}{2^1} \frac{(2^4 + 1)}{2^4} \frac{(2^8 + 1)}{2^8} \dots \frac{(2^{2^k} + 1)}{2^{2^k}} (u + 1) \right\rfloor = \left\lfloor \frac{8}{5} \frac{N - 1}{N} (u + 1) \right\rfloor$$

因此 $q = \lfloor u/10 + \epsilon_u \rfloor$, 其中 $\epsilon_u = \frac{1}{10}(1 - (u + 1)/N)$ 。由于 $N \bmod 10 = 6$ 而且对于 $0 \leq u < N$, $0 \leq \epsilon_u < 1/10$, 我们看到对于 $0 \leq u < N + 4$, $q = \lfloor u/10 \rfloor$ 。

当 u 在这个范围中时, 我们有 $r = u \bmod 10 + \lfloor 1 - (u + 1 + 5\theta_u)/N \rfloor$, 其中 $\theta_u = \frac{N-1}{5}(u+1) \bmod \frac{N}{8}$ 。如果 θ_u 很大, 比如说 $\theta_u = N/8 - t$, 其中 $0 < t < N/40$, 我们有 $u + 1 \equiv 5t \pmod{N/8}$; 因此如果 $u < N/2$, 则 $u + 1 + 5\theta_u \leq N$ 。否则 $\theta_u \leq N/8 - N/40 = N/10$, 如果 $u < N/2$ 再一次我们有 $u + 1 + \theta_u \leq N$ 。 $u = N/2, N/2 + 1, N/2 + 2$ 和 $N/2 + 3$ 的情况容易看出不会引起问题。但当 $u = N/2 + 4$ 时我们发现 $u \bmod 10 = 2$ 以及 $r = 1$ 。

[另一个方法 $r \leftarrow u - 8q$, $r \leftarrow r - 2q$ 将在一个更大的范围内有效, 但在一台 8 位计算机上就不太快了。本题是以 R.A. Vowels, *Australian Comp. J* **24** (1992), 81~85 的思想为基础的。]

10. (i) 右移 1 位; (ii) 抽取每个组的左边一位; (iii) (ii) 的结果右移两位; (iv) (iii) 的结果右移一位, 并把它加到 (iii) 的结果; (v) 从 (i) 的结果中减去 (iv) 的结果。

11. 5.7721

$$\begin{array}{r}
 -10 \\
 47.721 \\
 -94 \\
 \hline
 383.21
 \end{array}$$

$$\begin{array}{r} - 766 \\ 3066.1 \end{array}$$

$$\begin{array}{r} - 6132 \\ 24529 \end{array}$$

答案: $(24529)_{10}$

12. 首先把三进制数转换成九进制, 然后如同八进制到十进制的转换一样进行, 但无须加倍。十进制到九进制的转换是类似的。在给定的例子中, 我们有

$$\begin{array}{r} 1.764723 \\ - 1 \\ 16.64723 \end{array}$$

$$\begin{array}{r} - 16 \\ 150.4723 \end{array}$$

$$\begin{array}{r} - 150 \\ 1354.723 \end{array}$$

$$\begin{array}{r} - 1354 \\ 12193.23 \end{array}$$

$$\begin{array}{r} - 12193 \\ 109739.3 \end{array}$$

$$\begin{array}{r} - 109739 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} - 987654 \\ 987654 \end{array}$$

$$\begin{array}{r} 9.87654 \\ + 9 \\ 118.7654 \end{array}$$

$$\begin{array}{r} + 118 \\ 1316.654 \end{array}$$

$$\begin{array}{r} + 1316 \\ 14483.54 \end{array}$$

$$\begin{array}{r} + 14483 \\ 160428.4 \end{array}$$

$$\begin{array}{r} + 160428 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

$$\begin{array}{r} + 1764723 \\ 1764723 \end{array}$$

答案: $(987654)_{10}$

13.

BUF ALF .□□□□

(小数点在头一行)

ORIG *+39

START JOV OFLO

确保溢出开关断开

ENT2 -40

置缓冲区指针

8H ENT3 10

置循环计数器

1H ENT1 *m*

开始乘法程序

ENTX 0

2H STX CARRY

...

(见习题4.3.1-13, 且 $v=10^9$ 和 $w=U$)

J1P 2B

SLAX 5

 $rA \leftarrow$ 下9个数字

CHAR

STA BUF+40,2(2:5)

存下9个数字

STX	BUF + 41, 2	
INC2	2	缓冲区指针增值
DEC3	1	
3P	1B	重复 10 次
OUT	BUF + 20, 2(PRINTER)	
2N	8B	重复直到两行都打印完毕

14. 设 $K(n)$ 是为转换一个 n 位数字的十进制数成为二进制数, 同时计算 10^n 的二进表示所需要的步数。则我们有 $K(2n) \leq 2K(n) + O(M(n))$ 。证明: 给定数 $U = (u_{2n-1} \cdots u_0)_{10}$, 在 $2K(n)$ 步内计算 $U_1 = (u_{2n-1} \cdots u_n)_{10}$ 和 $U_0 = (u_{n-1} \cdots u_0)_{10}$ 以及 10^n , 然后在 $O(M(n))$ 步内计算 $U = 10^n U_1 + U_0$ 和 $10^{2n} = 10^n \cdot 10^n$ 。由此得出 $K(2^n) = O(M(2^n) + 2M(2^{n-1}) + 4M(2^{n-2}) + \cdots) = O(nM(2^n))$ 。

[类似地, Schönage 已经发现, 我们可以在 $O(nM(2^n))$ 步内把一个 $(2^n \lg 10)$ 位数 U 从二进制转换成十进制。首先在 $O(M(2^{n-1}) + M(2^{n-2}) + \cdots) = O(M(2^n))$ 步内形成 $V = 10^{2^{n-1}}$, 然后再用 $O(M(2^n))$ 步计算 $U_0 = (U \bmod V)$ 和 $U_1 = \lfloor U/V \rfloor$, 再转换 U_0 和 U_1 。]

17. 参见 W.D. Clinger, SIGPLAN Notices **25**, 6 (1990 年 6 月), 92~101 以及习题 3 的答案中引证的 Steele 和 White 的文章。

18. 设 $U = \text{round}_B(u, P)$ 和 $v = \text{round}_b(U, p)$ 。我们可以假设 $u > 0$, 使得 $U > 0$ 和 $v > 0$ 。情况 1, $v < u$: 确定 e 和 E , 使得 $b^{e-1} < u \leq b^e$, $B^{E-1} \leq U < B^E$ 。然后 $u \leq U + \frac{1}{2}B^{E-P}$ 和 $U \leq u - \frac{1}{2}b^{e-p}$; 因此, $B^{E-1} \leq B^{P-E}U < B^{P-E}u \leq b^{p-e}u \leq b^p$ 。情况 2, $v > u$: 确定 e 和 E 使得 $b^{e-1} \leq u < b^e$, $B^{E-1} < U \leq B^E$ 。然后 $u \geq U - \frac{1}{2}B^{E-P}$, 而且 $U \geq u + \frac{1}{2}b^{e-p}$; 因此 $B^{P-1} \leq B^{P-E}(U - B^{E-P}) < B^{P-E}u \leq b^{p-e}u < b^p$ 。于是我们证明了每当 $v \neq u$ 时, $B^{P-1} < b^p$ 。

反之, 如果 $B^{P-1} < b^p$, 则上述证明提示, 当 u 是 b 的乘方且同时接近于 B 的一个乘方时, $u \neq v$ 的最为可能的例子将出现。我们有 $B^{P-1}b^p < B^{P-1}b^p + \frac{1}{2}b^p - \frac{1}{2}B^{P-1} - \frac{1}{4} = \left(B^{P-1} + \frac{1}{2}\right)\left(b^p - \frac{1}{2}\right)$; 因此 $1 < \alpha = 1/\left(1 - \frac{1}{2}b^{-p}\right) < 1 + \frac{1}{2}B^{1-P} = \beta$ 。由习题 4.5.3-50, 有整数 e 和 E 使得 $\log_B \alpha < e \log_B b - E < \log_B \beta$ 。因此对于某个 e 和 E , $\alpha < b^e/B^E < \beta$ 。现在我们有 $\text{round}_B(b^e, P) = B^E$, 以及 $\text{round}_b(B^E, P) < b^e$ 。[CACM **11**(1968), 47~50; Proc. Amer. Math. Soc. **19** (1968), 716~723。]

例如, 如果 $b^p = 2^{10}$ 和 $B^P = 10^4$, 数 $u = 2^{6408} \approx .100049 \cdot 10^{1930}$ 降低成 $U = .1 \cdot 10^{1930} \approx (.1111111110111111111)_2 \cdot 2^{6408}$, 它降低成 $2^{6408} - 2^{6398}$ 。(由 Fred J. Tydeman 发现的最小的例子实际上是 $\text{round}((.1111111001)_2 \cdot 2^{784}) = .1011 \cdot$

$$10^{236}, \text{round}(.1011 \cdot 10^{235}) = (.11111110010)_2 \cdot 2^{784}。$$

19. $m_1 = (\text{FOFOFOFO})_{16}, c_1 = 1 - 10/16$ 使 $U = ((u_7 u_6)_{10} \cdots (u_1 u_0)_{10})_{256}$; 于是 $m_2 = (\text{FF00FF00})_{16}, c_2 = 1 - 10^2/16^2$ 使得 $U = ((u_7 u_6 u_5 u_4)_{10} (u_3 u_2 u_1 u_0)_{10})_{65536}$; 以及 $m_3 = (\text{FFFF0000})_{16}, c_3 = 1 - 10^4/16^4$ 完成了这一工作。[试同习题 14 中的 Schönhage 算法作比较。这个技术是由 Roy A. Keir 在大约 1958 年时给出的。]

4.5.1 小节

1. 测试是否 $uv' < u'v$, 因为分母为正。

2. 如果 $c > 1$ 既整除 u/d 也整除 v/d , 则 cd 既整除 u 也整除 v 。

3. 设 p 是素数。如果对于 $e \geq 1, p^e$ 是 uv 和 $u'v'$ 的一个因子, 则或者 $p^e \setminus u$ 和 $p^e \setminus v'$ 或者 $p^e \setminus u'$ 和 $p^e \setminus v$; 因此 $p^e \setminus \gcd(u, v') \gcd(u', v)$ 。把这个论证的次序反过来即得出它的逆。

4. 设 $d_1 = \gcd(u, v), d_2 = \gcd(u', v')$; 答案是 $w = (u/d_1)(v'/d_2)\text{sign}(v)$, $w' = |(u'/d_2)(v/d_1)|$, 而且如果 $v = 0$ 则带有“除以 0”的出错信息。

5. $d_1 = 10, t = 17 \cdot 7 - 27 \cdot 12 = -205, d_2 = 5, w = -41, w' = 168$ 。

6. 设 $u'' = u'/d_1, v'' = v'/d_1$; 我们的目标是证明 $\gcd(uv'' + u''v, d_1) = \gcd(uv'' + u''v, d_1 u''v'')$ 。如果 p 是整除 u'' 的一个素数, 则 p 不能整除 u 或 v'' , 所以 p 不能整除 $uv'' + u''v$ 。对于 v'' 的素因子类似的论断成立, 所以 $u''v''$ 没有影响给定的 \gcd 的素因子。

7. $(N-1)^2 + (N-2)^2 = 2N^2 - (6N-5)$ 。如果输入是 n 位二进位数, 则为表示 t 可能需要 $2n+1$ 位。

8. 假定 x 是有限的且非 0, 对于乘法和除法, 这些量将遵守规则 $x/0 = \text{sign}(x) \cdot \infty, (\pm\infty) \times x = x \times (\pm\infty) = (\pm\infty)/x = \pm \text{sign}(x) \infty, x/(\pm\infty) = 0$, 而无须改变所述的算法。其次, 这些算法很容易修改, 使得 $0/0 = 0 \times (\pm\infty) = (\pm\infty) \times 0 = “(0/0)”$, 其中后者是“不确定”的一个表示; 如果这两个操作数之一“不确定”, 则结果也将“不确定”。

由于乘法和除法子程序可以产生这些推广的算术的相当自然的规则, 有时值得修改加法和减法运算, 使得它们满足以下规则: 对于有限的 $x, x \pm \infty = \pm\infty, x \pm (-\infty) = \mp\infty; (\pm\infty) + (\pm\infty) = \pm\infty - (\mp\infty) = \pm\infty$; 而且 $(\pm\infty) + (\mp\infty) = (\pm\infty) - (\pm\infty) = (0/0)$; 而且如果两个操作数之一或两个都是 $(0/0)$, 则结果也是。相等测试和比较可以用类似的方式处理。

上边的注释是同“溢出”指示无关的。如果 ∞ 用来提示溢出, 那么命 $1/\infty$ 等于 0 是不正确的, 因为这样会把不精确的结果当成真正的答案。用 $(0/0)$ 表示溢出, 而且坚持这样的约定, 即如果至少有一个输入是不确定的, 则任何运算的结果也不确定, 这样做更好一些。这种类型的溢出指示有这样一个优点, 即推广计算的最后结果确切地揭示哪些答案是确定的和哪些是不确定的。

9. 如果 $u/u' \neq v/v'$, 则 $1 \leq |uv' - u'v| = u'v' |u/u' - v/v'| < |2^{2n}u/u' - 2^{2n}v/v'|$; 相差大于 1 的两个量不可能有相同的“底限”。(换言之, 当分母有 n 位时, 二进制点右边的头 $2n$ 位足以表征二进分数的值。我们不能把这改进成 $2n-1$ 位, 因为如果 $n=4$, 则我们有 $\frac{1}{13} = (.00010011\cdots)_2, \frac{1}{14} = (.00010010\cdots)_2$ 。)

11. 当 v 和 v' 不全为 0 时, 为了除以 $(v + v'\sqrt{5})/v''$, 可乘以倒数 $(v - v'\sqrt{5})v''/(v^2 - 5v'^2)$, 并约简到最低项。

12. $((2^{q-1}-1)/1); \text{round}(x) = (0/1)$ 当且仅当 $|x| \leq 2^{1-q}$ 。类似地, $\text{round}(x) = (1/0)$ 当且仅当 $x \geq 2^{q-1}$ 。

13. 一个想法是限制分子和分母总共为 27 个二进位, 其中我们仅需要存其中的 26 位(因为分母的前导位为 1 除非分母的长度为零)。这就有位置来保留一个符号位和另外的五位以指出分母大小。另一个想法是对分子和分母使用 28 个二进位, 它们至多总共有 7 个十六进数字, 连同 1 个符号位和 3 个二进位的字段在一起以指出分母中的十六进数字的个数。

[利用下题中的公式, 头一个办法精确地导致 2140040119 个有限的可表示数, 而第二个办法导致 1830986459 个这样的数。头一个办法是可取的, 因为它能表示更多的值, 而且因为它是更干净的并使范围之间的转换更为顺当。类似地, 对于 64 个二进位, 我们将把分子和分母限制成总共至多有 $64-6=58$ 个二进位。]

14. 在区间 $(a, b]$ 中 n 的倍数的个数为 $\lfloor b/n \rfloor - \lfloor a/n \rfloor$ 。因此, 由容斥原理, 这个问题的答案是 $S_0 - S_1 + S_2 - \cdots$, 其中 S_k 是 $\sum (\lfloor M_2/P \rfloor - \lfloor M_1/P \rfloor)(\lfloor N_2/P \rfloor - \lfloor N_1/P \rfloor)$, 这个求和对 k 个不同素数的所有乘积 P 进行。我们也可把这个答案表达为

$$\sum_{n=1}^{\min(M_2, N_2)} \mu(n) (\lfloor M_2/n \rfloor - \lfloor M_1/n \rfloor) (\lfloor N_2/n \rfloor - \lfloor N_1/n \rfloor)$$

4.5.2 小节

1. 把 $\text{gcd}, \text{lcm}, \times$ 分别一致地替换成 $\min, \max, +$ (在确保当任何变量为零时, 恒等式仍然正确之后)。

2. 对于素数 p , 设 $u_p, v_{1p}, \cdots, v_{np}$ 是 u, v_1, \cdots, v_n 的规范分解中 p 的指数。由假设, $u_p \leq v_{1p} + \cdots + v_{np}$ 。我们必须证明 $u_p \leq \min(u_p, v_{1p}) + \cdots + \min(u_p, v_{np})$, 如果 u_p 大于或等于每个 v_{jp} , 或者 u_p 小于某个 v_{jb} , 则这肯定为真。

3. 解法 1: 如果 $n = p_1^{e_1} \cdots p_r^{e_r}$, 在每种情况下的数是 $(2e_1+1) \cdots (2e_r+1)$ 。解法 2: 如果对于 n^2 的每个因子 d , 我们置 $u = \text{gcd}(d, n)$ 和 $v = n^2/\text{lcm}(d, n)$, 则得到一对一的对应关系。[E. Cesàro, *Annali di Matematica Pura ed Applicata* (2) 13 (1885), 235~250, § 12。]

4. 见习题 3.2.1.2-15(a)。

5. 右移 u 和 v 直到它们都不是 3 的倍数为止, 并记住将在 gcd 中出现的 3 的正确指数。每个随后的迭代置 $t \leftarrow u + v$ 或 $t \leftarrow u - v$ (不论哪个是 3 的倍数), 右移 t 直到它不是 3 的倍数为止, 然后用此结果代替 $\max(u, v)$ 。

u	v	t
13634	24140	10506, 3502;
13634	3502	17136, 5712, 1904;
1904	3502	5406, 1802;
1904	1802	102, 34;
34	1802	1836, 612, 204, 68;
34	68	102, 34;
34	34	0

现在 $\gcd(40902, 24140) = 34$ 是铁证如山了。

6. u 和 v 两者都是偶数的概率为 $\frac{1}{4}$; 两个都是 4 的倍数的概率为 $\frac{1}{16}$; 等等, 于是, A 有由生成函数

$$\frac{3}{4} + \frac{3}{16}z + \frac{3}{64}z^2 + \cdots = \frac{3/4}{1 - z/4}$$

给出的分布。均值是 $\frac{1}{3}$, 标准差是 $\sqrt{\frac{2}{9} + \frac{1}{3} - \frac{1}{9}} = \frac{2}{3}$ 。如果 u, v 对于 $1 \leq u, v < 2^N$ 是独立和一致地分布的, 则需要某些小的校正项; 而均值实际上是

$$(2^N - 1)^{-2} \sum_{k=1}^N (2^{N-k} - 1)^2 = \frac{1}{3} - \frac{4}{3}(2^N - 1)^{-1} + N(2^N - 1)^{-2}$$

7. 当 u, v 不全为偶数时, 情况(偶, 奇), (奇, 偶), (奇, 奇)的每一种都是同等可能的, 而且在这些情况下 $B = 1, 0, 0$ 。因此平均说来 $B = \frac{1}{3}$ 。实际上, 如同在习题 6 中那样, 当 $1 \leq u, v < 2^N$ 时应给出小的校正使之严格地精确; $B = 1$ 的概率实际上是:

$$(2^N - 1)^{-2} \sum_{k=1}^N (2^{N-k} - 1)2^{N-k} = \frac{1}{3} - \frac{1}{3}(2^N - 1)^{-1}$$

8. 令 F 是其中 $u > v$ 的减法步骤的个数。于是 $E = F + B$ 。如果我们把输入从 (u, v) 变成 (v, u) , 则 C 的值保持不变, 而 F 变成 $C - 1 - F$ 。因此 $E_{\text{ave}} = \frac{1}{2}(C_{\text{ave}} - 1) + B_{\text{ave}}$ 。

9. 二进算法首先以 $u = 1963, v = 1359$ 到达 B6。然后 $t \leftarrow 604, 302, 151$, 等等。

因此 \gcd 是 302。利用算法 X, 我们求得 $2 \cdot 31408 - 23 \cdot 2718 = 302$ 。

10. a) 两个整数互素当且仅当它们不同时被任何素数整除。b) 借助于分母 $k = p_1 \cdots p_r$, 重新排列 a) 中的和 (注意 a), b) 中的每个和实际上都是有限的)。c) 由于 $(n/k)^2 - \lfloor n/k \rfloor^2 = O(n/k)$, 所以我们有 $q_n = \sum_{k=1}^n \mu(k) (n/k)^2 = \sum_{k=1}^n O(n/k) = O(nH_n)$ 。而且 $\sum_{n > n} (n/k)^2 = O(n)$ 。d) $\sum_{d \mid n} \mu(d) = \delta_{1n}$ 。[事实上, 我们有更为一般的结果

$$\sum_{d \mid n} \mu(d) \left(\frac{n}{d} \right)^s = n^s - \sum \left(\frac{n}{p} \right)^s + \sum \left(\frac{n}{pq} \right)^s - \cdots$$

如在 b) 中那样, 其中右边的求和是对于 n 的素因子进行的, 如果 $n = p_1^{e_1} \cdots p_r^{e_r}$, 则这等于 $n^s (1 - 1/p_1^s) \cdots (1 - 1/p_r^s)$ 。]

注: 类似地, 我们发现 k 个整数的集合互素的概率是 $1/\zeta(k) = 1/(\sum_{n \geq 1} 1/n^k)$ 。定理 D 的这个证明见 F. Mertens, *Crelle* 77 (1874), 289~291。这一技术实际上给出了强得多的结果, 即当 $m \leq n$ 时, 对于任意的 f 和 g , 有 $6\pi^{-2}mn + O(n \log m)$ 对整数 $u \in [f(m), f(m) + m]$, $v \in [g(n), g(n) + n]$ 互素。

11. (a) $6/\pi^2$ 乘以 $1 + \frac{1}{4} + \frac{1}{9}$, 即 $49/(6\pi^2) \approx .82746$ 。(b) $6/\pi^2$ 乘以 $1/1 + 2/4 + 3/9 + \cdots$, 即 ∞ 。(不管习题 12 和 14 的结果怎样, 这都是真的。)

12. [*Annali di Mat.* (2) 13 (1885), 235~250, § 30] 设 $\sigma(n)$ 是 n 的正因子的个数。答案是

$$\sum_{k \geq 1} \sigma(k) \frac{6}{\pi^2 k^2} = \frac{6}{\pi^2} \left(\sum_{k \geq 1} \frac{1}{k^2} \right)^2 = \frac{\pi^2}{6}$$

[于是, 平均值小于 2, 尽管当 u, v 不互素时至少总有两个公因子。]

$$13. 1 + \frac{1}{9} + \frac{1}{25} + \cdots = 1 + \frac{1}{4} + \frac{1}{9} + \cdots = \frac{1}{4} \left(1 + \frac{1}{4} + \frac{1}{9} + \cdots \right).$$

14. (a) $L = (6/\pi^2) \sum_{d \geq 1} d^{-2} \ln d = -\zeta'(2)/\zeta(2) = \sum_{p \text{ 素数}} (\ln p)(2^p - 1) \approx 0.56996$ 。(b) $(8/\pi^2) \sum_{d \geq 1} [d \text{ 奇数}] d^{-2} \ln d = L - \frac{1}{3} \ln 2 \approx 0.33891$ 。

15. $v_1 = \pm v/u_3, v_2 = \mp u/u_3$ (符号依赖于迭代次数是偶数还是奇数)。这由 v_1 和 v_2 彼此互素 (在整个算法期间) 这一事实以及 $v_1 u = v_2 v$ 得出。[因此在算法结束时 $v_1 u = \text{lcm}(u, v)$, 但是这不是计算最低公倍数的特别有效的方法。关于它的一个推广, 请参见习题 4.6.1-18。]

在习题 4.5.3-48 中可以找到进一步的细节。

16. 应用算法 X 于 v 和 m , 于是得到一个值 x 使得 $xv \equiv 1 \pmod{m}$ 。(简化算法 X, 使得 u_2, v_2 和 t_2 不必被计算即可做到这一点, 因为在答案中绝不使用它们。) 然后置 $w \leftarrow ux \pmod{m}$ 。[像在习题 4.5.3-45 中那样, 因此得出, 在把它应用于

很大的 n 个二进位的数时,这一过程需要 $O(n^2)$ 个时间单位。关于算法 X 的其它变化,参见习题 17 和 39。]

17. 如同在牛顿方法中那样,我们可以令 $u' = (2u - vu^2) \bmod 2^{2^e}$ (参见 4.3.1 小节末尾)。等价地,如果 $uv \equiv 1 + 2^{2^e}w \pmod{2^{2^e}}$, 令 $u' = u + 2^e((-uw) \bmod 2^e)$ 。

18. 设除 u 和 v 之外, $u_1, u_2, u_3, v_1, v_2, v_3$ 也是多精度变量。推广的算法将和算法 L 对 u 和 v 所做的那样对 u_3 和 v_3 起作用。在步骤 L4 中,新的多精度运算是对于所有 j , 置 $t \leftarrow Au_j, t \leftarrow t + Bv_j, w \leftarrow Cu_j, w \leftarrow w + Dv_j, u_j \leftarrow t, v_j \leftarrow w$; 而且在这一步如果 $B \neq 0$, 则对于所有 j 和对 $q = \lfloor u_3/v_3 \rfloor$, 置 $t \leftarrow u_j - qv_j, u_j \leftarrow v_j, v_j \leftarrow t$ 。如果 v_3 很小, 则对步骤 L1 也做类似的修改。内循环(步骤 L2 和 L3)不变。

19. a) 置 $t_1 = x + 2y + 3z$; 则 $3t_1 + y + 2z = 1, 5t_1 - 3y - 20z = 3$ 。消去 y , 则 $14t_1 - 14z = 6$: 无解。b) 这时 $14t_1 - 14z = 0$ 。除以 14, 消去 t_1 ; 一般解是 $x = 8z - 2, y = 1 - 5z, z$ 是任意的。

20. 我们可以假定 $m \geq n$ 。如果 $m > n = 0$, 对于 $1 \leq t < m$, 我们可以以 2^{-t} 的概率得到 $(m-t, 0)$, 以 2^{1-m} 的概率得到 $(0, 0)$ 。正确地说, 对于 $n > 0$, 下列的值都可得到。

情况 1, $m = n$ 。对于 $2 \leq t < n$, 我们以 $t/2^t - 5/2^{t+1} + 3/2^{2t}$ 的概率从 (n, n) 转到 $(n-t, n)$ 。(这些值是 $\frac{1}{16}, \frac{7}{64}, \frac{27}{256}, \dots$) 转到 $(0, n)$ 的概率是 $n/2^{n-1} - 1/2^{n-2} + 1/2^{2n-2}$ 。转到 (n, k) 的概率和转到 (k, n) 的相同。这个算法以 $1/2^{n-1}$ 的概率终止。

情况 2, $m = n + 1$ 。当 $n > 1$ 时, 我们以 $\frac{1}{8}$ 的概率从 $(n+1, n)$ 达到 (n, n) , 或者当 $n = 1$ 时概率为 0; 对于 $1 \leq t < n-1$, 以 $11/2^{t+3} - 3/2^{2t+1}$ 的概率达到 $(n-t, n)$ 。(这些值是 $\frac{5}{16}, \frac{1}{4}, \frac{19}{128}, \dots$) 对于 $n > 1$, 我们以 $5/2^{n+1} - 3/2^{2n-1}$ 的概率达到 $(1, n)$; 以 $3/2^{2n} - 1/2^{2n-1}$ 的概率达到 $(0, n)$ 。

情况 3, $m \geq n - 2$ 。概率由下表给出:

$$\begin{aligned} (m-1, n): & \quad 1/2 - 3/2^{m-n+2} - \delta_{n1}/2^{m+1} \\ (m-t, n): & \quad 1/2^t + 3/2^{m-n+t+1}, \quad 1 < t < n \\ (m-n, n): & \quad 1/2^n + 1/2^m, \quad n > 1 \\ (m-n-t, n): & \quad 1/2^{n+t} + \delta_{n1}/2^{m-1}; \quad 1 \leq t < m-n \\ (0, n): & \quad 1/2^{m-1} \end{aligned}$$

关于这些结果惟一有趣的事情是它们竟如此地乱七八糟, 但这也使我们对我们毫无兴趣可言了。

21. 证明当 m 很大时, 对于固定的 v 和对于 $2^m < u < 2^{m+1}$, 算法的每个减法移位周期把 $\lfloor \lg u \rfloor$ 平均减 2。

22. 在 u 已经右移成为奇数之后, 在 $1 \leq u \leq 2^N$ 的范围内恰有 $(N-m) \cdot$

$2^{m-1+\delta_{m0}}$ 个整数 u 有 $\lfloor \lg u \rfloor = m$ 。于是

$$(2^N - 1)^2 C = N^2 C_{00} + 2N \sum_{1 \leq n \leq N} (N - n) 2^{n-1} C_{n0} + \\ 2 \sum_{1 \leq n < m \leq N} (N - m)(N - n) 2^{m+n-2} C_{mn} + \sum_{1 \leq n \leq N} (N - n)^2 2^{2n-2} C_{nn}$$

(借助于 D_{mn} 和对于 D 同样的公式成立。)

中间和是 $2^{2N-2} \sum_{0 \leq m < n < N} mn 2^{-m-n} ((\alpha + \beta)N + \gamma - \alpha m - \beta n)$ 。由于

$$\sum_{0 \leq m < n} m 2^{-m} = 2 - (n+1)2^{1-n} \text{ 和 } \sum_{0 \leq m < n} m(m-1)2^{-m} = 4 - (n^2 + n + 2)2^{1-n}$$

因此对 m 求和是

$$2^{2N-2} \sum_{0 \leq n < N} n 2^{-n} \left((\gamma - \alpha - \beta n + (\alpha + \beta)N)(2 - (n+1)2^{1-n}) - \right. \\ \left. \alpha(4 - (n^2 + n + 2)2^{1-n}) \right) = \\ 2^{2N-2} \left((\alpha + \beta)N \sum_{n \geq 0} n 2^{-n} (2 - (n+1)2^{1-n}) + O(1) \right)$$

因此答案中 $(\alpha + \beta)N$ 的系数被求出为 $2^{-2} \left(4 - \left(\frac{4}{3} \right)^3 \right) = \frac{11}{27}$ 。一个类似的论证也适用于其它的求和。

注:借助于一般的分部求和公式,在经过某些冗长的计算之后,可以得到这些和的精确的值。分部求和的一般公式为

$$\sum_{0 \leq k < n} k^m z^k = \frac{m! z^m}{(1-z)^{m+1}} - \sum_{k=0}^m \frac{m^k n^{m-k} z^{n+k}}{(1-z)^{k+1}}$$

23. 如果 $x \leq 1$, 它是 $\Pr(u \geq v \text{ 和 } v/u \leq x) = \frac{1}{2} (1 - G_n(x))$ 。而且如果 $x \geq 1$,

它是 $\frac{1}{2} + \Pr(u \leq v \text{ 和 } v/u \geq 1/x) = \frac{1}{2} + \frac{1}{2} G_n(1/x)$; 由(40), 这也等于 $\frac{1}{2} (1 - G_n(x))$ 。

24. $\sum_{k \geq 1} 2^{-k} G(1/(2^k + 1)) = S(1)$ 。这个值同经典的常数没有明显的关系, 它近似于 0.5432582959。

25. Richard Brent 已经指出, $G(e^{-y})$ 是一个对于所有 y 的实值解析的奇函数。如果我们令 $G(e^{-y}) = \lambda_1 y + \lambda_3 y^3 + \lambda_5 y^5 + \cdots = \rho(e^{-y} - 1)$, 我们有 $-\rho_1 = \lambda_1 = \lambda$, $\rho_2 = \frac{1}{2} \lambda$, $-\rho_3 = \frac{1}{3} \lambda + \lambda_3$, $\rho_4 = \frac{1}{4} \lambda + \frac{3}{2} \lambda_3$, $-\rho_5 = \frac{1}{5} \lambda + \frac{7}{4} \lambda_3 + \lambda_5$;

$$(-1)^n \rho_n = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} \frac{k!}{n!} \lambda_k; \quad \lambda_n = - \sum_k \begin{Bmatrix} n \\ k \end{Bmatrix} \frac{k!}{n!} \rho_k$$

最初的一些值是 $\lambda_1 \approx .3979226812$, $\lambda_3 \approx -.0210096400$, $\lambda_5 \approx .0013749841$, $\lambda_7 \approx -.0000960351$ 。乱猜: $\lim_{k \rightarrow \infty} (-\lambda_{2k+1}/\lambda_{2k-1}) = 1/\pi^2$ 。

26. 由(39)左边为 $2S(1/x) - 5S(1/2x) + 2S(1/4x) - 2S(x) + 5S(2x) - 2S(4x)$; 由(44)右边是 $S(2x) - 2S(4x) + 2S(1/x) - S(1/2x) - 2S(x) + 4S(2x) - 4S(1/2x) + 2S(1/4x)$ 。 $x=1, x=1/\sqrt{2}$ 和 $x=\phi$ 的情况也许是最有趣的; 例如 $x=\phi$ 给出 $2G(4\phi) - 5G(2\phi) + G(\phi^2/2) - G(\phi^3) = 2G(2\phi^2)$ 。

27. 当 $n > 1$ 时, 由习题 1.2.11.2-4, $2\phi_n = [z^n] z \sum_{k \geq 0} 2^{-2k} \sum_{j=0}^{2^k-1} \sum_{l=0}^{2^k-1} (jz/2^k)^l = \sum_{k \geq 1} 2^{-k(n+1)} \sum_{l=0}^{2^k-1} \binom{n}{l} B_l 2^{k(n-l)/n}$; 而且当然 $\sum_{k \geq 1} 2^{-k(l+1)} = 1/(2^{l+1} - 1)$ 。

28. 如同在习题 6.3-34(b) 中那样, 令 $S_n(m) = \sum_{k=1}^{n-1} (1 - k/m)^n$ 和 $T_n(m) = 1/(e^{n/m} - 1)$, 我们求得 $S_n(m) = T_n(m) + O(e^{-n/m} n/m^2)$ 和 $2\phi_{n+1} = \sum_{j \geq 1} 2^{-2j} S_n(2^j) = \tau_n + O(n^{-3})$, 其中 $\tau_n = \sum_{j \geq 1} 2^{-2j} T_n(2^j)$ 。由于 $\tau_{n+1} < \tau_n$, 而且 $4\tau_{2n} - \tau_n = 1/(e^n - 1)$ 为正但指数地很小, 由此得出 $\tau_n = \Theta(n^{-2})$ 。更多细节可以通过写出

$$\sum_{j \geq 1} \frac{1}{2^{2j}} \frac{1}{e^{n/2^j} - 1} = \frac{1}{2\pi i} \sum_{j \geq 1} \int_{3/2-i\infty}^{3/2+i\infty} \frac{\zeta(z)\Gamma(z)n^{-z}}{2^{j(2-z)}} dz = \frac{1}{2\pi i} \int_{3/2-i\infty}^{3/2+i\infty} \frac{\zeta(z)\Gamma(z)n^{-z}}{2^{z-1} - 1} dz$$

得到, 这个积分是在极点 $2 + 2\pi ik/\ln 2$ 的剩余之和, 即 n^{-2} 乘以 $\pi^2/(6 \ln 2) + f(n)$, 其中

$$f(n) = 2 \sum_{k \geq 1} \Re(\zeta(2 + 2\pi ik/\ln 2) \Gamma(2 + 2\pi ik/\ln 2) \exp(-2\pi ik \lg n)/\ln 2)$$

是其“平均值”为零的 $\lg n$ 的一个周期函数。

29. (由 P. Flajolet 和 B. Vallée 给出的解) 如果 $f(x) = \sum_{k \geq 1} 2^{-k} g(2^k x)$ 和 $g^*(s) = \int_0^\infty g(x) x^{s-1} dx$, 则在适当的条件下, $f^*(s) = \sum_{k \geq 1} 2^{-k(s+1)} g^*(s) = g^*(s)/(2^{s+1} - 1)$, 以及 $f(x) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} f^*(s) x^{-s} ds$ 。令 $g(x) = 1/(1+x)$, 我们求得, 当 $0 < \Re s < 1$ 时, 在这种情况下的变换是 $g^*(s) = \pi/\sin \pi s$; 因此

$$f(x) = \sum_{k=1}^{\infty} \frac{1}{2^k} \frac{1}{1+2^k x} = \frac{1}{2\pi i} \int_{1/2-i\infty}^{1/2+i\infty} \frac{\pi x^{-s} ds}{(2^{s+1} - 1) \sin \pi s}$$

由此得出, 对于 $\Re s \leq 0$, $f(x)$ 是 $\frac{\pi}{\sin \pi s} x^{-s}/(2^{s+1} - 1)$ 的剩余之和, 即 $1+x \lg x + \frac{1}{2}x + xP(\lg x) - \frac{2}{1}x^2 + \frac{4}{3}x^3 - \frac{8}{7}x^4 + \dots$, 其中

$$P(t) = \frac{2\pi}{\ln 2} \sum_{m=1}^{\infty} \frac{\sin 2\pi mt}{\sinh(2m\pi^2/\ln 2)}$$

是其绝对值绝不超过 8×10^{-12} 的一个周期函数。(P(t)是如此之小这一事实促使 Brent 在他原来的论文中对之忽略不计。)

对于 $-1 < \Re s < 0$, $f(1/x)$ 的 Mellin 变换是 $f^*(-s) = \pi/((1-2^{1-s}) \sin \pi s)$; 因此

$f(1/x) = \frac{1}{2\pi i} \int_{-1/2-i\infty}^{1/2+i\infty} \frac{\pi}{\sin \pi s} x^{-s} ds / (1 - 2^{1-s})$, 而且我们现在要求对于 $\Re s \leq -1$, 被积项的剩余: $f(1/x) = \frac{1}{3}x - \frac{1}{7}x^2 + \dots$ 。[这个公式也可直接得到。] 我们有 $S_1(x) = 1 - f(x)$, 由此得出

$$G_1(x) = f(x) - f(1/x) = x \lg x + \frac{1}{2}x + xP(\lg x) - \frac{x^2}{1+x} + (1-x^2)\phi(x)$$

其中 $\phi(x) = \sum_{k=0}^{\infty} (-1)^k x^k / (2^{k+1} - 1)$ 。

30. 我们有 $G_2(x) = \Sigma_1(x) - \Sigma_1(1/x) + \Sigma_2(x) - \Sigma_2(1/x)$, 其中

$$\Sigma_1(x) = \sum_{k,l \geq 1} \frac{1}{2^{k+l}} \frac{1}{1 + 2^l(1 + 2^k x)}, \quad \Sigma_2(x) = \sum_{k,l \geq 1} \frac{1}{2^k} \frac{1}{1 - 2^l + 2^k x}$$

Mellin 变换是 $\Sigma_1^*(s) = \frac{\pi}{\sin \pi s} a(s) / (2^{s+1} - 1)$, $\Sigma_2^*(s) = \frac{\pi}{\sin \pi s} b(s) / (2^{s+1} - 1)$,

其中

$$a(s) = \sum_{l \geq 1} \frac{(1 + 2^{-l})^{s-1}}{2^{2l}} = \sum_{k \geq 0} \binom{s-1}{k} \frac{1}{2^{k+2} - 1}$$

$$b(s) = \sum_{l \geq 1} (2^l + 1)^{s-1} = \sum_{k \geq 0} \binom{s-1}{k} \frac{1}{2^{k+1} - 1}$$

因此对于 $0 \leq x \leq 1$, 我们得到下列展开式:

$$\Sigma_1(x) = a(0) + a(-1)x(\lg x + \frac{1}{2}) - a'(1)x/\ln 2 + xA(\lg x) -$$

$$\sum_{k \geq 2} \frac{2^{k-1}}{2^{k-1} - 1} a(-k)(-x)^k$$

$$\Sigma_2(x) = b(0) + b(-1)x(\lg x + \frac{1}{2}) - b'(1)x/\ln 2 + xB(\lg x) -$$

$$\sum_{k \geq 2} \frac{2^{k-1}}{2^{k-1} - 1} b(-k)(-x)^k$$

$$\Sigma_1(1/x) = \sum_{k \geq 1} \frac{-a(k)(-x)^k}{2^{k+1} - 1}$$

$$\Sigma_2(1/x) = \sum_{k \geq 1} \frac{(-x)^k}{2^{k+1} - 1} \left(\lg x - \hat{b}(k) - \frac{1}{2} - \frac{1}{2^{k+1} - 1} + \frac{H_{k-1}}{\ln 2} + P_k(\lg x) \right)$$

$$\hat{b}(s) = \sum_{k=0}^{s-2} \binom{s-1}{k} \frac{1}{2^{k+1} - 1}$$

$$A(t) = \frac{1}{\ln 2} \sum_{m \geq 1} \Re \left(\frac{2\pi i}{\sinh(2m\pi^2/\ln 2)} a(-1 + 2m\pi i/\ln 2) e^{-2m\pi i t} \right)$$

$$B(t) = \frac{1}{\ln 2} \sum_{m \geq 1} \Re \left(\frac{2\pi i}{\sinh(2m\pi^2/\ln 2)} b(-1 + 2m\pi i/\ln 2) e^{-2m\pi i t} \right)$$

$$P_k(t) = \frac{1}{\ln 2} \sum_{m=1}^{\infty} \Re \left(\frac{2\pi i}{\sinh(2m\pi^2/\ln 2)} \binom{k-1}{k-1} \frac{2m\pi i/\ln 2}{1} e^{-2m\pi i t} \right)$$

34. 利用一个十分不同于 Brent 的方法, Brigitte Vallée [*Algorithmica* **22**(1998), 660~685] 已经求得对于算法 B 的一个优美而严格的分析。确实, 她的方法很不同, 人们还不知道用来预测和 Brent 的启发式模型一样的特性。因此, 分析二进制 gcd 算法的问题(现在已第一次获得严格解决), 仍在诱发高等数学的更多亟待解决的问题。

35. 由归纳法, 当 $m \geq n$ 时, 长度是 $m + \lfloor n/2 \rfloor + 1 - \lfloor m = n = 1 \rfloor$ 。但习题 37 表明, 算法不会像这么慢。

36. 设 $a_n = (2^n - (-1)^n)/3$; 则 $a_0, a_1, a_2, \dots = 0, 1, 1, 3, 5, 11, 21, \dots$ 。(这个数列的二进表示有有趣的 0 和 1 的模式。注意, $a_n = a_{n-1} + 2a_{n-2}$, 而且 $a_n + a_{n+1} = 2^n$ 。)对于 $m > n$, 令 $u = 2^{m+1} - a_{n+2}$, $v = a_{n+2}$ 。对于 $m = n > 0$, 令 $u = \max(a_{n+2}, 2a_{n+1})$ 和 $v = a_{n+3} - u$ 。对于 $m = n > 0$ 情况的另一个例子是 $u = 2^{n+1} - 2$, $v = 2^{n+1} - 1$; 这个选择要做更多的移位, 并给出 $B = 1, C = n + 1, D = 2n, E = n$, 这是对于程序 B 最糟的情况。

37. (由 J. O. Shallit 给出的解) 这个问题看起来有必要证明比要求证明的更多的东西。令 $S(u, v)$ 是在输入 u 和 v 时由算法 B 所采取的减法步骤的数目。我们将证明, $S(u, v) \leq \lg(u + v)$ 。这意味着, 如所要求的那样, $S(u, v) \leq \lfloor \lg(u + v) \rfloor \leq \lfloor \lg 2 \max(u, v) \rfloor = 1 + \lfloor \lg \max(u, v) \rfloor$ 。

注意 $S(u, v) = S(v, u)$ 。如果 u 为偶, $S(u, v) = S(u/2, v)$; 因此我们可以假定 u 和 v 为奇。我们还可以假定 $u > v$, 因为 $S(u, v) = 1$ 。于是由归纳法, $S(u, v) = 1 + S((u - v)/2, v) \leq 1 + \lg((u - v)/2 + v) = \lg(u + v)$ 。

顺便指出, 由此得出, 要求 n 个减法步骤的最小情况是 $u = 2^{n-1} + 1, v = 2^{n-1} - 1$ 。

38. 记住操作数的最高有效字和最低有效字(最高有效字用来猜测 t 的符号, 而最低有效字用来确定右移的数量), 同时构造单精度整数的一个 2×2 矩阵 A , 使得 $A \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u'w \\ v'w \end{pmatrix}$, 其中 w 是计算机字长而 u' 和 v' 比 u 和 v 小。(代替用 2 除被模拟的奇操作数, 我们用 2 来乘另一个操作数, 直到在恰好 $\lg w$ 次移位后得到 w 的倍数为止。)实验证明, 至少在一台计算机上, 这个算法的运行速度比算法 L 快 3 倍。使用习题 40 的一个类似的算法, 我们不需要最高有效字。

J. Sorenson, *J. Algorithms* **16** (1994), 110 ~ 144; Shallit 和 Sorenson, *Lecture Notes in Comp. Sci.* **877**(1994), 169~183, 已经描述了一个可能更快的二进制算法。

39. (Michael Penk 给出的解。)

Y1. [求 2 的幂] 和步骤 B1 相同。

Y2. [初始化] 置 $(u_1, u_2, u_3) \leftarrow (1, 0, u)$ 和 $(v_1, v_2, v_3) \leftarrow (v, 1 - u, v)$ 。如果

u 是奇数, 则置 $(t_1, t_2, t_3) \leftarrow (0, -1, -v)$ 并转到 Y4。否则 $(t_1, t_2, t_3) \leftarrow (1, 0, u)$ 。

Y3. [t_3 折半] 如果 t_1 和 t_2 都是偶数, 则置 $(t_1, t_2, t_3) \leftarrow (t_1, t_2, t_3)/2$; 否则置 $(t_1, t_2, t_3) \leftarrow (t_1 + v, t_2 - u, t_3)/2$ 。(在下一情况下, $t_1 + v$ 和 $t_2 - u$ 都将为偶数。)

Y4. [t_3 为偶数?] 如果 t_3 为偶数, 则转回 Y3。

Y5. [恢复 $\max(u_3, v_3)$] 如果 t_3 为正, 则置 $(u_1, u_2, u_3) \leftarrow (t_1, t_2, t_3)$; 否则置 $(v_1, v_2, v_3) \leftarrow (v - t_1, -u - t_2, -t_3)$ 。

Y6. [减] 置 $(t_1, t_2, t_3) \leftarrow (u_1, u_2, u_3) - (v_1, v_2, v_3)$ 。然后若 $t_1 \leq 0$, 则置 $(t_1, t_2) \leftarrow (t_1 + v, t_2 - u)$ 。若 $t_3 \neq 0$, 则转回 Y3, 否则算法终止且 $(u_1, u_2, u_3 \cdot 2^k)$ 作为输出。■

显然, (16) 中的关系被保持, 而且在执行步骤 Y2 ~ Y6 中的每一步后 $0 \leq u_1, v_1, t_1 \leq v, 0 \geq u_2, v_2, t_2 \geq -u, 0 < u_3 \leq u, 0 < v_3 \leq v$ 。如果在执行步骤 Y2 后, u 为奇数, 则步骤 Y3 可被简化, 因为 t_1 和 t_2 都为偶数当且仅当 t_2 为偶数; 类似地, 如果 v 为奇数, 则 t_1 和 t_2 都为偶数当且仅当 t_1 为偶数。于是, 如同在算法 X 中那样, 假定在执行步骤 Y2 之后 v 为奇数, 则有可能取消涉及 u_2, v_2 和 t_2 的所有计算。这个条件通常是预先知道的(例如, 当 v 为素数且我们试图计算 $u^{-1} \bmod v$ 时)。

关于习题 40 中算法的一个类似的扩充, 也请参见 A. W. Bojanczyk 和 R. P. Brent, *Computers and Math.* **14** (1987), 233。

40. 令 $m = \lg \max(|u|, |v|)$ 。在执行了步骤 K3 中的操作 $c \leftarrow c + 1$ s 次之后, 我们可归纳地证明 $|u| \leq 2^{m-(s-c)/2}, |v| \leq 2^{m-(s+c)/2}$ 。因此 $s \leq 2m$ 。如果 K2 被执行 t 次, 我们有 $t \leq s + 2$, 因为除了头一次和最后一次外, s 每次都增加。[参见 VL-SI'83 (North-Holland, 1983), 145 ~ 154。]

注: 当 $u = 1$ 和 $v = 3 \cdot 2^k - 1$ 以及 $k \geq 2$ 时, 我们有 $m = k + 2, s = 2k, t = k + 4$ 。当在由 $u_0 = 3, u_1 = 1, u_{j+1} = \min(|3u_j - 16u_{j-1}|, |5u_j - 16u_{j-1}|)$ 定义的序列中 $u = u_j$ 和 $v = 2u_{j-1}$ 时, 我们有 $s = 2j + 2, t = 2j + 3$ 以及(经验地) $m \approx \phi j$ 。 t 可以渐近地大于 $2m/\phi$ 吗?

41. 一般地说, 由于 $(a^k - 1) \bmod (a^v - 1) = a^{u \bmod v} - 1$ (参见等式 4.3.2-(20)), 对于所有正整数 a , 我们求得 $\gcd(a^m - 1, a^n - 1) = a^{\gcd(m, n)} - 1$ 。

42. 对于 $k = 1, 2, 3, \dots$ 从第 $2k, 3k, 4k$ 等列, 减去第 k 列。结果是 x_k 在 k 列的对角线上的一个三角矩阵, 其中 $m = \sum_{d \mid m} x_d$ 。由此得出, $x_m = \varphi(m)$, 所以行列式是 $\varphi(1)\varphi(2)\cdots\varphi(n)$ 。

[通常, “Smith 行列式”, 其中对于一个任意的函数 f , 元素 (i, j) 是 $f(\gcd(i, j))$, 由相同的论证, 等于 $\prod_{m=1}^n \sum_{d \mid m} \mu(m/d) f(d)$ 。参见 L. E. Dickson, *History of*

the Theory of Numbers 1 (Carnegie Inst. of Washington, 1919), 122~123.]

4.5.3 小节

1. 运行时间大约是 $19.02T + 6$, 仅仅比程序 4.5.2A 慢一点。

$$2. \begin{bmatrix} K_n(x_1, x_2, \dots, x_{n-1}, x_n) & K_{n-1}(x_1, x_2, \dots, x_{n-1}) \\ K_{n-1}(x_2, \dots, x_{n-1}, x_n) & K_{n-2}(x_2, \dots, x_{n-1}) \end{bmatrix}^0$$

3. $K_n(x_1, \dots, x_n)$ 。

4. 由归纳法, 或者取习题 2 中矩阵乘积的行列式。

5. 当诸 x 为正时, (9) 中诸 q 也为正, 且 $q_{n+1} > q_{n-1}$; 因此 (9) 是递减项的交错级数, 而且它收敛当且仅当 $q_n q_{n+1} \rightarrow \infty$ 。由归纳法, 如果诸 x 大于 ϵ , 我们有 $q_n \geq (1 + \epsilon/2)^n c$, 其中 c 选成足够小使得对于 $n=1$ 和 2 , 这个不等式成立。但如果 $x_n = 1/2^n$, 我们有 $q_n \leq 2 - 1/2^n$ 。

6. 只须证明 $A_1 = B_1$; 而且每当 x_1, \dots, x_n 为正整数时, $0 \leq //x_1, \dots, x_n// < 1$, 由此事实我们有 $B_1 = \lfloor 1/X \rfloor = A_1$ 。

7. 仅有 $12 \cdots n$ 和 $n \cdots 21$ 。(变量 x_k 恰在 $F_k F_{n+1-k}$ 项中出现; 因此 x_1 和 x_n 仅可排列成 x_1 和 x_n 。如果 x_1 和 x_n 都为这个排列所固定, 则由归纳法得出 x_2, \dots, x_{n-1} 也是固定的。)

8. 这等价于

$$\frac{K_{n-2}(A_{n-1}, \dots, A_2) - XK_{n-1}(A_{n-1}, \dots, A_1)}{K_{n-1}(A_n, \dots, A_2) - XK_n(A_n, \dots, A_1)} = -\frac{1}{X_n}$$

而且由 (6) 这等价于

$$X = \frac{K_{n-1}(A_2, \dots, A_n) + X_n K_{n-2}(A_2, \dots, A_{n-1})}{K_n(A_1, \dots, A_n) + X_n K_{n-1}(A_1, \dots, A_{n-1})}$$

9. a) 由定义。b), d) 先给出对于 $n=1$ 的证明, 然后应用 a) 得到对于一般 n 的结果。c) 当 $n=k+1$ 时, 证明之, 然后应用 a)。

10. 如果 $A_0 > 0$, 则 $B_0 = 0, B_1 = A_0, B_2 = A_1, B_3 = A_2, B_4 = A_3, B_5 = A_4, m = 5$ 。如果 $A_0 = 0$, 则 $B_0 = A_1, B_1 = A_2, B_2 = A_3, B_3 = A_4, m = 3$ 。如果 $A_0 = -1$, 且 $A_1 = 1$, 则 $B_0 = -(A_2 + 2), B_1 = 1, B_2 = A_3 - 1, B_3 = A_4, m = 3$ 。如果 $A_0 = -1$ 且 $A_1 > 1$, 则 $B_0 = -2, B_1 = 1, B_2 = A_1 - 2, B_3 = A_2, B_4 = A_3, B_5 = A_4, m = 5$ 。如果 $A_0 < -1$, 则 $B_0 = -1, B_1 = 1, B_2 = -A_0 - 2, B_3 = 1, B_4 = A_1 - 1, B_5 = A_2, B_6 = A_3, B_7 = A_4, m = 7$ 。[实际上, 最后三种情况包含八种子情况; 只要诸 B 中有一个被置为 0, 则利用习题 9c) 的规则可知, 这些值应该“一起消失”。例如, 如果 $A_0 = -1, A_1 = A_3 = 1$, 则我们实际上有 $B_0 = -(A_2 + 2), B_1 = A_4 + 1, m = 1$ 。当 $A_0 = -2, A_1 = 1$ 时, 产生双重消失。]

11. 设 $q_n = K_n(A_1, \dots, A_n)$, $q'_n = K_n(B_1, \dots, B_n)$, $p_n = K_{n+1}(A_0, \dots, A_n)$, $p'_n = K_{n+1}(B_0, \dots, B_n)$ 。由(5)和(11)我们有 $X = (p_m + p_{m-1}X_m)/(q_m + q_{m-1}X_m)$, $Y = (p'_m + p'_{m-1}Y_m)/(q'_m + q'_{m-1}Y_m)$; 因此, 如果 $X_m = Y_m$, 则由(8), X 和 Y 之间的上述关系成立。反之, 如果 $X = (qY + r)/(sY + t)$, 且 $|qt - rs| = 1$, 则我们可以假定 $s \geq 0$, 而且我们可以对 s 用归纳法证明 X 和 Y 的部分商终将一致。由习题 9(d), 当 $s = 0$ 时, 这结果显然。如果 $s > 0$, 设 $q = as + s'$, 其中 $0 \leq s' < s$, 则 $X = a + 1/((sY + t)/(s'Y + r - at))$; 由于 $s(r - at) - ts' = sr - tq$, 且 $s' < s$, 由归纳法和习题 10, 我们知道 X 和 Y 的部分商终将一致。[*J. de Math. Pures et Appl.* **15** (1850), 153 ~ 155。通过对这个证明更仔细的检查, 习题 10 中 m 总为奇数这个事实表明, $X_m = Y_m$ 当且仅当 $X = (qY + r)/(sY + t)$, 其中 $qt - rs = (-1)^{m-n}$ 。]

12. a) 由于 $V_n V_{n+1} = D - U_n^2$, 我们知道 $D - U_{n+1}^2$ 是 V_{n+1} 的倍数; 因此由归纳法, $X_n = (\sqrt{D} - U_n)/V_n$, 其中 U_n, V_n 为整数。[注: 基于这一过程的一个算法对于整数的二次方程求解有许多应用。比如说, 参见 H. Davenport, *The Higher Arithmetic* (London: Hutchinson, 1952); W. J. LeVeque, *Topics in Number Theory* (Reading, Mass.: Addison-Wesley, 1956); 也可参见 4.5.4 小节。由习题 1.2.4-35, 当 $V_{n+1} > 0$ 时我们有 $A_{n+1} = \lfloor (L\sqrt{D} + U_n)/V_{n+1} \rfloor$, 当 $V_{n+1} < 0$ 时 $A_{n+1} = \lfloor (\lfloor \sqrt{D} \rfloor + 1 + U_n)/V_{n+1} \rfloor$; 因此这样一个算法只须对正整数 $\lfloor \sqrt{D} \rfloor$ 来工作。而且当 V_{n+1} 正被确定时, 恒等式 $V_{n+1} = A_n(U_{n-1} - U_n) + V_{n-1}$ 使得没有必要来做除法。]

b) 设 $Y = (-\sqrt{D} - U)/V$, $Y_n = (-\sqrt{D} - U_n)/V_n$ 。在 a) 的证明中用 $-\sqrt{D}$ 代替 \sqrt{D} , 所述的恒等式显然成立。我们有

$$Y = (p_n/Y_n + p_{n-1})/(q_n/Y_n + q_{n-1})$$

其中 p_n 和 q_n 在本题 c) 中定义, 因此

$$Y_n = (-q_n/q_{n-1})(Y - p_n/q_n)/(Y - p_{n-1}/q_{n-1})$$

但由(12), p_{n-1}/q_{n-1} 和 p_n/q_n 都极其接近于 X ; 因为 $X \neq Y$, 对于所有大的 n , $Y - p_n/q_n$ 和 $Y - p_{n-1}/q_{n-1}$ 将有与 $Y - X$ 相同的符号。这证明, 对于所有大的 n , $Y_n < 0$; 因此 $0 < X_n < X_n - Y_n = 2\sqrt{D}/V_n$; V_n 必为正。而且 $U_n < \sqrt{D}$, 因为 $X_n > 0$ 。因此 $V_n < 2\sqrt{D}$, 因为 $V_n \leq A_n V_n < \sqrt{D} + U_{n-1}$ 。

最后我们要来证明 $U_n > 0$ 。由于 $X_n < 1$, 我们有 $U_n > \sqrt{D} - V_n$, 所以我们只须考虑 $V_n > \sqrt{D}$ 的情况。于是 $U_n = A_n V_n - U_{n-1} \geq V_n - U_{n-1} > \sqrt{D} - U_{n-1}$, 而且如同我们已经观察过的那样, 它是正的。

注: 在重复循环中, $\sqrt{D} + U_n = A_n V_n + (\sqrt{D} - U_{n-1}) > V_n$; 因此 $\lfloor (\sqrt{D} + U_{n+1})/V_{n+1} \rfloor = \lfloor A_{n+1} + V_n/(\sqrt{D} + U_n) \rfloor = A_{n+1} = \lfloor (\sqrt{D} + U_n)/V_{n+1} \rfloor$ 。换言之, A_{n+1} 由 U_{n+1} 和 V_{n+1} 确定; 我们在这个周期中可以由 (U_n, V_n) 的后继 $(U_{n+1},$

V_{n+1}) 确定 (U_n, V_n) 。事实上, 当 $0 < V_n < \sqrt{D} + U_n$ 和 $0 < U_n < \sqrt{D}$ 时, 上述论证证明 $0 < V_{n+1} < \sqrt{D} + U_{n+1}$ 和 $0 < U_{n+1} < \sqrt{D}$; 而且, 如果数偶 (U_{n+1}, V_{n+1}) 跟在 (U', V') 之后且 $0 < V' < \sqrt{D} + U'$ 和 $0 < U' < \sqrt{D}$, 则 $U' = U_n$ 和 $V' = V_n$ 。因此 (U_n, V_n) 是循环的一部分当且仅当 $0 < V_n < \sqrt{D} + U_n$ 和 $0 < U_n < \sqrt{D}$ 。

$$c) \frac{-V_{n+1}}{V_n} = X_n Y_n = \frac{(q_n X - p_n)(q_n Y - p_n)}{(q_{n-1} X - p_{n-1})(q_{n-1} Y - p_{n-1})}$$

也有一个伴随的恒等式, 即

$$V p_n p_{n-1} + U(p_n q_{n-1} + p_{n-1} q_n) + ((U^2 - D)/V) q_n q_{n-1} = (-1)^n U_n$$

d) 对于某个 $n \neq m$, 如果 $X_n = X_m$, 则 X 是满足二次方程 $(q_n X - p_n)/(q_{n-1} X - p_{n-1}) = (q_m X - p_m)/(q_{m-1} X - p_{m-1})$ 的一个无理数。

奠定本习题基础的思想至少可回溯到公元 1073 年之前在印度的 Jayadeva; 参见 K. S. Shukla, *Ganita* 5 (1954), 1~20; C. -O Selenius, *Historia Math.* 2 (1975), 167~184。在 1750 年之前, 在日本也发现了它的某些方面, 请见 Y. Mikami, *The Development of Mathematics in China and Japan* (1913), 223~229。但是关于二次式连分数理论的重要原理大部分归功于欧拉 [Novi. Comment Acad. Sci. Petrop 11 (1765), 28~66] 以及拉格朗日 [Hist. Acad. Sci. 24 (Berlin: 1768), 111~180]。

14. 像在习题 9 中一样, 我们只须当 e 是最后的部分商时验证所述的恒等式, 而这个验证是很简单的。现在 Hurwitz 的规则给出 $2/e = //1, 2, 1, 2, 0, 1, 1, 1, 1, 1, 0, 2, 3, 2, 0, 1, 1, 3, 1, 1, 0, 2, 5, \dots //$ 。取倒数, 像在习题 9 中那样把零都除去, 并且注意出现的模式, 我们发现 (参考习题 16) $e/2 = 1 + //2, 2m+1, 3, 1, 2m+1, 1, 3//$, $m \geq 0$ 。[Schriften der phys.-ökon. Gesellschaft zu Königsberg 32 (1891), 59~62。Hurwitz 也在 Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich 41 (1896), Jubelband II, 34~64, § 2 中说明怎样以一个任意的正整数来乘。]

15. (这一过程利用四个整数 (A, B, C, D) , 它的意义恒为“我们剩下的工作是输出 $(Ay + B)/(Cy + D)$ 的连分数, 其中 y 是有待输入的数”。) 开始时置 $j \leftarrow k \leftarrow 0$, $(A, B, C, D) \leftarrow (a, b, c, d)$; 然后输入 x_j 并置 $(A, B, C, D) \leftarrow (Ax_j + B, A, Cx_j + D, C)$, $j \leftarrow j + 1$ 一次或多次直到 $C + D$ 的符号和 C 相同。(当 $j \geq 1$ 且输入未结束时, 我们知道 $1 < y < \infty$; 因此当 $C + D$ 的符号和 C 相同时, 我们知道 $(Ay + B)/(Cy + D)$ 位于 $(A + B)/(C + D)$ 和 A/C 之间。) 一般的步骤是: 如果没有整数严格地位于 $(A + B)/(C + D)$ 和 A/C 之间, 则输出 $X_k \leftarrow \min(\lfloor A/C \rfloor, \lfloor (A + B)/(C + D) \rfloor)$, 并且置 $(A, B, C, D) \leftarrow (C, D, A - X_k C, B - X_k D)$, $k \leftarrow k + 1$; 否则输入 x_j 并置 $(A, B, C, D) \leftarrow (Ax_j + B, A, Cx_j + D, C)$, $j \leftarrow j + 1$ 。无限地重复这个一般步骤。然而, 一旦最后的 x_j 被输入, 则算法立即改换动作: 利用欧几里得算法输出 $(Ax_j + B)/(Cx_j + D)$ 的连分数, 并终止。

下列表格解决所要求的例子的问题, 其中矩阵 $\begin{pmatrix} B & A \\ D & C \end{pmatrix}$ 在左上角开始, 然后输

入时右移一位,输出时降一位:

	i_j	-1	5	1	1	1	2	1	2	∞
X_k	39	97	-58	-193						
-2	-25	-62	37	123						
2			16	53						
3			5	17	22					
7			1	2	3	5				
1				3	1	4	5	14		
1					2	1	3	7		
1							2	7	9	25
12							1	0	1	2
2										1
∞										0

M. Mendès France 已经证明:相对于每个商输入的商输出个数渐近地界于 $1/r$ 和 r 之间,其中 $r = 2\lfloor L(|ad - bc|)/2 \rfloor + 1$,且 L 是习题 38 中定义的函数;这个界是最好的。[Topics in Number Theory, P. Turán 编, Colloquia Math. Soc. János Bolyai 13 (1976), 183~194.]

Gosper 还证明了上述算法可以推广来从 x 和 y 的连分数计算 $(axy + bx + cy + d)/(Axy + Bx + Cy + D)$ 的连分数(特别是计算和与积。)[MIT AI Laboratory Memo 239(29, 1972 年 2 月 29 日), Hack 101.]

16. 用归纳法不难证明 $f_n(z) = z/(2n+1) + O(z^3)$ 是一个在原点的一个邻域中有一个收敛的幂级数的奇函数,而且它满足给定的微分方程。因此

$$f_0(z) = // z^{-1} + f_1(z) // = \cdots = // z^{-1}, 3z^{-1}, \cdots, (2n+1)z^{-1} + f_{n+1}(z) //$$

剩下的是要证明 $\lim_{n \rightarrow \infty} // z^{-1}, 3z^{-1}, \cdots, (2n+1)z^{-1} // = f_0(z)$ 。[实际上,欧拉在 24 岁时,就获得了对于更为一般得多的微分方程 $f'_n(z) = az^m + bf_n(z)z^{m-1} + cf_n(z)^2$ 的连分数展开,但他没有费心去证明收敛性,因为在 18 世纪时能有形式处理和这样的直觉就已很了不起了。]

有好几种方法可证明所要求的极限等式。首先,命 $f_n(z) = \sum_k a_{nk} z^k$,我们可以从等式

$$(2n+1)a_{n1} + (2n+3)a_{n3}z^2 + (2n+5)a_{n5}z^4 + \cdots = 1 - (a_{n1}z + a_{n3}z^3 + a_{n5}z^5 + \cdots)^2$$

论断 $(-1)^k a_{n(2k+1)}$ 是形如 $c_k/(2n+1)^{k+1}(2n+b_{k1})\cdots(2n+b_{kk})$ 的项之和,其中 c_k 和 b_{km} 都是同 n 无关的正整数。例如,我们有 $-a_{n7} = 4/(2n+1)^4(2n+3)(2n+5)(2n+7) + 1/(2n+1)^4(2n+3)^2(2n+7)$ 。于是 $|a_{(n+1)k}| \leq |a_{nk}|$,而且对于 $|z| < \pi/2$, $|f_n(z)| \leq \tan|z|$ 。 $f_n(z)$ 的这个一致的上限使得收敛性的证明非常简单。仔细研究这个论证即可发现,对于 $|z| < \pi\sqrt{2n+1}/2$, $f_n(z)$ 的幂级数实际上收敛;因此当 n 增大时 $f_n(z)$ 的奇异性越来越远离原点,所以在整个复平面上,这个连分数实际上表示 $\tanh z$ 。

另一个证明给出一种不同类型的进一步信息:如果我们命

$$A_n(z) = n! \sum_{k=0}^n \binom{2n-k}{n} \frac{z^k}{k!} = \sum_{k \geq 0} \frac{(n+k)!}{k!(n-k)!} z^{n-k} = n! z^n {}_2F_0(n+1, -n; -1/z)$$

$$\text{则 } A_{n+1}(z) = \sum_{k \geq 0} \frac{(n+k+1)!}{k!(n+1-k)!} \frac{((4n+2)k + (n+1-k)(n-k))}{k!(n+1-k)!} z^{n+1-k} = (4n+2)A_n(z) + z^2 A_{n-1}(z)$$

用归纳法,由此得出

$$K_n\left(\frac{1}{z}, \frac{3}{z}, \dots, \frac{2n-1}{z}\right) = \frac{A_n(2z) + A_n(-2z)}{2^{n+1}z^n}$$

$$K_{n-1}\left(\frac{3}{z}, \dots, \frac{2n-1}{z}\right) = \frac{A_n(2z) - A_n(-2z)}{2^{n+1}z^n}$$

因此

$$\|z^{-1}, 3z^{-1}, \dots, (2n-1)z^{-1}\| = \frac{A_n(2z) - A_n(-2z)}{A_n(2z) + A_n(-2z)}$$

我们来证明这个比例趋近于 $\tanh z$ 。由等式 1.2.9-(11) 和 1.2.6-(24) 知

$$e^z A_n(-z) = n! \sum_{m \geq 0} z^m \left(\sum_{k=0}^n \binom{m}{k} \binom{2n-k}{n} (-1)^k \right) = \sum_{m \geq 0} \binom{2n-m}{n} z^m \frac{n!}{m!}$$

因此

$$e^z A_n(-z) - A_n(z) = R_n(z) = (-1)^n z^{2n+1} \sum_{k \geq 0} \frac{(n+k)!}{(2n+k+1)! k!} z^k$$

我们现在有 $(e^{2z} - 1)(A_n(2z) + A_n(-2z)) - (e^{2z} + 1)(A_n(2z) - A_n(-2z)) = 2R_n(2z)$; 因此

$$\tanh z - \|z^{-1}, 3z^{-1}, \dots, (2n-1)z^{-1}\| = \frac{2R_n(2z)}{(A_n(2z) + A_n(-2z))(e^{2z} + 1)}$$

于是我们对于差有一个精确的公式。当 $|2z| \leq 1$ 时, 因子 $e^{2z} + 1$ 与 0 的距离有界 $|R_n(2z)| \leq e n! / (2n+1)!$, 而且

$$\frac{1}{2} |A_n(2z) + A_n(-2z)| \geq n! \left(\binom{2n}{n} - \binom{2n-2}{n} - \binom{2n-4}{n} - \binom{2n-6}{n} - \dots \right) \geq \frac{(2n)!}{n!} \left(1 - \frac{1}{4} - \frac{1}{16} - \frac{1}{64} - \dots \right) = \frac{2}{3} \frac{(2n)!}{n!}$$

于是, 甚至对于 z 的复值, 收敛仍是非常快的。

为了从这个连分数转到 e^z 的连分数, 我们有 $\tanh z = 1 - 2/(e^{2z} + 1)$; 因此, 通过简单的运算我们得到 $(e^{2z} + 1)/2$ 的连分数表示。Hurwitz 规则给出了 $e^{2z} + 1$ 的展开式, 由之我们可以减 1。对于 n 为奇数,

$$e^{-2/n} = // 1, 3mn + \lfloor n/2 \rfloor, (12m+6)n, (3m+2)n + \lfloor n/2 \rfloor, 1 //, m \geq 0$$

另一个推导已由 C. S. Davis 给出, 见 *J. London Math. Soc.* **20** (1945), 194~198。
e 的连分数首先是由 Roger Cotz 凭经验找出的, *Philosophical Transactions* **29** (1714), 5~45, Proposition 1, Scholium 3。欧拉于 1731 年 11 月 25 日写信给哥德巴赫介绍了他的结果 [P. H. Fuss 编, *Correspondance Mathématique et Physique*, **1** (St. Petersburg: 1843), 56~60], 而且最终他在 *Commentarii Acad. Sci. Petropolitansæ* **9** (1737), 98~137; **11** (1739), 32~81 上发表了更完全的描述。

17. (b) $// x_1 - 1, 1, x_2 - 2, 1, x_3 - 2, 1, \dots, 1, x_{2n-1} - 2, 1, x_{2n} - 1 //$ 。 [注: 使用恒等式

$$K_{m+n+1}(x_1, \dots, x_m, -x, y_n, \dots, y_1) = (-1)^{n-1} K_{m+n+2}(x_1, \dots, x_{m-1}, x_m - 1, 1, x - 1, -y_n, \dots, -y_1)$$

可以从连分数删去负参数, 两次应用该式后我们得到

$$K_{m+n+1}(x_1, \dots, x_m, -x, y_n, \dots, y_1) = -K_{m+n+3}(x_1, \dots, x_{m-1}, x_m - 1, 1, x - 2, 1, y_n - 1, y_{n-1}, \dots, y_1)$$

一个类似的恒等式出现于习题 41 中。]

$$(c) 1 + // 1, 1, 3, 1, 5, 1, \dots // = 1 + // 2m+1, 1 //, m \geq 0.$$

18. 由于由等式(5)和(8), 我们有 $K_m(a_1, a_2, \dots, a_m) // a_1, a_2, \dots, a_m, x // = K_{m-1}(a_2, \dots, a_m) + (-1)^m / (K_{m-1}(a_1, \dots, a_{m-1}) + K_m(a_1, a_2, \dots, a_m)x)$, 我们也有 $K_m(a_1, a_2, \dots, a_m) // a_1, a_2, \dots, a_m, x; a_1, a_2, \dots, a_m, x_2, a_1, a_2, \dots, a_m, x_3, a_1, \dots // = K_{m-1}(a_2, \dots, a_m) + // (-1)^m (C + Ax_1), C + Ax_2, (-1)^m (C + Ax_3), \dots //$, 其中 $A = K_m(a_1, a_2, \dots, a_m)$ 和 $C = K_{m-1}(a_2, \dots, a_m) + K_{m-1}(a_1, \dots, a_{m-1})$ 。结果由(6), 所述的差是 $(K_{m-1}(a_2, \dots, a_m) - K_{m-1}(a_1, \dots, a_{m-1})) / K_m(a_1, a_2, \dots, a_m)$ 。 [$m = 2$ 的情况是由欧拉在 *Commentarii Acad. Sci. Petropolitansæ* **9** (1737), 98~137, § 24~26 上讨论的。]

19. 对于 $1 \leq k \leq N$, 这个和是 $\log_b((1+x)(N+1)/(N+1+x))$ 。

20. 设 $H = SG$, $g(x) = (1+x)G'(x)$, $h(x) = (1+x)H'(x)$ 。则(37)意味着 $h(x+1)/(x+2) - h(x)/(x+1) = -(1+x)^{-2}g(1/(1+x))/(1+1/(1+x))$ 。

21. $\varphi(x) = c/(cx+1)^2 + (2-c)/((c-1)x+1)^2$, $U\varphi(x) = 1/(x+c)^2$ 。当 $c \leq 1$ 时, 在 $x=0$ 处出现 $\varphi(x)/U\varphi(x)$ 的极小值, 该值为 $2c^2 \leq 2$ 。当 $c \geq \phi$ 时, 在 $x=1$ 处出现极小值, 该值小于等于 ϕ^2 。当 $c \approx 1.31266$ 时, 在 $x=0$ 和 $x=1$ 处的值接近于相等且极小值大于 3.2; 得到界 $(0.29)^n \varphi \leq U^n \varphi \leq (0.31)^n \varphi$ 。由精心选定的线性组合 $Tg(x) = \sum a_j/(x+c_j)$ 可得到更好的界。

23. 对于 $x_0 = 0, x_1 = x, x_2 = x + \epsilon$, 并令 $\epsilon \rightarrow 0$, 通过习题 4.6.4-15 的内插公式, 我们有一般的恒等式如下, 每当 R_n 是具有连续的二阶导数的函数时, 对于 0 和 x 之间的某 $\theta_n(x)$, $R'_n(x) = (R_n(x) - R_n(0))/x + \frac{1}{2}xR''_n(\theta_n(x))$ 。因此在这种情况下

下, $R'_n(x) = O(2^{-n})$ 。

24. ∞ 。[A. Khinchin 在 *Compos. Math.* 1 (1935), 361~382 证明了对几乎所有的实数 X , X 的头 n 个部分商之和 $A_1 + \cdots + A_n$ 渐近地趋于 $n \lg n$ 。习题 35 表明对于有理数 X , 这个特性是不同的。]

25. 任何一组区间的并都可以写成一些不相交区间的并, 因为我们有 $\bigcup_{k \geq 1} I_k = \bigcup_{k \geq 1} (I_k \setminus \bigcup_{1 \leq j < k} I_j)$, 而这是一个不相交的并, 其中 $I_k \setminus \bigcup_{1 \leq j < k} I_j$ 可以表达成不相交区间的有限的并。因此, 利用有理数的某种枚举方法, 我们可以取 $\mathcal{J} = \bigcup I_k$, 其中 I_k 是包含 $[0, 1]$ 中的第 k 个有理数的长度为 $\epsilon/2^k$ 的一个区间。在这种情况下, $\mu(\mathcal{J}) \leq \epsilon$, 但对于所有的 n , $|\mathcal{J} \cap P_n| = n$ 。

26. 出现的连分数 $// A_1, \dots, A_t //$ 恰巧是使得 $A_1 > 1, A_t > 1$ 的那些, 而且 $K_1(A_1, A_2, \dots, A_t)$ 是 n 的一个因子。因此(6)完成了这个证明。[注: 如果 $m_1/n = // A_1, \dots, A_t //$ 而且 $m_2/n = // A_t, \dots, A_1 //$, 其中 m_1 和 m_2 同 n 互素, 则 $m_1 m_2 \equiv \pm 1 \pmod{n}$; 这个规则确定了对应关系。当 $A_1 = 1$ 时, 按照(46), 类似的对称性成立。]

27. 首先对于 $n = p^e$, 然后对于 $n = rs$ 证明这个结果, 其中 r 和 s 互素。或者, 使用下题中的公式。

28. a) 左边是乘法性的(见习题 1.2.4-31), 而且当 n 是一个素数的幂时, 容易计算它。c) 从 a), 我们有 Möbius 的逆公式: 如果 $f(n) = \sum_{d|n} g(d)$, 则 $g(n) = \sum_{d|n} \mu(n/d) f(d)$ 。

29. 由欧拉求和公式(见习题 1.2.11.2-7), 我们有 $\sum_{n=1}^N n \ln n = \frac{1}{2} N^2 \ln N + O(N^2)$, 也有 $\sum_{n=1}^N n \sum_{d|n} \Lambda(d)/d = \sum_{d=1}^N \Lambda(d) \sum_{1 \leq k \leq N/d} k$, 而这是 $O(\sum_{d=1}^N \Lambda(d) \cdot N^2/d^2) = O(N^2)$ 。事实上, $\sum_{d \geq 1} \Lambda(d)/d^2 = -\zeta'(2)/\zeta(2)$ 。

30. 当且仅当在未修改的算法中随后的除法步骤中的商为 1 时, 修正后的算法将影响这个计算, 在这种情况下它避免了随后的除法步骤。避免掉一个给定的除法步骤的概率, 相当于 $A_k = 1$ 且这个商前边有偶数个商等于 1 的概率。由对称性条件, 这是 $A_k = 1$ 而且它后边有偶数个商等于 1 的概率。当且仅当 $X_{k-1} > \phi - 1 = 0.618\dots$, 后一情况才出现, 其中 ϕ 是黄金比; 因为 $A_k = 1, A_{k+1} > 1$ 当且仅当 $\frac{2}{3} \leq X_{k-1} < 1; A_k = A_{k+1} = A_{k+2} = 1, A_{k+3} > 1$ 当且仅当 $\frac{5}{8} \leq X_{k-1} < \frac{2}{3}$; 等等。因此我们差不多节省了除法步骤的 $F_{k-1}(1) - F_{k-1}(\phi - 1) \approx 1 - \lg \phi \approx 0.306$ 。当 $v = n$ 且 u 同 n 互素时, 平均的步骤数近似于 $((12 \ln \phi)/\pi^2) \ln n$ 。

K. Vahlen [*Crelle* 115 (1895), 221~233] 考虑了当 $u \bmod v \neq 0$ 时在每次迭代中以 $(v, (\pm u) \bmod v)$ 替代 (u, v) 的所有算法。如果 $u \perp v$, 恰好有 v 个这样的算法, 而且它们可以表示为有 v 个叶的一个二叉树。在所有这样的 gcd 算法中, 对应于最

短可能的迭代次数的最浅的叶,在每步中取最小剩余时出现;而当总是选择最大的剩余时,出现最深的叶。[在 *Hist. Acad. Sci.* **23** (Berlin:1768), 111~180, § 58 中拉格朗日考虑了类似的思想。]关于进一步的结果,请见 N. G. de Bruijn 和 W. M. Zaring, *Nieuw Archief voor Wiskunde* (3) **1** (1953), 105~112; G. J. Rieger, *Math. Nachr.* **82** (1978), 157~180。

在许多计算机上,修改过的算法使得每个除法步骤更长;习题 1 的思想是当商为 1 时,它节省所有的除法步骤。在这样一些情况下,它将是可取的。

31. 设 $a_0 = 0, a_1 = 1, a_{u+1} = 2a_u + a_{u-1}$; 则 $a_n = ((1+\sqrt{2})^n - (1-\sqrt{2})^n)/2\sqrt{2}$, 而且当 $u = a_n + a_{n-1}, v = a_n, n \geq 2$ 时,出现最坏的情况(在定理 F 的意义下)。这一结果由 A. Dupré 给出 [*J. de Math.* **11** (1846), 41~64], 他还研究了由 J. Binet 提出的更为一般的“向前看”的过程。

32. (b) $K_{m-1}(x_1, \dots, x_{m-1})K_{n-1}(x_{m+2}, \dots, x_{m+n})$ 对应于长度为 $m+n$ 的 Morse 代码序列, 其中一个长划占有位置 m 和 $m+1$; 其它的项对应相反的情况。(或者, 使用习题 2。在欧拉的论文中, 也出现有更为一般的恒等式 $K_{m+n}(x_1, \dots, x_{m+n})K_k(x_{m+1}, \dots, x_{m+k}) = K_{m+k}(x_1, \dots, x_{m+k})K_n(x_{m+1}, \dots, x_{m+n}) + (-1)^k K_{m-1}(x_1, \dots, x_{m-1})K_{n-k-1}(x_{m+k+2}, \dots, x_{m+n})$ 。

33. a) 新的表示是对于 $\frac{1}{2}n < m < n, x = m/d, y = (n-m)/d, x' = y' = d = \gcd(m, n-m)$ 。b) 关系 $(n/x') - y \leq x < n/x'$ 确定 x 。c) 计算满足 b) 的 x' 。d) 一对整数 $x > y > 0$ 且 $x \perp y$ 可惟一地写成 $x = K_m(x_1, \dots, x_m), y = K_{m-1}(x_1, \dots, x_{m-1})$ 的形式, 其中 $x_1 \geq 2, m \geq 1$; 这里 $y/x = //x_m, \dots, x_1//$ 。e) 只需证明 $\sum_{1 \leq k \leq n/2} T(k, n) = 2\lfloor n/2 \rfloor + h(n)$; 这由习题 26 得出。

34. a) 以 $\gcd(x, y)$ 除 x 和 y 得到 $g(n) = \sum_{d \mid n} h(n/d)$; 应用习题 28c), 并使用素数和非素数变量之间的对称性。b) 对于固定的 y 和 t , 使得 $xd \geq x'$ 的表示有 $x' < \sqrt{nd}$; 因此有 $O(\sqrt{nd}/y)$ 种这样的表示。现在对 $0 < t \leq y < \sqrt{n/d}$ 求和。c) 如果 $s(y)$ 是给定的和, 则比如说 $\sum_{d \mid y} s(d) = y(H_{2y} - H_y) = k(y)$; 因此 $s(y) = \sum_{d \mid y} k(y/d)$ 。现在 $k(y) = y \ln 2 - \frac{1}{4} + O(1/y)$ 。d) $\sum_{y=1}^n \varphi(y)/y^2 = \sum_{y=1}^n [d \mid y] \mu(d)/yd = \sum_{cd \leq n} \mu(d)/cd^2$ 。(类似地, $\sum_{y=1}^n \sigma_{-1}(y)/y^2 = O(1)$ 。) e) $\sum_{k=1}^n \mu(k)/k^2 = 6/\pi^2 + O(1/n)$ (见习题 4.5.2-10d); 而且 $\sum_{k=1}^n \mu(k) \log k/k^2 = O(1)$ 。因此对于 $d \geq 1$ 我们有 $h_d(n) = n((3 \ln 2)/\pi^2) \ln(n/d) + O(n)$ 。最后 $h(n) = 2 \sum_{cd \leq n} \mu(d) h_c(n/cd) = ((6 \ln 2/\pi^2)n(\ln n - \sum - \sum') + O(n\sigma_{-1}(n)^2))$, 其中剩下的和是 $\sum = \sum_{cd \leq n} \mu(d) \ln(cd)/cd = 0$ 和 $\sum' = \sum_{cd \leq n} \mu(d) \ln c/cd = \sum_{d \leq n} \Lambda(d)/d$ 。[众所周知 $\sigma_{-1}(n) = O(\log \log n)$; 参看 Hardy 和 Wright, *An Introduction to the Theory of Numbers*, § 22.9。]

35. 见 *Proc. Nat. Acad. Sci.* **72** (1975), 4720~4722。M. L. V. Pitteway 和 C. M. A. Castla [*Bull. Inst. Math. and its Applications* **24** (1988), 17~20] 已经发现很强和吸引人的经验证据, 即所有部分商之和实际上是

$$\frac{\pi^2}{24(\ln 2)^2} \left(T_n + \frac{1}{2} - \frac{18(\ln 2)^2}{\pi^2} \right)^2 + \frac{6}{\pi^2} \sum_{\substack{p \leq n \\ p' \nmid n}} \left(\frac{4r}{p'} - \frac{p+1}{p^{2r}} \frac{p^r-1}{p-1} \right) (\ln p)^2 - 2.542875 + O(n^{-1/2})$$

36. 向后执行算法, 并假定对于一个给定的 k 值, $t_k - 1$ 个除法在步骤 C2 中出现, 当 $\gcd(u_{k+1}, \dots, u_n) = F_{t_1} \cdots F_{t_k}$ 和 $u_k \equiv F_{t_1} \cdots F_{t_{k-1}} F_{t_k-1} \pmod{\gcd(u_{k+1}, \dots, u_n)}$ 时, 我们得到极小值 u_n ; 这里诸 $t \geq 2, t_1 \geq 3$, 以及 $t_1 + \dots + t_{n-1} = N + n - 1$ 。在这些条件下极小化 $u_n = F_{t_1} \cdots F_{t_{n-1}}$ 的一个方法是取 $t_1 = 3, t_2 = \dots = t_{n-2} = 2, u_n = 2F_{N-n+2}$ 。如果我们还假设 $u_1 \geq u_2 \geq \dots \geq u_n$, 则解 $u_1 = 2F_{N-n+3} + 1, u_2 = \dots = u_{n-1} = 2F_{N-n+3}, u_n = 2F_{N-n+2}$ 有极小的 u_1 。[参见 *CACM* **13** (1970), 433~436, 447~448。]

37. 见 *Proc Amer. Math. Soc.* **7** (1956), 1014~1021; 也请参见习题 6.1-18。

38. 设 $m = \lceil n/\phi \rceil$, 使得 $m/n = \phi^{-1} + \epsilon = //a_1, a_2, \dots //$, 其中 $0 < \epsilon < 1/n$ 。设 k 是使得 $a_k \geq 2$ 的极小值; 则 $(\phi^{1-k} + (-1)^k F_{k-1}\epsilon) / (\phi^{-k} - (-1)^k F_k\epsilon) \geq 2$ 。因此 k 为偶数而 $\phi^{-2} = 2 - \phi \leq \phi^k F_{k+2}\epsilon = (\phi^{2k+2} - \phi^{-2})\epsilon/\sqrt{5}$ 。[*Ann. Polon. Math.* **1** (1954), 203~206。]

39. 至少 287 是轮上了的; $//2, 1, 95// = 96/287 \approx .33449477$, 没有分母 < 287 的分数位于区间

$$[.3335, .3345] = [//2, 1, 666//, //2, 1, 94, 1, 1, 3//]$$

中。

为了解决求具有最小分母的 $[a, b]$ (其中 $0 < a < b < 1$) 中分数的一般问题, 注意借助于正则连分数表示我们有 $//x_1, x_2, \dots// < //y_1, y_2, \dots//$, 当且仅当对于满足 $x_j \neq y_j$ 的最小的 j , $(-1)^j x_j < (-1)^j y_j$, 其中我们把“ ∞ ”放在一个有理数的最后的部分商之后。因此如果 $a = //x_1, x_2, \dots//$ 且 $b = //y_1, y_2, \dots//$, 且 j 是使 $x_j \neq y_j$ 之极小者, 则 $[a, b]$ 中的分数有 $c = //x_1, \dots, x_{j-1}, z_j, \dots, z_m//$ 的形式, 其中 $//z_j, \dots, z_m//$ 位于 $//x_j, x_{j+1}, \dots//$ 和 $//y_j, y_{j+1}, \dots//$ 之间。设 $K_{-1} = 0, c$ 的分母

$$K_{j-1}(x_1, \dots, x_{j-1})K_{m-j+1}(z_j, \dots, z_m) + K_{j-2}(x_1, \dots, x_{j-2})K_{m-j}(z_{j+1}, \dots, z_m)$$

当 $m = j$ 和 $z_j = (j \text{ 奇} \Rightarrow y_j + [y_{j+1} \neq \infty]; x_j + [x_{j+1} \neq \infty])$ 时取极小值。[推导这一方法的另一个途径由下边习题的理论得出。]

40. 用归纳法可以证明, 在每一节点处 $p_i q_l - p_l q_i = 1$, 因此 p_l 和 q_l 互素。由于 $p/q < p'/q'$ 意味着 $p/q < (p+p')/(q+q') < p'/q'$, 显然 p/q 的所有左后裔上的标号小于 p/q , 而所有右后裔上的标号大于 p/q 。因此每个有理数作为一个标号至多

出现一次。

剩下的是证明每个有理数确实出现。如果 $p/q = //a_1, \dots, a_r, 1//$, 其中每个 a_i 是正整数, 用归纳法可以证明标号为 p/q 的节点可通过往左走 a_1 次, 然后往右 a_2 次, 然后左走 a_3 次等等而被找到。

[这个树的逐级标号序列首先是由 M. A. Stern 研究的, *Crelle* 55 (1858) 193 ~ 220, 尽管在他的论文中未明显地给出与二叉树的关系。通过逐次地在相邻元素 p/q 和 p'/q' 之间插入 $(p+p')/(q+q')$ 而得到所有可能的分数的思想可以回溯得久远得多: 实质性的思想是由 Daniel Schwenter [*Deliciae Physico-Mathematicae* (Nürnberg: 1636), Part 1, Problem 87; *Geometria Practica*, 第 3 版 (1641), 68; 参见 M. Cantor, *Geschichte der Math.* 2 (1900), 763 ~ 765], 以及由 John Wallis 在他的 *Treatise of Algebra* (1685), 第 10 ~ 11 章给出的。C. Huygens 在设计他的天象仪的齿轮时很好地利用了这样一些思想 [参见他死后才发表的 *Descriptio Automati Planetarii* (1703)]。拉格朗日在 *Hist. Aca. Sci* 23 (Berlin: 1767), 311 ~ 352, § 24 以及在他对欧拉的代数学的法文版的补充 (1774), § 18 ~ § 20 中给出了充分的描述。也请参见习题 1.3.2-19; A. Brocot, *Revue Chronométrique* 3 (1861), 186 ~ 194; D. H. Lehmer, *AMM* 36 (1929), 59 ~ 67。]

41. 事实上, 对于一般形式的数

$$\frac{1}{l_1} + \frac{(-1)^{e_1}}{l_1^2 l_2} + \frac{(-1)^{e_2}}{l_1^4 l_2^2 l_3} + \dots$$

的正则连分数有一个有趣的模式, 它以连续恒等式

$$\begin{aligned} K_{m+n+1}(x_1, \dots, x_{n-1}, x_n - 1, 1, y_n - 1, y_{n-1}, \dots, y_1) = \\ x_m K_{m-1}(x_1, \dots, x_{m-1}) K_n(y_n, \dots, y_1) + \\ (-1)^n K_{m+n}(x_1, \dots, x_{m-1}, 0, -y_n, -y_{n-1}, \dots, -y_1) \end{aligned}$$

为基础。当 $y_n = x_{m-1}, y_{n-1} = x_{m-2}$ 等时, 这个恒等式是最有趣的, 因为

$$K_{n-1}(z_1, \dots, z_k, 0, z_{k+1}, \dots, z_n) = K_{n-1}(z_1, \dots, z_{k-1}, z_k + z_{k+1}, z_{k+2}, \dots, z_n)$$

特别是, 我们发现如果 $p_n/q_n = K_{n-1}(x_2, \dots, x_n)/K_n(x_1, \dots, x_n) = //x_1, \dots, x_n//$, 则 $p_n/q_n + (-1)^n/q_n^2 r = //x_1, \dots, x_n, r-1, 1, x_n-1, x_{n-1}, \dots, x_1//$ 。通过把 $//x_1, \dots, x_n//$ 改变成 $//x_1, \dots, x_{n-1}, x_n-1, 1//$, 我们可以随意控制符号 $(-1)^n$ 。

例如, 头一个级数的部分和有下列偶长度的连分数: $//1, 1//; //1, 1, 1, 1, 0, 1// = //1, 1, 1, 2//; //1, 1, 1, 2, 1, 1, 1, 1, 1, 1//; //1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1// = //1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1//$; 从这里开始, 序列平稳下来并遵守一个简单的反射模式。我们发现, 如果 $n-1 = 20q + r$, 其中 $0 \leq r < 20$, 则第 n 个部分商可迅速计算如下:

$$a_n = \begin{cases} 1, & \text{如果 } r = 0, 2, 4, 5, 6, 7, 9, 10, 12, \\ & 13, 14, 15, 17, \text{或 } 19 \\ 2, & \text{如果 } r = 3 \text{ 或 } 16 \\ 1 + (q + r) \bmod 2, & \text{如果 } r = 8 \text{ 或 } 11 \\ 2 - d_q, & \text{如果 } r = 1 \\ 1 + d_{q+1}, & \text{如果 } r = 18 \end{cases}$$

这里 d_n 是由规则 $d_0 = 1, d_{2n} = d_n, d_{4n+1} = 0, d_{4n+3} = 1$ 定义的“龙序列”;习题 4.1-18 中讨论的龙曲线在它的第 n 步向右转当且仅当 $d_n = 1$ 。

$l \geq 3$ 的 Liouville 数等于 $//l-1, l+1, l^2-1, 1, l, l-1, l^{l^2}-1, 1, l-2, l, 1, l^2-1, l+1, l-1, l^{l^2}-1, \dots//$ 。第 n 个部分商 a_n 依赖于 $n \bmod 4$ 的龙序列如下:如果 $n \bmod 4 = 1$, 则它是 $l-2 + d_{n-1} + (\lfloor n/2 \rfloor \bmod 4)$, 而如果 $n \bmod 4 = 2$, 则它是 $l+2 - d_{n+2} - (\lfloor n/2 \rfloor \bmod 4)$;如果 $n \bmod 4 = 0$, 依赖于 $d_n = 0$ 还是 1, 它是 1 或 $l^{k!(k-1)} - 1$, 其中 k 整除 n 的 2 的最大幂;当 $l=2$ 时, 可应用同一规则, 但必须删去所有的 0, 所以同 $n \bmod 24$ 有关的模式更为复杂。

[参考文献: J. O. Shallit, *J. Number Theory* **11** (1979), 209 ~ 217; Allouche, Lubiw, Mendès France, van der Poorten 和 Shallit, *Acta Arithmetica* **77** (1996), 77 ~ 96.]

42. 假设 $\|qX\| = |qX - p|$ 。我们总可以求整数 u 和 v 使得 $q = uq_{n-1} + vq_n$, $p = up_{n-1} + vp_n$, 其中 $p_n = K_{n-1}(A_2, \dots, A_n)$, 因为 $q_n p_{n-1} - q_{n-1} p_n = \pm 1$ 。如果 $v=0$, 则这个结果是显然的。否则我们必定有 $uv < 0$, 因此 $u(q_{n-1}X - p_{n-1})$ 和 $v(q_nX - p_n)$ 有相同的符号, 且 $|qX - p| = |u| |q_{n-1}X - p_{n-1}| + |v| |q_nX - p_n|$ 。这就完成了证明, 因为 $u \neq 0$ 。关于其推广见定理 6.4S。

43. 如果 x 是可表示的, 则习题 40 的 Stern-Brocot 树中 x 的父亲也是可表示的; 于是可表示的数形成该二叉树的一株子树。设 (u/u') 和 (v/v') 是相邻的可表示的数。则这两个当中有一个是另一个的祖先; 比如说可令 (u/u') 是 (v/v') 的祖先, 因为另一种情况与此类似。于是, (u/u') 是 (v/v') 最接近的左祖先, 所以 u/u' 和 v/v' 之间的所有数都是 v/v' 的左后裔, 而中间值 $((u+v)/(u'+v'))$ 是它的左儿子。按照正则连分数与二叉树的关系, 中间值和所有它的左后裔都将有 (u/u') 作为它们最后可表示的 p_i/q_i , 而所有中间值的左后裔都将有 (v/v') 作为 p_i/q_i 之一。(在通到 x 的通路上的所有“拐弯节点”的父亲标号是 p_i/q_i 。)

44. $M = N = 100$ 时的一个反例是 $(u/u') = \frac{1}{3}$, $(v/v') = \frac{67}{99}$ 。然而, 由于 (12) 的缘故, 这个恒等式几乎总是对的; 只有当 $u/u' + v/v'$ 非常接近等于比 (u/u') 更简单的分数时, 它才不对。

45. 使用通常的长除法, 为确定使得 $u = Av + r$, $0 \leq r < v$ 的 A 和 r , 要花费 $O((1 + \log A)(\log u))$ 个时间单位。如果在执行此算法期间的商是 A_1, A_2, \dots, A_m , 则 $A_1 A_2 \cdots A_m \leq u$, 所以 $\log A_1 + \cdots + \log A_m \leq \log u$ 。由定理 L 也有 $m =$

$O(\log u)$ 。

46. 是的,即使我们也需要计算将通过欧几里得算法计算的部分商的序列,也是 $O(n(\log n)^2(\log \log n))$;参见 A. Schönage, *Acta Informatica* 1 (1971), 139~144。而且相对于它所执行的乘法和除法, Schönage 的算法是计算连分数展开渐近最优的 [V. Strassen *SICOMP* 12 (1983), 1~27]。除非 n 十分地大,否则算法 4.5.2L 在实际使用中是更好的。但在 A. Schönage, A. F. W. Grotfeld 和 E. Vetter 的书 *Fast Algorithms* [(Heidelberg: Spektrum Akademischer Verlag, 1994), § 7.2] 中刻画了超过大约 1800 个二进位的数的一个有效实现。

48. $T_j = (K_{j-2}(-a_2, \dots, -a_{j-1}), K_{j-1}(-a_1, \dots, -a_{j-1}), K_{n-j}(a_{j+1}, \dots, a_n) \cdot d)$
 $= ((-1)^j K_{j-2}(a_2, \dots, a_{j-1}), (-1)^{j-1} K_{j-1}(a_1, \dots, a_{j-1}), K_{n-j}(a_{j+1}, \dots, a_n) d)$ 。

49. 由于 $\lambda x_1 + \mu z_1 = \mu v$ 和 $\lambda x_{n+1} + \mu z_{n+1} = -\lambda v/d$, 有一个 j 的奇数值使得 $\lambda x_j + \mu z_j > 0$ 和 $\lambda x_{j+2} + \mu z_{j+2} \leq 0$ 。如果 $\lambda x_j + \mu z_j > \theta$ 和 $\lambda x_{j+2} + \mu z_{j+2} < -\theta$, 我们有 $\mu > \theta/z_j$ 和 $\lambda > -\theta/x_{j+2}$ 。由此得出 $0 < \lambda x_{j+1} + \mu z_{j+1} < \lambda \mu x_{j+1} z_j / \theta - \lambda \mu z_{j+1} x_{j+2} / \theta \leq 2\lambda \mu v / \theta = 2\theta$, 因为对于所有 k 我们有 $|x_{k+1} z_k| = K_{k-1}(a_2, \dots, a_k) K_{n-k}(a_{k+1}, \dots, a_n) \leq K_{n-1}(a_2, \dots, a_n) = v/d$ 。[H. W. Lenstra, Jr., *Math. Comp.* 42 (1984), 331~340。]

50. 令 $k = \lceil \beta/\alpha \rceil$, 如果 $k\alpha < \gamma$, 则答案为 k ; 否则它是

$$k-1 + \left\lceil \frac{f((1/\alpha) \bmod 1, k - \gamma/\alpha, k - \beta/\alpha)}{\alpha} \right\rceil$$

51. 如果 $ax - mz = y$ 和 $x \perp y$, 我们有 $x \perp mz$ 。考虑习题 40 的 Stern-Brocot 树, 连同以 0/1 作为标号的一个附加节点。对于每个节点标号 z/x 附加标记值 $y = ax - mz$ 。我们要求其标记值的绝对值至多是 $\theta = \sqrt{m}/2$ 而其分母 x 也小于等于 θ 的所有节点 z/x 。到这样的节点的惟一可能的通路保持在左边是正的标记, 而在右边是负的标记。这样一个规则定义惟一的通路, 当标记为正时它向右动而当标记为负时它向左动, 当标记变成零时它停止下来。当以 $u = m$ 和 $v = a$ 执行算法 4.5.2X 时也隐含得出同样的通路, 除非这个算法向前跳——它仅仅访问标记改变符号之前的通路的节点(如同在习题 43 中的“拐弯节点”的父亲)。

令 z/x 是其标记 y 满足 $|y| \leq \theta$ 的通路的头一个节点。如果 $x > \theta$, 则无解, 因为在通路上随后的值甚至有更大的分母。否则, 假定 $x \perp y$, 则 $(\pm x, \mp y)$ 是一个解。

容易看出, 如果 $y = 0$ 则无解; 而如果 $y \neq 0$ 则在通路上的下个节点的标记将不会和 y 有相同的符号。因此由算法 4.5.2X, 将访问节点 z/x , 而且对于某个 j 我们将有 $x = x_j = K_{j-1}(a_1, \dots, a_{j-1})$, $y = y_j = (-1)^{j-1} K_{n-j}(a_{j+1}, \dots, a_n) d$, $z = z_j = K_{j-2}(a_2, \dots, a_{j-1})$ (见习题 48)。对于一个解的下一个可能性将是带有标记 $y' = y_{j-1} + ky_j$ 的标号 $z'/x' = (z_{j-1} + kz_j)/(x_{j-1} + kx_j)$ 的节点, 其中 k 尽可能地小使得 $|y'| \leq$

θ ; 我们有 $y'y < 0$ 。然而, x' 现在必须超过 θ ; 否则我们将有 $m = K_n(a_1, \dots, a_n)d = x'|y| + x|y'| \leq \theta^2 + \theta^2 = m$, 因此等式不能成立。

这个讨论证明, 通过对于 $u = m$ 和 $v = a$ 来应用算法 4.5.2X, 这个问题可有效地求解, 但通过替换步骤 X2 如下: “如果 $v_3 \leq \sqrt{m/2}$, 则算法结束。假定 $x \perp y$ 和 $x \leq \sqrt{m/2}$, 则数偶 $(x, y) = (|v_2|, v_3 \text{sign}(v_2))$ 是惟一的解; 否则无解。”[P. S. Wang, *Lecture Notes in Comp. Sci.* **162** (1983), 225 ~ 235; P. Kornerup 和 R. T. Gregory, *BIT* **23** (1983), 9 ~ 20.]

每当 $2\theta_1\theta_2 \leq m$ 时, 如果我们要求 $0 < x \leq \theta_1$ 和 $|y| \leq \theta_2$, 则一个类似的方法将有效。

4.5.4 小节

1. 如果 d_k 不是素数, 则在试验 d_k 之前它的素因子已经被分解了。

2. 否; 如果 $p_{t+1} = p_t$, 则这个算法将失败, 给出“1”作为假的素因子。

3. 设 P 是头 168 个素数的积。[注: 尽管 $P = 19590 \dots 5910$ 是一个 416 位数字, 求这样一个 gcd 比做 168 个除法要少花很多时间, 如果我们仅仅想测试 n 是否为素数的话。]

4. 在习题 3.1-11 的符号下

$$\sum_{\mu, \lambda} 2^{\lceil \lg \max(\mu+1, \lambda) \rceil} P(\mu, \lambda) = \frac{1}{m} \sum_{l \geq 1} f(l) \prod_{k=1}^{l-1} \left(1 - \frac{k}{m}\right)$$

其中 $f(l) = \sum_{1 \leq \lambda \leq l} 2^{\lceil \lg \max(l+1-\lambda, \lambda) \rceil}$ 。如果 $l = 2^{k+\theta}$, 其中 $0 < \theta \leq 1$, 我们有 $f(l) = l^2 (3 \cdot 2^{-\theta} - 2 \cdot 2^{-2\theta})$, 其中函数 $3 \cdot 2^{-\theta} - 2 \cdot 2^{-2\theta}$ 在 $\theta = \lg(4/3)$ 处达到极大值 $\frac{9}{8}$, 并且在 $\theta = 0$ 和 1 处有极小值 1。因此 $2^{\lceil \lg \max(\mu+1, \lambda) \rceil}$ 的平均值位于 1.0 和 1.125 乘以 $\mu + \lambda$ 的平均值之间, 由此得出结果。

注: Richard Brent 已经发现, 当 $m \rightarrow \infty$ 时, 密度 $\prod_{k=1}^l (1 - k/m) = \exp(-l(l-1)/2m + O(l^3/m^2))$ 趋近一个正态分布, 因此我们可以假定 θ 是一致分布的。于是 $3 \cdot 2^{-\theta} - 2 \cdot 2^{-2\theta}$ 取平均值 $3/(4 \ln 2)$, 而且算法 B 所需的平均迭代次数近似地为 $\left(3/(4 \ln 2) + \frac{1}{2}\right) \sqrt{\pi m/2} = 1.98277 \sqrt{m}$ 。当 $p \approx 2.4771366$ 被“最优地”选定为 $(p^2 - 1) \ln p = p^2 - p + 1$ 的根时, 对于习题 3.1-7 的答案中更一般方法的类似分析给出 $\sim 1.92600 \sqrt{m}$ 。参见 *BIT* **20** (1980), 176 ~ 184。

算法 B 是 Pollard 原来算法的一个改进, 它是以习题 3.1-6b) 而不是以习题 3.1-7 中还未发现的结果为基础的。他证明, 使得 $X_{2^n} = X_n$ 的最小的 n 有 $\sim (\pi^2/12) Q(m) \approx 1.0308 \sqrt{m}$ 的平均值; 由等式 4.5.3-(21) 可得到常数 $\pi^2/12$ 的解释。因此

在他原来的算法中所需要的平均工作量大约是 $1.03081 \sqrt{m}$ 个 gcd (或者乘法模 m) 以及 $3.09243 \sqrt{m}$ 个平方。当 gcd 的花销大于大约 1.17 乘以平方的花销时——通常对于很大的数是如此,则实际上这 will 比算法 B 更好。

然而, Brent 注意到, 通过当 $k > l/2$ 时, 不去检查 gcd, 可以改进算法 B; 如果重复步骤 B4 直到 $k \leq l/2$, 在进行了 $\lambda \lfloor \ell(\mu)/\lambda \rfloor = \ell(\mu) - (\ell(\mu) \bmod \lambda)$ 个进一步的迭代之后, 我们仍将探测这个循环。当我们取平方而不取 gcd 时, 平均花销现在近似地变成 $(3/(4 \ln 2)) \sqrt{\pi m/2} \approx 1.3561 \sqrt{m}$ 个迭代, 当我们两个都做时, 则加上 $\left((\ln \pi - \gamma)/(4 \ln 2) + \frac{1}{2} \right) \sqrt{\pi m/2} \approx .88319 \sqrt{m}$ 个迭代。[请见 Henri Cohen 在 *A Course in Computational Algebraic Number Theory* (Berlin: Springer, 1993), § 8.5 中的分析。]

5. 值得注意的是, $11111 \equiv 8616460799 \pmod{3 \cdot 7 \cdot 8 \cdot 11}$, 因此对于 $N = 11111$, (14) 也是正确的, 但相对于模 5 除外。由于剩余 $(x^2 - N) \bmod 5$ 是 4, 0, 3, 3, 0, 我们必有 $x \bmod 5 = 0, 1$ 或 4。满足所有条件的头一个 $x \geq \lceil \sqrt{N} \rceil = 106$ 是 $x = 144$; 但 $144^2 - 11111 = 9625$ 的平方根不是一个整数。然而, 下一个情况给出 $156^2 - 11111 = 13225 = 115^2$, 而且 $11111 = (156 - 115) \cdot (156 + 115) = 41 \cdot 271$ 。

6. 让我们计算同余 $N \equiv (x - y)(x + y) \pmod{p}$ 的解 (x, y) 的个数, 其中 $0 \leq x, y < p$ 。由于 $N \not\equiv 0$ 和 p 是素数, $x + y \not\equiv 0$ 。对于每个 $v \not\equiv 0$, 有惟一的 $u \pmod{p}$ 使得 $N \equiv uv$ 。现在同余式 $x - y \equiv u, x + y \equiv v$ 惟一地确定了 $x \bmod p$ 和 $y \bmod p$, 因为 p 是奇数。于是所述同余恰有 $p - 1$ 个解 (x, y) 。如果 (x, y) 是一个解, 则如果 $y \neq 0$, $(x, p - y)$ 也是解, 因为 $(p - y)^2 \equiv y^2$; 而且如果 (x, y_1) 和 (x, y_2) 是解且 $y_1 \neq y_2$, 则我们有 $y_1^2 \equiv y_2^2$; 因此 $y_1 = p - y_2$ 。于是如果 $N \equiv x^2$ 没有解, 则在诸解 (x, y) 当中不同 x 值的个数是 $(p - 1)/2$, 或者如果 $N \equiv x^2$ 有解, 则不同 x 值的个数为 $(p + 1)/2$ 。

7. 一个过程是对于每个模数记住两个下标, 一个作为当前字的位置, 另一个作为当前位的位置; 从表中取两个字, 然后执行一个变址移位指令, 可对表的项目进行适当的调整。(许多计算机都有特殊的设备执行这类位操作。)

8. (我们可以假定 $N = 2M$ 是偶数。) 下列算法使用一张辅助表 $X[1], X[2], \dots, X[M - 1]$, 其中 $X[k]$ 表示 $2k + 1$ 的素性。

S1. 对于 $1 \leq k < M$, 置 $X[k] \leftarrow 1$ 。并且置 $j \leftarrow 1, k \leftarrow 1, p \leftarrow 3, q \leftarrow 4$ 。(在此算法执行期间 $p = 2j + 1, q = 2j + 2j^2$ 。)

S2. 如果 $X[j] = 0$, 则转到 S4。否则输出 p (它是素数), 且置 $k \leftarrow q$ 。

S3. 如果 $k \leq M$, 则置 $X[k] \leftarrow 0, k \leftarrow k + p$, 并重复本步骤。

S4. 置 $j \leftarrow j + 1, p \leftarrow p + 2, q \leftarrow q + 2p - 2$ 。如果 $j < M$, 则返回步骤 S2。 ▮

如果在步骤 S4 中相对于 M 测试的是 q (而不是 j), 且如果附加一个新循环, 它对等于 1 的所有剩下的 $X[j]$ 输出 $2j + 1$, 则这个计算的主要部分将显著地加快, 并省去对 p 和 q 的操作。

注: Eratosthenes 有独创性的筛在 Nicomachus 的 *Introduction to Arithmetic* 的第 13 章, 卷 1 中被描述。众所周知, $\sum_{p \leq N} [p \leq N]/p = \ln \ln N + M + O((\log N)^{-10000})$, 其中 $M = \gamma + \sum_{k \geq 2} \mu(k) \ln \zeta(k)/k$ 是 Mertens 常数 0.26149 72128 47642 78375 54268 38608 69585 90516-; 参见 F. Mertens, *Grelle* 76 (1874), 46~62; Greene 和 Knuth, *Mathematics for the Analysis of Algorithms* (Boston, Birkhäuser, 1981), § 4.2.3。特别是, 在 Nicomachus 描述的原来的算法中运算的个数是 $N \ln \ln N + O(N)$ 。在习题 5.2.3-15 和 7.1 节中讨论了对于生成素数的筛方法的效率方面的改进。

9. 对于某个素数 p , 如果 p^2 是 n 的一个因子, 则 p 是 $\lambda(n)$ 的因子, 但不是 $n-1$ 的因子。如果 $n = p_1 p_2$, 其中 $p_1 < p_2$ 是两个素数, 则 $p_2 - 1$ 是 $\lambda(n)$ 的一个因子, 因此 $p_1 p_2 - 1 \equiv 0 \pmod{p_2 - 1}$ 。由于 $p_2 \equiv 1$, 这意味着 $p_1 - 1$ 是 $p_2 - 1$ 的倍数, 同 $p_1 < p_2$ 的假定矛盾。[使得 $\lambda(n)$ 真正整除 $n-1$ 的那样的 n 值, 称为 Carmichael 数。例如, 下边是具有至多六个素因子的一些小的 Carmichael 数: $3 \cdot 11 \cdot 17, 5 \cdot 13 \cdot 17, 7 \cdot 11 \cdot 13 \cdot 41, 5 \cdot 7 \cdot 17 \cdot 19 \cdot 73, 5 \cdot 7 \cdot 17 \cdot 73 \cdot 89 \cdot 107$ 。小于 10^{12} 的 Carmichael 数共有 8241 个, 而且小于 N 的 Carmichael 数至少有 $\Omega(N^{2/7})$ 个 [参见 W. R. Alford, A. Granville 及 C. Pomerance, *Annals of Math.* (2) 139 (1994), 703~722]。

10. 设 k_p 是 x_p 的模 n 阶, 并设 λ 是所有 k_p 的最小公倍数。则 λ 是 $n-1$ 的因子, 但不是任何 $(n-1)/p$ 的因子, 所以 $\lambda = n-1$ 。由于对所有的 $p, x_p^{\varphi(n)} \bmod n = 1$, $\varphi(n)$ 是 k_p 的倍数, 所以 $\varphi(n) \geq \lambda$ 。但当 n 不是素数时, $\varphi(n) < n-1$ 。(证明这一点的另一途径, 是用习题 3.2.1.2-15 的方法, 从诸 x_p 构造一个阶为 $n-1$ 的元素 x_0)

11.	U	V	A	P	S	T	输出
	1984	1	0	992	0	—	
	1981	1981	1	992	1	1981	
	1983	4	495	993	0	1	$993^2 \equiv +2^2$
	1983	991	2	98109	1	991	
	1981	4	495	2	0	1	$2^2 \equiv +2^2$
	1984	1981	1	99099	1	1981	
	1984	1	1984	99101	0	1	$99101^2 \equiv +2^0$

从头一个或最后的输出来看, 因子分解 $199 \cdot 991$ 是明显的。循环的简短性, 以及熟知的数 1984 的出现, 大概都仅仅是巧合。

12. 下列算法利用由整数 $E_{jk}, 0 \leq j, k \leq m$ 构成的一个 $(m+1) \times (m+1)$ 阶辅助矩阵, 一个单精度向量 $(b_0, b_1, b_2, \dots, b_m)$ 和其元素处于 $0 \leq x_k \leq N$ 范围内的一个多精度向量 (x_0, x_1, \dots, x_m) 。

F1. [初始化] 对 $0 \leq i \leq m$ 置 $b_i \leftarrow -1$; 然后置 $j \leftarrow 0$ 。

F2.[下一个解] 得到由算法 E 产生的下一个输出 $(x, e_0, e_1, \dots, e_m)$ 。(把算法 E 和 F 看做共行程序是方便的。)置 $k \leftarrow m$ 。

F3.[寻找奇数] 如果 $k < 0$, 则转到步骤 F5。否则如果 e_k 是偶数, 则置 $k \leftarrow k - 1$, 并重复这一步骤。

F4.[线性相关?] 如果 $b_k \geq 0$, 则对 $0 \leq r \leq m$, 置 $i \leftarrow b_k, x \leftarrow (x_i x) \bmod N, e_r \leftarrow e_r + E_{ir}$; 置 $k \leftarrow k - 1$ 并返回 F3。否则对于 $0 \leq r \leq m$, 置 $b_k \leftarrow j, x_j \leftarrow x, E_{jr} \leftarrow e_r$; 置 $j \leftarrow j + 1$ 并返回 F2。(在后一情况下, 我们有一个新的在模 2 下的线性无关的解, 它的头一个奇分量是 e_k 。不保证 E_{jr} 保留单精度, 但当 k 从 m 减少为 0 时, 如同 Morrison 和 Brillhart 所推荐的那样, 它们趋向于保持小的值。)

F5.[试分解因子] (现在 e_0, e_1, \dots, e_m 是偶数。)置

$$y \leftarrow ((-1)^{e_0/2} p_1^{e_1/2} \cdots p_m^{e_m/2}) \bmod N$$

如果 $x = y$ 或 $x + y = N$, 则返回 F2。否则计算 $\gcd(x - y, N)$, 它是 N 的真因子并终止本算法。 ─

只要能从算法 E 给定的输出导出一个因子, 这个算法就能找到一个因子。[证明。设对于 $1 \leq i \leq t$, 算法 E 的输出是 $(X_i, E_{i0}, \dots, E_{im})$, 并假设当 $x \equiv X_1^{a_1} \cdots X_t^{a_t}$ 和 $y \equiv (-1)^{e_0/2} p_1^{e_1/2} \cdots p_m^{e_m/2} \pmod{N}$ 时, 我们可以求出一个因子分解 $N = N_1 N_2$, 其中对所有的 j , $e_j = a_1 E_{1j} + \cdots + a_t E_{tj}$ 都是偶数。则 $x \equiv \pm y \pmod{N_1}$ 和 $x \equiv \mp y \pmod{N_2}$ 。不难看出, 这个解可以变换成为在 F5 步中出现的一对 (x, y) 。办法是通过系统地用 (xx', yy') 代替 (x, y) 的一系列步骤, 其中 $x' \equiv \pm y' \pmod{N}$ 。]

13. 有 2^d 个 x 的值有相同的指数 (e_0, \dots, e_m) , 因为当 $N = q_1^{f_1} \cdots q_d^{f_d}$ 时我们可任意选择 $x \pmod{q_i^{f_i}}$ 的符号, 这 2^d 个值当中恰有两个不能分解成因式。

14. 因为对于 V 的任何素因子 p , $P^2 \equiv kNQ^2 \pmod{p}$, 如果 $P \not\equiv 0$, 我们得到 $1 \equiv P^{2(p-1)/2} \equiv (kNQ^2)^{(p-1)/2} \equiv (kN)^{(p-1)/2} \pmod{p}$ 。

15. $U_n = (a^n - b^n)/\sqrt{D}$, 其中 $a = \frac{1}{2}(P + \sqrt{D})$, $b = \frac{1}{2}(P - \sqrt{D})$, $D = P^2 - 4Q$ 。于是 $2^{n-1} U_n = \sum_k \binom{n}{2k+1} P^{n-2k-1} D^k$; 所以如果 p 是一个奇素数, 则 $U_p \equiv D^{(p-1)/2} \pmod{p}$ 。类似地, 如果 $V_n = a^n + b^n = U_{n+1} - QU_{n-1}$, 则 $2^{n-1} V_n = \sum_k \binom{n}{2k} P^{n-2k} D^k$, 而且 $V_p \equiv P^p \equiv P$ 。因此, 若 $U_p \equiv -1$, 则我们求得 $U_{p+1} \bmod p = 0$ 。如果 $U_p \equiv 1$, 则求得 $(QU_{p-1}) \bmod p = 0$; 这里如果 Q 是 p 的倍数, 则对于 $n > 0$, $U_n \equiv P^{n-1} \pmod{p}$, 所以 U_n 绝不是 p 的倍数; 若 Q 不是 p 的倍数, 则 $U_{p-1} \bmod p = 0$ 。因此像在定理 L 中那样, 如果 $N = p_1^{e_1} \cdots p_r^{e_r}$, 则 $U_i \bmod N = 0, N \perp Q$,

且 $t = \text{lcm}_{1 \leq j \leq r} (p_j^{e_j-1} (p_j + c_j))$ 。在本题的假定下, N 出现的阶是 $N+1$; 因此 N 与 Q 互素, 而 t 是 $N+1$ 的倍数。又本题的假定意味着每个 p_j 是奇数且每个 c_j 是 ± 1 , 所以 $t \leq 2^{r-1} \prod_{j=1}^r p_j^{e_j-1} \left(p_j + \frac{1}{3} p_j \right) = 2 \left(\frac{2}{3} \right)^r N$; 因此 $r-1$ 和 $t = p_1^{e_1} + c_1 p_1^{e_1-1}$ 。最后, $e_1 = 1$ 和 $c_1 = 1$ 。

注: 如果要求关于不可约性的这个测试真能有点用处, 则我们必须以使得这个测试可能奏效的方式选择 P 和 Q 。Lehmer 提议取 $P = 1$ 使得 $D = 1 - 4Q$, 并选择 Q 使 $N \perp QD$ 。(如果后一条件失败, 则我们已经知道 N 不是素数, 除非 $|QD| \geq N$ 。)而且, 上面的推导表明我们将要求 $e_1 = 1$, 即 $D^{(N-1)/2} \equiv -1 \pmod{N}$ 。这是确定对 Q 的选择的另一个条件。此外, 如果 D 满足这个条件且 $U_{N+1} \pmod{N} \neq 0$, 则我们知道 N 不是素数。

例子: 如果 $P = 1, Q = -1$, 则我们有斐波那契序列, 且 $D = 5$ 。由于 $5^{11} \equiv -1 \pmod{23}$, 我们可能想通过使用斐波那契序列:

$$\langle F_n \pmod{23} \rangle = 0, 1, 1, 2, 3, 5, 8, 13, 21, 11, 9, 20, 6, 3, 9, 12, 21, 10, 8, 18, 3, 21, 1, 22, 0, \dots$$

来证明 23 是素数, 所以 24 是 23 出现的阶, 因而此测试有效。然而, 不能以这种方法用斐波那契序列来证明 13 和 17 的不可约性, 因为 $F_7 \pmod{13} = 0$ 和 $F_9 \pmod{17} = 0$ 。当 $p \equiv \pm 1 \pmod{10}$ 时, 我们有 $5^{(p-1)/2} \pmod{p} = 1$, 所以 F_{p-1} (不是 F_{p+1}) 可为 p 整除。

17. 设 $f(q) = 2 \lg q - 1$ 。当 $q = 2$ 或 3 时, 这树至多有 $f(q)$ 个节点。当 $q > 3$ 为素数时, 设 $q = 1 + q_1 \cdots q_t$, 其中 $t \geq 2$ 且 q_1, \dots, q_t 是素数。树的大小小于等于 $1 + \sum f(q_k) = 2 + f(q-1) - t < f(q)$ 。[SICOMP 7 (1975), 214~220。]

18. $x(G(\alpha) - F(\alpha))$ 是 $n \leq x$ 中其第二个最大的素因子 $\leq x^\alpha$ 和最大素因子 $> x^\alpha$ 的个数。因此

$$xG'(t)dt = (\pi(x^{t/(1-t)}) - \pi(x^t)) \cdot x^{1-t}(G(t/(1-t)) - F(t/(1-t)))$$

$p_{t-1} \leq \sqrt{p_t}$ 的概率是 $\int_0^1 F(t/2(1-t)) t^{-1} dt$ 。[奇怪的是, 可以证明这也等于

$\int_0^1 F(t/(1-t)) dt$, 即 $\log p_t / \log x$ 的平均值, 而且它也等于习题 1.3.3-23 和 3.1-

13 的 Dickman-Golomb 常数 .62433。导数 $G'(0)$ 可以证明是等于

$$\int_0^1 F(t/(1-t)) t^{-2} dt = F(1) + 2F\left(\frac{1}{2}\right) + 3F\left(\frac{1}{3}\right) + \dots = e^\gamma$$

第三个最大的素因子有 $H(\alpha) = \int_0^\alpha (H(t/(1-t)) - G(t/(1-t))) t^{-1} dt$ 和 $H'(0) =$

∞ 。见 P. Billingsley, *Period. Math. Hungar.* 2 (1972), 283~289; J. Galambos, *Acta Arith.* 31 (1976), 213~218; D. E. Knuth 和 L. Trapp Pardo, *Theoretical Comp. Sci.* 3 (1976), 321~

348; J. L. Hafner 和 K. S. McCurley, *J. Algorithms* **10** (1989) 531 ~ 556.]

19. $M = 2^D - 1$ 是所有这样的 p 的倍数, 对于这些 p , 2 的模 p 阶整除 D 。为了推广这一思想, 设 $a_1 = 2$ 和 $a_{j+1} = a_j^{q_j} \bmod N$, 其中 $q_j = p_j^{e_j}$, p_j 是第 j 个素数, $e_j = \lfloor \log 1000 / \log p_j \rfloor$; 设 $A = a_{169}$ 。现在对于 10^3 和 10^5 之间的所有素数 q 计算 $b_q = \gcd(A^q - 1, N)$ 。做这件事的一个办法是由 $A^{1009} \bmod N$ 开始, 而后交替地乘以 $A^4 \bmod N$ 和 $A^2 \bmod N$ (类似的方法曾为 D. N. Lehmer 于 20 世纪 20 年代使用过, 但他没有发表它)。如同对于算法 B 一样, 我们可以通过成批计算来避免大多数 gcd; 例如, 由于 $b_{30r+k} = \gcd(A^{30r+k} - A^k, N)$, 我们可以尝试 8 个一批地首先计算 $c_r = (A^{30r} - A^{20})(A^{30r} - A^{23}) \cdots (A^{30r} - A) \bmod N$, 然后对于 $33 < r \leq 3334$ 计算 $\gcd(c_r, N)$ 。

20. 参见 H. C. Williams, *Math. Comp.* **39** (1982), 225 ~ 234。

21. 有关这个猜测的某种有趣的理论已经由 Eric Bach 作了介绍, *Information and Computation* **90** (1991), 139 ~ 155。

22. 仅当随机数 x 没有揭示出 n 是非素数这一事实时, 算法 P 才算失败。比如说, 如果 $x^q \bmod n = 1$ 或者如果对于 $0 \leq j < k$, 数 $x^{2^j q}$ 之一同余于 -1 (模 n), 则认为 x 是坏的。由于 1 是坏的, 我们有 $p_n = (b_n - 1)/(n - 2) < b_n/(n - 1)$, 其中 b_n 是当 n 不是素数时, 使得 $1 \leq x < n$ 的坏 x 的个数。

每一个坏 x 满足 $x^{n-1} \equiv 1 \pmod{n}$ 。当 p 为素数时, 对于 $1 \leq x \leq p^e$, 同余式 $x^q \equiv 1 \pmod{p^e}$ 的解的个数是对于 $0 \leq y < p^{e-1}(p-1)$, $qy \equiv 0 \pmod{p^{e-1}(p-1)}$ 的解的个数, 即 $\gcd(q, p^{e-1}(p-1))$, 因为我们可以用 a^y 代替 x , 其中 a 是一个原根。

设 $n = n_1^{e_1} \cdots n_r^{e_r}$, 其中 n_i 是不同的素数。按照中国剩余定理, 同余式 $x^{n-1} \equiv 1 \pmod{n}$ 的解的个数是 $\prod_{i=1}^r \gcd(n-1, n_i^{e_i-1}(n_i-1))$, 而这至多是 $\prod_{i=1}^r (n_i-1)$, 因为 n_i 与 $n-1$ 互素。如果某个 $e_i > 1$, 我们有 $n_i - 1 \leq \frac{2}{9} n_i^{e_i}$, 因此解的个数至多是 $\frac{2}{9} n$; 在这种情况下 $b_n \leq \frac{2}{9} n \leq \frac{1}{4} (n-1)$, 因为 $n \geq 9$ 。

因此我们可以假定, n 是不同素数的乘积 $n_1 \cdots n_r$ 。设 $n_i = 1 + 2^{k_i} q_i$, 其中 $k_1 \leq \cdots \leq k_r$ 。则 $\gcd(n-1, n_i-1) = 2^{k_i} q_i'$, 其中 $k_i' = \min(k, k_i)$, 且 $q_i' = \gcd(q, q_i)$ 。按模 n_i 计算, 使得 $x^q \equiv 1$ 的 x 的个数是 q_i' ; 对于 $0 \leq j < k_i'$ 使得 $x^{2^j q} \equiv -1$ 的 x 的个数是 $2^j q_i'$, 否则为 0。由于 $k \geq k_1$, 我们有 $b_n = q_1' \cdots q_r' (1 + \sum_{0 \leq j < k_1} 2^{jr})$ 。

为完成这个证明, 只须证明 $b_n \leq \frac{1}{4} q_1 \cdots q_r 2^{k_1 + \cdots + k_r} = \frac{1}{4} \varphi(n)$, 因为 $\varphi(n) < n - 1$ 。我们有

$$(1 + \sum_{0 \leq j < k_1} 2^{jr}) / 2^{k_1 + \cdots + k_r} \leq (1 + \sum_{0 \leq j < k_1} 2^{jr}) / 2^{k_1 r} =$$

$$1/(2^r - 1) + (2^r - 2)/(2^{k_1 r}(2^r - 1)) \leq 1/2^{r-1}$$

所以除非 $r=2$ 和 $k_1=k_2$, 否则即得出结果。如果 $r=2$, 习题 9 表明, $n-1$ 不是 n_1-1 和 n_2-1 两者的倍数。因此如果 $k_1=k_2$, 我们不能有 $q'_1=q_1$ 和 $q'_2=q_2$ 两者。

由此得出, 在此情况下, $q'_1 q'_2 \leq \frac{1}{3} q_1 q_2$ 和 $b_n \leq \frac{1}{6} \varphi(n)$ 。

[参考文献: *J. Number Theory* **12** (1980), 128~138.] 这个证明表明, 仅在两种情况下, 即当 n 是 $(1+2q_1)(1+4q_1)$ 或特殊形式 $(1+2q_1)(1+2q_2)(1+2q_3)$ 的 Carmichael 数时 p_n 接近于 $\frac{1}{4}$ 。例如, 当 $n=49939 \cdot 99877$ 时, 我们有 $b_n = \frac{1}{4}(49938 \cdot 99876)$ 和 $p_n \approx .24999$; 当 $n=1667 \cdot 2143 \cdot 4523$ 时, 我们有 $b_n = \frac{1}{4}(1666 \cdot 2142 \cdot 4522)$, $p_n \approx .24968$ 。关于进一步的注释见下一答案。

23. a) 也许除了互反律外, 这些证明是简单的。设 $p = p_1 \cdots p_s$ 和 $q = q_1 \cdots q_r$ 其中 p_i 和 q_j 是素数, 则

$$\left(\frac{p}{q}\right) = \prod_{i,j} \left(\frac{p_i}{q_j}\right) = \prod_{i,j} (-1)^{(p_i-1)(q_j-1)/4} \left(\frac{q_j}{p_i}\right) = (-1)^{\sum_{i,j} (p_i-1)(q_j-1)/4} \left(\frac{q}{p}\right)$$

所以我们只需验证 $\sum_{i,j} (p_i-1)(q_j-1)/4 \equiv (p-1)(q-1)/4 \pmod{2}$ 。但是 $\sum_{i,j} (p_i-1)(q_j-1)/4 = (\sum_i (p_i-1)/2)(\sum_j (q_j-1)/2)$ 是奇数当且仅当奇数个 p_i 和奇数个 q_j 同余 3 (模 4), 而这成立当且仅当 $(p-1)(q-1)/4$ 为奇数。[G. G. J. Jacobi, *Bericht Königl. Preuß. Akad. Wiss. Berlin* **2** (1837), 127~136; V. A. Lebesgue, *J. Math. Pures. Appl.* **12** (1847), 497~520, 讨论了其有效性。]

b) 如同在习题 22 中那样, 我们可以假定 $n = n_1 \cdots n_r$, 其中 $n_i = 1 + 2^{k_i} q_i$ 是不同的素数, 且 $k_1 \leq \cdots \leq k_r$; 我们设 $\gcd(n-1, n_i-1) = 2^{k'_i} q'_i$ 而且如果它错误地把 n 弄成像素数, 则我们称 x 是坏的。命 $\Pi_n = \prod_{i=1}^r q'_i 2^{\min(k_i, k'_i)}$ 是 $x^{(n-1)/2} \equiv 1$ 的解的个数。使 $\left(\frac{x}{n}\right) = 1$ 的坏 x 的个数是 Π_n , 当 $k_1 < k$ 时还要乘以一个额外因子 $\frac{1}{2}$ 。(需要 $\frac{1}{2}$ 这个因子以确保当 $k_i < k$ 时有偶数个 n_i 使 $\left(\frac{x}{n_i}\right) = -1$ 。) 如果 $k_1 = k$, 则使 $\left(\frac{x}{n}\right) = -1$ 的坏 x 的个数是 Π_n , 否则为 0。[如果 $x^{(n-1)/2} \equiv -1 \pmod{n_i}$ 且如果 $k_i = k$, 我们有 $\left(\frac{x}{n_i}\right) = -1$; 如果 $k_i > k$, 则 $\left(\frac{x}{n_i}\right) = +1$; 如果 $k_i < k$ 则矛盾。如果 $k_1 = k$, 则有奇数个 k_i 等子 k 。]

注: 仅当 n 是 $k_r < k$ 的一个 Carmichael 数时, 一个坏的猜测的概率 $> \frac{1}{4}$; 例如, $n = 7 \cdot 13 \cdot 19 = 1729$, 这是在另一个场合被 Ramanujan 弄出名的一个数。Louis Monier 已经推广了上述分析以得出一般的坏 x 个数的下列准确的公式:

$$b_n = \left(1 + \frac{2^{k_1} - 1}{2^r - 1}\right) \prod_{i=1}^r q_i'$$

$$b_n' = \delta_n \prod_{i=1}^r \gcd\left(\frac{n-1}{2}, n_i - 1\right)$$

这里 b_n' 是本题中坏的 x 的个数, δ_n 是 2 (如果 $k_1 = k$) 或 $\frac{1}{2}$ (如果 $k_i < k$ 和对某个 i , e_i 是奇数) 或 1 (否则)。

c) 如果 $x^q \bmod n = 1$, 则 $1 = \left(\frac{x^q}{n}\right) = \left(\frac{x}{n}\right)^q = \left(\frac{x}{n}\right)$ 。如果 $x^{2^j q} \equiv -1 \pmod{n}$, 则对于 n 的所有素因子 n_i , x 的模 n_i 阶必定是 2^{j+1} 的奇数倍。设 $n = n_1^{e_1} \cdots n_r^{e_r}$ 和 $n_i = 1 + 2^{j_i+1} q_i''$; 则 $\left(\frac{x}{n_i}\right) = (-1)^{q_i''}$, 所以根据 $\sum e_i q_i''$ 是偶数或奇数, $\left(\frac{x}{n}\right) = +1$ 或 -1 。由于 $n \equiv (1 + 2^{j+1} \sum e_i q_i'') \pmod{2^{j+2}}$, 和 $\sum e_i q_i''$ 是奇数当且仅当 $j+1 = k$ [Theoretical Comp. Sci. 12 (1980), 97~108]。

24. 设 M_1 是一个矩阵, 对于在 $1 \leq n \leq N$ 范围内的每一非素奇数 n 它有一行, 并有编号由 2 到 N 的 $N-1$ 个列; 如果 n 通不过算法 P 的 x 检验, 则 n 行 x 列处的元素为 1, 否则为 0。当 $N = qn + r$ 和 $0 \leq r < n$ 时, 我们知道行 n 至多含 $-1 + q(b_n + 1) + \min(b_n + 1, r) < q\left(\frac{1}{4}(n-1) + 1\right) + \min(b_n + 1, r) \leq \frac{1}{3}qn + \min\left(\frac{1}{4}n, r\right) = \frac{1}{3}N + \min\left(\frac{1}{4}n - \frac{1}{3}r, \frac{2}{3}r\right) \leq \frac{1}{3}N + \frac{1}{6}n \leq \frac{1}{2}N$ 个元素等于 0, 所以在此矩阵中至少有一半元素是 1。于是, M_1 的某个列 x_1 至少有一半的元素等于 1。删去列 x_1 和在这列中含 1 的所有行, 就剩下了具有类似性质的一个矩阵 M_2 ; 重复运用这个构造能产生具有 $N-r$ 个列和少于 $N/2^r$ 个行的矩阵 M_r , 而且每行至少有 $\frac{1}{2}(N-1)$ 个元素等于 1 [参考 FOCS 19 (1978), 78]。

[一个类似的证明隐含着如下事实: 存在一个无穷序列 $x_1 < x_2 < \cdots$, 使得数 $n > 1$ 是素数当且仅当对子 $x = x_1, \cdots, x = x_m$ 它通过算法 P 的 x 检验, 其中 $m = \frac{1}{2}(\lfloor \lg n \rfloor (\lfloor \lg n \rfloor - 1))$ 。是否存在一个有此性质的序列 $x_1 < x_2 < \cdots$, 但 $m = O(\log n)$?)

25. 这一定理首先是由 von Mangoldt 严格地证明的 [Crelle 114 (1895), 255~305], 他事实上证明 $O(1)$ 项是 $C + \int_x^\infty dt / ((t^2 - 1)t \ln t)$, 如果 n 是一个素数的第 k 次幂, 则减去 $1/2k$ 。常数 C 是 $\text{li } 2 - \ln 2 = \gamma + \ln \ln 2 + \sum_{n \geq 2} (\ln 2)^n / nn! = 0.35201\ 65995\ 57547\ 47542\ 73567\ 67736\ 43656\ 84471 + \dots$

[关于在 von Mangoldt 的论文之后的 100 年期间发展的综述, 参见 A. A. Karatsuba, *Complex Analysis in Number Theory* (CRC Press, 1995)。关于黎曼假设和有关整数的具体问题之间的关系出色介绍, 也可参见 Eric Bach 和 Jeffrey Shallit, *Algorithmic Number Theory 1*, (MIT Press, 1996), 第 8 章。]

26. 如果 N 不是素数, 它有素因子 $q \leq \sqrt{N}$ 。由假设, f 的每个素因子 p 有一个整数 x_p 使得 x_p 的模 q 阶是 $N-1$ 的因子, 但不是 $(N-1)/p$ 的因子。因此如果 p^k 整除 f , 则 x_p 的模 q 阶是 p^k 的倍数。习题 3.2.1.2-15 现在告诉我们, 有一个模 q 阶为 f 的元素 x 。但这是不可能的, 因为它意味着 $q^2 \geq (f+1)^2 \geq (f+1)f \geq N$, 而且等式不可能成立。[*Proc. Camb. Phil. Soc.* **18** (1914), 29~30。]

27. 如果 k 不为 3 所整除且如果 $k \leq 2^n + 1$, 则数 $k \cdot 2^n + 1$ 是素数当且仅当 $3^{2^{n-1}k} \equiv -1 \pmod{k \cdot 2^n + 1}$ 。因为如果这个条件成立, 由习题 26 知 $k \cdot 2^n + 1$ 是素数; 而且如果 $k \cdot 2^n + 1$ 是素数, 则由二次互反律知数 3 是二次非剩余模 $k \cdot 2^n + 1$, 因为 $(k \cdot 2^n + 1) \bmod 12 = 5$ 。[Proth 在 *Comptes Rendus Acad. Sci. Paris* **87** (1878), 926 中不加证明地指出了这个测试。]

为了以必要的效率实现 Proth 测试, 我们需要有能力以大约和计算 $x^2 \bmod (2^n - 1)$ 相同的速度计算 $x^2 \bmod (k \cdot 2^n + 1)$ 。设 $x^2 = A \cdot 2^n + B$; 于是 $x^2 \equiv B - \lfloor A/k \rfloor + 2^n(A \bmod k)$, 所以当 k 是一个单精度的数时, 容易得到剩余。

[为检测形如 $3 \cdot 2^n + 1$ 的数的素约性, 工作只是稍微困难一点; 我们首先试验随机的单精度数直到通过二次互反律求出一个二次非剩余模 $3 \cdot 2^n + 1$, 然后使用这个数代替上述测试中的“3”。如果 $n \bmod 4 \neq 0$, 则可以使用数 5。当 $n = 1, 2, 5, 6, 8, 12, 18, 30, 36, 41, 66, 189, 201, 209, 276, 353, 408, 438, 534, 2208, 2816, 3168, 3189, 3912, 20909, 34350, 42294, 42665, 44685, 48150, 55182, 59973, 80190, 157169, 213321$ 时, $3 \cdot 2^n + 1$ 是素数; 而且在小于等于 300000 的数中, 没有其它的 n 了; 而当 $n = 1, 3, 7, 13, 15, 25, 39, 55, 75, 85, 127, 1947, 3313, 4687, 5947, 13165, 23473, 26607, 125413, 209787, 240937$ 时, $5 \cdot 2^n + 1$ 是素数而且在小于等于 300000 的数中没有其它的 n 了。见 R. M. Robinson, *Proc. Amer. Math. Soc.* **9** (1958), 673~681; G. V. Cormack 和 H. C. Williams, *Math. Comp.* **35** (1980), 1419~1421; H. Dubner 和 W. Keller, *Math. Comp.* **64** (1995), 397~405; J. S. Young, *Math. Comp.* **67** (1998), 1735~1738。]

28. $f(p, p^2d) = 2/(p+1) + f(p, d)/p$, 因为 $1/(p+1)$ 是 A 为 p 的倍数的概率。当 $d \bmod p \neq 0$ 时 $f(p, pd) = 1/(p+1)$ 。 $f(2, 4k+3) = \frac{1}{3}$, 因为 $A^2 - (4k+3)B^2$ 不可能是 4 的倍数; $f(2, 8k+5) = \frac{2}{3}$, 因为 $A^2 - (8k+5)B^2$ 不可能是 8 的倍数; $f(2, 8k+1) = \frac{1}{3} - \frac{1}{3} + \frac{1}{3} + \frac{1}{6} + \frac{1}{12} + \cdots = \frac{4}{3}$ 。对于奇的 p , 如果 $d^{(p-1)/2} \bmod p$ 分别等于 $(1, p-1)$, 则 $f(p, d) = (2p/(p^2-1), 0)$ 。

29. 在非负整数 x_i 中, 不等式 $x_1 + \cdots + x_m \leq r$ 的解的个数是 $\binom{m+r}{r} \geq m^r / r!$, 而且这些解的每一个对应于惟一的整数 $p_1^x \cdots p_m^x \leq n$ 。[在对于所有的 j , p_j 是第 j 个素数的特殊情况下, 关于更精确的估计, 见 N. G. de Bruijn, *Indag. Math.* **28** (1966), 240 ~ 247; H. Halberstam, *Proc. London Math. Soc.* (3) **21** (1970), 102 ~ 107.]

30. 如果 $p_1^{e_1} \cdots p_m^{e_m} \equiv x_i^2 \pmod{q_i}$, 我们可以求得 y_i 使得 $p_1^{e_1} \cdots p_m^{e_m} \equiv (\pm y_i)^2 \pmod{q_i^{d_i}}$, 因此由中国剩余定理我们得到 X 的 2^d 个值, 使得 $X^2 \equiv p_1^{e_1} \cdots p_m^{e_m} \pmod{N}$ 。这样的 (e_1, \dots, e_m) 至多对应于 $\binom{r}{r/2}$ 对有所提示的性质的 $(e'_1, \dots, e'_m; e''_1, \dots, e''_m)$ 。现在对于 2^d 个二进制数 $a = (a_1 \cdots a_d)_2$ 的每一个, 设 n_a 是使得 $(p_1^{e'_1} \cdots p_m^{e'_m})^{(q_i^{-1})/2} \equiv (-1)^{a_i} \pmod{q_i}$ 的指数 (e'_1, \dots, e'_m) 的个数; 我们已经证明所需要的整数 X 的个数 $\geq 2^d (\sum_a n_a^2) / \binom{r}{r/2}$ 。由于 $\sum_a n_a$ 是允许重复地从 m 个对象的一个集合选择至多 $r/2$ 个对象的方式个数, 即 $\binom{m+r/2}{r/2}$, 我们有 $\sum_a n_a^2 \geq \binom{m+r/2}{r/2}^2 / 2^d \geq m^r / (2^d (r/2)!^2)$ 。[参见 J. Algorithms 3 (1982), 101 ~ 127, 其中 Schnorr 介绍了对定理 D 的许多进一步的改进。]

31. 置 $n = M$, $pM = 4m$, $dM = 2m$, 以证明 $\Pr(X \leq 2m) \leq e^{-m/2}$ 。

32. 设 $M = \lfloor \sqrt[3]{N} \rfloor$, 并设每个信息的位置 x_i 被限制在范围 $0 \leq x < M^3 - M^2$ 中。如果 $x \geq M$, 则和以前一样把它编码为 $x^3 \bmod N$, 但如果 $x < M$ 就把编码改变成 $(x + yM)^3 \bmod N$, 其中 y 是在 $M^2 - M \leq y \leq M^2$ 范围内的随机数。为了译码, 首先取立方根; 而且如果结果是 $M^3 - M^2$ 或更多, 则取模 M 余数。

34. 设 P 是 $x^m \bmod p = 1$ 的概率且 Q 是 $x^m \bmod q = 1$ 的概率。 $\gcd(x^m - 1, N) = p$ 或 q 的概率是 $P(1 - Q) + Q(1 - P) = P + Q - 2PQ$ 。如果 $P \leq \frac{1}{2}$ 或 $Q \leq \frac{1}{2}$, 则这个概率 $\geq 2(10^{-6} - 10^{-12})$, 所以在大约 $10^6 \log m$ 次模 N 下的算术运算之后, 我们有一个好机会寻找一个因子。另一方面, 如果 $P > \frac{1}{2}$ 和 $Q > \frac{1}{2}$, 则 $P \approx Q \approx 1$, 因为我们有一般的公式 $P = \gcd(m, p-1)/p$; 于是在这种情况下 m 是 $\text{lcm}(p-1, q-1)$ 的倍数。设 $m = 2^k r$, 其中 r 为奇数, 构造序列 $x^r \bmod N, x^{2r} \bmod N, \dots, x^{2^k r} \bmod N$; 如同在算法 P 中那样, 我们将以大于等于 $\frac{1}{2}$ 的概率发现 1 的头一次出现的前边有不同于 $N-1$ 的一个 y 值, 因此 $\gcd(y-1, N) = p$ 或 q 。

35. 设 $f = (p^{q-1} - q^{p-1}) \bmod N$, 由于 $p \bmod 4 = q \bmod 4 = 3$, 我们有 $\left(\frac{-1}{p}\right) =$

$\left(\frac{-1}{q}\right) = \left(\frac{f}{p}\right) = -\left(\frac{f}{q}\right) = -1$, 我们还有 $\left(\frac{2}{p}\right) = -\left(\frac{2}{q}\right) = 1$ 。给定在 $0 \leq x \leq \frac{1}{8}(N-5)$ 范围内的一个消息 x , 令 $\bar{x} = 4x+2$ 或 $8x+4$, 按哪个 x 满足 $\left(\frac{\bar{x}}{N}\right) = 1$ 而定; 然后传送消息 $\bar{x}^2 \bmod N$ 。

为了对这消息进行译码, 我们首先使用一个 SQRT 框来求出惟一的数 y , 使得 $y^2 \equiv \bar{x}^2 \bmod N$ 且 $\left(\frac{y}{N}\right) = 1$ 和 y 是偶数。于是 $y = \bar{x}$, 因为 \bar{x}^2 的其它三个平方根是 $N - \bar{x}$ 和 $(\pm f\bar{x}) \bmod N$; 这些根的头一个是奇数, 而其它两个根有负的雅可比符号。如果 $y \bmod 4 = 2$, 则通过置 $x \leftarrow \lfloor y/4 \rfloor$, 否则 $x \leftarrow \lfloor y/8 \rfloor$ 来完成译码。

任何可以对这种编码进行译码的人也都能找到 N 的诸因子, 因为当 $\left(\frac{\bar{x}}{N}\right) = -1$ 时, 对一个错误消息 $\bar{x}^2 \bmod N$ 的译码揭示了 $(\pm f) \bmod N$, 而且 $((\pm f) \bmod N) - 1$ 和 N 有一个非平凡的 gcd。[参考文献: IEEE Transactions IT-26 (1980), 726 ~ 729。]

36. 由(4), 第 m 个素数等于 $m \ln m + m \ln \ln m - m + m \ln \ln m / \ln m - 2m / \ln m + O(m(\log \log m)^2(\log m)^{-2})$, 尽管对于这个问题我们只需要较弱的估计 $p_m = m \ln m + O(m \log \log m)$ 。(我们将假定, p_m 是第 m 个素数, 因为这对应于 V 是一致分布的假定。)如果我们选择 $\ln m = \frac{1}{2}c \sqrt{\ln N \ln \ln N}$, 其中 $c = O(1)$, 我们发现 $r = c^{-1} \sqrt{\ln N / \ln \ln N} - c^{-2} - c^{-2}(\ln \ln \ln N / \ln \ln N) - 2c^{-2} \left(\ln \frac{1}{2}c \right) / \ln \ln N + O(\sqrt{\ln N \ln \ln N} / \ln N)$ 。原来估计的运行时间(22)现在出人意外地简化成 $\exp(f(c, N) \sqrt{\ln N \ln \ln N} + O(\log \log N))$, 其中 $f(c, N) = c + (1 - (1 + \ln 2) / \ln \ln N) c^{-1}$ 。使 $f(c, N)$ 取极小的 c 的值是 $\sqrt{1 - (1 + \ln 2) / \ln \ln N}$, 所以我们得到估计式

$$\exp(2 \sqrt{\ln N \ln \ln N} \sqrt{1 - (1 + \ln 2) / \ln \ln N} + O(\log \log N))$$

当 $N = 10^{50}$ 时, 这给出 $\epsilon(N) \approx .33$, 它仍比我们所观察到的大得多。

注: \sqrt{D} 的部分商的情况似乎有与 4.5.3 小节中随机实数得到的分布一样的特性。例如, 在数 $10^{18} + 314159$ 的平方根的头 100 万个部分商中, A_n 分别是 (1, 2, 3, 4) 的情形恰好有 (415236, 169719, 93180, 58606) 种。而且, 由习题 4.5.3-12(c) 和等式 4.5.3-(12) 我们有 $V_{n+1} = |p_n^2 - Dq_n^2| = 2\sqrt{D}q_n |p_n - \sqrt{D}q_n| + O(q_n^{-2})$ 。因此我们可以期望 $V_n/2\sqrt{D}$ 实质上有和量 $\theta_n(x) = q_n |p_n - xq_n|$ 类似的特性, 其中 x 是一个随机实数。已知随机量 θ_n 对于 $0 \leq \theta \leq 1$ 有近似的密度 $\min(1, \theta^{-1} - 1) / \ln 2$ [见 Bosma, Jager 及 Wiedijk, Indag. Math. 45 (1983), 281 ~ 299], 当 $\theta \leq 1/2$ 时它是一致的。因此除 V_n 的大小之外还有其它原因解释算法 E 的不合理效率。

37. 把习题 4.5.3-12 应用于数 $\sqrt{D+R}$, 可以看到周期部分立即开始, 然后逆向遍历这个周期, 以检验它的回文性质。[由此得出, 这个周期的下一半给出和头一半相同的 V , 通过在步骤 E5 中当 $U=U'$ 或 $V=V'$ 时结束之, 可以使算法 E 较早地停下来。但是, 这个周期一般很长, 我们甚至不可能接近于它的一半, 所以没有理由使这个算法更复杂。]

38. 令 $r = (10^{50} - 1)/9$ 。于是 $P_0 = 10^{49} + 9$; $P_1 = r + 3 \cdot 10^{46}$; $P_2 = 2r + 3 \cdot 10^{47} + 7$; $P_3 = 3r + 2 \cdot 10^{49}$; $P_4 = 4r + 2 \cdot 10^{49} - 3$; $P_5 = 5r + 3 \cdot 10^{49} + 4$; $P_6 = 6r + 2 \cdot 10^{48} + 3$; $P_7 = 7r + 2 \cdot 10^{25}$ (非常漂亮); $P_8 = 8r + 10^{38} - 7$; $P_9 = 9r - 8000$ 。

39. 注意当 $q-1$ 恰有 2 和 p 作为素因子时, 容易证明 q 的素约性。2 的仅有的后继是费马素数, 而且第六个费马素数的存在与否是数论中最著名的未解决的问题之一。因此我们大概将永远不知道如何来确定一个任意的整数是否有任何后继。然而, 在某些情况下, 这是可能的; 例如, 在 W. Siewpiński 证明了没有后继的无穷多个奇数的存在性 [Elemente der Math. 15 (1960), 73 ~ 74] 之后, John Selfridge 于 1962 年证明 78557 和 271129 没有 [参见 AMM 70 (1963), 101 ~ 102]。也许 78557 是这些当中最小的, 尽管按照 G. Jaeschke 和 W. Keller [Math. Comp. 40 (1983), 381 ~ 384, 661 ~ 673; 45 (1985), 637] 的论述, 在 1983 年仍然存在其它 69 个竞争者来竞争这个荣誉。

关于更传统的素数链的 “Cunningham” 形式, 其中转换是 $p \rightarrow 2p \pm 1$, 参见 Gunter Löh, Math. Comp. 53 (1989), 751 ~ 759。特别是, Löh 发现, 对于 $0 \leq k < 12$, $554688278430 \cdot 2^k - 1$ 是素数。

40. [Inf. Proc. Letters 8 (1979), 28 ~ 31.] 注意在这样一台机器上, 可以很容易地计算 $x \bmod y = x - y \lfloor x/y \rfloor$, 而且我们可以得到像 $0 = x - x$, $1 = \lfloor x/x \rfloor$, $2 = 1 + 1$ 这样简单的常数; 通过测试 $x = 1$ 还是 $\lfloor x/(x-1) \rfloor \neq 0$, 我们可以知道是否 $x > 0$ 。

a) 首先通过重复地除以 2 在 $O(\log n)$ 步内计算 $l = \lfloor \lg n \rfloor$; 同时通过重复地置 $k \leftarrow 2k$, $A \leftarrow A^2$, 在 $O(\log n)$ 步内计算 $k \leftarrow 2^l$ 和 $A \leftarrow 2^{2^{l+1}}$ 。对于主要的计算, 假设我们知道 $t = A^m$, $u = (A+1)^m$, 且 $v = m!$; 于是通过置 $m \leftarrow m+1$, 我们可以把 m 的值加 1, 置 $t \leftarrow At$, $u \leftarrow (A+1)u$, $v \leftarrow vm$; 然后通过置 $m \leftarrow 2m$ 我们可以把 m 的值加倍, 假定 A 充分大, 置 $u \leftarrow u^2$, $v \leftarrow (\lfloor u/t \rfloor \bmod A) v^2$, $t \leftarrow t^2$ 。 (考虑用 A 进制表示的数 u ; A 必须大于 $\binom{2m}{m}$ 。) 现在如果 $n = (a_l \cdots a_0)_2$, 设 $n_j = (a_l \cdots a_j)_2$; 如果 $m = n_j$ 和 $k = 2^j$ 及 $j > 0$, 通过置 $k \leftarrow \lfloor k/2 \rfloor$, $m \leftarrow 2m + (\lfloor n/k \rfloor \bmod 2)$, 我们可以把 j 减 1。因此对于 $j = l, l-1, \dots, 0$, 我们可以在 $O(\log n)$ 步内计算 $n_j!$ 。 [由 Julia Robinson 给出的另一个解, 是当 $B > (2n)^{n+1}$ 时计算 $n! = \lfloor B^n / \binom{B}{n} \rfloor$; 参考 AMM 80 (1973), 250 ~ 251, 266。]

b) 如同在 a) 中一样, 首先计算 $A = 2^{2^{l+2}}$, 然后找使得 $2^{k+1}! \bmod n = 0$ 的最小

的 $k \geq 0$ 。如果 $\gcd(n, 2^k!) \neq 1$, 则命 $f(n)$ 是这个值; 注意, 用欧几里得算法可以在 $O(\log n)$ 步内计算这个 \gcd 。不然, 我们将找出使得 $\binom{m}{\lfloor m/2 \rfloor} \bmod n = 0$ 的最小整数 m , 并令 $f(n) = \gcd(m, n)$ 。(注意在这种情况下 $2^k < m \leq 2^{k+1}$, 因此, $\lfloor m/2 \rfloor \leq 2^k$, 而且 $\lfloor m/2 \rfloor!$ 同 n 互素; 因此 $\binom{m}{\lfloor m/2 \rfloor} \bmod n = 0$ 当且仅当 $m! \bmod n = 0$ 。而且 $n \neq 4$ 。)

为了用有限个寄存器计算 m , 我们可以使用斐波那契数(参考算法 6.2.1F)。假设我们知道 $s = F_j, s' = F_{j+1}, t = A^F, t' = A^{F_{j+1}}, u = (A+1)^{2F}, u' = (A+1)^{2F_{j+1}}, v = A^m, w = (A+1)^{2m}, \binom{2m}{m} \bmod n \neq 0$, 且 $\binom{2(m+s)}{m+s} \bmod n = 0$ 。若 $m = F_{j+1}$, 对于适当的 j , 容易在 $O(\log n)$ 步内做到这一点; 而且 A 将比 $2^{2(m+s)}$ 更大。如果 $s = 1$, 我们就置 $f(n) = \gcd(2m+1, n)$ 或 $\gcd(2m+2, n)$, 它们无论哪一个不等于 1, 就终止此算法。不然, 我们把 j 减 1 如下: 置 $r \leftarrow s, s \leftarrow s' - s, s' \leftarrow r, r \leftarrow t, t \leftarrow \lfloor t'/t \rfloor, t' \leftarrow r, r \leftarrow u, u \leftarrow \lfloor u'/u \rfloor, u' \leftarrow r$; 然后如果 $(\lfloor wu/vt \rfloor \bmod A) \bmod n \neq 0$, 则置 $m \leftarrow m + s, w \leftarrow wu, v \leftarrow vt$ 。

[这个问题能否用少于 $O(\log n)$ 次运算来解? n 的最小或最大的素因子能否在 $O(\log n)$ 次运算内算出?]

41. a) 当 $1 \leq m \leq x$ 时显然有 $\pi(x) = \pi(m) + f_1(x, m) = \pi(m) + f(x, m) - f_0(x, m) - f_2(x, m) - f_3(x, m) - \dots$ 。置 $x = N^3, m = N$, 并且注意对于 $k > 2, f_k(N^3, N) = 0$ 。

b) 我们有 $f_2(N^3, N) = \sum_{N < p \leq q} [pq \leq N^3] = \sum_{N < p \leq N^{3/2}} (\pi(N^3/p) - \pi(p) + 1) = \sum_{N < p \leq N^{3/2}} \pi(N^3/p) - \left(\binom{\pi(N^{3/2})}{2} \right) + \left(\binom{\pi(N)}{2} \right)$, 其中 p 和 q 为素数。因此 $f_2(1000, 10) = \pi\left(\frac{1000}{11}\right) + \pi\left(\frac{1000}{13}\right) + \pi\left(\frac{1000}{17}\right) + \pi\left(\frac{1000}{19}\right) + \pi\left(\frac{1000}{23}\right) + \pi\left(\frac{1000}{29}\right) + \pi\left(\frac{1000}{31}\right) - \left(\binom{\pi(31)}{2} \right) + \left(\binom{\pi(10)}{2} \right) = 24 + 21 + 16 + 15 + 14 + 11 + 11 - 55 + 6 = 63$ 。

c) 提示的恒等式简单指出, 一个 p_j 幸存者是非 p_j 的倍数的 p_{j-1} 幸存者。显然, $f(N^3, N) = f(N^3, p_{\pi(N)})$ 。应用这个恒等式直到达到项 $f(x, p_j)$ 为止, 其中或者 $j = 0$, 或者 $x \leq N^2$; 结果是

$$f(N^3, N) = \sum_{k=1}^{N-1} \mu(k) f\left(\frac{N^3}{k}, 1\right) - \sum_{j=1}^{\pi(N)} \sum_{N/p_j \leq k < N} \mu(k) f\left(\frac{N^3}{kp_j}, p_{j-1}\right) [k \text{ 是一个 } p_j \text{ 幸存者}]$$

现在 $f(x, 1) = \lfloor x \rfloor$, 所以当 $N = 10$ 时, 头一个和是 $1000 - 500 - 333 - 200 + 166 - 142 = -9$ 。第二个和是 $-f\left(\frac{1000}{10}, 1\right) - f\left(\frac{1000}{14}, 1\right) - f\left(\frac{1000}{15}, 2\right) - f\left(\frac{1000}{21}, 2\right) -$

$f\left(\frac{1000}{35}, 3\right) = -100 - 71 - 33 - 24 - 9 = -237$ 。因此 $f(1000, 10) = -9 + 237 = 228$, 且 $\pi(1000) = 4 + 228 - 1 - 63 = 168$ 。

d) 如果 $N^2 \leq 2^m$ 我们可以构造一个数组, 其中, 对于 $1 \leq n \leq N^2$, $a_{2^m - 1 + n} = [n + 1$ 是 p_j 幸存者]表示在 j 次扫描之后的筛, 而且对于 $1 \leq n < 2^m$, $a_n = a_{2n} + a_{2n+1}$ 。于是当 $x \leq N^2$ 时容易在 $O(m)$ 步内计算 $f(x, p_i)$, 并且在 $O(N^2 m/p)$ 步内从筛中删去 p 的倍数。计算 $f(N^2, N)$ 的总共运行时间将成为 $O(N^2 \log N \log \log N)$, 因为 $\sum_{j=1}^{\pi(N)} 1/p_j = O(\log \log N)$ 。

如果我们把筛分成为大小为 N 的 N 个部分, 而且在每个部分分开地工作, 则存储要求可以从 $2N^2 m$ 降低到 $2Nm$ 。在主计算开始之前, 对于 $1 \leq j \leq \pi(N)$, p_j 的辅助表和对于 $1 \leq k \leq N$, $\mu(k)$ 以及 k 的最小素因子的辅助表是有帮助和容易构造的。

[见 *Math. Comp.* **44** (1985), 537 ~ 560。一个类似的方法首先是由 D. F. E. Meissel 引入的, *Math. Annalen* **2** (1870), 636 ~ 642; **3** (1871), 523 ~ 525; **21** (1883), 304; **25** (1885), 251 ~ 257。D. H. Lehmer 在 *Illinois J. Math.* **3** (1959), 381 ~ 388 做了若干改进。但无论是 Meissel, 还是 Lehmer 都没有像上边所描述的方法那样有效的对于递推的一个停止规则。Lagarias 和 Odlyzko 也建立了一个完全不同的方法, 由此利用解析数论原理, 可以在 $O(N^{1/2+\epsilon})$ 步内计算 $\pi(N)$ 。后一个方法已经被用于目前的世界记录的素数的计算, 它是由 Deléglise 和 Rivat 得到的, *Math. Comp.* **65** (1996), 235 ~ 245; $\pi(10^{20}) = 2220819602560918840$ 。]

42. L1. [初始化] 求 \bar{r} 使得 $r\bar{r} \equiv 1 \pmod{s}$; 然后置 $r' \leftarrow n\bar{r} \pmod{s}$, $u \leftarrow r'\bar{r} \pmod{s}$, $v \leftarrow s$, $w \leftarrow (n - rr')\bar{r}/s \pmod{s}$, $\theta \leftarrow \lfloor \sqrt{N/s} \rfloor$, $(u_1, u_3) \leftarrow (1, u)$, $(v_1, v_3) \leftarrow (0, v)$ 。(我们要求所有的整数偶 (λ, μ) 使得 $(\lambda s + r)(\mu s + r') = N$; 这意味着 $\lambda u + \mu \equiv w \pmod{s}$, 且 $\sqrt{\lambda \mu v} \leq \theta$ 。我们将以抑制掉 t_2, u_2, v_2 来执行算法 4.5.2X; 关系

$$\lambda t^3 + \mu t_1 \equiv w t_1, \lambda u_3 + \mu u_1 \equiv w u_1, \lambda v_3 + \mu v_1 \equiv w v_1 \pmod{s}$$

将保持不变。)

L2. [试验因子] 如果 $v_1 = 0$, 则每当 $\lambda s + r$ 整除 N 和 $0 \leq \lambda \leq \theta/s$ 时, 输出 $\lambda s + r$ 。如果 $v_3 = 0$, 每当 $\mu s + r'$ 整除 N 和 $0 \leq \mu \leq \theta/s$ 时输出 $N/(\mu s + r')$ 。否则, 如果 $v_1 < 0$, 对于所有使得 $|wv_1 + ks| \leq \theta$ 的 k , 或者如果 $v_1 > 0$, 对所有使得 $0 < wv_1 + ks \leq 2\theta$ 的 k , 以及对于 $\sigma = +1$ 和 -1 , 如果 $d = (wv_1 s + ks^2 + v_3 r + v_1 r')^2 - 4v_1 v_3 N$ 是完全平方, 而且如果数

$$\lambda = \frac{wv_1 s + ks^2 - v_3 r + v_1 r' + \sigma \sqrt{d}}{2v_3 s}, \mu = \frac{wv_1 s + ks^2 + v_3 r - v_1 r' - \sigma \sqrt{d}}{2v_3 s}$$

是正整数, 则输出 $\lambda s + r$ 。(这些是对于 $\lambda v_3 + \mu v_1 = wv_1 + ks$, $(\lambda s + r)(\mu s + r') = N$ 的解。)

1.3. [完成了吗?] 如果 $v_3 = 0$, 则算法结束。

1.4. [除和减] 置 $q \leftarrow \lfloor u_3/v_3 \rfloor$ 。如果 $u_3 = qv_3$, 且 $v_1 < 0$, 则 q 减 1。然后置 $(t_1, t_3) \leftarrow (u_1, u_3) - (v_1, v_3)q$, $(u_1, u_3) \leftarrow (v_1, v_3)$, $(v_1, v_3) \leftarrow (t_1, t_3)$

并返回步骤 1.2。 ■

[参见 *Math. Comp.* 42 (1984), 331~340。步骤 1.2 中的界可以更精确化, 比如确保 $d \geq 0$ 。某些因子可能输出一次以上。]

43. a) 首先确保雅可比符号 $\left(\frac{y}{m}\right)$ 为 +1。(如果它为 0, 则任务是简单的; 如果它是 -1, 则 $y \notin Q_m$ 。) 于是在 $[0, m)$ 中选择随机整数 x_1, \dots, x_n 并令 $X_j = [G(y^2 x_j^4 \bmod m) = (yx_j^2 \bmod m) \bmod 2]$ 。如果 $y \in Q_m$, 我们有 $EX_j \geq \frac{1}{2} + \epsilon$; 否则 $m - y \in Q_m$ 和 $EX_j \leq \frac{1}{2} - \epsilon$ 。如果 $X_1 + \dots + X_n \geq \frac{1}{2}n$, 报告 $y \in Q_m$ 。由习题 1.2.10-21, 出错的概率至多为 $e^{-2\epsilon^2 n}$, 因此我们选择 $n = \lceil \frac{1}{2\epsilon^2} \ln \delta^{-1} \rceil$ 。

b) 求使雅可比符号 $\left(\frac{x}{m}\right) = -1$ 的一个 x 并置 $y \leftarrow x^2 \bmod m$ 。于是 m 的素因子是 $\gcd(x + \sqrt{y}, m)$ 和 $\gcd(x - \sqrt{y}, m)$, 所以我们的任务是当给定 $y \in Q_m$ 时求 $\pm\sqrt{y}$ 。如果对于任何非零的 v 我们可以求 τv , 则我们完成任务, 因为 $\sqrt{y} = (v^{-1} \tau v) \bmod m$, 除非 $\gcd(v, m)$ 是 m 的一个因子。

假定对于某个 $e \geq 1$, $\epsilon = 2^{-e}$ 。在 $[0, m)$ 中选择随机整数 a 和 b , 并且假定我们知道二进分数 α_0 和 β_0 使得

$$\left| \frac{\tau a}{m} - \alpha_0 \right| < \frac{\epsilon}{64}, \quad \left| \frac{\tau b}{m} - \beta_0 \right| < \frac{\epsilon^3}{64}$$

这里 α_0 是 $\epsilon/64$ 的奇数倍, 而 β_0 是 $\epsilon^3/64$ 的奇数倍。还假定我们知道 λa 和 λb 。当然我们并不真正知道 $\alpha_0, \beta_0, \lambda a$ 和 λb , 但我们将尝试所有 $32\epsilon^{-1} \times 32\epsilon^{-3} \times 2 \times 2$ 的可能性。程序的虚假分支是在不正确假定之下运行的, 但不会引起损害。

定义数 $u_{ij} = 2^{-i} \left(a + \left(j + \frac{1}{2} \right) b \right) \bmod m$ 和 $v_{ij} = 2^{-i-1} (a + jb) \bmod m$ 。 u_{ij} 和 v_{ij} 两者都在 $[0, m)$ 中一致分布, 因为 a 和 b 是随机选择的。其次, 对于固定的 i , 对于 $j_0 \leq j < j_0 + l$, 数 u_{ij} 是成对独立的, 而且对于 $j_0 \leq j < j_0 + l$, 只要 l 不超过 m 的最小素因子数, v_{ij} 也是成对独立的。我们将仅仅对于 $-2\epsilon^{-2} \leq j \leq 2\epsilon^{-2}$ 来利用 u_{ij} 和 v_{ij} ; 如果这些值当中有任何一个与 m 有非零公因子, 则我们就完成任务了。

对于所有 $v \perp m$, 如果 $v \in Q_m$, 我们定义 $\chi v = +1$, 如果 $-v \in Q_m$, 我们定义 $\chi v = -1$, 而如果 $\left(\frac{v}{m}\right) = -1$, 则 $\chi v = 0$ 。注意 $\chi^{u_{(t+2)j}} = \chi^{u_{ij}}$, 因为 $u_{ij} = (2^2 u_{(t+2)j}) \bmod m$ 。因此通过对 $0 \leq t \leq 1$ 和 $-2\epsilon^{-2} \leq j < 2\epsilon^{-2}$ 应用算法 A 到 u_{ij}

和 v_{ij} , 对于所有的 t 和 j 我们可确定 $\chi_{u_{ij}}$ 和 $\chi_{v_{ij}}$. 置 $\delta = \frac{1}{1440} \epsilon^2 r^{-1}$ 于该算法中将确保所有的 χ 值以大于等于 $1 - \frac{1}{90}$ 的概率正确。

这个算法至多工作 r 个阶段。对于 $0 \leq t < r$, 在阶段 t 开始处, 我们假定知道 $\lambda 2^{-t} a, \lambda 2^{-t} b$ 以及分数 α_t, β_t , 使得

$$\left| \frac{\tau 2^{-t} a}{m} - \alpha_t \right| < \frac{\epsilon}{2^{t+6}}, \quad \left| \frac{\tau 2^{-t} b}{m} - \beta_t \right| < \frac{\epsilon^3}{2^{t+6}}$$

定义 $\alpha_{t+1} = \frac{1}{2}(\alpha_t + \lambda 2^{-t} a)$ 和 $\beta_{t+1} = \frac{1}{2}(\beta_t + \lambda 2^{-t} b)$; 这保持这些不等式。下一步是求满足

$$\lambda u_{ij} + \lambda 2^{-t} a + j \lambda 2^{-t} b + \lambda 2^{-t-1} b + \left\lfloor \frac{\tau 2^{-t} a + j \tau 2^{-t} b + \tau 2^{-t-1} b}{m} \right\rfloor \equiv 0 \pmod{2}$$

的 $\lambda 2^{-t-1} b$ 。令 $n = 4 \min(r, 2^t) \epsilon^{-2}$; 于是当 $|j| \leq \frac{n}{2}$ 时, 我们有

$$\left| \frac{\tau 2^{-t} a}{m} + j \frac{\tau 2^{-t} b}{m} + \frac{\tau 2^{-t-1} b}{m} - (\alpha_t + j \beta_t + \beta_{t+1}) \right| < \frac{\epsilon}{16}$$

因此如果 $\chi_{u_{ij}} = 1$ 则有可能 $\lambda 2^{-t-1} b = G_j$, 其中 $G_j = (G(u_{ij}^2 y \bmod m) + \lambda 2^{-t} a + j \lambda 2^{-t} b + \lfloor \alpha_t j \beta_t + \beta_{t+1} \rfloor) \bmod 2$ 。更精确地说, 除非 $\tau u_{ij} \leq \frac{\epsilon}{16} m$ 或 $\tau u_{ij} > \left(1 - \frac{\epsilon}{16}\right) m$, 否则我们将有

$$\lfloor (\tau 2^{-t} a + j \tau 2^{-t} b + \tau 2^{-t-1} b) / m \rfloor = \lfloor \alpha_t + j \beta_t + \beta_{t+1} \rfloor$$

令 $Y_j = (2G_j - 1) \chi_{u_{ij}}$ 。如果 $Y_j = +1$, 它支持 $\lambda 2^{-t-1} b = 1$; 如果 $Y_j = -1$, 它支持 $\lambda 2^{-t-1} b = -1$; 如果 $Y_j = 0$, 它就弃权。我们将是民主的, 因此置 $\lambda 2^{-t-1} b = \left[\sum_{j=-n/2}^{n/2-1} Y_j \geq 0 \right]$ 。

$\lambda 2^{-t-1} b$ 是正确的概率是多少? 如果 $\chi_{u_{ij}} \neq 0$ 且 $(\tau u_{ij} < \frac{\epsilon}{16} m$ 或 $\tau u_{ij} > \left(1 - \frac{\epsilon}{16}\right) m$ 或 $G(u_{ij}^2 y \bmod m) \neq \lambda u_{ij})$, 则令 $Z_j = -1$; 否则令 $Z_j = |\chi_{u_{ij}}|$ 。由于 Z_j 是 u_{ij} 的函数, 随机变量 Z_j 成对独立而且有相同分布。令 $Z = \sum_{j=-n/2}^{n/2-1} Z_j$; 如果 $Z > 0$, 则 $\lambda 2^{-t-1} b$ 的值将是正确的。 $Z_j = 0$ 的概率是 $1/2$, $Z_j = +1$ 的概率 $\geq \frac{1}{4} + \frac{\epsilon}{2} - \frac{\epsilon}{8}$; 因此 $EZ_j \geq \frac{3}{4} \epsilon$ 。显然 $\text{var}(Z_j) \leq \frac{1}{2}$ 。所以在有正确假定的程序的分支中出错的机会, 按照切比雪夫不等式, 至多是 $\Pr(Z \leq 0) \leq \Pr((Z - nEZ_j)^2 \geq \frac{9}{16} n^2 \epsilon^2) \leq \frac{8}{9} n^{-1} \epsilon^2 = \frac{2}{9} \min(r, 2^t)^{-1}$ 。

以 v_{ij} 代替 u_{ij} 的一个类似的方法可用来确定 $\lambda 2^{-t-1} a$ 且误差 $\leq \frac{2}{9} \min(r, 2^t)^{-1}$ 。

最终我们将有 $\epsilon^3/2^{t+6} < 1/(2m)$, 所以 $\tau 2^{-t}b$ 将是最接近于 $m\beta_t$ 的整数。于是我们可以计算 $\sqrt{y} = (2^t b^{-1} \tau 2^{-t} b) \bmod m$; 把这个量平方将告诉我们, 我们是否正确。

在 $t < \lg n$ 的诸阶段中, 出错的总的机会以 $\frac{4}{9} \sum_{t=1}^{\infty} 2^{-t} = \frac{4}{9}$ 为上限, 而在随后的阶段中出错的总的机会以 $\frac{4}{9} \sum_{t=\lg n}^{\infty} 2^{-t} = \frac{4}{9}$ 为上限。因此总共的出错机会, 包括 χ 的值不全正确在内, 至多是 $\frac{4}{9} + \frac{4}{9} + \frac{1}{90} = \frac{9}{10}$ 。在寻求 \sqrt{y} 的过程中程序的所有运行的至少 $\frac{1}{10}$ 是成功的。因此平均说来, 在重复这个过程至多 10 次之后, 就将找出 m 的因子。

对于 χ 的计算, 总共的运行时间受 $O(r\epsilon^{-4} \log(r\epsilon^{-2}) T(G))$ 的支配, 加上用于随后猜测的 $O(r^2 \epsilon^{-2} T(G))$, 再加上在所有分支中用于计算 $\alpha_t, \beta_t, \lambda 2^{-t}a$ 及 $\lambda 2^{-t}b$ 的 $O(r^2 \epsilon^{-6})$ 。

这个过程很精彩地描述了随机化算法的许多基本范例, 是由 R. Fischlin 和 C.P. Schnorr [Lecture Notes in Comp. Sci. **1233** (1997), 267~279] 给出的。他们从 Alexi, Chor, Goldreich 及 Schnorr [SICOMP **17** (1988), 194~209] 和由 Ben-Or, Chor 和 Shamir [STOC **15** (1983), 421~430] 较早的方法导出了这一过程。当我们把它同引理 3.5P4 组合在一起时, 我们得到类似于定理 3.5P 的一个定理, 但以序列 3.2.2-(16) 代替 3.2.2-(17)。Fischlin 和 Schnorr 说明如何把这些计算流水线化使得它们的因子分解算法花费 $O(r\epsilon^{-4} \log(r\epsilon^{-1}) T(G))$ 步; 得到的用于“击破”3.2.2-(16) 的时间上限为 $T(F) = O(RN^4 \epsilon^{-4} \log(RN\epsilon^{-1}) (T(G) + R^2))$ 。由这个 O 所隐含的常数因子是较大的, 但并没有错。如果我们能以大于等于 $\frac{1}{2} + \epsilon$ 的概率猜测 $y^{1/a} \bmod 2$, 则当 $a \perp \varphi(m)$ 时, 一个类似的方法可从 RSA 函数 $y = x^a$ 求出 x 来。

44. 假设对于 $1 \leq i \leq k = d(d-1)/2 + 1$, $\sum_{j=0}^d a_{ij} x^j \equiv 0 \pmod{m_i}$, $\gcd(a_{i0}, a_{i1}, \dots, a_{i(d-1)}, m_i) = 1$, 且 $|x| < m_i$, 其中对于 $1 \leq i < j \leq k$, $m_i \perp m_j$ 。还假定 $m = \min \{m_1, \dots, m_k\} > n^{n/2} 2^{n^2/2} d^d$, 其中 $n = d + k$ 。首先求 u_1, \dots, u_k 使得 $u_j \bmod m_j = \delta_{ij}$ 。然后建立 $n \times n$ 矩阵:

$$L = \begin{pmatrix} M & & & & & \\ 0 & mM & & & & \\ \vdots & \vdots & \ddots & & & \\ 0 & 0 & \cdots & m^{d-1}M & & \\ a_{10}u_1 & ma_{11}u_1 & \cdots & m^{d-1}a_{1(d-1)}u_1 & M/m_1d & \\ a_{20}u_2 & ma_{21}u_2 & \cdots & m^{d-1}a_{2(d-1)}u_2 & 0 & M/m_2d \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ a_{k0}u_k & ma_{k1}u_k & \cdots & m^{d-1}a_{k(d-1)}u_k & 0 & 0 & \cdots & M/m_kd \end{pmatrix}$$

其中 $M = m_1 m_2 \cdots m_k$; 对角线以上的所有元素都为零, 因此 $\det L = M^{n-1} m^{k-1} d^{-k}$ 。现在令 $v = (t_0, \dots, t_{d-1}, v_1, \dots, v_k)$ 是长度 $(vL) \leq \sqrt{n 2^n M^{(n-1)/n} m^{(k-1)/n} d^{-k/n}}$ 的一个非零整向量。由于 $M^{(n-1)/n} < M/m^{k/n}$, 因此我们有长度 $(vL) < M/d$ 。令 $c_j = t_j M + \sum_{i=1}^k a_{ij} u_i v_i$ 和 $P(x) = c_0 + c_1 x + \cdots + c_{d-1} x^{d-1}$ 。于是对于 $1 \leq i \leq k$, $P(x) \equiv v_i (a_{i0} + a_{i1} x + \cdots + a_{i(d-1)} x^{d-1}) \equiv 0 \pmod{m_i}$; 因此 $P(x) \equiv 0 \pmod{M}$ 。还有 $|m'c_j| < M/d$; 由此得出 $P(x) = 0$ 。但 $P(x)$ 不恒等于零, 因为条件 $v_i a_{ij} \equiv 0 \pmod{m_i}$ 和 $\gcd(a_{i0}, \dots, a_{i(d-1)}, m_i) = 1$ 意味着 $v_i \equiv 0 \pmod{m_i}$, 而 $|v_i M/m_i d| < M/d$ 意味着 $|v_i| < m_i$; 我们不可能有 $v_1 = \cdots = v_k = 0$ 。因此我们可以求出 x (更精确地说, 是对于 x 的至多 $d-1$ 个可能性), 而且总共的运行时间是 $\lg M$ 的多项式。[Lecture Notes in Comp. Sci. 218 (1985), 403~408.]

45. 事实 1 一个解总存在。首先假设 n 是素数, 如果 $\left(\frac{b}{n}\right) = 1$ 则对于 $y=0$ 有一个解。如果 $\left(\frac{b}{n}\right) = -1$, 令 $j > 0$ 是使得我们有 $\left(\frac{-ja}{n}\right) = -1$ 的极小值; 于是对于某个 x_0 和 y_0 , $x_0^2 - a \equiv -ja$ 和 $b \equiv -ja y_0^2 \pmod{n}$, 因此 $(x_0 y_0)^2 - a y_0^2 \equiv b$ 。其次假设我们已经找到一个解 $x^2 - a y^2 \equiv b \pmod{n}$, 而我们要把它扩充成模 n^2 下的一个解。我们总可以求出 c 和 d , 使得 $(x + cn)^2 - a(y + dn)^2 \equiv b \pmod{n^2}$, 因为 $(x + cn)^2 - a(y + dn)^2 \equiv x^2 - a y^2 + (2cx - 2ayd)n$ 和 $\gcd(2x, 2ay) \perp n$ 。因此当 n 是一个奇素数的幂时, 一个解总存在。(我们需要假定 n 是奇数, 因为例如对于 $x^2 \pm y^2 \equiv 3 \pmod{8}$ 来说没有解。)最后, 由中国剩余定理, 对于所有奇数 n , 一个解总存在。

事实 2 给定 a 和 n 且 $a \perp n$, 解的个数和所有 $b \perp n$ 相同。这由提示的恒等式和事实 1 得出, 因为如果 $x_1^2 - a y_1^2 \equiv b$, 则当 (x_2, y_2) 取遍 $x^2 - a y^2 \equiv 1$ 的所有解时 $(x_1 x_2 - a y_1 y_2, x_1 y_2 + x_2 y_1)$ 取遍 $x^2 - a y^2 \equiv b$ 的所有解。换言之, 当 $x_1^2 - a y_1^2 \perp n$ 时, (x_2, y_2) 是由 (x_1, y_1) 和 (x, y) 惟一确定的。

事实 3 给定整数 (a, s, z) 使得 $z^2 \equiv a \pmod{s}$, 我们可以求整数 (x, y, m, t) 使 $x^2 - a y^2 = m^2 s t$, 其中 $(x, y) \neq (0, 0)$ 且 $t^2 \leq \frac{4}{3} |a|$ 。因为如果 $z^2 = a + ms$, 令 (u, v) 是极小化 $(zu + mv)^2 + |a| u^2$ 的非零整数对。使用 3.3.4 小节的诸方法, 和由习题 3.3.4-9 的 $(zu + mv)^2 + |a| u^2 \leq \left(\frac{4}{3} |a|\right)^{1/2}$, 我们可以有效地求出 (u, v) 。因此 $(zu + mv)^2 - a u^2 = mt$, 其中 $t^2 \leq \frac{4}{3} |a|$ 。提示的恒等式现在解决了 $x^2 - a y^2 = (ms)(mt)$ 。

事实 4 容易解 $x^2 - y^2 \equiv b \pmod{n}$, 因为我们可以令 $x = (b+1)/2, y = (b-1)/2$ 。

事实 5 不难来解 $x^2 + y^2 \equiv b \pmod{n}$, 因为习题 3.3.4-11 中的方法, 当 p

是素数和 $p \bmod 4 = 1$ 时可解 $x^2 + y^2 = p$; 数 $b, b+n, b+2n, \dots$ 之一将是这样一个素数。

现在当 $|a| > 1$ 时来解所述的问题, 我们可进行如下。在 1 和 $n-1$ 之间随机地选择 u 和 v , 然后计算 $w = (u^2 - av^2) \bmod n$ 和 $d = \gcd(w, n)$ 。如果 $1 < d < n$ 或如果 $\gcd(v, n) > 1$, 我们可以减少 n ; 用于证明事实 1 的方法将把对于 n 的因子的解提升成为对 n 本身的解。如果 $d = n$ 和 $v \perp n$, 我们有 $(u/v)^2 \equiv a \pmod{n}$, 因此我们可以把 a 减为 1。否则 $d = 1$; 令 $s = bw \bmod n$ 。由事实 2, 数 s 在与 n 互素的数当中一致分布。如果 $\left(\frac{a}{s}\right) = 1$, 假定 s 是素数, 尝试来求解 $z^2 \equiv a \pmod{s}$ (习题 4.6.2-15)。如果不成功, 则以 u 和 v 的另一个随机选择重新开始。如果成功, 令 $z^2 = a + ms$ 并计算 $d = \gcd(ms, n)$ 。如果 $d > 1$, 像前面那样简化这个问题。否则使用事实 3 来求有 $t^2 \leq \frac{4}{3}|a|$ 的 $x^2 - ay^2 = m^2 st$; 这使得 $(x/m)^2 - a(y/m)^2 \equiv st \pmod{n}$ 。如果 $t = 0$, 把 a 减为 1。否则就递归地应用这个算法求解 $X^2 - tY^2 \equiv a \pmod{n}$ 。(由于 t 比 a 小得多, 因此将仅仅需要 $O(\log \log n)$ 级的递归。) 如果 $\gcd(Y, n) > 1$, 我们可以减少 n 或 a ; 否则 $(X/Y)^2 - a(1/Y)^2 \equiv t \pmod{n}$ 。最后, 提示的恒等式产生对于 $x'^2 - ay'^2 \equiv s$ 的一个解 (参见事实 2), 它转而导致所求的解, 因为 $u^2 - av^2 \equiv s/b$ 。

实际使用中, 在本算法中所做的关于素数的假设证明为真之前, 仅需要 $O(\log n)$ 次随机试验。但一个形式的证明将要求我们假定推广的黎曼假设 [IEEE Trans. IT-33(1987), 702 ~ 709]。Adleman, Estes 和 McCurley [Math. Comp. 48(1987), 17 ~ 28] 建立了一个较慢且更复杂的算法, 它不依赖于任何未被证明的假设。

46. [FOCS 20 (1979), 55 ~ 60] 在对于足够的 n_i 求出 $a^{n_i} \bmod p = \prod_{j=1}^m p_j^{e_j}$ 之后, 对于 $1 \leq j, k \leq m$ 我们可以对整数 x_{ijk}, t_{jk} 求解 $\sum_i x_{ijk} e_{ij} + (p-1)t_{jk} = \delta_{jk}$ (例如, 像在 4.5.2-(23) 中那样), 由此知道对于 $a^{N_j} \bmod p = p_j$ 的解 $N_j = (\sum_i x_{ijk} e_{ij}) \bmod (p-1)$ 。于是如果 $ba^{n'} \bmod p = \prod_{j=1}^m p_j^{e'_j}$, 我们有 $n + n' \equiv \sum_{j=1}^m e'_j N_j \pmod{p-1}$ 。[已经知道一些改进了的算法, 例如 Coppersmith, Odlyzko 和 Schroepel, Algorithmica 1(1986), 1 ~ 15。]

4.6 节

1. $9x^2 + 7x + 7; 5x^3 + 7x^2 + 2x + 6$ 。

2. (a) 为真。(b) 如同在习题 1 中那样, 如果代数系统 S 包含零因子, 即其乘积为 0 的非 0 数, 则为假; 否则为真。(c) 当 $m \neq n$ 时为真, 但是一般地当 $m = n$ 时为假, 因为前导系数可能消失。

3. 假定 $r \leq s$ 。对于 $0 \leq k \leq r$, 极大值是 $m_1 m_2 (k+1)$; 对于 $r \leq k \leq s$, 它是 $m_1 m_2 (r+1)$; 对于 $s \leq k \leq r+s$, 它是 $m_1 m_2 (r+s+1-k)$ 。对于所有 k 有效的最

小上限是 $m_1 m_2 (r+1)$ 。(解答本习题者将知道怎样分解多项式 $x^7 + 2x^6 + 3x^5 + 3x^4 + 3x^3 + 3x^2 + 2x + 1$ 。)

4. 如果诸多项式之一有少于 2^t 个非 0 的系数, 则通过在每两个系数之间放置 $t-1$ 个 0 即可形成乘积, 然后在二进数系中相乘, 最后使用一个逻辑 AND 指令(在大多数二进制计算机上都存在, 参考算法 4.5.4D)把多余的位数化成 0。例如, 如果 $t=3$, 则正文中的乘法将变成 $(1001000001)_2 \times (1000001001)_2 = (1001001011001001001)_2$; 如果我们以常数 $(1001001 \cdots 1001)_2$ AND 此结果, 就得到所希望的答案。类似的技术可用于乘非负系数不太大的多项式。

5. 次数 $\leq 2n$ 的多项式可表示成 $U_1(x)x^n + U_0(x)$, 其中 $\deg(U_1) \leq n$ 和 $\deg(U_0) \leq n$; 而且 $(U_1(x)x^n + U_0(x))(V_1(x)x^n + V_0(x)) = U_1(x)V_1(x)(x^{2n} + x^n) + (U_1(x) + U_0(x))(V_1(x) + V_0(x))x^n + U_0(x)V_0(x)(x^n + 1)$ 。(该式假定算术是在模 2 之下进行的。)于是等式 4.3.3-(3) 和 4.3.3-(5) 成立。

注: S.A. Cook 证明了可以类似的方式推广算法 4.3.3T, 而且 A. Schönhage [Acta Informatica 7 (1977), 395~398] 已经说明怎样仅用 $O(n \log n \log \log n)$ 个二进位运算来乘多项式 mod 2。事实上, 任何环 S 上的多项式都可以仅使用 $O(n \log n \log \log n)$ 个代数运算来乘, 即使当 S 是一个其乘法不必是交换的和结合的代数系统时 [D.G. Cantor 和 E. Kaltofen, Acta Informatica 28 (1991), 693~701]。参见习题 4.6.4-57 和 4.6.4-58。但是这些思想对于“稀疏”多项式(大多数系数为 0)来说不是有用的。

4.6.1 小节

1. $q(x) = 1 \cdot 2^3 x^3 + 0 \cdot 2^2 x^2 - 2 \cdot 2x + 8 = 8x^3 - 4x + 8$; $r(x) = 28x^2 + 4x + 8$ 。

2. 在欧几里得算法期间产生的首一多项式序列有系数 $(1, 5, 6, 6, 1, 6, 3), (1, 2, 5, 2, 2, 4, 5), (1, 5, 6, 2, 3, 4), (1, 3, 4, 6), 0$ 。因此最大公因子是 $x^3 + 3x^2 + 4x + 6$ 。(一个多项式和它的逆的最大公因子, 在是它自己的逆的一个单位倍数的意义下, 总是对称的。)

3. 算法 4.5.2X 的过程是正确的, 其中以 S 上的多项式代替了整数。当此算法终止时, 我们有 $U(x) = u_2(x)$, $V(x) = u_1(x)$ 。设 $m = \deg(u)$, $n = \deg(v)$ 。用归纳法容易证明, 假定 $m \geq n$, 在步骤 X3 之后整个算法执行期间, $\deg(u_3) + \deg(v_1) = n$, $\deg(u_3) + \deg(v_2) = m$ 。因此如果 m 和 n 大于 $d = \deg(\gcd(u, v))$, 则我们有 $\deg(U) < m - d$, $\deg(V) < n - d$; 精确的次数是 $m - d_1$ 和 $n - d_1$, 其中 d_1 是倒数第二个非 0 余式的次数。如果 $d = \min(m, n)$, 比如说 $d = n$, 我们有 $U(x) = 0$ 和 $V(x) = 1$ 。

当 $u(x) = x^m - 1$ 和 $v(x) = x^n - 1$ 时, 恒等式 $(x^m - 1) \bmod (x^n - 1) = x^{m \bmod n} - 1$ 表明, 在计算期间出现的所有多项式都是具有整系数的首一多项式。当 $u(x) = x^{21} - 1$, $v(x) = x^{13} - 1$ 时, 我们有 $V(x) = x^{11} + x^8 + x^6 + x^3 + 1$, $U(x) = -(x^{19} + x^{16} + x^{14} + x^{11} + x^8 + x^5 + x^3 + x)$ 。[也见等式 3.3.3-(29); 它给出了对于 $U(x)$ 和

$V(x)$ 的另外一个公式。还可参见习题 4.3.2-6,其中以 x 代替 2.]

4. 由于商 $q(x)$ 仅仅依赖于 $v(x)$ 和 $u(x)$ 的头 $m-n$ 个系数,因此余式 $r(x) = u(x) - q(x)v(x)$ 是一致分布的且与 $v(x)$ 无关。因此这算法的每一个步骤均可以认为同其它步无关;这个算法比整数上的欧几里得算法有好得多的特性。

$n_1 = n - k$ 的概率是 $p^{1-k}(1-1/p)$, 而且 $t=0$ 的概率是 p^{-n} 。实质上,往下的每一步有相同的特性;因此我们可以看到,任何次数 $n, n_1, \dots, n_t, -\infty$ 的序列以 $(p-1)^t/p^n$ 的概率出现。为求 $f(n_1, \dots, n_t)$ 的平均值,命 S_t 为 $f(n_1, \dots, n_t)$ 在所有有着 t 的给定值的序列 $n > n_1 > \dots > n_t \geq 0$ 上的和。于是平均值是 $\sum_t S_t (p-1)^t/p^n$ 。

命 $f(n_1, \dots, n_t) = t$, 则 $S_t = \binom{n}{t} t$, 所以平均值是 $n(1-1/p)$ 。类似地,如果

$f(n_1, \dots, n_t) = n_1 + \dots + n_t$, 则 $S_t = \binom{n}{2} \binom{n-1}{t-1}$, 同时它的平均值为 $\binom{n}{2} (1-1/p)$ 。

最后如果 $f(n_1, \dots, n_t) = (n-n_1)n_1 + \dots + (n_{t-1}-n_t)n_t$, 则 $S_t = \binom{n+2}{t+2} - (n+1)\binom{n+1}{t+1} + \binom{n+1}{2}\binom{n}{t}$, 而且平均值是 $\binom{n+1}{2} - (n+1)p/(p-1) + (p/(p-1))^2(1-1/p^{n+1})$ 。

(通过置 $S_t = [t=n]$ 得到的对于 $1 \leq j \leq t=n, n_{j+1} = n_j - 1$ 的概率是 $(1-1/p)^n$, 所以当 $p \rightarrow \infty$ 时这个概率趋近于 1。作为一个推论,我们对于正文中的下述断言有了进一步的证据,即算法 C 几乎总求出 $\delta_2 = \delta_3 = \dots = 1$, 因为对于所有的 p , 模 p 下不能通过后面条件的任何多项式也将不能通过前面的条件。)

5. 利用习题 4 提供的公式,且 $f(n_1, \dots, n_t) = [n_t = 0]$, 我们求得如果 $n > 0$, 概率为 $1-1/p$, 如果 $n=0$, 概率等于 1。

6. 假定常数项 $u(0)$ 和 $v(0)$ 非 0, 想像一“自右到左”的除法算法, $u(x) = v(x) \cdot q(x) + x^{m-n}r(x)$, 其中 $\deg(r) < \deg(v)$ 。我们得到类似于算法 4.5.2B 的一个 gcd 算法,它实际上就是把欧几里得算法应用于原来输入的“逆”(参考习题 2), 然后对答案取逆并乘以 x 的适当乘方。

有类似于习题 4.5.2-40 的方法的一个类似的算法。G.H. Norton, *SICOMP* **18** (1989), 608~624; K. Ma 和 J. von zur Gathen, *J. Symbolic Comp.* **9** (1990), 429~455 已经求出这两个算法的平均迭代次数。

7. S 的单位(作为 0 次多项式)。

8. 如果 $u(x) = v(x)w(x)$, 其中 $u(x)$ 有整系数而 $v(x)$ 和 $w(x)$ 有有理系数, 则有非零整数 m 和 n 使得 $m \cdot v(x)$ 和 $n \cdot w(x)$ 有整系数。现在 $u(x)$ 是本原的, 所以等式(4)意味着

$$u(x) = \text{pp}((m \cdot v(x))(n \cdot w(x))) = \pm \text{pp}(m \cdot v(x))\text{pp}(n \cdot w(x))$$

9. 我们可以扩充算法 E 如下: 设 $(u_1(x), u_2(x), u_3, u_4(x))$ 和 $(v_1(x), v_2(x), v_3, v_4(x))$ 是满足关系式 $u_1(x)u(x) + u_2(x)v(x) = u_3u_4(x), v_1(x)$

$u(x) + v_2(x)v(x) = v_3v_4(x)$ 的四元组。扩充的算法以四元组 $(1, 0, \text{cont}(u), \text{pp}(u(x)))$ 和 $(0, 1, \text{cont}(v), \text{pp}(v(x)))$ 开始, 而且以保持上述条件这样一种方式来处理这些四元组, 其中 $u_4(x), v_4(x)$ 跑遍算法 E 中 $u(x)$ 和 $v(x)$ 所跑遍的同样的序列。如果 $au_4(x) = q(x)v_4(x) + br(x)$, 则我们有 $av_3(u_1(x), u_2(x)) - q(x) \cdot u_3(v_1(x), v_2(x)) = (r_1(x), r_2(x))$, 其中 $r_1(x)u(x) + r_2(x)v(x) = bu_3v_3r(x)$, 所以扩充的算法能保持所希望的关系。如果 $u(x)$ 与 $v(x)$ 互素, 则扩充的算法最终能找到 0 次的 $r(x)$, 而且如同所希望的那样, 我们得到 $U(x) = r_2(x), V(x) = r_1(x)$ 。(实际上, 我们将以 $\gcd(\text{cont}(r_1), \text{cont}(r_2))$ 来除 $r_1(x), r_2(x)$ 和 bu_3v_3 。) 反之, 如果这样的 $U(x)$ 和 $V(x)$ 存在, 则 $u(x)$ 和 $v(x)$ 没有公共的素因式, 因为它们是本原的, 因而没有正次数的公因子。

10. 通过逐次地把可约多项式分解为较小次数的多项式, 我们必然得到把任何多项式化成不可约多项式的有限因子分解。内容的分解是惟一的。为了证明至多有一个本原部分的因子分解, 关键之点是证明: 如果 $u(x)$ 是 $v(x)w(x)$ 的一个不可约因子, 但不是不可约多项式 $v(x)$ 的单位倍数, 则 $u(x)$ 是 $w(x)$ 的一个因子。注意由习题 9 的结果 $u(x)$ 是 $v(x)w(x)U(x) = rw(x) - w(x)u(x)V(x)$ 的一个因子, 其中 r 是一个非零常数, 即可证明这一点。

11. 所需要的行名仅为 $A_1, A_0, B_4, B_3, B_2, B_1, B_0, C_1, C_0, D_0$ 。一般来说, 设 $u_{j+2}(x) = 0$; 则对于这个证明所需要的行是 $A_{n_2-n_j}$ 到 $A_0, B_{n_1-n_j}$ 到 $B_0, C_{n_2-n_j}$ 到 $C_0, D_{n_3-n_j}$ 到 D_0 , 等等。

12. 如果 $n_k = 0$, 正文中 (24) 的证明表明, 行列式的值是 $\pm h_k$, 而且这等于 $\pm \ell_k^{n_k-1} / \prod_{1 \leq j < k} \ell_j^{\delta_j-1} (\delta_j-1)$ 。如果多项式有一个正次数的因子, 则我们可以人为地假定多项式 0 的次数为 0 并使用对于 $\ell_k = 0$ 的同一公式。

注: Sylvester 的行列式的值 $R(u, v)$ 称为 u 和 v 的结式, 而且量 $(-1)^{\deg(u)(\deg(u)-1)/2} \ell(u)^{-1} R(u, u')$ 称做 u 的判别式, 其中 u' 是 u 的导数。如果 $u(x)$ 有分解的形式 $a(x-\alpha_1) \cdots (x-\alpha_m)$, 而且如果 $v(x) = b(x-\beta_1) \cdots (x-\beta_n)$, 则结式 $R(u, v)$ 是 $a^n v(\alpha_1) \cdots v(\alpha_m) = (-1)^{mn} b^m u(\beta_1) \cdots u(\beta_n) = a^n b^m \prod_{i=1}^m \prod_{j=1}^n (\alpha_i - \beta_j)$ 。由此得出, 分别作为 $u(y-x), u(y+x), x^m u(y/x)$ 及 $u(yx)$ 的 $v(x)$ 的结式定义的 y 的 mn 次多项式, 它们的根分别是和 $\alpha_i + \beta_j$, 差 $\alpha_i - \beta_j$, 积 $\alpha_i \beta_j$ 和商 α_i / β_j (当 $v(0) \neq 0$ 时)。这一思想已由 R.G.K. Loos 用于构造代数数算术运算的算法 [Computing, Supplement 4 (1982), 173~187]。

如果用 $(b_0 A_i + b_1 A_{i+1} + \cdots + b_{n_2-1-i} A_{n_2-1}) - (a_0 B_i + a_1 B_{i+1} + \cdots + a_{n_2-1-i} B_{n_2-1})$ 代替 Sylvester 矩阵中的每行 A_i , 而后删去 B_{n_2-1} 到 B_0 诸行和最后的 n_2 列, 我们就得到结式的 $n_1 \times n_1$ 的行列式, 而不是原来的 $(n_1 + n_2) \times (n_1 + n_2)$ 行列式。在某些情况下借助于这个行列式, 可以有效地计算结式, 见 CACM 12 (1969), 23~

30, 302 ~ 303。

J. T. Schwartz 已经证明, 当 $n \rightarrow \infty$, 总共用 $O(n(\log n)^2)$ 次算术运算, 有可能计算 n 次多项式的诸结式和 Sturm 序列。[JACM 27 (1980), 701 ~ 717。]

13. 对 j 用归纳法可以证明, 对于 $j \geq 2$, $(u_{j+1}(x), g_{j+1}(x), h_j)$ 的值分别为 $(\ell^{1+p_j} \cdot w(x)u_j(x), \ell^{2+p_j}g_j, \ell^{p_j}h_j)$ 所代替, 其中 $p_j = n_1 + n_2 - 2n_j$ 。[无论这个增长是怎样的, 上限(26)保持正确。]

14. 设 p 是这个环的一个素数, 并设 j, k 是使得 $p^k \setminus v_n = \ell(v)$, $p^j \setminus v_{n-1}$ 的极大值。设 $P = p^k$ 。由算法 R, 我们可以写 $q(x) = a_0 + Pa_1x + \cdots + P^s a_s x^s$, 其中 $s = m - n \geq 2$ 。我们考察 $v(x)q(x)$ 中 x^{n+1}, x^n 及 x^{n-1} 的系数, 即 $Pa_1v_n + P^2a_2v_{n-1} + \cdots, a_0v_n + Pa_1v_{n-1} + \cdots$ 及 $a_0v_{n-1} + Pa_1v_{n-2} + \cdots$, 它的每一个都是 P^3 的倍数。我们从头一个推知 $p^3 \setminus a_1$, 从第二个推知 $p^{\min(k, 2j)} \setminus a_0$, 然后从第三个推知 $P \setminus a_0$ 。因此, $P \setminus r(x)$ 。[如果 m 仅是 $n+1$, 我们可以证明的最好结果是 $p^{\lceil k/2 \rceil}$ 整除 $r(x)$; 例如, 考虑 $u(x) = x^3 + 1, v(x) = 4x^2 + 2x + 1, r(x) = 18$ 。另一方面, 以类似于(21)和(22)的矩阵行列式为基础的论证可以用来证明 $\ell(r)^{\deg(v) - \deg(r) - 1} r(x)$ 总是 $\ell(v)^{(\deg(u) \cdot \deg(v))(\deg(v) - \deg(r) - 1)}$ 的倍数。]

15. 设 $c_{ij} = a_{i1}a_{j1} + \cdots + a_{in}a_{jn}$; 我们可以假设, 对所有 $i, c_{ii} > 0$ 。如果对于某个 $i \neq j, c_{ij} \neq 0$, 则我们可以用 $(c_{i1} - tc_{j1}, \cdots, c_{in} - tc_{jn})$ 代替行 i 和列 i , 其中 $t = c_{ij}/c_{jj}$; 这并不改变行列式 C 的值, 而且它减少了我们希望证明的上限的值, 因为 c_{ii} 为 $c_{ii} - c_{ij}^2/c_{jj}$ 所代替。这样的替换可以系统地对增长的 i 的和 $j < i$ 来进行, 直到对所有的 $i \neq j, c_{ij} = 0$ 为止。[后一算法称为 Gram-Schmidt 正交化过程: 见 Crelle 94 (1883), 41 ~ 73; Math. Annalen 63 (1907), 442。] 于是 $\det(A)^2 = \det(AA^T) = c_{11} \cdots c_{nn}$ 。

16. 在任何一个惟一因子分解整环上的 d 次单变量多项式至多有 d 个根 (参见习题 3.2.1.2-16b)); 所以, 如果 $n = 1$, 显然 $|r(S_1)| \leq d_1$ 。如果 $n > 1$, 我们有 $f(x_1, \cdots, x_n) = g_0(x_2, \cdots, x_n) + x_1 g_1(x_2, \cdots, x_n) + \cdots + x_1^{d_1} g_d(x_2, \cdots, x_n)$, 其中 g_k 对于至少一个 k 为非零。给定 (x_2, \cdots, x_n) , 由此得出, 除非 $g_k(x_2, \cdots, x_n) = 0$, 否则对于 x_1 的至多 d_1 个选择, $f(x_1, \cdots, x_n)$ 为零; 因此 $|r(S_1, \cdots, S_n)| \leq d_1(|S_n| - d_2) \cdots (|S_n| - d_n) + |S_1|(|S_2| \cdots |S_n| - (|S_2| - d_2) \cdots (|S_n| - d_n))$ 。[R. A. DeMillo 和 R. J. Lipton, Inf. Proc. Letters 7 (1978), 193 ~ 195。]

注: 所述上限是最佳可能的, 因为对于多项式 $f(x_1, \cdots, x_n) = \prod |x_j - s_k|$ $s_k \in S_j, 1 \leq k \leq d_j, 1 \leq j \leq n$ 。但是还有另一层意义, 其中上限可以大大改进: 令 $f_1(x_1, \cdots, x_n) = f(x_1, \cdots, x_n)$ 并令 $f_{j+1}(x_{j+1}, \cdots, x_n)$ 是在 $f_j(x_j, \cdots, x_n)$ 中 x_j 的一个乘幂的任何非零的系数。于是我们可以令 d_j 是 f_j 中 x_j 的次数而不是 f 中 x_j 的 (通常大得多) 的次数。例如, 我们可以在多项式 $x_1^3 x_2^9 - 3x_1^2 x_2 + x_2^{100} + 5$ 中令 $d_1 = 3$ 和 $d_2 = 1$ 。当 f 的每一项都有次数 $\leq d$ 时, 这个观察确保 $d_1 + \cdots + d_n \leq d$; 因此在这样的情况下当所有集合 S_j 相等时概率为

$$\frac{|r(S, \dots, S)|}{|S|} \leq 1 - \left(1 - \frac{d_1}{|S|}\right) \cdots \left(1 - \frac{d_n}{|S|}\right) \leq \frac{d_1 + \dots + d_n}{|S|} \leq \frac{d}{|S|}$$

如果此概率 $\leq \frac{1}{2}$, 而且如果对于 50 个随机地选择的向量 (x_1, \dots, x_n) , $f(x_1, \dots, x_n)$ 结果是零, 则 $f(x_1, \dots, x_n)$ 至少以 $1 - 2^{-50}$ 的概率恒等于零。

而且, 如果 $f_j(x_j, \dots, x_n)$ 有 $x_j^{e_j} f_{j+1}(x_{j+1}, \dots, x_n)$ 的特殊形式且 $e_j > 0$, 我们可以取 $d_j = 1$ 。因为当 $f_{j+1}(x_{j+1}, \dots, x_n) \neq 0$ 时, x_j 必为 0。因此一个仅有 m 个非零项的稀疏多项式对于 j 的至少 $n - \lg m$ 个值有 $d_j \leq 1$ 。

应用这个不等式到 gcd 的计算以及对于稀疏多变量多项式的其它运算是由 R. Zippel 引入的, *Lecture Notes in Comp. Sci.* 72 (1979), 216 ~ 226。J. T. Schwartz [*JACM* 27 (1980), 701 ~ 717] 给出了进一步的扩充, 包括一个借助于模算术来避免大数的方法: 如果 f 的系数是整数, 如果 P 是全部大于等于 q 的素数的集合, 而且如果每当每个 $x_j \in S_j$, $|f(x_1, \dots, x_n)| \leq L$, 则对于 $p \in P$, $f(x_1, \dots, x_n) \equiv 0 \pmod{p}$ 的解的个数至多为

$$|S_1| \cdots |S_n| |P| - (|S_1| - d_1) \cdots (|S_n| - d_n) (|P| - \log_q L)$$

17. a) 为方便起见, 让我们仅对 $A = \{a, b\}$ 来描述算法。这些假设意味着, $\deg(Q_1 U) = \deg(Q_2 V) \geq 0$, 且 $\deg(Q_1) \leq \deg(Q_2)$ 。如果 $\deg(Q_1) = 0$, 则 Q_1 是一非 0 有理数, 所以我们置 $Q = Q_2/Q_1$ 。否则设 $Q_1 = aQ_{11} + bQ_{12} + r_1$, $Q_2 = aQ_{21} + bQ_{22} + r_2$, 其中 r_1 和 r_2 是有理数; 因此得出

$$Q_1 U - Q_2 V = a(Q_{11} U - Q_{21} V) + b(Q_{12} U - Q_{22} V) + r_1 U - r_2 V$$

我们必然有 $\deg(Q_{11}) = \deg(Q_1) - 1$ 或 $\deg(Q_{12}) = \deg(Q_1) - 1$ 。在前一情况下, 通过考虑以 a 开始的最高次数的项, $\deg(Q_{11} U - Q_{21} V) < \deg(Q_{11} U)$; 所以我们可以以 Q_{11} 代替 Q_1 , 以 Q_{21} 代替 Q_2 , 并重复这一过程。在后一情况下, 类似地, 我们可以用 (Q_{12}, Q_{22}) 来代替 (Q_1, Q_2) 并重复这一过程。

b) 我们可以假设 $\deg(U) \geq \deg(V)$ 。如果 $\deg(R) \geq \deg(V)$, 注意, $Q_1 U - Q_2 V = Q_1 R - (Q_2 - Q_1 Q) V$ 有小于 $\deg(V) \leq \deg(Q_1 R)$ 的次数, 因此我们可以以 R 代替 U 并重复这一过程; 我们得到 $R = Q' V + R'$, $U = (Q + Q') V + R'$, 其中 $\deg(R') < \deg(R)$, 所以最终得到一个解。

c) b) 的算法给出 $V_1 = UV_2 + R$, $\deg(R) < \deg(V_2)$; 由齐次性, $R = 0$ 且 U 是齐次的。

d) 我们可以假设 $\deg(V) \leq \deg(U)$ 。如果 $\deg(V) = 0$, 则置 $W \leftarrow U$; 否则使用 c) 来求 $U = QV$, 使得 $QVV = VQV$, $(QV - VQ)V = 0$ 。这意味着 $QV = VQ$, 所以我们可以置 $U \leftarrow V$, $V \leftarrow Q$, 并重复这一过程。

有关这一课题进一步的细节, 见 P. M. Cohn, *Proc. Cambridge Phil. Soc.* 57 (1961), 18 ~ 30。表征所有使得 $UV = VU$ 的串多项式的更为困难的问题, 已由 G. M. Bergman 所解决 [*Ph.D. thesis, Harvard University, 1967*]。

18. [P. M. Cohn, *Transaction of the Amer. Math. Soc.* **109** (1963), 332~356.]

C1. 置 $u_1 \leftarrow U_1, u_2 \leftarrow U_2, v_1 \leftarrow V_1, v_2 \leftarrow V_2, z_1 \leftarrow z'_2 \leftarrow w_1 \leftarrow w'_2 \leftarrow 1, z'_1 \leftarrow z_2 \leftarrow w'_1 \leftarrow w_2 \leftarrow 0, n \leftarrow 0$ 。

C2. (这时习题中给出的恒等式成立, 并且 $u_1 v_1 = u_2 v_2; v_2 = 0$ 当且仅当 $u_1 = 0$ 。) 如果 $v_2 = 0$, 则这算法以 $\gcd(V_1, V_2) = v_1, \text{lcm}(V_1, V_2) = z'_1 V_1 = z'_2 V_2$ 终止。(由对称性, 我们还有 $\gcd(U_1, U_2) = u_2$, 及 $\text{lcm}(U_1, U_2) = U_1 w_1 = -U_2 w_2$ 。)

C3. 求 Q 和 R , 使得 $v_1 = Qv_2 + R$, 其中 $\deg(R) < \deg(v_2)$ 。(我们有 $u_1(Qv_2 + R) = u_2 v_2$, 所以 $u_1 R = (u_2 - u_1 Q)v_2 = R'v_2$ 。)

C4. 置 $(w_1, w_2, w'_1, w'_2, z_1, z_2, z'_1, z'_2, u_1, u_2, v_1, v_2) \leftarrow (w'_1 - w_1 Q, w'_2 - w_2 Q, w_1, w_2, z'_1, z'_2, z_1 - Qz'_1, z_2 - Qz'_2, u_2 - u_1 Q, u_1, v_2, v_1 - Qv_2)$ 和 $n \leftarrow n + 1$, 转回 C2。 |

欧几里得算法的这一推广概括了以前所有推广中的大多数特性, 所以它提供了对已经考虑过的一些特殊情况的新的认识。为了证明它是正确的, 首先注意, $\deg(v_2)$ 在步骤 C4 时减小, 所以此算法肯定要终止。在算法结尾时, v_1 是 V_1 和 V_2 的公共右因子, 因为 $w_1 v_1 = (-1)^n V_1$ 和 $-w_2 v_1 = (-1)^n V_2$; 而且如果 d 是 V_1 和 V_2 的任何公共右因子, 则它必是 $z_1 V_1 + z_2 V_2 = v_1$ 的右因子。因此 $v_1 = \gcd(V_1, V_2)$ 。又若 m 是 V_1 和 V_2 的任何公共左倍元, 则我们可以不失一般性地假设 $m = U_1 V_1 = U_2 V_2$, 因为 Q 的值的序列不依赖于 U_1 和 U_2 。因此 $m = (-1)^n (-u_2 z'_1) \cdot V_1 = (-1)^n (u_2 z'_2) V_2$ 是 $z'_1 V_1$ 的倍元。

在实际使用中, 如果我们仅仅要计算 $\gcd(V_1, V_2)$, 则可以免去计算 $n, w_1, w_2, w'_1, w'_2, z_1, z_2, z'_1, z'_2$ 。这些附加的量之所以加到此算法中来, 主要是为了使它的有效性更易于建立。

注: 串多项式的非平凡的因子分解, 例如由这个习题给出的例子, 可以从一些矩阵恒等式得出, 例如有

$$\begin{pmatrix} a & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} b & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -c \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -b \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -a \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

因为甚至当乘法不可交换时, 这些恒等式都成立。例如

$$(abc + a + c)(1 + ba) = (ab + 1)(cba + a + c)$$

(把这同 4.5.3 小节的“连项多项式”进行比较。)

19. [参考 Eugène Cahen, *Théorie des Nombres* **1** (Paris: 1914), 336~338.] 如果存在这样一个算法, 则由习题 18 的论证, D 是一个 \gcd 。现在让我们把 A 和 B 当做 $2n \times n$ 阶矩阵 C , 其头 n 行是 A 的那些行, 而其余 n 行是 B 的那些行。类似地, 把 P 和 Q 组合成一个 $2n \times n$ 阶矩阵 R ; 把 X 和 Y 组合成一个 $n \times 2n$ 阶矩阵 Z 。所求的条件现在归结为两个等式 $C = RD, D = ZC$ 。如果我们可以找出其行列式为 ± 1 的 $2n \times 2n$ 阶整数矩阵 U , 使得 $U^{-1}C$ 的最后 n 行全为 0, 则 $R = (U$ 的头 n

列), $D = (U^{-1}C$ 的头 n 行), $Z = (U^{-1}$ 的头 n 行) 解决了所求的条件。因此, 例如可以使用下列的算法(取 $m = 2n$):

算法 T(三角化) 设 C 是 $m \times n$ 阶整数矩阵。此算法求 $m \times m$ 阶整数矩阵 U 和 V , 使得 $UV = I$, 且 VC 是上三角。(这意味着如果 $i > j$, 则 VC 的 i 行 j 列元素为 0。)

T1. [初始化] 置 $U \leftarrow V \leftarrow I$, 即 $m \times m$ 阶单位矩阵; 并置 $T \leftarrow C$ 。(在整个算法中, 我们有 $T = VC, UV = I$ 。)

T2. [对 j 进行迭代] 对于 $j = 1, 2, \dots, \min(m, n)$, 执行步骤 T3, 然后终止此算法。

T3. [使 j 列为零] 实施下列动作 0 次或多次, 直到对所有 $i > j$, T_{ij} 为 0 为止: 设 T_{kj} 是 $\{T_{ij}, T_{(i+1)j}, \dots, T_{mj}\}$ 中有最小绝对值的非 0 元素。交换 T 和 V 的 k 行和 j 行; 交换 U 的 k 列和 j 列。然后, 在矩阵 T 和 V 中从第 i 行减去 $\lfloor T_{ij}/T_{jj} \rfloor$ 乘第 j 行, 而且对于 $j < i \leq m$, 在矩阵 U 中加 i 列的同样倍数到第 j 列。 ■

对于所描述的例子此算法产生 $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 4 & 5 \\ 2 & 3 \end{pmatrix}.$

$\begin{pmatrix} 1 & 2 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 2 & -2 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix}$ 。(实际上, 在此具体情况下, 任何具有行列式 ± 1 的矩阵都将是一个 gcd。)

20. 通过以小的数 ϵ 代替 p^m , 来考虑习题 4.6.2-22 的构造可能有帮助。

21. 为获得一个上限, 我们可以假定, 当 $m - n \leq 1$ 时, 算法 R 才被使用。其次, 诸系数以 (26) 为限且 $m = n$ 。[事实上所述的公式是在实践中所观察到的执行时间, 而不仅仅是上限。关于更详细的信息, 参见 G. E. Collins, *Proc. 1968 Summer Inst. On Symbolic Mathematical Computation*, Robert G. Tobey 编 (IBM Federal Systems Center; 1969 年 6 月), 195~231。]

22. 一个符号序列不可能含两个连续的 0, 因为 $u_{k+1}(x)$ 是 (29) 中非 0 的常数, 而且我们不能有 “+, 0, +” 或 “-, 0, -” 作为子序列。当 $b = a$ 时公式 $V(u, a) - V(u, b)$ 显然是正确的, 所以我们只须当 b 增加时检验它。多项式 $u_j(x)$ 有有限多个根, 而且仅当 b 遇到或越过这样的根时 $V(u, b)$ 才改变。命 x 是某个 (可能有好几个) u_j 的一个根。当 b 从 $x - \epsilon$ 增加到 x 时, 如果 $j > 0$, 这个 j 附近的符号序列从 “+, ±, -” 进行到 “+, 0, -”, 或者从 “-, ±, +” 进行到 “-, 0, +”; 如果 $j = 0$, 它从 “+, -” 进行到 “0, -”, 或从 “-, +” 进行到 “0, +” (由于 $u'(x)$ 是导数, 当 $u(x)$ 减少时 $u'(x)$ 为负)。于是 V 中的纯变化是 $-\delta_{j,0}$ 。当 b 从 x 增加到 $x + \epsilon$ 时, 类似的论证表明 V 保持不变。

[L. E. Heindel, *JACM* 18 (1971), 533~548 已经应用这些思想去构造用于分离一个给定的多项式 $u(x)$ 的实零点的算法, 它在时间上以 $\deg(u)$ 的一个多项式和

$\log N$ 为界, 其中所有系数 y_j 是满足 $|u_j| \leq N$ 的整数, 而且所有操作都保证是精确的.]

23. 如果 v 有 $n-1$ 个实根出现于 u 的 n 个实根之间, 则(通过考虑符号的变化) $u(x) \bmod v(x)$ 有 $n-2$ 个实根位于 v 的 $n-1$ 个根之间。

24. 首先证明 $h_j = g_j^{\delta_j-1} g_{j-1}^{\delta_{j-1}-(1-\delta_{j-1})} \cdots g_2^{\delta_1(1-\delta_2)\cdots(1-\delta_{j-1})}$. 然后证明(18)左边的 g_2 的指数有形式 $\delta_2 + \delta_1 x$, 其中 $x = \delta_2 + \cdots + \delta_{j-1} + 1 - \delta_2(\delta_3 + \cdots + \delta_{j-1} + 1) - \delta_3(1-\delta_2)(\delta_4 + \cdots + \delta_{j-1} + 1) - \cdots - \delta_{j-1}(1-\delta_2)\cdots(1-\delta_{j-2})(1)$. 但是 $x=1$, 因为可以看出它是与 δ_{j-1} 无关的, 而且我们可以置 $\delta_{j-1}=0$ 等等。一个类似的推导对 g_3, g_4, \cdots 适用而且可应用于(23)。

25. $u_j(x)$ 的每个系数都可表达为一个行列式, 其中一个列仅含 $\ell(u), \ell(v)$ 和一些零。为了利用这一事实, 修改算法 C 如下: 在步骤 C1 处置 $g \leftarrow \gcd(\ell(u), \ell(v))$ 和 $h \leftarrow 0$ 。在步骤 C3 处, 如果 $h=0$, 则置 $u(x) \leftarrow v(x), v(x) \leftarrow r(x)/g, h \leftarrow \ell(u)^{\delta}/g, g \leftarrow \ell(u)$, 并返回 C2; 否则如同在未修改的算法中那样进行。这个新的初始化的效果仅仅是对于 $j \geq 3$, 用 $u_j(x)/\gcd(\ell(u), \ell(v))$ 代替 $u_j(x)$; 于是 ℓ^{2j-4} 在(28)中将变成 ℓ^{2j-5} 。

26. 事实上, 还可推出更多的东西。注意, 对于 $n \geq -1$, 习题 3 中的算法计算 $\pm p_n(x)$ 和 $\mp q_n(x)$ 。令 $e_n = \deg(q_n)$ 和 $d_n = \deg(p_n u - q_n v)$; 我们在习题 3 中发现, 对于 $n \geq 0, d_{n-1} + e_n = \deg(u)$ 。我们将证明条件 $\deg(q) < e_n$ 和 $\deg(pu - qv) < d_{n-2}$ 意味着 $p(x) = c(x)p_{n-1}(x)$ 和 $q(x) = c(x)q_{n-1}(x)$: 给定这样的 p 和 q , 我们能求出 $c(x)$ 和 $d(x)$, 使得 $p(x) = c(x)p_{n-1}(x) + d(x)p_n(x)$ 和 $q(x) = c(x)q_{n-1}(x) + d(x)q_n(x)$, 因为 $p_{n-1}(x)q_n(x) - p_n(x)q_{n-1}(x) = \pm 1$ 。因此 $pu - qv = c(p_{n-1}u - q_{n-1}v) + d(p_nu - q_nv)$ 。如果 $d(x) \neq 0$, 我们必定有 $\deg(c) + e_{n-1} = \deg(d) + e_n$, 因为 $\deg(q) < \deg(q_n)$; 由此得出 $\deg(c) + d_{n-1} > \deg(d) + d_n$, 因为如果 $d_n = -\infty$ 这确实是对的, 否则我们有 $d_{n-1} + e_n = d_n + e_{n+1} > d_n + e_{n-1}$ 。因此 $\deg(pu - qv) = \deg(c) + d_{n-1}$ 。但是我们已假定 $\deg(pu - qv) < d_{n-2} = d_{n-1} + e_n - e_{n-1}$; 所以 $\deg(c) < e_n - e_{n-1}$ 和 $\deg(d) < 0$, 矛盾。

[这个结果实质上属于 L. Kronecker, *Monatsberichte Königl. Preuß. Akad. Wiss.* (Berlin: 1881), 535~600。它蕴涵了下列的定理: “设 $u(x)$ 和 $v(x)$ 是在一个域上的互素多项式并设 $d \leq \deg(v) < \deg(u)$ 。如果 $q(x)$ 是最小次的一个多项式, 使得存在多项式 $p(x)$ 和 $r(x)$ 且 $p(x)u(x) - q(x)v(x) = r(x)$ 且 $\deg(r) = d$, 则对某个 $n, p(x)/q_n(x) = p_n(x)/q_n(x)$ 。”因为如果 $d_{n-2} > d \geq d_{n-1}$, 则有解 $q(x)$ 使得 $\deg(q) = e_{n-1} + d - d_{n-1} < e_n$, 而且我们已经证明这样低次数的所有解都有所述的性质。]

27. 答案 4.3.1-40 的思想适用, 但以比较简单的方式, 因为多项式算术是无进位的; 自右至左的除法使用 4.7-(3)。或者, 对于大的 n 值, 倒过来使用习题 4.6.4-57, 我们可以除系数的傅里叶变换。

4.6.2 小节

1. 对于 $k \leq n$ 个不同根的任何一选择, 有 p^{n-k} 个至少有这些根一次的首一多项式。因此, 由 (1.3.3 小节的) 容斥原理, 没有线性因子的多项式的个数是 $\sum_{k \leq n} \binom{p}{k} p^{n-k} (-1)^k$, 而且它交替小于等于和大于等于这个级数的部分和。所述的限对应于 $k \leq 2$ 和 $k \leq 3$ 。当 $n \geq p$ 时至少一个线性因子的概率是 $1 - (1 - 1/p)^p$ 。线性因子的平均个数是 p 乘 x 整除 $u(x)$ 的次数, 所以它是 $1 + p^{-1} + \cdots + p^{1-n} = \frac{p}{p-1}(1 - p^{-n})$ 。

[以类似的方式, 我们求得, 有次数为 2 的一个不可约因子的概率是 $\sum_{k \leq n/2} \binom{p(p-1)/2}{k} (-1)^k p^{-2k}$; 当 $n \geq 2$ 时这个概率介于 $\frac{3}{8} - \frac{1}{4}p^{-1}$ 和 $\frac{1}{2} - \frac{1}{2}p^{-1}$ 之间, 而当 $n \rightarrow \infty$ 时它趋向于 $1 - e^{-1/2}(1 + \frac{1}{2}p^{-1}) + O(p^{-2})$ 。这样的因子的平均个数是 $\frac{1}{2} - \frac{1}{2}p^{-2\lfloor n/2 \rfloor}$ 。]

注: 设 $u(x)$ 是带有整系数的一个固定的多项式。Peter Weinberger 已经发现, 如果 $u(x)$ 是在整数上不可约的, 则当 $p \rightarrow \infty$ 时, $u(x)$ 的模 p 线性因子的平均个数趋于 1, 因为 $u(x)$ 的 Galois 群是传递的而且在任何传递排列群中任何随机选定元素的 1-循环的平均个数是 1。因此, $u(x)$ 的模 p 线性因子的平均个数是当 $p \rightarrow \infty$ 时在整数上 $u(x)$ 不可约因子的个数。[参见习题 37 的答案中的说明, 以及 Proc. Symp. Pure. Math. 24 (Amer. Math. Soc., 1972), 321~332。]

2. (a) 我们知道 $u(x)$ 有一个不可约多项式乘积的表示; 而且这些多项式的前导系数必须是 1, 因为它们整除 $u(x)$ 的前导系数。因此我们可以假定 $u(x)$ 有首一不可约多项式 $p_1(x)^{e_1} \cdots p_r(x)^{e_r}$ 的乘积的表示, 其中 $p_1(x), \dots, p_r(x)$ 是不同的。除了因子的次序外, 这个表示是惟一的。所以关于 $u(x), v(x), w(x)$ 的条件被满足当且仅当

$$v(x) = p_1(x)^{\lfloor e_1/2 \rfloor} \cdots p_r(x)^{\lfloor e_r/2 \rfloor}, \quad w(x) = p_1(x)^{e_1 \bmod 2} \cdots p_r(x)^{e_r \bmod 2}$$

(b) 次数为 n 的首一多项式的个数的生成函数是 $1 + pz + p^2z^2 + \cdots = 1/(1 - pz)$ 。次数为 n 有形式 $v(x)^2$ ($v(x)$ 为首一的) 的多项式个数的生成函数是 $1 + pz^2 + p^2z^4 + \cdots = 1/(1 - pz^2)$ 。如果次数为 n 的首一无平方多项式个数的生成函数是 $g(z)$, 则由 (a), 我们必有 $1/(1 - pz) = g(z)/(1 - pz^2)$ 。因此 $g(z) = (1 - pz^2)/(1 - pz) = 1 + pz + (p^2 - p)z^2 + (p^3 - p^2)z^3 - \cdots$ 。对于 $n \geq 2$ 的答案是 $p^n - p^{n-1}$ 。[奇怪, 这证明 $u(x) \perp u'(x)$ 的概率是 $1 - 1/p$; 由习题 4.6.1-5, 当 $u(x)$ 和 $v(x)$ 无关时, 它和 $u(x) \perp v(x)$ 的概率相同。]

注: 由一个类似的论证, 每个 $u(x)$ 有惟一的表示 $v(x)w(x)^r$, 其中 $v(x)$ 是不能由任何不可约多项式的第 r 次幂整除的; 对于 $n \geq r$, 这样的首一多项式 $v(x)$ 的

个数是 $p^n - p^{n-r+1}$ 。

3. 命 $v(x) = u_1(x) \cdots u_r(x)$ 。由定理 4.3.2C 的论证, 至多有一个这样的 $v(x)$ 。如果对于每个 j 我们能够以 $w_j(x) = 1, w_k(x) = 0, j \neq k$ 解系统, 则至少有一个 $v(x)$ 。后者的一个解是 $v_1(x) \prod_{k \neq j} u_k(x)$, 其中 $v_1(x)$ 和 $v_2(x)$ 通过欧几里得算法的推广(习题 4.6.1-3), 可以发现满足

$$v_1(x) \prod_{k \neq j} u_k(x) + v_2(x) u_j(x) = 1, \deg(v_1) < \deg(u_j)$$

当 $\deg(v) < 2$ 时在整数上我们不能使 $v(x) \equiv 1 \pmod{x}$ 和 $v(x) \equiv 0 \pmod{x-2}$ 。

4. 由惟一因子分解, 我们有 $(1-pz)^{-1} = \prod_{n \geq 1} (1-z^n)^{-a_{np}}$; 在取对数之后, 这可以重写成

$$\ln(1/(1-pz)) = \sum_{k, j \geq 1} a_{kp} z^{kj} / j = \sum_{j \geq 1} G_p(z^j) / j$$

所述的恒等式现在产生答案 $G_p(z) = \sum_{m \geq 1} \mu(m) m^{-1} \ln(1/(1-pz^m))$, 由此我们得到 $a_{np} = \sum_{d \mid n} \mu(n/d) p^d / n$; 于是 $\lim_{p \rightarrow \infty} a_{np} / p^n = 1/n$ 。

为证明所述的恒等式, 注意

$$\sum_{n, j \geq 1} \mu(n) g(z^n) n^{-1} j^{-1} = \sum_{m \geq 1} g(z^m) m^{-1} \sum_{n \mid m} \mu(n) = g(z)$$

[数 a_{np} 首先是由高斯发现的, 参见他的 *Werke* 2, 219~222。]

5. 设 a_{npr} 是在模 p 之下恰有 r 个不可约因子的 n 次首一多项式的个数, 则

$$\begin{aligned} \mathcal{G}_p(z, w) &= \sum_{n, r \geq 0} a_{npr} z^n w^r = \exp\left(\sum_{k \geq 1} G_p(z^k) w^k / k\right) = \\ &= \exp\left(\sum_{m \geq 1} a_{mw} \ln(1/(1-pz^{-m}))\right) \end{aligned}$$

参见等式 1.2.9-(38)。我们有

$$\begin{aligned} \sum_{n \geq 0} A_{np} z^n &= d \mathcal{G}_p(z/p, w) / dw \Big|_{w=1} = \\ &= \left(\sum_{k \geq 1} G_p(z^k/p^k)\right) \mathcal{G}_p(z/p, 1) = \\ &= \left(\sum_{n \geq 1} \ln(1/(1-p^{1-n} z^n)) \varphi(n)/n\right) / (1-z) \end{aligned}$$

因此对于 $n \geq 2$, $A_{np} = H_n + 1/2 p + O(p^{-2})$ 。 2^n 的平均值是 $[z^n] \mathcal{G}_p(z/p, 2) = n + 1 + (n-1)/p + O(np^{-2})$ 。(方差是 n^3 阶的, 但可置 $w=4$ 。)

6. 对于 $0 \leq s < p$, 由费马定理, $x-s$ 是 $x^p - x \pmod{p}$ 的一个因子, 所以 $x^p - x$ 是 $\text{lcm}(x-0, x-1, \dots, x-(p-1)) = x^{\frac{p-1}{2}}$ 的倍数。[注: 因此除了当 $k=1$ 或 $k=p$ 时外, 斯特林数 $\begin{bmatrix} p \\ k \end{bmatrix}$ 是 p 的倍数。等式 1.2.6-(45) 表明对于另一类斯特林数

$\begin{Bmatrix} p \\ k \end{Bmatrix}$ 同样的命题成立。]

7. 右边的因子互素, 而且每一个是 $u(x)$ 的因子, 所以它们的乘积整除 $u(x)$ 。

另一方面, $u(x)$ 整除

$$v(x)^p - v(x) = \prod_{0 \leq s < p} (v(x) - s)$$

所以由习题 4.5.2-2 它整除右边的式子。

8. 向量(18)是仅有的输出, 其第 k 个分量非零。

9. 例如, 由 $x \leftarrow 1$ 和 $y \leftarrow 1$ 开始; 然后重复地置 $R[x] \leftarrow y, x \leftarrow 2x \bmod 101, y \leftarrow 51y \bmod 101$, 共 100 次。

10. 下列矩阵 $Q-I$ 有由两个向量 $v^{[1]} = (1, 0, 0, 0, 0, 0, 0, 0)$, $v^{[2]} = (0, 1, 1, 0, 0, 1, 1, 1)$ 生成的零空间。因子分解是 $(x^6 + x^5 + x^4 + x + 1)(x^2 + x + 1)$ 。

$$\begin{array}{cc} p=2 & p=5 \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 2 & 0 & 4 & 3 & 4 & 0 \\ 0 & 1 & 4 & 4 & 4 & 2 & 1 & 0 \\ 2 & 2 & 2 & 3 & 4 & 3 & 2 & 0 \\ 0 & 0 & 4 & 0 & 1 & 3 & 2 & 0 \\ 3 & 0 & 2 & 1 & 4 & 2 & 1 & 0 \end{pmatrix} \end{array}$$

11. 删去平凡因子 x , 上边的 $Q-I$ 有一个由 $(1, 0, 0, 0, 0, 0, 0, 0)$ 和 $(0, 3, 1, 4, 1, 2, 1, 1)$ 生成的零空间。因子分解是 $x(x^2 + 3x + 4)(x^5 + 2x^4 + x^3 + 4x^2 + x + 3)$ 。

12. 如果 $p=2$, 则 $(x+1)^4 = x^4 + 1$ 。如果 $p=8k+1$, 则 $Q-I$ 是零矩阵, 所以有四个因子。对于 p 的其它值我们有

$$Q-I = \begin{array}{ccc} p=8k+3 & p=8k+5 & p=8k+7 \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -2 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & -2 & 0 \\ 0 & -1 & 0 & -1 \end{pmatrix} \end{array}$$

这里 $Q-I$ 有阶 2, 所以有 $4-2=2$ 个因子。[但是容易证明 x^4+1 在整数上是不可约的, 因为它没有线性因子, 而且由习题 20, 在任何次数为 2 的因子中 x 的系数的绝对值必定小于等于 2。(也见习题 32, 因为 $x^4+1 = \Psi_8(x)$ 。)对于所有 $k \geq 2$, H.P.F. Swinnerton-Dyer 已经指出了一些次数为 2^k 的多项式, 它们在整数上是不可约的, 但是在模每个素数之下完全分解成线性因子和二次因子。对于次数 8, 他的例子是 $x^8 - 16x^6 + 88x^4 + 192x^2 + 144$, 有根 $\pm\sqrt{2} \pm \sqrt{3} \pm i$ [见 *Math. Comp.* 24 (1970), 733~734]。按照习题 37 中所引的 Frobenius 的定理, 任何其 Galois 群不含 n 周期的 n 次不可约多项式含有模几乎所有素数的因子。]

13. 情况 $p=8k+1: (x + (1 + \sqrt{-1})/\sqrt{2})(x + (1 - \sqrt{-1})/\sqrt{2})(x - (1 + \sqrt{-1})/\sqrt{2})(x - (1 - \sqrt{-1})/\sqrt{2})$ 。情况 $p=8k+3: (x^2 + \sqrt{-2}x - 1)(x^2 - \sqrt{-2}x - 1)$ 。情况 $p=8k+5: (x^2 + \sqrt{-1})(x^2 - \sqrt{-1})$ 。情况 $p=8k+7: (x^2 + \sqrt{2}x + 1)(x^2 - \sqrt{2}x + 1)$ 。

1) $(x^2 - \sqrt{2}x + 1)$ 。在实数域上对于 $p = 8k + 7$ 的因子分解也成立。

14. 可以把算法 N 修改成求 w 的系数: 设 A 是 $(r+1) \times n$ 的矩阵, 对于 $0 \leq k \leq r$, 它的第 k 行包含 $v(x)^k \bmod u(x)$ 的系数。应用算法 N 的方法直到在步骤 N3 中找出第一个相关性为止; 然后算法以 $w(x) = v_0 + v_1x + \cdots + v_kx^k$ 结束, 其中 v_i 在 (18) 中定义。这时 $2 \leq k \leq r$, 不必预先知道 r , 因为在生成 A 的每行之后我们可以检验相关性。

15. 我们可以假定 $u \neq 0$ 和 p 为奇数。应用于多项式 $x^2 - u$ 的 Berlekamp 方法告诉我们, 一个平方根存在当且仅当 $Q - I = O$ 当且仅当 $u^{(p-1)/2} \bmod p = 1$; 但这是我们已经知道了的。(20) 中 Cantor 和 Zassenhaus 的方法, 或者对于 $d = 1$ 在 (21) 中更好的方法告诉我们, 当 s 是随机选择的时, $\gcd(x^2 - u, (x + s)^{(p-1)/2} - 1)$ 将以 $> 1/2$ 的概率是一个非平凡的因子。在实际使用中, 对 s 的顺序选择看起来就和随机选择一样好, 所以我们得到下列算法: “计算 $\gcd(x^2 - u, x^{(p-1)/2} - 1)$, $\gcd(x^2 - u, (x + 1)^{(p-1)/2} - 1)$, $\gcd(x^2 - u, (x + 2)^{(p-1)/2} - 1)$, \cdots , 直到找到 \gcd 有 $x + v$ 的形式的头一种情况为止。然后 $\sqrt{u} = \pm v$ 。”对于很大的 p , 预期的运行时间是 $O(\log p)^3$ 。

更仔细的观察表明, 这个算法的头一步成功当且仅当 $p \bmod 4 = 3$ 时。因为如果 $p = 2q + 1$, 其中 q 是奇数, 我们有 $x^q \bmod (x^2 - u) = u^{(q-1)/2}x$ 和 $\gcd(x^2 - u, x^q - 1) = x - u^{(q+1)/2}$, 因为 $u^q \equiv 1 \pmod{p}$ 。事实上, 我们看到, 每当 $p \bmod 4 = 3$ 时公式 $\sqrt{u} = \pm u^{(p+1)/4} \bmod p$ 直接给出平方根。

但当 $p \bmod 4 = 1$ 时, 我们将有 $x^{(p-1)/2} \bmod (x^2 - u) = u^{(p-1)/4}$, 而且 \gcd 将是 1。因此仅当 $p \bmod 4 = 1$ 时, 才应当使用上面的算法, 而且随后应省略头一个 \gcd 。

当 $p \bmod 8 = 5$ 时一个工作得很漂亮的直接方法是在 20 世纪 90 年代由 A. O. L. Atkin 发现的。在该情况下它是基于 $2^{(p-1)/2} \equiv -1$ 这一事实得出的: 置 $v \leftarrow (2u)^{(p-5)/8} \bmod p$ 和 $i \leftarrow (2uv^2) \bmod p$; 然后 $\sqrt{u} = \pm (uv(i-1)) \bmod p$, 而且我们还有 $\sqrt{-1} = \pm i$ 。[Computational Perspectives on Number Theory (Cambridge, Mass.: International Press, 1998), 1~11; 也见 H. C. Pocklington, Proc. Camb. Phil. Soc. 19 (1917), 57~59.]

当 $p \bmod 8 = 1$ 时, 一个反复试验的方法似乎是必要的。由 Daniel Shanks 随之给出的过程在这样一些情况下通常比所有其它已知的方法都要好: 假设 $p = 2^e q + 1$, 其中 $e \geq 3$ 。

S1. 在 $1 < x < p$ 的范围中随机选择 x , 并置 $z = x^q \bmod p$ 。如果 $z^{2^{e-1}} \bmod p = 1$, 则重复这一步骤。(重复的平均次数将小于 2。在步骤 S2 和 S3 中将不需要随机数。实践中通过试验小的奇素数 x 我们可以节省时间, 而且当 $p^{(e-1)/2} \bmod x = x - 1$ 时, 以 $z = x^q \bmod p$ 停止; 参见习题 1.2.4-47。)

S2. 置 $y \leftarrow z$, $r \leftarrow e$, $x \leftarrow u^{(q-1)/2} \bmod p$, $v \leftarrow ux \bmod p$, $w \leftarrow ux^2 \bmod p$ 。

S3. 如果 $w = 1$, 则停止; v 是答案。否则求使得 $w^{2^k} \bmod p$ 等于 1 的最小的 k 。

如果 $k = r$, 则停止 (无答案); 否则置 $(y, r, v, w) \leftarrow (y^{2^{r-k}}, k, vy^{2^{r-k}-1}, wy^{2^{r-k}})$, 并重复步骤 S3。 ▮

这个算法的正确性来自于不变同余式 $uw \equiv v^2, y^{2^{r-1}} \equiv -1, w^{2^{r-1}} \equiv 1 \pmod{p}$ 。当 $w \neq 1$ 时, 步骤 S3 执行 $r+2$ 次模 p 乘法; 因此在该步中极大的乘法次数小于 $\binom{e+3}{2}$, 而且平均次数小于 $\frac{1}{2} \binom{e+4}{2}$ 。因此对于步骤 S1 和 S2 的运行时间是 $O((\log p)^3)$, 加上对于步骤 S3 的阶为 $e^2(\log p)^2$, 这同基于 (21) 的随机化方法的仅为 $O((\log p)^3)$ 相当。但是 Shanks 方法中的常数因子是小的。[*Congressus Numerantium* 7(1972), 58~62。一个有关的但不大有效的方法由 A. Tonelli 发表在 *Göttinger Nachrichten* (1891), 344~346。头一个发现具有预期运行时间 $O((\log p)^3)$ 的平方根算法的人是 M. Cipolla, *Rendiconti Accad. Sci. Fis. Mat. Napoli* 9 (1903), 154~163。]

16. a) 在对于 $n=1$ 进行的证明中, 以多项式模 p 代替整数。b) 把对于 $n=1$ 的证明贯彻到任何有限域中。c) 由于对于某个 $k, x = \xi^k$, 因此由 $f(x)$ 定义的域中 $x^{p^m} = x$ 。而且, 在这个域中满足等式 $y^{p^m} = y$ 的元素 y 在加法之下封闭, 而且在乘法之下也封闭。所以, 如果 $x^{p^m} = x$, 则 ξ (作为有整系数的 x 的多项式) 满足 $\xi^{p^m} = \xi$ 。

17. 如果 ξ 是一个原根, 则每一非零元素是 ξ 的某个乘方。因此阶必定是 $13^2 - 1 = 2^3 \cdot 3 \cdot 7$ 的一个因子, 且 $\varphi(f)$ 个元素有阶 f 。

f	$\varphi(f)$	f	$\varphi(f)$	f	$\varphi(f)$	f	$\varphi(f)$
1	1	3	2	7	6	21	12
2	1	6	2	14	6	42	12
4	2	12	4	28	12	84	24
8	4	24	8	56	24	168	48

18. (a) 由高斯引理, $\text{pp}(p_1(u_n x)) \cdots \text{pp}(p_r(u_n x))$ 。例如, 设

$$u(x) = 6x^3 - 3x^2 + 2x - 1$$

$$v(x) = x^3 - 3x^2 + 12x - 36 = (x^2 + 12)(x - 3)$$

则 $\text{pp}(36x^2 + 12) = 3x^2 + 1, \text{pp}(6x - 3) = 2x - 1$ 。(这是 14 世纪时使用过多年的求解代数方程技巧的一种现代形式。)

(b) 设 $\text{pp}(w(u_n x)) = \bar{w}_m x^m + \cdots + \bar{w}_0 = w(u_n x)/c$, 其中 c 是 x 的多项式 $w(u_n x)$ 的容度。则 $w(x) = (c\bar{w}/u_n^m)x^m + \cdots + c\bar{w}_0$, 因此 $c\bar{w}_m = u_n^m$; 因为 \bar{w}_m 是 u_n 的一个因子, 因此 c 是 u_n^{m-1} 的倍数。

19. 如果 $u(x) = v(x)w(x)$ 且 $\deg(v)\deg(w) \geq 1$, 则 $u_n x^n \equiv v(x)w(x) \pmod{p}$, 通过模 p 的惟一因子分解可知, 除了 v 和 w 的前导系数外所有系数都是 p 的倍数, 且 p^2 整除 $v_0 w_0 = u_0$ 。

20. a) $\sum(\alpha u_j - u_{j-1})(\bar{\alpha} \bar{u}_j - \bar{u}_{j-1}) = \sum(u_j - \bar{\alpha} u_{j-1})(\bar{u}_j - \alpha \bar{u}_{j-1})$ 。b) 我们可以

假定 $u_0 \neq 0$ 。设 $m(u) = \prod_{j=1}^n \min(1, |\alpha_j|) = |u_0|/M(u)$ 。每当 $|\alpha_j| < 1$ 时, 把 $u(x)$ 中的因子 $x - \alpha_j$ 变成 $\bar{\alpha}_j x - 1$; 这并不影响 $\|u\|$, 但它把 $|u_0|$ 改变成 $M(u)$ 。
 c) $u_j = u_m \sum \alpha_{i_1} \cdots \alpha_{i_{m-j}}$, 即一个初等对称函数, 因此 $|u_j| \leq |u_m| \sum \beta_{i_1} \cdots \beta_{i_{m-j}}$, 其中 $\beta_i = \max(1, |\alpha_i|)$ 。只要证明当 $x_1 \geq 1, \dots, x_n \geq 1$ 和 $x_1 \cdots x_n = M$ 时, 初等对称函数 $\sigma_{nk} = \sum x_{i_1} \cdots x_{i_k} \leq \binom{n-1}{k-1} M + \binom{n-1}{k}$, 即当 $x_1 = \cdots = x_{n-1} = 1$ 和 $x_n = M$ 时所取的值, 整个证明即完成了。(因为如果 $x_1 \leq \cdots \leq x_n < M$, 变换 $x_n \leftarrow x_n \cdot x_{n-1}, x_{n-1} \leftarrow 1$ 使 σ_{nk} 增加正量 $\sigma_{(n-2)(k-1)}(x_n - 1)(x_{n-1} - 1)$ 。)
 d) $|v_j| \leq \binom{m-1}{j} M(v) + \binom{m-1}{j-1} |v_m| \leq \binom{m-1}{j} M(u) + \binom{m-1}{j-1} |u_n|$, 因为 $M(v) \leq M(u)$ 和 $|v_m| \leq |u_n|$ 。[见 M. Mignotte, *Math. Comp.* 28 (1974), 1153~1157。]

注: 这个解表明 $\binom{m-1}{j} M(u) + \binom{m-1}{j-1} |u_n|$ 是上限, 所以我们希望有对 $M(u)$ 的一个更好的估计。已经知道有若干方法 [W. Specht, *Math. Zeit.* 53 (1950), 357~363; Cerlicenco, Mignotte 和 Piras, *J. Symbolic Comp.* 4 (1987), 21~33]。最简单也最快捷的也许是下列过程 [参见 C. H. Graeffe, *Auflösung der höheren numerischen Gleichungen* (Zürich: 1837)]: 假定 $u(x) = u_n(x - \alpha_1) \cdots (x - \alpha_n)$, 令 $\hat{u}(x) = u(\sqrt{x}) \cdot u(-\sqrt{x}) = (-1)^n u_n^2(x - \alpha_1^2) \cdots (x - \alpha_n^2)$ 。于是 $M(u)^2 = M(\hat{u}) \leq \|\hat{u}\|$ 。因此我们可以置 $c \leftarrow \|u\|, v \leftarrow u/c, t \leftarrow 0$, 然后重复地置 $t \leftarrow t+1, c \leftarrow \|\hat{v}\|^{1/2}, v \leftarrow \hat{v}/\|\hat{v}\|$ 。不变关系 $M(u) = cM(v)^{1/2^t}$ 和 $\|v\| = 1$ 确保在迭代的每一步都有 $M(u) \leq c$ 。注意当 $v(x) = v_0(x^2) + xv_1(x^2)$, 我们有 $\hat{v}(x) = v_0(x)^2 - xv_1(x)^2$ 。可以证明, 如果每个 $|\alpha_j|$ 小于等于 ρ 或大于等于 $1/\rho$, 则 $M(u) = \|u\|(1 + O(\rho))$; 因此在 t 步之后, c 将是 $M(u)(1 + O(\rho^{2^t}))$ 。

例如, 如果 $u(x)$ 是 (22) 的多项式, 对于 $t = 0, 1, 2, \dots$, c 的逐次值是 10.63, 12.42, 6.85, 6.64, 6.65, 6.6228, 6.62246, 6.2246, \dots 。在这个例子中, $\rho \approx .90982$ 。注意收敛并非单调的。最终 $v(x)$ 将收敛到单项式 x^m , 其中 m 是使 $|\alpha_j| < 1$ 的根的个数, 假定对于所有的 $j, |\alpha_j| \neq 1$; 一般说来, 如果有满足 $|\alpha_j| = 1$ 的 k 个根, 则 x^m 和 x^{m+k} 的系数将不趋于零, 而 x 的较高和较低次幂的系数将趋于零。

由 Jensen [*Acta Math.* 22 (1899), 359~364] 给出的一个著名公式证明 $M(u)$ 是单位圆上 $|u(x)|$ 的几何平均, 即 $\exp\left(\frac{1}{2\pi} \int_0^{2\pi} \ln |f(e^{i\theta})| d\theta\right)$ 。类似地, 习题 21a) 将说明 $\|u\|$ 是单位圆上 $|u(x)|$ 的均方根。不等式 $M(u) \leq \|u\|$ 可追溯到 E. Landau [*Bull. Soc. Math. de France* 33 (1905), 251~261], 它可理解为均值之间的一个关系。数 $M(u)$ 通常称为一个多项式的 Mahler 测度, 因为 Kurt Mahler 在

Mathematika 7 (1960), 98 ~ 100 使用了它。碰巧, Jensen 也证明了, 当 $m > 0$ 时,

$$\frac{1}{2\pi} \int_0^{2\pi} e^{im\theta} \ln |f(e^{i\theta})| d\theta = - \sum_{j=1}^n a_j^m / (2m \max(|\alpha_j|, 1)^{2m}).$$

21. a) 除非 $\mathbf{p} + \mathbf{s} = \mathbf{q} + \mathbf{r}$, 否则两边 $a_p b_q c_r d_s$ 的系数为 0。而且当这个条件成立时右边的系数是 $(\mathbf{p} + \mathbf{s})!$; 左边是

$$\sum_{\mathbf{j}} \binom{\mathbf{p}}{\mathbf{j}} \binom{\mathbf{s}}{\mathbf{r} - \mathbf{j}} = \mathbf{q}! \mathbf{r}! = \binom{\mathbf{p} + \mathbf{s}}{\mathbf{r}} \mathbf{q}! \mathbf{r}! = (\mathbf{q} + \mathbf{r})!$$

[B. Beavzamy 和 J. Dégot, *Trans. Amer. Math. Soc.* 345 (1995), 2607 ~ 2619; D. Zeilberger, *AMM* 101 (1994), 894 ~ 896.]

b) 令 $a_p = v_p, b_q = w_q, c_r = \bar{v}_r, d_s = \bar{w}_s$; 于是 a) 的右边是 $B(u)$, 而左边是对于每一个 \mathbf{j} 和 \mathbf{k} 的非负项之和。如果我们只考虑其中 $\sum \mathbf{j}$ 是 v 的次数的项, 则除非当 $\mathbf{p} = \mathbf{j}$ 时否则项 $v_p / (\mathbf{p} - \mathbf{j})!$ 将消失。因此那些项归结为

$$\sum_{\mathbf{j}, \mathbf{k}} \frac{1}{\mathbf{j}! \mathbf{k}!} |v_{\mathbf{j}} w_{\mathbf{k}} \mathbf{j}! \mathbf{k}!|^2 = B(v) B(w)$$

[B. Beavzamy, E. Bombieri, P. Enflo 及 H. Montgomery, *J. Number Theory* 36 (1990), 219 ~ 245.]

c) 如果需要使每个项都成为齐次的, 增加一个新变量不改变关系 $u = vw$ 。因此, 如果 v 和 w 分别有总次数 m 和 n , 我们有 $(m+n)! [u]^2 \geq m! [v]^2 n! [w]^2$; 换言之, $[v][w] \leq \binom{m+n}{m}^{1/2} [u]$ 。

顺便指出, 考虑 Bombieri 范式的一个好方法是想像诸变量是非交换的。例如, 代替 $3xy^3 - z^2w^2$ 我们可以写 $\frac{3}{4}xyyy + \frac{3}{4}yxxy + \frac{3}{4}yyxy + \frac{3}{4}yyx - \frac{1}{6}zzww - \frac{1}{6}zwzw - \frac{1}{6}zwz - \frac{1}{6}wzww - \frac{1}{6}wzww - \frac{1}{6}wzww$ 。于是 Bombieri 范式是在新的系数上的 $\|\cdot\|$ 范式。当 u 是次数为 n 的齐次式时, 另一个有趣的公式是

$$[u]^2 = \frac{1}{n! \pi^n} \int_{\mathbf{x}, \mathbf{y}} e^{-x_1^2 - \dots - x_t^2 - y_1^2 - \dots - y_t^2} |u(\mathbf{x} + i\mathbf{y})|^2 d\mathbf{x} d\mathbf{y}$$

d) 一个变量的情况对应于 $t=2$ 。假设 $u = vw$, 其中 v 是 t 个变量中次数为 m 的齐次式。于是对于所有的 \mathbf{k} , $|v_{\mathbf{k}}|^2 \mathbf{k}! / m! \leq [v]^2$, 而且 $\mathbf{k}! \geq (m/t)!$, 因为对于 $x > 0$, $\log \Gamma(x)$ 是凸的; 因此 $|v_{\mathbf{k}}|^2 \leq m! [v]^2 / (m/t)!$ 。我们可以假定 $m! [v]^2 / (m/t)! \leq m'! [w]^2 / (m'/t)!$, 其中 $m' = n - m$ 是 w 的次数。则

$$\begin{aligned} |v_{\mathbf{k}}|^2 &\leq m! [v]^2 / (m/t)! \leq \\ &m!^{1/2} m'^{1/2} [v][w] / (m/t)!^{1/2} (m'/t)!^{1/2} \leq \\ &n!^{1/2} [u] / (n/2t)!^{1/2} \end{aligned}$$

(如果我们把所有次数为 m 的表达式中未排除一个因子的倒数第二个表达式极大化, 就得到一个更好的上限。) 量 $n!^{1/4} / (n/2t)!^{1/2}$ 是 $c_t (2t)^{n/4} n^{-(2t-1)/8}$

$\left(1 + O\left(\frac{1}{n}\right)\right)$, 其中当 $t=2$ 时, $c_t = 2^{1/8} \pi^{-(2t-1)/8} t^{t/4} \approx 1.004$ 。

注意我们还未证明有这样小的系数的一个不可约因子的存在性; 可能需要进一步的分裂运算, 参见习题 41c。

e) $[u]^2 = \sum_k \binom{n}{k}^2 / \binom{2n}{2k} = \sum_k \binom{2k}{k} \binom{2n-2k}{n-k} / \binom{2n}{n} = 4^n / \binom{2n}{n} = \sqrt{\pi n} + O(n^{-1/2})$ 。如果 $v(x) = (x-1)^n$ 和 $w(x) = (x+1)^n$, 我们有 $[v] = [w] = 2^n$; 因此在此情况下, c) 的不等式是一个等式。

f) 设 u 和 v 是 m 次和 n 次的齐次式, 则由柯西不等式

$$[uv]^2 \leq \sum_{\mathbf{k}} \frac{\left(\sum_{\mathbf{j}} |u_{\mathbf{j}} v_{\mathbf{k}-\mathbf{j}}|\right)^2}{\binom{m+n}{\mathbf{k}}} \leq \sum_{\mathbf{k}} \left(\sum_{\mathbf{j}} \frac{|u_{\mathbf{j}}|^2}{\binom{m}{\mathbf{j}}} \frac{|v_{\mathbf{k}-\mathbf{j}}|^2}{\binom{n}{\mathbf{k}-\mathbf{j}}}\right) \left(\sum_{\mathbf{j}} \frac{\binom{m}{\mathbf{j}} \binom{n}{\mathbf{k}-\mathbf{j}}}{\binom{m+n}{\mathbf{k}}}\right) = [u]^2 [v]^2$$

[B. Beuzamy, *J. Symbolic Comp.* **13** (1992), 465~472, 命题 5.]

g) 由习题 20, $\left(\binom{n}{\lfloor n/2 \rfloor}\right)^{-1} M(u)^2 \leq \left(\binom{n}{\lfloor n/2 \rfloor}\right)^{-1} \|u\|^2 = \left(\binom{n}{\lfloor n/2 \rfloor}\right)^{-1} \sum_{\mathbf{j}} |u_{\mathbf{j}}|^2 \leq [u]^2 = \sum_{\mathbf{j}} \left(\binom{n}{\mathbf{j}}\right)^{-1} |u_{\mathbf{j}}|^2 \leq \sum_{\mathbf{j}} \left(\binom{n}{\mathbf{j}}\right)^{-1} M(u)^2 = 2^n M(u)^2$ 。上面的不等式也从 f) 得出, 因为如果 $u(x) = u_n \prod_{j=1}^n (x - \alpha_j)$ 我们有 $[u]^2 \leq |u_n|^2 \prod_{j=1}^n [x - \alpha_j]^2 = |u_n|^2 \prod_{j=1}^n (1 + |\alpha_j|^2) \leq |u_n|^2 \prod_{j=1}^n (2 \max(1, |\alpha_j|)^2) = 2^n M(u)^2$ 。

22. 更一般地说, 设 $u(x) \equiv v(x)w(x) \pmod{q}$, $a(x)v(x) + b(x)w(x) \equiv 1 \pmod{p}$ 及 $c \cdot \ell(v) \equiv 1 \pmod{r}$, $\deg(a) < \deg(w)$, $\deg(b) < \deg(v)$, $\deg(u) = \deg(v) + \deg(w)$, 其中 $r = \gcd(p, q)$, p, q 不必是素数。我们将构造多项式 $V(x) \equiv v(x)$ 和 $W(x) \equiv w(x) \pmod{q}$ 使得 $u(x) \equiv V(x)W(x) \pmod{qr}$, $\ell(V) = \ell(v)$, $\deg(V) = \deg(v)$, $\deg(W) = \deg(w)$; 而且如果 r 是素数, 则结果在模 qr 下将是惟一的。

这个问题要求我们在 $V(x) = v(x) + q\bar{v}(x)$, $W(x) = w(x) + q\bar{w}(x)$, $\deg(\bar{v}) < \deg(v)$, $\deg(\bar{w}) \leq \deg(w)$ 的情况下求 $\bar{v}(x)$ 和 $\bar{w}(x)$; 而另一个条件

$$(v(x) + q\bar{v}(x))(w(x) + q\bar{w}(x)) \equiv u(x) \pmod{qr}$$

等价于 $\bar{w}(x)v(x) + v(x)\bar{w}(x) \equiv f(x) \pmod{r}$, 其中 $f(x)$ 满足 $u(x) \equiv v(x)w(x) + qf(x) \pmod{qr}$ 。对所有 $t(x)$ 我们有

$$(a(x)f(x) + t(x)w(x))v(x) + (b(x)f(x) + t(x)v(x))w(x) \equiv f(x) \pmod{r}$$

由于在模 r 之下 $\ell(v)$ 有一个逆, 我们可以通过算法 4.1.6D 求一个商 $t(x)$, 使得

$\deg(bf - tv) < \deg(v)$; 对于这个 $t(x)$, $\deg(af + tw) \leq \deg(w)$, 因为我们有 $\deg(f) \leq \deg(u) = \deg(v) + \deg(w)$ 。于是所求的解是 $\bar{v}(x) = b(x)f(x) - t(x)v(x) = b(x)f(x) \bmod v(x)$, $\bar{w}(x) = a(x)f(x) + t(x)w(x)$ 。如果 $(\bar{v}(x), \bar{w}(x))$ 是另一个解, 我们有 $(\bar{w}(x) - \bar{w}(x))v(x) \equiv (\bar{v}(x) - v(x))w(x) \pmod{r}$ 。于是如果 r 是素数, $v(x)$ 必定整除 $\bar{v}(x) - v(x)$; 但 $\deg(\bar{v} - v) < \deg(v)$, 所以 $\bar{v}(x) = v(x)$ 而且 $\bar{w}(x) = w(x)$ 。

如果 p 整除 q , 使得 $r = p$, 则如同 Hensel 引理所要求的那样, 我们对 $V(x)$ 和 $W(x)$ 的选择也满足 $a(x)V(x) + b(x)W(x) \equiv 1 \pmod{p}$ 。

对于 $p = 2$, 因子分解过程如下(只写系数, 而且对负的数字使用上横杠表示): 习题 10 指出, 在一位 2 的补码记法下, $v_1(x) = (\overline{111})$, $w_1(x) = (\overline{11} \overline{1001} \overline{1})$ 。欧几里得扩充算法产生 $a(x) = (100001)$, $b(x) = (10)$ 。由习题 20 知因子 $v(x) = x^2 + c_1x + c_0$ 必定有 $|c_1| \leq \lfloor 1 + \sqrt{113} \rfloor = 11$, $|c_0| \leq 10$ 。三次应用 Hensel 引理产生 $v_4(x) = (\overline{131})$, $w_4(x) = (\overline{13} \overline{5} \overline{4} \overline{43} \overline{5})$ 。于是 $c_1 \equiv 3$ 和 $c_0 \equiv -1 \pmod{16}$; $u(x)$ 惟一可能的二次因子是 $x^2 + 3x - 1$ 。除法做不了, 所以 $u(x)$ 不可约。(因为我们现在已经用四个独立方法证明这个可爱的多项式的不可约性, 它不可能有任何因子。)

Hans Zassenhaus 已经发现: 通过加大 p 和 q , 我们通常可以加速这样的计算: 在上述记法下当 $r = p$ 时, 我们可以求 $A(x), B(x)$, 使得 $A(x)V(x) + B(x)W(x) \equiv 1 \pmod{p^2}$, 即通过取 $A(x) = a(x) + p\bar{a}(x)$, $B(x) = b(x) + p\bar{b}(x)$, 其中 $\bar{a}(x)V(x) + \bar{b}(x)W(x) \equiv g(x) \pmod{p}$, $a(x)V(x) + b(x)W(x) \equiv 1 - pg(x) \pmod{p^2}$ 。我们也可以求满足 $\ell(V)C \equiv 1 \pmod{p^2}$ 的 C 。这样一来我们可以把一个无平方因子分解 $u(x) \equiv v(x)w(x) \pmod{p}$ 升格为惟一的模 p^2 , p^4, p^8, p^{16} 扩充等等。然而, 只要我们达到双精度模, 这“加速”的过程就会在实践中达到一个转折点而趋于下降, 因为在实用范围内做多精度数乘法所需要的时间抵消了直接对模取平方的优点。从计算的观点看, 用逐次的模 $p, p^2, p^4, p^8, \dots, p^E, p^{E+e}, p^{E+2e}, p^{E+3e}, \dots$ 进行工作似乎是最好的, 其中 E 是使 p^E 大于单精度的 2 的最小幂次, 而 e 是使 p^e 有单精度的最大整数。

“Hensel 引理”实际上是由 C. F. Gauss 于大约 1799 年在一部未完成的书稿中发明的, *Analysis Residuorum*, § 373 ~ 374。高斯把这本手稿的大部分内容加进他的 *Disquisitiones Arithmeticae* (1801) 中, 但直到他去世之前他关于多项式分解的思想一直未发表过[参见他的 *Werke* 2 (Göttingen, 1876), 238]。Hensel 的名字被附加到这个方法上, 这是因为它对于 p -adic 数理论是基本的(参见习题 4.1-31)。这个引理可以以好几种方式进行推广。首先, 如果有多个因子, 比如说 $u(x) \equiv v_1(x)v_2(x)v_3(x) \pmod{p}$, 我们可以找到 $a_1(x), a_2(x), a_3(x)$ 使得 $a_1(x)v_2(x)v_3(x) + a_2(x)v_1(x)v_3(x) + a_3(x)v_1(x)v_2(x) \equiv 1 \pmod{p}$ 且 $\deg(a_i) < \deg(v_i)$ 。(实际上, $1/u(x)$ 被展开成部分分式如 $\sum a_i(x)/v_i(x)$ 。)一个完全类似的构造现在允许我们把因子分解升格而不改变 v_1 和 v_2 的前导系数; 我们取 $\bar{v}_1(x) = a_1(x)f(x) \bmod v_1(x)$, $\bar{v}_2(x) = a_2(x)f(x) \bmod v_2(x)$ 等等。另一个重要的推广是当实现多

变量的 gcd 和因子分解时取分别形如 $p^e, (x_2 - a_2)^{n_2}, \dots, (x_t - a_t)^{n_t}$ 的好多个联立的模。参考 D. Y. Y. Yun, Ph. D. Thesis (M. I. T., 1974)。

23. $\text{pp}(u(x))$ 的判别式是一个非零整数(参考习题 4.6.1-12), 当且仅当 p 整除此判别式时, 它有多于一个模 p 因子。[(22) 的模 3 因子分解是 $(x+1)(x^2-x-1)^2(x^3+x^2-x+1)$; 这个多项式的平方因子仅对 $p=3, 23, 233$ 和 121702457 出现。不难证明, 如果 $n = \deg(u)$ 和 N 是 $u(x)$ 系数的上限, 则最小的非不幸素数至多是 $O(n \log Nn)$ 。]

24. 以一个适当的非 0 整数来乘有有理系数的首一多项式, 以得到整数上的一个本原多项式。在整数上分解这个多项式, 而后把这些因子转换成首一的。(这样因子分解不会有任何损失; 见习题 4.6.1-8。)

25. 考察常数项可知, 没有次数为 1 的因子, 所以如果这个多项式是可约的, 则它必定有一次数为 2 的因子和一次数为 3 的因子。这些因子在模 2 下是 $x(x+1)^2(x^2+x+1)$; 这没有多大用处。在模 3 下这些因子是 $(x+2)^2(x^3+2x+2)$ 。在模 5 下它们是 $(x^2+x+1)(x^3+4x+2)$ 。所以我们看出答案是 $(x^2+x+1)(x^3-x+2)$ 。

26. 由 $D \leftarrow (0 \cdots 01)$ 开始, 表示集合 $\{0\}$ 。然后对于 $1 \leq j \leq r$, 置 $D \leftarrow D \vee (D \leftarrow \leftarrow d_j)$, 其中 \vee 表示逻辑“或”, 而 $D \leftarrow \leftarrow d$ 表示 D 左移 d 个二进位。(实际上我们只需使用长度为 $\lceil (n+1)/2 \rceil$ 的一个二进位向量, 因为 $n-m$ 在这集合中当且仅当 m 在这集合中。)

27. 习题 4 指出, 次数 n 的一个随机多项式在模 p 之下不可约的概率比较低, 大约为 $1/n$ 。但由中国剩余定理可知, 整数上 n 次随机首一多项式相对于 k 个不同素数均可约的概率大概是 $(1-1/n)^k$, 而且当 $k \rightarrow \infty$ 时这趋于 0。因此, 整数上几乎所有本原多项式相对无穷多个素数不可约; 而且几乎所有整数上的本原多项式也不可约。[W. S. Brown 已给出另外一个证明, 见 AMM 70 (1963), 965~969。]

28. 参考习题 4; 概率是 $[z^n](1+a_1 p z/p)(1+a_2 p z^2/p^2)(1+a_3 p z^3/p^3) \cdots$, 它有极限值 $g(z) = (1+z) \left(1 + \frac{1}{2} z^2\right) \left(1 + \frac{1}{3} z^3\right) \cdots$ 。对于 $1 \leq n \leq 10$, 答案是 $1, \frac{1}{2}, \frac{5}{6}, \frac{7}{12}, \frac{37}{60}, \frac{79}{120}, \frac{173}{280}, \frac{101}{168}, \frac{127}{210}, \frac{1033}{1680}$ 。[设 $f(y) = \ln(1+y) - y = O(y^2)$, 我们有

$$g(z) = \exp\left(\sum_{n \geq 1} z^n/n + \sum_{n \geq 1} f(z^n/n)\right) = h(z)/(1-z)$$

而且可以证明, 当 $n \rightarrow \infty$ 时, 极限的概率是 $h(1) = \exp(\sum_{n \geq 1} f(1/n)) = e^{-\gamma} \approx .56146$ 。其实, N. G. de Bruijn 已经确定了渐近公式 $\lim_{p \rightarrow \infty} a_{np} = e^{-\gamma} + e^{-\gamma}/n + O(n^{-2} \log n)$ 。[参见 D. H. Lehmer, Acta Arith. 21 (1972), 379~388; D. H. Greene 和 D. E. Knuth, Math. for the Analysis of Algorithms (Boston: Birkhäuser, 1981), §4.1.6。]另一方面, 当 $p=2$ 时对于 $1 \leq n \leq 10$ 的答案是更小的: $1, \frac{1}{4}, \frac{1}{2}, \frac{7}{16}, \frac{7}{16}$,

$\frac{7}{16}, \frac{27}{64}, \frac{111}{256}, \frac{109}{256}, \frac{109}{256}$ 。A. Knopfmacher 和 R. Warlimont [Trans. Amer. Math. Soc.

347(1995), 2235~2243]已经证明, 对于固定的 p , 概率是 $c_p + O(1/n)$, 其中 $c_p = \prod_{m \geq 1} e^{-1/m} (1 + a_{mp}/p^m)$ 且 $c_2 \approx .397$ 。]

29. 设 $q_1(x)$ 和 $q_2(x)$ 是 $g(x)$ 的两个不可约因子。由中国剩余定理(习题 3), 选择次数 $< 2d$ 的一个随机多项式 $t(x)$ 等价于选择次数 $< d$ 的两个随机多项式 $t_1(x)$ 和 $t_2(x)$, 其中 $t_i(x) = t(x) \bmod q_i(x)$ 。如果 $t_1(x)^{(p^d-1)/2} \bmod q_1(x) = 1$ 和 $t_2(x)^{(p^d-1)/2} \bmod q_1(x) \neq 1$, 或者反过来, \gcd 将是一个真因子, 同时对于 $t_1(x)$ 和 $t_2(x)$ 的恰好 $2((p^d-1)/2)((p^d+1)/2) = (p^{2d}-1)/2$ 种选择这个条件成立。

注: 我们这里仅仅考虑关于两个不可约因子的特性, 但真正的特性要好得多。假设每个不可约因子 $q_i(x)$ 对于每个 $t(x)$ 有 $1/2$ 的概率整除 $t(x)^{(p^d-1)/2} - 1$, 而同别的 $q_j(x)$ 和 $t(x)$ 的特性无关; 并且假定 $g(x)$ 总共有 r 个不可约因子。然后如果按照对于逐次试验的 $t, q_i(x)$ 能否整除 $t(x)^{(p^d-1)/2} - 1$, 我们用一个 0 和 1 的序列来对每个 $q_i(x)$ 进行编码, 则我们得到一个随机的带有 r 个叶的二叉检索结构(见 6.3 节)。在这个检索结构中, 每个有 m 个叶作为后裔的内部节点的代价是 $O(m^2(\log p))$; 而且由习题 5.2.2-36, 递推式 $A_n = \binom{n}{2} + 2^{1-n} \sum \binom{n}{k} A_k$ 的解是 $A_n = 2 \binom{n}{2}$ 。因此在这个似真的假设下在给定的随机检索结构——表示完全分解 $g(x)$ 的预期时间——中花销之和是 $O(r^2(\log p)^3)$ 。如果我们随机地选择次数 $< rd$ 的 $t(x)$ 而不是限制它的次数 $< 2d$, 则这个似真的假设就成为严格正确的了。

30. 设 $T(x) = x + x^p + \cdots + x^{p^{d-1}}$ 是 x 的迹, 并设 $v(x) = T(t(x)) \bmod q(x)$ 。由于在多项式剩余模 $q(x)$ 的域中 $t(x)^{p^d} = t(x)$, 我们在该域便有 $v(x)^p = v(x)$; 换句话说, $v(x)$ 是方程 $y^p - y = 0$ 的 p 个根之一。因此 $v(x)$ 是一个整数。

由此得出, $\prod_{s=0}^{p-1} \gcd(g_d(x), T(t(x)) - s) = g_d(x)$ 。特别是, 当 $p=2$ 时, 我们可以像在习题 29 中那样论证, 当 $g_d(x)$ 至少有两个不可约因子和 $t(x)$ 是次数 $< 2d$ 的一个随机二进多项式时, $\gcd(g_d(x), T(t(x)))$ 将以大于等于 $1/2$ 的概率为 $g_d(x)$ 的一个真因子。

[注意, 从 $u(x) \leftarrow t(x)$ 开始, 并重复地置 $u(x) \leftarrow (t(x) + u(x)^p) \bmod g(x)$ 共 $d-1$ 次, 可以计算 $T(t(x)) \bmod g(x)$ 。本题的方法是以多项式因子分解 $x^{p^d} - x = \prod_{s=0}^{p-1} (T(x) - s)$ 为基础的, 它对任何 p 成立, 同时公式(21)是以对奇数 p 的多项式因子分解 $x^{p^d} - x = x(x^{(p^d-1)/2} + 1)(x^{(p^d-1)/2} - 1)$ 为基础的。]

迹是由 Richard Dedekind 引入的, 见 *Abhandlungen der Königl. Gesellschaft der Wissenschaften zu Göttingen* 29 (1882), 1~56。计算 $\gcd(f(x), T(x) - s)$ 来求 $f(x)$ 的因子的技术可追溯到 A. Arwin, *Arkiv för Mat., Astro. och Fys.* 14, 7 (1918), 1~46; 但他的方法是不完备的, 因为他没有考虑 $t(x) \neq x$ 时的 $T(t(x))$ 。后来由

R.J. McEliece 设计了使用迹的一个完全的因子分解算法, *Math. Comp.* **23** (1969), 861~867; 关于渐近的快的结果, 也参见 von zur Gathen 和 Shoup, *Computational Complexity* **2** (1992), 187~224, 算法 3.6。

Henri Cohen 已经发现, 对于 $p=2$, 当应用这个方法时, 只须检验至多 d 种特殊情况 $t(x) = x, x^3, \dots, x^{2^d-1}$ 就足够了。每当 $g_d(x)$ 是可约的时, $t(x)$ 的这些选择之一确保分裂 $g_d(x)$, 因为利用 $T(t(x)^p) \equiv T(t(x))$ 和 $T(u(x) + t(x)) \equiv T(u(x)) + T(t(x)) \pmod{g_d(x)}$ 这些事实, 由这些特殊情况, 我们可以得到次数 $< 2d$ 的所有多项式 $t(x)$ 的效果。[*A Course in Computational Algebraic Number Theory* (Springer, 1993), 算法 3.4.8。]

31. 如果 α 是 p^d 个元素的域中的元素, 设 $d(\alpha)$ 是 α 的次数, 即使得 $\alpha^{p^e} = \alpha$ 的最小指数 e , 则考虑多项式

$$P_\alpha(x) = (x - \alpha)(x - \alpha^p) \cdots (x - \alpha^{p^{d-1}}) = q_\alpha(x)^{d/d(\alpha)}$$

其中 $q_\alpha(x)$ 是次数为 $d(\alpha)$ 的一个不可约多项式。当 α 跑遍这个域所有元素时, 对应的 $q_\alpha(x)$ 跑遍其次数 e 能整除 d 的每个不可约多项式, 其中每个这样的不可约多项式恰出现 e 次。我们有 $(x+t)^{(p^d-1)/2} \pmod{q_\alpha(x)} = 1$ 当且仅当在这个域中 $(\alpha+t)^{(p^d-1)/2} = 1$ 。如果 t 是一个整数, 我们有 $d(\alpha+t) = d(\alpha)$, 因此 $n(p, d)$ 是 d^{-1} 乘以使得 $\alpha^{(p^d-1)/2} = 1$ 的次数为 d 的元素 α 的个数。类似地, 如果 $t_1 \neq t_2$, 我们要来计算使得 $(\alpha+t_1)^{(p^d-1)/2} = (\alpha+t_2)^{(p^d-1)/2}$, 或等价地 $((\alpha+t_1)/(\alpha+t_2))^{(p^d-1)/2} = 1$ 的次数为 d 的元素个数。当 α 跑遍次数为 d 的所有元素时, 量 $(\alpha+t_1)/(\alpha+t_2) = 1 + (t_1-t_2)/(\alpha+t_2)$ 也如此。

[我们有 $n(p, d) = \frac{1}{4} d^{-1} \sum_{c \mid d} (3 + (-1)^c) \mu(c) (p^{d/c} - 1)$, 它大约是不可约多项式总数的一半——事实上, 当 d 为奇数时, 恰好一半。这就证明当 t 固定而 $g_d(x)$ 随机地选择时, 用 $\gcd(g_d(x), (x+t)^{(p^d-1)/2} - 1)$ 求得 $g_d(x)$ 的因子的可能性很大; 但是一个随机算法应当如同在习题 29 中那样, 对固定的 $g_d(x)$ 和随机的 t , 以保证的概率进行工作。]

32. a) 显然 $x^n - 1 = \prod_{d \mid n} \Psi_d(x)$, 因为对于某个惟一的 $d \mid n$, 每个复数 n 次单位根是本原 d 次根。第二个恒等式从头一个得出; 而 $\Psi_n(x)$ 有整系数, 因为它借助于带有整系数的首一多项式的积和商表达。

b) 提示中的条件足以证明 $f(x) = \Psi_n(x)$, 所以我们应采纳提示。当 p 不整除 n 时, 我们有 $x^n - 1 \perp nx^{n-1} \pmod{p}$, 因此 $x^n - 1$ 在模 p 下是无平方的。如同在提示中那样给定 $f(x)$ 和 ζ , 设 $g(x)$ 是使得 $g(\zeta^p) = 0$ 的 $\Psi_n(x)$ 的不可约因子。如果 $g(x) \neq f(x)$, 则 $f(x)$ 和 $g(x)$ 两者都是 $\Psi_n(x)$ 的不同因子, 因此它们是 $x^n - 1$ 的不同因子, 因此它们在模 p 之下没有公共的不可约因子。然而 ζ 是 $g(x^p)$ 的一个

根,所以在整数上 $\gcd(f(x), g(x^p)) \neq 1$, 因此 $f(x)$ 是 $g(x^p)$ 的一个因子。由(5), $f(x)$ 是 $g(x)^p$ 的一个模 p 因子, 同 $f(x)$ 和 $g(x)$ 没有公共的不可约因子的假定矛盾。因此 $f(x) = g(x)$ 。[对于素数 n , $\Psi_n(x)$ 的不可约性首先是由 C.F. Gauss 在 *Diquisitiones Arithmeticae* (Leipzig, 1801), Atr. 341 中证明的, 而对于一般的 n , 是由 L. Kronecker 证明的, 见 *J. de Math. Pures et Appliquées* 19 (1854), 177~192。]

c) $\Psi_1(x) = x - 1$; 而且当 p 是素数时, $\Psi_p(x) = 1 + x + \cdots + x^{p-1}$ 。如果 $n > 1$ 是奇数, 不难证明 $\Psi_{2n}(x) = \Psi_n(-x)$ 。如果 p 整除 n , 则 a) 中第二个恒等式表明 $\Psi_{pn}(x) = \Psi_n(x^p)$ 。如果 p 不整除 n , 我们有 $\Psi_{pn}(x) = \Psi_n(x^p)/\Psi_n(x)$ 。对于非素数 $n \leq 15$, 我们有 $\Psi_4(x) = x^2 + 1$, $\Psi_6(x) = x^2 - x + 1$, $\Psi_8(x) = x^4 + 1$, $\Psi_9(x) = x^6 + x^3 + 1$, $\Psi_{10}(x) = x^4 - x^3 + x^2 - x + 1$, $\Psi_{12}(x) = x^4 - x^2 + 1$, $\Psi_{14}(x) = x^6 - x^5 + x^4 - x^3 + x^2 - x + 1$, $\Psi_{15}(x) = x^8 - x^7 + x^5 - x^4 + x^3 - x + 1$ 。[公式 $\Psi_{pq}(x) = (1 + x^p + \cdots + x^{(q-1)p})(x-1)/(x^q-1)$ 可以用来证明当 p 和 q 是素数时 Ψ_{pq} 的所有系数为 ± 1 或 0; 但 $\Psi_{pqr}(x)$ 的系数可以任意大。]

33. 假的; 我们损失所有 p_j 且 e_j 可为 p 所整除。如果 $p > \deg(u)$ 则为真。[见习题 36。]

34. [D. Y. Y. Yun. *Proc. ACM Symp. Symbolic and Algebraic Comp.* (1976), 26~35。] 置 $(t(x), v_1(x), w_1(x)) \leftarrow \text{GCD}(u(x), u'(x))$ 。如果 $t(x) = 1$, 置 $e \leftarrow 1$; 否则对于 $i = 1, 2, \dots, e-1$ 置 $(u_i(x), v_{i+1}(x), w_{i+1}(x)) \leftarrow \text{GCD}(v_i(x), w_i(x) - v'_i(x))$, 直到求出 $w_e(x) - v'_e(x) = 0$ 为止。最后置 $u_e(x) \leftarrow v_e(x)$ 。

为证明这个算法的正确性, 我们注意到, 它计算多项式 $t(x) = u_2(x)u_3(x)^2 \cdot u_4(x)^3 \cdots$, $v_i(x) = u_i(x)u_{i+1}(x)u_{i+2}(x) \cdots$, 而且

$$w_i(x) = u'_i(x)u_{i+1}(x)u_{i+2}(x) \cdots + 2u_i(x)u'_{i+1}(x)u_{i+2}(x) \cdots + 3u_i(x)u_{i+1}(x)u'_{i+2}(x) \cdots + \cdots$$

我们有 $t(x) \perp w_1(x)$, 因为 $u_i(x)$ 的一个不可约因子整除 $w_1(x)$ 的第 i 项之外的所有项, 因而它与该项互素, 而且显然有 $u_i(x) \perp v_{i+1}(x)$ 。

[尽管习题 2(b) 证明大多数多项式是无平方的, 但在实践中非无平方的多项式实际上经常出现。因此这个方法是十分重要的。关于如何改进其效率的讨论, 请参见 Paul S. Wang 和 Barry M. Trager, *SICOMP* 8 (1979), 300~305。Bach 和 Shallit, *Algorithmic Number Theory* 1 (MIT Press, 1996), 习题 7.27 的答案, 也讨论了模 p 的无平方因子分解的问题。]

35. 我们有 $w_j(x) = \gcd(u_j(x), v_j^*(x)) \cdot \gcd(u_{j+1}^*(x), v_j(x))$, 其中

$$u_j^*(x) = u_j(x)u_{j+1}(x) \cdots \text{ 且 } v_j^*(x) = v_j(x)v_{j+1}(x) \cdots$$

[Yun 注意到用习题 34 的方法进行无平方的因子分解的运行时间至多大约是计算 $\gcd(u(x), u'(x))$ 的运行时间的两倍。而且如果有了发现无平方因子分解的任意方法, 则本题的方法导致一个 gcd 过程。(当 $u(x)$ 和 $v(x)$ 是无平方时, 它们的 gcd 就是 $w_2(x)$, 其中 $w(x) = u(x)v(x) = w_1(x)w_2(x)^2$; 多项式 $u_j(x), v_j(x)$,

$u_j^*(x)$ 和 $v_j^*(x)$ 都是无平方的。)因此,把一个本原 n 次多项式转换成为它的无平方表示的问题在渐近运行时间最坏的意义下,在计算上等价于计算两个 n 次多项式的 gcd 的问题。]

36. 设 $U_j(x)$ 是用习题 34 的过程计算“ $u_j(x)$ ”所得的值。如果 $\deg(U_1) + 2\deg(U_2) + \cdots = \deg(u)$, 则对所有的 j , $u_j(x) = U_j(x)$ 。但是一般说来,我们将有 $e < p$, 而且对于 $1 \leq j < p$, $U_j(x) = \prod_{k \geq 0} u_{j+pk}(x)$ 。为了进一步分解这些因子,可以计算 $t(x)/(U_2(x)U_3(x)^2 \cdots U_{p-1}(x)^{p-2}) = \prod_{j \geq p} u_j(x)^{p-j/p} = z(x^p)$ 。在递归地寻求 $z(x) = (z_1(x), z_2(x), \cdots)$ 的无平方表示之后,我们将有 $z_k(x) = \prod_{0 \leq j < p} u_{j+pk}(x)$, 所以我们可以通过对于 $1 \leq j < p$ 的公式 $\gcd(U_j(x), z_k(x)) = u_{j+pk}(x)$, 来计算个别的 $u_i(x)$ 。当删去了 $z_k(x)$ 的其它因子时,就将剩下多项式 $u_{pk}(x)$ 。

注:这一过程是相当简单的,但程序较长。如果人们的目标是有用于模 p 完全因子分解的一个短程序,而不是特别有效的程序,则修改不同次的因子分解程序,使得它对相同的 d 值分离 $\gcd(x^{p^d} - x, u(x))$ 好多次,直到 gcd 为 1 为止,这可能是最容易的。在这种情况下,你不必像正文中所提示的那样,由计算 $\gcd(u(x), u'(x))$ 开始并删去多重因子,因为多项式 $x^{p^d} - x$ 是无平方的。

37. 精确的概率是 $\prod_{j \geq 1} (a_{jp}/p^j)^{k_j/k_j!}$, 其中 k_j 是等于 j 的 d_i 的个数。因为由习题 4, $a_{jp}/p^j \approx 1/j$, 我们得到习题 1.3.3-21 的公式。

注:本题指出,如果我们固定素数 p 并让多项式 $u(x)$ 是随机的,则在模 p 下将有以给定的方式分解的可能性。一个难得多的问题是固定多项式 $u(x)$ 并令 p “随机”;结果是对于几乎所有的 $u(x)$, 相同的渐近结果成立。G. Frobenius 于 1880 年证明,整数多项式 $u(x)$, 当 p 是随机地选定的很大素数时,在模 p 下分解成为次数为 d_1, \cdots, d_r 的因子,而且其概率等于具有循环长度 $\{d_1, \cdots, d_r\}$ 的 $u(x)$ 的 Galois 群 G 中的排列数除以 G 中排列的总数。[如果 $u(x)$ 有有理系数且在复数域上有不同的根 ξ_1, \cdots, ξ_n , 则它的伽罗瓦群是使得多项式 $\prod_{p(1) \cdots p(n) \in G} (z + \xi_{p(1)}y_1 + \cdots + \xi_{p(n)}y_n) = U(z, y_1, \cdots, y_n)$ 有有理系数,而且在有理数上不可约的(惟一的)排列群 G ; 参见 G. Frobenius, *Sitzungsberichte Königl. preuß Akad. Wiss.* (Berlin: 1896), 689 ~ 703。线性映射 $x \mapsto x^p$ 传统上称做 Frobenius 自同构的就由于这一著名论文所致。]而且 B. L. van der Waerden 已经于 1934 年证明,几乎所有的 n 次多项式都以全部 $n!$ 个排列的集合作为它们的伽罗瓦群 [*Math. Annalen* **109** (1934), 13 ~ 16]。因此,几乎所有固定的不可约多项式 $u(x)$ 都将如我们所希望的那样相对于随机选择的充分大的素数 p 进行分解。关于 Frobenius 定理用于猜测伽罗瓦群类的推广,也见 N. Chebotarev, *Math. Annalen* **95** (1926)。

38. 这些条件意味着,当 $|z| = 1$ 时或者 $|u_{n-2}z^{n-2} + \cdots + u_0| < |u_{n-1}| - 1 \leq |z^n + u_{n-1}z^{n-1}|$ 或者 $|u_{n-3}z^{n-3} + \cdots + u_0| < u_{n-2} - 1 \leq |z^n + u_{n-2}z^{n-2}|$ 。因此由 Rouché 定理 [*J. École Polytechnique* **21**, 37 (1858), 1 ~ 34], 在圆 $|z| = 1$ 内 $u(z)$ 至

少有 $n-1$ 或 $n-2$ 个根。如果 $u(z)$ 是可约的,则可以把它写成 $v(z)w(z)$, 其中 v 和 w 是首一整多项式。 v 和 w 的根的乘积是非零的整数,所以每个因子都有绝对值 ≥ 1 的一个根。因此惟一的可能性是 v 和 w 两者都恰有一个这样的根而且 $u_{n-1}=0$ 。这些根必须是实的,因为复共轭是根;因此 $u(z)$ 有一个实根 z_0 且有 $|z_0| \geq 1$ 。但这不可能,因为如果 $r=1/z_0$,我们就有 $0 = |1 + u_{n-2}r^2 + \cdots + u_0r^n| \geq 1 + u_{n-2}r^2 - |u_{n-3}|r^3 - \cdots - |u_0|r^n > 1$ 。[O. Perron, *Crelle* **132**(1907), 288~307; 关于推广,参见 A. Brauer, *Amer. J. Math.* **70** (1948), 423~432, **73** (1951), 717~720。]

39. 首先我们证明提示:令 $u(x) = a(x - \alpha_1) \cdots (x - \alpha_n)$ 有整系数, $u(x)$ 连同多项式 $y - t(x)$ 的结式是一个行列式,所以它是有整系数的一个多项式 $r_t(y) = a^{\deg(t)}(y - t(\alpha_1)) \cdots (y - t(\alpha_n))$ (参见习题 4.6.1-12)。如果 $u(x)$ 整除 $v(t(x))$, 则 $v(t(\alpha_1)) = 0$, 因此 $r_t(y)$ 和 $v(y)$ 有一个公因子。所以如果 v 是不可约的,则我们有 $\deg(u) = \deg(r_t) \geq \deg(v)$ 。

给定一个不可约多项式 $u(x)$, 其不可约性的一个短的证明是需要的,由习题 18, 我们可以假定 $u(x)$ 是首一的,而且, $\deg(u) \geq 3$ 。想法是如何来证明一个多项式 $t(x)$ 的存在性使得通过习题 38 的准则, $v(y) = r_t(y)$ 是不可约的。于是, $u(x)$ 的所有因子整除多项式 $v(t(x))$, 因而这将证明 $u(x)$ 是不可约的。如果 $t(x)$ 的系数都适当地小,则这个证明将是简明的。

如果 $n \geq 3$ 和 $\beta_1 \cdots \beta_n \neq 0$, 而且如果下列的“小性条件”成立,则可以证明多项式 $v(y) = (y - \beta_1) \cdots (y - \beta_n)$ 满足习题 38 的准则:即除了当 $j = n$ 或当 $\beta_j = \bar{\beta}_n$ 且 $|\Re \beta_j| \leq 1/(4n)$ 时, $|\beta_j| \leq 1/(4n)$ 。利用 $|v_0| + \cdots + |v_n| \leq (1 + |\beta_1|) \cdots (1 + |\beta_n|)$ 这一事实,这个计算是直截了当的。

令 $\alpha_1, \dots, \alpha_r$ 是实的和 $\alpha_{r+1}, \dots, \alpha_{r+s}$ 是复的,其中 $n = r + 2s$ 和对于 $1 \leq j \leq s$, $\alpha_{r+s+j} = \bar{\alpha}_{r+j}$ 。考虑对于 $1 \leq j \leq r+s$ 被定义为 $\Re(\sum_{i=0}^{n-1} a_i \alpha_j^i)$ 和对于 $r+s < j \leq n$ 被定义为 $\Re(\sum_{i=0}^{n-1} a_i \alpha_j^i)$ 的线性表达式 $S_j(a_0, \dots, a_{n-1})$ 。如果 $0 \leq a_i < b$ 和 $B = \lceil \max_{j=1}^{n-1} |\sum_{i=0}^{n-1} a_i \alpha_j^i| \rceil$, 我们有 $|S_j(a_0, \dots, a_{n-1})| < bB$ 。因此,如果我们选择 $b > (16nB)^{n-1}$, 则必存在不同的向量 (a_0, \dots, a_{n-1}) 和 (a'_0, \dots, a'_{n-1}) 使得对于 $1 \leq j < n$, $\lfloor 8nS_j(a_0, \dots, a_{n-1}) \rfloor = \lfloor 8nS_j(a'_0, \dots, a'_{n-1}) \rfloor$, 因为有 b^n 个向量但至多有 $(16nbB)^{n-1} < b^n$ 个可能的 $(n-1)$ 值元组。令 $t(x) = (a_0 - a'_0) + \cdots + (a_{n-1} - a'_{n-1})x^{n-1}$ 和 $\beta_j = t(\alpha_j)$ 。于是小性条件满足,而且 $\beta_j \neq 0$; 否则 $t(x)$ 将整除 $u(x)$ 。[*J. Algorithms* **2** (1981), 385~392。]

40. 给定一个候选因子 $v(x) = x^d + a_{d-1}x^{d-1} + \cdots + a_0$, 把每个 a_j 改成一个有理小数(模 p^e), 且分子和分母都小于等于 B 。然后以最小公分母来乘,并看看得到的多项式在整数上是否整除 $u(x)$ 。如果不,则系数以 B 为界的 $u(x)$ 没有在模 p^e 之下同余于 $v(x)$ 的倍数的因子。

41. David Boyd 注意到 $4x^8 + 4x^6 + x^4 + 4x^2 + 4 = (2x^4 + 4x^3 + 5x^2 + 4x + 2) \times (2x^4 - 4x^3 + 5x^2 - 4x + 2)$, 而且他发现了更高次的例子来证明如果它存在,则 c 必

定是大于 2 的。

4.6.3 小节

1. x^m , 其中 $m = 2^{\lfloor \lg n \rfloor}$ 即小于或等于 n 的 2 的最高次幂。
2. 假设 x 是寄存器 A 中的输入, 而且 n 在单元 NN 中; 输出在寄存器 X 中。

01	A1	ENTX	1	1	<u>A1. 初始化</u>
02		STX	Y	1	$Y \leftarrow 1$
03		STA	Z	1	$Z \leftarrow x$
04		LDA	NN	1	$N \leftarrow n$
05		JAP	2F	1	转到 A2
06		JMP	DONE	0	否则回答是 1
07	5H	SRB	1	$L + 1 - K$	
08		STA	N	$L + 1 - K$	$N \leftarrow \lfloor N/2 \rfloor$
09	A5	LDA	Z	L	<u>A5. 平方 Z</u>
10		MUL	Z	L	
11		STX	Z	L	$Z \leftarrow Z \times Z \bmod w$
12	A2	LDA	N	L	<u>A2. N 折半</u>
13	2H	JAE	5B	$L + 1$	如果 N 为偶数, 则转 A5
14		SRB	1	K	
15	A4	JAZ	4F	K	如果 $N = 1$, 则转移
16		STA	N	$K - 1$	$N \leftarrow \lfloor N/2 \rfloor$
17	A3	LDA	Z	$K - 1$	<u>A3. Y 乘以 Z</u>
18		MUL	Y	$K - 1$	
19		STX	Y	$K - 1$	$Y \leftarrow Z \times Y \bmod w$
20		JMP	A5	$K - 1$	转 A5
21	4H	LDA	Z	1	
22		MUL	Y	1	做最后的乘法 1

运行时间是 $21L + 16K + 8$, 其中 $L = \lambda(n)$, 比 n 的二进表示中的二进位数少 1, $K = \nu(n)$ 是在表示中二进位 1 的个数。

对于连续的程序, 我们可以假定 n 小到可以填入一个变址寄存器; 否则连续的求幂就不可能了。下列程序把输出保留在寄存器 A 中:

01	S1	LD1	NN	1	$r11 \leftarrow n$
02		STA	X	1	$X \leftarrow x$

03		JMP	2F	1	
04	1H	MUL	X	$N-1$	$rA \times X \bmod w$
05		SLAX	5	$N-1$	$\rightarrow rA$
06	2H	DEC1	1	N	$rI1 \leftarrow rI1 - 1$
07		J1P	1B	N	如果 $rI1 > 0$ 则再次乘

这个程序的运行时间是 $14N-7$; 当 $n \leq 7$ 时它比上边的程序运行得更快; 当 $n \geq 8$ 时, 则慢些。

3. 指数序列为: (a) 1, 2, 3, 6, 7, 14, 15, 30, 60, 120, 121, 242, 243, 486, 487, 974, 975 [16 个乘法]; (b) 1, 2, 3, 4, 8, 12, 24, 36, 72, 108, 216, 324, 325, 650, 975 [14 个乘法]; (c) 1, 2, 3, 6, 12, 15, 30, 60, 120, 240, 243, 486, 972, 975 [13 个乘法]; (d) 1, 2, 3, 6, 12, 15, 30, 60, 75, 150, 300, 600, 900, 975 [13 个乘法]。[最少可能的乘法数是 12; 这是通过把因子方法同二进方法合起来得到的, 因为 $975 = 15 \cdot (2^6 + 1)$ 。]

4. $(777777)_8 = 2^{18} - 1$ 。

5. T1. [初始化] 对于 $0 \leq j \leq 2^r$ 置 $LINKU[j] \leftarrow 0$, 并置 $k \leftarrow 0$, $LINKR[0] \leftarrow 1$, $LINKR[1] \leftarrow 0$ 。

T2. [改变层次] (现在树的第 k 层已经从左到右地连接在一起, 在 $LINKR[0]$ 处开始。) 如果 $k = r$, 则此算法终止。否则置 $n \leftarrow LINKR[0]$, $m \leftarrow 0$ 。

T3. [准备 n] (现在 n 是第 k 层上的节点, 而且 m 指向 $k+1$ 层上当前最右的节点。) 置 $q \leftarrow 0$, $s \leftarrow n$ 。

T4. [已在树中?] (现在 s 是在从根到 n 的通路中的一个节点。) 如果 $LINKU[n+s] \neq 0$, 则转到 T6 (值 $n+s$ 已在树中)。

T5. [插入 n 之下] 如果 $q = 0$, 则置 $m' \leftarrow n+s$ 。然后置 $LINKR[n+s] \leftarrow q$, $LINKU[n+s] \leftarrow n$, $q \leftarrow n+s$ 。

T6. [上移] 置 $s \leftarrow LINKU[s]$ 。如果 $s \neq 0$, 则返回 T4。

T7. [配组] 如果 $q \neq 0$, 则置 $LINKR[m] \leftarrow q$, $m \leftarrow m'$ 。

T8. [移动 n] 置 $n \leftarrow LINKR[n]$ 。如果 $n \neq 0$, 则返回 T3。

T9. [层的结束] 置 $LINKR[m] \leftarrow 0$, $k \leftarrow k+1$, 并返回 T2。 ▮

6. 用归纳法证明, 如果 $e_0 > e_1 > \cdots > e_r \geq 0$, 则通向数 $2^{e_0} + 2^{e_1} + \cdots + 2^{e_r}$ 的通路是 $1, 2, 2^2, \cdots, 2^{e_0}, 2^{e_0} + 2^{e_1}, \cdots, 2^{e_0} + 2^{e_1} + \cdots + 2^{e_r}$; 而且每层上的指数序列是在递减的字典序下。

7. 这个二进因子分解方法要求计算 x^{2^n} 比计算 x^n 多一步; 幂树方法要求至多一步。因此 a) $15 \cdot 2^k$; b) $33 \cdot 2^k$; c) $23 \cdot 2^k$; $k = 0, 1, 2, 3, \cdots$ 。

8. 幂树总是在 m 之下的一层处包括节点 $2m$, 除非它出现在同一层或更早的一层处; 而且它总是在 $2m$ 之下的一层处包括节点 $2m+1$, 除非它出现在同一层或更早的一层处。[对于所有的 m , 在幂树中 $2m$ 是 m 的儿子这一点并不是对的; 这一点不对的最小例子是 $m = 2138$, 它出现于 15 层上; 而 4276 出现在 16 层的某处。事实上, 有的 $2m$ 和 m 在同一层上; 最小的例子是 $m = 6029$ 。]

9. 从 $N \leftarrow n, Z \leftarrow x$, 及对于 $1 \leq q < m, q$ 为奇数, 置 $Y_q \leftarrow 1$ 开始; 一般来说, 当算法进行时, 我们将有 $x^n = Y_1 Y_3^3 Y_5^5 \cdots Y_{m-1}^{m-1} Z^N$. 假定 $N > 0$, 置 $k \leftarrow N \bmod m, N \leftarrow \lfloor N/m \rfloor$. 则如果 $k = 0$, 置 $Z \leftarrow Z^m$ 并重复; 否则如果 $k = 2^p q$, 其中 q 是奇数, 置 $Z \leftarrow Z^{2^p}, Y_q \leftarrow Y_q \cdot Z$, 而且如果 $N > 0$ 则置 $Z \leftarrow Z^{2^{p-1}}$ 并重复. 最后, 对于 $k = m-3, m-5, \dots, 1$ 置 $Y_k \leftarrow Y_k \cdot Y_{k+2}$; 答案为 $Y_1(Y_3 Y_5 \cdots Y_{m-1})^2$. (大约 $m/2$ 的乘法是乘以 1.)

10. 使用 2.3.3 小节中讨论的“PARENT”表示: 利用一张表 $p[j], 1 \leq j \leq 100$, 使得 $p[1] = 0$, 而对于 $j \geq 2, p[j]$ 是恰在 j 上边的节点个数. (这个树的每个节点至多次数为 2, 这对这个表示的效率没有影响; 它仅仅使这个树作为图示看起来更好些.)

11. 1, 2, 3, 5, 10, 20, (23 或 40), 43; 1, 2, 4, 8, 9, 17, (26 或 34), 43; 1, 2, 4, 8, 9, 17, 34, (43 或 68), 77; 1, 2, 4, 5, 9, 18, 36, (41 或 72), 77. 如果后两条通路中的任一条在树中, 则我们对于 $n = 43$ 将没有可能性, 因为这树必须包含 1, 2, 3, 5 或 1, 2, 4, 8, 9.

12. 这样的无穷树不能存在, 因为对某个 $n, l(n) \neq l^*(n)$.

13. 对于情况 1, 用一条类型 1 的链, 后边跟之以 $2^{A+C} + 2^{B+C} + 2^A + 2^B$; 或者用因子方法. 对于情况 2, 用一条类型 2 的链, 后边跟之以 $2^{A+C+1} + 2^{B+C} + 2^A + 2^B$. 对于情况 3, 用一条类型 5 的链, 后边跟之以 $2^A + 2^{A-1}$, 或者用因子方法. 对于情况 4, $n = 135 \cdot 2^D$, 所以我们可以用因子方法.

14. (a) 容易验证, 步 $r-1$ 和步 $r-2$ 不全都是小的, 所以我们假设步 $r-1$ 是小的而步 $r-2$ 不是. 如果 $c=1$, 则 $\lambda(a_{r-1}) = \lambda(a_{r-k})$, 所以 $k=2$; 而且因为 $4 \leq \nu(a_r) = \nu(a_{r-1}) + \nu(a_{r-k}) - 1 \leq \nu(a_{r-1}) + 1$, 我们有 $\nu(a_{r-1}) \geq 3$, 使 $r-1$ 成为一个星步 (以免 $a_0, a_1, \dots, a_{r-3}, a_{r-1}$ 仅包含一个小步). 于是对某个 $q, a_{r-1} = a_{r-2} + a_{r-q}$, 而且如果我们以 $a_{r-2}, 2a_{r-2}, 2a_{r-2} + a_{r-q} = a_r$ 代替 a_{r-2}, a_{r-1}, a_r , 那就得到另一条反例链, 其中步 r 是小的; 但这不可能. 另一方面, 如果 $c \geq 2$, 则 $4 \leq \nu(a_r) \leq \nu(a_{r-1}) + \nu(a_{r-k}) - 2 \leq \nu(a_{r-1})$; 因此 $\nu(a_{r-1}) = 4, \nu(a_{r-k}) = 2$, 且 $c=2$. 通过考虑定理 B 的证明中的六个类型, 很容易推出不可能的情况.

(b) 如果 $\lambda(a_{r-k}) < m-1$, 则我们有 $c \geq 3$, 所以由 (22), $\nu(a_{r-k}) + \nu(a_{r-1}) \geq 7$; 因此 $\nu(a_{r-k})$ 和 $\nu(a_{r-1})$ 两者都大于等于 3. 所有小步都必须小于等于 $r-k$, 而且 $\lambda(a_{r-k}) = m-k+1$. 如果 $k \geq 4$, 则我们必然有 $c=4, k=4, \nu(a_{r-1}) = \nu(a_{r-4}) = 4$; 于是 $a_{r-1} \geq 2^m + 2^{m-1} + 2^{m-2}$, 而且 a_{r-1} 必定等于 $2^m + 2^{m-1} + 2^{m-2} + 2^{m-3}$; 但现在 $a_{r-4} \geq \frac{1}{8} a_{r-1}$ 意味着 $a_{r-1} = 8a_{r-4}$. 于是 $k=3$ 和 $a_{r-1} > 2^m + 2^{m-1}$. 由于 $a_{r-2} < 2^m$ 和 $a_{r-3} < 2^{m-1}$, 步 $r-1$ 必然是倍步; 但步 $r-2$ 不是倍步的, 因为 $a_{r-1} \neq 4a_{r-3}$. 而且, 由于 $\nu(a_{r-3}) \geq 3, r-3$ 是一个星步; 而且 $a_{r-2} = a_{r-3} + a_{r-5}$ 将意味着 $a_{r-5} = 2^{m-2}$, 因此我们必然有 $a_{r-2} = a_{r-3} + a_{r-4}$. 像正文中处理的一个类似情

况那样,现在看来惟一的可能性是 $a_{r-4} = 2^{m-2} + 2^{m+3}$, $a_{r-3} = 2^{m-2} + 2^{m+3} + 2^{d+1} + 2^d$, $a_{r-1} = 2^m + 2^{m-1} + 2^{d+2} + 2^{d+1}$, 而且甚至这点可能性也是不可能的。

15. Achim Flammenkamp [Diplomarbeit in Mathematics (Bielefeld University, 1991), Part 1] 已经证明满足 $\lambda(n) + 3 = l(n) < l^*(n)$ 的数 n 都有 $2^A + 2^B + 2^C + 2^D + 2^E$ 的形式, 其中 $A > B > C > D > E$ 和 $B + E = C + D$; 而且, 通过不匹配下列八个模式的任何一个, 可以精确地描述它们, 这些模式是 $|e| \leq 1$: $2^A + 2^{A-3} + 2^C + 2^{C-1} + 2^{2C+2-A}$, $2^A + 2^{A-1} + 2^C + 2^D + 2^{C+D+1-A}$, $2^A + 2^E + 2^{2B-A+3} + 2^{2B+2-A} + 2^{3B+5-2A}$, $2^A + 2^B + 2^{2B-A+1} + 2^D + 2^{B+D+1-A}$, $2^A + 2^B + 2^{B-1} + 2^D + 2^{D-1}$, $2^A + 2^B + 2^{B-2} + 2^D + 2^{D-2}$ ($A > B + 1$), $2^A + 2^B + 2^C + 2^{2B+C-A} + 2^{B+C+1-A}$, $2^A + 2^B + 2^C + 2^{B+C+1-A} + 2^{2C+1-A}$ 。

16. $l^B(n) = \lambda(n) + \nu(n) - 1$; 所以如果 $n = 2^k$, 则 $l^B(n)/\lambda(n) = 1$, 但如果 $n = 2^{k+1} - 1$, 则 $l^B(n)/\lambda(n) = 2$ 。

17. 设 $i_1 < \dots < i_l$ 。删去不影响并 $I_1 \cup \dots \cup I_l$ 的任何区间 I_k (如果 $j_{k+1} \leq j_k$ 或 $j_1 < j_2 < \dots$ 且 $j_{k+1} \leq j_{k-1}$, 则可抛弃区间 $(j_k, i_k]$)。现在把重叠的区间 $(j_1, i_1], \dots, (j_d, i_d]$ 结合在一起成为一个区间 $(j', i'] = (j_1, i_d]$, 并且注意

$$a_{i'} < a_{j'}(1 + \delta)^{i_1 - j_1 + \dots + i_d - j_d} \leq a_{j'}(1 + \delta)^{2(i' - j')}$$

因为 $(j', i']$ 的每点在 $(j_1, i_1] \cup \dots \cup (j_d, i_d]$ 中至多被覆盖两次。

18. 如果当 $m \rightarrow \infty$ 时, $(\log f(m))/m \rightarrow 0$, 则称 $f(m)$ 为一个“好”的函数。 m 的多项式是好的。好函数的乘积是好的。如果 $g(m) \rightarrow 0$ 且 c 是一个正常数, 则 $c^{mg(m)}$ 是好的; $\binom{2m}{mg(m)}$ 也是好的, 因为由斯特林近似式这等于说, $g(m)\log(1/g(m)) \rightarrow 0$ 。

现在以极大项(通过任何 s, t, v 达到)来代替求和的每一项。这些项的总数是好的, 而且 $\binom{m+s}{t+v}, \binom{t+v}{v} \leq 2^{t+v}$ 和 β^{2v} 也是好的, 因为 $(t+v)/m \rightarrow 0$ 。最后, $\binom{(m+s)^2}{t} \leq (2m)^{2t}/t! < (4em^2/t)^t$, 其中 $(4e)^t$ 是好的; 把 t 以它的上限值 $(1 - \epsilon/2)m/\lambda(m)$ 代替表明 $(m^2/t)^t \leq 2^{m(1-\epsilon/2)} f(m)$, 其中 $f(m)$ 是好的。因此对于很大的 m , 如果 $\alpha = 2^{1-\eta}$, $0 < \eta < \frac{1}{2}\epsilon$, 则整个和小于 α^m 。

19. a) 分别为 $M \cap N, M \cup N, M \oplus N$; 见等式 4.5.2-(6), 4.5.2-(7)。

b) $f(z)g(z), \text{lcm}(f(z), g(z)), \text{gcd}(f(z), g(z))$ 。(出于和 a) 同样的原因, 因为复数上的首一不可约多项式恰是多项式 $z - \zeta$ 。)

c) 交换律 $A \oplus B = B \oplus A, A \cup B = B \cup A, A \cap B = B \cap A$ 。结合律 $A \oplus (B \oplus C) = (A \oplus B) \oplus C, A \cup (B \cup C) = (A \cup B) \cup C, A \cap (B \cap C) = (A \cap B) \cap C$ 。分配律 $A \cup (B \cap C) = (A \cup B) \cap (A \cup C), A \cap (B \cup C) = (A \cap B) \cup (A \cap C), A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C), A \oplus (B \cap C) = (A \oplus B) \cap (A \oplus C)$ 。幂等律 $A \cup A = A, A \cap A = A$ 。吸收律 $A \cup (A \cap B) = A, A \cap (A \cup B) = A, A \cap (A \oplus B) =$

$A, A \cup (A \uplus B) = A \uplus B$ 。恒等律和 0 律 $\emptyset \uplus A = A, \emptyset \cup A = A, \emptyset \cap A = \emptyset$, 其中 \emptyset 是空多重集。计数律 $A \uplus B = (A \cup B) \uplus (A \cap B)$ 。类似于集合的那些更进一步的性质由规则 $A \subseteq B$ 当且仅当 $A \cap B = A$ (当且仅当 $A \cup B = B$) 所定义的偏序给出。

注:多重集合的其它普通的应用是亚纯函数的零点和极点,规范形式下矩阵的不变量,有限阿贝尔群的不变量等等;多重集合在组合的计数论证中和在测度论的发展中是有用的。一个非循环的上下文无关的文法的所有终结串形成一个多重集,它是一个集合当且仅当这个文法是非二义性的。作者在由 J.D.Ullman 主编的 *Theoretical Studies in Computer Science* (Academic Press, 1992, 1~13) 的一篇文章中讨论了对上下文无关文法进一步的应用,并介绍了运算 $A \oslash B$, 其中在 A 中出现 a 次的每个元素和在 B 中出现 b 次的元素在 $A \oslash B$ 中出现 ab 次。

尽管多重集合在数学中经常出现,但由于当前没有标准的方式来处理具有重复元素的集合,因而通常对它们的处理必然略显笨拙。若干数学家认为,对这个普通的概念缺乏适当的术语和符号肯定已经成为数学发展的一个障碍。(当然,一个多重集合形式上等价于从一个集合到非负整数的一个映像,但这个形式上的等价性对于创造性的数学推理实用价值较小,甚至于根本没有实用价值。)自 20 世纪 60 年代起,作者就已经同许多人讨论过此事,以图找出一个好的补救办法。对于这个概念提出的某些名称是:表,串,袋,堆,样本,配权集合,汇集,套件;但这些词或者同现有的术语冲突,有不适当的含义,或者是说来太费口舌而写起来又不方便。最终已经很清楚,这样一个重要的概念本身值得有一个名字, N. G. de Bruijn 便提出了“多重集合”这个术语。他的建议 20 世纪 70 年代以来获得广泛的采用,因而现在它成了标准术语。

作者选择了“ $A \uplus B$ ”的符号以避免同现有的符号冲突,并强调它同集合的并的类似性。为此目的而使用“ $A + B$ ”是不合适的,因为代数学家已经发现 $A + B$ 是对于 $\{\alpha + \beta | \alpha \in A \text{ 和 } \beta \in B\}$ 的一个好记法。如果 A 是非负整数的一个多重集合,则命 $G(z) = \sum_{n \in A} z^n$ 是对应于 A 的一个生成函数(具有非负整系数的生成函数显然同非负整数的多重集合一一对应)。如果 $G(z)$ 对应于 A 且 $H(z)$ 对应于 B , 则 $G(z) + H(z)$ 对应于 $A \uplus B$, $G(z)H(z)$ 对应于 $A + B$ 。如果我们构造 Dirichlet 生成函数 $g(z) = \sum_{n \in A} 1/n^z$, $h(z) = \sum_{n \in B} 1/n^z$, 则乘积 $g(z)h(z)$ 对应于多重集合乘积 AB 。

20. 类型 3: $(S_0, \dots, S_r) = (M_{00}, \dots, M_{r0}) = (\{0\}, \dots, \{A\}, \{A-1, A\}, \{A-1, A, A\}, \{A-1, A-1, A, A, A\}, \dots, \{A+C-3, A+C-3, A+C-2, A+C-2, A+C-2\})$ 。类型 5: $(M_{00}, \dots, M_{r0}) = (\{0\}, \dots, \{A\}, \{A-1, A\}, \dots, \{A+C-1, A+C\}, \{A+C-1, A+C-1, A+C\}, \dots, \{A+C+D-1, A+C+D-1, A+C+D\})$; $(M_{01}, \dots, M_{r1}) = (\emptyset, \dots, \emptyset, \emptyset, \dots, \emptyset, \{A+C-2\}, \dots, \{A+C+D-2\})$, $S_i = M_{i0} \uplus M_{i1}$ 。

21. 例如, 设 $u = 2^{8q+5}$, $x = (2^{(q+1)u} - 1)/(2^u - 1) = 2^{qu} + \dots + 2^u + 1$, $y =$

$2^{(q+1)u} + 1$, 则 $xy = (2^{2(q+1)u} - 1)/(2^u - 1)$ 。如果 $n = 2^{4(q+1)u} + xy$, 则我们由定理 F 有 $l(n) \leq 4(q+1)u + q + 2$, 但由定理 H, $l^*(n) = 4(q+1)u + 2q + 2$ 。

22. 除了用于计算 x 的 $u-1$ 处插入外, 对其余的都画上下划线。

23. 定理 G(所有的都加上下划线)。

24. 使用数 $(B^2 - 1)/(B - 1)$, $0 \leq i \leq r$, 当 a_i 加上下划线时, 这些数也加上下划线; 而且对于 $0 \leq j < t$, $0 < i \leq b_{j-1} - b_j$, $1 \leq k \leq l^0(B)$, 当 c_k 加上下划线时, 数 $c_k B^{i-1} (B^{b_i} - 1)/(B - 1)$ 也加上下划线, 其中 c_0, c_1, \dots 是对于 B 的极小长度的 l^0 链。为证明第二个不等式, 令 $B = 2^m$ 并使用 (3)。(第二个不等式对定理 G 有极少改进, 如果有改进的话。)

25. 我们可以假设 $d_k = 1$ 。利用规则 $RA_{k-1} \cdots A_1$, 其中如果 $d_j = 1$, 则 $A_j = \text{“XR”}$, 否则 $A_j = \text{“R”}$, 而且其中“R”指的是取平方根, “X”指的是乘以 x 。例如, 如果 $y = (.1101101)_2$, 则规则是 R R XR XR R XR XR。(存在适合于计算机硬件的二进平方根开方算法, 需要同除法相当的执行时间; 因此有这种硬件的计算机利用本题的技术, 可以计算更为一般的小数乘方。)

26. 如果我们知道值 (F_k, F_{k-1}) , 则有 $(F_{k+1}, F_k) = (F_k + F_{k-1}, F_k)$ 和 $(F_{2k}, F_{2k-1}) = (F_k^2 + 2F_k F_{k-1}, F_k^2 + F_{k-1}^2)$; 所以利用 $O(\log n)$ 次算术运算, 一个二进方法可用来计算 (F_n, F_{n-1}) 。也许更好的是使用值偶 (F_k, L_k) , 其中 $L_k = F_{k-1} + F_{k+1}$ (参见习题 4.5.4~15); 则我们有 $(F_{k+1}, L_{k+1}) = \left(\frac{1}{2}(F_k + L_k), \frac{1}{2}(5F_k + L_k) \right)$, $(F_{2k}, L_{2k}) = (F_k L_k, L_k^2 - 2(-1)^k)$ 。

对于一般的线性递推式 $x_n = a_1 x_{n-1} + \cdots + a_d x_{n-d}$, 我们可以通过计算一个适当的 $d \times d$ 矩阵的 n 次方, 以 $O(d^3 \log n)$ 次算术运算来计算 x_n [这个发现属于 J. C.P. Miller 和 D.J. Spencer Brown, *Comp. J.* 9 (1966), 188~190]。事实上, 如同 Richard Brent 已经注意到的那样, 如果我们首先计算 $x^n \bmod (x^d - a_1 x^{d-1} - \cdots - a_d)$, 而后以 x_i 代替 x^i , 利用习题 4.7-6, 运算的次数可以减少成 $O(d^2 \log n)$ 或者甚至减少成 $O(d \log d \log n)$ 。

27. 要求 s 个小步的最小的 n 必须是对于某个 r 的 $c(r)$ 。因为如果 $c(r) < n < c(r+1)$, 我们有 $l(n) - \lambda(n) \leq r - \lambda(c(r)) = l(c(r) - \lambda(c(r)))$ 。对于 $1 \leq s \leq 6$, 答案因此是 3, 7, 29, 127, 1903, 65131; 大概 $c(28)$ 将要求 7。

28. a) $x \nabla y = x \vee y \vee (x + y)$, 其中“ \vee ”是逻辑“或”, 参见习题 4.6.2-26; 显然 $\nu(x \nabla y) \leq \nu(x \vee y) + \nu(x \wedge y) = \nu(x) + \nu(y)$ 。b) 首先注意对于 $1 \leq i \leq r$, $A_{i-1}/2^{d_{i-1}} \subseteq A_i/2^{d_i}$ 。其次, 注意在一个非加倍中 $d_j - d_{j-1}$; 因为否则 $a_{j-1} \geq 2a_j \geq a_j + a_k = a_i$ 。因此 $A_j \subseteq A_{i-1}$ 和 $A_k \subseteq A_{i-1}/2^{d_i - d_k}$ 。c) 对 i 的一个容易的归纳法, 只是对结束步需要更仔细注意。如果在 m 的二进表示中所有的 1 全都出现在一行中大于等于 α 的连续的块中, 我们就说 m 有性质 $P(\alpha)$ 。如果 m 和 m' 有 $P(\alpha)$, 则 $m \nabla m'$ 也有 $P(\alpha)$; 如果 m 有 $P(\alpha)$ 则 $\rho(m)$ 有 $P(\alpha + \delta)$ 。因此 B_i 有 $P(1 + \delta c_i)$ 。最后如

果 m 有 $P(a)$, 则 $\nu(\rho(m)) \leq (a + \delta)\nu(m)/a$; 对于 $\nu(m) = \nu_1 + \cdots + \nu_q$, 其中每个块大小 $\nu_j \geq a$, 因此 $\nu(\rho(m)) \leq (\nu_1 + \delta) + \cdots + (\nu_q + \delta) \leq (1 + \delta/a)\nu_1 + \cdots + (1 + \delta/a)\nu_q$. d) 设 $f = b_r + c_r$ 是非加倍的个数, s 是小步数。如果 $f \geq 3.271 \lg \nu(n)$, 则由(16)我们有所求的 $s \geq \lg \nu(n)$ 。否则对于 $0 \leq i \leq r$ 我们有 $a_i \leq (1 + 2^{-\delta})^{b_i 2^i + d_i}$, 因此 $n \leq ((1 + 2^{-\delta})/2)^{b_r} 2^r$, 且 $r \geq \lg n + b_r - b_r \lg(1 + 2^{-\delta}) \geq \lg n + \lg \nu(n) - \lg(1 + \delta c_r) - b_r \lg(1 + 2^{-\delta})$ 。设 $\delta = \lceil \lg(f + 1) \rceil$; 则 $\ln(1 + 2^{-\delta}) \leq \ln(1 + 1/(f + 1)) \leq 1/(f + 1) \leq \delta/(1 + \delta f)$, 而且由此得出, 对于 $0 \leq x \leq f$, $\lg(1 + \delta x) + (f - x)\lg(1 + 2^{-\delta}) \leq \lg(1 + \delta f)$ 。因此最后 $l(n) \geq \lg n + \lg \nu(n) - \lg(1 + (3.271 \lg \nu(n)) \lceil \lg(1 + 3.271 \lg \nu(n)) \rceil)$ 。[Theoretical Comp. Sci. 1 (1975), 1 ~ 12.]

29. 在刚才引证的论文中, Schönage 改进了习题 28 的方法以证明对所有的 n , $l(n) \geq \lg n + \lg \nu(n) - 2.13$ 。剩下的缺口能够弥合吗?

30. $n = 31$ 是最小的例子; $l(31) = 7$, 但 1, 2, 4, 8, 16, 32, 31 是长度为 6 的一个加减法链。[在证明了定理 E 后, Erdős 指出对于加减法链同样的结果也成立。Schönage 已经把习题 28 的下限扩大到加减法链, 并且像在习题 4.1-34 中所定义的那样, 用 $\nu(n)$ 代替 $\nu(n)$ 。基于该习题的表示 a_n , 当同时给出 x 和 x^{-1} 时, 可以给出对于乘幂的推广的自右至左的二进方法, 它使用 $\lambda(n) + \bar{\nu}(n) - 1$ 次乘法。]

32. 见 Discrete Math. 23 (1978), 115 ~ 119。[这一开销模型对应于通过算法 4.31M 那样的经典方法对很大的数做乘法。D. P. McCarthy, Math. Comp. 46 (1986), 603 ~ 608 已经得到其开销为 $(a_j a_k)^{2^{1/2}}$ 的一个更一般的模型的经验结果; 当两个 n 个二进位的数以 $O(n^{\beta})$ 步相乘时, 这个模型更接近于 4.3.3 小节中的“快速乘法”, 但是开销函数 $a_j a_k^{\beta-1}$ 实际上将是更合适的(参见习题 4.3.3-13)。H. Zantema 在步骤 i 的开销是 $a_j + a_k$ 而不是 $a_j a_k$ 时, 已经分析了类似的问题; 参见 J. Algorithms 12 (1991), 281 ~ 307。在这种情况下, 最优链有总开销 $\frac{5}{2}n + O(n^{1/2})$ 。而且

当 n 是奇数时最优的加法开销至少是 $\frac{5}{2}(n - 1)$, 且等于它当且仅当 n 可以写成形如 $2^k + 1$ 的数的一个乘积时。]

33. 八个; 有四种方法计算 $39 = 12 + 12 + 12 + 3$ 和两种方法计算 $79 = 39 + 39 + 1$ 。

34. 这个命题是真的, 在二进链的归约图中的标号, 对于 $k = e_0, \dots, 0$ 是 $\lfloor n/2^k \rfloor$; 在对偶图中它们是 $1, 2, \dots, 2^{e_0}$ 。[类似地, 习题 9 的自右至左的 m 进方法是自左至右方法的对偶。]

35. 2^t 等价于二进链; 如果 $e_0 = e_1 + 1$, 它将是 2^{t-1} 。等价于算法 A 方案的链数是计算 $t + 2$ 个数之和(其中有两个数相同)的方法数。这就是 $\frac{1}{2}f_{t+1} + \frac{1}{2}f_t$, 其中 f_m 是计算 $m + 1$ 个不同数之和的方法数。当我们考虑可交换性时, 我们看到 f_m 是

2^{-m} 乘 $(m+1)!$ 乘 m 个节点的二叉树的个数, 所以 $f_m = (2m-1)(2m-3)\cdots 1$ 。

36. 首先对于使得 $0 \leq e_k \leq 1$ 和 $e_1 + \cdots + e_m \geq 2$ 的所有指数序列, 构造 $2^m - m - 1$ 个乘积 $x_1^{e_1} \cdots x_m^{e_m}$ 。令 $n_k = (d_{k\lambda} \cdots d_{k1} d_{k0})_2$; 为完成计算, 取 $x_1^{d_{1\lambda}} \cdots x_m^{d_{m\lambda}}$, 然后平方之, 而且对于 $i = \lambda-1, \cdots, 1, 0$ 乘之以 $x_1^{d_{1i}} \cdots x_m^{d_{mi}}$ 。[Straus 在 AMM 71 (1964), 807~808 中证明, 通过像在定理 D 中那样, 把这个二进方法推广到 2^k 进制中, 对于任何 $\epsilon > 0$, $2\lambda(n)$ 可以用 $(1+\epsilon)\lambda(n)$ 来代替。]

37. 首先对于 $1 \leq q \leq \lambda(n_m)$ 计算 2^q , 然后用 2^k 进的方法的下列变形, 计算每个 $n = n_j$: 对于所有奇的 $q < 2^k$, 在至多 $\lfloor \frac{1}{k} \lg n \rfloor$ 步内, 计算 $f_q = \sum |2^{k+e} d_i = 2^q|$, 其中 $n = (\cdots d_1 d_0)_2$; 然后在顶多另外的 $\sum l(q) + 2^{k-1}$ 步内计算 $n \approx \sum q f_q$, 每个 n_j 所需的步数 $\leq \lfloor \frac{1}{k} \lg n \rfloor + O(k2^k)$, 而且当 $k = \lfloor \lg \lg n - 3 \lg \lg \lg n \rfloor$ 时, 这是 $\lambda(n)/\lambda\lambda(n) + O(\lambda(n)\lambda\lambda\lambda(n)/\lambda\lambda(n)^2)$ 。

[定理 E 的一个推广给出对应的下限。参考: SICOMP 5 (1976), 100~103。]

38. 下面由 D. J. Newman 给出的构造提供了最近知道的最好上限: 设 $k = p_1 \cdots p_r$ 是头 r 个素数的乘积。在 $O(2^{-r} k \log k)$ 步内计算 k 和所有二次剩余模 k (因为有近 $2^{-r} k$ 个二次剩余)。并且在大约另外的 m^2/k 个步内, 计算小于等于 m^2 的 k 的所有倍数。现在 m 个加法足以计算 $1^2, 2^2, \cdots, m^2$ 。我们有 $k = \exp(p_r + O(p_r/(\log p_r)^{1000}))$, 其中 p_r 由习题 4.5.4-36 中的公式给出; 例如, 参见 Greene 和 Knuth, *Math. for the Analysis of Algorithms* (Boston: Birkhäuser, 1981), § 4.1.6。所以通过选择

$$r = \left\lfloor \left(1 + \frac{1}{2} \ln 2 / \lg \lg m \right) \ln m / \ln \ln m \right\rfloor$$

即得出 $l(1^2, \cdots, m^2) = m + O\left(m \cdot \exp\left(-\left(\frac{1}{2} \ln 2 - \epsilon\right) \ln m / \ln \ln m\right)\right)$ 。

另一方面, D. Dobkin 和 R. Lipton 已经证明, 对任何 $\epsilon > 0$, 当 n 充分大时, $l(1^2, \cdots, m^2) > m + m^{2/3-\epsilon}$ [SICOMP 9 (1980), 121~125]。

39. 量 $l([n_1, n_2, \cdots, n_m])$ 是有向边数 - 顶点数 + m 中之极小者, 这里取遍所有如下的有向图, 它有 m 个入度为 0 的顶点 s_j 及一个出度为 0 的顶点 t , 其中对于 $1 \leq j \leq m$, 恰有 n_j 条由 s_j 到 t 的有向通路。量 $l(n_1, n_2, \cdots, n_m)$ 是有向边数 - 顶点数 + 1 中之极小者, 取遍有一个入度为 0 的顶点 s 和 m 个出度为 0 的顶点 t_j 的所有有向图, 其中对于 $1 \leq j \leq m$ 恰有 n_j 条从 s 到 t_j 的有向通路。如果我们改变所有有向边的方向, 则这些问题是彼此对偶的。[参考 J. Algorithms 2 (1981), 13~21。]

注: C. H. Papadimitriou 指出, 这是一个更为一般得多的定理的特殊情况。设 N

$= (n_{ij})$ 是没有全为 0 的行或列的非负整数的 $m \times p$ 矩阵。我们可以定义 $l(N)$ 为计算单项式集合 $\{x_1^{n_1} \cdots x_m^{n_m} \mid 1 \leq j \leq p\}$ 所需的最小乘法数。现在 $l(N)$ 也是有向边数 - 顶点数 + m 中极小者, 取遍 m 个入度是 0 的顶点 s_i 和 p 个出度为 0 的顶点 t_j 的所有有向图, 其中对于每个 i 和 j 恰有 n_{ij} 条从 s_i 到 t_j 的有向通路。由对偶性, 我们有 $l(N) = l(N^T) + m - p$ 。[*Bulletin of the Europ. Assoc. Theor. Comp. Sci.* 13 (1981 年 2 月), 2~3。]

N. Pippenger 已经证明了习题 36 和 37 的结果的一个广泛推广。设 $L(m, p, n)$ 是取遍非负整数 $n_{ij} \leq n$ 的所有 $m \times p$ 矩阵 N 的 $l(N)$ 的极大值。则 $L(m, p, n) = \min(m, p) \lg n + H! \lg H + O(m + p + H(\log \log H)^{1/2} (\log H)^{-3/2})$, 其中 $H = mp \lg(n+1)$ 。[*SICOMP* 9 (1980), 230~250。]

40. 由习题 39, 证明 $l(m_1 n_1 + \cdots + m_t n_t) \leq l(m_1, \cdots, m_t) + l([n_1, \cdots, n_t])$ 即可。但这是显然的, 因为我们可以首先构造 $\{x^{m_1}, \cdots, x^{m_t}\}$, 而后计算单项式 $(x^{m_1})^{n_1} \cdots (x^{m_t})^{n_t}$ 。

注: 表述 Olivos 定理的一个更强的方法是: 如果 a_0, \cdots, a_r 和 b_0, \cdots, b_s 是任意加法链, 则对于非负整数 c_{ij} 的任何 $(r+1) \times (s+1)$ 的矩阵, $l(\sum c_{ij} a_i b_j) \leq r + s + \sum c_{ij} - 1$ 。

41. [*SICOMP* 10 (1981), 638~646。] 每当 $A \geq 9m^2$ 时, 可以证明所述的公式。由于这是 m 的多项式, 而且由于求一个极小的顶点覆盖的问题是 NP 困难的 (参见 7.9 节), 故计算 $l(n_1, \cdots, n_m)$ 的问题是 NP 完全的。[计算 $l(n)$ 的问题是否 NP 完全尚不得知。但这样一点看起来是真的, 即当 A 充分大时, 对于比如说 $\sum_{k=0}^{m-1} n_{k+1} 2^{Ak^2}$ 的最优的链将要求对于 $\{n_1, \cdots, n_m\}$ 的最优的链。]

4.6.4 小节

1. 置 $y \leftarrow x^2$, 然后计算 $((\cdots (u_{2r+1}y + u_{2n-1})y + \cdots)y + u_1)x$ 。

2. 用多项式 $x + x_0$ 代替 (2) 中的 x 即得下列过程:

G1. 对于 $k = n, n-1, \cdots, 0$ (以这个顺序) 执行 G2, 并停止。

G2. 置 $v_k \leftarrow u_k$, 而后对于 $j = k, k+1, \cdots, n-1$ 置 $v_j \leftarrow v_j + x_0 v_{j+1}$ (当 $k = n$ 时, 这一步只是置 $v_n \leftarrow u_n$)。 ▮

这些计算同 H1 和 H2 中的那些计算刚好相同, 但是以不同的顺序执行。(这个应用事实上是牛顿原来使用方案 (2) 的动机。)

3. x^k 的系数是 y 的多项式, 它可以通过霍纳规则来求值: $((\cdots (u_{n,0}x + (u_{n-1,1}y + u_{n-1,0}))x + \cdots)x + ((\cdots (u_{0,n}y + u_{0,n-1})y + \cdots)y + u_{0,0}))$ 。[对于一个“齐次”多项式, 诸如 $u_n x^n + u_{n-1} x^{n-1} y + \cdots + u_1 x y^{n-1} + u_0 y^n$, 还有一个更有效的方案: 如果 $0 < |x| \leq |y|$, 首先 x 除以 y , 求 x/y 的多项式的值然后乘以 y^n 。]

4. 规则 (2) 包含 $4n$ 或 $3n$ 个实数乘法及 $4n$ 或 $7n$ 个实数加法; (3) 是坏的, 它花费 $4n+2$ 或 $4n+1$ 个乘法, $4n+2$ 或 $4n+5$ 个加法。

5. 用一个乘法计算 x^2 ; $\lfloor n/2 \rfloor$ 个乘法和 $\lfloor n/2 \rfloor$ 个加法计算第一行; $\lceil n/2 \rceil$ 个乘法和 $\lceil n/2 \rceil - 1$ 个加法计算第二行; 一个加法把两行加在一起。总共 $n+1$ 个乘法和 n 个加法。

6. J1. 计算和存储值 $x_0^2, x_0^3, \dots, x_0^{\lfloor n/2 \rfloor}$ 。

J2. 对于 $0 \leq j \leq n$ 置 $v_j \leftarrow u_j x_0^{\lfloor n/2 \rfloor - j}$ 。

J3. 对于 $k=0, 1, \dots, n-1$, 对 $j=n-1, \dots, k+1, k$, 置 $v_j \leftarrow v_j + v_{j+1}$ 。

J4. 对于 $0 \leq j \leq n$, 置 $v_j \leftarrow v_j x_0^{\lfloor n/2 \rfloor - j}$ 。 ■

有 $(n^2+n)/2$ 个加法, $n + \lceil n/2 \rceil - 1$ 个乘法, n 个除法。通过把 v_n 和 v_0 当做特殊情况, 可以节省另一个乘法和除法。参考文献: SIGACT News 7, 3(1975 年夏), 32~34。

7. 设 $x_j = x_0 + jh$, 并考虑(42)和(44)。对于 $0 \leq j \leq n$, 置 $y_j \leftarrow u(x_j)$ 。对于 $k=1, 2, \dots, n$ (以这个顺序), 对于 $j=n, n-1, \dots, k$ (以这个顺序) 置 $y_j \leftarrow y_j - y_{j-1}$ 。现在对所有 j , 置 $\beta_j \leftarrow y_j$ 。

然而, 即使(5)的运算是以完全的精度完成的, 但舍入误差仍会像在正文中所说明的那样积累。当(5)是以浮点算术进行时, 进行初始化的一个更好的方式是选择 β_0, \dots, β_n 使得

$$\begin{bmatrix} \binom{0}{0} & \binom{0}{1} & \cdots & \binom{0}{n} \\ \binom{d}{0} & \binom{d}{1} & \cdots & \binom{d}{n} \\ \vdots & \vdots & & \vdots \\ \binom{nd}{0} & \binom{nd}{1} & \cdots & \binom{nd}{n} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} = \begin{bmatrix} u(x_0) \\ u(x_d) \\ \vdots \\ u(x_{nd}) \end{bmatrix} + \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

其中 $|\epsilon_0|, |\epsilon_1|, \dots, |\epsilon_n|$ 尽可能地小。[H. Hassler, Proc. 12th. Spring Conf. Computer Graphics (Bratislava: Comenius University, 1996), 55~66。]

8. 见(43)。

9. [Combinatorial Mathematics (Buffalo: Math. Assoc. of America, 1963), 26~28。] 这个公式可以看做容斥原理(1.3.3 小节)的一个应用, 因为对于 $n - \epsilon_1 - \dots - \epsilon_n = k$ 的诸项的和是 j_i 的 k 个值不出现的所有那些 $x_{1j_1} x_{2j_2} \cdots x_{nj_n}$ 的和。注意到 $x_{1j_1} \cdots x_{nj_n}$ 的系数是

$$\sum (-1)^{n - \epsilon_1 - \dots - \epsilon_n} \epsilon_{j_1} \cdots \epsilon_{j_n}$$

可以给出一个直接的证明; 如果诸 j 是不同的, 则这等于 1, 但如果 $j_1, \dots, j_n \neq k$, 则它为 0, 因为 $\epsilon_k = 0$ 的项抵消 $\epsilon_k = 1$ 的项。

为了有效地求和, 我们可以从 $\epsilon_1 = 1, \epsilon_2 = \dots = \epsilon_n = 0$ 开始, 然后以下列方式遍历诸 ϵ 的所有可能的组合, 即从一个项进到下一个项时, 仅有一个 ϵ 发生变化(见第 7

章中的“Gray 码”)。计算头一项的工作量是 $n-1$ 个乘法; 随继的 2^n-2 项每一项需用 n 个加法, 再用 $n-1$ 个乘法, 然后又一个加法。总共 $(2^n-1)(n-1)$ 个乘法及 $(2^n-2)(n+1)$ 个加法。仅仅需要 $n+1$ 个临时存储单元, 一个用于主要部分的和, 其余的用于当前乘积的因子(每个因子用一个)。

10. $\sum_{1 \leq k < n} (k+1) \binom{n}{k+1} = n(2^{n-1} - 1)$ 个乘法和 $\sum_{1 \leq k < n} k \binom{n}{k+1} = n2^{n-1} - 2^n + 1$ 个加法。这近似于习题 9 方法的算术运算的一半那样多, 尽管它要求更复杂的程序以控制执行顺序。必须使用近似于 $\binom{n}{\lceil n/2 \rceil} + \binom{n}{\lceil n/2 \rceil - 1}$ 个临时存储单元, 而且这将以指数的方式增大(阶为 $2^n/\sqrt{n}$)。

本题中的方法等价于由 Jurkat 和 Ryser 在 *J. Algebra* **5** (1967), 342~357 中给出的积和式函数的非通常的矩阵因子分解。在一种适当意义下, 它也可以看做(39)和(40)的一个应用。

11. 如果矩阵是充分稠密的, 则计算一个适当的值的有效方法是已知的, 请参见 A. Sinclair, *Algorithms for Random Generation and Counting* (Boston: Birkhäuser, 1993)。但这个问题要求精确的值。对于某个 $c < 2$ 可能有一个方法以 $O(c^n)$ 次运算来计算积和式。

12. 下边是关于这一著名的研究问题的进展的简要概述: J. Hopcroft 和 L. R. Kerr 在其它场合证明了, 在 2×2 矩阵的乘法中, 需要 7 个模 2 乘法 [*SIAM J. Appl. Math.* **20** (1971), 30~36]。R. L. Probert 证明, 所有 7 个乘法的方案中必须至少有 15 个加法, 在这种方案中每个乘法取一个矩阵的元素的线性组合并乘以另一个矩阵的元素的线性组合 [*SICOMP* **5** (1976), 187~203]。在每个域中 2×2 矩阵乘法的张量的秩是 7 [V. Y. Pan, *J. Algorithms* **2** (1981), 301~310]; 2×3 矩阵和 3×2 矩阵的乘积的张量, 即 $T(2, 3, 2)$ 的张量, 其秩是 11 [V. B. Alekseyev, *J. Algorithms* **6** (1985), 71~85]。当 $n=3$ 时, 对于 $n \times n$ 矩阵乘法, 已知的最好方法是 J. D. Laderman 给出的 [*Bull. Amer. Math. Soc.* **82** (1976), 126~128], 他证明 23 个非可交换的乘法就足够了。他的构造已为 Ondrej Šýkora 所推广, Ondrej Šýkora 给出了需要 $n^3 - (n-1)^2$ 个非可交换的乘法和 $n^3 - n^2 + 11(n-1)^2$ 个加法的方法, 这个结果当 $n=2$ 时可归结为 (36) [*Lecture Notes in Comp. Sci.* **53** (1977), 504~512]。对于 $n=5$, 当前的记录是 100 个非交换的乘法 [O. M. Makarovsk, *USSR Comp. Math. and Math. Phys.* **27**, 1 (1987), 205~207]。迄今为止已知的最好下限是由 J.-C. Lafon 和 S. Winograd 给出的, 他们证明, $2n^2-1$ 次非纯量的乘法是必须的, 而且在 $m \times n \times s$ 的情况下是 $mn + ns + m + n - 1$ [A Lower bound for the multiplicative Complexity of the product of two matrices, Centre de Calcul, Univ. Louis Pasteur (Strasbourg, 1979)]。如果所有的计算都须不用除法进行, N. H. Bshouty [*SICOMP* **18** (1989), 759~765] 得到稍微更好的下限, 他证明当 $n \geq s \geq j \geq 1$ 时, $m \times n$ 乘以 $n \times s$ 的矩阵乘法 mod 2 至少需要 $\sum_{k=0}^{j-1} \lfloor ms/2^k \rfloor + \frac{1}{2}(n + (n \bmod j))(n - (n \bmod j) - j) +$

$n \bmod j$ 次乘法;置 $m = n = s$ 和 $j \approx \lg n$ 给出 $2.5n^2 - \frac{1}{2}n \lg n + O(n)$ 。

关于对于很大的 n 已知的最好的上限,在(36)之后的正文中作了讨论。

13. 通过对几何级数进行求和,我们求得 $F(t_1, \dots, t_n)$ 等于

$$\sum_{0 \leq s_1 < m_1, \dots, 0 \leq s_n < m_n} \exp(-2\pi i(s_1 t_1 / m_1 + \dots + s_n t_n / m_n)) f(s_1, \dots, s_n) / m_1 \dots m_n$$

通过进行一个正则变换,而且当 $t_j \neq 0$ 时交换 t_j 和 $m_j - t_j$,可以求得逆变换乘以 $m_1 \dots m_n$;参看习题 4.3.3-9。

[如果我们把 $F(t_1, \dots, t_n)$ 当做多变量多项式中 $x'_1 \dots x'_n$ 的系数,则离散傅里叶变换相当于求这个多项式在诸单位根处的值,而逆变换相当于求内插多项式。]

14. 设 $m_1 = \dots = m_n = 2$, $F(t_1, t_2, \dots, t_n) = F(2^{n-1}t_n + \dots + 2t_2 + t_1)$, $f(s_1, s_2, \dots, s_n) = f(2^{n-1}s_1 + 2^{n-2}s_2 + \dots + s_n)$;注意诸 t 和诸 s 之间的反演处理。并设 $g_k(s_k, \dots, s_n, t_k)$ 是 ω 取 $2^{k-1}t_k(s_n + 2s_{n-1} + \dots + 2^{n-k}s_k)$ 次方。

在每次迭代时,我们实际上取 2^{n-1} 对复数 (α, β) ,并用 $(\alpha + \zeta\beta, \alpha - \zeta\beta)$ 代替它们,其中 ζ 是 ω 的适当的幂,因此对某个 θ , $\zeta = \cos \theta + i \sin \theta$ 。如果我们在 $\zeta = \pm 1$ 或 $\pm i$ 时利用化简,则全部工作就成为 $((n-3) \cdot 2^{n-1} + 2)$ 次复数乘法和 $n \cdot 2^n$ 次复数加法;习题 41 所用的一些技术可用来减少为实现这些复数运算所需要的实数乘法和加法。

对于 $k=1, 3, \dots$,把第 k 次和第 $k+1$ 次扫描结合起来,可以把复数乘法的次数减少大约 25%,而不改变加法的次数;这意味着 2^{n-2} 个四元组 $(\alpha, \beta, \gamma, \delta)$ 为 $(\alpha + \zeta\beta + \zeta^2\gamma + \zeta^3\delta, \alpha + i\zeta\beta - \zeta^2\gamma - i\zeta^3\delta, \alpha - \zeta\beta + \zeta^2\gamma - \zeta^3\delta, \alpha - i\zeta\beta - \zeta^2\gamma + i\zeta^3\delta)$ 所代替。因此当 n 为偶数时复数乘法的次数减少为 $(3n-2)2^{n-3} - 5\lfloor 2^{n-1}/3 \rfloor$ 。

这些计算假定,给定的数 $F(t)$ 是复数。如果 $F(t)$ 是实数,则 $f(s)$ 是 $f(2^n - s)$ 的复共轭,所以通过仅计算 2^n 个独立的实数 $f(0), \Re f(1), \dots, \Re f(2^{n-1}-1), f(2^{n-1}), \Im f(1), \dots, \Im f(2^{n-1}-1)$,我们可以避免冗余性。在这种情况下整个计算可以通过对 2^n 个实数进行运算,并利用当 $(s_1 \dots s_n)_2 + (s'_1 \dots s'_n)_2 \equiv 0 \pmod{2^n}$ 时 $f_k(s_{n-k+1}, \dots, s_n, t_1, \dots, t_{n-k})$ 将是 $f_k(s'_{n-k+1}, \dots, s'_n, t_1, \dots, t_{n-k})$ 的复共轭这一事实来完成。大约需要在复数情况下的乘法和加法次数的一半。

[快速傅里叶变换的算法是由 C. F. Gauss 于 1805 年发现的,而且自此以后,已许多次被独立地重新发现,最著名的是 J. W. Cooley 和 J. W. Tukey 的发现, *Math. Comp.* **19** (1965), 297~301。它的有趣历史已由 J. W. Cooley, P. A. W. Lewis 以及 P. D. Welch, *Proc. IEEE* **55** (1967), 1675~1677; M. T. Heideman, D. H. Johnson 以及 C. S. Burrus, *IEEE ASSP Magazine* **1**, 4 (1984 年 10 月), 14~21 作了回顾。有关对于它的使用的细节已由数百个作者作了讨论,而由 Charles Van Lon 作了精彩概述, *Computational Frameworks for the Fast Fourier Transform* (Philadelphia: SIAM, 1992)。关于有限群上的快速傅里叶变换的概述,请参见 M. Clausen 和 U. Baum, *Fast Fourier Transforms* (Mannheim: Bibliographisches Institute Wissenschaftsverlag,

1993)。

15. (a) 提示由积分和归纳得出。设当 θ 由 $\min(x_0, \dots, x_n)$ 变化到 $\max(x_0, \dots, x_n)$ 时, $f^{(n)}(\theta)$ 取 A 和 B 之间(含)的所有值。在所述积分中, 用这些界的每一个代替 $f^{(n)}$, 产生 $A/n! \leq f(x_0, \dots, x_n) \leq B/n!$ 。(b) 只须对于 $j = n$ 来证明这一点就够了。设 f 是牛顿内插多项式, 则 $f^{(n)}$ 是常数 $n! \alpha_n$ 。[参见由 D. T. Whiteside 编辑的 *The Mathematical Papers of Isaac Newton*, 4 (1971), 36~51, 70~73。]

16. 如同对多项式的运算那样进行(43)的乘法和加法。(习题2中考虑了 $x_0 = x_1 = \dots = x_n$ 的特殊情况。在算法 4.3.3T 的步骤 T8 中我们已经使用了这一方法。)

17. 例如, 当 $n = 5$ 时, 我们有

$$u_{[5]}(x) = \frac{\frac{y_0}{x-x_0} - \frac{5y_1}{x-x_1} + \frac{10y_2}{x-x_2} - \frac{10y_3}{x-x_3} + \frac{5y_4}{x-x_4} - \frac{y_5}{x-x_5}}{\frac{1}{x-x_0} - \frac{5}{x-x_1} + \frac{10}{x-x_2} - \frac{10}{x-x_3} + \frac{5}{x-x_4} - \frac{1}{x-x_5}}$$

独立于 h 的值。

$$18. \alpha_0 = \frac{1}{2}(u_3/u_4 + 1), \beta = u_2/u_4 - \alpha_0(\alpha_0 - 1), \alpha_1 = \alpha_0\beta - u_1/u_4, \alpha_2 = \beta - 2\alpha_1,$$

$$\alpha_3 = u_0/u_4 - \alpha_1(\alpha_1 + \alpha_2), \alpha_4 = u_4.$$

19. 由于 α_5 是前导系数, 不失一般性我们可以假定 $u(x)$ 是首一的(即 $u_5 = 1$)。于是 α_0 是方程 $40z^3 - 24u_4z^2 + (4u_4^2 + 2u_3)z + (u_2 - u_3u_4) = 0$ 的根; 这个方程总是至少有一个实根, 而且还可能有三个实根。一旦确定了 α_0 , 我们就有 $\alpha_3 = u_4 - 4\alpha_0, \alpha_1 = u_3 - 4\alpha_0\alpha_3 - 6\alpha_0^2, \alpha_2 = u_1 - \alpha_0(\alpha_0\alpha_1 + 4\alpha_0^2\alpha_3 + 2\alpha_1\alpha_3 + \alpha_0^3), \alpha_4 = u_0 - \alpha_3(\alpha_0^4 + \alpha_1\alpha_0^2 + \alpha_2)$ 。

对于给定的多项式, 我们来解三次方程 $40z^3 - 120z^2 + 80z = 0$; 得到三个解 $(\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5) = (0, -10, 13, 5, -5, 1), (1, -20, 68, 1, 11, 1), (2, -10, 13, -3, 27, 1)$ 。

$$\begin{array}{llllll} 20. & \text{LDA} & X & & \text{STA} & \text{TEMP2} & \text{FADD} & = \alpha_1 = & \text{FMUL} & \text{TEMP1} \\ & \text{FADD} & = \alpha_3 = & & \text{FMUL} & \text{TEMP2} & \text{FMUL} & \text{TEMP2} & \text{FADD} & = \alpha_4 = \\ & \text{STA} & \text{TEMP1} & & \text{STA} & \text{TEMP2} & \text{FADD} & = \alpha_2 = & \text{FMUL} & = \alpha_5 = \\ & \text{FADD} & = \alpha_0 - \alpha_3 = & & & & & & & \end{array}$$

21. $z = (x+1)x - 2, w = (x+5)z + 9, u(x) = (w+z-8)w - 8$; 或者 $z = (x+9)x + 26, w = (x-3)z + 73, u(x) = (w+z-24)w - 12$ 。

22. $\alpha_6 = 1, \alpha_0 = -1, \alpha_1 = 1, \beta_1 = -2, \beta_2 = -2, \beta_3 = -2, \beta_4 = 1, \alpha_3 = -4, \alpha_2 = 0, \alpha_4 = 4, \alpha_5 = -2$ 。我们构造 $z = (x-1)x + 1, w = z + x$, 以及 $u(x) = ((z-x-4) \cdot w + 4)z - 2$ 。在这种特殊情况下, 我们看到, 如果我们计算 $w = x^2 + 1, z = w - x$, 则可以节省 7 次加法之一。

23. a) 我们可以对 n 用归纳法; 如果 $n < 2$, 则结果是不重要的。如果 $f(0) = 0$, 则这个结果对于多项式 $f(z)/z$ 为真, 所以它对 $f(z)$ 成立。如果对于某个实数 $y \neq 0$, $f(iy) = 0$, 则 $g(\pm iy) = h(\pm iy) = 0$; 由于此结果对 $f(z)/(z^2 + y^2)$ 为真, 它对 $f(z)$ 也成立。因此我们可以假设 $f(z)$ 没有其实部为零的根。而给定通路绕原点循环的净次数是 $f(z)$ 在这区域内根的个数, 故它至多为 1。当 R 很大时, 对于 $\pi/2 \leq t \leq 3\pi/2$ 的通路 $f(Re^{it})$ 将以顺时针的方向绕原点近似于 $n/2$ 次; 所以 $-R \leq t \leq R$, 通路 $f(it)$ 必然是以反时针方向绕原点至少 $n/2 - 1$ 次。对于 n 是偶数, 这意味着 $f(it)$ 至少穿过虚轴 $n - 2$ 次, 至少穿过实轴 $n - 3$ 次; 对于 n 为奇数, $f(it)$ 至少穿过实轴 $n - 2$ 次, 至少穿过虚轴 $n - 3$ 次。这些分别是 $g(it) = 0, h(it) = 0$ 的根。

b) 若否, 则 g 或 h 将有形如 $a + bi$ 的一个根, 且 $a \neq 0$ 和 $b \neq 0$ 。但这将意味着至少存在三个其它这样的根, 即 $a - bi$, 和 $-a \pm bi$, 而 $g(z)$ 和 $h(z)$ 至多有 n 个根。

24. u 的根是 $-7, -3 \pm i, -2 \pm i, -1$; 可以允许的 c 值是 2 和 4 (但 3 不是, 因为 $c = 3$ 使得根的和等于 0)。情况 1, $c = 2$: $p(x) = (x + 5)(x^2 + 2x + 2)(x^2 + 1)(x - 1) = x^6 + 6x^5 + 6x^4 + 4x^3 - 5x^2 - 2x - 10$; $q(x) = 6x^2 + 4x - 2 = 6(x + 1) \cdot \left(x - \frac{1}{3}\right)$ 。

设 $\alpha_2 = -1, \alpha_1 = \frac{1}{3}$; $p_1(x) = x^4 + 6x^3 + 5x^2 - 2x - 10 = \left(x^2 + 6x + \frac{16}{3}\right)\left(x^2 - \frac{1}{3}\right) - \frac{74}{9}$; $\alpha_0 = 6, \beta_0 = \frac{16}{3}, \beta_1 = -\frac{74}{9}$ 。情况 2, $c = 4$: 类似的分析给出 $\alpha_2 = 9, \alpha_1 = -3, \alpha_0 = -6, \beta_0 = 12, \beta_1 = -26$ 。

25. $\beta_1 = \alpha_2, \beta_2 = 2\alpha_1, \beta_3 = \alpha_7, \beta_4 = \alpha_6, \beta_5 = \beta_6 = 0, \beta_7 = \alpha_1, \beta_8 = 0, \beta_9 = 2\alpha_1 - \alpha_8$ 。

26. (a) $\lambda_1 = \alpha_1 \times \lambda_0, \lambda_2 = \alpha_2 + \lambda_1, \lambda_3 = \lambda_2 \times \lambda_0, \lambda_4 = \alpha_3 + \lambda_3, \lambda_5 = \lambda_4 \times \lambda_0, \lambda_6 = \alpha_4 + \lambda_5$ 。(b) $\kappa_1 = 1 + \beta_1 x, \kappa_2 = 1 + \beta_2 \kappa_1 x, \kappa_3 = 1 + \beta_3 \kappa_2 x, u(x) = \beta_4 \kappa_3 = \beta_1 \beta_2 \beta_3 \beta_4 x^3 + \beta_2 \beta_3 \beta_4 x^2 + \beta_3 \beta_4 x + \beta_4$ 。(c) 如果任何系数为 0, 则在 (b) 中 x^3 的系数也必为 0, 而 (a) 就产生了次数 ≤ 3 的任意多项式 $\alpha_1 x^3 + \alpha_2 x^2 + \alpha_3 x + \alpha_4$ 。

27. 否则对所有实数的集合 (q_n, \dots, q_1, q_0) 将有整系数的非 0 多项式 $f(q_n, \dots, q_1, q_0)$ 使得 $q_n \cdot f(q_n, \dots, q_1, q_0) = 0$ 。但这不能发生, 因为由对 n 的归纳法容易证明, 一个非 0 的多项式总是取某个非 0 的值。(然而, 参考习题 4.6.1-16, 这结果对于有限域而不是实数说来, 是不成立的。)

28. 中间量 $\alpha_1, \dots, \alpha_s$ 形成多项式整环 $Q[\alpha_1, \dots, \alpha_s]$ 的代数基, 其中 Q 是有理数域。因为 $s + 1$ 大于在一个基中的元素个数, 故多项式 $f_j(\alpha_1, \dots, \alpha_s)$ 代数相关; 这意味着有一个有理系数的非 0 多项式 g 使得 $g(f_0(\alpha_1, \dots, \alpha_s), \dots, f_s(\alpha_1, \dots, \alpha_s))$ 恒等于 0。

29. 给定 $j_0, \dots, j_t \in \{0, 1, \dots, n\}$, 存在整系数的非 0 多项式, 使得对于 $R_j (1 \leq j \leq m)$ 中的所有 (q_n, \dots, q_0) , 都有 $g_j(q_{j_0}, \dots, q_{j_t}) = 0$ 。因此对于 $R_1 \cup \dots \cup R_m$ 中所有的 (q_n, \dots, q_0) , 乘积 $g_1 g_2 \dots g_m$ 都为 0。

30. 由定理 M 的构造开始, 我们将证明有 $m_p + (1 - \delta_{0m_i})$ 个 β 可以有效地消去: 如果 μ_i 对应于一个参数乘法, 则我们有 $\mu_i = \beta_{2i-1} \times (T_{2i} + \beta_{2i})$; 把 $c\beta_{2i-1}\beta_{2i}$ 加到 $c\mu_i$ 在 T_j 中出现的每个 β_j 上, 并以 0 代替 β_{2i} 。这对每个参数乘法删去一个参数。如果 μ_i 是头一个链乘法, 则 $\mu_i = (\gamma_1 x + \theta_1 + \beta_{2i-1}) \times (\gamma_2 x + \theta_2 + \beta_{2i})$, 其中 $\gamma_1, \gamma_2, \theta_1, \theta_2$ 是整系数的 $\beta_1, \dots, \beta_{2i-2}$ 的多项式。这里 θ_1 和 θ_2 分别被“吸收”到 β_{2i-1} 和 β_{2i} 中, 所以我们可以假定 $\theta_1 = \theta_2 = 0$ 。现在把 $c\beta_{2i-1}\beta_{2i}$ 加到 $c\mu_i$ 在 T_j 中出现的每个 β_j 上; 把 $\beta_{2i-1}\gamma_2/\gamma_1$ 加到 β_{2i} 上; 并置 β_{2i-1} 成 0。除使得 γ_1 为 0 的 $\alpha_1, \dots, \alpha_s$ 的那些值之外, 消去 β_{2i-1} 不会改变结果集。[这个证明实质上是 V. Y. Pan 给出的, 见 *Uspekhi Mat. Nauk* 21, 1 (1996 年 1 月—2 月), 103~134。]后一种情况可以像在定理 A 的证明中那样处理, 因为具有 $\gamma_1 = 0$ 的多项式可通过消去 β_{2i} 来求值 (像在头一个构造中那样, 其中 μ_i 对应于一个参数乘法)。

31. 否则我们可以增加一个参数乘法作为最后一步, 因而违背定理 C。(在这种特殊情况下, 本题是对定理 A 的改进, 因为在 n 次首一多项式的系数中仅有 n 个自由度。)

32. $\lambda_1 = \lambda_0 \times \lambda_0, \lambda_2 = \alpha_1 \times \lambda_1, \lambda_3 = \alpha_2 + \lambda_2, \lambda_4 = \lambda_3 \times \lambda_1, \lambda_5 = \alpha_3 + \lambda_4$ 。由定理 A, 为计算 $u_4 x^4$ 我们至少需要三个乘法 (见 4.6.3 小节), 并且至少需要两次加法。

33. 我们必须有 $n+1 \leq 2m_c + m_p + \delta_{0m_i}$, 以及 $m_c + m_p = (n+1)/2$; 所以没有参数乘法。现在其前导系数 (作为 x 的一个多项式) 不是整数的头一个 λ_i , 必须通过一个链加法才得到; 而且至少必定有 $n+1$ 个参数, 所以至少有 $n+1$ 个参数加法。

34. 逐步变换给定的链, 还定义 λ_i 的“容度” c_i 如下: (直观上, c_i 是 λ_i 的前导系数。) 定义 $c_0 = 1$ 。(a) 如果此步有 $\lambda_i = \alpha_j + \lambda_k$ 的形式, 则以 $\lambda_i = \beta_j + \lambda_k$ 代替它, 其中 $\beta_j = \alpha_j/c_k$; 并且定义 $c_i = c_k$ 。(b) 如果此步有 $\lambda_i = \alpha_j - \lambda_k$ 的形式, 则以 $\lambda_i = \beta_j + \lambda_k$ 代替它, 其中 $\beta_j = -\alpha_j/c_k$; 而且定义 $c_i = -c_k$ 。(c) 如果此步有 $\lambda_i = \alpha_j \times \lambda_k$ 的形式, 则以 $\lambda_i = \lambda_k$ 代替它 (此步稍后将被删去); 并定义 $c_i = \alpha_j c_k$ 。(d) 如果此步有 $\lambda_i = \lambda_j \times \lambda_k$ 的形式, 则使它保持不变, 并定义 $c_i = c_j c_k$ 。

在这个过程完成以后, 删去形如 $\lambda_i = \lambda_k$ 的所有步, 在使用 λ_i 的每个未来的步中以 λ_k 代替 λ_i 。然后增加一个最后步 $\lambda_{r+1} = \beta \times \lambda_r$, 其中 $\beta = c_r$ 。这就是所求的方案, 因为容易验证新的 λ_i 恰巧是旧的那些除以因子 c_i 。诸 β 是诸 α 的给定函数; 不会有除以 0 的问题, 因为如果任何 $c_k = 0$, 则我们必然有 $c_r = 0$ (因此 x^n 的系数为 0), 否则 λ_k 绝不对最后结果做出贡献。

35. 由于至少有五个参数步, 所以结果是平凡的, 除非至少有一个参数乘法; 考虑用三个乘法可以构造 $u_4 x^4$ 的方式, 我们看到, 必定有一个参数乘法和两个链乘法。因此四个加减法必须每一个都是参数步, 而且可应用习题 34。我们现在可以假定仅仅使用加法, 而且我们有一条链来计算一般的首一四次多项式, 这条链包括

两个链乘法和四个参数加法。计算四次多项式的这种类型惟一可能的方案有形式

$$\lambda_1 = \alpha_1 + \lambda_0$$

$$\lambda_2 = \alpha_2 + \lambda_0$$

$$\lambda_3 = \lambda_1 \times \lambda_2$$

$$\lambda_4 = \alpha_3 + \lambda_3$$

$$\lambda_5 = \alpha_4 + \lambda_3$$

$$\lambda_6 = \lambda_4 \times \lambda_5$$

$$\lambda_7 = \alpha_5 + \lambda_6$$

实际上这条链多做了一个加法,但是如果我们把某些 α 限制成为其它的 α 的函数,则任何正确的方案均可表示为这个形式。现在 λ_7 有 $(x^2 + Ax + B)(x^2 + Ax + C) + D = x^4 - 2Ax^3 + (E + A^2)x^2 + EAx + F$ 的形式,其中 $A = \alpha_1 + \alpha_2, B = \alpha_1\alpha_2 + \alpha_3, C = \alpha_1\alpha_2 + \alpha_4, D = \alpha_6, E = B + C, F = BC + D$;而且由于这仅仅包含三个独立的参数,因此它不可能表示任何一般的首一四次多项式。

36. 如同在习题 35 的解中那样,我们可以假设这条链仅仅使用三个链乘法和六个参数加法来计算一般的六次首一多项式。这个计算必须采取两个一般形式之一:

$$\lambda_1 = \alpha_1 + \lambda_0 \quad \lambda_1 = \alpha_1 + \lambda_0$$

$$\lambda_2 = \alpha_2 + \lambda_0 \quad \lambda_2 = \alpha_2 + \lambda_0$$

$$\lambda_3 = \lambda_1 \times \lambda_2 \quad \lambda_3 = \lambda_1 \times \lambda_2$$

$$\lambda_4 = \alpha_3 + \lambda_0 \quad \lambda_4 = \alpha_3 + \lambda_3$$

$$\lambda_5 = \alpha_4 + \lambda_3 \quad \lambda_5 = \alpha_4 + \lambda_3$$

$$\lambda_6 = \lambda_4 \times \lambda_5 \quad \lambda_6 = \lambda_4 \times \lambda_5$$

$$\lambda_7 = \alpha_5 + \lambda_6 \quad \lambda_7 = \alpha_5 + \lambda_3$$

$$\lambda_8 = \alpha_6 + \lambda_6 \quad \lambda_8 = \alpha_6 + \lambda_6$$

$$\lambda_9 = \lambda_7 \times \lambda_8 \quad \lambda_9 = \lambda_7 \times \lambda_8$$

$$\lambda_{10} = \alpha_7 + \lambda_9 \quad \lambda_{10} = \alpha_7 + \lambda_9$$

其中,如同在习题 35 中一样,已经插入一次额外的加法以包括更一般的情况。这两个方案中没有一个是能计算一般的六次首一多项式,因为头一种情况是形如

$$(x^3 + Ax^2 + Bx + C)(x^3 + Ax^2 + Bx + D) + E$$

的一个多项式,而第二种情况是形如

$$(x^4 + 2Ax^3 + (E + A^2)x^2 + EAx + F)(x^2 + Ax + G) + H$$

的多项式;这两者都仅含五个独立的参数。

37. 设 $p_0(x) = u_n x^n + u_{n-1} x^{n-1} + \cdots + u_0$ 和 $q_0(x) = x^n + v_{n-1} x^{n-1} + \cdots + v_0$ 。对于 $1 \leq j \leq n$, $p_{j-1}(x)$ 除以首一多项式 $q_{j-1}(x)$, 得到 $p_{j-1}(x) = \alpha_j q_{j-1}(x) + \beta_j q_j(x)$ 。假设满足这一关系的 $n-j$ 次首一多项式 $q_j(x)$ 存在;这对于几乎所有有理函数都将是真的。设 $p_j(x) = q_{j-1}(x) - x v q_j(x)$ 。这些定义意味着 $\deg(p_n) <$

1. 所以我们可以设 $\alpha_{n+1} = p_n(x)$ 。

对于给定有理函数, 我们有

j	α_j	β_j	$q_j(x)$	$p_j(x)$
0			$x^2 + 8x + 19$	$x^2 + 10x + 29$
1	1	2	$x + 5$	$3x + 19$
2	3	4	1	5

所以 $u(x)/v(x) = p_0(x)/q_0(x) = 1 + 2/(x+3) + 4/(x+5)$ 。

注: 具有上述形式的一般有理函数有 $2n+1$ 个实质上独立的参数, 在此意义下, 它有 $2n+1$ 个“自由度”。如果我们把多项式链推广成算术链(它允许进行加减乘运算, 也允许进行除法运算)(参见习题 71), 则稍微修改定理 A 和 M 的证明即可得到下面的一些结果: 一条具有 q 个加减法步的算术链, 至多有 $q+1$ 个自由度。一条具有 m 个乘除法步的算术链, 至多有 $2m+1$ 个自由度。因此一条计算几乎所有具上述形式的有理函数的算术链, 必然至少有 $2n$ 个加减法, 以及 n 个乘除法; 本题中的方法是最优的。

38. 如果 $n=0$, 则这定理肯定是真的。假设 n 为正, 而且给定了计算 $P(x; u_0, \dots, u_n)$ 的多项式链, 其中每一个参数 α_j 都已为一个实数所代替。设 $\lambda_i = \lambda_j \times \lambda_k$ 是头一个涉及 u_0, \dots, u_n 之一的链乘法; 由于 A 的秩数, 这样一步必定存在。不失一般性, 我们可以假设 λ_j 涉及 u_n ; 于是, λ_j 有 $h_0 u_0 + \dots + h_n u_n + f(x)$ 的形式, 其中 h_0, \dots, h_n 是实数, $h_n \neq 0$, 而且 $f(x)$ 是实系数的多项式。(诸 h 和 $f(x)$ 的诸系数从赋予诸 α 的值导出。)

现在把步骤 i 变成 $\lambda_i = \alpha \times \lambda_k$, 其中 α 是任意实数。(我们可以取 $\alpha=0$; 这里使用一般的 α 只是说明在证明中可以有某种程度的灵活性。)增加一些步骤来计算

$$\lambda = (\alpha - f(x) - h_0 u_0 - \dots - h_{n-1} u_{n-1})/h_n$$

这些新步骤仅仅含加法和参数乘法(乘以适当的新参数)。最后, 在这条链的任何地方以这个新元素代替 $\lambda_{n-1} = u_n$ 。结果是计算

$Q(x; u_0, \dots, u_{n-1}) = P(x; u_0, \dots, u_{n-1}, (\alpha - f(x) - h_0 u_0 - \dots - h_{n-1} u_{n-1})/h_n)$ 的一条链; 而且这条链少一个链乘法。如果我们能够证明 Q 满足假设, 则这个证明即告完成。量 $(\alpha - f(x))/h_n$ 可能使 m 的值增加, 还导致新的向量 B' 。如果 A 的诸列是 A_0, A_1, \dots, A_n (这些向量在实数上是线性无关的), 则对应于 Q 的新矩阵 A' 有列向量

$$A_0 - (h_0/h_n)A_n, \dots, A_{n-1} - (h_{n-1}/h_n)A_n$$

也许要加上一些 0 的行以说明 m 增加值的原因, 而且这些列显然也是线性无关的。由归纳法, 计算 Q 的链至少有 $n-1$ 个链乘法, 所以原来的链至少有 n 个链乘法。

[Pan 还证明使用除法不给出任何改进; 参考 Problemy Kibernetiki 7 (1962), 21~30。S. Winograd 已经给出关于若干个变量的好些个多项式计算的推广, 包括带

和不带各种类型的先决条件在内,见 *Comm. Pure and Applied Math.* **23** (1970), 165 ~ 179.]

39. 对 m 用归纳法。设 $w_m(x) = x^{2m} + u_{2m-1}x^{2m-1} + \cdots + u_0$, $w_{m-1}(x) = x^{2m-2} + v_{2m-3}x^{2m-3} + \cdots + v_0$, $a = \alpha_1 + \gamma_m$, $b = \alpha_m$, 并设

$$f(r) = \sum_{i,j \geq 0} (-1)^{i+j} \binom{i+j}{j} u_{r+i+2j} a^i b^j$$

由此得出,对于 $r \geq 0$, $v_r = f(r+2)$, 而且 $\delta_m = f(1)$ 。如果 $\delta_m = 0$, 而且 a 是给定的, 我们就有 b 的 $m-1$ 次多项式, 而且前导系数是 $\pm(u_{2m-1} - ma) = \pm(\gamma_2 + \cdots + \gamma_m - m\gamma_m)$ 。

在 Motzkin 未发表的笔记中, 他通过选择诸 γ 使得这个前导系数当 m 为偶数时不等于 0, 当 m 为奇数时等于 0, 做到了使 δ_k 几乎总是等于 0; 因此我们几乎总是可以让 b 是一个奇次多项式的一个(实)根。

40. 否; S. Winograd 找到了只用 7 个(可能是复数的)乘法来计算所有 13 次多项式的方法 [*Comm. Pure and Applied Math.* **25** (1972), 455 ~ 457]。L. Revah 找到了一些方案, 它们用 $\lfloor n/2 \rfloor + 1$ 个(可能是复的)乘法来计算几乎所有 $n \geq 9$ 次多项式 [*SICOMP* **4** (1975), 381 ~ 392]; 她也证明了, 当 $n=9$ 时有可能仅仅通过至少 $n+3$ 次加法就实现 $\lfloor n/2 \rfloor + 1$ 次乘法。通过附加充分多的加法(参见习题 39), 可以删去“几乎所有”和“可能是复的”这些附带条件。V. Y. Pan [*STOC* **10** (1978), 162 ~ 172; *IBM Research Report RC 7754* (1979)] 发现了对于奇的 $n \geq 9$, 使用 $\lfloor n/2 \rfloor + 1$ 个(复的)乘法和极小数为 $n+2+\delta_{n,9}$ 的(复)加法的一些方案; 对于 $n=9$ 他的方法是

$$\begin{aligned} v(x) &= ((r+a)^2 + \beta)(x+\gamma), \quad w(x) = v(x) + x \\ t_1(x) &= (v(x) + \delta_1)(w(x) + \epsilon_1), \quad t_2(x) = (v(x) + \delta_2)(w(x) + \epsilon_2) \\ u(x) &= (t_1(x) + \zeta)(t_2(x) - t_1(x) + \eta) + \kappa \end{aligned}$$

对于 $n \geq 9$, 当(实)乘法的极小数被达到时, 所必需的实加法的极小数仍然是未知的。

41. $a(c+d) - (a+b)d + i(a(c+d) + (b-a)c)$ 。[注意数值的不稳定性。三次乘法是必要的, 因为复数乘法是 $p(u) = u^2 + 1$ 时(71)的特殊情况。如果没有对加法的限制, 就还有其它可能性。例如, Peter Ungar 于 1963 年提出了对称的公式 $ac - bd + i((a+b)(c+d) - ac - bd)$; 等式 4.3.3-(2)是类似的, 并用 2^n 代替 i 。见 I. Munro, *STOC* **3** (1971), 40 ~ 44; S. Winograd [*Linear Algebra and its Applications* **4** (1971), 381 ~ 388.]

或者, 如果 $a^2 + b^2 = 1$ 及 $t = (1-a)/b = b/(1+a)$, 则用于计算乘积 $(a+bi)(c+di) = u+iv$ 的算法“ $w = c - td$, $v = d + bw$, $u = w - tv$ ”已由 Oscar Buneman 提出 [*J. Comp. Phys.* **12** (1973), 127 ~ 128]。在这个方法中, 如果 $a = \cos \theta$ 和 $b = \sin \theta$, 则我们有 $t = \tan(\theta/2)$ 。

Helmut Alt 和 Jan van Leeuwen [*Computing* **27** (1981), 205 ~ 215] 证明了, 为计算 $1/(a+bi)$, 4 次实数乘法或除法是必要的, 为计算

$$\frac{a}{b+ci} = \frac{a}{b+c(c/b)} - i \frac{(c/b)a}{b+c(c/b)}$$

4 次就足够了。为计算 $(a+bi)/(c+di)$, 6 次乘除法运算和三次加减法运算是必要和充分的。[T. Lickteig, *SICOMP* 16 (1987), 278~311.]

尽管有这些下限, 人们应当记住, 复数运算不必借助于实数算术实现。例如, 为乘两个 n 位复数所需时间, 利用快速傅里叶变换, 渐近地只大约是为乘两个 n 位实数的时间的两倍。

42. (a) 设 π_1, \dots, π_m 是对应于链乘法的诸 λ_i ; 则 $\pi_i = P_{2i-1} \times P_{2i}$ 和 $u(x) = P_{2m+1}$, 其中每个 P_j 有 $\beta_j + \beta_{j0}x + \beta_{j1}\pi_1 + \dots + \beta_{jr(j)}\pi_{r(j)}$ 的形式, 其中 $r(j) \leq \lceil j/2 \rceil - 1$ 而且每个 β_j 和 β_{jk} 都是诸 α 的整系数多项式。我们可以系统地修改链 (参考习题 30), 使得对于 $1 \leq j \leq 2m$, $\beta_j = 0$ 和 $\beta_{jr(j)} = 1$; 而且我们可以假定 $\beta_{30} = 0$, 得到的集合现在至多有 $m+1 + \sum_{j=1}^{2m} (\lceil j/2 \rceil - 1) = m^2 + 1$ 个自由度。

(b) 任何这样的至多具有 m 个链乘法的多项式链可以通过具有 (a) 中考虑的形式链进行模拟, 除非对于 $1 \leq j \leq 2m+1$, 我们现在命 $r(j) = \lceil j/2 \rceil - 1$, 而且我们不假定 $\beta_{30} = 0$ 或对于 $j \geq 3$, $\beta_{jr(j)} = 1$ 。这个单典型形式涉及 $m^2 + 2m$ 个参数。当诸 α 跑遍所有整数以及当我们跑遍所有链时, 诸 β 跑遍至多 2^{m^2+2m} 个模 2 值集, 因此得到的集合也是如此。为了获得带有 0-1 系数的所有 2^n 个 n 次多项式, 我们需要 $m^2 + 2m \geq n$ 。

(c) 置 $m \leftarrow \lfloor \sqrt{n} \rfloor$ 并计算 x^2, x^3, \dots, x^m 。命 $u(x) = u_{m+1}(x)x^{(m+1)m} + \dots + u_1(x)x^m + u_0(x)$, 其中每个 $u_j(x)$ 是次数 $\leq m$ 的整系数多项式 (因此它可以被求值而不需要任何进一步的乘法)。现在用规则 (2) 把 $u(x)$ 作为具有已知系数的 x^m 的多项式来计算 (所用的加法次数近似于系数的绝对值之和, 所以这个算法对 0-1 多项式是有效的。Paterson 和 Stockmeyer 还给出另一个算法, 它使用大约 $\sqrt{2n}$ 个乘法)。

参考文献: *SICOMP* 2 (1973), 60~66; 也见 J. E. Savage, *SICOMP* 3 (1974), 150~158; J. Ganz, *SICOMP* 24 (1995), 473~483。关于加法的类似结果, 见 Borodin 和 Cook, *SICOMP* 5 (1976), 146~157; Rivest 和 Van de Wiele, *Inf. Proc. Letters* 8 (1979), 178~180)。

43. 当 $a_i = a_j + a_k$ 是对于 $n+1$ 的某个最优加法链中的一步时, 计算 $x^i = x^j x^k$ 和 $p_i = p_k x^j + p_j$, 其中 $p_i = x^{i-1} + \dots + x + 1$; 省去最后对 x^{n+1} 的计算。每当 $a_k = 1$ 时, 特别是当 $i = 1$ 时, 我们节省一次乘法。(参考习题 4.6.3-31, 且 $\epsilon = \frac{1}{2}$ 。)

44. 设 $l = \lfloor \lg n \rfloor$, 并假设 $x, x^2, x^4, \dots, x^{2^l}$ 已预先计算好了。如果 $u(x)$ 是次数 $n = 2m+1$ 的首一多项式, 我们可以写 $u(x) = (x^{m+1} + \alpha)v(x) + w(x)$, 其中 $v(x)$ 和 $w(x)$ 是次数为 m 的首一多项式。对于 $n = 2^{l+1} - 1 \geq 3$, 这产生一个要求 $2^l - 1$ 次进一步的乘法和 $2^{l+1} + 2^l - 2$ 次加法的方法。如果 $n = 2^l$, 我们可以应用霍纳规则对 n 减 1。而且如果 $m = 2^l < n < 2^{l+1} - 1$, 我们可以写 $u(x) = x^m v(x) +$

$w(x)$, 其中 v 和 w 分别是 $n-m$ 次和 m 次的首一多项式。对 l 用归纳法, 在做了预计算之后, 这至多要求 $\frac{1}{2}n + l - 1$ 次乘法和 $\frac{5}{4}n$ 次加法。[参见 S. Winograd, *IBM Tech. Disclosure Bull.* 13 (1970), 1133~1135。]

注: 在相同的基本规则下, 如果我们的目标是来极小化乘法 + 加法, 有可能以 $\frac{1}{2}n + O(\sqrt{n})$ 次乘法和 $n + O(\sqrt{n})$ 次加法来计算 $u(x)$ 。普通的多项式

$$p_{jkm}(x) = \left(\left(\cdots \left((x^m + \alpha_0)(x^{j+1} + \beta_1) + \alpha_1 \right)(x^{j+2} + \beta_2) + \alpha_2 \right) \cdots \right) (x^k + \beta_{k-j}) + \alpha_{k-j} \Big) (x^j + \beta_0)$$

“覆盖”指数 $\{j, j+k, j+k+(k-1), \dots, j+k+(k-1)+\dots+(j+1), m'-k, m'-k+1, \dots, m'-j\}$ 的系数, 其中

$$m' = m + j + (j+1) + \dots + k = m + \binom{k+1}{2} - \binom{j}{2}$$

对于 $m_j = \binom{j+1}{2} + \binom{k-j+2}{2}$, 通过把这样的多项式 $p_{1km_1}(x), p_{2km_2}(x), \dots, p_{kkm_k}(x)$ 加在一起, 我们得到次数为 $k^2 + k + 1$ 的任意首一多项式。[Rabin 和 Winograd, *Comm. on Pure and Applied Math.* 25 (1972), 433~458, §2; 这一论文还证明, 如果 n 足够大, 对于所有 $\epsilon > 0$, 则带有 $\frac{1}{2}n + O(\log n)$ 次乘法和小于等于 $(1+\epsilon)n$ 次加法的构造是可能的。

45. 证明 (T_{ijk}) 的秩至多是 (t_{ijk}) 的秩就行了, 因为通过对变换 F^{-1}, G^{-1}, H^{-1} 的相同方式来变换它, 我们可以从 (T_{ijk}) 重新得到 (t_{ijk}) 。如果 $t_{ijk} = \sum_{l=1}^s a_{il}b_{jl}c_{kl}$, 则立即得出

$$T_{ijk} = \sum_{1 \leq l \leq r} \left(\sum_{i'=1}^m F_{i'i} a_{i'l} \right) \left(\sum_{j'=1}^n G_{j'j} b_{j'l} \right) \left(\sum_{k'=1}^s H_{k'k} c_{k'l} \right)$$

[H. F. de Groote 已经证明, 用 7 个链乘法产生 2×2 矩阵乘积的所有正规方案都是等价的。等价的含义是: 如同本题中那样, 通过非奇异矩阵乘法它们可以相互转化。在这一意义下, Strassen 的算法是惟一的。参见 *Theor. Comp. Sci.* 7 (1978), 127~148。]

46. 由习题 45, 我们可以增加行、列或平面的任何倍数到另一个行、列或平面上而不改变秩; 我们也可以用非 0 常数, 乘一行, 一列或一个平面, 或者转置张量。总可以找到一系列这样的运算, 以把一个给定的 $2 \times 2 \times 2$ 张量归结成形式 $\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ q & r \end{pmatrix}$ 之一。由定理 W (参见 (74)), 根据多项式 $u^2 - ru - q$ 在所关心的域中有一个还是两个不可约因子, 最后的张量有秩 3 或 2。

47. 一般的 $m \times n \times s$ 张量有 mns 个自由度。由习题 28 知, 不可能只用实现

(A, B, C) 的 $(m+n+s)r$ 个元素来表达所有的 $m \times n \times s$ 张量, 除非 $(m+n+s)r \geq mns$ 。另一方面, 假设 $m \geq n \geq s$ 。一个 $m \times n$ 矩阵的秩至多为 n , 所以通过分别地实现每个矩阵平面, 我们可以用 ns 个链乘法实现任何张量。[习题 46 表明, 对于极大的张量秩, 这一下限不是最好的, 上限也不是最好的。Thomas D. Howell (Ph. D. thesis, Cornell Univ., 1976), 已经证明, 在复数上有秩 $\geq \lceil nms/(m+n+s-2) \rceil$ 的张量。]

48. 如果 (A, B, C) 和 (A', B', C') 是长度分别为 r 和 r' 的 (t_{ijk}) 和 (t'_{ijk}) 的实现, 则 $A'' = A \oplus A', B'' = B \oplus B', C'' = C \oplus C'$ 和 $A''' = A \otimes A', B''' = B \otimes B', C''' = C \otimes C'$, 是长度分别为 $r+r'$ 和 $r \cdot r'$ 的实现。

注: 许多人很自然地猜想 $\text{rank}((t_{ijk}) \oplus (t'_{ijk})) = \text{rank}(t_{ijk}) + \text{rank}(t'_{ijk})$, 但习题 60b) 和习题 65 的构造使得这个猜想比以前更不可信。

49. 由引理 T, $\text{rank}(t_{ijk}) \geq \text{rank}(t_{i(jk)})$ 。反之如果 M 是秩为 r 的一个矩阵, 我们可以通过行和列的操作来变换它, 即求非奇异矩阵 F 和 G , 使得 FMG 除了 r 个对角线元素为 1 外所有元素全为 0; 参见算法 4.6.2N。因此 FMG 的张量秩 $\leq r$; 而且由习题 45 它和 M 的张量秩相同。

50. 设 $i = \langle i', i'' \rangle$, 其中 $1 \leq i' \leq m$ 和 $1 \leq i'' \leq n$; 则 $t_{\langle i', i'' \rangle, jk} = \delta_{i'j} \delta_{i''k}$, 而且显然 $\text{rank}(t_{i(jk)}) = mn$, 因为 $(t_{i(jk)})$ 是一个排列矩阵。由引理 L, $\text{rank}(t_{ijk}) \geq mn$ 。反之, 由于 (t_{ijk}) 仅有 mn 个非 0 元素, 它的秩显然小于等于 mn 。(因此没有要求少于 mn 个明显乘法数的规范方案。也没有这种非规范方案 [Comm. Pure and Appl. Math. 3 (1970), 165~179]。但如果使用相同的矩阵并且有 $s > 1$ 个不同的列向量, 则可实现一些节省, 因为这等价于 $(m \times n)$ 乘 $(n \times s)$ 矩阵乘法。)

51. (a) $s_1 = y_0 + y_1, s_2 = y_0 - y_1; m_1 = \frac{1}{2}(x_0 + x_1)s_1, m_2 = \frac{1}{2}(x_0 - x_1)s_2; w_0 = m_1 + m_2, w_1 = m_1 - m_2$ 。(b) 这里是一些中间步骤, 使用正文中的方法: $((x_0 - x_2) + (x_1 - x_2)u)((y_0 - y_2) + (y_1 - y_2)u) \bmod (u^2 + u + 1) = ((x_0 - x_2)(y_0 - y_2) - (x_1 - x_2)(y_1 - y_2)) + ((x_0 - x_2)(y_0 - y_2) - (x_1 - x_0)(y_1 - y_0))u$ 。头一个实现是

$$\begin{pmatrix} 1 & 1 & \bar{1} & 0 \\ 1 & 0 & 1 & 1 \\ 1 & \bar{1} & 0 & \bar{1} \end{pmatrix}, \begin{pmatrix} 1 & 1 & \bar{1} & 0 \\ 1 & 0 & 1 & 1 \\ 1 & \bar{1} & 0 & \bar{1} \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 & \bar{2} \\ 1 & 1 & \bar{2} & 1 \\ 1 & \bar{2} & 1 & 1 \end{pmatrix} \times \frac{1}{3}$$

第二个实现是

$$\begin{pmatrix} 1 & 1 & 1 & \bar{2} \\ 1 & 1 & \bar{2} & 1 \\ 1 & \bar{2} & 1 & 1 \end{pmatrix} \times \frac{1}{3}, \begin{pmatrix} 1 & 1 & \bar{1} & 0 \\ 1 & \bar{1} & 0 & \bar{1} \\ 1 & 0 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & \bar{1} & 0 \\ 1 & 0 & 1 & 1 \\ 1 & \bar{1} & 0 & \bar{1} \end{pmatrix}$$

得到的算法计算 $s_1 = y_0 + y_1, s_2 = y_0 - y_1, s_3 = y_2 - y_0, s_4 = y_2 - y_1, s_5 = s_1 + y_2; m_1 = \frac{1}{3}(x_0 + x_1 + x_2)s_5, m_2 = \frac{1}{3}(x_0 + x_1 - 2x_2)s_2, m_3 = \frac{1}{3}(x_0 - 2x_1 + x_2)s_3, m_4 =$

$$\frac{1}{3}(-2x_0 + x_1 + x_2)s_4; t_1 = m_1 + m_2, t_2 = m_1 - m_2, t_3 = m_1 + m_3, w_0 = t_1 - m_3,$$

$$w_1 = t_3 + m_4, w_2 = t_2 - m_4.$$

52. 设当 $k \bmod n' = k'$ 和 $k \bmod n'' = k''$ 时 $k = \langle k', k'' \rangle$ 。于是我们希望计算对于 $i' + j' \equiv k' \pmod{n'}$ 和 $i'' + j'' \equiv k'' \pmod{n''}$ 求和的 $w_{\langle k', k'' \rangle} = \sum x_{\langle i', i'' \rangle} y_{\langle j', j'' \rangle}$ 。通过把 n' 算法应用于长度为 n'' 的 $2n'$ 个向量 X_i 和 Y_j , 得到 n' 个向量 W_k , 即可完成计算。每个向量加法变成 n'' 个加法, 每个参数乘法变成 n'' 个参数乘法, 而且向量的每个链乘法被次数为 n'' 的循环卷积代替。[由于习题 4.6.2-32 和定理 W, 如果子算法使用在有理数上极小次数的链乘法数, 则这个算法比这极小的乘法数多使用 $2(n' - d(n'))(n'' - d(n''))$ 次乘法, 其中 $d(n)$ 是 n 的因子数。]

53. a) 设对于 $0 \leq k < e, n(k) = (p-1)p^{e-k-1} = \varphi(p^{e-k})$, 而且对 $k \geq e$ 有 $n(k) = 1$ 。把数 $\{1, \dots, m\}$ 表示成形式 $a^i p^k \pmod{m}$, 其中 $0 \leq k \leq e$ 和 $0 \leq i < n(k)$, a 是模 p^e 下的一个固定的本原元。例如, 当 $m = 9$ 时, 我们可以令 $a = 2$; 值是 $\{2^0 3^0, 2^1 3^0, 2^0 3^1, 2^2 3^0, 2^3 3^0, 2^4 3^0, 2^3 3^1, 2^0 3^2\}$ 。于是 $f(a^i p^k) = \sum_{0 \leq l \leq e} \sum_{0 \leq j < n(l)} \omega^{g(i, j, k, l)} F(a^j p^l)$, 其中 $g(i, j, k, l) = a^{i+j} p^{k+l}$ 。

对于 $0 \leq i < n(k)$ 和对于每个 k 和 l , 我们将计算 $f_{ikl} = \sum_{0 \leq j < n(l)} \omega^{g(i, j, k, l)}$,

$F(a^j p^l)$ 。这是对于值 $x_i = \omega^{a^i p^{k+l}}$ 和 $y_s = \sum_{0 \leq j < n(l)} [s + j \equiv 0 \pmod{n(k+l)}] F(a^j p^l)$ 的 $n(k+l)$ 次循环卷积, 因为 f_{ikl} 是对 $r + s \equiv i \pmod{n(k+l)}$ 的求和 $\sum x_r y_s$ 。傅里叶变换通过对适当的 f_{ikl} 求和得到。[注: 例如, 当像在 (69) 中那样构造诸 x 的线性组合时, 如果已经使用规则 (59) 构造循环卷积算法且 $u^{n(k)} - 1 = (u^{n(k)/2} - 1)(u^{n(k)/2} + 1)$, 则结果将纯粹是实的或纯粹是虚的。原因是约简模 $(u^{n(k)/2} - 1)$ 产生带有实系数 $\omega^j + \omega^{-j}$ 的一个多项式, 而约简模 $(u^{n(k)/2} + 1)$ 产生带有虚系数 $\omega^j - \omega^{-j}$ 的一个多项式。]

当 $p = 2$ 时, 可应用一个类似的构造, 使用表示 $(-1)^i a^j 2^k \pmod{m}$, 其中 $0 \leq k \leq e$ 且 $0 \leq i \leq \min(e-k, 1)$ 且 $0 \leq j < 2^{e-k-2}$ 。在这种情况下我们以 $n' = 2$ 和 $n'' = 2^{e-k-2}$ 使用习题 52 的构造; 尽管这些数不是互素的, 但这个构造确实产生循环卷积所求的直接积。

b) 设 $a'm' + a''m'' = 1$; 并设 $\omega' = \omega^{a'm'}$, $\omega'' = \omega^{a''m''}$ 。定义 $s' = s \bmod m'$, $s'' = s \bmod m''$, $t' = t \bmod m'$, $t'' = t \bmod m''$, 使得 $\omega^s = (\omega')^{s'} (\omega'')^{s''}$ 。由此得出 $f(s', s'') = \sum_{t'=0}^{m'-1} \sum_{t''=0}^{m''-1} (\omega')^{s't'} (\omega'')^{s''t''} F(t', t'')$; 换句话说, 对 m 个元素的一维傅里叶变换实际上是对 $m' \times m''$ 个元素的二维傅里叶变换的一种改头换面。

我们将讨论一些“规范”算法, 它们的组成方式是: (i) 诸 F 和诸 s 的一些和 s_i ; (ii) 一些乘积 m_j , 它们中每一个都是通过用一个实数或虚数 α_j 乘诸 F 或诸 S 之一得到的; (iii) 一些和 t_k , 它们中的每一个由诸 m 或诸 t (不是诸 F 或者 s) 构成。最后的值必是诸 m 或诸 t 。例如从 (69) 和 a) 的方法构造的“规范”的 $m = 5$ 傅里叶变换方案如下: $s_1 = F(1) + F(4)$, $s_2 = F(3) + F(2)$, $s_3 = s_1 + s_2$, $s_4 = s_1 - s_2$, $s_5 = F(1)$

$-F(4), s_6 = F(2) - F(3), s_7 = s_5 - s_6; m_1 = \frac{1}{4}(\omega + \omega^2 + \omega^4 + \omega^3)s_3, m_2 = \frac{1}{4}(\omega - \omega^2 + \omega^4 - \omega^3)s_4, m_3 = \frac{1}{2}(\omega + \omega^2 - \omega^4 - \omega^3)s_5, m_4 = \frac{1}{2}(-\omega + \omega^2 + \omega^4 - \omega^3)s_6, m_5 = \frac{1}{2}(\omega^3 - \omega^2)s_7, m_6 = 1 \cdot F(5), m_7 = 1 \cdot s_3; t_0 = m_1 + m_6, t_1 = t_0 + m_2, t_2 = m_3 + m_5, t_3 = t_0 - m_2, t_4 = m_4 - m_5, t_5 = t_1 + t_2, t_6 = t_3 + t_4, t_7 = t_1 - t_2, t_8 = t_3 - t_4, t_9 = m_6 + m_7$ 。注意在 m_6 和 m_7 中有乘以 1 的乘法;这是为我们的约定所要求的,而且包括这样一些情况以便用于递归构造中是重要的(尽管这些乘法不必真正去做)。这里 $m_6 = f_{001}, m_7 = f_{010}, t_5 = f_{000} + f_{001} = f(2^0), t_6 = f_{100} + f_{101} = f(2^1)$, 等等。我们可以通过引入 $s_8 = s_3 + F(5)$ 改进此方案,并用 $(\frac{1}{4}(\omega + \omega^2 + \omega^4 + \omega^3) - 1)s_3$ [这是 $-\frac{5}{4}s_3$] 代替 m_1 , 用 $1 \cdot s_8$ 代替 m_6 , 并删去 m_7 和 t_9 ; 这就节省了乘以 1 的平凡乘法之一,而且当这一方案用来构造更大的傅里叶变换时它将是有益的。在改进的方案中, $f(5) = m_6, f(1) = t_5, f(2) = t_6, f(3) = t_8, f(4) = t_7$ 。

现在假设我们有 m' 和 m'' 的一维规范方案,并分别使用 (a', a'') 个复数加法, (t', t'') 个乘以 ± 1 或 $\pm i$ 的平凡乘法,总共有 (c', c'') 个复数乘法(包括平凡的乘法在内)。(那些非平凡的乘法也全都是“简单的”,因为它们只含两个实数乘法而无实数加法。)通过把 m' 的方案应用到长度为 m'' 的向量 $F(t', *)$ 上,我们可以构造适用于二维 $m' \times m''$ 情况的一个规范方案。每个 s_i 步变成 m'' 个加法;每个 m_j 变成 m'' 个元素上的一个傅里叶变换,但这算法中所有 a 都乘以 a_j ;而且每个 t_k 变成 m'' 个加法。于是一个新的算法有 $(a'm'' + c'a'')$ 个复数加法, $i't''$ 个平凡乘法,及总共 $c'c''$ 个复数乘法。

利用这些技术,Winograd 已经求出对于下列小的 m 值和下列代价 (a, t, c) 的一维规范方案:

$m = 2$	$(2, 2, 2)$	$m = 7$	$(36, 1, 9)$
$m = 3$	$(6, 1, 3)$	$m = 8$	$(26, 6, 8)$
$m = 4$	$(8, 4, 4)$	$m = 9$	$(46, 1, 12)$
$m = 5$	$(17, 1, 6)$	$m = 16$	$(74, 8, 18)$

通过像上边所介绍的那样把这些方案结合起来,我们就得到使用算术运算数比习题 14 中讨论的“快速傅里叶变换”(FFT)使用的算术运算数还要少的一些方法。例如,当 $m = 1008 = 7 \cdot 9 \cdot 16$ 时,代价是 $(17946, 8, 1944)$,所以我们可以用 3872 个实数乘法和 35892 个实数加法对 1008 个复数进行傅里叶变换。如 Nussbaumer 和 Qwandalle 在 *IBM J. Res. and Devel.* **22** (1978), 134~144 中所说明的,有可能通过使用多维卷积来组合互素的模,从而改进 Winograd 的方法;他们的巧妙方法把 1008 点的复数傅里叶变换所需计算数量减少到 3084 个实数乘法和 34668 个实数加法。与此相对照,1024 个复数的 FFT 要用 14344 个实数乘和 27652 个实数加法。但是如果

使用习题 14 答案中两遍扫描合一的改进,则 1024 个复数的 FFT 只须 10936 个实数乘法和 25948 个加法,而这不难于实现。因此更精巧的方法仅仅在乘法花的时间大大超过加法的机器上才是更快的。

[参考文献: *Proc. Nat. Acad. Sci. USA* **73** (1976), 1005~1006; *Math. Comp.* **32** (1978), 175~199; *Advances in Math.* **32** (1979), 83~117; *IEEE Trans. ASSP* **27** (1979), 169~181.]

54. $\max(2e_1 \deg(p_1) - 1, \dots, 2e_q \deg(p_q) - 1, q + 1)$ 。

55. $2n' - q'$, 其中 n' 是 P 的极小多项式的次数(即使得 $\mu(P)$ 为 0 矩阵的最小次数的首一多项式 μ) 而 q' 是它所有的不同不可约因子的个数。(用相似变换约简 P 。)

56. 设对于所有的 $i, j, k, t_{ijk} + t_{jik} = \tau_{ijk} + \tau_{jik}$ 。如果 (A, B, C) 是秩为 r 的 (t_{ijk}) 的一个实现, 则对所有 $k, \sum_{l=1}^r c_{kl} (\sum_i a_{il} x_i) (\sum_j b_{jl} x_j) = \sum_{i,j} t_{ijk} x_i x_j = \sum_{i,j} \tau_{ijk} \cdot x_i x_j$ 。反之, 设对于 $1 \leq l \leq r$, 一个多项式链的第 l 个链乘法是乘积 $(a_l + \sum_i a_{il} x_i) (\beta_l + \sum_j \beta_{jl} x_j)$, 其中 a_l 和 β_l 表示可能的常数项和(或)非线性项。出现于这个链的任何步上的所有二次项都可表达为一个线性组合 $\sum_{l=1}^r c_l (\sum_i a_{il} x_i) (\sum_j \beta_{jl} x_j)$; 因此这个链定义秩 $\leq r$ 的张量 (t_{ijk}) 使得 $t_{ijk} + t_{jik} = \tau_{ijk} + \tau_{jik}$ 。提示即由此而来。现在 $\text{rank}(\tau_{ijk} + \tau_{jik}) = \text{rank}(t_{ijk} + t_{jik}) \leq \text{rank}(t_{ijk}) + \text{rank}(t_{jik}) = 2 \text{rank}(t_{ijk})$ 。

$x_1, \dots, x_m, y_1, \dots, y_n$ 的一个双线性形式是 $m + n$ 个变量的一个二次形式, 其中对于 $i \leq m$ 和 $j > m$, 有 $\tau_{ijk} = t_{i, j-m, k}$, 否则 $\tau_{ijk} = 0$ 。现在 $\text{rank}(\tau_{ijk}) + \text{rank}(\tau_{jik}) \geq \text{rank}(t_{ijk})$, 因为我们通过在 $(\tau_{ijk} + \tau_{jik})$ 的一个实现 (A, B, C) 中去掉 A 的最后 n 行和 B 的头 m 行得到 (t_{ijk}) 的一个实现。

57. 设 N 是超过 $2n$ 的 2 的最小次幂, 并设 $u_{n+1} = \dots = u_{N-1} = v_{n+1} = \dots = v_{N-1} = 0$ 。如果 $U_s = \sum_{t=0}^{N-1} \omega^s u_t$ 和 $V_s = \sum_{t=0}^{N-1} \omega^s v_t, 0 \leq s < N, \omega = e^{2\pi i/N}$, 则 $\sum_{s=0}^{N-1} \omega^{-st} U_s V_s = N \sum u_{t_1} v_{t_2}$, 其中后一个求和是对于使得 $0 \leq t_1, t_2 < N, t_1 + t_2 \equiv t \pmod{N}$ 的所有 t_1 和 t_2 进行的。除了 $t_1 \leq n$ 和 $t_2 \leq n$ 使得 $t_1 + t_2 < N$ 以外, 其余项皆为 0; 于是这个和是在乘积 $u(z)v(z)$ 中 z^t 的系数。如果使用习题 14 的方法来计算傅里叶变换和逆变换, 则复运算的次数是 $O(N \log N) + O(N \log N) + O(N) + O(N \log N)$; 且 $N \leq 4n$ 。[参考 4.3.3C 小节和 J. M. Pollard 的文章, 见 *Math. Comp.* **25** (1971), 365~374.]

当整系数多项式相乘时, 有可能使用一个整数 ω , 它的阶是 2^t 模素数 p , 并且模充分多个素数来确定这些结果。在这方面有用的素数连同它们的最小原根 r (当 $p \bmod 2^t = 1$ 时我们取 $\omega = r^{(p-1)/2^t} \bmod p$) 可以像在 4.5.4 小节中所介绍的那样求出。对于 $t=9$, 小于 2^{35} 的 10 个最大的情况是 $p = 2^{35} - 512a + 1$, 其中 $(a, r) = (28, 7), (31, 10), (34, 13), (56, 3), (58, 10), (76, 5), (80, 3), (85, 11), (91, 5), (101, 3)$; 小于 2^{31} 的 10 个最大的情况是 $p = 2^{31} - 512a + 1$, 其中 $(a, r) = (1, 10), (11, 3), (19, 11), (20, 3), (29, 3), (35, 3), (55, 19), (65, 6), (95, 3), (121, 10)$ 。对于更大的

t , 形如 $2^t q + 1$ 的所有素数 p (其中 $q < 32$ 是奇数, 且 $2^{24} < p < 2^{36}$) 由 $(p-1, r) = (11 \cdot 2^{21}, 3), (25 \cdot 2^{20}, 3), (27 \cdot 2^{20}, 5), (25 \cdot 2^{22}, 3), (27 \cdot 2^{22}, 7), (5 \cdot 2^{25}, 3), (7 \cdot 2^{26}, 3), (27 \cdot 2^{26}, 13), (15 \cdot 2^{27}, 31), (17 \cdot 2^{27}, 3), (3 \cdot 2^{30}, 5), (13 \cdot 2^{28}, 3), (29 \cdot 2^{27}, 3), (23 \cdot 2^{29}, 5)$ 给出。对于适当小的 e , 当 $\omega = 2^e$ 时, 可以使用后边的一些素数。关于这样素数的讨论, 见 R. M. Robinson, *Proc. Amer. Math. Soc.* **9** (1958), 673 ~ 681; S. W. Golomb, *Math. Comp.* **30** (1976), 657 ~ 663。在习题 4.6-5 的答案中还引用了另外的全整数的方法。

但是习题 59 的方法在实践中几乎总是更可取的。

58. (a) 一般来说如果 (A, B, C) 实现 (t_{ijk}) , 则 $((x_1, \dots, x_m)A, B, C)$ 是其 j 行 k 列元素为 $\sum x_i t_{ijk}$ 的 $1 \times n \times s$ 矩阵的实现。所以 $(x_1, \dots, x_m)A$ 中的非 0 元素个数必定至少和这个矩阵的秩相同。在对应于 $m-1$ 次乘 $n-1$ 次的多项式乘法的 $m \times n \times (m+n-1)$ 张量情况下, 每当 $(x_1, \dots, x_m) \neq (0, \dots, 0)$ 时, 对应的矩阵有秩 n 。对于 $A \leftrightarrow B$ 和 $m \leftrightarrow n$, 一个类似的命题成立。

注: 特别是, 如果我们在 2 个元素的域上工作, 这表明每当 (A, B, C) 是完全由整数组成的一个实现时, A 模 2 的行构成至少有距离 n 的 m 个向量的一个“线性码”。这一发现源于 R. W. Brockett 和 D. Dobkin [*Linear Algebra and its Applications* **19** (1978), 207 ~ 235, 定理 14; 也参见 Lempel 和 Winograd, *IEEE Trans. IT-23* (1977), 503 ~ 508; Lempel, Seroussi 和 Winograd, *Theoretical Comp. Sci.* **22** (1983), 285 ~ 296], 它可以用来得到关于整数上的秩的一些非平凡的下限。例如, M. R. Brown 和 D. Dobkin [*IEEE Trans. C-29* (1980), 337 ~ 340] 已经用它证明了在整数上 $n \times n$ 多项式乘法的实现, 对所有充分大的 n 必定有大于等于 α_n 的秩, 当 α 是小于

$$\alpha_{\min} = 3.52762 \ 68026 \ 32407 \ 48061 \ 54754 \ 08128 \ 07512 \ 70182 +$$

的任何实数时; 这里 $\alpha_{\min} = 1/H(\sin^2 \theta, \cos^2 \theta)$, 其中 $H(p, q) = p \lg(1/p) + q \lg(1/q)$ 是二进熵函数, $\theta \approx 1.34686$ 是 $\sin^2(\theta - \pi/4) = H(\sin^2 \theta, \cos^2 \theta)$ 的根。M. Kaminski [*J. Algorithms* **9** (1988), 137 ~ 147] 已经构造了基于割圆多项式, 秩为 $O(n \log n)$ 的一个全整数实现。

$$(b) \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{1} & \bar{1} & 0 & 0 & 0 & 1 & 0 & 0 \\ \bar{1} & 1 & \bar{1} & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & \bar{1} & \bar{1} & \bar{1} & \bar{1} & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

H. Cohen 和 A. K. Lenstra [参见 *Math. Comp.* **48** (1987), S1 ~ S2] 已经给出实现次数为 2, 3 和 4 的一般多项式乘法的以下经济的方式:

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}, \text{相同}, \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \bar{1} & \bar{1} & 0 & 1 & 0 & 0 \\ \bar{1} & 1 & \bar{1} & 0 & 1 & 0 \\ 0 & \bar{1} & \bar{1} & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}, \text{相同}, \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{1} & \bar{1} & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \bar{1} & 1 & \bar{1} & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & \bar{1} & \bar{1} & \bar{1} & \bar{1} & 1 \\ 0 & \bar{1} & 1 & \bar{1} & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \bar{1} & \bar{1} & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}, \text{相同}, \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{1} & \bar{1} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{1} & 1 & \bar{1} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{1} & \bar{1} & \bar{1} & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \bar{1} & 0 & 0 \\ 1 & 1 & 1 & \bar{1} & 0 & 0 & \bar{1} & \bar{1} & 1 & 0 & 0 & 1 & 1 & \bar{1} \\ 1 & \bar{1} & 0 & 0 & \bar{1} & 0 & \bar{1} & 1 & 0 & 1 & 0 & 0 & \bar{1} & 0 \\ 0 & 1 & 0 & 0 & 0 & \bar{1} & 0 & \bar{1} & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \bar{1} & \bar{1} & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

在每种情况下, A 和 B 矩阵是相等的。

59. [IEEE Trans. ASSP-28 (1980), 205~215.] 注意循环卷积是多项式乘法模 $u^n - 1$, 而负循环卷积是多项式乘法模 $u^n + 1$ 。现在我们改变记号, 用 2^n 代替 n ; 我们将考虑 (x_0, \dots, x_{2^n-1}) 同 (y_0, \dots, y_{2^n-1}) 的循环和负循环卷积 (z_0, \dots, z_{2^n-1}) 的递归算法。为了简略和易于剖析起见, 这些算法以非最优的形式给出; 实现这些算法的读者将会注意到, 许多事情可以流水线化。例如, 在步骤 N5 中的 $Z_{2^m-1}(w)$ 的值将总是零。

C1. [测试简单情况] 如果 $n = 1$, 则置 $z_0 \leftarrow x_0 y_0 + x_1 y_1$, $z_1 \leftarrow (x_0 + x_1)(y_0 + y_1) - z_0$, 并终止。否则置 $m \leftarrow 2^{n-1}$ 。

C2. [剩余化] 对于 $0 \leq k < m$, 置 $(x_k, x_{m+k}) \leftarrow (x_k + x_{m+k}, x_k - x_{m+k})$, 及 $(y_k, y_{m+k}) \leftarrow (y_k + y_{m+k}, y_k - y_{m+k})$ 。(现在我们有 $x(u) \bmod (u^m - 1) = x_0 + \dots + x_{m-1} u^{m-1}$ 和 $x(u) \bmod (u^m + 1) = x_m + \dots + x_{2m-1} u^{m-1}$; 我们将计算 $x(u)y(u) \bmod (u^m - 1)$ 和 $x(u)y(u) \bmod (u^m + 1)$, 然后将通过

(59)把这些结果结合起来。)

C3.[递归] 置 (z_0, \dots, z_{m-1}) 成为 (x_0, \dots, x_{m-1}) 同 (y_0, \dots, y_{m-1}) 的循环卷积。并置 (z_m, \dots, z_{2m-1}) 成为 (x_m, \dots, x_{2m-1}) 同 (y_m, \dots, y_{2m-1}) 的负循环卷积。

C4.[非剩余化] 对于 $0 \leq k < m$, 置 $(z_k, z_{m+k}) \leftarrow \frac{1}{2}(z_k + z_{m+k}, z_k - z_{m+k})$ 。

现在 (z_0, \dots, z_{2m-1}) 是所求答案。■

N1.[测试简单情况] 如果 $n=1$, 则置 $t \leftarrow x_0(y_0 + y_1)$, $z_0 \leftarrow t - (x_0 + x_1)y_1$, $z_1 \leftarrow t + (x_1 - x_0)y_0$, 并终止。否则置 $m \leftarrow 2^{\lfloor n/2 \rfloor}$ 及 $r \leftarrow 2^{\lceil n/2 \rceil}$ 。(下列步骤使用 2^{n+1} 个辅助变量 X_{ij} ($0 \leq i < 2m, 0 \leq j < r$)来表示 $2m$ 个多项式 $X_i(w) = X_{i0} + X_{i1}w + \dots + X_{i(r-1)}w^{r-1}$;类似地, 有 2^{n+1} 个辅助变量 Y_{ij} 。)

N2.[辅助多项式初始化] 置 $X_{ij} \leftarrow X_{(i+m)_j} \leftarrow x_{mj+i}$, $Y_{ij} \leftarrow Y_{(i+m)_j} \leftarrow y_{mj+i}$, 其中 $0 \leq i < m$ 和 $0 \leq j < r$ 。(这时我们有 $x(u) = X_0(u^m) + uX_1(u^m) + \dots + u^{m-1}X_{m-1}(u^m)$, 而且对于 $y(u)$ 一个类似的公式成立。我们的策略将是按

如下方式把这些多项式在模 $(u^{mr} + 1) = (u^{2^n} + 1)$ 之下相乘: 对多项式 $X(w)$ 和 $Y(w)$ 做模 $(w^r + 1)$ 的运算, 找出它们的长度为 $2m$ 的循环卷积并由此得到 $x(u)y(u) \equiv Z_0(u^m) + uZ_1(u^m) + \dots + u^{2m-1}Z_{2m-1}(u^m)$ 。

N3.[变换] (现在要做的实质上是对多项式 $(X_0, \dots, X_{m-1}, 0, \dots, 0)$ 和 $(Y_0, \dots, Y_{m-1}, 0, \dots, 0)$ 做一个快速傅里叶变换, 并用 $w^{r/m}$ 作为 $-2m$ 次单位根。这是有效的, 因为乘以 w 的幂实际上不能算是一个乘法。)对于 $j = \lfloor n/2 \rfloor - 1, \dots, 1, 0$ (按此顺序), 对所有 m 个二进数 $s+t = (s_{\lfloor n/2 \rfloor} \dots s_{j+1} 0 \dots 0)_2 + (0 \dots 0 t_{j-1} \dots t_0)_2$ 做下列操作: 用多项式偶 $(X_{s+t}(w) + w^{(r/m)s'} X_{s+t+2'}(w))$, $X_{s+t}(w) - w^{(r/m)s'} X_{s+t+2'}(w)$ 代替 $(X_{s+t}(w), X_{s+t+2'}(w))$, 其中 $s' = 2^j(s_{j+1} \dots s_{\lfloor n/2 \rfloor})_2$ 。(我们正以 $K=2m$ 和 $\omega = w^{r/m}$ 计算4.3.3-(39); 注意 s' 中二进位的反序。更确切地说, 操作 $X_i(w) \leftarrow X_i(w) + w^k X_l(w)$ 意味着, 对于 $k \leq j < r$, 我们置 $X_{ij} \leftarrow X_{ij} + X_{l(j-k)}$, 而且对于 $0 \leq j < k$ 置 $X_{ij} \leftarrow X_{ij} - X_{l(j-k+r)}$ 。可以做 $X_l(w)$ 的一个副本而无须浪费许多空间。)对诸 Y 做同样的变换。

N4.[递归] 对 $0 \leq i < 2m$, 置 $(Z_{i0}, \dots, Z_{i(r-1)})$ 成为 $(X_{i0}, \dots, X_{i(r-1)})$ 和 $(Y_{i0}, \dots, Y_{i(r-1)})$ 的负循环卷积。

N5.[变换] 对于 $j = 0, 1, \dots, \lfloor n/2 \rfloor$ (按此顺序), 对于如步骤N3中那样的 s 和 t 的所有 m 种选择, 置 $(Z_{s+t}(w), Z_{s+t+2'}(w)) \leftarrow \frac{1}{2}(Z_{s+t}(w) + Z_{s+t+2'}(w), w^{-(r/m)s}(Z_{s+t}(w) - Z_{s+t+2'}(w)))$ 。

N6.[重装] (现在我们已经实现了在步骤N2结束时指出的目标, 因为容易看

出,诸 Z 的变换是诸 X 和诸 Y 变换的乘积。)对 $0 < j < r, 0 \leq i < m$, 置 $z_j \leftarrow Z_{i0} - Z_{(m+1)(r-1)}$ 和 $z_{mj+i} \leftarrow Z_j + Z_{(m+1)(j-1)}$ 。 ▮

容易验证,对于这个计算中的中间变量至多需要 n 位附加的精度;即如果在算法开始时对于 $0 \leq i < 2^n, |x_i| \leq M$, 则所有 x 和 X 变量将全都以 $2^n M$ 为界。所有的 z 和 Z 变量全部以 $(2^n M)^2$ 为界,它们比最后的卷积拥有的二进位数多出 n 位。

算法 N 执行 A_n 个加减法, D_n 个折半, 及 M_n 个乘法, 其中 $A_1 = 5, D_1 = 0, M_1 = 3$; 对于 $n > 1$, 我们有 $A_n = \lfloor n/2 \rfloor 2^{n-2} + 2^{\lfloor n/2 \rfloor + 1} A_{\lfloor n/2 \rfloor} + (\lfloor n/2 \rfloor + 1) 2^{n-1} + 2^n$, $D_n = 2^{\lfloor n/2 \rfloor + 1} D_{\lfloor n/2 \rfloor} + (\lfloor n/2 \rfloor + 1) 2^{n-1}$, 及 $M_n = 2^{\lfloor n/2 \rfloor + 1} M_{\lfloor n/2 \rfloor}$ 。这些解是 $A_n = 11 \cdot 2^{n-1+\lceil \lg n \rceil} - 3 \cdot 2^n + 6 \cdot 2^n S_n$, $D_n = 4 \cdot 2^{n-1+\lceil \lg n \rceil} - 2 \cdot 2^n + 2 \cdot 2^n S_n$, $M_n = 3 \cdot 2^{n-1+\lceil \lg n \rceil}$; 这里 S_n 满足递推式 $S_1 = 0, S_n = 2S_{\lfloor n/2 \rfloor} + \lfloor n/2 \rfloor$, 而且不难证明对于所有 $n \geq 1$ 不等式 $\frac{1}{2} n \lceil \lg n \rceil \leq S_n \leq S_{n+1} \leq \frac{1}{2} n \lg n + n$ 。算法 C 和算法 N 差不多做同样数量的工作。

60. a) 例如, 在 Σ_1 中, 我们可以把所有有公共的 j 和 k 值的项组成一个三线性项; 当 $(j, k) \in E \times E$ 时, 这给出 ν^2 个三线性项, 当 $(j, k) \in E \times O$ 时加上 ν^2 项, 且当 $(j, k) \in O \times E$ 时再加 ν^2 项。当 $j = k$ 时, 我们也可以不花代价地在 Σ_1 中包括 $-x_{jj} y_{jj} z_{jj}$ 。[在 $n = 10$ 的情况下, 这个方法用 710 个非可交换的乘法乘 10×10 矩阵, 这和在习题 12 的答案中引用的 Makarov 的方法等于 7 个 5×5 矩阵乘法几乎同样好, 尽管当允许交换时 Winograd 的方案 (35) 仅使用 600 次乘法。通过一个类似的方案, Pan 第一次证明, 对于所有很大的 $n, M(n) < n^{2.8}$, 因而这唤起了对于这个问题的强烈兴趣。参见 SICOMP 9 (1980), 321~342。]

b) 这里我们简单地令 S 是一个问题的所有下标 (i, j, k) , \bar{S} 是另一个问题的下标。[当 $m = n = s = 10$ 时, 结果是十分令人惊讶的: 我们可以用 1300 个非可交换乘法来乘两个分开的 10×10 矩阵, 然而却还不知道有用 650 个乘法来乘它们每一个的算法。]

61. a) 用 $ua_{il}(u)$ 代替 $a_{il}(u)$ 。b) 在长度为 $r = \text{rank}_d(t_{ijk})$ 的一个多项式实现中命 $a_{il}(u) = \sum_{\mu} a_{i\mu} u^{\mu}$, 等等。于是 $t_{ijk} = \sum_{\mu+\nu+\sigma=d} \sum_{l=1}^r a_{i\mu} b_{j\nu} c_{kl\sigma}$ 。[这一结果在无限域中可改进成为 $\text{rank}(t_{ijk}) \leq (2d+1)\text{rank}_d(t_{ijk})$, 因为 Bini 和 Pan 指出, 三线性形式 $\sum_{\mu+\nu+\sigma=d} a_{\mu} b_{\nu} c_{\sigma}$ 对应于多项式 modulo u^{d+1} 的乘法。参见 Calcolo 17 (1980), 87~97。] c), d) 由习题 48 中的一些实现看, 这是显然的。

e) 假设我们有 t 和 rt' 的实现, 使得 $\sum_{l=1}^r a_{il} b_{jl} c_{kl} = t_{ijk} u^d + O(u^{d+1})$ 和 $\sum_{L=1}^R A_{\langle ii' \rangle L} B_{\langle jj' \rangle L} C_{\langle kk' \rangle L} = [i = j = k] t'_{i'j'k'} u^{d'} + O(u^{d'+1})$ 。于是

$$\sum_{L=1}^R \sum_{l=1}^r a_{il} A_{\langle ii' \rangle L} \sum_{m=1}^r b_{jm} B_{\langle mj' \rangle L} \sum_{n=1}^r c_{kn} C_{\langle nk' \rangle L} = t_{ijk} t'_{i'j'k'} u^{d+d'} + O(u^{d+d'+1})$$

62. 对于 $P = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, 使用定理 W 的证明方法可得秩为 3。边界秩不能是 1, 因

为我们不能有 $a_1(u)b_1(u)c_1(u) \equiv a_1(u)b_2(u)c_2(u) \equiv u^d$ 和 $a_1(u)b_2(u)c_1(u) \equiv a_1(u)b_1(u)c_2(u) \equiv 0 \pmod{u^{d+1}}$ 。由于实现 $\begin{pmatrix} 1 & 1 \\ u & 0 \end{pmatrix}, \begin{pmatrix} u & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & -1 \\ 0 & u \end{pmatrix}$, 故边界秩为 2。

边界秩的概念是由 Bini, Capovani, Lotti 和 Romani 在 *Information Processing Letters* 8 (1979), 234~235 中引进的。

63. a) 设 $T(m, n, s)$ 和 $T(M, N, S)$ 的元素分别用 $t_{\langle i, j' \rangle \langle j, k' \rangle \langle k, i' \rangle}$ 和 $T_{\langle I, J' \rangle \langle J, K' \rangle \langle K, I' \rangle}$ 标记。直接积的每个元素 $\square_{\langle \mathcal{I}, \mathcal{J}' \rangle \langle \mathcal{J}, \mathcal{K}' \rangle \langle \mathcal{K}, \mathcal{I}' \rangle}$ (其中 $\mathcal{I} = \langle i, I \rangle, \mathcal{J} = \langle j, J \rangle$ 和 $\mathcal{K} = \langle k, K \rangle$) 由定义等于 $t_{\langle i, j' \rangle \langle j, k' \rangle \langle k, i' \rangle} \times T_{\langle I, J' \rangle \langle J, K' \rangle \langle K, I' \rangle}$ 。所以它是 $[\mathcal{I}' = \mathcal{I}$ 和 $\mathcal{J}' = \mathcal{J}$ 和 $\mathcal{K}' = \mathcal{K}]$ 。

b) 以 $M(N) = \text{rank}_0(T(N, N, N))$ 应用习题 61e)。

c) 我们有 $M_d(mns) \leq r^3$, 因为 $T(mns, mns, mns) = T(m, n, s) \otimes T(n, s, m) \otimes T(s, m, n)$ 。如果 $M(n) \leq R$, 对于所有 h , 我们有 $M(n^h) \leq R^h$, 而且由此得出 $M(N) \leq M(n^{\lceil \log_n N \rceil}) \leq R^{\lceil \log_n N \rceil} \leq RN^{\log R / \log n}$ 。[这一结果出现于 Pan 1972 年的文章。]

d) 对于某个 d 我们有 $M_d(mns) \leq r^3$, 其中 $M_d(n) = \text{rank}_d(T(n, n, n))$ 。如果 $M_d(n) \leq R$, 则对所有 h 我们有 $M_{hd}(n^h) \leq R^h$, 而且因为由习题 61b), $M(n^h) \leq \binom{hd+2}{2} R^h$, 所以得出所述公式。在无限域中, 我们节省了 $\log N$ 的一个因子。[这一结果是由 Bini 和 Schönhage 于 1979 年给出的。]

64. 当 $f_k(u) = (x_{k1} + u^2 x_{k2})(y_{2k} + u^2 y_{1k})z_{kk} + (x_{k1} + u^2 x_{k3})y_{3k}((1+u)z_{kk} - u(z_{k1} + z_{k2} + z_{k3})) - x_{k1}(y_{2k} + y_{3k})(z_{k1} + z_{k2} + z_{k3})$ 和 $g_{jk}(u) = (x_{k1} + u^2 x_{j2})(y_{2k} - uy_{1j})(z_{kj} - uz_{jk}) + (x_{k1} + u^2 x_{j3})(y_{2k} + uy_{1j})z_{kj}$ 时, 我们有 $\sum_k (f_k(u) + \sum_{j \neq k} g_{jk}(u)) = u^2 \sum_{1 \leq i, j, k \leq 3} x_{ij} y_{jk} z_{ki} + O(u^3)$ 。[对于 $\text{rank}(T(3, 3, 3))$ 已知的最好上限是 23; 参见习题 12 的答案。 $T(2, 2, 2)$ 的边界秩仍未知。]

65. 提示中的多项式是 $u^2 \sum_{i=1}^m \sum_{j=1}^n (x_{ij} y_{ij} z + X_{ij} Y_{ij} Z) + O(u^3)$ 。对于 $1 \leq i < m$ 和 $1 \leq j < n$ 令 X_{ij} 和 Y_{ij} 是不确定的; 还置 $X_{in} = Y_{mj} = 0, X_{mj} = -\sum_{i=1}^{m-1} X_{ij}, Y_{in} = -\sum_{j=1}^{n-1} Y_{ij}$ 。于是, 对于不确定量中的 $mn+1$ 个多项式乘法, 对于每个 i 和 j 我们可以计算 $x_{ij} y_{ij}$ 以及 $\sum_{i=1}^m \sum_{j=1}^n X_{ij} Y_{ij} - \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} X_{ij} Y_{ij}$ 。[SICOMP 10 (1981), 434~455。在这篇经典的论文中, Schönhage 还推导了习题 64, 66 及 67 i) 的结果。]

66. a) 设 $\omega = \liminf_{n \rightarrow \infty} \log M(n) / \log n$; 由引理 T, 我们有 $\omega \geq 2$ 。对于所有 $\epsilon > 0$, 有一个 N 使 $M(N) < N^{\omega+\epsilon}$ 。习题 63c) 的论证现在表明, 对于所有充分大的 $N, \log M(n) / \log n < \omega + 2\epsilon$ 。

b) 这是习题 63d) 的一个直接结果。

c) 令 $r = \text{rank}(t), q = (mns)^{\omega/3}, Q = (MNS)^{\omega/3}$ 。给定 $\epsilon > 0$, 对于所有正整数 p , 存在一个整常数 c_ϵ 使得 $M(p) \leq c_\epsilon p^{\omega+\epsilon}$ 。对于每个整数 $h > 0$, 我们有 $t^h =$

$\oplus_k \binom{h}{k} T(m^k M^{h-k}, n^k N^{h-k}, s^k S^{h-k})$, 以及 $\text{rank}(t^h) \leq r^h$ 。给定 h 和 k , 令 $p = \lfloor \binom{h}{k} \rfloor^{1/(\omega+\epsilon)}$ 。于是由习题 63b),

$$\begin{aligned} \text{rank}(T(p m^k M^{h-k}, p n^k N^{h-k}, p s^k S^{h-k})) &\leq \text{rank}(M(p) T(m^k M^{h-k}, n^k N^{h-k}, s^k S^{h-k})) \leq \\ &\text{rank}\left(c, \binom{h}{k} T(m^k M^{h-k}, n^k N^{h-k}, s^k S^{h-k})\right) \leq \\ &c r^h \end{aligned}$$

而且由 b) 得出

$$p^\omega q^k Q^{h-k} = (p m^k M^{h-k} p n^k N^{h-k} p s^k S^{h-k})^{\omega/3} \leq c r^h$$

由于 $p \geq \binom{h}{k}^{1/(\omega+\epsilon)}/2$, 我们有

$$\binom{h}{k} q^h Q^{h-k} \leq \binom{h}{k}^{1/(\omega+\epsilon)} (2p)^\omega q^k Q^{h-k} \leq 2^{h/(\omega+\epsilon)} 2^{\omega c} r^h$$

因此对于所有 h , $(q+Q)^h \leq (h+1) 2^{h/(\omega+\epsilon)} 2^{\omega c} r^h$ 。因而由此得出, 对于所有 $\epsilon > 0$, $q+Q \leq 2^{c/(\omega+\epsilon)} r$ 。

d) 在习题 65 中置 $m=n=4$, 并且注意 $10^{0.85} + 9^{0.85} > 17$ 。

67. a) $mn \times mns^2$ 矩阵 $(t_{(ij')(jk')(ki')})$ 有秩 mn , 因为对于 $k=k'=1$ 当把矩阵限制成 mn 行时, 它是排列矩阵。

b) $((t \oplus t')_{i(jk)})$ 实质上是 $(t_{i(jk)}) \oplus (t'_{i(jk)})$, 加上 $n's + sn'$ 个另外的零的列。[类似地对于直接乘积, 我们有 $((t \otimes t')_{i(jk)}) = (t_{i(jk)}) \otimes (t'_{i(jk)})$ 。]

c) 令 D 是对角矩阵 $\text{diag}(d_1, \dots, d_r)$, 使得 $ADB^T = O$ 。由引理 T 我们知道 $\text{rank}(A) = m$ 和 $\text{rank}(B) = n$; 因此 $\text{rank}(AD) = m$, $\text{rank}(DB^T) = n$ 。不失一般性, 我们可以假定 A 的头 m 列线性无关。由于 B^T 的列是在 AD 的零空间中, 因此我们也可以假定, B 的最后 n 列线性无关。把 A 写成分块的形式 (A_1, A_2, A_3) , 其中 A_1 是 $m \times m$ (且非奇异), A_2 是 $m \times q$ 以及 A_3 是 $m \times n$ 。也把 D 分块成使得 $AD = (A_1 D_1 A_2 D_2 A_3 D_3)$ 。于是有一个 $q \times r$ 的矩阵 $W = (W_1 IO)$ 使得 $ADW^T = O$, 即 $W_1 = -D_2 A_2^T A_1^{-T} D_1^{-1}$ 。类似地, 我们可以写 $B = (B_1 B_2 B_3)$, 而且我们发现当 $V = (OIV_3)$ 是 $q \times r$ 矩阵且 $V_3 = -D_2 B_2^T B_3^{-T} D_3^{-1}$ 时 $VDB^T = O$ 。注意 $UDV^T = D_2$, 因此提示被确立(终究, 它差不多只是一个提示)。

现在, 对于 $1 \leq i \leq m$, $A_{(m+i)l}(u) = uv_{il}/d_{m+i}$, 我们令 $A_{il}(u) = a_{il}$; 对于 $1 \leq j \leq n$, $B_{(n+j)l}(u) = w_{jl}u$, 令 $B_{jl}(u) = b_{jl}$; 对于 $1 \leq k \leq s$, $C_{(s+1)l}(u) = d_l$, 令 $C_{kl}(u) = u^2 c_{kl}$ 。由此得出, 如果 $k \leq s$, 则 $\sum_{l=1}^r A_{il}(u) B_{jl}(u) C_{kl}(u)$ 等于 $u^2 t_{ijk} + O(u^3)$, 如果 $k = s+1$, 则等于 $u^2 [i > m][j > n]$ 。[在这个证明中我们不必假定相对于 C , t 是不退化的。]

d) 对于 $r = mn+1$, 考虑 $T(m, 1, n)$ 的如下实现: 如果 $l \leq mn$, 则 $a_{il} = [\lfloor l/n \rfloor$

$= i-1], b_{jl} = [l \bmod n = j], b_{(ij)l} = [l = (i-1)n + j]; a_{ir} = 1, b_{jr} = -1, c_{(ij)r} = 0$ 。
对于 $1 \leq l \leq r$, 对 $d_l = 1$, 这是可改进的。

e) 想法是求出 $T(m, n, s)$ 的一个可改进的实现。假设 (A, B, C) 是长度为 r 的一个实现。给定任意整数 $\alpha_1, \dots, \alpha_m, \beta_1, \dots, \beta_s$, 对于 $1 \leq p \leq n$, 通过定义

$$A_{(i') (r+p)} = \alpha_i [j' = p], \quad B_{(j'') (r+p)} = \beta_k [j = p], \quad C_{(ki') (r+p)} = 0$$

扩充 A, B 和 C 。如果对于 $l \leq r, d_l = \sum_{i=1}^m \sum_{k=1}^s \alpha_i \beta_k c_{(ki')l}$, 否则 $d_l = -1$, 我们有

$$\sum_{i=1}^{r+n} A_{(i')l} B_{(j'')l} d_l = \sum_{i=1}^m \sum_{k=1}^s \alpha_i \beta_k \sum_{l=1}^r A_{(i')l} B_{(j'')l} C_{(ki')l} = \sum_{p=1}^n \alpha_i [j' = p] \beta_k [j = p] = [j = j'] \alpha_i \beta_k - [j = j'] \alpha_i \beta_k = 0$$

所以如果 $d_1 \cdots d_r \neq 0$, 这是可改进的。但 $d_1 \cdots d_r$ 是 $(\alpha_1, \dots, \alpha_m, \beta_1, \dots, \beta_s)$ 中的一个多项式, 不恒等于零, 因为不失一般性, 我们可以假定 C 没有全零的列。因此对于诸 α 和诸 β 的某个选择就行。

f) 如果 $M(n) = n^\omega$, 我们有 $M(n^h) = n^{h\omega}$, 因此

$$\text{rank}(T(n^h, n^h, n^h) \oplus T(1, n^{h\omega} - n^h(2n^h - 1), 1)) \leq n^{h\omega} + n^h$$

习题 66c) 现在意味着对于所有的 $h, n^{h\omega} + (n^{h\omega} - 2n^{2h} + n^h)^{\omega/3} \leq n^{h\omega} + n^h$ 。因此 $\omega = 2$; 但是这同下限 $2n^2 - 1$ 相矛盾 (参见习题 12 的答案)。

g) 令 $f(u)$ 和 $g(u)$ 是使得 $Vf(u)$ 和 $Wg(u)$ 的元素是多项式的多项式。于是我们重新定义

$$A_{(i+m)l} = u^{d+1} v_{il} f(u) / d_{i+m}, \quad B_{(j+n)l} = u^{d+1} w_{jl} g(u) / p, \quad C_{kl} = u^{d+e+2} c_{kl}$$

其中 $f(u)g(u) = pu^e + O(u^{e+1})$ 。由此得出, 如果 $k \leq s$, 则 $\sum_{l=1}^r A_{il}(u) B_{jl}(u) \cdot C_{kl}(u)$ 等于 $u^{d+e+2} t_{ijk} + O(u^{d+e+3})$; 如果 $k = s+1$, 则等于 $u^{d+e+2} [i > m][j > n]$ 。[注: 因此如果以 rank 来代替 rank_2 , 在任何域上, e) 的结果都成立, 因为我们可以选择诸 α 和 β 成为形如 $1 + O(u)$ 的多项式。]

h) 令 C 的行 p 指的是分量 $T(1, 16, 1)$ 。关键之点是对于在删除之后剩下的 i 和 $j, \sum_{l=1}^r a_{il}(u) b_{jl}(u) c_{pl}(u)$ 为零 (而不只是 $O(u^{d+1})$); 而且对于所有 $l, c_{pl}(u) \neq 0$ 。在 c) 和 g) 的构造中, 这些性质为真, 因此当我们取直接积时, 它们也仍保持为真。

i) 直接从二项式推广到多项式。

j) 在 h) 之后, 我们有 $81^{\omega/3} + 2(36^{\omega/3}) + 34^{\omega/3} \leq 100$, 所以 $\omega < 2.52$ 。再次平方给出 $\text{rank}(T(81, 1, 81) \oplus 4T(27, 4, 27) \oplus 2T(9, 34, 9) \oplus 4T(9, 16, 9) \oplus 4T(3, 136, 3) \oplus T(1, 3344, 1)) \leq 10000$; 这产生 $\omega < 2.4999$ 。成功! 连续的平方导致快速收敛到 $2.497723729083 \dots$ 的越来越好的下限。如果我们以 $T(4, 1, 4) \oplus T(1, 9, 1)$ 开始而不是 $T(3, 1, 3) \oplus T(1, 4, 1)$ 开始, 则极限的限已是 $2.51096309 \dots$ 。

[类似的技巧产生 $\omega < 2.496$; 参见 SICOMP 11 (1982), 472~492。]

68. T. M. Vari 通过证明 n 次乘法对计算 $x_1^2 + \dots + x_n^2$ 是必须的, 而证明 $n-1$ 次乘法是必须的 [Cornell Computer Science Report 120 (1972)]。C. Pandu Rangan

证明,如果我们把多项式计算为 $L_1 R_1 + \cdots + L_{n-1} R_{n-1}$, 其中诸 L 和诸 R 是诸 x 的线性组合, 则为构造诸 L 和诸 R , 至少需要 $n-2$ 次加法 [J. Algorithms 4 (1983), 282~285]。但他的下限不明显地适用于所有多项式链。

69. 令 $y_{ij} = x_{ij} - [i=j]$, 并且使用在 n^2 个变量 y_{ij} 的幂级数上的算术, 但忽略掉总次数 $> n$ 的所有项, 应用递归构造(31)到矩阵 $I + Y$ 上。数组的每一元素 h 被表示为一个和 $h_0 + h_1 + \cdots + h_n$, 其中 h_k 是次数为 k 的一个齐次多项式的值。于是每一个加法步变成 $n+1$ 次加法, 而每一个乘法步变成约等于 $\frac{1}{2}n^2$ 次乘法和约等于 $\frac{1}{2}n^2$ 次加法。其次, 每一个除法是形如 $1 + h_1 + \cdots + h_n$ 的一个量, 因为在递归构造中当 y_{ij} 完全为零时所有除法由 1 表示; 因此除法要稍微比乘法容易些 (参见当 $V_0 = 1$ 时的等式 4.7-(3))。由于在我们达到一个 2×2 行列式时就停止了, 当 $j > n-2$ 时, 我们不需要从 y_{ij} 减 1。结果是, 当删去冗余的计算时, 这个方法要求 $20\binom{n}{5} + 8\binom{n}{4} + 12\binom{n}{3} - 4\binom{n}{2} + 5n - 4$ 次乘法和 $20\binom{n}{5} + 8\binom{n}{4} + 4\binom{n}{3} + 24\binom{n}{2} - n$ 次加法, 于是每一个都是 $\frac{1}{6}n^5 - O(n^4)$ 。在许多其它情况下, 一个类似的方法可用来消除除法, 参见 Crelle 264 (1973), 184~202。(但下一习题对于行列式构造一个甚至更快的无除法方案。)

70. 在提示的恒等式中, 置 $A = \lambda - x$, $B = -u$, $C = -v$ 和 $D = \lambda I - Y$, 利用 $I/\lambda + Y/\lambda^2 + Y^2/\lambda^3 + \cdots$ 是 D 的逆这一事实作为 $1/\lambda$ 中的一个形式幂级数, 然后两边取行列式。我们仅对 $0 \leq k \leq n-2$ 需要计算 $uY^k v$, 因为我们知道 $f_X(\lambda)$ 是 n 次多项式。于是, 为从 $n-1$ 次推进到 n 次, 只需要 $n^3 + O(n^2)$ 次乘法和 $n^3 + O(n^2)$ 次加法。递归地处理, 我们在做了 $6\binom{n}{4} + 7\binom{n}{3} + 2\binom{n}{2}$ 次乘法和 $6\binom{n}{4} + 5\binom{n}{3} + 2\binom{n}{2}$ 次加减法之后从 X 的元素得到 f_X 的系数。

如果我们只要计算 $\det X = (-1)^n f_X(0)$, 我们可节省 $3\binom{n}{2} - n + 1$ 次乘法和 $\binom{n}{2}$ 次加法。当 n 有中等大小时, 行列式计算的无除法方法事实上十分经济, 当 $n > 4$ 时它击败显见的余子式展开方案。

如果 ω 是习题 66 中的矩阵乘法的指数, 同样的方法导致一个在 $O(n^{\omega+1})$ 步内的无除法方法, 因为对于 $0 \leq k < n$, 可在 $O(M(n) \log n)$ 步内计算向量 uY^k : 对于 $0 \leq k < 2^l$, 取其头 2^l 个行为 uY^k 的一个矩阵, 并以 Y^{2^l} 乘之; 于是对于 $2^l < k < 2^{l+1}$, 这个乘积的头 2^l 行是 uY^k 。[参见 S. J. Berkowitz, Inf. Processing Letters 18 (1984),

147~150。]当然这样渐近地“快”的矩阵乘法严格地仅是有理论价值。E. Kaltofen 已经说明怎样只以 $O(n^{2+\sqrt{M(n)}})$ 次加法, 减法和乘法来计算行列式 [Proc. Int. Symp. Symb. Alg. Comp. 17 (1992), 342~349]; 即使当 $M(n) = n^3$ 时他的方法都令人感兴趣。

71. 假设 $g_1 = u_1 \circ v_1, \dots, g_r = u_r \circ v_r$, 以及 $f = \alpha_1 g_1 + \dots + \alpha_r g_r + p_0$, 其中 $u_k = \beta_{k1} g_1 + \dots + \beta_{k(k-1)} g_{k-1} + p_k, v_k = \gamma_{k1} g_1 + \dots + \gamma_{k(k-1)} g_{k-1} + q_k$, 每个 \circ 是“ \times ”或“ $/$ ”, 而每个 p_j 或 g_j 是 x_1, \dots, x_n 中次数 ≤ 1 的一个多项式。对于 $k = r, r-1, \dots, 1$, 计算辅助变量 w_k, y_k, z_k 如下: $w_k = \alpha_k + \beta_{(k+1)k} y_{k+1} + \gamma_{(k+1)k} z_{k+1} + \dots + \beta_{rk} y_r + \gamma_{rk} z_r$, 以及

$$y_k = w_k \times v_k, \quad z_k = w_k \times u_k, \quad \text{如果 } g_k = u_k \times v_k$$

$$y_k = w_k / v_k, \quad z_k = w_k \times u_k, \quad \text{如果 } g_k = u_k / v_k$$

则 $f' = p'_0 + p'_1 y_r + q'_1 z_1 + \dots + p'_r y_r + q'_r z_r$, 其中'表示关于任何 x_1, \dots, x_n 的导数。[W. Baur 和 V. Strassen, Theoretical Comp. Sci. 22 (1983), 317~330。S. Linnainmaa, BIT 16 (1976), 146~160 发表了一个相关的方法, 用来做舍入误差的分析。]如果 $g_r = u_r \times v_r$ 我们节省两个链乘法, 因为 $w_r = \alpha_r$ 。重复此构造以至多 $9m + 3d$ 个链乘法和 $4d$ 次除法给出所有二阶偏微分。

72. 在像复数这样代数封闭的域上有一个计算张量的秩的算法, 因为这是 Alfred Tarski 的结果的一个特殊情况, A Decision Method for Elementary Algebra and Geometry, 第2版 (Berkeley, California: Univ. of California Press, 1951); 但除非对于很小的张量, 否则已知的方法并不使该计算真正可行。在有理数域上, 这个问题甚至还不知道在有限时间之内是否可解。

73. 在 N 个变量的这样一个多项式链中, 在 l 个加减法步之后已知的 N 个线性形式的任何 $N \times N$ 矩阵的行列式至多是 2^l 。而且在离散的傅里叶变换中, 最后的 $N = m_1 \cdots m_n$ 个线性形式的矩阵有行列式 $N^{N/2}$, 因为由习题 13, 它的平方是 N 乘以一个排列矩阵 [JACM 20 (1973), 305~306]。

74. a) 如果 $k = (k_1, \dots, k_s)^T$ 是互素整数的一个向量, 则 Uk 也是, 因为 Uk 的元素的任何公因子整除 $k = U^{-1}Uk$ 的所有元素。因此 VUk 不能有全部整分量。

b) 假设有一个带 t 个乘法的 Vx 的多项式链。如果 $t = 0$, V 的元素必是全为整数, 因此 $s = 0$ 。否则设 $\lambda_i = \alpha \times \lambda_k$ 或 $\lambda_i = \lambda_j \times \lambda_k$ 是头一个乘法步。我们可以假定 $\lambda_k = n_1 x_1 + \dots + n_s x_s + \beta$, 其中 n_1, \dots, n_s 是不全为零的整数, 而 β 是常数。求一个单模矩阵 U 使得 $(n_1, \dots, n_s)U = (0, \dots, 0, d)$, 其中 $d = \gcd(n_1, \dots, n_s)$ 。(在等式 4.5.2-(14)前边讨论的算法隐含了这样一个 U 的定义。)对于输入 y_1, \dots, y_{s-1} 构造一个新的多项式链如下: 首先计算 $x = (x_1, \dots, x_s)^T = U(y_1, \dots, y_{s-1}, -\beta/d)^T$, 然后以假定的 Vx 的多项式链继续。当达到该链的步骤 i 时, 我们将有 $\lambda_k = (n_1, \dots, n_s)x + \beta = 0$, 所以我们可以简单地置 $\lambda_i = 0$ 以代替乘法。在计算了 Vx 之后, 把常数向量 $w\beta/d$ 加到结果上, 其中 w 是 VU 的最右列, 并设 W 是 VU 的其它 $s-1$ 列。

新的多项式链已经以 $t-1$ 次乘法计算 $Vx + w\beta/d = VU(y_1, \dots, y_{s-1}, -\beta/d)^T + w\beta/d = W(y_1, \dots, y_{s-1})^T$ 。但由 a), W 的列是 Z 无关的; 因此, 通过对 s 的归纳法, $t-1 \geq s-1$, 因此我们有 $t \geq s$ 。

c) 设对于不在 Z -无关列的集合中的 j 的 $t-s$ 个值, $x_j = 0$ 。 Vx 的任何链于是计算 b) 适用的对于矩阵 V' 的 $V'x'$ 。

d) $\lambda_1 = x - y, \lambda_2 = \lambda_1 + \lambda_1, \lambda_3 = \lambda_2 + x, x_4 = (1/6) \times \lambda_3, \lambda_5 = \lambda_4 + \lambda_4, \lambda_6 = \lambda_5 + y (= x + y/3), \lambda_7 = \lambda_6 - \lambda_1, \lambda_8 = \lambda_7 + \lambda_4 (= x/2 + y)$ 。但是 $\{x/2 + y, x + y/2\}$ 需要两个乘法, 因为 $\begin{pmatrix} 1/2 & 1 \\ 1 & 1/2 \end{pmatrix}$ 的列是 Z 无关的。 [Journal of Information Processing 1 (1978), 125~129。]

4.7 节

1. 如同在 (4) 中那样, 求头一个非零系数 V_m , 并用 z^m 除 $U(z)$ 和 $V(z)$ (把系数左移 m 位)。商将是一个幂级数当且仅当 $U_0 = \dots = U_{m-1} = 0$ 。

2. 我们有 $V_0^{n+1} W_n = V_0^n U_n - (V_0^1 W_0)(V_0^{n-1} V_n) - (V_0^2 W_1)(V_0^{n-2} V_{n-1}) - \dots - (V_0^n W_{n-1})(V_0^1 V_1)$ 。于是, 我们可以以对于 $j \geq 1$, 用 $(V_0^j U_j, V_0^{j-1} V_j)$ 代替 (U_j, V_j) 开始, 然后对于 $n \geq 0$ 置 $W_n \leftarrow U_n - \sum_{k=0}^{n-1} W_k V_{n-k}$, 最后对于 $j \geq 0$, 用 W_j/V_0^{j+1} 代替 W_j 。类似的一些技术也可用于本节中的其它算法。

3. 是。当 $\alpha = 0$ 时, 由归纳法容易证明 $W_1 = W_2 = \dots = 0$ 。当 $\alpha = 1$ 时, 由漂亮的恒等式

$$\sum_{k=1}^n \left(\frac{k - (n-k)}{n} \right) V_k V_{n-k} = V_n V_0$$

我们发现 $W_n = V_n$ 。

4. 如果 $W(z) = e^{V(z)}$, 则 $W'(z) = V'(z)W(z)$; 我们发现 $W_0 = e^{V_0}$, 且

$$W_n = \sum_{k=1}^n \frac{k}{n} V_k W_{n-k}$$

其中 $n \geq 1$ 。如果 $W(z) = \ln V(z)$, 则 V 和 W 的作用被颠倒; 因此当 $V_0 = 1$ 时, 规则是 $W_0 = 0$ 以及对于 $n \geq 1$, $W_n = V_n + \sum_{k=1}^{n-1} (k/n - 1) V_k W_{n-k}$ 。

[由习题 6, 在 $O(n \log n)$ 次运算之下此算法可达到阶 n 。R. P. Brent 发现通过把牛顿方法应用于 $f(x) = \ln x - V(z)$ 上, 可以以这个渐近速度计算 $\exp(V(z))$; 因此一般的乘幂 $(1 + V(z))^\alpha = \exp(\alpha \ln(1 + V(z)))$ 也是 $O(n \log n)$ 。参考文献: J. F. Traub 编辑, *Analytic Computational Complexity* (New York: Academic Press, 1975), 172~176。]

5. 我们再次回到原来的级数, 这可以用来测试一个反演算法。

6. $\phi(x) = x + x(1 - xV(z))$; 参考算法 4.3.3R。于是在知道 W_0, \dots, W_{N-1} 之

后,办法是输入 V_N, \dots, V_{2N-1} , 计算 $(W_0 + \dots + W_{N-1}z^{N-1}) \times (V_0 + \dots + V_{2N-1} \cdot z^{2N-1}) = 1 + R_0z^N + \dots + R_{N-1}z^{2N-1} + O(z^{2N})$, 并设 $W_N + \dots + W_{2N-1}z^{N-1} = -(W_0 + \dots + W_{N-1}z^{N-1})(R_0 + \dots + R_{N-1}z^{N-1}) + O(z^N)$ 。 [Numer. Math. 22 (1974), 341~348; 实质上这个算法首先是由 M. Sieveking 发表的, 见 Computing 10 (1972), 153~156.] 注意, 如果我们使用“快速”多项式乘法(习题 4.6.4-57), 则 N 个系数共需时间 $O(N \log N)$ 。

7. 当 $n = (m-1)k+1$ 时, $W_n = \binom{mk}{k}/n$, 否则为 0。(参考习题 2.3.4.4-11。)

8. G1. 输入 G_1 和 V_1 ; 置 $n \leftarrow 1, U_0 \leftarrow 1/V_1$; 输出 $W_1 = G_1 U_0$ 。

G2. n 加 1。如果 $n > N$, 则结束算法; 否则输入 V_n 和 G_n 。

G3. 对于 $k=0, 1, \dots, n-2$ (以此次序) 置 $U_k \leftarrow (U_k - \sum_{j=1}^k U_{k-j} V_{j+1})/V_1$; 然后置 $U_{n-1} \leftarrow -\sum_{k=2}^n k U_{n-k} V_k / V_1$ 。

G4. 输出 $W_n = \sum_{k=1}^n k U_{n-k} G_k / n$ 并返回 G2。 I

(阶为 N^3 的算法的运行时间因此只增加 N^2 的阶。)

注: 算法 T 和 N 确定 $V^{[-1]}(U(z))$; 在本题中的算法确定 $G(V^{[-1]}(z))$, 它是稍微不同的。当然, 通过一系列的反演和复合运算的序列所有结果都可得到(习题 11), 但是对于每种情况都有更直接的算法是有益的。

9.	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$
T_{1n}	1	1	2	5	14
T_{2n}		1	2	5	14
T_{3n}			1	3	9
T_{4n}				1	4
T_{5n}					1

10. 借助式(9)构造 $y^{1/a} = x(1 + a_1x + a_2x^2 + \dots)^{1/a} = x(1 - c_1x + c_2x^2 + \dots)$; 然后求后边这个级数的逆。(见等式 1.2.11.3-(11)下边的附注。)

11. 对于 $1 \leq k \leq N$ 置 $W_0 \leftarrow U_0$, 并置 $(T_k, W_k) \leftarrow (V_k, 0)$ 。然后对于 $n=1, 2, \dots, N$, 如下操作: 对于 $n \leq j \leq N$, 置 $W_j \leftarrow W_j + U_n T_j$; 而后对于 $j=N, N-1, \dots, n+1$, 置 $T_j \leftarrow T_{j-1} V_1 + \dots + T_n V_{j-n}$ 。

这里 $T(z)$ 表示 $V(z)^N$ 。对于这个问题可构造一个类似于算法 T 的在线幂级数算法, 但是它将要求大约 $N^2/2$ 个存储单元。还有一个在线算法, 它解本题仅仅需要 $O(N)$ 个存储单元: 我们可以假定, 如果对所有 k 用 $U_k V_1^k$ 代替 U_k 和用 V_k/V_1 代替 V_k , 则 $V_1=1$ 。然后我们可以通过算法 L 求 $V(z)$ 的逆, 并使用它的输出作为习题 8 的算法的输入, 且 $G_1 = U_1, G_2 = U_2$ 等等, 这样就可以计算 $U(V^{[-1]}(z))$ 。

$-U_0$ 。也参见习题 20。

Brent 和 Kung 已经构造了好几个算法,它们在渐近的意义更快。例如,我们通过习题 4.6.4-42(c)的一个轻微变形,可以计算对于 $x = V(z)$ 的 $U(x)$,并进行代价为 $M(N)$ 的大约 $2\sqrt{N}$ 个链乘法及代价为 N 的大约 N 个参数乘法,其中 $M(N)$ 是为把幂级数相乘到阶 N 所需要的运算次数;因此共需时间 $O(\sqrt{N}M(N) + N^2) = O(N^2)$ 。一个稍微更快的方法可以以恒等式 $U(V_0(z) + z^m V_1(z)) = U(V_0(z)) + z^m U'(V_0(z)) V_1(z) + z^{2m} U''(V_0(z)) V_1(z)^2/2! + \cdots$ 为基础,一直展开到大约 N/m 项,其中我们选择 $m \approx \sqrt{N/\log N}$;使用稍微类似于习题 4.6.4-43 中的一个算法可在 $O(mN(\log N)^2)$ 个运算内计算头一项 $U(V_0(z))$ 。由于通过微分和除以 $V_0'(z)$,我们可以在 $O(N \log N)$ 次运算下从 $U^{(k)}(V_0(z))$ 进行到 $U^{(k+1)}(V_0(z))$,因此整个的过程花费 $O(mN(\log N)^2 + (N/m)N \log N) = O(N \log N)^{3/2}$ 次运算[JACM 25 (1978), 581~595]。

当多项式有 m 个二进位的整系数时,这个算法大约涉及 $N^{3/2+}$ 次 $N(\lg m)$ 个二进位数的乘法,所以总共的运行时间将多于 $N^{5/2}$ 。P. Ritzmann [Theoretical Comp. Sci. 44 (1986), 1~16] 已经提出具有渐近运行时间 $O(N^{2+})$ 的另一个方法。模一个小的素数 p 可以更快得多地完成复合运算(参见习题 26)。

12. 多项式除法非常简单,除非 $m \geq n \geq 1$ 。在后一种情况下,等式 $u(x) = q(x) \cdot v(x) + r(x)$ 等价于 $U(z) = Q(z) V(z) + z^{m-n+1} R(z)$, 其中 $U(x) = x^m u(x^{-1})$, $V(x) = x^n v(x^{-1})$, $Q(x) = x^{m-n} q(x^{-1})$, 且 $R(x) = x^{n-1} r(x^{-1})$ 是 u, v, q 和 r 的“逆”多项式。

为了求 $q(x)$ 和 $r(x)$, 计算幂级数 $U(z)/V(z) = W(z) + O(z^{m-n+1})$ 的头 $m-n+1$ 个系数;然后计算幂级数 $U(z) - V(z)W(z)$, 它有 $z^{m-n+1} T(z)$ 的形式, 其中 $T(z) = T_0 + T_1 z + \cdots$ 。注意, 对于所有 $j \geq n$, $T_j = 0$; 因此 $Q(z) = W(z)$ 和 $R(z) = T(z)$ 满足要求。

13. 以 $u(z) = z^N$ 和 $v(z) = W_0 + \cdots + W_{N-1} z^{N-1}$ 应用习题 4.6.1-3, 所求近似式是在此算法运行期间得到的 $v_3(z)/v_2(z)$ 的值。习题 4.6.1-26 告诉我们, 对于互素的分子和分母没有进一步的可能性。如果每个 W_i 是一个整数, 则算法 4.6.1C 的一个全整数的扩充将有所求的性质。

注: 关于进一步的信息, 参见 Claude Brezinski, *History of Continued Fractions and Padé Approximants* (Berlin: Springer, 1991)。情况 $N = 2n + 1$ 和 $\deg(w_1) = \deg(w_2) = n$ 是特别有趣的, 因为它等价于所谓的 Toeplitz 系统; 在 Bini 和 Pan 的 *Polynomial and Matrix Computations 1* (Boston: Birkhäuser, 1994) §2.5 中, 对 Toeplitz 系统的渐近快速方法做了综述。本题的方法可以推广到形式 $W(z) \equiv p(z)/q(z) \pmod{(z-z_1) \cdots (z-z_N)}$ 的任意有理内插, 其中诸 z_i 不需要不同; 于是, 我们能确定 $W(z)$ 的值及它在若干点的某些导数。见 Richard P. Brent, Fred G. Gustavson 和 David Y. Y. Yun, *J. Algorithms* 1 (1980), 259~295。

14. 如果 $U(z) = z + U_k z^k + \dots$ 和 $V(z) = z^k + V_{k+1} z^{k+1} + \dots$, 则我们求得差 $V(U(z)) - U'(z)V(z)$ 是 $\sum_{j \geq 1} z^{2k+1+j-1} j(U_k V_{k+j} - U_{k+j})$ (仅含 $U_k, \dots, U_{k+j-1}, V_{k+1}, \dots, V_{k+j-1}$ 的多项式); 因此给定 $U(z)$, 则 $V(z)$ 是惟一的, 而如果给定 $V(z)$ 和 U_k , 则 $U(z)$ 是惟一的。

这个解依赖于两个辅助算法, 它们的头一个对于 $V(z) = V_0 + V_1 z + \dots + V_{n-1} z^{n-1}$ 解方程 $V(z + z^k U(z)) = (1 + z^{k-1} W(z)) V(z) + z^{k-1} S(z) + O(z^{k-1+n})$, 其中 $U(z), W(z), S(z)$ 和 n 是给定的。如果 $n=1$, 令 $V_0 = -S(0)/W(0)$; 或者当 $S(0) = W(0) = 0$ 时令 V_0 为任意值。为了从 n 进行到 $2n$, 令

$$V(z + z^k U(z)) = (1 + z^{k-1} W(z)) V(z) + z^{k-1} S(z) - z^{k-1+n} R(z) + O(z^{k-1+2n})$$

$$1 + z^{k-1} \hat{W}(z) = (z/(z + z^k U(z)))^n (1 + z^{k-1} W(z)) + O(z^{k-1+n})$$

$$\hat{S}(z) = (z/(z + z^k U(z)))^n R(z) + O(z^n)$$

并令 $\hat{V}(z) = V_n + V_{n+1} z + \dots + V_{2n-1} z^{n-1}$ 满足

$$\hat{V}(z + z^k U(z)) = (1 + z^{k-1} \hat{W}(z)) \hat{V}(z) + z^{k-1} \hat{S}(z) + O(z^{k-1+n})$$

第二个算法对于 $U(z) = U_0 + U_1 z + \dots + U_{n-1} z^{n-1}$ 解 $W(z)U(z) + zU'(z) = V(z) + O(z^n)$, 其中 $V(z), W(z)$ 和 n 是给定的。如果 $n=1$, 令 $U_0 = V(0)/W(0)$, 或者在 $V(0) = W(0) = 0$ 的情况下, 令 U_0 是任意的。为了从 n 进行到 $2n$, 设 $W(z)U(z) + zU'(z) = V(z) - z^n R(z) + O(z^{2n})$, 并令 $\hat{U}(z) = U_n + \dots + U_{2n-1} z^{n-1}$ 是方程 $(n + W(z))\hat{U}(z) + z\hat{U}'(z) = R(z) + O(z^n)$ 的解。

恢复(27)的记号, 可以使用头一个算法解 $\hat{V}(U(z)) = U'(z)(z/U(z))^k \hat{V}(z)$ 到任何所希望的精度, 且置 $V(z) = z^k \hat{V}(z)$ 。为了求 $P(z)$, 假设我们有 $V(P(z)) = P'(z)V(z) + O(z^{2k-1+n})$, 这是当 $P(z) = z + \alpha z^k$ 和 α 任意时, 对 $n=1$ 成立的一个方程。通过设 $V(P(z)) = P'(z)V(z) + z^{2k-1+n}R(z) + O(z^{2k-1+2n})$ 和用 $P(z) + z^{k-n}\hat{P}(z)$ 代替 $P(z)$, 我们可以从 n 进行到 $2n$, 其中第二个算法用来求多项式 $\hat{P}(z)$, 使得 $(k+n-zV'(P(z))/V(z))\hat{P}(z) + z\hat{P}'(z) = (z^k/V(z))R(z) + O(z^n)$ 。

15. 微分方程 $U'(z)/U(z)^k = 1/z^k$ 意味着对于某个常数 c , $U(z)^{1-k} = z^{1-k} + c$ 。所以我们求得 $U^{[n]}(z) = z/(1 + cnz^{1-k})^{1/(k-1)}$ 。

一个类似的论证对于任意的 $V(z)$, 求解(27): 如果 $W'(z) = 1/V(z)$, 对于某个 c 我们有 $W(U^{[n]}(z)) = W(z) + nc$ 。

16. 我们要证明, $[t^n]t^{n+1}((n+1)R'_{k+1}(t)/V(t)^n - nR'_k(t)/V(t)^{n+1}) = 0$ 。

由于 $(n+1)R'_{k+1}(t)/V(t)^n - nR'_k(t)/V(t)^{n+1} = \frac{d}{dt}(R_k(t)/V(t)^{n+1})$, 这成立。

因此, 我们有 $n^{-1}[t^{n-1}]R'_1(t)t^n/V(t)^n = (n-1)^{-1}[t^{n-2}]R'_2(t)t^{n-1}/V(t)^{n-1} = \dots = 1^{-1}[t^0]R'_n(t)t/V(t) = [t]R_n(t)/V_1 = W_n$ 。

17. 等置 $x^l y^m$ 的系数, 卷积公式指出 $\binom{l+m}{m} v_{n(l+m)} = \sum_k \binom{n}{k} v_{kl} v_{(n-k)m}$, 它和 $[z^n] V(z)^{l+m} = \sum_k ([z^k] V(z)^l) ([z^{n-k}] V(z)^m)$ 相同, 是(2)的一个特殊情况。

注: “似次幂”这一名称, 是由 J. F. Steffensen 引进的。在许多作者中, 是他头一位一般地研究了这些多项式的令人惊奇的性质 [Acta. Mathematica 73 (1941), 333~366]。关于文献的综述, 以及关于在以下好几道题中的课题的进一步讨论, 参见 D. E. Knuth, *The Mathematica Journal* 2 (1992), 67~78。在该篇论文中的结果之一是, 如果 $V_1=1$ 和 $sV'(s) \triangleq y$ 以及当 $x \rightarrow \infty$ 和 $n \rightarrow \infty$ 时 $y = n/x$ 有界, 则有渐近公式

$$V_n(x) = e^{xV(s)} \left(\frac{n}{es} \right)^n (1 - V_2 y + O(y^2) + O(x^{-1})).$$

18. 我们有 $V_n(x) = \sum_k x^k n! [z^n] V(z)^k / k! = n! [z^n] e^{xV(z)}$ 。因此当 $n > 0$ 时, $V_n(x)/x = (n-1)! [z^{n-1}] V'(z) e^{xV(z)}$ 。通过在 $V'(z) e^{(x+y)V(z)} = V'(z) e^{xV(z)} e^{yV(z)}$ 中等置 z^{n-1} 的系数, 我们得到所述的恒等式。

19. 由多项式定理 1.2.6-(42), 我们有

$$v_{nm} = \frac{n!}{m!} [z^n] \left(\frac{v_1}{1!} z + \frac{v_2}{2!} z^2 + \frac{v_3}{3!} z^3 + \cdots \right)^m = \sum_{\substack{k_1+k_2+\cdots+k_n=m \\ k_1+2k_2+\cdots+nk_n=n \\ k_1, k_2, \dots, k_n \geq 0}} \frac{n!}{k_1! k_2! \cdots k_n!} \left(\frac{v_1}{1!} \right)^{k_1} \left(\frac{v_2}{2!} \right)^{k_2} \cdots \left(\frac{v_n}{n!} \right)^{k_n}$$

这些系数也出现在习题 1.2.5-21 的 Arbogast 公式中, 因此如同在该习题的答案中所说明的那样, 我们可以把这些项同集合的分划相关联。递推式

$$v_{nk} = \sum_j \binom{n-1}{j-1} v_j v_{(n-j)(k-1)}$$

表明如何由列 1 和列 $k-1$ 来计算列 k 。由于有 $\binom{n-1}{j-1}$ 种方法在大小为 j 的一个子集中包括元素 n , 这是容易相对于 $\{1, \dots, n\}$ 的分划解释的。矩阵的开头一些行是

$$\begin{array}{cccccc} v_1 & & & & & \\ v_2 & v_1^2 & & & & \\ v_3 & 3v_1v_2 & v_1^3 & & & \\ v_4 & 4v_1v_3 + 3v_2^2 & 6v_1^2v_2 & v_1^4 & & \\ v_5 & 5v_1v_4 + 10v_2v_3 & 15v_1v_2^2 + 10v_1^2v_3 & 10v_1^3v_2 & v_1^5 & \end{array}$$

20. $[z^n] W(z)^k = \sum_j ([z^j] U(z)^k) ([z^n] V(z)^j)$; 因此 $w_{nk} = (n!/k!) \cdot \sum_j ((k!/j!) u_{jk}) ((j!/n!) v_{nj})$ 。[E. Jabotinsky, *Comptes Rendus Acad. Sci. Paris* 224 (1947), 323~324.]

21. a) 如果 $U(z) = \alpha W(\beta z)$, 我们有 $u_{nk} = \frac{n!}{k!} [z^n] (\alpha W(\beta(z)))^k = \alpha^k \beta^n w_{nk}$; 特别是, 如果 $U(z) = V^{[-1]}(z) = -W(-z)$, 我们有 $u_{nk} = (-1)^{n-k} w_{nk}$ 。所以由习题 20, $\sum_k u_{nk} v_{km}$ 和 $\sum_k v_{nk} u_{km}$ 对应于单位函数 z 。

b) [由 Ira Gessel 给出的解。]事实上, 这个恒等式等价于拉格朗日的反演公式: 我们有 $w_{nk} = (-1)^{n-k} u_{nk} = (-1)^{n-k} \frac{n!}{k!} [z^n] V^{[-1]}(z)^k$, 而且由习题 16, 在 $V^{[-1]}(z)^k$ 中 z^n 的系数是 $n^{-1} [t^{n-1}] k t^{n+k-1} / V(t)^n$ 。另一方面, 我们已经把 $v_{(-k)(-n)}$ 定义为 $(-k)^{n-k} [z^{n-k}] (V(z)/z)^n$, 它等于 $(-1)^{n-k} (n-1) \cdots (k+1) k [z^{n-1}] z^{n+k-1} / V(z)^n$ 。

22. a) 如果 $V(z) = U^{[\alpha]}(z)$ 和 $W(z) = V^{[\beta]}(z)$, 我们有 $W(z) = V(zW(z)^\beta) = U(zW(z)^\beta V(zW(z)^\beta)^\alpha) = U(zW(z)^{\alpha+\beta})$ 。(注意这个定律和适合于迭代的类似公式 $U^{[1]}(z) = U(z)$, $U^{[\alpha][\beta]}(z) = U^{[\alpha\beta]}(z)$ 之间的对照。)

b) $B^{(2)}(z)$ 是对于 2.3.4.4-(12) 的二叉树的生成函数, 它是在算法 L 之后的例子 $z = t - t^2$ 中的 $W(z)/z$ 。而且, $B^{[1]}(z)$ 是习题 2.3.4.4-11 的 t 叉树的生成函数。

c) 提示等价于 $zU^{[\alpha]}(z)^\alpha = W^{[-1]}(z)$, 它等价于公式 $zU^{[\alpha]}(z)^\alpha / U(zU^{[\alpha]}(z)^\alpha)^\alpha = z$ 。现在拉格朗日的反演定理(习题 8)指出, 当 x 是正整数时, $[z^n] W^{[-1]}(z)^\alpha = \frac{x}{n} [z^{-x}] W(z)^{-n}$ 。(这里 $W(z)^{-n}$ 是一个 Laurent 级数——一个幂级数除以 z 的幂; 我们可以使用符号 $[z^m] V(z)$ 既表示幂级数也表示 Laurent 级数。)因此当 x/α 是正整数时, $[z^n] U^{[\alpha]}(z)^x = [z^n] (W^{[-1]}(z)/z)^{x/\alpha} = [z^{n+x/\alpha}] \cdot W^{[-1]}(z)^{x/\alpha}$ 等于 $\frac{x/\alpha}{n+x/\alpha} [z^{-x/\alpha}] W(z)^{-n-x/\alpha} = \frac{x}{n\alpha} [z^{-x/\alpha}] z^{-n-x/\alpha} U(z)^{n+x/\alpha}$ 。对于无穷多个 α 我们已经验证了这个结果; 因为 $U^{[\alpha]}(z)^x$ 的系数是 α 的多项式, 因此这是充分的。

在习题 1.2.6-25 和 2.3.4.4-29 中我们已经看到这一结果的特殊情况。提示的一个可记忆的结果是 $\alpha = -1$ 的情况:

$$W(z) = zU(z) \quad \text{当且仅当} \quad W^{[-1]}(z) = z/U^{[-1]}(z)$$

d) 如果 $U_0 = 1$ 和 $V_n(x)$ 是对于 $V(z) = \ln U(z)$ 的似次幂, 我们刚才已经证明 $xV_n(x+n\alpha)/(x+n\alpha)$ 是对于 $\ln U^{[\alpha]}(z)$ 的似次幂。所以我们可以把这个似次幂插入到以前的恒等式, 在第二个公式中把 y 改成 $y - \alpha n$ 。

23. a) 我们有 $U = I + T$, 其中 T^n 在小于等于 n 的行中为零。因此 $\ln U = T - \frac{1}{2}T^2 + \frac{1}{3}T^3 - \cdots$ 将有 $\exp(\alpha \ln U) = I + \binom{\alpha}{1}T + \binom{\alpha}{2}T^2 + \cdots = U^\alpha$ 的性质。 U^α 的每个元素是 α 的一个多项式, 而且每当 α 是正整数时习题 19 的关系成立; 因此对于所有 α , U^α 是一个幂矩阵, 而且它的头一行定义 $U^{[\alpha]}(z)$ 。(特别是, U^{-1} 是一个幂矩阵; 这是反演 $U(z)$ 的另一个方法。)

b) 因为 $U = I + \epsilon \ln U + O(\epsilon^2)$, 我们有

$$l_{nk} = [\epsilon] u_{nk}^{[\epsilon]} = \frac{n!}{k!} [z^n] [\epsilon] (z + \epsilon L(z) + O(\epsilon^2))^k = \frac{n!}{k!} [z^n] k z^{k-1} L(z)$$

c) $\frac{\partial}{\partial \alpha} U^{[\alpha]}(z) = [\epsilon] U^{[\alpha+\epsilon]}(z)$, 而且我们有

$$U^{[\alpha+\epsilon]}(z) = U^{[\alpha]}(U^{[\epsilon]}(z)) = U^{[\alpha]}(z + \epsilon L(z) + O(\epsilon^2))$$

还有

$$U^{[\alpha+\epsilon]}(z) = U^{(\epsilon)}(U^{[\alpha]}(z)) = U^{[\alpha]}(z) + \epsilon L(U^{[\alpha]}(z)) + O(\epsilon^2)$$

d) 由 U 同 $\ln U$ 可交换这一事实, 得出这个恒等式。当 $n \geq 4$ 时它确定 l_{n-1} ,

因为左边 l_{n-1} 的系数是 nu_2 , 而右边的系数是 $u_{n(n-1)} = \binom{n}{2} u_2$ 。类似地, 如果 $u_2 = \dots = u_{k-1} = 0$ 和 $u_k \neq 0$, 我们有 $l_k = u_k$, 而且对于 $n \geq 2k$ 的递推确定 l_{k+1}, l_{k+2}, \dots :

左边有 $l_n + \binom{n}{k-1} l_{n+1-k} u_k + \dots$ 的形式, 而右边有 $l_n + \binom{n}{k} l_{n+1-k} u_k + \dots$ 的形式。

一般说来, $l_2 = u_2, l_3 = u_3 - \frac{3}{2} u_2^2, l_4 = u_4 - 5u_2 u_3 + \frac{9}{2} u_2^3, l_5 = u_5 - \frac{15}{2} u_2 u_4 - 5u_3^2 + \frac{185}{6} u_2^3 u_3 - 20u_2^4$ 。

e) 我们有 $U = \sum_m (\ln U)^m / m!$, 而且对于固定的 m 从第 m 项起对 $u_n = u_{n1}$ 的贡献是对于 $n = n_m > \dots > n_1 > n_0 = 1$ 求和的 $\sum l_{n_m n_{m-1}} \dots l_{n_2 n_1} l_{n_1 n_0}$ 。现在应用 b) 的结果。[参见 *Trans. Amer. Math. Soc.* **108** (1963), 457~477。]

24. a) 由 (21) 和习题 20, 我们有 $U = VDV^{-1}$, 其中 V 是 Schröder 函数的幂矩阵, 而 D 是对角矩阵 $\text{diag}(u, u^2, u^3, \dots)$ 。所以我们可以取 $\ln U = V \text{diag}(\ln u, 2 \ln u, 3 \ln u, \dots) V^{-1}$ 。b) 等式 $WVDV^{-1} = VDV^{-1}W$ 意味着 $(V^{-1}WV)D = D(V^{-1}WV)$ 。 D 的对角元素是不同的, 所以 $V^{-1}WV$ 必定是对角矩阵 D' 。因此 $W = VD'V^{-1}$ 且 W 有和 U 一样的 Schröder 函数。由此得出, $W_1 \neq 0$ 和 $W = VD^\alpha V^{-1}$, 其中 $\alpha = (\ln W_1) / (\ln U_1)$ 。

25. 我们必定有 $k = l$, 因为 $[z^{k+l-1}] U(V(z)) = U_{k+l-1} + V_{k+l-1} + k U_k V_l$ 。为完成证明, 只须证明 $U_k = V_k$ 和 $U(V(z)) = V(U(z))$ 意味着 $U(z) = V(z)$ 。假设 l 是满足 $U_l \neq V_l$ 的极小值, 并令 $n = k + l - 1$ 。于是我们有 $u_{nk} - v_{nk} = \binom{n}{l} (u_l - v_l)$; 对于所有 $j > k$, $u_{nj} = v_{nj}$; $u_{nl} = \binom{n}{k} u_k$; 以及对于 $1 < j < n$, $u_{nj} = 0$ 。现在和 $\sum_j u_{nj} v_j = u_n + u_{nk} v_k + \dots + u_{nl} v_l + v_n$ 必然等于 $\sum_j v_{nj} u_j$; 所以我们求得 $\binom{n}{l} (u_l - v_l) v_k = \binom{n}{k} v_k (u_l - v_l)$ 。但我们有 $\binom{k+l-1}{k} = \binom{k+l-1}{l}$ 当且仅当 $k = l$ 。

[由本习题和上一习题, 我们可以猜测仅当 U 和 V 之一是另一个的迭代时,

$U(V(z)) = V(U(z))$ 。但当 U_1 和 V_1 是单位的根时,这并非必要。例如,如果 $V_1 = -1$ 和 $U(z) = V^{[2]}(z)$, V 不是 $U^{[1/2]}$ 的迭代, $U^{[1/2]}$ 也不是 V 的迭代。]

26. 写 $U(z) = U_{[0]}(z^2) + zU_{[1]}(z^2)$, 我们有 $U(V(z)) \equiv U_{[0]}(V_1 z^2 + V_2 z^4 + \cdots) + V(z)U_{[1]}(V_1 z^2 + V_2 z^4 + \cdots) \pmod{2}$ 。运行时间满足 $T(N) = 2T(N/2) + C(N)$, 其中 $C(N)$ 实质上是模 2^N 多项式乘法的时间。通过比如说习题 4.6.4-59 的方法,我们可以使 $C(N) = O(N^{1+\epsilon})$; 也可参见习题 4.6-5 的答案。

一个类似的方法在 $O(pN^{1+\epsilon})$ 的时间内进行模 p 的运算。[D. J. Bernstein, *J. Symbolic Computation* **26** (1998), 339~341。]

27. 由 $(W(qz) - W(z))V(z) = W(z)(V(q^m z) - V(z))$, 我们得到递推式 $W_n = \sum_{k=1}^n V_k W_{n-k} (q^{km} - q^{n-k}) / (q^n - 1)$ 。[*J. Difference Eqs. and Applics.* **16** (1995), 57~60。]

28. 首先注意 $\delta(U(z)V(z)) = (\delta U(z))V(z) + U(z)(\delta V(z))$, 因为 $t(mn) = t(m) + t(n)$ 。因此对于所有 $n \geq 0$, 由对 n 的归纳法, $\delta(V(z)^n) = nV(z)^{n-1}\delta V(z)$; 而这是我们要来证明 $\delta e^{V(z)} = \sum_{n \geq 0} \delta(V(z)^n/n!) = e^{V(z)}\delta V(z)$ 的恒等式。在这个等式中以 $\ln V(z)$ 代替 $V(z)$ 给出 $V(z)\delta \ln V(z) = \delta V(z)$; 因此对于所有的复数, $\delta(V(z)^\alpha) = \delta e^{\alpha \ln V(z)} = e^{\alpha \ln V(z)} \delta(\alpha \ln V(z)) = \alpha V(z)^{\alpha-1}$ 。

由此得出所想求的递推式是

$$(a) \quad W_1 = 1, W_n = \sum_{d \setminus n, d > 1} ((\alpha + 1)t(d)/t(n) - 1) V_d W_{n/d};$$

$$(b) \quad W_1 = 1, W_n = \sum_{d \setminus n, d > 1} (t(d)/t(n)) V_d W_{n/d};$$

$$(c) \quad W_1 = 0, W_n = V_n + \sum_{d \setminus n, d > 1} (t(d)/t(n) - 1) V_d W_{n/d}.$$

[参见 H. W. Gould, *AMM* **81** (1974), 3~14。当 t 是任何使得 $t(m) + t(n) = t(mn)$ 和 $t(n) = 0$ 的函数时这些公式成立当且仅当 $n = 1$, 但所建议的 t 是最简单的。这里讨论的方法对于任意多个变量的幂级数也有效; 于是 t 是一项的总次数。]

"It is certainly an idea you have there," said Poirot, with some interest.

"Yes, yes, I play the part of the computer.

One feeds in the information—"

"And supposing you come up with all the wrong answers?" said Mrs. Oliver.

"That would be impossible," said Hercule Poirot.

"Computers do not do that sort of a thing."

"They're not supposed to," said Mrs. Oliver,

"but you'd be surprised at the things that happen sometimes."

"你说的还挺有道理," Poirot 饶有兴趣地说。

"是的, 是的, 我还真像是台计算机。"

"你输入信息——"

“要是你出来的结果是错的呢？”Oliver 夫人问。
“那不可能，”Hercule Poirot 答道，“计算机不会出错的。”
“应该说不会，”Oliver 夫人说，“但有时结果会出乎意料。”
—AGATHA CHRISTIE, *Hallowe'en Party* (1969)*

* 此段对话出自阿加莎·克里斯蒂侦探小说《万圣节前夜的谋杀案》第6章。——本书责任编辑

附录 A 数值数量表

表 1 在标准子程序和对计算机的程序的
分析中经常使用的数量(到小数点后 40 位)

$\sqrt{2} =$	1.41421	35623	73095	04880	16887	24209	69807	85697 -
$\sqrt{3} =$	1.73205	08075	68877	29352	74463	41505	87236	69428 +
$\sqrt{5} =$	2.23606	79774	99789	69640	91736	68731	27623	54406 +
$\sqrt{10} =$	3.16227	76601	68379	33199	88935	44432	71853	37196 -
$\sqrt[3]{2} =$	1.25992	10498	94873	16476	72106	07278	22835	05703 -
$\sqrt[3]{3} =$	1.44224	95703	07408	38232	16383	10780	10958	83919 -
$\sqrt[3]{2} =$	1.18920	71150	02721	06671	74999	70560	47591	52930 -
$\ln 2 =$	0.69314	71805	59945	30941	72321	21458	17656	80755 +
$\ln 3 =$	1.09861	22886	68109	69139	52452	36922	52570	46475 -
$\ln 10 =$	2.30258	50929	94045	68401	79914	54684	36420	76011 +
$1/\ln 2 =$	1.44269	50408	88963	40735	99246	81001	89213	74266 +
$1/\ln 10 =$	0.43429	44819	03251	82765	11289	18916	60508	22944 -
$\pi =$	3.14159	26535	89793	23846	26433	83279	50288	41972 -
$1^\circ = \pi/180 =$	0.01745	32925	19943	29576	92369	07684	88612	71344 +
$1/\pi =$	0.31830	98861	83790	67153	77675	26745	02872	40689 +
$\pi^2 =$	9.86960	44010	89358	61883	44909	99876	15113	53137 -
$\sqrt{\pi} = \Gamma(1/2) =$	1.77245	38509	05516	02729	81674	83341	14518	27975 +
$\Gamma(1/3) =$	2.67893	85347	07747	63365	56929	40974	67764	41287 -
$\Gamma(2/3) =$	1.35411	79394	26400	41694	52880	28154	51378	55193 +
$e =$	2.71828	18284	59045	23536	02874	71352	66249	77572 +
$1/e =$	0.36787	94411	71442	32159	55237	70161	46086	74458 +
$e^2 =$	7.38905	60989	30650	22723	04274	60575	00781	31803 +
$\gamma =$	0.57721	56649	01532	86060	65120	90082	40243	10422 -
$\ln \pi =$	1.14472	98858	49400	17414	34273	51353	05871	16473 -
$\phi =$	1.61803	39887	49894	84820	45868	34365	63811	77203 +
$e^{\gamma} =$	1.78107	24179	90197	98523	65041	03107	17954	91696 +
$e^{\pi/4} =$	2.19328	00507	38015	45655	97696	59278	73822	34616 +
$\sin 1 =$	0.84147	09848	07896	50665	25023	21630	29899	96226 -
$\cos 1 =$	0.54030	23058	68139	71740	09366	07442	97660	37323 +
$-\zeta'(2) =$	0.93754	82543	15843	75370	25740	94567	86497	78979 -
$\zeta(3) =$	1.20205	69031	59594	28539	97381	61511	44999	07650 -
$\ln \phi =$	0.48121	18250	59603	44749	77589	13424	36842	31352 -
$1/\ln \phi =$	2.07808	69212	35027	53760	13226	06117	79576	77422 -
$-\ln \ln 2 =$	0.36651	29205	81664	32701	24391	58232	66946	94543 -

表 2 在标准子程序和对计算机程序的分析中经常使用的数量(到八进制 45 位)
“=”左边的名称是以十进制记号给出的*

0.1 =	0.06314	63146	31463	14631	46314	63146	31463	14631	46315	-
0.01 =	0.00507	53412	17270	24365	60507	53412	17276	24365	60510	-
0.001 =	0.00040	61115	64570	65176	76355	44264	15254	62930	44672	+
0.0001 =	0.00003	21555	13536	70414	54512	75170	39021	15002	35223	-
0.00001 =	0.00000	24761	32610	76664	36041	06077	17401	56063	34417	-
0.000001 =	0.00000	02061	57364	05536	66151	55323	07746	44476	26033	+
0.0000001 =	0.00000	60153	27745	15274	53644	12741	72312	20354	02151	+
0.00000001 =	0.00000	00012	57143	56106	04363	47374	77341	01512	69327	+
0.000000001 =	0.00000	00001	04560	27640	46655	12262	71426	40124	21742	+
0.0000000001 =	0.00000	00000	06676	33766	35367	55653	37265	34042	01627	-
$\sqrt{2}$ =	1.32404	74631	77167	46220	42627	66115	46725	12575	17435	+
$\sqrt{3}$ =	1.56653	65641	30231	25163	54453	50265	66361	34073	42223	-
$\sqrt{5}$ =	2.17667	36334	57722	47602	57471	09003	00563	55620	32021	-
$\sqrt{10}$ =	3.12305	40726	64555	22444	02242	57101	41466	33775	22532	+
$\sqrt[3]{2}$ =	1.20505	65746	15345	05342	10756	65934	25574	22415	03024	+
$\sqrt[3]{3}$ =	1.34233	50444	22175	73134	67363	76193	05934	31147	60121	-
$\sqrt[3]{2}$ =	1.14067	74056	61556	12455	72152	64436	66271	02755	73136	+
$\ln 2$ =	0.54271	02775	75071	73632	57117	07316	30607	71366	53640	+
$\ln 3$ =	1.06237	24752	55606	05227	32446	63065	25012	35574	55337	+
$\ln 10$ =	2.23273	06735	52524	25405	56512	66542	56026	46050	50765	+
$1/\ln 2$ =	1.34252	16624	53405	77627	35750	37766	46644	35175	04353	+
$1/\ln 10$ =	0.39626	75425	11562	41614	52325	33525	27655	14756	06220	-
π =	3.14037	55242	10264	30215	14230	63650	56006	70163	21122	+
$1^\circ = \pi/180$ =	0.01073	72152	11224	72344	25663	54276	63351	22056	11544	+
$1/\pi$ =	0.24276	36155	62344	20251	23760	47257	50765	15156	76067	-
π^2 =	11.67517	14467	62135	71322	25561	15466	30621	46654	34103	-
$\sqrt{\pi} = \Gamma(1/2)$ =	1.61337	61106	64736	65247	47035	40510	15273	34470	17762	-

* 作者计划在本书下一版中给出这些常数的 36 位十六进制值,而不是现在的 45 位八进制值。——本书责任编辑

(续)

$\Gamma(1/3) =$	2.53347	35234	51013	61916	79106	47644	54653	60166	66646 -
$\Gamma(2/3) =$	1.26523	57112	14154	74312	54572	37655	60126	23231	62457 +
$e =$	2.55760	52136	50535	51246	52773	42542	00471	72363	61661 +
$1/e =$	0.27426	53066	13167	46751	52726	75436	02440	52371	03955 +
$e^2 =$	7.38714	45615	23355	33460	53507	35040	32664	25356	50217 +
$\gamma =$	0.44742	14770	67566	06172	23215	74376	01002	51313	25521 -
$\ln \pi =$	1.11209	45443	47503	36413	65372	52661	52410	37511	46052 +
$\phi =$	1.47433	57156	27751	23701	27634	71401	46271	66716	15016 +
$e^{\gamma} =$	1.61772	13452	61152	65761	22477	36553	53327	17554	21260 +
$e^{e^{\gamma}} =$	2.14275	31512	16162	52376	35536	11342	53525	44307	02171 -
$\sin 1 =$	0.65665	24436	04414	73402	03067	25644	11612	07474	14505 -
$\cos 1 =$	0.42456	50037	32406	42711	07022	14666	27320	70675	12321 +
$-\zeta'(2) =$	0.74001	45144	53253	42362	42167	23350	50074	46160	27766 +
$\zeta(3) =$	1.14735	06623	60014	20470	15613	42561	31715	10177	06614 +
$\ln \phi =$	0.36630	26256	61213	01145	13706	41004	52264	30706	40646 +
$1/\ln \phi =$	2.04776	60111	17144	41512	11436	16575	00355	43630	40651 +
$-\ln \ln 2 =$	0.27351	71233	67265	63650	17401	56637	26334	31455	57005 -

在本书的第1版中,表1中的一些40位的数值是由John W. Wrench, Jr. 在一架台式计算机上计算出来的。到了20世纪70年代,当做此计算的计算机软件出现时,证明了他所做的所有计算都是正确的。在等式4.5.2-(60), 4.5.3-(26), 4.5.3-(41), 4.5.4-(9)以及习题4.5.4-8, 4.5.4-25, 4.6.4-58的答案中,可以找到其它基本常数的40位的值。

表3 对于小的 n 值,调和数、伯努利数以及斐波那契数的值

n	H_n	B_n	F_n	n
0	0	1	0	0
1	1	-1/2	1	1
2	3/2	1/6	1	2
3	11/6	0	2	3
4	25/12	-1/30	3	4
5	137/60	0	5	5
6	49/20	1/42	8	6
7	363/140	0	13	7

(续)

n	H_n	B_n	F_n	n
8	761/280	- 1/30	21	8
9	7129/2520	0	34	9
10	7381/2520	5/66	55	10
11	83711/27720	0	89	11
12	86021/27720	- 691/2730	144	12
13	1145993/360360	0	233	13
14	1171733/360360	7/6	377	14
15	1195757/360360	0	610	15
16	2436559/720720	- 3617/510	987	16
17	42142223/12252240	0	1597	17
18	14274301/4084080	43867/798	2584	18
19	275295799/77597520	0	4181	19
20	55835135/15519504	- 174611/330	6765	20
21	18858053/5173168	0	10946	21
22	19093197/5173168	854513/138	17711	22
23	444316699/118982864	0	28657	23
24	1347822955/356948592	- 236364091/2730	46368	24
25	34052522467/8923714800	0	75025	25
26	34395742267/8923714800	8553103/6	121393	26
27	312536252003/80313433200	0	196418	27
28	315404588903/80313433200	- 23749461029/870	317811	28
29	9227046511387/23290895662800	0	514229	29
30	9304682830147/23290895662800	8615841276005/14322	832040	30

对于任意的 x , 命 $H_x = \sum_{n \geq 1} \left(\frac{1}{n} - \frac{1}{n+x} \right)$, 于是

$$H_{1/2} = 2 - 2 \ln 2$$

$$H_{1/3} = 3 - \frac{1}{2}\pi/\sqrt{3} - \frac{3}{2}\ln 3$$

$$H_{2/3} = \frac{3}{2} + \frac{1}{2}\pi/\sqrt{3} - \frac{3}{2}\ln 3$$

$$H_{1/4} = 4 - \frac{1}{2}\pi - 3 \ln 2$$

$$H_{3/4} = \frac{4}{3} + \frac{1}{2}\pi - 3 \ln 2$$

$$H_{1/5} = 5 - \frac{1}{2}\pi\phi^{3/2}5^{-1/4} - \frac{5}{4}\ln 5 - \frac{1}{2}\sqrt{5} \ln \phi$$

$$H_{2/5} = \frac{5}{2} - \frac{1}{2}\pi\phi^{-3/2}5^{-1/4} - \frac{5}{4}\ln 5 + \frac{1}{2}\sqrt{5} \ln \phi$$

$$H_{3/5} = \frac{5}{3} + \frac{1}{2}\pi\phi^{-3/2}5^{-1/4} - \frac{5}{4}\ln 5 + \frac{1}{2}\sqrt{5} \ln \phi$$

$$H_{4/5} = \frac{5}{4} + \frac{1}{2}\pi\phi^{3/2}5^{-1/4} - \frac{5}{4}\ln 5 - \frac{1}{2}\sqrt{5} \ln \phi,$$

$$H_{1/6} = 6 - \frac{1}{2}\pi\sqrt{3} - 2\ln 2 - \frac{3}{2}\ln 3$$

$$H_{5/6} = \frac{6}{5} + \frac{1}{2}\pi\sqrt{3} - 2\ln 2 - \frac{3}{2}\ln 3$$

而且,一般地,当 $0 < p < q$ (参见习题 1.2.9-19) 时,

$$H_{p/q} = \frac{q}{p} - \frac{\pi}{2} \cot \frac{p}{q} \pi - \ln 2q + 2 \sum_{1 \leq n < q/2} \cos \frac{2pn}{q} \pi \cdot \ln \sin \frac{n}{q} \pi$$

附录 B 符号索引

在以下的诸公式中,未被进一步定性的字母有如下的意义:

j, k 整数值算术表达式

m, n 非负整数值算术表达式

x, y 实数值算术表达式

z 复数值算术表达式

f 实数或复数值函数

S, T 集合或多重集合

形式符号	意 义	定义的位置
 	算法,程序或证明的结束	1.1
A_n 或 $A[n]$	线性数组 A 的第 n 个元素	1.1
A_{mn} 或 $A[m, n]$	矩形数组 A 的第 m 行第 n 列元素	1.1
$V \leftarrow E$	把表达式 E 的值赋给变量 V	1.1
$U \leftrightarrow V$	交换变量 U 和 V 的值	1.1
$(B \Rightarrow E; E')$	条件表达式,如果 B 为真指向 E ,如果 B 为假则指向 E'	
$[B]$	条件 B 的特征函数($B \Rightarrow 1; 0$)	1.2.3
δ_{kj}	Kronecker δ : $[j = k]$	1.2.3
$[z^n]g(z)$	在幂级数 $g(z)$ 中 z^n 的系数	1.2.9
$\sum_{R(k)} f(k)$	使得变量 k 为整数且关系 $R(k)$ 为真的所有 $f(k)$ 之和	1.2.3
$\prod_{R(k)} f(k)$	使得变量 k 为整数且关系 $R(k)$ 为真的所有 $f(k)$ 之积	1.2.3
$\min_{R(k)} f(k)$	使得变量 k 为整数且关系 $R(k)$ 为真的所有 $f(k)$ 之极小值	1.2.3
$\max_{R(k)} f(k)$	使得变量 k 为整数且关系 $R(k)$ 为真的所有 $f(k)$ 之极大值	1.2.3
$\Re z$	z 的实部	1.2.2
$\Im z$	z 的虚部	1.2.2
\bar{z}	复共轭: $\Re z - i\Im z$	1.2.2
A^T	矩形数组 A 的转置: $A^T[j, k] = A[k, j]$	1.2.3
x^y	x 的 y 次方(当 x 为正时)	1.2.2
x^k	x 的 k 次方: $(k \geq 0 \Rightarrow \prod_{0 \leq j < k} x; 1/x^{-k})$	1.2.2
$x^{\bar{k}}$	x 的 k 升阶乘: $\Gamma(x+k)/\Gamma(x) = (k \geq 0 \Rightarrow \prod_{0 \leq j < k} (x+j); 1/(x+y)^{-k})$	1.2.5

(续)

形式符号	意 义	定义的位置
$x^{\underline{k}}$	x 的 k 降阶乘: $x^{\underline{k}}/(x-k)! = (k \geq 0 \Rightarrow \prod_{0 \leq j < k} (x-j); 1/(x-k)^{\underline{-k}})$	1.2.5
$n!$	n 的阶乘: $\Gamma(n+1) = n!$	1.2.5
$f'(x)$	f 在 x 处的导数	1.2.9
$f''(x)$	f 在 x 处的二阶导数	1.2.10
$f^{(n)}(x)$	n 阶导数: $(n=0 \Rightarrow f(x); g'(x))$, 其中 $g(x) = f^{(n-1)}(x)$	1.2.11.2
$f^{[n]}(x)$	第 n 次迭代: $(n=0 \Rightarrow x; f(f^{[n-1]}(x)))$	4.7
$f^{[n]}(x)$	第 n 次归纳函数: $f^{[n]}(x) = f(x f^{[n-1]}(x))$	4.7
$H_n^{(r)}$	阶为 r 的调和数: $\sum_{1 \leq k \leq n} 1/k^r$	1.2.7
H_n	调和数: $H_n^{(1)}$	1.2.7
F_n	斐波那契数: $(n \leq 1 \Rightarrow n; F_{n-1} + F_{n-2})$	1.2.8
B_n	伯努利数: $n![z^n]z/(e^z - 1)$	1.2.11.2
$X \cdot Y$	向量 $X = (x_1, \dots, x_n)$ 和 $Y = (y_1, \dots, y_n)$ 的点乘: $x_1 y_1 + \dots + x_n y_n$	3.3.4
$j \mid k$	j 整除 k : $k \bmod j = 0$ 且 $j > 0$	1.2.4
$S \setminus T$	集合差: $\{a \mid a \text{ 在 } S \text{ 中且 } a \text{ 不在 } T \text{ 中}\}$	
$\oplus \ominus \otimes \oslash$	舍入或特殊运算	4.2.1
$(\dots a_1 a_0, a_{-1} \dots)_b$	b 进制的按位表示: $\sum_k a^k b^k$	4.1
$// x_1, x_2, \dots, x_n //$	连分数: $1/(x_1 + 1/(x_2 + 1/(\dots + 1/(x_n) \dots)))$	4.5.3
$\binom{x}{k}$	二项式系数: $(k < 0 \Rightarrow 0; x^k/k!)$	1.2.6
$\binom{n}{n_1, n_2, \dots, n_m}$	多项式系数 (仅当 $n = n_1 + n_2 + \dots + n_m$ 时才有定义)	1.2.6
$\left[\begin{matrix} n \\ m \end{matrix} \right]$	第一类斯特林数: $\sum_{0 < k_1 < k_2 < \dots < k_{n-m} < n} k_1 k_2 \dots k_{n-m}$	1.2.6
$\left\{ \begin{matrix} n \\ m \end{matrix} \right\}$	第二类斯特林数: $\sum_{1 \leq k_1 \leq k_2 \leq \dots \leq k_{n-m} \leq n} k_1 k_2 \dots k_{n-m}$	1.2.6
$\{a \mid R(a)\}$	使得关系 $R(a)$ 为真的所有 a 的集合	
$\{a_1, \dots, a_n\}$	集合或多重集合 $\{a_k \mid 1 \leq k \leq n\}$	
$ x $	分数部分 (用于意味着是一个实数值而不是一个集合的上下文中): $x - \lfloor x \rfloor$	1.2.11.2
$[a, b]$	闭区间 $\{x \mid a \leq x \leq b\}$	1.2.2
(a, b)	开区间 $\{x \mid a < x < b\}$	1.2.2
$[a, b)$	半开区间: $\{x \mid a \leq x < b\}$	1.2.2
$(a, b]$	半开区间: $\{x \mid a < x \leq b\}$	1.2.2
$ S $	基数: 在集合 S 中元素的个数	

(续)

形式符号	意 义	定义的位置
$ x $	x 的绝对值: $(x \geq 0 \Rightarrow x; -x)$	
$ z $	z 的绝对值: $\sqrt{z\bar{z}}$	1.2.2
$\lfloor x \rfloor$	x 的底限函数, 即最大整数函数: $\max_{k \leq x} k$	1.2.4
$\lceil x \rceil$	顶限函数, 即最小整数函数: $\min_{k \geq x} k$	1.2.4
$((x))$	锯齿函数	3.3.3
$\langle X_n \rangle$	无穷序列 X_0, X_1, X_2, \dots (这里字母 n 是符号的一部分)	1.2.9
γ	欧拉常数: $\lim_{n \rightarrow \infty} (H_n - \ln n)$	1.2.7
$\gamma(x, y)$	不完全 Γ 函数: $\int_0^y e^{-t} t^{x-1} dt$	1.2.11.3
$\Gamma(x)$	Γ 函数: $(x-1)! = \Gamma(x, \infty)$	1.2.5
$\delta(x)$	整数的特征函数	3.3.3
e	自然对数的底: $\sum_{n \geq 0} 1/n!$	1.2.2
$\zeta(x)$	ζ 函数: $\lim_{s \rightarrow \infty} H_n^{(x)}$ (当 $x > 1$ 时)	1.2.7
$K_n(x_1, \dots, x_n)$	连续多项式	4.5.3
$\ell(u)$	多项式 u 的前导系数	4.6
$l(u)$	对于 n 的最短加法链的长度	4.6.3
$\Lambda(n)$	Von Mangoldt 函数	4.5.3
$\mu(n)$	Möbius 函数	4.5.2
$\nu(n)$	边路和	4.6.3
$O(f(n))$	当 $n \rightarrow \infty$ 时, $f(n)$ 的大 O	1.2.11.1
$O(f(z))$	当 $z \rightarrow 0$ 时, $f(z)$ 的大 O	1.2.11.1
$\Omega(f(n))$	当 $n \rightarrow \infty$ 时, $f(n)$ 的大 Ω	1.2.11.1
$\Theta(f(n))$	当 $n \rightarrow \infty$ 时, $f(n)$ 的大 Θ	1.2.11.1
$\pi(x)$	素数计数 $\sum_{n \leq x} [n \text{ 是素数}]$	4.5.4
π	圆周率: $4 \sum_{n \geq 0} (-1)^n / (2n+1)$	4.3.1
ϕ	黄金比: $\frac{1}{2}(1+\sqrt{5})$	1.2.8
\emptyset	空集: $\{x 0=1\}$	
$\varphi(n)$	欧拉 φ 函数: $\sum_{0 \leq k < n} [k \perp n]$	1.2.4
∞	无穷大: 比任何数都大	4.2.2
$\det(A)$	矩形数组 A 的行列式	1.2.3
$\text{sign}(x)$	x 的符号: $(x=0 \Rightarrow 0; x/ x)$	
$\deg(u)$	多项式 u 的次数	4.6

(续)

形式符号	意 义	定义的位置
$\text{cont}(u)$	多项式 u 的容度	4.6.1
$\text{pp}(u(x))$	多项式 u 的本原部分	4.6.1
$\log_b x$	(当 $x > 0, b > 0$ 和 $b \neq 1$ 时) x 的以 b 为底的对数,即使得 $x = b^y$ 的 y 值	1.2.2
$\ln x$	自然对数: $\log_e x$	1.2.2
$\lg x$	二进制对数: $\log_2 x$	1.2.2
$\exp x$	x 的指数: e^x	1.2.9
$j \perp k$	j 与 k 互素: $\text{gcd}(j, k) = 1$	1.2.4
$\text{gcd}(j, k)$	j 和 k 的最大公因子: $(j = k = 0 \Rightarrow 0; \max_{d \mid j, d \mid k} d)$	4.5.2
$\text{lcm}(j, k)$	j 和 k 的最小公倍数: $(jk = 0 \Rightarrow 0; \min_{d > 0, j \mid d, k \mid d} d)$	4.5.2
$x \bmod y$	mod 函数: $(y = 0 \Rightarrow x; x - y \lfloor x/y \rfloor)$	1.2.4
$u(r) \bmod v(r)$	多项式 u 除以多项式 v 之后的余式	4.6.1
$x \equiv x' \pmod{y}$	同余关系: $x \bmod y = x' \bmod y$	1.2.4
$x \approx y$	x 近似地等于 y	3.5, 4.2.2
$\text{Pr}(S(n))$	对于随机正整数 n , 命题 $S(n)$ 为真的概率	3.5
$\text{Pr}(S(X))$	对于 X 的随机值, 命题 $S(X)$ 为真的概率	1.2.10
$E X$	X 的期望值: $\sum_r r \text{Pr}(X = r)$	1.2.10
$\text{mean}(g)$	由生成函数 g 表示的概率分布的均值: $g'(1)$	1.2.10
$\text{var}(g)$	由生成函数 g 表示的概率分布的方差: $g''(1) + g'(1) - g'(1)^2$	1.2.10
$(\min x_1, \text{ave } x_2, \max x_3, \text{dev } x_4)$	有极小值 x_1 , 均(期望)值 x_2 , 极大值 x_3 , 标准差 x_4 的一个随机变量	1.2.10
\square	一个空格	1.3.1
rA	MIX 的寄存器 A(累加器)	1.3.1
rX	MIX 的寄存器 X(扩充)	1.3.1
$r11, \dots, r16$	MIX 的(变址)寄存器 I_1, \dots, I_6	1.3.1
rJ	MIX 的(转移)寄存器 J	1.3.1
$(L:R)$	MIX 字的部分字段, $0 \leq L \leq R \leq 5$	1.3.1
OP ADDRESS, I(F)	MIX 指令符号	1.3.1, 1.3.2
μ	MIX 中的时间单位	1.3.1
$*$	MIXAL 中的“本身”	1.3.2
OF, 1F, 2F, \dots , 9F	MIXAL 的“向前”局部符号	1.3.2
OB, 1B, 2B, \dots , 9B	MIXAL 中的“向后”局部符号	1.3.2
OH, 1H, 2H, \dots , 9H	MIXAL 中的“这里”局部符号	1.3.2

索引与词汇表

Seek and ye shall find.

寻找吧你会找到的。

—Matthew 7:7

当一条索引所指页码包含相关习题时,也请参看对该习题的答案,以获得更进一步信息。习题答案未参与索引,除非其中有习题中未包含的话题。

- 0-origin indexing 以 0 开始的下标
[0,1) sequence [0,1) 序列
2-adic numbers 2-adic 数
10-adic numbers 10-adic 数
 ∞ , representation of 无穷的表示
 ∞ -distributed sequence 无穷分布序列
 γ (Euler's constant) 欧拉常数
 π (circle ratio) 圆周率
 $\pi(x)$ (prime count) 素数计数
 ϕ (golden ratio) 黄金比
 logarithm of $\ln \phi$
 number system ϕ 数系
 $\varphi(n)$ (totient function) 欧拉 φ 函数
 χ^2 见 Chi-square
A priori tests 先验检验
Abacus 算盘
 binary 二进制算盘
Abel, Niels Henrik, binomial theorem 阿贝尔·尼尔斯·亨里克二项式理论
Abramowitz, Milton 阿布拉莫维茨, 米尔顿
Absolute error 绝对误差
Absorption laws 吸收率
Abuse of probability 概率滥用
Abuse of theory 理论滥用
ACC; Floating point accumulator 浮点累加器
Acceptance-rejection method 接受—拒绝方法
Accuracy of floating point arithmetic 浮点算术精度
Accuracy of random number generation 随机数生成精度
Adaptation of coefficients 系数修正
Add-with-carry sequence 带进位加序列
Addition 加法
 complex 复数加法
 continued fractions 连分数加法
 double-precision 双精度加法
 floating point 浮点加法
 fractions 分数加法
 left to right 从左到右加法
 mixed-radix 混合进制加法
 mod m 模 m 加法
 modular 模加法
 multiprecision 多精度加法
 polynomial 多项式加法
 power series 幂级数加法
 sideways 侧向加法
Addition chains 加法链
 ascending 递增加法链
 dual 加法链的对偶
 l^0 l^0 加法链
 star 星加法链
Addition-subtraction chains 加减法链
Additive random number generation 加法随机数生成
Adleman, Leonard Max 艾德勒曼, 伦纳德·马克斯
Admissible numbers 允许的数
Ahrens, Joachim Heinrich Ludecke 阿伦斯, 乔基姆·海因里希·卢德克
Ahrens, Wilhelm Ernst Martin Georg 阿伦斯, 威廉·厄恩斯特·马丁·乔治
Akushsky, Izrail Yakovlevich (Акушский, Израиль Яковлевич) 阿库斯基, 伊兹赖尔·雅科夫列维

- 奇
- al-Biruni, Abū al-Rayhān Muḥammad ibn Ahmad 阿尔-毕鲁尼, 阿布·阿尔-拉伊汗·穆罕默德·伊本·阿哈麦德
- al-Kāshī, Jamshīd ibn Mas'ūd 阿尔·卡西, 杰姆希德·伊本·梅苏德
- al-Khwarizmi, Abū 'Abd Allāh Muḥammad ibn Musa 阿尔·科瓦利茨米, 阿布阿布达·阿拉·穆罕默德·伊本·穆萨
- al-Samaw' al(= as-Samaw' al), ibn Yahyā ibn Yahūdā al-Maghribī 阿尔·萨马瓦尔, 伊本·亚希亚·伊本·亚护达·阿尔-马尼比
- al-Uqlidisi, Abū al-Ḥasan Ahmad ibn Ibrāhīm 阿尔-乌克里迪西, 阿布·阿尔-哈山·阿哈麦德·伊本·伊布拉希姆
- Ala-Nissilä, Tapio 阿拉-尼西拉, 塔皮欧
- Aanen, Jack David 阿兰宁, 杰克·戴维
- Aldous, David John 奥尔德斯, 戴维·约翰
- Alekseev, Boris Vasilievich (Алексеев, Борис Васильевич) 阿历克谢耶夫, 波里斯·瓦西里耶维奇
- Alekseyev, Valery Borisovich (Алексеев, Валерий Борисович) 阿历克谢耶夫, 瓦莱里·波里茨维奇
- Alexi, Werner 亚里克西, 沃纳
- Alford, William Robert 奥尔福德, 威廉·罗伯特
- Algebra, free associative 自由结合的代数
- Algebraic dependence 代数相关性
- Algebraic functions 代数函数
- Algebraic integers 代数整数
- Algebraic number fields 代数数域
- Algebraic system 代数系统; 元素及其上定义的运算的集合. 见 Field, Ring, Unique factorization domain
- ALGOL language ALGOL 语言
- Algorithms 算法; 在有限步骤中将指定输入转换成指定输出的精确规则
- analysis of 算法分析
- complexity of 算法复杂性
- discovery 算法的发现
- historical development of 算法的历史发展
- proof of 算法证明
- Alias method 别名方法
- Allouche, Jean-Paul 亚鲁奇, 琼·保罗
- ALPAK system ALPAK 系统
- Alt, Helmut 阿尔特, 赫尔穆特
- American National Standards Institute 美国国家标准协会
- AMM: American Mathematical Monthly 《美国数学期刊》: 美国数学会机关刊物, 1894 年创刊
- Amplification of guesses 猜测的扩大
- Analysis of algorithms 算法分析
- history 算法分析的历史
- Analytical Engine 分析机
- Ananthanarayanan, Kasi 阿南塔纳拉雅尼, 卡西
- AND (bitwise and) 逻辑乘, 位“与”
- Anderson, Stanley F 安德森, 斯坦利·F
- ANSI 见 The American National Standards Institute
- Antanairesis 安塔奈里锡斯
- Apollonius of Perga 佩加的阿波罗尼尤斯
- Apparently random numbers 明显随机数
- Apparition, rank of 出现的秩
- Approximate associative law 近似结合律
- Approximate equality 近似等式
- Approximately linear density 近似线性密度
- Approximation, by rational functions 有理函数逼近
- by rational numbers 有理数逼近
- Arabic mathematics 阿拉伯数学
- Arazi, Benjamin 阿雷兹, 本杰明
- Arbitrary precision 任意精度, 也见 Multiple-precision
- Arbogast, Louis Francois Antoine 阿博加斯特, 路易斯·弗兰科伊斯·安托万
- Archibald, Raymond Clare 阿奇博尔德, 雷蒙德·克莱尔
- Arctangent 反正切
- Aristotle of Stagira, son of Nicomachus 斯塔吉拉的亚理士多德·尼可曼彻斯之子
- Arithmetic 算术, 见 Addition Comparison, Division, Doubling, Exponentiation, Greatest common divisor, Halving, Multiplication, Reciprocal, Square root, Subtraction
- complex 复数算术
- floating point 浮点算术
- fractions 分数算术
- fundamental theorem of 算术基本定理
- mod m 模 m 算术
- modular 同余算术
- multiprecision 多精度算术
- polynomial 多项式算术
- power series 幂级数算术
- rational 有理算术
- Arithmetic chains 算术链, 见 Quolynomial chains

- Armengaud, Joel 阿门戈德, 乔尔
 Arney, James W 阿尼, 詹姆斯·W
 Arrival time 到达时间
 Arwin, Axel 阿尔文, 阿克塞尔
 Āryabhata 阿里亚哈塔
 ASCII: The American Standard Code for Information Interchange 美国信息交换标准代码
 Ashenhurst, Robert Lovett 阿申赫斯特, 罗伯特·洛维特
 Associative law 结合律
 approximate 近似结合律
 Asymptotic values 渐近值; 表达数值量趋向的极限特性的函数
 Atanasoff, John Vincent 阿塔纳索夫, 约翰·文森特
 Atkin, Arthur Oliver Lonsdale 阿特金, 阿瑟·奥利弗·朗斯代尔
 Atrubin, Alan Joseph 艾特鲁宾, 艾伦·约瑟夫
 Automata (plural of Automaton) 自动机 (automation 的复数形式)
 Automorphic numbers 自守数
 Avogadro di Quaregna e Cerreto, Lorenzo Romano Amedeo Carlo, number 阿伏加德罗·迪·夸里格纳·厄·塞拉托, 洛伦佐·罗马诺·阿米蒂奥·卡洛, 阿伏加德罗常数
 Axioms for floating point arithmetic 浮点算术的公理
 b -ary number b 进数
 b -ary sequence b 进序列
 Babbage, Charles 巴贝奇, 查理斯
 Babenko, Konstantin Ivanovich (Бабенко, Константин Иванович) 巴宾戈, 康斯坦丁·伊凡诺维奇
 Babington-Smith, Bernard 巴宾顿·史密斯, 伯纳德
 Babylonian mathematics 巴比伦数学
 Bach, Carl Eric 巴赫, 卡尔·埃里克
 Bachet, Claude Gaspard, sieur de Méziriac 贝彻特, 克劳德·加斯帕德, 西尤尔·德·梅兹里亚克
 Bag 袋, 包
 Bailey, David Harold 贝利, 戴维·哈罗德
 Baker, Kirby Alan 贝克, 柯尔比·艾伦
 Balanced binary number system 平衡二进(制)数系
 Balanced decimal number system 平衡十进(制)数系
 Balanced mixed-radix number system 平衡混合进制数系
 Balanced ternary number system 平衡三进(制)数系
 Ballantine, John Perry 巴兰坦, 约翰·佩里
 Bareiss, Erwin Hans 巴莱士, 欧文·汉斯
 Barlow, Jesse Louis 巴洛, 杰西·路易斯
 Barnard, Robert 巴纳德, 罗伯特
 Barnsley, Michael Fielding 巴恩斯利, 迈克尔·菲尔丁
 Barton, David Elliott 巴顿, 戴维·埃利奥特
 Barycentric coordinates 巴里森特里克坐标
 Base of representation 表示的进制
 floating point 浮点表示的进制
 Baseball 棒球
 Bauer, Friedrich Ludwig 鲍尔, 弗里德里克·路德维格
 Baum, Ulrich 鲍姆, 乌尔里克
 Baur, Walter 鲍尔, 沃尔特
 Bays, John Carter 贝斯, 约翰·卡特
 Beauzamy, Bernard 博扎米, 伯纳德
 Beckenbach, Edwin Ford 贝肯巴赫, 埃德温·福特
 Becker, Oskar Joachim 贝克尔, 奥斯卡·乔基姆
 Béjani, Robert 贝简, 罗伯特
 Belaga, Edward Grigorievich (Белага, Эдуард Григорьевич) 贝拉加, 爱德华·格里戈耶维奇
 Bell Telephone Laboratories Model V 贝尔电话实验室模型 V
 Bellman, Richard Ernest 贝尔曼, 理查德·欧内斯特
 Ben-Or, Michael 本·欧, 迈克尔
 Bender, Edward Anton 本德, 爱德华·安东
 Benford, Frank 本福德, 弗兰克
 Bentley, Jor. Lous 本特利, 乔恩·路易斯
 Bergman, George Mark 伯格曼, 乔治·马克
 Berkowitz, Stuart J. 贝克威茨, 斯图尔特·J
 Berlekamp, Elwyn Ralph 伯勒坎普, 埃尔温·拉尔夫
 algorithm 伯勒坎普算法
 Bernoulli, Jacques (= Jakob = James) 伯努利, 雅克 (= 雅各布 = 詹姆斯)
 numbers B_n 伯努利数
 numbers, table 伯努利数表
 sequences 伯努利序列
 Bernstein, Daniel Julius 伯恩斯坦, 丹尼尔·朱利叶斯
 Besicovitch, Abram Samoilovitch (Безикович, Абрам Самойлович) 贝西科维茨, 艾布拉姆·萨莫尔沃维茨
 Beta distribution β 分布
 Beyer, William Aaron 拜尔, 威廉·艾伦
 Bharati Krishna Tirthaji Maharaja, Jagadguru Swami Sri, Shankaracharya of Goverdhana Matha 哥弗达那马塔的巴拉第·克里斯纳·蒂塔基·马哈拉雅·贾加德古鲁·斯威米·斯里, 香卡拉克里亚
 Bhaskara Āchārya I 巴斯卡拉·阿喀利亚一世

- Bienaymé, Irénée Jules 比恩纳米, 艾琳尼·朱尔斯
 Bilinear forms 二次线性形式
 Billingsley, Patrick Paul 比林斯利, 帕特里克·保罗
 Bin-packing problem 装箱问题
 Binary abacus 二进算盘
 Binary basis 二进基
 Binary-coded decimal 二进编码的十进制
 Binary computer 二进计算机: 主要处理二进数的计算机
 Binary-decimal conversion 二进-十进转换
 Binary digit 二进数字
 Binary gcd algorithms 二进制最大公因子算法
 compared to Euclid's 与欧几里得算法比较
 extended 扩充二进制最大公因子算法
 Binary method for exponentiation 求指数的二进方法
 Binary number systems 二进数系
 Binary point 二进小数点
 Binary recurrences 二进递推
 Binary search 二分查找
 Binary shift 二进制移位
 Binary trees 二叉树
 BINEG BINEG 计算机
 Binet, Jacques Philippe Marie 比内, 雅克·菲利普·玛丽
 identity: $\sum_{j=1}^n a_j x_j, \sum_{k=1}^n b_k y_k =$
 $\sum_{j=1}^n a_j y_j, \sum_{k=1}^n b_k x_k +$
 $\sum_{1 \leq j < k \leq n} (a_j b_k - a_k b_j)(x_j y_k - x_k y_j)$ 比
 利特恒等式
 Bini, Dario Andrea 比尼, 达里欧·安德烈亚
 Binomial coefficients 二项式系数
 Binomial distribution 二项式分布
 tail of 二项式分布的尾部
 Binomial number system 二项式数系统, 见 Combinatorial number system
 Binomial theorem 二项式定理
 Birnbaum, Zygmunt Wilhelm 伯恩鲍姆, 齐格蒙特·威廉
 Birthday spacings 生日间隔
 BIT; Nordisk Tidskrift for Informations-Behandling, an international journal published in Scandinavia since 1961 哥本哈根出版的一种信息处理杂志, 创刊于 1961 年
 Bit: "Binary digit" 位(比特): "二进制数字", 或 0 或 1
 random 随机位
 Bitwise operations 按位运算, 见 Boolean operations
 Björk, Johan Harry 比约克, 约翰·哈里
 Blachman, Nelson Merle 布莱克曼, 纳尔逊·默尔
 Black box 黑盒
 Bleichenbacher, Daniel 布莱肯巴切, 丹尼尔
 Blinn, James Frederick 布林, 詹姆斯·弗雷德里克
 Blöte, Hendrik Willem Jan 布洛特, 亨德里克·威廉·简
 Blouin, François Joseph Raymond Marcel 布劳因, 弗朗科伊斯·约瑟夫·雷蒙德·马塞尔
 Bluestein, Leo Isaac 布卢斯坦, 利奥·艾萨克
 Blum, Bruce Ivan 布卢姆, 布鲁斯·伊凡
 Blum, Fred 布卢姆, 弗雷德
 Blum, Lenore Carol 布卢姆, 勒诺尔·卡罗尔
 Blum, Manuel 布卢姆, 曼纽尔
 integer 布卢姆整数
 Bofinger, Eve 博芬格, 伊夫
 Bofinger, Victor John 博芬格, 维克多·约翰
 Bohlender, Gerd 波伦德尔, 格尔德
 Bojańczyk, Adam Wojciech 博杰恩兹克, 亚当·沃杰
 Bombieri, Enrico 邦比耶里, 恩里科
 norm 邦比耶里范式
 Boolean operations 布尔运算
 and 与
 exclusive or 异或
 or 或
 shifts 移位
 Border rank 边界秩
 Borel, Émile Félix Édouard Justin 博雷尔, 埃米尔·费利克斯·埃杜瓦德·贾斯廷
 Borodin, Allan Bertram 博罗登, 阿伦·伯特伦
 Borosh, Itzhak 博罗斯, 伊特扎克
 Borrow 借位; 负进位
 Borwein, Peter Benjamin 博尔文, 彼得·本杰明
 Bosma, Wiebren 博斯马, 威伯伦
 Bouyer, Martine 布伊尔, 马蒂尼
 Bowden, Joseph 鲍登, 约瑟夫
 Box, George Edward Pelham 博克斯, 乔治·爱德华·佩勒姆
 Boyar, Joan 博亚尔, 乔
 Boyd, David William 博伊德, 戴维·威廉
 Bradley, Gordon Hoover 布雷德利, 戈登·胡佛
 Brakke, Kenneth Allen 布雷克, 肯尼思·艾伦
 Bramhall, Janet Natalie 布拉姆霍尔, 珍妮特·纳塔利
 Brauer, Alfred Theodor 布劳尔, 阿尔弗雷德·西奥多
 Bray, Thomas Arthur 布雷, 托马斯·阿瑟
 Brent, Richard Peirce 布伦特, 理查德·皮尔斯

- Brezinski, Claude 布里津斯基, 克劳德
 Brillhart, John David 布里尔哈特, 约翰·戴维
 Brockett, Roger Ware 布罗克特, 罗杰·韦尔
 Brocot, Achille 布罗科特, 艾基尔
 Brontë, Emily Jane 布朗蒂, 艾米利·简
 Brooks, Frederick Phillips, Jr 小布鲁克斯, 弗雷德里克·菲利普斯
 Brouwer, Luitzen Egbertus Jan 布劳威尔, 卢易特·埃格伯特·简
 Brown, David 布朗, 戴维, 见 Spencer Brown
 Brown, George William 布朗, 乔治·威廉
 Brown, Mark Robbin 布朗, 马克·罗宾
 Brown, Robert 布朗, 罗伯特, 见 Brownian motion
 Brown, William Stanley 布朗, 威廉·斯坦利
 Brownian motion 布朗运动
 Bruijn, Nicolaas Govert de 布鲁因, 尼古拉斯·戈维特·德
 cycle 布鲁因循环
 Brute force 暴力, 指硬算, 蛮干
 Bshouty, Nader Hanna 毕肖蒂, 纳德·汉纳
 Buchholz, Werner 巴克霍尔兹, 沃纳
 Bunch, James Raymond 本奇, 詹姆斯·雷蒙德
 Buneman, Oscar 邦尼曼, 奥斯卡
 Bürgisser, Peter 伯吉塞, 彼得
 Burks, Arthur Walter 伯克斯, 阿瑟·沃尔特
 Burrus, Charles Sidney 巴勒斯, 查尔斯·西德尼
 Butler, James Preston 巴特勒, 詹姆斯·普雷斯顿
 Butler, Michael Charles Richard 巴特勒·迈克尔·查尔斯·理查德
 C language C 语言
 CACM: *Communications of the ACM*, a publication of the Association for Computing Machinery since 1958 《ACM 通信》, 美国计算机协会出版物, 创刊于 1958 年
 Cahen, Eugène 卡亨, 尤金
 Calculating prodigies 计算天才
 Camion, Paul Frédéric Roger 卡米昂, 保罗·弗雷德里克·罗杰
 Campbell, Edward Fay, Jr 小坎贝尔, 爱德华·菲
 Campbell, Sullivan Graham 坎贝尔, 沙利文·格雷厄姆
 Cancellation error 消去误差
 avoiding 避免消去误差
 Cantor, David Geoffrey 坎托, 戴维·杰弗里
 Cantor, Georg Ferdinand Ludwig Philip 坎托, 乔治·费迪南德·路德维格·菲利普
 Cantor, Moritz Benedikt 坎托, 莫里茨·本尼迪克特
 Capovani, Milvie 卡波瓦尼, 米尔韦欧
 Caramuel Lobkowitz, Juan de 卡拉米尔·洛布科维茨, 胡安·德
 Cards, playing 扑克牌
 Carissan, Eugène Olivier 卡里桑, 尤金·奥利弗
 Carling, Robert Laurence 卡林, 罗伯特·劳伦斯
 Carltz, Leonard 卡利茨, 伦纳德
 Carmichael, Robert Daniel, numbers 卡迈克尔, 罗伯特·丹尼尔数
 Carr, John Weber 卡尔, 约翰·韦伯
 Carroll, Lewis (= Dodgson, Charles Lutwidge) 卡罗尔, 刘易斯 (= 道奇森, 查尔斯·路特威奇)
 Carry 进位: 从较低位传递到当前位的量
 Cassels, John William Scott 卡塞尔斯, 约翰·威廉·斯科特
 Casting out nines 舍九法
 Castle, Clive Michael Anthony 卡斯尔, 克利夫·迈克尔·安东尼
 Catalan, Eugène Charles, numbers 卡特兰, 尤金·查尔斯数
 Cauchy, Augustin Louis 柯西, 奥古斯丁·路易斯
 inequality 柯西不等式
 matrices 柯西矩阵
 CCITT: The International Telegraph and Telephone Consultative Committee of the ITU (International Telecommunication Union) 国际电报电话咨询委员会: 国际电信联盟的下属机构
 CDC 1604 computer CDC 1604 计算机
 CDC 7600 computer CDC 7600 计算机
 CDROM: Compact disk read-only memory 只读光盘存储器
 Ceiling function $\lceil x \rceil$ 顶限函数
 Cerlienco, Luigi 塞利恩科, 卢齐
 Certificate of irreducibility 不可约性鉴定
 Certificate of primality 素性鉴定
 Cesàro, Ernesto 塞萨罗, 欧内斯托
 Ceulen, Ludolph van 塞乌伦, 鲁道夫·范
 Chace, Arnold Buffum 蔡斯, 阿诺德·巴法姆
 Chain multiplications 链乘法
 Chain steps 链步
 Chains of primes 素数链
 Chaitin, Gregory John 蔡廷, 格雷戈里·约翰
 Chan, Tony Fan-Cheong 陈繁昌
 Chapple, M. A. 查普尔, M. A.
 CHAR (convert to characters) 转换成字符指令

- Characteristic 特征, 见 Exponent part
- Characteristic polynomial 特征多项式
- Charles XII of Sweden 瑞典的查尔斯十二世
- Chartres, Bruce Aylwin 查特里斯·布鲁斯·艾尔文
- Chebotarev, Nikolai Grigorievich (Чеботарев, Николай Григорьевич) 契波塔列夫, 尼古拉伊·格里戈列维奇
- Chebyshev, Pafuntii Lvovich (Чебышев, Пафнутий Львович), inequality 切比雪夫, 帕夫努季·利沃维奇不等式
- Cheng, Russell Ch'uan Hsun 郑川训
- Chesterton, Gilbert Keith 切斯特顿, 吉尔伯特·凯恩
- Chi-square distribution χ^2 分布
table χ^2 分布表
- Chi-square test χ^2 检验
- Ch'in Chiu-Shao 秦九韶
- Chinese mathematics 中国数学
- Chinese remainder algorithm 中国剩余算法
- Chinese remainder theorem 中国剩余定理
for polynomials 多项式中国剩余定理
generalized 推广的中国剩余定理
- Chirp transform 齐尔普变换
- Chiu Chang Suan Shu 《九章算术》
- Choice, random 随机选择
- Chor, Ben-Zion 科尔, 本·蔡恩
- Christiansen, Hanne Delgas 克里斯琴森, 汉尼·德尔加斯
- Christie Mallowan, Agatha Mary Clarissa Miller 克里斯蒂·马洛温, 阿加萨·玛丽·克拉里沙·米勒
- Chudnovsky, David Volfovich (Чудновский, Давид Вольфович) 丘德诺夫斯基, 戴维·沃尔夫科维奇
- Chudnovsky, Gregory Volfovich (Чудновский, Григорий Вольфович) 丘德诺夫斯基, 格里戈里·沃尔夫科维奇
- Church, Alonzo 丘奇, 阿伦佐
- Cipolla, Michele 西波拉, 米歇尔
- Classical algorithms 经典算法
- Clausen, Michael Hermann 克劳森, 迈克尔·赫尔曼
- Clinger, William Douglas 克林格, 威廉·道格拉斯
- CMath: Concrete Mathematics, a book by R. L. Graham, D. E. Knuth, and O. Patashnik 《具体的数学》
- Cochran, William Gemmell 科克伦, 威廉·格默尔
- Cocke, John 科克, 约翰
- Codes, linear 线性代码
- Codes for difficulty of exercises 习题难度代码
- Cody, William James, Jr. 小科迪, 威廉·詹姆斯
- Coefficients of a polynomial 多项式系数
adaptation of 多项式系数的修改
leading 前导多项式系数
size of 多项式系数的大小
- Cohen, Daniel Isaac Aryeh 科恩, 丹尼尔·伊萨克·阿里耶
- Cohen, Henri José 科恩, 亨利·乔斯
- Cohn, Paul Moritz 科恩, 保罗·莫里茨
- Coincidence 巧合
- Colenne, Joseph Désiré 科伦, 约瑟夫·德席尔
- Collins, George Edwin 科林斯, 乔治·埃德温
- Collision test 冲突检验
- Color values 颜色值
- Colson, John 科尔森, 约翰
- Colton, Charles Caleb 科尔顿, 查尔斯·凯莱布
- Column addition 列加法
- Combination, random 随机组合
- Combination of random number generators 随机数生成程序组合
- Combinations with repetitions 带重复的组合
- Combinatorial matrices 组合矩阵
- Combinatorial number system 组合数系统
- Commutative law 交换律
- Commutative ring with identity 有么元的交换环
- Comp. J.: The Computer Journal, a publication of the British Computer Society since 1958 《计算机杂志》: 英国计算机学会出版的杂志, 创刊于 1958 年
- Compagner, Aaldert 康培纳, 阿尔德特
- Companion matrix 伴随矩阵
- Comparison 比较: 测试 $<$, $=$ 或 $>$
continued fractions 连分数比较
floating point numbers 浮点数比较
fractions 小数比较
mixed-radix 混合进制比较
modular 模数比较
multiprecision 多精度比较
- Complement notations for numbers 数的补码记法
- Complete binary tree 完备二叉树
- Completely equidistributed sequence 完备等分布序列
- Complex arithmetic 复数算术
- Complex numbers 复数
representation of 复数的表示
- Complex radices 复数进制

- Complexity of calculation 计算复杂性
Composition of power series 幂级数的组成
Computability 可计算性
Concave function 凹函数
Conditional expression 条件表达式
Congruential sequence, inversive 逆同余序列
Congruential sequence, linear 线性同余序列
 choice of increment 线性同余序列增量的选择
 choice of modulus 线性同余序列模数的选择
 choice of multiplier 线性同余序列乘数的选择
 choice of seed 线性同余序列种子的选择
 period length 线性同余序列的周期长度
 subsequence of 线性同余序列的子序列
Congruential sequence, quadratic 二次同余序列
Conjugate of a complex number 复数的共轭
Connection Machine 连接机
Content of a polynomial 多项式容度
Context-free grammar 上下文无关文法
Continuant polynomials 连续多项式
Continued fractions 连分数
 infinite 无穷连分数
 quadratic irrationalities 二次无理连分数
 regular 正则连分数
 with polynomial quotients 带多项式商的连分数
Continuous binomial distribution 连续二项分布
Continuous distribution functions 连续分布函数
Continuous Poisson distribution 连续泊松分布
Convergents 收敛式
Conversion of representations 表示的转换, 也见 Radix conversion
Convex function 凸函数
Convolution 卷积
 cyclic 循环卷积
 multidimensional 多维卷积
 negacyclic 反循环卷积
Conway, John Horton 康韦, 约翰·霍顿
Cook, Stephen Arthur 库克, 斯蒂芬·阿瑟
Cooley, James William 库利, 詹姆斯·威廉
Coolidge, Julian Lowell 库利奇, 朱利安·洛厄尔
Coonen, Jerome Toby 库南, 杰罗姆·托比
Copeland, Arthur Herbert 科普兰, 阿瑟·赫伯特
Coppersmith, Don 科珀史密斯, 唐
Cormack, Gordon Villy 科马克, 戈登·维利
Coroutine 共行程序
Corput, Johannes Gualtherus van der 科普特, 约翰尼斯·格尔塞勒斯·范·德
Correlation coefficient 相关系数
Cosine 余弦
Cotes, Roger 科茨, 罗杰
Couffignal, Louis 库菲格纳尔 路易斯
Counting law 计数律
Coupon collector's test 集券检验
Couture, Raymond 库切尔, 雷蒙德
Covariance 协方差
 matrix 协方差矩阵
Cover, Thomas Merrill 科弗, 托马斯·梅里尔
Coveyou, Robert Reginald 科维佑, 罗伯特·雷金纳德
Cox, Albert George 考克斯, 艾伯特·乔治
Crandall, Richard Eugene 克兰多尔, 理查德·尤金
Craps 双骰子游戏
Cray T94 computer 克雷 T94 计算机
Cray X-MP computer 克雷 X-MP 计算机
Creative writing 创造性写作
Crelle: *Journal für die reine und angewandte Mathematik*, an international journal founded by A. L. Crelle in 1826 《克雷勒》: 即《科学院统计报告》, 克雷勒于 1826 年创立的国际性杂志
Cryptanalysis 密码分析
Cube root modulo m 模 m 立方根
Cunningham, Allan Joseph Champneys 坎宁安, 阿伦·约瑟夫·钱普尼斯
Cusick, Thomas William 库席克, 托马斯·威廉
Cut-and-riffle 切牌与洗牌
Cycle in a random permutation 随机排列的周期
Cycle in a sequence 序列周期
 detection of 序列周期的发现
Cyclic convolution 循环卷积
Cyclotomic polynomials 分圆多项式
Dahl, Ole-Johan 达尔, 奥利·约翰
Daniels, Henry Ellis 丹尼尔斯, 亨利·艾利斯
Dase, Johann Martin Zacharias 达斯, 约翰·马丁·扎卡赖亚斯
Datta, Bibhutibhusan 戴塔, 比布胡底布胡森
Daudé, Hervé 多德, 赫维
Davenport, Harold 达文波特, 哈罗德
David, Florence Nightingale 戴维, 弗洛伦斯·南丁格尔
Davis, Chandler 戴维斯, 钱德勒
Davis, Clive Selwyn 戴维斯, 克莱夫·塞尔温
de Bruijn, Nicolaas Govert 德布鲁因, 尼克拉斯·戈维特
 cycle 德布鲁因循环

- de Finetti, Bruno 德芬内蒂, 布鲁诺
 de Groote, Hans Friedrich 德格鲁特, 汉斯·弗里德里克
 de Jong, Lieuwe Sytse 德荣, 利欧沃·席特思
 de Jonquières, Jean Philippe Ernest de Fauque 德琼李里斯, 琼·菲利普·欧内斯特·德福基
 de La Vallee Poussin, Charles-Jean-Gustave-Nicolas 德拉·瓦利·波欣, 查尔斯·琼-古斯塔夫·尼古拉斯
 de Lagny, Thomas Fantet 德拉格尼, 托马斯·范特特
 Debugging 排错, 调试
 Decimal computer 十进计算机: 主要处理十进数的计算机
 Decimal digits 十进数字
 Decimal fractions, history 十进小数的历史
 Decimal number system 十进数系统
 Decimal point 小数点
 Decimation 十进表示
 Decision, unbiased 无偏倚决策
 DECsystem 20 computer DEC 系统 20 计算机
 Decuple-precision floating point 十倍精度浮点
 Dedekind, Julius Wilhelm Richard 戴德金, 朱利亚斯·威廉·理查德
 sums, generalized 戴德金和数, 广义戴德金和数
 Definitely greater than 确定地大于
 Definitely less than 确定地小于
 Definition of randomness 随机性定义
 Dégot, Jérôme 德戈特, 杰罗姆
 Degree of a polynomial 多项式次数
 Degrees of freedom 自由度
 Dekker, Theodorus Jozef 德克尔, 西奥多罗斯·乔泽夫
 Deléglise, Marc 德利格利斯, 马克
 Dellac, H. 德拉克, H
 DeMillo, Richard Allan 德米罗, 理查德·阿伦
 Denneau, Monty Montague 丹尼欧, 蒙蒂·蒙特古
 Denormal floating point number 非正规浮点数
 Density function 密度函数
 nearly linear 近线性密度函数
 Dependent normal deviates 相关正态偏离
 Derandomization 非随机化
 Derflinger, Gerhard 德弗林格, 格哈特
 Derivatives 可导的, 可微的
 Descartes, René 德谢卡特斯, 伦尼
 Determinants 行列式
 Deviate: A random number 随机数的偏离
 Devroye, Luc Piet-Jan Arthur 德弗罗尔, 卢克·皮特-简·阿瑟
 简·阿瑟
 Dewey, Melvil, notation for trees 杜威, 梅尔维尔的树记法
 Diaconis, Persi Warren 黛亚戈尼斯, 珀西·沃伦
 Diamond, Harold George 戴蒙德, 哈罗德·乔治
 Dice 骰子
 Dickman, Karl Daniel 迪克曼, 卡尔·丹尼尔
 Dickman-Golomb constant 迪克曼-戈龙常数
 Dickson, Leonard Eugene 迪克森, 伦纳德·尤金
 Dictionaries 字典
 Dieter, Ulrich Otto 迪特尔, 乌尔里克·奥托
 Differences 差
 Differential equations 微分方程
 Differentiation 微分法, 见 Derivatives
 Diffie, Bailey Whitfield 迪菲, 贝利·惠特菲尔德
 Digit 数字: 定位记号下所用符号: 通常指十进数字
 0, 1, ..., 9 之一
 binary 二进制数字
 decimal 十进数字
 hexadecimal 十六进数字
 octal 八进数字
 Dilcher, Karl Heinrich 迪尔切尔, 卡尔·海因里希
 Dilogarithm 二重对数
 Diophantine equations 刁藩都方程
 Diophantus of Alexandria (Διόφαντος Ἀλεξανδρ εως) 亚历山大的刁藩都, 见 Diophantine equations
 Direct product 直接积
 Direct sum 直接和
 conjecture 直接和猜想
 Directed graph 有向图
 Dirichlet, Peter Gustav Lejeune 迪里赫里, 彼得·古斯塔夫·勒琼
 series 迪里赫里级数
 Discrepancy 差异
 Discrete distribution functions 离散分布函数
 Discrete Fourier transforms 离散傅里叶变换
 Discrete logarithms 离散对数
 Discriminant of a polynomial 多项式的判别式
 Distinct-degree factorization 不同次数的因子分解
 Distribution 分布: 支配一个随机变量的值的概率的描述
 beta β (贝塔)分布
 binomial 二项分布
 chi-square χ^2 分布
 exponential 指数分布
 F · F 分布

- of floating point numbers 浮点数分布
- gamma Γ (伽玛)分布
- geometric 几何分布
- integer-valued 整值分布
- Kolmogorov-Smirnov 科尔莫戈罗夫-斯米尔诺夫分布, KS 分布
- of leading digits 前导数字的分布
- negative binomial 负的二项分布
- normal 正态分布
- partial quotients of regular continued fractions 正则连分数部分商的分布
- Poisson 泊松分布
- of prime factors 素因子分布
- of prime numbers 素数的分布
- Student's 学生分布
- t - t 分布
- tail of binomial 二项分布的尾部
- tail of normal 正态分布的尾部
- uniform 一致分布
- variance-ratio 方差比分布
- wedge-shaped 楔形分布
- Distribution functions 分布函数
 - continuous 连续分布函数
 - discrete 离散分布函数
 - empirical 经验分布函数
 - mixture of 分布函数的混合
 - polynomial 多项式分布函数
 - product of 分布函数的积
- Distributive laws 分配律
- Divide-and-correct 除并校正
- Divided differences 除后的差
- Dividend 被除数
- Division 除法
 - algebraic numbers 代数数除法
 - avoiding 除法的避免
 - balanced ternary 平衡三进制除法
 - by ten 除以 10
 - by zero 除以 0
 - complex 复数除法
 - continued fractions 连分数除法
 - double-precision 双精度除法
 - exact 精确除法
 - floating point 浮点除法
 - fractions 分数除法
 - long 长除法
 - mixed-radix 混合进制除法
 - mod m 模 m 除法
 - multiprecision 多精度除法
 - multiprecision by single-precision 通过单精度的多精度除法
 - polynomial 多项式除法
 - power series 幂级数除法
 - pseudo- 伪除法
 - quater-imaginar 虚 4 除法
 - short 短除法
 - string polynomials 串多项式除法
- Divisor 除数
- Divisor 因子; 若 $y \bmod x = 0$ 且 $x > 0$, 则 x 是 y 的因子; 若同时 $1 < x < y$, 则 x 是 y 的真因子
- polynomial 多项式因子
- Dixon, John Douglas 狄克逊, 约翰·道格拉斯
- Dixon, Wilfrid Joseph 狄克逊, 威尔弗雷德·约瑟夫
- Dobell, Alan Rodney 多贝尔, 艾伦·罗德尼
- Dobson, David Paul 多布金, 戴维·保罗
- Dodgson, Charles Lutwidge 道奇森, 查尔斯·勒特威奇
- Donsker, Monroe David 唐斯克, 门罗·戴维
- Doob, Joseph Leo 杜布, 约瑟夫·利奥
- Dorn, William Schroeder 多恩, 威廉·施罗德
- Dot product 点积
- Double-precision arithmetic 双精度算术
- Doubling 加倍
 - continued fraction 连分数加倍
- Doubling step 倍步
- Downey, Peter James 唐尼, 彼得·詹姆斯
- Dragon curve 龙曲线
- Dragon sequence 龙序列
- Dresden, Arnold 德雷斯登, 阿诺德
- Drift 漂移
- Du Shiran 杜石然
- Dual of an addition chain 一条加法链的对偶
- Duality formula 对偶公式
- Duality principle 对偶原理
- Dubner, Harvey Allen 杜布内, 哈维·艾伦
- Dumas, Philippe 杜马斯, 非利普
- Duncan, Robert Lee 邓肯, 罗伯特·李
- Duodecimal number system 十二进制数系统
- Dupré, Athanase 杜普雷, 阿塔纳西
- Durbin, James 德宾, 詹姆斯
- Durham, Stephen Daniel 德拉姆, 斯蒂芬·丹尼尔
- Durstenfeld, Richard 德斯特菲尔德, 理查德
- e 自然对数的底

- Earle, John Goodell 厄尔, 约翰·古德爾
 Eckhardt, Roger Charles 艾克哈特, 罗杰·查尔斯
 L'Ecuyer, Pierre 利凯尔, 皮埃尔
 Edelman, Alan Stuart 埃德尔曼, 艾伦·斯图尔特
 Edinburgh rainfall 爱丁堡的降雨
 EDVAC computer EDVAC 计算机
 Effective algorithms 能行算法
 Effective information 有效信息
 Egyptian mathematics 埃及数学
 Eichenauer-Herrman, Jürgen 艾克瑞尔-赫尔曼, 朱尔金
 Eisenstein, Ferdinand Gotthold Max 艾森斯坦, 费迪南德·戈特霍尔德·马克斯
 Electrológica X8 computer Electrológica X8 计算机
 Electronic mail 电子邮件(函件)
 Elementary symmetric functions 初等对称函数
 Elkies, Noam David 埃尔克斯, 诺姆·戴维
 Ellipsoid 椭球面
 random point on 椭球面上的随机点
 Elliptic curve method 椭圆曲线方法
 Empirical distribution functions 经验分布函数
 Empirical tests for randomness 随机性的经验检验
 Encoding a permutation 对排列编码
 Encoding secret messages 对秘密消息编码
 Enflo, Per 恩弗洛, 佩尔
 Engineering Research Associates 工程研究协会
 Enhancing randomness 强化随机性
 ENIAC computer ENIAC 计算机
 Entropy 熵
 Enumerating binary trees 枚举二叉树
 Enumerating prime numbers 枚举素数
 Equality, approximate 近似相等
 essential 实质上的相等
 Equidistributed sequence 等分布序列
 Equidistribution test 等分布检验
 Equitable distribution 等分布
 Equivalent addition chains 等价加法链
 Eratosthenes of Cyrene 昔兰尼的厄拉托森斯
 Erdős, Pál (= Paul) 厄尔多斯, 保罗
 ERH 见 Extended Riemann Hypothesis
 ERNIE machine 厄尼计算机
 Error, absolute 绝对误差
 Error, relative 相对误差
 Error estimation 误差估计
 Espelid, Terje Oskar 埃斯皮利德, 特吉·奥斯卡
 Essential equality 实质上的相等
 Estes, Dennis Ray 埃斯特斯, 丹尼斯·雷
 Estrin, Gerald 埃斯特林, 杰拉尔德
 Euclid (Εὐκλείδης) 欧几里得
 Euclid's algorithm 欧几里得算法
 analysis of 欧几里得算法的分析
 compared to binary algorithm 与二进算法的比较
 extended 扩充欧几里得算法
 for polynomials 多项式的欧几里得算法
 for polynomials, extended 扩充的多项式欧几里得算法
 for string polynomials 串多项式的欧几里得算法
 generalized to the hilt 推广到 hilt 的欧几里得算法
 multiprecision 多精度欧几里得算法
 original form 欧几里得算法的原始形式
 Eudoxus of Cnidus 克尼达斯的尤杜古斯
 Euler, Leonhard (Ейлеръ, Леонардъ = Эйлер, Леонард) 欧拉, 伦哈德
 constant γ 欧拉常数 γ
 theorem 欧拉定理
 totient function $\varphi(n)$ 欧拉计数函数
 Eulerian numbers 欧拉数
 Evaluation 求值
 of determinants 行列式求值
 of mean and standard deviation 均值与标准差的求值
 of monomials 单项式求值
 of polynomials 多项式求值
 of powers 求幂值
 Eve, James 厄维, 詹姆斯
 Eventually periodic sequence 最终的周期序列
 Exact division 精确除法
 Excess q exponent 加余量 q 的指数
 Exclusive or 异或
 Exercises, notes on 关于习题的说明
 Exhaustive search 穷尽查找
 Exponent overflow 指数上溢
 Exponent part of a floating point number 浮点数的指数部分
 Exponent underflow 指数下溢
 Exponential deviates, generating 生成指数偏离
 Exponential distribution 指数分布
 Exponential function 指数函数
 Exponential sums 指数和
 Exponentiation 求幂
 multiprecision 多精度求幂
 of polynomials 多项式求幂

- of power series 幂级数求幂
- Extended arithmetic 扩充算术
- Extended Euclidean algorithm 扩充欧几里得算法
for polynomials 扩充的多项式欧几里得算法
- Extended Riemann Hypothesis 扩充黎曼假设
- F-distribution F 分布
- Factor 因子
- Factor method of exponentiation 指数的因子方法
- Factorial number system 阶乘数系统
- Factorial powers 阶乘的幂
- Factorials 阶乘
- Factorization 因子分解
of integers 整数的因子分解
of polynomials 多项式的因子分解
of polynomials mod p 多项式模 p 因子分解
of polynomials over the integers 整数上的多项式因子分解
of polynomials over the rationals 有理数上的多项式因子分解
optimistic estimates of running time 因子分解运行时间的最优估计
uniqueness of 因子分解的惟一性
- FADD (floating add) 浮点加指令
- Fagin, Barry Steven 费金, 巴里·史蒂文
- Fallacious reasoning 不合理推理
- Falling powers 降幂
- Fan, Chung Teh 范崇德
- Fast Fourier transform 快速傅里叶变换
history of 快速傅里叶变换的历史
- Fateman, Richard J 费特曼, 理查德·J
- Faure, Henri 福里, 亨利
- FCMP (floating compare) 浮点比较指令
- FDIV (floating divide) 浮点除指令
- Feijen, Wilhelmus (= Wim) Hendricus Johannes 法伊詹, 威廉姆斯(= 威姆)·亨德里卡斯·约翰尼斯
- Ferguson, Donald Fraser 弗格森, 唐纳德·弗雷泽
- Fermat, Pierre de 费马, 皮埃尔·德
factorization method 费马因子分解方法
numbers 费马数
theorem 费马定理
- Ferranti Mark I computer 费兰蒂·马克 I 型计算机
- Ferrenberg, Alan Milton 费伦伯格, 艾伦·米尔顿
- FFT 见 Fast Fourier transform
- Fibonacci, Leonardo, of Pisa 皮萨的斐波那契, 伦纳德
generator 斐波那契生成
- number system 斐波拉契数系统
- numbers F_n 斐波那契数; 斐波那契序列的元素
- numbers, table of 斐波那契数表
- sequence 斐波那契序列
- sequence, lagged 延搁斐波那契序列
- Field 域: 允许进行加、减、乘、除的代数系统
finite 有限域
- Fike, Charles Theodore 菲克, 查尔斯·西奥多
- Finck, Pierre Joseph Étienne 芬克, 皮埃尔·约瑟夫·厄蒂恩
- Fine, ti, Bruno de 芬内蒂, 布鲁诺·德
- Finite fields 有限域
- Finite Fourier transform 有限傅里叶变换, 见 Discrete Fourier transform
- Finite sequences, random 有限随机序列
- Fischer, Michael John 费希尔, 迈克尔·约翰
- Fischer, Patrick Carl 费希尔, 帕特里克·卡尔
- Fischlin, Roger 费施林, 罗杰
- Fisher, Ronald Aylmer 费希尔, 罗纳尔德·埃尔默
- Fishman, George Samuel 非什曼, 乔治·塞缪尔
- FIX (convert to fixed point) 转换到定点指令
- Fix-to-float conversion 定点-浮点转换
- Fixed point arithmetic 定点算术
- Fixed slash arithmetic 开缝定点算术
- Flajolet, Philippe Patrick Michel 弗莱乔利特, 菲利普·帕特里克·迈克尔
- Flammenkamp, Achim 弗拉门坎普, 阿基姆
- Flat distribution 平坦分布, 见 Uniform distribution
- Flehinger, Betty Jeanne 弗勒欣格尔, 贝蒂·珍妮
- Float-to-fix conversion 浮点-定点转换
- Floating binary numbers 浮点二进制数
- Floating decimal numbers 浮点十进制数
- Floating hexadecimal numbers 浮点十六进制数
- Floating point arithmetic 浮点算术
accuracy of 浮点算术的精确性
addition 浮点加
addition, exact 精确浮点加
axioms 浮点算术公理
comparison 浮点比较
decuple-precision 十倍精度浮点算术
division 浮点除法
double-precision 双精度浮点算术
hardware 硬件浮点算术
intervals 区间浮点算术
mod 浮点取模
multiplication 浮点乘

- multiplication, exact 精确浮点乘
 operators of MIX MIX 的浮点操作符
 quadruple-precision 四倍精度浮点算术
 reciprocal 倒数的浮点算术
 single-precision 单精度浮点算术
 subtraction 浮点减
 summation 浮点求和
 triple-precision 三精度浮点算术
 unnormalized 非规格化浮点算术
 Floating point numbers 浮点数
 radix- b , excess- q b 进制余量 q 浮点数
 statistical distribution 浮点数统计分布
 two's complement 2 的补码浮点数
 Floating point radix conversion 浮点进制转换
 Floating point trigonometric subroutines 浮点三角子程序
 Floating slash arithmetic 开键浮点算术
 Floor function $\lfloor x \rfloor$ 底限函数
 FLOTT (convert to floating point) 转换到浮点指令
 Floyd, Robert W 弗洛伊德, 罗伯特·W
 Fluhrer, Scott 弗卢雷尔, 斯科特
 FMUL (floating multiply) 浮点乘指令
 Foata, Dominique Cyprien 福阿塔, 多米尼克·西普里恩
 FOCS: *Proceedings of the IEEE Symposia on Foundations of Computer Science* (1975—), formerly called the *Symposia on Switching Circuit Theory and Logic Design* (1960—1965), *Symposia on Switching and Automata Theory* (1966—1974)
 《IEEE 计算机科学基础研讨会讨论集》
 Forsythe, George Elmer 福赛思, 乔治·埃尔默
 FORTRAN language FORTRAN 语言
 Fourier, Jean Baptiste Joseph 傅里叶, 琼·巴普蒂斯特·约瑟夫
 division method 傅里叶除法
 series 傅里叶级数
 transform, discrete 离散傅里叶变换
 Fractals 碎片
 Fraction overflow 小数上溢
 Fraction part of a floating point number 浮点数的小数部分
 distribution of 浮点数小数部分的分布
 Fractions 小数: $[0, 1)$ 中的数
 conversion 小数的转换
 decimal, history 十进小数的历史
 exponentiation 小数的指数
 random 随机小数, 见 Uniform deviates
 terminating 有穷小数
 Fractions: Rational numbers 分数; 有理数
 Fraenkel, Aviezri S 弗伦克尔, 阿维泽里·S
 Franel, Jérôme 弗兰尼尔, 杰罗姆
 Franklin, Joel Nick 富兰克林, 乔尔·尼克
 Franta, William Ray 弗兰塔, 威廉·雷
 Fredricksen, Harold Marvin 弗雷德里克森, 哈罗德·马文
 Free associative algebra 自由结合代数
 Frequency function 频率函数, 见 Density function
 Frequency test 频率检验
 Friedland, Paul 弗里德兰, 保罗
 Frieze, Alan Michael 弗里兹, 艾伦·迈克尔
 Fritz, Kurt von 弗里茨, 库尔特·冯
 Frobenius, Ferdinand Georg 弗罗比尼尤斯, 费迪南德·乔治
 automorphism 弗罗比尼尤斯自同构
 Frye, Roger Edward 弗赖伊, 罗杰·爱德华
 FSUB (floating subtract) 浮点减指令
 Fuchs, Aimé 富克斯, 艾梅
 Fundamental theorem of arithmetic 算术基本定理
 Fuss, Paul Heinrich von (Фус, Павел Николаевич) 富斯, 保罗·海因里希
 Gage, Paul Vincent 盖奇, 保罗·文森特
 Galambos, János 加拉姆博斯, 雅诺什
 Galois, Évariste 伽罗瓦, 埃瓦里斯特
 fields, 伽罗瓦域, 见 Finite fields
 groups 伽罗瓦群
 Gambling systems 赌博系统
 Gamma distribution Γ (伽玛) 分布
 Gamma function, incomplete 不完备的 Γ 函数
 Ganz, Jürg Werner 甘兹, 朱尔格·沃纳
 Gap test 间隔检验
 Gardner, Martin 加德纳, 马丁
 Garner, Harvey Louis 加纳, 哈维·路易斯
 Gathen, Joachim Paul Rudolf von zur 盖森, 乔基姆·保罗·鲁道夫·冯·泽
 Gauß (= Gauss), Johann Friedrich Carl (= Carl Friedrich) 高斯, 约翰·弗里德里克·卡尔
 lemma about polynomials 高斯多项式引理
 Gaussian integers 高斯整数
 Gay, John 盖伊, 约翰
 gcd; Greatest common divisor 最大公因子
 Gebhardt, Friedrich 格布哈特, 弗里德里克
 Gehhardt, Karl Immanuel 格哈特, 卡尔·伊曼纽尔

- Geiger, Hans, counter 盖革, 汉斯, 盖革计数器
 Geiringer, Hilda, von Mises 盖令格尔, 希尔达, 冯·迈席斯
 Generalized Dedekind sums 广义戴德金和数
 Generalized Riemann hypothesis 广义黎曼假设
 Generating functions 生成函数
 Generation of uniform deviates 一致偏离的生成
 Genuys, François 吉纽斯, 弗朗科伊斯
 Geometric distribution 几何分布
 Geometric mean 几何平均
 Geometric series 几何级数
 Gessel, Ira Martin 格塞尔, 艾拉·马丁
 Gibb, Allan 吉布, 阿伦
 Gilbert, William John 吉尔伯特, 威廉·约翰
 Gill, Stanley 古尔, 斯坦利
 GIMPS project GIMPS 工程
 Gioia, Anthony Alfred 乔亚, 安东尼·艾尔弗雷德
 Girard, Albert 吉拉德, 艾伯特
 Givens, James Wallace Jr. 小吉文斯, 詹姆斯·华莱士
 Glaser, Anton 格拉泽, 安东
 Globally nonrandom behavior 全局非随机特性
 Goertzel, Gerald 戈策尔, 杰拉尔德
 Goffinet, Daniel 戈芬尼特, 丹尼尔
 Goldbach, Christian 哥德巴赫, 克里斯琴
 Goldberg, David Marc 戈德堡, 戴维·马克
 Golden ratio 黄金比
 Goldreich, Oded 戈德里奇, 奥迪德
 Goldschmidt, Robert Elliott 戈德施米特, 罗伯特·埃利奥特
 Goldstine, Herman Heine 戈德斯坦, 赫尔曼·海因
 Goldwasser, Shafira 戈德瓦泽, 沙夫里拉
 Golomb, Solomon Wolf 哥伦布, 所罗门·沃尔夫
 Golub, Gene Howard 戈卢布, 吉恩·霍华德
 Gonzalez, Teofilo 冈萨雷斯, 特奥菲罗
 Good, Irving John 古德, 欧文·约翰
 Goodman, Allan Sheldon 古德曼, 阿伦·谢尔登
 Gosper, Ralph William, Jr. 小戈斯珀, 拉尔夫·威廉
 Goulard, A. 古拉德, A
 Gould, Henry Wadsworth 古尔德, 亨利·沃兹沃思
 Gourdon, Xavier Richard 古尔登, 泽维尔·理查德
 Goyal, Girish Kumar 高耶尔, 吉里什·库默
 Gradual underflow 逐渐下溢
 Gräffe, Carl Heinrich 格拉夫, 卡尔·海因里希
 Graham, Ronald Lewis 格雷厄姆, 罗纳德·刘易斯
 Gram, Jørgen Pedersen 格拉姆, 乔根·佩德森
 Gram-Schmidt orthogonalization process 格拉姆·施密特正交化方法
 Granville, Andrew James 格兰维尔, 安德鲁·詹姆斯
 Graph 图
 Graphics 图解
 Gray, Frank, code 格雷, 弗兰克码, 格雷码
 Gray, Herbert L. 格雷, 赫伯特·L
 gray levels, multiplication of 灰度乘法
 Great Internet Mersenne Prime Search 因特网大梅森素数搜索
 Greater than, definitely 确定地大于
 Greatest common divisor 最大公因子
 binary algorithms for 最大公因子二进算法
 Euclidean algorithm for 最大公因子欧几里得算法, 见 Euclid's algorithm
 multiprecision 多精度最大公因子
 of n numbers n 个数的最大公因子
 of polynomials 多项式的最大公因子
 within a unique factorization domain 惟一因子分解整环中的最大公因子
 Greatest common right divisor 最大公共右因子
 Greedy algorithm 贪婪算法
 Greek mathematics 希腊数学
 Green, Bert Franklin, Jr. 小格林, 伯特·富兰克林
 Greenberger, Martin 格林伯格, 马丁
 Greene, Daniel Hill 格林, 丹尼尔·希尔
 Greenwood, Joseph Arthur 格林伍德, 约瑟夫·阿瑟
 Greenwood, Robert Ewing 格林伍德, 罗伯特·尤因
 Gregory, Robert Todd 格里戈里, 罗伯特·托德
 GRH 广义黎曼假设: 代数数的扩充黎曼假设
 Groote, Hans Friedrich de 格鲁特, 汉斯·弗里德里克·德
 Grosswald, Emil 格罗斯瓦尔德, 艾米尔
 Grotefeld, Andreas Friedrich Wilhelm 格罗特菲尔德, 安德烈亚斯·弗里德里克·威廉
 Groups 群
 Galois 伽罗瓦群
 Grube, Andreas 格鲁伯, 安德烈亚斯
 Grünwald, Vittorio 格伦沃尔德, 维托里奥
 Guaranteed randomness 保证随机性
 Guard digits 保护数字
 Gudenberg 古登伯格, 见 Wolff von Gudenberg
 Guessing, amplified 猜测扩大
 Guldoud, Jean 古罗德, 琼
 Gustavson, Fred Gehrung 古斯塔夫森, 弗雷德·格龙
 Guy, Michael John Thirion 盖伊, 迈克尔·约翰·瑟里安

- Guy, Richard Kenneth 盖伊, 理查德·肯尼思
- Haber, Seymour 哈伯, 西摩
- Habicht, Walter 哈比希特, 沃尔特
- Hadamard, Jacques Salomon 阿达马, 雅克·所罗门
inequality 阿达马不等式
- Hafner, James Lee 哈夫纳, 詹姆斯·李
- Hajjar, Mansour 哈加, 曼素尔
- HAKMEM MIT AI Laboratory Memo 麻省人工智能实验室备忘录
- Ha.berstam, Henri 哈伯斯塔姆, 海尼
- Halewyn, Christopher Neil van 哈勒维, 克里斯托弗·内尔·范
- Halliwell-Phillipps, James Orchard 哈利韦尔·菲利普斯, 詹姆斯·奥查德
- Halton, John Henry 霍尔顿, 约翰·亨利
- Halving 折半
continued fraction 连分数折半
modular 模折半
- Hamblin, Charles Leonard 汉布林, 查尔斯·伦纳德
- Hamlet, Prince of Denmark 哈姆莱特, 丹麦王子
- Hammersley, John Michael 哈默斯利, 约翰·迈克尔
- Hamming, Richard Wesley 汉明, 理查德·韦斯利
- Handscorn, David Christopher 汉德斯康布, 戴维·克里斯托弗
- Hardy identities 容易得到的恒等式
- Hansen, Eldon Robert 汉森, 埃尔登·罗伯特
- Hansen, Walter 汉森, 沃尔特
- Hanson, Richard Joseph 汉森, 理查德·约瑟夫
- Haralambous, Yannis (Χαραλ άμπος, 'Ιωάννης) 哈拉拉姆波斯, 扬尼斯
- Hard-core bit 核心位
- Hardware 硬件: 计算机线路
algorithms suitable for 适合于硬件的算法
- Hardy, Godfrey Harold 哈迪, 戈弗雷·哈罗德
- Harmonic numbers H_n 调和数
fractional 分数调和数
table of 调和数表
- Harmonic probability 调和概率
- Harmonic sums 调和和
- Harmuth, Henning Friedolf 哈姆斯, 亨宁·弗里多夫
- Harriot, Thomas 哈里奥特, 托马斯
- Harris, Bernard 哈里斯, 伯纳德
- Harris, Vincent Crockett 哈里斯, 文森特·克罗克特
- Harrison, Charles, Jr. 小哈里森, 查尔斯
- Hashing 散列
- Hassler, Hannes 哈斯勒, 汉尼斯
- Håstad, Johan Torkel 哈斯特德, 约翰·托克尔
- Haynes, Charles Edmund, Jr. 小海恩斯, 查尔斯·埃德蒙德
- hcf Highest common factor 最大公因子 见 Greatest common divisor
- Hebb, Kevin Ralph 赫布, 凯文·拉尔夫
- Heideman, Michael Thomas 海德曼, 迈克尔·托马斯
- Heilbronn, Hans Arnold 海尔布伦, 汉斯·阿诺德
- Heindel, Lee Edward 海因德尔, 李·爱德华
- Hellman, Martin Edward 赫尔曼, 马丁·爱德华
- Henrici, Peter Karl Eugen 亨利奇, 彼得·卡尔·尤金
- Hensel, Kurt Wilhelm Sebastian 亨塞尔, 库尔特·威廉·赛巴斯蒂安
lemma 亨塞尔引理
- Hensley, Douglas Austin 亨斯利, 道格拉斯·奥斯丁
- Heringa, Jouke Reyn 赫林加, 朱克·雷恩
- Hermelink, Heinrich 赫姆林克, 海因里希
- Hermite, Charles 赫米特, 查尔斯
- Herrmann, Hans Jürgen 赫尔曼, 汉斯·朱尔金
- Hershberger, John Edward 赫什伯格, 约翰·爱德华
- Herzog, Thomas Nelson 赫佐格, 托马斯·纳尔逊
- Hexadecimal digits 十六进制数字
- Hexadecimal number system 十六进制数系统
floating point 浮点十六进制数
nomenclature for 十六进制数系统的记名法
- Higham, Nicholas John 海厄姆, 尼古拉斯·约翰
- Hilferty, Margare: Mary 希尔费蒂, 玛格丽特·玛丽
- Hill, Ian David 希尔, 伊恩·戴维
- himult himult 运算
- Hindu mathematics 印度数学
- HITACHI SR2201 computer 日立 SR2201 计算机
- Hitchcock, Frank Lauren 希契科克, 弗兰克·劳伦
- Hlawka, Edmund 赫劳卡, 埃德蒙德
- HLT (halt) 停止指令
- Hobby, John Douglas 霍比, 约翰·杜格尔斯
- Hoffmann, Immanuel Carl Volkmar 霍夫曼, 伊曼纽尔·卡尔·沃尔克默
- Holte, John Myrom 霍尔特, 约翰·迈罗姆
- Homogeneous polynomial 齐次多项式
- Hopcroft, John Edward 霍布克洛夫特, 约翰·爱德华
- Hörmann, Wolfgang 霍尔曼, 沃尔夫冈
- Hörner, Horst Helmut 霍纳, 霍斯特·赫尔穆特
- Horner, William George 霍纳, 威廉·乔治
rule for polynomial evaluation 多项式计算的霍纳规则

- Horowitz, Ellis 霍罗威茨, 埃利斯
 Howard, John Vernon 霍华德, 约翰·弗农
 Howell, Thomas David 豪厄尔, 托马斯·戴维
 Hoyle, Edmond, rules 霍尔, 埃德蒙德规则
 Huang, Ming-Deh Alfred 黄铭德
 Huff, Darrell Burton 赫夫, 达雷尔·伯顿
 Hull, Thomas Edward 赫尔, 托马斯·爱德华
 Hurwitz, Adolf 赫尔威茨, 阿道夫
 Huygens(= Huyghens), Christiaan 体金斯, 克里斯琴
 Hyde, John Porter 海德, 约翰·波特
 Hyperbolic tangent 双曲正切
 Hyperplanes 超平面
 IBM 704 computer IBM 704 计算机
 IBM 7090 computer IBM 7090 计算机
 IBM System/360 computers, early models IBM System/360 计算机的早期模型
 IBM System/360 Model 91 computer IBM System/360 Model 91 计算机
 IBM System/370 computers IBM System/370 计算机
 Ibn Ezra, Abraham ben Meir 伊布恩·埃兹拉·亚伯拉罕·本·梅尔
 Idempotent 等幂
 Identity element 等元
 IEEE standard floating point IEEE 标准浮点
 Ikebe, Yasuhiko 池边八洲彦
 Ill-conditioned matrix 病态矩阵
 Images, digitized 数字化图像
 Imaginary radix 虚数进制
 Impagliazzo, Russell Graham 伊姆佩格利亚佐, 拉塞尔·格拉姆
 Improving randomness 改进随机性
 IMSL: The International Mathematics and Statistics Library 国际数学与统计学图书馆
 Inclusion and exclusion principle 容斥原理
 Incomplete gamma function 不完备的 Γ 函数
 Increment in a linear congruential sequence 线性同余序列的增量
 Independence, algebraic 代数独立性
 Independence, Linear 线性独立性
 Independence of random numbers 随机数独立性
 Index modulo p 模 p 指数
 Indian mathematics 印度数学
 Induced functions 归纳函数
 Induction, mathematical 数学归纳法
 on the course of computation 关于计算过程的数学归纳法
 Inductive assertions 归纳断言
 Infinite continued fractions 无穷连分数
 Infinity, representation of 无穷的表示
 Inner product 内积
 Integer, random 随机整数
 among all positive integers 所有正整数中的随机整数
 in a bounded set 有界集中的随机整数
 Integer solution to equations 方程的整数解
 Integer-valued distributions 整数值分布
 Integrated circuit module 集成电路模块
 Integration 积分
 Interesting point 有趣点
 Internet 因特网
 Interpolation 内插多项式
 Interpretive routines 解释程序
 Interval arithmetic 区间算术
 Inverse Fourier transform 逆傅里叶变换
 Inverse function 反函数, 也见 Reversion of power series
 Inverse matrix 逆矩阵
 Inverse modulo m 模 m 逆
 Inverse congruential sequence 逆同余序列
 Irrational number 无理数
 multiples of, mod 1 无理数的模 1 倍数
 transcendental 超越无理数
 Irrational radix 无理数进制
 Irrationality, quadratic 二次无理性
 Irreducible polynomial 不可约多项式
 Ishibashi, Yoshihiro 石桥善弘
 Islamic mathematics 伊斯兰数学
 Iteration of power series 幂级数的迭代
 Iterative n -source 迭代 n 源
 Iverson, Kenneth Eugene 艾弗森, 肯尼思·尤金
 Jabotinsky, Eri 雅博金斯基, 埃里
 JACM: Journal of the ACM, a publication of the Association for Computing Machinery since 1954
 《ACM 杂志》: 美国计算机学会的刊物, 创刊于 1954 年
 Jacobi, Carl Gustav Jacob 雅可比, 卡尔·古斯塔夫·雅可比
 symbol 雅可比符号
 JAE (jump A even) A 为偶转移指令
 Jaeschke, Gerhard Paul Werner 贾斯切克, 格哈特·保罗·沃纳
 Jager, Hendrik 贾格尔, 亨德里克

- Ja'Ja', Joseph Fand 杰杰,约瑟夫·费里德
 Jaansens, Frank 詹森,弗兰克
 Jansson, Briger 詹森,布里格
 JAO (jump A odd) A 奇转移指令
 Japanese mathematics 日本数学
 Jayadeva, Acharya 杰亚德瓦,阿查里亚
 Jebelean, Tudor 杰伯利恩,图德
 Jefferson, Thomas 杰斐逊,托马斯
 Jensen, Johan Ludvig William Valdemar 詹森,约翰·路德维格·威廉·瓦尔德马
 Jevons, William Stanley 杰文斯,威廉·斯坦利
 Jiuzhang Suanshu 《九章算术》
 Johnk, Max Detlev 约翰,马克斯·德特尔夫
 Johnson, Don Herrick 约翰逊,唐·赫里克
 Johnson, Jeremy Russell 约翰逊,杰里米·罗素
 Johnson, Samuel 约翰逊,塞缪尔
 Jokes 乔克斯
 Jones, Hugh 琼斯,休
 Jones, Terence Cordon 琼斯,特伦斯·戈登
 Jong, Lieuwe Sytse de 乔恩,利尤韦·赛特西·德
 Jonquières, Jean Philippe Ernest de Fauque de 琼奎里
 斯,琼·菲利普·欧内斯特·德·福克·德
 Jordane, Joshua 乔丹尼,乔舒亚
 Judd, John Stephen 贾德,约翰·斯蒂芬
 Jurkat, Wolfgang Bernhard 朱尔卡特,沃尔夫冈·伯恩哈德
 Justeson, John Stephen 贾斯特森,约翰·斯蒂芬
 JXE (jump X even) X 偶转移指令
 JXO (jump X odd) X 奇转移指令
 k -distributed sequence k 分布序列
 Kac, Mark 卡克,马克
 Kahan, William Morton 卡亨,威廉·莫顿
 summation formula 卡亨求和公式
 Kaib, Michael Andreas 凯布,迈克尔·安德烈亚斯
 Kaltofen, Erich Leo 卡尔托芬,埃里克·利奥
 Kaminski, Michael 卡明斯基,迈克尔
 Kanada, Yasumasa 金田康正
 Kankaala, Kari Veli Antero 坎加拉,卡里·维利·安特
 罗
 Kannan, Ravindran 卡南,雷文德兰
 Kanner, Herbert 卡恩纳,赫伯特
 Karatsuba, Anatolii Alekseevich (Карацуба, Анатолий
 Алексеевич) 卡拉特萨巴,阿纳托里·阿历克谢
 维奇
 Karlsruhe, University of 卡尔苏鲁赫大学
 Kátai, Imre 卡泰,伊姆里
 Katz, Victor Joseph 卡茨,维克多·约瑟夫
 Keir, Roy Alex 基尔,罗伊·亚历克斯
 Keller, Wilfrid 凯勒,威尔弗里德
 Kempner, Aubrey John 肯珀,奥伯里·约翰
 Kendall, Maurice George 肯德尔,莫里斯·乔治
 Kermack, William Ogilvy 克马克,威廉·奥格尔维
 Kerr, Leslie Robert 克尔,莱斯利·罗伯特
 Kesner, Oliver 基斯尼尔,奥利弗
 Khirchin, Alexander Yakovlevich (Хинчин, Александр
 Яковлевич) 辛钦,亚历山大·雅科夫列维奇
 Kinderman, Albert John 金德曼,艾伯特·约翰
 Klarner, David Anthony 克拉内尔,戴维·安东尼
 Klem, Laura 克莱姆,劳拉
 Knop, Robert Edward 诺普,罗伯特·爱德华
 Knopfmacher, Arnold 诺普夫梅切,阿诺德
 Knopfmacher, John Peter Louis 诺普夫梅切,约翰·
 彼得·路易斯
 Knopp, Konrad Hermann Theodor 诺普,康拉德·赫
 尔曼·西奥多
 Knorr, Wilbur Richard 诺尔,威尔伯·理查德
 Knott, Cargill Gilston 诺特,卡吉尔·吉尔斯顿
 Knuth, Donald Ervin 克努特,唐纳德·欧文(高德纳)
 Knuth, Jennifer Sierra 克努特,詹尼弗·西拉(高小
 珍)
 Knuth, John Martin 克努特,约翰·马丁(高小强)
 Kohavi, Zvi 科哈韦,兹维
 Koksma, Jurjen Ferdinand 科克斯马,朱尔珍·费迪南
 德
 Kolmogorov, Andrei Nikolaevich (Колмогоров, Андрей
 Николаевич) 科尔莫戈罗夫,安德列·尼古拉耶
 维奇
 Kolmogorov-Smirnov distribution 科尔莫戈罗夫-斯
 米尔诺夫分布
 table 科尔莫戈罗夫-斯米尔诺夫分布表
 Kolmogorov-Smirnov test 科尔莫戈罗夫-斯米尔诺夫
 检验
 Koons, Florence 库恩斯,弗洛伦斯
 Kornerup, Peter 康纳勒普,彼得
 Korobov, Nikolai Mikhailovich (Коробов, Николай
 Михайлович) 科罗波夫,尼古拉·米哈伊洛维
 奇
 Kovács, Béla 科瓦茨,贝拉
 Kraitchik, Maurice Borisovitch (Крайчик, Меер
 Борисович) 克莱特奇克,莫里斯·鲍里索维茨
 Krandick, Werner 克兰迪克,沃纳
 Krishnamurthy, Edayathumangalam Venkataraman 克

- 里斯纳穆尔思,埃达亚苏曼加朗·文卡塔拉曼
 Kronecker, Leopold 克罗内克,利奥波德
 Kruskal, Martin David 克鲁斯卡尔,马丁·戴维
 KS test KS 检验,见 Kolmogorov-Smirnov test
 Kucznik, Marek 库茨尼克,马雷科
 Kuipers, Lauwerens 凯珀,劳伦斯
 Kulisch, Ulrich Walter Heinz 库利斯,乌尔里克·沃
 尔特·海因茨
 Kung, Hsiang Tsung 孔祥重
 Kurita, Yoshiharu 栗田良春
 Kuttaka 库塔卡
 Kuz'min, Rodion Osievich (Кузьмин, Родион
 Осиевич) 库兹明,罗迪昂·奥西耶维奇
 I^0 -chain I^0 链
 L^3 algorithm L^3 算法
 La Touche, Maria Price 拉塔奇,玛丽亚·普赖斯
 La Vallée Poussin, Charles-Jean-Gustave-Nicolas de 拉
 瓦利·波辛,查尔斯·琼-古斯塔夫·尼古拉斯·德
 Laderman, Julian David 拉德曼,朱丽安·戴维
 Lafon, Jean-Claude 拉丰,琼·克劳德
 Lagarias, Jeffrey Clark 拉加利亚斯,杰弗里·克拉克
 Lagged Fibonacci sequences 延搁斐波那契序列
 Lagny, Thomas Fantet de 拉格尼,托马斯·范特·德
 Lagrange (= de la Grange), Joseph Louis, Comte 拉格
 朗日 (= 德拉·格朗日),约瑟夫·路易斯·科姆特
 interpolation formula 拉格朗日插值公式
 inversion formula 拉格朗日反演公式
 Lags 延搁
 Lake, George Thomas 莱克,乔治·托马斯
 Lakshman, Yagati Narayana 拉克斯曼,亚加蒂·纳拉
 亚纳
 Lalanne, Léon Louis Chrétien 莱兰尼,利昂·路易斯·
 克里坦
 Lamé, Gabriel 拉默,加布里埃尔
 Landau, Edmund Georg Hermann 兰多,埃德蒙德·乔
 治·赫尔曼
 Laplace (= de la Place), Pierre Simon, Marquis de 拉
 普拉斯,皮埃尔·西蒙,马夸斯·德
 Lapko, Olga Georgievna (Ляпко, Ольга Георгиевна)
 拉普科,奥尔加·格奥尔吉维纳
 Large Prime numbers 大素数
 Las Vegas algorithm 拉斯维加斯算法:使用随机数
 的计算方法,若计算终止,则总能产生正确结果
 Lattice of points 点格
 Lattice reduction 格约简,见 Short vectors
 Laughlin, Harry Hamilton 劳克林,哈里·哈密顿
 Laurent, Paul Mathieu Hermann, series 劳伦特,保罗
 ·马休·赫尔曼级数
 Lauwerier, Hendrik Adolf 劳韦里亚,亨德里克·阿道
 夫
 Lavaux, Michel 拉瓦克,米歇尔
 Lavington, Simon Hugh 拉文顿,西蒙·休
 Lawrence, Frederick William 劳伦斯,弗雷德里克·威
 廉
 Lcm: Least common multiple 最小公倍数
 Leading coefficient 前导系数
 Leading digit 前导数字
 Leading zeros 前导零
 Least common left multiple 最小左公倍数
 Least common multiple 最小公倍数
 Least remainder algorithm 最小剩余算法
 Least significant digit 最低有效位
 Lebesgue, Henri Léon, measure 勒贝格,亨利·利昂测
 度
 Lebesgue (= Le Besgue), Victor Amédée 勒贝格,维
 克多·埃米迪
 L'Ecuyer, Pierre 勒凯耶,皮埃尔
 Leeb, Hannes 利布,汉尼斯
 Leeuwen, Jan van 纽文,简·范
 Legendre (= Le Gendre), Adrien Marie 勒让德,艾德
 里安·玛丽
 symbol 勒让德符号
 Léger, Émile 莱杰,艾米尔
 Léger R. 莱杰·R
 Lehman, Russell Sherman 莱曼,拉塞尔·谢尔曼
 Lehmer, Derrick Henry 莱默,德里克·亨利
 Lehmer, Derrick Norman 莱默,德里克·诺曼
 Lehmer, Emma Markovna Trotskaia 莱默,埃玛·马尔
 科夫纳·特罗斯卡亚
 Lehn Jürgen 莱恩·朱尔金
 Leibniz, Gottfried Wilhelm, Freiherr von 莱布尼茨,
 戈特弗里德·威廉,弗雷赫尔·冯
 Lempel, Abraham 伦珀尔,亚伯拉罕
 Lenstra, Arjen Klaas 伦斯特拉,阿金·克拉斯
 Lenstra, Hendrik Willem, Jr. 小伦斯特拉,亨德里克
 ·威廉
 Leonardo Pisano 伦纳多·比萨诺,见 Fibonacci
 Leong, Benton Lau 利昂,本顿·劳(梁捷安)
 Leslic, John 莱斯利,约翰
 Less than, definitely 确定地小于
 Levene, Howard 莱文,霍华德

- LeVeque, William Judson 莱维格, 威廉·贾德森
 Levin, Leonid Anatolevich (Левин, Ленид Анатольевич) 莱文, 利奥尼德·阿纳托里耶维奇
 Levine, Eugene 莱文, 尤金
 Lévy, Paul 莱维, 保罗
 Levy, Silvio Vieira Ferreira 莱维, 西尔维奥·维拉·费雷拉
 Lewis, John Gregg 刘易斯, 约翰·格雷格
 Lewis, Peter Adrian Walter 刘易斯, 彼得·艾德里安·沃尔特
 Lewis, Theodore Gyle 刘易斯, 西奥多·盖尔
 Lexicographic order 字典序
 li; Logarithmic integral function 对数整数函数
 Li, Ming 李明
 Li Yan 李伊
 Lickteig, Thomas Michael 利克特格, 托马斯·迈克尔
 Lindholm, James H 林霍姆, 詹姆斯·H
 Linear congruential sequene 线性同余序列
 choice of increment 线性同余序列增量的选择
 choice of modulus 线性同余序列模数的选择
 choice of multiplier 线性同余序列乘数的选择
 choice of seed 线性同余序列种子的选择
 period length 线性同余序列的周期长度
 subsequence of 线性同余序列的子序列
 Linear equations 线性方程组
 integer solution to 线性方程组的整数解
 Linear factors mod p 模 p 线性因子
 Linear iterative array 线性迭代数组
 Linear lists 线性表
 Linear operators 线性算子
 Linear probing 线性探查
 Linear recurrences 线性递归(推)
 mod m 模 m 线性递归(推)
 Linearly independent vectors 线性无关向量
 Linked memory 链接存储
 Linking automaton 链接自动机
 Linnainmaa, Seppo Ilmari 林南马, 塞普波·伊尔马里
 Liouville, Joseph 莱奥维尔, 约瑟夫
 Lipton, Richard Jay 利普顿, 理查德·杰伊
 Liquid measure 液量单位
 Little Fermat computer 小费马计算机
 Littlewood, John Edensor 利特尔伍德, 约翰·伊登索尔
 LLL algorithm LLL 算法
 Local arithmetic 局部算术
 Locally nonrandom behavior 局部非随机性
 Loewenthal, Dan 洛温特尔, 唐
 Logarithm 对数
 discrete 离散对数
 of a matrix 矩阵的对数
 of a power series 幂级数的对数
 of a uniform deviate 一致偏离的对数
 of ϕ ϕ 的对数
 Logarithmic integral 对数积分
 Logarithmic law of leading digits 前导数字的对数律
 Logarithmic sums 对数和
 Logical operations 逻辑运算, 见 Boolean operations
 Loh, Günter 洛, 冈特
 lomult lomult 函数
 Long division 长除法
 Loos, Rüdiger Georg Konrad 卢斯, 鲁迪格·乔治·康拉德
 Lotti, Grazia 洛蒂, 格拉齐亚
 Lovász, László 洛瓦兹, 拉兹罗
 Lovelace, Augusta Ada Byron King 洛夫莱斯, 奥古斯塔·艾达·拜伦·金
 Countess of 洛夫莱斯伯爵夫人
 Loveland, Donald William 洛夫兰, 唐纳德·威廉
 Lubiw, Anna 卢比, 安娜
 Lubkin, Samuel 卢布金, 塞缪尔
 Luby, Michael George 卢比, 迈克尔·乔治
 Lucas, Francois Édouard Anatole 卢卡斯, 弗朗科伊
 斯·埃杜瓦德·阿纳托尔
 numbers L_n 卢卡斯数 L_n
 Lukes, Richard Francis 卢克斯, 理查德·弗朗西斯
 Lüscher, Martin 勒斯切, 马丁
 Luther, Herber Adesla 卢瑟, 赫伯·艾德斯拉
 m -ary method of exponention 指数的 m 进方法
 Ma, Keju 马柯驹
 machine language versus higher-level languages 机器语言与高级语言
 MacLaren, Malcolm Donald 麦克拉伦, 马尔科姆·唐纳德
 MacMahon, Percy Alexander 麦克马洪, 珀西·亚历山大
 MacMillan, Donald B. 麦克米伦, 唐纳德·B
 MacPherson, Robert Duncan 麦克弗斯, 罗伯特·邓肯
 MacSorley, Ohin Lowe 麦克索利, 奥林·洛
 Maeder, Roman Erich 梅德, 罗曼·艾里奇
 Mahler, Kurt 马勒, 库尔特

- measure 马勒测度
- Makarov, Oleg Mikhailovich (Макаров, Олег Михайлович) 马卡罗夫, 奥列格·米哈伊洛维奇
- Mallows, Colin Lingwood 马洛斯, 科林·林格伍德
- Manasse, Mark Steven 马纳西, 马克·史蒂文
- Manchester University Computer 曼彻斯特大学计算机
- Mandelbrot, Benoit Baruch 曼德尔伯洛特, 贝诺瓦·巴鲁克
- Mangoldt, Hans Carl Friedrich von 曼戈尔德, 汉斯·卡尔·弗里德里克·冯
- function 曼戈尔德函数
- MANIAC III computer MANIAC III 计算机
- Mansour, Yishay 曼苏尔, 易沙
- Mantel, Willem 曼特尔, 威廉
- Mantissa 尾数, 见 Fraction part
- Marczynski, Romuald Wladyslaw 马金斯基, 罗米尔·德·伍拉迪斯罗
- Mariage, Aimé 玛丽亚格, 艾梅
- Mark I computer (Ferranti) Mark I 计算机(费伦蒂)
- Mark II Calculator (Harvard) Mark II 计算器(哈佛)
- Marsaglia, George 马萨格里亚, 乔治
- Martin, Monroe Harnish 马丁, 门罗·哈尼什
- Martin-Löf, Per Erik Rutger 马丁-洛夫, 佩尔·埃里克·拉特格
- Masking 屏蔽
- Math. Comp.: Mathematics of Computation (1960 -), a publication of the American Mathematical Society since 1965; founded by the National Research Council of the National Academy of Sciences under the original title *Mathematical Tables and Other Aids to Computation* (1943—1959) 《计算数学》: 美国数学协会出版的杂志
- Mathematical aesthetics 数学美学
- Matias, Yossi 马蒂亚斯, 约斯
- Matrix 矩阵
- characteristic polynomial 矩阵的特征多项式
- determinant 矩阵的行列式
- greatest common right divisor 矩阵的最大公共右因子
- inverse 逆矩阵
- multiplication 矩阵乘法
- null space 零空间矩阵
- permanent 矩阵的积和式
- rank 矩阵的秩
- semidefinite 半确定的矩阵
- singular 奇异矩阵
- triangularization 矩阵的三角化
- Matrix (Bush), Irving Joshua 马特里克(布什), 欧文·乔舒亚
- Matsumoto, Makoto 松本真
- Matthew, Saint 马修, 圣
- Matula, David William 马图拉, 戴维·威廉
- Mauchly, John William 莫奇利, 约翰·威廉
- Maximum of random deviates 随机偏离的极大值
- Maximum-of- t test t 的极大值检验
- Maya Indians 玛雅印第安人
- Mayer, Dieter Heinz Jörg 迈耶, 迪特尔·海因茨·乔格
- McCarthy, Daniel Patrick 麦卡锡, 丹尼尔·帕特里克
- McClellan, Michael Terence 麦克莱伦, 迈克尔·特伦斯
- McCracken, Daniel Delbert 麦克拉肯, 丹尼尔·德尔伯特
- McCurley, Kevin Snow 麦克利, 凯文·斯诺
- McEliece, Robert James 麦克利斯, 罗伯特·詹姆斯
- McKendrick, Anderson Gray 麦肯德里克, 安德森·格雷
- Mean, evaluation of 均值计算
- Measure, units of 测度单位
- Measure theory 测度论
- Mediant rounding 中间舍入
- Messel, Daniel Friedrich Ernst 迈塞尔·丹尼尔·弗雷德里克·厄恩斯特
- Mellin, Robert Hjalmar, transforms 梅伦, 罗伯特·亚尔马变换
- Mendelsohn, Nathan Saul 门德尔松, 内森·索尔
- Mendès France, Michel 门德兹·弗朗斯, 迈克尔
- Mental arithmetic 智力算术
- Merit, figure of 品质指标
- Mersenne, Marin 梅森, 马林
- multiplication 梅森乘法
- numbers 梅森数
- primes 梅森素数
- Mertens Franz Carl Joseph 默滕斯·弗朗兹·卡尔·约瑟夫
- constant 默滕斯常数
- METAFONT METAFONT 语言
- METAPOST METAPOST 语言
- Metrology 计量学, 计量衡制
- Metropolis, Nicolas Constantine 梅特罗波利斯, 尼古拉斯·康斯坦丁

- Metze, Gernot 梅茨, 杰诺特
 Meyer, Albert Ronald da Silva 迈耶, 艾伯特·罗纳德·达·席尔瓦
 Micali, Silvio 迈克利, 席尔瓦
 Michigan, University of 密执安大学
 Middle-square method 平方取中法
 Midpoint 中点
 Mignotte, Maurice 米格诺特, 莫里斯
 Mikami, Yoshio 三上义夫
 Mikusinski, Jan 米库辛斯基·简
 Miller, Gary Lee 米勒, 加里·李
 Miller, James M 米勒, 詹姆斯·M
 Miller, Jeffrey Charles Percy 米勒, 杰弗里·查尔斯·珀西
 Miller, Kenneth William 米勒, 肯尼思·威廉
 Miller, Victor Saul 米勒, 维克多·索尔
 Miller, Webb Colby 米勒, 韦布·科尔比
 Milne-Thompson, Louis Melville 米尔恩-汤姆森, 路易斯·梅尔维尔
 Minimizing a quadratic form 极小化一个二次型
 Minimum polynomial 极小多项式
 Minkowski, Hermann 明柯维斯基, 赫尔曼
 Minus zero 负零
 MIP-years MIP 年
 Miranker, Willard Lee 米朗克, 威拉德·李
 Mises, Richard, Edler von 迈西泽, 理查德·艾德勒·冯
 Mitchell, Gerard Joseph Francis Xavier 米切尔, 杰拉德·约瑟夫·弗朗西斯·泽维尔
 MIX computer MIX 计算机
 binary version MIX 计算机的二进形式
 floating point attachment MIX 计算机的浮点附件
 Mixed congruential method 混合同余方法, 见 Linear congruential sequence
 Mixed-radix number systems 混合进制数系统
 addition and subtraction 混合进制数的加减法
 balanced 平衡的混合进制数系统
 comparison 混合进制数的比较
 counting by 1s 以 1 计数的混合进制数
 multiplication and division 混合进制数的乘除法
 radix conversion 混合进制数的进制转换
 Mixture of distribution functions 分布函数的混合
 Mobius, August Ferdinand, function 莫比尤斯, 奥古斯特·费迪南德函数
 inversion formula 莫比尤斯反演公式
 mod 取模
 mod m arithmetic 模 m 算术
 addition 模 m 加法
 division 模 m 除法
 halving 模 m 折半
 multiplication 模 m 乘法
 on polynomial coefficients 多项式系数的模 m 算术
 square root 模 m 平方根
 subtraction 模 m 减法
 Model V computer Model V 计算机
 Modular arithmetic 模算术, 同余算术
 complex 复数的模算术
 Modular method for polynomial gcd 多项式最大公因子的模方法
 Modulus in a linear congruential sequence 线性同余序列中的模数
 Moench, Robert Thomas 芒克, 罗伯特·托马斯
 Møller, Ole 莫勒, 奥利
 Monahan, John Francis 莫纳汉, 约翰·弗朗西斯
 Monic polynomial 首一多项式
 Monier, Louis Marcel Gino 莫尼尔, 路易斯·马塞尔·吉诺
 Monkey tests 猴子检验
 Monomials, evaluation of 首一多项式的求值
 Monotonicity 单调性
 Monte Carlo 蒙特卡罗
 Monte Carlo method 蒙特卡罗方法: 任何使用随机数的计算方法(可能不产生正确答案), 也见 Las Vegas algorithms, Randomized algorithms
 Montgomery, Hugh Lowell 蒙哥马利, 休·洛厄尔
 Montgomery, Peter Lawrence 蒙哥马利, 彼得·劳伦斯
 multiplication mod m 蒙哥马利模 m 乘法
 Moore, Donald Philip 穆尔, 唐纳德·菲利普
 Moore, Louis Robert 穆尔, 路易斯·罗伯特
 Moore, Ramon Edgar 穆尔, 拉蒙·埃德加
 Moore School of Electrical Engineering 穆尔电气工程学校
 Morain, Francois 莫兰, 弗朗科伊斯
 Morgenstern, Jacques 摩根斯坦, 雅克
 Morley, Geoffrey Hugh 莫利, 杰弗里·休
 Morris, Robert 莫里斯, 罗伯特
 Morrison, Michael Allan 莫里森, 迈克尔·阿伦
 Morse, Harrison Reed 莫尔斯, 哈里森·里德
 Morse, Samuel Finley Breese, code 莫尔斯, 塞缪尔·芬利·布里斯代码

- Moses, Joel 摩西·乔尔
- Most significant digit 最高有效数字
- Motzkin, Theodor Samuel 莫茨金, 西奥多·塞缪尔
- Middle-square method 混合平方方法
- Muller, Mervin Edgar 马勒, 默文·埃德加
- Multinomial coefficients 多项式系数
- Multinomial theorem 多项式定理
- Multiple-precision arithmetic 多精度算术
- addition 多精度加法
 - comparison 多精度比较
 - division 多精度除法
 - greatest common divisor 多精度最大公因子
 - multiplication 多精度乘法
 - radix conversion 多精度进制转换
 - subtraction 多精度减法
 - table of constants 多精度常数表
- Multiples 倍数
- Multiples of an irrational number mod 1 无理数的模 1 倍数
- Multiplication 乘法
- complex 复数乘法
 - double-precision 双精度乘法
 - fast(asymptotically) (渐近地)快速乘法
 - floating point 浮点乘法
 - fractions 小数乘法
 - matrix 矩阵乘法
 - Mersenne 梅森乘法
 - mixed-radix 混合进制乘法
 - mod m 模 m 乘法
 - mod $u(x)$ 模 $u(x)$ 乘法
 - modular 模数乘法
 - multiprecision 多精度乘法
 - multiprecision by single-precision 通过单精度的多精度乘法
 - polynomial 多项式乘法
 - power series 幂级数乘法
- Multiplicative congruential method 乘同余法
- Multiplier in a linear congruential sequence 线性同余序列中的乘数
- Multiply-and-add algorithm 乘和加算法
- Multiprecision 多精度:任意精度
- Multisets 多重集合
- operations on 多重集合的运算
 - terminological discussion 多重集合术语讨论
- Multivariate polynomials 多变量多项式
- chains 多变量多项式链
 - factors 多变量多项式的因子
 - noncommutative 非交换的多变量多项式
 - roots of 多变量多项式的根
- Munro, James Ian 芒罗, 詹姆斯·伊恩
- Musical notation 音乐记号
- Musinski, Jean Elisabeth Abramson 穆辛斯基, 琼·伊丽莎白·艾布拉姆森
- Musser, David Rea 马瑟, 戴维·雷亚
- N-source N 源
- Nadler, Morton 纳德勒, 莫顿
- Nance, Richard Earle 南斯, 理查德·厄尔
- Nandi, Salil Kumar 南迪, 萨利尔·孔默
- NaNs 无穷、无定义或不寻常的量
- Napier, John, Laird of Merchiston 内皮尔, 约翰, 马其顿领主
- Native American mathematics 美国数学
- Needham, Joseph 尼达姆, 约瑟夫
- Negabinary number system 负二进制数系统
- Negacyclic convolution 负循环卷积
- Negadecimal number system 负十进制数系统
- Negative binomial distribution 负二项分布
- Negative digits 负数字
- Negative numbers, representation of 负数的表示
- Negative radices 负进制
- Neighborhood of a floating point number 一个浮点数的邻域
- Neugebauer, Otto Eduard 诺伊格鲍尔, 奥托·埃杜阿德
- Neumann, John von (= Margittai Neumann János 诺伊曼, 约翰·冯(= 马吉塔·诺伊曼·雅诺什))
- Newcomb, Simon 纽科姆, 西蒙
- Newman, Donald Joseph 纽曼, 唐纳德·约瑟夫
- Newton, Isaac 牛顿, 艾萨克
- interpolation formula 牛顿内插公式
 - method for rootfinding 牛顿求根法
- Ni, Wen-Chun 倪文君
- Nicomachus of Gerasa 杰拉萨的尼科马彻斯
- Niederreiter, Harald Gunther 尼德尔雷特, 哈拉德·冈瑟
- Nijenhuis, Albert 尼詹惠斯, 艾伯特
- Nine Chapters on Arithmetic 《九章算术》
- Nines, casting out 舍九法
- Nines' complement notation 九的补码记法
- Nisan, Noam 尼桑, 诺姆
- Niven, Ivan Morton 尼文, 伊凡·莫顿
- Nonary (radix 9) number system 九进制数系统

- Noncommutative multiplication 非可交换乘法
- Nonconstructive proofs 非结构化证明
- Nonnegative 非负的
- Nonsingular matrix 非奇异矩阵:具有非零行列式值的矩阵
- Norm of a polynomial 多项式的范数
- Normal deviates 正态偏离:具有正态分布的随机数
dependent 相关正态偏离
direct generation 正态偏离的直接生成
square of 正态偏离的平方
- Normal distribution 正态分布
tail of 正态分布的尾部
variations 正态分布变形
- Normal evaluation schemes 正常计算方案
- Normal numbers 正规数
- Normalization of divisors 除数的规格化
- Normalization of floating point numbers 浮点数的规格化
- Normand, Jean-Marie 诺曼德, 琼-玛丽
- Norton, Graham Hilton 诺顿, 格拉姆·希尔顿
- Norton, Karl Kenneth 诺顿, 卡尔·肯尼思
- Norton, Victor Thane, Jr. 小诺顿, 维克多·塞恩斯
- Notations, index to 符号索引
- Nozaki, Akihiro 野山奇昭弘
- NP-complete problems NP 完全问题
- Null space of a matrix 矩阵的零空间
- Number field sieve 数域筛
- Number fields 数域
- Number sentences 数的句子
- Number system 数系统:用于表示数的语言
balanced binary 平衡二进制数系统
balanced decimal 平衡的十进制数系统
balanced mixed-radix 平衡的混合进制数系统
balanced ternary 平衡的三进制数系统
binary (radix 2) 二进制数系统
combinatorial 组合数系统
complex 复数系统
decimal (= denary, radix ten) 十进制数系统
duodecimal (radix twelve) 十二进制数系统
factorial 阶乘数系统
Fibonacci 斐波那契数系统
floating point 浮点数系统
hexadecimal (radix sixteen) 十六进制数系统
mixed-radix 混合进制数系统
modular 模数系统
negabinary (radix-2) 负二进制数系统
negadecimal 负十进制数系统
nonary (radix 9) 九进制数系统
octal (= octonary = octonol, radix 8) 八进制数系统
 p -adic p -adic 数系统
 ϕ ϕ 数系统
positional 定位计数系统
primitive tribal 本原族数系统
quater-imaginary (radix 2i) 虚四数系统
quaternary (radix 4) 四进制数系统
quinary (radix 5) 五进制数系统
rational 有理数系统
regular continued fraction 正则连分数系统
reversing binary 反序二进制数系统
revolving binary 旋转二进制数系统
sedecimal (= hexadecimal) 十六进制数系统
senary (radix 6) 六进制数系统
sendenary (= hexadecimal) 十六进制数系统
septenary (radix 7) 七进制数系统
sexagesimal (radix sixty) 六十进制数系统
slash 开缝数系统
ternary (radix 3) 三进制数系统
vigesimal (radix twenty) 二十进制数系统
Numerical instability 数值不稳定性
Nunes (= Nuñez Salaciense = Nonius)
Pedro 努涅斯·佩德罗
Nussbaumer, Henri Jean 努斯鲍默, 亨利·琼
Octal (radix 8) number system 八进制数系统
Octavation 八进制转换
Odd-even method 奇偶方法
Odlyzko, Andrew Michael 奥德里兹科, 安德鲁·迈克尔
DFLO DFLO 子程序
Oldham, Jeffrey David 奥尔德姆, 杰弗里·戴维
Oliver, Ariadne 奥利弗, 阿里亚德内
Olivos Aravena, Jorge Augusto Octavio 奥利沃斯·阿拉维纳, 乔治·奥古斯托·奥克塔维欧
One-way function 单向函数
Ones' complement notation 1 的补码记法
Online algorithms 在线算法
Operands 操作数:运算中的量,例如,在计算 $u + v$ 时的 u 和 v
Ophelia, daughter of Polonius 奥非利娅, 波洛尼亚斯之女
Optimum methods of computation 计算的最优方法, 见 Complexity
OR (bitwise or) 或

- Order of a modulo m a 的模 m 阶
 Order of an element in a field 一个域中元素的阶
 Order of magnitude zero 量 0 的阶
 Ordered hash table 有序散列表
 Oriented binary tree 有向二叉树
 Oriented tree 有向树
 Ostrowski, Alexander Markus 奥斯特罗斯基, 亚历山大·马卡斯
 Oughtred, William 奥特莱德, 威廉
 Overflow 上溢出
 exponent 指数上溢
 fraction 小数上溢
 rounding 舍入上溢
 Overstreet, Claude Lee, Jr. 小奥弗斯特里特, 克劳德·李
 Owen, John 欧文, 约翰
 Owings, James Claggett, Jr. 小奥因斯, 詹姆斯·克拉杰特
 Ozawa, Kazufumi 小泽一文
 p -adic numbers p -adic 数
 packing 打包, 组装
 Padé, Henri Eugène 帕德, 亨利·尤金
 Padags Andris 帕德格斯·安德里斯
 Pairwise independence 对偶独立
 Palindromes 回文
 Palmer, John Franklin 帕尔默, 约翰·富兰克林
 Pan, Victor Yakovlevich (Пан, ВЯКТОР Яковлевич) 潘, 维克托·雅科夫列维奇
 Panario Rodríguez, Daniel Nelson 帕纳里奥·罗德里格茨, 丹尼尔·纳尔逊
 Pandu Rangan, Chandrasekaran 潘都·兰根, 钱德雷斯卡兰
 Papadimitriou, Christos Harilaos 帕帕迪米特里欧, 克里斯托斯·哈里劳斯
 Pappus of Alexandria 亚历山德里亚的帕帕斯
 Paradox 悖论
 Parallel computation 并行计算
 Parameter multiplications 参数乘法
 Parameter step 参数步
 Pardo 帕多, 见 Trabb Pardo
 Park, Stephen Kent 帕克, 史蒂芬·肯特
 Parlett, Beresford Neall 帕利特, 贝雷斯弗德·尼尔
 Parry, William 帕里, 威廉
 Partial derivatives 部分偏导
 Partial fraction expansion 部分小数展开
 Partial ordering 偏序
 Partial quotients 部分商
 distribution of 部分商的分布
 Partition test 分划检验
 Partitions of a set 集合的分划
 Partitions of an integer 整数的分划
 Pascal, Blaise 帕斯卡, 布莱斯
 Pascal-SC language Pascal-SC 语言
 Patashnik, Oren 帕特什尼克, 奥伦
 Paterson, Michael Stewart 佩特森, 迈克尔·斯图尔特
 Patience 耐心, 忍耐力
 Patterson, Cameron Douglas 佩特森, 卡梅伦·道格拉斯
 Paul, Nicholas John 保罗, 尼古拉斯·约翰
 Pawlak, Zdzislaw 波拉克, 泽德吉斯洛
 Payne, William Harris 佩恩, 威廉·哈里斯
 Paz, Azaria 帕兹, 阿扎里亚
 Peano, Giuseppe 皮亚诺, 吉乌塞普
 Pearson, Karl 皮尔逊, 卡尔
 Perice, Charles Santiago Sanders 佩里斯, 查尔斯·圣地亚哥·桑德斯
 Penk, Michael Alexander 彭克, 迈克尔·亚历山大
 Penney, Walter Francis 彭尼, 沃尔特·弗朗西斯
 Pentium computer chip 奔腾计算机芯片
 Percentage points 百分比点
 Perfect numbers 完全数
 Perfect squares 完全平方
 Period in a sequence 序列的周期
 length of 序列的周期长度
 Periodic continued fraction 周期连分数
 Permanent 积和式
 Permutation 排列: 一个集合的有序安排
 mapped to integers 整数映像的排列
 random 随机排列
 Permutation test 排列检验
 Perron, Oskar 佩龙, 奥斯卡
 Persian mathematics 波斯数学
 Pervushin, Ivan Mikhchevich (Первушин, Иван Микеевич) 佩武欣, 伊凡·迈克希维茨
 Pethő, Attila 佩托, 艾蒂拉
 Petkovšek, Marko 佩特科夫塞克, 马可
 Petr, Karel 佩特, 卡莱尔
 Pfeiffer, John Edward 法伊弗, 约翰·爱德华
 Phalen, Harold Romaine 费伦, 哈罗德·罗曼
 Phi ϕ
 Phillips, Ernest William 菲利普斯, 欧内斯特·威廉
 Pi π

- Picutti, Ettore 皮丘蒂, 埃托尔
 pigeonhole principle 鸽巢原理
 Pingala, Ācharya 平加拉, 阿卡里亚
 Pipeline 管道
 Pippenger, Nicholas John 皮彭格尔, 尼古拉斯·约翰
 Piras, Francesco 皮雷斯, 弗朗西斯科
 Pitfalls of random number generation 随机数生成的缺陷
 Pitteway, Michael Lloyd Victor 皮特威, 迈克尔·劳夫德·维克多
 Places 位置
 Planck, Max Karl Ernst Ludwig, constant 普朗克, 马克斯·卡尔·厄恩斯特·路德维格常数, 普朗克常数
 Plaüger, Phillip James 普罗杰, 菲利普·詹姆斯
 Playwriting 剧本创作
 Plouffe, Simon 普劳夫, 西蒙
 PM system PM 系统
 Pocklington, Henry Cabourn 波克林顿, 亨利·卡博恩
 Pointer machine 指针机器
 Poiror, Hercule 波伊罗特, 赫尔克利
 Poisson, Siméon Denis, distribution 泊松, 西蒙·丹尼斯分布
 Poker test 扑克检验
 Polar coordinates 极坐标
 Polar method 配极方法
 Pollard, John Michael 波拉德, 约翰·迈克尔
 Pólya, György (= George) 波利亚, 乔治
 polynomial 多项式
 addition 多项式加法
 arithmetic modulo m 模 m 多项式算术
 degree of 多项式的次数
 derivative of 多项式的导数
 discriminant of 多项式的判别式
 distribution function 多项式的分布函数
 division 多项式除法
 evaluation 多项式的计算
 factorization 多项式的因子分解
 greatest common divisor 多项式的最大公因子
 interpolation 多项式插值
 irreducible 不可约多项式
 leading coefficient 多项式的首项系数
 monic 首一多项式
 multiplication 多项式乘法
 multivariate 多变量多项式
 norms 多项式范数
 over a field 一个域上的多项式
 over a unique factorization domain 一个唯一因子分解整环上的多项式
 primitive 本原多项式
 primitive modulo p 模 p 本原多项式
 primitive part 多项式的本原部分
 random 随机多项式
 remainder sequence 剩余序列多项式
 resultant 冗余多项式
 reverse of 多项式的逆
 roots of 多项式的根
 sparse 稀疏多项式
 squarefree 无平方多项式
 string 串多项式
 subtraction 多项式减法
 Polynomial chains 多项式链
 Pomerance, Carl 波默兰斯, 卡尔
 Poorten, Alfred Jacobus van der 普尔顿, 阿尔弗雷德·雅各布斯·范·德
 Pope, Alexander 波普, 亚历山大
 Pope, David Alexander 波普, 戴维·亚历山大
 Popper, Karl Raimund 波珀, 卡尔·雷蒙德
 Portable random number generators 可移植的随机数生成程序
 Porter, John William 波特, 约翰·威廉
 Positional representation of numbers 数的定位表示法
 Positive definite quadratic form 正定二次型
 Positive operator 正算子
 Positive semidefinite matrix 半正定矩阵
 Potency 效能
 Power matrix 幂矩阵
 Power series 幂级数, 见 Generating functions
 manipulation of 幂级数的操作
 Power tree 幂树
 Poweroids 似次幂
 Powers, Don M. 鲍尔斯, 唐·M
 Powers, evaluation of 幂的计算
 multiprecision 多精度的幂计算
 polynomial 多项式幂计算
 power series 幂级数幂计算
 Powers, Ralph Earnest 鲍尔斯, 拉尔夫·厄内斯特
 pp: Primitive part 本原部分
 Pr: Probability 概率
 Pratt, Vaughan Ronald 普拉特, 沃恩·罗纳德
 Precision 精度: 一个表示中数字的个数
 double 双精度

- multiple 多精度
 quadruple 四倍精度
 single 单精度:能被一个计算机字容纳
 unlimited 无限精度,也见 Multiple-precision
 Preconditioning 预先处理,见 Adaptation
 Prediction tests 预测检验
 Preston, Richard McCann 普雷斯頓,理查德·麦卡恩
 Primality testing 素性检验
 Prime chains 素数链
 Prime numbers 素数:大于1且无真因子的整数
 distribution of 素数分布
 enumeration of 素数枚举
 factorization into 分解成素因子
 largest known 已知的最大素数
 Mersenne 梅森素数
 size of m th 第 m 个素数的大小
 useful 有用的素数
 useless 无用的素数
 verifying primality of 检验素约性
 Primes in a unique factorization domain 惟一因子分解整环上的素数
 Primitive element modulo m 模 m 本原元
 Primitive notations for numbers 数的本原表示法
 Primitive part of a polynomial 多项式的本原部分
 Primitive polynomial 本原多项式
 Primitive polynomial modulo p 模 p 本原多项式
 Primitive recursive function 原始递归函数
 Primitive root 原根:一个有限域中的本原元
 Pritchard, Paul Andrew 普里查德,保罗·安德鲁
 Probabilistic algorithms 概率算法,见 Randomized algorithms
 Probability 概率:发生的机率
 abuse of 概率的滥用
 over the integers 整数上的概率
 Probert, Robert Lorne 普罗伯特,罗伯特·洛恩
 Programming languages 程序设计语言
 Pronouncing hexadecimal numbers 十六进制数的读音
 Proof of algorithms 算法证明
 Proofs, constructive versus nonconstructive 构造性与非构造性的证明
 Proper factor of v v 的真因子:既非1也非 v 本身的 v 的因子
 Proth, Francois Toussaint 普罗思,弗朗科伊斯·图森特
 Proulx, René 普罗克斯,雷纳
 Pseudodivision of polynomials 多项式的伪除法
 Pseudorandom sequences 伪随机序列
 Ptolemy, Claudius (Πτολεμαῖος Κλαύδιος) 普托里米,克劳迪乌斯
 Public key cryptography 公钥密码
 Purdom, Paul Walton, Jr. 小珀多姆,保罗·沃尔顿
 Pyke, Ronald 派克,罗纳德
 q series q 级数
 Quadratic congruences, solving 二次同余的解
 Quadratic congruential sequences 二次同余序列
 Quadratic forms 二次型
 minimizing, over the integers 整数上的二次型极小化
 Quadratic irrationalities, continued fractions for 二次无理性的连分数
 Quadratic reciprocity law 二次互反律
 Quadratic residues 二次剩余
 Quadratic sieve method 二次筛方法
 Quadruple-precision arithmetic 四倍精度算法
 Quandalle, Philippe 匡特尔,菲利普
 Quasirandom numbers 拟随机数
 Quater imaginary number system 虚四数系统
 Quaternary number system 四进制数系统
 Quick, Jonathan Horatio 奎克,乔纳森·霍雷肖
 Quinary number system 五进制数系统
 Quolynomial chains 四项链
 Quotient: $\lfloor u/v \rfloor$ 商,见 Division
 of polynomials 多项式的商
 partial 部分商
 trial 试除的商
 Rabin, Michael Oser 拉宾,迈克尔·奥泽
 Rabinowitz, Philip 拉宾诺维茨,菲利普
 Rademacher, Hans 拉德梅歇尔,汉斯
 Radioactive decay 放射性衰变
 Radix: Base of positional notation 进制;定位记法的基
 complex 复数进制
 irrational 无理进制
 mixed 混合进制
 negative 负进制
 Radix conversion 进制转换
 floating point 浮点进制转换
 multiprecision 多精度进制转换
 Radix point 小数点
 Rami, Ralph Alexis 雷米,拉尔夫·亚历克西斯

- Raleigh, Walter 雷利, 沃尔特
 Rall, Louis Baker 拉尔, 路易斯·贝克
 Ramage, John Gerow 拉梅奇·约翰·杰罗
 Ramanujan Iyengar, Srinivasa 拉马努扬·艾扬加尔, 斯里尼瓦沙
 Ramaswami, Vammi 拉马斯瓦米, 范米
 Ramshaw, Lyle Harold 拉姆肖, 莱尔·哈罗德
 ran-array ran array 子程序
 RAND Corporation 兰德公司
 Randell, Brian 兰德尔, 布赖恩
 Random bits 随机位
 Random combinations 随机组合
 Random directions 随机方向
 Random fractions 随机分数, 见 Uniform deviates
 Random functions 随机函数
 Random integers 随机整数
 among all positive integers 正整数上的随机整数
 in a bounded set 有界集合中的随机整数
 Random mappings 随机映像
 Random number generators 随机数生成器(程序)
 for nonuniform deviates 非一致偏离随机数生成程序
 for uniform deviates 一致偏离随机数生成程序
 machines 随机数生成器
 summary 随机数生成程序小结
 tables 随机数表
 testing 随机数检验
 using 使用随机数, 也见 Randomized algorithms
 Random permutations 随机排列
 of a random combination 随机组合的排列
 Random point, in a circle 一个圆内的随机点
 in a sphere 一个球内的随机点
 on an ellipsoid 一个椭球内的随机点
 on a sphere 一个球面上的随机点
 Random polynomials 随机多项式
 Random random number generators 随机的随机数生成程序
 Random real numbers 随机实数
 Random samples 随机抽样
 Random sequences, meaning of 随机序列的意义
 finite 有限随机序列
 Randomized algorithms 随机化算法: 使用随机数且通常产生正确答案的算法
 Randomness, guaranteed 保证的随机性
 RANDU RANDU 随机数生成器
 Rangan 兰根, 见 Pandu Rangan
 Range arithmetic 范围算术
 Rank, of apparition 出现的秩
 of a matrix 矩阵的秩
 of a tensor 张量的秩
 RANLUX RANLOX 随机数生成器
 Rap music 说唱乐
 Rapoport, Anatol 拉波波特, 阿纳托尔
 Ratio method 比例方法
 Rational arithmetic 有理算术
 Rational function approximation 有理函数逼近
 Rational functions 有理函数
 approximation and interpolation 有理函数逼近和
 内插
 Rational numbers 有理数
 approximation by 用有理数逼近
 mod m 模 m 有理数
 polynomials over 有理数上的多项式
 positional representation of 有理数的定位表示
 Rational reconstruction 有理重构
 Real numbers 实数
 Real time 实时
 Realization of a tensor 张量的实现
 Reciprocal differences 倒数差
 Reciprocals 倒数
 floating point 浮点倒数
 mod 2^k 模 2^k 倒数
 mod m 模 m 倒数
 power series 幂级数倒数
 Reciprocity laws 互反律
 Records, Robert 雷科德, 罗伯特
 Rectangle-wedge-tail method 矩形-楔形-尾形方法
 Rectangular distribution 矩阵分布, 见 Uniform distribution
 Recurrence relations 递推关系
 Recursive processes 递归过程
 Reeds, James Alexander 里德斯, 詹姆斯·亚历山大
 Rees, David 里斯, 戴维
 Registers 寄存器
 Regular continued fractions 正则连分数
 Reiser, John Fredrick 赖泽, 约翰·弗雷德里克
 Reitwiesner, George Walter 赖特维斯纳, 乔治·沃尔特
 Rejection method 拒绝方法
 Relative error 相对误差
 Relatively prime 互素: 没有公共素因子
 polynomials 互素多项式

- Remainder 余数, 余式, 剩余: 被除数减去商与除数的积, 也见 mod
- Replicative law 重复律
- Representation of numbers 数的表示, 见 Number systems
- Representation of trees 树的表示
- Representation of ∞ ∞ 的表示
- Reservoir sampling “水库”抽样, 储蓄器抽样
- Residue arithmetic 剩余算术
- Result set 结果集
- Resultant of polynomials 多项式的结式
- Revah, Ludmila 雷维, 勒德米拉
- Reverse of a polynomial 多项式的反演
- Reversing binary number system 反序二进制数系统
- Reversion of power series 幂级数的反演
- Revolving binary number system 旋转的二进制数系统
- Rezucha, Ivan 雷朱卡, 伊凡
- Rhind papyrus 莱因德文稿
- Rho method for factoring 因子分解的罗方法
- Riccati, Jacopo Francesco, equation 里卡蒂, 雅各波·弗朗西斯科方程
- Rieger, Georg Johann 里格, 乔治·约翰
- Riemann, Georg Friedrich Bernhard 黎曼, 乔治·弗雷德里克·伯恩哈德
- hypothesis 黎曼假说
- hypothesis, generalized 广义黎曼假说
- integration 黎曼积分
- Riffle shuffles 里弗尔洗牌法
- Ring with identity, commutative 可交换的幺元的环
- Riordan, John 赖尔登, 约翰
- Rising powers 上升幂
- Ritzmann, Peter 里茨曼, 彼得
- Rivat, Joël 里瓦特, 乔尔
- Rivest, Ronald Linn 里夫斯特, 罗纳德·林
- Robber 罗珀
- Robinson, Donald Wilford 鲁宾逊, 唐纳德·威尔福德
- Robinson, Julia Bowman 鲁宾逊, 朱莉娅·鲍曼
- Robinson, Raphael Mitchel 鲁宾逊, 拉斐尔·米切尔
- Roepstorff, Gert 罗普斯托夫·格特
- Rolletschek, Heinrich Franz 罗赖茨切克, 海因里希·弗朗兹
- Roman numerals 罗马数字
- Romani, Francesco 罗马尼, 弗朗西斯科
- Roof, Raymond Bradley 鲁夫, 雷蒙德·布雷德利
- Roots of a polynomial 多项式的根
- multivariate 多变量多项式的根
- Roots of unity 单位根, 见 Cyclotomic polynomials, Exponential sums
- Ross, Douglas Taylor 罗斯, 道格拉斯·泰勒
- Rotenberg, Aubey 罗登贝格, 奥比
- Rothe, Heinrich August 罗思, 海因里希·奥古斯特
- Rouché, Eugène, theorem 鲁彻, 尤金定理
- Roulette 轮盘赌
- Round to even 舍入成偶数
- Round to odd 舍入成奇数
- Rounding 舍入
- mediant 中间舍入
- Rounding errors 舍入误差
- Rounding overflow 舍入上溢
- Rozier, Charles P. 罗齐尔, 查尔斯·P
- RSA box RSA 箱
- RSA encryption RSA 密码
- Rudolf, Christof 鲁道夫·克里斯托弗
- Rumely, Robert Scott 鲁姆利, 罗伯特·斯科特
- Run test 运行检验
- Runs above (or below) the mean 在均值之上(或之下)运行
- Runs in a permutation 排列中的运行
- Russian peasant method 俄罗斯农民方法
- Ruzsa, Imre Zoltán 鲁齐沙, 伊姆里·佐尔坦
- Ryser, Herbert John 赖施尔, 赫伯特·约翰
- S_N N 源
- Saariinen, Jukka Pentti Päiviö 萨里南, 朱卡·彭蒂·佩维欧
- Sachau, Karl Eduard 萨考, 卡尔·埃杜瓦德
- Saddle point method 鞍点方法
- Sahni, Sartaj Kumar 萨尼, 萨尔塔·库默
- Saiden, Ahmad Salim 塞旦, 阿马德·萨利姆
- Salamin, Eugene 萨拉明, 尤金
- Salfi, Robert 萨尔菲, 罗伯特
- Samelson, Klaus 萨梅尔森, 克劳斯
- Samet, Paul Alexander 萨梅特, 保罗·亚历山大
- Sampling (without replacement) 抽样(无替换)
- Sands, Arthur David 桑兹, 阿瑟·戴维
- Saunders, Benjamin David 桑德斯, 本杰明·戴维
- Savage, John Edmund 萨维奇, 约翰·埃德蒙德
- Sawtooth function $((x))$ 锯齿函数
- Saxe, James Benjamin 萨克斯, 詹姆斯·本杰明
- Scarborough, James Blaine 斯卡巴勒, 詹姆斯·布莱恩
- Schatte, Peter 沙特, 彼得
- Schelling, Hermann von 谢林, 赫尔曼·冯

- Schmid, Larry Philip 施米德, 拉利·菲利普
 Schmidt, Erhard 施米特, 埃哈德
 Schmidt, Wolfgang M 施米特, 沃尔夫冈·M
 Schnorr, Claus-Peter 施罗尔, 克劳斯-彼得
 Scholz, Arnold 肖尔茨, 阿诺德
 Scholz-Brauer conjecture 肖尔茨-布劳尔猜想
 Schönemann, Theodor 舍尼曼, 西奥多
 Schönhage, Arnold 舍恩哈格, 阿诺德
 Schönhage-Strassen algorithm 舍恩哈格-斯特拉森算法
 Schooling, William 斯库林, 威廉
 Schreyer, Helmut 施赖伊尔, 赫尔穆特
 Schröder, Friedrich Wilhelm Karl Ernst 施罗德, 弗里德里克·威廉·卡尔·厄恩斯特
 function 施罗德函数
 Schroepfel, Richard Crabtree 施罗皮尔, 理查德·克雷布特里
 Schubert, Friedrich Theodor von 舒伯特, 弗里德里克·西奥多
 Schwartz, Jacob Theodore 施瓦茨, 雅各布·西奥多
 Schwarz = (Švarc), Stefan 施瓦茨, 斯蒂芬
 Schwenter, Daniel 施文特, 丹尼尔
 Secrest, Don 西克雷斯特, 唐
 Secret keys 密钥
 Secure communications 安全通信
 Sedgewick, Robert 塞奇威克, 罗伯特
 Seed (starting value) in a linear congruential sequence 一个线性同余序列中的种子(开始值)
 Seidenberg, Abraham 塞登伯格, 亚伯拉罕
 Selection sampling 选择抽样
 Selenius, Clas-Olof 塞莱尼厄斯, 克拉斯-奥洛夫
 Self-reproducing numbers 自再生数
 Selfridge, John Lewis 塞尔弗里奇, 约翰·刘易斯
 Semi-online algorithm 半在线算法
 Semigroup 半群
 Seneschal, David 塞内斯科尔, 戴维
 Septenary (radix 7) number system 七进制数系统
 Serial correlation coefficient 序列相关系数
 Serial correlation test 序列相关检验
 Serial test 序列检验
 Seroussi, Gadiel 塞路斯, 加迭尔
 Serret, Joseph Alfred 塞雷特, 约瑟夫·阿尔弗雷德
 Sethi, Ravi 塞思, 拉维
 SETUN computer SETUN 计算机
 Sexagesimal number system 六十进制数系统
 Seysen, Martin 塞森, 马丁
 Shakespeare (= Shakspeare), William 莎士比亚, 威廉
 Shallit, Jeffrey Outlaw 沙利特, 杰弗里·奥特洛
 Shamir, Adi 沙米尔, 阿迪
 Shand, Mark Alexander 香德, 马克·亚历山大
 Shanks, Daniel Charles 香克斯, 丹尼尔·查尔斯
 Shanks, William 香克斯, 威廉
 Shannon, Claude Elwood, Jr. 小香龙, 克劳德·埃尔伍德
 Shaw, Mary Margaret 肖, 玛丽·玛格丽特
 Shen, Kangshen 沈康身
 Sheriff 行政司法长官, 警长
 Shubata, Akihiko 柴田昭彦
 Shift operators of MIX MIX 的移位操作符
 Shift register recurrences 移位寄存器递推
 Shift-symmetric N -source 移位对称 N 源
 Shirley, John William 雪莉, 约翰·威廉
 Shokrollahi, Mohammad Amin 肖克罗拉西, 穆哈默德·阿米恩
 Short vectors 短向量
 Shoup, Victor John 肖普, 维克多·约翰
 Shub, Michael Ira 舒布, 迈克尔·艾拉
 Shuffled digits 混洗后的数字
 Shuffling a sequence 混洗一个序列
 Shuffling cards 洗牌
 Shukla, Kripa Shankar 舒克拉, 克里帕·香克尔
 Sibuya, Masaaki 涉谷政昭
 SICOMP; SIAM Journal on Computing, published by the Society for Industrial and Applied Mathematics since 1972 《SIAM 计算杂志》; 工业与应用数学学会 1972 年创立的杂志
 Sideways addition 侧向加法
 Sierpiński, Waclaw 西彭斯基, 瓦克劳
 Sieve methods 筛方法
 Sieve of Eratosthenes 伊拉托斯尼斯筛
 Sieveking, Malte 西夫金, 马尔蒂
 Signatures, digital 数字签名
 Signed magnitude representation 带符号量的表示
 Significant digits 有效数字
 Sikdar, Kripasindhu 西格达, 克里帕辛杜
 Silverman, Joseph Hillel 西尔弗曼, 约瑟夫·希勒尔
 Simplex, recursively subdivided 递归再分的单纯形
 Simulation 模拟
 Sinclair, Alistair 辛克莱, 阿利斯泰尔
 Sine 正弦
 Singh, Avadhesh Narayan 辛格, 阿瓦德斯·纳拉燕
 Sink vertex 下沉顶点

- SKRZAT 1 computer SKRZAT 1 计算机
- Slash arithmetic 开缝算术
- SLB(shift left rAX binary) 左移寄存器 AX 指令
- Slide rule 计算尺
- Sloane, Neil James Alexander 斯隆, 尼尔·詹姆斯·亚历山大
- Slowinski, David Allen 斯洛文斯基, 戴维·艾伦
- Small step 小步
- Smirnov, Nikolai Vasilievich (Смирнов, Николай Васильевич) 斯米尔诺夫, 尼古拉伊·瓦西里耶维奇
- Smith, David Eugene 史密斯, 戴维·尤金
- Smith, David Michael 史密斯, 戴维·迈克尔
- Smith, Henry John Stephen 史密斯, 亨利·约翰·斯蒂芬
- Smith, James Everett Keith 史密斯, 詹姆斯·埃弗雷特·基思
- Smith, Robert Leroy 史密斯, 罗伯特·勒鲁瓦
- Sobol, Ilya Meerovich (Соболев, Илья Меерович) 索伯尔, 伊利亚·米罗维茨
- SODA: Proceedings of the ACM-SIAM Symposia on Discrete Algorithms, inaugurated in 1990 《ACM-SIAM 离散算术会议论文集》
- Soden, Walter 索登, 沃尔特
- Solitaire 单人纸牌游戏
- Solomonoff, Ray Joseph 所罗莫诺夫, 雷·约瑟夫
- Solvay, Robert Martin 索洛韦, 罗伯特·马丁
- Sorenson, Jonathan Paul 索伦森, 乔纳森·保罗
- Sorted uniform deviates 排好序的一致偏离
- Source vertex 源顶点
- Sowey, Eric Richard 索韦, 埃里克·理查德
- Space-filling curves 间隔填充曲线
- Spacings 间隔
- Sparse polynomials 稀疏多项式
- Specht, Wilhelm 施佩希特, 威廉
- Species of measure zero 测度为 0 的种类
- Spectral test 谱检验
algorithm for 谱检验的算法
examples 谱检验举例
generalized 广义谱检验
- Spence, Gordon McDonald 斯彭斯, 戈登·麦克唐纳
- Spencer Brown, David John 斯潘塞·布朗, 戴维·约翰
- Sphere, n -dimensional n 维球
random point in n 维球内的随机点
random point on n 维球面上的随机点
volume of n 维球的体积
- Spherical coordinates 球坐标
- SQRT box SQRT 盒
- Square root 平方根
modulo m 模 m 平方根
modulo p 模 p 平方根
of power series 幂级数的平方根
of uniform deviate 一致偏离的平方根
- Squarefree factorization 无平方因子分解
- Squarefree polynomials 无平方多项式
- Squeeze method 挤压方法
- SRB (shift right rAX binary) rAX 二进制右移指令
- Stability of polynomial evaluation 多项式求值的稳定性
- Stack 栈: 具有后进先出增长模式的线性表
- Stahnke, Wayne Lee 斯坦克, 韦恩·李
- Standard deviation, evaluation of 标准差的计算
- Stanley, Richard Peter 斯坦利, 理查德·彼得
- Star chains 星链
- Star step 星步
- Stark, Richard Harlan 斯塔克, 理查德·哈兰
- Starting value in a linear congruential sequence 线性同余序列的初始值
- Statistical tests 统计检验, 见 Testing
- Steele, Guy Lewis, Jr 小斯蒂尔, 盖伊·刘易斯
- Steffensen, Johan Frederik 斯蒂芬森, 约翰·弗雷德里克
- stegun, Inene Anne 斯蒂冈, 艾琳·安妮
- Stein, Josef 斯坦, 约瑟夫
- Stein, Marvin Leonard 斯坦, 马文·伦纳德
- Stern, Moritz Abraham 斯特恩, 莫里兹·亚伯拉罕
- Stern-Brocot tree 斯特恩-布罗科特树
- Stevin, Simon 史蒂文, 西蒙
- Stibitz, George Roberto 斯蒂比兹, 乔治·罗伯特
- Stillington, Edward 斯蒂林弗利特, 爱德华
- Stirling, James 斯特林, 詹姆斯
approximation 斯特林近似
numbers 斯特林数
- STOC: Proceedings of the ACM Symposia on Theory of Computing, inaugurated in 1969 《ACM 计算理论会议论文集》
- Stockmeyer, Larry Joseph 斯托克迈耶, 拉里·约瑟夫
- Stoneham, Richard George 斯托纳姆, 理查德·乔治
- Stoppard, Tom (= Straussler, Tomas) 斯托帕德, 汤姆 (= 斯特劳斯拉勒, 托马斯)
- Storage modification machines 存储修改机器
- Strachey, Christopher 斯特雷奇, 克里斯托弗

- Straight-line program 直线程序
 Strassen, Volker 斯特拉森, 沃尔克
 Straus, Ernst Gabor 施特劳斯, 厄恩斯特·加波尔
 String polynomials 串多项式
 Stringent tests 斯特林金特检验
 Stroud, Arthur Howard 斯特劳德, 阿瑟·霍华德
 Struve, Wassilj Wassiliewitsch (Струве, Василий Васильевич) 斯特拉弗, 瓦西利·瓦西列维奇
 Sturm, Jacob Karl Franz 斯特姆, 雅各布·卡尔·弗朗兹
 Subbarao, Mathukumalli Venkata 萨巴劳, 马图古马里·文卡塔
 Subexponential (nice) functions 好函数
 Subresultant algorithm 子结式算法
 Subsequence rules 子序列规则
 Subsequence tests 子序列检验
 Subsequences 子序列
 Subset FORTRAN language 子集 FORTRAN 语言
 Subtract-and-shift cycle 减和移位循环
 Subtract-with-borrow sequence 带借位减序列
 Subtraction 减法
 complex 复数减法
 continued fractions 连分数减法
 double-precision 双精度减法
 floating point 浮点减法
 fractions 分数减法
 mod m 模 m 减法
 modular 模数减法
 multiprecision 多精度减法
 polynomial 多项式减法
 power series 幂级数减法
 Subtractive random number generator 减法随机数生成程序
 Sugunamma, Mantri 萨古南姆马, 曼特里
 Sukhatme, Pandurang Vasudeo 苏克哈特米, 潘都兰·瓦苏迪奥
 Sum of periodic sequences, mod m 模 m 周期序列的和
 Summation by parts 分部求和
 Sun Tsu (= Sünzi, Master Sun) 孙子
 Sun SPARCstation Sun SPARC 工作站
 Suokonautio, Vilho 索科瑙蒂奥, 维尔霍
 Svoboda, Antonin 斯沃博达, 安托宁
 Swartrauber, Paul Noble 斯沃茨特劳博, 保罗·诺布尔
 Swedenborg, Emanuel 斯维登博格, 伊曼纽尔
 Sweeney, Dura Warren 斯威尼, 杜拉·沃伦
 Swinnerton-Dyer, Henry Peter Francis 斯温纳顿-戴尔, 亨利·彼得·弗朗西斯
 Sýkora, Ondrej 西科拉, 翁德列
 Sylvester, James Joseph, matrix 西尔威斯特, 詹姆斯·约瑟夫矩阵
 Szabó, József 萨博, 约瑟夫
 Szabó, Nicholas Sigismund 萨博, 尼古拉斯·西格斯蒙德
 Szekeres, George 塞克里斯, 乔治
 Szymanski, Thomas Gregory 西曼斯基, 托马斯·格里戈里
 t -ary trees t 叉树
 Tabari, Mohammed ben Ayyūb 塔巴里, 穆哈默德·本·艾尤比
 Tables of fundamental constants 基本常数表
 Tabulating polynomial values 表格多项式值
 Tague, Berkley Arnold 塔格, 伯克利·阿诺德
 Tail of a floating point number 浮点数的尾部
 Tail of the binomial distribution 二项式分部的尾部
 Tail of the normal distribution 正态分布的尾部
 Takahashi, Daisuke 高桥大介
 Takahasi, Hidetosi 高桥秀俊
 Tamura, Yoshiaki 田村良明
 Tanaka, Richard Isamu 田中リチャード勇
 Tangent 正切
 tanh 双曲正切
 Tannery, Jules 坦尼里, 朱尔斯
 Taranto, Donald Howard 塔兰托, 唐纳德·霍华德
 Tarski, Alfred 塔斯基, 艾尔弗雷德
 Tate, John Torrence, Jr. 小塔特, 约翰·托伦斯
 Tate, Stephen Ralph 塔特, 斯蒂芬·拉尔夫
 Tausky Todd, Olga 陶斯基·托德, 奥尔加
 Tausworthe, Robert Clem 陶斯沃思, 罗伯特·克莱姆
 Taylor, Alfred Eower 泰勒, 艾尔弗雷德·鲍尔
 Taylor, Brook, theorem 泰勒, 布鲁克定理
 Taylor, William Johnson 泰勒, 威廉·约翰逊
 Television script 电视脚本
 Ten's complement notation 十的补码记法
 Tensors 张量
 Term 项; 被加的量
 Terminating fractions 有穷小数
 Ternary number system 三进制数系统
 balanced 平衡的三进制数系统
 Testing for randomness 随机性的检验
 a priori tests 随机性检验的先验检验

- chi-square test χ^2 检验
collision test 冲突检验
coupon collector's test 集券检验
empirical tests 经验检验
equidistribution test 等分布检验
frequency test 频率检验
gap test 间隔检验
Kolmogorov-Smirnov test 科尔莫戈罗夫-斯米尔诺夫大检验
maximum-of- t test t 的极大值检验
partition test 分划检验
permutation test 排列检验
run test 运行检验
serial correlation test 序列相关检验
serial test 序列检验
spectral test 谱检验
subsequence tests 子序列检验
theoretical tests 理论检验
torture test 折磨检验
TEX TEX 排版系统
Tozuka, Shu 手塚集
Thacher, Henry Clarke, Jr. 小撒切尔, 亨利·克拉克
Theoretical tests for randomness 随机性的理论检验
Thiele, Thorvald Nicolai 蒂利, 索瓦尔德·尼古拉
Thompson, John Eric Sidney 汤普森, 约翰·埃里克·西德尼
Thomson, William Ettrick 汤姆森, 威廉·埃特里克
Thurber, Edward Gerrish 瑟伯, 爱德华·格里什
Tichy, Robert Franz 蒂奇, 罗伯特·弗朗茨
Tienari, Martti Johannes 蒂纳里, 马蒂·约翰尼斯
Tingey, Fred Hollis 廷吉, 弗雷德·霍利斯
Tippett, Leonard Henry Caleb 蒂皮特, 伦纳德·亨利·凯莱布
Tiware, Prasoon 蒂瓦里, 普拉苏恩
Tobey, Robert George 托比, 罗伯特·乔治
Tocher, Keith Douglas 托切尔, 基斯·道格拉斯
Todd, John 托德, 约翰
Todd, Olga Taussky 托德, 奥尔加·陶斯基
Toeplitz, Otto 托耶普利兹, 奥托
Tonal System 音调系统
Tonelli, Alberto 托内利, 艾伯托
Toolkit philosophy 工具箱原理
Toom, Andrei Leonovich (Тоом, Андрей Леонович) 图姆, 安德雷·利奥诺维奇
Toom-Cook algorithm 图姆-库克算法
Topological sorting 拓扑排序
Topuzoglu, Alev 托普佐格鲁, 艾利弗
Torrelli, Gabriele 托雷利, 加布里埃尔
Torresy Quevedo, Leonardo de 托里斯-奎韦多, 伦纳德·德·德
Torture test 折磨检验
Trabb Pardo, Luis Isidoro 特拉布·帕多, 卢斯·伊西多洛
Trace of a field element 一个域元素的迹
Trager, Barry Marshal 特拉格尔, 巴里·马歇尔
Trailing digit 尾部数字
Transcendental numbers 超越数
Transitive permutation groups 传递排列组
Transpose of a tensor 张量的转置
Transpositions 转置
Taub, Joseph Frederick 特劳布, 约瑟夫·弗雷德里克
Trees 树: 分支信息结构
 binary 二叉树
 complete binary 完备二叉树
 enumeration of 树的枚举
 oriented 有向树
 t -ary t 叉树
Trevisan, Vilmar 特里维桑, 维尔默
Trial quotients 试验的商
Triangularization of matrices 矩阵的三角化
Tries 检索结构
Trigonometric functions 三角函数
Trilinear representation of tensors 张量的三线性表示
Triple-precision floating point 三精度浮点
Trits 特里兹
Truncation 截取; 去掉末尾数字
Tsang, Wai Wan 曾卫寰
Tsu Ch'ung-Chih (= Zü Chongzhī) 祖冲之
Tsuji, Masatsugu 辻正次
Tukey, John Wilder 图基, 约翰·怀尔德
Turán, Paul 图兰, 保罗
Turing, Alan Mathison 图灵, 艾伦·马西森
 machines 图灵机
Twindragon fractal 双龙碎片
Two squares, sum of 平方和
Two's complement notation 2 的补码记法
Twos' complement notation 2 的补码记法
Tydeman, Fredrick John 泰德曼, 弗雷德里克·约翰
Ulam, Stanislaw Mercin 乌兰姆, 斯坦尼斯劳·马辛
Ullman, Jeffrey David 乌尔曼, 杰弗里·戴维
Ulrich, Christian 乌尔里克, 克里斯琴

- Ulp 最后一位的 1
- Underflow, exponent 指数的下溢
gradual 逐渐下溢
- Ungar, Peter 昂加尔, 彼得
- Uniform deviates 一致偏离: 一致分布的随机数
generating 生成一致偏离
logarithm of 一致偏离的对数
sorted 排序的一致偏离
square root of 一致偏离的平方根
- Uniform distribution 一致分布
- Unimodular matrix 单模矩阵
- Unique factorization domain 惟一因子分解整环
- Units in a unique factorization domain 惟一因子分解整环上的单位
- Unity: The number one 单位: 数 1
roots of 单位根, 也见 Cyclotomic polynomials, Exponential sums
- Unlimited precision 无限精度, 也见 Multiple-precision
- Unnormalized floating point arithmetic 未规格化浮点算术
- Unusual correspondence 非常相关
- Useful primes 有用的素数
- Uspensky, James Victor 乌斯宾斯基, 詹姆斯·维克托
- Vahlen, Karl Theodor 瓦伦, 卡尔·西奥多
- Valach, Miroslav 瓦拉克, 米罗斯拉夫
- Valiant, Leslie Gabriel 瓦利昂特, 莱斯利·加布里埃尔
- Vallée, Brigitte 瓦利, 布里吉特
- Vallée Poussin, Charles-Jean-Gustave Nicolas de la 瓦利·波辛, 查尔斯-琼·古斯塔瓦-尼古拉斯·德拉
- Valtat, Raymond 瓦尔塔特, 雷蒙德
- van Ceulen, Ludolph 范修林, 卢多尔夫
- van de Wiele, Jean-Paul 范德威勒, 琼·保罗
- van der Corput, Johannes Gualtherus 范德科普特, 约翰尼斯·高尔锡拉斯
- van der Poorten, Alfred Jacobus 范德普尔顿, 阿尔弗雷德·雅各布斯
- van der Waerden, Bartel Leendert 范德瓦尔登, 巴特·利恩德特
- van Halewyn, Christopher Neil 范哈勒维, 克里斯托夫·内尔
- van Leeuwen, Jan 范利夫文, 简
- Van Loan, Charles Francis 范劳恩, 查尔斯·弗朗西斯
- van Wijngaarden, Adriaan 范维恩加登, 艾德里安
- Vari, Thomas Michael 瓦里, 托马斯·迈克尔
- Variables 变量
- Variance, unbiased estimate of 方差的无偏倚估计
- Variance-ratio distribution 方差比分布
- Vattulainen, Ilpo Tapio 瓦图莱伦, 伊尔波·塔皮奥
- Vaughan, Robert Charles 沃恩, 罗伯特·查尔斯
- Velthuis, Frans Jozef 维尔修斯, 弗朗兹·约瑟夫
- Veltkamp, Gerhard Willem 维尔特卡姆, 格哈特·威廉
- Vershik, Anatoly Moiseevich (Вершик, Анатолий Моисеевич) 韦尔斯基, 阿纳托利·莫伊谢耶维奇
- Vertex cover 顶点覆盖
- Vetter, Herbert Dieter Ekkhart 维特, 赫伯特·迪特·埃克哈特
- Viète, Francois 维特, 弗朗科伊斯
- Ville, Jean 维利, 琼
- Vitanyi, Paul Michaël Béla 维坦尼, 保罗·迈克尔·贝拉
- Vitter, Jeffrey Scott 魏杰甫
- Vogel, Otto Hermann Kurt 沃格尔, 奥托·赫曼库尔特
- Voltaire, de (= Arouet, Francois Marie) 沃尔特, 德(= 阿罗特, 弗朗科伊斯·玛丽)
- Volume of sphere 球的体积
- von Fritz, Kurt 冯弗里茨, 库尔特
- von Mangoldt, Hans Carl Friedrich 冯曼戈尔特, 汉斯·卡尔·弗里德里克
- function 冯曼戈尔特函数
- von Mises, Richard, Edler 冯迈西斯, 理查德·埃德勒
- von Neumann, John (= Margittai Neumann János) 冯诺伊曼, 约翰(= 马古泰·诺伊曼·雅诺什)
- von Schelling, Hermann 冯谢林, 赫曼
- von Schubert, Friedrich Theodor 冯舒伯特, 弗里德里克·西奥多
- von zur Gathen, Joachim Paul Rudolf 冯朱尔盖森, 乔基姆·保罗·鲁道夫
- Vowels, Robin Anthony 沃韦尔斯, 罗宾·安东尼
- Vuillemin, Jean Etienne 维勒明, 琼·埃蒂恩
- Wadel, Louis Burnett 韦德尔, 路易斯·伯内特
- Wadey, Water Geoffrey 韦迪, 沃尔特·杰弗里
- Waerden, Bartel Leendert van der 瓦尔登, 巴特·利恩德特·范·德
- Waiting time 等候时间
- Wakulicz, Andrzej 韦古里茨, 安德烈

- Wald, Abraham 沃尔德, 亚伯拉罕
sequence 沃尔德序列
- Wales, Francis Herbert 韦尔斯, 弗朗西斯·赫伯特
- Walfisz, Arnold 沃尔费茨, 阿诺德
- Walker, Alastair John 沃克, 阿拉斯泰尔·约翰
- Wall, Donald Dines 沃尔, 唐纳德·丹斯
- Wall, Hubert Stanley 沃尔, 休伯特·斯坦利
- Wallace, Christopher Stewart 华莱士, 克里斯托弗·斯图尔特
- Wallis, John 沃利斯·约翰
- Walsh, Joseph Leonard 沃尔什, 约瑟夫·伦纳德
transform 沃尔什变换
- Wang, Paul Shyh-Horng 王士弘
- Ward, Morgan 沃德, 摩根
- Waring, Edward 韦林, 爱德华
- Warlimont, Richard Clemens 沃利蒙特, 理查德·克茨门斯
- Watanabe, Masatoshi 渡边雅俊
- Waterman, Alan Gaisford 沃特曼, 艾伦·盖斯福德
- Weather 天气
- Wedge-shaped distributions 楔形分布
- Weigel, Erhard 韦格尔, 埃哈德
- Weighing problem 称重问题
- Weights and measures 度量衡
- Weinberger, Peter Jay 温伯格, 彼得·杰伊
- Welch, Peter Dunbar 韦尔奇, 彼得·邓巴
- Welford, B. P. 韦尔福德, B. P.
- Weyl, Claus Hugo Hermann 韦尔, 克劳斯·雨果·赫尔曼
- Wheeler, David John 惠勒, 戴维·约翰
- White, Jon L. 怀特, 乔恩·L.
- White sequence 白序列(可靠的序列)
- Whiteside, Derek Thomas 怀特赛德, 德里克·托马斯
- Whitworth, William Allen 惠特沃思, 威廉·艾伦
- Wichmann, Brian Anderson 威奇曼, 布赖恩·安德森
- Wiedijk, Frederik 威迪克, 弗雷德里克
- Wiele, Jean-Paul van de 威尔, 琼-保罗·范·德
- Wijngaarden, Adriaan van 维恩加登, 安德里安·范
- Wilf, Herbert Saul 威尔弗, 赫伯特·索尔
- Wilkes, Maurice Vincent 威尔克斯, 莫里斯·文森特
- Wilkinson, James Hardy 威尔金森, 詹姆斯·哈迪
- Williams, Hugh Cowie 威廉斯·休·考伊
- Williams, John Hayden 威廉斯, 约翰·海登
- Williamson, Dorothy 威廉森, 多萝西
- Wilson, Edwin Bidwell 威尔森, 埃德温·比德韦尔
- Winograd, Shmuel 维诺格拉德, 斯米尔
- Wirsing, Eduard 威欣, 爱杜瓦德
- WMI (word size minus one) 字长减1指令
- Wolf, Thomas Howard 沃尔夫, 托马斯·霍华德
- Wolff von Gudenberg, Jürgen Freiherr 沃尔夫·冯·古登伯格, 朱尔金·弗莱赫尔
- Wollowitz, Jacob 沃尔福威茨, 雅各布
- Woltman, George Frederick 沃尔特曼, 乔治·弗雷德里克
- Wood, William Wayne 伍德, 威廉·韦恩
- Word length 字长: 字大小的对数
- Word size 字的大小
- Wrench, John William, Jr. 小伦奇, 约翰·威廉
- Wright, Edward Maitland 赖特, 爱德华·梅特兰
- Wunderlich, Charles Marvin 旺德利奇, 查尔斯·马文
- Wynn, Peter 温, 彼得
- Wynn-Williams, Charles Eryl 温-威廉斯, 查尔斯·埃里尔
- Xie, Shenquan 谢深泉
- XOR (exclusive or) 异或指令
- Yagati 亚加蒂, 见 Lakshman
- Yaglom, Akiva Moiseevich (Яглом, Акива Моисеевич) 亚格洛姆, 阿基维·莫伊谢耶维奇
- Yaglom, Isaak Moiseevich (Яглом, Исаак Моисеевич) 亚格洛姆, 艾萨克·莫伊谢耶维奇
- Yao, Andrew Chi-Chih 姚期智
- Yao, Frances Foong Chu 姚储枫
- Yates, Frank 耶茨, 弗朗克
- Yohe, James Michael 约埃, 詹姆斯·迈克尔
- Young, Jeffery Stagg 扬, 杰弗里·斯塔格
- Younis, Saed Ghalib 汤尼斯, 萨德·格利布
- Yuditsky, Davit Islam Gireevich (Юдицкий, давит Ислам Тиреевич) 尤季茨基, 达维·伊斯拉姆·季里维奇
- Yun, David Yuan-Yue 恽元一
- Yuriev, Sergei Petrovich (Юрьев, Сергей Петрович) 尤里耶夫, 谢尔盖·彼得罗维奇
- Z-independent vectors Z 独立向量
- Zacher, Hans-Joachim 札策尔, 汉斯·乔基姆
- Zaman, Arif 札曼, 艾里夫
- Zantema, Hantsje 赞特马, 汉兹杰
- Zaremba, Stanislaw Krystyn 札仁姆巴, 斯坦尼斯劳·克里斯廷
- Zaring, Wilson Miles 札林, 威尔逊·迈尔斯
- Zassenhaus, Hans Julius 札森豪斯, 汉斯·朱利叶斯
- Zeilberger, Doron 蔡伯格, 多罗恩

Zero 零	Zhang, Linbo 张林波
leading 前导零	Zierler, Neal 齐勒, 尼尔
minus 负零	Zippel, Richard Eliot 齐佩尔, 理查德·埃利奥特
order of magnitude 量零的阶	Zuckerman, Herbert Samuel 朱克曼, 赫伯特·塞缪尔
polynomial 零多项式	Zuse, Konrad 朱斯, 康拉德
Zero divisors 零除数	Zvonkin, Alexander Kalmanovich (Звонкин, Александр Калманович) 兹冯金, 亚历山大·卡尔马诺维奇
Zeta function 仄塔函数	

本书的写作是在安装了 Computer Modern 字体的一台 Sun SPARCstation 上完成的, 使用了 TEX 和 META-FONT 软件[作者的 *Computers & Typesetting* (Reading, Mass.: Addison-Wesley, 1986), Volumes A-E 中对此作了介绍]。插图用 John Hobby 的 METAPOST 系统制作。索引中的一些名词用到了其它一些字体, 它们是由 Yannis Haralambous (Greek, Hebrew, Arabic), Olga G. Lapko (Cyrillic), Frans J. Velthuis (Devanagari), Masatoshi Watanabe (Japanese), 和 Linbo Zhang (Chinese) 开发的。——原注