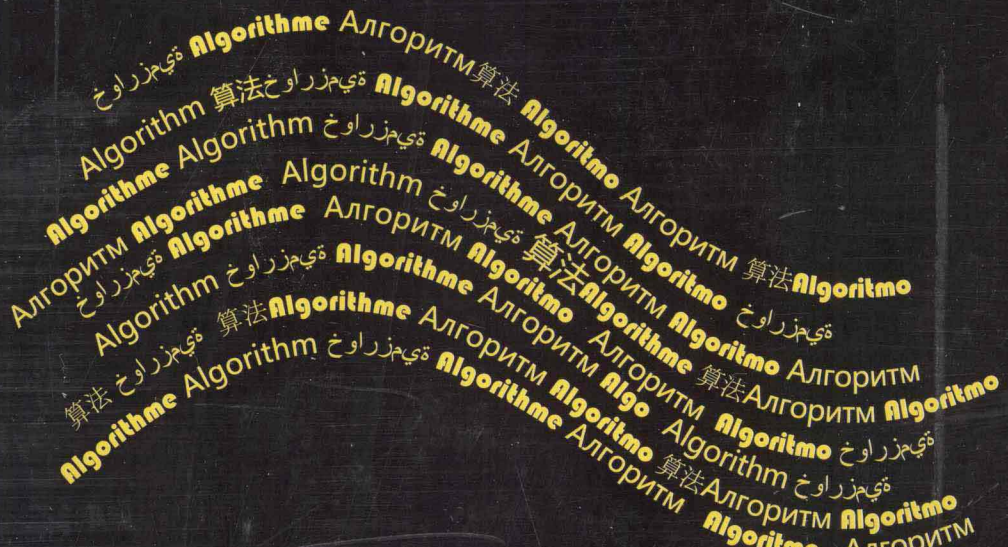
PEARSON

Introduction to the Design and Analysis of Algorithms Third Edition

算法设计与分析基础

(第3版 影印版)

(美) Anany Levitin 著



清华大学出版社

PEARSON

算法设计与分析基础

(第3版 影印版)

(美) Anany Levitin 著

清华大学出版社
北京

Original edition, entitled: INTRODUCTION TO THE DESIGN AND ANALYSIS OF ALGORITHMS, 3E, 9780132316811 by ANANY LEVITIN, Published by Pearson Education, Inc., publishing as Addison-Wesley, Copyright © 2012 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.
China edition published by PEARSON EDUCATION ASIA LTD., and TSINGHUA UNIVERSITY PRESS LIMITED Copyright © 2013.
This edition is manufactured in the People's Republic of China, and is authorized for sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macau SAR).
本书影印版由 Pearson Education, Inc.授权给清华大学出版社出版发行。

Authorized for sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macau SAR).

仅限于中华人民共和国境内(不包括中国香港、澳门特别行政区和中国台湾地区)销售。

北京市版权局著作权合同登记号　图字：01-2013-3028

本书封面贴有 Pearson Education (培生教育出版集团) 激光防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目(CIP)数据

算法设计与分析基础 (第 3 版 影印版) = Introduction to the Design and Analysis of Algorithms, Third Edition / (美)莱维丁(Levitin, A.)著. --影印本. --北京：清华大学出版社，2013
ISBN 978-7-302-31185-0

Ⅰ. ①算… Ⅱ. ①莱… Ⅲ. ①电子计算机—算法设计—英文 ②电子计算机—算法分析—英文 Ⅳ. ①TP301.6

中国版本图书馆 CIP 数据核字(2013)第 002529 号

责任编辑：文开琪　汤涌涛
封面设计：杨玉兰
责任校对：周剑云
责任印制：刘海龙

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185mm×230mm　　**印　　张**：37.25　　**字　　数**：1047 千字
版　　次：2013 年 5 月第 1 版　　**印　　次**：2013 年 5 月第1 次印刷
印　　数：1～4000
定　　价：79.00 元

产品编号：050764-01

前　言

“一个人接受科技教育时所能获得的最珍贵的收获，是那些能够受用终身的通用智能工具[①]。”

——George Forsythe，*What to do till the computer scientist comes*，1968

无论是计算科学还是计算实践，算法都在其中扮演着重要角色。因此，这门学科中出现了大量的教材。它们在介绍算法的时候，基本上都选择了以下两种方案中的一种。第一种方案是按照问题的类型对算法分类。这类教材安排了不同的章节分别讨论排序、查找、图等算法。这种做法的优点是，对于解决同一问题的不同算法，它能够立即比较这些算法的效率。其缺点在于，由于过于强调问题的类型，它忽略了对算法设计技术的讨论。

第二种方案围绕着算法设计技术来组织章节。在这种结构中，即使算法来自于不同的计算领域，如果它们采用了相同的设计技术，就会被编成一组。从各方(例如[BaY95])获得的信心使我相信，这种结构更适合于算法设计与分析的基础课程。强调算法设计技术有三个主要原因。第一，学生们在解决新问题时，可以运用这些技术设计出新的算法。从实用的角度看，这使得学习算法设计技术颇有价值。第二，学生们会试图按照算法的内在设计方法对已知的众多算法进行分类。计算机科学教育的一个主要目的，就是让学生们知道如何发掘不同应用领域的算法间的共性。毕竟，每门学科都会倾向于把它的重要主题归纳为几个甚至一个规则。第三，依我看来，算法设计技术作为问题求解的一般性策略，在解决计算机领域以外的问题时，也能发挥相当大的作用。

遗憾的是，无论是从理论还是从教学的角度，传统的算法设计技术分类法都存在一些严重的缺陷。其中最显著的缺陷就是无法对许多重要的算法进行分类。由于这种局限性，这些书的作者不得不在按照设计技术进行分类的同时，另外增加一些章节来讨论特殊的问题类型。但这种改变将导致课程缺乏一致性，而且很可能会使学生感到迷惑。

算法设计技术的新分类法

传统算法设计技术分类法的缺陷令我感到失望，它激发我开发一套新的分类法[Lev99]，这套分类法就是本书的基础。以下是这套新分类法的几个主要优势。

- 新分类法比传统分类法更容易理解。它包含的某些设计策略，例如蛮力法、减治法、变治法、时空权衡和迭代改进，几乎从不曾被看作重要的设计范例。
- 新分类法很自然地覆盖了许多传统方法无法分类的经典算法(欧几里得算法、堆排序、查找树、散列法、拓扑排序、高斯消去法、霍纳法则等，不胜枚举)。所以，新分类法能够以一种连贯的、一致的方式表达这些经典算法的标准内容。
- 新分类法很自然地容纳了某些设计技术的重要变种(例如，它能涵盖减治法的 3 个变种

① 译注：George Forsythe 认为，在这些工具当中，最重要的三项依次是自然语言、数学和计算机科学。

和变治法的3个变种)。

- 在分析算法效率时，新分类法与分析方法结合得更好(参见附录B)。

设计技术作为问题求解的一般性策略

在本书中，主要将设计技术应用于计算机科学中的经典问题(这里唯一的创新是引入了一些数值算法的内容，我们也是用同样的通用框架来表述这些算法的)。但把这些设计技术看作问题求解的一般性工具时，它们的应用就不仅限于传统的计算问题和数学问题了。有两个因素令这一点变得尤其重要。第一，越来越多的计算类应用超越了它们的传统领域，并且有足够的理由使人相信，这种趋势会愈演愈烈。第二，人们渐渐认识到，提高学生们的问题求解能力是高等教育的一个主要目标。为了满足这个目标，在计算机科学课程体系中安排一门算法设计和分析课程是非常合适的，因为它会告诉学生如何应用一些特定的策略来解决问题。

虽然我并不建议将算法设计和分析课程变成一门教授一般性问题求解方法的课程，但我的确认为，我们不应错过算法设计和分析课程提供的这样一个独一无二的机会。为了这个目标，本书包含了一些和谜题相关的应用。虽然利用谜题来教授算法课程绝不是我的创新，但本书打算通过引进一些全新的谜题来系统地实现这个思路。

如何使用本书

我的目标是写一本既不泛泛而谈，又可供学生们独立阅读的教材。为了实现这个目标，本书做了如下努力。

- 根据George Forsythe的观点(参见引言)，我试图着重强调那些隐藏在算法设计和分析背后的主要思想。在选择特定的算法来阐述这些思想的时候，我并不倾向于涉及大量的算法，而是选择那些最能揭示其内在设计技术或分析方法的算法。幸运的是，大多数经典算法满足了这个要求。
- 第2章主要分析算法的效率，该章将分析非递归算法的方法和分析递归算法的典型方法区别开来。这一章还花了一些篇幅介绍算法经验分析和算法可视化。
- 书中系统地穿插着一些面向读者的提问。其中有些问题是经过精心设计的，而且答案紧随其后，目的是引起读者的注意或引发疑问。其余问题的用意是防止读者走马观花，不能充分理解本书的内容。
- 每一章结束时都会对本章最重要的概念和结论做一个总结。
- 本书包含600多道习题。有些习题是为了给大家练习，另外一些则是为了指出书中正文部分所涉及内容的重要意义，或是为了介绍一些书中没有涉及的算法。有一些习题利用了因特网上的资源。较难的习题数量不多，会在教师用书中用一种特殊的记号标注出来(因为有些学生可能没有勇气做那些标有难度的习题，所以本书没有对习题标注难度)。谜题类的习题用一种特殊的图标做标注。
- 本书所有的习题都附有提示。除了编程练习，习题的详细解法都能够在教师用书中找到，符合条件的教师可以填写书后的教师证明表，发传真到010-62791865，以获得教师用书(也可联系培生公司的当地销售代表，或者访问 *www. pearsonhighered.com/irc*)。本书的任何读者都可以在CS支持网站 *http://cssupport.pearsoncmg.com* 上找到PowerPoint格式的

试读结束：需要全本请在线购买： www.ertongbook.com

幻灯片文件。

第 3 版的变化

第 3 版有若干变化。其中最重要的变化是介绍减治法和分治法的先后顺序。第 3 版会先介绍减治法，后介绍分治法，这样做有以下几个优点。

- 较之分治法，减治法更简单。
- 在求解问题方面，减治法应用更广。
- 这样的编排顺序便于先介绍插入排序，后介绍合并排序和快速排序。
- 数组划分的概念通过选择性问题引入，这次利用 Lomuto 算法的单向扫描来实现，而将 Hoare 划分方法的双向扫描留至后文与快速排序一并介绍。
- 折半查找归入介绍减常量算法的章节。

另一重要变化是重新编排第 8 章关于动态规划的内容，具体如下所述。

- 导述部分的内容是全新的。在前两版中用计算二项式系数的例子来引入动态规划这一重要技术，但在第 3 版中会介绍 3 个基础性示例，这样介绍的效果更好。
- 8.1 节的习题是全新的，包括一些在前两版中没有涉及的流行的应用。
- 第 8 章其他小节的顺序也做了调整，以便达到由浅入深、循序渐进的效果。

此外，还有其他一些变化。增加了不少与本书所述算法相关的应用。遍历图算法不再随减治法介绍，而是纳入蛮力算法和穷举查找的范畴，我认为这样更合理。在介绍生成组合对象的算法时，会新增格雷码算法。对求解最近对问题的分治法会有更深入的探讨。改进的内容包括算法可视化和求解旅行商问题的近似算法，当然参考文献也相应更新。

第 3 版新增约 70 道习题，其中涉及算法谜题和面试问题。

读者所需的知识背景

本书假定读者已经学习了离散数学的标准课程和一门基础性的编程课程。有了这样的知识背景，读者应该能够掌握本书的内容而不会遇到太大的困难。尽管如此，1.4 节、附录 A 和附录 B 仍然对基本的数据结构，必须用到的求和公式和递推关系分别进行了复习和回顾。只有 3 个小节(2.2 节、11.4 节和 12.4 节)会用到一些简单的微积分知识，如果读者缺少必要的微积分知识，完全可以跳过这 3 个涉及微积分的小节，这并不会妨碍对本书其余部分的理解。

进度安排

如果打算开设一门围绕算法设计技术来讲解算法设计和分析理论的基础课程，可以采用本书作为教材。但要想在一个学期内完成该课程，本书涵盖的内容可能过于丰富了。大体上来说，跳过第 3～12 章的部分内容不会影响读者对后面部分的理解。本书的任何一个部分都可以安排学生自学。尤其是 2.6 节和 2.7 节，它们分别介绍了经验分析和算法可视化，这两小节的内容可以结合练习[①]布置给学生。

① 译注：“练习”的原文为“project”，一般应该翻译成“项目”，但国外一般将布置在课后完成的、较大型的、要求实际演练的习题称为 project，国内没有相应的称呼，所以姑且译为“练习”。

下面给出了一种针对一个学期课程的教学计划，这是按照40课时的集中教学来设计的。

课 次	主 题	小 节
1	课程简介	1.1～1.3
2，3	分析框架；O、Θ和Ω符号	2.1，2.2
4	非递归算法的数学分析	2.3
5，6	递归算法的数学分析	2.4，2.5(+附录B)
7	蛮力算法	3.1，3.2(+3.3)
8	穷举查找	3.4
9	深度优先查找和广度优先查找	3.5
10～11	减一算法：插入排序、拓扑排序	4.1，4.2
12	折半查找和其他减常量算法	4.4
13	减变量算法	4.5
14～15	分治法：合并排序、快速排序	5.1～5.2
16	其他分治法示例	5.3、5.4或5.5
16	减变量算法	5.6
17～19	实例化简：预排序、高斯消去法、平衡查找树	6.1～6.3
20	改变表现：堆和堆排序 或者霍纳法则和二进制幂	6.4或 6.5
21	问题化简	6.6
22～24	时空权衡：串匹配、散列法、B树	7.2～7.4
25～27	动态规划算法	8.1～8.4(选3节)
28～30	贪婪算法：Prim算法、Kruskal算法、Dijkstra算法、哈夫曼算法	9.1～9.4
31～33	迭代改进算法	10.1～10.4(选3节)
34	下界的参数	11.1
35	决策树	11.2
36	*P*、*NP*和*NP*完全问题	11.3
37	数值算法	11.4(+12.4)
38	回溯法	12.1
39	分支界限法	12.2
40	*NP*困难问题的近似算法	12.3

Anany Levitin
anany.levitin@villanova.edu

New to the Third Edition

- Reordering of chapters to introduce decrease-and-conquer before divide-and-conquer
- Restructuring of chapter 8 on dynamic programming, including all new introductory material and new exercises focusing on well-known applications
- More coverage of the applications of the algorithms discussed
- Reordering of select sections throughout the book to achieve a better alignment of specific algorithms and general algorithm design techniques
- Addition of the Lomuto partition and Gray code algorithms
- Seventy new problems added to the end-of-chapter exercises, including algorithmic puzzles and questions asked during job interviews

Preface

The most valuable acquisitions in a scientific or technical education are the general-purpose mental tools which remain serviceable for a life-time.

—George Forsythe, "What to do till the computer scientist comes." (1968)

Algorithms play the central role both in the science and practice of computing. Recognition of this fact has led to the appearance of a considerable number of textbooks on the subject. By and large, they follow one of two alternatives in presenting algorithms. One classifies algorithms according to a problem type. Such a book would have separate chapters on algorithms for sorting, searching, graphs, and so on. The advantage of this approach is that it allows an immediate comparison of, say, the efficiency of different algorithms for the same problem. The drawback of this approach is that it emphasizes problem types at the expense of algorithm design techniques.

The second alternative organizes the presentation around algorithm design techniques. In this organization, algorithms from different areas of computing are grouped together if they have the same design approach. I share the belief of many (e.g., [BaY95]) that this organization is more appropriate for a basic course on the design and analysis of algorithms. There are three principal reasons for emphasis on algorithm design techniques. First, these techniques provide a student with tools for designing algorithms for new problems. This makes learning algorithm design techniques a very valuable endeavor from a practical standpoint. Second, they seek to classify multitudes of known algorithms according to an underlying design idea. Learning to see such commonality among algorithms from different application areas should be a major goal of computer science education. After all, every science considers classification of its principal subject as a major if not the central point of its discipline. Third, in my opinion, algorithm design techniques have utility as general problem solving strategies, applicable to problems beyond computing.

Unfortunately, the traditional classification of algorithm design techniques has several serious shortcomings, from both theoretical and educational points of view. The most significant of these shortcomings is the failure to classify many important algorithms. This limitation has forced the authors of other textbooks to depart from the design technique organization and to include chapters dealing with specific problem types. Such a switch leads to a loss of course coherence and almost unavoidably creates a confusion in students' minds.

New taxonomy of algorithm design techniques

My frustration with the shortcomings of the traditional classification of algorithm design techniques has motivated me to develop a new taxonomy of them [Lev99], which is the basis of this book. Here are the principal advantages of the new taxonomy:

- The new taxonomy is more comprehensive than the traditional one. It includes several strategies—brute-force, decrease-and-conquer, transform-and-conquer, space and time trade-offs, and iterative improvement—that are rarely if ever recognized as important design paradigms.
- The new taxonomy covers naturally many classic algorithms (Euclid's algorithm, heapsort, search trees, hashing, topological sorting, Gaussian elimination, Horner's rule—to name a few) that the traditional taxonomy cannot classify. As a result, the new taxonomy makes it possible to present the standard body of classic algorithms in a unified and coherent fashion.
- It naturally accommodates the existence of important varieties of several design techniques. For example, it recognizes three variations of decrease-and-conquer and three variations of transform-and-conquer.
- It is better aligned with analytical methods for the efficiency analysis (see Appendix B).

Design techniques as general problem solving strategies

Most applications of the design techniques in the book are to classic problems of computer science. (The only innovation here is an inclusion of some material on numerical algorithms, which are covered within the same general framework.) But these design techniques can be considered general problem solving tools, whose applications are not limited to traditional computing and mathematical problems. Two factors make this point particularly important. First, more and more computing applications go beyond the traditional domain, and there are reasons to believe that this trend will strengthen in the future. Second, developing students' problem solving skills has come to be recognized as a major goal of college education. Among all the courses in a computer science curriculum, a course on the design and analysis of algorithms is uniquely suitable for this task because it can offer a student specific strategies for solving problems.

I am not proposing that a course on the design and analysis of algorithms should become a course on general problem solving. But I do believe that the

unique opportunity provided by studying the design and analysis of algorithms should not be missed. Toward this goal, the book includes applications to puzzles and puzzle-like games. Although using puzzles in teaching algorithms is certainly not a new idea, the book tries to do this systematically by going well beyond a few standard examples.

Textbook pedagogy

My goal was to write a text that would not trivialize the subject but would still be readable by most students on their own. Here are some of the things done toward this objective.

- Sharing the opinion of George Forsythe expressed in the epigraph, I have sought to stress major ideas underlying the design and analysis of algorithms. In choosing specific algorithms to illustrate these ideas, I limited the number of covered algorithms to those that demonstrate an underlying design technique or an analysis method most clearly. Fortunately, most classic algorithms satisfy this criterion.
- In Chapter 2, which is devoted to efficiency analysis, the methods used for analyzing nonrecursive algorithms are separated from those typically used for analyzing recursive algorithms. The chapter also includes sections devoted to empirical analysis and algorithm visualization.
- The narrative is systematically interrupted by questions to the reader. Some of them are asked rhetorically, in anticipation of a concern or doubt, and are answered immediately. The goal of the others is to prevent the reader from drifting through the text without a satisfactory level of comprehension.
- Each chapter ends with a summary recapping the most important concepts and results discussed in the chapter.
- The book contains over 600 exercises. Some of them are drills; others make important points about the material covered in the body of the text or introduce algorithms not covered there at all. A few exercises take advantage of Internet resources. More difficult problems—there are not many of them—are marked by special symbols in the Instructor's Manual. (Because marking problems as difficult may discourage some students from trying to tackle them, problems are not marked in the book itself.) Puzzles, games, and puzzle-like questions are marked in the exercises with a special icon.
- The book provides hints to all the exercises. Detailed solutions, except for programming projects, are provided in the Instructor's Manual, available to qualified adopters through Pearson's Instructor Resource Center. (Please contact your local Pearson sales representative or go to www.pearsonhighered.com/irc to access this material.) Slides in PowerPoint are available to all readers of this book via anonymous ftp at the CS Support site: http://cssupport.pearsoncmg.com/.

试读结束：需要全本请在线购买： www.ertongbook.com

Changes for the third edition

There are a few changes in the third edition. The most important is the new order of the chapters on decrease-and-conquer and divide-and-conquer. There are several advantages in introducing decrease-and-conquer before divide-and-conquer:

- Decrease-and-conquer is a simpler strategy than divide-and-conquer.
- Decrease-and-conquer is applicable to more problems than divide-and-conquer.
- The new order makes it possible to discuss insertion sort before mergesort and quicksort.
- The idea of array partitioning is now introduced in conjunction with the selection problem. I took advantage of an opportunity to do this via the one-directional scan employed by Lomuto's algorithm, leaving the two-directional scan used by Hoare's partitioning to a later discussion in conjunction with quicksort.
- Binary search is now considered in the section devoted to decrease-by-a-constant-factor algorithms, where it belongs.

The second important change is restructuring of Chapter 8 on dynamic programming. Specifically:

- The introductory section is completely new. It contains three basic examples that provide a much better introduction to this important technique than computing a binomial coefficient, the example used in the first two editions.
- All the exercises for Section 8.1 are new as well; they include well-known applications not available in the previous editions.
- I also changed the order of the other sections in this chapter to get a smoother progression from the simpler applications to the more advanced ones.

The other changes include the following. More applications of the algorithms discussed are included. The section on the graph-traversal algorithms is moved from the decrease-and-conquer chapter to the brute-force and exhaustive-search chapter, where it fits better, in my opinion. The Gray code algorithm is added to the section dealing with algorithms for generating combinatorial objects. The divide-and-conquer algorithm for the closest-pair problem is discussed in more detail. Updates include the section on algorithm visualization, approximation algorithms for the traveling salesman problem, and, of course, the bibliography.

I also added about 70 new problems to the exercises. Some of them are algorithmic puzzles and questions asked during job interviews.

Prerequisites

The book assumes that a reader has gone through an introductory programming course and a standard course on discrete structures. With such a background, he or she should be able to handle the book's material without undue difficulty.

Still, fundamental data structures, necessary summation formulas, and recurrence relations are reviewed in Section 1.4, Appendix A, and Appendix B, respectively. Calculus is used in only three sections (Section 2.2, 11.4, and 12.4), and to a very limited degree; if students lack calculus as an assured part of their background, the relevant portions of these three sections can be omitted without hindering their understanding of the rest of the material.

Use in the curriculum

The book can serve as a textbook for a basic course on design and analysis of algorithms organized around algorithm design techniques. It might contain slightly more material than can be covered in a typical one-semester course. By and large, portions of Chapters 3 through 12 can be skipped without the danger of making later parts of the book incomprehensible to the reader. Any portion of the book can be assigned for self-study. In particular, Sections 2.6 and 2.7 on empirical analysis and algorithm visualization, respectively, can be assigned in conjunction with projects.

Here is a possible plan for a one-semester course; it assumes a 40-class meeting format.

Lecture	Topic	Sections
1	Introduction	1.1–1.3
2, 3	Analysis framework; O, Ω, Θ notations	2.1, 2.2
4	Mathematical analysis of nonrecursive algorithms	2.3
5, 6	Mathematical analysis of recursive algorithms	2.4, 2.5 (+ App. B)
7	Brute-force algorithms	3.1, 3.2 (+ 3.3)
8	Exhaustive search	3.4
9	Depth-first search and breadth-first search	3.5
10, 11	Decrease-by-one: insertion sort, topological sorting	4.1, 4.2
12	Binary search and other decrease-by-a-constant-factor algorithms	4.4
13	Variable-size-decrease algorithms	4.5
14, 15	Divide-and-conquer: mergesort, quicksort	5.1–5.2
16	Other divide-and-conquer examples	5.3 or 5.4 or 5.5
17–19	Instance simplification: presorting, Gaussian elimination, balanced search trees	6.1–6.3
20	Representation change: heaps and heapsort or Horner's rule and binary exponentiation	6.4 or 6.5
21	Problem reduction	6.6
22–24	Space-time trade-offs: string matching, hashing, B-trees	7.2–7.4
25–27	Dynamic programming algorithms	3 from 8.1–8.4

28–30	Greedy algorithms: Prim's, Kruskal's, Dijkstra's, Huffman's	9.1–9.4
31–33	Iterative improvement algorithms	3 from 10.1–10.4
34	Lower-bound arguments	11.1
35	Decision trees	11.2
36	*P*, *NP*, and *NP*-complete problems	11.3
37	Numerical algorithms	11.4 (+ 12.4)
38	Backtracking	12.1
39	Branch-and-bound	12.2
40	Approximation algorithms for *NP*-hard problems	12.3

Acknowledgments

I would like to express my gratitude to the reviewers and many readers who have shared with me their opinions about the first two editions of the book and suggested improvements and corrections. The third edition has certainly benefited from the reviews by Andrew Harrington (Loyola University Chicago), David Levine (Saint Bonaventure University), Stefano Lombardi (UC Riverside), Daniel McKee (Mansfield University), Susan Brilliant (Virginia Commonwealth University), David Akers (University of Puget Sound), and two anonymous reviewers.

My thanks go to all the people at Pearson and their associates who worked on my book. I am especially grateful to my editor, Matt Goldstein; the editorial assistant, Chelsea Bell; the marketing manager, Yez Alayan; and the production supervisor, Kayla Smith-Tarbox. I am also grateful to Richard Camp for copyediting the book, Paul Anagnostopoulos of Windfall Software and Jacqui Scarlott for its project management and typesetting, and MaryEllen Oliver for proofreading the book.

Finally, I am indebted to two members of my family. Living with a spouse writing a book is probably more trying than doing the actual writing. My wife, Maria, lived through several years of this, helping me any way she could. And help she did: over 400 figures in the book and the Instructor's Manual were created by her. My daughter Miriam has been my English prose guru over many years. She read large portions of the book and was instrumental in finding the chapter epigraphs.

Anany Levitin
anany.levitin@villanova.edu
June 2011

Brief Contents

Contents

试读结束：需要全本请在线购买： www.ertongbook.com