

CONTENTS

Preface xv

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 7
- 1.4 Implementing Generic Components Pre Java 5 11
 - 1.4.1 Using **Object** for Genericity 12
 - 1.4.2 Wrappers for Primitive Types 12
 - 1.4.3 Using Interface Types for Genericity 13
 - 1.4.4 Compatibility of Array Types 15
- 1.5 Implementing Generic Components Using Java 5 Generics 16
 - 1.5.1 Simple Generic Classes and Interfaces 16
 - 1.5.2 Autoboxing/Unboxing 17
 - 1.5.3 Wildcards with Bounds 18
 - 1.5.4 Generic Static Methods 19
 - 1.5.5 Type Bounds 20
 - 1.5.6 Type Erasure 21
 - 1.5.7 Restrictions on Generics 22
- 1.6 Function Objects 23
 - Summary 25
 - Exercises 25
 - References 26

vii

Chapter 2 Algorithm Analysis

29

- 2.1 Mathematical Background 29
- 2.2 Model 32
- 2.3 What to Analyze 32
- 2.4 Running Time Calculations 35
 - 2.4.1 A Simple Example 35
 - 2.4.2 General Rules 36
 - 2.4.3 Solutions for the Maximum Subsequence Sum Problem 38
 - 2.4.4 Logarithms in the Running Time 44
 - 2.4.5 Checking Your Analysis 48
 - 2.4.6 A Grain of Salt 48
- Summary 50
- 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 60
 - 3.3.1 Collection Interface 61
 - 3.3.2 Iterators 62
 - 3.3.3 The List Interface, ArrayList, and LinkedList 63
 - 3.3.4 Example: Using remove on a LinkedList 65
 - 3.3.5 ListIterators 66
- 3.4 Implementation of ArrayList 67
 - 3.4.1 The Basic Class 68
 - 3.4.2 The Iterator and Java Nested and Inner Classes 68
- 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 83
- 3.7 The Queue ADT 91
 - 3.7.1 Queue Model 91
 - 3.7.2 Array Implementation of Queues 91
 - 3.7.3 Applications of Queues 94
- Summary 95
- Exercises 95

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` 115
 - 4.3.4 `remove` 117
 - 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 135
 - 4.5.1 A Simple Idea (That Does Not Work) 135
 - 4.5.2 Splaying 137
- 4.6 Tree Traversals (Revisited) 143
- 4.7 B-Trees 145
- 4.8 Sets and Maps in the Standard Library 150
 - 4.8.1 Sets 151
 - 4.8.2 Maps 151
 - 4.8.3 Implementation of `TreeSet` and `TreeMap` 152
 - 4.8.4 An Example That Uses Several Maps 152
- 4.9 Summary 157
 - Exercises 159
 - References 165

Chapter 5 Hashing

169

- 5.1 General Idea 169
- 5.2 Hash Function 170
- 5.3 Separate Chaining 172
- 5.4 Hash Tables Without Linked Lists 177
 - 5.4.1 Linear Probing 177
 - 5.4.2 Quadratic Probing 179
 - 5.4.3 Double Hashing 181
- 5.5 Rehashing 186

5.6	Hash Tables in the Standard Library	187
5.7	Extendible Hashing	190
	Summary	193
	Exercises	194
	References	198

Chapter 6 Priority Queues (Heaps)

201

6.1	Model	201
6.2	Simple Implementations	202
6.3	Binary Heap	202
	6.3.1 Structure Property	203
	6.3.2 Heap Order Property	205
	6.3.3 Basic Heap Operations	205
	6.3.4 Other Heap Operations	210
6.4	Applications of Priority Queues	214
	6.4.1 The Selection Problem	214
	6.4.2 Event Simulation	215
6.5	<i>d</i> -Heaps	216
6.6	Leftist Heaps	217
	6.6.1 Leftist Heap Property	217
	6.6.2 Leftist Heap Operations	218
6.7	Skew Heaps	225
6.8	Binomial Queues	227
	6.8.1 Binomial Queue Structure	228
	6.8.2 Binomial Queue Operations	229
	6.8.3 Implementation of Binomial Queues	232
6.9	Priority Queues in the Standard Library	237
	Summary	237
	Exercises	239
	References	243

Chapter 7 Sorting

247

7.1	Preliminaries	247
7.2	Insertion Sort	248
	7.2.1 The Algorithm	248
	7.2.2 Analysis of Insertion Sort	248
7.3	A Lower Bound for Simple Sorting Algorithms	249
7.4	Shellsort	250
	7.4.1 Worst-Case Analysis of Shellsort	252

7.5	Heapsort	254
7.5.1	Analysis of Heapsort	256
7.6	Mergesort	258
7.6.1	Analysis of Mergesort	260
7.7	Quicksort	264
7.7.1	Picking the Pivot	264
7.7.2	Partitioning Strategy	266
7.7.3	Small Arrays	268
7.7.4	Actual Quicksort Routines	268
7.7.5	Analysis of Quicksort	271
7.7.6	A Linear-Expected-Time Algorithm for Selection	274
7.8	A General Lower Bound for Sorting	276
7.8.1	Decision Trees	276
7.9	Bucket Sort	278
7.10	External Sorting	279
7.10.1	Why We Need New Algorithms	279
7.10.2	Model for External Sorting	279
7.10.3	The Simple Algorithm	279
7.10.4	Multiway Merge	281
7.10.5	Polyphase Merge	282
7.10.6	Replacement Selection	283
	Summary	284
	Exercises	285
	References	290

Chapter 8 The Disjoint Set Class

293

8.1	Equivalence Relations	293
8.2	The Dynamic Equivalence Problem	294
8.3	Basic Data Structure	295
8.4	Smart Union Algorithms	299
8.5	Path Compression	301
8.6	Worst Case for Union-by-Rank and Path Compression	303
8.6.1	Analysis of the Union/Find Algorithm	304
8.7	An Application	309
	Summary	312
	Exercises	312
	References	314

Chapter 9 Graph Algorithms 317

- 9.1 Definitions 317
 - 9.1.1 Representation of Graphs 318
- 9.2 Topological Sort 320
- 9.3 Shortest-Path Algorithms 323
 - 9.3.1 Unweighted Shortest Paths 325
 - 9.3.2 Dijkstra's Algorithm 329
 - 9.3.3 Graphs with Negative Edge Costs 338
 - 9.3.4 Acyclic Graphs 338
 - 9.3.5 All-Pairs Shortest Path 342
 - 9.3.6 Shortest-Path Example 342
- 9.4 Network Flow Problems 344
 - 9.4.1 A Simple Maximum-Flow Algorithm 344
- 9.5 Minimum Spanning Tree 349
 - 9.5.1 Prim's Algorithm 351
 - 9.5.2 Kruskal's Algorithm 353
- 9.6 Applications of Depth-First Search 355
 - 9.6.1 Undirected Graphs 357
 - 9.6.2 Biconnectivity 358
 - 9.6.3 Euler Circuits 361
 - 9.6.4 Directed Graphs 366
 - 9.6.5 Finding Strong Components 367
- 9.7 Introduction to NP-Completeness 369
 - 9.7.1 Easy vs. Hard 369
 - 9.7.2 The Class NP 370
 - 9.7.3 NP-Complete Problems 371
- Summary 373
- Exercises 373
- References 381

Chapter 10 Algorithm Design Techniques 385

- 10.1 Greedy Algorithms 385
 - 10.1.1 A Simple Scheduling Problem 386
 - 10.1.2 Huffman Codes 389
 - 10.1.3 Approximate Bin Packing 395
- 10.2 Divide and Conquer 403
 - 10.2.1 Running Time of Divide and Conquer Algorithms 404
 - 10.2.2 Closest-Points Problem 406
 - 10.2.3 The Selection Problem 411
 - 10.2.4 Theoretical Improvements for Arithmetic Problems 414

10.3	Dynamic Programming	418
10.3.1	Using a Table Instead of Recursion	418
10.3.2	Ordering Matrix Multiplications	421
10.3.3	Optimal Binary Search Tree	424
10.3.4	All-Pairs Shortest Path	426
10.4	Randomized Algorithms	429
10.4.1	Random Number Generators	431
10.4.2	Skip Lists	435
10.4.3	Primality Testing	437
10.5	Backtracking Algorithms	440
10.5.1	The Turnpike Reconstruction Problem	440
10.5.2	Games	445
	Summary	452
	Exercises	452
	References	461

Chapter 11 Amortized Analysis

465

11.1	An Unrelated Puzzle	466
11.2	Binomial Queues	466
11.3	Skew Heaps	471
11.4	Fibonacci Heaps	473
11.4.1	Cutting Nodes in Leftist Heaps	474
11.4.2	Lazy Merging for Binomial Queues	476
11.4.3	The Fibonacci Heap Operations	480
11.4.4	Proof of the Time Bound	480
11.5	Splay Trees	483
	Summary	487
	Exercises	487
	References	489

Chapter 12 Advanced Data Structures and Implementation

491

12.1	Top-Down Splay Trees	491
12.2	Red-Black Trees	497
12.2.1	Bottom-Up Insertion	499
12.2.2	Top-Down Red-Black Trees	501
12.2.3	Top-Down Deletion	506
12.3	Deterministic Skip Lists	508
12.4	AA-Trees	513

12.5	Treaps	521
12.6	k -d Trees	523
12.7	Pairing Heaps	527
	Summary	532
	Exercises	534
	References	538

Index	541
--------------	------------