

# Table of Contents

## Chapter 1 Programming: A General Overview 1

- 1.1 What's This 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 C++ Classes 12
  - 1.4.1 Basic class Syntax 12
  - 1.4.2 Extra Constructor Syntax and Accessors 13
  - 1.4.3 Separation of Interface and Implementation 16
  - 1.4.4 vector and string 19
- 1.5 C++ Details 21
  - 1.5.1 Pointers 21
  - 1.5.2 Lvalues, Rvalues, and References 23
  - 1.5.3 Parameter Passing 25
  - 1.5.4 Return Passing 27
  - 1.5.5 std::swap and std::move 29
  - 1.5.6 The Big-Five: Destructor, Copy Constructor, Move Constructor, Copy Assignment operator=, Move Assignment operator= 30
  - 1.5.7 C-style Arrays and Strings 35
- 1.6 Templates 36
  - 1.6.1 Function Templates 37
  - 1.6.2 Class Templates 38
  - 1.6.3 Object, Comparable, and an Example 39
  - 1.6.4 Function Objects 41
  - 1.6.5 Separate Compilation of Class Templates 44
- 1.7 Using Matrices 44
  - 1.7.1 The Data Members, Constructor, and Basic Accessors 44
  - 1.7.2 operator[] 45
  - 1.7.3 Big-Five 46
- Summary 46
- Exercises 46
- References 48

## Chapter 2 Algorithm Analysis 51

- 2.1 Mathematical Background 51
- 2.2 Model 54
- 2.3 What to Analyze 54
- 2.4 Running-Time Calculations 57
  - 2.4.1 A Simple Example 58
  - 2.4.2 General Rules 58

|   |    |
|---|----|
| 2.4.3 Solutions for the Maximum Subsequence Sum Problem | 60 |
| 2.4.4 Logarithms in the Running Time                    | 66 |
| 2.4.5 Limitations of Worst Case Analysis                | 70 |
| Summary   | 70 |
| Exercises   | 71 |
| References  | 76 |

## **Chapter 3 Lists, Stacks, and Queues 77**

|  |     |
|--|-----|
| 3.1 Abstract Data Types (ADTs)             | 77  |
| 3.2 The List ADT                           | 78  |
| 3.2.1 Simple Array Implementation of Lists | 78  |
| 3.2.2 Simple Linked Lists                  | 79  |
| 3.3 vector and list in the STL             | 80  |
| 3.3.1 Iterators                            | 82  |
| 3.3.2 Example: Using erase on a List       | 83  |
| 3.3.3 const_iterators                      | 84  |
| 3.4 Implementation of vector               | 86  |
| 3.5 Implementation of list                 | 91  |
| 3.6 The Stack ADT                          | 103 |
| 3.6.1 Stack Model                          | 103 |
| 3.6.2 Implementation of Stacks             | 104 |
| 3.6.3 Applications                         | 104 |
| 3.7 The Queue ADT                          | 112 |
| 3.7.1 Queue Model                          | 113 |
| 3.7.2 Array Implementation of Queues       | 113 |
| 3.7.3 Applications of Queues               | 115 |
| Summary                                    | 116 |
| Exercises                                  | 116 |

## **Chapter 4 Trees 121**

|   |     |
|---|-----|
| 4.1 Preliminaries                           | 121 |
| 4.1.1 Implementation of Trees               | 122 |
| 4.1.2 Tree Traversals with an Application   | 123 |
| 4.2 Binary Trees                            | 126 |
| 4.2.1 Implementation                        | 128 |
| 4.2.2 An Example: Expression Trees          | 128 |
| 4.3 The Search Tree ADT—Binary Search Trees | 132 |
| 4.3.1 contains                              | 134 |
| 4.3.2 findMin and findMax                   | 135 |
| 4.3.3 insert                                | 136 |
| 4.3.4 remove                                | 139 |
| 4.3.5 Destructor and Copy Constructor       | 141 |
| 4.3.6 Average-Case Analysis                 | 141 |
| 4.4 AVL Trees                               | 144 |
| 4.4.1 Single Rotation                       | 147 |
| 4.4.2 Double Rotation                       | 149 |

- 4.5 Splay Trees 158
  - 4.5.1 A Simple Idea (That Does Not Work) 158
  - 4.5.2 Splaying 160
- 4.6 Tree Traversals (Revisited) 166
- 4.7 B-Trees 168
- 4.8 Sets and Maps in the Standard Library 173
  - 4.8.1 Sets 173
  - 4.8.2 Maps 174
  - 4.8.3 Implementation of set and map 175
  - 4.8.4 An Example That Uses Several Maps 176
- Summary 181
- Exercises 182
- References 189

## **Chapter 5 Hashing 193**

- 5.1 General Idea 193
- 5.2 Hash Function 194
- 5.3 Separate Chaining 196
- 5.4 Hash Tables without Linked Lists 201
  - 5.4.1 Linear Probing 201
  - 5.4.2 Quadratic Probing 202
  - 5.4.3 Double Hashing 207
- 5.5 Rehashing 208
- 5.6 Hash Tables in the Standard Library 210
- 5.7 Hash Tables with Worst-Case  $O(1)$  Access 212
  - 5.7.1 Perfect Hashing 213
  - 5.7.2 Cuckoo Hashing 215
  - 5.7.3 Hopscotch Hashing 224
- 5.8 Universal Hashing 230
- 5.9 Extendible Hashing 233
- Summary 236
- Exercises 238
- References 242

## **Chapter 6 Priority Queues (Heaps) 245**

- 6.1 Model 245
- 6.2 Simple Implementations 246
- 6.3 Binary Heap 247
  - 6.3.1 Structure Property 247
  - 6.3.2 Heap-Order Property 248
  - 6.3.3 Basic Heap Operations 249
  - 6.3.4 Other Heap Operations 252
- 6.4 Applications of Priority Queues 257
  - 6.4.1 The Selection Problem 258
  - 6.4.2 Event Simulation 259
- 6.5  $d$ -Heaps 260

|       |   |     |
|-------|---|-----|
| 6.6   | Leftist Heaps                           | 261 |
| 6.6.1 | Leftist Heap Property                   | 261 |
| 6.6.2 | Leftist Heap Operations                 | 262 |
| 6.7   | Skew Heaps                              | 269 |
| 6.8   | Binomial Queues                         | 271 |
| 6.8.1 | Binomial Queue Structure                | 271 |
| 6.8.2 | Binomial Queue Operations               | 271 |
| 6.8.3 | Implementation of Binomial Queues       | 276 |
| 6.9   | Priority Queues in the Standard Library | 283 |
|       | Summary                                 | 283 |
|       | Exercises                               | 283 |
|       | References                              | 288 |

## **Chapter 7 Sorting 291**

|        |   |     |
|--------|---|-----|
| 7.1    | Preliminaries                                     | 291 |
| 7.2    | Insertion Sort                                    | 292 |
| 7.2.1  | The Algorithm                                     | 292 |
| 7.2.2  | STL Implementation of Insertion Sort              | 293 |
| 7.2.3  | Analysis of Insertion Sort                        | 294 |
| 7.3    | A Lower Bound for Simple Sorting Algorithms       | 295 |
| 7.4    | Shellsort   | 296 |
| 7.4.1  | Worst-Case Analysis of Shellsort                  | 297 |
| 7.5    | Heapsort  | 300 |
| 7.5.1  | Analysis of Heapsort                              | 301 |
| 7.6    | Mergesort   | 304 |
| 7.6.1  | Analysis of Mergesort                             | 306 |
| 7.7    | Quicksort   | 309 |
| 7.7.1  | Picking the Pivot                                 | 311 |
| 7.7.2  | Partitioning Strategy                             | 313 |
| 7.7.3  | Small Arrays                                      | 315 |
| 7.7.4  | Actual Quicksort Routines                         | 315 |
| 7.7.5  | Analysis of Quicksort                             | 318 |
| 7.7.6  | A Linear-Expected-Time Algorithm for Selection    | 321 |
| 7.8    | A General Lower Bound for Sorting                 | 323 |
| 7.8.1  | Decision Trees                                    | 323 |
| 7.9    | Decision-Tree Lower Bounds for Selection Problems | 325 |
| 7.10   | Adversary Lower Bounds                            | 328 |
| 7.11   | Linear-Time Sorts: Bucket Sort and Radix Sort     | 331 |
| 7.12   | External Sorting                                  | 336 |
| 7.12.1 | Why We Need New Algorithms                        | 336 |
| 7.12.2 | Model for External Sorting                        | 336 |
| 7.12.3 | The Simple Algorithm                              | 337 |
| 7.12.4 | Multiway Merge                                    | 338 |
| 7.12.5 | Polyphase Merge                                   | 339 |
| 7.12.6 | Replacement Selection                             | 340 |
|        | Summary   | 341 |

Exercises 341  
References 347

## **Chapter 8 The Disjoint Sets Class 351**

8.1 Equivalence Relations 351  
8.2 The Dynamic Equivalence Problem 352  
8.3 Basic Data Structure 353  
8.4 Smart Union Algorithms 357  
8.5 Path Compression 360  
8.6 Worst Case for Union-by-Rank and Path Compression 361  
8.6.1 Slowly Growing Functions 362  
8.6.2 An Analysis by Recursive Decomposition 362  
8.6.3 An  $O(M \log^* N)$  Bound 369  
8.6.4 An  $O(M a(M, N))$  Bound 370  
8.7 An Application 372  
Summary 374  
Exercises 375  
References 376

## **Chapter 9 Graph Algorithms 379**

9.1 Definitions 379  
9.1.1 Representation of Graphs 380  
9.2 Topological Sort 382  
9.3 Shortest-Path Algorithms 386  
9.3.1 Unweighted Shortest Paths 387  
9.3.2 Dijkstra's Algorithm 391  
9.3.3 Graphs with Negative Edge Costs 400  
9.3.4 Acyclic Graphs 400  
9.3.5 All-Pairs Shortest Path 404  
9.3.6 Shortest Path Example 404  
9.4 Network Flow Problems 406  
9.4.1 A Simple Maximum-Flow Algorithm 408  
9.5 Minimum Spanning Tree 413  
9.5.1 Prim's Algorithm 414  
9.5.2 Kruskal's Algorithm 417  
9.6 Applications of Depth-First Search 419  
9.6.1 Undirected Graphs 420  
9.6.2 Biconnectivity 421  
9.6.3 Euler Circuits 425  
9.6.4 Directed Graphs 429  
9.6.5 Finding Strong Components 431  
9.7 Introduction to NP-Completeness 432  
9.7.1 Easy vs. Hard 433  
9.7.2 The Class NP 434  
9.7.3 NP-Complete Problems 434  
Summary 437

Exercises 437  
References 445

## **Chapter 10 Algorithm Design Techniques 449**

10.1 Greedy Algorithms 449  
10.1.1 A Simple Scheduling Problem 450  
10.1.2 Huffman Codes 453  
10.1.3 Approximate Bin Packing 459  
10.2 Divide and Conquer 467  
10.2.1 Running Time of Divide-and-Conquer Algorithms 468  
10.2.2 Closest-Points Problem 470  
10.2.3 The Selection Problem 475  
10.2.4 Theoretical Improvements for Arithmetic Problems 478  
10.3 Dynamic Programming 482  
10.3.1 Using a Table Instead of Recursion 483  
10.3.2 Ordering Matrix Multiplications 485  
10.3.3 Optimal Binary Search Tree 487  
10.3.4 All-Pairs Shortest Path 491  
10.4 Randomized Algorithms 494  
10.4.1 Random-Number Generators 495  
10.4.2 Skip Lists 500  
10.4.3 Primality Testing 503  
10.5 Backtracking Algorithms 506  
10.5.1 The Turnpike Reconstruction Problem 506  
10.5.2 Games 511  
Summary 518  
Exercises 518  
References 527

## **Chapter 11 Amortized Analysis 533**

11.1 An Unrelated Puzzle 534  
11.2 Binomial Queues 534  
11.3 Skew Heaps 539  
11.4 Fibonacci Heaps 541  
11.4.1 Cutting Nodes in Leftist Heaps 542  
11.4.2 Lazy Merging for Binomial Queues 544  
11.4.3 The Fibonacci Heap Operations 548  
11.4.4 Proof of the Time Bound 549  
11.5 Splay Trees 551  
Summary 555  
Exercises 556  
References 557

## **Chapter 12 Advanced Data Structures and Implementation 559**

12.1 Top-Down Splay Trees 559  
12.2 Red-Black Trees 566

|        |  |     |
|--------|--|-----|
| 12.2.1 | Bottom-Up Insertion  | 567 |
| 12.2.2 | Top-Down Red-Black Trees                                   | 568 |
| 12.2.3 | Top-Down Deletion  | 570 |
| 12.3   | Treaps   | 576 |
| 12.4   | Suffix Arrays and Suffix Trees                             | 579 |
| 12.4.1 | Suffix Arrays  | 580 |
| 12.4.2 | Suffix Trees   | 583 |
| 12.4.3 | Linear-Time Construction of Suffix Arrays and Suffix Trees | 586 |
| 12.5   | $k$ -d Trees   | 596 |
| 12.6   | Pairing Heaps  | 602 |
|        | Summary  | 606 |
|        | Exercises  | 608 |
|        | References   | 612 |

## **Appendix A Separate Compilation of Class Templates 615**

|     |                          |     |
|-----|--------------------------|-----|
| A.1 | Everything in the Header | 616 |
| A.2 | Explicit Instantiation   | 616 |
|     | Index                    | 619 |