Contents

Foreword vi

Preface viii

Authors' Profiles xix

List of Abbreviations xx

List of Tables xxi

List of Figures xxii

1 Introduction **1**
- 1.1 Competitive Programming . . . 1
- 1.2 Tips to be Competitive . . . 3
  - 1.2.1 Tip 1: Type Code Faster! . . . 3
  - 1.2.2 Tip 2: Quickly Identify Problem Types . . . 4
  - 1.2.3 Tip 3: Do Algorithm Analysis . . . 6
  - 1.2.4 Tip 4: Master Programming Languages . . . 10
  - 1.2.5 Tip 5: Master the Art of Testing Code . . . 13
  - 1.2.6 Tip 6: Practice and More Practice . . . 15
  - 1.2.7 Tip 7: Team Work (for ICPC) . . . 16
- 1.3 Getting Started: The Easy Problems . . . 16
  - 1.3.1 Anatomy of a Programming Contest Problem . . . 16
  - 1.3.2 Typical Input/Output Routines . . . 17
  - 1.3.3 Time to Start the Journey . . . 19
- 1.4 The Ad Hoc Problems . . . 21
- 1.5 Solutions to Non-Starred Exercises . . . 27
- 1.6 Chapter Notes . . . 32

2 Data Structures and Libraries **33**
- 2.1 Overview and Motivation . . . 33
- 2.2 Linear DS with Built-in Libraries . . . 35
- 2.3 Non-Linear DS with Built-in Libraries . . . 43
- 2.4 Data Structures with Our Own Libraries . . . 49
  - 2.4.1 Graph . . . 49
  - 2.4.2 Union-Find Disjoint Sets . . . 52
  - 2.4.3 Segment Tree . . . 55
  - 2.4.4 Binary Indexed (Fenwick) Tree . . . 59
- 2.5 Solution to Non-Starred Exercises . . . 64
- 2.6 Chapter Notes . . . 67

3 Problem Solving Paradigms 69

3.1 Overview and Motivation 69

3.2 Complete Search 70

3.2.1 Iterative Complete Search 71

3.2.2 Recursive Complete Search 74

3.2.3 Tips 76

3.3 Divide and Conquer 84

3.3.1 Interesting Usages of Binary Search 84

3.4 Greedy 89

3.4.1 Examples 89

3.5 Dynamic Programming 95

3.5.1 DP Illustration 95

3.5.2 Classical Examples 103

3.5.3 Non-Classical Examples 112

3.6 Solution to Non-Starred Exercises 118

3.7 Chapter Notes 120

4 Graph 121

4.1 Overview and Motivation 121

4.2 Graph Traversal 122

4.2.1 Depth First Search (DFS) 122

4.2.2 Breadth First Search (BFS) 123

4.2.3 Finding Connected Components (Undirected Graph) 125

4.2.4 Flood Fill - Labeling/Coloring the Connected Components 125

4.2.5 Topological Sort (Directed Acyclic Graph) 126

4.2.6 Bipartite Graph Check 128

4.2.7 Graph Edges Property Check via DFS Spanning Tree 128

4.2.8 Finding Articulation Points and Bridges (Undirected Graph) 130

4.2.9 Finding Strongly Connected Components (Directed Graph) 133

4.3 Minimum Spanning Tree 138

4.3.1 Overview and Motivation 138

4.3.2 Kruskal's Algorithm 138

4.3.3 Prim's Algorithm 139

4.3.4 Other Applications 141

4.4 Single-Source Shortest Paths 146

4.4.1 Overview and Motivation 146

4.4.2 SSSP on Unweighted Graph 146

4.4.3 SSSP on Weighted Graph 148

4.4.4 SSSP on Graph with Negative Weight Cycle 151

4.5 All-Pairs Shortest Paths 155

4.5.1 Overview and Motivation 155

4.5.2 Explanation of Floyd Warshall's DP Solution 156

4.5.3 Other Applications 158

4.6 Network Flow 163

4.6.1 Overview and Motivation 163

4.6.2 Ford Fulkerson's Method 163

4.6.3 Edmonds Karp's Algorithm 164

4.6.4 Flow Graph Modeling - Part 1 166

4.6.5 Other Applications 167

4.6.6 Flow Graph Modeling - Part 2 168

4.7 Special Graphs . . . 171
4.7.1 Directed Acyclic Graph . . . 171
4.7.2 Tree . . . 178
4.7.3 Eulerian Graph . . . 179
4.7.4 Bipartite Graph . . . 180
4.8 Solution to Non-Starred Exercises . . . 187
4.9 Chapter Notes . . . 190

5 Mathematics 191
5.1 Overview and Motivation . . . 191
5.2 Ad Hoc Mathematics Problems . . . 192
5.3 Java BigInteger Class . . . 198
5.3.1 Basic Features . . . 198
5.3.2 Bonus Features . . . 199
5.4 Combinatorics . . . 204
5.4.1 Fibonacci Numbers . . . 204
5.4.2 Binomial Coefficients . . . 205
5.4.3 Catalan Numbers . . . 205
5.4.4 Remarks about Combinatorics in Programming Contests . . . 206
5.5 Number Theory . . . 210
5.5.1 Prime Numbers . . . 210
5.5.2 Greatest Common Divisor & Least Common Multiple . . . 211
5.5.3 Factorial . . . 212
5.5.4 Finding Prime Factors with Optimized Trial Divisions . . . 212
5.5.5 Working with Prime Factors . . . 213
5.5.6 Functions Involving Prime Factors . . . 214
5.5.7 Modified Sieve . . . 216
5.5.8 Modulo Arithmetic . . . 216
5.5.9 Extended Euclid: Solving Linear Diophantine Equation . . . 217
5.5.10 Remarks about Number Theory in Programming Contests . . . 217
5.6 Probability Theory . . . 221
5.7 Cycle-Finding . . . 223
5.7.1 Solution(s) using Efficient Data Structure . . . 223
5.7.2 Floyd's Cycle-Finding Algorithm . . . 223
5.8 Game Theory . . . 226
5.8.1 Decision Tree . . . 226
5.8.2 Mathematical Insights to Speed-up the Solution . . . 227
5.8.3 Nim Game . . . 228
5.9 Solution to Non-Starred Exercises . . . 229
5.10 Chapter Notes . . . 231

6 String Processing 233
6.1 Overview and Motivation . . . 233
6.2 Basic String Processing Skills . . . 234
6.3 Ad Hoc String Processing Problems . . . 236
6.4 String Matching . . . 241
6.4.1 Library Solutions . . . 241
6.4.2 Knuth-Morris-Pratt's (KMP) Algorithm . . . 241
6.4.3 String Matching in a 2D Grid . . . 244
6.5 String Processing with Dynamic Programming . . . 245

6.5.1 String Alignment (Edit Distance) 245
6.5.2 Longest Common Subsequence 247
6.5.3 Non Classical String Processing with DP 247
6.6 Suffix Trie/Tree/Array 249
6.6.1 Suffix Trie and Applications 249
6.6.2 Suffix Tree 250
6.6.3 Applications of Suffix Tree 251
6.6.4 Suffix Array 253
6.6.5 Applications of Suffix Array 258
6.7 Solution to Non-Starred Exercises 264
6.8 Chapter Notes 267

7 (Computational) Geometry 269
7.1 Overview and Motivation 269
7.2 Basic Geometry Objects with Libraries 271
7.2.1 0D Objects: Points 271
7.2.2 1D Objects: Lines 272
7.2.3 2D Objects: Circles 276
7.2.4 2D Objects: Triangles 278
7.2.5 2D Objects: Quadrilaterals 281
7.3 Algorithm on Polygon with Libraries 285
7.3.1 Polygon Representation 285
7.3.2 Perimeter of a Polygon 285
7.3.3 Area of a Polygon 285
7.3.4 Checking if a Polygon is Convex 286
7.3.5 Checking if a Point is Inside a Polygon 287
7.3.6 Cutting Polygon with a Straight Line 288
7.3.7 Finding the Convex Hull of a Set of Points 289
7.4 Solution to Non-Starred Exercises 294
7.5 Chapter Notes 297

8 More Advanced Topics 299
8.1 Overview and Motivation 299
8.2 More Advanced Search Techniques 299
8.2.1 Backtracking with Bitmask 299
8.2.2 Backtracking with Heavy Pruning 304
8.2.3 State-Space Search with BFS or Dijkstra's 305
8.2.4 Meet in the Middle (Bidirectional Search) 306
8.2.5 Informed Search: A* and IDA* 308
8.3 More Advanced DP Techniques 312
8.3.1 DP with Bitmask 312
8.3.2 Compilation of Common (DP) Parameters 313
8.3.3 Handling Negative Parameter Values with Offset Technique 313
8.3.4 MLE? Consider Using Balanced BST as Memo Table 315
8.3.5 MLE/TLE? Use Better State Representation 315
8.3.6 MLE/TLE? Drop One Parameter, Recover It from Others 316
8.4 Problem Decomposition 320
8.4.1 Two Components: Binary Search the Answer and Other 320
8.4.2 Two Components: Involving 1D Static RSQ/RMQ 322
8.4.3 Two Components: Graph Preprocessing and DP 322

8.4.4 Two Components: Involving Graph . . . 324
8.4.5 Two Components: Involving Mathematics . . . 324
8.4.6 Two Components: Complete Search and Geometry . . . 324
8.4.7 Two Components: Involving Efficient Data Structure . . . 324
8.4.8 Three Components . . . 325
8.5 Solution to Non-Starred Exercises . . . 332
8.6 Chapter Notes . . . 333

9 Rare Topics 335
9.1 2-SAT Problem . . . 336
9.2 Art Gallery Problem . . . 338
9.3 Bitonic Traveling Salesman Problem . . . 339
9.4 Bracket Matching . . . 341
9.5 Chinese Postman Problem . . . 342
9.6 Closest Pair Problem . . . 343
9.7 Dinic's Algorithm . . . 344
9.8 Formulas or Theorems . . . 345
9.9 Gaussian Elimination Algorithm . . . 346
9.10 Graph Matching . . . 349
9.11 Great-Circle Distance . . . 352
9.12 Hopcroft Karp's Algorithm . . . 353
9.13 Independent and Edge-Disjoint Paths . . . 354
9.14 Inversion Index . . . 355
9.15 Josephus Problem . . . 356
9.16 Knight Moves . . . 357
9.17 Kosaraju's Algorithm . . . 358
9.18 Lowest Common Ancestor . . . 359
9.19 Magic Square Construction (Odd Size) . . . 361
9.20 Matrix Chain Multiplication . . . 362
9.21 Matrix Power . . . 364
9.22 Max Weighted Independent Set . . . 368
9.23 Min Cost (Max) Flow . . . 369
9.24 Min Path Cover on DAG . . . 370
9.25 Pancake Sorting . . . 371
9.26 Pollard's rho Integer Factoring Algorithm . . . 374
9.27 Postfix Calculator and Conversion . . . 376
9.28 Roman Numerals . . . 378
9.29 Selection Problem . . . 380
9.30 Shortest Path Faster Algorithm . . . 383
9.31 Sliding Window . . . 384
9.32 Sorting in Linear Time . . . 386
9.33 Sparse Table Data Structure . . . 388
9.34 Tower of Hanoi . . . 390
9.35 Chapter Notes . . . 391

A uHunt 393

B Credits 396

Bibliography 398

Foreword

A long time ago (on the 11th of November in 2003, Tuesday, 3:55:57 UTC), I received an e-mail with the following message:

> "I should say in a simple word that with the UVa Site, you have given birth to a new CIVILIZATION and with the books you write (he meant "Programming Challenges: The Programming Contest Training Manual" [60], coauthored with Steven Skiena), you inspire the soldiers to carry on marching. May you live long to serve the humanity by producing super-human programmers."

Although that was clearly an exaggeration, it did cause me to think. I had a dream: to create a community around the project I had started as a part of my teaching job at UVa, with people from all around the world working together towards the same ideal. With a little searching, I quickly found a whole online community running a web-ring of sites with excellent tools that cover and provide whatever the UVa site lacked.

To me, 'Methods to Solve' by Steven Halim, a very young student from Indonesia, was one of the more impressive websites. I was inspired to believe that the dream would become real one day, because in this website lay the result of the hard work of a genius of algorithms and informatics. Moreover, his declared objectives matched the core of my dream: to serve humanity. Even better, he has a brother with similar interests and capabilities, Felix Halim.

It's a pity that it takes so much time to start a real collaboration, but life is like that. Fortunately, all of us have continued working together in a parallel fashion towards the realization of that dream—the book that you have in your hands now is proof of that.

I can't imagine a better complement for the UVa Online Judge. This book uses lots of examples from UVa carefully selected and categorized both by problem type and solving technique, providing incredibly useful help for the users of the site. By mastering and practicing most programming exercises in this book, a reader can easily solve at least 500 problems in the UVa Online Judge, which will place them in the top 400-500 amongst ≈100000 UVa OJ users.

It's clear that the book "Competitive Programming: Increasing the Lower Bound of Programming Contests" is suitable for programmers who want to improve their ranks in upcoming ICPC regionals and IOIs. The two authors have gone through these contests (ICPC and IOI) themselves as contestants and now as coaches. But it's also an essential colleague for newcomers—as Steven and Felix say in the introduction 'the book is not meant to be read once, but several times'.

Moreover, it contains practical C++ source code to implement given algorithms. Understanding a problem is one thing, but knowing the algorithm to solve it is another, and implementing the solution well in short and efficient code is tricky. After you have read this extraordinary book three times you will realize that you are a much better programmer and, more importantly, a happier person.