Contents

Foreword **v**

Preface **vi**

Authors' Profiles and Copyright **xi**

Convention and Problem Categorization **xii**

List of Abbreviations **xiii**

List of Tables **xiv**

List of Figures **xv**

1 Introduction **1**
- 1.1 Competitive Programming . . . 1
- 1.2 Tips to be Competitive . . . 2
  - 1.2.1 Tip 2: Quickly Identify Problem Types . . . 4
  - 1.2.2 Tip 3: Do Algorithm Analysis . . . 5
  - 1.2.3 Tip 4: Master Programming Languages . . . 8
  - 1.2.4 Tip 5: Master the Art of Testing Code . . . 10
  - 1.2.5 Tip 6: Practice and More Practice . . . 12
  - 1.2.6 Tip 7: Team Work (ICPC Only) . . . 12
- 1.3 Getting Started: The Ad Hoc Problems . . . 13
- 1.4 Chapter Notes . . . 19

2 Data Structures and Libraries **21**
- 2.1 Overview and Motivation . . . 21
- 2.2 Data Structures with Built-in Libraries . . . 22
  - 2.2.1 Linear Data Structures . . . 22
  - 2.2.2 Non-Linear Data Structures . . . 26
- 2.3 Data Structures with Our-Own Libraries . . . 29
  - 2.3.1 Graph . . . 29
  - 2.3.2 Union-Find Disjoint Sets . . . 30
  - 2.3.3 Segment Tree . . . 32
  - 2.3.4 Fenwick Tree . . . 35
- 2.4 Chapter Notes . . . 38

3 Problem Solving Paradigms **39**
- 3.1 Overview and Motivation . . . 39
- 3.2 Complete Search . . . 39
  - 3.2.1 Examples . . . 40
  - 3.2.2 Tips . . . 41
- 3.3 Divide and Conquer . . . 47

3.3.1 Interesting Usages of Binary Search . . . 47
3.4 Greedy . . . 51
3.4.1 Examples . . . 51
3.5 Dynamic Programming . . . 55
3.5.1 DP Illustration . . . 55
3.5.2 Classical Examples . . . 61
3.5.3 Non Classical Examples . . . 66
3.6 Chapter Notes . . . 70

4 Graph **71**
4.1 Overview and Motivation . . . 71
4.2 Graph Traversal . . . 71
4.2.1 Depth First Search (DFS) . . . 71
4.2.2 Breadth First Search (BFS) . . . 72
4.2.3 Finding Connected Components (in an Undirected Graph) . . . 73
4.2.4 Flood Fill - Labeling/Coloring the Connected Components . . . 74
4.2.5 Topological Sort (of a Directed Acyclic Graph) . . . 75
4.2.6 Bipartite Graph Check . . . 76
4.2.7 Graph Edges Property Check via DFS Spanning Tree . . . 76
4.2.8 Finding Articulation Points and Bridges (in an Undirected Graph) . . . 77
4.2.9 Finding Strongly Connected Components (in a Directed Graph) . . . 80
4.3 Minimum Spanning Tree . . . 84
4.3.1 Overview and Motivation . . . 84
4.3.2 Kruskal's Algorithm . . . 84
4.3.3 Prim's Algorithm . . . 85
4.3.4 Other Applications . . . 86
4.4 Single-Source Shortest Paths . . . 90
4.4.1 Overview and Motivation . . . 90
4.4.2 SSSP on Unweighted Graph . . . 90
4.4.3 SSSP on Weighted Graph . . . 91
4.4.4 SSSP on Graph with Negative Weight Cycle . . . 93
4.5 All-Pairs Shortest Paths . . . 96
4.5.1 Overview and Motivation . . . 96
4.5.2 Explanation of Floyd Warshall's DP Solution . . . 96
4.5.3 Other Applications . . . 98
4.6 Maximum Flow . . . 101
4.6.1 Overview and Motivation . . . 101
4.6.2 Ford Fulkerson's Method . . . 101
4.6.3 Edmonds Karp's . . . 102
4.6.4 Other Applications . . . 104
4.7 Special Graphs . . . 107
4.7.1 Directed Acyclic Graph . . . 107
4.7.2 Tree . . . 112
4.7.3 Eulerian Graph . . . 113
4.7.4 Bipartite Graph . . . 114
4.8 Chapter Notes . . . 119

5 Mathematics **121**
5.1 Overview and Motivation . . . 121
5.2 Ad Hoc Mathematics Problems . . . 121
5.3 Java BigInteger Class . . . 125
5.3.1 Basic Features . . . 125
5.3.2 Bonus Features . . . 126

5.4 Combinatorics 129
5.4.1 Fibonacci Numbers 129
5.4.2 Binomial Coefficients 130
5.4.3 Catalan Numbers 131
5.4.4 Other Combinatorics 131
5.5 Number Theory 133
5.5.1 Prime Numbers 133
5.5.2 Greatest Common Divisor (GCD) & Least Common Multiple (LCM) 135
5.5.3 Factorial 136
5.5.4 Finding Prime Factors with Optimized Trial Divisions 136
5.5.5 Working with Prime Factors 137
5.5.6 Functions Involving Prime Factors 138
5.5.7 Modulo Arithmetic 140
5.5.8 Extended Euclid: Solving Linear Diophantine Equation 141
5.5.9 Other Number Theoretic Problems 142
5.6 Probability Theory 142
5.7 Cycle-Finding 143
5.7.1 Solution using Efficient Data Structure 143
5.7.2 Floyd's Cycle-Finding Algorithm 143
5.8 Game Theory 145
5.8.1 Decision Tree 145
5.8.2 Mathematical Insights to Speed-up the Solution 146
5.8.3 Nim Game 146
5.9 Powers of a (Square) Matrix 147
5.9.1 The Idea of Efficient Exponentiation 147
5.9.2 Square Matrix Exponentiation 148
5.10 Chapter Notes 149

6 String Processing **151**
6.1 Overview and Motivation 151
6.2 Basic String Processing Skills 151
6.3 Ad Hoc String Processing Problems 153
6.4 String Matching 156
6.4.1 Library Solution 156
6.4.2 Knuth-Morris-Pratt (KMP) Algorithm 156
6.4.3 String Matching in a 2D Grid 158
6.5 String Processing with Dynamic Programming 160
6.5.1 String Alignment (Edit Distance) 160
6.5.2 Longest Common Subsequence 161
6.5.3 Palindrome 162
6.6 Suffix Trie/Tree/Array 163
6.6.1 Suffix Trie and Applications 163
6.6.2 Suffix Tree 163
6.6.3 Applications of Suffix Tree 164
6.6.4 Suffix Array 166
6.6.5 Applications of Suffix Array 170
6.7 Chapter Notes 174

7 (Computational) Geometry **175**
7.1 Overview and Motivation 175
7.2 Basic Geometry Objects with Libraries 176
7.2.1 0D Objects: Points 176
7.2.2 1D Objects: Lines 177

7.2.3 2D Objects: Circles . . . 181
7.2.4 2D Objects: Triangles . . . 183
7.2.5 2D Objects: Quadrilaterals . . . 185
7.2.6 3D Objects: Spheres . . . 186
7.2.7 3D Objects: Others . . . 187
7.3 Polygons with Libraries . . . 188
7.3.1 Polygon Representation . . . 188
7.3.2 Perimeter of a Polygon . . . 188
7.3.3 Area of a Polygon . . . 188
7.3.4 Checking if a Polygon is Convex . . . 189
7.3.5 Checking if a Point is Inside a Polygon . . . 189
7.3.6 Cutting Polygon with a Straight Line . . . 190
7.3.7 Finding the Convex Hull of a Set of Points . . . 191
7.4 Divide and Conquer Revisited . . . 195
7.5 Chapter Notes . . . 196

8 More Advanced Topics 197
8.1 Overview and Motivation . . . 197
8.2 Problem Decomposition . . . 197
8.2.1 Two Components: Binary Search the Answer and Other . . . 197
8.2.2 Two Components: SSSP and DP . . . 198
8.2.3 Two Components: Involving Graph . . . 199
8.2.4 Two Components: Involving Mathematics . . . 199
8.2.5 Three Components: Prime Factors, DP, Binary Search . . . 199
8.2.6 Three Components: Complete Search, Binary Search, Greedy . . . 199
8.3 More Advanced Search Techniques . . . 203
8.3.1 Informed Search: A* . . . 203
8.3.2 Depth Limited Search . . . 204
8.3.3 Iterative Deepening Search . . . 204
8.3.4 Iterative Deepening A* (IDA*) . . . 204
8.4 More Advanced Dynamic Programming Techniques . . . 205
8.4.1 Emerging Technique: DP + bitmask . . . 205
8.4.2 Chinese Postman/Route Inspection Problem . . . 205
8.4.3 Compilation of Common DP States . . . 206
8.4.4 MLE/TLE? Use Better State Representation! . . . 207
8.4.5 MLE/TLE? Drop One Parameter, Recover It from Others! . . . 208
8.4.6 Your Parameter Values Go Negative? Use Offset Technique . . . 209
8.5 Chapter Notes . . . 211

A Hints/Brief Solutions 213

B uHunt 225

C Credits 227

D Plan for the Third Edition 228

Bibliography 229

Foreword

Long time ago (exactly the Tuesday November 11th 2003 at 3:55:57 UTC), I received an e-mail with the following sentence: 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" [34], 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 it's clear that was an exaggeration, to tell the truth I started thinking a bit about and I had a dream: to create a community around the project I had started as a part of my teaching job at UVa, with persons from everywhere around the world to work together after that ideal. Just by searching in Internet I immediately found a lot of people who was already creating a web-ring of sites with excellent tools to cover the many lacks of the UVa site.

The more impressive to me was the 'Methods to Solve' from Steven Halim, a very young student from Indonesia and I started to believe that the dream would become real a day, because the contents of the site were the result of a hard work of a genius of algorithms and informatics. Moreover his declared objectives matched the main part of my dream: to serve the humanity. And the best of the best, he has a brother with similar interest and capabilities, Felix Halim.

It's a pity it takes so many time to start a real collaboration, but the life is as it is. Fortunately, all of us have continued working in a parallel way and the book that you have in your hands is the best proof.

I can't imagine a better complement for the UVa Online Judge site, as it uses lots of examples from there carefully selected and categorized both by problem type and solving techniques, an incredible useful help for the users of the site. By mastering and practicing most programming exercises in this book, reader can easily go to 500 problems solved in UVa online judge, which will place them in top 400-500 within ≈100000 UVa OJ users.

Then it's clear that the book "Competitive Programming: Increasing the Lower Bound of Programming Contests" is suitable for programmers who wants 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 the newcomers, because 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 codes to implement the given algorithms. Because understand the problems is a thing, knowing the algorithms is another, and implementing them well in short and efficient code is tricky. After you read this extraordinary book three times you will realize that you are a much better programmer and, more important, a more happy person.

@ ACM ICPC World Finals Harbin 2010
L-R: Carlos, Miguel Jr, Steven, Miguel Revilla

Miguel A. Revilla, University of Valladolid
UVa Online Judge site creator; ACM-ICPC International Steering Committee Member and Problem Archivist