

# CONTENTS IN DETAIL

<b>ACKNOWLEDGMENTS</b>	<b>xix</b>
------------------------	------------

<b>INTRODUCTION</b>	<b>xxi</b>
---------------------	------------

Who Is This Book For? . . . . .	xxii
Analogies and Examples . . . . .	xxii
Language and Coding Conventions . . . . .	xxii
Terminology and Definitions . . . . .	xxiii
How to Use This Book . . . . .	xxiv

## **PART I: GRAPH BASICS** **1**

<b>1</b>	
<b>REPRESENTING GRAPHS</b>	<b>3</b>

Graph Structure . . . . .	4
Weighted Edges . . . . .	5
Directed Edges . . . . .	5
Edges with Both Weight and Direction . . . . .	7
The Adjacency List Representation . . . . .	7
Edges . . . . .	9
Nodes . . . . .	10
The Graph Class . . . . .	12
The Adjacency Matrix Representation . . . . .	15
Why This Matters . . . . .	18

<b>2</b>	
<b>NEIGHBORS AND NEIGHBORHOODS</b>	<b>19</b>

Neighbors in Undirected Graphs . . . . .	20
Neighbors in Directed Graphs . . . . .	21
Self-Loops . . . . .	22
Degree . . . . .	22
Clustering Coefficient . . . . .	24
Computing the Average Clustering Coefficient . . . . .	25
Handling Limitations . . . . .	26
Generating Neighborhood Subgraphs . . . . .	26
The Code . . . . .	27
An Example . . . . .	28
Why This Matters . . . . .	30

<b>3</b>	<b>PATHS THROUGH GRAPHS</b>	<b>31</b>
	Paths . . . . .	32
	Path Representation . . . . .	33
	Lists of Nodes . . . . .	33
	Lists of Edges . . . . .	34
	Lists of Previous Nodes . . . . .	36
	Calculating Path Cost . . . . .	39
	Reachability . . . . .	41
	Why This Matters . . . . .	42

## **PART II: SEARCH AND SHORTEST PATHS** **43**

<b>4</b>	<b>DEPTH-FIRST SEARCH</b>	<b>45</b>
	Use Cases . . . . .	46
	Exploring a Hedge Maze . . . . .	46
	Learning a New Subject . . . . .	47
	Checking Reachability . . . . .	48
	Recursive Depth-First Search . . . . .	48
	The Code . . . . .	48
	An Example . . . . .	50
	Depth-First Search with a Stack . . . . .	52
	The Code . . . . .	52
	An Example . . . . .	53
	Finding Connected Components . . . . .	55
	The Code . . . . .	55
	An Example . . . . .	56
	Depth-First Search Trees and Forests . . . . .	57
	Iterative Deepening . . . . .	59
	Why This Matters . . . . .	61

<b>5</b>	<b>BREADTH-FIRST SEARCH</b>	<b>63</b>
	Use Cases . . . . .	64
	Learning New Topics . . . . .	64
	Exploring a New City . . . . .	64
	The Breadth-First Search Algorithm . . . . .	65
	The Code . . . . .	65
	An Example . . . . .	66
	Finding Shortest Paths . . . . .	68
	Simple Path Planning . . . . .	70
	Constructing a Graph from a Grid . . . . .	70
	Adding Obstacles . . . . .	71
	Running Breadth-First Search . . . . .	73
	Why This Matters . . . . .	73

<b>6</b>		
<b>SOLVING PUZZLES</b>		<b>75</b>
State Spaces and Graphs . . . . .		76
The Tower of Hanoi . . . . .		76
River-Crossing Puzzles . . . . .		78
Slider Puzzles . . . . .		79
Constructing a Graph with Search . . . . .		80
Representing the Puzzle's States . . . . .		81
Generating the Graph . . . . .		83
Solving a Puzzle with Search . . . . .		85
Why This Matters . . . . .		88
<b>7</b>		
<b>SHORTEST PATHS</b>		<b>89</b>
Lowest-Cost Paths . . . . .		90
Dijkstra's Algorithm . . . . .		91
The Code . . . . .		92
An Example . . . . .		94
Disconnected Graphs . . . . .		95
Negative Edge Weights . . . . .		96
Bellman-Ford Algorithm . . . . .		97
The Code . . . . .		99
An Example . . . . .		100
All-Pairs Shortest Paths . . . . .		102
The Floyd-Warshall Algorithm . . . . .		103
The Code . . . . .		105
An Example . . . . .		106
Computing Graph Diameter . . . . .		108
Why This Matters . . . . .		110
<b>8</b>		
<b>HEURISTIC-GUIDED SEARCHES</b>		<b>111</b>
Choosing Appropriate Heuristics . . . . .		112
Euclidean Distance . . . . .		112
Admissible Heuristics . . . . .		113
Heuristic Design Challenges . . . . .		113
Greedy Best-First Search . . . . .		114
The Code . . . . .		115
An Example . . . . .		116
A* Search . . . . .		119
The Code . . . . .		120
An Example . . . . .		121
Why A* Finds the Optimal Path . . . . .		124
Applying A* to Puzzles . . . . .		124
Searching Unknown Graphs . . . . .		126
The Code . . . . .		127
An Example . . . . .		130
Why This Matters . . . . .		132

## PART III: CONNECTIVITY AND ORDERING 133

### 9 TOPOLOGICAL SORT 135

How Topological Sort Algorithms Work . . . . .	136
Use Cases . . . . .	137
Code Dependencies . . . . .	137
Task Lists . . . . .	138
Teaching and Learning . . . . .	139
Kahn's Algorithm . . . . .	139
The Code . . . . .	140
An Example . . . . .	141
Depth-First Search . . . . .	143
The Code . . . . .	143
An Example . . . . .	144
Order of Starting Nodes . . . . .	146
Detecting Cycles . . . . .	147
Reordering Lists . . . . .	148
Why This Matters . . . . .	150

### 10 MINIMUM SPANNING TREES 153

The Structure of Minimum Spanning Trees . . . . .	154
Use Cases . . . . .	154
Physical Networks . . . . .	155
Social Networks . . . . .	156
Prim's Algorithm . . . . .	156
The Code . . . . .	157
An Example . . . . .	159
Kruskal's Algorithm . . . . .	161
Union-Find . . . . .	161
The Code . . . . .	162
An Example . . . . .	163
Maze Generation . . . . .	165
Representing Grid-Based Mazes . . . . .	165
Generating Mazes . . . . .	165
The Code . . . . .	167
An Example . . . . .	167
Single-Linkage Hierarchical Clustering . . . . .	169
The Code . . . . .	170
An Example . . . . .	172
Why This Matters . . . . .	173

### 11 BRIDGES AND ARTICULATION POINTS 175

Defining Bridges and Articulation Points . . . . .	176
Use Cases . . . . .	177
Designing Resilient Networks . . . . .	178
Preventing the Spread of Diseases . . . . .	179
Designing Magical Labyrinths . . . . .	180

A Bridge-Finding Algorithm . . . . .	180
The Code . . . . .	182
An Example . . . . .	184
An Algorithm for Finding Articulation Points . . . . .	186
The Code . . . . .	188
An Example . . . . .	189
Why This Matters . . . . .	191

**12**  
**STRONGLY CONNECTED COMPONENTS** **193**

Defining Strongly Connected Components . . . . .	194
Determining Which Nodes Are Mutually Reachable . . . . .	195
Determining Whether Nodes Are Strongly Connected . . . . .	196
Use Cases . . . . .	196
Modeling Computer Program States . . . . .	197
Understanding a Gossip Network . . . . .	197
Planning a Travel Network . . . . .	198
Kosaraju-Sharir’s Algorithm . . . . .	199
Transposed Graphs . . . . .	200
The Code . . . . .	201
An Example . . . . .	202
Why This Matters . . . . .	204

**13**  
**RANDOM WALKS** **207**

Introducing Random Walks . . . . .	208
Probabilities in Random Walks . . . . .	208
Random Walks as Markov Chains . . . . .	210
Transition Probabilities . . . . .	211
Matrix Formulation . . . . .	211
Use Cases . . . . .	213
Information Chains in Social Networks . . . . .	213
Exploration . . . . .	214
Games of Chance . . . . .	214
Simulating Random Walks . . . . .	215
Statistical Measures . . . . .	216
Hitting and Absorption Time . . . . .	217
Stationary Distribution . . . . .	218
Luck-Based Board Games . . . . .	219
Transition Probabilities . . . . .	221
Maximum Likelihood Estimations . . . . .	221
A Transition Matrix Estimation Algorithm . . . . .	222
Limitations of Working with Finite Data . . . . .	224
Random Starting Nodes . . . . .	224
Choosing a Random Starting Node . . . . .	224
Estimating the Probability Distribution for Starting Nodes . . . . .	225
Why This Matters . . . . .	226

# PART IV: MAX FLOW AND BIPARTITE MATCHING

227

## 14

### MAX-FLOW ALGORITHMS

229

The Maximum-Flow Problem . . . . .	230
Use Cases . . . . .	232
Physical Pipelines . . . . .	232
Transportation Networks . . . . .	232
Communication Networks . . . . .	233
Extending the Data Structures . . . . .	233
Edges with Capacity . . . . .	233
Residual Graphs . . . . .	235
The Ford-Fulkerson Algorithm . . . . .	237
Defining Augmenting Paths . . . . .	238
Finding an Augmenting Path . . . . .	240
Updating a Path's Capacity . . . . .	243
Putting It All Together . . . . .	244
An Example . . . . .	245
The Edmonds-Karp Algorithm . . . . .	246
The Code . . . . .	246
An Example . . . . .	249
Modeling Increasingly Complex Real-World Situations . . . . .	251
Multiple Sources . . . . .	251
Multiple Sinks . . . . .	252
Anti-parallel Edges . . . . .	253
Why This Matters . . . . .	254

## 15

### BIPARTITE GRAPH MATCHING

255

Matching . . . . .	256
Bipartite Graphs . . . . .	257
Bipartite Labeling . . . . .	258
The Code . . . . .	258
An Example . . . . .	260
Use Cases . . . . .	263
Scheduling Jobs . . . . .	263
Assigning Office Space . . . . .	263
Planning Quest Battles . . . . .	264
Exhaustive Algorithms . . . . .	264
Matching Data . . . . .	264
Exhaustive Scoring . . . . .	265
The Code . . . . .	266
An Example . . . . .	267
Solving the Maximum-Cardinality Bipartite Problem . . . . .	269
The Code . . . . .	270
An Example . . . . .	271
Why This Matters . . . . .	273

## PART V: HARD GRAPH PROBLEMS

275

### 16

#### GRAPH COLORING

277

The Graph-Coloring Problem . . . . .	278
Use Cases . . . . .	279
Coloring Maps . . . . .	279
Organizing Seating Arrangements . . . . .	280
Assigning Parking Spaces . . . . .	281
Planning Magical Labyrinths . . . . .	282
Graph-Coloring Algorithms . . . . .	282
Exhaustive Search . . . . .	283
Backtracking Search . . . . .	284
Greedy Search . . . . .	290
Node Removal . . . . .	293
Why This Matters . . . . .	297

### 17

#### CLIQUES, INDEPENDENT SETS, AND VERTEX COVERS

299

Backtracking Search for Sets of Nodes . . . . .	300
Cliques . . . . .	301
Use Cases . . . . .	303
Greedy Search . . . . .	303
Backtracking Search . . . . .	305
Independent Sets . . . . .	308
Use Cases . . . . .	309
Greedy Search . . . . .	309
Backtracking Search . . . . .	312
Vertex Cover . . . . .	315
Use Cases . . . . .	316
Greedy Search . . . . .	316
Backtracking Search . . . . .	318
Randomized Algorithms . . . . .	321
Basic Randomized Search . . . . .	321
Weighted Randomized Search . . . . .	322
Why This Matters . . . . .	323

### 18

#### TOURS THROUGH GRAPHS

325

Hamiltonian Paths and Cycles . . . . .	326
Validating Hamiltonian Paths . . . . .	327
Finding Hamiltonian Paths with Depth-First Search . . . . .	328
The Traveling Salesperson Problem . . . . .	331
Depth-First Search . . . . .	331
The Code . . . . .	332
An Example . . . . .	333
Eulerian Paths and Cycles . . . . .	334
Validating Eulerian Paths . . . . .	336
Finding Eulerian Cycles with Hierholzer's Algorithm . . . . .	337
Why This Matters . . . . .	341

**CONCLUSION 343**

**A  
CONSTRUCTING GRAPHS 345**

Constructing Graphs from Edges . . . . . 346  
    Inserting Edges from a List . . . . . 346  
    Loading Edge Lists from Files . . . . . 347  
    Saving Edge Lists to Files . . . . . 349  
Inserting Nodes by Name . . . . . 350  
Co-occurrences . . . . . 351  
Spatial Points . . . . . 353  
Preconditions . . . . . 355

**B  
MODIFIABLE PRIORITY QUEUES 357**

Heaps . . . . . 358  
    Heap Items . . . . . 358  
    Array-Based Storage . . . . . 359  
    Element Swaps . . . . . 359  
Modifiable Priority Queue . . . . . 361  
The Data Structure . . . . . 362  
    Defining Helper Functions . . . . . 363  
    Adding Items . . . . . 365  
    Removing Items . . . . . 366  
    Modifying Priorities . . . . . 367  
Peek Functions . . . . . 367

**C  
UNION-FIND 369**

The Union-Find Data Structure . . . . . 370  
UnionFind . . . . . 371  
    UnionFindNode . . . . . 371  
    UnionFind Class . . . . . 371

**INDEX 375**