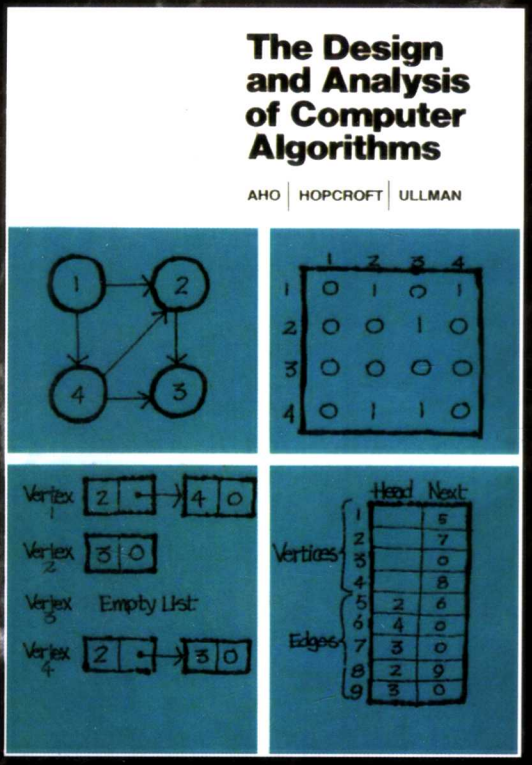
计 算 机 科 学 丛 书

计算机算法的设计与分析

新增经典算法的C/C++实现

(美) Alfred V. Aho 哥伦比亚大学 John E. Hopcroft 康奈尔大学 Jeffrey D. Ullman 斯坦福大学 著
黄林鹏 王德俊 张仕 上海交通大学 译



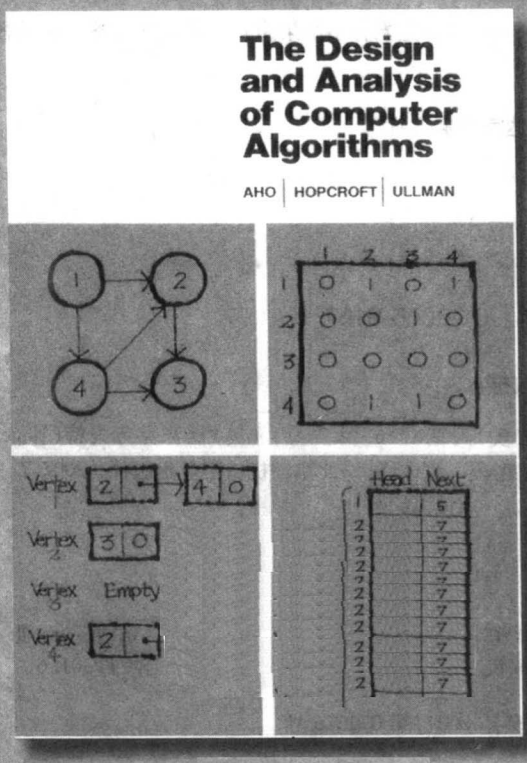
The Design and Analysis of Computer Algorithms

机械工业出版社
China Machine Press

计 算 机 科 学 丛

计算机算法的设计与分析

(美) Alfred V. Aho 哥伦比亚大学 John E. Hopcroft 康奈尔大学 Jeffrey D. Ullman 斯坦福大学 著 黄林鹏 王德俊 张 仕 上海交通大学 译



The Design and Analysis of Computer Algorithms

机械工业出版社
China Machine Press

本书是一部设计与分析领域的经典著作，着重介绍了计算机算法设计领域的基本原则和根本原理。书中深入分析了一些计算机模型上的算法，介绍了一些和设计有效算法有关的数据结构和编程技术，为读者提供了有关递归方法、分治方法和动态规划方面的详细实例和实际应用，并致力于更有效算法的设计和开发。同时，对NP完全等问题能否有效求解进行了分析，并探索了应用启发式算法解决问题的途径。另外，本书还提供了大量富有指导意义的习题。

本书可以作为高等院校计算机算法设计与分析课程的本科生或研究生教材，也可以作为计算机理论研究人员、计算机算法设计人员的参考书。

本书版权登记号：图字：01-2006-3157

图书在版编目（CIP）数据

计算机算法的设计与分析/(美)阿霍(Aho, A. V.)，(美)霍普克劳夫特(Hopcroft, J. E.)，(美)乌尔曼(Ullman, J. D.)著；黄林鹏，王德俊，张仕译. -北京：机械工业出版社，2007.7

(计算机科学丛书)

书名原文：The Design and Analysis of Computer Algorithms

ISBN 978-7-111-21543-1

Ⅰ.计…　Ⅱ.①阿…　②霍…　③乌…　④黄…　⑤王…　⑥张…　Ⅲ.①电子计算机-算法设计-高等学校-教材　②电子计算机-算法分析-高等学校-教材　Ⅳ.TP301.6

中国版本图书馆CIP数据核字(2007)第074676号

机械工业出版社(北京市西城区百万庄大街22号　邮政编码　100037)

责任编辑：王　玉

三河市明辉印装有限公司印刷·新华书店北京发行所发行

2007年7月第1版第1次印刷

184mm×260mm·26.75印张

定价：49.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：(010)68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自 1998 年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与 Prentice Hall，Addison－Wesley，McGraw-Hill，Morgan Kaufmann 等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出 Tanenbaum，Stroustrup，Kernighan，Jim Gray 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”。为了保证这两套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这两套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为 M. I. T.，Stanford，U. C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色—有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高

校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzjsj@ hzbook. com
联系电话：(010)68995264
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

译 者 序

本书是算法设计与分析领域的经典之作。

本书和 Knuth 所著的《计算机程序设计艺术》[⊖]、Cormen、Leiserson 和 Rivest 等著的《算法导论》[⊖]合称为算法设计领域的 3 大名著。和《算法导论》力求浅显易懂及颇具全面性相比，本书更倾向于数学和形式化描述，特别对图灵机模型和空间复杂度等有更深入的讨论。本书的另一个特点是简洁，直指问题核心，并给出严谨的分析及解决方案和改进措施。

不管从内容上还是形式上，目前市场上所有算法设计的书籍都深受本书的影响(截至本书中文版出版之前，据 Amazon 统计，已有 271 本书籍引用了本书)，可以说，不论是计算机理论研究人员、计算机算法设计者还是研究生、本科生，都可以通过阅读本书受到启发。

本书的特点

- 介绍了若干计算模型，包括图灵机模型、随机存取计算机模型以及它们的变体在算法设计和分析中的作用。
- 介绍了和设计有效算法相关的数据结构，给出递归、分治法、动态规划技术以及相关的算法例子。
- 解释了排序算法、集合算法、图算法、模式匹配算法、快速傅里叶变换的技术细节和应用技巧。
- 讨论了矩阵乘法、多项式运算以及整数算法的内在机理。
- 探讨了一些问题的时间复杂度和空间复杂度，涉及 NP 完全问题以及一些可以证明是不存在有效算法的问题，并将计算难度的概念和向量空间的线性独立性关联在一起。

关于本书附录

本书使用一种称为 Pidgin ALGOL 的语言来描述算法，Pidgin 意思是“由两种或多种语言混合而成的一种简化的说话方式，语法和词汇较初等，用于不同语言者进行交流”。算法的开发可分为设计、分析和实现 3 个环节。该过程可能不断重复，直到需求满足为止。一般说来，算法的描述只要清晰明了就可以了。书中所用的 Pidgin ALGOL 语言，对于算法的描述是充分的。但 ALGOL 是一个比较古老的语言，如果读者想测试书中给出的例子，那么找到一个 ALGOL 语言的编译器可是一件不容易的事(译者推荐使用 PLT Scheme 附带的 ALGOL 语言解释器，读者可参阅本书译者的另一本译作《程序设计方法》，人民邮电出版社 2003 年出版)。因此，在翻译本书的时候，经和机械工业出版社讨论协商，本书的代码保持原貌，另将书中的算法用 C/C++ 实现并作为译本的附录。本书附录中的代码由译者的研究生王慧芳实现，并经杨欢、彭冲、许赵云等同学测试，以保证其准确性。

本书的翻译工作由黄林鹏负责并与其博士生共同完成。其中第 9 章和第 10 章由王德俊翻译，第 7 章和第 8 章由张仕翻译，参加翻译的还有王欣、徐小辉、伍建焜等，全书由黄林鹏统稿和

⊖ 该书第 1 卷第 1 册和第 4 卷第 2、3 和 4 册的双语版已由机械工业出版社出版。——编辑注

⊖ 该书的第 2 版已由机械工业出版社翻译出版。——编辑注

审校。

本书的英文版，曾学习过两次。一次是1980年，当时我在浙江大学读计算机本科，另一次是1986年，那时我已经在上海交通大学攻读硕士学位了。这本教材给我的印象很深，但没有想到在20多年后我有机会翻译这本书，感谢机械工业出版社华章分社给我重温巨著并用中文表述的机会。在此，我要向所有为本书的翻译提供帮助的人表示感谢。由于我们水平有限，翻译中难免出现不妥和错误之处，敬请读者谅解，并请将意见寄至 lphuang@ sjtu. edu. cn，我们将不胜感激。

黄林鹏
于上海交通大学
2007年5月

前　言

算法研究是计算机科学的核心。近年来，算法领域取得了很多重要的进展。这些进展包括快速算法的开发，如发明了傅里叶变换快速算法，以及不存在有效算法的本质问题的惊人发现。这些结果点燃了计算机学者对算法研究的兴趣。算法设计与分析已成为一个受到广泛注意的领域。本书旨在将这个领域的一些基本成果汇集在一起，使算法设计的基本原则和根本原理的教学变得容易。

本书内容

分析一个算法的性能需要一种计算机模型。本书首先介绍一些形式模型，使用它们不仅可以进行算法分析，还能确切反映实际计算机的一些显著特性。这些模型包括随机存取计算机 RAM、随机存取存储程序计算机 RASP 以及一些变体。为了证明第 10、11 章中的一些算法的复杂度下界是指数的，本书还将讨论图灵机模型。由于程序设计趋向于远离机器语言，因此本书将引入一种称为简化 ALGOL 语言的高级程序语言作为描述算法的主要工具。简化 ALGOL 语言程序的复杂度和机器模型是相关的。

第 2 章介绍在设计有效的算法中使用的基本数据结构和程序设计技术。涉及列表、下推式存储结构、队列、树和图等。同时给出递归、分治法、动态规划的详细解释以及一些应用实例。

第 3 ~ 9 章给出一些不同的应用第 2 章所介绍的基本技术的领域。这些章节致力于设计已知最有效的渐近算法。由于要考虑渐近性，一些算法可能仅对那些输入规模比目前实际所遇到的问题更大时才有效。特别是第 6 章中的一些矩阵乘法算法、第 7 章中的 Schönhage - Strassen 整数乘法算法以及第 8 章中的一些多项式和整数算法。

另一方面，第 3 章中的大多数排序算法、第 4 章中的搜索算法、第 5 章的图算法、第 7 章的快速傅里叶算法以及第 9 章中的串匹配算法都是广泛应用的算法。这些算法对许多实际应用问题的输入规模来说都是有效的。

第 10 ~ 12 章讨论计算复杂度的下界。不管对于程序设计者还是希望了解计算本质的人都会对了解一个问题固有的计算难度感兴趣。第 10 章讨论一类重要的问题，即 NP 完全问题。该类中所有问题对于计算难度来说是等价的，即，若该类中有一个问题存在有效的算法(多项式时间界)，则该类中的所有问题都有有效的算法。由于该问题类包含了许多在现实中非常重要且被广泛研究的问题，如整数规划问题和旅行商问题，有理由怀疑此类问题不存在有效的算法。因此，如果一个程序设计者了解到他欲寻找有效算法的问题属于此类，他就应该尝试寻找启发式的方法。尽管有丰富的经验证据，NP 完全问题是否存在有效的算法还是一个有待解决的问题。

第 11 章定义了一些确实可以证明不存在有效算法的问题。第 10 章和第 11 章的内容依赖于本书 1.6 和 1.7 节引入的图灵机的概念。

最后一章将计算难度的概念和向量空间的线性独立性关联在一起。该章给出了证明较第 10 章和第 11 章简单很多的一些问题的复杂度下界的方法。

如何使用本书

本书旨在作为涉及算法设计与分析课程的入门教材。强调算法的思想以及如何方便地理解

算法而不是算法实现的技巧和细节。因此，常使用内容直观的解释而不是冗长的证明。本书是自包含的，读者无需具有数学和程序设计语言的专业背景。然而，还是希望读者具备一定的处理数学概念的能力，熟悉某种高级程序设计语言如 FORTRAN 或 ALGOL 等。要完全理解第 6、7、8 和 12 章等内容，读者还需要具备线性代数的知识。

本书曾用于算法设计的本科生和研究生课程。一个学期的课程一般涉及第 1～5 章以及第 9 和 10 章的大部分内容，其他章节则可浅尝辄止。在算法的介绍性课程中，可以第 1～5 章为重点，而 1.6、1.7、4.13、5.11 节和定理 4.5 一般可不涉及。本书也可用于强调算法理论的高级课程，第 6～12 章的内容可作为此类课程的基础资料。

每章最后都提供有大量的习题，教师可选用作为学生的作业。习题按难度进行分类。没有星号的可用于介绍性的课程，有一个星号(*)的可用于高级课程，有两个星号的可用于研究生高级课程。每章后面的参考文献和注释提供了正文以及习题包含的相应算法和结果的公共资源。

致谢

本书的材料取自作者在康奈尔大学、普林斯顿大学和史迪温理工学院开设的算法课程的讲义。作者要感谢认真读过本书部分手稿的许多人所提出的批评和建议。作者特别要感谢 Kellogg Booth、Stan Brown、Steve Chen、Allen Cypher、Arch Davis、Mike Fischer、Hania Gajewska、Mike Garey、Udai Gupta、Mike Harrison、Matt Hecht、Harry Hunt、Dave Johnson、Marc Kaplan、Don Johnson、Steve Johnson、Brian Kernighan、Don Knuth、Richard Ladner、Anita LaSalle、Doug McIlroy、Albert Meyer、Christos Papadimitriou、Bill Plauger、John Savage、Howard Siegel、Ken Steiglitz、Larry Stockmeyer、Tom Szymanski 和 Theodore Yen 等。

特别感谢 Gemma Carnevale、Pauline Cameron、Hannah Kresse、Edith Purser 和 Ruth Suzuki 等对手稿所进行的细心和赏心悦目的排版。

感谢贝尔实验室、康奈尔大学、普林斯顿大学和加州大学伯克利分校为作者准备本书手稿所提供的设施。

A. V. A.
J. E. H.
J. D. U.
1974 年 6 月

目　录

第1章 计算模型

给定一个问题，如何寻求一个有效的求解算法？如果得到了一个算法又该如何与解决同一问题的其他算法进行比较？如何判断一个算法的好坏？程序设计者和理论计算机学者常对此类问题的本质感兴趣。本书将从不同的方面探讨这些问题。

本章将讨论几个典型的计算机模型，包括随机存取计算机 RAM、随机存取存储程序计算机 RASP 和图灵机 TM 等。比较这些模型的能力和算法复杂度的关系，从其导出若干更成熟的计算模型，如直线状程序模型、位计算模型、位向量计算模型和决策树模型等。本章的最后一节介绍一种描述算法的“简化 ALGOL 语言”。

1.1 算法和复杂度

评价算法有多个标准。最通常的是考虑其求解越来越大的问题实例所需的时间或空间开销的增长率。一般将一个称为问题规模的整数和算法相关联，该整数通常是问题的输入数据的规模。例如，对于矩阵乘法，其规模可能是相乘的矩阵的最大维数，而图问题的规模可能是边的数目。

算法所需要的时间是问题规模的函数，称为算法的时间复杂度。当问题规模增加时，复杂度的极限行为称为算法的渐近时间复杂度。类似的定义也适用于空间复杂度和渐近空间复杂度。

算法的渐近复杂度确定了可用该算法解决的问题的规模。对于某个常数 c，如果一个算法能在 cn^2 时间内处理输入规模为 n 的问题，则称该算法的时间复杂度为 $O(n^2)$，读为“阶为 n^2”。更精确地，如果存在常数 c，除了有限的若干个值(可能为空)外，对所有非负的 n 都有 $g(n) \leqslant cf(n)$，则称函数 $g(n)$ 是 $O(f(n))$ 的。

也许有人认为，现代数字计算机的出现带来的计算机速度的大幅度提高会削弱对有效算法的需求。然而，结果恰恰相反。计算速度的提高使我们可以处理更大的问题，但算法复杂度决定了由于计算速度的提高带来的可处理的问题规模的增量的大小。

假定有时间复杂度如下的 5 个算法 $A_1 \sim A_5$：

算法	时间复杂度
A_1	n
A_2	$n\log n$[⊖]
A_3	n^2
A_4	n^3
A_5	2^n

这里的时间复杂度是处理一个输入规模为 n 的问题的时间单元数。假定一个时间单元等于 1 毫秒，算法 A_1 能在一秒时间内处理输入规模为 1000 的问题，而算法 A_5 可在一秒内处理输入规模至多为 9 的问题。图 1-1 分别给出了 1 秒、1 分和 1 个小时内这 5 个算法可以处理的问题规模。

⊖ 除非特别说明，本书所有对数都是以 2 为底。

算法	时间复杂度	最大问题规模		
		1秒	1分	1小时
A_1	n	1000	6×10^4	3.6×10^6
A_2	$n\log n$	140	4893	2.0×10^5
A_3	n^2	31	244	1897
A_4	n^3	10	39	153
A_5	2^n	9	15	21

图 1-1 由增长率所确定的可解决的问题规模的限度

算法	时间复杂度	加速前最大问题规模	加速后最大问题规模
A_1	n	s_1	$10s_1$
A_2	$n\log n$	s_2	约 $10s_2$ 对于大的 s_2
A_3	n^2	s_3	$3.16s_3$
A_4	n^3	s_4	$2.15s_4$
A_5	2^n	s_5	$s_5+3.3$

图 1-2 十倍加速的效果

假定下一代计算机的速度是目前的 10 倍。图 1-2 是计算速度增加后在相同的时间内可以解决的问题规模的增量。注意对于算法 A_5，10 倍的加速得到的结果是可以解决的问题规模比原来的规模仅大 3 左右，而对于算法 A_3 则是原来的 3 倍。

撇开速度的增加，现在考虑使用效率更高的算法的效果。回到图 1-1，使用 1 分钟为比较基础，用算法 A_3 代替算法 A_4，此时可以解决规模为原来 6 倍的问题；用算法 A_2 代替算法 A_4，则可以解决规模为原来 125 倍的问题。这些结果比用速度快 10 倍的计算机却得到 2 倍的改进更使人印象深刻。如果以 1 个小时作为比较基础，差异将更加明显。由此可以得到结论，一个算法的渐近时间复杂度是衡量算法好坏的一个重要标准，随着未来计算速度的提高，它必将变得更加重要。

除了注重阶量级外，也要认识到有时一个增长率较快的算法的比例常数可能比一个增长率较低的算法小。在这种情况下，对于小的问题，甚至是使我们感兴趣的同一规模下的所有问题，增长率较快的算法可能会比增长率低的算法优越。例如，假定算法 A_1、A_2、A_3、A_4 和 A_5 的时间复杂度分别是 $1000n$，$100n\log n$，$10n^2$，n^3 和 2^n，则对于问题规模为 $2\leqslant n\leqslant 9$，A_5 将是最好的算法；对于 $10\leqslant n\leqslant 58$，$A_3$ 将是最好的算法；对于 $59\leqslant n\leqslant 1024$，$A_2$ 将是最好的算法；而对于大于 1024 的规模，A_1 才是最好的算法。

在进一步讨论算法和复杂度之前，必须定义执行算法的计算设备模型，并指出计算中的基本步的含义。遗憾的是，并没有一个对所有情况都适用的模型。主要的困难在于如何对计算机字的大小进行定义。例如，如果假定一个计算机字可以存储任意大小的整数，则通过编码，整个问题就可以存放于一个计算机字中的一个整数。另一方面，如果假定一个计算机字的长度是有限的，则必须考虑如何存储任意大小的整数，以及通常在处理规模适当的问题时可以避免的其他困难。对于每个问题，都必须选择一个合适的模型来反映给定的算法在真正的计算机上运行时所耗费的时间。

接下来的几节讨论计算设备的一些基本模型，比较重要的模型是随机存取计算机、随机存取程序计算机和图灵机。在计算能力上它们是等价的，但在速度上不一样。

研究形式计算模型的最主要动机是揭示不同问题的固有计算难度。本章将证明一些问题的计算时间的下界。为了说明对于给定的问题不存在一个算法能在某一时间内解决它，需要一个精确的、常常是高度程式化的算法定义。图灵机(参见1.6节)就是定义算法的一个好的工具。

在描述算法和交流算法时，还需要一个比随机存取计算机、随机存取存储程序计算机或图灵机更加自然并易于理解的算法描述记法。由此，本书引进了称为Pidgin(Pidgin的含义是由两种或多种语言混合而成的一种简化的语言。——译者注)ALGOL的高级语言，即简化ALGOL语言。本书将使用该语言进行算法描述。然而，为了解使用简化ALGOL语言描述的算法的复杂度，还必须将简化ALGOL语言和更加形式化的计算模型相关联。本章最后一节讨论该问题。

1.2 随机存取计算机

随机存取计算机(RAM)模拟一台带有一个累加器的计算机，该计算机的程序指令不允许对程序自身进行修改。

一台RAM计算机包括只读输入带、只写输出带、一个程序和一个存储器(见图1-3)。输入带是一个方格序列，每个方格可存放一个整数(可以是负整数)。每从输入带读取一个整数，带首就往右移动一个方格。只写操作的输出带的初始状态全部为空。每执行一次写指令，就在当前输出带首的方格中写入一个整数，并将带首往右移动一个方格。不允许对输出带上写入的符号进行修改。

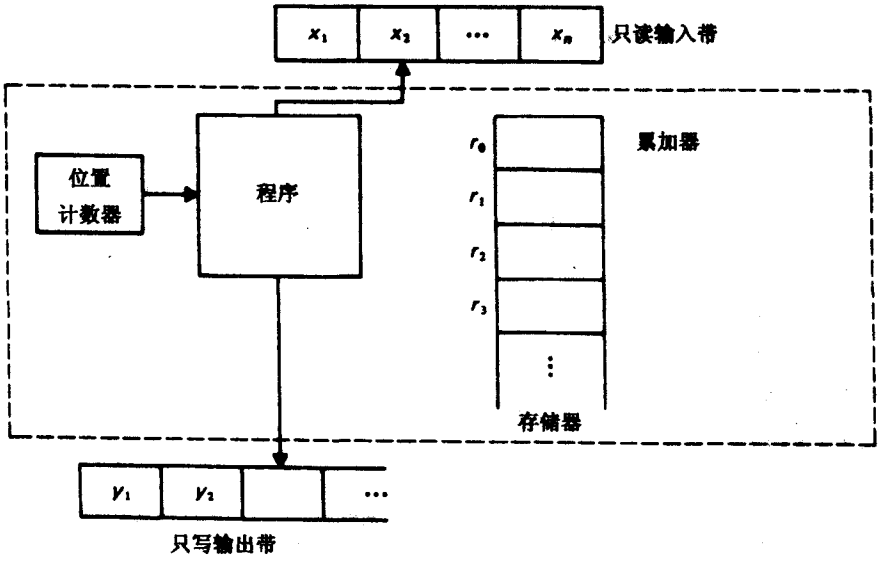


图1-3 一台RAM计算机

存储器包含了寄存器r_0，r_1，…，r_i，…，每个都可存储任意大小的整数，可用寄存器的数目没有限制。该抽象形式适用于下述情景：

1. 问题的规模足够小，可以存放在计算机的主存储器内；
2. 计算中用到的整数足够小，可以用一个计算机字表示。

在RAM模型中，程序并不存放在存储器中，因此程序不能修改自身。程序只是(可选)带标号的指令序列，同实际计算机上使用的指令类似，但其确切特性并不重要，通常可假定有算术运

算指令、输入输出指令、间接寻址(例如数组的下标)和转移指令等。所有的运算都在称为累加器的寄存器 r_0 中进行，累加器和其他的寄存器一样可以存储任意的整数。一个 RAM 指令集范例如图 1-4 所示。每条指令包含两个部分，操作码和地址。

操作码	地址
1. LOAD	操作数
2. STORE	操作数
3. ADD	操作数
4. SUB	操作数
5. MULT	操作数
6. DIV	操作数
7. READ	操作数
8. WRITE	操作数
9. JUMP	标号
10. JGTZ	标号
11. JZERO	标号
12. HALT	

图 1-4 RAM 指令表

原则上说，可以将图 1-4 中的指令集扩充为任何实际计算机的指令集，如添加逻辑或字符操作指令，而不改变解决问题的复杂度的阶量级。读者可以按照需要设想合适的指令集。

操作数可以是如下之一：

1. $=i$ 表示整数 i 本身；
2. 非负整数 i 表示寄存器 i 的内容；
3. $*i$ 表示间接寻址。即操作数是寄存器 j 的内容，j 是寄存器 i 中存放的整数。如果 $j<0$，则停机。

对于使用汇编语言编程的程序员来说，这些指令是再熟悉不过的了。定义程序 P 的含义，借助两个量，一个是从非负整数到整数的映射 c，另一个是“位置计数器”，其确定下一条要执行的指令。函数 c 是内存映射，$c(i)$ 是存储在寄存器 i 中的整数(即寄存器 i 的内容)。

初始时，对所有 $i\geq 0$，$c(i)=0$，位置计数器指向 P 中第一条指令，输出带全部为空白。在执行 P 的第 k 条指令之后，位置计数器的值自动设为 $k+1$(即下一条指令)，除非第 k 条指令是 JUMP、HALT、JGTZ 或 JZERO。

为了说明指令的含义，可定义操作数 a 的值 $v(a)$ 如下

$$v(=i)=i$$
$$v(i)=c(i)$$
$$v(*i)=c(c(i))$$

图 1-5 中的表定义了图 1-4 中每条指令的含义。没有定义的指令，如 STORE $=i$，可以认为和 HALT 一样。类似地，除以 0 也将使机器停机。

指令	含义
1. LOAD a	$c(0)\leftarrow v(a)$
2. STORE i	$c(i)\leftarrow c(0)$
STORE $*i$	$c(c(i))\leftarrow c(0)$
3. ADD a	$c(0)\leftarrow c(0)+v(a)$
4. SUB a	$c(0)\leftarrow c(0)-v(a)$
5. MULT a	$c(0)\leftarrow c(0)\times v(a)$
6. DIV a	$c(0)\leftarrow \lfloor c(0)/v(a)\rfloor$ ①
7. READ i	$c(i)\leftarrow$ 当前输入符号
READ $*i$	$c(c(i))\leftarrow$ 当前输入符号。这两种情况下，输入带首都右移一个方格
8. WRITE a	在输出带首的当前输出带方格打印 $v(a)$。然后带首右移一个方格
9. JUMP b	将位置计数器设置为标号为 b 的指令
10. JGTZ b	如果 $c(0)>0$，则将位置计数器设置为标号为 b 的指令；否则，将位置计数器的值设置为下一条指令
11. JZERO b	如果 $c(0)=0$，则将位置计数器设置为标号为 b 的指令；否则，将位置计数器设置为下一条指令
12. HALT	执行停止

① 在本书中，$\lceil x\rceil$(x 的上取整)表示大于或等于 x 的最小整数，$\lfloor x\rfloor$(x 的下取整)表示小于或等于 x 的最大整数。

图 1-5 RAM 指令的含义。操作数 a 是 $=i$、i 或 $*i$

在执行头8条指令的每一条时，位置计数器增1。因此，除非遇到JUMP或HALT指令，或者当累加器的内容大于0时遇到JGTZ指令，或者当累加器的内容等于0时遇到JZERO指令，否则程序中的指令将按照顺序依次执行。

一般来说，RAM程序定义了一个从输入带到输出带的映射。因为程序并不是对所有的输入带都会停机，因此映射是一个部分映射(即，对一些输入映射可能没有定义)。映射可以使用不同的方式解释，其中两种重要的解释是把映射看作一个函数或一种语言。

假定程序 P 总是从输入带读取 n 个整数并最多在输出带上写进一个整数。如果 x_1，x_2，…，x_n 是输入带前 n 个方格中的整数，程序 P 的执行结果是将 y 写入输出带的第一个方格并停机，则称程序 P 计算了函数 $f(x_1, x_2, \cdots, x_n) = y$。容易看出，RAM模型和其他合理的计算机模型一样，可以计算*部分递归函数*。即，给定任意的部分递归函数 f，可以定义一个计算 f 的RAM程序，给定一个RAM程序，可以定义一个等价的部分递归函数。(参阅Davis[1958]或Rogers[1967]对递归函数的讨论。)

另外一种解释是将RAM程序作为一个语言的接收器。*字母表*是有限个符号的集合，*语言*是该字母表上字符串的一个集合。字母表的符号可以对于某个 k，表示为整数1，2，…，k。RAM可按以下方式接受一种语言。将输入串 $s = a_1 a_2 \cdots a_n$ 放置到输入带，其中符号 a_1 位于第一个方格、符号 a_2 位于第2个方格，以此类推。另外，将作为终止符号的0放置于第 $n+1$ 个方格，作为输入符号串的结束。

如果 P 读入所有 s 中的符号以及结束符，则输入串 s 被RAM程序 P *接受*，将1写入输出带的第1个方格并停机。P *接受的语言*是所有被 P 接受的输入符号串的集合。对不被 P 接受的输入字符串，程序 P 将输出非1的符号并停机，也可能不停机。容易证明，一种语言能被RAM程序接受，当且仅当其是*递归可枚举*的语言。一种语言能被对所有输入都可停机的RAM接受，当且仅当其是*递归*语言(参见Hopcroft和Ullman[1969]对递归语言和递归可枚举语言的讨论)。

考虑下面两个RAM程序示例。前者定义一个函数，后者接受一种语言。

例1.1 考虑如下给定的函数 $f(n)$

$$f(n) = \begin{cases} n^n & \text{对所有整数 } n \geqslant 1 \\ 0 & \text{其他} \end{cases}$$

```
begin
    read r1;
    if r1 ≤ 0 then write 0
    else
        begin
            r2 ← r1;
            r3 ← r1 − 1;
            while r3 > 0 do
                begin
                    r2 ← r2 * r1;
                    r3 ← r3 − 1
                end;
            write r2
        end
end
```

图1-6 计算 n^n 的简化ALGOL程序

图1-6所示是一个简化ALGOL语言的程序[⊖]，通过将 n 自乘 $n-1$ 次来计算 $f(n)$。相应的RAM程序见图1-7。其中变量 $r1$、$r2$ 和 $r3$ 分别存放在寄存器1、2和3之中。由于没有进行程序优化，图1-6和图1-7的对应比较明显。□

例1.2 考虑一个RAM程序，它接受输入字母表{1，2}上的语言，该语言包含1和2的数目相等的任何字符串。程序将每个输入符号读入寄存器1，而寄存器2则保留了目前所看到的1和2的数目的差异数 d。当读入终止符0时，程序检查差异数是否为0，如是，则打印1并停机。假定0、1和2是仅有的可能输入符号。

图1-8中的程序包含了基本的算法细节。等价的RAM程序如图1-9所示，其中寄存器1存放 x，而寄存器2存放 d。□

⊖ 参见1.8节对于简化ALGOL语言的描述。

RAM 程序			相应的简化 ALGOL 语句
	READ	1	read r1
	LOAD	1	
	JGTZ	pos	if r1 ≤ 0 then write 0
	WRITE	=0	
	JUMP	endif	
pos:	LOAD	1	r2 ← r1
	STORE	2	
	LOAD	1	
	SUB	=1	r3 ← r1 − 1
	STORE	3	
while:	LOAD	3	
	JGTZ	continue	while r3 > 0 do
	JUMP	endwhile	
continue:	LOAD	2	
	MULT	1	r2 ← r2 * r1
	STORE	2	
	LOAD	3	
	SUB	=1	r3 ← r3 − 1
	STORE	3	
	JUMP	while	
endwhile:	WRITE	2	write r2
endif:	HALT		

图 1-7 计算 n^n 的 RAM 程序

```
begin
    d ← 0;
    read x;
    while x ≠ 0 do
        begin
            if x ≠ 1 then d ← d − 1 else d ← d + 1;
            read x
        end;
    if d = 0 then write 1
end
```

图 1-8 识别具有相同数目的 1 和 2 的字符串的程序

RAM 程序			对应的混合 ALGOL 语句
	LOAD	=0	d ← 0
	STORE	2	
	READ	1	read x
while:	LOAD	1	while x ≠ 0 do
	JZERO	endwhile	
	LOAD	1	
	SUB	=1	if x ≠ 1
	JZERO	one	
	LOAD	2	
	SUB	=1	then d ← d − 1
	STORE	2	
	JUMP	endif	
one:	LOAD	2	
	ADD	=1	else d ← d + 1
	STORE	2	
endif:	READ	1	read x
	JUMP	while	
endwhile:	LOAD	2	
	JZERO	output	if d = 0 then write 1
	HALT		
output:	WRITE	=1	
	HALT		

图 1-9 与图 1-8 中算法相对应的 RAM 程序