

经 典 原 版 书 库

数据结构与算法分析

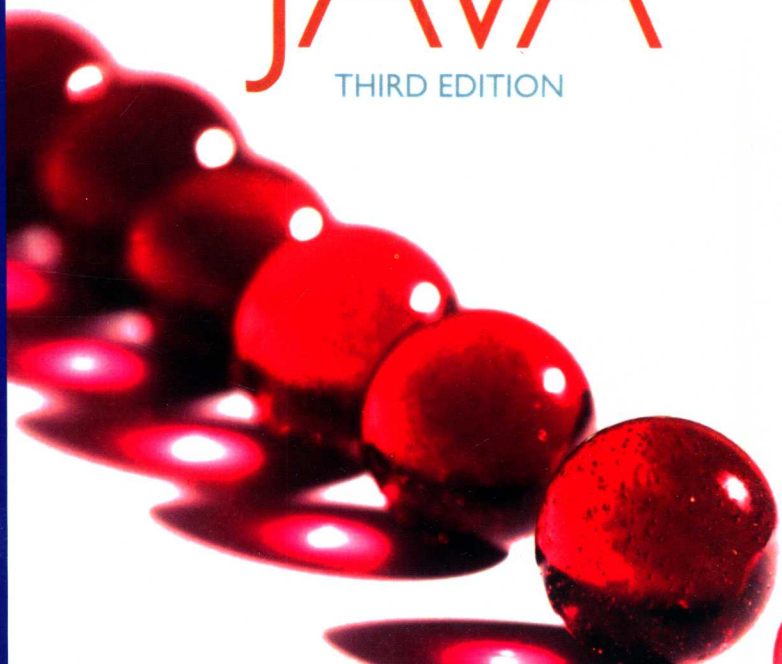
Java语言描述

(美) Mark Allen Weiss 著
佛罗里达国际大学

(英文版·第3版)

MARK ALLEN WEISS

DATA STRUCTURES
AND
ALGORITHM ANALYSIS
IN
JAVA™
THIRD EDITION



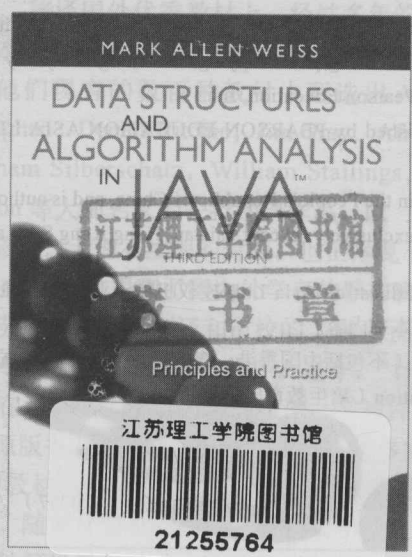
经 典 原 版 书 库

数据结构与算法分析


Java语言描述

(英文版·第3版)

Data Structures and Algorithm Analysis in Java (Third Edition)



(美) Mark Allen Weiss 著
佛罗里达国际大学

 机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

数据结构与算法分析：Java 语言描述（英文版·第3版）/（美）韦斯（Weiss, M. A.）著. —北京：机械工业出版社，2013.1

（经典原版书库）

书名原文：Data Structures and Algorithm Analysis in Java, Third Edition

ISBN 978-7-111-41236-6

I. 数… II. 韦… III. ① 数据结构—教材—英文 ② 算法分析—教材—英文 ③ JAVA 语言—程序设计—教材—英文 IV. ① TP311.12 ② TP312

中国版本图书馆 CIP 数据核字（2013）第 012558 号

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问 北京市展达律师事务所

本书版权登记号：图字：01-2013-0213

Original edition, entitled: Data Structures and Algorithm Analysis in Java, 3E, 9780132576277 by Mark Allen Weiss, published by Pearson Education, Inc, publishing as Pearson, Copyright © 2012, 2007, 1999.

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.

English reprint edition published by PEARSON EDUCATION ASIA LTD., and CHINA MACHINE PRESS, 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).

本书英文影印版由 Pearson Education Asia Ltd. 授权机械工业出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

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）销售发行。

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。

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：迟振春

北京瑞德印刷有限公司印刷

2013年2月第1版第1次印刷

170mm × 242mm · 39.5印张

标准书号：ISBN 978-7-111-41236-6

定价：79.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010) 88378991 88361066

投稿热线：(010) 88379604

购书热线：(010) 68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

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，不仅肇划了研究的范畴，还揭示了学术的源变，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和发展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到积极的推动作用，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公司较早意识到“出版要为教育服务”。自1998年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson, McGraw-Hill, Elsevier, MIT, John Wiley & Sons, Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum, Bjarne Stroustrup, Brian W. Kernighan, Dennis Ritchie, Jim Gray, Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, Abraham Silberschatz, William Stallings, Donald E. Knuth, John L. Hennessy, Larry L. Peterson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、研究及珍藏。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”已经出版了近两百个品种，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。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

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。随着计算机科学与技术专业学科建设的不断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将步入一个新的阶段，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。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，我们的联系方式如下：

华章网站：www.hzbook.com

电子邮件：hzjsj@hzbook.com

联系电话：(010) 88379604

联系地址：北京市西城区百万庄南街1号

邮政编码：100037



华章科技图书出版中心

PREFACE

Purpose/Goals

This new Java edition describes *data structures*, methods of organizing large amounts of data, and *algorithm analysis*, the estimation of the running time of algorithms. As computers become faster and faster, the need for programs that can handle large amounts of input becomes more acute. Paradoxically, this requires more careful attention to efficiency, since inefficiencies in programs become most obvious when input sizes are large. By analyzing an algorithm before it is actually coded, students can decide if a particular solution will be feasible. For example, in this text students look at specific problems and see how careful implementations can reduce the time constraint for large amounts of data from centuries to less than a second. Therefore, no algorithm or data structure is presented without an explanation of its running time. In some cases, minute details that affect the running time of the implementation are explored.

Once a solution method is determined, a program must still be written. As computers have become more powerful, the problems they must solve have become larger and more complex, requiring development of more intricate programs. The goal of this text is to teach students good programming and algorithm analysis skills simultaneously so that they can develop such programs with the maximum amount of efficiency.

This book is suitable for either an advanced data structures (CS7) course or a first-year graduate course in algorithm analysis. Students should have some knowledge of intermediate programming, including such topics as object-based programming and recursion, and some background in discrete math.

Summary of the Most Significant Changes in the Third Edition

The third edition incorporates numerous bug fixes, and many parts of the book have undergone revision to increase the clarity of presentation. In addition,

- Chapter 4 includes implementation of the AVL tree deletion algorithm—a topic often requested by readers.
- Chapter 5 has been extensively revised and enlarged and now contains material on two newer algorithms: cuckoo hashing and hopscotch hashing. Additionally, a new section on universal hashing has been added.
- Chapter 7 now contains material on radix sort, and a new section on lower bound proofs has been added.

- Chapter 8 uses the new union/find analysis by Seidel and Sharir, and shows the $O(M\alpha(M, N))$ bound instead of the weaker $O(M \log^* N)$ bound in prior editions.
- Chapter 12 adds material on suffix trees and suffix arrays, including the linear-time suffix array construction algorithm by Karkkainen and Sanders (with implementation). The sections covering deterministic skip lists and AA-trees have been removed.
- Throughout the text, the code has been updated to use the diamond operator from Java 7.

Approach

Although the material in this text is largely language independent, programming requires the use of a specific language. As the title implies, we have chosen Java for this book.

Java is often examined in comparison with C++. Java offers many benefits, and programmers often view Java as a safer, more portable, and easier-to-use language than C++. As such, it makes a fine core language for discussing and implementing fundamental data structures. Other important parts of Java, such as threads and its GUI, although important, are not needed in this text and thus are not discussed.

Complete versions of the data structures, in both Java and C++, are available on the Internet. We use similar coding conventions to make the parallels between the two languages more evident.

Overview

Chapter 1 contains review material on discrete math and recursion. I believe the only way to be comfortable with recursion is to see good uses over and over. Therefore, recursion is prevalent in this text, with examples in every chapter except Chapter 5. Chapter 1 also presents material that serves as a review of inheritance in Java. Included is a discussion of Java generics.

Chapter 2 deals with algorithm analysis. This chapter explains asymptotic analysis and its major weaknesses. Many examples are provided, including an in-depth explanation of logarithmic running time. Simple recursive programs are analyzed by intuitively converting them into iterative programs. More complicated divide-and-conquer programs are introduced, but some of the analysis (solving recurrence relations) is implicitly delayed until Chapter 7, where it is performed in detail.

Chapter 3 covers lists, stacks, and queues. This chapter has been significantly revised from prior editions. It now includes a discussion of the Collections API `ArrayList` and `LinkedList` classes, and it provides implementations of a significant subset of the collections API `ArrayList` and `LinkedList` classes.

Chapter 4 covers trees, with an emphasis on search trees, including external search trees (B-trees). The UNIX file system and expression trees are used as examples. AVL trees and splay trees are introduced. More careful treatment of search tree implementation details is found in Chapter 12. Additional coverage of trees, such as file compression and game trees, is deferred until Chapter 10. Data structures for an external medium are considered as the final topic in several chapters. New to this edition is a discussion of the Collections API `TreeSet` and `TreeMap` classes, including a significant example that illustrates the use of three separate maps to efficiently solve a problem.

Chapter 5 discusses hash tables, including the classic algorithms such as separate chaining and linear and quadratic probing, as well as several newer algorithms, namely cuckoo hashing and hopscotch hashing. Universal hashing is also discussed, and extendible hashing is covered at the end of the chapter.

Chapter 6 is about priority queues. Binary heaps are covered, and there is additional material on some of the theoretically interesting implementations of priority queues. The Fibonacci heap is discussed in Chapter 11, and the pairing heap is discussed in Chapter 12.

Chapter 7 covers sorting. It is very specific with respect to coding details and analysis. All the important general-purpose sorting algorithms are covered and compared. Four algorithms are analyzed in detail: insertion sort, Shellsort, heapsort, and quicksort. New to this edition is radix sort and lower bound proofs for selection-related problems. External sorting is covered at the end of the chapter.

Chapter 8 discusses the disjoint set algorithm with proof of the running time. The analysis is new. This is a short and specific chapter that can be skipped if Kruskal's algorithm is not discussed.

Chapter 9 covers graph algorithms. Algorithms on graphs are interesting, not only because they frequently occur in practice, but also because their running time is so heavily dependent on the proper use of data structures. Virtually all the standard algorithms are presented along with appropriate data structures, pseudocode, and analysis of running time. To place these problems in a proper context, a short discussion on complexity theory (including *NP*-completeness and undecidability) is provided.

Chapter 10 covers algorithm design by examining common problem-solving techniques. This chapter is heavily fortified with examples. Pseudocode is used in these later chapters so that the student's appreciation of an example algorithm is not obscured by implementation details.

Chapter 11 deals with amortized analysis. Three data structures from Chapters 4 and 6 and the Fibonacci heap, introduced in this chapter, are analyzed.

Chapter 12 covers search tree algorithms, the suffix tree and array, the *k*-d tree, and the pairing heap. This chapter departs from the rest of the text by providing complete and careful implementations for the search trees and pairing heap. The material is structured so that the instructor can integrate sections into discussions from other chapters. For example, the top-down red-black tree in Chapter 12 can be discussed along with AVL trees (in Chapter 4).

Chapters 1–9 provide enough material for most one-semester data structures courses. If time permits, then Chapter 10 can be covered. A graduate course on algorithm analysis could cover Chapters 7–11. The advanced data structures analyzed in Chapter 11 can easily be referred to in the earlier chapters. The discussion of *NP*-completeness in Chapter 9 is far too brief to be used in such a course. You might find it useful to use an additional work on *NP*-completeness to augment this text.

Exercises

Exercises, provided at the end of each chapter, match the order in which material is presented. The last exercises may address the chapter as a whole rather than a specific section. Difficult exercises are marked with an asterisk, and more challenging exercises have two asterisks.

References

References are placed at the end of each chapter. Generally the references either are historical, representing the original source of the material, or they represent extensions and improvements to the results given in the text. Some references represent solutions to exercises.

Supplements

The following supplements are available to all readers at www.pearsonhighered.com/cssupport:

- Source code for example programs

In addition, the following material is available only to qualified instructors at Pearson's Instructor Resource Center (www.pearsonhighered.com/irc). Visit the IRC or contact your campus Pearson representative for access.

- Solutions to selected exercises
- Figures from the book

Acknowledgments

Many, many people have helped me in the preparation of books in this series. Some are listed in other versions of the book; thanks to all.

As usual, the writing process was made easier by the professionals at Pearson. I'd like to thank my editor, Michael Hirsch, and production editor, Pat Brown. I'd also like to thank Abinaya Rajendran and her team in Integra Software Services for their fine work putting the final pieces together. My wonderful wife Jill deserves extra special thanks for everything she does.

Finally, I'd like to thank the numerous readers who have sent e-mail messages and pointed out errors or inconsistencies in earlier versions. My World Wide Web page www.cis.fiu.edu/~weiss contains updated source code (in Java and C++), an errata list, and a link to submit bug reports.

M.A.W.
Miami, Florida

CONTENTS

Chapter 1 Introduction

1

- 1.1 What's the Book About? 1
- 1.2 Mathematics Review 2
 - 1.2.1 Exponents 3
 - 1.2.2 Logarithms 3
 - 1.2.3 Series 4
 - 1.2.4 Modular Arithmetic 5
 - 1.2.5 The P Word 6
- 1.3 A Brief Introduction to Recursion 8
- 1.4 Implementing Generic Components Pre-Java 5 12
 - 1.4.1 Using `Object` for Genericity 13
 - 1.4.2 Wrappers for Primitive Types 14
 - 1.4.3 Using Interface Types for Genericity 14
 - 1.4.4 Compatibility of Array Types 16
- 1.5 Implementing Generic Components Using Java 5 Generics 16
 - 1.5.1 Simple Generic Classes and Interfaces 17
 - 1.5.2 Autoboxing/Unboxing 18
 - 1.5.3 The Diamond Operator 18
 - 1.5.4 Wildcards with Bounds 19
 - 1.5.5 Generic Static Methods 20
 - 1.5.6 Type Bounds 21
 - 1.5.7 Type Erasure 22
 - 1.5.8 Restrictions on Generics 23

- 1.6 Function Objects 24
- Summary 26
- Exercises 26
- References 28

Chapter 2 Algorithm Analysis

29

- 2.1 Mathematical Background 29
- 2.2 Model 32
- 2.3 What to Analyze 33
- 2.4 Running Time Calculations 35
 - 2.4.1 A Simple Example 36
 - 2.4.2 General Rules 36
 - 2.4.3 Solutions for the Maximum Subsequence Sum Problem 39
 - 2.4.4 Logarithms in the Running Time 45
 - 2.4.5 A Grain of Salt 49
- Summary 49
- Exercises 50
- References 55

Chapter 3 Lists, Stacks, and Queues

57

- 3.1 Abstract Data Types (ADTs) 57
- 3.2 The List ADT 58
 - 3.2.1 Simple Array Implementation of Lists 58
 - 3.2.2 Simple Linked Lists 59
- 3.3 Lists in the Java Collections API 61
 - 3.3.1 Collection Interface 61
 - 3.3.2 Iterators 61
 - 3.3.3 The List Interface, ArrayList, and LinkedList 63
 - 3.3.4 Example: Using remove on a LinkedList 65
 - 3.3.5 ListIterators 67
- 3.4 Implementation of ArrayList 67
 - 3.4.1 The Basic Class 68
 - 3.4.2 The Iterator and Java Nested and Inner Classes 71
- 3.5 Implementation of LinkedList 75
- 3.6 The Stack ADT 82
 - 3.6.1 Stack Model 82

- 3.6.2 Implementation of Stacks 83
- 3.6.3 Applications 84
- 3.7 The Queue ADT 92
 - 3.7.1 Queue Model 92
 - 3.7.2 Array Implementation of Queues 92
 - 3.7.3 Applications of Queues 95
- Summary 96
- Exercises 96

Chapter 4 Trees

101

- 4.1 Preliminaries 101
 - 4.1.1 Implementation of Trees 102
 - 4.1.2 Tree Traversals with an Application 103
- 4.2 Binary Trees 107
 - 4.2.1 Implementation 108
 - 4.2.2 An Example: Expression Trees 109
- 4.3 The Search Tree ADT—Binary Search Trees 112
 - 4.3.1 contains 113
 - 4.3.2 findMin and findMax 115
 - 4.3.3 insert 116
 - 4.3.4 remove 118
 - 4.3.5 Average-Case Analysis 120
- 4.4 AVL Trees 123
 - 4.4.1 Single Rotation 125
 - 4.4.2 Double Rotation 128
- 4.5 Splay Trees 137
 - 4.5.1 A Simple Idea (That Does Not Work) 137
 - 4.5.2 Splaying 139
- 4.6 Tree Traversals (Revisited) 145
- 4.7 B-Trees 147
- 4.8 Sets and Maps in the Standard Library 152
 - 4.8.1 Sets 152
 - 4.8.2 Maps 153
 - 4.8.3 Implementation of TreeSet and TreeMap 153
 - 4.8.4 An Example That Uses Several Maps 154
- Summary 160
- Exercises 160
- References 167

Chapter 5 Hashing	171
5.1 General Idea	171
5.2 Hash Function	172
5.3 Separate Chaining	174
5.4 Hash Tables Without Linked Lists	179
5.4.1 Linear Probing	179
5.4.2 Quadratic Probing	181
5.4.3 Double Hashing	183
5.5 Rehashing	188
5.6 Hash Tables in the Standard Library	189
5.7 Hash Tables with Worst-Case $O(1)$ Access	192
5.7.1 Perfect Hashing	193
5.7.2 Cuckoo Hashing	195
5.7.3 Hopscotch Hashing	205
5.8 Universal Hashing	211
5.9 Extendible Hashing	214
Summary	217
Exercises	218
References	222
Chapter 6 Priority Queues (Heaps)	225
6.1 Model	225
6.2 Simple Implementations	226
6.3 Binary Heap	226
6.3.1 Structure Property	227
6.3.2 Heap-Order Property	229
6.3.3 Basic Heap Operations	229
6.3.4 Other Heap Operations	234
6.4 Applications of Priority Queues	238
6.4.1 The Selection Problem	238
6.4.2 Event Simulation	239
6.5 d -Heaps	240
6.6 Leftist Heaps	241
6.6.1 Leftist Heap Property	241
6.6.2 Leftist Heap Operations	242
6.7 Skew Heaps	249

- 6.8 Binomial Queues 252
 - 6.8.1 Binomial Queue Structure 252
 - 6.8.2 Binomial Queue Operations 253
 - 6.8.3 Implementation of Binomial Queues 256
- 6.9 Priority Queues in the Standard Library 261
 - Summary 261
 - Exercises 263
 - References 267

Chapter 7 Sorting 271

- 7.1 Preliminaries 271
- 7.2 Insertion Sort 272
 - 7.2.1 The Algorithm 272
 - 7.2.2 Analysis of Insertion Sort 272
- 7.3 A Lower Bound for Simple Sorting Algorithms 273
- 7.4 Shellsort 274
 - 7.4.1 Worst-Case Analysis of Shellsort 276
- 7.5 Heapsort 278
 - 7.5.1 Analysis of Heapsort 279
- 7.6 Mergesort 282
 - 7.6.1 Analysis of Mergesort 284
- 7.7 Quicksort 288
 - 7.7.1 Picking the Pivot 290
 - 7.7.2 Partitioning Strategy 292
 - 7.7.3 Small Arrays 294
 - 7.7.4 Actual Quicksort Routines 294
 - 7.7.5 Analysis of Quicksort 297
 - 7.7.6 A Linear-Expected-Time Algorithm for Selection 300
- 7.8 A General Lower Bound for Sorting 302
 - 7.8.1 Decision Trees 302
- 7.9 Decision-Tree Lower Bounds for Selection Problems 304
- 7.10 Adversary Lower Bounds 307
- 7.11 Linear-Time Sorts: Bucket Sort and Radix Sort 310
- 7.12 External Sorting 315
 - 7.12.1 Why We Need New Algorithms 316
 - 7.12.2 Model for External Sorting 316
 - 7.12.3 The Simple Algorithm 316

- 7.12.4 Multiway Merge 317
- 7.12.5 Polyphase Merge 318
- 7.12.6 Replacement Selection 319
- Summary 321
- Exercises 321
- References 327

Chapter 8 The Disjoint Set Class

331

- 8.1 Equivalence Relations 331
- 8.2 The Dynamic Equivalence Problem 332
- 8.3 Basic Data Structure 333
- 8.4 Smart Union Algorithms 337
- 8.5 Path Compression 340
- 8.6 Worst Case for Union-by-Rank and Path Compression 341
 - 8.6.1 Slowly Growing Functions 342
 - 8.6.2 An Analysis By Recursive Decomposition 343
 - 8.6.3 An $O(M \log^* N)$ Bound 350
 - 8.6.4 An $O(M \alpha(M, N))$ Bound 350
- 8.7 An Application 352
 - Summary 355
 - Exercises 355
 - References 357

Chapter 9 Graph Algorithms

359

- 9.1 Definitions 359
 - 9.1.1 Representation of Graphs 360
- 9.2 Topological Sort 362
- 9.3 Shortest-Path Algorithms 366
 - 9.3.1 Unweighted Shortest Paths 367
 - 9.3.2 Dijkstra's Algorithm 372
 - 9.3.3 Graphs with Negative Edge Costs 380
 - 9.3.4 Acyclic Graphs 380
 - 9.3.5 All-Pairs Shortest Path 384
 - 9.3.6 Shortest-Path Example 384
- 9.4 Network Flow Problems 386
 - 9.4.1 A Simple Maximum-Flow Algorithm 388

- 9.5 Minimum Spanning Tree 393
 - 9.5.1 Prim's Algorithm 394
 - 9.5.2 Kruskal's Algorithm 397
- 9.6 Applications of Depth-First Search 399
 - 9.6.1 Undirected Graphs 400
 - 9.6.2 Biconnectivity 402
 - 9.6.3 Euler Circuits 405
 - 9.6.4 Directed Graphs 409
 - 9.6.5 Finding Strong Components 411
- 9.7 Introduction to NP-Completeness 412
 - 9.7.1 Easy vs. Hard 413
 - 9.7.2 The Class NP 414
 - 9.7.3 NP-Complete Problems 415
 - Summary 417
 - Exercises 417
 - References 425

Chapter 10 Algorithm Design Techniques

429

- 10.1 Greedy Algorithms 429
 - 10.1.1 A Simple Scheduling Problem 430
 - 10.1.2 Huffman Codes 433
 - 10.1.3 Approximate Bin Packing 439
- 10.2 Divide and Conquer 448
 - 10.2.1 Running Time of Divide-and-Conquer Algorithms 449
 - 10.2.2 Closest-Points Problem 451
 - 10.2.3 The Selection Problem 455
 - 10.2.4 Theoretical Improvements for Arithmetic Problems 458
- 10.3 Dynamic Programming 462
 - 10.3.1 Using a Table Instead of Recursion 463
 - 10.3.2 Ordering Matrix Multiplications 466
 - 10.3.3 Optimal Binary Search Tree 469
 - 10.3.4 All-Pairs Shortest Path 472
- 10.4 Randomized Algorithms 474
 - 10.4.1 Random Number Generators 476
 - 10.4.2 Skip Lists 480
 - 10.4.3 Primality Testing 483

- 10.5 Backtracking Algorithms 486
 - 10.5.1 The Turnpike Reconstruction Problem 487
 - 10.5.2 Games 490
 - Summary 499
 - Exercises 499
 - References 508

Chapter 11 Amortized Analysis

513

- 11.1 An Unrelated Puzzle 514
- 11.2 Binomial Queues 514
- 11.3 Skew Heaps 519
- 11.4 Fibonacci Heaps 522
 - 11.4.1 Cutting Nodes in Leftist Heaps 522
 - 11.4.2 Lazy Merging for Binomial Queues 525
 - 11.4.3 The Fibonacci Heap Operations 528
 - 11.4.4 Proof of the Time Bound 529
- 11.5 Splay Trees 531
 - Summary 536
 - Exercises 536
 - References 538

Chapter 12 Advanced Data Structures and Implementation

541

- 12.1 Top-Down Splay Trees 541
- 12.2 Red-Black Trees 549
 - 12.2.1 Bottom-Up Insertion 549
 - 12.2.2 Top-Down Red-Black Trees 551
 - 12.2.3 Top-Down Deletion 556
- 12.3 Treaps 558
- 12.4 Suffix Arrays and Suffix Trees 560
 - 12.4.1 Suffix Arrays 561
 - 12.4.2 Suffix Trees 564
 - 12.4.3 Linear-Time Construction of Suffix Arrays and Suffix Trees 567
- 12.5 k-d Trees 578