TEXTS IN COMPUTER SCIENCE

Editors
David Gries
Fred B. Schneider

Springer
New York
Berlin
Heidelberg
Hong Kong
London
Milan
Paris
Tokyo

Steven S. Skiena Miguel A. Revilla

PROGRAMMING CHALLENGES

The Programming Contest Training Manual

With 65 Illustrations

Springer

Steven S. Skiena
Department of Computer Science
SUNY Stony Brook
Stony Brook, NY 11794-4400, USA
skiena@programming-challenges.com

Miguel A. Revilla
Department of Applied Mathematics
and Computer Science
Faculty of Sciences
University of Valladolid
Valladolid, 47011, SPAIN
revilla@programming-challenges.com

Series Editors:
David Gries
Department of Computer Science
415 Boyd Graduate Studies
Research Center
The University of Georgia
Athens, GA 30602-7404, USA

Fred B. Schneider
Department of Computer Science
Upson Hall
Cornell University
Ithaca, NY 14853-7501, USA

Cover illustration: "Spectator," by William Rose © 2002.

Library of Congress Cataloging-in-Publication Data
Skeina, Steven S.
Programming challenges : the programming contest training manual / Steven S. Skiena, Miguel A. Revilla.
p. cm. — (Texts in computer science)
Includes bibliographical references and index.
ISBN 0-387-00163-8 (softcover : alk. paper)
1. Computer programming. I. Revilla, Miguel A. II. Title. III. Series.
QA76.6.S598 2003
005.1—dc21 2002044523

ISBN 0-387-00163-8 Printed on acid-free paper.

© 2003 Springer-Verlag New York, Inc.
All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY 10010, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.
The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed in the United States of America.

9 8 7 6 5 4 3 2 1 SPIN 10901052

Photocomposed pages prepared by the author using Springer-Verlag's LaTeX macros.

www.springer-ny.com

Springer-Verlag New York Berlin Heidelberg
A member of BertelsmannSpringer Science+Business Media GmbH

To our wives, Renee and Carmela,
and children, Bonnie, Emilio, and Miguel.

The challenges in this book
are not nearly as difficult
as the challenge of making enough time
for those who we love.

Preface

There are many distinct pleasures associated with computer programming. Craftsmanship has its quiet rewards, the satisfaction that comes from building a useful object and making it work. Excitement arrives with the flash of insight that cracks a previously intractable problem. The spiritual quest for elegance can turn the hacker into an artist. There are pleasures in parsimony, in squeezing the last drop of performance out of clever algorithms and tight coding.

The games, puzzles, and challenges of problems from international programming competitions are a great way to experience these pleasures while improving your algorithmic and coding skills. This book contains over 100 problems that have appeared in previous programming contests, along with discussions of the theory and ideas necessary to attack them. Instant online grading for all of these problems is available from two WWW robot judging sites. Combining this book with a judge gives an exciting new way to challenge and improve your programming skills.

This book can be used for self-study, for teaching innovative courses in algorithms and programming, and in training for international competition.

To the Reader

The problems in this book have been selected from over 1,000 programming problems at the Universidad de Valladolid online judge, available at *http://online-judge.uva.es*. The judge has ruled on well over one million submissions from 27,000 registered users around the world to date. We have taken only the best of the best, the most fun, exciting, and interesting problems available.

We have organized these problems by topic and provided enough tutorial material (primarily in mathematics and algorithms) to give you a fair chance to solve them.

Sample programs are provided to illustrate many important concepts. By reading this book and trying the problems you will gain a concrete understanding of algorithmic techniques such as backtracking and dynamic programming, and advanced topics such as number theory and computational geometry. These subjects are well worth your attention even if you never intend to compete in programming contests.

Many of the problems are flat-out fun. They address fascinating topics in computer science and mathematics, sometimes disguised by an amusing story. These make interesting subjects for additional study, so we provide notes with further readings where appropriate.

We have found that people whose training is in the pragmatics of programming and software engineering often fail to appreciate the power of algorithmics. Similarly, the theoretically inclined typically underestimate what it takes to turn an algorithm into a program, and how clever programming can make short work of a tough problem.

For this reason, the first portion of the book focuses primarily on programming techniques, such as the proper use of data types and program libraries. This lays the foundation for the more algorithmic sections in the second part of the book. Mastery of both is required to be a complete problem solver.

To the Instructor

This book has been designed to serve as a textbook for three types of courses:

- Algorithm courses focusing on programming.
- Programming courses focusing on algorithms.
- Elective courses designed to train students to participate in competitions such as the Association for Computing Machinery (ACM) International Collegiate Programming Contest and the International Olympiad in Informatics.

Such courses can be a lot of fun for all involved. Students are easily motivated by the thrill of competition, and get positive feedback each time the judge accepts their solution. The most obvious algorithm may result in a "Time Limit Exceeded" message from the judge, thus motivating a search for efficiency. The correct insight can make for a dozen-line program instead of a huge mass of code. The best students will be inspired to try extra problems just for kicks.

Such courses are fun to teach, too. Many problems are quite clever, putting a fresh face on standard topics in programming and algorithms. Finding the best solution requires insight and inspiration. It is exciting to figure out the right way to do each of the problems, and even more exciting when the students figure it out for themselves.

Pedagogical features of this book include:

- *Complements Standard Algorithm Texts* — Although this book is self-contained, it has been written with the understanding that most students will have some prior exposure to algorithm design. This book has been designed (and priced) so it can serve as a supplementary text for traditional algorithms courses, complementing abstract descriptions with concrete implementations and theoretical analysis with

hands-on experience. Further, it covers several interesting topics that are not universally included in standard algorithm texts.

- *Provides Complete Implementations of Classical Algorithms* — Many students have a difficult time going from abstract algorithm descriptions to working code. To help them, we provide carefully written implementations of all important algorithms we discuss using a subset of C designed to be easily readable by C++ and Java programmers. Several of our programming challenge problems can be solved by appropriately modifying these routines, thus providing a concrete path to get students started.

- *Integrated Course Management Environment* — We have created a special course management environment that makes it shamefully easy to administer such a course, as it will handle all testing and grading for you! Our website *http://www.programming-challenges.com* lets you assign problems to students, maintain rosters, view each student's score and programs, and even detect suspicious similarity among their solutions!

- *Help for Students at All Levels* — The challenges included in this book have been selected to span a wide range of difficulty. Many are suitable for introductory students, while others will prove challenging to those ready for international competition. Hints for most problems are provided.

  To help identify the most appropriate problems for any given student, we have annotated each problem with three distinct measures of difficulty. The *popularity* of a problem (A, B, or C) refers to how many people try it, while the *success rate* (low to high) measures how often they succeed. Finally, the *level* of a problem (1 to 4, corresponding roughly from freshman to senior) indicates how advanced a student needs to be in order to have a fair chance of solving the problem.

To the Coach or Competitor

This book has been particularly designed to serve as a training manual for programming competitions at the high school and collegiate levels. We provide a convenient summary/reference of important topics in mathematics and algorithms, along with appropriate challenges to help you master the material.

The robot judge checks the correctness of submitted programs just like the human judges of the ACM International Collegiate Programming Contest. Once you set up a personal account with the judge, you can submit solutions written in C, C++, Pascal, or Java and wait for the verdict announcing success or failure. The judge keeps statistics on how you are doing, so you can compare yourself to the thousands of other participants.

To help the competitor, we include an appendix with training secrets from finalists for the three major programming contest venues: the ACM International Collegiate Programming Contest (ICPC), the International Olympiad in Informatics (IOI), and the TopCoder Programmer Challenge. We present the history of these competitions, show how you can get involved, and help you make your best possible showing.

Roughly 80% of all finalists in the most recent ACM contest trained on the Universidad de Valladolid online judge. That the finals are held in exotic locals like Hawaii provides extra incentive to study. Good luck!

Associated Websites

This book has been designed to work hand-in-hand with two websites. Online grading for all problems is available at *http://www.programming-challenges.com*, along with lots of supporting material. In particular, we provide complete source code of all the programs that appear in the text as well as lecture notes to help integrate this material into courses.

All of the problems in this book (and many, many more) can also be graded by the Universidad de Valladolid online judge, *http://online-judge.uva.es*. In particular each programming challenge in this book has been given an ID number on both judging websites, so you can take advantage of their special features.

Acknowledgments

The existence of this book is due in great part to the generosity of all the people who let us incorporate their contest problems into the robot judge as well as in this book. No less than 17 people contributed problems to this volume, from four different continents. We are particularly indebted to Gordon Cormack and Shahriar Manzoor, problem posers on the scale of Sam Loyd and H. E. Dudeney!

A complete mapping of people to problems appears in the appendix, but we particularly thank the following contest organizers for their contributions: Gordon Cormack (38 problems), Shahriar Manzoor (28), Miguel Revilla (10), Pedro Demasi (8), Manuel Carro (4), Rujia Liu (4), Petko Minkov (4), Owen Astrakan (3), Alexander Denisjuk (3), Long Chong (2), Ralf Engels (2), Alex Gevak (1), Walter Guttmann (1), Arun Kishore (1), Erick Moreno (1), Udvranto Patik (1), and Marcin Wojciechowski (1). Several of these problems were developed by third parties, who are acknowledged in the appendix.

Tracking down the original authors of some of these problems proved almost as difficult as tracking down the author of the Bible. We have tried very hard to identify the author of each problem, and in each case received permission from someone claiming to speak for the author. We apologize in advance if there are any oversights. If so, please let us know so we can award proper credit.

The robot judge project is the work of many hands. Ciriaco García is the primary author of the robot judge software and a key supporter of the project. Fernando P. Nájera is responsible for many of the tools that help the judge in a friendly manner. Carlos M. Casas maintains the correctness of the test files, ensuring that they are both fair and demanding. José A. Caminero and Jesús Paúl help with problem curation and solution integrity. We particularly thank Miguel Revilla, Jr. for building and maintaining the *http://www.programming-challenges.com* website.

This book was partially debugged during a course taught at Stony Brook by Vinhthuy Phan and Pavel Sumazin in spring 2002. The students from our terrific programing teams this year (Larry Mak, Dan Ports, Tom Rothamel, Alexey Smirnov, Jeffrey

Versoza, and Charles Wright) helped review the manuscript and we thank them for their interest and feedback. Haowen Zhang made a significant contribution by carefully reading the manuscript, testing the programs, and tightening the code.

We thank Wayne Yuhasz, Wayne Wheeler, Frank Ganz, Lesley Poliner, and Rich Putter of Springer-Verlag for all their help turning a manuscript into a published book. We thank Gordon Cormack, Lauren Cowles, David Gries, Joe O'Rourke, Saurabh Sethia, Tom Verhoeff, Daniel Wright, and Stan Wagon for thoughtful manuscript reviews that significantly improved the final product. The Fulbright Foundation and the Department of Applied Mathematics and Computation at the Universidad de Valladolid provided essential support, enabling the two authors to work together face to face. Citigroup CIB, through the efforts of Peter Remch and Debby Z. Beckman, significantly contributed to the ACM ICPC effort at Stony Brook. Its involvement helped spark the writing of this book.

Steven S. Skiena
Stony Brook, NY

Miguel A. Revilla
Valladolid, Spain

February 2003

Contents

1 Getting Started **1**

1.1 Getting Started With the Judge . . . 1

1.1.1 The Programming Challenges Robot Judge . . . 2

1.1.2 The Universidad de Valladolid Robot Judge . . . 2

1.1.3 Feedback From the Judge . . . 3

1.2 Choosing Your Weapon . . . 4

1.2.1 Programming Languages . . . 5

1.2.2 Reading Our Programs . . . 6

1.2.3 Standard Input/Output . . . 7

1.3 Programming Hints . . . 9

1.4 Elementary Data Types . . . 11

1.5 About the Problems . . . 13

1.6 Problems . . . 15

1.6.1 The $3n + 1$ Problem . . . 15

1.6.2 Minesweeper . . . 16

1.6.3 The Trip . . . 17

1.6.4 LCD Display . . . 18

1.6.5 Graphical Editor . . . 19

1.6.6 Interpreter . . . 21

1.6.7 Check the Check . . . 23

1.6.8 Australian Voting . . . 25

1.7 Hints . . . 26

1.8 Notes . . . 26

2 Data Structures **27**
2.1 Elementary Data Structures . . . 27
2.1.1 Stacks . . . 28
2.1.2 Queues . . . 28
2.1.3 Dictionaries . . . 30
2.1.4 Priority Queues . . . 31
2.1.5 Sets . . . 32
2.2 Object Libraries . . . 33
2.2.1 The C++ Standard Template Library . . . 33
2.2.2 The Java `java.util` Package . . . 33
2.3 Program Design Example: Going to War . . . 34
2.4 Hitting the Deck . . . 35
2.5 String Input/Output . . . 37
2.6 Winning the War . . . 38
2.7 Testing and Debugging . . . 39
2.8 Problems . . . 42
2.8.1 Jolly Jumpers . . . 42
2.8.2 Poker Hands . . . 43
2.8.3 Hartals . . . 45
2.8.4 Crypt Kicker . . . 47
2.8.5 Stack 'em Up . . . 48
2.8.6 Erdös Numbers . . . 50
2.8.7 Contest Scoreboard . . . 52
2.8.8 Yahtzee . . . 53
2.9 Hints . . . 55
2.10 Notes . . . 55

3 Strings **56**
3.1 Character Codes . . . 56
3.2 Representing Strings . . . 58
3.3 Program Design Example: Corporate Renamings . . . 59
3.4 Searching for Patterns . . . 61
3.5 Manipulating Strings . . . 62
3.6 Completing the Merger . . . 63
3.7 String Library Functions . . . 64
3.8 Problems . . . 66
3.8.1 WERTYU . . . 66
3.8.2 Where's Waldorf? . . . 67
3.8.3 Common Permutation . . . 69
3.8.4 Crypt Kicker II . . . 70
3.8.5 Automated Judge Script . . . 71
3.8.6 File Fragmentation . . . 73
3.8.7 Doublets . . . 74
3.8.8 Fmt . . . 75
3.9 Hints . . . 77

3.10 Notes . . . 77

4 Sorting **78**
4.1 Sorting Applications . . . 78
4.2 Sorting Algorithms . . . 79
4.3 Program Design Example: Rating the Field . . . 82
4.4 Sorting Library Functions . . . 83
4.5 Rating the Field . . . 85
4.6 Problems . . . 88
4.6.1 Vito's Family . . . 88
4.6.2 Stacks of Flapjacks . . . 89
4.6.3 Bridge . . . 91
4.6.4 Longest Nap . . . 92
4.6.5 Shoemaker's Problem . . . 94
4.6.6 CDVII . . . 95
4.6.7 ShellSort . . . 97
4.6.8 Football (aka Soccer) . . . 99
4.7 Hints . . . 101
4.8 Notes . . . 101

5 Arithmetic and Algebra **102**
5.1 Machine Arithmetic . . . 102
5.1.1 Integer Libraries . . . 103
5.2 High-Precision Integers . . . 103
5.3 High-Precision Arithmetic . . . 105
5.4 Numerical Bases and Conversion . . . 110
5.5 Real Numbers . . . 112
5.5.1 Dealing With Real Numbers . . . 113
5.5.2 Fractions . . . 113
5.5.3 Decimals . . . 114
5.6 Algebra . . . 115
5.6.1 Manipulating Polynomials . . . 115
5.6.2 Root Finding . . . 116
5.7 Logarithms . . . 116
5.8 Real Mathematical Libraries . . . 117
5.9 Problems . . . 119
5.9.1 Primary Arithmetic . . . 119
5.9.2 Reverse and Add . . . 120
5.9.3 The Archeologist's Dilemma . . . 121
5.9.4 Ones . . . 122
5.9.5 A Multiplication Game . . . 123
5.9.6 Polynomial Coefficients . . . 124
5.9.7 The Stern-Brocot Number System . . . 125
5.9.8 Pairsumonious Numbers . . . 127
5.10 Hints . . . 128

5.11 Notes 128

6 Combinatorics 129
6.1 Basic Counting Techniques 129
6.2 Recurrence Relations 131
6.3 Binomial Coefficients 131
6.4 Other Counting Sequences 133
6.5 Recursion and Induction 135
6.6 Problems 137
6.6.1 How Many Fibs? 137
6.6.2 How Many Pieces of Land? 138
6.6.3 Counting 139
6.6.4 Expressions 140
6.6.5 Complete Tree Labeling 141
6.6.6 The Priest Mathematician 142
6.6.7 Self-describing Sequence 144
6.6.8 Steps 145
6.7 Hints 146
6.8 Notes 146

7 Number Theory 147
7.1 Prime Numbers 147
7.1.1 Finding Primes 148
7.1.2 Counting Primes 149
7.2 Divisibility 149
7.2.1 Greatest Common Divisor 150
7.2.2 Least Common Multiple 151
7.3 Modular Arithmetic 152
7.4 Congruences 154
7.4.1 Operations on Congruences 154
7.4.2 Solving Linear Congruences 155
7.4.3 Diophantine Equations 155
7.5 Number Theoretic Libraries 156
7.6 Problems 157
7.6.1 Light, More Light 157
7.6.2 Carmichael Numbers 158
7.6.3 Euclid Problem 159
7.6.4 Factovisors 160
7.6.5 Summation of Four Primes 161
7.6.6 Smith Numbers 162
7.6.7 Marbles 163
7.6.8 Repackaging 164
7.7 Hints 166
7.8 Notes 166

8 Backtracking **167**
8.1 Backtracking . . . 167
8.2 Constructing All Subsets . . . 169
8.3 Constructing All Permutations . . . 170
8.4 Program Design Example: The Eight-Queens Problem . . . 172
8.5 Pruning Search . . . 173
8.6 Problems . . . 176
8.6.1 Little Bishops . . . 176
8.6.2 15-Puzzle Problem . . . 177
8.6.3 Queue . . . 179
8.6.4 Servicing Stations . . . 180
8.6.5 Tug of War . . . 181
8.6.6 Garden of Eden . . . 182
8.6.7 Color Hash . . . 184
8.6.8 Bigger Square Please... . . . 186
8.7 Hints . . . 188
8.8 Notes . . . 188

9 Graph Traversal **189**
9.1 Flavors of Graphs . . . 189
9.2 Data Structures for Graphs . . . 191
9.3 Graph Traversal: Breadth-First . . . 194
9.3.1 Breadth-First Search . . . 194
9.3.2 Exploiting Traversal . . . 195
9.3.3 Finding Paths . . . 196
9.4 Graph Traversal: Depth-First . . . 198
9.4.1 Finding Cycles . . . 198
9.4.2 Connected Components . . . 199
9.5 Topological Sorting . . . 200
9.6 Problems . . . 203
9.6.1 Bicoloring . . . 203
9.6.2 Playing With Wheels . . . 204
9.6.3 The Tourist Guide . . . 206
9.6.4 Slash Maze . . . 208
9.6.5 Edit Step Ladders . . . 210
9.6.6 Tower of Cubes . . . 211
9.6.7 From Dusk Till Dawn . . . 213
9.6.8 Hanoi Tower Troubles Again! . . . 215
9.7 Hints . . . 216

10 Graph Algorithms **217**
10.1 Graph Theory . . . 217
10.1.1 Degree Properties . . . 217
10.1.2 Connectivity . . . 218
10.1.3 Cycles in Graphs . . . 219

10.1.4 Planar Graphs . . . 220
10.2 Minimum Spanning Trees . . . 220
10.3 Shortest Paths . . . 223
10.3.1 Dijkstra's Algorithm . . . 223
10.3.2 All-Pairs Shortest Path . . . 225
10.4 Network Flows and Bipartite Matching . . . 227
10.5 Problems . . . 231
10.5.1 Freckles . . . 231
10.5.2 The Necklace . . . 232
10.5.3 Fire Station . . . 234
10.5.4 Railroads . . . 235
10.5.5 War . . . 237
10.5.6 Tourist Guide . . . 239
10.5.7 The Grand Dinner . . . 241
10.5.8 The Problem With the Problem Setter . . . 242
10.6 Hints . . . 244

11 Dynamic Programming 245
11.1 Don't Be Greedy . . . 245
11.2 Edit Distance . . . 246
11.3 Reconstructing the Path . . . 250
11.4 Varieties of Edit Distance . . . 251
11.5 Program Design Example: Elevator Optimization . . . 253
11.6 Problems . . . 257
11.6.1 Is Bigger Smarter? . . . 257
11.6.2 Distinct Subsequences . . . 258
11.6.3 Weights and Measures . . . 259
11.6.4 Unidirectional TSP . . . 260
11.6.5 Cutting Sticks . . . 262
11.6.6 Ferry Loading . . . 263
11.6.7 Chopsticks . . . 265
11.6.8 Adventures in Moving: Part IV . . . 266
11.7 Hints . . . 267
11.8 Notes . . . 267

12 Grids 268
12.1 Rectilinear Grids . . . 268
12.1.1 Traversal . . . 269
12.1.2 Dual Graphs and Representations . . . 270
12.2 Triangular and Hexagonal Grids . . . 271
12.2.1 Triangular Lattices . . . 271
12.2.2 Hexagonal Lattices . . . 272
12.3 Program Design Example: Plate Weight . . . 275
12.4 Circle Packings . . . 277
12.5 Longitude and Latitude . . . 278

12.6 Problems . . . 279
12.6.1 Ant on a Chessboard . . . 279
12.6.2 The Monocycle . . . 280
12.6.3 Star . . . 282
12.6.4 Bee Maja . . . 283
12.6.5 Robbery . . . 284
12.6.6 (2/3/4)-D Sqr/Rects/Cubes/Boxes? . . . 286
12.6.7 Dermuba Triangle . . . 287
12.6.8 Airlines . . . 288
12.7 Hints . . . 290

13 Geometry **291**
13.1 Lines . . . 291
13.2 Triangles and Trigonometry . . . 294
13.2.1 Right Triangles and the Pythagorean Theorem . . . 295
13.2.2 Trigonometric Functions . . . 295
13.2.3 Solving Triangles . . . 296
13.3 Circles . . . 298
13.4 Program Design Example: Faster Than a Speeding Bullet . . . 299
13.5 Trigonometric Function Libraries . . . 302
13.6 Problems . . . 304
13.6.1 Dog and Gopher . . . 304
13.6.2 Rope Crisis in Ropeland! . . . 305
13.6.3 The Knights of the Round Table . . . 306
13.6.4 Chocolate Chip Cookies . . . 307
13.6.5 Birthday Cake . . . 308
13.6.6 The Largest/Smallest Box 309
13.6.7 Is This Integration? . . . 310
13.6.8 How Big Is It? . . . 311
13.7 Hints . . . 312

14 Computational Geometry **313**
14.1 Line Segments and Intersection . . . 313
14.2 Polygons and Angle Computations . . . 315
14.3 Convex Hulls . . . 316
14.4 Triangulation: Algorithms and Related Problems . . . 319
14.4.1 Van Gogh's Algorithm . . . 320
14.4.2 Area Computations . . . 322
14.4.3 Point Location . . . 322
14.5 Algorithms on Grids . . . 324
14.5.1 Range Queries . . . 324
14.5.2 Lattice Polygons and Pick's Theorem . . . 325
14.6 Geometry Libraries . . . 326
14.7 Problems . . . 327
14.7.1 Herding Frosh . . . 327

14.7.2 The Closest Pair Problem 328
14.7.3 Chainsaw Massacre 329
14.7.4 Hotter Colder 330
14.7.5 Useless Tile Packers 331
14.7.6 Radar Tracking 333
14.7.7 Trees on My Island 334
14.7.8 Nice Milk 336
14.8 Hints 337

A Appendix **339**
A.1 The ACM International Collegiate Programming Contest 339
A.1.1 Preparation 340
A.1.2 Strategies and Tactics 341
A.2 International Olympiad in Informatics 343
A.2.1 Participation 343
A.2.2 Format 344
A.2.3 Preparation 345
A.3 Topcoder.com 345
A.4 Go to Graduate School! 346
A.5 Problem Credits 348

References **350**

Index **353**