

Contents

Preface *xiii*

I Foundations

	Introduction	3
1	The Role of Algorithms in Computing	5
	1.1 Algorithms	5
	1.2 Algorithms as a technology	10
2	Getting Started	15
	2.1 Insertion sort	15
	2.2 Analyzing algorithms	21
	2.3 Designing algorithms	27
3	Growth of Functions	41
	3.1 Asymptotic notation	41
	3.2 Standard notations and common functions	51
4	Recurrences	62
	4.1 The substitution method	63
	4.2 The recursion-tree method	67
	4.3 The master method	73
★	4.4 Proof of the master theorem	76
5	Probabilistic Analysis and Randomized Algorithms	91
	5.1 The hiring problem	91
	5.2 Indicator random variables	94
	5.3 Randomized algorithms	99
★	5.4 Probabilistic analysis and further uses of indicator random variables	106

II Sorting and Order Statistics

	Introduction	123
6	Heapsort	127
6.1	Heaps	127
6.2	Maintaining the heap property	130
6.3	Building a heap	132
6.4	The heapsort algorithm	135
6.5	Priority queues	138
7	Quicksort	145
7.1	Description of quicksort	145
7.2	Performance of quicksort	149
7.3	A randomized version of quicksort	153
7.4	Analysis of quicksort	155
8	Sorting in Linear Time	165
8.1	Lower bounds for sorting	165
8.2	Counting sort	168
8.3	Radix sort	170
8.4	Bucket sort	174
9	Medians and Order Statistics	183
9.1	Minimum and maximum	184
9.2	Selection in expected linear time	185
9.3	Selection in worst-case linear time	189

III Data Structures

	Introduction	197
10	Elementary Data Structures	200
10.1	Stacks and queues	200
10.2	Linked lists	204
10.3	Implementing pointers and objects	209
10.4	Representing rooted trees	214
11	Hash Tables	221
11.1	Direct-address tables	222
11.2	Hash tables	224
11.3	Hash functions	229
11.4	Open addressing	237
★	11.5 Perfect hashing	245

- 12 Binary Search Trees 253**
 - 12.1 What is a binary search tree? 253
 - 12.2 Querying a binary search tree 256
 - 12.3 Insertion and deletion 261
 - ★ 12.4 Randomly built binary search trees 265
- 13 Red-Black Trees 273**
 - 13.1 Properties of red-black trees 273
 - 13.2 Rotations 277
 - ★ 13.3 Insertion 280
 - ★ 13.4 Deletion 288
- 14 Augmenting Data Structures 302**
 - 14.1 Dynamic order statistics 302
 - 14.2 How to augment a data structure 308
 - 14.3 Interval trees 311

IV Advanced Design and Analysis Techniques

- Introduction 321**
- 15 Dynamic Programming 323**
 - 15.1 Assembly-line scheduling 324
 - 15.2 Matrix-chain multiplication 331
 - 15.3 Elements of dynamic programming 339
 - 15.4 Longest common subsequence 350
 - 15.5 Optimal binary search trees 356
- 16 Greedy Algorithms 370**
 - 16.1 An activity-selection problem 371
 - 16.2 Elements of the greedy strategy 379
 - 16.3 Huffman codes 385
 - ★ 16.4 Theoretical foundations for greedy methods 393
 - ★ 16.5 A task-scheduling problem 399
- 17 Amortized Analysis 405**
 - 17.1 Aggregate analysis 406
 - 17.2 The accounting method 410
 - 17.3 The potential method 412
 - 17.4 Dynamic tables 416

V Advanced Data Structures

	Introduction	431
18	B-Trees	434
18.1	Definition of B-trees	438
18.2	Basic operations on B-trees	441
18.3	Deleting a key from a B-tree	449
19	Binomial Heaps	455
19.1	Binomial trees and binomial heaps	457
19.2	Operations on binomial heaps	461
20	Fibonacci Heaps	476
20.1	Structure of Fibonacci heaps	477
20.2	Mergeable-heap operations	479
20.3	Decreasing a key and deleting a node	489
20.4	Bounding the maximum degree	493
21	Data Structures for Disjoint Sets	498
21.1	Disjoint-set operations	498
21.2	Linked-list representation of disjoint sets	501
21.3	Disjoint-set forests	505
★	21.4 Analysis of union by rank with path compression	509

VI Graph Algorithms

	Introduction	525
22	Elementary Graph Algorithms	527
22.1	Representations of graphs	527
22.2	Breadth-first search	531
22.3	Depth-first search	540
22.4	Topological sort	549
22.5	Strongly connected components	552
23	Minimum Spanning Trees	561
23.1	Growing a minimum spanning tree	562
23.2	The algorithms of Kruskal and Prim	567
24	Single-Source Shortest Paths	580
24.1	The Bellman-Ford algorithm	588
24.2	Single-source shortest paths in directed acyclic graphs	592
24.3	Dijkstra's algorithm	595
★	24.4 Difference constraints and shortest paths	601
	24.5 Proofs of shortest-paths properties	607

- 25 All-Pairs Shortest Paths 620**
 - 25.1 Shortest paths and matrix multiplication 622
 - 25.2 The Floyd-Warshall algorithm 629
 - 25.3 Johnson's algorithm for sparse graphs 636
- 26 Maximum Flow 643**
 - 26.1 Flow networks 644
 - 26.2 The Ford-Fulkerson method 651
 - 26.3 Maximum bipartite matching 664
 - ★ 26.4 Push-relabel algorithms 669
 - ★ 26.5 The relabel-to-front algorithm 681

VII Selected Topics

- Introduction 701**
- 27 Sorting Networks 704**
 - 27.1 Comparison networks 704
 - 27.2 The zero-one principle 709
 - 27.3 A bitonic sorting network 712
 - 27.4 A merging network 716
 - 27.5 A sorting network 719
- 28 Matrix Operations 725**
 - 28.1 Properties of matrices 725
 - 28.2 Strassen's algorithm for matrix multiplication 735
 - 28.3 Solving systems of linear equations 742
 - 28.4 Inverting matrices 755
 - 28.5 Symmetric positive-definite matrices and least-squares approximation 760
- 29 Linear Programming 770**
 - 29.1 Standard and slack forms 777
 - 29.2 Formulating problems as linear programs 785
 - 29.3 The simplex algorithm 790
 - 29.4 Duality 804
 - 29.5 The initial basic feasible solution 811
- 30 Polynomials and the FFT 822**
 - 30.1 Representation of polynomials 824
 - 30.2 The DFT and FFT 830
 - 30.3 Efficient FFT implementations 839

31	Number-Theoretic Algorithms	849
31.1	Elementary number-theoretic notions	850
31.2	Greatest common divisor	856
31.3	Modular arithmetic	862
31.4	Solving modular linear equations	869
31.5	The Chinese remainder theorem	873
31.6	Powers of an element	876
31.7	The RSA public-key cryptosystem	881
★	31.8 Primality testing	887
★	31.9 Integer factorization	896
32	String Matching	906
32.1	The naive string-matching algorithm	909
32.2	The Rabin-Karp algorithm	911
32.3	String matching with finite automata	916
★	32.4 The Knuth-Morris-Pratt algorithm	923
33	Computational Geometry	933
33.1	Line-segment properties	934
33.2	Determining whether any pair of segments intersects	940
33.3	Finding the convex hull	947
33.4	Finding the closest pair of points	957
34	NP-Completeness	966
34.1	Polynomial time	971
34.2	Polynomial-time verification	979
34.3	NP-completeness and reducibility	984
34.4	NP-completeness proofs	995
34.5	NP-complete problems	1003
35	Approximation Algorithms	1022
35.1	The vertex-cover problem	1024
35.2	The traveling-salesman problem	1027
35.3	The set-covering problem	1033
35.4	Randomization and linear programming	1039
35.5	The subset-sum problem	1043

VIII Appendix: Mathematical Background

Introduction 1057

A Summations 1058

- A.1 Summation formulas and properties 1058
- A.2 Bounding summations 1062

Preface

B	Sets, Etc.	1070
B.1	Sets	1070
B.2	Relations	1075
B.3	Functions	1077
B.4	Graphs	1080
B.5	Trees	1085
C	Counting and Probability	1094
C.1	Counting	1094
C.2	Probability	1100
C.3	Discrete random variables	1106
C.4	The geometric and binomial distributions	1112
★ C.5	The tails of the binomial distribution	1118

	Bibliography	1127
--	---------------------	-------------

	Index	1145
--	--------------	-------------

Each chapter presents an algorithm, a design technique, an application area, or a related topic. Algorithms are described in English and in a “pseudocode” designed to be readable by anyone who has done a little programming. The book contains over 230 figures illustrating how the algorithms work. Because we emphasize efficiency as a design criterion, we include careful analyses of the running times of all our algorithms.

The text is intended primarily for use in undergraduate or graduate courses in algorithms or data structures. Because it discusses engineering as well as algorithm design, as well as mathematical aspects, it is equally well suited for self-study by technical professionals.

In this, the second edition, we have updated the entire book. The changes range from the addition of new chapters to the rewriting of individual chapters.

To the teacher

This book is designed to be both versatile and complete. It will be useful for a variety of courses, from an undergraduate course in data structures up through a graduate course in algorithms. Because we have prepared considerably more material than can fit in a typical one-term course, you may view the book as a “buffer” or “smorgasbord” from which you can pick and choose the material that best supports the course you wish to teach.

You should find it easy to organize your course around just the chapters you need. We have made chapters relatively self-contained, so that you need not worry about an unexpected and unnecessary dependence of one chapter on another. Each chapter presents the easier material first and the more difficult material later, with