世界著名计算机教材精选

PEARSON Prentice Hall

算法基础

Gilles Brassard
Paul Bratley 著

邱仲潘 柯 渝 徐 锋 译

FUNDAMENTALS OF ALGORITHMICS

清华大学出版社

Pearson Education

世界著名计算机教材精选

算 法 基 础

Gilles Brassard
Paul Bratley 著

邱仲潘 柯 渝 徐 锋 译

清华大学出版社
北京

北京市版权局著作权合同登记号 图字：01-2003-1781

图书在版编目(CIP)数据

算法基础 (　)布拉萨德(Brassard, G.),(　)布拉特里(Bratley, P.)著；邱仲潘等译. 北京：清华大学出版社，2005.7
(世界著名计算机教材精选)
书名原文：Fundamentals of Algorithmics
ISBN 7-302-10609-6

Ⅰ. 算… Ⅱ. ① 布… ② 布… ③ 邱… Ⅲ. 电子计算机—算法理论—高等学校—教材 Ⅳ. TP301.6

中国版本图书馆 CIP 数据核字(2005)第 017659 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969
组稿编辑：龙啟铭
文稿编辑：王敏稚
印 刷 者：北京市清华园胶印厂
装 订 者：三河市金元装订厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：26.25　**字数**：676 千字
版　　次：2005 年 7 月第 1 版　2005 年 7 月第 1 次印刷
书　　号：ISBN 7-302-10609-6/TP・7195
印　　数：1～3000
定　　价：49.00 元

前　言

分析仪一经存在，它就必将引导科学未来的进程。无论何时，只要借助于它来寻找结果，就会引发一个问题——用什么样的计算过程可以让机器在最短的时间内得到结果？

——Charles Babbage，1864

1977 年 8 月，《科学美国人》(*Scientific American*)杂志给其读者出了一道难题：解密一段加密的消息，赢者将获得 100 美元的奖赏。这听起来就不可能：据预测，以当时最快的计算机，用当时已知的最高效的算法，即使一刻不停地计算也要花费几百万倍于宇宙年龄的时间才可能得到这个奖赏。然而，16 年后，仅需 8 个月时间的计算就足以完成这个任务。这些年中到底发生了什么事？运算能力的提高固然不可忽视，但更重要的是发现了解决问题的更高效的算法——详情请参阅 7.8 节、7.10 节和 10.7.4 小节。2.7 节和全书中也给出了一些补充的例子以说明高效算法的发展是如何扩展了可行性计算的能力。

早在电子计算机时代之前，人们就已经意识到高效算法的重要性。最有名的一个有关算法的例子是欧几里德(Euclid)的最大公约数算法。人们经历了用手工(笔和纸)也能进行算术运算的算法的发展，如乘法。许多数学家——如高斯——都曾为很多问题研究高效的算法。可能没有人比查尔斯·巴比奇(Charles Babbage)更具有预见性。他在 19 世纪成功制造出的分析仪可能是第一台(机械的)计算机。他杰出的先进思想，如在前言开始处所引用的他的话，在今天看来更具合理性：因为你的计算设备速度越快，你就会从高效算法中获得越大的利益。

这本书不是一本编程手册。书里至多给出一些在设计算法时可以说是含糊不清的基本原则。它也不是类似食谱那样提供了一长串程序目录，以便碰到某个特定问题时直接拿来使用。相反，它讲的是算法学：系统地研究算法的设计与分析。这本书的目的是提供给读者一些必需的基本工具，这样他们可以在任何应用领域使用这些工具来开发自己的算法。

我们关注在那些用于设计和分析高效算法的基本技术上，包括贪婪法(greedy algorithms)、分治法(divide-and-conquer)、动态规划(dynamic programming)、图技术(graph techniques)、概率算法(probabilistic algorithms)和并行算法(parallel algorithms)。每种技术会先给出比较广泛的概念，然后以一些实例作为讲解。这些实例都是从不同的应用领域中提取出来的，如最优化、线性代数、密码学、计算数论、图论、运筹学、人工智能等。我们特别注意算法的设计与性能分析的结合。尽管本书行文严肃精确，但也注意到实践者的需要：除了演示设计的技术外，书中大部分的算法都有现实的应用。

为了能够尽量从书中获益，你还应当需要有些编程经验。不过我们也没有使用某种特定的编程语言，也不会把例子局限于特定类型的机器。这样做可以使得书中的各种思想保持一致和关联。另一方面，也不可能直接使用书中给出的算法，而需要一定的努力把算法转

换为相应的编程语言。书中使用了 Pascal 或类似的结构化语言，这样也许可以帮助读者尽量减少转换的精力。

本书的宗旨是作为本科生算法课程的教材。书中提供了 500 多个问题，这样教师可以以此来布置课程作业。第 1 章包含了大部分必需的数学预备知识。此章还专门详细讨论了在计算机科学的本科教育中常常忽视的一个基本工具——数学归纳法。书中的某些段落时常需要更高级的数学知识，但是在第一次阅读的时候也可以跳过而不失连贯性。本书也可以用于自学：适用于任何需要写出更好、更高效的算法的人。

为了能够一开始就吸引学生的注意力，很有效的做法是在第一次讲课时，以讨论一个常见的问题的算法，如整数乘法，作为开始。James A. Foster 曾在 Idaho 大学（美国爱达荷州）使用本书的初版进行教学。他是这样描述他的经验的。“我的第一次算法课以讨论‘如何进行两个数相乘’作为开始，并以此引导向对传统乘法算法的分析。接着我演示了 à la russe 乘法算法，学生们很快理解了。然后我们讨论了分治法（7.1 节）。课讲得很通俗，但是在这次课结束时（只是一次课！），学生们都理解了创建和分析算法的基本原理。……以上就是我所说的方法，它确实起作用！”

如果希望在大约 45 个课时的本科课程中涵盖本书的所有内容是不太现实的。因此，在选择内容的时候，教师可以让学生们自学第 1 章，其中 1.7.4 小节可选可不选。第 2 章的大部分内容也可以自学，然而还是有必要在课堂上花一两个课时的时间进行总结。这样有助于涵盖一些概念的区别。比如问题和实例，平均和最坏情况分析，也有助于强调高效算法的重要性。第 3 章是渐近记法，虽然简短但是非常重要。如果希望从本书中获益，前 3 节必须完全理解。3.4 节中条件渐近记法的一些内容在以后会常常提到。第 4 章的前 5 节给出了算法分析的后面隐含的基本思想。内容虽然重要但是简单明了。4.6 节定义了分期分析的概念。这节内容可能比较深奥，在第一次阅读的时候可以跳过。4.7 节给出了解决递归问题的技巧。虽然长，这一节与许多算法的分析相互独立。第 5 章是预备知识的最后一章，它讲述了对于设计高效算法来说一些重要的基本数据结构。前 5 节涵盖了一些概念，如数组、列表和图，这些章节学生们都可以独立阅读。第 5 章的剩下部分讲述了如关联表（哈希），普通堆与二项堆以及不相交集结构等更复杂的数据结构。普通堆（5.7 节）和不相交集结构（5.9 节）在以后的章节中会用到几次，而关联表（5.6 节）和二项堆（5.8 节）则只有在必要时偶尔出现。

其他章节在内容上关联不大。每一章（除了第 12 章外）给出了一种设计高效算法的技巧。如果是算法基础课的话，就一定要讲第 6 章（贪婪法），第 7 章（分治法）和第 8 章（动态规划）。这些章节中的例子在讲课时不一定都要讲到，这就取决于教师的个人偏好了。许多学生可能会对 7.8 节很感兴趣，在这一节中展示了如何用先前所学的知识来产生一个加密的消息。第 9 章讲述图的搜索技巧，包括使用如回溯法和分支界限等技巧来搜索图。9.7 节和 9.8 节在介绍性的算法课程中不妨跳过。

最后 4 章是算法更高级的内容。对本科生的第一次算法课来说，不可能一下子把它们都讲得很详细。但是如果不在讲课时提及每一章中的一些内容又很可惜。教师可能会发现在本科生课程中能简要地讨论一下这些章节中的一些主题还是比较有趣的，甚至可能为学生们后续的研究生课程打下一些基础。第 10 章讲述概率算法。10.6.2 和 10.7.1 小节给出了一些最惊人的例子，它们展示了在计算中随机是多么有用！第 11 章介绍了目前越来越

重要的并行算法。第 12 章介绍了计算复杂性。任何一门计算机科学本科生编程课程中都应当讲到 12.5 节中的 *NP* 完全性问题，但在一门算法学的课程上就不是那么必需了。最后，第 13 章会告诉你当面对一个从没解决过但必须解决的难题时，你能做什么，或者什么是你根本就别想解决的。

在每一章结尾处，为进一步深入阅读，都给出了一些建议。每一章提到的参考书目都会在本书末尾列出来，加上补充的一些，书目总共超过了 300 项。虽然在参考书目中我们给出了很多算法和思想的出处，但还是不可能事无巨细。如果有些是没有列出来的，也不必惊讶。我们的目标是提供一些补充材料以有助于你深入理解书中思想。对一些更具挑战性的问题的解决方法也会在那里列出来。

熟悉我们在 1988 年出版的研究生教材《算法学：理论与实践》的读者，就会注意到与这本书的共同之处。我们尤其坚信算法学应该作为一种设计技巧来教，而非侧重其应用。所以这就是为什么这两本书的目录都类似，甚至于本书的一些例子都从那一本逐字摘抄过来的。不过无论对本科生还是老师，都需要花些力气去学或教这本书。仅在全美，就有超过 100 所大学采用前者作为教材。从这些教这门课的教授们的反馈意见中，我们受益匪浅。此外我们还专门收集了一些问题，列在本书每一章的末尾处。第 1 章关于数学预备知识的内容就是新加上的。第 4 章为算法的分析提供了更多背景资料，还增加介绍了分期分析的概念。第 5 章对简单数据结构的回顾比上一版本更具有扩展性。每一章中的算法设计技巧都重新修订过，采用了更简单、更贴切和更有趣的算法。第 10 章关于概率算法的内容在前一版本中就已经是重头戏了，在这一版中修订得更干净利落，也更清晰。目前越来越热门的并行算法则是作为全新的一章加上去的。在前一版中我们收到最多的批评正是因为忽略了该主题。第 12 章的计算复杂性内容中增加了一些反方的论证，还加上了复杂性分类等节。第 13 章是新加的，包括了启发式算法和近似算法。最后，我们还更新和扩充了大量的参考书目，并完全修订了书中的问题。

如果没有许多人的帮助，我们根本不可能写出这本书。我们首先感谢那些过去超过 15 年的时间里，在 Montréal 大学，California 大学 Berkeley 分校，Lausanne 的 École 综合技术学院和一些国际短期课程中学过我们的算法课的学生们。我们在写 1988 年版及其前一版本——《算法学：概念与分析》(法语)过程中获得的经验对于这本书的提高起到了不可估量的作用。我们也感谢那些承认我们早期工作的人们。朋友和同事们给了我们大量的鼓励和建设性的意见。比如 Manuel Blum，他就使用了我们前一版的书。Prentice Hall 出版社中评审者的意见对我们来说也很重要。他们包括 Michael Atkinson，James A. Foster，Richard Newman-Wolfe，Clifford Shaffer，Doug Tygar 和 Steve Hoover。我们的同事 Pierre L'Écuyer，Christian Léger 和 Pierre McKenzie 帮我们把一些章节整理得更清晰。Faith Fich 写的课堂笔记促成了第 12 章中的反方论证材料的产生。

我们同样感谢那些使我们可以在 Montréal 大学之外花费长时间专心进行写作的人们。Paul Bratley 利用在 Strasbourg 大学、Fribourg 大学和 Nice 大学期间的年休假写了本书的部分内容。Gilles Brassard 则是利用在巴黎的 École 高等师范学院和在澳大利亚的 Wollongong 大学期间的年休假进行编写。我们特别感谢 Jean-François Dufourd，Jürg Kohlas，Olivier Lecarme，Jacques Stern 和 Jennifer Seberry，感谢他们的友善和好客。本书的前面一些章节是 Gilles Brassard 利用他的 E. W. R Steacie 基金会所提供的空闲时间写的。所以

在此他也感谢加拿大自然科学与工程委员会。他还感谢得以使他住进了乡村小屋的 Lise DuPlessis，这里森林宁静的气息鼓舞着 Gilles Brassard 写出本书的很多章节。

我们感谢在 Prentice Hall 出版社中的小组的热心和支持。Tom McElwee，计算机科学的编辑，是他第一次建议我们应该把此书作为面向本科学生的教材。而开始写书后，我们发现所花的时间远远超出了我们的预料，实际上还超过了编写第一本书的时间。这本书还接受了几位计算机科学编辑的审查，这里我们感谢 Bill Zobrist 和 Irwin Zucker 两位的一直支持。Phyllis Morgan 保证了本书的连贯性。Paul Mailhot 负责从我们的 LaTeX 格式的文档进行最后的排版。Montréal 大学信息与运筹学系的实验室负责人 Robert Gérin-Lajoie 和他的技术人员与分析人员组成的团队对我们的项目自始至终给予了无私的支持。

最后，但同样也很重要的是我们也非常感谢我们的妻子，Isabelle 和 Pat，感谢她们的鼓励和理解，感谢她们在我们忙着一起编写另一本书时给予我们的极大耐心。Gilles Brassard 也感谢他那两个有耐心的女儿，Alice 和 Léonore。她们俩在上一版的时候还没出生呢。Paul Bratley 则是感谢他的那些猫，它们可对算法一点都不感兴趣。

Gilles Brassard, Fairy Meadow
Paul Bratley, Pointe-Claire

目　录

第 1 章　预备知识 …… 1
1.1　简介 …… 1
1.2　什么是算法 …… 1
1.3　程序符号 …… 4
1.4　数学符号 …… 5
1.4.1　命题演算 …… 5
1.4.2　集合论 …… 6
1.4.3　整数、实数和区间 …… 7
1.4.4　函数和关系 …… 7
1.4.5　量词 …… 8
1.4.6　累加与累积 …… 9
1.4.7　杂项 …… 10
1.5　证明技巧 1——反证法 …… 11
1.6　证明技巧 2——数学归纳法 …… 12
1.6.1　数学归纳法的法则 …… 15
1.6.2　不同颜色的马 …… 18
1.6.3　一般化的数学归纳法 …… 19
1.6.4　构造性归纳 …… 22
1.7　一些回顾 …… 24
1.7.1　极限 …… 24
1.7.2　简单级数 …… 26
1.7.3　基本组合 …… 30
1.7.4　概率基础 …… 32
1.8　习题 …… 38
1.9　参考与深入阅读 …… 43

第 2 章　基本算法 …… 45
2.1　简介 …… 45
2.2　问题和实例 …… 45
2.3　算法的效率 …… 46
2.4　平均和最坏情况分析 …… 48
2.5　什么是基本运算？ …… 50

2.6 为什么寻求效率? …… 52
2.7 一些例子 …… 53
2.7.1 计算行列式的值 …… 53
2.7.2 排序 …… 53
2.7.3 大整数的乘法 …… 55
2.7.4 计算最大公约数 …… 55
2.7.5 计算斐波纳契序列 …… 56
2.7.6 傅立叶变换 …… 57
2.8 什么时候算法是确定的? …… 58
2.9 习题 …… 58
2.10 参考与深入阅读 …… 61

第3章 渐近记法 …… 62
3.1 引言 …… 62
3.2 同阶记法 …… 62
3.3 其他渐近记法 …… 67
3.3.1 Ω记法 …… 67
3.3.2 Θ记法 …… 68
3.4 条件渐近记法 …… 69
3.5 具有多个参数的渐近记法 …… 71
3.6 渐近记法的操作 …… 71
3.7 习题 …… 72
3.8 参考与深入阅读 …… 75

第4章 算法分析 …… 76
4.1 引言 …… 76
4.2 分析控制结构 …… 76
4.2.1 先后顺序 …… 76
4.2.2 For 循环 …… 76
4.2.3 递归调用 …… 78
4.2.4 while 和 repeat 循环 …… 79
4.3 使用标称 …… 80
4.4 补充例子 …… 82
4.4.1 选择排序 …… 82
4.4.2 插入排序 …… 83
4.4.3 欧几里德算法 …… 83
4.4.4 汉诺塔 …… 85
4.4.5 计算行列式的值 …… 86
4.5 平均情况分析 …… 86

4.6 分期分析 …… 87
4.6.1 势函数 …… 88
4.6.2 账户戏法 …… 90
4.7 求解递归式 …… 90
4.7.1 推测 …… 90
4.7.2 齐次递归式 …… 92
4.7.3 非齐次递归式 …… 96
4.7.4 变量变换 …… 100
4.7.5 范围转换 …… 105
4.7.6 渐近递归式 …… 106
4.8 习题 …… 107
4.9 参考与深入阅读 …… 113

第5章 一些数据结构 …… 114
5.1 数组、栈和队列 …… 114
5.2 记录和指针 …… 116
5.3 链表 …… 117
5.4 图 …… 118
5.5 树 …… 119
5.6 关联表 …… 124
5.7 堆 …… 126
5.8 二项堆 …… 132
5.9 不相交集结构 …… 136
5.10 习题 …… 141
5.11 参考与深入阅读 …… 144

第6章 贪婪算法 …… 146
6.1 找零钱(1) …… 146
6.2 贪婪算法的一般特性 …… 147
6.3 图:最小生成树 …… 148
6.3.1 Kruskal 算法 …… 150
6.3.2 Prim 算法 …… 152
6.4 图:最短路径 …… 154
6.5 背包问题(1) …… 157
6.6 日程安排 …… 160
6.6.1 最小化时间 …… 160
6.6.2 有期限的日程安排 …… 162
6.7 习题 …… 168
6.8 参考与深入阅读 …… 170

第7章 分治算法 …… 171
7.1 简介:大整数乘法 …… 171
7.2 通用模板 …… 174
7.3 二分搜索 …… 176
7.4 排序 …… 178
7.4.1 归并排序 …… 178
7.4.2 快速排序 …… 180
7.5 查找中值 …… 184
7.6 矩阵乘法 …… 188
7.7 求幂 …… 189
7.8 综合:密码学简介 …… 192
7.9 习题 …… 194
7.10 参考与深入阅读 …… 200

第8章 动态规划 …… 202
8.1 两个简单的例子 …… 202
8.1.1 计算二项式系数 …… 202
8.1.2 系列赛 …… 203
8.2 找零钱(2) …… 205
8.3 最优性原则 …… 207
8.4 背包问题(2) …… 208
8.5 最短路径 …… 209
8.6 链式矩阵乘法 …… 211
8.7 使用递归 …… 214
8.8 存储函数 …… 216
8.9 习题 …… 217
8.10 参考与深入阅读 …… 221

第9章 搜索图 …… 223
9.1 图和游戏:简介 …… 223
9.2 遍历树 …… 228
9.2.1 预处理 …… 228
9.3 深度优先搜索:无向图 …… 230
9.3.1 关节点 …… 232
9.4 深度优先搜索:有向图 …… 233
9.4.1 无环图:拓扑排序 …… 234
9.5 广度优先搜索 …… 236
9.6 回溯法 …… 239

9.6.1 背包问题(3) …… 240
9.6.2 八皇后问题 …… 242
9.6.3 通用模板 …… 244
9.7 分支界限 …… 244
9.7.1 分配任务问题 …… 245
9.7.2 背包问题(4) …… 248
9.7.3 一般考虑 …… 248
9.8 极小化原则 …… 248
9.9 习题 …… 251
9.10 参考与深入阅读 …… 256

第 10 章 概率算法 …… 257
10.1 简介 …… 257
10.2 概率不意味着不确定 …… 258
10.3 预期与平均时间 …… 259
10.4 生成伪随机数 …… 259
10.5 数值概率算法 …… 261
10.5.1 Buffon 的针 …… 261
10.5.2 数值积分 …… 264
10.5.3 概率计数 …… 265
10.6 Monte Carlo 算法 …… 267
10.6.1 验证矩阵乘法 …… 267
10.6.2 素数性测试 …… 269
10.6.3 一个数可能是素数吗? …… 272
10.6.4 随机优势的扩大 …… 274
10.7 Las Vegas 算法 …… 276
10.7.1 重访八皇后问题 …… 278
10.7.2 概率选择与排序 …… 281
10.7.3 通用散列 …… 282
10.7.4 大整数分解因数 …… 284
10.8 习题 …… 287
10.9 参考与深入阅读 …… 293

第 11 章 并行算法 …… 295
11.1 并行计算的模型 …… 295
11.2 一些基本的技术 …… 297
11.2.1 计算完全二叉树 …… 297
11.2.2 指针倍增 …… 298
11.3 工作量与效率 …… 301

11.4 图论的两个例子 …… 303
11.4.1 最短路径 …… 303
11.4.2 连通分量 …… 304
11.5 表达式的并行求值 …… 308
11.6 并行排序网络 …… 312
11.6.1 0-1 原理 …… 313
11.6.2 并行合并网络 …… 314
11.6.3 改进的排序网络 …… 315
11.7 并行排序 …… 316
11.7.1 预备工作 …… 316
11.7.2 核心思想 …… 317
11.7.3 算法 …… 317
11.7.4 细节概述 …… 318
11.8 EREW 和 CRCW p-ram 的一些注意点 …… 319
11.9 分布式计算 …… 320
11.10 习题 …… 321
11.11 参考与深入阅读 …… 323

第 12 章 计算复杂性 …… 324
12.1 引言:一个简单的例子 …… 324
12.2 信息-理论论证 …… 325
12.2.1 排序的复杂性 …… 327
12.2.2 复杂性对算法设计的帮助 …… 330
12.3 对手论证 …… 331
12.3.1 查找数组的最大项 …… 332
12.3.2 测试图的连通性 …… 333
12.3.3 中值再考察 …… 333
12.4 线性归约 …… 335
12.4.1 形式定义 …… 337
12.4.2 矩阵问题中的归约 …… 338
12.4.3 最短路径问题中的归约 …… 342
12.5 *NP* 完全性介绍 …… 344
12.5.1 *P* 和 *NP* 类 …… 345
12.5.2 多项式归约 …… 348
12.5.3 *NP* 完全性问题 …… 351
12.5.4 一些 *NP* 完全性证明 …… 353
12.5.5 *NP* 难问题 …… 356
12.5.6 非确定性算法 …… 357
12.6 复杂性类纵览 …… 359

12.7 习题 …… 362
12.8 参考与深入阅读 …… 366

第 13 章 启发式和近似算法 …… 369
13.1 启发式算法 …… 369
13.1.1 图着色 …… 369
13.1.2 旅行商 …… 371
13.2 近似算法 …… 372
13.2.1 度量旅行商 …… 372
13.2.2 背包问题(5) …… 374
13.2.3 装箱问题 …… 375
13.3 *NP* 难近似问题 …… 377
13.3.1 绝对难近似问题 …… 378
13.3.2 相对难近似问题 …… 379
13.4 相同,惟一不同 …… 380
13.5 近似模式 …… 383
13.5.1 重访装箱问题 …… 383
13.5.2 背包问题(6) …… 384
13.6 习题 …… 386
13.7 参考与深入阅读 …… 389

参考文献 …… 390

第1章 预备知识

1.1 简 介

本书将讲述算法和算法学。本章将以对这两个词的定义的讨论作为开始。在讨论中会用不同方法来进行一个简单的乘法运算。甚至像这样一个简单平常的问题里头都大有深意！还会解释为什么算法的研究是一件既有用又有趣的事情。

接下来会解释用以描述全书算法的一些符号。本章剩下的内容可能读者在别的地方就见过了，这里会再提及一些重要的内容以便让读者能够回忆起来。简要复习一些标准的数学符号后，会谈到两个常用的证明技巧：反证法和数学归纳法。最后列出一些关于极限、级数求和、基本组合和概率的结论。

如果读者对上述内容熟悉的话，应当看一下 1.2 和 1.3 节，然后就可以快速浏览本章的其余部分，也可以跳过那些你已经知道的内容。不过对那些久没碰过基础数学和计算机科学的读者来说，1.6.4 节应当通读，至少你应该读一下那里给出的一些主要结论。我们的表达简洁通俗，因为这一章节不是为了起到取代基本分析、微积分或者编程这些课程的作用。书的后续部分中将会用到本章给出的大部分结论。反之我们也尽量不让书中出现超出本章范围的一些结论。

1.2 什么是算法

算法(*algorithm*)一词来自 9 世纪波斯的数学家 al-Khowârizmî。算法，是一个(由人，或机器进行的)关于某种运算规则的集合。本书主要谈论在计算机中使用的算法。不过只要是为能够得到一个结果而进行运算的系统化方法也都是算法，比如我们在学校里学的加法、乘法和除法等。许多对无趣的布道深感厌烦的英国唱诗班孩子们，就用祈祷书(*the Book of Common Prayer*)上的算法来计算耶稣的复活节日期，以消磨度过布道冗长的时光。从古希腊起，历史上最有名的算法是：欧几里德(Euclid)的求两个整数最大公约数的算法。

一个算法的执行通常不能包含任何主观的决定，也不能有类似直觉或者创造力等因素掺杂其中。因此，如果在一本烹饪菜谱里精确描述了如何做某一道菜，严格规定了各种原料的数量和精确的烹饪时间的话，那么它也是算法。相反，如果在菜谱里只有诸如“加点盐巴后尝尝味道”或者“把食物煮软”等含糊的字眼，那就不能称得上是一个算法。

这条规则有个例外。允许算法在某个特定情况下，对下一步做什么可以有个随机的选择。第 10 章就专门讲述这些概率算法。这里值得读者注意的是“随机”并不意味着随意。相反，在碰到要从多个值中选择的情况下，选择每一个值的概率是已知的并且是可控制的。

算法是不会接受一条诸如“在 1 和 6 之间选一个数”而没有提供更多的细节的指令的。但是如果说成“在 1 和 6 之间选一个数，而且每个数被选中的概率相同”，那么这样的指令就可以接受。在碰到这条指令的情况时，如果算法要手工执行，我们可能会通过掷骰子来得到一个公正的结果。在计算机中则是通过一个伪随机数生成器来实现的。

当使用算法来求取一个特定问题的答案时，我们通常假设如果这些规则应用正确的话，就会确确实实得出正确的结果。一个计算 23 乘以 51 得到 1170 的规则集(23 乘以 51 应为 1173)在实际中一般没什么用处。然而在某些情况下，如近似算法(*approximate algorithms*)中，就会有用。例如想计算 2 的平方根，因为$\sqrt{2}$是一个无限不循环的无理数，所以在这种情况下，如果算法能得出一个可以精确到小数点后 4 位、10 位或任意位数的答案，那我们就说这个算法令人满意。

我们将会在第 12 章里看到还有很多问题没有实用的算法！对于此类问题，如果想得到精确解，采用已知的算法进行计算，大多要花费很长的时间，譬如几个世纪。所以一旦碰到此类问题，而又必须求解时，我们将被迫转而寻找这么一个规则集：它在可接受的时间内会算出足够好的近似解。如果能够证明用这个规则集得出的解误差很小，那就太好了。有时甚至还得不出满意的近似结果，此时只能碰运气——此类算法的运行结果通常大部分依赖于乐观主义，我们把它叫做“启发式算法”(*heuristic algorithm*)，或者简单地说成“启发”(*heuristic*)。注意近似算法和启发式算法的一个关键不同点是，前者我们可以根据需要规定算法可接受的误差，而后者的误差是无法控制的，但我们可以预计误差有多大。

在本书的前 12 章，除非上下文中有明确说明，否则其中所指的算法都是指对某个问题能够得出**正确**答案的规则集合。但第 13 章的算法就完全指近似算法和启发式算法。

在研究过算法后，现在可以来定义算法学。当开始着手解决问题时，此时可能有很多算法可供选择。根据优先需要和计算设备的限制，可能会选择满足某些条件的算法，譬如那些时间开销最短的，或者占用最少存储空间的，或最容易编程的，等等。而得出的答案也取决于很多因素，诸如参与运算的数、问题表达的方式或者计算设备的速度和存储容量的大小等等。可能没有哪个算法完全符合我们的要求，这时候就要根据自己需要设计一种新算法。算法学是这样的一门科学：它告诉我们如何评估不同的外在因素对可供选择算法的影响，好让我们可以选择一种最符合特定情况的算法；它也告诉我们如何为一个特定任务设计一种新算法。

以初等算术为例。假设你现在只能用笔和纸计算两个正整数的乘法。如果你在北美长大，你可能会从右到左取出被乘数的每一位，与乘数相乘。接下来你会把中间结果写在上一个中间结果的下面，同时最低位往左边偏移一位。最后你把每一行全部加起来就得到了你的结果。因此 981 乘以 1234 你可能产生如图 1.1(a)中给出的排列方式。相反如果你是在英国上的学，则可能是从左到右地运算，产生如图 1.1(b)所示的排列。

```
  981                  981
 1234                 1234
-----                -----
 3924                  981
2943                  1962
1962                  2943
 981                  3924
-----                -----
1210554              1210554
```

(a) 美国算法　　(b) 英国算法

图 1.1　乘法

这两种乘法算法太类似了，以致我们在讲到“传统”的乘法算法时，二者皆可为所指。第三种乘法算法在图 1.2 中列出。

981	1234	1234
490	2468	
245	4936	4936
122	9872	
61	19744	19744
30	39488	
15	78976	78976
7	157952	157952
3	315904	315904
1	631808	631808
		1210554

图 1.2　*à* la russe 乘法

把乘数和被乘数并排写在一起，每个操作数一列。将左边操作数整除 2，在该操作数下面写下商；将右边操作数乘以 2，在该操作数的下面写下积。以上一次的商和积作为操作数重复以上规则，直到左边的操作数为 1 为止。接着把左列中商为偶数的行全部删除，最后把右列中剩下的数字加起来就得到结果。图 1.2 说明了如何计算 981 乘以 1234。最后的答案这么得来：

$$1234 + 4936 + 19744 + 78976 + \cdots + 631808 = 1210554$$

这个算法有时候也叫做 *à* la russe(俄罗斯式)乘法算法，在计算机硬件里就用到了类似的算法进行乘法运算。它的优点是不要保存乘法表，只需要知道如何进行加法、如何把一个数整除以 2。虽然通常在学校里不会教这个算法，但它确实提供了一个手工进行两个正整数相乘运算的替代方法。

图 1.3 和图 1.4 列出了另一个计算两个正整数相乘的算法。这里再次举 981 乘以 1234 为例。不过在使用这个算法时，要求乘数和被乘数的位数要相等，并且它们的位数还须是 2 的幂，如 1,2,4,8,16,等等。这很好办，只要在必要的时候，在乘数或被乘数的左边补上若干个 0 就可以使它们的位数相等。这个例子里只需要在被乘数的左边补上 1 个 0，变成 0981，这样两个操作数就都是 4 位数了。

	乘		移位	结果
1)	09	12	4	108····
2)	09	34	2	306··
3)	81	12	2	972··
4)	81	34	0	2754
				1210554

图 1.3　用分治法计算 0981 乘以 1234

先将被乘数的左半部分(09)去乘乘数的左半部分(12)，然后把结果(108)向左移动，移动的位数就是乘数的位数，本例子中是 4 位。接下来将被乘数的左半部分(09)去乘乘数的右半部分(34)，然后把结果(306)向左移动，移动的位数是乘数位数的一半，本例子中是 2。第三步将被乘数的右半部分(81)去乘乘数的左半部分(12)，同样把结果(972)向左移动乘数位数的一半(2)；第四步将被乘数的右半部分(81)去乘乘数的右半部分(34)，得到结果(2754)，这次不移动。最后将 4 个中间结果加起来，如图 1.3 所示，得到答案 1 210 554。

如果照着刚才算法一步一步做一次的话，你会发现把两个 4 位数的乘法运算简化到了