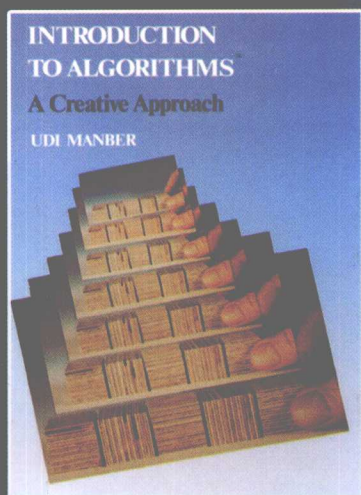


算法引论

—— 一种创造性方法

Introduction to Algorithms A Creative Approach



[美] Udi Manber 著

黄林鹏 谢瑾奎 陆首博 等译



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

国外计算机科学教材系列

算法引论

—— 一种创造性方法

Introduction to Algorithms
A Creative Approach

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书是国际算法大师乌迪·曼博 (Udi Manber) 博士撰写的一本享有盛誉的著作。全书共分 12 章: 第 1 章到第 4 章为介绍性内容, 涉及数学归纳法、算法分析、数据结构等内容; 第 5 章提出了与归纳证明进行类比的算法设计思想; 第 6 章到第 9 章分别给出了 4 个领域的算法, 如序列和集合的算法、图算法、几何算法、代数和数值算法; 第 10 章涉及归约, 也是第 11 章的序幕, 而后者涉及 NP 完全问题; 第 12 章则介绍了并行算法; 最后是部分习题的答案及参考文献。本书的特色有二, 旨在提高读者的问题求解能力, 使读者能够理解算法设计的过程和思想: 一是强调算法设计的创造性过程, 注重算法设计背后的创造性思想, 而不拘泥于某个具体算法的详细讨论; 二是将算法设计类比为定理归纳证明, 揭示了算法设计的基本思想和本质。

本书的组织结构清晰且易于理解, 强调了创造性, 具有浓郁特色, 时至今日仍有其巨大的价值, 并且适合作为计算机及相关专业算法和高级算法课程的教材。

Authorized translation from the English language edition, entitled INTRODUCTION TO ALGORITHMS: A CREATIVE APPROACH, 9780201120370 by Udi Manber, published by Pearson Education, Inc, publishing as Addison-Wesley, Copyright © 1989 by Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD., and PUBLISHING HOUSE OF ELECTRONICS INDUSTRY Copyright © 2010.

本书中文简体字版专有版权由 Pearson Education (培生教育出版集团) 授予电子工业出版社。未经出版者预先书面许可, 不得以任何方式复制或抄袭本书的任何部分。

本书贴有 Pearson Education (培生教育出版集团) 激光防伪标签, 无标签者不得销售。

版权贸易合同登记号 图字: 01-2005-0327

图书在版编目 (CIP) 数据

算法引论——一种创造性方法 / (美) 曼博 (Manber, U.) 著; 黄林鹏等译.

北京: 电子工业出版社, 2010.1

(国外计算机科学教材系列)

书名原文: Introduction to Algorithms: A Creative Approach

ISBN 978-7-121-09813-0

I. 算… II. ①曼… ②黄… III. 电子计算机—算法理论—教材 IV. TP301.6

中国版本图书馆 CIP 数据核字 (2009) 第 201517 号

责任编辑: 冯小贝

印 刷: 北京市海淀区四季青印刷厂

装 订: 涿州市桃园装订有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编: 100036

开 本: 787 × 1092 1/16 印张: 21.75 字数: 571 千字

印 次: 2010 年 1 月第 1 次印刷

定 价: 36.00 元

凡所购买电子工业出版社的图书有缺损问题, 请向购买书店调换; 若书店售缺, 请与本社发行部联系。联系及邮购电话: (010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。

出版说明

21世纪初的5至10年是我国国民经济和社会发展的关键时期,也是信息产业快速发展的关键时期。在我国加入WTO后的今天,培养一支适应国际化竞争的一流IT人才队伍是我国高等教育的重要任务之一。信息科学和技术方面人才的优劣与多寡,是我国面对国际竞争时成败的关键因素。

当前,正值我国高等教育特别是信息科学领域的教育调整、变革的重大时期,为使我国教育体制与国际化接轨,有条件的高等院校正在为某些信息学科和技术课程使用国外优秀教材和优秀原版教材,以使我国在计算机教学上尽快赶上国际先进水平。

电子工业出版社秉承多年来引进国外优秀图书的经验,翻译出版了“国外计算机科学教材系列”丛书,这套教材覆盖学科范围广、领域宽、层次多,既有本科专业课程教材,也有研究生课程教材,以适应不同院系、不同专业、不同层次的师生对教材的需求,广大师生可自由选择 and 自由组合使用。这些教材涉及的学科方向包括网络与通信、操作系统、计算机组织与结构、算法与数据结构、数据库与信息处理、编程语言、图形图像与多媒体、软件工程等。同时,我们也适当引进了一些优秀英文原版教材,本着翻译版本和英文原版并重的原则,对重点图书既提供英文原版又提供相应的翻译版本。

在图书选题上,我们大都选择国外著名出版公司出版的高校教材,如Pearson Education培生教育出版集团、麦格劳-希尔教育出版集团、麻省理工学院出版社、剑桥大学出版社等。撰写教材的许多作者都是蜚声世界的教授、学者,如道格拉斯·科默(Douglas E. Comer)、威廉·斯托林斯(William Stallings)、哈维·戴特尔(Harvey M. Deitel)、尤利斯·布莱克(Uyless Black)等。

为确保教材的选题质量和翻译质量,我们约请了清华大学、北京大学、北京航空航天大学、复旦大学、上海交通大学、南京大学、浙江大学、哈尔滨工业大学、华中科技大学、西安交通大学、国防科学技术大学、解放军理工大学等著名高校的教授和骨干教师参与了本系列教材的选题、翻译和审校工作。他们中既有讲授同类教材的骨干教师、博士,也有积累了几十年教学经验的老教授和博士生导师。

在该系列教材的选题、翻译和编辑加工过程中,为提高教材质量,我们做了大量细致的工作,包括对所选教材进行全面论证;选择编辑时力求达到专业对口;对排版、印制质量进行严格把关。对于英文教材中出现的错误,我们通过与作者联络和网上下载勘误表等方式,逐一进行了修订。

此外,我们还将与国外著名出版公司合作,提供一些教材的教学支持资料,希望能为授课老师提供帮助。今后,我们将继续加强与各高校教师的密切联系,为广大师生引进更多的国外优秀教材和参考书,为我国计算机科学教学体系与国际教学体系的接轨做出努力。

电子工业出版社

教材出版委员会

主 任	杨芙清	北京大学教授 中国科学院院士 北京大学信息与工程学部主任 北京大学软件工程研究所所长
委 员	王 珊	中国人民大学信息学院院长、教授
	胡道元	清华大学计算机科学与技术系教授 国际信息处理联合会通信系统中国代表
	钟玉琢	清华大学计算机科学与技术系教授 中国计算机学会多媒体专业委员会主任
	谢希仁	中国人民解放军理工大学教授 全军网络技术研究中心主任、博士生导师
	尤晋元	上海交通大学计算机科学与工程系教授 上海分布计算技术中心主任
	施伯乐	上海国际数据库研究中心主任、复旦大学教授 中国计算机学会常务理事、上海市计算机学会理事长
	邹 鹏	国防科学技术大学计算机学院教授、博士生导师 教育部计算机基础课程教学指导委员会副主任委员
	张昆藏	青岛大学信息工程学院教授

译者序

本书作者乌迪·曼博 (Udi Manber) 博士是美国著名的计算机科学家, 国际公认的算法大师, 在线信息搜索引擎的先驱。乌迪·曼博曾是美国亚利桑那大学计算机专业的教授, 离开学校后在雅虎公司担任执行官, 目前是亚马逊 (Amazon.com) 的副总裁和首席算法师 (CAO), 也是亚马逊旗下搜索网站 A9.com 的首席执行官, 他提出的 UDI 测试已经成为衡量搜索引擎质量的评估标准。

本书的特色有二, 一是强调算法设计的创造性过程, 而不是拘泥于某个具体算法的详细讨论, 正如常言道“予之以鱼, 不如授之以渔”。在教学过程中常常发现有许多同学 (包括译者自己) 不是去掌握算法背后的思想, 而是钻牛角尖, 针对某个算法反复地思考, 或者只是掌握了几个具体的算法就沾沾自喜。事实上学习算法不外乎两个目的, 一是了解各种算法, 在遇到问题时能灵活地应用所掌握的方法技巧; 二个研究算法设计技术, 当没有现成可用的算法时, 能够创造出问题的求解方法。前者非常实际, 但后者更加重要, 它涉及创新, 是推动学科发展的源泉。本书中的习题就很有特色, 大致分为两类, 一类是操练型的, 意在巩固对具体算法的了解, 锻炼学生应用书中介绍的算法以及各种变体的设计能力; 另一类是创造型的, 旨在让学生掌握设计算法的技术而非算法本身, 由此锻炼学生的算法设计能力而不是算法应用能力。这种体系结构是一种尝试, 虽然目前市面上有许多大部头的算法著作, 包括一些算法大师的名著, 但本书强调的是创造性, 具有浓郁特色, 因此时至今日仍有巨大的价值。

本书的特色之二是将算法设计类比于归纳证明。事实上, 归纳证明、递归算法、递推关系, 它们之间存在着千丝万缕的联系。对事物, 人们往往是从简单到复杂, 从具体到抽象地进行思考, 猜测存在的性质并试图证明, 最后得到一般性的规律再加以应用。问题求解也是如此, 我们先从简单的情况出发, 考虑简单问题的解决方案, 然后再推广到一般的情况, 在此过程中需要验证的是这种推广的有效性, 这无疑就是一种归纳证明。作者将算法设计类比于归纳证明并贯穿全书, 将各种归纳方法演绎得淋漓尽致, 使通常的算法设计有章可循, 更使设计算法成为一种享受。

译者黄林鹏教授与本书的翻译有着不解之缘。2003 年初, 译者的研究生导师左孝凌教授拟修改他的离散数学教材, 准备添置论述归纳法的章节, 由此译者为他收集了许多关于归纳法的材料, 并粗读了其中一部分, 也了解到了一些皮毛。不料 2004 年暑假左老师因操劳过度, 竟不辞西去。译者希望能以这本与归纳法紧密相关的翻译教材的出版向左老师及其未尽事业致敬。此外就是关于并行计算中的脉动阵列算法, 15 年前译者在读博士的时候, 恰好其博士生导师孙永强教授在做关于脉动阵列的研究, 译者得以参与, 并对此稍有了解。2000 年译者在哈佛大学毕业进修, 遇到脉动阵列体系结构的创始人之一, 并有幸旁听了他的课程。而本书最后就是关于脉动阵列算法的, 读来译来, 会有些如偶遇故知和聆听智者教诲的感觉。虽然脉动阵列是本书成书时的研究热点, 现在时过境迁, 已经是集群计算、分布对象计算、网格计算的年代, 但书中涉及的并行算法设计的思想依然璀璨, 富有启迪。事实上, 正是因为强调的是算法设计的创造性过程而非算法本身, 才使得本书可以永保活力。

本书的翻译工作由上海交通大学的黄林鹏教授负责, 并与一些老师和博士生共同完成, 其中的第 2 章、第 4 章、第 6 章由陆首博初译, 第 3 章、第 5 章、第 7 章由谢谨奎初译, 参与本书翻译工作的还有欧佳凡、黄青、杨欢、林海源、毛宏燕、黄晓琴、李东、刘飞、林琳、王佳钰、耿向阳等同志, 陆朝俊和任庆生副教授对本书一些章节的翻译提供了大量帮助。在此, 对所有为本书出版提供了帮助的人们表示诚挚的感谢! 稿件由黄林鹏修改、整理和定稿, 并对最终出现的问题负责, 请将批评意见寄至 lp Huang@sjtu.edu.cn, 不胜感激。

前言

编写本书的动机来源于我在教学实践中常常无法为给定算法给出清晰解析的困惑。与许多教师一样，我发现对一些学生来说，要他们亲自动手解决一些简单问题有困难，而让他们理解给定问题的解决方案同样有困难。我相信，事物的两个方面——创造和解释——是相关而不可分离的。为了完全了解一个问题，考察最后的答案远远不够，我们必须了解问题的求解过程。

本书强调了算法设计的创造性方面，其主要目的是要告诉读者如何去设计一个新的算法。本书描述算法的顺序不是“问题 X 、算法 A 、算法 A' 、程序 P 、程序 P' ”，而是像（但并不总是）“问题 X 、直接明了的问题求解算法、缺点、改进这些缺点的困难、（可能包含一些错误的）好的算法、进一步的改进、分析以及其他方法和算法的关系”。本书的目标不是给出一个容易转换为程序代码的算法，而是希望读者理解算法的原理。算法因此被解释为创造过程而不是最终产品。我们讲授算法的目标不仅是说明如何求解特定的问题，还包括传授如何求解未来将产生或遇到的新问题的技术。可以说，讲授算法设计的思维过程与讲授问题求解细节是同样重要的。

为了进一步帮助读者构建算法创造的思维过程，本书在算法设计过程中使用的是一种“老兵新传”式的方法学。该方法学除了涵盖许多已知的算法设计技术外，还是一个优雅的、能深入解释算法设计的直觉的框架。然而，该方法学并不能遍及算法设计所有可能的方法，我们也不打算专用该方法学于算法设计过程。该方法学的核心在于使用归纳法进行数学定理证明的智能过程与组合算法设计过程之间的相似性。尽管这两个过程的目的不同、取得结果的类型有异，但它们的相似程度却比表面上看起来更多。此相似性也已被许多人注意。本书的新颖性在于对该相似性所进行的深度发掘，并将说明该相似性除了涵盖许多已知的算法设计技术外，还可以在相当程度上帮助算法的创建。对于该方法学，本书第1章先做了简单的介绍，然后在第5章中进行形式化讨论。

考虑下面的类比。你来到一座陌生的城市，租了一辆车，现在想知道旅馆的方向。这时如果别人喋喋不休地告诉你城市的历史、布局 and 交通状况，你肯定会觉得非常厌烦。你所需要的其实是诸如“向前开、过两个街区、右转、再直着向前开三英里”这样的信息。然而，如果计划在这座城市里生活较长的时间，那么你的看法就会有所改变。你可能希望在另一个方向上行驶一段时间（假定能找到一个人告诉你方向），慢慢地你将对城市有更多的了解。虽然本书也包含了许多问题的求解和解析，但它不是一本告诉你所有具体目标其方位的书籍，主要强调的是一般性的原理和方法。本书的学习具有挑战性，它需要读者的深入思考，而我相信额外的努力是值得的。

在许多不同领域，如数学、统计学、细胞生物学和工程学科，高效非数值算法的设计已变得非常重要，本书可作为算法和非数值计算的导论性书籍。许多专业人士，甚至那些未曾和计算机有过紧密接触的科学家，会认为程序设计是一种低级且无需智能的活动。这种看法将导致简单、低效、价值不高的问题解决方案，而忽视优雅、高效算法的存在。本书的目的就是要向读者说明算法设计是一门既重要又雅致的学问。

本书是自包含的，大部分内容很直观，技术性细节尽量保持到最少或者与讨论的主题内容分离，特别是我们将算法实现细节和算法思想尽可能分离，书中许多算法的设计都强调了这一原则。本书并不是以哪些内容必须掌握和牢记这样的方式来设计的，而是通过思想、例子、反例、修改、

改进等一系列过程加以阐述。在大多数算法描述内容的后面给出了算法的伪代码，每章都包含习题和相关的文献目录。大部分章节后面的习题可分为两类：操练型和创造型。操练型习题的目的在于检查读者对于书中特定例子和算法的理解，创造型习题的目的在于检查读者应用相应章节所介绍的特定算法和技术于新问题求解的能力。本书最后给出了一些精选习题（习题编号加上了下划线）的答案。

本书组织如下。第 1 章到第 4 章的内容是介绍性的。第 2 章涉及数学归纳法，就我看来，数学归纳法对于算法设计是非常重要的，熟悉归纳法对于算法设计来说有莫大的帮助。遗憾的是计算机科学系的学生对于归纳证明不是十分了解，因此对一些学生来说，第 2 章的内容可能比较困难，建议他们在第一次阅读本书的时候跳过那些有难度的例子，待适当时候再回过头来研究。第 3 章涉及算法分析，描述了算法分析的过程，并给出对本书所讨论的算法进行简单分析所需要的工具。第 4 章介绍了数据结构，那些对数据结构熟悉且具备一定数学基础的同学可以直接跳到第 5 章（但阅读一下也是有益的）学习。第 5 章提出了与归纳证明进行类比的算法设计思想，并给出一些简单的例子，描述了算法设计过程。如果你没有太多时间，只打算阅读本书的一章，则请直接阅读第 5 章。

一般来说，有关算法的书籍有两种不同的组织结构，一种是按照领域进行分类的，如分成图算法、几何算法等；另一种是按照算法设计技术进行分类的。虽然本书强调的是算法设计技术，但我选择了前者。本书第 6 章到第 9 章分别给出了下面 4 个领域的算法：序列和集合的算法（如排序、序列比较、匹配等）、图算法（如生成树、最短路径、匹配等）几何算法（如凸包和交集问题等）、代数和数值算法（如矩阵乘法、快速傅里叶变换等）。这种组织结构不但清晰且易于理解。

第 10 章涉及归约或约简。尽管归约的例子也出现在本书前面其他章节之中，但该主题不但独特而且重要，因此有必要作为一章的主题。第 10 章的内容也是第 11 章的序幕，后者涉及 NP 完全问题。复杂度理论已经成为算法的基本组成部分，任何算法设计者都应该了解 NP 完全问题以及证明相关性质的技术。第 12 章介绍了并行算法，本章包含不同并行计算模型上的几个有意义的算法。

本书内容或许不能在一个学期之内讲授完毕，因此对于教师来说就有多种不同的选择。开设的第一门算法设计课程可在某种深度上涉及第 3 章、第 5 章、第 6 章、第 7 章和第 8 章的部分内容，这些章节的高级部分以及第 9 章、第 10 章、第 11 章和第 12 章的内容是可选的，它们也可作为高级算法课程的基础。

致谢

首先要感谢我的妻子 Rachel 对于本书的帮助，对本书所基于的方法学她给出了指导性的建议，对于本书的内容她也给出了许多意见和忠告，可以说没有她就没有本书。

特别感谢 Jan Van Leuwen 对本书大部分章节所给出的全面的评论。这些详细的意见、建议和修正方案为本书增色不少。感谢 Eric Bach、Darrah Chavey、Kirk Pruhs 和 Sun Wu，他们在阅读了本书的部分手稿后给出了建设性的意见，感谢 Guy T. Almes（赖斯大学）、Agnes H. Chan（美国东北大学）、Dan Gusfield（加州大学 Davis 分校）、David Harel（以色列魏兹曼学院）、Daniel Hirschberg（加州大学 Irvine 分校）、Jefferey H. Kingston（爱荷华大学）、Victor Klee（华盛顿大学）、Charles Martel（加州大学 Davis 分校）、Michael J. Quinn（新罕布什尔大学）和 Diane M. Spresser（詹姆士·麦迪逊大学）。

感谢 Addison-Wesley 出版公司的编辑没有施加给我通常作者所经历的有关书籍出版的恐怖经历，他们乐于助人，并且充满耐心。特别感谢主管 Bette Aaronson，本书的编辑 Jim DeWolf 和 Lyn Dupre，他们在写作过程中引导我，尽管知道有更好的方法，但他们仍然允许我按照自己的方式行事。在这里，我要感谢美国自然科学基金总统青年研究奖的资助以及 AT&T、数字设备公司、惠普公司和 Tektronix 公司的资助。

然而，必须强调的是，最后的手稿是由排版人员（即作者自己）准备的，他决定了这里所列举的许多人员对本书的评论和建议的取舍，并且将对最后的结果负责。

乌迪·曼博
于美国亚利桑那州图森市

目 录

第 1 章 引论	1
第 2 章 数学归纳法	6
2.1 引言	6
2.2 三个简单的例子	7
2.3 平面内区域的计数	8
2.4 简单的着色问题	10
2.5 复杂一些的加法题	10
2.6 一个简单的不等式	11
2.7 欧拉公式	12
2.8 图论中的一个问题	13
2.9 格雷码	14
2.10 在图上寻找无重边的路	16
2.11 数学平均数和几何平均数定理	17
2.12 循环不变量：将十进制数转换为二进制数	19
2.13 常见的错误	20
2.14 小结	21
第 3 章 算法分析	27
3.1 引言	27
3.2 符号 O	28
3.3 时间与空间复杂度	30
3.4 求和	31
3.5 递推关系	33
3.5.1 巧妙地猜测	34
3.5.2 分治关系	36
3.5.3 涉及全部历史的递推关系	37
3.6 一些有用的证明论据	39
3.7 小结	40
第 4 章 数据结构简介	44
4.1 引言	44
4.2 基本数据结构	44
4.2.1 元素	44
4.2.2 数组	45
4.2.3 记录	45

4.2.4	链表	46
4.3	树	47
4.3.1	树的表示	48
4.3.2	堆	49
4.3.3	二叉搜索树	51
4.3.4	AVL 树	54
4.4	散列	56
4.5	合并—查找问题	58
4.6	图	60
4.7	小结	61
第 5 章	基于归纳的算法设计	66
5.1	引言	66
5.2	多项式求值	66
5.3	最大导出子图	68
5.4	寻找一对一映射	69
5.5	社会名流问题	71
5.6	分治算法：轮廓问题	73
5.7	在二叉树中计算平衡因子	75
5.8	寻找最大连续子序列	76
5.9	增强归纳假设	77
5.10	动态规划：背包问题	77
5.11	常见的错误	80
5.12	小结	80
第 6 章	序列和集合的算法	85
6.1	引言	85
6.2	二叉搜索的几种形式	85
6.2.1	纯二叉搜索	85
6.2.2	循环序列的二叉搜索	86
6.2.3	二叉搜索特殊下标	86
6.2.4	二叉搜索长度未知的序列	88
6.2.5	重叠子序列问题	88
6.2.6	解方程	89
6.3	内插搜索	89
6.4	排序	90
6.4.1	桶排序和基数排序	90
6.4.2	插入排序和选择排序	92
6.4.3	归并排序	93
6.4.4	快速排序	93
6.4.5	堆排序	98

6.4.6	排序问题的下界	100
6.5	顺序统计	102
6.5.1	最大数和最小数	102
6.5.2	查找第 k 小的数	102
6.6	数据压缩	103
6.7	串匹配	105
6.8	序列比较	110
6.9	概率算法	112
6.9.1	随机数	113
6.9.2	着色问题	114
6.9.3	将拉斯维加斯算法变换成确定性算法	114
6.10	查找众数	116
6.11	三个展现有趣证明方法的问题	118
6.11.1	最长递增序列	118
6.11.2	查找集合中两个最大的元素	119
6.11.3	计算多重集合的模	121
6.12	小结	122
第 7 章	图算法	130
7.1	引言	130
7.2	欧拉图	131
7.3	图的遍历	133
7.3.1	深度优先搜索	133
7.3.2	广度优先搜索	139
7.4	拓扑排序	140
7.5	单源最短路径	142
7.6	最小代价生成树	146
7.7	全部最短路径	150
7.8	传递闭包	151
7.9	图的分解	153
7.9.1	双连通分支	153
7.9.2	强连通分支	159
7.9.3	利用图分解的例子	164
7.10	匹配	166
7.10.1	非常稠密图中的完美匹配	166
7.10.2	偶图匹配	167
7.11	网络流量	169
7.12	哈密尔顿旅行	173
7.12.1	反向归纳	173
7.12.2	在非常稠密图中找哈密尔顿回路	173
7.13	小结	174

第 8 章 几何算法	188
8.1 引言	188
8.2 判定点是否在多边形内部	189
8.3 构造简单多边形	191
8.4 凸包	193
8.4.1 直接方法	193
8.4.2 礼品包裹算法	194
8.4.3 Graham 扫描算法	195
8.5 最近点对	197
8.6 水平线段和竖直线段的交点	200
8.7 小结	203
第 9 章 代数和数值算法	207
9.1 引言	207
9.2 求幂运算	207
9.3 欧几里得算法	210
9.4 多项式乘法	211
9.5 矩阵乘法	212
9.5.1 Winograd 算法	212
9.5.2 Strassen 算法	213
9.5.3 布尔矩阵	215
9.6 快速傅里叶变换	218
9.7 小结	224
第 10 章 归约	228
10.1 引言	228
10.2 归约的例子	229
10.2.1 简单字符串匹配问题	229
10.2.2 特殊代表集	229
10.2.3 关于序列比较的归约	230
10.2.4 在无向图中寻找三角形	231
10.3 有关线性规划的归约	232
10.3.1 概述与定义	232
10.3.2 归约到线性规划的例子	233
10.4 下界的归约	235
10.4.1 寻找简单多边形算法复杂度的下界	235
10.4.2 关于矩阵的简单归约	236
10.5 常见的错误	237
10.6 小结	238
第 11 章 NP 完全问题	242
11.1 引言	242

11.2	多项式时间归约	242
11.3	非确定性和 Cook 定理	244
11.4	NP 完全性的证明例子	246
11.4.1	顶点覆盖问题	246
11.4.2	支配集问题	247
11.4.3	3SAT 问题	248
11.4.4	团问题	249
11.4.5	3 着色问题	250
11.4.6	一般经验	251
11.4.7	更多的 NP 完全问题	252
11.5	处理 NP 完全问题的技术	253
11.5.1	回溯法和分枝限界法	254
11.5.2	确保性能的近似算法	257
11.6	小结	261
第 12 章	并行算法	266
12.1	引言	266
12.2	并行计算模型	267
12.3	共享存储器算法	268
12.3.1	并行加	268
12.3.2	寻找最大数的算法	269
12.3.3	并行前缀问题	271
12.3.4	在链表中查寻秩	273
12.3.5	欧拉遍历技术	274
12.4	互连网络上的算法	275
12.4.1	阵列上的排序	276
12.4.2	排序网络	279
12.4.3	在树中查找第 k 个最小元素	280
12.4.4	网孔上的矩阵乘法	283
12.4.5	超立方体中的路由	284
12.5	脉动计算	286
12.5.1	矩阵与向量相乘	286
12.5.2	卷积问题	287
12.5.3	序列的比较	288
12.6	小结	290
部分习题答案		296
参考文献		317

第1章 引 论

“构筑”过程确实蕴涵巨大的重要性，有人宣称由此可以看到科学发展的充分必要条件。但我认为，必要性，是无疑的，但并不是充分的。一个成功而不浪费心智的构筑，由于要作为高层事物的基石，首先应该具备某种形式的协调性，而我们看到的，不应该是一些东西的简单堆砌，而是更多。

——亨利·庞加莱（1902）

在《韦氏大学词典（第九版）》中，算法的解释是“求解数学问题（如寻找最大公约数）的一个过程，该过程步骤有限，通常还涉及重复的操作；广义地说，算法是按部就班解决一个问题或完成某个目标的过程。本书取广义的算法来定义。算法设计是一个古老的研究领域。自古以来，人们总是对发现更好的目标求解方法充满兴趣，不论是取火、建造金字塔还是对邮件进行排序。而计算机算法的研究当然是一个新的领域。一些计算机算法采用的方法早在计算机发明之前就存在，但大多数计算机算法的设计需要新的方法和技术。首先，告诉计算机诸如“察看小山，如果发现敌情就拉响警报”是不够的。一台计算机必须了解“察看”的确切含义，知道如何识别敌情，懂得如何拉响警报（基于技术原因，拉响警报是最容易的）。一台计算机可接受的指令应当是定义明确、长度有限的基本操作序列。将通常的命令转换为计算机可以理解的指令是一个困难的过程，而该过程就是编程，目前有成百万的程序员正在不同层次上进行编程。

计算机上的编程，所需要的不仅仅是将那些为人所理解的命令转换为计算机可以理解的语言。在大多数情况下，程序员必须设计出完全崭新的算法来求解问题。只学习与计算机交谈所用的怪异语言会使编程变得困难，因为只有计算机才知道你说了什么。计算机不仅能以极快的速度执行先前由人完成的操作，它还可以做得更多。计算机能处理数十亿、数万亿比特单位的信息，能在一秒内完成数百万条基本指令。在这个数量级上进行算法设计是一种崭新的实践，有很多方面会与我们的直觉相反，因为我们通常只对自己能感知的事物进行思考。遗憾的是，一些能很好解决小问题的程序在处理大问题时就变得很糟。因此当进行大规模计算时不要忽视算法的复杂度和有效性。

问题的另一个方面是，我们在日常生活中所执行的算法一般不太复杂，执行的次数也不太频繁，因为回报非常低，通常不值得耗费太多精力来开发完美的算法。例如，在日常生活中，从超市采购回家，打开杂货袋放置食品的过程就可能不是最有效的，有效的过程应该考虑杂货袋中的食品以及橱柜的结构，但很少有人会去想这件事，更不会有人为它去专门设计算法。反之，进行大规模商业包装和拆装就必须有一种好的方法。另一个例子是修剪草坪，可以考虑如何使来回次数最少、修剪时间最短，或者到垃圾堆的往返距离最短。除非你非常讨厌修剪草坪，否则就不会花一个小时的时间来思考如何使修剪时间缩短几分钟这样的问题。但另一方面，计算机可以处理非常复杂的任务，而且同一任务往往必须执行多次，因此值得花费时间来设计有效的方法，就算最终得到的方法更加复杂或难以理解也无妨，因为潜在的回报是巨大的。（当然我们也不要太乐观，有时花费数小时的程序设计时间带来的仅仅是节约几秒钟的计算机运算时间。）

对与直觉相悖的大型问题求解算法的需要以及了解这些算法的复杂度是学习算法这门学科的动力所在。首先,必须认识到直观的方法并不总是最好的,寻找更好的方法有时非常重要,为此,程序员必须学习新的方法和技术。本书给出了多种算法的说明。然而,就像要成为一个好棋手仅仅靠记住许多棋局是不够的一样,就算你学习了大量的算法也不足以应付各种情况,因此我们必须了解算法背后的原理,必须知道如何应用它们,当然更重要的是,何时应用它们。

算法的设计和实现类似于房屋的设计和建筑^①。我们先从最基本的概念出发考虑房子的修建。设计师的工作是给出满足要求的规划;工程师的工作是确认规划是正确且可行的(以保证房子不会在短时间内倒塌);建设工人的工作是按照规划建筑房子。当然,在所有阶段,都要考虑造价并进行分析。每个阶段的工作是不同的,但它们又有所联系甚至交织在一起。算法的设计也如此,先考虑基本的思想和方法,然后做出规划。我们必须证明规划是可行的并且代价是可接受的。最后的工作是在具体的计算机上实现。简单地讲,可将过程分为4步:设计、正确性证明、分析和实现。再次,每步都不同但又相关,每步都不能不顾及其他而在真空中完成。当然,我们很少能按照线性次序走过历程。在构造算法的每个阶段都可能出现困难。由此可能需要修改设计,还可能再次进行可行性证明、调整代价以及改变算法实现。

本书主要注重算法的第一个阶段,即算法的设计。按照类比,本书合适的名字应该是算法体系结构。然而,体系结构在计算机科学中有它自己的含义,使用这个名字容易造成混淆。当然本书也不会忽略算法的其他方面,书中在对大多数算法描述之后紧接着是算法的正确性证明、算法的复杂度分析以及对算法实现的讨论,有些详细,有些简单,但主要还是围绕算法设计展开的。

要成为一个好的体系结构师并设计出崭新的算法,仅靠学习许多算法是不够的,还必须了解算法设计背后的原理。本书中,我们使用一个与以往不同的方式来解释算法。我们认为学习如何创造事物的最好方式就是尝试去创造它,因此本书首先引导读者给出其自己的算法;其次,更重要的是,我们将遵循有助于培养读者创新思想的方法学。该方法学(由Manber[1988]首先提出)是一种能对算法进行深入解释的直观的优雅框架。该方法学包括许多不同的方法及变体。在许多不同的方法间做出选择并且应用它们更需要系统的方法学。该方法学并没有完全覆盖算法设计的所有可能方式,但它对于本书中大多数算法的设计很有帮助。

该方法学基于数学归纳法,核心是对证明数学定理的智力过程与设计组合算法的过程进行类比。数学归纳法其原理的主要思想是命题无须从什么都没有开始证明,可以通过先证明对于一个小的基本实例集命题是正确的,然后由“若对相对较小的命题实例成立则可导出较大的命题实例成立”的方式,来证明对所有实例来说命题都是成立的。将该原理转化为算法设计则提示了一种侧重于如何将简单问题的解扩充为较大问题的解的方法。给定一个问题,如果能说明如何应用类似问题(但参数较小)的解决方法进行问题求解,那么任务就完成了。因此它的基本思想是如何扩充一个解决方案而不是从零开始进行构造。正如在本书后面章节中将看到的那样,对于这个问题有许多方法,由此产生了多种不同的算法设计技术。

数学归纳法在这里主要用于高层次算法的设计和解释,我们不对方法本身进行形式化和公理化的研究。但有很多人做出了许多贡献,包括非常杰出的Dijkstra[1976]、Manna[1980]、Gries[1981]、Dershowitz[1983]和Paull[1988]等。本书是对以上书籍的补充,主要用于教学。显然,对一个事物解释得更清晰,它就能被理解得更通透。在证明技术上,我们讨论了如何加强归纳假设,如何更精明地选择归纳序列,如何应用双重归纳和反向归纳等。我们的方法的重要性有两个方面:其一,

① 该比喻是受Tracy Kidder所著的两本书的启发,这两本书分别是“The Soul of a New Machine”(Little Brown, 1981)和“House”(Houghton Mifflin, 1985)。

我们收集了许多似乎无关的算法设计技术并归类于一个框架之下；其二，将已知的数学证明技术应用于算法设计。后者特别重要，因为它为已在另一学科中开发多年的强有力技术的使用打开了一扇门。

这种方法的一个缺点在于它不是一种通用的方法，并不是所有程序都可以或者应该按照归纳的思想进行设计。然而，在算法设计中归纳原理非常普遍，使得我们可以侧重对其应用的研究。对于其他原理，本书也不会忽视。对新出现的一种方法学最通常的批评是它可以为已有的东西给出一个有意义的解释，但它无助于事物的创造。这是一个实在的批评，因为只有未来才能告诉你一种方法学的有效性和广泛应用的程度。我坚信归纳法不仅仅是一种算法解释工具，它还是理解算法的必要途径。就个人来说，虽然在设计算法时我也有不遵循该方法学的经历，但碰到了两个例子，我发现它是非常有用的，它能让我快速开发出算法（Manber and McVoy [1988]、Manber and Myers[1989]）。

算法描述记法

除了通过对算法开发的创造性过程进行算法描述外，本书还包含许多算法的伪代码。包含程序代码的目的是加强算法的形象描述。我们并未对程序做太多优化，也不推荐直接使用它们。在许多情况下，我们有意不包括程序的优化版本，因为这可能会引入额外的复杂度，使读者的注意力离开算法的主要思想。有时也没有详细解释如何将算法思想转换为程序，这种转换有时很显然有时却不是。本书的侧重点，正如前面所提到的那样，在于算法设计的原理。

在大多数情况下，我们使用类 Pascal 语言（有时甚至是纯 Pascal 语言）。在许多情况下，基于可读性，Pascal 代码会包含一些高级描述。要注意的一个例外是对标识 Pascal 块的 **begin** 和 **end** 语句的修改，我们仅在程序的开始和结束时用到它们，块之间使用缩排进行标识。这种使用约定在不引起混淆的情况下将大大节省空间。在数据类型清晰的情况下我们通常不准备给出变量的详细说明和数据类型（比如说 G 是一个图， T 是一棵树等）。

习题

对于那些编号包括下划线的习题，本书后面有其解答，标记为星号的习题，则可能会比其他习题稍难。

本章中的习题无须任何与算法相关的先期知识，它们是一些有特定输入的相对简单的问题，这里要求读者手工给出问题的解答。习题的目的是说明处理大规模问题的困难。换言之，这些习题的目标之一是说明直接方法将受挫于此类问题。这里给出的问题将在后面的章节中继续讨论。

1.1 在不同的卡片上分别写下数字 1 到 100，打乱后按顺序进行排列。

1.2 在不同的卡片上分别写下下列 100 个数字并排序。考虑这道习题与上一道习题的不同点。

```
32918 21192 11923 4233 88231 8312 11 72 971 8234 22238 49283 3295
29347 3102 32883 20938 2930 16 823 9234 9236 29372 2218 9222 21202
83721 9238 8221 30234 93920 81102 1011 18152 2831 29133 9229 10039
9235 48395 2832 37927 73492 8402 48201 38024 2800 32155 2273 82930
* 2221 3841 311 3022 38099 29920 28349 74212 7011 1823 903 2991 9335
29123 28910 29281 3772 20012 70458 30572 38013 72032 28001 83835
3017 92626 73825 29263 2017 262 8362 77302 8593 3826 9374 2001
83261 48402 4845 79794 27271 39992 22836 444 2937 37201 37322
49472 11329 2253
```

1.3 考虑由下列数所组成的表。你的工作是删去尽可能少的数使得留下来的数以升序排列。例