

# Contents

12.1	What is a binary search tree?	147
12.2	Querying a binary search tree	147
12.3	Insertion and deletion	147
12.4	Randomly built binary search trees	147
13	Red-Black Trees	151
13.1	Properties of the red property	151
13.2	Rotations	151
13.3	Insertion	151
13.4	Deletion	151

## Preface *xiii*

## I Foundations

	<b>Introduction</b>	<b>3</b>
<b>1</b>	<b>The Role of Algorithms in Computing</b>	<b>5</b>
	1.1 Algorithms	5
	1.2 Algorithms as a technology	11
<b>2</b>	<b>Getting Started</b>	<b>16</b>
	2.1 Insertion sort	16
	2.2 Analyzing algorithms	23
	2.3 Designing algorithms	29
<b>3</b>	<b>Growth of Functions</b>	<b>43</b>
	3.1 Asymptotic notation	43
	3.2 Standard notations and common functions	53
<b>4</b>	<b>Divide-and-Conquer</b>	<b>65</b>
	4.1 The maximum-subarray problem	68
	4.2 Strassen's algorithm for matrix multiplication	75
	4.3 The substitution method for solving recurrences	83
	4.4 The recursion-tree method for solving recurrences	88
	4.5 The master method for solving recurrences	93
★	4.6 Proof of the master theorem	97
<b>5</b>	<b>Probabilistic Analysis and Randomized Algorithms</b>	<b>114</b>
	5.1 The hiring problem	114
	5.2 Indicator random variables	118
	5.3 Randomized algorithms	122
★	5.4 Probabilistic analysis and further uses of indicator random variables	130

---

**II Sorting and Order Statistics**


---

- Introduction** 147
- 6 Heapsort** 151
- 6.1 Heaps 151
- 6.2 Maintaining the heap property 154
- 6.3 Building a heap 156
- 6.4 The heapsort algorithm 159
- 6.5 Priority queues 162
- 7 Quicksort** 170
- 7.1 Description of quicksort 170
- 7.2 Performance of quicksort 174
- 7.3 A randomized version of quicksort 179
- 7.4 Analysis of quicksort 180
- 8 Sorting in Linear Time** 191
- 8.1 Lower bounds for sorting 191
- 8.2 Counting sort 194
- 8.3 Radix sort 197
- 8.4 Bucket sort 200
- 9 Medians and Order Statistics** 213
- 9.1 Minimum and maximum 214
- 9.2 Selection in expected linear time 215
- 9.3 Selection in worst-case linear time 220
- 

**III Data Structures**


---

- Introduction** 229
- 10 Elementary Data Structures** 232
- 10.1 Stacks and queues 232
- 10.2 Linked lists 236
- 10.3 Implementing pointers and objects 241
- 10.4 Representing rooted trees 246
- 11 Hash Tables** 253
- 11.1 Direct-address tables 254
- 11.2 Hash tables 256
- 11.3 Hash functions 262
- 11.4 Open addressing 269
- ★ 11.5 Perfect hashing 277
-

- 12 Binary Search Trees 286**
  - 12.1 What is a binary search tree? 286
  - 12.2 Querying a binary search tree 289
  - 12.3 Insertion and deletion 294
  - ★ 12.4 Randomly built binary search trees 299
- 13 Red-Black Trees 308**
  - 13.1 Properties of red-black trees 308
  - 13.2 Rotations 312
  - 13.3 Insertion 315
  - 13.4 Deletion 323
- 14 Augmenting Data Structures 339**
  - 14.1 Dynamic order statistics 339
  - 14.2 How to augment a data structure 345
  - 14.3 Interval trees 348

---

## **IV Advanced Design and Analysis Techniques**

---

- Introduction 357**
- 15 Dynamic Programming 359**
  - 15.1 Rod cutting 360
  - 15.2 Matrix-chain multiplication 370
  - 15.3 Elements of dynamic programming 378
  - 15.4 Longest common subsequence 390
  - 15.5 Optimal binary search trees 397
- 16 Greedy Algorithms 414**
  - 16.1 An activity-selection problem 415
  - 16.2 Elements of the greedy strategy 423
  - 16.3 Huffman codes 428
  - ★ 16.4 Matroids and greedy methods 437
  - ★ 16.5 A task-scheduling problem as a matroid 443
- 17 Amortized Analysis 451**
  - 17.1 Aggregate analysis 452
  - 17.2 The accounting method 456
  - 17.3 The potential method 459
  - 17.4 Dynamic tables 463

---

**V Advanced Data Structures**


---

- Introduction** 481
- 18 B-Trees** 484
- 18.1 Definition of B-trees 488
- 18.2 Basic operations on B-trees 491
- 18.3 Deleting a key from a B-tree 499
- 19 Fibonacci Heaps** 505
- 19.1 Structure of Fibonacci heaps 507
- 19.2 Mergeable-heap operations 510
- 19.3 Decreasing a key and deleting a node 518
- 19.4 Bounding the maximum degree 523
- 20 van Emde Boas Trees** 531
- 20.1 Preliminary approaches 532
- 20.2 A recursive structure 536
- 20.3 The van Emde Boas tree 545
- 21 Data Structures for Disjoint Sets** 561
- 21.1 Disjoint-set operations 561
- 21.2 Linked-list representation of disjoint sets 564
- 21.3 Disjoint-set forests 568
- ★ 21.4 Analysis of union by rank with path compression 573
- 

**VI Graph Algorithms**


---

- Introduction** 587
- 22 Elementary Graph Algorithms** 589
- 22.1 Representations of graphs 589
- 22.2 Breadth-first search 594
- 22.3 Depth-first search 603
- 22.4 Topological sort 612
- 22.5 Strongly connected components 615
- 23 Minimum Spanning Trees** 624
- 23.1 Growing a minimum spanning tree 625
- 23.2 The algorithms of Kruskal and Prim 631
-

- 24 Single-Source Shortest Paths 643**
- 24.1 The Bellman-Ford algorithm 651
  - 24.2 Single-source shortest paths in directed acyclic graphs 655
  - 24.3 Dijkstra's algorithm 658
  - 24.4 Difference constraints and shortest paths 664
  - 24.5 Proofs of shortest-paths properties 671
- 25 All-Pairs Shortest Paths 684**
- 25.1 Shortest paths and matrix multiplication 686
  - 25.2 The Floyd-Warshall algorithm 693
  - 25.3 Johnson's algorithm for sparse graphs 700
- 26 Maximum Flow 708**
- 26.1 Flow networks 709
  - 26.2 The Ford-Fulkerson method 714
  - 26.3 Maximum bipartite matching 732
  - ★ 26.4 Push-relabel algorithms 736
  - ★ 26.5 The relabel-to-front algorithm 748

---

## VII Selected Topics

---

- Introduction 769**
- 27 Multithreaded Algorithms 772**
- 27.1 The basics of dynamic multithreading 774
  - 27.2 Multithreaded matrix multiplication 792
  - 27.3 Multithreaded merge sort 797
- 28 Matrix Operations 813**
- 28.1 Solving systems of linear equations 813
  - 28.2 Inverting matrices 827
  - 28.3 Symmetric positive-definite matrices and least-squares approximation 832
- 29 Linear Programming 843**
- 29.1 Standard and slack forms 850
  - 29.2 Formulating problems as linear programs 859
  - 29.3 The simplex algorithm 864
  - 29.4 Duality 879
  - 29.5 The initial basic feasible solution 886

- 30 Polynomials and the FFT 898**
- 30.1 Representing polynomials 900
  - 30.2 The DFT and FFT 906
  - 30.3 Efficient FFT implementations 915
- 31 Number-Theoretic Algorithms 926**
- 31.1 Elementary number-theoretic notions 927
  - 31.2 Greatest common divisor 933
  - 31.3 Modular arithmetic 939
  - 31.4 Solving modular linear equations 946
  - 31.5 The Chinese remainder theorem 950
  - 31.6 Powers of an element 954
  - 31.7 The RSA public-key cryptosystem 958
  - ★ 31.8 Primality testing 965
  - ★ 31.9 Integer factorization 975
- 32 String Matching 985**
- 32.1 The naive string-matching algorithm 988
  - 32.2 The Rabin-Karp algorithm 990
  - 32.3 String matching with finite automata 995
  - ★ 32.4 The Knuth-Morris-Pratt algorithm 1002
- 33 Computational Geometry 1014**
- 33.1 Line-segment properties 1015
  - 33.2 Determining whether any pair of segments intersects 1021
  - 33.3 Finding the convex hull 1029
  - 33.4 Finding the closest pair of points 1039
- 34 NP-Completeness 1048**
- 34.1 Polynomial time 1053
  - 34.2 Polynomial-time verification 1061
  - 34.3 NP-completeness and reducibility 1067
  - 34.4 NP-completeness proofs 1078
  - 34.5 NP-complete problems 1086
- 35 Approximation Algorithms 1106**
- 35.1 The vertex-cover problem 1108
  - 35.2 The traveling-salesman problem 1111
  - 35.3 The set-covering problem 1117
  - 35.4 Randomization and linear programming 1123
  - 35.5 The subset-sum problem 1128

---

**VIII Appendix: Mathematical Background**


---

	<b>Introduction</b>	<b>1143</b>
<b>A</b>	<b>Summations</b>	<b>1145</b>
	A.1 Summation formulas and properties	1145
	A.2 Bounding summations	1149
<b>B</b>	<b>Sets, Etc.</b>	<b>1158</b>
	B.1 Sets	1158
	B.2 Relations	1163
	B.3 Functions	1166
	B.4 Graphs	1168
	B.5 Trees	1173
<b>C</b>	<b>Counting and Probability</b>	<b>1183</b>
	C.1 Counting	1183
	C.2 Probability	1189
	C.3 Discrete random variables	1196
	C.4 The geometric and binomial distributions	1201
★	C.5 The tails of the binomial distribution	1208
<b>D</b>	<b>Matrices</b>	<b>1217</b>
	D.1 Matrices and matrix operations	1217
	D.2 Basic matrix properties	1222

---

**Bibliography** 1231

**Index** 1251

### To the teacher

We have designed this book to be both versatile and complete. You should find it useful for a variety of courses, from an undergraduate course in data structures up through a graduate course in algorithms. Because we have provided considerably more material than can fit in a typical one-term course, you can consider this book to be a "buffet" or "smorgasbord" from which you can pick and choose the material that best supports the course you wish to teach.