algorithmic PUZZLES

anany levitin | maria levitin

Algorithmic Puzzles

This page intentionally left blank

ALGORITHMIC PUZZLES

Anany Levitin
and
Maria Levitin

OXFORD
UNIVERSITY PRESS

OXFORD
UNIVERSITY PRESS

Oxford University Press, Inc., publishes works that further
Oxford University's objective of excellence
in research, scholarship, and education.

Oxford New York
Auckland Cape Town Dar es Salaam Hong Kong Karachi
Kuala Lumpur Madrid Melbourne Mexico City Nairobi
New Delhi Shanghai Taipei Toronto

With offices in
Argentina Austria Brazil Chile Czech Republic France Greece
Guatemala Hungary Italy Japan Poland Portugal Singapore
South Korea Switzerland Thailand Turkey Ukraine Vietnam

Copyright © 2011 by Oxford University Press

Published by Oxford University Press, Inc.
198 Madison Avenue, New York, New York 10016
www.oup.com

Oxford is a registered trademark of Oxford University Press

All rights reserved. No part of this publication may be reproduced,
stored in a retrieval system, or transmitted, in any form or by any means,
electronic, mechanical, photocopying, recording, or otherwise,
without the prior permission of Oxford University Press.

Library of Congress Cataloging-in-Publication Data

Levitin, Anany.
Algorithmic puzzles / Anany Levitin, Maria Levitin.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-19-974044-4 (pbk.)
1. Mathematical recreations. 2. Algorithms.
I. Levitin, Maria. II. Title.
QA95.L475 2011
793.74—dc22 2010052043

9 8 7 6 5 4 3 2 1

Printed in the United States of America
on acid-free paper

To Max with love

This page intentionally left blank

Contents

Preface ix
Acknowledgments xiii
List of Puzzles xv
Tutorial Puzzles xv
Main Section Puzzles xvi
The Epigraph Puzzle: Who said what? xxi

1. Tutorials 3
General Strategies for Algorithm Design 3
Analysis Techniques 22

2. Puzzles 32
Easier Puzzles (#1 to #50) 32
Puzzles of Medium Difficulty (#51 to #110) 45
Harder Puzzles (#111 to #150) 60

3. Hints 72

4. Solutions 82

References 241
Design Strategy and Analysis Index 247
Index of Terms and Names 254

This page intentionally left blank

Preface in Questions and Answers

WHAT IS THIS BOOK ABOUT?

This book is a collection of algorithmic puzzles—puzzles that involve, explicitly or implicitly, clearly defined procedures for solving problems. It is a unique collection of such puzzles. The book includes some old classics, which have become a part of mathematics and computer science folklore. It also contains newer examples, some of which have been asked during job interviews at major companies.

The book has two main goals:

- To entertain a wide range of readers interested in puzzles
- To promote development of high-level algorithmic thinking (with no computer programming), supported by a carefully developed list of general algorithm design strategies and analysis techniques

Although algorithms do constitute the cornerstone of computer science and no sensible computer programming is possible without them, it is a common misconception to equate the two. Some algorithmic puzzles predate computers by more than a thousand years. It is true, however, that the proliferation of computers has made algorithmic problem solving important in many areas of modern life, from hard and soft sciences to art and entertainment. Solving algorithmic puzzles is the most productive and definitely most enjoyable way to develop and strengthen one's algorithmic thinking skills.

WHOM IS THIS BOOK FOR?

There are three large categories of readers who should be interested in this book:

- Puzzle lovers
- People interested in developing algorithmic thinking, including teachers and students
- People preparing for interviews with companies giving puzzles as well as people conducting such interviews

All we have to say to puzzle lovers is to reassure them that they could enjoy this collection as they would a collection not dedicated to any particular theme or type of puzzle. They will encounter a few all-time favorites, but, hopefully, will also find a number of little-known puzzle gems. No computing background

or even an interest in it is assumed; such a reader can simply ignore references to specific algorithm design strategies and analysis techniques in the solutions given.

Algorithmic thinking has recently become somewhat of a buzz word among computer science educators, and with some justice: ubiquity of computers in today's world does make algorithmic thinking a very important skill for almost any student. Puzzles are an ideal vehicle for mastering this important skill for two reasons. First, puzzles are fun, and a person is normally willing to put more effort into solving them than in doing routine exercises. Second, algorithmic puzzles force a solver to think on a more abstract level. Even computer science students have a tendency to think about algorithmic problems in terms of a computer language they know instead of applying general design and analysis strategies. Puzzles can rectify this important deficiency.

The puzzles in this book can certainly be used for individual study. Together with the tutorials, they provide, in our view, a good introduction to main algorithmic ideas. They can also be used by teachers of computing courses—both at the college and secondary school level—as supplemental exercises and project topics. The book might also be of interest for problem-solving courses, especially those based on puzzles.

As to people preparing for interviews, they should find the book helpful in two ways. First, it contains many examples of puzzles they may encounter, with complete solutions and comments. Second, the book also provides concise tutorials on algorithm design strategies and analysis techniques. After all, managers offering puzzles during interviews claim that they are more interested in the way an interviewee approaches a puzzle than in an actual solution to it. Showing an expertise in applying general design strategies and analysis techniques should then be a highly attractive way to impress the potential employer.

WHAT PUZZLES ARE INCLUDED IN THE BOOK?

Algorithmic puzzles constitute a small fraction among thousands of mathematical puzzles invented over the years. In selecting puzzles for the book, we have sought puzzles that satisfy the following criteria.

First, we wanted puzzles that illustrate some general principle in design or analysis of algorithms.

Second, we were looking for beauty and elegance, the subjectivity of those qualities notwithstanding.

Third, we wanted the puzzles to run a wide range of difficulty levels. Puzzle difficulty is hard to pinpoint; math professors have been occasionally stumped by puzzles easily solved by middle school students. Still, we have divided the book's puzzles into three sections—Easier Puzzles, Puzzles of Medium Difficulty, and Harder Puzzles—to give readers some help in gauging the puzzles' difficulty. Within each of these three sections, we have tried to order the puzzles in increasing level of difficulty as well. The puzzles in the Easier Puzzles section require only middle school mathematics. Although solutions to a few problems in the other two

sections do use proofs by induction, high school mathematics should, in general, suffice for solving all the book's puzzles. In addition, topics such as binary numbers and simple recurrence relations are briefly reviewed in the second tutorial. This does not mean, of course, that all the puzzles in the book are easy. Some of them—especially those at the end of the last section—are truly hard. But their difficulty does not lie in some sophisticated mathematics, and the reader should not be intimidated by them.

Fourth, we have felt compelled to include a few puzzles because of their historical importance. Finally, we only included puzzles with clear statements and solutions devoid of any tricks such as intentional ambiguity, word play, and so on.

One more important comment needs to be made here. Many puzzles in this book can be solved by exhaustive search or backtracking. (These strategies are explained in the book's first tutorial.) It is *not* the approach the reader is expected to employ to solve the puzzles, unless explicitly stated otherwise. Therefore, we have excluded categories of puzzles such as Sudoku and cryptarithms, which have to be solved either by exhaustive search/backtracking or by some ingenious insight in the specific data given in the puzzle. We have also decided against inclusion of puzzles based on some physical objects that are not very easy to describe, such as the Chinese Rings and Rubik's Cube.

HINTS, SOLUTIONS, AND COMMENTS

The book contains hints, solutions, and comments for every puzzle. Puzzle books rarely include hints, but we see them as a valuable addition. Hints may provide a small push in a right direction, still leaving the reader with a chance to solve the puzzle. All the hints are collected at the end of the book in a separate section.

Solutions are provided to every puzzle. As a rule, they start with a short answer. This is done to provide the reader with the last opportunity to solve the puzzle on his or her own: if the reader's answer disagrees with the one in the book, the reader can stop reading the complete solution and try to solve the puzzle again.

Algorithms are described in free-style English, with no special formatting or pseudocode notations. The emphasis is on the ideas rather than insignificant details. Rewriting the solutions in a more formal manner may, in fact, provide useful exercises in their own right.

Most comments point out a general algorithmic idea that the puzzle and its solution illustrate. Occasionally, they also include references to similar puzzles in the book and elsewhere.

Many puzzle books do not indicate the puzzle sources. The reason usually given is that trying to find an author of a puzzle is akin to trying to find an author of a joke. While there is a lot of truth to this observation, we have decided to mention the earliest sources of the puzzles known to us. The reader should keep in mind, however, that we have not conducted anything close to an extensive search for the puzzles' origin; doing that would have resulted in a very different book.

WHAT ARE THE TWO TUTORIALS ABOUT?

The book includes two tutorials, with puzzle examples, on general strategies for algorithm design and techniques for algorithm analysis. Although almost all puzzles in the book can be solved without any knowledge of the topics discussed in these tutorials, there is no question that they can make solving the puzzles much easier and, importantly, more useful. Besides, solutions, comments, and a few hints use some special terminology explained in the tutorials.

The tutorials are written on the most elementary level possible to make them comprehensible for a wide variety of readers. A reader with a computer science degree will hardly find there anything new, except, possibly, for puzzle examples. At the same time, such a reader might use them as a concise refresher of the fundamental ideas in the design and analysis of algorithms.

WHY ARE THERE TWO INDICES IN THE BOOK?

In addition to a standard index, the book contains an index indicating puzzles based on a particular design strategy or a type of analysis. This index should help the reader to locate problems on a specific strategy or technique and can also serve as a list of additional hints.

We conclude with a hope that the readers will find the book both enjoyable and useful. We also hope they will share our delight in beauty of and amazing feats of human ingenuity behind many of the book's puzzles.

Anany Levitin
Maria Levitin
May 2011
algorithmicpuzzles.book@gmail.com

Acknowledgments

We would like to express our deep gratitude to the book's reviewers: Tim Chartier (Davidson College), Stephen Lucas (James Madison University), and Laura Taalman (James Madison University). Their enthusiastic support of the book's idea and specific suggestions about its contents have certainly been very helpful to us.

We are also thankful to Simon Berkovich of the George Washington University for several discussions of the puzzle topics and for reading a portion of the book's manuscript.

Our thanks go to all the people at the Oxford University Press and their associates who worked on the book. We are especially grateful to our editor, Phyllis Cohen, for her ceaseless efforts to make the book better. We are also thankful to the editorial assistant, Hallie Stebbins, the book's cover designer, Natalya Balnova, and the marketing manager, Michelle Kelly. We appreciate the work of Richard Camp, the book's copy editor, as well as the efforts of Jennifer Kowing and Kiran Kumar who supervised the book's production.

This page intentionally left blank

List of Puzzles

TUTORIAL PUZZLES

The list below contains all the puzzles in the book's two tutorials. The puzzles are listed in the order of their appearance. The page numbers indicate locations of the puzzles; their solutions are given directly in the tutorials following the puzzle statements.

Magic Square 4
The n-Queens Problem 6
Celebrity Problem 8
Number Guessing (Twenty Questions) 9
Tromino Puzzle 10
Anagram Detection 11
Cash Envelopes 12
Two Jealous Husbands 12
Guarini's Puzzle 14
Optimal Pie Cutting 15
Non-Attacking Kings 16
Bridge Crossing at Night 17
Lemonade Stand Placement 18
Positive Changes 19
Shortest Path Counting 20
Chess Invention 23
Square Build-Up 24
Tower of Hanoi 25
Domino Tiling of Deficient Chessboards 28
The Königsberg Bridges Problem 29
Breaking a Chocolate Bar 30
Chickens in the Corn 30

MAIN SECTION PUZZLES

This list contains all 150 puzzles included in the main section of the book. The page numbers indicate locations of the puzzle, hint, and solution with comments, respectively.

1. **A Wolf, a Goat, and a Cabbage** 32, 72, 82
2. **Glove Selection** 32, 72, 83
3. **Rectangle Dissection** 32, 72, 83
4. **Ferrying Soldiers** 32, 72, 84
5. **Row and Column Exchanges** 33, 72, 85
6. **Predicting a Finger Count** 33, 72, 85
7. **Bridge Crossing at Night** 33, 72, 86
8. **Jigsaw Puzzle Assembly** 33, 72, 87
9. **Mental Arithmetic** 34, 72, 87
10. **A Fake Among Eight Coins** 34, 73, 88
11. **A Stack of Fake Coins** 34, 73, 89
12. **Questionable Tiling** 34, 73, 90
13. **Blocked Paths** 34, 73, 90
14. **Chessboard Reassembly** 35, 73, 91
15. **Tromino Tilings** 35, 73, 92
16. **Making Pancakes** 36, 73, 93
17. **A King's Reach** 36, 73, 93
18. **A Corner-to-Corner Journey** 36, 73, 94
19. **Page Numbering** 36, 73, 94
20. **Maximum Sum Descent** 36, 73, 95
21. **Square Dissection** 37, 73, 96
22. **Team Ordering** 37, 73, 97
23. **Polish National Flag Problem** 37, 73, 97
24. **Chessboard Colorings** 37, 73, 98
25. **The Best Time to Be Alive** 37, 73, 99
26. **Find the Rank** 37, 73, 100
27. **The Icosian Game** 38, 73, 100

28. **Figure Tracing** 38, 74, 101

29. **Magic Square Revisited** 39, 74, 103

30. **Cutting a Stick** 39, 74, 105

31. **The Three Pile Trick** 39, 74, 105

32. **Single-Elimination Tournament** 40, 74, 106

33. **Magic and Pseudo-Magic** 40, 74, 106

34. **Coins on a Star** 40, 74, 107

35. **Three Jugs** 40, 74, 109

36. **Limited Diversity** 41, 74, 110

37. **2*n*-Counters Problem** 41, 74, 111

38. **Tetromino Tiling** 41, 74, 112

39. **Board Walks** 42, 74, 114

40. **Four Alternating Knights** 42, 74, 115

41. **The Circle of Lights** 42, 74, 115

42. **The Other Wolf-Goat-Cabbage Puzzle** 43, 74, 116

43. **Number Placement** 43, 74, 117

44. **Lighter or Heavier?** 43, 74, 117

45. **A Knight's Shortest Path** 43, 74, 118

46. **Tricolor Arrangement** 43, 74, 118

47. **Exhibition Planning** 43, 75, 119

48. **McNugget Numbers** 44, 75, 120

49. **Missionaries and Cannibals** 44, 75, 121

50. **Last Ball** 44, 75, 122

51. **Missing Number** 45, 75, 123

52. **Counting Triangles** 45, 75, 124

53. **Fake-Coin Detection with a Spring Scale** 45, 75, 124

54. **Cutting a Rectangular Board** 45, 75, 125

55. **Odometer Puzzle** 45, 75, 125

56. **Lining Up Recruits** 46, 75, 126

57. **Fibonacci's Rabbits Problem** 46, 75, 126

58. **Sorting Once, Sorting Twice** 46, 75, 128

59. **Hats of Two Colors** 46, 75, 129
60. **Squaring a Coin Triangle** 46, 75, 129
61. **Checkers on a Diagonal** 47, 76, 131
62. **Picking Up Coins** 47, 76, 132
63. **Pluses and Minuses** 47, 76, 133
64. **Creating Octagons** 47, 76, 134
65. **Code Guessing** 47, 76, 135
66. **Remaining Number** 48, 76, 136
67. **Averaging Down** 48, 76, 137
68. **Digit Sum** 48, 76, 137
69. **Chips on Sectors** 48, 76, 138
70. **Jumping into Pairs I** 48, 76, 139
71. **Marking Cells I** 48, 76, 139
72. **Marking Cells II** 49, 76, 140
73. **Rooster Chase** 49, 76, 141
74. **Site Selection** 50, 76, 143
75. **Gas Station Inspections** 50, 76, 144
76. **Efficient Rook** 51, 76, 145
77. **Searching for a Pattern** 51, 76, 146
78. **Straight Tromino Tiling** 51, 76, 147
79. **Locker Doors** 51, 77, 148
80. **The Prince's Tour** 51, 77, 148
81. **Celebrity Problem Revisited** 51, 77, 149
82. **Heads Up** 52, 77, 150
83. **Restricted Tower of Hanoi** 52, 77, 151
84. **Pancake Sorting** 52, 77, 153
85. **Rumor Spreading I** 53, 77, 155
86. **Rumor Spreading II** 53, 77, 156
87. **Upside-Down Glasses** 53, 77, 157
88. **Toads and Frogs** 53, 77, 157
89. **Counter Exchange** 53, 77, 159

90. **Seating Rearrangements** 54, 77, 160
91. **Horizontal and Vertical Dominoes** 54, 77, 161
92. **Trapezoid Tiling** 54, 77, 162
93. **Hitting a Battleship** 55, 77, 164
94. **Searching a Sorted Table** 55, 77, 165
95. **Max-Min Weights** 55, 77, 166
96. **Tiling a Staircase Region** 55, 77, 167
97. **The Game of Topswops** 55, 78, 169
98. **Palindrome Counting** 56, 78, 170
99. **Reversal of Sort** 56, 78, 171
100. **A Knight's Reach** 57, 78, 172
101. **Room Painting** 57, 78, 173
102. **The Monkey and the Coconuts** 57, 78, 174
103. **Jumping to the Other Side** 57, 78, 175
104. **Pile Splitting** 58, 78, 176
105. **The MU Puzzle** 58, 78, 178
106. **Turning on a Light Bulb** 59, 78, 178
107. **The Fox and the Hare** 59, 78, 180
108. **The Longest Route** 59, 78, 181
109. **Double-*n* Dominoes** 59, 78, 181
110. **The Chameleons** 59, 78, 183
111. **Inverting a Coin Triangle** 60, 78, 183
112. **Domino Tiling Revisited** 60, 78, 186
113. **Coin Removal** 60, 78, 187
114. **Crossing Dots** 61, 79, 188
115. **Bachet's Weights** 61, 79, 188
116. **Bye Counting** 61, 79, 190
117. **One-Dimensional Solitaire** 62, 79, 192
118. **Six Knights** 62, 79, 193
119. **Colored Tromino Tiling** 62, 79, 195
120. **Penny Distribution Machine** 62, 79, 196

121. **Super-Egg Testing** 63, 79, 197

122. **Parliament Pacification** 63, 79, 198

123. **Dutch National Flag Problem** 63, 79, 199

124. **Chain Cutting** 63, 79, 199

125. **Sorting 5 in 7** 64, 79, 202

126. **Dividing a Cake Fairly** 64, 79, 203

127. **The Knight's Tour** 64, 79, 204

128. **Security Switches** 64, 80, 205

129. **Reve's Puzzle** 64, 80, 207

130. **Poisoned Wine** 64, 80, 209

131. **Tait's Counter Puzzle** 65, 80, 209

132. **The Solitaire Army** 65, 80, 211

133. **The Game of Life** 65, 80, 214

134. **Point Coloring** 66, 80, 215

135. **Different Pairings** 66, 80, 216

136. **Catching a Spy** 66, 80, 217

137. **Jumping into Pairs II** 67, 80, 219

138. **Candy Sharing** 67, 80, 220

139. **King Arthur's Round Table** 67, 80, 221

140. **The *n*-Queens Problem Revisited** 67, 80, 222

141. **The Josephus Problem** 67, 80, 223

142. **Twelve Coins** 67, 80, 225

143. **Infected Chessboard** 68, 81, 227

144. **Killing Squares** 68, 81, 227

145. **The Fifteen Puzzle** 68, 81, 229

146. **Hitting a Moving Target** 69, 81, 231

147. **Hats with Numbers** 69, 81, 232

148. **One Coin for Freedom** 69, 81, 234

149. **Pebble Spreading** 69, 81, 236

150. **Bulgarian Solitaire** 70, 81, 238

THE EPIGRAPH PUZZLE: WHO SAID WHAT?

Match the following quotations with the authors listed below:

The man with a hammer sees every problem as a nail. Our age's great hammer is the algorithm.

Solving problems is a practical skill like, let us say, swimming. We acquire any practical skill by imitation and practice.

There is no better way to relieve the tedium than by injecting recreational topics into a course, topics strongly tinged with elements of play, humor, beauty, and surprise.

It is not knowledge, but the act of learning, not possession but the act of getting there, which grants the greatest enjoyment.

If I have perchance omitted anything more or less proper or necessary, I beg indulgence, since there is no one who is blameless and utterly provident in all things.

William Poundstone, the author of *How Would You Move Mount Fuji? Microsoft's Cult of the Puzzle: How the World's Smartest Companies Select the Most Creative Thinkers*

George Pólya (1887–1985), a prominent Hungarian mathematician, the author of *How To Solve It,* the classic book on problem solving

Martin Gardner (1914–2010), an American writer, best known for his "Mathematical Games" column in *Scientific American* and books on recreational mathematics

Carl Friedrich Gauss (1777–1855), a great German mathematician

Leonardo of Pisa a.k.a. Fibonacci (1170–c1250), a remarkable Italian mathematician, the author of *Liber Abaci* ("The Book of Calculation"), one of the most consequential mathematical book in history

This page intentionally left blank

Algorithmic Puzzles

This page intentionally left blank

1 Tutorials

GENERAL STRATEGIES FOR ALGORITHM DESIGN

The purpose of this tutorial is to briefly review a few general strategies for designing algorithms. While these strategies are not all applicable to every puzzle, taken collectively they provide a powerful tool kit. Not surprisingly, these strategies are also used for solving many problems in computer science. Therefore, learning to apply these strategies to puzzles can serve as an excellent introduction to this important field.

But before we embark on reviewing major algorithm design strategies, we need to make an important comment on two types of algorithmic puzzles. Every algorithmic puzzle has an input. An input defines an *instance* of the puzzle. The instance can be either specific (e.g., find a false coin among eight coins with a balance) or general (e.g., find a false coin among n coins with a balance). When dealing with a specific instance of a puzzle, the solver has no obligations beyond solving the instance given. In fact, it might be the case that other instances of the puzzle do not have the same solution or even do not have solutions at all. On the other hand, specific numbers in a puzzle's statement may be of no significance whatsoever. Then solving the general instance of the puzzle could be not only more satisfying but, on occasion, even easier. But whether a puzzle is presented by a specific instance or given in its most general form, it is almost always a good idea to solve a few small instances of it anyway. On rare occasions the solver might be misled by such investigation, but much more often it can provide useful insights into the puzzle given.

Exhaustive Search

Theoretically, many puzzles can be solved by *exhaustive search*—a problem-solving strategy that simply tries all possible candidate solutions until a solution to the problem is found. Little ingenuity is typically required in applying exhaustive search. Therefore, puzzles are rarely offered to a person (as opposed to a computer) in the expectation that a solution will be found by applying this strategy. The most important limitation of exhaustive search is its inefficiency: as a rule, the number of solution candidates that need to be processed grows at least exponentially with the problem size, making the approach inappropriate not only for a human but often for a computer as well. As an example, consider the problem of constructing a *magic square* of order 3.

Magic Square Fill the 3×3 table with nine distinct integers from 1 to 9 so that the sum of the numbers in each row, column, and corner-to-corner diagonal is the same (Figure 1.1).

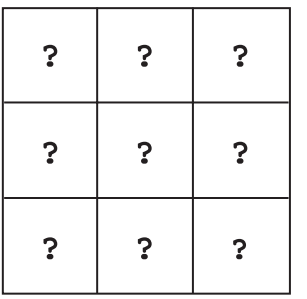


FIGURE 1.1 3×3 table to be filled with integers 1 through 9 to form a magic square.

How many ways are there to fill such a table? Let us think of the table as filled with one number at a time, starting with placing the 1 somewhere and ending with placing the 9. There are nine ways to place 1, followed by eight ways to place 2, and so on until the last number 9 is placed in the only unoccupied cell of the table. Hence, there are $9! = 9 \cdot 8 \cdot \ldots \cdot 1 = 362{,}880$ ways to arrange the nine numbers in the cells of the 3×3 table. (We just used the standard notation, $n!$, called *n factorial*, for the product of consecutive integers from 1 to n.) Therefore, solving this problem by exhaustive search would imply generating all 362,880 possible arrangements of distinct integers from 1 to 9 in the table and checking, for each of the arrangements, whether all its row, column, and diagonal sums are the same. This amount of work is clearly impossible to do by hand.

Actually, it is not difficult to solve this puzzle by proving first that the value of the common sum is equal to 15 and that 5 must be put at the center cell (see the *Magic Square Revisited* puzzle (#29) in the main section of the book). Alternatively, one can take advantage of several known algorithms for constructing

magic squares of an arbitrary order $n \geq 3$, which are especially efficient for odd n's (e.g., [Pic02]). Of course, these algorithms are not based on exhaustive search: the number of candidate solutions the exhaustive search algorithm would have to consider becomes prohibitively large even for a computer for n as small as 5. Indeed, $(5^2)! \simeq 1.5 \cdot 10^{25}$, and hence it would take a computer making 10 trillion operations per second about 49,000 years to finish the job.

Backtracking

There are two major difficulties in applying exhaustive search. The first one lies in the mechanics of generating all possible solution candidates. For some problems, such candidates compose a well-structured set. For example, candidate arrangements of the first nine positive integers in the cells of the 3×3 table (see the *Magic Square* example above) can be obtained as *permutations* of these numbers, for which several algorithms are known. There are many problems, however, where solution candidates do not form a set with such a regular structure. The second, and more fundamental, difficulty lies in the number of solution candidates that need to be generated and processed. Typically, the size of this set grows at least exponentially with the problem size. Therefore, exhaustive search is practical only for very small instances of such problems.

Backtracking is an important improvement over the brute-force approach of exhaustive search. It provides a convenient method for generating candidate solutions while making it possible to avoid generating unnecessary candidates. The main idea is to construct solutions one component at a time and evaluate such partially constructed candidates as follows: If a partially constructed solution can be developed further without violating the problem's constraints, it is done by taking the first remaining legitimate option for the next component. If there is no legitimate option for the next component, no alternatives for *any* remaining component need to be considered. In this case, the algorithm backtracks to replace the last component of the partially constructed solution with the next option for that component.

Typically, backtracking involves undoing a number of wrong choices—the smaller this number, the faster the algorithm finds a solution. Although in the worst-case scenario a backtracking algorithm may end up generating all the same candidate solutions as an exhaustive search, this rarely happens.

It is convenient to interpret a backtracking algorithm as a process of constructing a tree that mirrors decisions being made. Computer scientists use the term *tree* to describe hierarchical structures such as family trees and organizational charts. A tree is usually shown with its *root* (the only node without a parent) on the top and its *leaves* (nodes without children) on or closer to the bottom of the diagram. This is nothing but a convenient typographical convention, however. For a backtracking algorithm, such a tree is called a *state-space tree*. The root of a state-space tree corresponds to the start of a solution construction process; we consider the root to be on the zero level of the tree. The root's children—on the first level of the tree—correspond to possible choices of the first component

of a solution (e.g., the cell to contain 1 in the magic square construction). Their children—the nodes on the second level—correspond to possible choices of the second component of a solution, and so on. Leaves can be of two kinds. The first kind—called *nonpromising nodes* or *dead ends*—correspond to partially constructed candidates that cannot lead to a solution. After establishing that a particular node is nonpromising, a backtracking algorithm terminates the node (the tree is said to be *pruned*), undoes the decision regarding the last component of the candidate solution by backtracking to the parent of the nonpromising node, and considers another choice for that component. The second kind of a leaf provides a solution to the problem. If a single solution suffices, the algorithm stops; if other solutions need to be searched for, the algorithm continues searching for them by backtracking to the leaf's parent.

The following example is a perennial favorite for showing an application of backtracking to a particular problem.

The n-Queens Problem Place n queens on an $n \times n$ chessboard so that no two queens attack each other by being in the same column, row, or diagonal.

For $n = 1$, the problem has a trivial solution, and it is easy to see that there is no solution for $n = 2$ and $n = 3$. So let us consider the 4-queens problem and solve it by backtracking. Since each of the four queens has to be placed in its own column, all we need to do is to assign a row for each queen on the board shown in Figure 1.2.

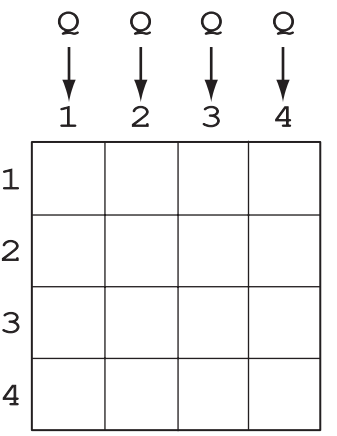


FIGURE 1.2 Board for the 4-queens problem.

We start with the empty board and then place queen 1 in the first possible position, which is in row 1 of column 1. Then we place queen 2, after trying unsuccessfully rows 1 and 2 of the second column, in the first acceptable position for it, which is square (3, 2), the square in row 3 and column 2. This proves to be a dead end because there is no acceptable position in the third column for

queen 3. Therefore, the algorithm backtracks and puts queen 2 in the next possible position (4, 2). Then queen 3 is placed at (2, 3), which proves to be another dead end. The algorithm then backtracks all the way to queen 1 and moves it to (2, 1). Queen 2 then goes to (4, 2), queen 3 to (1, 3), and queen 4 to (3, 4), which is a solution to the problem. The state-space tree of this search is given in Figure 1.3.

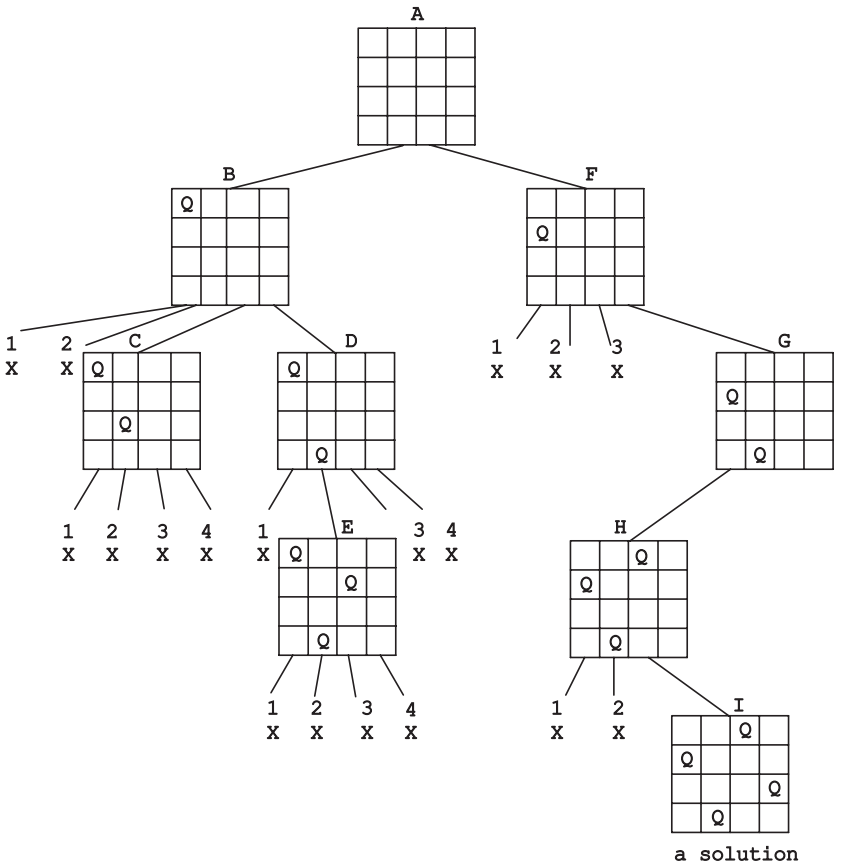


FIGURE 1.3 State-space tree of finding a solution to the 4-queens problem by backtracking. X denotes an unsuccessful attempt to place a queen in the indicated row. The letters above the nodes show the order in which the nodes are generated.

If other solutions need to be found (there is just one other solution to the 4-queens problem), the algorithm can simply resume its operations at the leaf at which it stopped. Alternatively, one can use the board's symmetry for this purpose.

How much faster is this solution by backtracking compared to exhaustive search? If we are to consider all possible placements of four queens on four different squares of the 4×4 board, the number of such placements is

$$\frac{16!}{4!(16-4)!} = \frac{16 \cdot 15 \cdot 14 \cdot 13}{4 \cdot 3 \cdot 2} = 1820.$$

(The general formula for the number of ways to choose k different objects, the order of which is not of interest, from a given set of n different objects, called by mathematicians *combinations* of n objects taken k at a time and denoted by either $\binom{n}{k}$ or $C(n, k)$, is $\frac{n!}{k!(n-k)!}$.) If we consider only the placements with the queens in different columns, the total number of solution candidates decreases to $4^4 = 256$. And if we add to the latter the constraint that the queens must also be in different rows, the number of choices drops to $4! = 24$. While the last number is quite manageable, it would not be the case for larger instances of the problem. For example, for a regular 8×8 chessboard, the number of such solution candidates is $8! = 40,320$.

The reader might be interested to know that the total number of different solutions to the 8-queens problem is 92, twelve of which are qualitatively distinct, with the remaining 80 obtainable from the basic twelve by rotations and reflections. As to the general n-queens problem, it has a solution for every $n \geq 4$ but no convenient formula for the number of solutions for an arbitrary n has been discovered. It is known that this number grows very fast with the value of n. For example, the number of solutions for $n = 10$ is 724, of which 92 are distinct, while for $n = 12$ the respective numbers are 14,200 and 1787.

Many puzzles in this book can be solved by backtracking. For each of them, however, there is a more efficient algorithm the reader is expected to strive for. In particular, *The n-Queens Problem Revisited* (#140) in the main section of the book asks the reader to design a much faster algorithm for the n-queens problem.

Decrease-and-Conquer

The *decrease-and-conquer* strategy is based on finding a relationship between a solution to a given problem and a solution to its smaller instance. Once found, such a relationship leads naturally to a *recursive algorithm*, which reduces the problem to a sequence of its diminishing instances until it becomes small enough to be solved directly.[1] Here is an example.

Celebrity Problem A celebrity among a group of n people is a person who knows nobody but is known by everybody else. The task is to identify a celebrity by only asking questions to people of the following form: "Do you know this person?"

Assuming for simplicity that a celebrity is known to exist among a given group of n people, the problem can be solved by the following decrease-by-one algorithm. If $n = 1$, that one person is vacuously a celebrity by the definition. If $n > 1$, select two people from the group, say, A and B, and ask A whether he or she knows B. If A knows B, remove A from the remaining people who can be a celebrity; if A does

[1] The notion of a *recursion* is one of the most important in computer science. The reader unfamiliar with it will find a wealth of information by, say, following the references and links in the Wikipedia article on "recursion (computer science)."

not know B, remove B from this group. Then solve the problem recursively (i.e., by the same method) for the remaining group of $n - 1$ people who can be a celebrity.

As an easy exercise, the reader may want to solve the *Ferrying Soldiers* puzzle (#4) in the main section of the book.

In general, a smaller instance in the decrease-and-conquer paradigm need not necessarily be of size $n - 1$. Although *decrease-by-one* is the most common case of size reduction, there are examples of size reduction by a larger amount. We get a particularly fast algorithm if we manage reducing an instance size by a constant factor, for example, by half, on each iteration. A well-known example of such an algorithm arises in the following well-known game.

Number Guessing (*Twenty Questions*) Determine a selected integer in the range from 1 to n, inclusive, by asking questions with yes/no answers.

The fastest algorithm for this problem asks a question that reduces the size of the set containing the answer by about half on each iteration. For example, the first question can be whether the selected number is greater than $\lceil n/2 \rceil$, which is the standard notation for $n/2$ rounded up to the nearest integer.[2] If the answer is "no," the selected number is among the integers 1 to $\lceil n/2 \rceil$; if the answer is "yes," the selected number is among the integers $\lceil n/2 \rceil + 1$ to n. In either case, the algorithm reduced the problem of size n to an instance of the same problem of about half the size of the original instance. Repeating this step until the instance size is reduced to 1 solves the problem.

Since this algorithm reduces the size of an instance (the range of the numbers that still can contain the selected number) by about half on each iteration, it works amazingly fast. For example, for $n = 1{,}000{,}000$, the algorithm requires no more than 20 questions! As fast as it is, an algorithm would be even faster if it could reduce the instance size by a larger factor, say, 3.

A Fake Among Eight Coins (#10) in the main section of the book provides another illustration of the *decrease-by-constant-factor* variation of the decrease-and-conquer strategy and can serve as a good exercise here.

It should be noted that sometimes it is easier to exploit a relationship between larger and smaller instances bottom up. This means solving first the puzzle for the smallest possible instance, then for the next larger one, and so on. This method is sometimes called the *incremental approach*. For a specific example, see the first solution of the *Rectangle Dissection* puzzle (#3) in the book's main section.

Divide-and-Conquer

The *divide-and-conquer* strategy is to partition a problem into several smaller subproblems (usually of the same or related type and ideally of about the same

[2] $\lceil x \rceil$, called the *ceiling* of a real number x, is the smallest integer larger than or equal to x. For example, $\lceil 2.3 \rceil = 3$ and $\lceil 2 \rceil = 2$. $\lfloor x \rfloor$, called the *floor* of x, is the largest integer smaller than or equal to x. For example, $\lfloor 2.3 \rfloor = 2$ and $\lfloor 2 \rfloor = 2$.

size), solve each of them, and then, if necessary, combine their solutions to get a solution to the original problem. This strategy underlines many efficient algorithms for important problems in computer science. Surprisingly, there are not many puzzles solvable by divide-and-conquer algorithms. Here is a well-known example, however, that perfectly illustrates this strategy.

Tromino Puzzle Cover a $2^n \times 2^n$ board missing one square with right trominoes, which are L-shaped tiles formed by three adjacent squares. The missing square can be any of the board squares. Trominoes should cover all the squares except the missing one exactly with no overlaps.

The problem can be solved by a recursive divide-and-conquer algorithm that places a tromino at the center of the board in such a way that the problem's instance of size n is reduced to four instances of the same problem, each of size $n - 1$ (Figure 1.4). The algorithm stops after every 2×2 region with one missing square generated by it is covered with a single tromino.

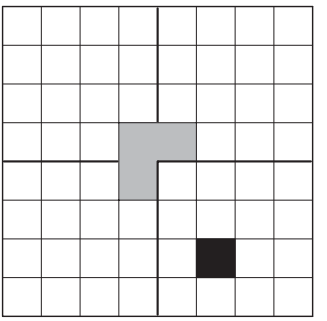


FIGURE 1.4 First step in tromino tiling of a $2^n \times 2^n$ board without one square by a divide-and-conquer algorithm.

The reader may want to finish the tiling of the 8×8 square in Figure 1.4 by this algorithm as a quick but useful exercise.

Most divide-and-conquer algorithms solve smaller subproblems recursively because, as in the above example, they represent smaller instances of the same problem. This need not always be the case, however. For some problems involving boards, in particular, a board may need to be divided into subboards that are not necessarily smaller versions of the board given. For such examples, see *2n-Counters Problem* (#37) and *Straight Tromino Tiling* (#78) in the main section of the book.

One more comment needs to be made about the divide-and-conquer strategy. Although some people consider decrease-and-conquer (discussed above) as a special case of divide-and-conquer, it is better to consider it as distinct design strategy. The crucial difference between the two lies in the number of smaller subproblems that need to be solved on each step: several in divide-and-conquer algorithms, and just one in decrease-and-conquer algorithms.

Transform-and-Conquer

The *transform-and-conquer* is a well-known approach to problem solving that is based on the idea of transformation. A problem is solved in two stages. First, in the transformation stage, it is modified or transformed into another problem that, for one reason or another, is more amenable to solution. Then, in the second, conquering stage, it is solved. In our realm of algorithmic problem solving, one can identify three varieties of this strategy. The first variety—called *instance simplification*—solves a problem by first transforming an instance given into another instance of the same problem with some special property that makes the problem easier to solve. The second variety—called *representation change*—is based on the transformation of a problem's input to a different representation that is more conducive to an efficient algorithmic solution. The third variety of the transformation strategy is *problem reduction*, in which an instance of a given problem is transformed into an instance of a different problem altogether.

As our first example, let us consider a puzzle-like problem from Jon Bentley's book *Programming Pearls* [Ben00, pp. 15–16].

Anagram Detection Anagrams are words that are composed of the same letters; for example, the words "eat," "ate," and "tea" are anagrams. Devise an algorithm to find all sets of anagrams in a large file of English words.

An efficient algorithm for this problem works in two stages. First, it assigns each word a "signature" obtained by sorting its letters (representation change) and then sorts the file in alphabetical order of the signatures (sorting data is a special case of instance simplification) to put anagrams next to each other.

As an exercise, the reader is invited to solve the *Number Placement* puzzle (#43), which exploits the same idea.

Another occasionally useful type of representation change is to employ binary or ternary representation of the problem's input. Just in case the reader is unfamiliar with this important topic, here is a one-paragraph introduction. In the decimal positional system, which most of the world has been using for the last eight hundred years, an integer is represented as a combination of powers of 10, for example, $1069 = 1 \cdot 10^3 + 0 \cdot 10^2 + 6 \cdot 10^1 + 9 \cdot 10^0$. In the binary and ternary systems, an integer is represented as a combination of powers of 2 and 3, respectively. For example, $1069_{10} = 10000101101_2$ because $1069 = 1 \cdot 2^{10} + 0 \cdot 2^9 + 0 \cdot 2^8 + 0 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$, and $1069_{10} = 1110121_3$ because $1069 = 1 \cdot 3^6 + 1 \cdot 3^5 + 1 \cdot 3^4 + 0 \cdot 3^3 + 1 \cdot 3^2 + 2 \cdot 3^1 + 1 \cdot 3^0$. While a decimal number is composed of some of 10 digits (0 through 9), there are only two possible digits in a binary number (0 and 1), and there are three possible digits (0, 1, and 2) in a ternary number. Every decimal integer has a unique representation in either of these systems, which can be found by repeatedly dividing the integer by 2 and 3, respectively. The binary system is particularly important because it has proved to be most convenient for computer implementation.

As an example of a puzzle that takes advantage of the binary system, consider an instance of the problem mentioned in W. Poundstone's book [Pou03, p. 84].

Cash Envelopes You have one thousand $1 bills. How can you distribute them among 10 envelops so that any amount between $1 and $1000, inclusive, can be given as some combination of these envelopes? No change is allowed, of course.

Let us put 1, 2, $2^2, \ldots, 2^8$ dollar bills in the first nine envelopes and $1000 - (1 + 2 + \cdots + 2^8) = 489$ dollar bills in the tenth envelope. Any amount A smaller than 489 can be obtained as a combination of the powers of 2: $b_8 \cdot 2^8 + b_7 \cdot 2^7 + \cdots + b_0 \cdot 1$, where the coefficients $b_8, b_7, \ldots, b_0$ are either 0 or 1. (These coefficients compose A's representation in the binary system. The largest integer a nine digit binary number can represent is $2^8 + 2^7 + \cdots + 1 = 2^9 - 1 = 511$.) Any amount A between 489 and 1000, inclusive, can be represented as $489 + A'$ where $0 \le A' \le 511$; hence, it can be obtained as the contents of the tenth envelope and a combination of the first nine, the latter given by the binary representation of A'. Note that the solution to the puzzle is not unique for some amounts A.

A good exercise for the reader would be to solve the two versions of the *Bachet's Weights* puzzle (#115), which take advantage of the binary and a variation of the ternary system, respectively.

Finally, many problems can be solved by transforming them into questions about graphs. A *graph* can be thought of as a finite collection of points in the plane with lines connecting some of them. The points and lines are called, respectively, *vertices* and *edges* of the graph. Edges may have no directions on them or may be directed from one vertex to another. In the former case, the graph is said to be *undirected*; in the later case, it is called a *directed graph*, or a *digraph,* for short. In applications to puzzles and games, vertices of a graph typically represent possible states of the problem in question, and edges indicate permitted transitions between the states. One of the graph's vertices represents an initial state, while another represents a goal state of the problem. (There might be several vertices of the latter kind.) Such a graph is called a *state-space graph.* Thus, the transformation just described reduces the problem to the question about a path from the initial-state vertex to a goal-state vertex.

As a specific example, let us consider a smaller instance of a very old and well-known puzzle.[3]

Two Jealous Husbands There are two married couples who need to cross a river. They have a boat that can hold no more than two people at a time. To complicate matters, both husbands are jealous and require that no wife can be in the presence

[3] The classic version of this puzzle for three couples was included in the earliest known collection of mathematical problems in Latin, titled *Propositiones ad Acuendos Juvenes* (*Problems to Sharpen the Young*), which is attributed to Alcuin of York (c.735–804), a renowned medieval scholar. It had even more offensive wording to today's norms [Had92].

of the other man without her husband being present. Can they cross the river under such constraints?

A state-space graph for this puzzle is shown in Figure 1.5, where H_i, W_i denote the husband and wife of couple i $(i = 1, 2)$, respectively; the two bars | | denote the river; the boat's location, which defines the direction of the next trip, is shown by the gray oval. (For the sake of simplicity, the graph does not include crossings that differ by obvious index substitutions such as starting with the first couple H_1W_1 crossing the river instead of the second couple H_2 W_2.) The vertices corresponding to the initial and final states are shown in bold.

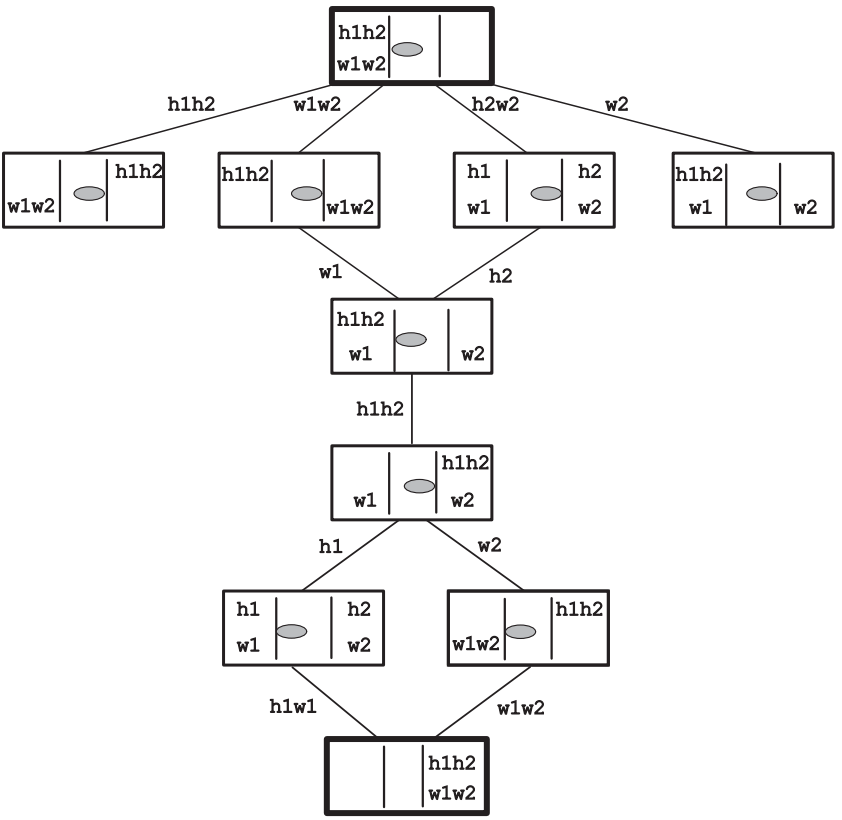


FIGURE 1.5 State-space tree for the *Two Jealous Husbands* puzzle.

There are four shortest paths from the initial-state vertex to the final-state vertex, each five edges long, in this graph. Specified by their edges, they are as follows:

$$
\begin{array}{lllll}
W_1W_2 & W_1 & H_1H_2 & H_1 & H_1W_1 \\
W_1W_2 & W_1 & H_1H_2 & W_2 & W_1W_2 \\
H_2W_2 & H_2 & H_1H_2 & H_1 & H_1W_1 \\
H_2W_2 & H_2 & H_1H_2 & W_2 & W_1W_2
\end{array}
$$

Hence, there are four (to within obvious symmetric substitutions) optimal solutions to this problem, each requiring five river crossings.

The *Missionaries and Cannibals* puzzle (#49) can be used as another exercise of this kind.

Two notes need to be made about solving puzzles via a graph representation. First, the creation of a state-space graph for more sophisticated puzzles can pose an algorithmic problem in its own right. In fact, the task might be infeasible because of a very large number of states and transformations. For example, the graph representing the states of the Rubik's Cube puzzle would have more than 10^{19} vertices. Second, although a specific location of points representing vertices of a graph has no theoretical significance, a good selection of the way the vertices are placed in the plane can provide an important insight into the puzzle in question. For example, consider the following puzzle, which is often attributed to Paolo Guarini (1512) but in fact was found in Arab chess manuscripts dating from around 840.

Guarini's Puzzle There are four knights on the 3×3 chessboard: the two white knights are at the two bottom corners, and the two black knights are at the two upper corners of the board (Figure 1.6). The goal is to switch the knights in the minimum number of moves so that the white knights are at the upper corners and the black knights are at the bottom corners.

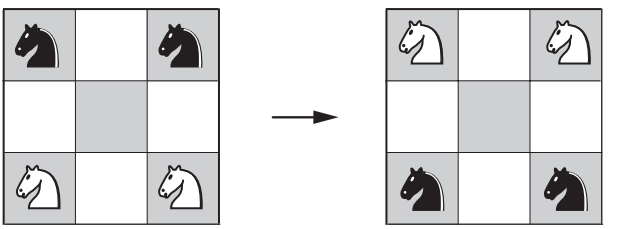


FIGURE 1.6 *Guarini's Puzzle.*

It is natural to represent the squares of the board (numbered for simplicity by consecutive integers in Figure 1.7a) by vertices of a graph in which an edge connects two vertices if a knight can make a move between the squares represented by the vertices. Placing the vertices in a way mimicking the positions of the squares on the board, we obtain the graph shown in Figure 1.7b. (Since the square number 5 at the center of the board cannot be reached by any of the knights, it is omitted there.) The graph in Figure 1.7b does not seem to help much in solving the problem. If, on the other hand, we place the vertices along a circumference in the order they can be reached from vertex 1 by knight moves—as shown in Figure 1.7c—we will obtain a much more revealing picture.[4] It is clear from Figure 1.7c that every

[4] Dudeney [Dud58, p. 230] called such a transformation the *buttons and strings method*: if vertices and edges of the graph are thought as buttons and strings, respectively, one can get the same graph as shown in Figure 1.7c by lifting and carrying "buttons" 2, 8, 4, and 6 to the opposite sides of the graph to "untangle" it.

legitimate move of a knight preserves the knights' relative ordering in the clockwise and counterclockwise directions. Therefore, there are only two ways to solve the puzzle in the minimum number of moves: move the knights along the edges in either a clockwise or counterclockwise direction until each of the knights reaches the diagonally opposite corner for the first time. Either of these symmetric options requires a total of 16 moves.

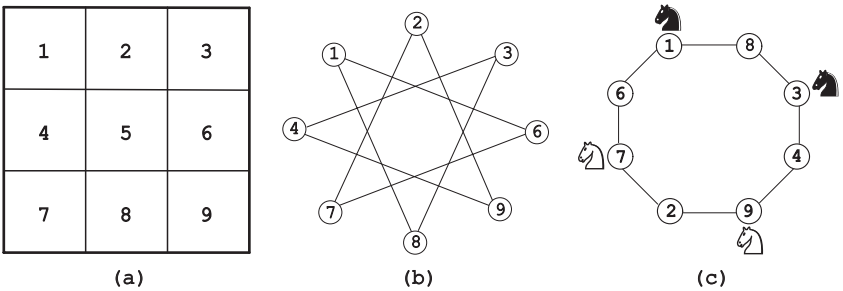


FIGURE 1.7 (a) Numbering of the board's squares for *Guarini's Puzzle*. (b) Straightforward representation of the puzzle's graph. (c) Better representation of the puzzle's graph.

We recommend solving the *Coins on a Star* puzzle (#34) as an exercise in the graph unfolding method.

There are also puzzles that can be solved by reducing them to a mathematical problem such as solving an equation or finding the maximum or minimum of a function. Here is an example of such a puzzle.

Optimal Pie Cutting What is the maximum number of pieces one can get by making n straight cuts in a rectangular pie, if each cut has to be parallel to one of the pie's sides, vertical or horizontal?

If the pie is cut by h horizontal and v vertical cuts, the total number of pieces obtained will be $(h + 1)(v + 1)$. Since the total number of cuts $h + v$ is equal to n, the problem is reduced to maximizing

$$(h + 1)(v + 1) = hv + (h + v) + 1 = hv + n + 1 = h(n - h) + n + 1$$

among all integer values of h between 0 and n, inclusive. Since $h(n - h)$ is a quadratic function of h, the maximum is obtained for $h = n/2$ if n is even and for $h = n/2$ rounded down (denoted $\lfloor n/2 \rfloor$) or up (denoted $\lceil n/2 \rceil$) if n is odd. Hence, the puzzle has a unique solution $h = v = n/2$ if n is even and two solutions (which can be considered symmetric) $h = \lfloor n/2 \rfloor$, $v = \lceil n/2 \rceil$ and $h = \lceil n/2 \rceil$, $v = \lfloor n/2 \rfloor$ if n is odd.

Greedy Approach

The *greedy approach* solves an optimization problem by a sequence of steps, each expanding a partially constructed solution until a complete solution is reached.

At each step—and this is the central point of this strategy—the choice is to produce the largest immediate gain without violating the problem's constraints. Such a "greedy" grab of the best alternative available at each step is made in the hope that a sequence of locally optimal choices will yield a (globally) optimal solution to the entire problem. This simple-minded approach works in some cases and fails in others.

One should not expect a rich bounty from a hunt for puzzles solvable by the greedy approach: good puzzles are usually too "tricky" to be solvable in such a straightforward fashion. Still, there are puzzles that can be solved by a greedy algorithm. Usually in these cases it is not difficult to design a greedy algorithm itself; rather, a more difficult task is to prove that it indeed yields an optimal solution. The following puzzle provides an example.

Non-Attacking Kings Place the greatest possible number of kings on an 8×8 chessboard so that no two kings are placed on adjacent—vertically, horizontally, or diagonally—squares.

Following the prescription of the greedy strategy, we can start by placing the maximum number of nonadjacent kings (four) in the first column of the board. Then, after skipping the next column because each of its squares is adjacent to one of the placed kings in the first column, we can place four kings in the third column, skip the fourth, and so on, until we end up with the total of 16 kings on the board (Figure 1.8a).

In order to show that it is impossible to place more than 16 nonadjacent kings on the board, we partition the board into 16 four-by-four squares, as shown in Figure 1.8b. Obviously, it is impossible to place more than one king in each of these squares, which implies that the total number of nonadjacent kings on the board cannot exceed 16.

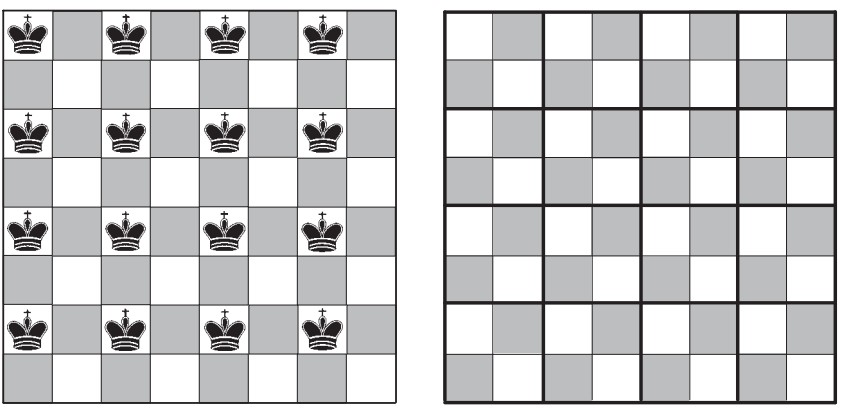


FIGURE 1.8 (a) Placement of 16 non-attacking kings. (b) Partition of the board proving impossibility of placing more than 16 non-attacking kings.

As our second example, we consider a puzzle that has become especially popular since it was reported to have been asked during job interviews at Microsoft.

Bridge Crossing at Night A group of four people, who have one flashlight, need to cross a rickety bridge at night. A maximum of two people can cross the bridge at one time, and any party that crosses (either one or two people) must have the flashlight with them. The flashlight must be walked back and forth; it cannot be thrown. Person A takes 1 minute to cross the bridge, person B takes 2 minutes, person C takes 5 minutes, and person D takes 10 minutes. A pair must walk together at the rate of the slower person's pace. Find the fastest way they can accomplish this task.

The greedy algorithm, illustrated in Figure 1.9, would start by sending to the other side the two fastest people, that is, persons A and B (it will take 2 minutes) and then return the flashlight with the fastest of the two, that is, with person A (1 more minute). Then it will send to the other side the two fastest persons available, that is, persons A and C (5 minutes) and return the flashlight with the fastest person A (1 minute). Finally, the two persons remaining will cross the river together (10 minutes). The total amount of time this greedy-based schedule requires is $(2 + 1) + (5 + 1) + 10 = 19$ minutes, but this is *not* the fastest possible solution (see this puzzle again later in the book (#7)).

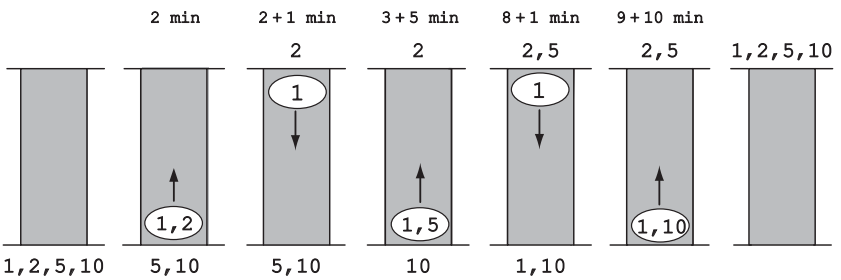


FIGURE 1.9 Greedy solution to the *Bridge Crossing at Night* puzzle.

It would be instructive for the reader to revisit the *Coins On a Star* puzzle (#34) and solve it by the greedy approach without benefits of the graph's unfolding.

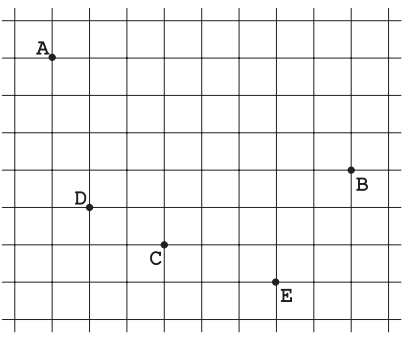
Iterative Improvement

While a greedy algorithm constructs a solution piece by piece, an iterative improvement algorithm starts with some easily obtainable approximation to a solution and improves upon it by repeated applications of some simple step. To validate such an algorithm, one needs to make sure that the algorithm in question does stop after a finite number of steps and that the final approximation obtained indeed solves the problem. Consider the following puzzle, which is a politically

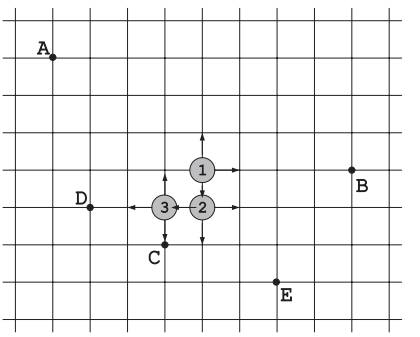
correct version of a problem discussed by Martin Gardner in his remarkable book *aha!Insight* [Gar78, pp. 131–132].

Lemonade Stand Placement Five friends—Alex, Brenda, Cathy, Dan, and Earl—want to set up a lemonade stand. They live at the locations denoted by letters A, B, C, D, and E on the map in Figure 1.10a. At which street intersection should they place their stand to minimize the distance to their homes? Assume that they measure the distance by the total number of blocks—horizontally and vertically—from their homes to the stand.

Initially, the friends decided to locate their stand at intersection 1 (Figure 1.10b), which is the middle point horizontally between the leftmost and rightmost points A and B and the middle point vertically between the highest and lowest points A and E. But then somebody noticed that it is not the best location possible. So they decided on the following iterative improvement algorithm: Starting with their initial candidate, consider in turn the locations one block from it in some order, say,



(a)



(b)

	①	↑①	①→	① = ② ↓
A	4 + 3	4 + 2	5 + 3	4 + 4
B	4 + 0	4 + 1	3 + 0	4 + 1
C	1 + 2	1 + 3	2 + 2	1 + 1
D	3 + 1	3 + 2	4 + 1	3 + 0
E	2 + 3	2 + 4	1 + 3	2 + 2
Total	23	26	24	22

	↑② = ①	②→	② ↓	←② = ③
A		5 + 4	4 + 5	3 + 4
B		3 + 1	4 + 2	5 + 1
C		2 + 1	1 + 0	0 + 1
D		4 + 0	3 + 1	2 + 0
E		1 + 2	2 + 1	3 + 2
Total		23	23	21

	↑③	③→ = ②	③ ↓	←③
A	3 + 3		3 + 5	2 + 4
B	5 + 0		5 + 2	6 + 1
C	0 + 2		0 + 0	1 + 1
D	2 + 1		2 + 1	1 + 0
E	3 + 3		3 + 1	4 + 2
Total	22		22	22

(c)

FIGURE 1.10 (a) Instance of the *Lemonade Stand Placement* puzzle. (b) The algorithm's steps. (c) Distances computed by the algorithm.

up (north), right (east), down (south), and left (west). As soon as a new location is closer to their homes, replace the old location with the new candidate and repeat the same verification operation; if none of the four neighboring intersections turns out to be better, consider the current location optimal and stop the algorithm. The algorithm's operation is shown in Figure 1.10b, with the computed distances given in Figure 1.10c.

While the final location marked by number 3 in Figure 1.10b certainly looks like a good choice, the algorithm does not provide a proof of the location's *global* optimality. In other words, how do we know that not only the four intersections one block away from it are inferior choices but also that it will be true for any other intersection? Well, we need not worry about our young entrepreneurs: this location is indeed the best, and the reader will have a chance to see this by solving the *Site Selection* puzzle (#74)—the general instance of this puzzle.

Here is another example of a puzzle that can be solved by iterative improvement.

Positive Changes Given an $m \times n$ table of real numbers, is there an algorithm to make all the row sums and column sums nonnegative by changing the signs of all the numbers in any row or column as the only operation allowed for the algorithm?

It would be natural to try finding an algorithm that increases the number of lines (rows and columns) with nonnegative sums on each of its iterations. However, changing the signs in a row (column) with a negative sum may make the sums in some column (row) negative! A neat way to overcome this difficulty is to pay attention to the total sum of the numbers in the table. Since it can be computed as the total of either all the row sums or all the column sums, changing the signs in a line with a negative sum definitely increases the total sum of the numbers in the table. Therefore, we can simply repeatedly search for a line with a negative sum. If we find such a line, we change the signs of all its numbers; if we do not find such a line, we have achieved our goal and can stop.

Is that all? Not quite. We also need to show that the algorithm's operation cannot continue indefinitely without stopping. This is indeed the case, because repeated applications of the algorithm's operation can create only a finite number of different tables (each of the mn elements can be in no more than two states). Therefore, the number of all element sums is also finite. Since the algorithm generates a sequence of tables with increasing sums, it must stop after a finite number of steps.

In both examples considered above, we took advantage of some quantity with the following characteristics:

- It could change its value only in a desired direction (decreasing in the first problem and increasing in the second).
- It could attain only a finite number of values, which guaranteed a stop after a finite number of steps.
- When it reached its final value, the problem was solved.

Such a quantity is called a *monovariant*. Finding an appropriate monovariant can be a tricky task. This has made puzzles involving monovariants a popular

topic in mathematical competitions. For example, the second example given above was used among practice problems for the first All-Russian Mathematical Olympiad in 1961 [Win04, p. 77]. It would be wrong, however, to dismiss iterative improvement and monovariants as just mathematical toys. Some of the most important algorithms in computer science, such as the *simplex method*, are based on this approach. The interested reader can find a few other puzzles involving monovariants in the harder puzzle section of this book.

Dynamic Programming

Dynamic programming is interpreted by computer scientists as a technique for solving problems with overlapping subproblems. Rather than solving overlapping subproblems again and again, it suggests solving each of the smaller subproblems only once and recording the results in a table from which a solution to the original problem can then be obtained. Dynamic programming was invented by a prominent U.S. mathematician, Richard Bellman, in the 1950s as a general method for optimizing multistage decision processes. For an optimization problem to be solved by this technique, the problem must have a so-called optimal substructure so that its optimal solution can be constructed efficiently from optimal solutions to its subproblems.

As an example, consider a problem of counting shortest paths.

Shortest Path Counting Find the number of the shortest paths from intersection A to intersection B in a city with perfectly horizontal streets and vertical avenues shown in Figure 1.11a.

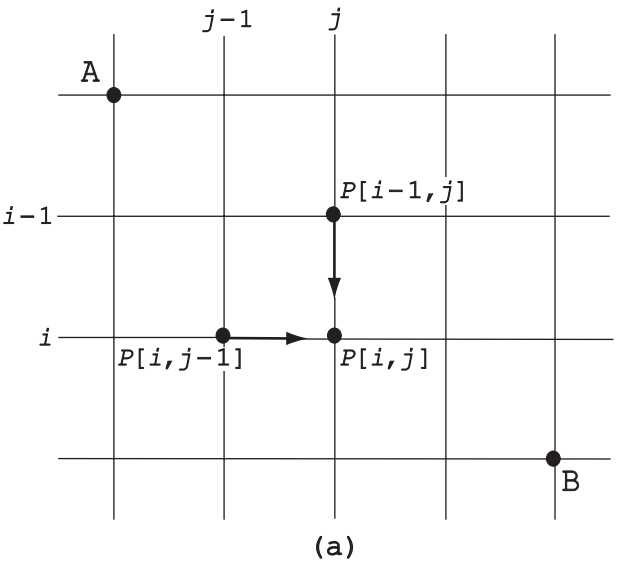
Let $P[i, j]$ be the number of shortest paths from intersection A to the intersection of street i $(1 \le i \le 4)$ and avenue j $(1 \le j \le 5)$. Any shortest path here is composed of horizontal segments going right along the streets and vertical segments going down the avenues. Therefore, the number of shortest paths from A to the intersection of street i and avenue j can be found as the sum of the number of shortest paths from A to the intersection of street $i - 1$ and avenue j ($P[i - 1, j]$ in our notation) and the number of shortest paths from A to the intersection of street i and avenue $j - 1$ ($P[i, j - 1]$ in our notation):

$$P[i, j] = P[i - 1, j] + P[i, j - 1] \text{ for every } 1 < i \le 4,\ 1 < j \le 5$$

where

$$P[1, j] = 1 \text{ for every } 1 \le j \le 5,\ \ P[i, 1] = 1 \text{ for every } 1 \le i \le 4.$$

Using these formulas, we can compute the values of $P[i, j]$ either row by row starting with row 1 and moving left to right along each row, or column by column moving down along each column.



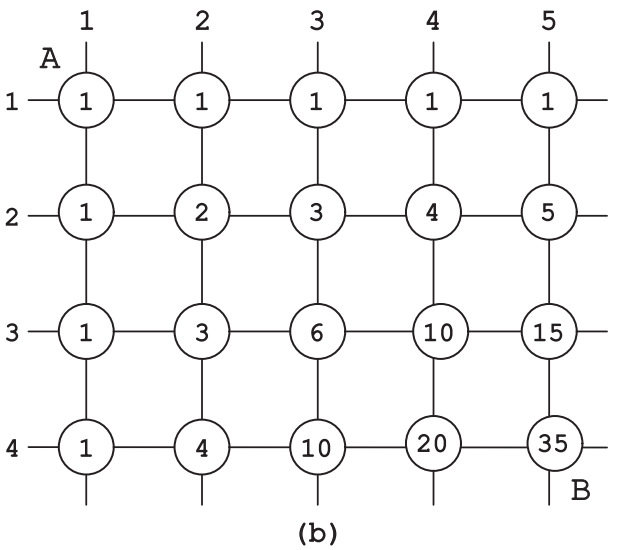


FIGURE 1.11 (a) Dynamic programming way to computing the number of shortest paths to the (i, j) intersection. (b) Numbers of the shortest paths from intersection A to each street-avenue intersection.

The problem can also be solved by a simple combinatorial argument. Every shortest path is made up of four horizontal segments and three vertical segments; the paths differ from each other in the choice of the three vertical segments out of the seven possibilities. Hence, the total number of the shortest paths can be obtained as the number of ways to choose three out of seven different things, that is, as $C(7, 3) = 7!/(3!4!) = 35$.[5] But while the combinatorial solution is faster than dynamic programming for this simple example, it is often not the case for counting paths in less regular grids. To see this, the reader may want to solve the *Blocked Paths* puzzle (#13) in the main section.

While some applications of dynamic programming are anything but straightforward, *Maximum Sum Descent* (#20) and *Picking Up Coins* (#62) can also be recommended as simple applications of this strategy.

ANALYSIS TECHNIQUES

Although most puzzles in this book deal with algorithm design, some of them do require an algorithm analysis. The goal of this short tutorial is to review the standard techniques for algorithm analysis, illustrating them on a few puzzles. We will try to do this on the most accessible level; a more in-depth treatment can be found in textbooks such as [Lev06], [Kle05], and [Cor09], which are listed in increasing order of difficulty.

An algorithm analysis usually seeks to ascertain the algorithm's time efficiency. This is done by counting the number of times the algorithm's basic step is executed. For almost all algorithmic problems, such a count grows with the problem size. Figuring out *how fast* it grows is the main goal of the algorithm analysis. Counting, not surprisingly, involves mathematics. So let us start by reviewing some important mathematical formulas that go surprisingly far in analyzing algorithms.

A few summation formulas and efficiency of algorithms

There is a well-known apocryphal story about Carl Friedrich Gauss (1777–1855), one of the greatest mathematicians in history. When Gauss was about 10 years old, his teacher asked the class to compute the sum of the first 100 positive integers

$$1 + 2 + \cdots + 99 + 100,$$

probably in the hope to occupy his students for quite a while. The teacher could not know, of course, that a mathematical genius happened to be among them. Supposedly, it took Carl just a few minutes to come with the answer by grouping

[5] The reader familiar with elementary combinatorics will note that the values of $P[i, j]$ can also be computed along southwest to northeast diagonals through the intersections, starting at point A. These values are elements of the famous combinatorial structure called *Pascal's Triangle* (e.g., [Ros07, Section 5.4]).

the numbers into 50 pairs with the same sum of 101:

$$(1 + 100) + (2 + 99) + \cdots + (50 + 51) = 101 \cdot 50 = 5050.$$

Generalizing this to the sum of the first n positive integers yields the formula

$$1 + 2 + \cdots + (n - 1) + n = \frac{(n + 1)n}{2}. \tag{1}$$

As an exercise, we recommend to the reader the *Mental Arithmetic* puzzle (#9), which exploits this formula and its derivation.

Formula (1) is almost indispensable in algorithm analysis. It also leads to several other useful formulas. For example, for the sum of the first n positive even numbers, we obtain

$$2 + 4 + \cdots + 2n = 2(1 + 2 + \cdots + n) = n(n + 1),$$

and for the sum of the first n positive odd numbers, we get

$$\begin{aligned} &1 + 3 + \cdots + (2n - 1) \\ &= (1 + 2 + 3 + 4 + \cdots + (2n - 1) + 2n) - (2 + 4 + \cdots + 2n) \\ &= \frac{2n(2n + 1)}{2} - n(n + 1) = n^2. \end{aligned}$$

Another very important formula is the sum of consecutive powers of 2, which we already used in the first tutorial:

$$1 + 2 + 2^2 + \cdots + 2^n = 2^{n+1} - 1. \tag{2}$$

Now we are ready to consider our first example dealing with an algorithm analysis.

Chess Invention Presumably, the game of chess was invented many centuries ago in northwestern India by a sage named Shashi. When Shashi took his invention to his king, the king liked the game so much that he offered the inventor any reward he wanted. Shashi asked for some grain to be obtained as follows: just a single grain of wheat was to be placed on the first square of the chess board, two on the second, four on the third, eight on the fourth, and so on, until all 64 squares had been filled. Was it a reasonable request on the inventor's part?

Well, according to formula (2), the total number of grains Shashi requested is equal to

$$1 + 2 + \cdots + 2^{63} = 2^{64} - 1.$$

If it took him just 1 second to count each grain, the total amount of time he would have needed to count all the grains comes to about 585 billion years, over 100 times

more than the estimated age of the planet Earth. This is a good demonstration of the monstrous rate of *exponential growth*. Obviously, algorithms that require time that grow exponentially with the problem size are impractical for all but very small instances of the problem.

What would have happened if instead of doubling the number of grains for each square of the chessboard, Shashi asked for adding two grains? Then the total amount of grains would have been equal to

$$1 + 3 + \cdots + (2 \cdot 64 - 1) = 64^2.$$

With the same speed of counting one grain per second, he would have needed less than 1 hour and 14 minutes to count his modest reward. The *quadratic* rate of growth is clearly much more acceptable for the running time of an algorithm.

Even faster are algorithms that are *linear*. These algorithms require time proportional to their input's size. Still more efficient are *logarithmic* algorithms. These algorithms are usually based on the decrease-by-a-constant-factor strategy (see the tutorial on the algorithm design strategies) and work by repeatedly reducing the problem size by, say, half. This turns the exponential rate of growth to our advantage by making the size of the problem that remains to be solved shrink very fast. The algorithm for the *Number Guessing* game discussed in the first tutorial falls into this category.

Analysis of a nonrecursive algorithm

The reader will not be surprised by the fact that a *nonrecursive algorithm* is an algorithm that is not recursive. This means that it does not work by applying itself to smaller and smaller instances of the same problem until a trivial instance with an obvious solution is reached. Typically, a nonrecursive algorithm can be analyzed by setting up a sum for the number of times its principal step is executed. Then the sum is simplified with the goal of finding either an exact compact formula for the count or an approximate formula indicating its rate of growth. As an example, we consider the following puzzle-like question [Gar99, p. 88].

Square Build-Up An algorithm starts with a single square and on each of its next iterations adds new squares all around the outside. How many unit squares will be there after the nth iteration? The results for the first few iterations are shown in Figure 1.12.

The basic step of this algorithm is adding a unit square. Therefore, counting the number of basic steps for this algorithm is simply equivalent to counting the total number of unit squares. After n iterations, the longest horizontal row will contain $(2n - 1)$ such squares. The rows above and below it will contain odd numbers of squares from 1 to $2n - 3$. Since the sum of the first $n - 1$ odd numbers is equal to

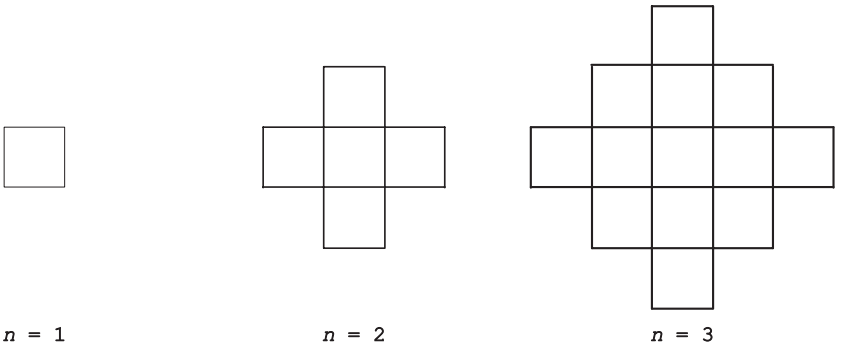


FIGURE 1.12 First iterations of the *Square Build-Up* algorithm.

$(n-1)^2$, the total number of unit squares will be equal to

$$\begin{aligned} &2(1+3+\cdots+(2n-3))+(2n-1) \\ &\quad = 2(n-1)^2+(2n-1) = 2n^2-2n+1. \end{aligned}$$

One can also solve the puzzle by noting that the number of unit squares added on the ith iteration $(1 < i \leq n)$ is equal to $4(i-1)$. Therefore, the total number of unit squares after n iterations can be computed as

$$\begin{aligned} &1+4\cdot 1+4\cdot 2+\cdots+4(n-1) = 1+4(1+2+\cdots+(n-1)) \\ &\quad = 1+4(n-1)n/2 = 2n^2-2n+1. \end{aligned}$$

While there is certainly nothing wrong with using standard techniques, it is always a good idea to try exploiting the specifics of the problem given. Here, one can count the squares by the diagonals of the shape obtained after the nth iteration. It comprises n diagonals with n unit squares that alternate with $n-1$ diagonals each with $n-1$ unit squares for the total of $n^2+(n-1)^2 = 2n^2-2n+1$ squares.

For another example, we suggest to the reader solving the *Counting Triangles* puzzle (#52).

Analysis of a recursive algorithm

We will illustrate the standard technique for analysis of a recursive algorithm by considering the classic *Tower of Hanoi* puzzle.

Tower of Hanoi In the general instance of this puzzle, there are n disks of different sizes and three pegs. Initially, all the disks are on the first peg in order of size, the largest on the bottom and the smallest on top. The objective is to transfer all the disks to another peg by a sequence of moves. Only one disk can be moved at a time, and it is forbidden to place a larger disk on top of a smaller one.

The problem has an elegant recursive solution that is illustrated in Figure 1.13. To transfer $n > 1$ disks from peg 1 to peg 3 (with peg 2 as auxiliary), transfer $n - 1$ disks from peg 1 to peg 2 (with peg 3 as auxiliary), then move the largest disk directly from peg 1 to peg 3, and, finally, transfer $n - 1$ disks from peg 2 to peg 3 (using peg 1 as auxiliary). Of course, if $n = 1$, simply move the single disk directly from peg 1 to peg 3.

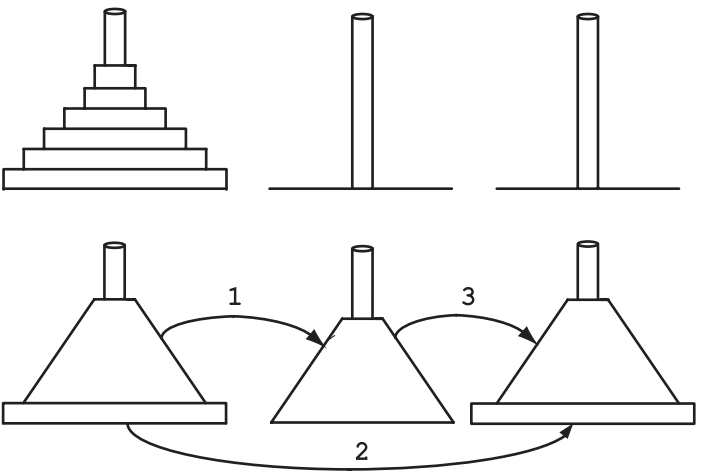


FIGURE 1.13 Recursive solution to the *Tower of Hanoi* puzzle.

Obviously, moving a disk from one peg to another is the basic operation of this algorithm. Let $M(n)$ be the number of disk moves made by the algorithm to solve the puzzle with n disks. Following the algorithm's description (see also Figure 1.13), we obtain the following equation for $M(n)$:

$$M(n) = M(n-1) + 1 + M(n-1) \quad \text{for } n > 1.$$

Of course, this can be simplified as

$$M(n) = 2M(n-1) + 1 \quad \text{for } n > 1.$$

Such equations are called *recurrence relations* because they specify how the nth term of a sequence relates to its preceding terms. In our case, it says that the nth term of the sequence, $M(n)$, is larger by 1 than twice its preceding term $M(n-1)$. Note that this does not specify the sequence uniquely, because it does not say anything about the first term of the sequence. Since the algorithm makes just one move to solve the problem for one disk, we have the following addition to the recurrence relation: $M(1) = 1$, which, not surprisingly, is called the *initial condition*. To summarize, we have the following recurrence relation with the initial

condition for the number of moves made by the recursive algorithm for the Tower of Hanoi puzzle with n disks:

$$M(n) = 2M(n-1) + 1 \quad \text{for } n > 1,$$
$$M(1) = 1.$$

Rather than using one of the standard methods for solving such equations, which can be found in the textbooks mentioned in the first paragraph of this tutorial, let us try the inductive approach: compute the first few values of $M(n)$ by the formulas above, try to identify a general pattern, and then prove that the pattern holds for all positive n's.

n	$M(n)$
1	1
2	3
3	7
4	15

The inspection of the first values of $M(n)$ suggests the formula $M(n) = 2^n - 1$. Obviously, $M(1) = 2^1 - 1 = 1$. The easiest way to prove that it is valid for all $n > 1$ is to substitute the formula into the equation to see whether we get the equality for all such n's. This is indeed the case because

$$M(n) = 2^n - 1 \text{ and } 2M(n-1) + 1 = 2(2^{n-1} - 1) + 1 = 2^n - 1.$$

Thus, we have an exponential algorithm, which will run for an unimaginably long time even for moderate values of n. This is not due to the fact that this particular algorithm is poor; in fact, it is not difficult to prove that this is the most efficient algorithm possible for this problem. It is the problem's intrinsic difficulty that makes it so computationally hard. This might be a good news: in the original version of the Tower of Hanoi puzzle, as published by its inventor, the French mathematician Édouard Lucas, in the 1880s, the world will end after 64 disks have been moved by monks from a mystical Tower of Brahma. Assuming that the monks do not eat, sleep, or die and move one disk per minute, the world will end after about $3 \cdot 10^{13}$ years, which more than one thousand times longer than the estimated age of the universe.

As as exercise, the reader may want to solve the recurrence for the minimum number of moves in the *Restricted Tower of Hanoi* puzzle (#83), one of many variations of the classic version:

$$M(n) = 3M(n-1) + 2 \quad \text{for } n > 1,$$
$$M(1) = 2.$$

Invariants

We conclude this tutorial with a brief discussion of the idea of an invariant. For our purposes, an *invariant* is a property that is preserved by any algorithm that solves the problem. For puzzle-like questions, an invariant is often used to show that the problem has no solution because the invariant property holds for the initial state of the puzzle but fails for the required final state. Let us consider a few examples.

Domino Tiling of Deficient Chessboards a. Is it possible to tile with dominoes an 8×8 chessboard missing one corner square (Figure 1.14a)? b. Is it possible to tile with dominoes an 8×8 chessboard missing two diagonally opposite corner squares (Figure 1.14b)?

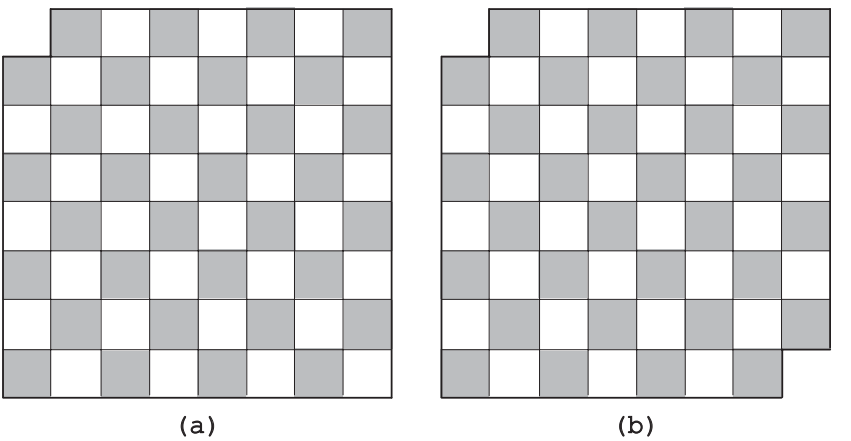


FIGURE 1.14 (a) Chessboard missing one corner square. (b) Chessboard missing two diagonally opposite squares.

The answer to the first question is "no": in any domino tiling, the number of covered squares is always even (this even parity of the number of covered squares is the invariant here), while the number of squares on the board in question is odd.

The answer to the second question is also "no" even though the number of the squares on such a board is obviously even. The invariant here is different: since one domino covers one dark and one light square, any domino tiling covers the same number of dark and light squares of the board. Hence, a desired tiling of the entire board is impossible because the numbers of dark and light squares on the board without two diagonally opposite corners differ by two.

In general, the even-odd parity and coloring are two of the most widely used ways to exploit the invariant idea. Puzzles *Last Ball* (#50) and *A Corner-to-Corner Journey* (#18) are recommended to the reader as typical representatives of such puzzles.

The importance of an invariant in a different setting can be seen in the famous puzzle about walks in the old Prussian city of Königsberg.

The Königsberg Bridges Problem Was it possible, in a single stroll, to cross all seven bridges of Königsberg exactly once and return to the starting point? A sketch of the river with its two islands and seven bridges is shown in Figure 1.15.

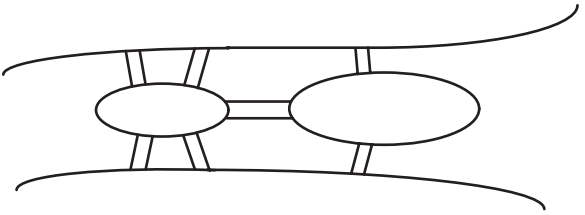


FIGURE 1.15 Diagram of Königsberg's seven bridges over the river connecting the mainland and two islands.

The puzzle was solved by the great Swiss-born mathematician Leonhard Euler (1707–1783). First, Euler realized that walking along a land mass—a bank of the river or an island—is irrelevant to the problem. The only pertinent information is connections provided by the bridges. In modern terms, this insight enabled him to transform the problem to the question about the graph shown in Figure 1.16. (Actually, it is a *multigraph* since there is more than one edge connecting some of its vertices.)

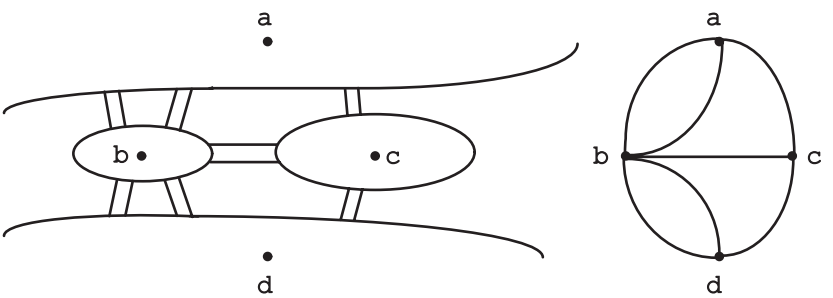


FIGURE 1.16 Multigraph of *The Königsberg Bridges Problem*.

The question then is whether the multigraph in Figure 1.16 has an *Euler circuit*: a sequence of adjacent vertices that traverses all the edges exactly once before returning to the starting vertex. Euler noticed that any such circuit would have to enter a vertex exactly the same number of times as it leaves the vertex. Hence, such a circuit can only exist in a multigraph in which the number of edges touching a vertex—called its *degree*—is even for every vertex. This invariant property implies that *The Königsberg Bridges Problem* has no solution because it fails this necessary condition: every vertex of the multigraph in Figure 1.16 has an odd degree. Moreover, the same analysis implies that there was no walk crossing all

the bridges exactly once even without the requirement of returning to the starting point. For such a walk, called an *Euler path*, to exist, all the vertices of the multigraph would have to have an even degree except exactly two vertices in which the walk would have to start and end.

By the way, these conditions turn out to be not only necessary but also sufficient for existence of an Euler circuit and an Euler path in a connected multigraph. (A multigraph is connected if there is a path between every pair of its vertices. Of course, if it is not the case, neither an Euler circuit not an Euler path can exist.) This fact was noted by Euler himself and later formally proved by another mathematician. The reader may take advantage of this analysis to solve the *Figure Tracing* puzzle (#28) in the main section.

Today, *The Königsberg Bridges Problem* is considered a springboard for graph theory, a branch of mathematics with important applications to computing and operations research. It is also often pointed to as an example of potential usefulness of puzzles for serious science, education, and practical applications.

The next puzzle provides an example where an invariant plays a role other than implying the nonexistence of a solution.

Breaking a Chocolate Bar Break an $n \times m$ chocolate bar into nm squares with the minimum number of bar breaks if a bar can be broken only in a straight line and only one bar can be broken at a time.

To appreciate this puzzle, well-known among mathematicians and computer scientists, the reader should stop and try to solve it before reading the solution given in the next two sentences. Since only one bar piece can be broken at a time, any break increases the number of pieces by 1. Hence, $nm - 1$ breaks are needed to get from a single $n \times m$ piece to nm 1×1 pieces, which is obtained by *any* sequence of $nm - 1$ allowed breaks.

Finally, we consider an example where an invariant plays a more constructive role by pointing out a way an algorithm must proceed. Here is a puzzle published—with trivial variations in the board size and wording—by the two most renowned creators of puzzles in history: Henry E. Dudeney [Dud02, p. 95], and Sam Loyd [Loy59, p. 8].[6]

Chickens in the Corn The game is played on a 5×8 board representing a cornfield, with two counters of one color representing a farmer and his wife and two counters of another color representing a rooster and a hen. On each move, a person and a chicken can move to a neighboring square, directly up and down or right and left but not diagonally. Starting with the positions indicated in Figure 1.17a, the man (M) and woman (W) each move one square, and then the rooster (r) and the hen (h) each make a move. The play continues by turns until both chickens are captured. A capture

[6] Dudeney and Loyd had collaborated for some years until Dudeney broke off the correspondence and accused Loyd of stealing his puzzles and publishing them under his own name.

occurs when the farmer or his wife can pounce on a square occupied by the chicken. The objective is to accomplish this task in the minimum number of moves.

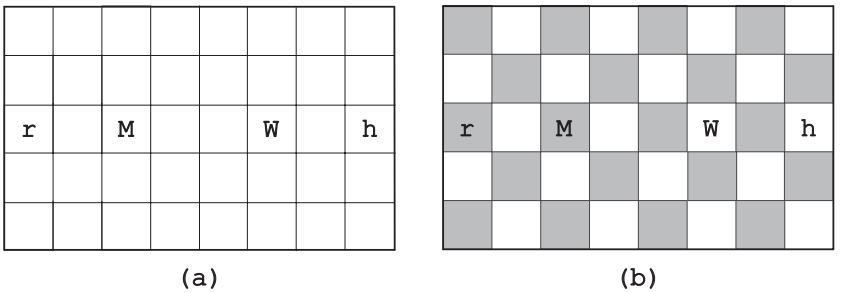


FIGURE 1.17 (a) Board of the *Chickens in the Corn* puzzle. (b) The board colored as a chessboard.

It does not take much effort to realize that the man cannot capture the rooster and that the woman cannot capture the hen. Indeed, a capture can only occur when a person and a bird occupy two adjacent squares, which are always of the opposite colors if the board's squares are colored as those of a chessboard (Figure 1.17b). But the man and the rooster start at the same-color squares, and this property remains valid after both make one and hence any finite number of moves. The same observation is true for the woman and the hen. Hence, the man should go for the hen and the woman should go for the rooster. Even if the chickens do not cooperate in their capture, one of them can be taken on the eighth move, the other on the ninth.

This concludes the tutorials. As to the question of which of the strategies needs to be applied to a particular puzzle, there is no answer! (If there was, puzzles would have lost their attraction as an intellectual entertainment.) The strategies are just general tools, which may and may not work successfully for a particular puzzle. With a lot of practice, one can hope to develop some intuition about when a particular tool might work successfully, but such intuition is certainly not going to be foolproof.

Still, the strategies and techniques discussed above provide a powerful set of tools for solving problems of an algorithmic nature. They are also more specific than those that mathematicians have at their disposal, despite such famous efforts as Pólya's "How to Solve It" [Pol57].

Of course, even if one knows which strategy needs to be applied, it might still be far from a simple task. For example, proving that a puzzle has no solution is typically done by using an invariant. But finding a specific invariant for a particular problem might be difficult, even if one knows that, say, using parity or board coloring might lead to a solution. Again, with practice the task gets easier but not necessarily easy.

2 Puzzles

EASIER PUZZLES

1. **A Wolf, a Goat, and a Cabbage**
   A man finds himself on a riverbank with a wolf, a goat, and a head of cabbage. He needs to transport all three to the other side of the river in his boat. However, the boat has room for only the man himself and one other item (either the wolf, the goat, or the cabbage). In his absence, the wolf would eat the goat, and the goat would eat the cabbage. Show how the man can get all these "passengers" to the other side.

2. **Glove Selection**
   There are 20 gloves in a drawer: 5 pairs of black gloves, 3 pairs of brown, and 2 pairs of gray. You select the gloves in the dark and can check them only after a selection has been made. What is the smallest number of gloves you need to select to guarantee getting the following?
   (a) At least one matching pair
   (b) At least one matching pair of each color

3. **Rectangle Dissection**
   Find all values of $n > 1$ for which one can dissect a rectangle into n right triangles, and outline an algorithm for doing such a dissection.

4. **Ferrying Soldiers**
   A detachment of 25 soldiers must cross a wide and deep river with no bridge in sight. They notice two 12-year-old boys playing in a rowboat by

the shore. The boat is so tiny, however, that it can only hold two boys or one soldier. How can the soldiers get across the river and leave the boys in joint possession of the boat? How many times does the boat pass from shore to shore in your algorithm?

5. **Row and Column Exchanges**
   Can one transform the left table in Figure 2.1 into the right table by exchanging its rows and columns?

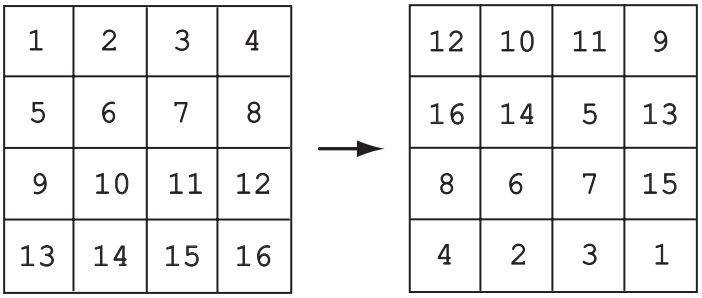


FIGURE 2.1 Initial and target tables in the *Row and Column Exchanges* puzzle.

6. **Predicting a Finger Count**
   A little girl counts from 1 to 1000 using the fingers of her left hand as follows. She starts by calling her thumb 1, the first finger 2, middle finger 3, ring finger 4, and little finger 5. Then she reverses direction, calling the ring finger 6, middle finger 7, the first finger 8, and her thumb 9, after which she calls her first finger 10, and so on. If she continues to count in this manner, on which finger will she stop?

7. **Bridge Crossing at Night**
   Four people need to cross a rickety footbridge; they all begin on the same side. It is dark, and they have one flashlight. A maximum of two people can cross the bridge at one time. Any party that crosses, either one or two people, must have the flashlight with them. The flashlight must be walked back and forth; it cannot be thrown, for example. Person 1 takes 1 minute to cross the bridge, person 2 takes 2 minutes, person 3 takes 5 minutes, and person 4 takes 10 minutes. A pair must walk together at the rate of the slower person's pace. For example, if person 1 and person 4 walk together, it will take them 10 minutes to get to the other side. If person 4 returns the flashlight, a total of 20 minutes have passed. Can they cross the bridge in 17 minutes?

8. **Jigsaw Puzzle Assembly**
   A jigsaw puzzle contains 500 pieces. A "section" of the puzzle is a set of one or more pieces that have been connected to each other. A "move" consists

of connecting two sections. What is the minimum number of moves in which the puzzle can be completed?

9. **Mental Arithmetic**
   A 10 × 10 table is filled with repeating numbers on its diagonals as shown in Figure 2.2. Calculate the total sum of the table's numbers in your head.

1	2	3			⋯			9	10
2	3						9	10	11
3						9	10	11	
					9	10	11		
				9	10	11			
⋮			9	10	11				⋮
		9	10	11					
	9	10	11						17
9	10	11						17	18
10	11				⋯		17	18	19

FIGURE 2.2 Table of numbers to be summed up in the *Mental Arithmetic* puzzle.

10. **A Fake Among Eight Coins**
    There are eight identical-looking coins; one of these coins is counterfeit and is known to be lighter than the genuine coins. What is the minimum number of weighings needed to identify the fake coin with a two-pan balance scale without weights?

11. **A Stack of Fake Coins**
    There are 10 stacks of 10 identical-looking coins. All of the coins in one of these stacks are counterfeit, and all the coins in the other stacks are genuine. Every genuine coin weighs 10 grams, and every fake weighs 11 grams. You have an analytical scale that can determine the exact weight of any number of coins. What is the minimum number of weighings needed to identify the stack with the fake coins?

12. **Questionable Tiling**
    Is it possible to tile an 8 × 8 board with dominoes (2 × 1 tiles) so that no two dominoes form a 2 × 2 square?

13. **Blocked Paths**
    Find the number of different shortest paths from point A to point B in a city with perfectly horizontal streets and vertical avenues as shown in Figure 2.3. No path can cross the fenced off area shown in grey in the figure.

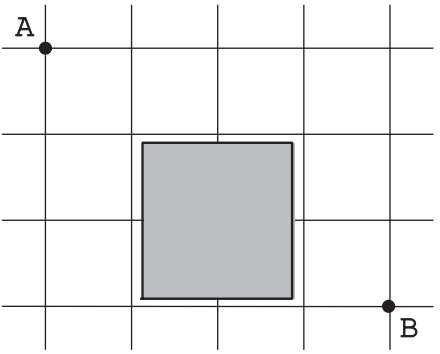


FIGURE 2.3 Grid of city streets with a fenced off area (shown in gray).

14. **Chessboard Reassembly**
The squares of an 8×8 chessboard are mistakenly colored in blocks of two colors as shown in Figure 2.4. You need to cut this board along some lines separating its rows and columns so that the standard 8×8 chessboard can be reassembled from the pieces obtained. What is the minimum number of pieces into which the board needs to be cut and how should they be reassembled?

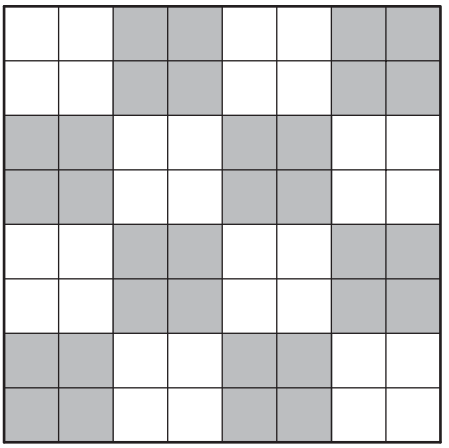


FIGURE 2.4 An 8×8 board to be reassembled into the standard chessboard.

15. **Tromino Tilings**
For each of the three cases, prove or disprove that for every $n > 0$ all the boards of the following dimensions can be tiled by right trominoes:

(a) $3^n \times 3^n$
(b) $5^n \times 5^n$
(c) $6^n \times 6^n$

Recall that right trominoes are L-shaped tiles formed by three adjacent squares (see the tutorial on algorithm design strategies). In a tiling, trominoes can be oriented in different ways, but they should cover all the squares of the board exactly with no overlaps.

16. **Making Pancakes**
    You have to make $n \geq 1$ pancakes using a skillet that can hold only two pancakes at a time. Each pancake has to be fried on both sides; frying one side of a pancake takes 1 minute, regardless of whether one or two pancakes are fried at the same time. Design an algorithm to do this job in the minimum amount of time. What is the minimum amount of time as a function of n?

17. A **King's Reach**

    (a) The king in chess can move to any neighboring square horizontally, vertically, or diagonally. Assuming that the king starts on some square of an infinite chessboard, in how many different squares can it be after n moves?
    (b) Answer the same question if the king makes no diagonal moves.

18. **A Corner-to-Corner Journey**
    Is there a way for a chess knight to start at the lower left corner of a standard 8×8 chessboard, visit all the squares of the board exactly once, and end at the upper right corner? (The knight's moves are L-shaped jumps: two squares horizontally or vertically followed by one square in the perpendicular direction.)

19. **Page Numbering**
    Pages of a book are numbered sequentially starting with 1. If the total number of decimal digits used is equal to 1578, how many pages are there in the book?

20. **Maximum Sum Descent**
    Some positive integers are arranged in a triangle like the one shown in Figure 2.5. Design an algorithm (more efficient than an exhaustive search, of course) to find the largest sum in a descent from its apex to the base through a sequence of adjacent numbers, one number per each level.

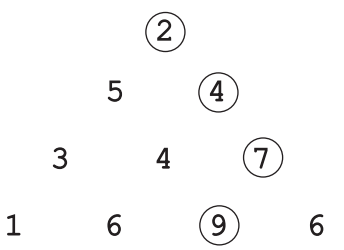


FIGURE 2.5 Triangle of numbers with the maximum-sum path shown by the circles.

21. **Square Dissection**
Find all values of n for which one can dissect a square into n smaller squares and outline an algorithm for doing such a dissection.

22. **Team Ordering**
Suppose you have the results of a completed round-robin tournament in which n teams played each other once. Assuming there were no ties, is it always possible to list the teams in a sequence so that every team won the game with the team listed immediately after it?

23. **Polish National Flag Problem**
There is a row of $n > 1$ checkers on the table, some of them are red and some are white. (Red and white are the colors of the Polish national flag.) Design an algorithm to rearrange the checkers so that all the red checkers precede all the white ones. The only operations allowed are the examination of a checker's color and the swapping of two checkers. Try to minimize the number of swaps made by your algorithm.

24. **Chessboard Colorings**
For each of the following chess pieces, find the minimum number of colors needed to color an $n \times n$ chessboard ($n > 1$) so that no two pieces in question placed on two squares of the same color can threaten each other:
    (a) The knight. (The knight threatens any square that is two squares horizontally and one square vertically, or two squares vertically and one square horizontally from the square it occupies.)
    (b) The bishop. (The bishop threatens any square that is on the same diagonal.)
    (c) The king. (The king threatens any square adjacent to it horizontally, vertically, or diagonally.)
    (d) The rook. (The rook threatens any square in the same row or in the same column.)

    The squares threatened by each of these pieces are shown in Figure 2.6.

25. **The Best Time to Be Alive**
An editor of *The History of the World Science* wants to find out the time when the largest number of prominent scientists were alive. The prominent scientists are, by definition, the people mentioned in the book with the dates of their birth and death. (No living scientists are included in the book.) Devise an algorithm for this task if it has the book's index as its input. The entries in the index are sorted alphabetically and give the persons' birth and death years. If person A died the same year person B was born, assume that the former event happened before the latter one.

26. **Find the Rank**
If we generate a list of all "words" made of letters G, I, N, R, T, and U in lexicographic order starting with GINRTU and ending with UTRNIG, what position in the list will be occupied by TURING? (Alan Turing

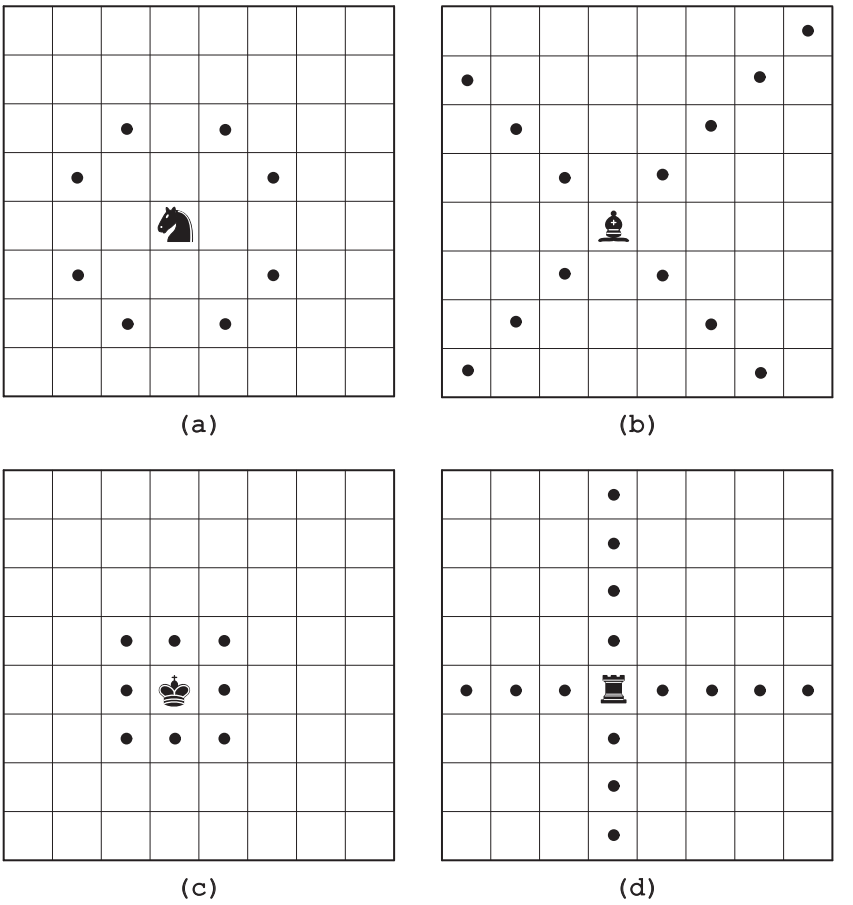


FIGURE 2.6 Squares threatened by (a) knight, (b) bishop, (c) king, (d) rook.

(1912–1954) was an English mathematician and computer scientist who, among other remarkable achievements, played a leading role in developing theoretical computer science.)

27. **The Icosian Game**
This is a 19th-century puzzle invented by the renowned Irish mathematician Sir William Hamilton (1805–1865) and presented to the world as the "Icosian Game." The game was played on a wooden board with holes representing major world cities and grooves representing connections between them (see Figure 2.7). The object of the game was to find a circular route that would pass through all the cities exactly once before returning to the starting point. Can you find such a route?

28. **Figure Tracing**
For each of the three figures in Figure 2.8, either trace the figure without lifting your pen off the paper or going back over any line in it, or prove that it is impossible to do so.

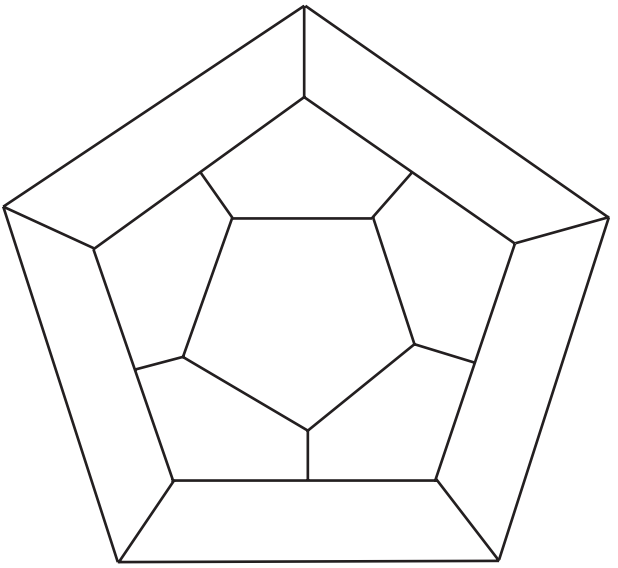


FIGURE 2.7 The graph of the *Icosian Game*.

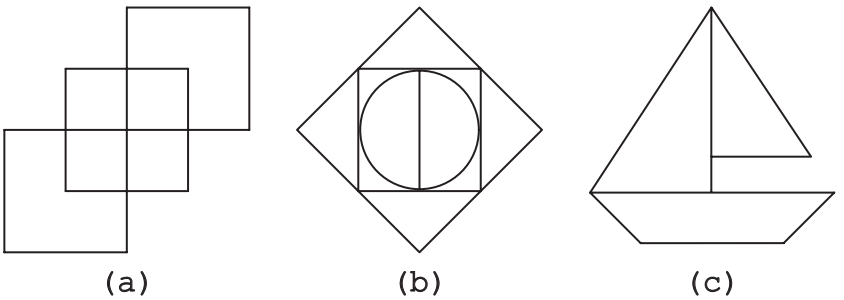


FIGURE 2.8 Three figures to trace.

29. **Magic Square Revisited**
A magic square of order 3 is a 3×3 table filled with nine distinct integers from 1 to 9 so that the sum of the numbers in each row, column, and two corner-to-corner diagonals is the same. Find *all* the magic squares of order 3.

30. **Cutting a Stick**
A stick 100 units long needs to be cut into 100 unit pieces. What is the minimum number of cuts required if you are allowed to cut several stick pieces at the same time? Also outline an algorithm that performs this task with the minimum number of cuts for a stick of n units long.

31. **The Three Pile Trick**
A magician asks a person to select one of 27 cards in a pack and then return it back without showing the card to the magician. After the choice

is made, the magician shuffles the pack and deals all the cards face up into three piles, one card to each pile in turn. The person who selected the card is asked which pile contains it. The magician then places the pile with that card between the other two piles and, without any shuffling, deals the cards into three piles as before. After the magician is informed about the pile containing the selected card again, he places the pile between the two and deals the cards into three piles for the last time. When he is told which of the piles contains the selected card in the final layout, he names that card. Explain the trick.

32. **Single-Elimination Tournament**
In a single-elimination tournament—such as the tennis Grand Slam championships—every losing player is immediately eliminated from the subsequent rounds of play until a single winner is determined. If such a tournament starts with n players, determine the following:

    (a) What is the total number of matches needed to get a winner?
    (b) How many rounds are there in such a tournament?
    (c) How many more matches need to be played to determine the second-best player based on the information produced by the tournament?

33. **Magic and Pseudo-Magic**

    (a) You have an $n \times n$ table whose cells are to be filled with integers 1 through 9, inclusive, one number per cell. The object is to do this in such a way that every 3×3 square in it is a magic square. Find all the values of $n \geq 3$ for which this can be done.
    (b) Find all the values of $n \geq 3$ for which it is possible to fill an $n \times n$ table with integers 1 through 9, inclusive, so that every 3×3 square in it is a *pseudo-magic square*. In a pseudo-magic square, all the row and column sums must be the same but the diagonal sums may be different.

34. **Coins on a Star**
The object of this puzzle is to place the largest possible number of coins at points of the eight-pointed star depicted in Figure 2.9. The coins should be placed one after another, with the following restrictions: (i) a coin needs to be placed first on an unoccupied point and then moved along a line to another unoccupied point, and (ii) once a coin has been positioned in this manner, it cannot be moved again. For example, we can start by placing the first coin on point 6 and then moving it to point 1 (denoted $6 \rightarrow 1$), where the coin will have to remain. We can continue, say, with the following sequence of moves: $7 \rightarrow 2, 8 \rightarrow 3, 7 \rightarrow 4, 8 \rightarrow 5$, which places five coins.

35. **Three Jugs**
Given an 8-pint jug full of water and two empty jugs of 5- and 3-pint capacity, get exactly 4 pints of water in one of the jugs by completely filling up and/or emptying jugs into others.

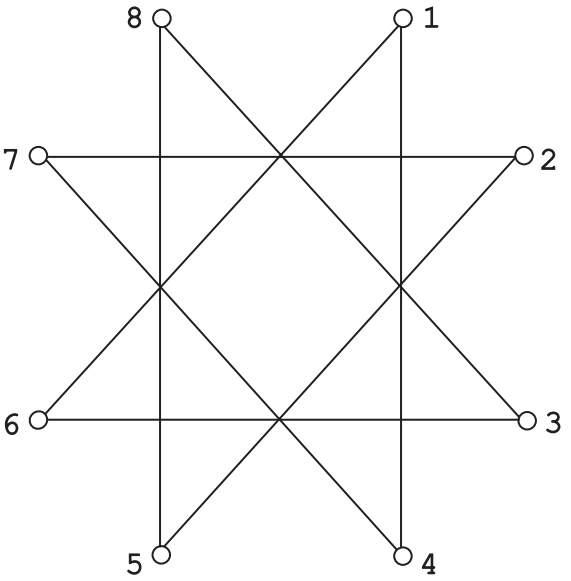


FIGURE 2.9 A star for placing coins at its points.

36. **Limited Diversity**
Find all values of n for which one can fill an $n \times n$ table with $+$'s and $-$'s (one per cell) so that every cell has exactly one neighbor with the opposite sign. Two cells are considered neighbors if they are either in the same row or the same column.

37. **$2n$-Counters Problem**
For any $n > 1$, place $2n$ counters on an $n \times n$ board so that no more than two counters are in the same row, column, or diagonal.

38. **Tetromino Tiling**
There are five types of tetrominoes—tiles made of four 1×1 squares, which are shown in Figure 2.10.

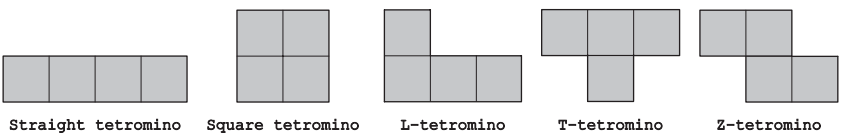


FIGURE 2.10 Five tetromino types.

Is it possible to tile (i.e., cover exactly without overlaps) an 8×8 chessboard with the following?

(a) 16 straight tetrominoes
(b) 16 square tetrominoes
(c) 16 L-tetrominoes

(d) 16 T-tetrominoes
(e) 16 Z-tetrominoes
(f) 15 T-tetrominoes and one square tetromino

39. **Board Walks**
For each of the two boards in Figure 2.11, either find a path that passes through every square of the board exactly once or prove that no such path exists. A path may proceed through any sequence of horizontally or vertically adjacent squares and is not required to return to its starting square.

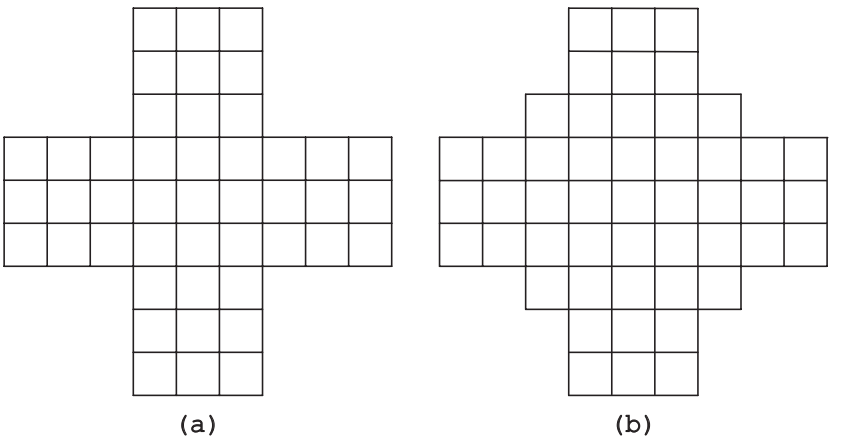


FIGURE 2.11 Two boards to find a path through all the squares.

40. **Four Alternating Knights**
There are four knights on a 3×3 chessboard: the two white knights are at the two bottom corners, and the two black knights are at the two upper corners of the board. Find the shortest sequence of moves to achieve the position shown on the right of Figure 2.12 or prove that no such sequence exists. Of course, no two knights can ever occupy the same square of the board.

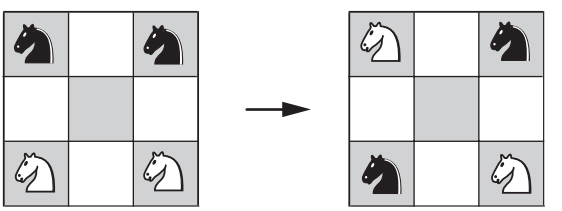


FIGURE 2.12 The *Four Alternating Knights* puzzle.

41. **The Circle of Lights**
There is a circle of $n > 2$ lights with a switch next to each of them. Each switch can be flipped between two positions, thereby toggling the on/off

states of three lights: its own and the two lights adjacent to it. Initially all the lights are off. Design an algorithm for turning all the lights on by flipping the minimum number of switches.

42. **The Other Wolf-Goat-Cabbage Puzzle**
You have $4n$ counters of four types: n wolfs, n goats, n cabbages, and n hunters. The object is to place the counters in a row such that no one is in danger; that is, no hunter is next to a wolf, no wolf is next to a goat, and no goat is next to a cabbage. In addition, no two counters of the same kind may be next to each other either. How many ways are there to solve the puzzle?

43. **Number Placement**
Given a list of n distinct integers and a sequence of n boxes with preset inequality signs inserted between them, design an algorithm that places the numbers into the boxes to satisfy those inequalities. For example, the numbers 2, 5, 1, and 0 can be placed in the four boxes as shown below:

$$\boxed{0} < \boxed{5} > \boxed{1} < \boxed{2}$$

44. **Lighter or Heavier?**
You have $n > 2$ identical-looking coins and a two-pan balance scale with no weights. One of the coins is a fake, but you do not know whether it is lighter or heavier than the genuine coins, which all weigh the same. Design an algorithm to determine in the minimum number of weighings whether the fake coin is lighter or heavier than the others.

45. **A Knight's Shortest Path**
What is the minimum number of moves needed for a chess knight to go from one corner of a 100×100 board to the diagonally opposite corner?

46. **Tricolor Arrangement**
A rectangular board with three rows and n columns is filled with $3n$ counters, of which n are red, n are white, and n are blue. The object is to rearrange the counters to have counters of each of the three different colors in every column. The only operation allowed is to swap counters in the same row. Design an algorithm to accomplish this task or prove that such an algorithm does not exist.

47. **Exhibition Planning**
A museum has an exhibition space made up of 16 rooms. Its floor plan is presented in Figure 2.13. There is a door between every pair of horizontally or vertically adjacent rooms. In addition, each room on the north and south side of the building (the top and bottom lines of the plan) has one door to the outside. In planning a new exhibition, the curator has to decide which of the doors need to be open so that a visitor can enter the exhibition through a door on the north side, visit each and every room exactly once, and get out through a door on the south side. Of course, the curator also wants to have as few doors open as possible.

(a) What is the minimum number of doors that need to be open for the exhibition?
(b) What entrance and exit doors need to be open for the exhibition?

Indicate all the entrance-exit pairs that can be open for the exhibition.

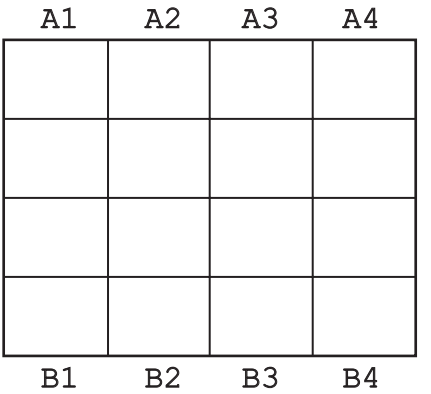


FIGURE 2.13 Floor plan of 16 rooms.

48. **McNugget Numbers**
A McNugget number is a positive integer that can be obtained by adding together orders of McDonald's Chicken McNuggets, which come in boxes of 4, 6, 9, and 20 pieces.

(a) Find all the positive integers that are *not* McNugget numbers.
(b) Design an algorithm to compute the number of boxes of each order size for any given McNugget number.

49. **Missionaries and Cannibals**
Three missionaries and three cannibals must cross a river. Their boat can only hold two people, and it cannot cross the river by itself with no people on board. Each missionary and each cannibal can row the boat. If present, missionaries cannot be outnumbered by cannibals. How can all six get across the river with the fewest crossings?

50. **Last Ball**

(a) You have 20 black balls and 16 white balls in a bag. You repeat the following operation until a single ball is left in the bag. You remove two balls at a time. If they are of the same color, you add a black ball to the bag; if they are of different colors, you add a white ball to the bag. Can you predict the color of the last ball left in the bag?
(b) Answer the same question if there are 20 black balls and 15 white balls to start with.

51. **Missing Number**
Jill bets Jack that she can do the following trick. Jack will recite 99 different numbers from 1 to 100 in a random order and she will be able to name the only number in that range that he will have missed. What is the best way for Jill to do the trick? Of course, she will have to perform the task in her head, without taking any notes.

52. **Counting Triangles**
An algorithm starts with a single equilateral triangle and on each subsequent iteration adds new triangles all around the outside. The results for the first three values of n are shown in Figure 2.14. How many small triangles will be there after the nth iteration?

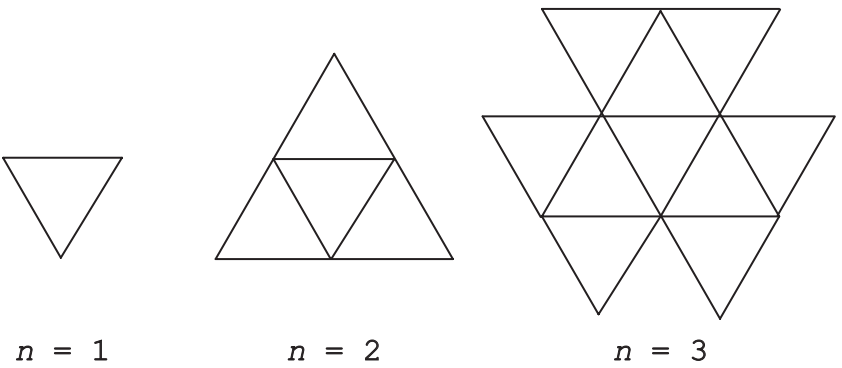


FIGURE 2.14 First iterations of the algorithm for the *Counting Triangles* puzzle.

53. **Fake-Coin Detection with a Spring Scale**
You have $n > 1$ identical-looking coins: $n - 1$ of them are genuine with a known weight g, and one of them—of an unknown weight different from g—is counterfeit. Design an algorithm that determines the fake in the minimum number of weighings on a spring scale. (Assume that the spring scale indicates the exact weight of the coins being weighed.)

54. **Cutting a Rectangular Board**
There is an $m \times n$ rectangular board drawn on graph paper. You need to cut it into mn 1×1 squares by straight cuts along the grid lines. You are allowed to stack several pieces together to cut them at the same time, which is considered one cut. Design an algorithm that performs this task with the minimum number of cuts.

55. **Odometer Puzzle**
A car odometer can display any six-digit combination from 000,000 to 999,999, inclusive. If it runs through its entire range, how many such combinations will have at least one digit 1 in them? What is the total

number of times the digit 1 will be displayed? (For example, 101,111 contributes five toward the total count of 1's, and the next reading of 101,112 adds four more.)

56. **Lining Up Recruits**
The good soldier Schweik had been ordered to line up a band of new recruits before their officer gave them a speech. The desired line sought to minimize the average difference in height of adjacent men. Schweik put the tallest recruit first, the shortest one last, and let the remaining men stand between them in random order. Did Schweik execute his order as stated? How would you arrange the recruits?

57. **Fibonacci's Rabbits Problem**
A man put a pair of rabbits in a place surrounded on all sides by a wall. The initial pair of rabbits (male and female) are newborn. All rabbit pairs are not fertile during their first month of life but give birth to one new male/female pair at the end of the second month and every month thereafter. How many pairs of rabbits will be there in a year?

58. **Sorting Once, Sorting Twice**
Shuffle a 52-card deck and lay the cards face up on a table to form an array of 4 rows and 13 columns. Sort each row by the cards' rank, that is in nondecreasing order of their numerical value (where the Ace is 1, and the Jack, Queen, and King are 11, 12, 13, respectively). Resolve ties among cards of the same rank by some predefined rule, say, clubs (lowest), followed by diamonds, hearts, and spades (highest). Then sort each column of the new array. If you decide to sort each row again, what is the largest number of card pairs that will have to be swapped to do this?

59. **Hats of Two Colors**
There are 12 very smart prisoners in a jail. To get rid of them, the warden comes up with the following test. He will put a hat, either black or white, on the head of each of these prisoners. There will be at least one hat of each color, and the prisoners will be informed about this fact. They will be able to see everyone else's hat but their own; there will be no communications of any kind among the prisoners. The warden will line up the prisoners every 5 minutes starting at 12:05 pm and ending at 12:55 pm. To pass the test, all the prisoners with a black hat and only those prisoners will have to step forward during the same line up. If they do, all the prisoners will be freed, otherwise they will be executed. How can the prisoners pass the test?

60. **Squaring a Coin Triangle**
Given a right isosceles triangle made of $n > 1$ lines containing $1, 3, \ldots, 2n - 1$ identical coins, respectively (see an example of such a triangle for $n = 3$ in Figure 2.15), what is the minimum number of coins that need to be moved to create a square made up of all the coins given? How many different squares can be obtained in the minimum number of moves?

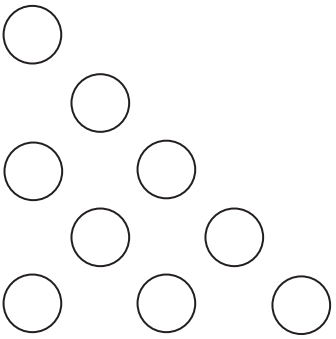


FIGURE 2.15 Instance of the *Squaring a Coin Triangle* puzzle for $n = 3$.

61. **Checkers on a Diagonal**
    On an $n \times n$ checkerboard, where $n \geq 4$, there is one checker on each square of its main diagonal from upper left to lower right. On each move, one can select any pair of the checkers and move each of them down one square, provided this will not move a checker off the board. The goal is to move all the checkers to the lowest row of the board. Find all the values of n for which this can be done and indicate an algorithm that performs the task. Also determine the number of moves made by your algorithm.

62. **Picking Up Coins**
    Some coins are spread in the cells of an $n \times m$ board, one coin per cell. A robot, located in the upper left cell of the board, needs to collect as many of the coins as possible and bring them to the bottom right cell. On each step, the robot can move either one cell to the right or one cell down from its current location. When the robot visits a cell with a coin, it picks up that coin. Devise an algorithm to find the maximum number of coins the robot can collect and a path it needs to follow to do this.

63. **Pluses and Minuses**
    The n consecutive integers from 1 to n are written in a row. Design an algorithm that puts signs "+" and "−" in front of them so that the expression obtained is equal to 0 or, if the task is impossible to do, returns the message "no solution." Your algorithm should be much more efficient than an examination of all possible ways to place the signs.

64. **Creating Octagons**
    There are 2000 points in the plane, no three of them on the same line. Devise an algorithm to construct 250 octagons with their vertices at these points. Each octagon has to be simple, that is, its boundary should not cross itself, and no two octagons may have a common point.

65. **Code Guessing**
    Your friend thinks of an n-bit string (some sequence of 0's and 1's such as 01011 for $n = 5$) that we will refer to as a code. Your goal is to identify the code by asking your friend questions. Each question will provide your

friend with an n-bit string of your choice, and your friend will tell you how many bits in your string coincide with the corresponding bits in the code. For example, if the code is 01011 and your question's string is 11001, the answer should be 3 because the two strings have the same bits in the second, third, and fifth positions. Devise an algorithm that can identify any n-bit code in no more than n questions.

66. **Remaining Number**
The first 50 natural numbers—1, 2, . . . , 50—are written on a board. You have to apply the following operation 49 times: select two of the numbers on the board, a and b, write the absolute value of their difference $|a - b|$ on the board, and then erase both a and b. Determine all possible values of the remaining number that can be obtained in this manner.

67. **Averaging Down**
There are 10 identical vessels, one of them with a pints of water and the others empty. You are allowed to perform the following operation: take two of the vessels and split the total amount of water in them equally between them. The object is to achieve a minimum amount of water in the vessel containing all the water in the initial set up by a sequence of such operations. What is the best way to do this?

68. **Digit Sum**
Without the help of a computer or calculator, find the total sum of the digits in all integers from 1 to a million, inclusive.

69. **Chips on Sectors**
A circle is divided into $n > 1$ sectors, and one chip is placed on each of them. A move is made by moving two chips to their neighboring sectors (in the same or opposite directions). For which values of n is there an algorithm to collect all the chips on the same sector?

70. **Jumping into Pairs I**
There are n coins placed in a row. The goal is to form $n/2$ pairs of them by a sequence of moves. On each move a single coin can jump right or left over two coins adjacent to it (i.e., over either two single coins or one previously formed pair), to land on the next single coin; no triples are allowed. Any empty space between adjacent coins is ignored. Determine all the values of n for which the puzzle has a solution and devise an algorithm that solves the puzzle in the minimum number of moves for such n's.

71. **Marking Cells I**
Mark n cells on an infinite sheet of graph paper so that each marked cell has a positive even number of marked neighbors. Two cells are considered neighbors if they are next to each other either horizontally or vertically, but not diagonally. The marked cells must form a contiguous region, that is, a region in which there is a path between any pair of marked cells that

goes through a sequence of marked neighbors. For example, a solution for $n = 4$ is shown in Figure 2.16. For which values of n can this be done?

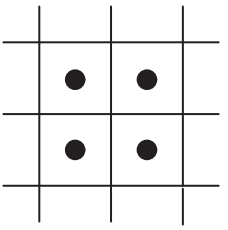


FIGURE 2.16 Four marked cells with an even number of marked neighbors.

72. **Marking Cells II**
Mark n cells on an infinite sheet of graph paper so that each marked cell has an odd number of marked neighbors. Two cells are considered neighbors if they are next to each other either horizontally or vertically, but not diagonally. The marked cells must form a contiguous region, that is, a region in which there is a path between any pair of marked cells that goes through a sequence of marked neighbors. For example, a solution for $n = 4$ is shown in Figure 2.17. For which values of n can this be done?

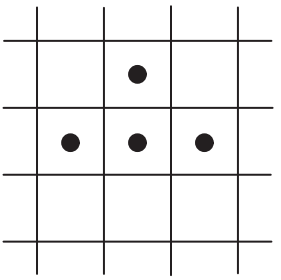


FIGURE 2.17 Four marked cells with an odd number of marked neighbors.

73. **Rooster Chase**
This game is played on the grid shown in Figure 2.18. The F counter at the lower left corner represents a farmer; the R counter at the upper right corner represents a rooster. The farmer and the rooster move alternately until the rooster is captured. On each move, each of them can move to a neighboring point on the grid: up, down, left, or right. A capture occurs when the farmer can move on a point occupied by the rooster.

(a) Can the farmer catch the rooster if he moves first? If he can, provide an algorithm to do this in the minimum number of moves. If he cannot, explain why.

(b) Can the farmer catch the rooster if he moves second? If he can, provide an algorithm to do this in the minimum number of moves. If he cannot, explain why.

Of course, you should assume that the rooster is not going to cooperate in his capture.

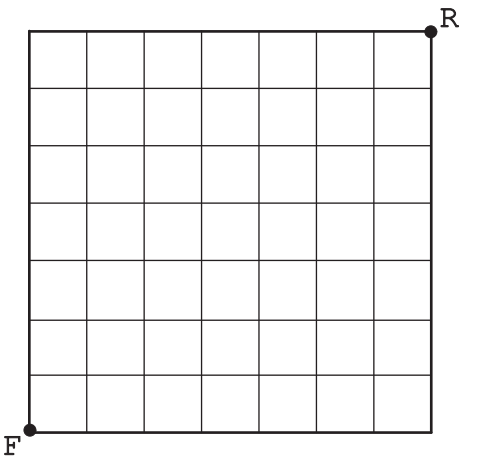


FIGURE 2.18 Initial position of the *Rooster Chase* game.

74. **Site Selection**

Consider the general case of the *Lemonade Stand Placement* puzzle, which is discussed in the tutorial on algorithm design techniques. Let $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ indicate the locations of n houses in a perfectly planned city with horizontal streets and vertical avenues (see Figure 1.10a in the first tutorial for an example). The object is to devise an algorithm to find a location (x, y) minimizing the sum of the Manhattan distances from the houses to that location, computed as $|x_1 - x| + |y_1 - y| + \cdots + |x_n - x| + |y_n - y|$.

75. **Gas Station Inspections**

A gas station inspector needs to inspect $n > 1$ gas stations located equidistantly along a straight highway. The stations are numbered consecutively from 1 to n. The inspector starts at station 1, which he will need to visit one more time. He will also need to visit station n twice, where he may but does not have to finish his tour. As to stations 2 through $n - 1$, he needs to inspect them an equal number of times. For example, he might go from station 1 to station n, turn back and return to station 1, and then go to station n again to complete his task there. (It is assumed, of course, that the inspector visits the stations every time he passes them.) The question is whether this is the shortest tour possible that satisfies all the requirements stated. If it is, prove it; if it is not, find the shortest one.

76. **Efficient Rook**
The rook in chess can move either horizontally to any square that is in the same row or vertically to any square that is in the same column as its current position. What is the minimum number of moves needed for the rook to pass over all the squares of an $n \times n$ chessboard? (Here, a tour does not have to start and end at the same square; the squares where it starts and ends are considered "passed over" by default.)

77. **Searching for a Pattern**
Perform the following multiplications:

$$1 \times 1, \quad 11 \times 11, \quad 111 \times 111, \quad 1111 \times 1111.$$

Will the pattern you observe in the products continue if you use longer strings of 1's?

78. **Straight Tromino Tiling**
A straight tromino is a 3×1 tile. Obviously, one can tile any $n \times n$ square with straight trominoes if n is divisible by 3. Is it true that for every $n > 3$ that is not divisible by 3, one can tile an $n \times n$ square with straight trominoes and a single 1×1 tile called a monomino? If it is possible, explain how; if it is not, explain why.

79. **Locker Doors**
There are n lockers in a hallway, numbered sequentially from 1 to n. Initially, all the locker doors are closed. You make n passes by the lockers, each time starting with locker #1. On the ith pass, $i = 1, 2, \ldots, n$, you toggle the door of every ith locker: if the door is closed, you open it; if it is open, you close it. Thus, after the first pass every door is open; on the second pass, you only toggle the even-numbered lockers (#2, #4, ...) so that after the second pass the even doors are closed and the odd ones are open; the third time through you close the door of locker #3 (opened from the first pass), open the door of locker #6 (closed from the second pass), and so on. After the last pass, which locker doors are open and which are closed? How many of them are open?

80. **The Prince's Tour**
Consider a special chess piece—to be called here a "prince"–that can move one square to the right, or one square downward, or one square diagonally upward to the left. Find all values of n for which a prince can visit all the squares of an $n \times n$ board exactly once on the same tour.

81. **Celebrity Problem Revisited**
A celebrity among a group of n people is a person who knows nobody but is known by everybody else. The task is to identify a celebrity by only asking questions to people of the form "Do you know him/her?" Design an efficient algorithm to identify a celebrity or determine that the group has

no such person. What is the largest number of questions your algorithm needs to solve the problem for n people?

82. **Heads Up**
There are n coins in a line, heads and tails in random order. On each move, one can turn over any number of coins laying in succession. Design an algorithm to turn all the coins heads up in the minimum number of moves. How many moves are required in the worst case?

83. **Restricted Tower of Hanoi**
There are n disks of different sizes and three pegs. Initially, all the disks are on the first peg in order of size, the largest on the bottom and the smallest on top. The object is to transfer all the disks to the third peg. Only one disk can be moved at a time, and it is forbidden to place a larger disk on top of a smaller one. In addition, any move should either place a disk on the middle peg or move a disk from that peg (Figure 2.19). Design an algorithm that solves the puzzle in the minimum number of moves.

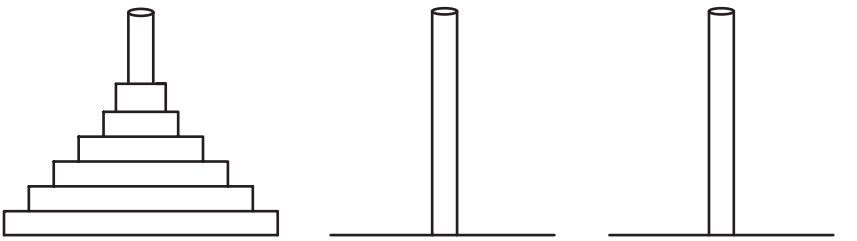


FIGURE 2.19 *Restricted Tower of Hanoi* puzzle: transfer through the middle peg all the disks from the left peg to the right peg.

84. **Pancake Sorting**
There are n pancakes, all of different sizes, that are stacked on top of each other. You are allowed to slip a spatula under one of the pancakes and flip over the whole stack above the spatula. The objective is to arrange the pancakes according to their size with the biggest at the bottom. Figure 2.20 shows an instance of the puzzle for $n = 7$. Design an algorithm for solving this puzzle and determine the number of flips made by the algorithm in the worst case.

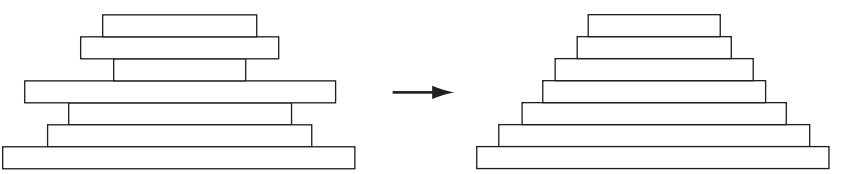


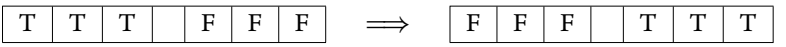
FIGURE 2.20 *Pancake Sorting* puzzle for $n = 7$.

85. **Rumor Spreading I**
There are n people, each in possession of a different rumor. They want to share the news with each other by sending electronic messages. What is the minimum number of messages they need to send to guarantee that everyone of them gets all the rumors? Assume that a sender includes all the rumors he or she knows at the time the message is sent and that a message may only have one addressee.

86. **Rumor Spreading II**
There are n people, each in possession of a different rumor. They want to share all the rumors through a series of bilateral conversations (e.g., via a telephone). Devise an efficient (in terms of the total number of conversations) algorithm for this task. Assume that in every conversation both parties exchange all the rumors they know at the time.

87. **Upside-Down Glasses**
There are n glasses on the table, all standing upside down. In one move, you are allowed to turn over exactly $n - 1$ of them. Determine all values of n for which all the glasses can be turned up, and outline an algorithm that does this in the minimum number of moves.

88. **Toads and Frogs**
On a one-dimensional board with $2n + 1$ cells, there are n counters in the first n cells representing Toads and n counters in the last n cells representing Frogs. Toads and Frogs take turns moving. Moves consist of sliding a Toad or Frog into the empty cell or jumping over one opposing creature to the empty cell. (Toads cannot jump over themselves and neither can Frogs.) Toads can only move rightward; Frogs can only move leftward. The object is to make them switch their positions. For example, for $n = 3$, the task can be depicted by the following diagram:



Devise an algorithm to accomplish this task.

89. **Counter Exchange**
This solitaire game is played on a two-dimensional board with $2n + 1$ rows and $2n + 1$ columns. All the $(2n + 1)^2$ positions of the board, except the central one, are occupied by counters of two colors, say, white (W) and black (B), as follows. In the first n rows, the first $n + 1$ positions are occupied by W's followed by n B's. In row $n + 1$, the first n positions are occupied by W's followed by one vacant position followed by n B's. In the last n rows, the first n positions are occupied by n W's followed by $n + 1$ B's. The W counters can move horizontally right or vertically down; the B counters can move horizontally left or vertically up. A move can be either a slide to the empty neighboring position or a jump over one counter of

the opposite color to the empty position right beyond it. No counter can jump over another counter of the same color. The object is to switch all the counters to the positions initially occupied by the counters of the opposite color (see Figure 2.21 for $n = 3$ for an illustration).

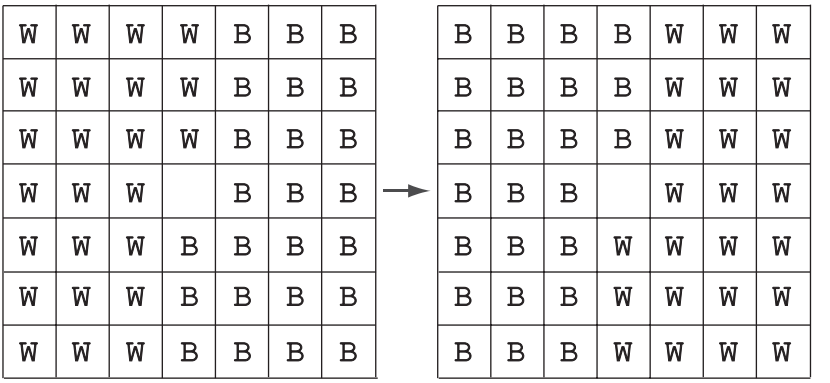


FIGURE 2.21 The *Counter Exchange* puzzle for $n = 3$.

90. **Seating Rearrangements**
There is a row of n chairs occupied by n children. Is it possible to design an algorithm for getting all possible seating arrangements of the children if a transition from one arrangement to another is made by two children sitting next to each other exchanging seats?

91. **Horizontal and Vertical Dominoes**
Determine all the values of n for which one can tile an $n \times n$ board so that the number of horizontal and vertical dominoes is the same.

92. **Trapezoid Tiling**
An equilateral triangle is partitioned into smaller equilateral triangles by parallel lines dividing each of its sides into $n > 1$ equal segments. The topmost equilateral triangle is chopped off yielding a region such as the one shown in Figure 2.22 for $n = 6$. This region needs to be tiled with

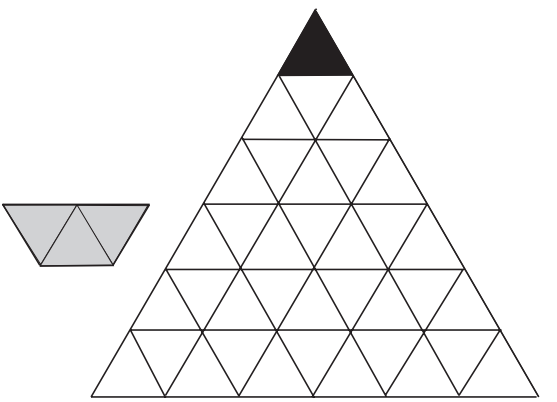


FIGURE 2.22 Region to tile with trapezoid tiles (in gray) for $n = 6$.

trapezoid tiles made of three equilateral triangles of the same size as the small triangles composing the region. (Tiles need not be oriented the same way, but they need to cover the region exactly with no overlaps.) Determine all values of n for which this can be done and devise a tiling algorithm for such n's.

93. **Hitting a Battleship**
What is the minimum number of shots needed to guarantee hitting a battleship (a 4×1 rectangle) on a 10×10 board? The battleship can be located anywhere on the board and may be oriented either horizontally or vertically. You may assume that there are no other ships. (A "shot" is a blind guess of a square on the board.)

94. **Searching a Sorted Table**
One hundred different numbers are written on 100 cards, one number per card. The cards are arranged in 10 rows and 10 columns, in increasing order in each row (left to right) and each column (top down). All the cards are turned faced down so that you cannot see the numbers written on them. Can you devise an algorithm to determine whether a given number is written on one of the cards by turning up less than 20 cards?

95. **Max-Min Weights**
Given $n > 1$ items and a two-pan balance scale with no weights, determine the lightest and the heaviest items in $\lceil 3n/2 \rceil - 2$ weighings.

96. **Tiling a Staircase Region**
Find all values of $n > 1$ for which a staircase region S_n (Figure 2.23 for $n = 8$) can be tiled with right trominoes?

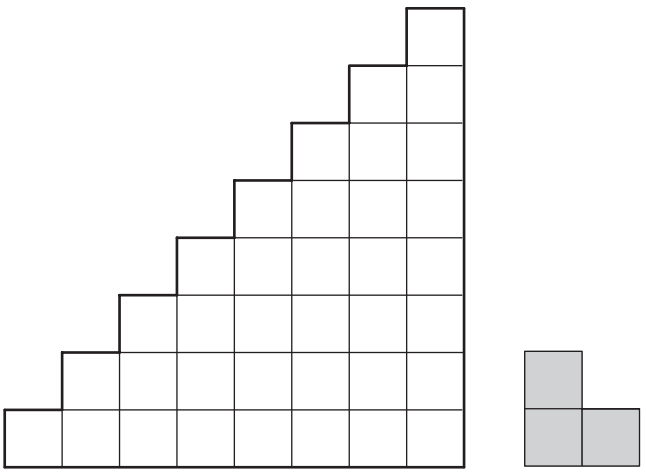


FIGURE 2.23 Staircase region S_8 to be tiled with right trominoes (in gray).

97. **The Game of Topswops**
Consider the following one-person card game. It is played with 13 cards of the same suit; each card is considered as having a numerical value: the ace

is 1, the deuce is 2, and so on, where the jack, queen, and king are 11, 12, and 13, respectively. Before the game begins, the cards are shuffled. After that the following operation is repeated. The top card of the deck is turned up. If it is the ace, the game stops. Otherwise, the top n cards, where n is the value of the top card, are removed from the deck and then returned there in reverse order. An example of the game's step is shown below:

$$5\,7\,10\,K\,8\,A\,3\,Q\,J\,4\,9\,2\,6 \implies 8\,K\,10\,7\,5\,A\,3\,Q\,J\,4\,9\,2\,6.$$

Does the game always stop after a finite number of iterations for every initial state of the deck?

98. **Palindrome Counting**
In how many different ways can the palindrome

WAS IT A CAT I SAW

be read in the diamond-shaped arrangement shown in Figure 2.24? You may start at any W and go in any direction on each step—up, down, left, or right—through adjacent letters. The same letter can be used more than once in the same sequence.

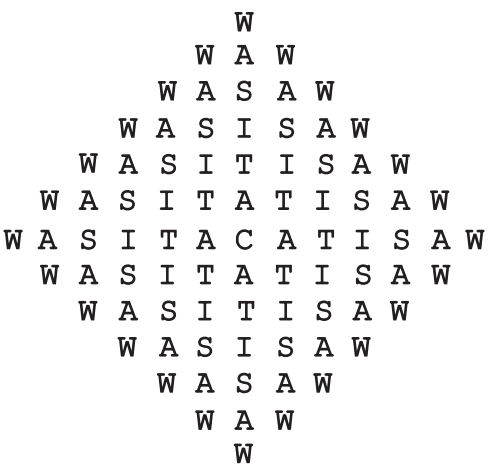


FIGURE 2.24 Letter arrangement for the *Palindrome Counting* puzzle.

99. **Reversal of Sort**
There are n index cards in a row, with n distinct integers written on them (one number per card) so that the numbers are sorted in decreasing order. You are allowed to exchange any pair of cards that have exactly one card between them. For which values of n is it possible to make the cards sorted in increasing order with a sequence of such operations? When it is possible, indicate an algorithm with the minimum number of exchanges.

100. **A Knight's Reach**
How many distinct squares can a chess knight reach after n moves on an infinite chessboard? (The knight's moves are L-shaped: two squares either up, down, left, or right and then one square in a perpendicular direction.)

101. **Room Painting**
There once lived a king who liked chess. He had a palace whose floor plan mimicked an 8×8 chessboard, with each of the 64 rooms having a door in each of its four walls. Originally, all the floors in all the rooms were painted white. Then the king ordered the floors to be repainted so that they alternated like the squares of a chessboard (Figure 2.25). To do this, his painter had to walk through the palace repainting floors in all the rooms he visited from white to black, and vice versa. The painter was allowed to exit the palace and reenter it at another door. Was there a way to execute the order by repainting the rooms not more than 60 times?

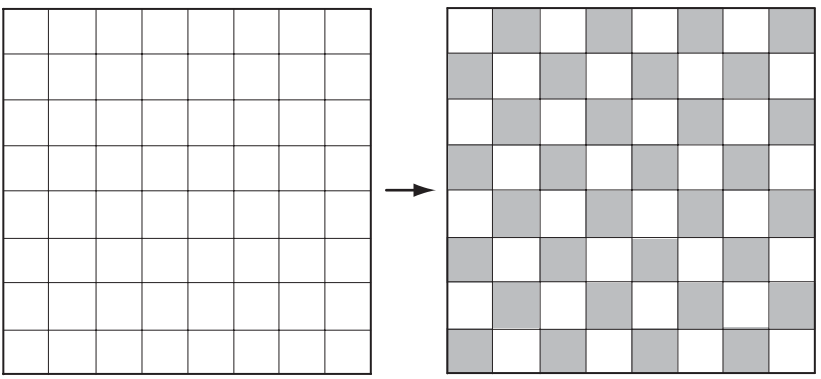


FIGURE 2.25 The *Room Painting* puzzle.

102. **The Monkey and the Coconuts**
Five sailors and a monkey were shipwrecked on a desert island. During the first day, they collected some coconuts to eat the next morning. At night, one sailor woke up, divided the coconuts into five equal piles after giving one coconut to the monkey, hid his pile, recombined the other four piles, and went back to sleep. Later that night, each of the other four sailors did the same thing one after the other: each gave one coconut to the monkey and then took one-fifth of the remaining ones for himself. In the morning, they divided all the remaining coconuts among themselves after giving one coconut to the monkey for the last time. What is the minimum number of coconuts that could have been in the original pile?

103. **Jumping to the Other Side**
On a 5×6 board, the 15 positions shown in black in Figure 2.26 are occupied by counters. The task is to move all the counters, which are

above the line, to the positions below the line. On each move, a counter can jump over an adjacent counter to an unoccupied position immediately beyond it. Jumps may be horizontal, vertical, or diagonal in any direction. Can this task be accomplished?

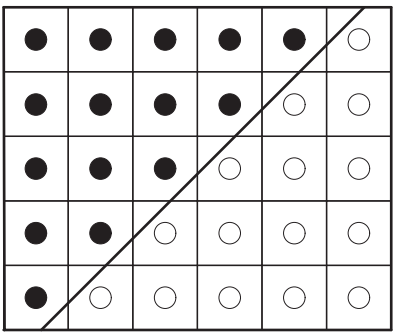


FIGURE 2.26 The board of the *Jumping to the Other Side* puzzle.

104. **Pile Splitting**

(a) Given n counters in a pile, split the counters into two smaller piles and compute the product of the numbers of the counters in the two piles obtained. Continue to split each pile into two smaller piles and to compute the products until there are n piles of size one. Once there are n piles, sum all the products computed. How should one split the piles to maximize the sum of the products? What is this maximal sum equal to?

(b) How does the solution to the puzzle change if we are to compute the sum of the numbers of the counters in the two piles obtained after every split and have a goal of maximizing the total of such sums?

105. **The MU Puzzle**

Consider strings composed of the three symbols M, I, and U that can be obtained by starting with the string MI and then applying the following transformation rules a finite number of times:

Rule 1. Add U at the end of any string ending in I, for example, changing MI to MIU.
Rule 2. Double any string after the M (that is, change Mx to Mxx), for example, changing MIU to MIUIU.
Rule 3. Replace any III with a U, for example, changing MUIIIU to MUUU.
Rule 4. Remove any UU, for example, changing MUUU to MU.

Is it possible to obtain the string MU using these rules?

106. **Turning on a Light Bulb**
A light bulb is connected to n switches in such a way that it lights up only when all the switches are closed. Each switch is controlled by a push button; pressing the button toggles the switch, but there is no way to know the state of the switch. Design an algorithm to turn on the light bulb with the minimum number of button pushes needed in the worst case.

107. **The Fox and the Hare**
Consider a chase game that we call *The Fox and the Hare*. The game is played on a one-dimensional board with 30 cells numbered left to right from 1 to 30. A chip representing the fox starts at cell 1, and a chip representing the hare starts at some cell $s > 1$. They move alternately, with the fox moving first. On each move, the fox can move left or right to a neighboring cell; the hare jumps left or right over two cells landing on the third. The hare cannot land on a cell occupied by the fox; if he does not have another move, he loses the game. And, of course, neither of them can move outside the board. The fox's goal is to catch the hare, which he can do if they occupy adjacent cells on the fox's move; the hare's goal is to avoid the capture. Find all the values of s for which the fox can win the game.

108. **The Longest Route**
If someone wants to attach a copy of a note to each of n posts, located at equal distances along a straight road, the best way is to start with the first post and put the notes as one passes the posts until the last one is reached. What would be the worst (i.e., longest) way to accomplish this task? It is not required to start at the first post and end at the last one, but all the turns must be made at the posts.

109. **Double-*n* Dominoes**
Dominoes are small rectangular tiles with dots called spots or pips embossed at both halves of the tiles. They are used to play a variety of games involving patterns on a tabletop. A standard "double-six" domino set has 28 tiles: one for each unordered pair of values from $(0, 0)$ to $(6, 6)$. In general, a "double-n" domino set would consist of domino tiles for each unordered pair of values from $(0, 0)$ to (n, n).

(a) Find the number of tiles in a double-n domino set.
(b) Find the total number of spots on all the tiles of a double-n domino set.
(c) Design an algorithm for constructing a ring made up of all the tiles in a double-n domino set or prove that no such algorithm exists. (Of course, every pair of adjacent tiles in the ring must have the same number of spots on their adjacent halves.)

110. **The Chameleons**
A researcher puts three types of chameleons on an island: 10 brown, 14 gray, and 15 black. When two chameleons of different colors meet, they both change their colors to the third one. Will it be possible for all the chameleons to become the same color?

HARDER PUZZLES

111. **Inverting a Coin Triangle**

Consider an equilateral triangle formed by closely packed pennies or other identical coins like the one shown in Figure 2.27. (The centers of the coins are assumed to be at the points of the equilateral triangular lattice.) Design an algorithm to flip the triangle upside down in the minimum number of moves if on each move you can slide one coin at a time to its new position. Give a compact formula for the number of minimum moves.

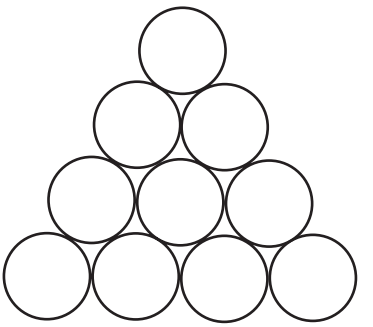


FIGURE 2.27 Equilateral triangle of coins to invert.

112. **Domino Tiling Revisited**

Find all the values of n for which an $n \times n$ chessboard with two missing squares of opposite colors can be tiled with 2×1 dominoes.

113. **Coin Removal**

There is a line of n coins on the table; some of them are heads up and the rest are tails up, in no particular order. The object of the puzzle is to remove all the coins by a sequence of moves. On each move, one can remove any head-up coin, after which its neighboring coin or coins, if any, must be turned over. Coins are considered "neighbors" if they are next to each other in the original line; if there is a gap between two coins after some moves, the coins are no longer considered "neighbors." For example, the following sequence of moves solves the puzzle for the coin line shown below. (The head-up coin being removed is shown in bold.)

T	H	H	T	**H**	H	H
T	**H**	H	H	_	T	H
H	_	T	H	_	T	H
_	_	T	**H**	_	T	H
_	_	**H**	_	_	T	H
_	_	_	_	_	T	**H**
_	_	_	_	_	**H**	_
_	_	_	_	_	_	_

Determine the property of the starting line that is necessary and sufficient for the puzzle to have a solution. For those lines that can be removed by the puzzle's rules, design an algorithm for doing so.

114. **Crossing Dots**
Given an $n \times n$ point lattice (intersection points of n consecutive horizontal and n consecutive vertical lines on common graph paper), where $n > 2$, cross out all the points by $2n - 2$ straight lines without lifting your pen from the paper. You may cross the same point more than once, but you cannot redraw any portion of the same line. (A "greedy" solution for $n = 4$, shown in Figure 2.28, has seven lines instead of the six required by the puzzle.)

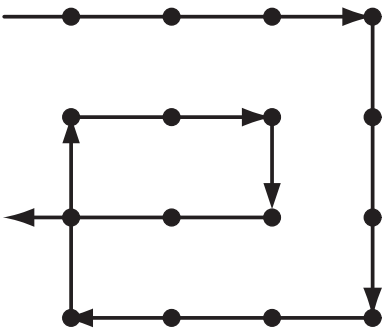


FIGURE 2.28 Crossing 16 points by 7 lines.

115. **Bachet's Weights**
Find an optimal set of n weights $\{w_1, w_2, \ldots, w_n\}$ so that it would be possible to weigh on a two-pan balance scale any integral load in the largest possible range from 1 to W, assuming the following:

(a) Weights can be put only on the free pan of the scale.
(b) Weights can be put on both pans of the scale.

116. **Bye Counting**
If a single-elimination tournament starts with the number of players n not equal to a power of 2, some players need to be given byes, which are transfers of players directly to the next round because they have no opponent assigned to them. Determine the total number of byes in such a tournament under the two different definitions of byes given below.

(a) Byes are given to the fewest players in the first round so that the number of players left for the second round is equal to a power of 2.
(b) Byes are given to the fewest players in each round so that there is an even number of players in that round.

117. **One-Dimensional Solitaire**

Consider the one-dimensional version of peg solitaire played on an array of n cells, where n is even and greater than 2. Initially, all but one cell are occupied by some counters (pegs), one peg per cell. On each move, a peg jumps over its immediate neighbor to the left or to the right to land on an empty cell; after the jump, the jumped-over neighbor is removed from the board. The object is to remove all but one peg by a sequence of such moves. Find all the locations of the empty cell in the initial setup for which the puzzle can be solved and the corresponding locations of the single remaining peg.

118. **Six Knights**

There are six knights on a 3×4 chessboard: the three white knights are at the bottom row, and the three black knights are at the top row. Exchange the knights to get the position shown on the right of Figure 2.29 in the minimum number of knight moves, not allowing more than one knight on a square at any time.

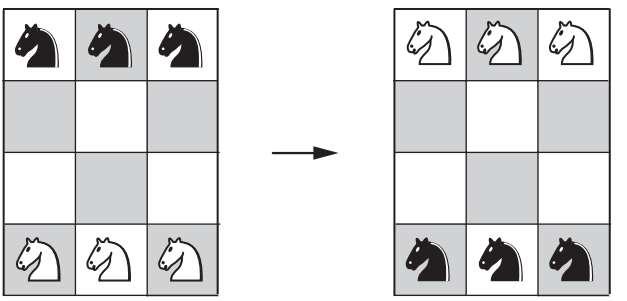


FIGURE 2.29 The *Six Knights* puzzle.

119. **Colored Tromino Tiling**

Devise an algorithm for the following task: given a $2^n \times 2^n (n > 1)$ board with one missing square, tile it with right trominoes of only three colors so that no pair of trominoes that share an edge have the same color. Recall that the right tromino is an L-shaped tile formed by three adjacent squares (see Figure 1.4).

120. **Penny Distribution Machine**

A "machine" consists of a row of boxes. To start, one places n pennies in the leftmost box. The machine then redistributes the pennies as follows. On each iteration, it replaces a pair of pennies in one box with a single penny in the next box to the right. The iterations stop when there is no box with more than one coin. For example, Figure 2.30 shows the work of the machine in distributing six pennies by always selecting a pair of pennies in the leftmost box with at least two coins.

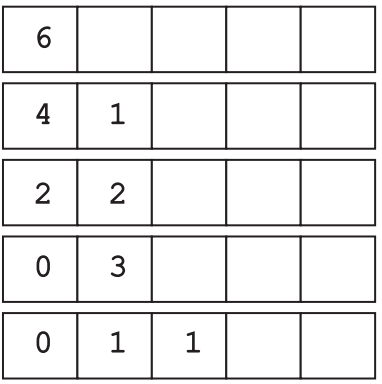


FIGURE 2.30 Example of penny distribution.

(a) Does the final distribution of pennies depend on the order in which the machine processes the coin pairs?
(b) What is the minimum number of boxes needed to distribute n pennies?
(c) How many iterations does the machine make before stopping?

121. **Super-Egg Testing**
A firm has invented a super-strong egg. For publicity purposes, it wants to determine the highest floor in a 100-story building from which such an egg can fall without breaking. The firm has given a tester two identical eggs to experiment with. Of course, the same egg can be dropped multiple times unless it breaks. What is the minimum number of droppings that is guaranteed to determine the highest safe floor in all cases?

122. **Parliament Pacification**
In a parliament, each member has at most three enemies. (We assume that enmity is always mutual.) True or false: one can always divide the parliament into two chambers in such a way that no parliamentarian has more than one enemy in his or her chamber?

123. **Dutch National Flag Problem**
There is a row of n checkers of three colors: red, white, and blue. Devise an algorithm to rearrange the checkers so that all the red checkers come first, all the white ones come next, and all the blue checkers come last. The only operations allowed are examination of a checker's color and swap of two checkers. Try to minimize the number of swaps made by your algorithm.

124. **Chain Cutting**
You have a chain of $n > 1$ paper clips. What is the minimum number of single clips that must be removed from the chain so that it would be possible to create a chain of any integer length between 1 and n clips, inclusive, from the resulting pieces?

125. **Sorting 5 in 7**
There are five items of different weights and a two-pan balance scale with no weights. Order the items in increasing order of their weights, making no more than seven weighings.

126. **Dividing a Cake Fairly**
There are $n > 1$ friends who want to divide a cake among them so that everyone is satisfied with the portion he gets. Devise an algorithm for this task.

127. **The Knight's Tour**
Is it possible for a chess knight to visit all the cells of an 8×8 chessboard exactly once, ending at a cell one knight's move away from the starting cell? (Such a tour is called closed or re-entrant. Note that a cell is considered visited only when the knight lands on it, not just passes over it on its move.)

128. **Security Switches**
There is a row of n security switches protecting a military installation entrance. The switches can be manipulated as follows:

(i) The rightmost switch may be turned on or off at will.
(ii) Any other switch may be turned on or off only if the switch to its immediate right is on and all the other switches to its right, if any, are off.
(iii) Only one switch may be toggled at a time.

Devise an algorithm to turn off all the switches, which are initially all on, in the minimum number of moves. (Toggling one switch is considered one move.) Also find the minimum number of moves.

129. **Reve's Puzzle**
There are eight disks of different sizes and four pegs. Initially, all the disks are on the first peg in order of size, the largest on the bottom and the smallest on the top. The objective is to transfer all the disks to another peg by a sequence of moves. Only one disk can be moved at a time, and it is forbidden to place a larger disk on top of a smaller one. Devise an algorithm that solves the puzzle in 33 moves.

130. **Poisoned Wine**
An evil king is informed that one of his 1000 wine barrels has been poisoned. The poison is so potent that a miniscule amount of it, no matter how diluted, kills a person in exactly 30 days. The king is prepared to sacrifice 10 of his slaves to determine the poisoned barrel.

(a) Can this be done before a feast scheduled in 5 weeks?
(b) Can the king achieve his goal with just eight slaves?

131. **Tait's Counter Puzzle**

There is a line of $2n$ counters with no spaces between adjacent counters. The counters alternate between black and white: B W B W . . . B W. The objective is to rearrange the counters so that all the white counters are before all the black ones with no gaps between the counters: W W . . . W B B . . . B. The counters are to be moved in pairs; on each move, a pair of adjacent counters can be moved, without changing their order, into a vacant location. Design an algorithm that solves the problem in n moves for any $n \geq 3$.

132. **The Solitaire Army**

This variation of peg solitaire is played on an infinite two-dimensional board with a horizontal line separating the board into two halves. In an initial position, a number of pegs (the "soldiers of the solitaire army") are placed below this line. The object is to advance one of the pegs (the army's "scout") as far above the line as possible by horizontal or vertical jumps. On each move, a peg jumps over its immediate neighbor vertically or horizontally to land on an empty cell; after the jump, the jumped-over neighbor is removed from the board. For example, to advance a peg to a cell in the first row above the line, two pegs will suffice (Figure 2.31a); to advance a peg to a cell in the second row, four pegs are both necessary and sufficient (Figure 2.31b).

Find an initial configuration of the following:

(a) 8 pegs to advance 1 peg to a cell in the third row above the line
(b) 20 pegs to advance 1 peg to a cell in the fourth row above the line

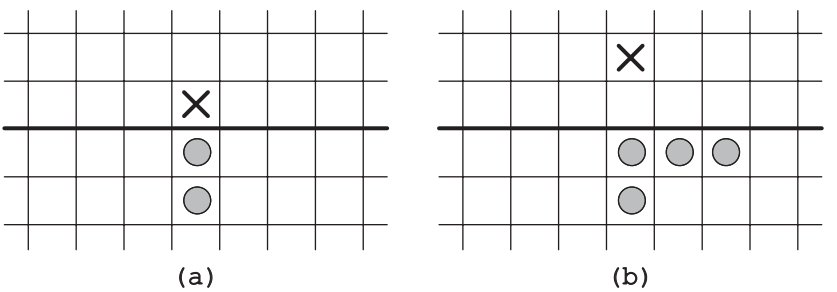


FIGURE 2.31 Solutions to the *Solitaire Army* puzzle. (a) Advancing a peg to one row. (b) Advancing a peg two rows above the "enemy" line. X denotes the target cell.

133. **The Game of Life**

This solitaire game is "played" on an infinite two-dimensional grid of square cells. Each of the cells is always in one of two possible states: live or dead. After some initial configuration of live cells is selected—for example, by marking them with a black dot—a sequence of new

configurations called "generations" is obtained by the following rules, which are applied simultaneously to every cell in the current generation. Every cell interacts with its eight neighbors, which are the cells that are adjacent to it horizontally, vertically, or diagonally. At each step in time, the following transitions occur:

(i) *Death by underpopulation* Any live cell with fewer than two live neighbors dies.
(ii) *Death by overcrowding* Any live cell with more than three live neighbors dies.
(iii) *Survival* Any live cell with two or three live neighbors lives on to the next generation.
(iv) *Birth* Any dead cell with exactly three live neighbors becomes the live cell.

(a) Find the smallest initial configuration of live cells that will remain the same with every generation. (Such configurations are called "still lifes.")
(b) Find the smallest initial configuration of live cells that will oscillate between two states. (Such configurations are called "oscillators.")
(c) Find the smallest initial configuration of live cells that will move itself across the board. (Such configurations are called "spaceships.")

134. **Point Coloring**

Design an algorithm for the following task. Given n arbitrary points on the grid, paint them in two colors, say, black and white, so that for every line, horizontal or vertical, the numbers of black and white points on the line are either the same or differ by one.

135. **Different Pairings**

A kindergarten teacher has to arrange $2n$ children in n pairs for daily walks. Design an algorithm for this task so that for $2n - 1$ days no pair would be the same.

136. **Catching a Spy**

In a computer game, a spy is located on a one-dimensional line. At time 0, the spy is at location a. With each time interval, the spy moves b units to the right if $b \geq 0$, and $|b|$ units to the left if $b < 0$. Both a and b are fixed integers, but they are unknown to you. Your goal is to identify the spy's location by asking at each time interval (starting at time 0) whether the spy is currently at some location of your choosing. For example, you can ask whether the spy is currently at location 19, to which you will receive a truthful yes/no answer. If the answer is "yes," you reach your goal; if the answer is "no," you can ask the next time whether the spy is at the same or another location of your choice. Devise an algorithm that will find the spy after a finite number questions.

137. **Jumping into Pairs II**
There are n coins placed in a row. The goal is to form $n/2$ pairs of them by a sequence of moves. On the first move a single coin has to jump over one coin adjacent to it, on the second move a single coin has to jump over two adjacent coins, on the third move a single coin has to jump over three adjacent coins, and so on, until after $n/2$ moves $n/2$ coin pairs are formed. (On each move, a coin can jump right or left but it has to land on a single coin. Jumping over a coin pair counts as jumping over two coins. Any empty space between adjacent coins is ignored.) Determine all the values of n for which the problem has a solution and design an algorithm that solves it in the minimum number of moves for those n's.

138. **Candy Sharing**
In a kindergarten, there are n children sitting in a circle facing their teacher in the center. Each child initially has an even number of candy pieces. When the teacher blows a whistle, each child simultaneously gives half of his or her candy pieces to the neighbor on the left. Any child who ends up with an odd number of pieces is given another piece by the teacher. Then the teacher blows her whistle again, unless all the children have the same number of candies, in which case the game stops. Can this game go on forever or will it eventually stop to let the children go on with their lives?

139. **King Arthur's Round Table**
King Arthur wants to seat $n > 2$ knights around his Round Table so that none of the knights is seated next to his enemy. Show how this can be done if the number of friends for each knight is not smaller than $n/2$. You may assume that friendship and enmity are always mutual.

140. **The n-Queens Problem Revisited**
Consider the problem of placing n queens on an $n \times n$ chessboard so that no two queens are in the same row, column, or diagonal. Design a linear-time algorithm for finding a solution to this problem for any $n > 3$.

141. **The Josephus Problem**
There are n people numbered 1 to n standing in a circle. Starting the count with person number 1, every second person is eliminated until only one person is left. Where in the circle should a person stand to remain the last person standing?

142. **Twelve Coins**
There are 12 coins identical in appearance; either all are genuine or exactly one of them is fake. It is unknown whether the fake coin is lighter or heavier than the genuine one. You have a two-pan balance scale without weights. The problem is to find whether all the coins are genuine and, if not, to find the fake coin and establish whether it is lighter or heavier than the genuine ones. Design an algorithm to solve the problem in the minimum number of weighings.

143. **Infected Chessboard**

A virus spreads through squares of an $n \times n$ chessboard by infecting any square that has two infected neighbors (horizontally or vertically, but not diagonally). What is the minimum number of unit squares that need to be infected initially for the virus to spread to the entire board?

144. **Killing Squares**

Consider an $n \times n$ board formed by $2n(n + 1)$ toothpicks that serve as borders of the 1×1 squares (see Figure 2.32 for an example). Design an algorithm to remove the minimum number of toothpicks that will break the perimeter of every square of any size.

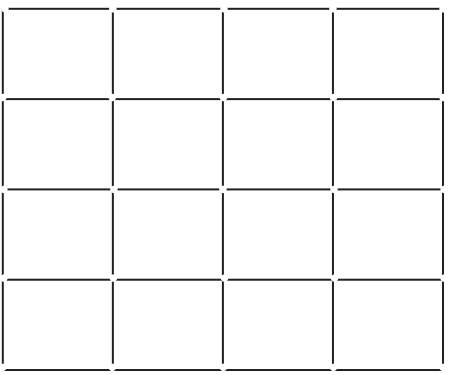


FIGURE 2.32 The 4×4 board for the *Killing Squares* puzzle.

145. **The Fifteen Puzzle**

This famous puzzle consists of fifteen square tiles numbered from 1 to 15 which are placed in a 4×4 box leaving one square out of the sixteen empty. The goal is to reposition the tiles from a given starting arrangement by sliding them one at a time into the configuration in which the tiles are ordered sequentially. Is it possible to solve the puzzle for the initial configuration shown in Figure 2.33?

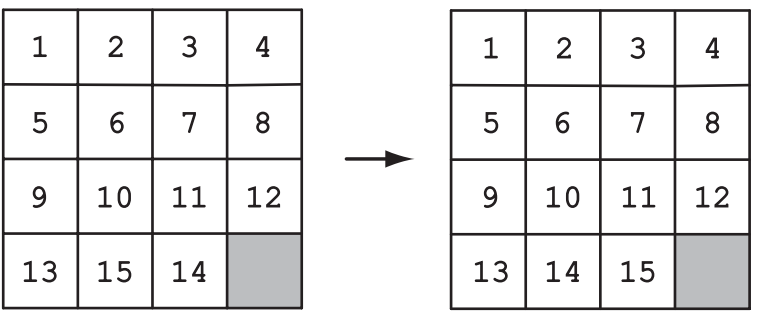


FIGURE 2.33 Initial and final positions of the *Fifteen Puzzle*.

146. **Hitting a Moving Target**
A computer game has a shooter and a moving target. The shooter can hit any of $n > 1$ hiding spots located along a straight line in which the target can hide. The shooter can never see the target; all he knows is that the target moves to an adjacent hiding spot between every two consecutive shots. Design an algorithm that guarantees hitting the target or prove that no such algorithm exists.

147. **Hats with Numbers**
At a university-wide New Year's party, there were $n > 1$ mathematicians. They conspired to challenge the university's president, who also attended the party, to the following bet. The president will write any number from 0 to $n - 1$, inclusive, on the party hats the mathematicians are wearing. The numbers may but need not be all different. After seeing the numbers written on all the hats but their own, with no communication whatsoever among themselves or anyone else, each mathematician will write his hat's number on a piece of paper and give it to the president. Of course, none of them will see the numbers written by the others. If at least one of the numbers is correct, they win their collective bet and the president will increase the next year's budget for their department by 5 percent. If none of them guessed right, the budget will be frozen for the next 5 years. Are the mathematicians bluffing or do they have a way to win the bet?

148. **One Coin for Freedom**
A jailer offers to free two imprisoned programmers—we call them A and B—if they manage to win the following guessing game. The jailer sets up an 8×8 board with one coin on each cell, some tails up and the others tails down. While B is absent, the jailer points out to A the board's cell the jailer has selected to be guessed by B. Prisoner A is required to turn over exactly one coin on the board before leaving the room. Then B enters and guesses the selected cell. A and B are allowed to plan their strategy beforehand, but there should be no communication between them after the game begins. Of course, B is allowed to see the board after he enters the room and even to perform any calculations he may wish to do. Can the prisoners win their freedom or is the game impossible to win?

149. **Pebble Spreading**
Consider the following one-person game that is played on an infinite board obtained by dividing the first quadrant into square cells. It starts with a single pebble placed at the corner of the board. On each move, the player can replace a pebble by placing two pebbles in the cells adjacent to it: one immediately to the right and the other immediately above, provided these two cells are empty. The object of the game is to remove all the pebbles from a staircase region S_n, which is composed of the n consecutive diagonals at the board's corner (see Figure 2.34 for examples).

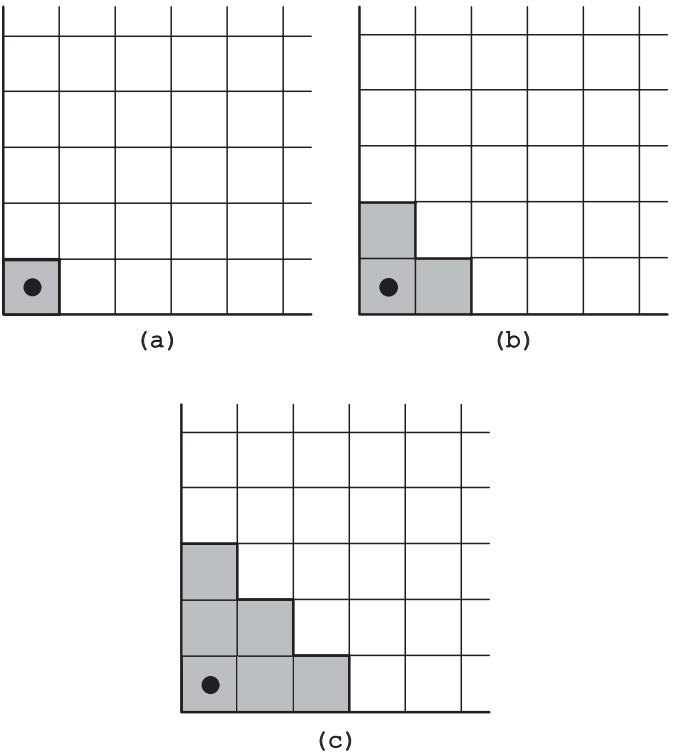


FIGURE 2.34 Starting positions of the *Pebble Spreading* game for (a) $n = 1$, (b) $n = 2$, and (c) $n = 3$.

For example, for $n = 1$, the first and only possible move from the starting position frees S_1 (Figure 2.35).

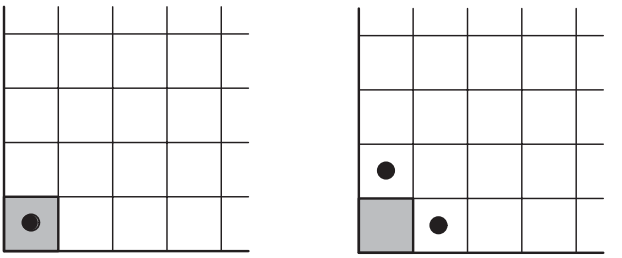


FIGURE 2.35 First move in the *Pebble Spreading* game for $n = 1$ frees staircase S_1 (shaded).

Find all the values of n for which the game's objective can be achieved.

150. **Bulgarian Solitaire**

Take n coins where n is a triangular number (i.e., $n = 1 + 2 + \ldots + k$ for some positive integer k), and divide them into $s \geq 1$ piles, with

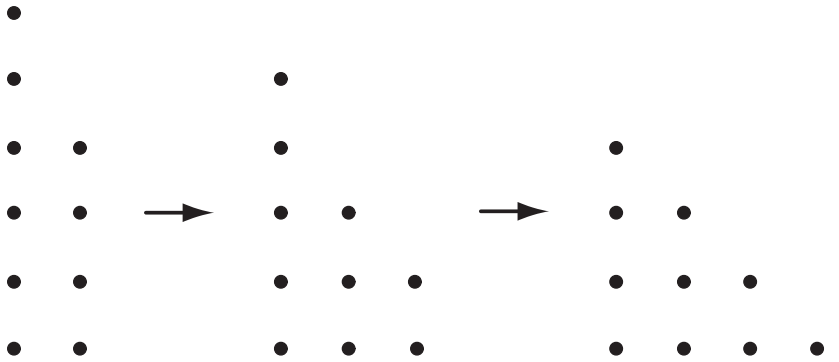


FIGURE 2.36 *Bulgarian Solitaire* example.

no restriction on the number of piles or the numbers of coins in the piles. Then perform repeatedly the following operation. Take one coin from each pile and put all of them in a new pile. Show that no matter what the initial partition of n coins is, the process will always arrive at k piles containing $1, 2, \ldots, k$ coins, respectively, after a finite number of iterations. (After reaching this state, the algorithm will obviously remain in it.) For example, Figure 2.36 illustrates the process for 10 coins initially divided into two piles of 6 and 4 coins. Note that since Bulgarian solitaire does not distinguish the pile order, it is convenient to list the piles in nonincreasing order of their size, as Figure 2.36 does.

3 Hints

1. **A Wolf, a Goat, and a Cabbage** With one insignificant exception, the puzzle can be solved by making a sequence of the only moves available in each situation.
2. **Glove Selection** Imagine a malevolent adversary who wants you to draw as many gloves as possible before getting what you need. Note that gloves are not socks: they can be right-handed and left-handed.
3. **Rectangle Dissection** Triangles in question need not be of the same size.
4. **Ferrying Soldiers** Solve the problem of ferrying one soldier first.
5. **Row and Column Exchanges** The answer is "no"; determine why.
6. **Predicting a Finger Count** Reenact the girl's count long enough to see a pattern that makes the answer obvious.
7. **Bridge Crossing at Night** The answer is "yes," and the solution does not involve any tricks.
8. **Jigsaw Puzzle Assembly** A similar problem is discussed in the book's tutorial on algorithm analysis techniques.
9. **Mental Arithmetic** There are at least two different ways to compute this sum. Both use the methods discussed in the tutorial on algorithm analysis techniques.

10. **A Fake Among Eight Coins** "Three" is not the correct answer to the puzzle.

11. **A Stack of Fake Coins** The answer is "one." Take advantage of the fact that the scale gives the exact weight.

12. **Questionable Tiling** The answer is "no."

13. **Blocked Paths** Use dynamic programming as explained in the tutorial on algorithm design strategies.

14. **Chessboard Reassembly** What parts of the board do you have to cut to solve the puzzle?

15. **Tromino Tilings** Only one of the three questions has a "yes" answer.

16. **Making Pancakes** What is the fastest way to make three pancakes? Also note that $n = 1$ is, in fact, a special case here.

17. A **King's Reach** The puzzle statement does not forbid the king to visit the same square more than once. Also, make sure that your answer is correct for every value of $n \geq 1$.

18. **A Corner-to-Corner Journey** Observe the colors of the squares the knight jumps through.

19. **Page Numbering** Derive a formula expressing the total number of digits as a function of the number of pages.

20. **Maximum Sum Descent** Use the dynamic programming strategy.

21. **Square Dissection** There are just a few values of n for which no such dissection exists. Also note that smaller squares need not be of the same size.

22. **Team Ordering** A required ordering can be easily obtained by one of the algorithm design strategies outlined in the first tutorial.

23. **Polish National Flag Problem** Advance toward the goal two checkers at a time.

24. **Chessboard Colorings** For each of the pieces except the rook, the solution can be found by a straightforward application of the greedy strategy. A simple solution for the rook is not hard to find either.

25. **The Best Time to Be Alive** You may manipulate the index given in alphabetical order.

26. **Find the Rank** It might be easier to find the number of "words" following TURING.

27. **The Icosian Game** Keep in mind that your path does not need to traverse every edge; it just needs to visit all the vertices. You may use backtracking here, but prepare to be lucky or very patient.

28. **Figure Tracing** *The Königsberg Bridges Problem* discussed in the tutorial on analysis techniques is based on the insight that is the key to this puzzle as well.

29. **Magic Square Revisited** See the discussion of magic square construction in the tutorial on algorithm design strategies.

30. **Cutting a Stick** Concentrate on the longest piece remaining.

31. **The Three Pile Trick** Denote cards in the three piles of the first layout, say, a1, a2, . . . , a9; b1, b2, . . . , b9; c1, c2, . . . , c9 and trace the algorithm.

32. **Single-Elimination Tournament** Start by answering the questions when n is a power of 2.

33. **Magic and Pseudo-Magic** There are $(n-2)^2$ 3×3 squares inside an $n \times n$ table. Answer first the puzzle's questions for a 4×4 table.

34. **Coins on a Star** The puzzle can be solved by applying the greedy strategy or by using the "buttons and strings" method mentioned in the tutorial on general strategies for algorithm design.

35. **Three Jugs** The puzzle can be solved in six steps.

36. **Limited Diversity** Solve the puzzle for $n = 2$, 3, and 4 for a critical insight.

37. **2*n*-Counters Problem** The problem is mentioned in the divide-and-conquer discussion in the first tutorial.

38. **Tetromino Tiling** The answer to four of the questions is "yes."

39. **Board Walks** A required path exists for one of the boards and does not exists for the other.

40. **Four Alternating Knights** See the tutorial on design strategies where the classic version of this puzzle is discussed.

41. **The Circle of Lights** The solution is not the same for all values of n. Considering a few small instances of the puzzle should help to see the difference.

42. **The Other Wolf-Goat-Cabbage Puzzle** Solve first the instance of the puzzle for $n = 1$ to see all possible arrangements of the counters.

43. **Number Placement** Start by sorting the numbers given.

44. **Lighter or Heavier?** You are not asked to determine the fake coin, just whether it is lighter or heavier than the others.

45. **A Knight's Shortest Path** A minimum-move sequence is rather obvious here. Proving its optimality is somewhat harder but becomes easy with the right way to measure the distance between the start and finish squares.

46. **Tricolor Arrangement** The task can be accomplished for any $n \geq 1$.

47. **Exhibition Planning** The answer to the first question is all but obvious. The answer to the second stems from a standard application of the invariant idea, which is discussed in the tutorial on algorithm analysis techniques.

48. **McNugget Numbers** There are just six integers that are not McNugget numbers. For the rest, the algorithm in question can be based on the decrease-and-conquer strategy.

49. **Missionaries and Cannibals** The puzzle can be solved in 11 river crossings. Note that a similar problem is discussed in the book's tutorial on algorithm design strategies.

50. **Last Ball** Think parities.

51. **Missing Number** After you get the main idea, try to improve on it to make Jill's task easy enough to perform in her head.

52. **Counting Triangles** Use a pattern in the number of small triangles added on each iteration. A similar example is discussed in the tutorial on algorithm analysis techniques.

53. **Fake-Coin Detection with a Spring Scale** What information does a weighing of a subset of the coins provide?

54. **Cutting a Rectangular Board** You might want to start by solving *Cutting a Stick* (#30), which is a one-dimensional version of this puzzle.

55. **Odometer Puzzle** The first question can be answered by standard combinatorial reasoning. With some ingenuity, the second question can be answered without cumbersome computations.

56. **Lining Up Recruits** One may answer the first question affirmatively. The intended order was, of course, different; it could have been executed in two different ways.

57. **Fibonacci's Rabbits Problem** Set up an equation expressing the number of rabbits after n months in terms of the number of rabbits in some previous months.

58. **Sorting Once, Sorting Twice** Solve the problem for a small two-dimensional array of cards or numbers to get a crucial insight.

59. **Hats of Two Colors** Suppose only one of the hats is black. How can the prisoner wearing it figure this out? What about the other prisoners? After answering these questions, generalize your answers to solve the puzzle.

60. **Squaring a Coin Triangle** The formula for the sum of the first n odd numbers $S_n = 1 + 3 + \cdots + (2n - 1) = n^2$ is useful for solving the puzzle. It might also be helpful to visualize the triangle with its right angle as the apex and the horizontal coin rows parallel to the hypotenuse.

61. **Checkers on a Diagonal** Although solving the puzzle for a few small values of n could definitely help, a better approach is to find an invariant—some feature that remains constant from move to move.

62. **Picking Up Coins** Dynamic programming seems to be the most appropriate strategy to use here.

63. **Pluses and Minuses** Use the formula $1 + 2 + \ldots + n = n(n + 1)/2$ and explore the sum's parity.

64. **Creating Octagons** Solve the problem for eight points first.

65. **Code Guessing** A sequence of n bit strings with a simple pattern can be used to identify the code one bit at a time.

66. **Remaining Number** Think parity.

67. **Averaging Down** You may use the greedy strategy, but you will need to prove that it does achieve the stated goal.

68. **Digit Sum** It might be easier to solve the general instance of the problem by computing the total digit sum in all integers from 1 to 10^n, where n is a positive integer. In fact, there are at least three different ways to solve this general instance of the problem.

69. **Chips on Sectors** Think parity.

70. **Jumping into Pairs I** You may use backtracking to find the minimum number of coins for which the problem has a solution.

71. **Marking Cells I** There are six values of n for which the puzzle cannot be solved, three of which are obvious.

72. **Marking Cells II** The answers are different for even and odd values of n.

73. **Rooster Chase** How can the farmer force a capture of the rooster if the rooster tries to avoid it? What algorithm makes reaching such a position as quick as possible?

74. **Site Selection** There is a much more efficient algorithm for this problem than the one used in the tutorial. Solving a few small special instances of the puzzle where all the homes are located on the same street should lead you to this algorithm for the general instance of the puzzle.

75. **Gas Station Inspections** You may need to consider the cases of odd and even n separately.

76. **Efficient Rook** An optimal tour can be constructed by the greedy strategy. Proving the tour's optimality, however, is not as straightforward.

77. **Searching for a Pattern** The answer is different for decimal and binary numbers.

78. **Straight Tromino Tiling** The answer is "yes."

79. **Locker Doors** Tracing the algorithm by hand for, say, $n = 10$, and studying its outcome should help to answer both questions.

80. **The Prince's Tour** The same algorithm can make a prince to visit all the squares of an $n \times n$ board exactly once for any positive value of n. Note that the puzzle's statement does not require the tour to be re-entrant, that is, to end one move away from its starting square.

81. **Celebrity Problem Revisited** A simpler version of the problem is solved in the book's first tutorial.

82. **Heads Up** Think blocks of successive heads and tails.

83. **Restricted Tower of Hanoi** The puzzle can be solved by a recursive algorithm similar to the one for the classic version of this puzzle (see the second tutorial).

84. **Pancake Sorting** Your algorithm does not have to be optimal, but it should definitely be more efficient than exhaustive search.

85. **Rumor spreading I** The minimum number of messages for $n = 4$ is six.

86. **Rumor Spreading II** There are several algorithms that require $2n - 4$ conversations for $n > 3$.

87. **Upside-Down Glasses** Think parity.

88. **Toads and Frogs** The puzzle can be solved by a sequence of moves that are all but uniquely defined because the alternative moves lead to obvious dead ends. You may also take advantage of several visualizations of the puzzle on the Internet.

89. **Counter Exchange** Identify a puzzle closely related to this one and use the algorithm for the former to devise an algorithm for the latter.

90. **Seating Rearrangements** There is a simple algorithm for generating permutations by adjacent element exchanges.

91. **Horizontal and Vertical Dominoes** The nontrivial part of the question is to prove that if n is even but not divisible by 4, no domino tiling of an $n \times n$ board with an equal number of horizontal and vertical tiles is possible. This can be done by using an invariant.

92. **Trapezoid Tiling** The obvious necessary condition for a tiling existence is also sufficient here.

93. **Hitting a Battleship** Mark the minimum number of the board's cells so that any 4×1 rectangle there would contain at least one marked cell.

94. **Searching a Sorted Table** The answer is "yes."

95. **Max-Min Weights** Consider the $n = 4$ instance for a crucial insight.

96. **Tiling a Staircase Region** The obvious necessary condition for a tiling existence is not quite sufficient here.

97. **The Game of Topswops** The answer is "yes": the game always stops after a finite number of iterations.

98. **Palindrome Counting** It is easier to count first the number of ways to spell CAT I SAW.

99. **Reversal of Sort** Solving the puzzle for a few small values of n should point you in the right direction.

100. **A Knight's Reach** Identify a shape of the region containing the squares in question. Also note that the answer can be given by the same formula for any $n > 2$.

101. **Room Painting** The answer is "yes."

102. **The Monkey and the Coconuts** While there are several ingenious methods for solving the puzzle, it can be solved in a rather straightforward fashion by setting up some equations and finding their smallest positive integer solutions.

103. **Jumping to the Other Side** The answer is "no."

104. **Pile Splitting** Considering a few small instances of the puzzle should help.

105. **The MU Puzzle** The answer is "no."

106. **Turning on a Light Bulb** Thinking about the switches as bits of a bit string could be helpful but not necessary.

107. **The Fox and the Hare** The fox can catch the hare for one-half of the possible values of s.

108. **The Longest Route** You may want to use a greedy strategy as a rough guide to a solution.

109. **Double-*n* Dominoes** The first two questions can be answered by straightforward summations. A ring can be constructed for a half of n's possible values either recursively or via reduction to a well-known graph problem.

110. **The Chameleons** Investigate changes in the differences between the chameleon counts after a meeting of two chameleons.

111. **Inverting a Coin Triangle** Find the best way to invert the triangle by making the kth coin row $(1 \leq k \leq n)$ the base of the inverted triangle and then determine the optimal value of k.

112. **Domino Tiling Revisited** The answer to the question is easy; the proof of its correctness is less so because the missing squares can be anywhere on the board.

113. **Coin Removal** Solving a few small instances of the puzzle should lead you to the right general strategy to employ.

114. **Crossing Dots** Solve the puzzle for $n = 3$ and then generalize the solution. Note that the only restriction on lines is that they must be straight.

115. **Bachet's Weights** For both versions of the puzzle, the solutions are not difficult to guess by applying the greedy approach. It is proving the solutions' optimality that is at the heart of this puzzle.

116. **Bye Counting** The answer to the first question can be obtained by solving a simple equation; the answer to the second can be extracted from the answer to the first.

117. **One-Dimensional Solitaire** Not counting the symmetric solutions, the empty cell can be in one of two board locations; each of the two may lead to two final locations of the remaining peg.

118. **Six Knights** See the tutorial on design strategies where the simpler *Guarini's Puzzle* is discussed.

119. **Colored Tromino Tiling** You may use the strategy employed in the first tutorial to tile the same boards with uncolored trominoes.

120. **Penny Distribution Machine** Number the boxes left to right starting with a 0 and represent final distributions of pennies by bit strings.

121. **Super-Egg Testing** Consider the function $H(k)$, the maximum number of floors for which the problem can be solved in k drops.

122. **Parliament Pacification** Start by partitioning the parliamentarians into two chambers in an arbitrary way and find a way to advance the chamber configuration toward the desired state.

123. **Dutch National Flag Problem** You may want to solve first the *Polish National Flag Problem* (#23), which deals with checkers of just two colors.

124. **Chain Cutting** Start by reversing the question and find the maximum length of the chain for which the problem can be solved by removing k clips. For $k = 1$, the maximum length of the chain is seven clips.

125. **Sorting 5 in 7** Although the puzzle can be solved in seven weighings, each comparing the weights of two items, no general sorting algorithm provides a right approach for achieving this goal.

126. **Dividing a Cake Fairly** The case of $n = 2$ has a simple but ingenious solution that can be extended to the general case.

127. **The Knight's Tour** There are a very large number of different solutions to this problem, which may all be assumed to start at the board's corner. You can find one by making moves that put the knight as close to the board's edge as possible.

128. **Security Switches** Solving a few small instances of the problem definitely helps, but for the general case you may want to apply the decrease-and-conquer strategy.

129. **Reve's Puzzle** Use the approach similar to the one utilized to solve the *Tower of Hanoi* puzzle (see, e.g., the tutorial on algorithm analysis techniques in this book).

130. **Poisoned Wine** The answer is "yes" to both questions; the task is to devise algorithms efficient enough to achieve the goal under the constraints given. In fact, you don't need the entire 5 weeks.

131. **Tait's Counter Puzzle** You will need to solve several instances of the puzzle before a useful pattern can be ascertained. In particular, the solution for $n = 3$ can be especially misleading, whereas the solution for $n = 4$ does contain important components of the solutions to larger instances. The general instance can be solved by a decrease-and-conquer algorithm, but the reduction to a smaller instance is far from obvious here.

132. **The Solitaire Army** Take advantage of the instances of the puzzle you already know how to solve.

133. **The Game of Life** The minimum numbers of live cells for still lifes, oscillators, and spaceships are 4, 3, and 5, respectively.

134. **Point Coloring** Apply the decrease-and-conquer strategy.

135. **Different Pairings** One can solve the problem either by using a $2 \times n$ table or by placing equidistant points on a circumference.

136. **Catching a Spy** Solve a simpler problem first: suppose you know that the spy starts at location 0 but you can only ask questions starting at time 1.

137. **Jumping into Pairs II** Thinking backward can help both to answer the question posed and design an algorithm required.

138. **Candy Sharing** Observe what happens with the largest number and the smallest number of candy pieces a child can have.

139. **King Arthur's Round Table** Use the iterative improvement strategy with the number of enemy pairs seated next to each other as the monovariant.

140. **The *n*-Queens Problem Revisited** Consider separately six cases of different remainders of the division of n by 6. The cases of $n \bmod 6 = 2$ and $n \bmod 6 = 3$ are harder than the others and require an adjustment of a greedy placement of the queens.

141. **The Josephus Problem** Set up two recurrences for the survivor's position $J(n)$: one for even n's and the other for odd n's.

142. **Twelve Coins** This difficult problem can be solved in three weighings, which is the minimum needed.

143. **Infected Chessboard** The answer is n. Prove that this number is both sufficient and necessary for infecting the entire board.

144. **Killing Squares** Tiling the board with dominoes in a certain way may help to find a solution. The minimum number of toothpicks that need to be removed from a 4×4 board is 9.

145. **The Fifteen Puzzle** Find an invariant that implies that the task is impossible for the given configuration.

146. **Hitting a Moving Target** Such an algorithm does exist. Assuming the hiding spots are numbered 1 to n, consider first the case when the target is at some even-numbered spot.

147. **Hats with Numbers** Use the sum of the hat numbers each mathematician observes.

148. **One Coin for Freedom** The game can be won by using a standard computing operation on bit strings.

149. **Pebble Spreading** The game's objective can be achieved only for $n = 1$ and $n = 2$. Find an invariant that implies that it is impossible to do for any $n > 2$.

150. **Bulgarian Solitaire** First, show that the algorithm creates a loop of coin partitions in which the piles are formed in nonincreasing order of their size. Then, trace trajectories of each coin in such a loop to show that the loop may contain just one partition for a triangular number of coins.

4 Solutions

The authors of the quotes in the epigraph puzzle in their given order are William Poundstone, George Pólya, Martin Gardner, Carl Friedrich Gauss, and Fibonacci.

1. A Wolf, a Goat, and a Cabbage

Solution Let M, w, g, and c stand for the man (with the boat, which is always on the same bank of the river as the man), wolf, goat, and cabbage head, respectively. Figure 4.1 depicts two sequences of trips that solve the problem.

Comments Most puzzles do not lend themselves to such simple solutions. This one is a rare exception in that the man has just one meaningful choice on all but the third trip The puzzle can also be solved by using the state-space graph (see [Lev06, Section 6.6]), similar to the solution of the *Two Jealous Husbands* puzzle in the tutorial on general design strategies. The states of the puzzle can also be represented by vertices of a cube (see, e.g., [Ste09, p. 256]). These alternative representations make it obvious that the seven trips is the fewest possible here.

This classic puzzle was included in Alcuin's collection—the earliest known collection of mathematical problems in Latin—which we already had a chance to mention in the first tutorial. On the appearance of this puzzle in other parts of the world, see [Ash90]. In modern times, it has become a standard feature in puzzle collections (e.g., [Bal87, p. 118]; [Kor72, Problem 11]). Surprisingly, the puzzle still attracts the attention of mathematicians and computer scientists (see [Cso08]).

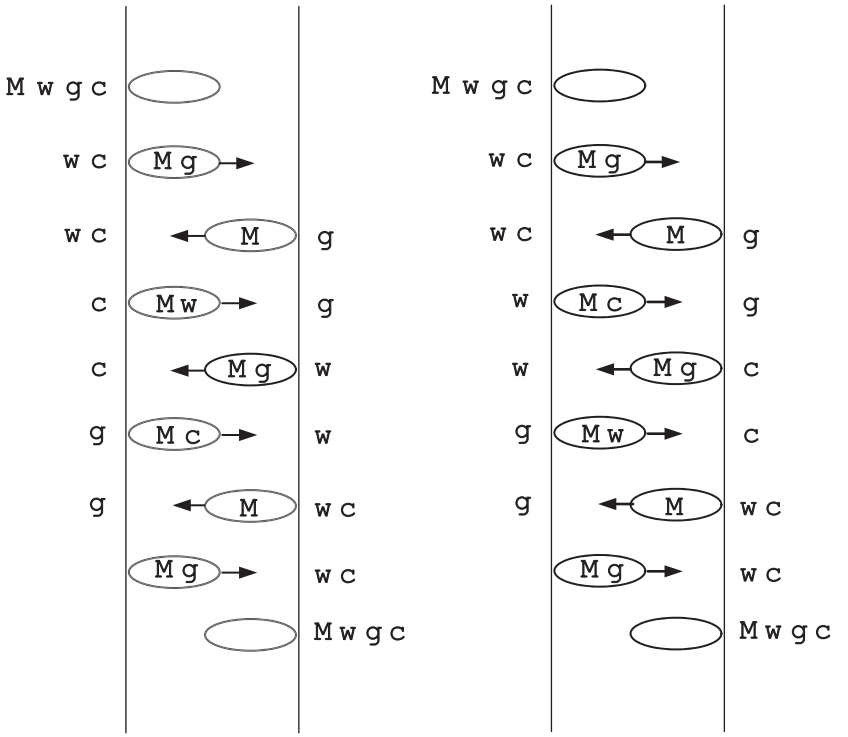


FIGURE 4.1 Two solutions to the *Wolf-Goat-Cabbage* puzzle.

2. Glove Selection

Solution The answers are 11 and 19 gloves for parts (a) and (b), respectively.

a. In the worst case, before you get at least one matching pair, you will select 5 black gloves, 3 brown gloves, and 2 gray gloves—all for the same hand. The next glove will have to yield a matching pair. Thus, the answer is 11 gloves.

b. In the worst case, before you get one matching pair of each color, you will select all 10 black, all 6 brown, and 2 gray gloves for the same hand. The next gray glove will have to yield a matching pair. Thus, the answer is 19 gloves.

Comments The puzzle provides a simple example of the worst-case analysis of an algorithm efficiency.

In most puzzle books, a version of this problem is stated for balls of different colors (e.g., [Gar78, pp. 4–5]). The glove version, which adds an extra twist to it, was used in [Mos01, Problem 18].

3. Rectangle Dissection

Solution A rectangle can be dissected into n right triangles for any integer $n > 1$.

A dissection for $n = 2$ is obtained by cutting the rectangle along its diagonal (Figure 4.2a). If $n > 2$, one can make the first cut along the rectangle's diagonal

and follow it up with $n - 2$ cuts of any of the available right triangles into two right triangles. The dissection of a right triangle into two right triangles is obtained by cutting along the height onto its hypotenuse. An example is shown in Figure 4.2b.

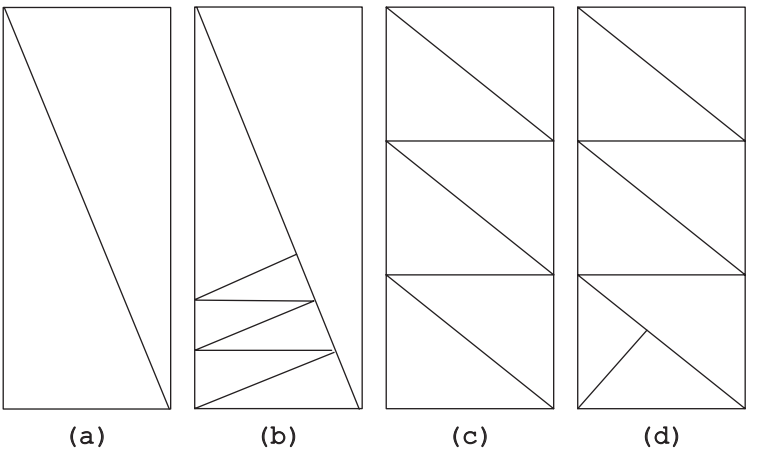


FIGURE 4.2 Rectangle dissection into right triangles. Results of the first method for (a) $n = 2$ and (b) $n = 7$. Results of the second method for (c) $n = 6$ and (d) $n = 7$.

We can also solve the puzzle by first considering the case of even n: cut the rectangle into $n/2$ smaller rectangles (e.g., by $n/2 - 1$ cuts parallel to the base of the rectangle) and then cut each of them along the rectangle's diagonal into two right triangles (Figure 4.2c). If n is odd, we first dissect the rectangle into $(n - 1)$ smaller triangles by the previous method and then cut any of the triangles along the height onto its hypotenuse (Figure 4.2d).

Comments The first solution is based on the incremental approach (decrease-by-one strategy applied bottom up). The second solution can be interpreted as a transform-and-conquer example: transferring the odd case to the simpler even case.

4. Ferrying Soldiers

Solution First, the two boys take the boat to the other side, after which one of them returns with the boat. Then a soldier takes the boat to the other side and stays there while the other boy returns the boat. These four trips reduce the problem size—measured by the number of soldiers to be ferried—by 1. Thus, if this four-trip procedure is repeated the total of 25 times, the problem will be solved after the total of 100 trips. (Of course, for the general instance of n soldiers, $4n$ trips will need to be made.)

Comments This easy puzzle provides a good illustration of the decrease (by one)-and-conquer strategy of algorithm design; this strategy is discussed in the book's first tutorial.

The puzzle is quite old and well-known. Henry E. Dudeney published it in the *Strand Magazine* in 1913 (see also [Dud67, Problem 450]); it was also included in a Russian puzzle collection [Ign78, Problem 43], which was first published in 1908.

5. Row and Column Exchanges

Solution The answer is "no."

Row exchanges preserve the numbers in rows, and column exchanges preserve the numbers in columns. This is not the case for the tables given (Figure 4.3): for example, 5 and 6 are in the same row in the initial table but in the different rows in the target table.

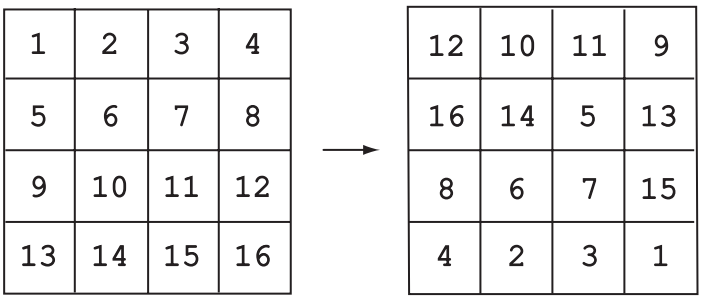


FIGURE 4.3 The initial and target tables for the *Row and Column Exchanges* puzzle.

Comments The puzzle provides a good example of an invariant that is different from the more common parity and coloring.

The puzzle is similar to Problem 713 in A. Spivak's collection [Spi02].

6. Predicting a Finger Count

Solution She will stop on her first finger.

Here is how the finger count starts:

finger	thumb	first	middle	ring	little	ring	middle	first
count	1	2	3	4	5	6	7	8
count	9	10	11	12	13	14	15	16
count	17	18	19	20	21	22	23	24
count	25	26	27	28	29	30	31	32

It is easy to see that the counting falls on the same finger every eighth number called. Therefore to answer the question, all one needs is to find the remainder of the division of 1000 by 8, which is equal to 0. This implies that when the girl reaches 1000, she will be on her first finger (moving from the middle finger), the same one she will be on while calling any number divisible by 8.

Comments The puzzle belongs to a rather rare type of algorithmic puzzles in which the object is to determine the output of a given algorithm (here, the finger count procedure) for a specified input (here, the number 1000).

The puzzle is from Martin Gardner's *Colossal Book of Short Puzzles and Problems* [Gar06, Problem 3.11]. A similar problem was included in Henry Dudeney's collection *536 Puzzles & Curious Problems* [Dud67, Problem 164].

7. Bridge Crossing at Night

Solution The sequence of moves solving the puzzle is shown in Figure 4.4.

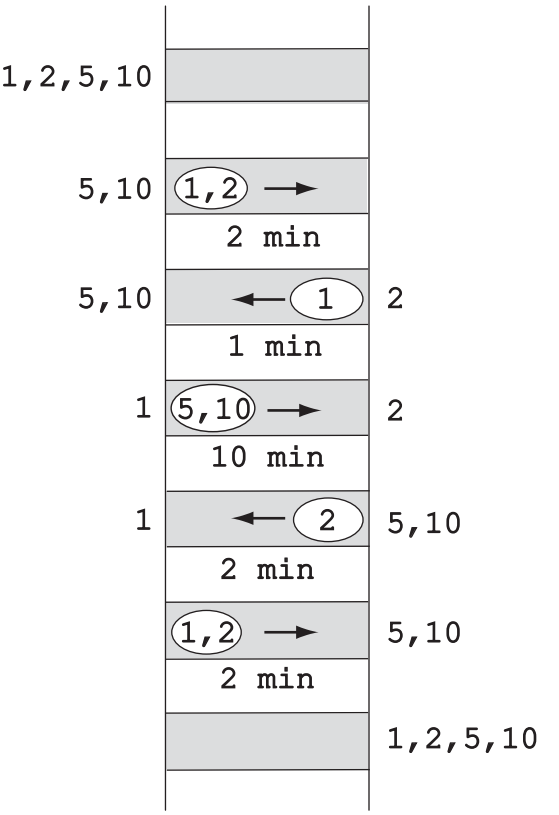


FIGURE 4.4 Solution to the *Bridge Crossing at Night* puzzle: labels 1, 2, 5, 10 represent the four people, respectively. The arrows indicate the crossing directions (always with the flashlight).

In the obvious alternative solution, person 2 returns the flashlight after the first trip to the other side while person 1 returns it after the second trip.

Actually, 17 minutes is the minimum amount of time needed here. It is obvious (and can be formally proved) that in an optimal solution two persons have to cross the bridge together and one person has to return the flashlight, if not, all the people are already on the other side. Thus three two-person trips and two one-person trips are needed for four people to get to the other side in a minimum amount of time. If the flashlight is returned by the fastest person on both back trips, the fastest person

will have to participate in each pair going to the other side, for the total time of $(10 + 1) + (5 + 1) + 2 = 19$ minutes. If one of the two back trips is not done by the fastest person, then the return crossing times will be at least $2 + 1 = 3$ minutes, and trips to the other side will be at least $10 + 2 + 2 = 14$ minutes because at least one pair will have to include the slowest person and hence take 10 minutes to cross the bridge whereas the other two pairs will take at least 2 minutes each. Hence, the total crossing time will have to take at least 17 minutes.

Comments As discussed in the first tutorial, the puzzle cannot be solved correctly by the straightforward application of the greedy approach. This might be one of the reasons many people find it harder than it appears at the first glance.

This puzzle, also known as the *Bridge and Torch Problem*, became a hot topic on the Internet some years ago; it has also been included in William Poundstone's book as one of the Microsoft interview puzzles [Pou03, p. 86]. Torsten Sillke's web page [Sillke] contains some interesting material related to this puzzle, including the earliest reference to the problem in the book by Levmore and Cook [Lev81] and an algorithm for the general instance of the problem, in which n people need to cross a bridge under the same constraints as described above and arbitrary individual crossing times. The proof of this algorithm's optimality was published by Günter Rote in 2002 [Rot02]. For further extensions, see Moshe Sniedovich's website [Sni02] and the paper by Roland Backhouse [Bac08].

8. Jigsaw Puzzle Assembly

Solution The answer is 499 moves.

Any move decreases the number of remaining sections by 1. Therefore, after k moves, the total number of remaining sections will be $500 - k$ irrespective of an order in which the sections are assembled. Hence 499 moves will be made before the entire puzzle is assembled.

Comments The solution is based on the same invariant idea that underlines the better known *Breaking a Chocolate Bar* puzzle (see the second tutorial).

Submitted by Leo Moser, the puzzle was published in the January 1953 issue of *Mathematics Magazine* (p. 169); it was later included in [Ave00, Problem 9.22].

9. Mental Arithmetic

Solution The sum is equal to 1000.

The object is to compute (in one's head) the sum of the numbers in the table shown in Figure 4.5.

The first method is based on the observation that the sum of any two numbers in the squares symmetric with respect to the diagonal connecting the lower left and upper right corners is equal to 20: $1 + 19$, $2 + 18$, $2 + 18$, and so on. So, since there are $(10 \cdot 10 - 10)/2 = 45$ such pairs (we subtracted the number of the squares on that diagonal from the total number of the squares), the sum of the

1	2	3			⋯			9	10
2	3						9	10	11
3						9	10	11	
					9	10	11		
				9	10	11			
⋮			9	10	11				⋮
		9	10	11					
	9	10	11						17
9	10	11						17	18
10	11				⋯		17	18	19

FIGURE 4.5 Table of numbers to be summed up in the *Mental Arithmetic* puzzle.

numbers outside that diagonal is equal to $20 \cdot 45 = 900$. With $10 \cdot 10 = 100$ on the diagonal, the total sum is equal to $900 + 100 = 1000$.

The second method computes the sum row by row (or column by column). The sum in the first row, as discussed in the second tutorial, is equal to $10 \cdot 11/2 = 55$. The sum of the numbers in second row is $55 + 10$ since each of the numbers is larger by 1 than their counterparts in the row above. The same is true for all the other rows as well. Hence the total sum is equal to $55 + (55 + 10) + (55 + 20) + \cdots + (55 + 90) = 55 \cdot 10 + (10 + 20 + \cdots + 90) = 55 \cdot 10 + 10 \cdot (1 + 2 + \cdots + 9) = 55 \cdot 10 + 10 \cdot 45 = 1000$.

Comments The first method uses the same trick Carl Gauss presumably used to find the sum of the first hundred integers, as described in the tutorial on algorithm analysis techniques. We also mentioned there that the formula itself is extremely useful in algorithm analysis. We used this formula twice in the second solution to the problem along with reducing the sums to the simpler one.

The problem is similar to Question 1.33 in the Wall Street interview question book [Cra07].

10. A Fake Among Eight Coins

Solution The answer is two weighings.

Select from the given coins two groups of three coins each and put them on the opposite cups of the scale. If they weigh the same, the fake is among the other two coins, and weighing these two coins will identify the lighter fake. If the first weighing does not yield a balance, the lighter fake is among the three lighter coins. Take any two of them and put them on the opposite cups of the scale. If they weigh the same, it is the third coin in the lighter group that is fake; if they do not weigh

the same, the lighter one is the fake. Since the problem cannot be solved in one weighing, the above algorithm requiring just two weighings is optimal.

Comments Since $8 = 2^3$ and halving the problem size does usually yield extremely efficient algorithms, it is quite understandable why many people solve the problem in three weighings instead of two. This, however, is a rare example of a situation when a decrease by a factor larger than 2 is possible. The puzzle also highlights a dilemma of dealing with problems stated with specific numerical data. Sometimes it is possible to take advantage of some special property of the data given while in others (as here) one can be misled by it.

The problem has an alternative solution in which the second weighing does not depend on the results of the first one. Label the coins by the letters A, B, C, D, E, F, G, H. On the first weighing, weigh A, B, C against F, G, H. On the second weighing, weigh A, D, F against C, E, H. If ABC = FGH (the first weighing results in a balance), all these six coins are genuine, and therefore the second weighing is equivalent to weighing D against E. If ABC < FGH, only A, B, and C may still be fake. Therefore if on the second weighing ADF = CEH, B is the fake; if ADF < CEH, A is the fake; and if ADF > CEH, C is the fake. The case of ABC > FGH is symmetric to the case just discussed.

The puzzle has an obvious generalization to an arbitrary number of coins, although the optimality proof of the division-into-thirds algorithm is usually done by using more advanced techniques such as *decision trees* (e.g., [Lev06, Section 11.2]).

According to T. H. O'Beirne [Obe65, p. 20], the puzzle dates from the First World War. Nowadays, it is often offered during job interviews in the United States. For a much harder coin weighing problem, see the *Twelve Coins* puzzle (#142).

11. A Stack of Fake Coins

Solution The puzzle can be solved in one weighing.

Number the coin stacks from 1 to 10. Take 1 coin from the first stack, 2 coins from the second, and so on, until all 10 coins are taken from the last stack. Weigh all these coins together. The difference between this weight and 550, the weight of $(1 + 2 + \cdots + 10) = 55$ genuine coins, indicates the number of the fake coins weighted, which is equal to the number of the stack with the fake coins. For example, if the selected coins weigh 553 grams, 3 coins are fake and hence it is the third stack that contains the fake coins.

Comments The solution is based on the representation change idea.

The puzzle was included, among others, in the first collection of Martin Gardner's columns in *Scientific American* [Gar88a, p. 26] and Averbach and Chein's *Problem Solving Through Recreational Mathematics* [Ave00, Problem 9.11].

12. Questionable Tiling

Solution A requested tiling does not exist.

This can be proved by contradiction. Assume that such a tiling does exist. Because of the board's symmetry, we may assume that its upper left square is covered by a horizontal domino numbered 1 in Figure 4.6. Then the square in the first column of the second row from the top must be covered by a vertical domino, and therefore the square in the second column of that row must be covered by a horizontal domino. By applying the same logic further, we end up with a tiling shown in Figure 4.6. Placing a domino horizontally below tile 13, which is the only option available, contradicts the assumption that no two dominoes form a 2×2 square.

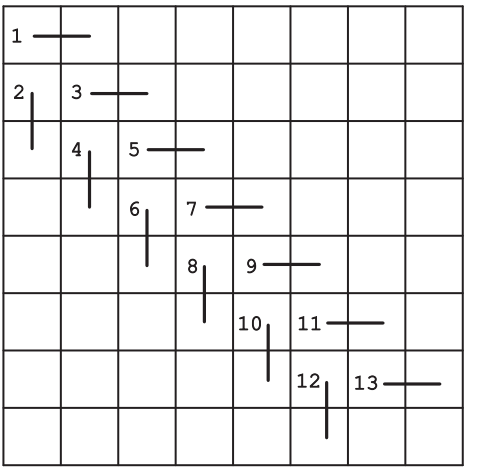


FIGURE 4.6 Solution to the *Questionable Tiling* puzzle.

Comments This is a rather rare example of a nonexistence proof that is not based on some version of the invariant idea discussed in the second tutorial.

The puzzle is Problem 102 in [Fom96, p. 74].

13. Blocked Paths

Solution The answer is 17 paths.

The easiest way to get it is to apply dynamic programming—one of the algorithm design strategies discussed in the first tutorial. This approach finds the number of shortest paths from A to every intersection in the grid outside the fenced-off area (see Figure 4.7). Starting with the assignment of 1 to intersection A, these numbers can be computed row by row and left to right within each row. If an intersection has both the left and upper neighbors, its number is computed as the sum of the neighboring numbers; if an intersection has just one of such neighbors, it gets the same number as that of the neighbor.

Comments A similar problem was discussed in the book's tutorial on algorithm design strategies. Counting paths is a well-known applications of dynamic

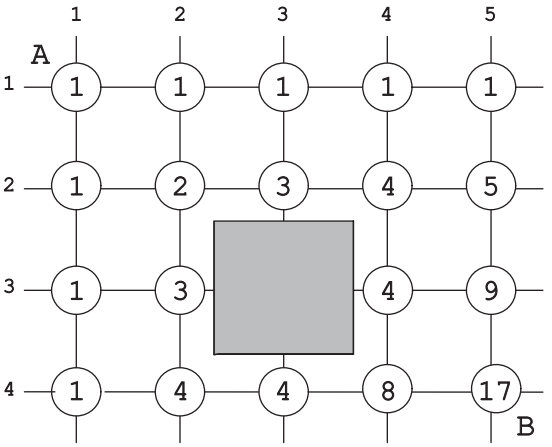


FIGURE 4.7 Counting shortest paths from A to B with a fenced-off area shown in gray.

programming (see, e.g., [Gar78, pp. 9–11]). Other applications of dynamic programming are usually less straightforward.

14. Chessboard Reassembly

Solution The answer is 25 pieces.

Since the standard chessboard has no 2×1 or 1×2 part colored the same color, each 4×4 region of the board given has to be cut both horizontally and vertically. Four horizontal and four vertical cuts shown in Figure 4.8 dissect the board into 25 pieces, which is the minimum: four 1×1 squares, twelve 1×2 rectangles, and nine 2×2 squares each colored as in the standard chessboard. The standard chessboard can be assembled from the pieces obtained in a variety of ways. For example, one can simply rotate eight 1×2 rectangles at the board's border by 180 degrees and rotate four 2×2 squares by 90 degrees.

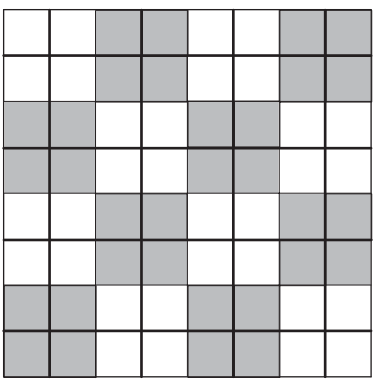


FIGURE 4.8 Optimal cutting of the board to reassemble into a chessboard.

Comments The puzzle is from Serhiy Grabarchuk's *The New Puzzle Classics* [Gra05, p. 31].

15. Tromino Tilings

Solution The answer is "no" for parts (a) and (b) and "yes" for part (c).

a. The answer is "no" because a 3×3 board cannot be tiled with right trominoes. Indeed, a corner of the board, say, the lower left one, can be tiled in three different ways, and each of them leaves a room only for one more right tromino (Figure 4.9).

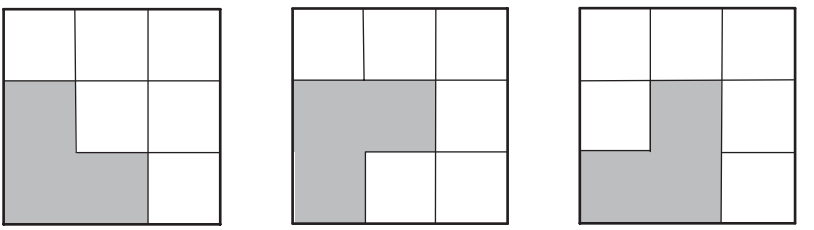


FIGURE 4.9 Three ways to start tiling a 3×3 board by placing a right tromino covering the lower left corner.

b. The answer is "no" because the total number of squares in any $5^n \times 5^n$ board is not divisible by 3.

c. A desired tiling can be easily obtained by dividing the board into 2×3 rectangles to tile each of them with two trominoes (see Figure 4.10 for an example).

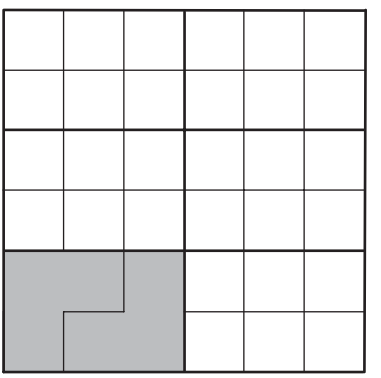


FIGURE 4.10 Tiling a 6×6 board with trominoes.

Comments The answer to the first question is obtained by the exhaustive inspection of its smallest instance. (The smallest instance is an exception here: all $3^n \times 3^n$ boards where $n > 1$ can be tiled with trominoes [Mar96, p. 31].) The answer to the second question is based on the invariant idea. Finally, the solution to the third problem can be considered a divide-and-conquer application.

The puzzle is similar to Problem 50 in Ian Parberry's *Problems on Algorithms* [Par95].

16. Making Pancakes

Solution The minimum amount of time needed is n minutes for every $n > 1$ and 2 minutes for $n = 1$.

If n is even, the solution is obvious: for every pair of pancakes, fry them both simultaneously, first on one side and then on the other.

If $n = 1$, then 2 minutes are needed to fry the pancake on both sides. If $n = 3$, one can do the job in 3 minutes as follows. First, fry pancakes 1 and 2 on one side. Then fry pancake 1 on the second side together with pancake 3 on its first side. Finally, fry both pancakes 2 and 3 on the second side. If n is odd and greater than 3, an optimal algorithm can fry the first three pancakes as just described and then fry the remaining $n - 3$ pancakes, which is even, as indicated above.

For every $n > 1$, the above algorithm requires n minutes to do the job. This is the minimum time possible because n pancakes have $2n$ sides to be fried and any algorithm can fry no more than two sides in 1 minute.

Comments The above algorithm can be considered a decrease-by-two algorithm; but the key to the puzzle is, of course, the optimal way frying of three pancakes.

The earliest reference to this puzzle in David Singmaster's bibliography [Sin10, Section 5.W], is dated 1943, although he says that it is probably older than that. Since then, it was included in a variety of puzzle books (e.g., [Gar61, p. 96]; [Bos07, p. 9, Problem 38]).

17. A King's Reach

Solution a. The answer is $(2n + 1)^2$ for $n > 1$ and 8 for $n = 1$.

In one move, the king can reach any of the eight squares adjacent to the starting one. After two moves, it can be at any of the following: the starting square (by first leaving and then returning there), any of the 8 squares adjacent to it (by moving first to a neighboring square and then reaching the target square), and any of the 16 squares forming the middle square shown by connected points in Figure 4.11a. Thus, all the squares reachable in two moves are either on the perimeter of this square or within it. In general, after $n > 1$ moves, the king can reach those and only those squares that are within or on the perimeter of the $(2n + 1) \times (2n + 1)$ square with the center at the starting square (see Figure 4.11a for $n = 3$). The number of such squares is equal to $(2n + 1)^2$. If $n = 1$, the king can only reach eight squares adjacent to its starting square but, unlike the case of $n > 1$, it cannot return to the starting square.

b. The answer is $(n + 1)^2$ squares.

If the king moves only horizontally or vertically, after n moves it will always be on a square of the same or opposite color to the color of its starting square for an even and odd n, respectively. Consider the farthest squares the king can reach in n moves. These squares form a border: all the squares on this border and all

the squares of the same color within it are reachable by the king in n moves (see Figure 4.11b for an illustration.) There are $(n+1)^2$ of such squares.

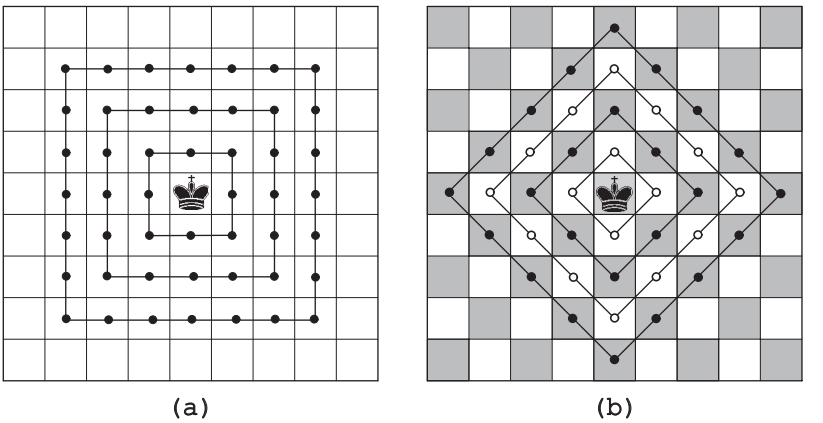


FIGURE 4.11 (a) Reachable squares in three standard king moves (plus the starting square). (b) Reachable squares in three horizontal and vertical moves (white circles) and in four horizontal and vertical moves (black circles plus the starting square).

Comments The correctness of the solutions can be proved more rigorously by mathematical induction. The same question for the chess knight is the subject of the *Knight's Reach* puzzle (#100) later in the book.

18. A Corner-to-Corner Journey

Solution The journey in question is impossible.

The squares where the knight starts and ends its move are always of the opposite color. To visit all the squares of the board once, it would need to make 63 moves; since this number is odd, such a journey will have to start and end on squares of the opposite color. But the squares of the lower left and upper right corners of the board are colored the same, making the journey in question impossible.

Comments The puzzle is a standard exercise exploiting square coloring as the invariant idea. Note that the problem of finding a knight's tour through all the squares of a standard 8×8 chessboard, known as the *Knight's Tour* problem (#127), does have solutions if the tour is not required to start and end at diagonally opposite corners of the board.

19. Page Numbering

Solution The answer is 562 pages.

Let $D(n)$ be the total number of decimal digits in the first n positive integers (book pages). The first nine numbers are one-digit, therefore $D(n) = n$ for $1 \leq n \leq 9$. The next 90 numbers from 10 to 99, inclusive, are two-digits. Hence,

$$D(n) = 9 + 2(n - 9) \text{ for } 10 \leq n \leq 99.$$

The maximal value of $D(n)$ for this range is $D(99) = 189$, which means that some three-digit numbers are needed to reach the total digit count of 1578 given in the puzzle. There are 900 three-digit decimals, which leads to the formula

$$D(n) = 189 + 3(n - 99) \text{ for } 100 \leq n \leq 999.$$

To answer the puzzle's question, we need to solve the equation

$$189 + 3(n - 99) = 1578.$$

Its solution is $n = 562$.

Comments The puzzle is included in the book as an example of a simple algorithm analysis.

Similar questions have been a common feature in elementary recreational mathematics books.

20. Maximum Sum Descent

Solution Using the standard dynamic programming technique, discussed in the first tutorial, compute the maximum sum along a descending path from the apex to each number in the triangle. Start with the apex, for which this sum is obviously equal to the number itself. Then compute the sums moving top down and, say, left to right across the triangle's rows as follows. For any number that is either the first or the last in its row, add the sum previously computed for the adjacent number in the preceding row and the number itself; for any number that is neither the first nor the last in its row, add the larger of the previously computed sums for the two adjacent numbers in the preceding row and the number itself. When all such sums are computed for the numbers at the base of the triangle, find the largest among them.

Figure 4.12 illustrates the algorithm for the triangle given in the problem's statement.

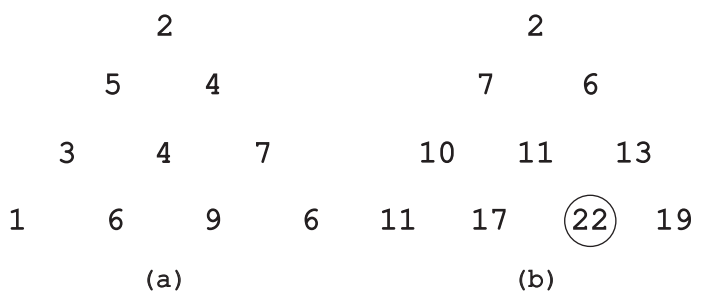


FIGURE 4.12 Illustration of the dynamic programming algorithm for the *Maximum Sum Descent* puzzle. (a) Input triangle. (b) Triangle of maximum sums along descending paths with 22 being the largest.

Comments The puzzle is from the Project Euler website [ProjEuler].

21. Square Dissection

Solution A square can be dissected into n smaller squares for every $n > 1$ except for $n = 2$, 3, and 5.

Considering the fact that the four right angles of the given square must be in smaller squares, it is obvious that the problem has no solution for these three values of n. It does have one obvious solution for $n = 4$, shown in Figure 4.13a. This solution can be generalized to any even $n = 2k$ by having $2k - 1$ equal squares along two adjacent sides of the given square, with the side length of each smaller square equal $1/k$th of the side length of the given square. Figure 4.13b illustrates this solution for $n = 6$.

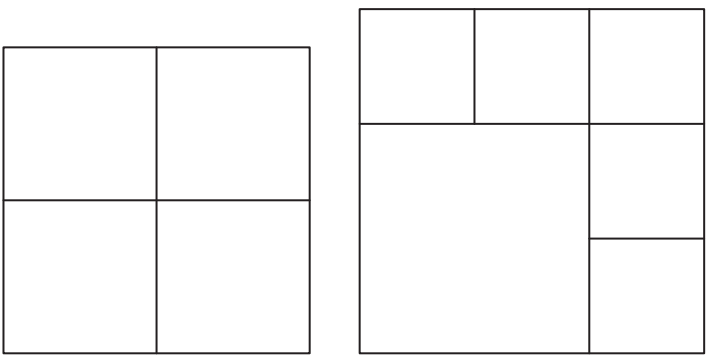


FIGURE 4.13 Square dissection into (a) four squares, and (b) six squares.

If $n > 5$ and odd, that is, $n = 2k + 1$ where $k > 2$, then $n = 2(k - 1) + 3$, and we can first dissect the given square into $2(k - 1)$ squares as described above and then dissect any of the obtained squares (e.g., the one in the top left corner) into four smaller ones, which will increase the total number of the obtained squares by 3. This solution is illustrated in Figure 4.14 for $n = 9$.

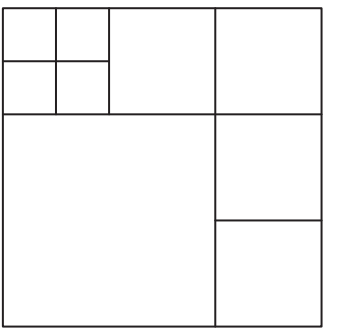


FIGURE 4.14 Square dissection into nine squares.

Comments Here, we considered the cases of even and odd n's separately and solved the more difficult of the two (the odd case) by transforming it to the easier one (the even case).

The puzzle has been included in several books (e.g., [Sch04, pp. 9–11]). The related problem of dissecting a square into squares of different sizes is much harder, of course—see [Ste04, Chapter 13], for a good exposition of its history and pertinent results.

22. Team Ordering

Solution The following recursive algorithm solves the puzzle. If $n = 1$, the problem is solved. If $n > 1$, solve the problem recursively for an arbitrarily chosen group of $n - 1$ teams. Then scan the obtained list of these $n - 1$ teams to insert the team not included in the group right before the first team on the list that lost to it in the tournament. If there is no such team—that is, if the team not yet on the list lost all its games to the teams on the list—insert that team at the end of the list.

Comments The algorithm is a perfect illustration of the decrease-and-conquer strategy. It can also be implemented bottom up (incrementally) by starting with some ordering of the teams, initializing a list with the first team, and then successively inserting teams $2, 3, \ldots, n$ into the list right before the first team on the list that lost to it in the tournament. If there is no such team—that is, if the team not yet on the list lost all its games to the teams on the list—insert that team at the end of the list.

As far as the puzzle's origins are concerned, it has been known for a long time. For example, its version for chess tournaments (a chess game can end with a tie) was mentioned in E. Gik's book [Gik76, p. 179].

23. Polish National Flag Problem

Solution Here is one of the algorithms to solve the puzzle. Find the leftmost white checker and the rightmost red checker. If the leftmost white checker is to the right of the rightmost red checker, the problem is solved; if not, swap the two and repeat the operation.

Figure 4.15 illustrates the algorithm.

Comments The above algorithm is similar to the principal part of *quicksort*, one of the most important sorting algorithms (e.g., [Lev06, Section 4.2]). It can be considered a decrease-and-conquer algorithm in which an instance size decrease can vary from one iteration to another.

The puzzle is a simplified version of the *Dutch National Flag Problem* puzzle (#123) given later in the book.

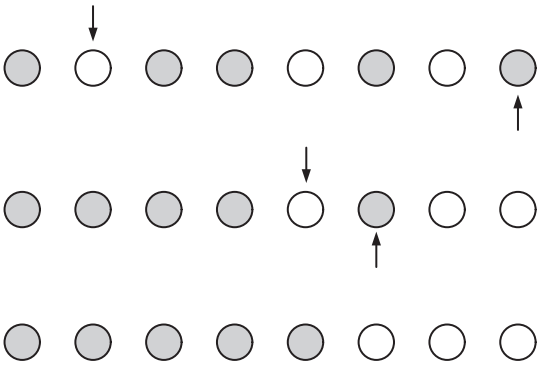


FIGURE 4.15 Illustration of the algorithm for the *Polish National Flag Problem*.

24. Chessboard Colorings

Solution a. For the knight, the minimum number of colors for $n > 2$ is two: more than one color is obviously needed, and the standard coloring of the board in two opposite colors provides a coloring required by the question. It is one for $n = 2$: no two knights threaten each other on such a small board.

b. Since the bishop threatens all the squares on the same diagonal and no other squares, at least n colors are needed to color the main diagonal of the board from its upper left corner to the lower right corner. The easiest way to extend this coloring to the entire board is to color all the squares in the same column the same color as the square on the main diagonal. Thus, the answer for the bishop is n.

c. Since the king threatens only every square adjacent to it horizontally, vertically, or diagonally, at least four colors are needed to color each 2×2 region of the board. Dividing the board into such disjoint regions (some of which may degenerate to smaller rectangles that can be thought of as parts 4×4 regions with some squares outside the board) and coloring each 2×2 region with four colors using the same coloring scheme implies that the answer for the king is four.

d. Since the rook only threatens all the squares in the same row or column, at least n colors are needed to color every line (row or column). This number is not only necessary but also sufficient. The easiest way to get a coloring in n colors so that no two squares in the same line are colored the same, is to color, say, the first row in n colors and then shift the coloring scheme one column to the right with wrapping the colors that fall outside the board to color the squares in the leftmost columns. An example for $n = 5$ is given in Figure 4.16.

Comments The straightforward solutions given above for the knight, bishop, and king can be interpreted as based on the greedy strategy. As to the rook's coloring, it yields a *Latin square* of order n: an $n \times n$ table filled with n different symbols in such a way that each symbol occurs exactly once in each row and exactly once in each column. And while the questions about the minimal coloring for the knight, bishop, and king are easy, it is not the case for the queen (see [Iye66]).

1	2	3	4	5
5	1	2	3	4
4	5	1	2	3
3	4	5	1	2
2	3	4	5	1

FIGURE 4.16 Coloring a 5×5 board in five colors with no two squares in the same row or column colored the same color.

25. The Best Time To Be Alive

Solution In general terms, the problem can be stated as follows. Given n intervals $(b_1, d_1), \ldots, (b_n, d_n)$ —where in our case b_i, d_i are the dates of birth and death of the ith person in the index $(1 \leq i \leq n)$—find an interval that is an intersection of the largest number of the intervals given. All the intervals are open, that is, do not include their end points. If $d_i = b_j$, the closing parenthesis of the ith interval precedes the opening parenthesis of the jth interval. If several intervals have the same opening parenthesis, it should be counted for each of these intervals; of course, the same is true for a coinciding closing parenthesis.

It is helpful to visualize the intervals depicted on the real line, as it is done in Figure 4.17. The sequence of the interval parenthesis holds the key to an efficient solution to the problem. It is not difficult to see that scanning the sequence left to right while increasing the parenthesis count by 1 on every opening parenthesis and decreasing it by 1 on every closing one solves the problem: the count reaches its maximum on the left parenthesis of the desired interval, whose right parenthesis is simply the next one.

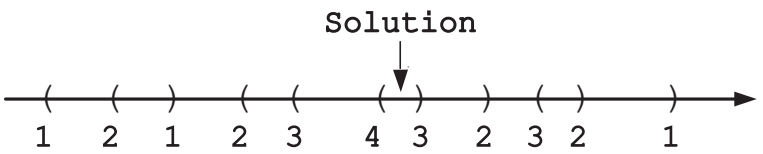


FIGURE 4.17 Illustration of the algorithm for the *Best Time To Be Alive* puzzle.

Comments Representing the input data as intervals on the real line is an example of the representation change variety of transform-and-conquer—one of the strategies outlined in the first tutorial.

26. Find the Rank

Solution The answer is 598.

The total number of words made up of the six letters is equal to $6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 720$ (see the first tutorial). The "words" following TURING in the alphabetically ordered list are either of the form U $*****$ or of the form TURN $**$, where a $*$ stands for one of the six letters not already in the "word." Since these letters can be in any order, there are $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$ and $2! = 2 \cdot 1$ of them, respectively (see the first tutorial). Thus, the total number of "words" following TURING in the alphabetically ordered list is equal to $5! + 2! = 120 + 2 = 122$. This means that TURING will be in position $720 - 122 = 598$ in the list, if its elements are numbered from 1 to 720.

Comments The puzzle is an instance of a well-known problem called *permutation ranking*. For a decrease-by-one algorithm for this problem, see, for example, [Kre99, pp. 54–55].

27. The Icosian Game

Solution The puzzle has 30 solutions, one of which is shown in Figure 4.18.

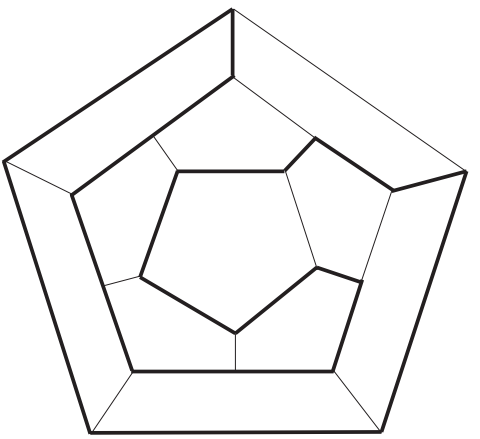


FIGURE 4.18 One of the solutions to the *Icosian Game*.

Comments The puzzle presents a special case of one of the most intriguing problems dealing with graphs. The problem in question is the existence of a *Hamilton circuit*—a sequence of adjacent (connected by an edge) vertices that starts at some vertex, visits each of the other vertices exactly once, and then returns to the starting vertex. Some graphs have Hamilton circuits—for example, the graph of the Icosian Game—some do not. No efficient algorithm to determine the existence of a Hamilton circuit in an arbitrary graph is known. In fact, most computer scientists believe that such an algorithm does not exists. Despite a more than a 50-year long search for a proof of this conjecture, and a $1-million prize

established in year 2000 for resolving the issue one way or another, it remains unresolved.

28. Figure Tracing

Solution The analysis of *The Königsberg Bridges Problem* in the second tutorial implies that a figure can be traced without lifting pen off the paper or going back over any line in it if and only if the multigraph of the figure is connected and satisfies one of the two conditions:

- All the vertices of the multigraph have even degrees (i.e., an even number of edges for which the vertex is an endpoint)—then a trace can start at any of its vertices where it will end.
- Exactly two of its vertices have odd degrees—then a trace must start at one of these odd vertices and end at the other.

a. The first figure can be traced with the restrictions imposed: its graph (Figure 4.19a) is connected and all its vertices have even degrees.

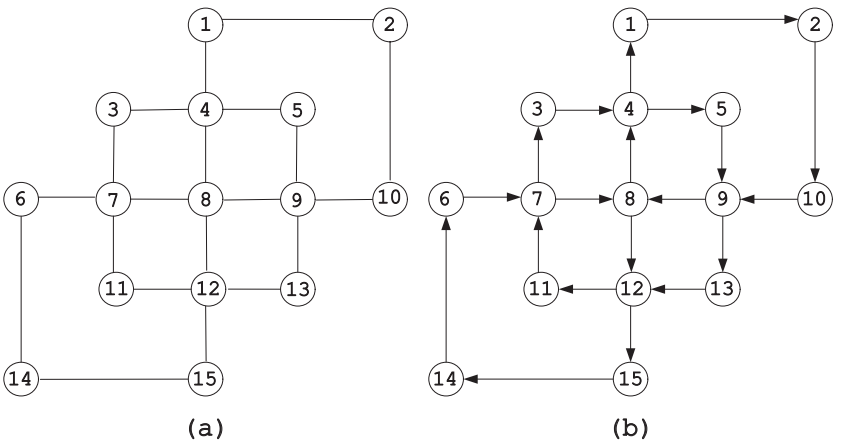


FIGURE 4.19 (a) Graph of the figure to trace. (b) Its Euler circuit.

There is a well-known algorithm for constructing an Euler circuit. It starts at an arbitrary chosen vertex and proceeds along previously untraversed edges until either all of them are traversed or the path returns to the starting vertex with no untraversed edge out of it available while some of the graph's edges remain untraversed. In the latter case, the obtained circuit is removed from the graph and the same operation is repeated recursively starting at a vertex that is in both the remaining part of the graph and the removed circuit. (The existence of such a vertex follows from the graph's connectivity and the fact that all its vertices have even degrees.) Once an Euler circuit is constructed for the remaining part of the graph, it is "spliced" into the first circuit to yield an Euler circuit for the entire graph.

For example, starting at vertex 1 of the graph in Figure 4.19a and following its "outside" edges, we get the circuit

$$1 - 2 - 10 - 9 - 13 - 12 - 15 - 14 - 6 - 7 - 3 - 4 - 1.$$

Picking, say, vertex 4 as a common vertex with the remaining part of the graph, we will get the following Euler circuit for the remaining part of the graph:

$$4 - 5 - 9 - 8 - 12 - 11 - 7 - 8 - 4.$$

"Splicing" the latter circuit into the former yields the following Euler circuit for the entire graph (Figure 4.19b):

$$1 - 2 - 10 - 9 - 13 - 12 - 15 - 14 - 6 - 7 - 3 - 4 - 5 - 9 -$$
$$8 - 12 - 11 - 7 - 8 - 4 - 1.$$

b. The second figure can be traced with the restrictions imposed: considered as a graph (see Figure 4.20a), it is connected and all its vertices have even degrees except two: vertices 3 and 8. Starting at vertex 3 and using essentially the same algorithm, we can get the following path:

$$3 - 4 - 7 - 11 - 10 - 9 - 6 - 2 - 3 - 7 - 10 - 6 - 3 - 10.$$

Then picking, say, vertex 2 as a common vertex with the remaining part of the graph, we get the following Euler circuit for the remaining part of the graph:

$$2 - 1 - 4 - 8 - 11 - 12 - 9 - 5 - 2.$$

"Splicing" the latter circuit into the former path yields the following Euler path for the entire graph (Figure 4.20b):

$$3 - 4 - 7 - 11 - 10 - 9 - 6 - 2 - 1 - 4 - 8 - 11 - 12$$
$$- 9 - 5 - 2 - 3 - 7 - 10 - 6 - 3 - 10.$$

c. It is impossible to trace the third figure because its graph has more than two vertices of an odd degree.

Comments Since the size of Euler circuits constructed by the above algorithm varies unpredictably, the algorithm falls in the decrease-by-variable-number category.

Figure tracing is a standard problem in puzzle books. This application of Euler's theorem goes back to Peter G. Tait (1831–1901), a prominent Scottish mathematician and physicist [Pet09, p. 232].

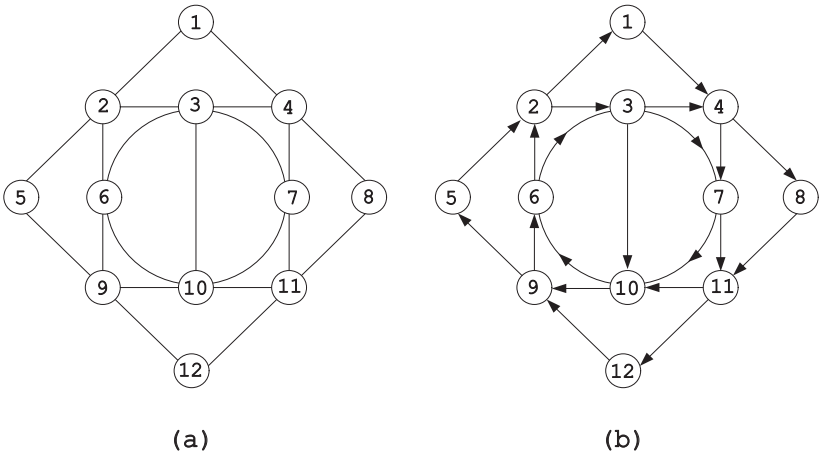


FIGURE 4.20 (a) Graph of the figure to trace. (b) Its Euler path.

29. Magic Square Revisited

Solution The first helpful step is to find the value of the common sum for the magic squares in question. This sum, called sometimes the *magic sum*, is equal to the sum of all the numbers in its rows divided by the number of rows: $(1 + 2 + \cdots + 9)/3 = 15$. Second, the central cell must contain 5. Indeed, denoting the numbers in the row 1, 2, and 3 a, b, c; d, e, f; g, h, i, respectively, and adding the numbers in the second row, the second column, and the two main diagonals, we obtain

$$\begin{aligned}&(d+e+f)+(b+e+h)+(a+e+i)+(g+e+c)\\&=3e+(a+b+c)+(d+e+f)+(g+h+i)=3e+3\cdot 15=4\cdot 15,\end{aligned}$$

which implies that $e = 5$. What remains is to arrange pairs $(1, 9)$, $(2, 8)$, $(3, 7)$, and $(4, 6)$ around it.

Taking into account the table's symmetries, there are only two qualitatively different ways to put 1 and hence 9: in the table's corners and not in the table's corners (see Figure 4.21).

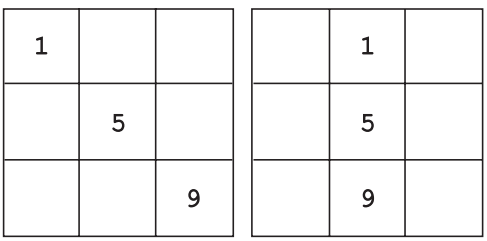


FIGURE 4.21 Two possible placements of 1 and 9 in a 3×3 magic square construction.

But the first of these arrangements cannot be completed to form a magic square: if we put a number less than 5 in the upper right corner, we will not be able to have the magic sum of 15 in the first row, and if we put there a number larger than 5, we will have the same problem with the last column.

Thus, we can abandon the first partially filled table in Figure 4.21 and concentrate on the second one. There are also three other ways to put 1 and 9 in the same row or the same column with 5. These symmetric alternatives are shown in Figure 4.22.

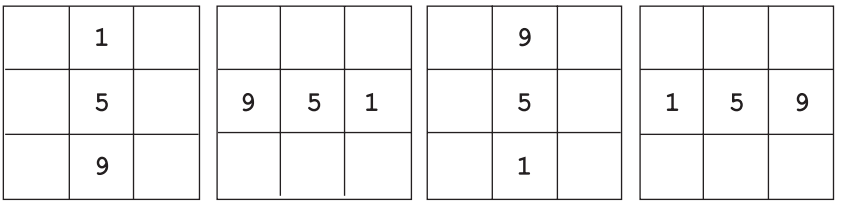


FIGURE 4.22 Four possible positions of 1, 5, and 9 in a 3×3 magic square.

The line (row or column) containing 1 must then be filled with 6 and 8, which can be done in two ways. The numbers for the remaining cells are determined uniquely afterward. All eight magic squares of order 3 are shown in Figure 4.23. Of course, all of them are symmetric and can be obtained from one of the eight by rotation and reflection.

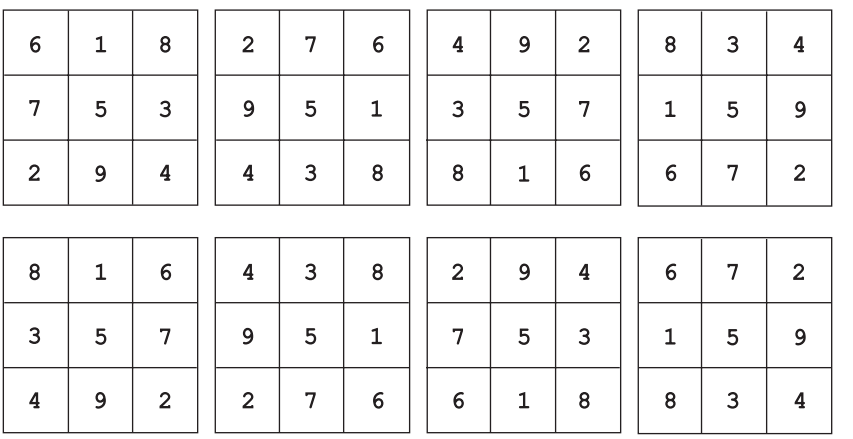


FIGURE 4.23 Eight magic squares of order 3.

Comments Magic squares have fascinated people for thousands of years since their appearance in ancient China. While several algorithms for construction of magic squares of order $n > 2$ have been devised, no formula for the number of magic squares of an arbitrary order has been discovered. For more information on magic squares, the reader can check several monographs (e.g., [Pic02]), chapters in many recreational mathematics books (e.g., [Kra53, Chapter 7]), and numerous sites on the World Wide Web devoted to them.

30. Cutting a Stick

Solution The minimum number of cuts for a 100-unit stick is seven.

Since cutting several pieces of a given stick at the same time is allowed, we need to concern ourselves only with finding a cutting algorithm that reduces the size of the longest piece present to size 1. This implies that on each iteration an optimal algorithm must cut the longest piece—and simultaneously all the other pieces whose size is greater than 1—by half (or as close to this as possible). That is, if the length l of a piece is even, it is cut into two pieces of length $l/2$; if l is odd and greater than 1, it is cut into pieces of lengths $\lceil l/2 \rceil = (l+1)/2$ and $\lfloor l/2 \rfloor = (l-1)/2$, respectively. The iterations stop after the longest—and, hence, all the other pieces of the stick—has length 1.

The number of cuts (iterations) such an optimal algorithm makes for an n-unit stick is equal to $\lceil \log_2 n \rceil$, which is the least k such that $2^k \geq n$. In particular, for $n = 100$, $\lceil \log_2 100 \rceil = 7$, since $2^7 > 100$ and $2^6 < 100$.

Comments This is one of several puzzles exploiting optimality of the decrease-by-half strategy. This strategy was also used to solve the *Number Guessing* game in the first tutorial. For a two-dimensional version of the problem, see the *Cutting a Rectangular Board* puzzle (#54) in the main section.

31. The Three Pile Trick

Solution The card in question will always be exactly in the middle of the pile selected by the chooser in the final layout.

Let us denote cards in the three piles after the first deal as, say, a1, a2, ... a9; b1, b2, ..., b9; c1, c2, ..., c9 (Figure 4.24a). If the selected card is, to be specific, in pile 1, after the second deal the piles will look as in Figure 4.24b. Note that all the cards that were in pile 1 after the first deal are now exactly in the three middle positions in each of the piles. If the selected card is now in, say, pile 3—that is, it is either a3, or a6, or a9—it will be exactly in the middle of a pile in the final

Pile 1	Pile 2	Pile 3
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5
a6	b6	c6
a7	b7	c7
a8	b8	c8
a9	b9	c9

(a)

Pile 1	Pile 2	Pile 3
b1	b2	b3
b4	b5	b6
b7	b8	b9
a1	a2	a3
a4	a5	a6
a7	a8	a9
c1	c2	c3
c4	c5	c6
c7	c8	c9

(b)

Pile 1	Pile 2	Pile 3
b1	b4	b7
a1	a4	a7
c1	c4	c7
b3	b6	b9
a3	a6	a9
c3	c6	c9
b2	b5	b8
a2	a5	a8
c2	c5	c8

(c)

FIGURE 4.24 The *Three Pile Trick* illustration.

layout (Figure 4.24c) Pointing out the pile containing the selected card in the final layout uniquely identifies that card. It is easy to see that this would also be the case if a selected card was in piles other than 1 and 3 after the first and second deal, respectively.

Comments Many card tricks are based on some general ideas of algorithm design and analysis. This puzzle illustrates a situation in which a simple analysis of the output of a given algorithm solves the puzzle in question.

According to Ball and Coxeter [Bal87, p. 328], the trick was mentioned by Bachet in his 17th-century classic [Bac12, p. 143]. It was also included in *Mathematical Recreations* by Maurice Kraitchik [Kra53, p. 317].

32. Single-Elimination Tournament

Solution a. The total number of matches is equal to $n - 1$: each match yields one loser, and exactly $n - 1$ losers need to be produced to get a single winner of the tournament.

b. If $n = 2^k$, the total number of rounds is equal to $k = \log_2 n$: each round reduces the number of remaining players by half, and the rounds continue until the number of remaining player is reduced to 1. If n is not necessarily a power of 2, the answer is the smallest power of 2 that is greater than or equal to n, that is, using the standard notation for rounding up to the nearest integer, $\lceil \log_2 n \rceil$. For example, for $n = 10$, the number of rounds is equal to $\lceil \log_2 10 \rceil = 4$.

c. The second-best player can be any player who lost to the winner and nobody else. These players can be made play their own single-elimination tournament that can be organized as follows. In the tree representing all the played matches, find a leaf representing the tournament's winner and follow the path from this leaf up to the root assuming that the winner lost the first match. This will require no more than $\lceil \log_2 n \rceil - 1$ matches.

Comments The tournament can be considered an algorithm, and the first two questions ask about the number of steps—understood as either individual matches or rounds of play—in it. The different interpretations of what the algorithm's step is leads to the different step counts, of course. It is also worth pointing out that tournament trees have some interesting applications in computer science (see [Knu98]).

Martin Gardner included a similar puzzle in his *aha!Insight* [Gar78, p. 6]. The *Bye Counting* puzzle (#116) in this book deals with byes, which are transfers of players directly to the next round because they have no opponent assigned to them.

33. Magic and Pseudo-Magic

Solution The answers are $n = 3$ and $n \geq 3$ for parts (a) and (b), respectively.

a. Since the central square in a 3×3 magic square must be filled with number 5 (see the solution to the *Magic Square Revisited* puzzle (#29)), the puzzle's objective can only be achieved for $n = 3$ by making any of the eight magic squares of order 3 given there.

b. For $n = 3$, the answer is obvious, because any 3×3 magic square is also a pseudo-magic square. If we fill a 3×3 square at the upper left corner of an $n \times n$ table where $n > 3$ with numbers 1 through 9 to have a magic square there and then copy the contents of the first column as the contents of column four, the 3×3 square formed by the first three rows and columns 2, 3, and 4 will be a pseudo-magic square. Similarly, if we copy the first row into the fourth, we will get a pseudo-magic square in rows 2, 3, and 4 and the first three columns. This leads to the following algorithm for the task in question.

Fill the 3×3 square at the upper left corner with numbers forming any 3×3 magic square. Then fill the first three cells of columns 4, 5, ..., n with the corresponding entries at the first three cells at columns 1, 2, ..., $n - 3$, respectively. Then fill rows 4, 5, ..., n of the table with the contents of rows 1, 2, ..., $n - 3$, respectively. An example for $n = 5$ is shown in Figure 4.25.

Alternatively, the algorithm can be phrased as tiling the table given with the same 3×3 magic square, ignoring the parts of some squares, if any, that fall outside the table.

4	9	2	4	9
3	5	7	3	5
8	1	6	8	1
4	9	2	4	9
8	1	6	8	1

FIGURE 4.25 Solution to the *Magic and Pseudo-Magic* puzzle for $n = 5$.

Comments The algorithm's idea is based on the incremental approach (see the first tutorial) starting with the smallest instance of $n = 3$.

34. Coins on a Star

Solution The largest number of coins that can be placed is seven.

With no loss of generality, we can start by placing the first coin on point 6 and then moving it to point 1, denoted $6 \rightarrow 1$ (Figure 4.26).

This will make both lines connecting point 1 with points 4 and 6 unusable for another coin placement. Following the logic of the greedy strategy, we should

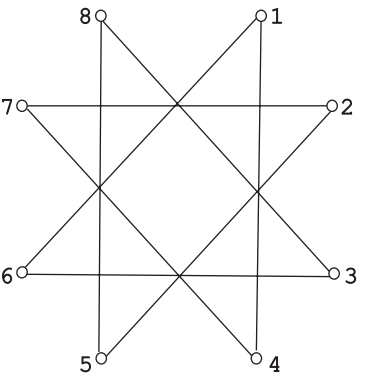


FIGURE 4.26 The star for placing coins at its vertices.

try placing each coin in a manner that minimizes the number of unusable lines and hence maximizes the number of usable lines for the coins still to be placed. This implies that for each coin after the first one, we should try placing it on an unoccupied point of an unusable line by moving it there along a usable line, of course. The easiest way to do this is to always move the next coin so that it ends at the point from which the previously coin has been placed. For example, seven coins can be placed by the following sequence of moves:

$$6 \to 1,\ 3 \to 6,\ 8 \to 3, 5 \to 8,\ 2 \to 5,\ 7 \to 2, 4 \to 7.$$

Obviously, we cannot place eight coins, because after placing seven coins there will be no unoccupied point from which the eighth coin could have been moved.

Alternatively, we can solve the puzzle by "unfolding" the graph by the "buttons and strings" method. (This method was mentioned in the first tutorial's discussion of the representation change strategy.) Lifting vertex 2 of the graph in Figure 4.26 and carrying it over to the left side of the graph and lifting vertex 6 and carrying it over to the right side of the graph yields the graph depicted in Figure 4.27a. Then lifting vertices 8 and 4 and carrying them on the opposite sides of that graph yields the graph in Figure 4.27b. The solution mentioned above—as well as several alternative solutions—follow immediately from this representation of the puzzle's star.

Comments The two solutions given above take advantage of the greedy strategy and a particular method of graph representation change, respectively.

Different versions of this puzzle, also known as the *Octogram Puzzle*, have been known for centuries (see [Sin10, Section 5.R.6]). In modern times, it has been included in such puzzle books as [Dud58, p. 230], [Sch68, p. 15], and [Gar78, p. 38]. It is closely related to *Guarini's Puzzle*, discussed in the tutorial on algorithm design strategies.

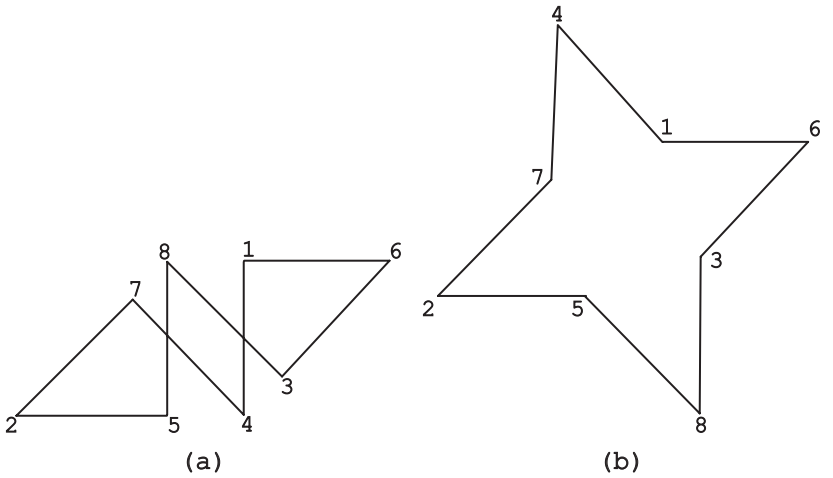


FIGURE 4.27 Unfolding the graph of Figure 4.26.

35. Three Jugs

Solution The sequence shown in Figure 4.28 solves the puzzle in six steps:

Step#	8-pint jug	5-pint jug	3-pint jug
	8	0	0
1	3	5	0
2	3	2	3
3	6	2	0
4	6	0	2
5	1	5	2
6	1	4	3

FIGURE 4.28 Solution to the *Three Jugs* puzzle.

Although the solution can be obtained by trial and error, there is a systematic way of getting to it. We can represent a state of the jars by a triple of nonnegative integers indicating the amount of water in the 3-pint, 5-pint, and 8-pint jugs, respectively. Thus, we start with the triple 008. We will consider all legal transformations from a current state of the jugs to new possible states. To do this, we will take advantage of the queue—one of the basic data structures in computing.

A *queue* is a list of items, which, as its name implies, operates like a queue of customers to a single cashier: the customers are served in the order they arrive.

Items are deleted from one end of the queue, called the *front,* and new items are added to the other end, called the *rear*.

In our application, we initialize a queue with the given state triple 008 and repeat the following steps until a desired state—a triple containing a 4—is encountered for the first time. For the state at the front of the queue, label all the *new* states reachable from it, add them to the queue, and then delete the front state from the queue. After the desired state is reached for the first time, follow the labels backward to get the shortest sequence of transformations that solve the puzzle.

The application of this algorithm to the puzzle's data yields the following sequence of the queue contents, where the subscripts show the state labels when they are assigned for the first time:

$$008|305_{008}, 053_{008}|053, 035_{305}, 350_{305}|035, 350, 323_{053}|350, 323, 332_{035}|$$

$$323, 332|332, 026_{323}|026, 152_{332}|152, 206_{026}|206, 107_{152}|107, 251_{206}|$$

$$251, 017_{107}|017, 341_{251}$$

Tracing the labels from that of 341 backward, we get the following transformation sequence that solves the puzzle in the minimum number of six steps:

$$008 \rightarrow 053 \rightarrow 323 \rightarrow 026 \rightarrow 206 \rightarrow 251 \rightarrow 341.$$

Comments The solution mimics the so-called *breadth-first search* traversal (e.g., [Lev06, Section 5.2]) of the puzzle's state-space graph. We did not draw this graph explicitly for the sake of simplicity. The algorithm clearly has an exhaustive search flavor.

Some interesting tidbits about this very old puzzle, its variations and later developments, can be found in two MAA online columns: one by Alex Bogomolny [Bog00], which contains a link to a visualization applet, and the other by Ivar Peterson [Pet03]. The puzzle can also be solved by a surprising representation in the trilinear coordinates discovered by M. C. K. Tweedie [Twe39] (see also [OBe65, Chapter 4]).

36. Limited Diversity

Solution The puzzle has a solution if n is even, and it does not have a solution if n is odd.

If n is even, one can fill the top row with pluses, rows 2 and 3 with minuses, rows 4 and 5 with pluses again, and so on, until the last row is filled with pluses. Under this scheme, every cell will have exactly one neighbor with the opposite sign either above or below the cell in question. Of course, as alternative solutions, one can interchange the pluses and minuses in the previous solution or have every column, rather than every row, contain the same sign.

If we put a plus in the upper left corner, we have to put a plus and a minus in the neighboring cells. We will consider the case when a plus is put in its row neighbor and a minus in its column neighbor; the other case is symmetric. Thus, let us prove

that if we put pluses in the first two cells of row 1 and a minus in the first cell of row 2, we will have to fill the remaining cells of row 1 with pluses and the remaining cells of row 2 with minuses. Since the first cell in the second row already has a neighbor with a plus above it, the second cell in row 2 will have to contain a minus. But then the third cell in row 1 will have to contain a plus since the second cell in row 1 already has a neighbor with a minus below it. By the same argument, all the remaining cells of row 1 will have to contain a plus, and all the remaining cells in row 2 will have to contain a minus. But then all the cells in row 3 must contain minuses because their neighbors in row 2 already have the plus neighbors in row 1. This implies that row 3 cannot be last but must be followed by row 4 filled with all pluses. Row 4 can be either last or must be followed with the all plus row, and so on. (More formally, the same arguments can be made via mathematical induction.) This proves that the puzzle has no other solutions than those described above when n is even, and that it does not have a solution when n is odd.

Comments The puzzle is motivated by its instance for $n = 4$, which appeared in A. Spivak's collection [Spi02, Problem 67b].

37. $2n$-Counters Problem

Solution Since $2n$ counters need to be placed into n rows and n columns of the board with at most two counters in the same row or in the same column, exactly two counters have to be placed in each row and column.

For even $n = 2k$, a solution can be obtained by identical placement of n counters in the first k columns and the last k columns as follows. (We assume that rows and columns of the board are numbered top to bottom and left to right, respectively.) Place two counters in the first two rows of columns 1 and $k + 1$, two counters in rows 3 and 4 of columns 2 and $k + 2$, and and so on, until finally counters are placed in rows $n - 1$ and n of columns k and $2k$ (see Figure 4.29a for the case of $n = 8$).

For odd $n = 2k + 1$, $k > 0$, a solution can be obtained by placing two counters in rows 1 and 2 of column 1, two counters in rows 3 and 4 of column 2, and so on until counters are placed in rows $n - 2$ and $n - 1$ of column k. Then two counters are placed in the first and last rows of column $k + 1$. After that, k counters are placed in the right part of the board symmetrically with respect to the board's central square to those in the left part: two counters are placed in rows 2 and 3 of column $k + 2$, in rows 4 and 5 of column $k + 3$, and so on, until rows $n - 1$ and n of the last column (see Figure 4.29b for the case of $n = 7$).

For $n \geq 4$, the problem can also be solved by interposing two solutions to the *n-Queens Problem* (#140) that do not have a queen on the same square of the board. This is hardly an advisable way to solve the *2n-Counters Problem*, however, because it is, in fact, easier than the *n-Queens Problem*.

Comments Sam Loyd [Loy60, Problem 48] considered the puzzle for an 8×8 board with the additional stipulation that two counters must be placed on two

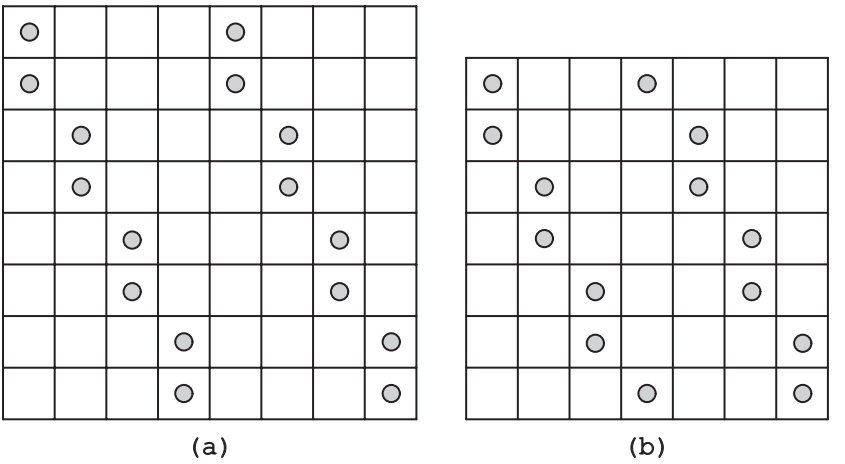


FIGURE 4.29 Solutions to 2*n-Counters Problem* for (a) $n = 8$ and (b) $n = 7$.

central squares of the board. Henry Dudeney [Dud58, Problem 317] also imposed a more stringent requirement that no three pieces shall be in any straight line, that is, not limited to rows, columns, and diagonals. While both authors gave the same solution for an 8×8 board, Dudeney's version remains unsolved for boards of an arbitrary size. For a further discussion of the problem, see Chapter 5 in Martin Gardner's *Penrose Tiles to Trapdoor Chiphers* [Gar97a].

38. Tetromino Tiling

Solution Tilings of an 8×8 chessboard with straight tetrominoes, square tetrominoes, L-tetrominoes, and T-tetrominoes are shown in Figure 4.30. Note that in each case the tiling of one quarter of the board is repeated for the three other quarters.

It is impossible to cover an 8×8 chessboard with Z-tetrominoes. Indeed, putting such a tile to cover a corner of the board makes it necessary to continue putting two more tiles along the boarder with no possibility to cover the two remaining squares in the first row (Figure 4.30e).

Finally, it is impossible to tile an 8×8 chessboard with 15 T-tetrominoes and 1 square tetromino. The number of, say, dark squares covered by one T-tetromino—and hence by any odd number of them—is odd, whereas a single square tetromino always covers two darks squares. Hence 15 T-tetrominoes and 1 square tetromino always cover an odd number of dark squares while the number of such squares on an 8×8 chessboard is even.

Comments One can consider the tilings with straight tetrominoes, square tetrominoes, L-tetrominoes, and T-tetrominoes as based either on brute force or the divide-and-conquer strategy. The impossibility proof of tiling with 15 T-tetrominoes and 1 square tetromino is clearly based on the invariant (parity and coloring) idea.

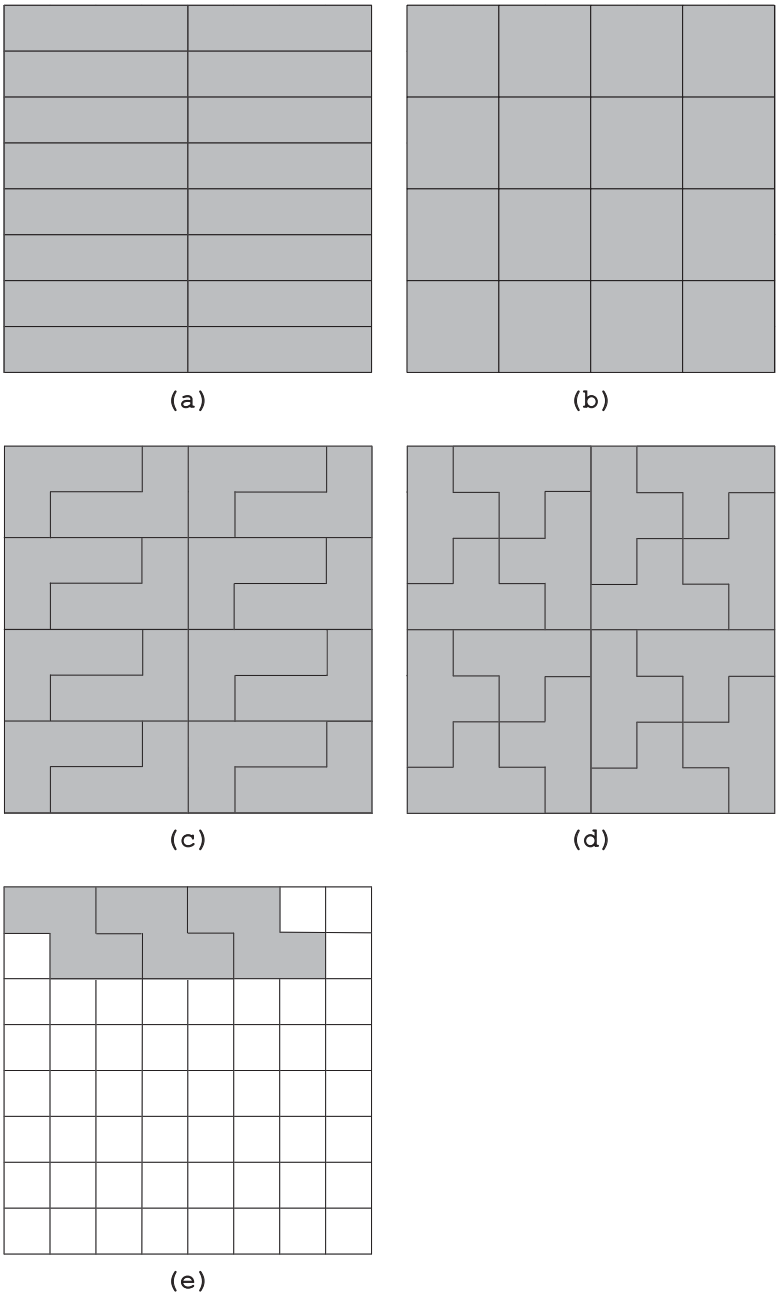


FIGURE 4.30 Tiling a chessboard with tetrominoes. (a) Straight tetrominoes. (b) Square tetrominoes. (c) L-tetrominoes. (d) T-tetrominoes. (e) Z-tetrominoes (failed).

This puzzle is from the seminal paper on polyomino tiling by Solomon Golomb [Gol54].

39. Board Walks

Solution No path through all the squares of the board in Figure 2.11a is possible. If one colors the board's squares alternatingly as a chessboard (Figure 4.31a), it will have six more dark squares than light squares. Since the colors of squares a path goes through must alternate, such a path through all the squares of the board in Figure 4.31a is impossible.

Since the number of light squares on the board of Figure 2.11b is more by 1 than the number of dark squares, a path through all the squares must start and end at light squares. One of such paths, which exploits the board's symmetry, is shown in Figure 4.31b.

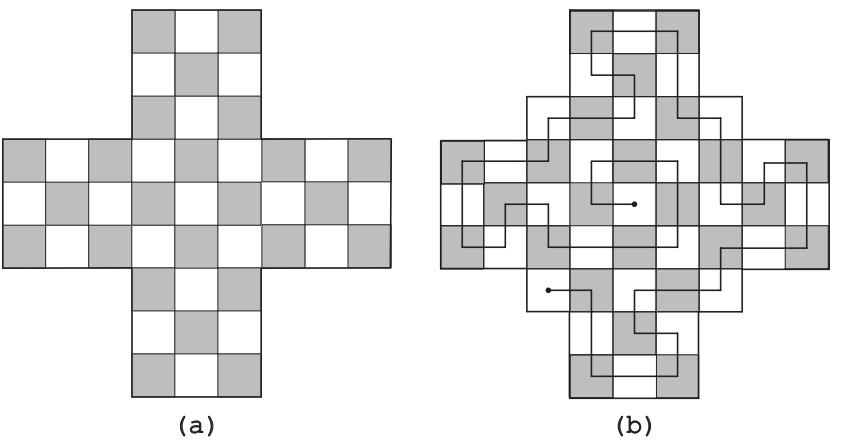


FIGURE 4.31 (a) Coloring of the first board given. (b) Coloring of the second board with a path through all its squares.

Comments The solutions take advantage of coloring—one of the most frequently used ways to exploit the invariant idea (see the tutorial on algorithm analysis techniques).

Note that the problem asks for an existence of a *Hamilton path* in a graph whose vertices represent the board's squares and edges "connect" adjacent squares. Unlike a Hamilton circuit (see the comments to *The Icosian Game* (#27)), a Hamilton path is not required to return to its starting point. The absence of this requirement does not make the problem easier, however: no efficient algorithm to determine the existence of a Hamilton path in an arbitrary graph is known either.

Part (b) is based on Problem 459 in A. Spivak's collection [Spi02].

40. Four Alternating Knights

Solution The puzzle has no solution.

As explained in the tutorial on algorithm design strategies, the initial state of the puzzle can be conveniently represented by a graph in Figure 4.32.

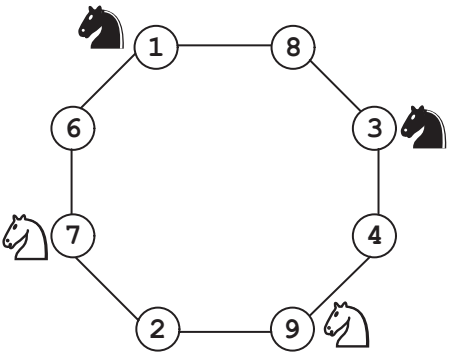


FIGURE 4.32 Initial state of the *Four Alternating Knights* puzzle represented by the unfolded graph.

The knights can move only to adjacent vertices of that graph preserving their clockwise (and counterclockwise) order: the two knights of one color followed by the two knights of the other color. Since the objective configuration of the puzzle requires the four knights of different colors alternate, the puzzle cannot be solved.

Comments The puzzle's solution exploits two themes in algorithmic problem solving: representation change (the board's graph and its unfolding) and invariant (the knights' clockwise order).

This variation of *Guarini's Puzzle*, discussed in the first tutorial, is from [Fom96, Problem 2, p. 39]. A similar variation of the classic puzzle is mentioned by M. Gardner in *aha!Insight* [Gar78, p. 36].

41. The Circle of Lights

Solution The minimum number of switches that need to be flipped is $n/3$ if n is divisible by 3 and n if n is not divisible by 3.

It is not difficult to realize that the final state of the lights depends only on the parity (odd or even) of the number of times each switch is flipped and does not depend on the order in which the switches are manipulated. Therefore all that needs to be determined is which of the switches should be flipped once and which should be left in the off position to have all the lights on. To turn on a light, we need either flip its switch while leaving the two adjacent switches in their initial positions or flip all these three switches. Clearly, at least one switch needs to be flipped. Let us number the lights and their corresponding switches from 1 to n—say, clockwise—starting with that switch/light. Then to have light 1 to remain

on in the final state, its adjacent switches (numbered 2 and n) must be either both flipped or both not flipped. In the former case, switch 3 and switch $n-1$ must also be flipped; continuing this process we come to the situation in which all the switches must be flipped once.

In the latter case—when switch 1 is flipped and switches 2 and n are not—switch 3 must not be flipped to leave light 2 on; switch 4 must be flipped to turn on light 3 as well as lights 4 and 5. Continuation of this process will have every third switch flipped, that is, the switches numbered $1, 4, \ldots, 3k+1, \ldots, n-2$, which is possible if and only if n is a multiple of 3. For such n's, this alternative flips only $n/3$ switches, which is smaller than flipping all the n switches—the minimum needed for all other values of n.

Comments Although we were able to solve the puzzle by essentially brute-force thinking, its more general version—the one requiring a transformation from any given state of lights to some designated state—would require a more sophisticated approach.

The puzzle was offered, as the November 2004 problem of the month, on *Math Central*—an Internet service for students and teachers of mathematics, which is maintained at the University of Regina in Regina, Saskatchewan, Canada [MathCentral]. The site mentions that a demo of the puzzle for $n = 7$ was exhibited in the mathematics museum in Giessen, Germany. Two-dimensional versions of the puzzle—*Merlin's Magic Squares* and *Lights Out*—are, in fact, better known than this one-dimensional version.

42. The Other Wolf-Goat-Cabbage Puzzle

Solution Let W, G, C, and H represent a wolf, a goat, a cabbage, and a hunter, respectively. Then the puzzle has two symmetric solutions:

$$\text{WCWC}\ldots\ \text{WCHGHG}\ldots\ \text{HG and GHGH}\ldots\ \text{GHCWCW}\ldots\ \text{CW}.$$

The key observation here is that a W may only be next to a C, and a G may only be next to an H. For $n = 1$, this immediately implies that the puzzle has two symmetric solutions: WCHG and GHCW.

The solutions for $n = 2$ can be obtained by appending WC and GH to the front and rear of the solutions for $n = 1$ to get WCWCHGHG and GHGHCWCW. In general, appending WC and GH $n-1$ times to the front and rear of the solutions for $n = 1$ yields the two solutions for any value of n, which are given above.

The fact that there is no other solution follows from the following argument. As it was the case for $n = 1$, any solution must have a W-counter and a G-counter at the two ends of the counters' ordering. We can prove this by contradiction. Assume, to the contrary, that. there exists a solution for which this is not the case. Since the W's and G's have the symmetric restrictions imposed on them, we may assume without loss of generality that this solution has all its n W's inside

the solution's ordering. But then it would have to have $n + 1$ C's adjacent to them, which is impossible. Thus, any solution must have a W and a G at its two ends; the other $n - 1$ W's must be interleaved with n C's to form a sequence CWCW. . . C and the other $n - 1$ G's must be interleaved with n H's to form a sequence HGHG. . . H inside the solution's ordering. Finally, there is just one way to put CWCW. . . C and HGHG. . . H between the W and G and one way to put them between the G and W without violating the puzzle's constraints.

Comments The solution to the puzzle can be considered based on decrease-and-conquer applied bottom up: the solution is obtained by solving smaller instances of the same puzzle first and then extending their solutions to form the same patterns.

According to M. Kraitchik's *Mathematical Recreations* [Kra53, p. 214], the puzzle is due to Aubry, who considered its $n = 3$ instance.

43. Number Placement

Solution Start by sorting the list in increasing order. Then repeat the following $n - 1$ times: If the first inequality sign is "<," place the first (smallest) number in the first box; otherwise, place there the last (largest) number. After that, delete the number from the list and the box it was put in. Finally, when just a single number remains, place it in the remaining box.

Comments The above algorithm is based on two algorithm design strategies: transform-and-conquer (presorting) and decrease-and-conquer (decrease-by-one). Note that it does not give all possible solutions.

The problem was posted on a web page of *The Math Circle* [MathCircle].

44. Lighter or Heavier?

Solution The puzzle can be solved in two weighings.

Start by taking aside one coin if n is odd and two coins if n is even. After that, divide the remaining even number of coins into two equal-size groups and put them on the opposite pans of the scale. If they weigh the same, all these coins are genuine and the fake coin is among the coins set aside. So we can weigh the set-aside group of one or two coins against the same number of genuine coins: if the former weighs less, the fake coin is lighter; otherwise, it is heavier.

If the first weighing does not result in a balance, take the lighter group and, if the number of coins in it is odd, add to it one of the coins initially set aside (which must be genuine). Divide all these coins into two equal-size groups and weigh them. If they weigh the same, all these coins are genuine and therefore the fake coin is heavier; otherwise, they contain the fake, which is lighter. ■

Since the puzzle cannot, obviously, be solved in one weighing, the above algorithm solves it in the minimum possible number of weighings.

Comments The puzzle provides a very rare example of a problem that can be solved in the same number of basic steps (namely, two weighings) irrespective of how large the problem's instance (here, the number of coins) is. Another such example in the book is *A Stack of Fake Coins* (#11).

Instances of this puzzle are included in a puzzle book by Dick Hess [Hes09, Problem 72], and a Russian collection for middle school students [Bos07, p. 41, Problem 4].

45. A Knight's Shortest Path

Solution The minimum number of moves is 66.

Although the knight cannot move along the straight line toward its goal, it can stay on the main diagonal after every pair of its moves. Thus, if its start and finish squares are (1, 1) and (100, 100), respectively, a sequence of 66 moves such as

$$(1, 1) - (3, 2) - (4, 4) - \cdots - (97, 97) - (99, 98) - (100, 100)$$

solves the problem. (The number k of two-move advances can be obtained from the equation $1 + 3k = 100$.)

Given the nature of the knight's moves, it is convenient to measure the distance between two squares on the board by the so-called *Manhattan distance,* which is computed as the number of rows plus the number of columns between two squares in question. Here the Manhattan distance between the start and finish squares is $(100 - 1) + (100 - 1) = 198$. Since one knight's move can decrease this distance by no more than 3, the knight will need at least 66 moves to reach its destination. This proves that the sequence of moves given above is indeed optimal.

Comments The solution to the puzzle can be considered "greedy" (see the tutorial on algorithm design techniques) since on each step the algorithm decreases the Manhattan distance to the destination square as much as possible. It is not unique, of course. The *Knight's Reach* puzzle (#100) deals with a more general question for an arbitrary $n \times n$ board.

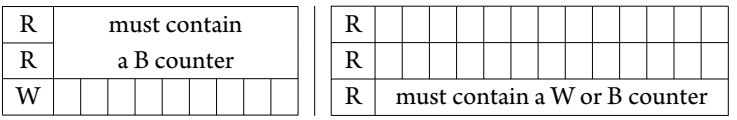
46. Tricolor Arrangement

Solution If $n = 1$, the problem is already solved: all three counters in the only column are of the three different colors. If $n > 1$, we will show that the counters can be rearranged so that the three counters in the first column are of the three different colors; repeating such a rearrangement for the boards of decreasing sizes will solve the problem.

Consider the counters in the first column. There are three possibilities: (i) all three of them are of the different colors, (ii) exactly two are of the same color, and (iii) all of them are of the same color. In case (i), nothing obviously needs to be done with the counters in the first column.

Consider case (ii); we assume without loss of generality that the two counters of the same color are, say, red and they are in the first two rows, whereas the counter in the first column and the third row is, say, white (see the diagram below). Since there are n blue counters on the board and no more than $n - 1$ of them can be in the third row, at least one of them must be in the first two rows. Hence the algorithm in question can scan the first two rows starting with column 2 until a blue counter is encountered; after a blue counter is found, it should be swapped with the red counter in the first column.

Finally, consider case (iii). We assume, without loss of generality, that the three counters in the first column are red. Since each of the three rows must then contain at least one counter of the color other than red, we can scan, say, row 3 to find such a counter and swap it with the counter in the first column to get the situation of case (ii).



Case (ii) Case (iii)

Comments The algorithm solving the puzzle takes advantage of the decrease-by-one and transformation ideas, which are discussed in the tutorial on algorithm design strategies.

The puzzle has been included in A. Spivak's collection [Spi02, Problem 670].

47. Exhibition Planning

Solution a. Since a path through the exhibition must visit each room exactly once, it will have to enter and leave each room through different doors. This implies that the minimum of 17 doors need to be open, including one entrance door and one exit door.

b. On coloring the rooms as squares of a 4×4 chessboard (Figure 4.33), it becomes obvious that any path through the exhibition will have to pass through squares of alternating colors. Since the total of 16 rooms have to be visited, the first and last squares must be colored in the opposite colors. The possible outside door pairs that need to be open are (A1, B1), (A1, B3), (A2, B2), (A2, B4) and, symmetrically, (A4,B4), (A4, B2), (A3, B3), (A3, B1). Figure 4.34 shows one path (of the several possible) for each of the first four pairs listed above. Of course, all the room doors indicated by the path intersection with the square boundaries on the floor plan are assumed to be open as well.

Comments Although the puzzle can be interpreted as a question about Hamilton paths in the graph representing the 4×4 chessboard, it is much easier to solve it directly by exploiting the standard square coloring argument.

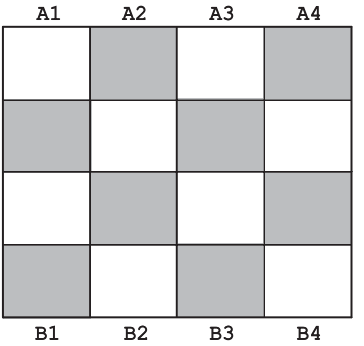


FIGURE 4.33 Sixteen-room plan colored as a chessboard.

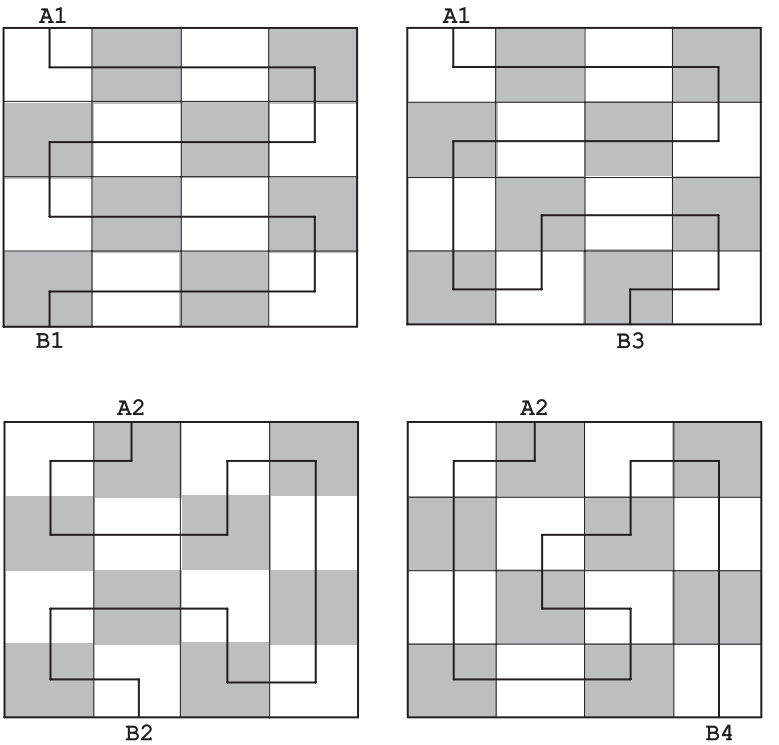


FIGURE 4.34 Four paths through each room with the marked entrance and exit doors.

48. McNugget Numbers

Solution a. It is obvious that the following six numbers are not McNugget numbers: 1, 2, 3, 5, 7, and 11.

b. Every positive number n of McNuggets, which is distinct from the six values listed in part (a), can be represented as some combination of the boxes containing 4, 6, 9, and 20 McNuggets by the following algorithm. If $n \leq 15$, the following

combinations can be used:

$$4 = 1 \cdot 4,\ 6 = 1 \cdot 6,\ 8 = 2 \cdot 4,\ 9 = 1 \cdot 9,\ 10 = 1 \cdot 4 + 1 \cdot 6,$$
$$12 = 3 \cdot 4 (\text{or } 2 \cdot 6),\ 13 = 1 \cdot 4 + 1 \cdot 9,\ 14 = 2 \cdot 4 + 1 \cdot 6,\ 15 = 1 \cdot 6 + 1 \cdot 9.$$

If $n > 15$, solve the problem by the same method (i.e., recursively) for $n - 4$ and add one 4-nugget box to that box combination.

Alternatively—to avoid the recursion—one can find the quotient k and remainder r of division $n - 12$ by 4 (i.e., $n - 12 = 4k + r$ where $k \geq 0$ and $0 \leq r \leq 3$). This yields the representation of n as $4k + (12 + r)$. Using one of the four combinations for $12 + r$ given above and k 4-piece boxes, we get n McNuggets in total.

Comments The above algorithm is based on the decrease (by 4)-and-conquer strategy. The algorithm does not use 20-piece boxes at all. (This happens because 20 is a multiple of 4.) Obviously, any five 4-piece boxes in the solution obtained by the above algorithm can be replaced by one 20-piece box.

In general, finding the smallest positive integer that can *not* be presented as a linear combination of a given set of natural numbers with greatest common divisor 1 is known as the *Frobenius Coin Problem* (see, e.g., [Mic09, Section 6.7]).

49. Missionaries and Cannibals

Solution The problem can be solved by creating a state-space graph for it (Figure 4.35).

The four paths from the initial-state vertex to the final-state vertex, each 11 edges long, represent the solutions with the minimum number of river crossings:

$$2c \to c \to 2c \to c \to 2m \to mc \to 2m \to c \to 2c \to c \to 2c$$
$$2c \to c \to 2c \to c \to 2m \to mc \to 2m \to c \to 2c \to m \to mc$$
$$mc \to m \to 2c \to c \to 2m \to mc \to 2m \to c \to 2c \to c \to 2c$$
$$mc \to m \to 2c \to c \to 2m \to mc \to 2m \to c \to 2c \to m \to mc$$

Comments For the above solution to work, it suffices that one of the missionaries and two of the cannibals can row.

The solution is based on creating a state-space graph, which is a standard method for solving such problems. For an alternative graphical solution, see [Pet09, p. 253].

The puzzle is a 19th-century variation of one of the three river-crossing puzzles in the medieval collection of recreational problems attributed to Alcuin of York (c.735–804). It is very closely related to the *Three Jealous Husbands* problem whose two-couple version is solved in the book's first tutorial. For other references and variations, see David Singmaster's annotated bibliography [Sin10, Section 5.B].

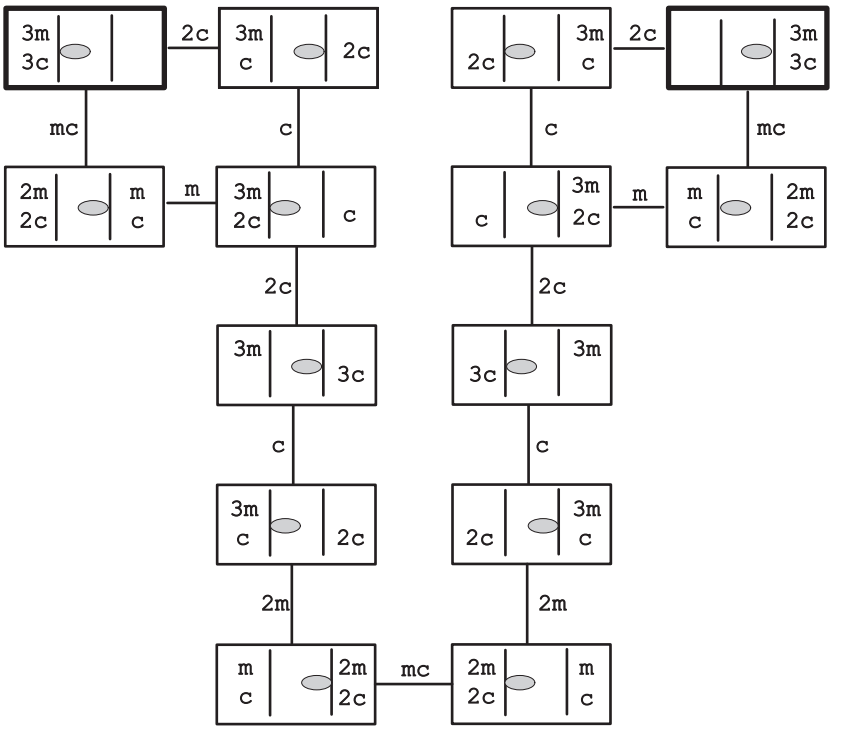


FIGURE 4.35 State-space graph for the *Missionaries and Cannibals* puzzle. The vertices are represented by rectangles, with the two bars || denoting the river and the shaded oval showing the boat's location. The vertices corresponding to the initial and final states are shown in bold. Edge labels indicate the trip's participants.

50. Last Ball

Solution The remaining ball must be black for the (a) part and white for the (b) part of the puzzle.

(a) It is not difficult to note that no matter what color two balls removed from the bag are, the total number of balls will decrease by 1, the number of black balls will change its parity, and the number of white balls will retain the same parity. Therefore, if the algorithm starts with 20 black and 16 white balls, when a single ball remains in the bag, it cannot be white, because 1 is an odd number whereas 16 is even.

(b) If there are 20 black balls and 15 white balls to begin with, a single remaining ball must be white, since the initial number of white balls is odd and the parity of the number of white balls stays the same.

Comments The solution is obviously based on the parity invariant. Note that although in a majority of puzzles an invariant indicates that the puzzle cannot be solved, this is not the case here.

A different instance of this puzzle can be found at [techInt]—one of several sites dedicated to interview questions.

51. Missing Number

Solution Since the sum of all the consecutive integers from 1 to 100 is equal to $S = 1 + 2 + \cdots + 100 = 100 \times 101/2 = 5050$ (see the tutorial on algorithm analysis techniques where this formula is derived), one can find the missing number m by subtracting the sum of the numbers recited, $J = 1 + 2 + \cdots + (m-1) + (m+1) + \cdots + 100$, from this sum: $m = S - J$. Therefore Jill can sum up the numbers recited by Jack and then compute this difference. For example, if the missing number m is 10, $J = 5040$ and m is obtained as the difference $5050 - 5040$.

Since adding three- and four-digit numbers in one's head is not an easy operation to perform, the algorithm can be simplified by computing just the last two digits of all the sums. This fact follows from the observation that the 100 possible values of J, given in the table below, run the range from 4950 to 5049, inclusive, and can be uniquely represented by their last two digits. One can then easily obtain the missing number m by the formula

$$m = \begin{cases} 50 - j & \text{if } 0 \le j \le 49, \\ 150 - j & \text{if } 50 \le j \le 99, \end{cases}$$

where j is the integer between 0 and 99 formed by the last two digits of J. Thus we have

m	1	2	...	49	50	51	...	99	100
J	5049	5048	...	5001	5000	4999	...	4951	4950
j	49	48	...	1	0	99	...	51	50
$50 - j$	1	2	...	49	50				
$150 - j$						51	...	99	100

More formally, the above formulas can be derived from the following property of the modulo division:

$$(S - J) \bmod 100 = (S \bmod 100 - J \bmod 100) \bmod 100,$$

where $S \bmod 100 = 50$ and $j = J \bmod 100$ are the remainders of the division of S and J by 100, which are the numbers formed by the last two decimal digits of S and J, respectively. Then for the missing number m's values between 1 and 99, we have the formula

$$\begin{aligned} m = m \bmod 100 &= (S - J) \bmod 100 = (50 - j) \bmod 100 \\ &= \begin{cases} 50 - j & \text{if } 0 \le j \le 49, \\ 150 - j & \text{if } 51 \le j \le 99, \end{cases} \end{aligned}$$

and if $m = 100$, we can still use the second of the above formulas, because then $J = 5050 - 100 = 4950$, $j = 50$, and $m = 150 - j = 100$.

Comments The solution to the puzzle takes advantage of the transform-and-conquer strategy: we represent the missing number by the difference between S and J and then represent the latter by its last two digits.

The problem of recovering a missing integer in a set of the first positive integers is well-known. This particular version was posed to listeners of the U.S. National Public Radio talk show "Car Talk" on December 6, 2004 [CarTalk].

52. Counting Triangles

Solution There will be $\frac{3}{2}(n-1)n+1$ small triangles after the nth iteration.

The following table contains the number of small triangles $T(n)$ after n iterations of the algorithm for the first few values of n:

n	$T(n)$
1	1
2	$1+3=4$
3	$4+6=10$
4	$10+9=19$

It is not difficult to notice (and prove by induction) that the number of the new small triangles added on the nth iteration is equal to $3(n-1)$ for $n>1$. This implies that the total number of small triangles after the nth iteration can be computed as

$$1+3\cdot 1+3\cdot 2+\cdots+3(n-1)$$
$$=1+3(1+2+\cdots+(n-1))=\frac{3}{2}(n-1)n+1.$$

Comments The puzzle provides a rather straightforward exercise in an algorithm analysis. A similar example is discussed in the tutorial on algorithm analysis techniques.

The puzzle is from *Mathematical Puzzling* by A. Gardiner [Gar99, p. 88, Problem 1].

53. Fake-Coin Detection with a Spring Scale

Solution The minimum number of weighings needed to guarantee identification of the fake coin is $\lceil \log_2 n \rceil$.

Consider an arbitrary subset S of $m \geq 1$ coins selected from the n coins given. If its total weight W is equal to gm, all the coins in S are genuine; otherwise, one of the coins in S is fake. Hence we can proceed searching for the fake among the coins not in S in the former case and among the coins in S in the latter case. If S contains half (or nearly half) the coins given, after one weighing we can proceed searching for the fake in a set half the size. We will have to repeat this halving and weighing operation $\lceil \log_2 n \rceil$ times before n, the size of the original set, can be guaranteed to

be reduced to 1. (Formally, the number of weighings in the worst case is defined by the recurrence $W(n) = W(\lceil n/2 \rceil) + 1$ for $n > 1$, $W(1) = 0$, whose solution is $W(n) = \lceil \log_2 n \rceil$.)

Comments The above algorithm is based on the descrease-by-half strategy. It is almost identical to the algorithm for the *Number Guessing* (*Twenty Questions*) game described in the book's first tutorial.

The problem of detecting a defective coin is one of the most popular puzzle types in recreational mathematics. The version with a spring scale is somewhat less common than the one with a balance scale. For an in-depth discussion of this problem, see the paper by C. Christen and F. Hwang [Chr84].

54. Cutting a Rectangular Board

Solution Since any vertical (horizontal) cut of an $h \times w$ rectangle creates a rectangle with width (height) not smaller than $\lceil w/2 \rceil$ ($\lceil h/2 \rceil$), $\lceil \log_2 n \rceil + \lceil \log_2 m \rceil$ cuts are necessary to cut the $m \times n$ rectangle given into mn 1×1 squares. This number of cuts is also sufficient as performed by the following algorithm. First, make $\lceil \log_2 n \rceil$ vertical cuts along the board's lines—as close to the middle as possible—of all the rectangles with the width greater than 1. Then perform $\lceil \log_2 m \rceil$ horizontal cuts along the board's lines—as close to the middle as possible—of all the rectangles (strips with width equal to 1) with the height greater than 1.

Comments As its one-dimensional version—see the *Cutting a Stick* puzzle (#30)—this puzzle exploits optimality of the decrease-by-half technique. Note that the variant of this puzzle that does not allow stacking, that is, cutting more than one rectangle at a time (the *Breaking a Chocolate Bar* puzzle in the tutorial on algorithm analysis techniques) exploits the quite different idea—that of an invariant.

The earliest reference to this puzzle in David Singmaster's bibliography [Sin10, Section 6.AV], dates 1880; it concerns cutting a 2×4 rectangle into eight unit squares with three cuts. The general instance of the puzzle is included by James Tanton in his book [Tan01], where he gives an induction proof of the formula for the minimum number of cuts (p. 118).

55. Odometer Puzzle

Solution The answers to the first and second questions are 468,559 and 600,000, respectively.

The answer to the first question can be computed as the difference between the total number of readings and those without any 1's. The total number of readings is obviously equal to 10^6. All the readings without 1's are obtained by filling six display positions with one of the nine digits; therefore there are 9^6 such readings. Hence, the total number of readings with digit 1 is equal to $10^6 - 9^6 = 468{,}559$.

To answer the second question, one can simply note that each of the 10 digits will appear in all the readings the same number of times. Therefore the total number of times a 1 will appear can be obtained as one tenth of the total number of all the digits in all the readings: $0.1(6 \cdot 10^6) = 600{,}000$.

Comments The point of this puzzle is that in some cases it is much easier to analyze an algorithm by exploiting its specifics rather than to apply the general techniques discussed in the second tutorial.

The second question was given as a puzzle to listeners of the U.S. National Public Radio talk show "Car Talk" on October 27, 2008 [CarTalk].

56. Lining Up Recruits

Solution Schweik must have understood his order as one to minimize

$$\frac{1}{n}[(h_2 - h_1) + (h_3 - h_2) + \cdots + (h_n - h_{n-1})] = \frac{1}{n}(h_n - h_1), \qquad (1)$$

where n is the number of recruits and h_i, $i = 1, 2, \ldots, n$, is the height of the recruit in the ith position. Since $(h_n - h_1)/n$ can be negative, its minimization means making it a negative number of the largest absolute value. This is achieved if h_1 and h_n are the largest and smallest heights, respectively, with the heights of all the others not contributing to the value of expression (1).

The intended order was, of course, to minimize the average *magnitude* of the difference in height of adjacent men, that is,

$$\frac{1}{n}(|h_2 - h_1| + |h_3 - h_2| + \cdots + |h_n - h_{n-1}|). \qquad (2)$$

Since n is fixed, the multiple $1/n$ can be ignored. Sum (2) is minimized when the heights are sorted in either ascending or descending order to yield the difference $h_{\max} - h_{\min}$ between the maximal and minimal heights. For any other ordering, this sum can be interpreted as the sum of lengths of segments covering the interval with endpoints at $h_{\min}$ and $h_{\max}$ with some overlaps, which makes it larger than $h_{\max} - h_{\min}$. A more formal proof by mathematical induction is not difficult.

Comments The sum $(h_2 - h_1) + (h_3 - h_2) + \cdots + (h_n - h_{n-1})$ in the left-hand side of equality (1) is called the *telescopic series* because of the physical analogy of the folding up of a telescope. It is an occasionally useful tool for algorithm analysis.

The puzzle alludes to the hero of the world-famous novel *The Good Soldier Schweik*, by the Czech writer Yaroslav Hašek (1883–1923). In this satirical novel, Schweik is depicted as a simple-minded man who appears to be eager to execute orders but does it in a manner that, in fact, contradicts their intended goal.

57. Fibonacci's Rabbits Problem

Solution After 12 months, there will be 233 pairs of rabbits.

Let $R(n)$ be the number of rabbit pairs at the end of month n. Clearly, $R(0) = 1$ and $R(1) = 1$. For every $n > 1$, the number of rabbit pairs, $R(n)$, is equal to the number of pairs at the end of month $n - 1$, $R(n - 1)$, plus the number of rabbit pairs born at the end of month n. According to the problem's assumptions, the number of newborns is equal to $R(n - 2)$, the number of rabbit pairs at the end of month $n - 2$. Thus, we have the recurrence relation

$$R(n) = R(n-1) + R(n-2) \text{ for } n > 1, \quad R(0) = 1, R(1) = 1.$$

The following table gives the values of the first 13 terms of the sequence, called the *Fibonacci numbers*, defined by this recurrence relation:

n	0	1	2	3	4	5	6	7	8	9	10	11	12
$R(n)$	1	1	2	3	5	8	13	21	34	55	89	144	233

Comments $R(n)$ differs slightly from the canonical *Fibonacci sequence*, which is usually defined by the same recurrence equation $F(n) = F(n-1) + F(n-2)$ but with different initial conditions, namely, $F(0) = 0$ and $F(1) = 1$. Obviously, $R(n) = F(n+1)$ for $n \geq 0$. Also note that we could find R_{12} by using one of the two well-known formulas for the Fibonacci numbers (e.g., [Gra94, Section 6.6]):

$$R(n) = F(n+1) = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right]$$

and

$$R(n) = F(n+1) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} \quad \text{rounded to the nearest integer.}$$

From the algorithmic perspective, the puzzle specifies an algorithm and asks to determine its output. This is a relatively rare type of algorithmic puzzle; for most of them, the object is to design an algorithm rather than to determine the output of an algorithm given.

The puzzle appeared in *The Book of Calculation* (*Liber Abaci*), written by the Italian mathematician Leonardo of Piza, also known as Fibonacci, in 1202. (The other and more important contribution of the book was the promotion of the Hindu-Arabic numeral system in Europe.) The sequence that solves the problem is one of the most interesting and important sequences ever discovered. Not only does it have many fascinating properties, but it arises, often quite unexpectedly, in many areas of nature and science. Nowadays there exist a large number of books and websites, as well as a special journal called *Fibonacci Quarterly,* devoted to the Fibonacci sequence and its applications. In particular, the website by the British mathematician Ron Knott [Knott] contains a score of puzzles related to the Fibonacci numbers.

58. Sorting Once, Sorting Twice

Solution The answer is zero.

This unexpected result follows from the following property: If there are two n-element sequences $A = a_1, a_2, \ldots, a_n$ and $B = b_1, b_2, \ldots, b_n$ such that all the elements of A are smaller than or equal to the corresponding elements of B (i.e., $a_j \leq b_j$ for $j = 1, 2, \ldots, n$), then the ith smallest element in A is smaller than or equal to the ith smallest element in B for every $i = 1, 2, \ldots, n$. For example, if

$$\begin{array}{lcccc} A: & 3 & 4 & 1 & 6, \\ B: & 5 & 9 & 5 & 8, \end{array}$$

the smallest elements in A and B are, respectively, 1 and 5, the second smallest are 3 and 5, the third smallest are 4 and 8, the fourth smallest are 6 and 9. The simplest way to obtain these values is, of course, to sort the sequences:

$$\begin{array}{lcccc} A': & 1 & 3 & 4 & 6, \\ B': & 5 & 5 & 8 & 9. \end{array}$$

To prove this property, consider $b'_i = b_j$, the ith smallest element in B, which is in some jth position in sequence B. (For instance, if $i = 3$ in the above example, $b'_3 = b_4 = 8$, the 3rd smallest element in B, which is in its fourth position.) Since b'_i is the ith smallest element in B, there are $i - 1$ elements in B that are smaller than or equal to b'_i. (In our example, they are 5 and 5.) For each of these elements and b'_i itself, considered in their original positions in B, there are i elements in the same positions in A that are smaller than or equal to these elements in B. (For the example, they are 3, 1, and 6 .) Hence b'_i is greater than or equal to (at least) i elements in A. This implies that $a'_i \leq b'_i$, where a'_i is the ith smallest element in A: otherwise, a'_i would have had (at least) i elements in A, mentioned above, that are smaller than a'_i.

Consider now any two columns k and l ($k < l$) of cards, after each row is sorted by the cards' rank for the first time, as A and B. After sorting the array's columns, the ith row will contain the ith smallest cards of the columns for every $i = 1, 2, \ldots, n$. According to the property stated above, the card in column k will be smaller than or equal to the card in column l. Since columns k and l were chosen arbitrarily, this means that row i will be sorted.

Comments Of course, this property holds for any doubly sorted two-dimensional arrays. Donald Knuth included it as an exercise in the third volume of *The Art of Computer Programming* [Knu98, p. 238, Problem 27]; he also traced its origin to Hermann Boerner's 1955 book (p. 669). The problem was also included by Peter Winkler into his second book of mathematical puzzles [Win07, p. 21]; in writing a solution, he noted that "this is one of those things that alternate between being mysterious and obvious, each time you think about it" (p. 24).

59. Hats of Two Colors

Solution Suppose only one of the hats is black. Then the prisoner wearing it would see only white hats on the other prisoners. Since he knows that there is at least one black hat, it must be his. Each of the other prisoners sees one black hat and can make no definite conclusion about the color of their hats. So the prisoner who sees only white hats is the only one who steps forward during the first line up, which brings the freedom to all of them.

Suppose there are two black hats. Then no prisoner steps forward during the first line up because none of them can be certain about the color of his hat. But during the second line up both prisoners who see one black hat can step forward. Indeed, since no prisoner stepped forward during the first line up, they know that there are at least two black hats. If a prisoner sees just one black hat, he can conclude that the second black hat is his. All the prisoners who see two black hats cannot be certain about the color of their hats and therefore will stay in line.

In general, if there are k black hats, $1 \le k \le 11$, no prisoner will step forward during the first $k-1$ line ups. But on the kth line up, each of the prisoners who sees $k-1$ black hats will be able to step forward by the same reasoning as above: he knows that there are at least k black hats since the previous line up did not stop the test. If he sees $k-1$ on the other prisoners, then there are exactly k black hats, one of which is his. At the same time, all the $n-k$ prisoners with white hats should—and given their smartness will— stay in line because they cannot deduce the color of their hats. The smarts can win freedom after all!

Comments The solution is in the decrease-and-conquer spirit, applied bottom up by the number of black hats.

Several versions of this problem have been published both in print and electronically. This particular version follows, in a slightly different wording, W. Poundstone's book [Pou03, p. 85]. The earliest version of three men with spots on foreheads appeared around 1935 and is attributed to the prominent Princeton logician Alonzo Church. For a more challenging puzzle with a similar set up, see the *Hats with Numbers* puzzle (#147) in this book.

60. Squaring a Coin Triangle

Solution The minimum number of coins that need to be moved is $\lfloor n/2 \rfloor \lceil n/2 \rceil$. The puzzle has two solutions for every even $n > 2$, one solution for every odd $n > 1$, and three solutions for $n = 2$.

Since the total number of coins is equal to the nth square number $S_n = \sum_{j=1}^{n} (2j-1) = n^2$, each of the n rows composing the square in question must have n coins. It is natural to create such a square by adjusting the triangle's rows by moving the extra coins from the $\lfloor n/2 \rfloor$ rows with more than n coins to the $\lfloor n/2 \rfloor$ rows with less than n coins. This can be done by moving $n-1$ coins from the longest (i.e., hypotenuse) row with $2n-1$ coins to the row with 1 coin,

then moving $n-3$ coins from the next row with $2n-3$ coins to the row with 3 coins, and so on as illustrated in Figures 4.36 and 4.37 for the even and odd cases of n, respectively. (In these figures, a given triangle is shown with its right angle as the apex and the coin rows that are parallel to the hypotenuse being horizontal.) Obviously, any algorithm that on each of its moves takes a coin from a row that needs to be shorten and puts it into a row that needs to be lengthen solves the problem in the minimum number of moves.

FIGURE 4.36 Two symmetric solutions for S_4 coins. + and − indicate centers of added and moved coins, respectively. The stationary coins are shown by black dots.

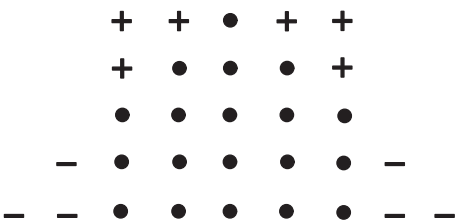


FIGURE 4.37 Solution for S_5 coins.

The total number of coin moves needed to accomplish such a transformation, $M(n)$, can be obtained as follows:

$$M(n) = \sum_{j=1}^{\lfloor n/2 \rfloor} (n-(2j-1)) = \sum_{j=1}^{\lfloor n/2 \rfloor} n - \sum_{i=1}^{\lfloor n/2 \rfloor} (2j-1)$$

$$= n\lfloor n/2 \rfloor - \lfloor n/2 \rfloor^2 = \lfloor n/2 \rfloor (n - \lfloor n/2 \rfloor) = \lfloor n/2 \rfloor \lceil n/2 \rceil.$$

The only contender for a square obtainable in fewer moves could be the one composed of the odd-numbered lines that are parallel to one of the triangle's leg. But this would require moving all the coins in the even numbered lines, whose total number is equal to

$$\bar{M}(n) = \sum_{j=1}^{n-1} j = (n-1)n/2 > \lfloor n/2 \rfloor \lceil n/2 \rceil \text{ for } n > 2.$$

For $n = 2$, however, $\bar{M}(2) = M(2) = 1$, and the problem does have the third, and quite different, solution shown in Figure 4.38.

FIGURE 4.38 Three solutions for S_2 coins.

Comments An instance of this puzzle by S. Grabarchuk was posted on the website Puzzles.com [Graba].

61. Checkers on a Diagonal

Solution The puzzle has a solution if and only if either $n - 1$ is a multiple of 4 or n is a multiple of 4. The number of moves needed is equal to $(n - 1)n/4$.

We can measure the distance between a current position of the checkers and their destination at the bottom row by the sum of the squares below all the checkers. At the initial position, this distance is equal to $(n - 1) + (n - 2) + \ldots + 1 = (n - 1)n/2$; at destination, it is equal to 0. Since any move decreases the sum exactly by 2, its parity does not change. Therefore, for the problem to have a solution, it is necessary for $(n - 1)n/2$ to be even.

Note that $(n - 1)n/2$ is even if and only if either $n - 1$ is a multiple of 4 or n is a multiple of 4. Indeed, if $(n - 1)n/2 = 2k$, then $(n - 1)n = 4k$, and since either $n - 1$ or n is odd, the other must be divisible by 4. Conversely, if either $n - 1$ or n is a multiple of 4, $(n - 1)n/2$ is obviously even.

Thus, to have a solution, it is necessary that either $n = 4k$ or $n = 4k + 1$ where $k \geq 1$. This condition is also sufficient since the problem can be solved by the following algorithm. Move the $\lceil (n - 2)/2 \rceil$ pairs of checkers in consecutive columns—that is, columns 1 and 2, 3 and 4, and so on—down as far as possible. Then move the $\lfloor n/4 \rfloor$ pairs in consecutive odd columns—that is, columns 1 and 3, 5 and 7, and so on—down one square. (Note that if n or $n - 1$ is a multiple of 4, there is an even number of odds less than n and $n - 1$, respectively.) The algorithm is illustrated in Figures 4.39 and 4.40.

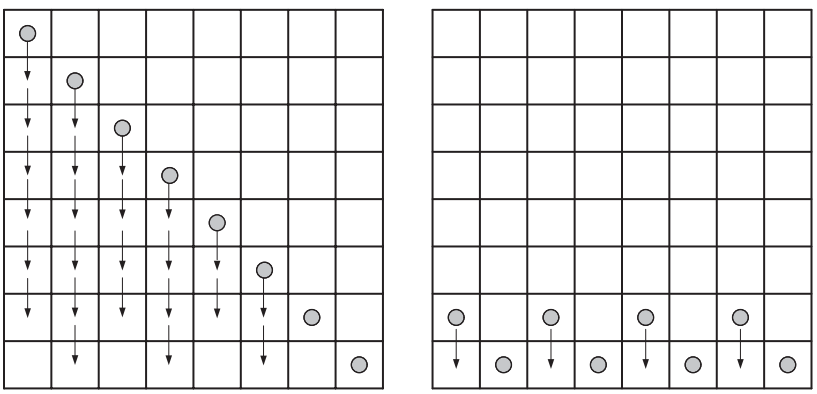


FIGURE 4.39 Moving $n = 8$ checkers from the diagonal to the bottom row one pair at a time.

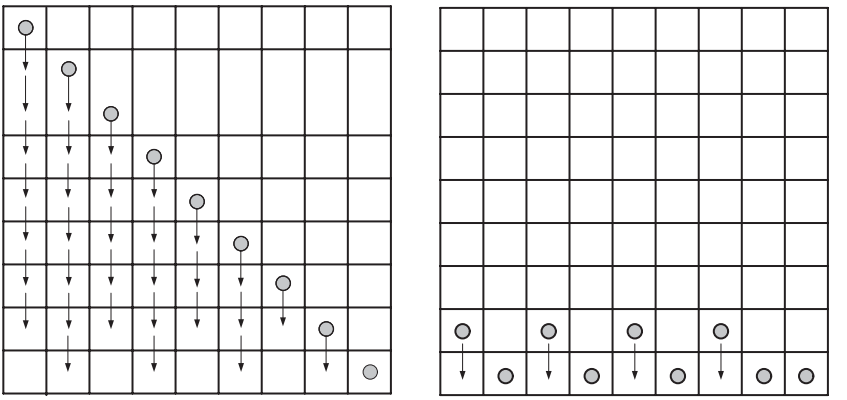


FIGURE 4.40 Moving $n = 9$ checkers from the diagonal to the bottom row one pair at a time.

Finally, any algorithm solving the problem, including the one given above, decreases the distance between the initial and final positions by 2 on each of its iterations. Hence, any such algorithm makes the same $(n-1)n/4$ moves.

Comments The puzzle exploits the formula for the sum of the first consecutive integers and the parity version of the invariant idea. Both are discussed in the tutorial on algorithm analysis techniques.

The puzzle is a generalization of Problem 448 in [Spi02], which asks about a solution existence for $n = 10$.

62. Picking Up Coins

Solution As suggested by the hint to this puzzle, we can use dynamic programming to solve it. Let $C[i, j]$ be the largest number of coins the robot can collect and bring to the cell (i, j) in the ith row and jth column of the board. It can reach this cell either from the adjacent cell $(i-1, j)$ above it or from the adjacent cell $(i, j-1)$ to the left of it. The largest number of coins that can be brought to these cells are $C[i-1, j]$ and $C[i, j-1]$, respectively. (Of course, there are no adjacent cells above cells in the first row, and there are no adjacent cells to the left of cells in the first column. For such cells we assume that $C[i-1, j]$ and $C[i, j-1]$ are equal to 0 for their nonexistent neighbors.) Therefore, the largest number of coins the robot can bring to cell (i, j) is the maximum of these two numbers plus one possible coin at cell (i, j) itself. In other words, we have the following formula for $C[i, j]$:

$$C[i, j] = \max\{C[i-1, j],\ C[i, j-1]\} + c_{ij} \text{ for } 1 \le i \le n,\ 1 \le j \le m, \quad (1)$$

where $c_{ij} = 1$ if there is a coin in cell (i, j) and $c_{ij} = 0$ otherwise, and $C[0, j] = 0$ for $1 \le j \le m$ and $C[i, 0] = 0$ for $1 \le i \le n$.

Using these formulas, we can fill in the $n \times m$ table of $C[i, j]$ values either row by row or column by column, as it is typically done in dynamic programming algorithms. This is illustrated in Figure 4.41b for the coin setup in Figure 4.41a.

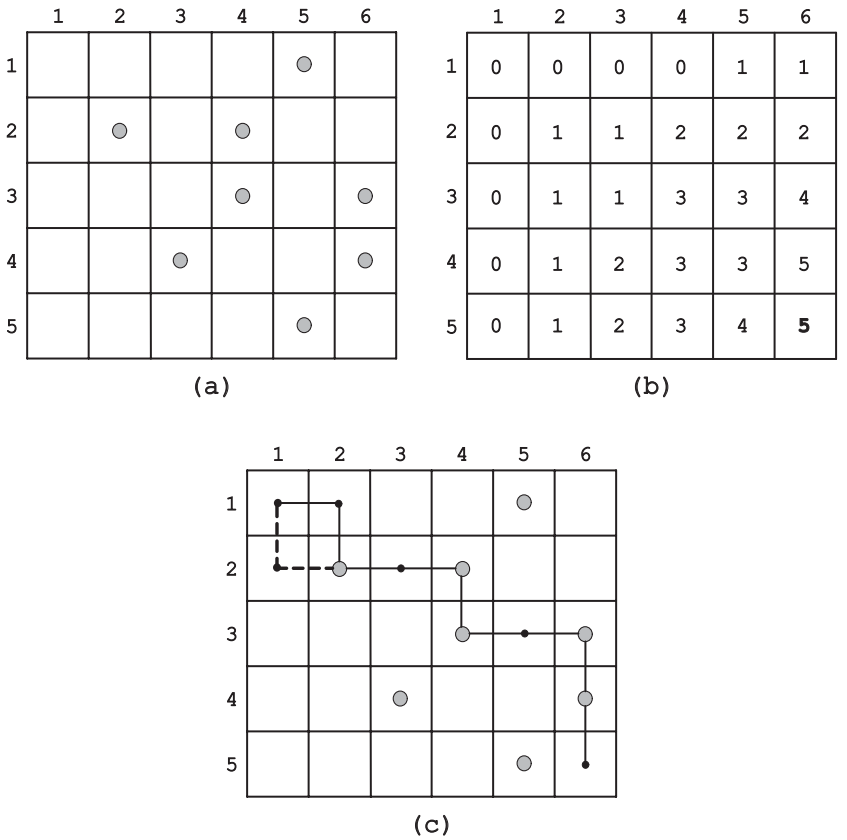


FIGURE 4.41 (a) Coins given. (b) Dynamic programming algorithm results. (c) Two paths to collect 5 coins, the maximum number of coins possible.

To get an actual path that yields the maximum number of coins, one needs to ascertain whether the maximum in equality (1) was provided by the value above the cell or the one to the left of it: in the former case, the path comes down to that cell; in the latter case, it comes from the left. In case of a tie, an optimal path is not unique, and either option provides a step in some optimal path. For example, Figure 4.41c shows two such optimal paths for the coin setup in Figure 4.41a.

Comments The puzzle is an easier version of a problem in [Gin03]. It provides a nice example of a dynamic programming application.

63. Pluses and Minuses

Solution The puzzle has a solution if and only if either n or $n + 1$ is divisible by 4.

The problem is equivalent to partitioning n integers from 1 to n into two disjoint (i.e., without common elements) subsets with the same sum: a subset of numbers with a plus before them and a subset of numbers with a minus before them. Since $S = 1 + 2 + \cdots + n = n(n + 1)/2$, the sum of the numbers in each of the subsets must be equal to exactly one half of S. This implies that $n(n + 1)/2$ must be even as a necessary condition for the puzzle to have a solution. As shown below, this condition is also sufficient for a solution's existence.

Note that $n(n + 1)/2$ is even if and only if either n is a multiple of 4 or $n + 1$ is a multiple of 4. Indeed, if $n(n + 1)/2 = 2k$, then $n(n + 1) = 4k$, and since either n or $n + 1$ is odd, the other must be divisible by 4. Conversely, if either n or $n + 1$ is a multiple of 4, $n(n + 1)/2$ is obviously even.

If n is divisible by 4, we can, for example, partition the sequence of integers from 1 to n into $n/4$ groups of four consecutive integers and then put pluses before the first and fourth number and minuses before the second and third number in each of these groups:

$$(1 - 2 - 3 + 4) + \cdots + ((n - 3) - (n - 2) - (n - 1) + n) = 0. \qquad (1)$$

If $n + 1$ is divisible by 4, then $n = 4k - 1 = 3 + 4(k - 1)$, and we can exploit the same idea by first taking care of the first three numbers as follows:

$$(1+2-3)+(4-5-6+7)+\cdots+((n-3)-(n-2)-(n-1)+n) = 0. \quad (2)$$

To summarize, the problem can be solved by the following algorithm. Compute $n \bmod 4$ (the remainder of the division of n by 4). If the remainder is equal to 0, insert the "+" and "−" signs as indicated by formula (1); if the remainder is equal to 3, insert the "+" and "−" signs as indicated by formula (2); otherwise, return the "no solution" message.

Comments This is an algorithmic version of a well-known puzzle, which exploits several themes, including the ubiquitous formula for the sum of consecutive integers from 1 to n and its parity. It should also be noted that for an arbitrary sequence of integers there is no known efficient algorithm for the puzzle known as the *Partition Problem*. Moreover, most computer scientists believe that no efficient algorithm for it exists because the partition problem is known to be NP-complete.

64. Creating Octagons

Solution We assume without loss of generality that the points are numbered left to right from 1 to n, with ties, if any, resolved by numbering a lower of the two points first. (More formally, we assume that the points are sorted according their first coordinate in the plane's Cartesian coordinate system.) Consider the first eight points $p_1, \ldots, p_8$ and draw the straight line connecting p_1 and p_8 There are two cases: either all the other six points $p_2, \ldots, p_7$ lie on the same side of this line (Figure 4.42a) or they lie on both sides of the line (Figure 4.42b). In the former

case, an octagon can be obtained by connecting the points $p_1, \ldots, p_8$ in this order, with the line connecting p_1 and p_8 also serving as one of the octagon's sides. If the points $p_2, \ldots, p_7$ lie on both sides of the line connecting p_1 and p_8, an octagon is obtained by connecting consecutive pairs of the points on or above this line to form its upper boundary and connecting consecutive pairs of the points on or below this line to form its lower boundary (Figure 4.42b).

Note that all the 1992 remaining points lie either to the right of the constructed octagon, or just one leftmost point p_9 lies on the same vertical line as p_8, the rightmost point of the constructed pentagon. Hence, using the same method, we can construct an octagon with its vertices at the next eight points $p_9, \ldots, p_{16}$ and it will not intersect with the first octagon. Repeating this step for every consecutive eight points on the list solves the problem.

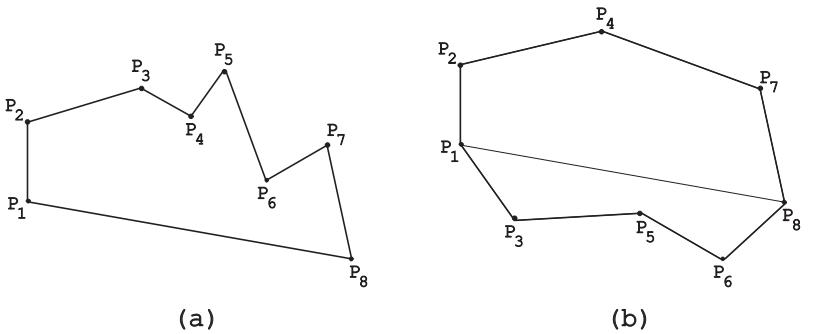


FIGURE 4.42 Creating a simple octagon with vertices at eight given points.

Comments The solution exploits the presorting, divide-and-conquer, and decrease-and-conquer ideas. It is similar to a well-known algorithm in computer science called *quickhull*, which is used for constructing *convex hulls* of point sets (see, e.g., [Lev06, Section 4.6]).

65. Code Guessing

Solution The following sequence of bit strings can be used in n consecutive questions to identify a code $b_1 b_2 \ldots b_n$:

$$000\ldots0,\ 100\ldots0,\ 1100,\ldots,\ 11\ldots10.$$

The first answer, a_1, gives the total number of 0's in the code being identified. Let a_2 be the answer to the second question. Since the bit strings used in the first two questions differ only by their first bits, the corresponding numbers of the correct bits, a_1 and a_2, differ by one, identifying the first bit in the code, b_1, as follows: if $a_1 < a_2$, $b_1 = 1$; if $a_1 > a_2$, $b_1 = 0$. (For example, for the code 01011, $a_1 = 2$ and

$a_2 = 1$, and therefore $b_1 = 0$ since $a_1 > a_2$.) Repeating the same argument for the next $n - 2$ questions makes it possible to identify in turn the bits $b_3, \ldots, b_{n-1}$ in the code in question. The last bit b_n can be found by using the total number of 0's in the code determined by the first question: if this number is equal to the number of zeros among the code's first $n - 1$ bits, which are now known, then $b_n = 1$; otherwise, $b_n = 0$.

In fact, one can start with any n-bit string and make a sequence of n questions by changing a different bit one at a time.

Comments The puzzle, which is closely related to the popular board game called *Mastermind*, is from the book by Dennis Shasha [Sha07]. The solution given above is based on the decrease-and-conquer strategy. It is not optimal for every value of n. For example, any 5-bit code can be identified by the sequence 00000, 11100, 01110, 00101 (pp. 105–106 in Shasha's book). No general formula for the minimum number of questions needed for an arbitrary n is known.

66. Remaining Number

Solution Any positive odd integer less than 50 can be obtained.

The initial sum of the numbers on the board is equal to $1 + 2 + \ldots + 50 = 1275$, which is odd. Each operation of replacing a and b by $|a - b|$, where $a \leq b$ without loss of generality, decreases the sum by $2a$:

$$S_{\text{new}} = S_{\text{old}} - a - b + |a - b| = S_{\text{old}} - b - a + b - a = S_{\text{old}} - 2a.$$

This immediately implies that the new sum must be odd if the old sum was odd, and therefore no even number can result from repeated applications of such an operation starting with 1275.

Further, all the numbers on the board are always nonnegative. They are also less than or equal to 50, since $|a - b|$ is always less than or equal to the maximum of a and b for nonnegative a and b.

We will show now that any odd integer from 1 to 49, inclusive, can be obtained by applying the puzzle's operation 49 times. Let k be such a number. We can obtain it on the first iteration by subtracting 1 from $k + 1$. Then we can apply the operation to the pairs of the remaining consecutive integers,

$$(2, 3), (4, 5), \ldots, (k - 1, k), (k + 2, k + 3), \ldots, (49, 50),$$

to get 24 ones on the board while erasing the above pairs. Applying the operation to the 24 pairs of ones yields 12 zeros, which can be reduced to a single zero after applying the operation 11 times. Finally, applying the operation to the two remaining numbers, k and 0, yields k.

Comments The puzzle is a version of the parity-based problem well-known in mathematical circles around the world. The *Pluses and Minuses* puzzle (#63) in this book is another variation on this theme.

67. Averaging Down

Solution Averaging the amount of water in the nonempty vessel successively with each of the nine empty vessels leaves $a/2^9$ pints in it. This is the minimum amount of water achievable for that vessel. Indeed, consider m, the minimum positive amount of water among all the vessels in their current state. (Initially, $m = a$ and our goal is to minimize it.) Since the average of two numbers is always greater than or equal to the smaller of the two, the value of m can be decreased by the averaging operation only if this operation involves a vessel containing m pints and an empty vessel. After repeating the averaging operation with each empty vessel, no empty vessel will remain, making an increase in m impossible. Hence, the ultimate minimal value of m we can get here is equal to $a/2^9 = a/512$ pints.

Comments The puzzle is solved by the greedy approach. Its justification is based on the monovariant argument, which is discussed in the first tutorial.

The puzzle was included, in a different wording, in the exercises to the *Kvant* article on monovariants [Kur89, Problem 6], after being offered at the Leningrad mathematical olympiad in 1984.

68. Digit Sum

Solution The total digit sum in integers from 1 to 10^n, inclusive, is $45n10^{n-1} + 1$. In particular, for $n = 6$, it is equal to 27,000,001.

It is convenient to disregard 10^n, which obviously contributes 1 to the sum in question, and compute the digit sum in all integers from 1 to $10^n - 1$; we will denote the latter sum by $S(n)$. It is also convenient to pad every integer smaller than 10^{n-1} by an appropriate number of leading zeros so that all the integers are composed of n digits.

A simple way to find $S(n)$ is by pairing 0 with $10^n - 1$, 1 with $10^n - 2$, 2 with $10^n - 3$, and so on, because the sum of the digits in each of these pairs is equal to $9n$. (The pairing trick was used in the algorithm analysis tutorial to compute the sum of the first n positive integers.) Since there are obviously $10^n/2$ such pairs, the sum $S(n)$ is equal to $9n \cdot 10^n/2 = 45n \cdot 10^{n-1}$.

One can also follow an approach used for solving the *Odometer Puzzle* (#55) in this book. Assuming the padding, there are 10^n n-digit integers, in which each of the 10 digits occurs the same $n \cdot 10^n/10 = n \cdot 10^{n-1}$ number of times. $S(n)$, the total sum of the digits in them, can therefore be computed as

$$\begin{aligned} &0 \cdot n \cdot 10^{n-1} + 1 \cdot n \cdot 10^{n-1} + 2 \cdot n \cdot 10^{n-1} + \cdots + 9 \cdot n \cdot 10^{n-1} \\ &= (1 + 2 + \cdots + 9)n \cdot 10^{n-1} = 45n \cdot 10^{n-1}. \end{aligned}$$

The third way to solve the problem is to set up a recurrence relation for $S(n)$. Obviously, $S(1) = 1 + 2 + \cdots + 9 = 45$. For each of the 10 possible leftmost digits in a n-digit integer, the total sum of the other digits is equal to $S(n-1)$. The sum

contributed by the leftmost digits is equal to $10^{n-1}(0 + 1 + 2 + \cdots + 9) = 45 \cdot 10^{n-1}$. This yields the following recurrence:

$$S(n) = 10S(n-1) + 45 \cdot 10^{n-1} \text{ for } n > 1,\ S(1) = 45.$$

Solving the recurrence by backward substitutions (see the algorithm analysis tutorial) yields $S(n) = 45n \cdot 10^{n-1}$.

Thus, for $n = 6$, $S(6) = 45 \cdot 6 \cdot 10^{6-1} = 27{,}000{,}000$ and the total sum of the digits is equal to 27,000,001.

Comments Pairing the numbers with the same digit sum can be considered either representation change or an invariant as Z. Michalewicz and M. Michalewicz did in their book [Mic08, pp. 61–62]. The pairing method was also used by B. A. Kordemsky [Kor72, p. 202]. The recurrence-based approach is an application of the decrease-by-one strategy.

69. Chips on Sectors

Solution The problem has a solution if and only if n is either odd or a multiple of 4.

We will first prove that this condition is necessary for the existence of a solution. If n is not odd, we need to show that the problem can have a solution only if n is a multiple of 4. Let us number all the sectors from 1 to n, starting with an arbitrary sector and moving, say, clockwise. Consider the sum S of the sector numbers currently occupied by the chips. (If there are several chips on the same sector, the sector's number is included in the sum several times as well—once for each chip.) It is easy to see that if n is even, the parity of S remains the same after each move shifting two chips to neighboring sectors. Indeed, moving one checker to a neighboring sector changes the sum's parity and hence moving two of them preserves it. If the problem has a solution, that is, all the chips can be brought to some sector j $(1 \le j \le n)$, then in the final position $S = nj$ will be even because we consider the case of even n. Hence, in the initial position, the sum $1 + 2 + \cdots + n = n(n+1)/2$ must be even as well: $n(n+1)/2 = 2k$. But then $n(n+1) = 4k$, and since $n + 1$ is odd, n must be divisible by 4. This completes the proof of the condition's necessity.

Let us now prove that the stated condition is also sufficient. If n is odd, all the chips can be brought to the middle sector (i.e., the sector $j = (1+n)/2$) by simply moving together each pair of chips equidistant from the middle one—that is, the chips from sectors 1 and n, 2 and $n-1$, $\ldots$, $j-1$ and $j+1$—until the pair reaches the middle sector. If, on the other hand, n is a multiple of 4, we can, for example, start by moving all the chips in the odd-numbered sectors to their even-numbered neighbors, two at a time in the clockwise direction. (There is an even number of odd numbers less than $n = 4i$.) Then we can move the pair of chips from sector 2 to sector n, do the same for the pair from sector 4, and so on, until all the chips are collected in the last sector.

Comments This puzzle, exploiting the notion of parity, is a generalization of Problem 2 in [Fom96, p. 124].

70. Jumping into Pairs I

Solution Obviously, the problem cannot be solved for an odd number of coins because in the final state all the coins must be arranged in pairs making their total number even. Considering all possible moves, it is not difficult to see that the puzzle does not have a solution for $n = 2$, 4, and 6. For $n = 8$, it has several solutions. One of them—which can be found by backtracking—is as follows: 4 on 7, then 6 on 2, then 1 on 3, and 5 on 8. Any instance with an even number of coins greater than 8 (i.e., $n = 8 + 2k$ where $k > 0$) can be reduced to the instance with two fewer coins (i.e., $n = 8 + 2(k - 1)$) by moving the coin in position $8 + 2k - 3$ on the last coin in position $8 + 2k$, which reduces the size of the instance by 2. Repeating this operation k times reduces the instance with $n = 8 + 2k$ coins to the instance with 8 coins, whose solution was given above.

This algorithm obviously makes the minimum number of moves possible because it forms a new pair of coins on each move.

Comments The algorithm above takes advantage of the backtracking and decrease-and-conquer strategies.

The oldest reference to this puzzle in David Singmaster's bibliography [Sin10, Section 5.R.7], is Japanese, dated 1727. The puzzle has been discussed, among many others, by Ball and Coxeter [Bal87, p. 122], and by Martin Gardner [Gar89], who called it "one of the oldest and best coin puzzles" (p. 12).

71. Marking Cells I

Solution Besides $n = 4$, the puzzle has a solution for every $n > 6$ except $n = 9$.

Obviously, the puzzle does not have a solution for $n = 1$, 2, and 3. For $n = 4$, the solution is given in the puzzle's statement. There is also no solution for $n = 5$. This assertion can be proved by contradiction. Assume that a required region of five marked cells, each with a positive even number of neighbors, exists. Consider the highest in the leftmost column of the cells in this region. This cell must have one neighbor to the right and one neighbor below it. The lowest cell in the leftmost column of the region must have one neighbor to the right and one neighbor above. These two possible configurations of the marked cells are shown in Figure 4.43.

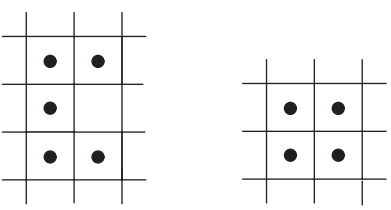


FIGURE 4.43 Two possible configurations of the highest and lowest marked cells in the leftmost column with an even number of neighbors for $n = 5$.

But the region of five marked cells in that figure fails the requirement that each marked cell must have a positive even number of marked neighbors, and it is impossible to mark one more cell to the four-cell configuration to satisfy this requirement either. A similar analysis for $n = 6$ and $n = 9$ shows that the puzzle does not have solutions for these values of n as well.

Solutions for $n = 7$ and $n = 11$ are shown in the Figures 4.44a and 4.44b. The latter can be extended to a solution for any odd $n > 11$ by simply expanding the larger loop by two marked cells, as shown in Figure 4.44c for $n = 13$.

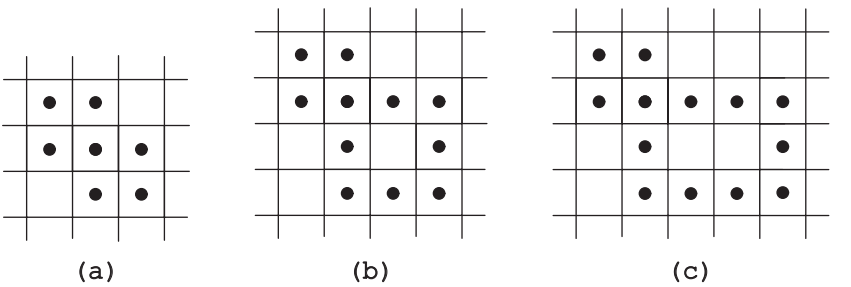


FIGURE 4.44 Solutions to the *Marking Cells I* puzzle for (a) $n = 7$, (b) $n = 11$, and (c) $n = 13$.

For every even $n > 6$, a solution is provided by a frame-like region of length $(n - 2)/2$ and height 3, as shown in Figure 4.45 for $n = 8$ and $n = 10$.

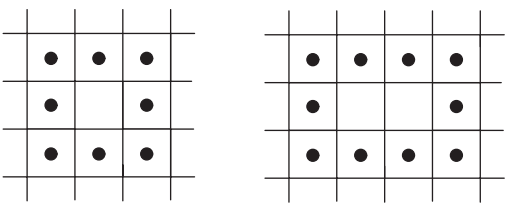


FIGURE 4.45 Solutions to the *Marking Cells I* puzzle for $n = 8$ and $n = 10$.

Comments The above solutions are based on the incremental approach (decrease-and-conquer strategy applied bottom up). For most values of n, there are also other solutions. As to the case of an odd number of marked neighbors, it is the subject of the next puzzle, *Marking Cells II* (#72).

The puzzle is included in B. A. Kordemsky's last book [Kor05, pp. 376–377].

72. Marking Cells II

Solution The puzzle can be solved only for an arbitrary even number of cells.

For $n = 2$, the obvious solution is depicted in Figure 4.46a. Marking two cells adjacent to, say, the rightmost cell in this solution—one horizontally and the other

vertically (say, up)—yields a solution for $n = 4$ (Figure 4.46b). Repeating the same operation again but marking the vertical neighbor below rather than above the rightmost cell, yields a solution for $n = 6$ (Figure 4.46c). In this manner, we can solve the puzzle for any even value of n.

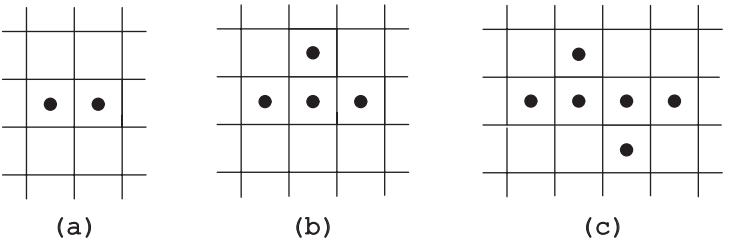


FIGURE 4.46 Solutions to the *Marking Cells II* puzzle for (a) $n = 2$, (b) $n = 4$, and (c) $n = 6$.

Now we will prove that it is impossible to mark an odd number of cells, each with an odd number of marked neighbors. If this were possible, we would have the following contradiction. On the one hand, if we add the number of shared edges for each marked cell, we should get an even number since each common edge is counted twice. On the other hand, this number must be odd as the sum of an odd number of terms each of which is odd.

Comments The solution for even n's is clearly based on the incremental approach (decrease-and-conquer strategy applied bottom up). The impossibility proof for odd n's is identical to that of the well-known mathematical proposition called the *Handshaking Lemma*: the number of people who have shaken hands at a party with an odd number of people must be even (e.g., [Ros07, p. 599]). In our case, people shaking hands are marked cells sharing edges with other marked cells.

The puzzle is included in B. A. Kordemsky's last book [Kor05, pp. 376–377].

73. Rooster Chase

Solution It is helpful to consider the game's grid as obtained by connecting centers of the squares of a standard 8×8 chessboard (Figure 4.47a). Under this interpretation, the capture can only occur on the farmer's move when the counters occupy two adjacent squares (horizontally or vertically), which always have the opposite colors. Initially, the counters are on the squares of the same color. Since both counters move to a square of the opposite color on each move, the capture cannot happen if the farmer moves first.

If the farmer moves second, he can always push the rooster into a corner to guarantee capture by always moving to a square on the same diagonal with the rooster and closer to him. We will denote positions of the farmer and the rooster as (i_F, j_F) and (i_R, j_R), respectively, where i and j are the row and column of the squares they occupy. Geometrically, the farmer's position (i_F, j_F) and squares

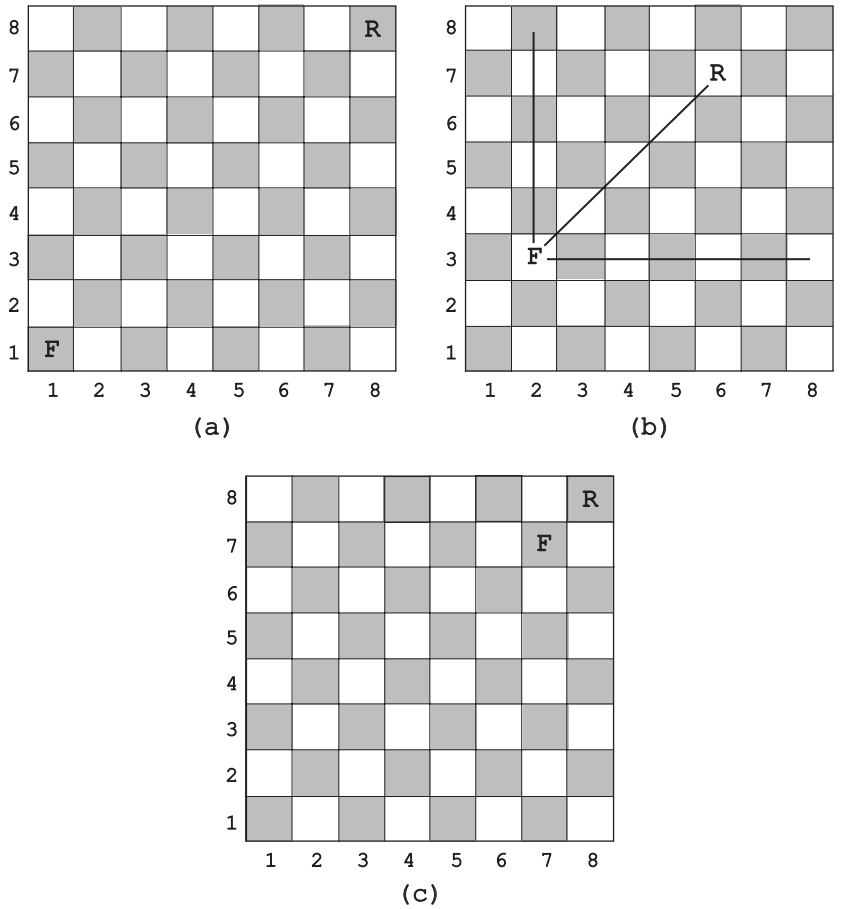


FIGURE 4.47 The *Rooster Chase* game. (a) Starting position. (b) Winning strategy. (c) Position before the last rooster move.

$(8, j_F), (8, 8), (i_F, 8)$ form a rectangle that contains the rooster and from which the rooster cannot escape (Figure 4.47b). The rectangle shrinks along one of its side on each move until the rooster is pushed into the upper right corner (Figure 4.47c).

A more formal way to determine the farmer's move is to compute $i_R - i_F$, the row distance between his current square (i_F, j_F) and the rooster's current square (i_R, j_R); compute $j_R - j_F$, the column distance between these squares; and then find the maximum of these two values:

$$d = \max\{i_R - i_F,\ j_R - j_F\}.$$

The farmer makes the move to decrease d, that is, he moves to the right if the column distance $j_R - j_F$ is larger than the row distance $i_R - i_F$ and he moves up otherwise. (Either the column distance or the row distance is always larger than the other after a move by the rooster.)

Since the Manhattan distance from the current position of the farmer to the upper right corner—computed as $(8 - i_F) + (8 - j_F)$—decreases by 1 after each move of the farmer, after 12 moves the farmer and the rooster will be in the position shown in Figure 4.47c, from which the farmer will catch the rooster on his next move. Thus, at the longest, it will take 14 moves of each piece for the game to end, provided the rooster goes first from the starting position in Figure 4.47a. Of course, the rooster may hasten the inevitable by getting to a square adjacent to the one occupied by the rooster in as little as seven moves.

Comments The solution exploits the invariant idea to determine who needs to go first and the greedy strategy for the capture algorithm. The transformation from the grid to the chessboard may also be noted, although it does not play the crucial role here.

The problem is a simplification of the *Chickens in the Corn* puzzle discussed in the book's second tutorial. Similar puzzles has been included in many puzzle collections, for example, [Gar61, p. 57] and [Tan01, Problem 29.3].

74. Site Selection

Solution The optimal values for x and y are the medians of $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, respectively.

The problem is obviously equivalent to minimizing independently the sum of the horizontal distances $|x_1 - x| + \cdots + |x_n - x|$ and the sum of the vertical distances $|y_1 - y| + \cdots + |y_n - y|$. Thus, we have two instances of the same problem with different inputs.

To solve the problem of minimizing $|x_1 - x| + \cdots + |x_n - x|$, it is convenient to assume that the points $x_1, \ldots, x_n$ are sorted in nondecreasing order. (If it is not the case, we can always sort them first and then rename.) It is also useful to take advantage of the geometric interpretation of $|x_i - x|$ as the distance between points x_i and x on the real line. We will consider the cases of even and odd n separately.

Let n be even. Consider first the case of $n = 2$. The sum $|x_1 - x| + |x_2 - x|$ is equal to $x_2 - x_1$, the length of the interval with the endpoints at x_1 and x_2, for any point x of this interval (including the endpoints), and it is larger than $x_2 - x_1$ for any point x outside of this interval. This implies that for any even n, the sum

$$\begin{aligned}
&|x_1 - x| + |x_2 - x| + \cdots + |x_{n-1} - x| + |x_n - x| \\
&\quad = (|x_1 - x| + |x_n - x|) + (|x_2 - x| + |x_{n-1} - x|) \\
&\qquad + \cdots + (|x - x_{n/2}| + |x_{n/2+1} - x|)
\end{aligned}$$

is minimized when x belongs to each of the intervals with the endpoints at x_1 and x_n, x_2 and x_{n-1}, $\ldots$, $x_{n/2}$ and $x_{n/2+1}$. Since every interval in this sequence is embedded in its predecessor, it is both necessary and sufficient that x belongs to the last of them. In other words, any value of x (and only such value) for which $x_{n/2} \le x \le x_{n/2+1}$ solves the problem.

Let n be odd. Since the case of $n = 1$ is trivial—to minimize $|x_1 - x|$ we need to make $x = x_1$—let us consider the case of $n = 3$ first. The sum

$$|x_1 - x| + |x_2 - x| + |x_3 - x| = (|x_1 - x| + |x_3 - x|) + |x_2 - x|$$

is minimized when $x = x_2$ because this value minimizes both $|x_1 - x| + |x_3 - x|$ and $|x_2 - x|$. The same arguments extends to any odd value of n:

$$\begin{aligned}&|x_1 - x| + |x_2 - x| + \cdots + |x - x_{\lceil n/2 \rceil}| + \cdots + |x_{n-1} - x| + |x_n - x| \\ &= (|x_1 - x| + |x_n - x|) + (|x_2 - x| + |x_{n-1} - x|) + \cdots + |x - x_{\lceil n/2 \rceil}|\end{aligned}$$

is minimized when $x = x_{\lceil n/2 \rceil}$, the point for which the number of the given points to the left of it is equal to the number of the given points to the right of it.

Note that the middle point $x_{\lceil n/2 \rceil}$—the $\lceil n/2 \rceil$th smallest among $x_1, \ldots, x_n$—solves the problem for even n's as well.

Comments The solution takes advantage of transform-and-conquer by reducing the puzzle to two instances of the mathematical problem of finding a middle value among n given numbers. Mathematicians call such a value the *median*; it plays a very important role in statistics. Computing the median efficiently is a task called the *selection problem*. It has a straightforward solution, of course: sort the numbers in nondecreasing order and take the $\lceil n/2 \rceil$th element in the sorted list. For faster but more sophisticated algorithms, see, for example, [Lev06, pp. 188–189] and [Cor09, Sections 9.2 and 9.3].

75. Gas Station Inspections

Solution Since the problem requires visiting station n twice, every legitimate tour will have to visit station $n - 1$ at least twice as well. Hence, it will also have to visit each of the other intermediate stations at least twice. Taking into account that station 1 has to be visited twice by the puzzle's statement, the total number of station visits will be at least $2n$. Therefore, the total length of any tour satisfying the requirements will be at least $(2n - 1)d$, where d is the distance between two neighboring stations. If n is even, the tour that returns back to the visited neighbor from each even numbered station has the length equal to this lower bound:

$$1, 2, 1, 2, 3, 4, 3, 4, \ldots, n - 1, n, n - 1, n.$$

This tour for $n = 8$ is shown in Figure 4.48.

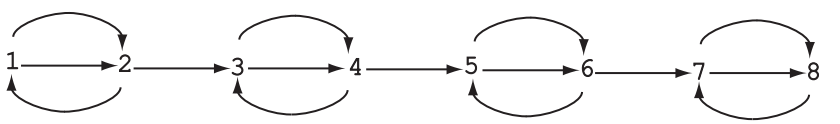


FIGURE 4.48 Solution to the *Gas Station Inspections* puzzle for $n = 8$.

Let us prove by induction that if n is odd, there is no tour satisfying the requirements that visits intermediate stations $2, \ldots, n-1$ exactly twice. For the base case of $n = 3$, it is easy to verify directly that a required tour has to visit station 2 at least three times. For the general case, assume that there is no tour satisfying the requirements that visits intermediate stations $2, \ldots, n-1$ exactly twice, where $n \geq 3$ is odd. We will show by contradiction that this is also the case for $n+2$ stations. If such a tour existed, it would have to end at station $n+2$; otherwise, it would have to visit station $n+1$ at least three times. Further, the tour would have to end with $n+1, n+2, n+1, n+2$ and would have neither station $n+1$ nor station $n+2$ among the stations visited before this last segment. But then this preceding segment of the tour would be a legitimate tour for the problem of visiting n stations, with each of the intermediate stations $2, \ldots, n-1$ visited twice. This contradicts the inductive assumption and completes the proof.

Thus, when n is odd, any tour satisfying the requirements must visit intermediate stations at least three times. This implies that for odd values of n, the obvious tour

$$1, 2, \ldots, n-1, n, n-1, \ldots, 2, 1, 2, \ldots, n-1, n$$

is optimal.

Comments The puzzle is a generalization of Henry Dudeney's "Stepping Stones" in [Dud67, Problem 522], and Sam Loyd's "Hod Carrier's Problem" in [Loy59, Problem 88]. Both of them considered the $n = 10$ instance of the puzzle.

76. Efficient Rook

Solution The minimum number of moves needed is $2n-1$ for any $n > 1$.

An optimal tour can start in the top left corner and, following the greedy strategy, proceed as far as it can before making a turn. The resulting tour for the 8×8 board is shown in Figure 4.49a. The number of moves in such a tour on the $n \times n$ board is equal to $2n-1$.

Let us prove now that any rook tour that passes over all the squares of an $n \times n$ board where $n > 1$ comprises at least $2n-1$ moves. Indeed, any such tour will have to contain moves either along every column or along every row of the board. (If it does not contain a move along some column, then each square of that column has to be passed over or landed on by a row move, and if it does not contain a move along some row, then each square of that row has to be passed over or landed on by a column move.) Hence, such a tour will have to contain either n vertical moves interspersed with $n-1$ horizontal moves needed to switch to another column, or n horizontal moves interspersed with $n-1$ vertical moves needed to switch to another row. This proves that any rook tour that passes through all the squares of the $n \times n$ board has to make at least $2n-1$ moves, which proves optimality of the tour described above.

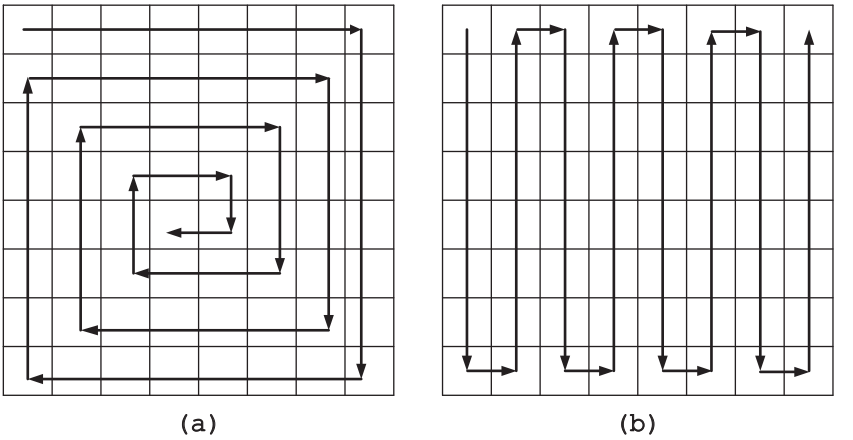


FIGURE 4.49 Rook's minimum-move paths through the 8×8 board. (a) A greedy solution. (b) An alternative solution.

The above solution is not unique. One of alternative solutions for the 8×8 board is shown in Figure 4.49b; of course, a similar tour exists on any $n \times n$ board where $n > 1$.

Comments The first solution given above is based on the greedy strategy. As it is often the case with greedy algorithms, it is a proof of the algorithm's correctness rather than the algorithm itself that constitutes the main difficulty here.

The problem is discussed in E. Gik's *Mathematics on a Chessboard* [Gik76, p. 72].

77. Searching for a Pattern

Solution If the numbers are considered as decimals, the first nine products do exhibit an obvious pattern:

$$1 \times 1 = 1,\ 11 \times 11 = 121,\ 111 \times 111 = 12321,$$
$$1111 \times 1111 = 1234321, \ldots, 111111111 \times 111111111 = 12345678987654321.$$

But after that it breaks down because of the digit carryovers:

$$1{,}111{,}111{,}111 \times 1{,}111{,}111{,}111 = 1234567900987654321, \text{ and so on.}$$

(Of course, this does not preclude a possibility that some pattern may eventually appear as the number of 1's increases.)

However, if the numbers are considered as binary,

$$\underbrace{11\ldots1}_{k} \times \underbrace{11\ldots1}_{k} = \underbrace{11\ldots1}_{k-1}\underbrace{00\ldots0}_{k}1,$$

where we assume that 01 is 1 for $k = 1$. Indeed, $\underbrace{11\ldots1}_{k} = 2^k - 1$ and therefore

$$\underbrace{11\ldots1}_{k} \times \underbrace{11\ldots1}_{k} = (2^k - 1)^2 = 2^{2k} - 2 \cdot 2^k + 1 = 2^{2k} + 1 - 2^{k+1}.$$

Since

$$2^{2k} + 1 = 1\underbrace{00\ldots00\ldots01}_{k\quad k} \text{ and } 2^{k+1} = 1\underbrace{00\ldots0}_{k+1},$$

we obtain

$$2^{2k} + 1 - 2^{k+1} = \underbrace{11\ldots1}_{k-1}\underbrace{00\ldots0}_{k}1.$$

Comments The decimal version of the puzzle is from [Ste09, p. 6].

78. Straight Tromino Tiling

Solution A tiling in question is always possible.

Consider first the case of $n \bmod 3 = 1$ and $n > 3$, that is, $n = 4 + 3k$ where $k \geq 0$. We can divide the square into three subregions (note the divide-and-conquer idea in action!): the 4×4 square in, say, the upper left corner, the $4 \times 3k$ rectangle, and $3k \times (4 + 3k)$ rectangle (see Figure 4.50a). The 4×4 square requires a monomino placed in one of its corners to make it possible to tile the rest of it; tiling the other two rectangles (if $k > 0$) is trivial since both of them have a side equal to $3k$.

Similarly, if $n \bmod 3 = 2$ and $n > 3$, that is, $n = 5 + 3k$ where $k \geq 0$, we can tile the board as shown in Figure 4.50b.

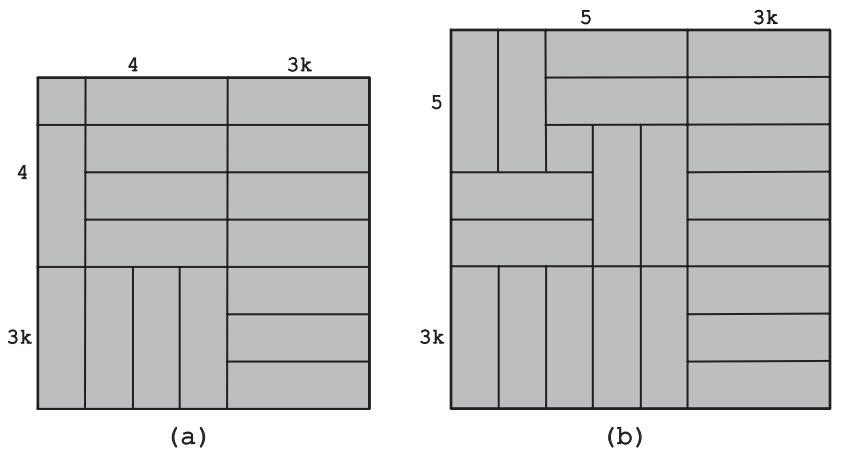


FIGURE 4.50 Tiling an $n \times n$ square with straight trominoes and one monomino. (a) $n = 4 + 3k$. (b) $n = 5 + 3k$.

Comments This tiling algorithm can also be considered an application of the decrease-by-three strategy: there is a simple way to tile an $n \times n$ square once an $(n-3) \times (n-3)$ square has already been tiled.

The problem of tiling the 8×8 checker board with straight trominoes was introduced in the seminal paper on polyomino tiling by Solomon Golomb [Gol54]. In particular, he proved that the board's tiling is possible if and only if the monomino is placed in one of the four positions shown in Figure 4.51.

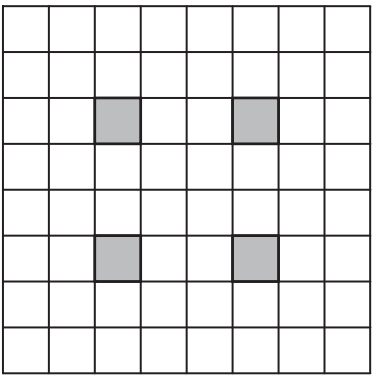


FIGURE 4.51 Four possible monomino positions (shown in gray) for straight tromino tilling of an 8×8 board.

79. Locker Doors

Solution The number of open doors after the nth pass is $\lfloor \sqrt{n} \rfloor$.

Since all the doors are initially closed, a door will be open after the last pass if and only if it is toggled an odd number of times. Door $i (1 \le i \le n)$ is toggled on pass $j (1 \le j \le n)$ if and only if j divides i; therefore, the total number of times door i is toggled is equal to the number of its divisors. Note that if j divides i, that is, $i = jk$, then k divides i too. Hence, all the divisors of i can be paired (e.g., for $i = 12$, such pairs are 1 and 12, 2 and 6, 3 and 4) unless i is a perfect square (e.g., for $i = 16$, 4 does not have another divisor to be matched with). This implies that i has an odd number of divisors if and only if it is a perfect square, that is, $i = l^2$ where l is a positive integer. Hence, doors that are in the positions that are perfect squares and only such doors will be open after the last pass. The total number of such positions not exceeding n is equal to $\lfloor \sqrt{n} \rfloor$: these numbers are the squares of the positive integers between 1 and $\lfloor \sqrt{n} \rfloor$, inclusive.

Comments The puzzle was published in the April 1953 issue of the *Pi Mu Epsilon Journal* (p. 330). Since then, it appeared in many other puzzle collections both in print (e.g., [Tri85, Problem 141]; [Gar88b, pp. 71–72]) and on the Internet.

80. The Prince's Tour

Solution The puzzle has a solution for any n.

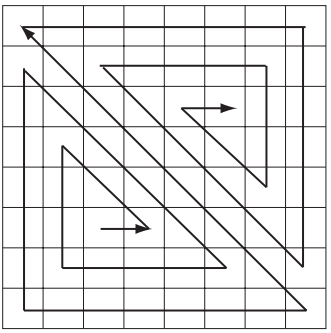


FIGURE 4.52 Prince's tour of an 8×8 board.

A tour of an 8×8 board by a prince is shown in Figure 4.52.

It can be considered composed of three parts: the main diagonal of the board from its lower right to upper left corner, the spiral-like path from the upper left corner to the end point that passes through each square above the main diagonal exactly once, and the symmetric spiral-like path that passes through each square below the main diagonal.

It is easy to see that such a path can be constructed for any $n \times n$ board where $n > 1$, and the puzzle also has the trivial solution for $n = 1$.

The solution to the puzzle is not unique. Figure 4.53 shows an alternative solution for $n = 6$, 7, and 8, representing the general cases of $n = 3k$, $n = 3k + 1$, and $n = 3k + 2$.

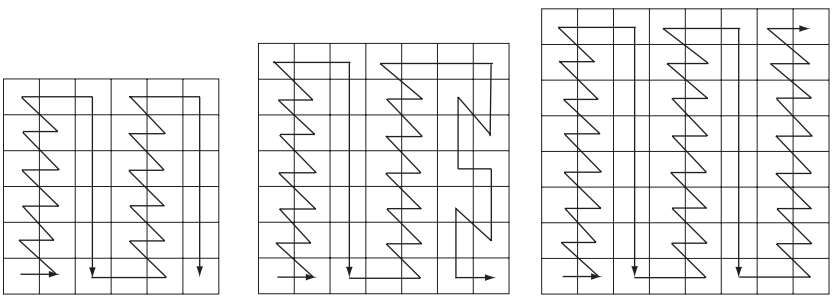


FIGURE 4.53 Prince's tour of an $n \times n$ board for $n = 3k$, $n = 3k + 1$, and $n = 3k + 2$.

Comments Both solutions can be considered applications of the divide-and-conquer strategy.

The puzzle was motivated by a question about existence of a closed (i.e., re-entrant) prince tour on a 10×10 board, which was included in a collection of problems for Russian physics and mathematics schools [Dyn71, Problem 139].

81. Celebrity Problem Revisited

Solution The problem can be solved by an algorithm asking no more than $3n - 4$ questions for $n > 1$.

Unlike the simpler version discussed in the book's first tutorial, we cannot assume here that there is a celebrity in the group. Still, the same algorithmic idea works as follows. If $n = 1$, the one person is a celebrity by the definition. If $n > 1$, select two people from the group, say, A and B, and ask A whether A knows B. If A knows B, remove A from the remaining people who can be a celebrity; if A does not know B, remove B from this group. Solve the problem recursively for the remaining group of $n - 1$ people who can still be a celebrity. If the solution returned indicates that there is no celebrity among the group of $n - 1$ people, the larger group of n people cannot contain a celebrity either because the person removed after the first question is known not to be a celebrity. If the identified celebrity is a person other than either A or B, say, C, ask whether C knows the person removed after the first question and, if the answer is "no," whether the person removed after the first question knows C. If the answer to the second question is "yes," return C as a celebrity, and "no celebrity" otherwise. If the celebrity found among the group of $n - 1$ people is B, just ask whether B knows A: return B as a celebrity if the answer is "no," and "no celebrity" otherwise. If the celebrity found among the group of $n - 1$ people is A, ask whether B knows A: return A as a celebrity if the answer is "yes," and "no celebrity" otherwise.

The recurrence for $Q(n)$, the number of questions needed in the worst case, is as follows:

$$Q(n) = Q(n-1) + 3 \quad \text{for } n > 2, \quad Q(2) = 2, \quad Q(1) = 0.$$

It can be solved by forward substitutions, backward substitutions, or by using the formula for the generic term of the arithmetical progression. The solution is $Q(n) = 2 + 3(n-2) = 3n - 4$ for $n > 1$ and $Q(1) = 0$.

Comments The algorithm provides an excellent example of decrease-by-one problem solving. Udi Manber discussed the algorithm and its computer implementation in his book [Man89, Section 5.5]. He traced the problem's origin to S. O. Aanderaa and also pointed to a paper by King and Smith-Thomas [Kin82] that further improves the algorithm described above.

82. Heads Up

Solution The minimum number of moves needed to solve the puzzle in the worst case is $\lceil n/2 \rceil$.

We can think of the coin line as composed of alternating blocks of heads and tails, where a block may have as few as one coin and as many as n coins of the same type. Flipping any number of successive coins can decrease the number of tail blocks by no more than 1 since flipping more than one such block also flips all the head blocks between them. Therefore, the number of moves needed to get to the zero tail blocks must be at least as large as the number of tail blocks in the initial line. That number may be as small as zero (in a line of all heads) and as large $\lceil n/2 \rceil$ (in a line of alternating tails and heads that starts with a tail). An algorithm that

solves the puzzle in the minimum number of moves can simply flip all the coins in the first tail block in the current line until no tails are left; it will need $\lceil n/2 \rceil$ iterations in the worst case.

Comments The algorithm suggested above can be considered an application of the decrease-by-one strategy. This puzzle was also used by L. D. Kurlandchik and D. B. Fomin in their *Kvant* article on monovariants [Kur89]. As the monovariant, they considered the number of TH and HT pairs in the coin line, which cannot change by more than two on any move. The 100-coin instance of this puzzle for the HT. . . HT line was also included in [Fom96, p. 194, Problem 90].

83. Restricted Tower of Hanoi

Solution The minimum number of moves needed to solve the puzzle is $3^n - 1$.

If $n = 1$, move the single disk from the source peg first to the middle peg and then from there to the destination peg. If $n > 1$, do the following:

- Transfer recursively the top $n - 1$ disks from the source peg to the destination peg through the middle peg.
- Move the bottom disk from the source peg to the middle peg.
- Transfer recursively $n - 1$ disks from the destination peg to the source peg through the middle peg.
- Move the disk from the middle peg to the destination peg.
- Transfer recursively $n - 1$ disks from the source peg to the destination peg through the middle peg.

The algorithm is illustrated in Figure 4.54.

The recurrence relation for the number of moves $M(n)$ is

$$M(n) = 3M(n-1) + 2 \quad \text{for } n > 1, \quad M(1) = 2.$$

The first few values of $M(n)$ are given in the following table:.

n	$M(n)$
1	2
2	8
3	26
4	80

They suggest the formula $M(n) = 3^n - 1$ for the solution, which can be verified by the substitution into the recurrence:

$$M(n) = 3^n - 1 \quad \text{and} \quad 3M(n-1) + 2 = 3(3^{n-1} - 1) + 2 = 3^n - 1.$$

Alternatively, the recurrence can be solved by the standard method of backward substitutions explained, for example, in [Lev06, Section 2.4].

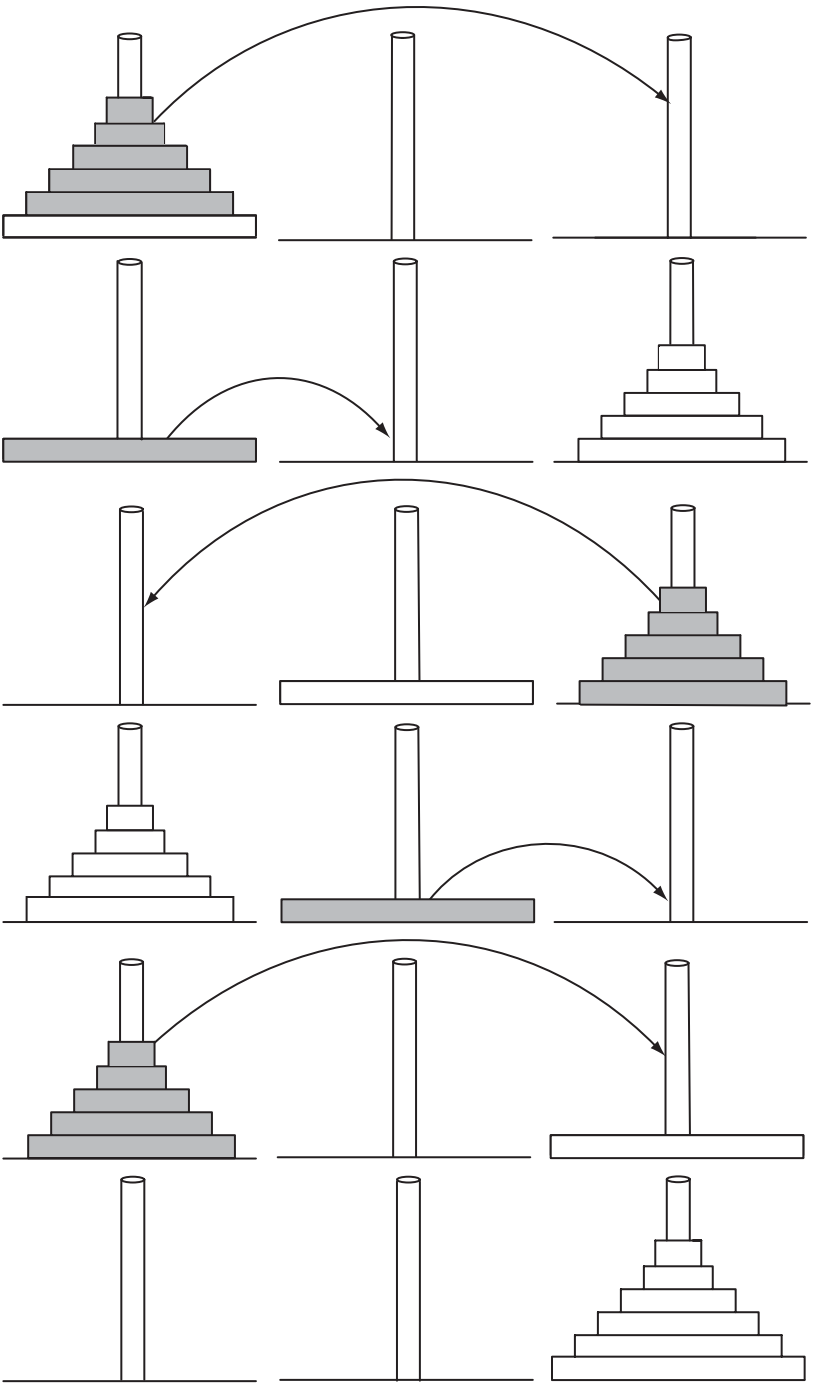


FIGURE 4.54 Recursive algorithm for the *Restricted Tower of Hanoi* puzzle.

It is not difficult to prove that the above algorithm solves the problem in the fewest moves possible. Let $A(n)$ be the number of disk moves made by some algorithm solving the puzzle. We will prove by induction that

$$A(n) \geq 3^n - 1 \text{ for } n \geq 1.$$

For the basis case of $n = 1$, $A(1) \geq 3^1 - 1$ holds. Assume now that the inequality holds for $n \geq 1$ disks and consider the case of $n + 1$ disks. Before the largest disk can be moved, all n smaller disks must be in a tower on the destination peg. By the inductive assumption, it requires at least $3^n - 1$ disk moves. Moving the largest disk to the middle peg takes at least one move. Then before the largest disk can be moved on the destination peg, all n smaller disks must be in a tower on the left peg; by the inductive assumption, it requires at least $3^n - 1$ disk moves to transfer them there. Moving the largest disk from the middle peg to the right peg requires at least one move, and transferring the $n - 1$ disks from the left peg to the right peg requires at least $3^n - 1$ more disk moves yet again. To summarize, the total number of moves made by the algorithm must satisfy the inequality

$$A(n + 1) \geq (3^n - 1) + 1 + (3^n - 1) + 1 + (3^n - 1) = 3^{n+1} - 1,$$

which completes the proof.

Comments The classic version of the *Tower of Hanoi* puzzle allows direct moves between the left and right pegs; as shown in the second tutorial, this makes it possible to solve the puzzle in the minimum number of $2^n - 1$ moves. The above algorithm solves the puzzle in the maximum number of moves that do not repeat the same configuration of the disks (see the "Tower of Hanoi, the Hard Way" page from [Bogom]).

As far as the underlying design strategy is concerned, the algorithm is definitely based on the decrease-by-one approach, but unlike the standard version of this strategy, it solves three rather than two instances of size $n - 1$.

This version of the *Tower of Hanoi* puzzle was mentioned in the 1944 paper by R. S. Scorer et al. [Sco44].

84. Pancake Sorting

Solution The problem can be solved in $2n - 3$ flips where $n \geq 2$ is the number of pancakes given.

The decrease-and-conquer strategy leads to the following outline of an algorithm. Repeat the following step until the problem is solved: bring the largest pancake not yet in its final position to the top with one flip, and then take it down to its final position with one more flip. Here is a more detailed implementation of this outline.

Initialize k, the number of pancakes at the bottom of the stack that are in their final positions to 0. Repeat the following until $k = n - 1$, that is, until the problem

is solved. Find the largest pancake above the kth from the bottom that is larger than the one immediately below it. (If there is no such pancake, the problem is solved.) If this largest pancake is not on the top of the stack, slide the spatula under it and flip to put it on the top. Starting with the $(k+1)$th pancake from the bottom, scan the stack up to find the first pancake that is smaller than the top pancake. Let it be the jth pancake from the bottom. (Note that all the pancakes from the $(k+1)$th to the jth, inclusive, are in proper order because the pancake currently at the top has been selected as the largest out-of-order pancake.) Slide the spatula under the jth pancake and flip to increase the number of pancakes in their final place at least by one. Finally, update the value of k by replacing it by j.

The first iteration of the algorithm on the instance in Figure 2.20 is shown in Figure 4.55.

The number of flips made by this algorithm in the worst case is $W(n) = 2n - 3$, where $n \geq 2$ is the number of pancakes. (Obviously, $W(1) = 0$.) This formula stems from the following recurrence relation:

$$W(n) = W(n-1) + 2 \quad \text{for } n > 2, \qquad W(2) = 1.$$

The initial condition $W(2) = 1$ is true because the algorithm solves the problem in one flip if the larger of the two pancakes is on the top and it makes no flips if the larger of the two pancakes is on the bottom. Consider an arbitrary stack of $n > 2$ pancakes. With two flips or less, the algorithm puts the largest pancake at the bottom of the stack, where it does not participate in any further flips. Hence, the total number of flips needed for any stack of n pancakes is bounded above by $W(n-1) + 2$. In fact, this upper bound is attained on the stack of n pancakes constructed as follows. Flip a worst-case stack of $n - 1$ pancakes upside down and insert a pancake larger than all the others right below the top pancake. On the new stack, the algorithm will make two flips to reduce the problem to flipping the worst-case stack of $n - 1$ pancakes.

Since the above recurrence relation defines an arithmetical progression, we get the following explicit formula for its nth element:

$$W(n) = 1 + 2(n-2) = 2n - 3 \quad \text{for } n \geq 2.$$

Comments The above algorithm is an excellent example of the decrease-and-conquer strategy, where the problem's size decreases irregularly, that is, neither by a constant nor by a constant factor. It is not optimal, however. The minimum number of flips needed in the worst case lies between $(15/14)n$ and $(5/3)n$, but its exact value is not known.

There is a visualization applet of the puzzle on the "Flipping pancakes" page of the *Interactive Mathematics Miscellany and Puzzles* website by Alexander Bogomolny [Bogom]. One can also find there a few interesting facts about this puzzle. It mentions, in particular, that the only research paper published by Microsoft's founder Bill Gates was devoted to this problem.

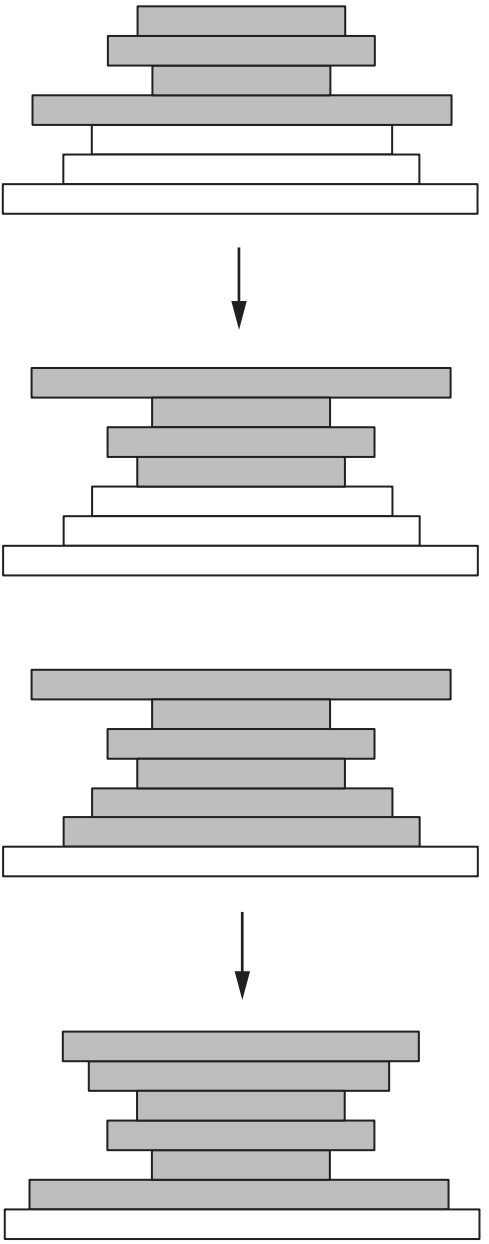


FIGURE 4.55 Two flips of the first iteration of the *Pancake Sorting* algorithm.

85. Rumor Spreading I

Solution The minimum number of messages is equal to $2n - 2$.

There are several ways to do this. For example, they can designate one person, say, person 1, to whom everybody else sends the message with the rumor they know. After receiving all these messages, person 1 combines all the rumors with his or her rumor and send this combined message to each of the other $n - 1$ persons.

The same number of messages is required by the following greedy algorithm, which seeks to increase as much as possible the total number of known rumors after each message sent. Number the persons from 1 to n and send the first $n-1$ messages as follows: from 1 to 2, from 2 to 3, and so on, until the message combining the rumors initially known to persons $12, \ldots, n-1$ is sent to person n. Then send the message combining all the n rumors from person n to persons $1, 2, \ldots, n-1$.

The fact that $2n-2$ messages is the smallest number needed to solve the puzzle stems from the fact that an increase of the number of persons by one requires at least two extra messages: to and from the extra person—exactly what the above algorithms provide.

Comments The puzzle was offered at a Canadian mathematical olympiad in 1971 [Ton89, Problem 3]. Problems similar to this puzzle are obviously of great importance to specialists dealing with communication networks.

86. Rumor Spreading II

Solution For $n = 1$, 2, and 3, the solutions—requiring 0, 1, and 3 conversations, respectively—are obvious. Here is one of several algorithms that achieve the puzzle's goal with $2n-4$ conversations for any $n \geq 4$. For $n = 4$, four conversations are needed: for example, P_1 with P_2, P_3 with P_4, P_1 with P_4, and P_2 with P_3. For $n > 4$, the solution for $n = 4$ can be extended by making each of persons $P_5, P_6, \ldots, P_n$ talk with person P_1 before P_1 talks with P_2, P_3 with P_4, P_1 with P_4 and P_2 with P_3 and then making P_1 talk with each of persons $P_5, P_6, \ldots, P_n$ for the second time. The algorithm is illustrated in Figure 4.56. The total number of conversations in this algorithm is equal to $2(n-4)+4 = 2n-4$ where $n \geq 4$.

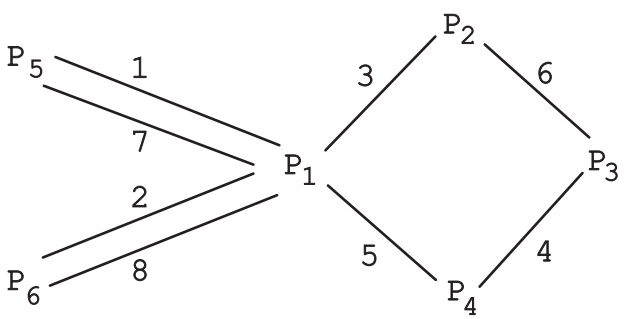


FIGURE 4.56 Optimal rumor spreading via bilateral conversations for $n = 6$: the sequential order of conversations is indicated by the labels on the edges connecting the vertices representing the conversing parties.

Comments The algorithm given above exploits the decrease-and-conquer idea bottom up by expanding a solution for $n = 4$. Although it is not difficult to devise

an algorithm that solves the puzzle with $2n - 4$ conversations, it is much harder to prove that this number is the minimum for $n \geq 4$. For a proof, see the article by C. A. J. Hurkens [Hur00], which also contains the above algorithm and some further references. D. Niederman gives another algorithm for the puzzle in his book [Nie01, Problem 55], prefacing the puzzle's statement by saying that the puzzle "is destined to become a classic."

87. Upside Down Glasses

Solution There is no solution for any odd n; if n is even, the problem can be solved in n moves, which is the fewest number of moves required.

If n is odd, $n - 1$, the number of glasses turned over in one move, is even. Therefore, the parity of the number of glasses that are upside down will always remain odd, as it was in the initial position. Hence, the final position in which the number of upside down glasses is 0, which is even, cannot be reached.

If n is even, the problem can be solved by making the following move n times: turn over all the glasses except the ith glass, where $i = 1, 2, \ldots, n$. (We assume that the glasses are numbered from 1 to n.)

Here is an example of the algorithm's application for $n = 6$. The upside down glasses are denoted by 1's, the others are denoted by 0's; a glass that is not turned over on the next move is shown in bold.

$$\mathbf{1}11111 \rightarrow 1\mathbf{0}0000 \rightarrow 00\mathbf{1}111 \rightarrow 111\mathbf{0}00 \rightarrow 0000\mathbf{1}1 \rightarrow 11111\mathbf{0} \rightarrow 000000$$

Since any two consecutive moves will either have no impact on the state of the glasses or change the state of exactly two of them, no algorithm can solve the problem in fewer than n moves.

Comments The puzzle is one of several versions of the well-known problem in which some objects can be in one of two states and the goal is to transform them from one state to the other. Their solutions often exploit parity and, when the problem can be solved, the decrease-and-conquer strategy.

The puzzle was included by Charles Trigg in his *Mathematical Quickies* [Tri85, Problem 22].

88. Toads and Frogs

Solution We can find the number of slides and jumps in an algorithm solving the puzzle as follows. The only way a Toad and a Frog can pass each other is with a jump. (Which of the creatures jumps over which does not matter here.) Hence, n^2 jumps are necessary for n Toads and n Frogs to pass each other. Further, since Toads cannot jump over each other, the first Toad has to move $n + 1$ cells to end up in cell $n + 2$, the second Toad has to move $n + 1$ cells to end up in cell $n + 3$, and so on. Added together, all the Toads have to move $n(n + 1)$ cells to their final positions. By the similar reasoning, the Frogs have to move $n(n + 1)$

cells to their final positions as well. Added together, all the creatures have to move $2n(n+1)$ cells. Since one jump covers two cells, and there are n^2 of them, the number of slides is equal to $2n(n+1) - 2n^2 = 2n$.

There are two symmetric algorithms for solving the puzzle: one starting with the slide of the last Toad and the other starting with the slide of the first Frog. Without loss of generality, we will describe the former one. The algorithm can be obtained by essentially brute-force thinking: its moves are uniquely defined because the alternative moves lead to obvious dead ends. In particular, whenever there is a choice between a slide and a jump, the jump must be made. The algorithm is illustrated in Figure 4.57 and Figure 4.58 for $n = 2$ and $n = 3$, respectively. They show the board's states and the moves being made. S and J indicate a slide and a jump while the subscripts T and F indicate a move by a Toad and a Frog, respectively. A jump, in fact, is always uniquely defined, and therefore does not require a subscript indicator.

	Cells					
Move #	1	2	3	4	5	Move
1	T	T		F	F	S_T
2	T		T	F	F	J
3	T	F	T		F	S_F
4	T	F	T	F		J
5	T	F		F	T	J
6		F	T	F	T	S_F
7	F		T	F	T	J
8	F	F	T		T	S_T
	F	F		T	T	

FIGURE 4.57 Solution to the *Toads and Frogs* puzzle for $n = 2$.

In general, the algorithm can be described by the following string of $2n + n^2$ letters indicating the moves made:

$$S_T\, J\, S_F\, J\, J \ldots S \underbrace{J \ldots J}_{n-1} S\, |\, \underbrace{J \ldots J}_{n}\, |\, S \underbrace{J \ldots J}_{n-1} \ldots J\, J\, S_F\, J\, S_T$$

The above string is a palindrome, that is, it reads the same left to right and right to left. The moves in the left part of the string (left from the vertical line) make Toads and Frogs alternate in the TFTF. . . TF pattern, followed or preceded by the empty cell for even and odd n's, respectively. It is composed of n S's (slides) with the subscripts alternating between T (Toad's slide) and F (Frog's slide), which are interposed with groups of J's (jumps) whose number increases from 1 to $n - 1$. The central part of the string transforms the pattern into FTFT. . . FT; it is composed of n J's. The right part of the string finishes the task by the moves of the left part performed in reverse order.

	Cells							
Move #	1	2	3	4	5	6	7	Move
1	T	T	T		F	F	F	S_T
2	T	T		T	F	F	F	J
3	T	T	F	T		F	F	S_F
4	T	T	F	T	F		F	J
5	T	T	F		F	T	F	J
6	T		F	T	F	T	F	S_T
7		T	F	T	F	T	F	J
8	F	T		T	F	T	F	J
9	F	T	F	T		T	F	J
10	F	T	F	T	F	T		S_T
11	F	T	F	T	F		T	J
12	F	T	F		F	T	T	J
13	F		F	T	F	T	T	S_F
14	F	F		T	F	T	T	J
15	F	F	F	T		T	T	S_T
	F	F	F		T	T	T	

FIGURE 4.58 Solution to the *Toads and Frogs* puzzle for $n = 3$.

Comments The puzzle can be easily generalized to the one with m Toads and n Frogs. Then the total number of moves required is equal to $mn + m + n$, of which mn are jumps and $m + n$ are slides. In other variations of the puzzle, the number of empty cells separating Toads from Frogs may be greater than 1.

Ball and Coxeter [Bal87, p. 124] reference the book by Lucas [Luc83, pp. 141–143] as the puzzle's source. For other references to this puzzle, whose alternative names have included *Sheep and Goats* and *Hares and Tortoises*, see David Singmaster's annotated bibliography [Sin10, sec. 5.R.2]. Alexander Bogomolny's website [Bogom] provides a good visualization applet of the puzzle and a discussion of its solution on the "Toads and Frogs puzzle: theory and solution" page.

89. Counter Exchange

Solution The puzzle is a two-dimensional version of the *Toads and Frogs* puzzle (#88). It can be solved by applying the algorithm for that puzzle to the middle column. Whenever this algorithm creates a vacant cell in the board's row for the first time, we can switch to exchanging the counters in that row by applying the same algorithm. Thus, the algorithm for solving the one-dimensional *Toads and Frogs* puzzle will have to be applied the total of $(2n + 2)$ times: once for each row and once for the central column. Since that algorithm makes n^2 jumps and $2n$ slides on a line with $2n + 1$ cells, the two-dimensional algorithm will make $n^2(2n + 2)$ jumps and $2n(2n + 2)$ slides, for the total of $2n(n + 1)(n + 2)$ moves.

Comments The solution is clearly based on the transform-and-conquer strategy. In general, a reduction of a two-dimensional version of a problem to its one-dimensional counterpart is a common problem-solving method. Of course, such a reduction is not always possible.

Ball and Coxeter [Bal87, p. 125] reference the book by Lucas [Luc83, p. 144] as the puzzle's source. For other references, see the "Frogs and Toads" section in David Singmaster's annotated bibliography [Sin10]. Alexander Bogomolny's website [Bogom] provides a good visualization applet for this puzzle on its "Toads and Frogs puzzle in two dimensions" page.

90. Seating Rearrangements

Solution After numbering the children from 1 to n in the order of their initial seating, the problem is reduced to generating all permutations of $1, 2, \ldots, n$ by transposing two adjacent elements. These permutations can be obtained by first generating all permutations of $1, 2, \ldots, n-1$ recursively and then inserting n into all possible positions in each permutation of $1, 2, \ldots, n-1$. To make sure that every pair of consecutive permutations differs only by transposition of two adjacent elements, we have to insert n in the alternating directions.

We can insert n into the previously generated permutations either left to right or right to left. It turns out that it is beneficial to start with inserting n into $12\ldots(n-1)$ by moving right to left and then switch direction every time a new permutation of $1, 2, \ldots, n-1$ needs to be processed. An example of applying this approach bottom up for $n \leq 4$ is given in Figure 4.59.

Start	1
Insert 2 into 1 right to left	12 21
Insert 3 into 12 right to left	123 132 312
Insert 3 into 21 left to right	321 231 213
Insert 4 into 123 right to left	1234 1243 1423 4123
Insert 4 into 132 left to right	4132 1432 1342 1324
Insert 4 into 312 right to left	3124 3142 3412 4312
Insert 4 into 321 left to right	4321 3421 3241 3214
Insert 4 into 231 right to left	2314 2341 2431 4231
Insert 4 into 213 left to right	4213 2413 2143 2134

FIGURE 4.59 Generating permutations bottom up.

Comments The algorithm is a perfect illustration of the decrease-by-one strategy. The nonrecursive version of this algorithm is known in computer science as the Johnson-Trotter algorithm, after two researchers who independently published it around 1962. According to Martin Gardner [Gar88b, p. 74], the algorithm was, in fact, discovered by the Polish mathematician Hugo Steinhaus to solve the abacus problem [Ste64, Problem 98]. The problem of generating

permutations by transposition of adjacent elements has sometimes been referred to as *Lehmer's Motel Problem* after the paper by D. H. Lehmer [Leh65], who considered a more general problem of permuting numbers that may not all be distinct.

91. Horizontal and Vertical Dominoes

Solution A required tiling is possible if and only if n is divisible by 4.

If n is odd, the $n \times n$ board has an odd number of squares. Therefore, no domino tiling of such a board is possible, because any domino tiling covers an even number of squares.

If n is divisible by 4, that is, $n = 4k$, the board can be divided into $4k^2$ 2×2 squares. Since $4k^2$ is an even number, one can tile one half of the 2×2 squares with horizontal dominoes and the other half with vertical dominoes to get a desired tiling.

If n is even but not divisible by 4, that is, $n = 2m$ where m is odd, no tiling of an $n \times n$ board with an equal number of horizontal and vertical dominoes is possible. To prove this, we color the board's rows with two alternating colors (see Figure 4.60 for an example). Note that any horizontal tile would cover two squares of the same color, and any vertical tile would cover two squares of different colors. We want to cover the board's $n^2 = 4m^2$ cells with t horizontal tiles and t vertical tiles, where $t = m^2$. The colored board has $2m^2$ squares of one color and $2m^2$ squares of the other color. The vertical dominoes would cover m^2 squares of each color, leaving the remaining m^2 squares of each color to be covered by horizontal dominoes. But this is impossible for either of the two colors because m^2 is odd whereas the number of squares covered by dominoes must be even.

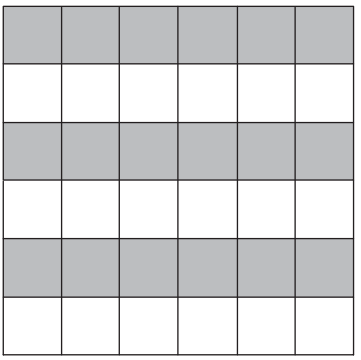


FIGURE 4.60 Coloring of an $n \times n$ board where n is even but not divisible by 4.

Comments The solution is typical for tiling puzzles. When it is possible, a tiling is obtained by partitioning the board into regions that are easy to tile. When it is impossible, an invariant argument is used, often based on either parity or board coloring. Such examples are given in the book's tutorial on algorithm analysis techniques.

This tiling puzzle is well-known: see, for example, a similar problem in Arthur Engel's *Problem-Solving Strategies* [Eng99, p. 26, Problem 9].

92. Trapezoid Tiling

Solution The puzzle has a solution if and only if n is not divisible by 3.

Counting the small triangles by the layers starting with the base of the triangular region (Figure 4.61), we get the following formula:

$$\begin{aligned} T(n) &= [n + 2(n-1) + 2(n-2) + \cdots + 2 \cdot 1] - 1 \\ &= n + 2(n-1)n/2 - 1 = n^2 - 1. \end{aligned}$$

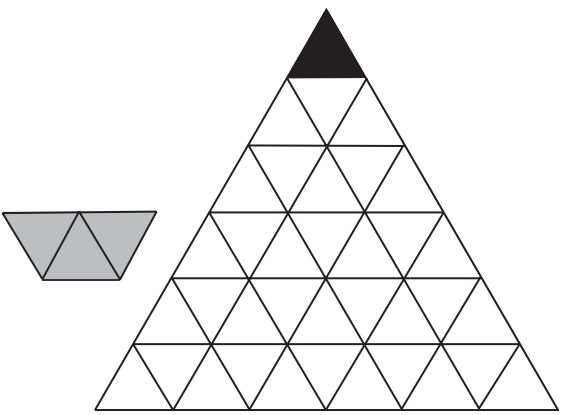


FIGURE 4.61 Region to tile with trapezoid tiles (in gray) for $n = 6$.

Since one trapezoid tile is made up of three small triangles, $n^2 - 1$ must be divisible by 3 for a tiling to exist. But $n^2 - 1$ is divisible by 3 if and only if n is not divisible by 3, which immediately is established by considering the cases of $n = 3k$, $n = 3k+1$, and $n = 3k + 2$.

Before we show that this condition is not only necessary but also sufficient for a tiling existence, let us prove that if $n = 3k$ and no small triangle is removed from the region, it can be tiled by trapezoids. This is easy to prove by induction on k. For $k = 1$, the region can be tiled by three trapezoids (Figure 4.62a). Let us prove that if a trapezoid tiling exists for $n = 3k$ where $k \geq 1$, it is also exists for $n = 3(k + 1)$. Consider the line parallel to the base through the points dividing the region's sides $3 : 3k$ (Figure 4.62b). This line dissects the region into the trapezoid below this line and the equilateral triangle above the line. The former can be dissected into $(k + 1) + k$ equilateral triangles of side size 3 and hence can be tiled by trapezoid tiles; the latter can be tiled by the inductive assumption.

Now consider the case of an equilateral triangle of side size $n = 3k + 1$ with its topmost equilateral triangle chopped off. After placing $2k$ trapezoids along one of the sides of the region (see Figure 4.63a for an example), we will be left with the task of tiling the whole equilateral triangle of side size $3k$, which can be

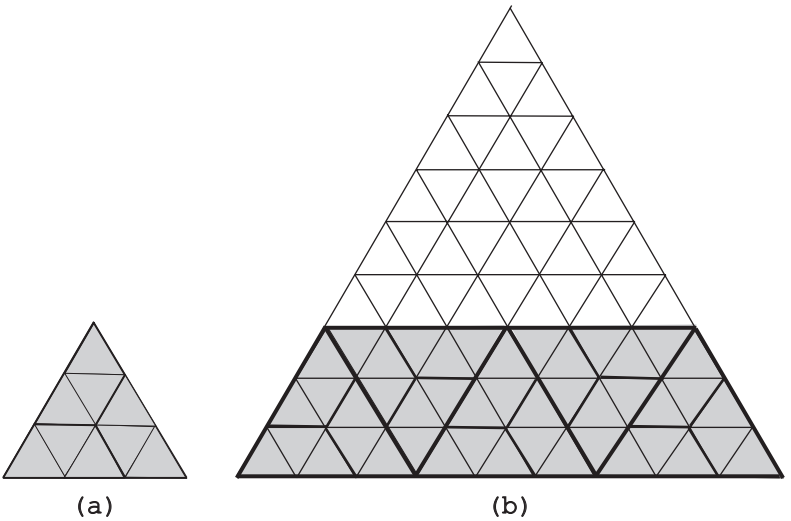


FIGURE 4.62 (a) Trapezoid tiling of the whole triangular region for $n = 3$. (b) Recursive tiling of a whole triangular region for $n = 3k$, $k > 1$.

done as shown above. And if $n = 3k + 2$, we can place $2k + 1$ trapezoids along the region's base to reduce this instance to that of $n = 3k + 1$ (see Figure 4.63b for an example).

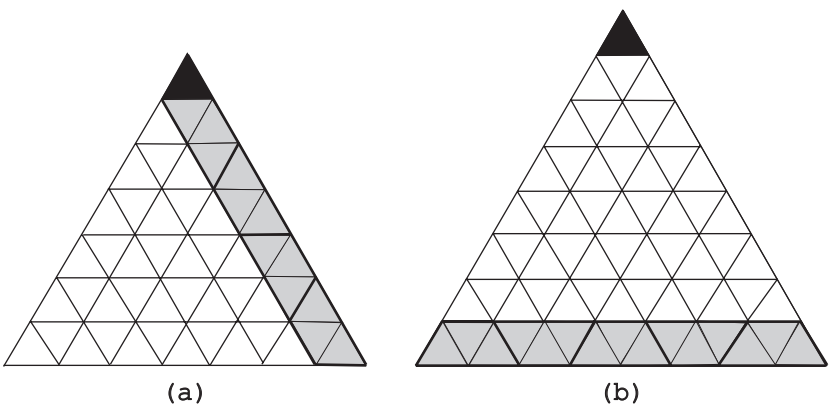


FIGURE 4.63 First step in tiling the triangular region for (a) $n = 7$ and (b) $n = 8$.

Comments The above solution takes advantage of the invariant, decrease-and-conquer, and transform-and-conquer ideas.

It is interesting that for $n = 2^k$, the puzzle can also be solved by the following divide-and-conquer algorithm. If $n = 2$, the region is congruent to one trapezoid tile. If $n = 2^k$ where $k > 1$, place the longer base of the first tile in the middle of the regions's base. Then draw three straight lines connecting the middle points of the three sides of the triangle from which the region was obtained. This will dissect the region into four congruent subregions similar to the original region but exactly half

its size (Figure 4.64a). Hence, each of them can be tiled by the same algorithm, that is, recursively. The algorithm is illustrated in Figure 4.64b for $n = 8$. It is obviously similar to the algorithm for tromino tiling a $2^n \times 2^n$ region with a missing square discussed in the book's first tutorial.

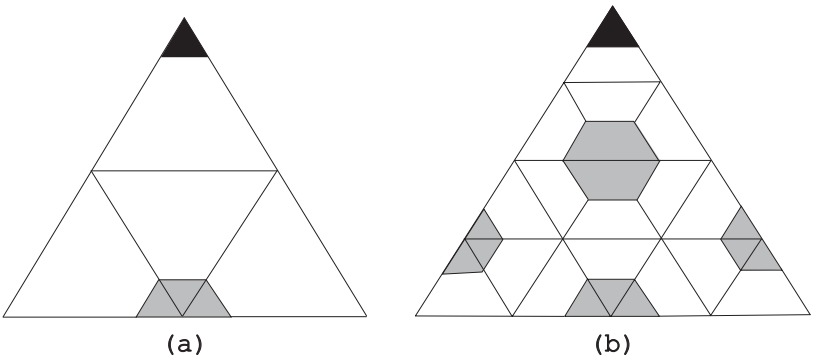


FIGURE 4.64 (a) Dissection of the triangular region into four similar regions half the size. (b) Illustration of the divide-and-conquer tiling for $n = 8$.

The puzzle's instance for $n = 2^k$ was used as an induction exercise by Roland Backhouse in his course on algorithmic problem solving at the University of Nottingham [Backh].

93. Hitting a Battleship

Solution The minimum number of shots needed to guarantee hitting a battleship (a 4×1 or 1×4 rectangle) is 24; one of the possible solutions is shown in Figure 4.65.

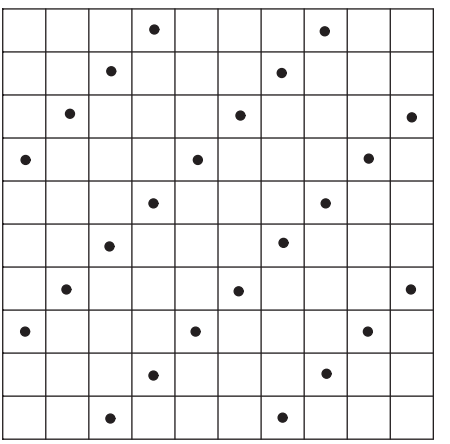


FIGURE 4.65 A solution to the *Hitting a Battleship* puzzle.

The fact that less than 24 shots are not going to do the job follows from Figure 4.66, which shows 24 possible positions of the battleship, with one shot being necessary to hit each of them.

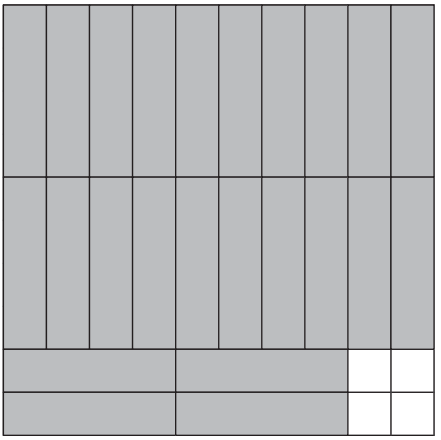


FIGURE 4.66 Twenty-four possible positions of the battleship.

Comments The puzzle illustrates the idea of the worst-case analysis in a rather unusual setting. An article in the Russian journal *Kvant* [Gik80] gave another 24-shot solution, shown in Figure 4.67.

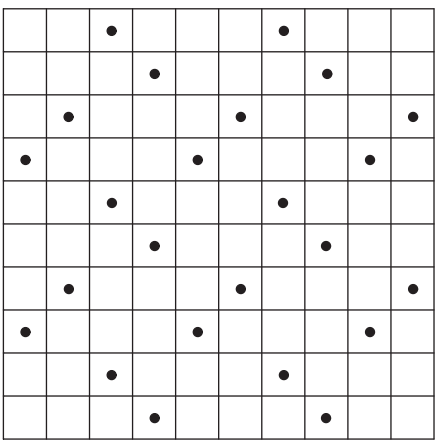


FIGURE 4.67 Alternative solution to the *Hitting a Battleship* puzzle.

The answer to the same question for an 8×8 board, for which the minimum of 21 shots are needed, was given earlier by Solomon Golomb in the first edition of his book on polyominoes [Gol94].

94. Searching a Sorted Table

Solution Let us start by turning up the card at the upper right corner of the array and compare its number with the number we are searching for. If the numbers are

the same, the problem is solved. If the search number is smaller than the number on the card, the search number cannot be in the last column, and we can move left to the card in the preceding column. If the search number is larger than the number on the card, the search number cannot be in the first row, and we can move down to the card in the next row. Repeating this operation until either a search number is found or the search leads outside the array solves the problem.

The algorithm's sequence of turned up cards forms a zigzag line made up of segments going left or down from the upper right corner to some card in the array. The longest such line ends at the lower left corner, turning up the total of 19 cards. No such line can be longer, because it cannot have more than nine horizontal and nine vertical segments.

Comments The algorithm is an application of the decrease-by-one strategy, because on each iteration it decreases either the number of rows or the number of columns that may contain the search number.

The puzzle has appeared—both in print and on the Web—in several publications devoted to technical interviews (e.g., [Laa10, Problem 9.6]).

95. Max-Min Weights

Solution Divide the coins into $\lfloor n/2 \rfloor$ pairs, leaving one coin aside if n is odd. For each pair, weigh the coins to find the lighter and heavier of the two. (If they weigh the same, the tie can be broken arbitrarily.) In $\lfloor n/2 \rfloor - 1$ weighings, find the lightest among the $\lfloor n/2 \rfloor$ lighter coins and then the heaviest among the $\lfloor n/2 \rfloor$ heavier coins. If n is even, the problem is solved; if n is odd, weigh the coin set aside against the lightest and heaviest coins previously found to determine the lightest and heaviest coins overall.

The total number of weighings made by the algorithm, $W(n)$, is given by the following formulas. If n is even,

$$W(n) = \frac{n}{2} + 2\left(\frac{n}{2} - 1\right) = \frac{3n}{2} - 2.$$

If n is odd,

$$W(n) = \left\lfloor \frac{n}{2} \right\rfloor + 2\left(\left\lfloor \frac{n}{2} \right\rfloor - 1\right) + 2 = 3\left\lfloor \frac{n}{2} \right\rfloor = 3\frac{n-1}{2} = \frac{3n}{2} - \frac{3}{2}.$$

It is easy to check that the formulas for both even and odd cases can be combined into

$$W(n) = \left\lceil \frac{3n}{2} \right\rceil - 2.$$

Indeed, if n is even, $\lceil 3n/2 \rceil - 2 = 3n/2 - 2$. If $n = 2k + 1$ is odd, $\lceil 3n/2 \rceil - 2 = \lceil 3(2k+1)/2 \rceil - 2 = \lceil 3k + 3/2 \rceil - 2 = 3k = 3(n-1)/2 = 3n/2 - 3/2$.

Comments Essentially the same algorithm can be obtained by applying the divide-and-conquer strategy: divide the coins into two equal (or about equal) groups, find the lightest and heaviest coins in each of them, and then weight the two lightest and two heaviest coins to determine the lightest and heaviest coins overall.

The puzzle is a well-known problem in computer science, usually stated as the problem of finding the largest and smallest elements in a set of n numbers. It has been proved that $\lceil 3n/2 \rceil - 2$ is, in fact, the minimum number of comparisons needed to solve the problem in the worst case by any comparison-based algorithm (see [Poh72]).

96. Tiling a Staircase Region

Solution In addition to the obvious tiling of S_2 with a single tromino, the tiling in question is possible for $n > 2$ if and only if either $n = 3k$ where $k > 1$ or $n = 3k + 2$ where $k > 1$.

Obviously, for a tiling of S_n with trominoes to exist, it is necessary that the total number of squares in S_n is divisible by 3. The total number of squares in S_n is equal to the nth triangular number

$$T_n = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}.$$

If $n = 3k$, where k is even (i.e., $k = 2m$), $T_n = 6m(6m + 1)/2 = 3m(6m + 1)$ is divisible by 3. If $n = 3k$, where k is odd (i.e., $k = 2m + 1$), $T_n = (6m + 3)(6m + 4)/2 = 3(2m + 1)(3m + 2)$ is also divisible by 3. Similarly, one can check that if $n = 3k + 1$, where k is even or odd, T_n is not divisible by 3. Finally, if $n = 3k + 2$, T_n is divisible by 3 for both even and odd values of k.

The only possible tilings of the first step in S_3 and of the first and last steps in S_5 (Figure 4.68) show that these regions cannot be tiled with right trominoes.

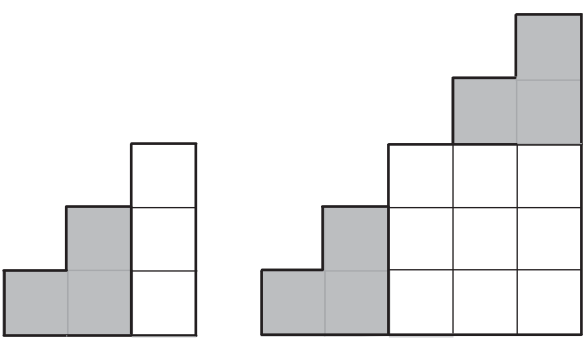


FIGURE 4.68 Impossibility of tiling of S_3 and S_5.

Now we will show that any staircase S_n where $n = 3k$, $k > 1$, can be tiled with right trominoes by the following recursive algorithm. Tilings of S_6 and S_9 are shown in Figure 4.69.

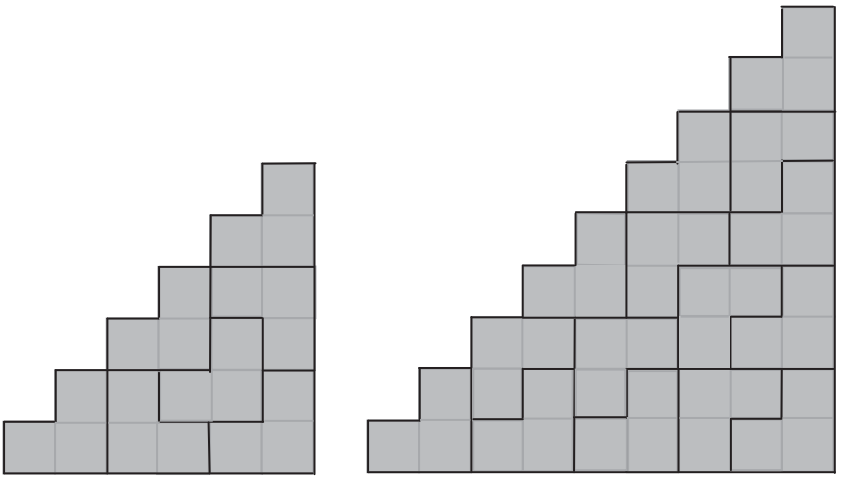


FIGURE 4.69 Tilings of S_6 and S_9.

If $n = 3k$, where k is an even positive integer greater than 2 (i.e., $n = 6m = 6 + 6(m - 1)$ where $m > 1$), the staircase region S_n can be partitioned into the staircase regions S_6 and $S_{6(m-1)}$ and the $6 \times 6(m - 1)$ rectangle. The tiling of S_6 is shown in Figure 4.79, $S_{6(m-1)}$ can be tiled recursively, and the rectangle can be tiled by partitioning it into 3×2 rectangles, each of which to be tiled by two right trominoes (Figure 4.70a).

If $n = 3k$, where k is an odd positive integer greater than 3 (i.e., $n = 6m + 3 = 9 + 6(m - 1)$ where $m > 1$), the staircase region S_n can be partitioned into the

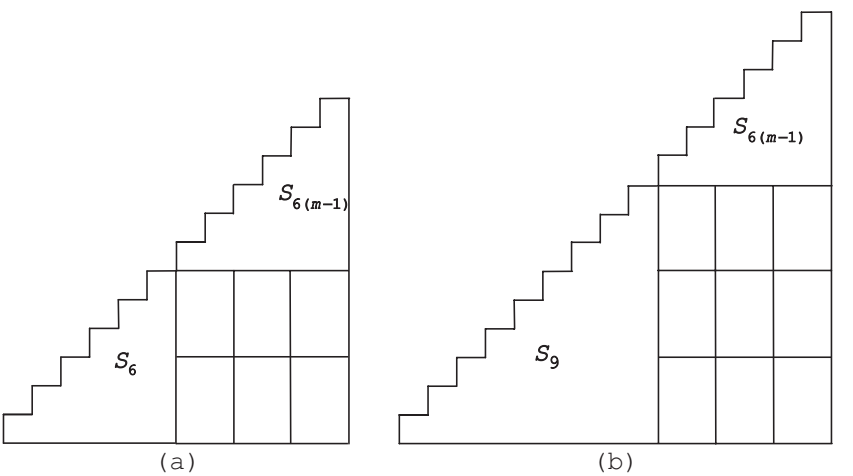


FIGURE 4.70 Tiling of (a) $S_{6+6(m-1)}$ and (b) $S_{9+6(m-1)}$.

staircase regions S_9 and $S_{6(m-1)}$ and the $9 \times 6(m-1)$ rectangle. The tiling of S_9 is shown in Figure 4.79, $S_{6(m-1)}$ can be tiled as shown above, and the rectangle can be tiled by partitioning it into 3×2 rectangles, each of which to be tiled by two right trominoes (Figure 4.70b).

Thus, we have an algorithm to tile any staircase region S_n where $n = 3k, k > 1$.

Finally, we will show that any staircase S_n where $n = 3k + 2, k > 1$, can be tiled with right trominoes as follows. We can partition S_n into S_2, S_{3k}, and the $2 \times 3k$ rectangle (Figure 4.71). S_2 can be tiled by a single tromino, S_{3k} can be tiled by the above algorithm, and the rectangle can be tiled by partitioning it into 2×3 rectangles, each of which to be tiled by two right trominoes.

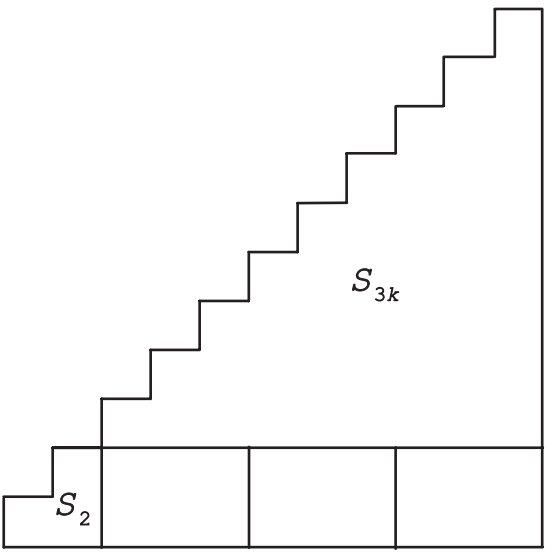


FIGURE 4.71 Tiling of S_{3k+2}.

Comments The solution utilizes several ideas in design and analysis of algorithms: a formula for triangular numbers, an invariant ($T_n \bmod 3 = 0$), divide-and-conquer (region partitioning), and decrease-(by 6)-and-conquer.

The $n = 8$ instance of the puzzle was included in A. Spivak's book [Spi02, Problem 80].

97. The Game of Topswops

Solution The game always stops after a finite number of iterations.

Indeed, the king may appear at the top of the deck no more than once because if it does, the algorithm will move it to the last 13th position, and no other card (which has a smaller value) at the top of the deck will be able to move the king from there. Similarly, the queen may appear at the top no more than twice: after its first appearance there and subsequent move to the 12th position, only the king at the top will be able to move it from there, which cannot happen more than once.

The jack may appear as the top card no more than four times: after the first appearance, it can be moved from its final position only by the king or the queen, which can appear at the top no more than $1 + 2 = 3$ times. In general, which can be formally proved by induction, the card of value i, $2 \le i \le 13$, can appear at the top of the deck no more than $1 + (1 + 2 + \cdots + 2^{12-i}) = 2^{13-i}$ times, where $(1 + 2 + \cdots + 2^{12-i}) = 2^{13-i} - 1$ is an upper bound for the number of times the cards of greater value than i may appear at the top. In particular, the ace will appear on the top of the deck to stop the game no later than after $2^{12} - 1$ appearances there of all the other cards. In fact, the longest game requires 80 moves, which was established with a help of a computer program [Knu11, p. 721].

Comments This is an example of an algorithmic puzzle whose objective is to prove that the puzzle's procedure stops after a finite number of iterations for every possible input.

The game was invented and named by John H. Conway, a British mathematician working since 1986 at Princeton University [Gar88b, p. 76]. Of course, the playing card version of the topswops game can be extended to any set of $n \ge 1$ cards with values from 1 to n written on them.

98. Palindrome Counting

Solution The answer is 63,504.

As suggested by the hint to this puzzle, we can start by counting the number of ways to spell CAT I SAW. Any such string starts at the C at the center and is contained in one of the four triangles formed by the diamond's diagonals. One of these triangles is shown in Figure 4.72. The number of strings spelling

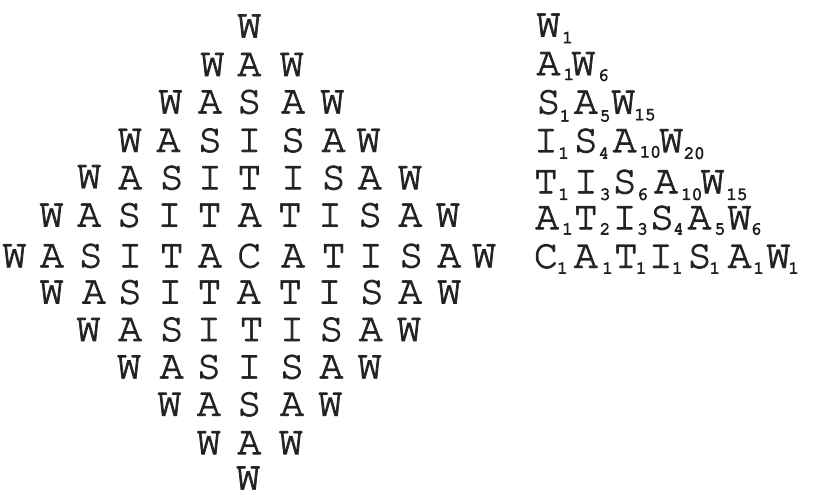


FIGURE 4.72 Diamond of letters and the number of ways to reach each letter from C while spelling CAT I SAW in the triangle.

CAT I SAW in the triangle can be found by the standard application of dynamic programming (see the tutorial on algorithm design strategies). These numbers can be computed by diagonals parallel to the triangle's hypotenuse by adding up the adjacent numbers to the left and below the letter in question. These numbers form Pascal's triangle. The sum of these numbers on the triangle's hypotenuse—the diamond's border—is equal to 2^6.

The total number of strings spelling CAT I SAW for the entire diamond is then obtained by the formula $4 \cdot 2^6 - 4$. (We need to subtract 4 to compensate the overcount of the strings along the diamond's diagonals.) This implies that the total number of ways to spell WAS IT A CAT I SAW is given by the formula $(4 \cdot 2^6 - 4)^2 = 63,504$.

Comments In addition to dynamic programming, the solution exploits symmetry twice by counting occurrences of the half of the palindrome in the quarter of the given shape.

The puzzle is from *Mathematical Puzzles of Sam Loyd* [Loy59, Problem 109]; it also appeared in Dudeney's *The Canterbury Puzzles* [Dud02, Problem 30] with the letter R instead of the letter C. Dudeney also gave a general formula for the number of ways to read such a palindrome of $2n + 1$ letters ($n > 0$) in a diamond-shaped arrangement: $(4 \cdot 2^n - 4)^2$.

99. Reversal of Sort

Solution The puzzle can be solved with $(n - 1)^2/4$ card exchanges, which is the minimum, for any odd n; it has no solution for any even n.

Any sequence of allowed exchanges can only swap cards that are either both in even positions or both in odd positions. Therefore, if n is even, the first card with the largest number cannot be moved to the last position that is even.

If n is odd ($n = 2k - 1$, $k > 0$), the problem can be solved by applying a sorting algorithm such as *bubble sort* or *insertion sort* first to the numbers in odd positions and then to the numbers in even positions. Both these algorithms work by exchanging out-of-order pairs of adjacent elements. For example, if bubble sort is applied to the odd-position numbers, it will exchange the first number with the third, then the third with the fifth, and so on, until the largest number ends up in the last position. Then, on the second pass, the second largest number in an odd position will "bubble up" to its final position and so on. After $k - 1$ such passes the odd-position numbers will be sorted.

Bubble sort makes $(s - 1)s/2$ exchanges on strictly decreasing arrays of size s. Therefore, it will make $(k - 1)k/2$ exchanges for the cards in the odd positions and $(k - 2)(k - 1)/2$ exchanges for the cards in the even positions, yielding the total of

$$(k - 1)k/2 + (k - 2)(k - 1)/2 = (k - 1)^2 = (n - 1)^2/4.$$

This number of exchanges cannot be decreased by the following reason. We are only allowed to exchange adjacent elements in the two sorted-in-reverse sequences

formed by the cards in the odd-numbered and even-numbered positions. This operation decreases the total number of *inversions*—two elements out of order—in their sequence by one. The total number of inversions in a strictly decreasing sequence of s elements is equal to $(s-1)s/2$: the first element is larger than all $s-1$ elements following it, the second one is larger that $s-2$ elements, and so on, for the total of $(s-1)+(s-2)+\cdots+1=(s-1)s/2$ inversions. Hence, the same number is the minimum of exchanges necessary for any sequence of operations reducing the number of inversions by one at a time.

Comments The main themes the puzzle exploits is those of parity and inversion.

The puzzle is an extension of Problem 155 in [Dyn71], which only considered the case of $n=100$.

100. A Knight's Reach

Solution The answer is $7n^2+4n+1$ for $n \geq 3$, and it is 8 and 33 for $n=1$ and 2, respectively.

The squares reachable by the knight in $n=1$, 2, and 3 moves are shown in Figure 4.73. As can be seen from that figure, the number of distinct squares reachable by the knight in n moves, $R(n)$, for $n=1$, 2, and 3, are $R(1)=8$, $R(2)=33$, and $R(3)=76$. It is not difficult to prove by mathematical induction that for any odd $n \geq 3$, all the squares reachable in n moves are the squares of the color opposite to the color of the starting square that lie on the boundary or within the octagon with the center at the starting square and horizontal and vertical sides made of $2n+1$ squares (see Figure 4.73c for $n=3$). For any even $n \geq 3$, the only difference is the color of the squares, which is the same as that of the starting one. To derive a formula for $R(n)$ where $n \geq 3$, we can, for example, partition the octagon into the central $(2n+1) \times (4n+1)$ rectangle and two congruent trapezoids above and below the rectangle. The rectangle is made up of $n+1$ rows each with $2n+1$ reachable squares, interposed with n rows each with $2n$ reachable squares. The number of reachable squares in the trapezoids can be obtained by applying the standard formula for the sum of the terms in an arithmetical progression:

$$2[(n+1)+(n+2)+\cdots+2n]=2\frac{n+1+2n}{2}n=(3n+1)n.$$

This immediately implies the following formula for the total number of reachable squares for $n \geq 3$:

$$R(n)=(2n+1)(n+1)+2n^2+(3n+1)n=7n^2+4n+1.$$

Comments The problem was included in E. Gik's *Mathematics on the Chessboard* [Gik76, pp. 48–49].

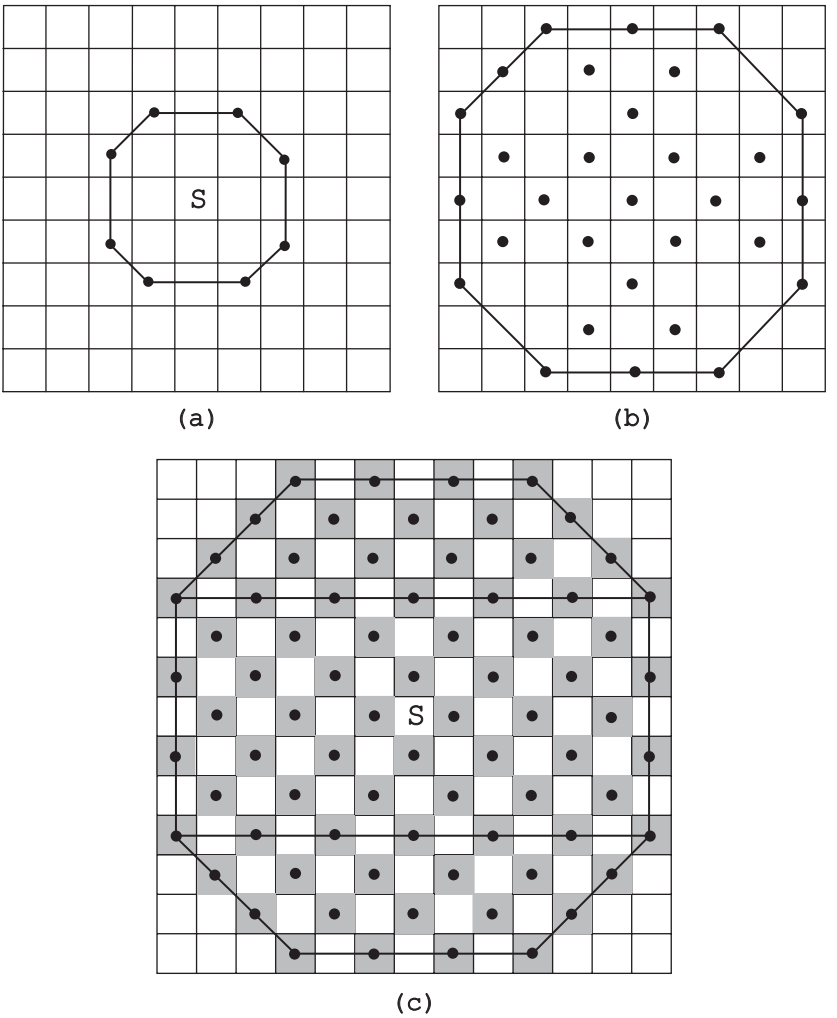


FIGURE 4.73 Squares reachable by the knight from square S in (a) one move, (b) two moves, and (c) three moves.

101. Room Painting

Solution The algorithm presented in Figure 4.74 requires the total of $13 + 11 + (1 + 1 + 3 + 1) = 30$ room repaintings for the half of the rooms in the palace. The other half—symmetric to the first one with respect to the main diagonal—can be taken care of in the same manner. Thus, the entire job can be done by repainting the rooms 60 times.

Comments The solution takes advantage of the palace's symmetry, which can be considered a divide-and-conquer application.

The puzzle is based on Problem 32 in *Mathematical Circles* [Fom96, p. 68].

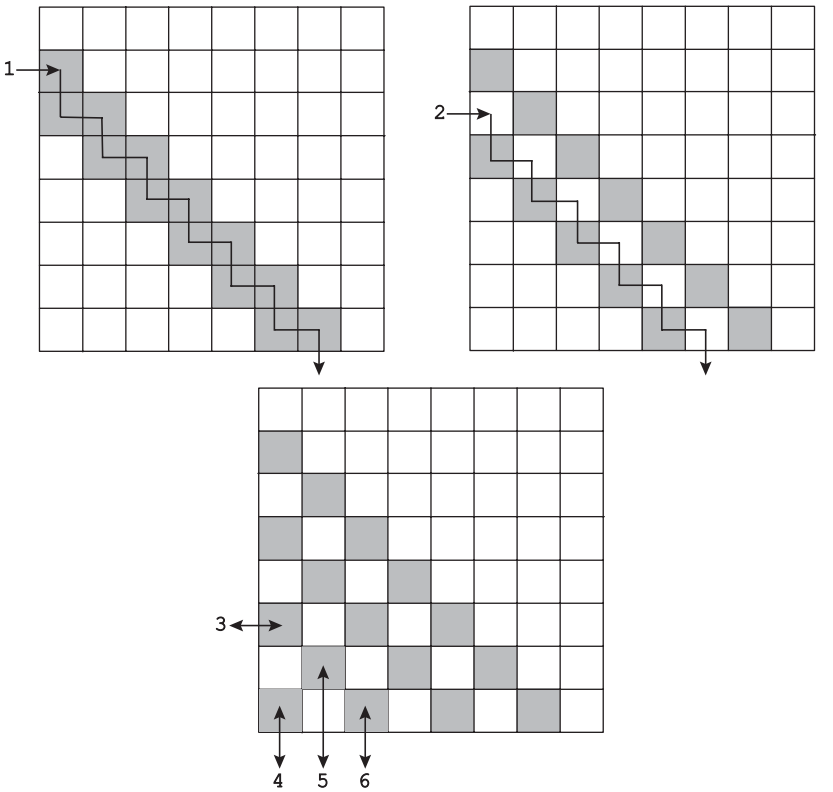


FIGURE 4.74 Solution to the *Room Painting* puzzle: the numbers indicate a path ordering to paint the squares below the main diagonal.

102. The Monkey and the Coconuts

Solution The answer is 15,621.

Let n be the initial number of coconuts, let a, b, c, d, and e be the number of coconuts taken by the first, second, third, fourth, and fifth sailor, respectively, and let f be the number of coconuts each of them received in the morning. Then the following system of equations is obtained:

$$n = 5a + 1$$

$$4a = 5b + 1$$

$$4b = 5c + 1$$

$$4c = 5d + 1$$

$$4d = 5e + 1$$

$$4e = 5f + 1.$$

The easiest way to solve the system is to add 4 to each of the equations to get the following:

$$\begin{aligned} n+4 &= 5(a+1) \\ 4(a+1) &= 5(b+1) \\ 4(b+1) &= 5(c+1) \\ 4(c+1) &= 5(d+1) \\ 4(d+1) &= 5(e+1) \\ 4(e+1) &= 5(f+1). \end{aligned}$$

After multiplying the left- and right-hand sides of the equations, we obtain

$$\begin{aligned} &4^5(n+4)(a+1)(b+1)(c+1)(d+1)(e+1) \\ &= 5^6(a+1)(b+1)(c+1)(d+1)(e+1)(f+1), \end{aligned}$$

or

$$4^5(n+4) = 5^6(f+1).$$

The last equation implies that if it has an integer solution, $n+4$ and $f+1$ must be divisible by 5^6 and 4^5, respectively. Hence, $n+4 = 5^6$ and $f+1 = 4^5$ are the smallest natural numbers that satisfy the equation. Therefore, the smallest value of n is $5^6 - 4 = 15{,}621$. (Obviously, all the other unknowns, i.e., a, b, c, d, e, and f, will also be positive integers as well.)

Comments The solution given above was suggested by R. Gibson, a high school student in South Africa, in 1958.

As mentioned in the first tutorial, some puzzles can be solved by reducing them to mathematical problems. The above solution provides a good example of this approach.

Different versions of this puzzle has been known for a very long time. David Singmaster's annotated bibliography [Sin10, Section 7.E] devotes more than a dozen pages to references dealing with it. Martin Gardner discussed this problem in his *Scientific American* column and then in [Gar87, Chapter 9]. His discussion contained some entertaining anecdotes about the puzzle's history and a few methods for solving it, including an ingenious method of adding four fictitious or colored coconuts to simplify the computations.

103. Jumping to the Other Side

Solution The answer is "no."

Color the board's positions as squares of a chessboard (Figure 4.75).

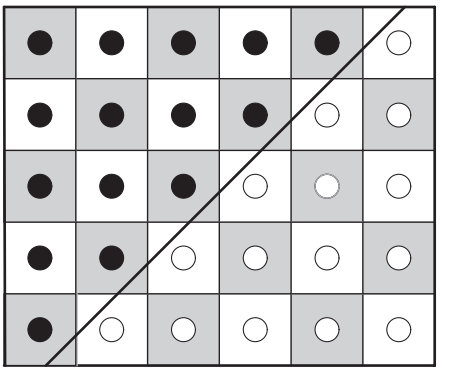


FIGURE 4.75 Coloring the board of the *Jumping to the Other Side* puzzle.

Out of the 15 positions occupied by the counters, 9 are on the dark squares, whereas there are only 6 destination positions on the dark squares. Since any allowed move preserves the color of a counter's position, the puzzle cannot be solved.

Comments The puzzle is based on an invariant idea. It is a minor modification, suggested to Martin Gardner by a reader, of a problem from *The Last Recreations* [Gar97b, pp. 335–336]; Gardner attributed the problem to Mark Wegman of IBM's Thomas J. Watson Research Center.

104. Pile Splitting

Solution a. The sum of the products is equal to $(n-1)n/2$ irrespective of the splits done.

Computing the sum of the products for different ways to split a pile of n counters into n piles for the first few values of n leads one to hypothesize that the sum does not depend on a way the splitting is done. As to a formula for this sum of the products, $P(n)$, one can either note the pattern of the so-called *triangular numbers* (the sums of the first consecutive positive integers—see the second tutorial) emerging in inspection of small instances of the puzzle, or obtain it by considering the simplest way of splitting the piles by setting aside one counter on each split. The latter leads to the recurrence

$$P(n) = 1 \cdot (n-1) + P(n-1) \text{ for } n > 1, \quad P(1) = 0,$$

which can be easily solved by backward substitutions:

$$\begin{aligned} P(n) &= (n-1) + P(n-1) = (n-1) + (n-2) + P(n-2) = \dots \\ &= (n-1) + (n-2) + \cdots + 1 + P(1) = (n-1)n/2. \end{aligned}$$

Here is a straightforward proof by strong mathematical induction that $P(n)$ is indeed equal to $n(n-1)/2$ and does not depend on a way the splitting into n

piles is done. If $n = 1$, $P(1) = 0$. Assume now that the assertion is correct for any $1 \le j < n$; we will prove that it is also correct for $j = n$. Indeed, if the initial pile of n counters is split into piles of k and $n - k$ counters where $1 \le k < n$, then by the inductive assumption the sums for these two piles are equal to $k(k-1)/2$ and $(n-k)(n-k-1)/2$, respectively. Hence, we have the following for the sum in question:

$$\begin{aligned} P(n) &= k(n-k) + k(k-1)/2 + (n-k)(n-k-1)/2 \\ &= \frac{2k(n-k) + k(k-1) + (n-k)(n-k-1)}{2} \\ &= \frac{2kn - 2k^2 + k^2 - k + n^2 - 2nk + k^2 - n + k}{2} \\ &= \frac{n^2 - n}{2} = \frac{n(n-1)}{2}. \end{aligned}$$

b. The maximal sum of the sums is equal to $n(n+1)/2 - 1$.

Denoting the maximal sum that can be obtained for the initial pile of n counters by $M(n)$, we have the following recurrence:

$$M(n) = n + \max_{1 \le k \le n-1} [M(k) + M(n-k)] \quad \text{for } n > 1, \quad M(1) = 0. \qquad (1)$$

Computing the values of $M(n)$ for the first few values of n leads one to hypothesize that the sum of sums is maximized when $k = 1$ and hence

$$M(n) = n + M(n-1) \quad \text{for } n > 1, \quad M(1) = 0.$$

The solution to the last recurrence can be easily obtained by backward substitutions:

$$\begin{aligned} M(n) &= n + M(n-1) = n + (n-1) + M(n-2) = \dots \\ &= n + (n-1) + \dots + 2 + M(1) = n(n+1)/2 - 1. \end{aligned}$$

It is not difficult to verify by strong mathematical induction that $M(n) = n(n+1)/2 - 1$ indeed satisfies recurrence (1) as follows. The basis $M(1) = 0$ checks out immediately. Assume now that $M(j) = j(j+1)/2 - 1$ satisfies the equation $M(j) = j + \max_{1 \le k \le j-1} [M(k) + M(j-k)]$ for every $1 \le j < n$. We will show that then $M(n) = n(n+1)/2 - 1$ satisfies the equation $M(n) = n + \max_{1 \le k \le n-1} [M(k) + M(n-k)]$. Using the inductive assumption, we obtain:

$$\begin{aligned} M(n) &= n + \max_{1 \le k \le n-1} [M(k) + M(n-k)] \\ &= n + \max_{1 \le k \le n-1} [k(k+1)/2 - 1 + (n-k)(n-k+1)/2 - 1] \\ &= n + \max_{1 \le k \le n-1} [k^2 - nk + (n^2+n)/2 - 2]. \end{aligned}$$

Since the quadratic function $k^2 - nk + (n^2 + n)/2 - 2$ attains its minimum at $n/2$, the middle point of the interval $1 \le k \le n - 1$, it attains its maximal value at the interval's endpoints $k = 1$ and $k = n - 1$. Therefore,

$$n + \max_{1 \le k \le n-1} [k^2 - nk + (n^2 + n)/2 - 2] = n + [1^2 - n \cdot 1 + (n^2 + n)/2 - 2]$$
$$= n(n + 1)/2 - 1,$$

which completes the proof.

Comments The problem in part (a)—see K. Rosen's *Discrete Mathematics and Its Applications* [Ros07, p. 292, Problem 14]—is based on an invariant idea. The problem in part (b) has a dynamic programming flavor.

105. The MU Puzzle

Solution It is impossible to change MI into MU by using the four rules of the puzzle.

By inspecting the rules given, it is easy to see that any string that can be derived starts with the symbol M, which is the only occurrence of this symbol in the string. (We need this observation to clarify Rule 2.)

Consider now n, the number of I's occurrences in the strings that can be obtained. For the starting string MI, $n = 1$, which is not divisible by 3. Only Rules 2 and 3 change n by doubling it and decreasing it by 3, respectively. Neither of these operations can result in a number that is divisible by 3 if it was not divisible by 3 before the change. Since for the target string MU, $n = 0$ and hence is divisible by 3, this string cannot be derived from MI for which $n = 1$ and hence is not divisible by 3.

Comments The solution to the puzzle is based on the invariant method discussed in the tutorial on algorithm analysis techniques. The question of which strings can be obtained by applying rules similar to those of this puzzle is important to computer science for several reasons; in particular, high-level computer languages are defined using such rules.

The puzzle was introduced in the first chapter of Douglas Hofstadter's *Gödel, Escher, Bach* [Hof79] as an example of a formal system.

106. Turning on a Light Bulb

Solution The puzzle can be solved by the following recursive algorithm for pushing the buttons numbered from 1 to n. If $n = 1$ and the light is off, push button 1. If $n > 1$ and the light is off, apply the algorithm recursively to the first $n - 1$ buttons. If this fails to turn the light on, push button n and, if the light is still off, apply the algorithm to the first $n - 1$ buttons again.

The recurrence for the number of button pushes in the worst case is

$$M(n) = 2M(n-1) + 1 \text{ for } n > 1, \quad M(1) = 1,$$

whose solution is $M(n) = 2^n - 1$. (See the second tutorial where the same recurrence for the *Tower of Hanoi* puzzle is solved.)

Alternatively, since a switch can be in one of the two states, it can be thought of as a bit in an n-bit string in which 0 and 1 represent, say, the initial and opposite states of the switch, respectively. The total number of such bit strings (switch configurations) is equal to 2^n; one of them represents an initial state, the remaining $2^n - 1$ bit strings contain the one that will turn on the light bulb. In the worst case, all these $2^n - 1$ switch combinations will have to be checked. To accomplish this with the minimum number of button pushes, every push must produce a new switch combination.

There are several algorithms that start with a bit string of n zeros and generate all the other $2^n - 1$ bit strings by changing just a single bit at a time. The most well-known of them is the so-called *binary reflected Gray code*, which can be constructed as follows. If $n = 1$, return the list 0, 1. If $n > 1$, generate recursively the list of bit strings of size $n - 1$ and make a copy of this list; then add 0 in front of each bit string in the first list and add 1 in front of each bit string in the second list; finally, append the second list in reversed order to the first list. For example, the algorithm generates the following bit string sequence for $n = 4$:

0000	0001	0011	0010	0110	0111	0101	0100
1100	1101	1111	1110	1010	1011	1001	1000

To return to the puzzle, we can number the switches from 1 to n right to left and use the sequence of the Gray's code bit strings for guidance about which buttons to push: if the next bit string differs from its immediate predecessor in the ith bit from the right, we push button number i. For example, for four switches, it will guide us to push the buttons in the following sequence:

121312141213121.

Comments The first solution is based on the decrease-by-one strategy. The second solution takes advantage of two strategies: representation change (to represent switches and their buttons by bit strings) and decrease-by-one (to generate the Gray code).

The puzzle was proposed and solved by J. Rosenbaum in 1938 [Ros38] by the method described above—years before the U.S. patents were granted for the Gray code invention. M. Gardner mentioned this puzzle in his article about Gray codes [Gar86, p. 21]; the article also included applications to two better-known puzzles: the *Chinese Rings* and the *Tower of Hanoi*. After briefly reviewing a rather convoluted history of Gray codes, D. Knuth concluded that it is the Frenchman

Lois Gros who should be considered a true inventor of this code in view of Gros' 1872 book on the *Chinese Rings* [Knu11, pp. 284–285].

107. The Fox and the Hare

Solution The fox can catch the hare if s is even, and he cannot if s is odd.

Consider the parity (even or odd) of the cells $F(n)$ and $H(n)$ in which, respectively, the fox and hare can be before the nth move. For $n = 1$, $F(1) = 1$, $H(1) = s$. On each move, their positions change by 1 for the fox and by 3 for the hare. Hence, the parity of their positions change to the opposite on each move, leaving the parity of the difference between their positions unchanged. Therefore, if the initial position s of the hare is odd, the parity of the difference will remain even before a move by the fox, and he will never be in a position adjacent to that of the hare.

Nor will the hare lose because of finding himself without a legitimate move, as could have happened if the board were less than 11 cells long. If $s = 3$, the hare can jump to the right on his first two moves to position 9, making the new situation similar to either that of $s = 7$ if the fox moves to position 3 on his first two moves or to that of $s = 9$ if the fox first moves to position 2 and then returns to position 1. Similarly, if $s = 5$, the hare can jump to position 8 on his first move, also making the new situation similar to that of $s = 7$. Let now $s = 7$, that is, $F(1) = 1$ and $H(1) = 7$. After the fox moves to position 2 on his first move, the hare can jump to position 4. If the fox then returns to 1, the hare returns to 7; if the fox moves to 3, the hare jumps back to 7 (not to 1, which would have lead to the immediate defeat after the fox's move to 4 and leaving the hare without a possible move). After that, the hare can simply jump back and forth between 7 and 10 until the fox tries to catch him by advancing to 6. Then the hare jumps over the fox to 4 and starts alternating between 1 and 4. If the fox returns to 5, the hare jumps back to 7, and so on. Thus, the fox will never catch the hare in this case. If s is an odd integer between 9 and 27, inclusive, the same strategy used for $s = 7$ allows the hare to avoid the fox. If $s = 29$, the first jump by the hare to 26 makes the situation analogous to the fox starting in 1 and the hare starting at 25.

If s is even, the fox can force the hare into a position next to it by simply always moving to the right. Indeed the hare starts to the right of the fox, with the difference between their positions being odd. If the difference is equal to 1, the fox simply catches the hare on the next move. If the difference is 3, the fox moves to the right, leaving the hare with a choice of jumping to the left of the fox to be caught on the next move or jumping to the right; if the difference is 5 or more, the hare can jump in either direction. In all these cases, either the hare can be caught on the next move or the situation becomes equivalent to restarting the game with the board shortened by one position, the fox in the leftmost position and the hare some odd positions to the right of the fox. Hence, if the fox does not catch the hare before reaching position 26, he will do this on his next move.

Comments The solution exploits two ideas in algorithm design and analysis: parity and decrease-and-conquer.

The puzzle is a modification of a similar game from [Dyn71, p. 74, Problem 54].

108. The Longest Route

Solution Assuming that the distance between two adjacent posts is equal to 1, the length of the longest route for any $n \geq 2$ is

$$\frac{(n-1)n}{2} + \left\lfloor \frac{n}{2} \right\rfloor - 1.$$

Let us number the posts sequentially from 1 to n. It is easy to see that the routes obtained by the greedy strategy—that is, $1, n, 2, n-1, \ldots, \lceil n/2 \rceil$ for odd n's and $1, n, 2, n-1, \ldots, n/2, n/2+1$ for even n's—can be made longer by replacing the last segments of these routes by the longer segments connecting the last posts of the greedy routes to post 1 (see Figure 4.76 illustrating the cases of $n = 5$ and $n = 6$). The proof that the greedy paths adjusted in this fashion are indeed the longest is not difficult but rather tedious; it can be found in Hugo Steinhaus's *One Hundred Problems in Elementary Mathematics* [Ste64].

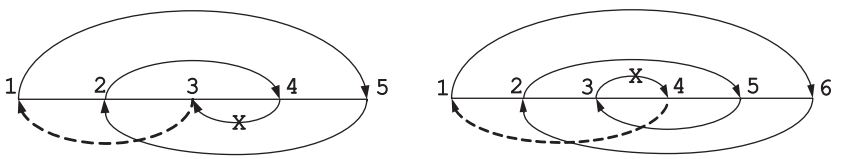


FIGURE 4.76 The longest routes obtained by adjustments to the greedy solutions for $n = 5$ and $n = 6$.

The longest path for $n = 5$ is $3 \to 1 \to 5 \to 2 \to 4$; for $n = 6$, the longest path is $4 \to 1 \to 6 \to 2 \to 5 \to 3$. The solutions to the problem are not unique for $n > 4$, but any longest route must either start or end at one of the three middle posts if n is odd and at one of the two middle posts if n is even.

Comments The puzzle provides an interesting example of a greedy algorithm yielding a solution that is not optimal but which can be easily adjusted to be one.

As mentioned above, the puzzle is from *One Hundred Problems in Elementary Mathematics* by Hugo Steinhaus [Ste64, Problem 64]. It was also included in one of Martin Gardner's articles in *Scientific American*, later republished in [Gar71, pp. 235, 237–238].

109. Double-*n* Dominoes

Solution a. A double-n domino set consists of $n+1$ $(0, j)$ tiles where $0 \leq j \leq n$, n $(1, j)$ tiles where $1 \leq j \leq n, \ldots$, and finally one (n, n) tile. Therefore, the total number of tiles is equal to $(n+1) + n + \cdots + 1 = (n+1)(n+2)/2$.

b. For every value of k, $0 \le k \le n$, there are n tiles with one half containing k spots and the other half containing a different number of spots, and there is also one double with both halves containing k spots. So, the total number of the halves with k spots is equal to $n + 2$. Hence, the total number of spots on all the tiles is equal to $\sum_{k=0}^{n} k(n+2) = n(n+1)(n+2)/2$.

c. A ring in question can be constructed if and only if n is a positive even number. Since every pair of adjacent tiles in a ring must have the same number of spots on their adjacent halves, the number of tiles with halves containing k spots must be even for every $0 \le k \le n$. Since this number is equal to $n + 2$ for every k (see part (b)), no such ring is possible when n is odd.

When n is even and positive, a required ring can be constructed recursively as follows. If $n = 2$, there is just one ring:

$$R(2)\text{: } (0, 0)(0, 1)(1, 1)(1, 2)(2, 2)(2, 0).$$

If $n = 2s$, where $s > 1$, construct recursively a ring $R(2s - 2)$ of all the dominoes in the double-$(2s - 2)$ domino subset. Then construct a chain comprising the remaining dominoes (i, j), where $j = 2s - 1$ and $2s$, $0 \le i \le j$, for example, by sequencing quartets

$$(2t, 2s - 1)(2s - 1, 2t + 1)(2t + 1, 2s)(2s, 2t + 2) \text{ for } t = 0, 1, \ldots, s - 1$$

followed by $(2s, 0)$. Finally, splice this chain between the adjacent pair $(0, 0)$ and $(0, 1)$ in $R(2s - 2)$ to get a ring made up of all the double-$2s$ dominoes.

Alternatively, one can reduce the problem to the question about existence of an Euler circuit in a complete graph with $n + 1$ vertices. Vertex i, $0 \le i \le n$, in this graph represents a possible number of spots on one of the two halves of an n-domino, and an edge between vertices i and j represents the domino with i and j spots on its halves. The doubles can be either eliminated until a ring including all the other dominoes is constructed and then inserted between any two dominoes with the same number of spots, or they can be represented by loops (edges connecting vertices to themselves). Obviously, an Euler circuit in such a graph would specify a ring of all n-dominoes, and vice versa. By a well-known theorem—see, for example, this book's tutorial on algorithm analysis—a connected graph has an Euler circuit if and only if all its vertices have even degrees. It is the case for the graph in question if and only if n is even. An algorithm for constructing an Euler circuit is outlined in the solution to the *Figure Tracing* puzzle (#28).

Comments The puzzle exploits several themes: parity, decrease-and-conquer, and, in the alternative solution, problem reduction. Rouse Ball [Bal87, p. 251] attributed the reduction idea to the French mathematician Gaston Tarry, who used it to find the number of different arrangements of a double-n domino set.

110. The Chameleons

Solution The answer is "no."

After two chameleons of different colors meet, the number of the chameleons of their colors will decrease by 1 and the number of chameleons of the other color will increase by 2. Hence, one difference between the numbers of the differently colored chameleons will not change and the two others will increase by 3. Therefore, the division-by-3 remainders of all the three differences will not change. This immediately implies that for the data given it will be impossible for all the chameleons become the same color. Indeed, the initial differences are 4, 1, and 5, and therefore their division-by-3 remainders are 1, 1, and 2. But if all the chameleons could become the same color, one of the differences would have to be 0, with the division-by-3 remainder equal to 0 as well.

Comments The puzzle is based on a rather rare version of the invariant idea. Different versions of the puzzle have been included in several puzzle books (e.g., [Hes09, Problem 24] and [Fom96, p. 130, Problem 21]).

111. Inverting a Coin Triangle

Solution The minimum number of coin moves needed to invert a triangle with $T_n = n(n+1)/2$ coins is $\lfloor T_n/3 \rfloor$.

To invert a coin triangle in the minimum number of moves, we obviously need to use one of its horizontal rows as the base of the inverted triangle. Consider the kth horizontal row of the triangle given, $1 \le k \le n$. (We assume that the rows are counted top to bottom, so that the kth row contains k coins.) Let us find the minimum number of coin moves needed to invert the triangle by making this row the base of the inverted triangle. To do this, $n - k$ coins must be moved into this row. It is convenient to take these coins from the last row as follows: $\lceil (n-k)/2 \rceil$ leftmost coins and $\lfloor (n-k)/2 \rfloor$ rightmost coins in the last row are moved to the right and left ends of row k, respectively; this will leave exactly $n - (\lceil (n-k)/2 \rceil + \lfloor (n-k)/2 \rfloor) = n - (n-k) = k$ coins in the last row. Then $n - k - 2$ coins must be moved into row $k+1$ of the given triangle; it is convenient to take these coins from the penultimate row: its $\lceil (n-k)/2 \rceil - 1$ leftmost coins and $\lfloor (n-k)/2 \rfloor - 1$ rightmost coins are moved to the right and left ends of row $k+1$, respectively. This operation is repeated until $\lceil (n-k)/2 \rceil - \lfloor (n-k)/2 \rfloor$ (which is equal to 0 or 1 depending on whether $n-k$ is even or odd) leftmost coins from row $n - \lfloor (n-k)/2 \rfloor$ are moved to the right end of row $k + \lfloor (n-k)/2 \rfloor$. Finally, the coins from the first $k-1$ rows of the given triangle are moved in the reverse order, that is, from the longest to the shortest, to form successive rows below the original base row.

Figure 4.77 illustrates the solution for the case of even $n-k$. If $n-k$ is odd, we can also move one more coin from the right end of each row than from its left end to get a symmetric solution to the one described above. These two solutions are illustrated in Figure 4.78.

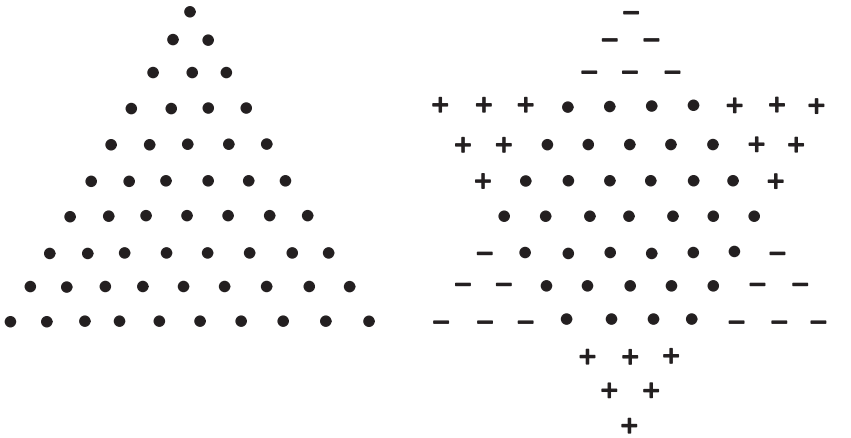


FIGURE 4.77 Inverting a triangle with T_{10} coins by making row $k = 4$ the base of the inverted triangle (+, −, and small circles indicate centers of the added, moved, and stationary coins, respectively).

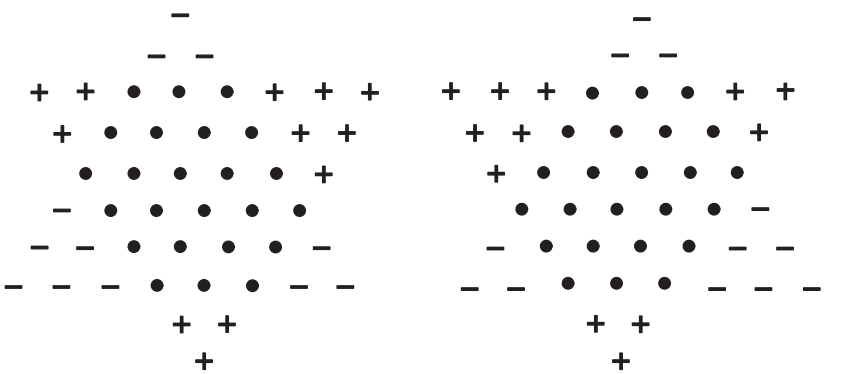


FIGURE 4.78 Inverting a triangle with T_8 coins by making row $k = 3$ the base of the inverted triangle.

The total number of moves made by the algorithm, $M(k)$, is obviously the minimum needed to make the kth row the base of the inverted triangle, because each coin move increases the number of coins in a row that must be lengthened and simultaneously decreases the number of coins in a row that must be shortened. $M(k)$ can be computed as follows:

$$M(k) = \sum_{j=0}^{\lfloor (n-k)/2 \rfloor} (n-k-2j) + \sum_{j=1}^{k-1} j = \sum_{j=0}^{\lfloor (n-k)/2 \rfloor} (n-k) - \sum_{j=0}^{\lfloor (n-k)/2 \rfloor} 2j + \sum_{j=1}^{k-1} j$$

$$= (n-k)\left(\left\lfloor \frac{n-k}{2} \right\rfloor + 1\right) - \left\lfloor \frac{n-k}{2} \right\rfloor \left(\left\lfloor \frac{n-k}{2} \right\rfloor + 1\right) + \frac{(k-1)k}{2}$$

$$= \left(\left\lfloor \frac{n-k}{2} \right\rfloor + 1\right)\left\lceil \frac{n-k}{2} \right\rceil + \frac{(k-1)k}{2}.$$

If $n - k$ is even, the above formula can be further simplified to

$$M(k) = \left(\frac{n-k}{2} + 1\right)\frac{n-k}{2} + \frac{(k-1)k}{2} = \frac{3k^2 - (2n+4)k + n^2 + 2n}{4}.$$

If $n - k$ is odd, the formula for $M(k)$ can be further simplified to

$$M(k) = \left(\frac{n-k-1}{2} + 1\right)\frac{n-k+1}{2} + \frac{(k-1)k}{2} = \frac{3k^2 - (2n+4)k + (n+1)^2}{4}.$$

In either case, the minimum of the quadratic $M(k)$ is attained when $k = (n + 2)/3$. So, if $(n + 2)/3$ is an integer (i.e., if $n = 3i + 1$), the problem has a unique solution (in terms of the coins that remain stationary) because then $n - k = (3i + 1) - (3i + 1 + 2)/3 = 2i$ is even (see Figure 4.77 for an example).

If $(n + 2)/3$ is not an integer, the problem has two qualitatively different solutions[7] obtained by the above algorithm with $(n + 2)/3$ rounded up and down, that is, with $k^+ = \lceil (n + 2)/3 \rceil$ and $k^- = \lfloor (n + 2)/3 \rfloor$.

The minimum number of coins moved can also be expressed by the formula mentioned in references [Gar89, p. 23], [Tri69], and [Epe70]:

$$\left\lfloor \frac{n(n+1)}{6} \right\rfloor = \left\lfloor \frac{T_n}{3} \right\rfloor$$

where $T_n = n(n + 1)/2$ is the total number of coins in the triangle. One can easily check this assertion by substituting each of the possible three cases—$n = 3i$, $n = 3i + 1$, and $n = 3i + 2$—into this formula to verify that it does yield the same answers as the ones obtained above for the optimal choices of k.

Comments As pointed out in the book's first tutorial, reducing a puzzle to a mathematical problem is one of the varieties of the transform-and-conquer strategy in algorithm design.

This puzzle has been around for a long time. For example, Maxey Brooke included the 10-coin instance in his book [Bro63], referring to it as a "coffee winner," that is, a bet—which should win more often than lose—that someone won't be able to find a three-move solution" (p. 15). Martin Gardner asked readers of his March 1966 column in *Scientific American* to find a compact formula for the minimum number of coins that need to be moved in this general instance of the problem. As a solution, he used the following "geometric" approach ([Gar89, p. 23]): "In working on the general problem, for equilateral formations of

[7] We consider solutions qualitatively different if they have distinct rows of the given triangle as their bases. One of these solutions—the one with an odd number of coins moved from the base of the given triangle to the base of an inverted triangle—will have a symmetric counterpart (see Figure 4.78 for an example).

any size, readers may have realized that the problem is that of drawing a bounding triangle (like the frame used to group the 15 balls for a game of billiard), inverting it and placing it over the figure so that it encloses a maximum number of coins. In every case the smallest number of coins that must be moved to invert the pattern is obtained by dividing the number of coins by 3 and ignoring the remainder."

Both Trigg [Tri69] and Eperson [Epe70] described the solution in terms of triangles that need to be cut off a triangle given, but neither of the authors provided a proof of the solution's optimality. Using the cut-off triangles makes it easy to modify the algorithm outlined above to satisfy the additional requirement that "on each move a coin must be slid to a new position so that to touch two other coins that rigidly determine its new position" [Gar89, p. 13]. Rather than moving coins a horizontal row by a horizontal row, we can slide all the coins in a cut-off triangle to their designated locations in the inverted triangle by always taking an outside coin from the triangle given and sliding it to its new location where it touches at least two other coins. For a general exploration of conditions under which one coin arrangement can be transformed to another under such a constraint, see [Dem02].

112. Domino Tiling Revisited

Solution The puzzle has a solution if and only if n is even.

Of course, the problem has no solution for odd n's because of the parity argument: the number of squares to be covered is odd whereas any domino covering can cover only an even number of squares.

When n is even, an $n \times n$ chessboard without two arbitrary 1×1 squares of opposite color can always be tiled with dominoes. For $n = 2$, this is obviously true. For $n > 2$, the proof requires an ingenious device called *Gomory barriers,* named after the U.S. mathematician Ralph Gomory, who had introduced them. These barriers partition the board into two "corridors" with the endpoints at the removed squares if the latter are not next to each other and just one "corridor" if the removed squares are next to each other (see Figures 4.79a and 4.79b). Dominoes of a tiling can always be laid in a unique way along these "corridors."

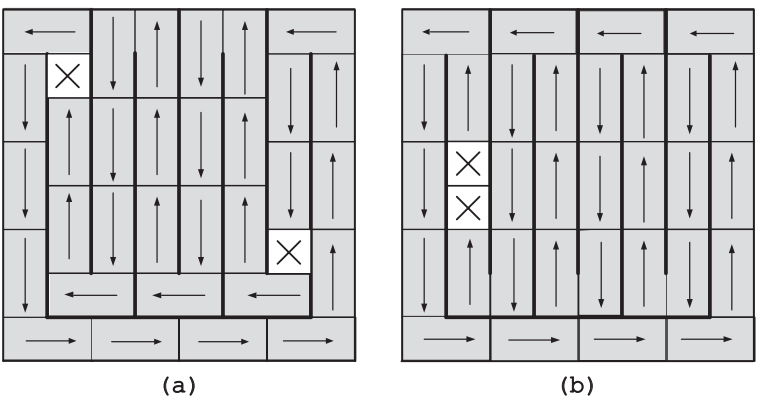


FIGURE 4.79 Domino tiling of an 8×8 board with two missing squares using Gomory barriers. (a) Two missing squares are not adjacent. (b) Two missing squares are next to each other.

Comments Gomory barriers used in the solution are not unique. Solomon Golomb gave four alternative patterns in his book on polyominoes, noting that "there are literally hundreds of others" [Gol94, p. 112]. Of course, if two missing squares of an $n \times n$ board, where n is even, are of the same color—for example, its two diagonally opposite corners (the *Domino Tiling of Deficient Chessboards* puzzle in this book's second tutorial)—no tiling is possible because the numbers of white and black squares to be covered are not the same.

113. Coin Removal

Solution The puzzle can be solved if and only if the number of heads (head-up coins) in the initial line is odd. If it is the case for a line given, the puzzle can be solved by removing repeatedly the leftmost head until no coins are left.

Let us first prove that this algorithm does solve the puzzle for a line with a single head in a line of n coins. If the head is at one of the line's ends (say, the left), its removal will result in the same kind of line with $n - 1$ coins:

$$\underbrace{\mathbf{H}\ \ \mathrm{T}\ \ \ldots\ \ \mathrm{T}}_{n} \quad \Longrightarrow \quad \underbrace{\mathbf{H}\ \ \mathrm{T}\ \ \ldots\ \ \mathrm{T}}_{n-1}$$

Hence, repeating this step n times will remove all the coins.

If the single head is not at one of the line's ends, its removal will result in two shorter lines with a single head at one of each line's ends and a gap between them:

$$\underbrace{\mathrm{T}\ \ldots\ \mathrm{T}\ \mathrm{T}\ \mathbf{H}\ \mathrm{T}\ \mathrm{T}\ \ldots\ \mathrm{T}}_{n} \quad \Longrightarrow \quad \underbrace{\mathrm{T}\ \ldots\ \mathrm{T}\ \mathrm{H}\ _\ \mathrm{H}\ \mathrm{T}\ \ldots\ \mathrm{T}}_{n-1}$$

Thereafter, the puzzle can be solved by removing all the coins in each of these shorter lines as shown above.

Consider now the general case of removing the leftmost head in a line with an odd number of heads that is greater than 1. Let $k \geq 0$ be the number of tails preceding the leftmost head. Whether that head is followed by a tail or by another head, its removal will yield a line with $k - 1$ tails followed by a single head (which is either empty if $k = 0$ or can be removed as described above) and a shorter line with an odd number of heads that can be removed by the same method:

$$\underbrace{\mathrm{T}\ldots\mathrm{T}}_{k}\mathrm{T}\mathbf{H}\underbrace{\mathrm{T}\ldots}_{\text{odd H's}} \Longrightarrow \underbrace{\mathrm{T}\ldots\mathrm{T}}_{k-1}\mathrm{H}_\underbrace{\mathrm{H}\ldots}_{\text{odd H's}} \qquad \underbrace{\mathrm{T}\ldots\mathrm{T}}_{k}\mathrm{T}\mathbf{H}\underbrace{\mathrm{H}\ldots}_{\text{odd H's}} \Longrightarrow \underbrace{\mathrm{T}\ldots\mathrm{T}}_{k-1}\mathrm{H}_\underbrace{\mathrm{T}\ldots}_{\text{odd H's}}$$

The second part of the proof—that if a coin line has an even number of heads the puzzle has no solution—is similar to the proof of the first part. If the number of heads is zero, the puzzle cannot be solved, because only head-up coins can be removed. If there are heads, then removing any one of them either yields one shorter line with an even number of heads or two shorter lines separated by a gap at least one of which having an even number of heads.

Comments While the main idea is clearly based on the decrease-by-one strategy, it gets somewhat unexpected help from divide-and-conquer. Also note that removing an arbitrary head-up coin may not lead to a solution: for example, removing the middle head from the line of three heads makes the removal of the remaining two tails impossible.

The puzzle appeared in J. Tanton's *Solve This* [Tan01, Problem 29.4], with a reference to its circular version by D. Beckwith [Bec97]. The puzzle was also included in *Professor Stewart's Cabinet of Mathematical Curiosities* [Ste09, p. 245].

114. Crossing Dots

Solution Figure 4.80 presents solutions for $n = 3$, 4, and 5. They show how a $2n - 2$ segment path through the n^2 points can be obtained from a $2(n - 1) - 2$ segment path through the $(n - 1)^2$ points by adding to the latter two segments—one vertical and one horizontal—each passing through the next column and row of the n points.

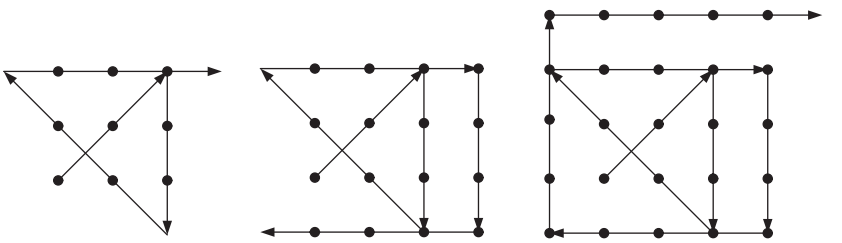


FIGURE 4.80 A $2n - 2$ segment path through n^2 lattice points for $n = 3$, $n = 4$, and $n = 5$.

Comments After the puzzle's instance for $n = 3$ is solved, the path-construction algorithm implements the decrease-and-conquer strategy bottom up.

The case of $n = 3$ is one of the all-time favored puzzles. Because of the nature of the solution, it is often claimed as the source of the expression "thinking outside the box." Both Henry Dudeney and Sam Loyd published it about a century ago. Dudeney also considered the cases of $n = 7$ and $n = 8$ [Dud58, Problems 329–332]. The general case given here was included by Charles Trigg in his *Mathematical Quickies* [Tri85, Problem 261], with a reference to the solution by M. S. Klamkin published in the February 1955 issue of *American Mathematical Monthly* (p. 124).

115. Bachet's Weights

Solution The answers are n consecutive powers of 2 starting with 2^0 and n consecutive powers of 3 starting with 3^0 for parts (a) and (b), respectively.

a. Let us apply the greedy approach to the first few instances of the problem in question. For $n = 1$, we have to use $w_1 = 1$ to balance weight 1. For $n = 2$, we simply add $w_2 = 2$ to balance the first previously unattainable weight of 2.

The weights $\{1, 2\}$ can balance every integral weights up to their sum 3. For $n = 3$, in the spirit of greedy thinking, we take the next previously unattainable weight: $w_3 = 4$. The three weights $\{1, 2, 4\}$ allow to weigh any integral load l between 1 and their sum 7, with l's binary expansion indicating the weights needed for load l:

load l	1	2	3	4	5	6	7
l in binary	1	10	11	100	101	110	111
weights for load l	1	2	$2+1$	4	$4+1$	$4+2$	$4+2+1$

Generalizing these observations, we could hypothesize that for any positive integer n the set of consecutive powers of 2 $\{w_i = 2^i, i = 0, 1, \ldots, n-1\}$ makes it possible to balance every integral load in the largest possible range from 1 to their sum $\sum_{i=0}^{n-1} 2^i = 2^n - 1$, inclusive. The fact that every integral weight l in the range $1 \le l \le 2^n - 1$ can be balanced with this set of weights follows immediately from the binary expansion of l, which yields the weights needed for weighing l.

Since for any set of n weights there are only $2^n - 1$ different subsets (less if some of the weights are the same) that can be put on one pan of the scale, no more than $2^n - 1$ different loads can be measured with them. This proves that no set of n weights can cover a larger range of consecutive integral loads than $1 \le l \le 2^n - 1$ if weights can be put only on the free pan of the scale.

b. If weights can be put on both pans of the scale, a larger range can be reached with n weights for $n > 1$. For $n = 1$, the single weight still has to be 1, of course. The weights $\{1, 3\}$ enable weighing of every integral load up to 4. The weights $\{1, 3, 9\}$ enable weighing of every integral load up to 13, as shown in the following table:

load l	1	2	3	4	5	6	7
l in ternary	1	2	10	11	12	20	21
weights for load l	1	$3-1$	3	$3+1$	$9-3-1$	$9-3$	$9+1-3$
load l	8	9	10	11	12	13	
l in ternary	22	100	101	102	110	111	
weights for load l	$9-1$	9	$9+1$	$9+3-1$	$9+3$	$9+3+1$	

In general, the weights $\{w_i = 3^i, i = 0, 1, \ldots, n-1\}$ enable weighing of every integral load from 1 to their sum $\sum_{i=0}^{n-1} 3^i = (3^n - 1)/2$, inclusive. A load's expansion in the ternary system indicates the weights needed as follows. If the ternary expansion of load l, $l \le (3^n - 1)/2$, contains only 0's and 1's, the load requires putting the weights corresponding to the 1's on the opposite pan of the balance. If the ternary expansion of l contains one or more 2's, we can replace each

2 by $(3-1)$ to represent l uniquely in the *balanced ternary system* (see [Knu98, pp. 207–208]):

$$l = \sum_{i=0}^{n-1} \beta_i 3^i, \text{ where } \beta_i \in \{0, 1, -1\}.$$

For example,

$$\begin{aligned} 5 = 12_3 &= 1 \cdot 3^1 + 2 \cdot 3^0 = 1 \cdot 3^1 + (3-1) \cdot 3^0 = 2 \cdot 3^1 - 1 \cdot 3^0 \\ &= (3-1) \cdot 3^1 - 1 \cdot 3^0 = 1 \cdot 3^2 - 1 \cdot 3^1 - 1 \cdot 3^0. \end{aligned}$$

(Note that if we start with the rightmost 2, after a simplification, the new rightmost 2, if any, will be at some position to the left of the starting one. This proves that after a finite number of such replacements, we will be able to eliminate all the 2's.) Using the representation $l = \sum_{i=0}^{n-1} \beta_i 3^i$, where $\beta_i \in \{0, 1, -1\}$, we can weigh load l by placing all the weights $w_i = 3^i$ for negative β_i's along with the load on one pan of the scale and all the weights $w_i = 3^i$ for positive β_i's on the opposite pan.

Finally, any weight in a set of n weights can be put on the left pan, or on the right pan, or not put on either of them. Hence, there are $3^n - 1$ ways to use n of them for weighing a positive load. Taking into account the symmetry, they will be able to weigh no more than $(3^n - 1)/2$ different loads. This proves that no set of n weights can cover a larger range of consecutive integral loads than $1 \le l \le (3^n - 1)/2$.

Comments The puzzle provides good examples of a successful application of the greedy approach and occasional usefulness of numeral systems other than decimal. It is also worth pointing out that if an exact balancing of each integral load is not required, twice as many integral loads can be determined with the same number of weights. For example, the four weights 2, 6, 18, and 54 can determine any integral load from 1 to 80, inclusive: the even loads $2, 4, \ldots, 80$ can be balanced exactly, whereas any odd load is determined by finding two consecutive combinations of 2, 6, 18, and 54 that underweight and overweight the load in question. For example, if an integral load is heavier than 10 but lighter than 12, its weight is obviously 11 [Sin10, Section 7.L.3, p. 95].

The puzzle is named after Claude Gaspar Bachet de Méziriac, the author of *Problèmes* [Bac12]—the pioneering classic in recreational mathematics published in 1612, which contained a solution to it. Modern research in history of recreational mathematics considers Hasib Tabari (c.1075) and Fibonacci (1202) as the first mathematicians who solved the problem [Sin10, Sections 7.L.2.c and 7.L.3].

116. Bye Counting

Solution The answers to part (a) is $2^{\lceil \log_2 n \rceil} - n$; the answer to part (b) is the number of 1's in the binary representation of $2^{\lceil \log_2 n \rceil} - n$.

a. The number of byes, defined as transfers of the fewest players directly to the second round so that the number of players left for that round is a power of 2, is equal to $2^{\lceil \log_2 n \rceil} - n$. For example, for $n = 10$, the number of byes is equal to $2^{\lceil \log_2 10 \rceil} - 10 = 6$. Indeed, let $2^{k-1} < n \le 2^k$, where n is the number of players. If b players get a bye, $n - b$ players play in the first round, with $(n - b)/2$ winners advancing to the second round. Thus, we have the equation $b + (n - b)/2 = 2^{k-1}$. Its solution is $b = 2^k - n$, where $k = \lceil \log_2 n \rceil$. (Note that $n - b = 2n - 2^k$ and therefore is even as it should be.)

b. Let n be the number of players in the tournament ($2^{k-1} < n \le 2^k$) and $B(n)$ be the total number of byes, defined as transfers of fewest players in each round so that there is an even number of players in each round. In other words, if the number of players is even, there are no byes in that round; if the number of players is odd, one of them gets a bye, making the number of players in the next round equal to $1 + (n - 1)/2 = (n + 1)/2$. Thus, we have the following recurrence for this definition of the number of byes $B(n)$:

$$B(n) = \begin{cases} B(\frac{n}{2}) & \text{if } n > 0 \text{ is even,} \\ 1 + B(\frac{n+1}{2}) & \text{if } n > 1 \text{ is odd,} \end{cases} \qquad B(1) = 0.$$

While there is no closed-form (i.e., direct) formula for the solution to this recurrence, it can be obtained by the following algorithm, mentioned by Martin Gardner in his *aha!Insight* [Gar78, p. 6]: $B(n)$ can be computed by counting the number of 1's in the binary representation of the difference $b(n) = 2^k - n$, where $k = \lceil \log_2 n \rceil$. In other words, the number of byes as defined in part (b) is equal to the number of 1's in the binary representation of the number of byes as defined in part (a). For example, for $n = 10$, this algorithm computes $b(10) = 2^4 - 10 = 6 = 110_2$ and therefore yields 2 byes, whereas for $n = 9$, it computes $b(9) = 2^4 - 9 = 7 = 111_2$ and therefore yields 3 byes.

So let $G(n)$ be the number obtained by Gardner's algorithm for a positive integer n ($2^{k-1} < n \le 2^k$); we will show that $G(n)$ satisfies the recurrence equation and initial condition given above for the number of byes $B(n)$. If $n = 1$, $2^0 - 1 = 0$, and hence $G(1)$, the number of 1's in the binary representation of 0, is equal to 0. Let n be a positive even number; $2^{k-1} < n \le 2^k$ and hence $2^{k-2} < \frac{n}{2} \le 2^{k-1}$. Since $b(n) = 2^k - n$ is even in this case, it has a 0 as its rightmost binary digit. Therefore,

$$b\left(\frac{n}{2}\right) = 2^{k-1} - \frac{n}{2} = \frac{b(n)}{2}$$

has the same number of 1's as $b(n)$. Hence $G(n) = G(\frac{n}{2})$ for any positive even n.

Let n be an odd integer greater than 1; $2^{k-1} < n < 2^k$ and hence $2^{k-2} < \frac{n+1}{2} \le 2^{k-1}$. By definition, $G(\frac{n+1}{2})$ is equal to the number of 1's in the binary representation of $b(\frac{n+1}{2}) = 2^{k-1} - \frac{n+1}{2}$. Since $b(n) = 2^k - n$ is odd in this

case, it has a 1 as its rightmost binary digit. Dropping that last 1 in the binary representation of $2^k - n$ yields the binary representation of

$$(2^k - n - 1)/2 = 2^{k-1} - (n+1)/2 = b\left(\frac{n+1}{2}\right).$$

Hence, for any odd $n > 1$, we get $G(n) = 1 + G(\frac{n+1}{2})$, which is exactly what we needed to show.

Comments Our solution to part (b) is based on interpreting a single-elimination tournament as a divide-by-half algorithm. One can get another solution by adding $2^{\lceil \log_2 n \rceil} - n$ imaginary players to get a simpler instance of the problem. The number of byes is then obtained by counting the number of games between the imaginary and real players. This approach leads to the alternative formula for the numbers of byes, which is the number of 0's in the binary representation of $n - 1$, where $n > 1$ is the number of real players in the tournament (see [MathCentral], the October 2009 problem of the month).

By the way, beyond their usefulness for tournament organization, tournament trees have applications in computer science (see, e.g., [Knu98]).

117. One-Dimensional Solitaire

Solution Assuming that the board's cells are numbered consecutively from 1 to n, the empty cell at the start may be in locations 2 or 5 (symmetrically, $n - 1$ or $n - 4$), and the remaining peg may end up at either $n - 1$ or $n - 4$ (symmetrically, 2 or 5).

The solution follows from investigation of the pattern that arises after any first move

$$\underbrace{1\ldots1}_{l}00\underbrace{1\ldots1}_{r}$$

where 1's and 0's represent cells with and without pegs, respectively. Without loss of generality, we may assume that $l \le r$.

If $l = 0$ and $r = 2$, after the only possible jump by the last peg, it will remain a single peg on the board; if $l = 0$ and $r > 2$, after $\lfloor r/2 \rfloor$ compulsory jumps to the left, $\lceil r/2 \rceil \ge 2$ pegs separated by empty cells will remain.

Similarly, if $l = 1$ and $r \ge 1$, after $\lfloor r/2 \rfloor$ compulsory jumps to the left, $\lceil r/2 \rceil + 1 \ge 2$ pegs separated by empty cells will remain.

If $l = 2$ and $r \ge 2$ and even, all the pegs but one can be eliminated by a series of permitted jumps, which is easy to prove by induction on r. Indeed, if $r = 2$, this it achieved by jumps of the first, last, and then either of the two remaining pegs:

$$110011 \implies 001011 \implies 001100 \implies \text{either } 010000 \text{ or } 000010.$$

The same first two jumps reduce the position with an even $r > 2$ pegs to the same pattern with $r - 2$ pegs, which proves the inductive step:

$$110\underbrace{11\ldots1}_{r} \implies 001\underbrace{011\ldots1}_{r} \implies 0011001\underbrace{\ldots1}_{r-2}$$

Moreover, since it is impossible to reduce $0\underbrace{11\ldots1}_{r}$, where $r > 2$, to a single peg, there is just one way to reduce $110\underbrace{11\ldots1}_{r}$ to a single peg—the one described above.

Finally, if $l > 2$ and $r \geq l$, it is impossible to reduce the position to a single peg. Indeed, neither $l > 2$ pegs to the left of 00 nor $r > 2$ pegs to the right of them can be reduced to a single peg without "help" from a peg from the other side. Such a peg can be brought in with a jump from the other side over the neighboring peg, but the pegs on both sides can not be reduced to a single peg in this fashion:

$$\underbrace{1\ldots110}_{l}\underbrace{011\ldots1}_{r} \implies \ldots \implies \underbrace{1\ldots1}_{l-2}001100\underbrace{1\ldots1}_{r-2}.$$

Comments The main tool in solving the puzzle is the decrease-and-conquer strategy. Using the same approach, one can also show that the only board with an odd number of cells that has a solution is of size 3, with the empty cell either at 1 or 3. C. Moore and D. Eppstein [Moo00] gave a formal description of all positions that can arise in this game.

The puzzle is a one-dimensional version of the old but still popular game played by the same rules on two-dimensional boards. For an in-depth account of the game's history and winning strategies, see, for example, the monograph by John D. Beasley [Bea92] and Chapter 23 in *Winning Ways for Your Mathematical Plays* [Ber04], not to mention several websites devoted to it.

118. Six Knights

Solution The minimum number of moves needed to solve the puzzle is 16.

As explained in the book's first tutorial, the initial state of the puzzle can be conveniently represented by the graph in Figure 4.81.

The graph can be unfolded as shown in Figure 4.82: move vertex 8 up and vertex 5 down to get the first figure there; swap the locations of vertices 4 and 6 to get the second figure; swap the locations of vertices 10 and 12 to get the third one; move vertex 11 up and vertex 2 down to get the unfolded representation of the puzzle's graph.

To achieve the puzzle's objective, one of the two black knights at initial positions 1 and 3 has to move to either 12 or 10. Without loss of generality, we consider the knight at 1 moving to 12, which is closer than 10 to its initial position. Taking the shortest path, it will take 3 moves. The other two black knights will need at least 4 moves total to get from their initial positions at 2 and 3 to their goal

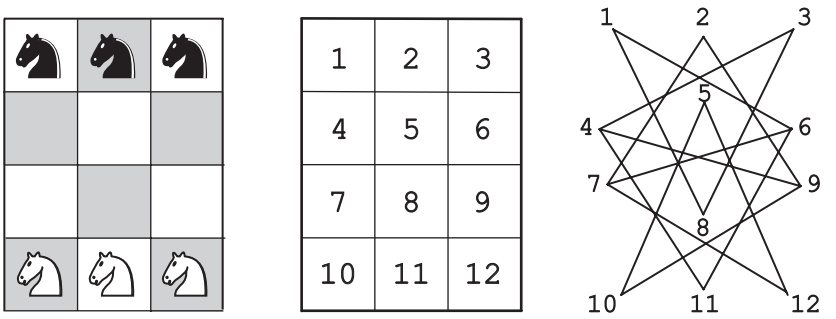


FIGURE 4.81 Graph representing possible moves in the *Six Knights* puzzle.

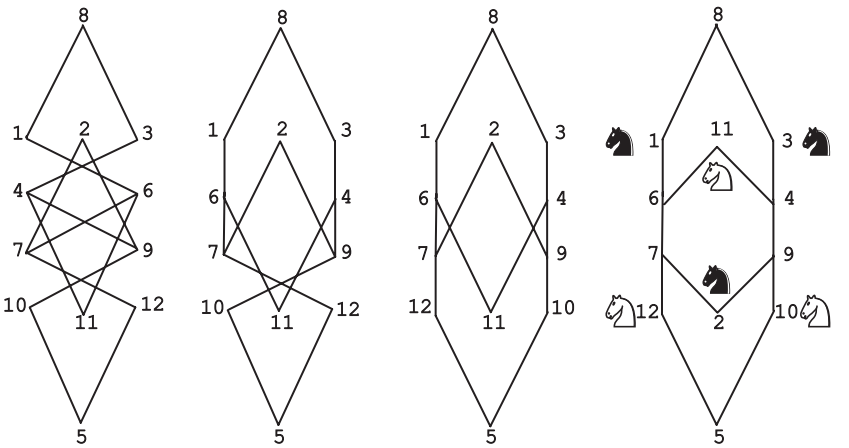


FIGURE 4.82 Unfolding of the graph in Figure 4.81.

positions at 10 and 11. The symmetry implies the same lower bound of 7 moves for the white knights to reach their goal positions. Thus, the puzzle cannot be solved in fewer than 14 moves. This cannot be done in 14 or 15 moves, however, which is easy to prove by contradiction. Assume that there is a sequence of moves that solves the puzzle in less than 16 moves. In such a sequence, the black knight at 1 will have to reach 12 in 3 moves because it cannot do this in 4 moves and if it does this in 5 or more moves the total number of moves will be at least $(5 + 4) + 7 = 16$. But before the black knight in 1 can move from 6 to 7, the white knight at 12 must go to 2, which must be preceded by the black knight at 2 move to 9, which must be preceded by the white knight at 10 move to 4, which must be preceded by the black knight at 3 move to 11, which must be preceded by the white knight at 11 move to 6, blocking the black knight at 1 and thus making the necessary number of moves exceed 15. But the puzzle can be solved in 16 moves, for example, those in the following sequence:

$$B(1-6-7),\ W(11-6-1),\ B(3-4-11),\ W(10-9-4-3),$$
$$B(2-9-10),\ B(7-6),\ W(12-7-2),\ B(6-7-12).$$

Comments The puzzle is a well-known extension of *Guarini's Puzzle*, discussed in the tutorial on algorithm design strategies. The principal idea in both puzzles is representation change: first, the board is represented by a graph and then the graph is unfolded to give a clearer picture of the task at hand.

The *Journal of Recreational Mathematics* published two papers with incorrect answers to this problem in 1974 and 1975 [Sch80, pp. 120–124]. Dudeney [Dud02, Problem 94] considered the puzzle with an additional constraint that no move may put two knights of opposite color in a position to attack each other. For other variations, see Loyd's "Knights Crossing over the Danube" (e.g., [Pet97, pp. 57–58]) and Grabarchuk's *The New Puzzle Classics* [Gra05, pp. 204–206].

119. Colored Tromino Tiling

Solution The puzzle can be solved by the following recursive algorithm. If $n = 2$, divide the board into four 2×2 boards and place one gray tromino to cover the three central squares that are not in the 2×2 board with the missing square. Then place one black tromino in the upper left 2×2 board, one white tromino in the upper right 2×2 board, one black tromino in the lower right 2×2 board, and one white tromino in the lower left 2×2 board (see Figure 4.83).

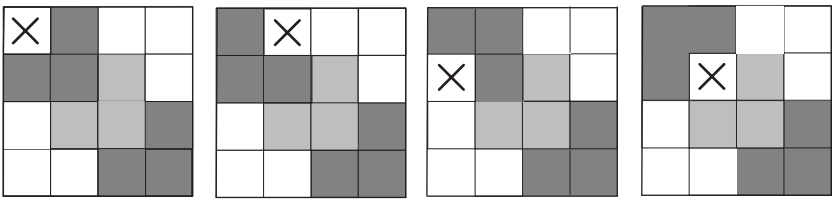


FIGURE 4.83 Colored tromino tiling of a 4×4 board with four possible locations of a missing square (marked by X).

Note that for any location of the missing square, the following properties hold:

- Going left to right, the upper edge of the board is covered by an alternating sequence of two black squares followed by two white squares, with the missing square possibly replacing one square in this sequence.
- Going down, the right edge of the board is covered by an alternating sequence of two white squares followed by two black squares, with the missing square possibly replacing one square in this sequence.
- Going right to left, the lower edge of the board is covered by an alternating sequence of two black squares followed by two white squares, with the missing square possibly replacing one square in this sequence.
- Going up, the left edge of the board is covered by an alternating sequence of two white squares followed by two black squares, with the missing square possibly replacing one square in this sequence.

If $n > 2$, divide the board into four $2^{n-1} \times 2^{n-1}$ boards and place one gray tromino to cover the three central squares that are not in the $2^{n-1} \times 2^{n-1}$ board with the missing square. Then tile each of the three $2^{n-1} \times 2^{n-1}$ boards recursively by the same algorithm (see an example in Figure 4.84).

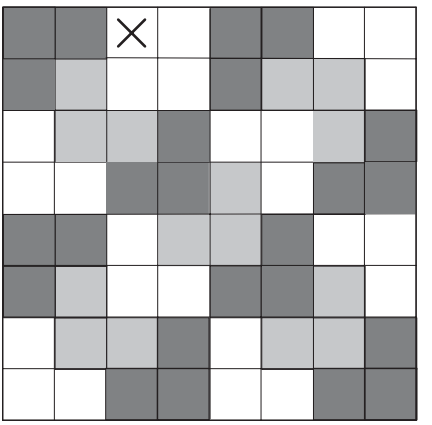


FIGURE 4.84 Colored tromino tiling of a $2^3 \times 2^3$ board with a missing square (marked by X).

The proof of the algorithm's correctness is based on the fact that the properties enumerated above remain valid on each iteration of the algorithm, which can be easily checked by mathematical induction.

Comments As it was the case for the tiling algorithm with noncolored trominoes, which was discussed in the first tutorial, we have here an excellent illustration of the divide-and-conquer strategy.

Both the puzzle and its solution is from a paper by I-Ping Chu and Richard Johnsonbaugh [Chu87].

120. Penny Distribution Machine

Solution a. We assume that the boxes are numbered left to right starting with a 0 for the leftmost box. Let $b_0 b_1 \dots b_k$ be a bit string representing a result of the machine's distribution of n pennies, where b_i is equal to 1 or 0 depending on whether the ith box, $0 \le i \le k$, has a coin or not; in particular, $b_k = 1$, where k is the number of the last box with a penny. This coin was obtained by replacing two coins in box $k - 1$, which in turn replaced four coins in box $k - 2$, and so on. Applying the same reasoning to any box i with a coin in the final distribution, we obtain the formula

$$n = \sum_{i=0}^{k} b_i 2^i.$$

In other words, the bit string of the final distribution represents the binary expansion of the initial number of pennies n in reverse order. Since the binary expansion of any natural number is unique, the machine always ends up with the same coin distribution for a given n, irrespective of an order in which coin pairs are processed.

b. The minimum number of boxes needed to distribute n pennies is equal to the number of bits in the binary expansion of n, which is $\lfloor \log_2 n \rfloor + 1 = \lceil \log_2 (n+1) \rceil$.

c. Let $b_k b_{k-1} \dots b_0$ be the binary expansion of n. According to the answer to part (a), when the machine stops, the number of pennies in box i is equal to b_i, $0 \le i \le k$. We have the following recurrence for the number of iterations needed to place one penny in box i:

$$C(i) = 2C(i-1) + 1 \text{ for } 0 < i \le k, \quad C(0) = 0.$$

Solving the recurrence by backward substitutions yields the following:

$$\begin{aligned} C(i) &= 2C(i-1) + 1 \\ &= 2(2C(i-2) + 1) + 1 = 2^2 C(i-2) + 2 + 1 \\ &= 2^2(2C(i-3) + 1) + 2 + 1 = 2^3 C(i-3) + 2^2 + 2 + 1 \\ &= \dots \\ &= 2^i C(i-i) + 2^{i-1} + 2^{i-2} + \dots + 1 = 2^i \cdot 0 + (2^i - 1) = 2^i - 1. \end{aligned}$$

Hence, the total number of iterations made by the machine before stopping can be found as

$$\sum_{i=0}^{k} b_i C(i) = \sum_{i=0}^{k} b_i (2^i - 1) = \sum_{i=0}^{k} b_i 2^i - \sum_{i=0}^{k} b_i = n - \sum_{i=0}^{k} b_i,$$

which is the difference between the number of pennies given at the start and the number of 1's in its binary expansion.

Comments The puzzle takes advantage of n's binary expansion, which serves as an invariant here, and backward thinking. It is a minor modification of a problem invented by James Propp (see [MathCircle]).

121. Super-Egg Testing

Solution The answer is 14.

Let $H(k)$ be the maximum number of floors for which the problem can be solved in k drops. The first drop has to be made from floor k because if the egg breaks, each of the lower $k-1$ floors will need to be tested sequentially starting with the first floor. If the first drop does not break the egg, the second drop has to be made from floor $k + (k-1)$ to be prepared for the possibility that if the egg breaks, each

of the $k-2$ floors from floor $k+1$ to floor $2k-2$ need to be tested sequentially. On repeating this argument for the remaining $k-2$ drops, we obtain the following formula for $H(k)$:

$$H(k) = k + (k-1) + \cdots + 1 = k(k+1)/2.$$

(Alternatively, one can get the same formula for $H(k)$ by solving the recurrence $H(k) = k + H(k-1)$ for $k > 1$, $H(1) = 1$.)

What remains to be done to answer the puzzle's question is to find the smallest value of k such that $k(k+1)/2 \geq 100$. This value is $k = 14$. The first egg can be dropped from floors 14, 27, 39, 50, 60, 69, 77, 84, 90, 95, 99, and 100 until it breaks, and if this happens, the second egg is to be dropped from each consecutive floor starting with the next floor after the last successfully tested one. This solution is not unique: the first drop can also be executed from floors 13, 12, 11, and 10, with corresponding adjustments of the other drops, of course.

Comments While the algorithm solving the puzzle is clearly based on the greedy approach, it is rather unusual in analyzing the worst case backward: it finds the largest size of the problem for a given number of times its basic operation (the egg drop) is executed.

After its appearance in the book by Joseph Konhauser et al. [Kon96, Problem 166], this puzzle has become quite popular: see, for example, Peter Winkler's *Mathematical Mind-Benders* [Win07, p. 10] and a paper by Moshe Sniedovich [Sni03].

122. Parliament Pacification

Solution The answer is "true": the following algorithm will divide the parliament into two chambers in such a way that no parliamentarian has more than one enemy in his or her chamber.

Start by dividing all the parliamentarians into two chambers in an arbitrary way. (For example, divide them about equally between the chambers.) Let p be the total number of enemy pairs in both chambers who are currently in the same chamber. As long as there is a parliamentarian who has at least two enemies in the same chamber, transform this parliamentarian into the other chamber. Since the transformed parliamentarian will have no more than one enemy in the new chamber and the number of enemy pairs in his old chamber will decrease by two, the total number of enemy pairs will decrease at least by one. Hence, the algorithm will terminate after no more than p iterations, with a final partition of the parliamentarians in which every parliamentarian has no more than one enemy in the same chamber.

Comments One can consider the algorithm solving the puzzle as based on the iterative improvement strategy, which is outlined in the book's first tutorial.

A detailed discussion of this strategy, along with several important algorithms using it, can be found in A. Levitin's textbook [Lev06, Chapter 10].

The earliest reference to the puzzle we have been able to find is Problem M580 (p. 38) in the August 1979 issue of *Kvant*, a Russian popular science magazine in physics and mathematics for school students and teachers. It was later included in such books in English as *Mathematical Miniatures* by S. Savchev and T. Andreescu [Sav03, p. 1, Problem 4].

123. Dutch National Flag Problem

Solution The following algorithm is based on considering the checker row as made up of four contiguous possibly empty sections: red checkers on the left, then white checkers, then the checkers whose colors are yet to be identified, and finally blue checkers:

all red	all white	unknown	all blue

Initially, the red, white, and blue sections are empty, with all the checkers being in the unknown section. On each iteration, the algorithm shrinks the size of the unknown section by one element either from the left or from the right: If the first (i.e., leftmost) checker in the unknown section is red, swap it with the first checker after the red section and advance to the next checker; if it is white, advance to the next checker; if it is blue, swap it with the last checker before the blue section. (See an illustration in Figure 4.85.) This step is repeated as long as there are checkers in the unknown section.

Comments Both the problem and the algorithm were proposed by W. H. J. Feijen; it was made famous in computer science circles by Edsger Dijkstra, a prominent computer scientist, by inclusion in his book *A Discipline of Programming* [Dij76, Chapter 14]. The colorful name of the problem has an obvious explanation: both Feijen and Dijkstra were Dutch, and red, white, and blue are colors of the Dutch national flag.

124. Chain Cutting

Solution The answer is the smallest integer k such that $(k+1)2^{k+1} - 1 \geq n$.

As suggested in the hint to this puzzle, it is easier to reverse the question and find first $n_{\max}(k)$, the maximum length of the chain for which the problem can be solved by removing k single clips, where k is a given positive integer. If we remove k clips spread along the chain, the chain will split into $k+1$ pieces. The shortest of them, S_1, must be $k+1$ clips long: together with k removed clips, it will enable us to combine them in a chain of any length from 1 to $k + (k+1) = 2k+1$. The second shortest piece, S_2, should be one clip longer, that is $(2k+1)+1 = 2(k+1)$ clips long. With it, we will be able to make a chain of any length from 1 to $(2k+1) + 2(k+1) = 4k+3$. The third shortest piece, S_3, should be of length

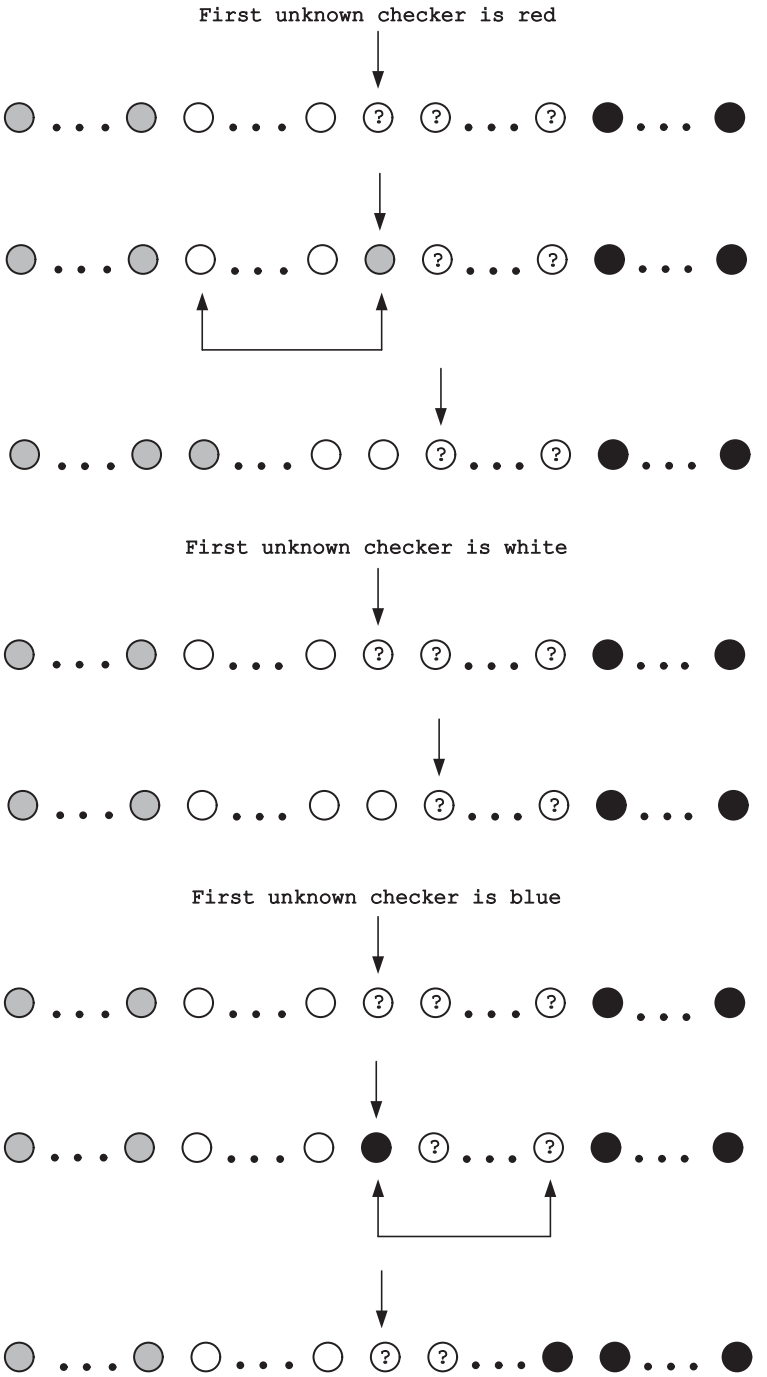
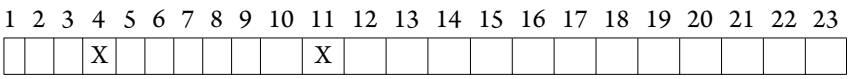


FIGURE 4.85 Three cases of the algorithm for the *Dutch National Flag Problem*.

$(4k+3)+1 = 2^2(k+1)$, and so on, until we get the longest piece, S_{k+1}, of length $2^k(k+1)$. (This assertion can be formally proved by mathematical induction.) Having k removed clips and these $k+1$ pieces, we will be able to get a chain of any integer length from 1 to $n_{\max}(k)$, inclusive, where

$$\begin{aligned} n_{\max}(k) &= k + (k+1) + 2(k+1) + 2^2(k+1) + \cdots + 2^k(k+1) \\ &= k + (k+1)(1 + 2 + 2^2 + \cdots + 2^k) = k + (k+1)(2^{k+1} - 1) \\ &= (k+1)2^{k+1} - 1. \end{aligned}$$

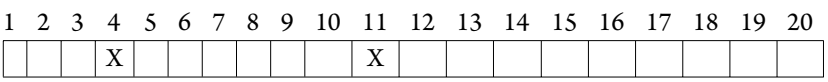
For example, for $k = 2$, we can form a chain of any length between 1 and 23, as illustrated in the following diagram:



Now, to return to the puzzle as stated, let $n > 1$ be the length of a chain given. Obviously, there exists a unique positive integer k such that

$$n_{\max}(k-1) < n \le n_{\max}(k)$$

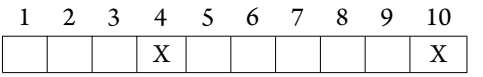
where $n_{\max}(k) = (k+1)2^{k+1} - 1$. From the preceding discussion we know that at least k single clips must be removed from the chain to solve the problem. Now we will show that removing k clips as explained above is, in fact, sufficient to achieve our goal, with a minor adjustment explained below. If $n = n_{\max}(k)$, removing k single clips as described above solves the problem. If $|S_1| + \cdots + |S_k| + k \le n < n_{\max}(k)$, where $|S_1|, \ldots, |S_k|$ denote the lengths of the first k chain pieces formed as above, the last piece, $\tilde{S}_{k+1}$, will be shorter than the last piece S_{k+1} in the above solution. But it is easy to see that this partition of the given chain will still solve the problem. For example, if $n = 20$ and hence $k = 2$, the solution just described requires removing clips 4 and 11:



A chain of any length from 1 to $|S_1| + |S_2| + 2 = 3 + 6 + 2 = 11$ can be obtained as a combination of some of S_1, S_2, and 2 single clips; hence, together with $\tilde{S}_3$, we can also obtain a chain of any length from $|\tilde{S}_3| = 9$ to $|\tilde{S}_3| + 11 = 20$.

If $n_{\max}(k-1) < n < |S_1| + \cdots + |S_k| + k$, the kth single clip can be removed from the right end of the chain given, while the first $k-1$ single clips are removed

at the same positions as above. For example, if $n = 10$ and hence $k = 2$, the clips to be removed are at positions 4 and 10:



A chain of any length from 1 to $|S_1| + 2 = 3 + 2 = 5$ can be obtained as a combination of some of S_1 and 2 single clips; hence, together with $\tilde{S}_2$ of length 5, we can also obtain a chain of any length from 5 to 10.

Comments The above solution (e.g., [Sch80, pp. 128–130]) nicely illustrates the greedy strategy for algorithm design.

The most common version of this puzzle is that of a seven-link gold chain that a traveler needs to cut to pay to an inn's owner for every day of the week the traveler stays at the inn. For a few related references, see D. Singmaster's bibliography [Sin10, Section 5.S.1].

125. Sorting 5 in 7

Solution It is helpful to start with a brief investigation of a natural but ultimately fruitless idea: sort four weights first and then find an appropriate place for the fifth among them. Sorting four numbers by comparisons cannot be done in less than five comparisons. (It is not difficult to prove this directly; it also follows from the general theorem that the sorting of n arbitrary real numbers by comparisons requires at least $\lceil \log_2 n! \rceil$ comparisons in the worst case.) Further, it is easy to see that to find a proper place of the fifth number among four sorted numbers requires at least three more comparisons. (This is especially clear if one considers the sorted numbers as points on the real line.) The conclusion from this failure is that a desired algorithm must involve all five numbers before four of them are completely sorted.

Here is an algorithm for solving the problem. Order the items arbitrarily from 1 to 5. Let $w_1, \ldots, w_5$ be the unknown weights of the items. Start by weighing items 1 and 2, and items 3 and 4. Without loss of generality, we may assume that $w_1 < w_2$ and $w_3 < w_4$. Weigh the heavier items found in the first two weighings, that is, compare w_2 and w_4. Two possible cases may arise as the results of these three weighings (and the assumption made):

$$\text{Case 1: } w_1 < w_2 < w_4 \quad \text{and } w_3 < w_4,$$
$$\text{Case 2: } w_3 < w_4 < w_2 \quad \text{and } w_1 < w_2.$$

Clearly, Case 2 is analogous to Case 1 since they differ only in the roles played by the first and second pair of weights participating in the first two weighings. Therefore, with no loss of generality, we will consider only Case 1 as the representative of the two. It is helpful, as mentioned above, to represent the

numbers involved in sorting as points on the real line. So, after the first three comparisons indicated above, we have the following diagram:

$$-w_1 - w_2 - w_4 - \quad \text{and } w_3 < w_4 \text{ (point } w_3 \text{ is to the left of } w_4).$$

As the fourth weighing, compare w_5 and w_2. If $w_5 < w_2$, as the fifth weighing compare w_5 with w_1 to obtain one of the two situations:

$$\text{If } w_5 < w_1\text{:} \quad -w_5 - w_1 - w_2 - w_4 - \quad \text{and } w_3 < w_4 \text{ (w_3 is to the left of } w_4).$$

$$\text{If } w_5 > w_1\text{:} \quad -w_1 - w_5 - w_2 - w_4 - \quad \text{and } w_3 < w_4 \text{ (w_3 is to the left of } w_4).$$

Since these two situations differ only by the relative order of w_1 and w_5, we will give further comparisons only for the first of these situations. As the sixth weighing compare w_3 and w_1. If $w_3 < w_1$, as the seventh weighing compare w_3 and w_5. If $w_3 < w_5$, we obtain $w_3 < w_5 < w_1 < w_2 < w_4$; if $w_3 > w_5$, we obtain $w_5 < w_3 < w_1 < w_2 < w_4$. Similarly, if on the sixth weighing $w_3 > w_1$, as the seventh weighing compare w_3 and w_2. If $w_3 < w_2$, we obtain $w_5 < w_1 < w_3 < w_2 < w_4$; if $w_3 > w_2$, we obtain $w_5 < w_1 < w_2 < w_3 < w_4$.

Now we consider the situation that arises if $w_5 > w_2$ on the fourth weighing. As the fifth weighing, compare w_5 with w_4 to obtain one of the two situations:

$$\text{If } w_5 < w_4\text{:} \quad -w_1 - w_2 - w_5 - w_4 - \quad \text{and } w_3 < w_4 \text{ (w_3 is to the left of } w_4).$$

$$\text{If } w_5 > w_4\text{:} \quad -w_1 - w_2 - w_4 - w_5 - \quad \text{and } w_3 < w_4 \text{ (w_3 is to the left of } w_4).$$

In the first of these situations, as the sixth weighing compare w_3 and w_2. If $w_3 < w_2$, as the seventh weighing compare w_3 and w_1. If $w_3 < w_1$, we obtain $w_3 < w_1 < w_2 < w_5 < w_4$; if $w_3 > w_1$, we obtain $w_1 < w_3 < w_2 < w_5 < w_4$. In the second situation, as the sixth weighing also compare w_3 and w_2. If $w_3 < w_2$, as the seventh weighing compare w_3 and w_1. If $w_3 < w_1$, we obtain $w_3 < w_1 < w_2 < w_4 < w_5$; if $w_3 > w_1$, we obtain $w_1 < w_3 < w_2 < w_4 < w_5$. Finally, if $w_3 > w_2$ on the sixth weighing, no seventh weighing is necessary since we know that $w_3 < w_4$; hence, we have $w_1 < w_2 < w_3 < w_4 < w_5$.

Comments The puzzle presents a well-known problem in sorting small-size files. Donald Knuth discusses it in Section 5.3.1, Volume 3, of his *The Art of Computer Programming* [Knu98]. In particular, he gives a very elegant diagramming way to present the above algorithm discovered by H. B. Demuth (pp. 183–184).

126. Dividing a Cake Fairly

Solution A fair solution for two people is to let one of them cut the cake in two pieces and let the other choose his piece. For more than two people, this procedure can be generalized as follows. First, all the persons are assigned numbers from 1 to n. Person 1 cuts off a piece X of the cake. He will have an incentive to make it

as close to $1/n$th of the cake as possible: if he makes it smaller than that, he might end up getting this piece, if he makes it larger, the piece will likely be decreased by the others. Person 2 will have an option to cut off a slice of X, if he believes that X is larger than $1/n$th of the cake, and add the slice to the remaining portion of the cake. If Person 2 does not believe that X is larger than $1/n$th of the cake, he does nothing. Persons 3, 4, . . . , n exercise in turn the same option of either decreasing X or doing nothing. Then the last person to touch X takes it for himself, and the remaining $n - 1$ persons apply the same procedure to divide the remaining part of the cake, which includes all the extra slices, if any, added to adjust X. When just two people are left, they divide the remaining part (with all the extra slices) by one person cutting and the other choosing.

Comments The algorithm is clearly based on the decrease-by-one strategy.

According to Ian Stewart [Ste06, pp. 4–5], the above solution for two people has been known for 2800 years, and its extension for three people was given by the Polish mathematician Hugo Steinhaus in 1944. Since then, several different algorithms have been suggested for the original problem as well as for its variations and extensions. The interested reader will find them in the monograph by Jack Robertson and William Webb [Rob98].

127. The Knight's Tour

Solution Because of the board's symmetry, we can, without loss of generality, start the knight's tour at the upper left corner of the board and finish it at the square numbered 64 in Figure 4.86. We will always move to an unvisited square that is as close as possible to the nearest edge of the board. More precisely, we will try first visiting squares in the two outer layers of the board, jumping to one of the

1	38	17	34	3	48	19	32
16	35	2	49	18	33	4	47
39	64	37	54	59	50	31	20
36	15	56	51	62	53	46	5
11	40	63	60	55	58	21	30
14	25	12	57	52	61	6	45
41	10	27	24	43	8	29	22
26	13	42	9	28	23	44	7

FIGURE 4.86 Knight's closed tour of a standard chessboard.

16 central squares only as the last resort. In addition, we will always visit a corner square as soon as it becomes feasible to do so. A tour generated by these rules is shown in Figure 4.86, in which the squares are numbered sequentially in the order they are visited by the knight.

Comments *The Knight's Tour* is one of the two most studied puzzles involving a chessboard (the other is the *n-Queens Problem* discussed in the first tutorial and in #140). The corresponding section of D. Singmaster's annotated bibliography [Sin10, Section 5.F.1] runs for eight pages and goes back to the 9th century. Over this time, it has attracted the attention of several noted mathematicians, including the greats Leonhard Euler and Carl Gauss. Since there are more than 10^{13} closed knight's tours on an 8×8 board, it is surprising that finding even one of them with no computer help is not a trivial task.

The tour given above is based on the idea proposed at the beginning of the 18th century by Montmort and De Moivre for constructing an open tour, that is, a tour that need not return to a starting square. The idea clearly has a greedy flavor by giving preference to squares reachable from fewer other squares. This greedy thinking is used more formally by the method suggested a century later by Warnsdorff: among available alternatives, proceed to a square with the fewest possible choices for the next move. Warnsdorff's method is more powerful of the two greedy methods, but it is more computationally demanding. It is important to note that both these methods are just *heuristics*: logical rules of thumb that may fail to lead to a solution (in our case, by an unlucky break of a tie). In general, the idea of a heuristic-based algorithm is an important one for difficult computational problems. For other algorithms for generating knight's tours—several of which based on a version of the divide-and-conquer strategy—see, for example, [Bal87, pp. 175–186], [Kra53, pp. 257–266], [Gik76, pp. 51–67], and quite a few websites devoted to this problem and its extensions.

It is also noteworthy that the *Knight's Tour* problem is a special case of trying to find a Hamilton circuit, and Warnsdorff's heuristic is one that is regularly used for trying to find Hamilton circuits.

128. Security Switches

Solution The puzzle can be solved in the minimum of $\frac{2}{3}2^n - \frac{1}{6}(-1)^n - \frac{1}{2}$ switch toggles.

We will number the switches left to right from 1 to n and denote the "on" and "off" states of a switch by a 1 and 0, respectively. To help ourselves with solving the general instance of the puzzle, let us start by solving its four smallest instances (see Figure 4.87).

Consider now the general instance of the puzzle represented by the bit string of n 1's: 111. . . 1. Before we can turn off the first (leftmost) switch, the switches should be in the state 110. . . 0. Hence, to begin with, we need to turn off the last $n - 2$ switches and do this in the minimum number of moves if we want to have an optimal algorithm. In other words, we should first solve the same problem as

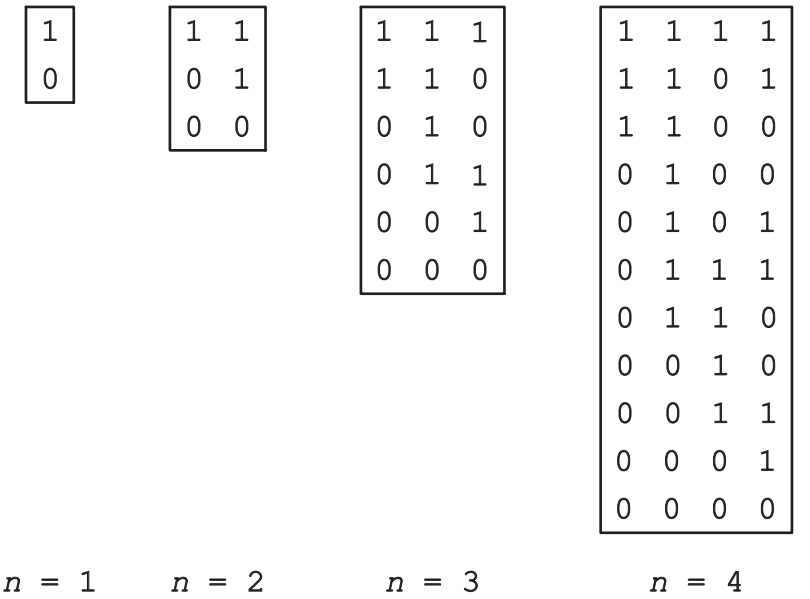


FIGURE 4.87 Optimal solutions for the first four instances of the *Security Switches* puzzle.

given for the last $n - 2$ switches. This can be done recursively with the $n = 1$ and $n = 2$ instances solved directly, as shown in Figure 4.87. After that, we can toggle the first switch to get $010\ldots0$. Now, before we can toggle the second switch we will need to pass through the state with all the switches following it "on," which can be easily proved by mathematical induction. Getting all the switches from the third to the last one "on" can be achieved by reversing the optimal sequence of moves made previously to toggle the last $n - 2$ switches from the "on" position to the "off" position. This will give us $011\ldots1$. Ignoring the first 0, we have now the $n - 1$ instance of the original puzzle, which can be solved recursively.

Let $M(n)$ be the number of moves (switch toggles) made by the above algorithm. We have the following recurrence for $M(n)$:

$$M(n) = M(n-2) + 1 + M(n-2) + M(n-1) \text{ or}$$
$$M(n) = M(n-1) + 2M(n-2) + 1 \text{ for } n \geq 3,\ M(1) = 1,\ M(2) = 2.$$

Solving the recurrence by the standard technique for second-order linear nonhomogeneous recurrence relations with constant coefficients (see, e.g., [Lev06, pp. 476–478] or [Ros07, Sec. 7.2]), we obtain the following closed-form solution:

$$M(n) = \frac{2}{3}2^n - \frac{1}{6}(-1)^n - \frac{1}{2} \text{ for } n \geq 1.$$

For even n's, this formula reduces to $M(n) = (2^{n+1} - 2)/3$; for odd n's, it yields $M(n) = (2^{n+1} - 1)/3$.

Comments The algorithm solving the puzzle is based on the decrease-and-conquer strategy. Although solving the second-order recurrence for the number of moves by applying the standard techniques is both natural and easy, one can avoid this by following either Ball and Coxeter [Bal87, pp. 318–320], or Averbach and Chein [Ave00, p. 414]. In our view, both methods are more cumbersome than the above solution. An entirely different approach was proposed by the French mathematician Louis A. Gros in 1872. His method amounted to representing states of the switches by bit strings anticipating modern-day Gray codes; for details see [Bal87, pp. 320–322] and [Pet09, 182–184].

The puzzle was proposed by C. E. Greenes [Gre73]. It imitates operations of a very old and well-known mechanical puzzle called the *Chinese Rings*. The abounding literature on the Chinese Rings is annotated by D. Singmaster [Sin10, Sec. 7.M.1]; a few recent references can be found in M. Petković 's book [Pet09, p. 182].

129. Reve's Puzzle

Solution Since the problem is an obvious extension of the *Tower of Hanoi* puzzle, it is natural to use a similar recursive approach. Namely, if $n > 2$, transfer k smallest disks to an intermediate peg recursively using all four pegs, then move the remaining $n - k$ disks to the destination peg by the classic recursive algorithm for the three-peg *Tower of Hanoi* puzzle (see, e.g., the second tutorial), and, finally, transfer the k smallest disks to the destination peg recursively using all four pegs. If $n = 1$ or 2, solve the trivial instances of the problem in one and three moves, respectively, as it is done in the three-peg *Tower of Hanoi* solution. The value of parameter k can be selected to minimize the total number of disk moves made by the algorithm. This leads to the following recurrence relation for the number of moves, $R(n)$, made by this algorithm to move n disks:

$$R(n) = \min_{1 \le k < n} [2R(k) + 2^{n-k} - 1] \text{ for } n > 2, \ R(1) = 1, \ R(2) = 3.$$

Starting with $R(1) = 1$ and $R(2) = 3$ and using this recurrence, we can find successively the values of $R(3), R(4), \ldots, R(8)$. These values are shown in bold in the tables below.

n	k	$2R(k) + 2^{n-k} - 1$	n	k	$2R(k) + 2^{n-k} - 1$
3	1	$2 \cdot 1 + 2^2 - 1 = \mathbf{5}$	5	1	$2 \cdot 1 + 2^4 - 1 = 17$
	2	$2 \cdot 3 + 2^1 - 1 = 7$		2	$2 \cdot 3 + 2^3 - 1 = \mathbf{13}$
				3	$2 \cdot 5 + 2^2 - 1 = \mathbf{13}$
4	1	$2 \cdot 1 + 2^3 - 1 = \mathbf{9}$		4	$2 \cdot 9 + 2^1 - 1 = 19$
	2	$2 \cdot 3 + 2^2 - 1 = \mathbf{9}$			
	3	$2 \cdot 5 + 2^1 - 1 = 11$			

n	k	$2R(k)+2^{n-k}-1$	n	k	$2R(k)+2^{n-k}-1$
6	1	$2\cdot 1+2^5-1=33$	7	1	$2\cdot 1+2^6-1=65$
	2	$2\cdot 3+2^4-1=21$		2	$2\cdot 3+2^5-1=37$
	3	$2\cdot 5+2^3-1=\mathbf{17}$		3	$2\cdot 5+2^4-1=\mathbf{25}$
	4	$2\cdot 9+2^2-1=21$		4	$2\cdot 9+2^3-1=\mathbf{25}$
	5	$2\cdot 13+2^1-1=27$		5	$2\cdot 13+2^2-1=29$
				6	$2\cdot 17+2^1-1=35$

n	k	$2R(k)+2^{n-k}-1$
8	1	$2\cdot 1+2^7-1=129$
	2	$2\cdot 3+2^6-1=69$
	3	$2\cdot 5+2^5-1=41$
	4	$2\cdot 9+2^4-1=\mathbf{33}$
	5	$2\cdot 13+2^3-1=\mathbf{33}$
	6	$2\cdot 17+2^2-1=37$
	7	$2\cdot 25+2^1-1=51$

Thus, there are several ways to transfer eight disks in 33 moves according to the above computations. In particular, for the eight-disk instance of the puzzle, one can always use $k = n/2$ on each iteration of the algorithm.

Comments The main algorithmic idea behind the solution is decrease-and-conquer. Here the algorithm explicitly seeks the optimal size reduction on each iteration.

The extension of the *Tower of Hanoi* puzzle to more than three pegs was suggested by the French mathematician Édouard Lucas in 1889, who invented the original three-peg version a few years earlier. Under the name *The Reve's Puzzle*, it appeared in Henry E. Dudeney's first puzzle book *The Canterbury Puzzles* [Dud02], where he gave the solutions for $n = 8$, 10, and 21. Later, more detailed analyses of the above algorithm have produced alternative formulas for the optimal value of the partition parameter k: for example, according to Ted Roth [Schw80, pp. 26–29],

$$k = n - 1 - m \text{ where } m = \lfloor \sqrt{8n-7} - 1)/2 \rfloor,$$
$$R(n) = [n - 1 - m(m-1)/2]2^m + 1,$$

and Michael Rand [Ran09] simplified it further to

$$k = n - \lfloor \sqrt{2n} + 0.5 \rfloor.$$

The algorithm's extension to an arbitrary number of pegs is known as the *Frame-Stewart algorithm*. It is presumed to be optimal for any number of pegs, but this conjecture has never been proved. For some other references, see David Singmaster's annotated bibliography [Sin10, Section 7.M.2.a].

130. Poisoned Wine

Solution a. The poisoned barrel can be identified in 30 days as follows. Number the barrels from 0 to 999 and represent these numbers by 10-bit strings, padding, if necessary, their binary expansions with leading zeros. For example, barrels 0 and 999 will be represented by 0000000000 and 1111100111, respectively. Make the first slave drink from each of the barrels that have 1 as their rightmost bit, the second slave from each of the barrels that have 1 as their second from the right bit, and so on, until the tenth slave is made to drink from each of the barrels that have 1 as their leftmost bit. Note that each slave may have a "cocktail" made of the wine from all the barrels assigned to him, because the poison dilution by the wine from the good barrels does not effect the poison's potency. In 30 days, the poisoned barrel can be determined by its bit string as follows: if the ith slave $(1 \le i \le 10)$—who was "responsible" for the ith bit from the right—dies from the poison, set the ith bit of the poisoned barrel to 1; otherwise, set it to 0. For example, if only slaves 1, 3, and 10 die from the poison in 30 days, the number of the poisoned barrel is $2^0 + 2^2 + 2^9 = 517$.

b. To achieve his goal with just eight slaves, the king can divide the barrels into four groups of 250 each. Since $2^8 > 250$, eight slaves are enough to identify the poisoned barrel in one group in 30 days by the algorithm analogous to the procedure outlined in part (a). Since the poison kills a person in exactly 30 days, the king can apply the same algorithm to the second, third, and fourth barrel group after 1 day, 2 days, and 3 days, respectively. Since there is just one poisoned barrel, the day the slaves die will uniquely identify the group containing the poisoned barrel and a particular subset of dying slaves will determine the poison barrel's number in the group.

Comments Identifying a number by bits of its binary representation is a very old idea, going back at least 500 years (see [Sin10, Section 7.M.4]). It is closely related to *binary search* (e.g., [Lev06, Section 4.3]), with the advantage of the iterations performed in parallel. Part (b) takes the parallelism idea one step further.

Different versions of the puzzle appeared in Martin Gardner's November 1965 column in *Scientific American* (republished in [Gar06, Problem 9.23]) and Dennis Shasha's *Doctor Ecco's Cyberpuzzles* [Sha02, pp. 16–22]. The puzzle has also been actively discussed on interview puzzle sites on the Internet.

131. Tait's Counter Puzzle

Solution Here is a solution for $n = 3$. It is the only instance of the puzzle whose solution uses four spaces to the left of the line given; all the others use only two.

				1	2	3	4	5	6
				B	W	B	W	B	W
		W	B	B			W	B	W
		W	B	B	B	W	W		
W	W	W	B	B	B				

Here is a solution for $n = 4$. Note the WBBW_ _BBWW pattern, which enables two obvious last moves:

		1	2	3	4	5	6	7	8
		B	W	B	W	B	W	B	W
W	B	B	W	B	W	B			W
W	B	B	W			B	B	W	W
W			W	B	B	B	B	W	W
W	W	W	W	B	B	B	B		

Here is a solution for $n = 5$:

		1	2	3	4	5	6	7	8	9	10
		B	W	B	W	B	W	B	W	B	W
W	B	B	W	B	W	B	W	B			W
W	B	B	W			B	W	B	B	W	W
W	B	B	W	W	B	B			B	W	W
W			W	W	B	B	B	B	B	W	W
W	W	W	W	W	B	B	B	B	B		

Here is a solution for $n = 6$:

		1	2	3	4	5	6	7	8	9	10	11	12
		B	W	B	W	B	W	B	W	B	W	B	W
W	B	B	W	B	W	B	W	B	W	B			W
W	B	B	W	B	W	B	W			B	B	W	W
W	B	B			W	B	W	W	B	B	B	W	W
W	B	B	W	W	W	B			B	B	B	W	W
W			W	W	W	B	B	B	B	B	B	W	W
W	W	W	W	W	W	B	B	B	B	B	B		

Here is a solution for $n = 7$:

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
		B	W	B	W	B	W	B	W	B	W	B	W	B	W
W	B	B	W	B	W	B	W	B	W	B	W	B			W
W	B	B	W	B	W			B	W	B	W	B	B	W	W
W	B	B	W	B	W	W	B	B			W	B	B	W	W
W	B	B	W			W	B	B	B	W	W	B	B	W	W
W	B	B	W	W	W	W	B	B	B			B	B	W	W
W			W	W	W	W	B	B	B	B	B	B	B	W	W
W	W	W	W	W	W	W	B	B	B	B	B	B	B		

Starting with $n \geq 8$, we can get a solution recursively by reducing the instance given to the instance of size $n - 4$. The first two moves transform the line to

WBBW followed by two empty spaces followed by $2n - 8$ alternating counters followed by BBWW:

		1	2	3	4	5	6	7	8		$2n-5$	$2n-4$	$2n-3$	$2n-2$	$2n-1$	$2n$
		B	W	B	W	B	W	B	W	...	B	W	B	W	B	W
W	B	B	W	B	W	B	W	B	W	...	B	W	B			W
W	B	B	W			B	W	B	W	...	B	W	B	B	W	W

The sequence of $2n - 8$ alternating counters with two preceding empty spaces forms the puzzle's instance of size $n - 4 \geq 4$. After its solution (with two trailing spaces) is obtained recursively, only two more moves are needed to complete the solution to the instance of size n:

		1	2	3	4	5	6			$2n-5$	$2n-4$	$2n-3$	$2n-2$	$2n-1$	$2n$
W	B	B	W	W	...	W	B	...	B			B	B	W	W
W			W	W	...	W	B	...	B	B	B	B	B	W	W
W	W	W	W	W	...	W	B	...	B	B	B	B	B		

The above algorithm solves the instance with $2n$ counters in n moves. This assertion can be easily proved by strong induction or by solving the recurrence for the number of moves: $M(n) = M(n-4) + 4$ for $n > 7$, $M(n) = n$ for $3 \leq n \leq 7$.

Comments The algorithm presented above is based on the decrease-(by 4)-and-conquer strategy.

An extensive list of publications discussing this puzzle and its variations can be found in D. Singmaster's bibliography [Sin10, Section 5.O]. The earliest paper was published in 1884 by P. G. Tait; several authors attributed the solution to the general case of the puzzle to Henri Delannoy, a French military officer and mathematician.

132. The Solitaire Army

Solution a. To advance a peg three rows above the line, it is natural to target a known configuration capable of advancing a peg two rows above the line after this configuration is moved one row up (Figure 4.88a). Such a configuration of 8 pegs is shown in Figure 4.88b. The alternative configuration of 8 pegs advancing 1 peg three rows above the line is shown in Figure 4.88c.

b. Here, it is logical to take advantage of the solution given in part (a) by moving it one row above the line and then looking for an initial configuration of 20 pegs below the line to reach it. An execution of this plan is shown in Figure 4.89. Figure 4.89a shows a configuration obtained in part (a) (Figure 4.88b), which is capable of reaching the cell marked with an X three rows above it The cells of this 8-peg configuration are partitioned into five regions, each composed of the cells marked with the same number from 1 to 5. Figure 4.89b presents a 20-peg configuration that solves the problem via a transformation to the configuration in

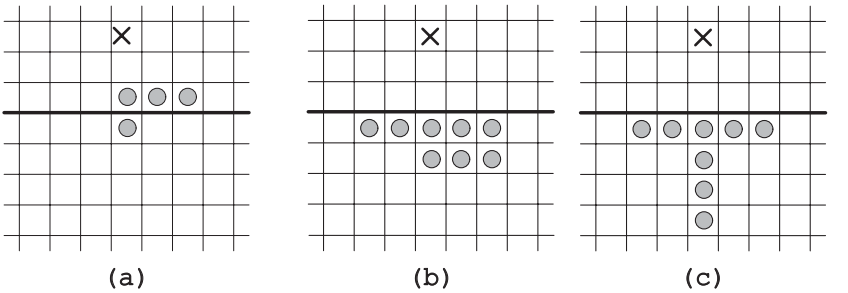


FIGURE 4.88 (a) Intermediate target configuration (X denotes the target cell). Parts (b) and (c) show initial configurations of 8 pegs to reach the target configuration.

Figure 4.89a. The configuration in Figure 4.89b is composed of five regions, with the cells of the same region marked with the same number as well. The peg pairs numbered with 1's and 2's in Figure 4.89b transform into the respective 1-peg regions in Figure 4.89a, the 8 pegs numbered with 3's in Figure 4.89b transform into the 4-peg region numbered with 3's in Figure 4.89a, and so on.

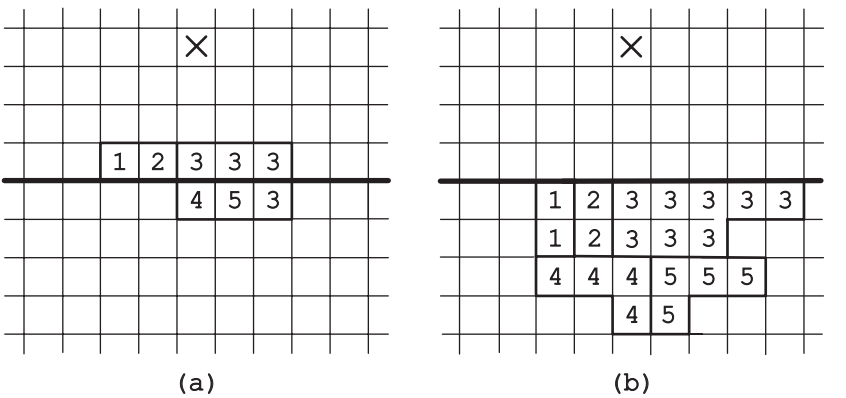


FIGURE 4.89 (a) Intermediate target configuration of 8 pegs. (b) Initial configuration of 20 pegs to reach the target configuration.

The 20-peg solution shown in Figure 4.89b is not unique; Beasley [Bea92, p. 212] mentions two others shown in Figure 4.90.

Comments The solution given above is principally based on the transform-and-conquer strategy, which is helped by the divide-and-conquer thinking.

We did not ask the reader to prove that the peg numbers given in the puzzle's statement are the fewest possible for reaching the specified rows. To prove this, one needs to take advantage of special functions called *resource counts* or *pagoda functions*, invented by J. H. Conway and J. M. Boardman in 1961 [Bea92, p. 71]. A resource count is a function that assigns a numeric value to each cell of the board

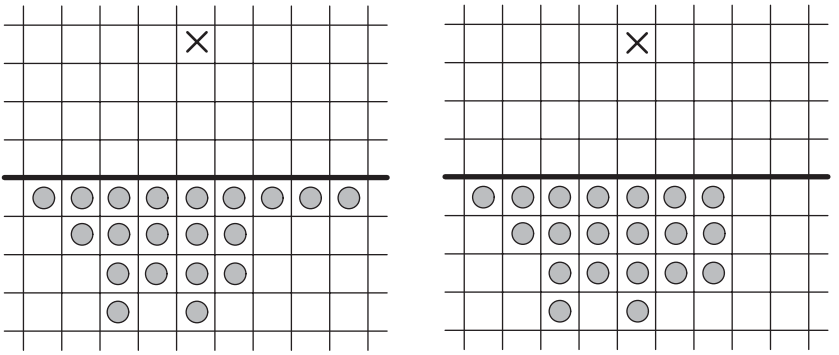


FIGURE 4.90 Two alternative configurations of 20 pegs capable of reaching a cell in the fourth row above the line.

so that for any legitimate jump of a peg, the sum of the values assigned to all the occupied cells after the jump is not greater than the sum of the values for all the occupied cells before the jump. Among infinitely many resource counts possible, the one given by J. D. Beasley in [Bea92, p. 212] and shown here in Figure 4.91 proves that fewer than 8 pegs cannot reach a designated cell in the third row above the line. (It is assumed that all the values not shown in that figure are 1's for the cells below the line and are computed by adding two values immediately below the cell for the cells above the line. The value of 21 is assigned to the cell in the third row above the line that is designated by the puzzle as the target cell.) Since for any set of seven or fewer cells below the line, the resource count is not greater than $5 \cdot 1 + 3 \cdot 3 + 2 \cdot 3 = 20$, no seven or fewer pegs in any cells below the line can be transformed to a single peg at the target cell with the resource count of 21.

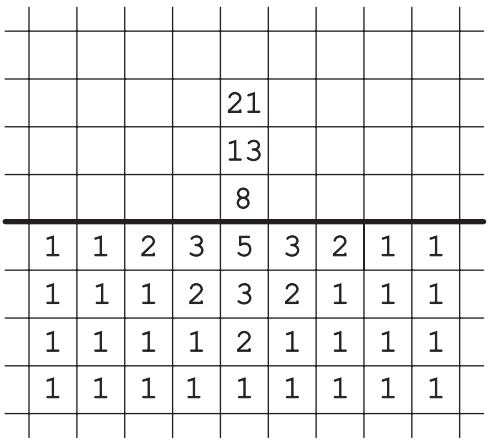


FIGURE 4.91 Resource count proving that at least 8 pegs are necessary to reach a cell in the third row.

The most spectacular application of resource counts is arguably that by J. H. Conway himself, who proved in 1961 that no amount of pegs can reach a cell at the

fifth row above the line. To prove this counterintuitive fact, Conway used powers of $(\sqrt{5}-1)/2$, the reciprocal of the so-called *golden ratio* $(\sqrt{5}+1)/2$. The details of this remarkable proof can be found in the classic book on mathematical games [Ber04] coauthored by Conway as well as on several websites. (Note that the puzzle is also known under the names *Conway's Soldiers* and *Sending Scouts into the Desert.*) J. Tanton proved the same fact by using a resource count based on the Fibonacci numbers [Tan01, pp. 197–198].

133. The Game of Life

Solution The smallest still lifes (the "block" and the "tub"), oscillator (the "blinker"), and spaceship (the "glider") are shown in Figure 4.92.

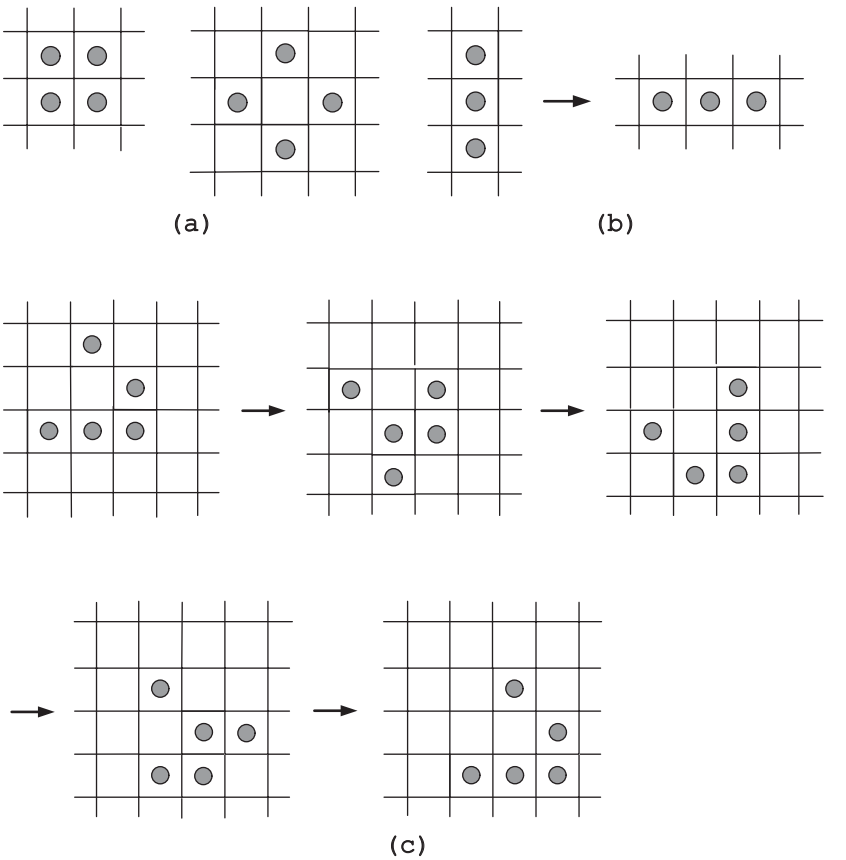


FIGURE 4.92 The *Game of Life* configurations: (a) The "block" and the "tub." (b) The "blinker." (c) The "glider" descending diagonally one cell down and to the right in four generations.

Comments The puzzle asks to determine inputs to produce specified outputs by the puzzle's algorithm.

The *Game of Life* was invented by the British mathematician John Conway in 1970. It became widely popular after Martin Gardner wrote about the game in his *Scientific American* columns [Gar83, Chapters 20–22]. It still commands a significant interest, as can be seen from a number of the websites devoted to it. The game is interesting in several ways. First, Conway's simple rules lead to fascinating and quite unexpected patterns. Second, the game can be used as a model of a universal computer [Ber04, Chapter 25], which leads to profound questions from mechanics of evolution to the nature of the Universe.

134. Point Coloring

Solution The problem can be solved by the following recursive algorithm. If $n = 1$, color the point either color, say, black. If $n > 1$, proceed as follows. Select a line l (horizontal or vertical) that contains an odd number of the given points; if there is no such line, select any line with at least one point on it. Select a point P on line l. Recursively, color all the points except P as required by the problem's statement. We will show that P can always be colored to satisfy the problem's requirement. Let m be the other grid line passing through point P. If the number of the colored points on each of the lines l and m is even, exactly one half of its points colored black and the other half colored white; therefore, P can be colored either color. If the number of the colored points is even on one of these lines and odd on the other, P should be colored to even out the color occurrences on that other line. If the number of the colored points on each of the lines l and m is odd and the same color occurs one more time than the other on both lines, P should be colored to even out the color occurrences on each of these lines.

Finally, we need to show that the situation in which one color—say, black—occurs more often on l while the other color—white—occurs more often on m is impossible if the number of colored points on each of the lines l and m is odd Indeed, in such a situation, the number of all the points on l would have been even (odd colored points plus P). Because of the way line l was selected, this would imply that every line contains an even number of points, with each line except l and m having exactly one half of its points colored black and the other half colored white. But if we then compute the total number of points colored each color by adding the colored points on all the lines parallel to l and l itself, we would have to conclude that the total number of black points is one more than the total number of white points; if, on the other hand, we compute the total number of points colored each color by adding the colored points on all the lines parallel to m and m itself, we would have to conclude that the total number of white points is one more than the total number of black points. This contradiction completes the proof of the algorithm's correctness.

Comments The problem, which was offered to participants of the XXVII International Mathematical Olympiad, appeared in the December 1986 issue of *Kvant*, the Soviet Union magazine in physics and mathematics for school students and teachers (Problem M1019, p. 26). The solution by A. P. Savin,

reproduced above, was published in the April 1987 issue (pp. 26–27). In our terminology, it is based on the decrease-by-one variety of the decrease-and-conquer strategy.

135. Different Pairings

Solution Here is one of the ways to generate $2n - 1$ sets of different pairs efficiently. For convenience, number the children from 1 to $2n$ and place these numbers in a $2 \times n$ table. The pairs for the first set are given by the columns of this table. To generate the next $2n - 2$ sets, rotate—say, clockwise—all the entries except 1 in the last generated table. Figure 4.93 shows an example for $n = 3$.

1	2	3
6	5	4

1	6	2
5	4	3

1	5	6
4	3	2

1	4	5
3	2	6

1	3	4
2	6	5

FIGURE 4.93 Five sets of three different pairs.

The other way to describe the algorithm is to mark with 1 the center of a circle and divide its circumference by $2n - 1$ equidistant points numbered clockwise from 2 to $2n$. A set of pairs is obtained by drawing the diameter through the center and one of the points on the circumference, which yields one pair, and the other $n - 1$ pairs are obtained by the chords perpendicular to this diameter. Figure 4.94 illustrates this for $n = 3$. New sets of pairs are obtained by rotating the diameter, which yields a new mate for the center point and a new pairing of the other points by the set of the new chords.

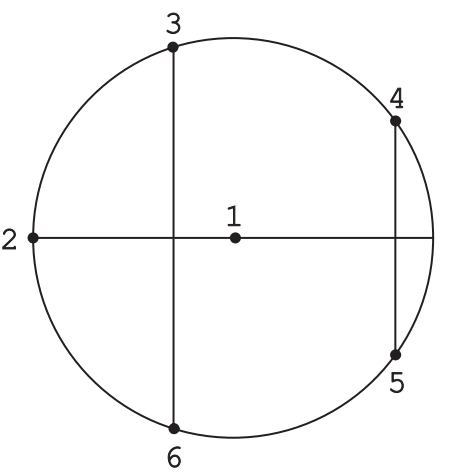


FIGURE 4.94 Geometric way to generate pairings for $n = 3$.

Comments The algorithm can be thought of as based on the representation change strategy.

Both interpretations of the algorithm outlined above were given by Maurice Kraitchik in *Mathematical Recreations* [Kra53, pp. 226–227]. Of course, the problem is another wording of game scheduling in a round-robin tournament with n participants.

136. Catching a Spy

Solution Since the spy is at location a at $t = 0$ and moves b units each time interval, the spy's location at time t can be computed by the obvious formula $x_{ab}(t) = a + bt$. Therefore, the problem can be solved by any algorithm that enumerates the set of all integer pairs (a, b) in a sequence and checks the spy's location at consecutive time intervals $t = 0, 1, \ldots$ computed by this formula for the corresponding (a, b) in the sequence. Then whatever the parameters a and b defining the spy's movement, the algorithm will reach this combination after a finite number of steps to compute the spy's exact location. Obviously, if the spy is at location 0 at $t = 0$, he can be discovered by probing this location at $t = 0$, no matter what the value of b is; hence, after checking the pair $(0, 0)$ at $t = 0$, no other pair $(0, b)$ needs to be checked.

Integer pairs (a, b) can be enumerated in many different ways. In particular, one can think of the pairs as integer points of the Cartesian plane and enumerate them by moving along a spiral originating at $(0, 0)$ (Figure 4.95). This will lead to the

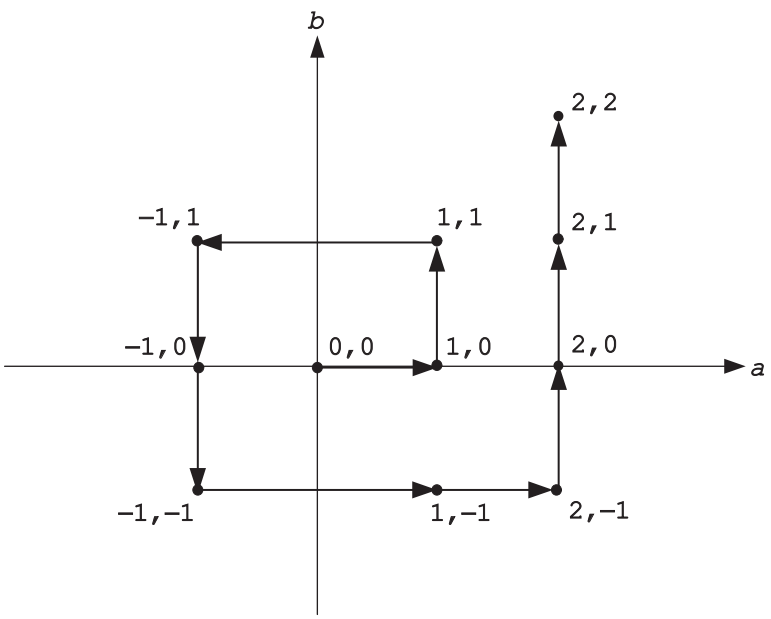


FIGURE 4.95 Spiral solution of the *Catching a Spy* puzzle.

probing the spy's locations at

$$0+0\cdot 0=0,\ 1+0\cdot 1=1,\ 1+1\cdot 2=3,\ -1+1\cdot 3=2,$$
$$-1+0\cdot 4=-1,\ -1-1\cdot 5=-6,\ 1-1\cdot 6=-5,\ \ldots$$

Alternatively, one can follow a standard enumeration method of mathematical set theory. This method considers the (a, b) pairs as elements of an infinite matrix X whose rows and columns correspond to different values of a ($a = 0, \pm 1, \pm 2, \ldots$) and b ($b = 0, \pm 1, \pm 2, \ldots$), respectively (Figure 4.96). The method lists the elements of the matrix in order of its northeast to southwest diagonals shown in that figure.

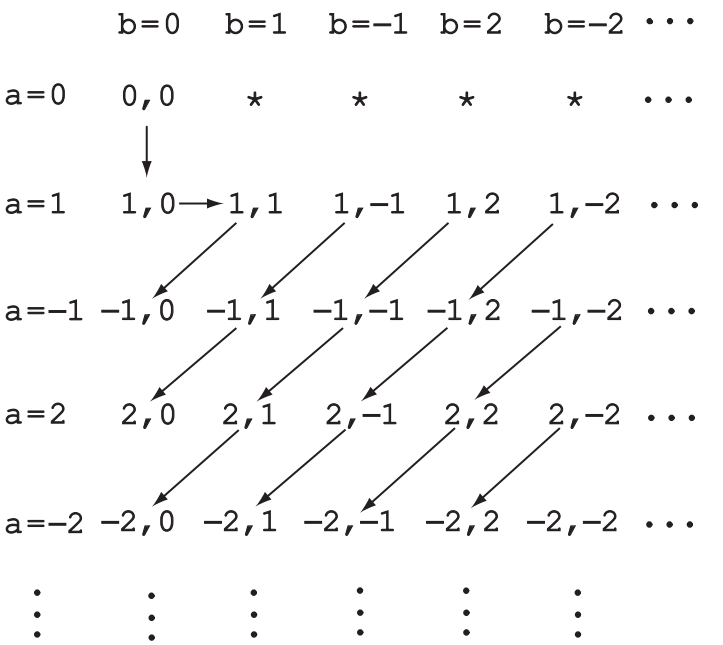


FIGURE 4.96 Matrix solution of the *Catching a Spy* puzzle. The elements in row 0 except x_{00} are marked by an asterisk because they are not used by the algorithm.

The sequence of the locations probed by the algorithm will start as follows:

$$0+0\cdot 0=0,\ 1+0\cdot 1=1,\ 1+1\cdot 2=3,\ -1+0\cdot 3=-1,$$
$$1+(-1)\cdot 4=-3,\ -1+1\cdot 5=4,\ 2+0\cdot 6=2.$$

Comments The first solution given above was suggested to the authors by Stephen Lucas of James Madison University, one of the book's reviewers. It seems to be simpler than the authors' solution based on the diagonal enumeration of the infinite matrix.

The above solutions are, in our terminology, representation change examples.

The authors came across this puzzle perusing the Internet puzzle collection by K. R. M. Leino of Microsoft Research [Leino]. They have not be able to determine its origin, which is not totally surprising given its subject matter.

137. Jumping into Pairs II

Solution The puzzle has a solution if and only if n is a multiple of 4.

Obviously, the problem cannot be solved for an odd number of coins, because in the final state all the coins are arranged in pairs, making their total number even. Moreover, n must be a multiple of 4. To prove this, consider the state of the puzzle before the last move: $(n - 2)$ coins are already paired, and hence one of the two remaining single coins should make a jump over an even number of coins to land on the other single coin. Since a coin is required to jump over an even number of coins on move i, $1 \leq i \leq n/2$, if and only if i is even, the last move, numbered $n/2$, must be even, which implies that n must be a multiple of 4.

One can devise an algorithm for solving the problem by backward thinking as follows. Consider the puzzle in its final state in which n coins $(n = 4k,\ k > 0)$ are paired, with the pairs numbered left to right from 1 to $n/2$. To arrive at the initial state of the puzzle with n single coins in a row, take the top coin of the pair numbered $n/4 + 1$ and move it to the left over all $n/2$ coins. Then move the top coin of the pair numbered $n/4$ and move it to the left over all $n/2 - 1$ coins. Continue this process of moving the top coins of the pairs over all the coins to the left of them until the top coin of the leftmost pair is moved to the left over $n/4$ coins. Then, starting with the leftmost remaining pair (initially numbered $n/4 + 2$) and ending with the rightmost pair (initially numbered $n/2$), move the top coin of the pair to the left over $n/4 - 1, \ldots, 1$ coins respectively. The top coins should be placed as single coins as just described—recall that we are supposed to disregard spacing between adjacent coins.

"Reversing" the above, we obtain the following algorithm for pairing n coins placed in a row and numbered left to right from 1 to n. First, perform the following operation for $i = 1, 2, \ldots, n/4 - 1$: for the rightmost single coin, find the coin to its left to have i coins between them and place that coin on the rightmost single coin. Then repeat the following operation for $i = n/4, n/4 + 1, \ldots, n/2$: take the leftmost single coin and jump it over i coins to the right to land on a single coin. The operations of the algorithm for $n = 8$ are illustrated in Figure 4.97.

This algorithm obviously makes the minimum number of moves possible because it forms a new pair of coins on each move.

Comments The solution to the puzzle provides an excellent example of the occasional usefulness of backward thinking in algorithm design.

The puzzle and its solution was attributed by Martin Gardner to W. Lloyd Milligan (see [Gar83, pp. 172, 180]).

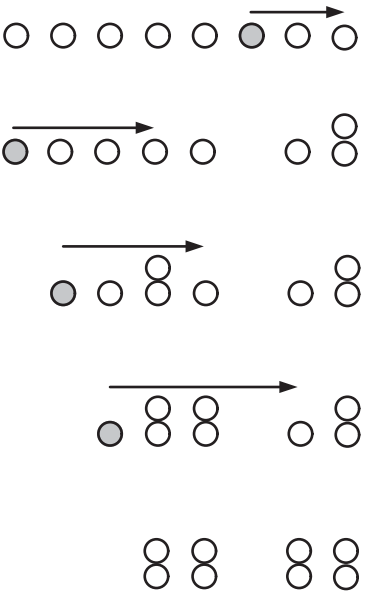


FIGURE 4.97 Solution to the *Jumping into Pairs II* puzzle for $n = 8$. The jumping coins are shown in gray.

138. Candy Sharing

Solution Let i and j be, respectively, the number of candy pieces a child and his or her neighbor to the right have before a whistle blow. After the whistle blow, the child who had i pieces will have i' pieces, where i' is defined by the following formula:

$$i' = \begin{cases} i/2 + j/2 & \text{if this number is even,} \\ i/2 + j/2 + 1 & \text{otherwise.} \end{cases}$$

This formula implies that if $i = M$, where M is largest number of candy pieces in the initial distribution (recall that it is even), then $i' \leq M$. Thus, the number of candy pieces any child can have never exceeds M. Let now m be the smallest number of candy pieces a child has before a particular whistle blow. After that whistle blow, every child will have at least $m/2 + m/2 = m$ candy pieces; moreover, a child will have at least $m + 1$ pieces unless the child and his or her neighbor to the right both had m pieces before the whistle blow. More generally, if there were $1 \leq k < n$ children in a row with m candy pieces while the $(k + 1)$th child (counting counterclockwise) having more than m, the first $k - 1$ of them will still have m pieces after the whistle blow but the kth child will have more than m. Hence, after k iterations, the minimum number of pieces of candies among these children will increase. Since the number of candy pieces a child can have is also limited from above, the process will result in each child having the same number of candy pieces after a finite number of iterations.

Comments The problem exploits the idea of a monovariant, which is the minimum number of pieces in a candy distribution before a whistle blow. (The notion of a monovariant is discussed in the book's tutorial on algorithm design techniques.)

The problem is rather well-known; in particular, it was given in the 1962 Beiging Olympiad and 1983 Leningrad All-City Mathematical Olympiad. For some variations of the problem, see a paper by G. Iba and J. Tanton [Iba03].

139. King Arthur's Round Table

Solution Since each knight has at least $n/2$ friends, the number of his enemies cannot exceed $n - 1 - n/2 = n/2 - 1$. If $n = 3$, each of the three knights is friends with the two others and hence they can be seated in any order. In $n > 3$, we can start with an arbitrary seating of the knights and count the number of enemy pairs seating next to each other. If this count is zero, we are done; if it is not, it can be decreased at least by one as follows. Let knights A and B be two enemies seating next to each other, B to the left of A (Figure 4.98). Then there exists among the friends of A knight C whose neighbor to the left D is friendly with B. (If it were not true, the number of the knights who are enemies with B would have been at least $n/2$.) If we now swap the seats of all the knights between B and C clockwise including knights B and C themselves (shown by the arrows in Figure 4.98a), we will get a seating arrangement with at least one fewer pair of enemy neighbors (Figure 4.98b).

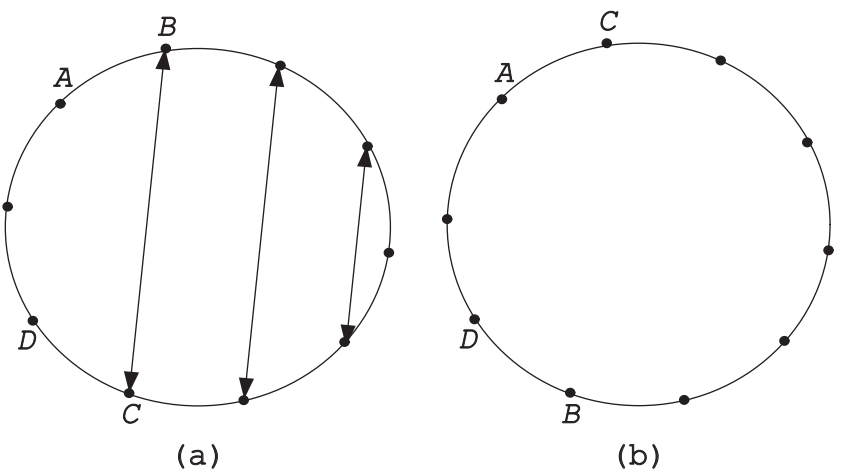


FIGURE 4.98 Iterative improvement of a seating in the *King Arthur's Round Table* puzzle.

Comments The puzzle and its solution are from the paper on monovariants [Kur89] published in *Kvant* in 1989. (We have omitted the assumption that the number of knights is even, which is unnecessary.) It should be noted that the existence of a solution to the puzzle follows from *Dirac's theorem*: Any graph with

$n \geq 3$ vertices in which the degree of each vertex (i.e., the number of vertices connected to it by an edge) is at least $n/2$ has a Hamilton circuit. For this puzzle, the vertices of the graph are the knights and there is an edge between two vertices if and only if the knights represented by the vertices are friendly.

140. The *n*-Queens Problem Revisited

Solution The solution for $n = 4$ shown in Figure 4.99a suggests the following structure of solutions for other even n's. For the first $n/2$ columns, place the queens in rows $2, 4, \ldots, n/2$; for the last $n/2$ columns, place the queens in rows $1, 3, \ldots, n-1$. Indeed, this works not only for any $n = 4 + 6k$ but also for any $n = 6k$ (e.g., Figure 4.99b). Moreover, since none of these solutions places a queen on the main diagonal, the solution can be extended to yield a solution for the next value of n by just adding a queen in the last row of the last column (see Figure 4.99c for an example).

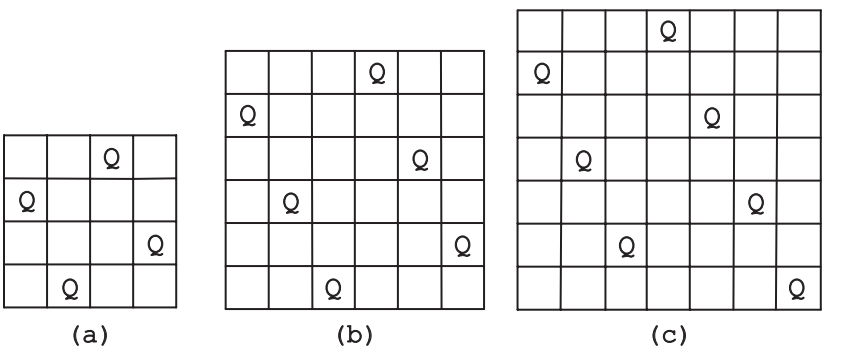


FIGURE 4.99 Solutions to the *n-Queens Problem* for (a) $n = 4$, (b) $n = 6$, and (c) $n = 7$.

Unfortunately, this does not work for $n = 8 + 6k$. Although it is possible to place queens on such boards as it was done above and then rearrange them, it is easier to start with queens in odd-numbered rows of the first $n/2$ columns followed by queens in even-number rows. Then it is possible to get a nonattacking position by exchanging the rows of the queens in columns 1 and $n/2 - 1$ and the rows of the queens in columns $n/2 + 2$ and n (see an example for $n = 8$ in Figure 4.100a). Since the solution obtained in this way has no queen on the main diagonal, it can be expanded to a solution for $n = 9 + 6k$ by placing the extra queen at the last row of the last column of the larger board (Figure 4.100b).

Thus, we have the following algorithm that generates row numbers for placing $n > 3$ queens in consecutive columns of an $n \times n$ board.

Compute the remainder r of division n by 6.

Case 1 (r is neither 2 nor 3): Write a list of the consecutive even numbers from 2 to n, inclusive, and append to it the consecutive odd numbers from 1 to n, inclusive.

Case 2 (r is 2): Write a list of the consecutive odd numbers from 1 to n, inclusive, and then swap the first and penultimate numbers. Append to the list the

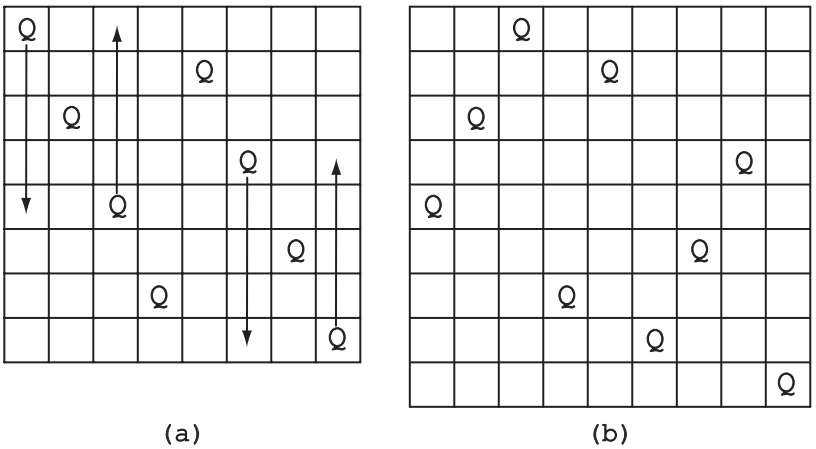


FIGURE 4.100 Solutions to the *n-Queens Problem* for (a) $n = 8$ and (b) $n = 9$.

consecutive even numbers from 2 to n, inclusive, and then swap number 4 and the last number in the expanded list.

Case 3 (r is 3) Apply the directives of Case 2 to $n - 1$ instead of n and append n to the created list.

Comments The above algorithm, which follows an outline by D. Ginat [Gin06], is rather straightforward in that it places queens in the first available square of each column of the board. The nontrivial part is that for some n's we have to start with the second square in the first column, and for some n's we have to swap two pairs of queens to satisfy the nonattacking requirement. It is also worth noting that the solutions for $n \times n$ boards where n is odd are obtained, in fact, as extensions of the solutions for $(n - 1) \times (n - 1)$ boards.

The *n-Queens Problem* is one of the most well-known problems in recreational mathematics and has attracted the attention of mathematicians since the middle of the 19th century. The search for efficient algorithms for solving it started much later, of course. Among a variety of approaches, solving the problem by backtracking—as we do in this book's first tutorial—has become a standard way to introduce this general technique in algorithm textbooks. In addition to the educational merits, backtracking has the advantage of finding, at least in principle, all the solutions to the problem. If just one solution is the goal, much simpler methods can be employed. The survey paper by J. Bell and B. Stevens [Bel09] contains more than half a dozen sets of formulas for direct computation of queens' positions, including the earliest one by E. Pauls published in 1874. Surprisingly, this alternative had been all but ignored by computer scientists until B. Bernhardsson alerted the community of its existence [Ber91].

141. The Josephus Problem

Solution Let $J(n)$ be the survivor's number. It is convenient to consider the cases of even and odd n's separately. If n is even, that is, $n = 2k$, the first pass through

the circle yields an instance of the same problem but half its initial size. The only difference is in position numbering; for example, a person in initial position 3 will be in position 2 for the second pass, a person in initial position 5 will be in position 3, and so on. Thus, to get the initial position of a person, we simply need to multiply his new position by two and subtract one. In particular, for the survivor, we obtain

$$J(2k) = 2J(k) - 1.$$

Let us now consider the case of an odd $n > 1$, that is, $n = 2k + 1$. The first pass eliminates people in all even positions. If we add to this the elimination of the person in position 1 right after that, we are left with an instance of size k. Here, to get the initial position that corresponds to the new position numbering, we have to multiply the new position number by 2 and add 1. Thus, for odd values of n, we get

$$J(2k + 1) = 2J(k) + 1.$$

To find an explicit formula for $J(n)$, we will follow the plan outlined in [Gra94, Section 1.3]. Using the initial condition $J(1) = 1$ and the above recurrences for even and odd values of n, we obtain the following values of $J(n)$ for $n = 1, 2, \ldots, 15$:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$J(n)$	1	1	3	1	3	5	7	1	3	5	7	9	11	13	15

On inspecting these values, it is not difficult to observe that for the n's values between consecutive powers of 2, that is, for $2^p \le n < 2^{p+1}$ or $n = 2^p + i$ where $i = 0, 1, \ldots, 2^p - 1$, the corresponding values of $J(n)$ run the range of odd numbers from 1 to $2^{p+1} - 1$. This observation can be expressed by the formula

$$J(2^p + i) = 2i + 1 \text{ for } i = 0, 1, \ldots, 2^p - 1.$$

It is not difficult to prove by induction on p that this formula does solve the recurrences of the Josephus problem for any nonnegative integer p. For the basis value $p = 0$, we have $J(2^0 + 0) = 2 \cdot 0 + 1 = 1$, as it should be for the initial condition. Assuming that for a given nonnegative integer p and for every $i = 0, 1, \ldots, 2^p - 1$, $J(2^p + i) = 2i + 1$, we need to show that

$$J(2^{p+1} + i) = 2i + 1 \text{ for } i = 0, 1, \ldots, 2^{p+1} - 1.$$

If i is even, it can be represented as $2j$ where $j = 0 \le j < 2^p$. Then using the induction's assumption, we obtain

$$\begin{aligned} J(2^{p+1} + i) &= J(2^{p+1} + 2j) = J(2(2^p + j)) = 2J(2^p + j) - 1 \\ &= 2(2j + 1) - 1 = 2i + 1. \end{aligned}$$

If i is odd, it can be expressed as $2j + 1$ where $0 \le j < 2^p$. Then using the induction's assumption, we obtain

$$J(2^{p+1} + i) = J(2^{p+1} + 2j + 1) = J(2(2^p + j) + 1) = 2J(2^p + j) + 1$$
$$= 2(2j + 1) + 1 = 2i + 1.$$

Finally, one can also get $J(n)$ by a 1-bit cyclic shift left of the binary representation of n [Gra94, p. 12]. For example, if $n = 40 = 101000_2$, $J(101000_2) = 10001_2 = 17$. Alternatively, one can also find $J(n)$ by the formula

$$J(n) = 1 + 2n - 2^{1+\lfloor \log_2 n \rfloor},$$

mentioned in [Weiss]. Using the same value of 40 for an example, we get $J(40) = 1 + 2 \cdot 40 - 2^{1+\lfloor \log_2 40 \rfloor} = 17$.

Comments This puzzle is named for Flavius Josephus, a Jewish historian who participated in and chronicled the Jewish revolt of 66–70 C.E. against the Romans. Josephus, as a general, managed to hold the fortress of Jotapata for 47 days, but after the fall of the city he took refuge with 40 diehards in a nearby cave. There, the rebels voted to perish rather than surrender. Josephus proposed to form a circle and, proceeding around it, kill every third man until just one man was left, who was supposed to kill himself. Josephus contrived to be in a position to be that last person and when just he and another man remained alive, he prevailed upon his intended victim to surrender to the Romans.

The puzzle discusses the version in which every *second* person is eliminated, which makes it easier to solve. For other variations and historical references, see David Singmaster's annotated bibliography [Sin10, Section 7.B] and, among many other less exhaustive sources, the book by Ball and Coxeter [Bal87, pp. 32–36].

142. Twelve Coins

Solution The decision tree in Figure 4.101 presents an algorithm that solves the fake-coin puzzle for 12 coins in three weighings. In this tree the coins are numbered from 1 to 12. Internal nodes indicate weighings, with the coins being weighted listed inside the node. For example, the root corresponds to the first weighing in which coins 1, 2, 3, 4 and coins 5, 6, 7, 8 are put on the left and right pan of the scale, respectively. Edges to a node's children are marked according to the node's weighing outcome: $<$ means that the left pan's weight is smaller than that of the right's pan, $=$ means that the weights are equal, and $>$ means that the left pan's weight is larger than that of the right pan. Leaves indicate final outcomes: $=$ means that all the coins are genuine, and a number followed by $+$ or $-$ means that the coin with that number is heavier or lighter, respectively. A list above an internal node indicates the outcomes that are still possible before the weighing indicated inside the node. For example, before the first weighing either all the coins are genuine $(=)$ or each coin is either heavier $(+)$ or lighter $(-)$.

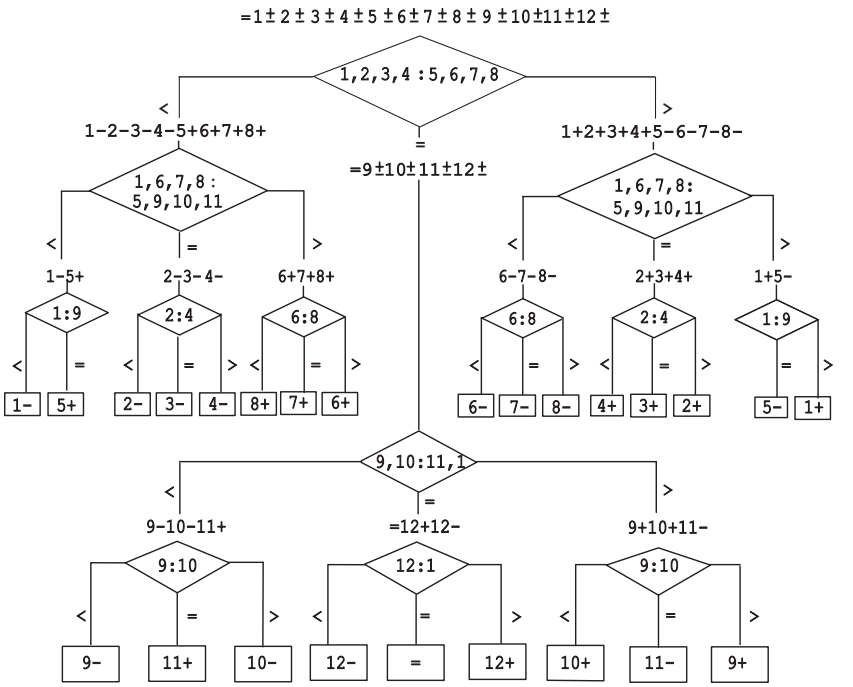


FIGURE 4.101 Decision tree for the *Twelve Coins* puzzle.

The problem cannot be solved in fewer than three weighings. A decision tree associated with any algorithm solving the problem will have to have at least $2 \cdot 12 + 1 = 25$ leaves reflecting all the possible outcomes. Therefore its height, equal to the number of weighings W performed by the algorithm in the worst case, will have to satisfy the inequality $W \geq \lceil \log_3 25 \rceil$ or $W \geq 3$.

Comments The attractiveness of the solution given above lies in its symmetry: the second weighings involve the same coins if the first weighing tips the scale either way, and the subsequent round of weighings involves the same pairs of coins. In fact, the problem has a completely nonadaptive solution, that is, a choice of what to put on the balance for the second weighing does not depend on the outcome of the first one, and a choice of what to weigh in the third round does not depend on what happened on either the first or second weighing (e.g., [OBe65, pp. 22–25]). For other solutions, including the one based on the ternary system, see the references on the "Weighing 12 coins, an Odd Ball (A Selection of Treatments) puzzle" page from [Bogom].

An optimal algorithm for the general instance of this problem with $n \geq 3$ coins requires $\lceil \log_3 (2n + 3) \rceil$ weighings. This fact was established independently by several mathematicians who published their findings in *Mathematical Gazette* in 1946.

The first published accounts of this famous puzzle appeared in 1945 [Sin10, Section 5.C]. It seems to have arisen either on the eve or during the Second World War. T. H. O'Beirne writes in his book that it has been alleged that

this puzzle "diverted so much war-time scientific effort that there was serious consideration of a proposal to inflict compensating damage by dropping it in enemy territory" [OBe65, p. 20]. Since then the puzzle has become one of the most popular puzzles ever, both in puzzle books and on the Internet puzzle sites.

143. Infected Chessboard

Solution The answer is n.

There are many initial configurations of n infected squares that will infect the entire $n \times n$ board; Figure 4.102 shows two of them.

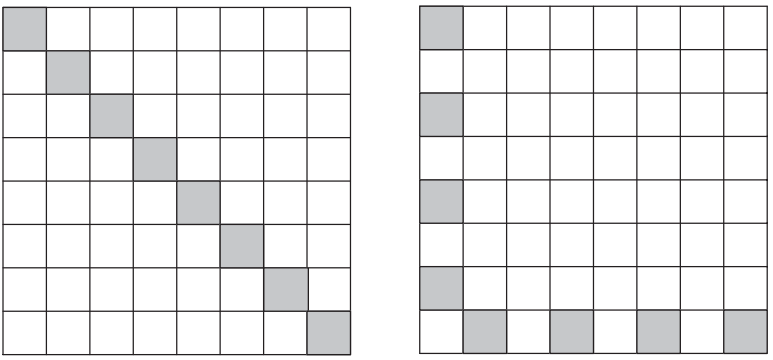


FIGURE 4.102 Two initial configurations of infected squares that will spread to the entire board.

In order to prove that n squares is the minimum number needed, note that the total perimeter of the infected region (which, in general, is equal to the sum of the perimeters of contiguous subregions infected by the virus) cannot increase while the virus spreads over the board. Indeed, when a new square is infected, at least two of its boundary edges are absorbed into the infected region, and at most two boundary edges are added to it. Therefore, if there are less than n squares that are initially infected, the total perimeter of the infected region will be less than $4n$ initially, and it will remain less than $4n$ as the virus spreads. Hence, the entire board with the perimeter of $4n$ will never be entirely infected.

Comments The puzzle exploits the idea of monovariant mentioned in the discussion of iterative improvement in the first tutorial, but it is used here to prove impossibility of a certain kind of a solution—namely, of an initial configuration of fewer than n infected squares capable of infecting the entire $n \times n$ board.

The puzzle was included in Peter Winkler's *Mathematical Puzzles* [Win04, p. 79]. Béla Bollobás [Bol07, p. 171] considered the version of the puzzle in which at least three infected neighbors are needed to infect a new square.

144. Killing Squares

Solution The following recursive algorithm solves the puzzle by removing the minimum number of toothpicks, which is equal to $\lceil n^2/2 \rceil + 1$ for $n > 1$

(and, of course, is equal to 1 when $n = 1$). If $n = 1$, 2, or 3, the solutions are given in Figure 4.103.

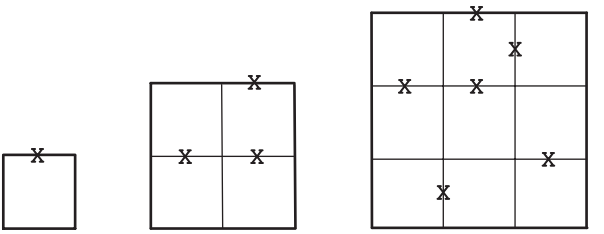


FIGURE 4.103 The *Killing Squares* solutions for $n = 1$, $n = 2$, and $n = 3$.

If $n > 3$, do the following. Consider the frame of width 1 formed by the perimeters of the square given and the square of size $n - 2$ inside it. Starting with the top left corner of the frame and going counterclockwise, remove every toothpick that would have been the middle line of the domino ring tiling the frame except for the toothpick of the last domino in the ring. For that last domino, remove the second horizontal toothpick in the upper side of the square given. Then solve recursively the puzzle for the square of size $n - 2$ inside the smaller border of the frame.

The algorithm's application to the $n \times n$ boards for even and odd values of n is illustrated in Figure 4.104.

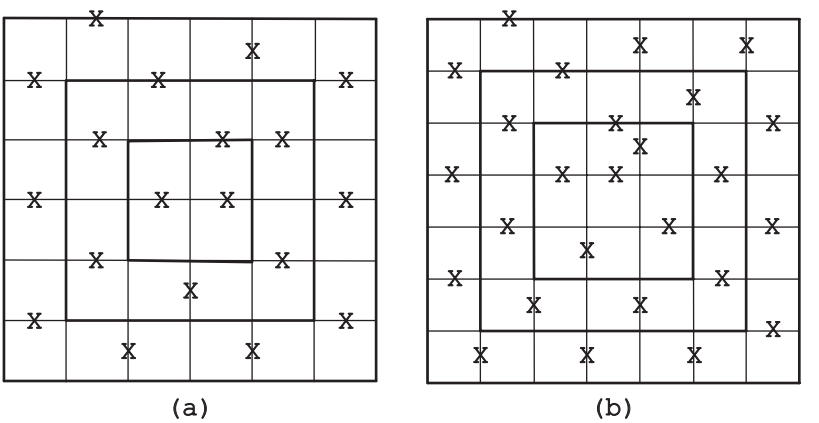


FIGURE 4.104 The *Killing Squares* solutions for (a) $n = 6$ and (b) $n = 7$.

In a nutshell, the algorithm works because the middle lines of the dominoes tiling the adjacent frames break the straight lines inside the board, and its perimeter is taken care of by removing a single toothpick. This observation can be easily formalized by using mathematical induction.

The domino tiling also immediately yields the formula for $K(n)$, the number of toothpicks removed by the algorithm. Indeed, if n is even, there are $n^2/2$ dominoes in the tiling; each of this dominoes contributes one removed toothpick, except for one of the dominoes covering the central 2×2 square, which contributes two. Thus, the total number of removed toothpick for an even n is equal to $n^2/2 + 1$. If $n > 1$ is odd, there are $(n^2 - 1)/2$ dominoes in the tiling of the frames; each of these dominoes contributes one removed toothpick except for one of the horizontal dominoes covering the central 3×3 square, which contributes two. In addition, one more toothpick is removed from the central 1×1 square. Thus, the total number of removed toothpicks for an odd n is equal to $(n^2 - 1)/2 + 2 = \lceil n^2/2 \rceil + 1$.

It is easy to show that $K(n) = \lceil n^2/2 \rceil + 1$ is the minimum number of toothpicks that need to be removed to eliminate all the squares when n is even. If we color the cells of the board in two alternating colors as those in a chessboard, there will be $n^2/2$ dark cells and $n^2/2$ light cells. To eliminate unit squares formed by the perimeters of the dark cells, at least $n^2/2$ toothpicks need to be removed. These $n^2/2$ toothpicks can also eliminate all the unit squares formed by the perimeters of the light squares as well, but only if each removed toothpick separates a dark and a light square. Since there is no such toothpick on the board's perimeter, at least one more toothpick needs to be removed for the total minimum of $n^2/2 + 1$. If $n > 1$ is odd, a somewhat more sophisticated argument (see [Gar06, pp. 31–32]) shows that the same formula, rounded up to the nearest integer, still gives the minimum number of toothpicks that need to be removed.

Comments The algorithm solving the problem is clearly based on the decrease-and-conquer strategy. Interestingly, solving this puzzle for $n = 1$, 2, and 3 may lead one to the wrong expectation of triangular numbers $(1, 3, 6, 10, \ldots)$ as the correct answer to the puzzle.

The puzzle had been found by Martin Gardner in a collection of puzzles by Sam Loyd. Gardner used it in his November 1965 *Scientific American* column, which was later included in his *Colossal Book of Short Puzzles and Problems* [Gar06, Problem 1.20].

145. The Fifteen Puzzle

Solution Reading the tile numbers top down and left to right, we can associate a list of numbers from 1 to 15 with every position in the game. Then the goal is to get from the initial list

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 14 \tag{1}$$

to its permutation

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 \tag{2}$$

by a sequence of allowed moves.

It is useful to consider the *parity* of permutations representing the board's positions. In general, to find the parity of a permutation, one counts the number of its *inversions*, which are pairs of its elements that are out of order: a larger element precedes a smaller one. For example, for permutation 32154 there are four inversions: (3, 2), (3, 1), (2, 1), and (5, 4). Since 4 is an even number, permutation 32154 is said to be *even*. On the other hand, permutation 23154 is *odd*, since it has an odd number of inversions: (2, 1), (3, 1), and (5, 4). The following general property of a permutation's parity is obvious but important: a swap of two adjacent elements in a permutation changes its parity to the opposite one.

Returning to our puzzle, note that the initial and final positions have different parities: (1) is odd, and (2) is even. Let us now investigate how the parity of a permutation representing a board's position may change by the game's moves. The puzzle allows two kinds of moves: a horizontal slide and a vertical slide of a tile to the empty location next to it. A horizontal slide does not change the permutation and hence its parity. A vertical slide of a tile creates a cyclical shift of four consecutive elements in the permutation: for example, the ordering of tiles *j, k, l, m* in Figure 4.105 becomes *k, l, m, j*.

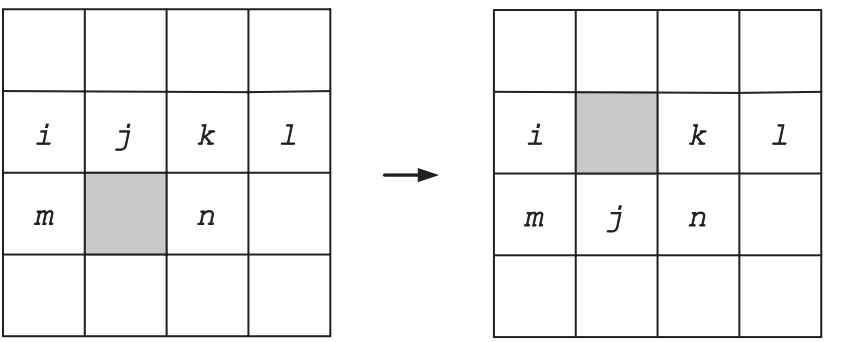


FIGURE 4.105 Impact of sliding of the *j*-tile down.

The same cyclical shift can also be obtained by three swaps of adjacent elements in this quartet:

$$\ldots jklm \ldots \rightarrow \ldots kjlm \ldots \rightarrow \ldots kljm \ldots \rightarrow \ldots klmj \ldots$$

(Although these swaps cannot be done according to the puzzle's rules, they are useful for figuring out the impact of a vertical slide on the position's parity.) Since one swap of adjacent numbers changes the parity to the opposite one, the same outcome is true for three such swaps as well.

It is convenient to interpret the game's moves as a sequence of slides of the empty square. In our puzzle, the empty square occupies the same location in both the initial and goal positions. Therefore, the number of horizontal slides and the number of vertical slides in any sequence solving the puzzle would have to be even

to compensate each slide to the right with a slide to the left and each slide up with a slide down. Since neither horizontal slides nor an even number of vertical slides change the position's parity, the parities of initial and goal positions must be the same for the puzzle to have a solution. Since this necessary condition fails in our case, the puzzle has no solution.

Comments The puzzle is solved by comparing parities of its initial and final positions, which is a standard application of the invariant idea (see the tutorial on algorithm analysis techniques for a discussion and other examples). Note that if we add the row number to the number of inversions, the parity of this sum would remain the same after a vertical slide of a tile as well. Another option is to treat the empty square as having number 16.

While no odd permutation is solvable, every even permutation is, although devising an efficient algorithm to do this is a rather difficult task. In particular, finding the shortest sequence of moves leading to a solution has been proved to be especially hard (NP-complete) for the $n \times n$ boards.

The *Fifteen Puzzle*, also known as the *14–15 Puzzle* and the *Boss Puzzle*, is one of the best known puzzles, rarely missed in general puzzle collections. The puzzle caused a world-wide craze in 1880, partly fueled by a $1000 offer to anybody who could solve this (unsolvable) problem. The invention of the puzzle is often attributed to Sam Loyd (1841–1911), the foremost American inventor of puzzles and a prominent chess composer. This attribution, promoted by Loyd himself, is, however, incorrect: the puzzle was invented by Noyes Chapman, the Postmaster of Canastota, New York, who applied for a patent in March 1880. The patent application was rejected, likely because of its similarity to the patent granted 2 years earlier to Ernest U. Kinsey. For details of this story and many other facts about the puzzle, see the monograph by J. Slocum and D. Sonneveld [Slo06].

146. Hitting a Moving Target

Solution Let us start by numbering the hiding spots left to right from 1 to n. It is reasonable for the gunner to make the first shot at spot 2 (or, symmetrically, at spot $n - 1$) since these are the only shots that can guarantee either hitting the target or ensuring a spot (1 or, respectively, n) where the target can *not* move to after the first shot. We will consider first the case when the target was originally in an even-numbered spot. Then after the first shot, either the shooter hit the target or the target moved to a spot labeled with an odd number greater than or equal to 3. Therefore, if the shooter makes his second shot at spot 3, he will either hit the target or guarantee that the target will be at a spot labeled with an even number greater than or equal to 4. Thus, by continue shooting at consecutive spots $4, 5, \ldots, n - 1$, the shooter will definitely hit the target.

If before the first shot the target was at an odd-numbered spot, it will not be hit by the $n - 2$ shots described above, because its locations and those of the shots will always have the opposite parities. But after the shot at spot $n - 1$, the target will move to a spot with the same parity as the parity of $n - 1$. Therefore, repeating

the symmetric sequence of shots hitting consecutively spots $n-1, n-2, \ldots, 2$ will guarantee hitting the target in this case as well.

To summarize, the sequence of $2(n-2)$ shots at the spots numbered

$$2, 3, \ldots, n-1, n-1, n-2, \ldots, 2$$

guarantees hitting the target for any $n > 2$. When $n = 2$, two shots at the same spot solve the problem as well.

This solution can be nicely illustrated and, in fact, derived from the transformation diagrams shown in Figure 4.106. The diagrams show possible and impossible positions of the target before each shot for $n = 5$ and $n = 6$.

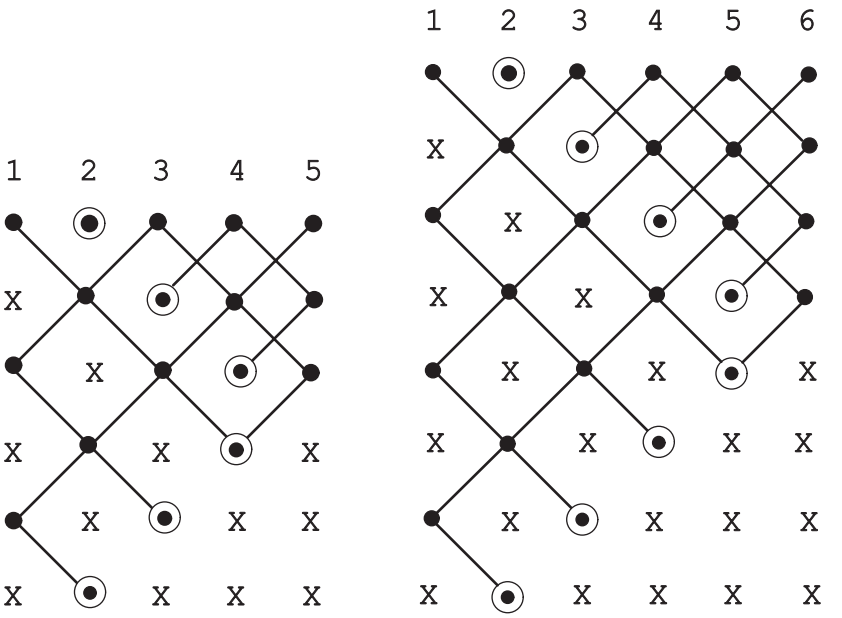


FIGURE 4.106 Illustration of the *Hitting a Moving Target* algorithm for $n = 5$ and $n = 6$. Row i, $i = 1, \ldots, 2n-4$, depicts the possible (shown by the small black circles) and impossible (shown by the X's) locations for the moving target before the ith shot; the shot is indicated by the ring around the location.

Comments The number of spots still available for the target plays the role of a monovariant in this solution. On the other hand, using the transformation diagrams of Figure 4.106 can certainly be viewed as an excellent example of the representation change strategy.

The problem's instance for $n = 1000$ was offered at a Russian mathematical competition in 1999 and published in *Kvant in* 2000 (no. 2, p. 21).

147. Hats with Numbers

Solution The mathematicians can win the bet as follows.

Beforehand, they assign themselves sequential numbers from 0 to $n-1$, say, in alphabetical order of their names. Then on seeing the numbers on the hats of all the other mathematicians, the ith mathematician, $0 \le i \le n-1$, computes the sum S_i of these numbers and guesses his/her hat's number as the smallest nonnegative solution to the equation

$$(S_i + x_i) \bmod n = i,$$

which is

$$x_i = (i - S_i) \bmod n.$$

(In other words, x_i is computed as the remainder of division of the difference between the mathematician's sequential number and the sum of the numbers on all the other hats by the total number of the mathematicians.)

Let $S = h_1 + h_2 + \cdots + h_n$ be the sum of the numbers on all the hats. Obviously, $S = S_i + h_i$ for every $0 \le i \le n-1$. Since the remainders of dividing $0, 1, \ldots, n-1$ by n run the entire range of nonnegative integers from 0 to $n-1$, inclusive, there will exist exactly one such integer j so that

$$j \bmod n = S \bmod n.$$

It is the jth mathematician who will guess correctly his/her number. Indeed,

$$j = j \bmod n = S \bmod n = (S_j + h_j) \bmod n = (S_j + x_j) \bmod n.$$

This implies $h_j \bmod n = x_j \bmod n$, and since both h_j and x_j are between 0 and $n-1$, $x_j = h_j$.

For example, let $n = 5$ and the hat numbers be 3, 4, 0, 3, 2. Then we have the following table:

i	h_i	S_i	x_i	Correctness of the guess
0	3	9	1	no
1	4	8	3	no
2	0	12	0	yes ($j = 2$)
3	3	9	4	no
4	2	10	4	no

Comments In recent years, the puzzle has appeared under the names *Rainbow Hats* and *88 Hats* on several Internet sites and in books devoted to technical interview questions (e.g., [Zho08, p. 31]). We have been unable to determine its origin.

148. One Coin for Freedom

Solution The prisoners have to devise a method that will allow prisoner A to turn one coin to signal prisoner B the location of the cell selected by the jailer. To do this, they can take advantage of the locations of the coins turned, say, tails up. More specifically, they should find a function that maps the locations of all the tails-up coins into the location of the selected cell. A's task is to turn one coin to ensure the mapping; B's task is simply to compute the value of the function for the board presented to him. Here is how it can be done.

First, number the board's cells from 0 to 63, for example, going left to right along every row top to bottom. Let $T_1, T_2, \ldots, T_n$ be the 6-bit binary representations of the sequential numbers assigned to all the cells with tails-up coins on the board presented to A; let J be its 6-bit binary representation of the number assigned to the cell selected by the jailer. Let X be the 6-digit binary representation of the sequential number assigned to the coin to be turned over by A. To find X, find the "exclusive or" (XOR, denoted by $\oplus$) complement of the sum $T = T_1 \oplus T_2 \oplus \cdots \oplus T_n$ to J:

$$T \oplus X = J \text{ or } X = T \oplus J. \quad (1)$$

(If $n = 0$, we assume that $T = O$, the all-zero bit string, and hence $X = O \oplus J = J$.)

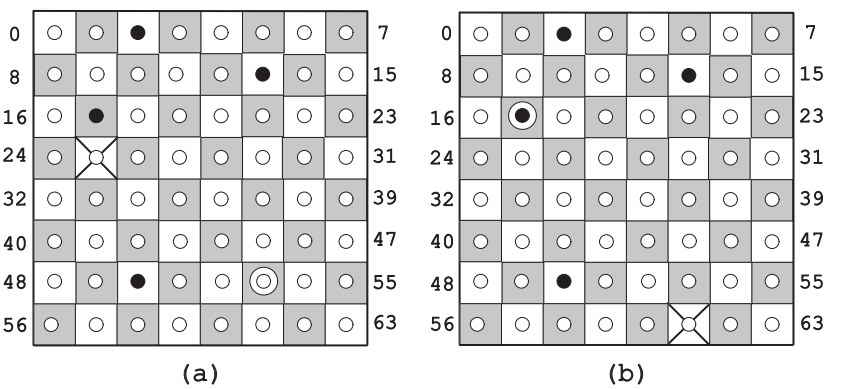


FIGURE 4.107 Two instances of the *One Coin for Freedom* puzzle. Tails-up coins and heads-up coins in the initial configurations are shown as black and white circles, respectively; the cell selected to be guessed is indicated by a cross. The coin to be turned over by the first prisoner is shown by an extra circle around that coin.

For example, for the board in Figure 4.107a,

$$\begin{aligned} T_1 &= \ \ 2_{10} = 000010 \\ T_2 &= 13_{10} = 001101 \\ T_3 &= 17_{10} = 010001 \qquad J = 25_{10} = 011001 \\ T_4 &= 50_{10} = 110010 \\ \hline T &= 101100 \end{aligned}$$

and hence

$$X = T \oplus J = 101100 \oplus 011001 = 110101 = 53_{10}.$$

So, after prisoner A turns the coin at location 53 tails up, prisoner B will see the board with the tails-up coins at locations 2, 13, 17, 50, and 53 and will compute the location of the selected cell as

$$T_1 \oplus T_2 \oplus \cdots \oplus T_n \oplus X = T \oplus X = J = 011001 = 25_{10}.$$

The above example demonstrates the first of the two possible cases—the case where the coin at location X computed by formula (1) is turned heads up. Then turning it over indeed adds X to the other tails-up coins. But what if the coin at that location is already tails up, that is, it is the ith tails-up coin for some $1 \le i \le n$? In this case, turning that coin over makes it heads up, leaving prisoner B to compute the location of the selected cell as $T_1 \oplus \cdots \oplus T_{i-1} \oplus T_{i+1} \oplus \cdots \oplus T_n$. Fortunately, formula (1) works in this case as well because $S \oplus S = O$ for any bit string S. Indeed, if prisoner A computes $X = T \oplus J = T_i$, prisoner B will get the same location of the selected cell as the one used by prisoner A:

$$\begin{aligned} J = T \oplus X &= T_1 \oplus \cdots \oplus T_{i-1} \oplus T_i \oplus T_{i+1} \oplus \cdots \oplus T_n \oplus T_i \\ &= T_1 \oplus \cdots \oplus T_{i-1} \oplus T_{i+1} \oplus \cdots \oplus T_n \oplus T_i \oplus T_i \\ &= T_1 \oplus \cdots \oplus T_{i-1} \oplus T_{i+1} \oplus \cdots \oplus T_n. \end{aligned}$$

For example, if the jailer selects cell 61 on the board with the same four tails-up coins (see Figure 4.107b),

$$\begin{array}{rl} T_1 = \ \ 2_{10} = 000010 & \\ T_2 = 13_{10} = 001101 & \\ T_3 = 17_{10} = 010001 & \qquad J = 61_{10} = 111101 \\ T_4 = 50_{10} = 110010 & \\ \hline T = 101100 & \end{array}$$

and

$$X = T \oplus J = 101100 \oplus 111101 = 010001 = T_3 = 17_{10}.$$

After prisoner A turns the coin at cell 17 heads up, prisoner B will see the board with the tails-up coins at locations 2, 13, and 50 and will compute the location of the selected cell as

$$000010 \oplus 001101 \oplus 110010 = 111101 = 61_{10}.$$

Comments The puzzle's solution provides one more example of the occasional utility of binary numerals.

The one-dimensional version of the puzzle was offered at the International Mathematics Tournament of Towns in the Fall of 2007, but its participants were not asked to provide a complete algorithm for its solution. Since then, the puzzle has appeared in the form given above on several websites.

149. Pebble Spreading

Solution The puzzle has a solution only for $n = 1$ and $n = 2$.

The only legitimate first move solves the puzzle for $n = 1$. Figure 4.108 shows a solution for $n = 2$.

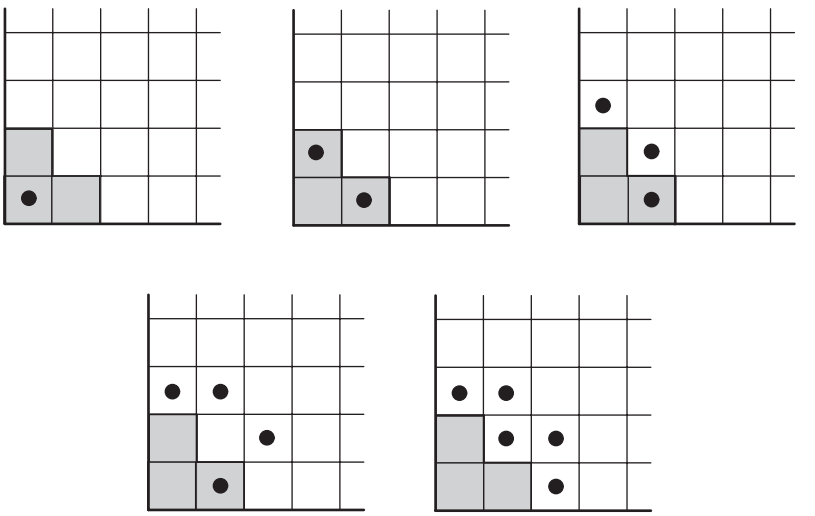


FIGURE 4.108 Sequence of moves freeing the staircase region S_2 (shaded) in the *Pebble Spreading* puzzle.

Now we will show that no staircase region S_n for $n > 2$ can be freed of pebbles by a finite sequence of allowed moves. To do this, we will assign to each cell (i, j) on the board, $i, j \geq 1$, the weight $w(i, j) = 2^{2-i-j}$ (Figure 4.109).

Note that all the cells on the dth diagonal with the equation $i + j = d + 1$, $d = 1, 2, \ldots$, get the same weight, and the sum of the weights of all the cells—which can be computed by summing up the weights in each row—is equal to 4:

$$(1+1/2+1/4+\cdots)+(1/2+1/4+\cdots)+\cdots+(1/2^{j-1}+1/2^{j}+\cdots)+\cdots$$
$$=\frac{1}{1-1/2}+\frac{1/2}{1-1/2}+\cdots+\frac{1/2^{j-1}}{1-1/2}+\cdots=2(1+1/2+\cdots 1/2^{j-1}+\cdots)$$
$$=2\cdot\frac{1}{1-1/2}=4.$$

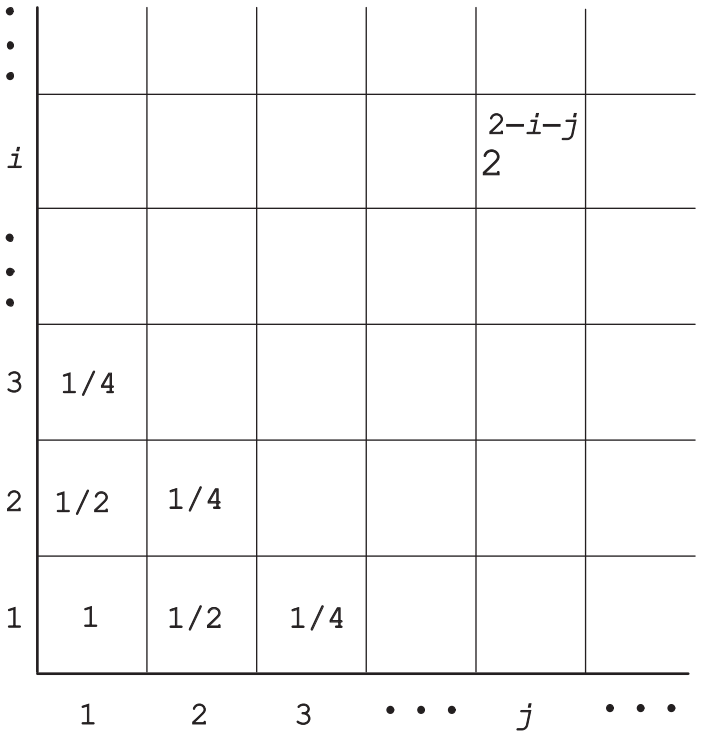


FIGURE 4.109 Weights assigned to the board's cells in the *Pebble Spreading* puzzle.

We can then define the weight of a position as the sum of the weights of all the cells occupied by the pebbles. The weight of the initial position is equal to 1 for any $n \geq 1$. A single move—and hence a finite sequence of moves—cannot change the position's weight, because the weight of a pebble removed from cell (i, j) is compensated by the weights of the new pebbles in cells $(i + 1, j)$ and $(i, j + 1)$: $2^{2-(i+j)} = 2^{2-(i+1+j)} + 2^{2-(i+j+1)}$.

This immediately implies that no staircase region S_n where $n \geq 4$ can be freed from pebbles by a finite sequence of moves. If it were possible, in their final position the pebbles would have occupied some region R_n composed of a finite number of cells outside S_n. The total weight of that region, $W(R_n)$, would have to be smaller than the total weight of all the cells outside S_4. The latter can be found by subtracting the weight of the cells composing S_4 from the weight of all the cells on the board (Figure 4.110a):

$$W(R_4) < 4 - W(S_4) = 4 - (1 + 2 \cdot 1/2 + 3 \cdot 1/4 + 4 \cdot 1/8) = 3/4.$$

Thus, since $W(R_n) \leq W(R_4) < 1$ for $n \geq 4$, no staircase region S_n where $n \geq 4$ can be freed from pebbles by a finite sequence of moves.

The staircase S_3 cannot be freed from its pebbles either. To prove this, we note that the first row and first column of the board always contain one pebble each. Therefore, if all the pebbles from S_3 were transformed to occupy some region R_3

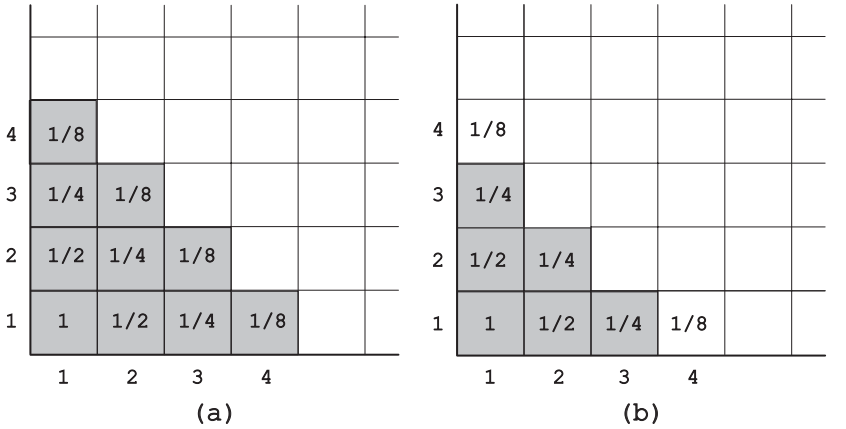


FIGURE 4.110 (a) Weights of the cells for staircase region S_4 and (b) weights of the cells for staircase region S_3 plus two outside cells in row 1 and column 1 in the *Pebble Spreading* puzzle.

outside S_3, R_3 would have been composed of one cell in the first row, one cell in the first column, and a set Q_3 of cells in the other rows and columns (Figure 4.110b). The upper bound for $W(Q_3)$ can be calculated as the difference between the weight of the whole quadrant and the weights of the cells in the first column, the first row, and the weight of cell (2,2):

$$W(Q_3) < 4 - [1 + 2(1/2 + 1/4 + \cdots) + 1/4] = 3/4.$$

This implies the following upper bound for $W(R_3)$:

$$W(R_3) < 1/8 + 1/8 + 3/4 = 1.$$

Since $W(R_3) < 1$, the pebbles of S_3 cannot be transformed to occupy R_3.

Comments The invariant idea—the weight of the current position—exploited in this solution is similar to that used by J. Conway in the *Solitaire Army* puzzle (#132).

The puzzle was proposed by M. Kontsevich in the November 1981 issue of *Kvant* (p. 21, Problem M715); A. Khodulev provided a detailed solution to and a generalization of the puzzle in the magazine's July 1982 issue.

150. Bulgarian Solitaire

Solution It is convenient to think of each pile as a stack of coins. On each iteration, the algorithm goes through the row of such stacks, taking one coin from each of them—which we may assume, without loss of generality, is taken from the bottom of each stack—and puts these coins in the order taken in a new stack. We will first put the new stack before the others and then, if it is smaller than the next

one, insert it in its proper place in the pile row to have the entire row sorted in nonincreasing order of the pile sizes. This process is illustrated in Figure 4.111, where the coins are labeled by letters.

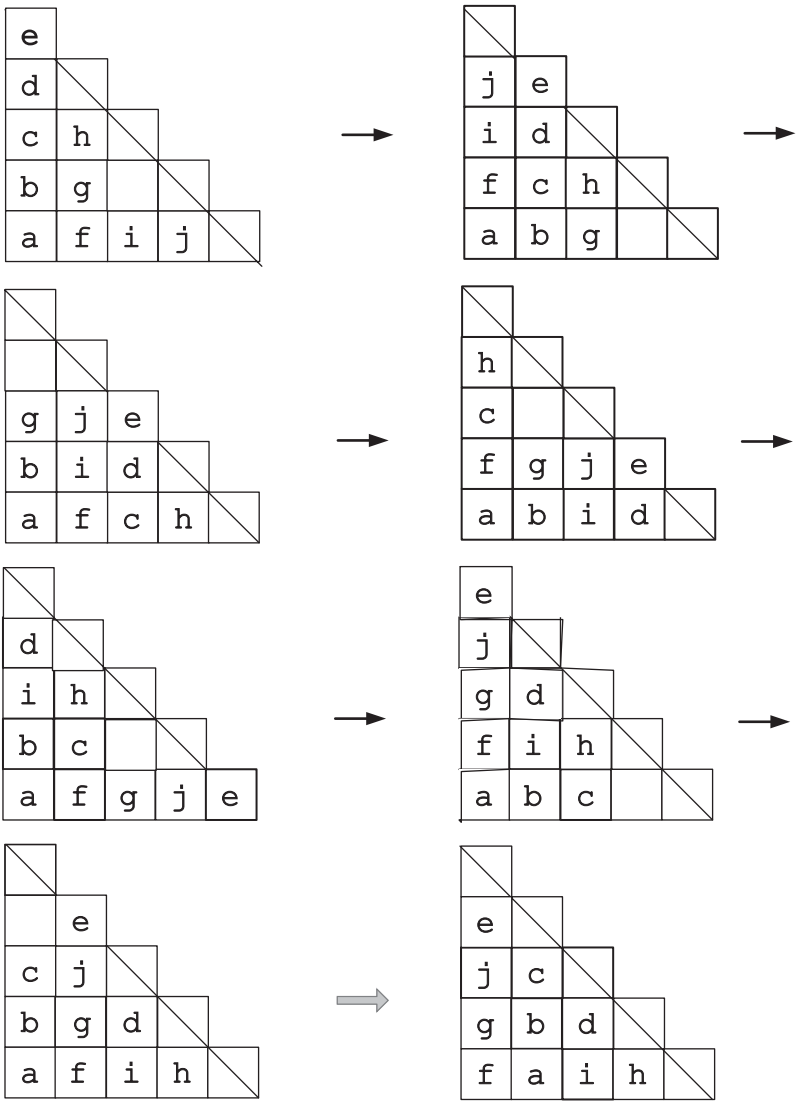


FIGURE 4.111 *Bulgarian Solitaire* example. The last iteration requires sorting of the piles.

Since the number of ways to partition n into a sum of positive integers (pile sizes) is finite, the algorithm will have to enter a loop after a finite number of iterations. Our task is to show that for any initial partition of $n = 1 + 2 + \cdots + k$, including the possibility of starting with a single pile of n coins, the algorithm will always reach the partition $(k, k-1, \ldots, 1)$. This partition is a one-state loop: the algorithm transforms it into itself. There is no other one-state loop. Indeed, if the algorithm transforms $(n_1, n_2, \ldots, n_s)$, where $n_1 \geq n_2 \geq \cdots \geq n_s$, into

$(s, n_1 - 1, n_2 - 1, \ldots, n_{s-1} - 1)$ and the two tuples are the same, s must be the largest component in the second tuple and $n_s = 1$ (otherwise, the tuples would not be of the same size). Equating the corresponding components of the two tuples yields a system of linear equations $n_1 = s$ and $n_i = n_{i-1} - 1$ for $i = 2, \ldots, s$, which can be easily solved by backward substitution: $n_{s-1} = n_s + 1 = 2$, and so on, until $n_1 = n_2 + 1 = k$ and $s = k$.

Now we will show that there exists no loop containing more than one partition. First, no pile sorting can happen within a loop. One can prove this assertion by assigning a weight to each coin equal to the sum of its stack number and the coin's position in its stack counting from the stack's bottom. For example, the following weights are assigned to the coins in the starting partition in Figure 4.111: $a(1 + 1)$, $b(1 + 2), \ldots, j(4 + 1)$. Then we can define the weight of a partition (not necessarily sorted) as the sum of the weights of all the coins composing it. For example, the weight of the starting partition in Figure 4.111 is 41. It is easy to see that applying the algorithm's operation to a partition does not change the weight if sorting of the output is not required: the weight $i + j$ of a coin in stack i and position j becomes $(i + 1) + (j - 1)$ if $j > 1$ and $j + i$ if $j = 1$, respectively. But if stacks composing a partition are not in nonincreasing order of their size, then sorting them does decrease the partition's weight. Hence, no sorting can happen within a loop of coin partitions.

Second, in any sequence of coin partitions produced by the algorithm that does not require sorting, each of the coins circles along its diagonal, as do the diagonal's empty spots if any (see Figure 4.111 for an example). But then, if we have at least one coin on the dth diagonal, we cannot have any empty spots on the $(d - 1)$th diagonal. Indeed, if it were possible, the empty spot on the $(d - 1)$th diagonal and the coin on the dth diagonal would have reached the same height in their respective stacks after a finite number of steps—the situation requiring sorting of the stacks (see the last iteration in Figure 4.111).

This immediately implies that only the largest diagonal can have some missing coins in a sequence of states forming a loop. But if $n = 1 + 2 + \cdots + k$, all the k diagonals must be full. Indeed, if there were no coins on the kth diagonal, the total number of coins would have been not larger than $1 + 2 + \cdots + (k - 1)$. And if there were a coin on the $(k + 1)$th diagonal, all the smaller diagonals would have to be full as proved above and the total number of coins would have been larger then $1 + 2 + \cdots + k$. Thus, we have the single-partition loop for such n's.

Comments Bulgarian solitaire was popularized by Martin Gardner in his *Scientific American* column in 1983 (see also [Gar97b, pp. 36–43]). It was based on a paper by the Danish mathematician Jørgen Brandt, which was published a year earlier. Since then several interesting results about this game and its variations have been obtained (e.g., [Gri98]). In particular, it has been proved that no more than $k^2 - k$ iterations are required to reach the stable partition $(k, k - 1, \ldots, 1)$ from any initial one.

References

[Ash04] Ash, J. M., and Golomb, S. W. Tiling deficient rectangles with trominoes. *Mathematics Magazine*, vol. 77, no. 1 (Feb. 2004), 46–55.

[Ash90] Asher, M. A river-crossing problem in cross-cultural perspective. *Mathematics Magazine*, vol. 63, no. 1 (Feb. 1990), 26–29.

[Ave00] Averbach, B., and Chein, O. *Problem Solving Through Recreational Mahtematics*. Dover, 2000.

[Bac12] Bachet, C. *Problèmes plaisans et delectables qui se font par les nombres*. Paris, 1612.

[Backh] Backhouse, R. *Algorithmic problem solving course website*. www.cs.nott.ac.uk/~rcb/G51APS/exercises/InductionExercises.pdf (accessed Oct. 4, 2010).

[Bac08] Backhouse, R. The capacity-C torch problem. *Mathematics of Program Construction 9th International Conference* (*MPC* 2008), Marseille, France, July 15–18, 2008, Springer-Verlag, 57–78.

[Bal87] Ball, W. W. Rouse, and Coxeter, H. S. M. *Mathematical Recreations and Essays*, 13th edition. Dover, 1987. www.gutenberg.org/ebooks/26839 (1905 edition; accessed Oct. 10, 2010).

[Bea92] Beasley, J. D. *The Ins and Outs of Peg Solitaire*. Oxford University Press, 1992.

[Bec97] Beckwith, D. Problem 10459, in Problems and Solutions, *American Mathematical Monthly*, vol. 104, no. 9 (Nov. 1997), 876.

[Bel09] Bell, J., and Stevens, B. A survey of known results and research areas for *n*-queens. *Discrete Mathematics*, vol. 309, issue 1 (Jan. 2009), 1–31.

[Ben00] Bentley, J. *Programming Pearls*, 2nd ed. Addison-Wesley, 2000.

[Ber04] Berlekamp, E. R., Conway, J. H., and Guy, R. K. *Winning Ways for Your Mathematical Plays*, Volume 4, 2nd ed. A K Peters, 2004.

[Ber91] Bernhardsson, B. Explicit solutions to the *n*-queens problem for all *n*. *SIGART Bulletin*, vol. 2, issue 2 (April 1991), 7.

[Bogom] Bogomolny, A. *Interactive Mathematics Miscellany and Puzzles.* www.cut-the-knot.org (accessed Oct. 4, 2010).

[Bog00] Bogomolny, A. The three jugs problem. The Mathematical Association of America, May 2000. www.maa.org/editorial/knot/water.html#kasner (accessed Oct. 10, 2010).

[Bol07] Bollobás, B. *The Art of Mathematics: Coffee Time in Memphis*. Cambridge University Press, 2007.

[Bos07] Bosova, L. L., Bosova, A. Yu, and Kolomenskaya, Yu. G. *Entertaining Informatics Problems*, 3rd ed., BINOM, 2007 (in Russian).

[Bro63] Brooke, M. *Fun for the Money*. Charles Scribner's Sons, 1963.

[CarTalk] Archive of the U.S. National Public Radio talk show *Car Talk*. www.cartalk.com/content/puzzler (accessed Oct. 4, 2010).

[Chr84] Christen, C., and Hwang, F. Detection of a defective coin with a partial weight information. *American Mathematical Monthly*, vol. 91, no. 3 (March 1984), 173–179.

[Chu87] Chu, I-Ping, and Johnsonbaugh, R. Tiling and recursion. *ACM SIGCSE Bulletin*, vol. 19, issue 1 (Feb. 1987), 261–263.

[Cor09] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*, 3rd edition. MIT Press, 2009.

[Cra07] Crack, T. F. *Heard on the Street: Quantitative Questions from Wall Street Job Interviews,* 10th ed. Self-published, 2007.

[Cso08] Csorba, P., Hurkens, C. A., and Woeginger, G. J. The Alcuin number of a graph. *Proceedings of the 16th Annual European Symposium on Algorithms. Lecture Notes in Computer Science*, vol. 5193, 2008, 320–331.

[Dem02] Demaine, E. D., Demaine, M. L., and Verrill, H. Coin-moving puzzles. In R. J. Nowakowski, editor, *More Games of No Chance*. Cambridge University Press, 2002, 405–431.

[Dij76] Dijkstra, E. W. *A Discipline of Programming*. Prentice Hall, 1976.

[Dud02] Dudeney, H. E. *The Canterbury Puzzles and Other Curious Problems*. Dover, 2002. www.gutenberg.org/ebooks/27635 (1919 edition; accessed Oct. 10, 2010).

[Dud58] Dudeney, H. E. *Amuzements in Mathematics*. Dover, 1958. www.gutenberg.org/ebooks/16713 (first published in 1917; accessed Oct. 10, 2010).

[Dud67] Dudeney, H. E. (edited by Martin Gardner). *536 Puzzles & Curious Problems*. Charles Scribner's Sons, 1967.

[Dyn71] Dynkin, E. B., Molchanov, S. A., Rozental, A. L., and Tolpygo, A. K. *Mathematical Problems*, 3rd revised edition, Nauka, 1971 (in Russian).

[Eng99] Engel, A. *Problem-Solving Strategies*. Springer, 1999.

[Epe70] Eperson, D. B. Triangular (Old) Pennies. *The Mathematical Gazette*, vol. 54, no. 387 (Feb. 1970), 48–49.

[Fom96] Fomin, D., Genkin, S., and Itenberg, I. *Mathematical Circles* (*Russian Experience*). American Mathematical Society, Mathematical World, Vol. 7, 1996 (translated from Russian).

[Gar99] Gardiner, A. *Mathematical Puzzling*. Dover, 1999.

[Gar61] Gardner, M. *Mathematical Puzzles*. Thomas Y. Crowell, 1961.

[Gar71] Gardner, M. *Martin Gardner's 6th Book of Mathematical Diversions from Scientific American*. W. H. Freeman, 1971.

[Gar78] Gardner, M. *aha! Insight*. Scientific American/W. H. Freeman, 1978.

[Gar83] Gardner, M. *Wheels, Life, and Other Mathematical Amusements.* W. H. Freeman, 1983.

[Gar86] Gardner, M. *Knotted Doughnuts and Other Mathematical Entertainments*. W. H. Freeman, 1986.

[Gar87] Gardner, M. *The Second Scientific American Book of Puzzles and Games*. University of Chicago Press, 1987.

[Gar88a] Gardner, M. *Hexaflexagons and Other Mathematical Diversions*: *The First Scientific American Book of Puzzles and Games*. University of Chicago Press, 1988.

[Gar88b] Gardner, M. *Time Travel and Other Mathematical Bewilderments*. W. H. Freeman, 1988.

[Gar89] Gardner, M. *Mathematical Carnival*. The Mathematical Association of America, 1989.

[Gar97a] Gardner, M. *Penrose Tiles to Trapdoor Chiphers ... and the Return of Dr. Matrix*, revised edition. The Mathematical Association of America, 1997.

[Gar97b] Gardner, M. *The Last Recreations: Hidras, Eggs, and Other Mathematical Mystifications.* Springer, 1997.

[Gar06] Gardner, M. *Colossal Book of Short Puzzles and Problems*. W. W. Norton, 2006.

[Gik76] Gik, E. Ya. *Mathematics on the Chessboard*. Nauka, 1976 (in Russian).

[Gik80] Gik, E. The Battleship game. *Kvant*, Nov. 1980, 30–32, 62–63 (in Russian).

[Gin03] Ginat, D. The greedy trap and learning from mistakes. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, ACM, 2003, 11–15.

[Gin06] Ginat, D. Coloful Challenges column. *inroads—SIGCSE Bulletin*, vol. 38, no. 2 (June 2006), 21–22.

[Gol54] Golomb, S. W. Checkerboards and polyominoes. *American Mathematical Monthly*, vol. 61, no. 10 (Dec. 1954), 675–682.

[Gol94] Golomb, S. W. *Polyominoes: Puzzles, Patterns, Problems, and Packings*, 2nd edition. Princeton University Press, 1994.

[Graba] Grabarchuk, S. Coin triangle. From *Puzzles.com.* www.puzzles.com/PuzzlePlayground/CoinTriangle/CoinTriangle.htm (accessed Oct. 4, 2010).

[Gra05] Grabarchuk, S. *The New Puzzle Classics: Ingenious Twists on Timeless Favorites*. Sterling Publishing, 2005.

[Gra94] Graham, R. L., Knuth, D. E. and Patashnik, O. *Concrete Mathematics: A Foundation for Computer Science*, 2nd ed. Addison-Wesley, 1994.

[Gre73] Greenes, C. E. Function generating problems: the row chip switch. *Arithmetic Teacher*, vol. 20 (Nov. 1973), 545–549.

[Gri98] Griggs, J. R., and Ho, Chih-Chang. The cycling of partitions and compositions under repeated shifts. *Advances in Applied Mathematics*, vol. 21, no. 2 (1998), 205–227.

[Had92] Hadley, J., and Singmaster, D. Problems to sharpen the young. *Mathematical Gazette*, vol, 76, no. 475 (March 1992), 102–126.

[Hes09] Hess, D. *All-Star Mathlete Puzzles*. Sterling, 2009.

[Hof79] Hofstadter, D. *Gödel, Escher, Bach: An Eternal Golden Braid*. Basic Books, 1979.

[Hur00] Hurkens, C. A. J. Spreading gossip efficiently. *NAW*, vol. 5/1 (June 2000), 208–210.

[Iba03] Iba, G., and Tanton, J. Candy sharing. *American Mathematical Monthly*, vol. 110, no. 1 (Jan. 2003), 25–35.

[Ign78] Ignat'ev, E. I. *In the Kindom of Quick Thinking*. Nauka, 1978 (in Russian).

[Iye66] Iyer, M., and Menon, V. On coloring the $n \times n$ chessboard. *American Mathematical Monthly,* vol. 73, no. 7 (Aug.–Sept. 1966), 721–725.

[Kho82] Khodulev, A. Relocation of chips. *Kvant*, July 1982, 28–31, 55 (in Russian).

[Kin82] King, K. N., and Smith-Thomas, B. An optimal algorithm for sink-finding. *Information Processing Letters*, vol. 14, no. 3 (May 1982), 109–111.

[Kle05] Kleinberg, J., and Tardos, E. *Algorithm Design*. Addison-Wesley, 2005.

[Knott] Knott, R. *Fibonacci Numbers and the Golden Section*. www.mcs.surrey.ac.uk/Personal/R.Knott/Fibonacci/ (accessed Oct. 4, 2010).

[Knu97] Knuth, D. E. *The Art of Computer Programming, Volume 1: Fundamental Algorithms,* 3rd ed. Addison-Wesley, 1997.

[Knu98] Knuth, D. E. *The Art of Computer Programming, Volume 3: Sorting and Searching*, 2nd ed. Addison-Wesley, 1998.

[Knu11] Knuth, D. E. *The Art of Computer Programming, Volume 4A, Combinatorial Algorithms, Part 1*. Pearson, 2011.

[Kon96] Konhauser J. D. E., Velleman, D., and Wagon, S. *Which Way Did the Bicycle Go?: And Other Intriguing Mathematical Mysteries*. The Dolciani Mathematical Expositions, No. 18, The Mathematical Association of America, 1996.

[Kor72] Kordemsky, B. A. *The Moscow Puzzles: 359 Mathematical Recreations*. Scribner, 1972 (translated from Russian).

[Kor05] Kordemsky, B. A. *Mathematical Charmers*. Oniks, 2005 (in Russian).

[Kra53] Kraitchik, M. *Mathematical Recreations*, 2nd revised edition. Dover, 1953.

[Kre99] Kreher, D. L., and Stinson, D. R. *Combinatorial Algorithms: Generation, Enumeration, and Search*. CRC Press, 1999.

[Kur89] Kurlandchik, L. D., and Fomin, D. B. Etudes on the semi-invariant. *Kvant*, no. 7, 1989, 63–68 (in Russian).

[Laa10] Laakmann, G. *Cracking the Coding Interview*, 4th ed. CareerCup, 2010.

[Leh65] Lehmer, D. H. Permutation by adjacent interchanges. *American Mathematical Monthly*, vol. 72, no. 2 (Feb. 1965), 36–46.

[Leino] Leino, K. R. M. *Puzzles*. research.microsoft.com/en-us/um/people/leino/puzzles.html (accessed Oct. 4, 2010).

[Lev06] Levitin, A. *Introduction to the Design and Analysis of Algorithms*, 2nd edition. Pearson, 2006.

[Lev81] Levmore, S. X., and Cook, E. E. *Super Strategies for Puzzles and Games*. Doubleday, 1981.

[Loy59] Loyd, S. (edited by M. Gardner) *Mathematical Puzzles of Sam Loyd*. Dover, 1959.

[Loy60] Loyd, S. (edited by M. Gardner) *More Mathematical Puzzles of Sam Loyd*. Dover, 1960.

[Luc83] Lucas, E. *Récréations mathématiques*, Vol. 2. Gauthier Villars, 1883.

[Mac92] Mack, D. R. *The Unofficial IEEE Brainbuster Gamebook: Mental Workouts for the Technically Inclined*. IEEE Press, 1992.

[Man89] Manber, U. *Introduction to Algorithms: A Creative Approach*. Addison-Wesley, 1989.

[Mar96] Martin, G. E. *Polyominoes: A Guide to Puzzles and Problems in Tiling*. The Mathematical Association of America, 1996.

[MathCentral] *Math Central*. mathcentral.uregina.ca/mp (accessed Oct. 4, 2010).

[MathCircle] *The Math Circle*. www.themathcircle.org/researchproblems.php (accessed Oct. 4, 2010).

[Mic09] Michael, T. S. *How to Guard an Art Gallery*. John Hopkins University Press, 2009.

[Mic08] Michalewicz, Z., and Michalewicz, M. *Puzzle-Based Learning: An Introduction to Critical Thinking, Mathematics, and Problem Solving*. Hybrid Publishers, 2008.

[Moo00] Moore, C., and Eppstein, D. One-dimensional peg solitaire and Duotaire. *Proceedings of MSRI Workshop on Combinatorial Games*, Berkeley, CA. MSRI Publications 42. Springer, 2000, 341–350.

[Mos01] Moscovich, I. 1000 *Play Thinks: Puzzles, Paradoxes, Illusions, and Games*. Workman Publishing, 2001.

[Nie01] Niederman, *Hard-to-Solve Math Puzzles*. Sterling Publishing, 2001.

[OBe65] O'Beirne, T. H. *Puzzles & Paradoxes*. Oxford University Press, 1965.

[Par95] Parberry, I. *Problems on Algorithms*. Prentice-Hall, 1995.

[Pet03] Peterson, Ivar. Measuring with jugs. The Mathematical Association of America, June 2003. www.maa.org/mathland/mathtrek_06_02_03.html (accessed Oct. 4, 2010).

[Pet97] Petković, M. *Mathematics and Chess: 110 Entertaining Problems and Solutions*. Dover, 1997.

[Pet09] Petković, M. *Famous Puzzles of Great Mathematicians*. The American Mathematical Society, 2009.

[Pic02] Pickover, C. A. *The Zen of Magic Squares, Circles, and Stars: An Exhibition of Surprising Structures across Dimensions*. Princeton University Press, 2002.

[Poh72] Pohl, I. A sorting problem and its complexity. *Communications of the ACM*, vol. 15, issue 6 (June 1972), 462–464.

[Pol57] Pólya, G. *How to Solve It*: *A New Aspect of Mathematical Method*, 2nd ed. Princeton University Press, 1957.

[Pou03] Poudstone, W. *How Would You Move Mount Fuji*? *Microsoft's Cult of the Puzzle—How the World's Smartest Companies Select the Most Creative Thinkers*. Little-Brown, 2003.

[Pre89] Pressman, I., and Singmaster, D. "The Jealous Husbands" and "The Missionaries and Cannibals." *Mathematical Gazette*, 73, no. 464 (June 1989), 73–81.

[ProjEuler] *Project Euler*. projecteuler.net (accessed Oct. 4, 2010).

[Ran09] Rand, M. On the Frame-Stewart algorithm for the Tower of Hanoi. www2.bc.edu/~grigsbyj/Rand_Final.pdf (accessed Oct. 4, 2010).

[Rob98] Robertson, J., and Webb, W. *Cake Cutting Algorithms*. A K Peters, 1998.

[Ros07] Rosen, K. *Discrete Mathematics and Its Applications*, 6th edition. McGraw-Hill, 2007.

[Ros38] Rosenbaum, J. Problem 319, *American Mathematical Monthly*, vol. 45, no. 10 (Dec. 1938), 694–696.

[Rot02] Rote, G. Crossing the bridge at night. *EATCS Bulletin*, vol. 78 (Aug. 2002), 241–246.

[Sav03] Savchev, S., and Andreescu, T. *Mathematical Miniatures*. The Mathematical Association of America, Anneli Lax New Mathematical Library, Volume #43, Washington, DC, 2003.

[Sch68] Schuh, F. *The Master Book of Mathematical Recreations*. Dover, 1968 (translated from Dutch).

[Sch04] Schumer, P. D. *Mathematical Journeys*. Wiley, 2004.

[Sch80] Schwartz, B. L., ed. *Mathematical Solitaires & Games*. (Excursions in Recreational Mathematics Series 1), Baywood Publishing, 1980.

[Sco44] Scorer, R. S., Grundy, P. M., and Smith, C. A. B. Some binary games. *Mathematical Gazette*, vol. 28, no. 280 (July 1944), 96–103.

[Sha02] Shasha, D. *Doctor Ecco's Cyberpuzzles*. Norton, 2002.

[Sha07] Shasha, D. *Puzzles for Programmers and Pros*. Wiley, 2007.

[Sillke] Sillke, T. Crossing the bridge in an hour. www.mathematik.uni-bielefeld.de/~sillke/PUZZLES/crossing-bridge (accessed Oct. 4, 2010).

[Sin10] Singmaster, D. *Sources in Recreational Mathematics: An Annotated Bibliography*, 8th preliminary edition. www.g4g4.com/MyCD5/SOURCES/SOURCE1.DOC (accessed Oct. 4, 2010).

[Slo06] Slocum, J. and Sonneveld, D. *The 15 Puzzle: How It Drove the World Crazy. The Puzzle That Started the Craze of 1880. How America's Greatest Puzzle Designer, Sam Loyd, Fooled Everyone for 115 Years*. Slocum Puzzle Foundation, 2006.

[Sni02] Sniedovich, M. The bridge and torch problem. Feb. 2002. www.tutor.ms.unimelb.edu.au/bridge (accessed Oct. 4, 2010).

[Sni03] Sniedovich, M. OR/MS Games: 4. The Joy of Egg-Dropping in Braunschweig and Hong Kong. *INFORMS Transactions on Education*, vol. 4, no. 1 (Sept. 2003), 48–64.

[Spi02] Spivak, A. V. *One Thousand and One Mathematical Problems*. Education, 2002 (in Russian).

[Ste64] Steinhaus, H. *One Hundred Problems in Elementary Mathematics*. Basic Books, 1964 (translated from Polish).

[Ste04] Stewart, I. *Math Hysteria*. Oxford University Press, 2004.

[Ste06] Stewart, I. *How to Cut a Cake: And Other Mathematical Conundrums*. Oxford University Press, 2006.

[Ste09] Stewart, I. *Professor Stewart's Cabinet of Mathematical Curiosities*. Basic Books, 2009.

[Tan01] Tanton, J. *Solve This: Math Activities for Students and Clubs*. The Mathematical Association of America, 2001.

[techInt] *techInterviews*. www.techinterview.org/archive (accessed Oct. 4, 2010).

[Ton89] Tonojan, G. A. Canadian mathematical olympiads. *Kvant*, 1989, no. 7, 75–76 (in Russian).

[Tri69] Trigg, C. W. Inverting coin triangles. *Journal of Recreational Mathematics*, vol. 2 (1969), 150–152.

[Tri85] Trigg, C. W. *Mathematical Quickies*. Dover, 1985.

[Twe39] Tweedie, M. C. K. A graphical method of solving Tartaglian measuring puzzles. *Mathematical Gazette*, vol. 23, no. 255 (July 1939), 278–282.

[Weiss] Weisstein, E. W. Josephus Problem. From *MathWorld*–A Wolfram Web Resource. mathworld.wolfram.com/JosephusProblem.html (accessed Oct. 4, 2010).

[Win04] Winkler, P. *Mathematical Puzzles: Connoisseur's Collection*. A K Peters, 2004.

[Win07] Winkler, P. *Mathematical Mind-Benders*. A K Peters, 2007.

[Zho08] Zhow, X. *A Practical Guide to Quantitative Finance Interview*. Lulu.com, 2008.

Design Strategy and Analysis Index

This index groups the puzzles by the design strategies and analysis questions. The puzzles used in the tutorials are marked by "tut"; all the other puzzles are indexed by their number in the main portion of the book. Some of the puzzles are listed in more than one category.

ANALYSIS

Output Analysis

tut *Chess Invention*
tut *Square Build-Up*
6. Predicting a Finger Count
17. A King's Reach
26. Find the Rank
31. The Three Pile Trick
52. Counting Triangles
55. Odometer Puzzle
57. Fibonacci's Rabbits Problem
66. Remaining Number
68. Digit Sum
77. Searching for a Pattern
79. Locker Doors
100. A Knight's Reach
120. Penny Distribution Machine
141. The Josephus Problem

Step Counting

tut *Tower of Hanoi*
2. Glove Selection
19. Page Numbering
32. Single-Elimination Tournament
61. Checkers on a Diagonal
120. Penny Distribution Machine

Other

56. Lining Up Recruits
63. Pluses and Minuses
69. Chips on Sectors
93. Hitting a Battleship
97. The Game of Topswops
109. Double-*n* Dominoes
133. The Game of Life
150. Bulgarian Solitaire

INVARIANTS

Parity

tut *Domino Tiling of Deficient Chessboards*
tut *The Königsberg Bridges Problem*
28. Figure Tracing
38. Tetromino Tiling
50. Last Ball
61. Checkers on a Diagonal
63. Pluses and Minuses
66. Remaining Number
69. Chips on Sectors
87. Upside Down Glasses
91. Horizontal and Vertical Dominoes
99. Reversal of Sort
107. The Fox and the Hare
109. Double-*n* Dominoes
112. Domino Tiling Revisited
145. The Fifteen Puzzle

Coloring

tut *Domino Tiling of Deficient Chessboards*
tut *Chickens in the Corn*
18. A Corner-to-Corner Journey
39. Board Walks
47. Exhibition Planning
73. Rooster Chase
91. Horizontal and Vertical Dominoes
103. Jumping to the Other Side

Other Invariants

tut *Breaking a Chocolate Bar* number of pieces
5. Row and Column Exchanges row and column elements

8.	Jigsaw Puzzle Assembly	number of pieces
15.	Tromino Tilings	divisibility by 3
40.	Four Alternating Knights	clockwise order
92.	Trapezoid Tiling	divisibility by 3
96.	Tiling a Staircase Region	divisibility by 3
104.	Pile Splitting	sum of the products
105.	The MU Puzzle	divisibility by 3
110.	The Chameleons	differences mod 3
120.	Penny Distribution Machine	binary representation
132.	The Solitaire Army	position weight (monovariant)
143.	Infected Chessboard	region's perimeter (monovariant)
149.	Pebble Spreading	position weight

BACKTRACKING

tut	*The n-Queens Problem*
27.	The Icosian Game
29.	Magic Square Revisited
70.	Jumping into Pairs I

DECREASE-AND-CONQUER

Decrease-by-1

tut	*Celebrity Problem*	
3.	Rectangle Dissection	bottom-up
4.	Ferrying Soldiers	
22.	Team Ordering	
32.	Single-Elimination Tournament	
33.	Magic and Pseudo-Magic	bottom-up
42.	The Other Wolf-Goat-Cabbage Puzzle	bottom-up
43.	Number Placement	
46.	Tricolor Arrangement	
59.	Hats Of Two Colors	bottom-up
65.	Code Guessing	
68.	Digit Sum	
81.	Celebrity Problem Revisited	
82.	Heads Up	
83.	Restricted Tower of Hanoi	
86.	Rumor Spreading II	bottom-up
90.	Seating Rearrangements	bottom-up
94.	Searching a Sorted Table	
106.	Turning on a Light Bulb	
107.	The Fox and the Hare	

113. Coin Removal
114. Crossing Dots bottom-up
126. Dividing a Cake Fairly
134. Point Coloring

Decrease-by-2

16. Making Pancakes
70. Jumping into Pairs I
71. Marking Cells I bottom-up
72. Marking Cells II bottom-up
87. Upside Down Glasses
109. Double-*n* Dominoes
117. One-Dimensional Solitaire
128. Security Switches
144. Killing Squares

Decrease-by-another-constant

48. McNugget Numbers by 4
64. Creating Octagons by 8
78. Straight Tromino Tiling by 3
96. Tiling a Staircase Region by 6
131. Tait's Counter Puzzle by 4

Decrease-by-constant-factor

tut *Number Guessing* factor of 2
10. A Fake Among Eight Coins factor of 2 or 3
30. Cutting a Stick factor of 2
31. The Three Pile Trick factor of 3
32. Single-Elimination Tournament factor of 2
53. Fake-Coin Detection with a Spring Scale factor of 2
54. Cutting a Rectangular Board factor of 2
116. Bye Counting factor of 2
141. The Josephus Problem factor of 2

Decrease-by-variable-number

23. Polish National Flag Problem
28. Figure Tracing
84. Pancake Sorting
129. Reve's Puzzle

DIVIDE-AND-CONQUER

tut *Tromino Puzzle*
15. Tromino Tilings
37. 2*n*-Counters Problem
38. Tetromino Tiling
64. Creating Octagons
78. Straight Tromino Tiling
80. The Prince's Tour
91. Horizontal and Vertical Dominoes
92. Trapezoid Tiling
95. Max-Min Weights
96. Tiling a Staircase Region
101. Room Painting
113. Coin Removal
119. Colored Tromino Tiling
132. The Solitaire Army

DYNAMIC PROGRAMMING

tut *Shortest Path Counting*
13. Blocked Paths
20. Maximum Sum Descent
62. Picking Up Coins
98. Palindrome Counting
104. Pile Splitting

EXHAUSTIVE SEARCH

tut Magic Square
15. Tromino Tilings
35. Three Jugs

GREEDY APPROACH

tut *Non-Attacking Kings*
tut *Bridge Crossing at Night*
24. Chessboard Colorings
34. Coins on a Star
45. A Knight's Shortest Path
67. Averaging Down
73. Rooster Chase
76. Efficient Rook
85. Rumor Spreading I

108. The Longest Route
115. Bachet's Weights
121. Super-Egg Testing
124. Chain Cutting
127. The Knight's Tour

ITERATIVE IMPROVEMENT

tut *Lemonade Stand Placement*
tut *Positive Changes*
67. Averaging Down
82. Heads Up
122. Parliament Pacification
138. Candy Sharing
139. King Arthur's Round Table
146. Hitting a Moving Target

TRANSFORM-AND-CONQUER

Instance Simplification

tut	*Anagram Detection*	presorting
3.	Rectange Dissection	odd to even
21.	Square Dissection	odd to even
43.	Number Placement	presorting
46.	Tricolor Arrangement	
64.	Creating Octagons	presorting
116.	Bye Counting	
132.	The Solitaire Army	

Representation Change

tut	*Anagram Detection*	"signatures"
tut	*Cash Envelopes*	binary
tut	*Two Jealous Husbands*	state-space graph
tut	*Guarini's Puzzle*	graph, graph unfolding
9.	Mental Arithmetic	addend rearrangement
11.	A Stack of Fake Coins	stacks by coin numbers
25.	The Best Time to Be Alive	intervals on the real line
34.	Coins on a Star	graph unfolding
40.	Four Alternating Knights	graph, graph unfolding
49.	Missionaries and Cannibals	state-space graph
51.	Missing Number	two sums' difference, last two digits
68.	Digit Sum	addend rearrangement
77.	Searching for a Pattern	binary

106.	Turning on a Light Bulb	bit string
115.	Bachet's Weights	binary & ternary
118.	Six Knights	graph, graph unfolding
120.	Penny Distribution Machine	binary
125.	Sorting 5 in 7	points on the real line
130.	Poisoned Wine	binary
135.	Different Pairings	table, points on a circumference
136.	Catching a Spy	integer pairs
146.	Hitting a Moving Target	diagram
148.	One Coin for Freedom	binary

Problem Reduction

tut	*Optimal Pie Cutting*	to a math problem
74.	Site Selection	to a math problem
89.	Counter Exchange	to 1D Toads and Frogs
92.	Trapezoid Tiling	to a whole triangle
102.	The Monkey and the Coconuts	to a math problem
109.	Double-*n* Dominoes	to the Euler circuit problem
111.	Inverting a Coin Triangle	to a math problem

OTHER

1. A Wolf, a Goat, and a Cabbage
7. Bridge Crossing at Night
12. Questionable Tiling
14. Chessboard Reassembly
27. The Icosian Game
36. Limited Diversity
38. Tetromino Tiling
41. The Circle of Lights
44. Lighter or Heavier?
58. Sorting Once, Sorting Twice
60. Squaring a Coin Triangle
75. Gas Station Inspections
88. Toads and Frogs
123. Dutch National Flag Problem
137. Jumping into Pairs II
140. The *n*-Queens Problem Revisited
142. Twelve Coins
147. Hats with Numbers

Index of Terms and Names

Aanderaa, S. O., 150
Alcuin of York, 12, 82, 121
algorithm
  nonrecursive, 24
  recursive. *See also* recursive algorithm
algorithm analysis
  index of applications, 247–249
  techniques of, 22–31
algorithm design
  index of applications, 249–253
  strategies of, 3–22
algorithm efficiency
  class of. *See* rate of growth
  optimal, 9, 27, 30, 86–87, 88–89, 105, 125, 129, 153, 156, 157, 183–186, 199–202, 219
  worst-case analysis of, 52, 59, 83, 125, 150, 154, 165, 167, 179, 198, 202, 226
anagram, 11
Andreescu T., 199
Averbach, B., 89, 207

Bachet, Claude Gaspar, 106, 190
Backhouse, Roland, 87, 164
backtracking, 5–8
  index of applications, 249
backward substitutions, method of, 138, 150, 151, 176, 177, 197
backward thinking, 197, 219
Ball, W. W. Rouse, 106, 139, 159, 160, 182, 207, 225
Beasley, John D., 193, 212, 213
Beckwith, D., 188
Bell, J., 223
Bellman, Richard, 20
Bernhardsson, B., 223
binary numbers, 11–12, 146–147, 189, 191–192, 197, 209, 225, 234–236
binary search, 209
Boardman, J. M., 212
Boerner, Hermann, 128
Bogomolny, Alexander, 110, 154, 159, 160
Bollobás, Béla, 227
Brandt, Jørgen, 240
breadth-first search, 110
Bridge and Torch problem, 87
Brook, Maxey, 185
Bulgarian solitaire, 70–71
buttons and strings method, 14, 108

card puzzles, 39, 46, 55
ceiling (mathematics), 9
Chapman, Noyes, 231
Chein, O., 89, 207
chessboard problems, 23, 28, 35, 41, 57, 60, 67, 68
  bishop, 37
  king, 16, 36, 37
  knight, 14, 36, 37, 42, 43, 57, 62, 64
  prince, 51
  queen, 6, 67
  rook, 37, 51
Chinese Rings, 179, 180, 207
Christen, C., 125
Chu, I-Ping, 196
Church, Alonzo, 129
coin problems, 52, 60, 69. *See also* weighing problems
coloring (invariant), 28, 248
combination (mathematics), 8, 22
convex hull (mathematics), 135
Conway, John Horton, 170, 212, 213, 215
Cook, E. E., 87
Coxeter, H.S.M., 106, 139, 159, 160, 207, 225
cutting problems, 15, 30, 32, 37, 39, 45, 63

De Moivre, Abraham, 205
dead end, in a state-space tree, 6, 7
decimal numbers, 11, 36, 146
decision tree, 89, 225–226

decrease-and-conquer, 8–9
  index of applications
    bottom-up, 249–250
    by another constant, 250
    by constant factor, 250
    by one, 249–250
    by two, 250
    by variable number, 250
decrease-by-constant-factor, 9, 24, 250
decrease-by-one, 8–9, 249–250
degree, of a vertex, 29, 101, 102, 182, 222
Delannoy, Henri, 211
Demuth, H. B., 203
digraph. *See* graph, directed
Dijkstra, Edsger W., 199
Dirac's theorem, 221
divide-and-conquer, 9–10
  index of applications, 251
domino. *See* tiling, domino
double-*n* domino, 59
Dudeney, Henry E., 14, 30, 85, 86, 112, 145, 171, 188, 195, 208
dynamic programming, 20–22
  index of applications, 251

edge, of a graph, 12
Engel, Arthur, 162
enumeration method, 217–218
Eperson, D. B., 186
Eppstein D., 193
Euler circuit, 29, 30, 101, 102, 182
Euler path, 30, 102
Euler, Leonhard, 29, 30, 205
exhaustive search, 4–5, 7–8
  index of applications, 251
exponential rate of growth, 5, 24, 27

factorial, 4
Feijen, W. H., 199
Fibonacci numbers, 46, 127, 214
Fibonacci, Leonardo, 82, 127, 190
Fifteen puzzle, 68
floor (mathematics), 9
Fomin, D. B., 151
Frame-Stewart algorithm, 208
Frobenius coin problem, 121

Game of Life, 65–66, 215
Gardiner, A., 124
Gardner, Martin, 18, 82, 86, 89, 106, 112, 115, 139, 175, 179, 181, 185, 191, 209, 215, 229, 240
Gates, Bill, 154
Gauss, Carl Friedrich, 22, 82, 88, 205
Gibson, R., 175
Gik, E., 97, 146, 172
Ginat, D., 223
Golomb, Solomon W., 114, 148, 165, 187
Gomory barriers, 186
Gomory, Ralph E., 186
Grabarchuk, Serhiy, 92, 131, 195
graph, 12–15. *See also* Dirac's theorem; Euler circuit; Euler path; Hamilton circuit; Hamilton path; multigraph
  applications of, 252–253
  directed, 12
  edge of, 12
  state-space, 12–14
  undirected, 12
  unfolding of, 14–15
  vertex of, 12
Gray code, 179, 207
greedy approach, 15–17
  index of applications, 251–252
Greenes, C. E., 207
Gros, Lois, 180, 207
Guarini's puzzle, 14–15, 108, 115, 195
Guarini, Paolo, 14

Hamilton circuit, 100, 205, 222
Hamilton path, 114. *See also* path problems, Hamilton
Hamilton, Sir William, 38
Handshaking Lemma, 141
Hess, Dick, 118
heuristic, 205
Hofstadter, Douglas, 178
Hurkens, C.A.J., 157
Hwang, F., 125

Iba, G., 221
Icosian Game, 38
incremental approach, 9, 84, 140, 141
initial condition, of recurrence relation, 26
instance simplification, 11, 252
instance, of a puzzle, 3
interview question, 17, 87, 88, 89, 122, 166, 209, 233
invariant, 28–31
  index of applications, 248–249
    coloring, 248
    other, 248–249
    parity, 248

inversion, 172, 230
iterative improvement, 17–20
  index of applications, 252

Johnson-Trotter algorithm, 160
Johnsonbaugh, R., 196
Josephus problem, 67
Josephus, Flavius, 225

Khodulev, A., 238
King, K. N., 150
Kinsey, Ernest U., 231
Klamkin, M. S., 188
Knott, Ron, 127
Knuth, Donald E., 128, 179, 203
Konhauser, Joseph, 198
Kontsevich, M., 238
Kordemsky, B. A., 138, 140, 141
Kraitchik, Maurice, 106, 117, 217
Kurlandchik, L. D., 151
Königsberg Bridges problem, 29–30, 101

Latin square, 98
leaf, of a tree, 5
Lehmer's motel problem, 161
Lehmer, D. H., 161
Leino, K.R.M., 219
Levitin, A., 199
Levmore, S. X., 87
linear rate of growth, 24, 67
liquid pouring problems, 40, 48
logarithmic rate of growth, 24
Loyd, Sam, 30, 111, 145, 171, 188, 195, 229, 231
Lucas, Stephen, 218
Lucas, Édouard, 27, 159, 160, 208

magic square, 4–5, 39, 40, 103–104
magic sum, 103
Manber, Udi, 150
Manhattan distance, 50, 118, 143
Mastermind (game), 136
mathematical induction, 94, 111, 124, 126, 145, 153, 162, 164, 170, 172, 176, 177, 192, 196, 201, 206, 211, 224, 228
McNugget number, 44
median, 143, 144
Michalewicz, M., 138
Michalewicz, Z., 138
Microsoft, 17, 87
Milligan, W. Lloyd, 219
monomino, 51
monovariant, 19–20, 137, 151, 221, 227, 232
Montmort, Pierre Rémond, 205
Moore, C., 193
Moser, Leo, 87
multigraph, 29–30

n-queens problem, 6–8, 67, 111, 222–223
Niederman, D., 157
nonpromising node, 6
nonrecursive algorithm, 24
  analysis of, 24–25
NP-complete, 134, 231

O'Beirne, T. H., 89, 226
octagon, 47, 134–135, 172

pagoda function. *See* resource count
palindrome, 56, 158, 171
Parberry, Ian, 92
parity (invariant), 28, 248
partition problem, 134
Pascal's triangle, 22, 171
path problems
  dot crossing, 61
  Euler, 30, 38
  Hamilton, 36, 38, 42, 43, 64
  optimal, 36, 47, 50, 51, 59
  shortest path counting, 20, 34
Pauls, E., 223
permutation, 5, 160–161, 229
  inversion in, 230–231
  parity of, 230–231
  ranking of, 100
Peterson, Ivar, 110
Petković M., 207
Poundstone, William, 12, 82, 87, 129
presorting, 11
  index of applictions, 252
problem reduction, 11, 253
Project Euler, 95
Propp, James, 197
pseudo-magic square, 40
Pólya, George, 31, 82

quadratic rate of growth, 24
queue, 109–110
quickhull, 135
quicksort, 97

Rand, Michael, 208
rate of growth, 24
  exponential, 5, 24, 27
  linear, 24, 67
  logarithmic, 24
  quadratic, 24
recurrence relation, 26, 125, 127, 137–138, 150, 151, 154, 176, 179, 191, 197, 198, 207, 211, 224
  second-order linear, 206–207
recursion, 8
recursive algorithm, 8, 10, 26, 97, 101, 121, 151, 160, 163, 164, 168–169, 178, 179, 182, 195–196, 206, 207, 209–211, 215, 227–228
  analysis of, 25–27
representation change, 11
  index of applications, 252–253
resource count, 212–214, 236–238
river-crossing problems, 12–14, 17, 32, 33, 44
Robertson, Jack, 204
root, of a tree, 5
Rosen, Kenneth, 178
Rosenbaum, J., 179
Rote, Günter, 87
Roth, Ted, 208
Rubik's Cube, 14

Savchev, S., 199
Savin, A. P., 215
Schweik (literary hero), 46, 126
Scorer, R. S., 153
selection problem, 144
Shasha, Dennis, 136, 209
Shashi, 23
Sillke, Torsten, 87
simplex method, 20
Singmaster, David, 93, 121, 125, 139, 159, 160, 175, 202, 205, 207, 208, 211, 225
Slocum, J., 231
Smith-Thomas, B., 150
Sniedovich, Moshe, 87, 198
Sonneveld D., 231
sorting, 46, 52, 56, 63, 64, 126, 144. *See also* presorting
  by bubble sort, 171
  by insertion sort, 171
  by quicksort, 97
Spivak, A., 85, 111, 114, 119, 169
state-space graph, 12–14, 82, 110, 121
state-space tree, 5–7. *See also* backtracking
  pruned, 6
Steinhaus, Hugo, 160, 181, 204
Stevens, B., 223
Stewart, Ian, 204

Tabari, Hasib, 190
Tait, Peter Guthrie, 65, 102, 211
Tanton, James, 125, 188, 214, 221
Tarry, Gaston, 182
telescopic series, 126
ternary numbers, 11, 12, 189, 226
  balanced, 190
tetromino. *See* tiling, tetromino
tiling
  domino, 28, 34, 54, 60, 228–229
  tetromino, 41–42
  trapezoid, 54–55
  tromino
    right, 10, 35–36, 55, 62
    straight, 51
Topswops, game of, 55–56
tournament
  round-robin, 37, 217
  single-elimination, 40, 61
  tree, 106
Tower of Brahma, 27
Tower of Hanoi, 25–27, 153, 179
  extension of (Reve's puzzle), 64, 207–208
  restricted, 27, 52
transform-and-conquer, 11–15
  index of applications, 252–253
tree, 5
  decision, 89, 225–226
  leaf of, 5
  root of, 5
  state-space, 5
triangular numbers, 167, 176
Trigg, Charles, 157, 186, 188
tromino. *See* tiling, tromino
Turing, Alan, 37–38
Tweedie, M.C.K., 110

vertex, of a graph, 12

Warnsdorff, H. C., 205
Webb, William, 204
Wegman, Mark, 176
weighing problems, 34, 43, 45, 55, 61, 64, 67
Winkler, Peter, 128, 198, 227