

Contents

| | |
|--|------------------|
| Preface | <i>page</i> xiii |
| Introduction | 1 |
| Part I Iterative Algorithms and Loop Invariants | |
| 1 Iterative Algorithms: Measures of Progress and Loop Invariants | 5 |
| 1.1 A Paradigm Shift: A Sequence of Actions vs. a Sequence of Assertions | 5 |
| 1.2 The Steps to Develop an Iterative Algorithm | 9 |
| 1.3 More about the Steps | 13 |
| 1.4 Different Types of Iterative Algorithms | 21 |
| 1.5 Code from Loop Invariants | 28 |
| 1.6 Typical Errors | 31 |
| 1.7 Exercises | 32 |
| 2 Examples Using More-of-the-Input Loop Invariants | 33 |
| 2.1 Coloring the Plane | 33 |
| 2.2 Deterministic Finite Automaton | 35 |
| 2.3 More of the Input vs. More of the Output | 42 |
| 3 Abstract Data Types | 47 |
| 3.1 Specifications and Hints at Implementations | 47 |
| 3.2 Link List Implementation | 55 |
| 3.3 Merging with a Queue | 61 |
| 3.4 Parsing with a Stack | 62 |
| 4 Narrowing the Search Space: Binary Search | 64 |
| 4.1 Binary Search Trees | 64 |
| 4.2 Magic Sevens | 66 |
| 4.3 VLSI Chip Testing | 68 |
| 4.4 Exercises | 72 |
| 5 Iterative Sorting Algorithms | 74 |
| 5.1 Bucket Sort by Hand | 74 |

| | | |
|-----------|---|-----|
| viii | Contents | |
| | 5.2 Counting Sort (a Stable Sort) | 75 |
| | 5.3 Radix Sort | 78 |
| 6 | More Iterative Algorithms | 80 |
| | 6.1 Euclid's GCD Algorithm | 80 |
| | 6.2 Multiplying | 84 |
| 7 | The Loop Invariant for Lower Bounds | 88 |
| 8 | Key Concepts Summary: Loop Invariants and Iterative Algorithms | 97 |
| | 8.1 Loop Invariants and Iterative Algorithms | 97 |
| | 8.2 System Invariants | 99 |
| 9 | Additional Exercises: Part I | 102 |
| 10 | Partial Solutions to Additional Exercises: Part I | 124 |
| | Part II Recursion | |
| 11 | Abstractions, Techniques, and Theory | 133 |
| | 11.1 Thinking about Recursion | 133 |
| | 11.2 Looking Forward vs. Backward | 134 |
| | 11.3 With a Little Help from Your Friends | 135 |
| | 11.4 The Towers of Hanoi | 138 |
| | 11.5 Checklist for Recursive Algorithms | 139 |
| | 11.6 The Stack Frame | 144 |
| | 11.7 Proving Correctness with Strong Induction | 146 |
| 12 | Some Simple Examples of Recursive Algorithms | 149 |
| | 12.1 Sorting and Selecting Algorithms | 149 |
| | 12.2 Operations on Integers | 157 |
| | 12.3 Ackermann's Function | 162 |
| | 12.4 Fast Fourier Transformations | 163 |
| | 12.5 Exercise | 168 |
| 13 | Recursion on Trees | 169 |
| | 13.1 Tree Traversals | 174 |
| | 13.2 Simple Examples | 177 |
| | 13.3 Heap Sort and Priority Queues | 180 |
| | 13.4 Representing Expressions with Trees | 187 |
| 14 | Recursive Images | 192 |
| | 14.1 Drawing a Recursive Image from a Fixed Recursive and a Base Case Image | 192 |
| | 14.2 Randomly Generating a Maze | 195 |

| | Contents | ix |
|--|----------|-----|
| 15 Parsing with Context-Free Grammars | | 198 |
| 16 Key Concepts Summary: Recursion | | 208 |
| 17 Additional Exercises: Part II | | 211 |
| 18 Partial Solutions to Additional Exercises: Part II | | 230 |
| Part III Optimization Problems | | |
| 19 Definition of Optimization Problems | | 241 |
| 20 Graph Search Algorithms | | 243 |
| 20.1 A Generic Search Algorithm | | 243 |
| 20.2 Breadth-First Search for Shortest Paths | | 248 |
| 20.3 Dijkstra's Shortest-Weighted-Path Algorithm | | 253 |
| 20.4 Depth-First Search | | 259 |
| 20.5 Recursive Depth-First Search | | 263 |
| 20.6 Linear Ordering of a Partial Order | | 264 |
| 20.7 Exercise | | 267 |
| 21 Network Flows and Linear Programming | | 268 |
| 21.1 A Hill-Climbing Algorithm with a Small Local Maximum | | 270 |
| 21.2 The Primal–Dual Hill-Climbing Method | | 276 |
| 21.3 The Steepest-Ascent Hill-Climbing Algorithm | | 284 |
| 21.4 Linear Programming | | 288 |
| 21.5 Exercises | | 293 |
| 22 Greedy Algorithms | | 294 |
| 22.1 Abstractions, Techniques, and Theory | | 294 |
| 22.2 Examples of Greedy Algorithms | | 307 |
| 22.3 Exercises | | 320 |
| 23 Recursive Backtracking | | 321 |
| 23.1 Recursive Backtracking Algorithms | | 321 |
| 23.2 The Steps in Developing a Recursive Backtracking | | 325 |
| 23.3 Pruning Branches | | 329 |
| 23.4 Satisfiability | | 331 |
| 23.5 Exercises | | 334 |
| 24 Dynamic Programming Algorithms | | 336 |
| 24.1 Start by Developing a Recursive Backtracking Algorithm | | 336 |
| 24.2 The Steps in Developing a Dynamic Programming Algorithm | | 340 |
| 24.3 Subtle Points | | 346 |

| | | |
|---|--|------------|
| x | Contents | |
| | | <hr/> |
| | 24.4 The Longest-Common-Subsequence Problem | 364 |
| | 24.5 Dynamic Programs as More-of-the-Input Iterative Loop Invariant Algorithms | 368 |
| | 24.6 A Greedy Dynamic Program: The Weighted Job/Event Scheduling Problem | 371 |
| | 25 Designing Dynamic Programming Algorithms via Reductions | 375 |
| | 26 The Game of Life | 380 |
| | 26.1 Graph G from Computation | 380 |
| | 26.2 The Graph of Life | 382 |
| | 26.3 Examples of the Graph of Life | 385 |
| | 27 Solution Is a Tree | 390 |
| | 27.1 The Solution Viewed as a Tree: Chains of Matrix Multiplications | 390 |
| | 27.2 Generalizing the Problem Solved: Best AVL Tree | 395 |
| | 27.3 All Pairs Using Matrix Multiplication | 397 |
| | 27.4 Parsing with Context-Free Grammars | 398 |
| | 28 Reductions and NP-Completeness | 402 |
| | 28.1 Satisfiability Is at Least as Hard as Any Optimization Problem | 404 |
| | 28.2 Steps to Prove NP-Completeness | 407 |
| | 28.3 Example: 3-Coloring Is NP-Complete | 415 |
| | 28.4 An Algorithm for Bipartite Matching Using the Network Flow Algorithm | 419 |
| | 29 Randomized Algorithms | 423 |
| | 29.1 Using Randomness to Hide the Worst Cases | 423 |
| | 29.2 Solutions of Optimization Problems with a Random Structure | 427 |
| | 30 Machine Learning | 431 |
| | 31 Key Concepts Summary: Greedy Algorithms and Dynamic Programming | 439 |
| | 31.1 Greedy Algorithms | 439 |
| | 31.2 Dynamic Programming | 444 |
| | 32 Additional Exercises: Part III | 454 |
| | 32.1 Graph Algorithms | 454 |
| | 32.2 Greedy Algorithms | 457 |
| | 32.3 Dynamic Programming | 465 |
| | 32.4 Reductions and NP-Completeness | 476 |
| | 33 Partial Solutions to Additional Exercises: Part III | 482 |
| | 33.1 Graph Algorithms | 482 |

| | Contents | xi |
|--|-----------------|-----|
| 33.2 Greedy Algorithms | | 482 |
| 33.3 Dynamic Programming | | 485 |
| 33.4 Reductions and NP-Completeness | | 492 |
| Part IV Additional Topics | | |
| 34 Existential and Universal Quantifiers | | 499 |
| 35 Time Complexity | | 508 |
| 35.1 The Time (and Space) Complexity of an Algorithm | | 508 |
| 35.2 The Time Complexity of a Computational Problem | | 513 |
| 36 Logarithms and Exponentials | | 515 |
| 37 Asymptotic Growth | | 518 |
| 37.1 Steps to Classify a Function | | 519 |
| 37.2 More about Asymptotic Notation | | 525 |
| 38 Adding-Made-Easy Approximations | | 529 |
| 38.1 The Technique | | 530 |
| 38.2 Some Proofs for the Adding-Made-Easy Technique | | 534 |
| 39 Recurrence Relations | | 540 |
| 39.1 The Technique | | 540 |
| 39.2 Some Proofs | | 543 |
| 40 A Formal Proof of Correctness | | 549 |
| 41 Additional Exercises: Part IV | | 551 |
| 41.1 Existential and Universal Quantifiers | | 551 |
| 41.2 Time Complexity | | 553 |
| 41.3 Asymptotic Growth | | 554 |
| 41.4 Adding Made-Easy Approximations | | 554 |
| 42 Partial Solutions to Additional Exercises: Part IV | | 556 |
| 42.1 Existential and Universal Quantifiers | | 556 |
| 42.2 Time Complexity | | 560 |
| Exercise Solutions | | 561 |
| Conclusion | | 588 |
| Index | | 589 |