

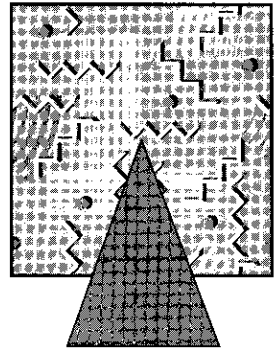
The book cover features a dark, textured background with several glowing spheres in shades of blue, teal, and purple. Thin, wavy orange lines swirl around the spheres, creating a sense of motion and depth. The title is centered in a white, sans-serif font.

Data Structures and Algorithms in Java™

A small logo is visible on a white rectangular element in the bottom left corner of the cover. The logo consists of a stylized, geometric shape that resembles a letter 'A' or a similar symbol, rendered in a light blue or grey color.

Adam Drozdek

Contents



1	OBJECT-ORIENTED PROGRAMMING USING JAVA	1
1.1	Rudimentary Java	1
1.1.1	Variable Declarations	1
1.1.2	Operators	4
1.1.3	Decision Statements	5
1.1.4	Loops	6
1.1.5	Exception Handling	6
1.2	Object-Oriented Programming in Java	8
1.2.1	Encapsulation	8
1.2.2	Abstract Data Types	14
1.2.3	Inheritance	15
1.2.4	Polymorphism	18
1.3	Input and Output	21
1.4	Java and Pointers	24
1.5	Vectors in <code>java.util</code>	28
1.6	Data Structures and Object-Oriented Programming	34
1.7	Case Study: Random Access File	35
1.8	Exercises	43
1.9	Programming Assignments	45

2	COMPLEXITY ANALYSIS	49
2.1	Computational and Asymptotic Complexity	49
2.2	Big-O Notation	50
2.3	Properties of Big-O Notation	53
2.4	Ω and Q Notations	54
2.5	Possible Problems	55
2.6	Examples of Complexities	56
2.7	Finding Asymptotic Complexity: Examples	57
2.8	The Best, Average, and Worst Cases	59
2.9	Amortized Complexity	62
2.10	Exercises	67
3	LINKED LISTS	71
3.1	Singly Linked Lists	71
3.1.1	Insertion	77
3.1.2	Deletion	79
3.1.3	Search	84
3.2	Doubly Linked Lists	86
3.3	Circular Lists	90
3.4	Skip Lists	92
3.5	Self-Organizing Lists	98
3.6	Sparse Tables	102
3.7	Linked Lists in <code>java.util</code>	106
3.8	Concluding Remarks	111
3.9	Case Study: A Library	112
3.10	Exercises	121
3.11	Programming Assignments	123
4	STACKS AND QUEUES	129
4.1	Stacks	129
4.1.1	Stacks in <code>java.util</code>	137
4.2	Queues	138
4.3	Priority Queues	147

- 4.4 Case Study: Exiting a Maze 148
- 4.5 Exercises 153
- 4.6 Programming Assignments 156

5

RECURSION**159**

- 5.1 Recursive Definitions 159
- 5.2 Method Calls and Recursion Implementation 162
- 5.3 Anatomy of a Recursive Call 164
- 5.4 Tail Recursion 168
- 5.5 Nontail Recursion 169
- 5.6 Indirect Recursion 174
- 5.7 Nested Recursion 176
- 5.8 Excessive Recursion 177
- 5.9 Backtracking 180
- 5.10 Concluding Remarks 187
- 5.11 Case Study: A Recursive Descent Interpreter 188
- 5.12 Exercises 196
- 5.13 Programming Assignments 199

6

BINARY TREES**203**

- 6.1 Trees, Binary Trees, and Binary Search Trees 203
- 6.2 Implementing Binary Trees 208
- 6.3 Searching a Binary Search Tree 210
- 6.4 Tree Traversal 212
 - 6.4.1 Breadth-First Traversal 213
 - 6.4.2 Depth-First Traversal 213
 - 6.4.3 Stackless Depth-First Traversal 221
- 6.5 Insertion 228
- 6.6 Deletion 231
 - 6.6.1 Deletion by Merging 232
 - 6.6.2 Deletion by Copying 235
- 6.7 Balancing a Tree 238
 - 6.7.1 The DSW Algorithm 241
 - 6.7.2 AVL Trees 243

6.8	Self-Adjusting Trees	249
6.8.1	Self-Restructuring Trees	250
6.8.2	Splaying	251
6.9	Heaps	256
6.9.1	Heaps as Priority Queues	258
6.9.2	Organizing Arrays as Heaps	261
6.10	Polish Notation and Expression Trees	265
6.10.1	Operations on Expression Trees	266
6.11	Case Study: Computing Word Frequencies	270
6.12	Exercises	276
6.13	Programming Assignments	280

7

MULTIWAY TREES

287

7.1	The Family of B-Trees	288
7.1.1	B-Trees	289
7.1.2	B*-Trees	300
7.1.3	B+-Trees	301
7.1.4	Prefix B+-Trees	304
7.1.5	Bit-Trees	305
7.1.6	R-Trees	308
7.1.7	2-4 Trees	311
7.1.8	Sets in java.util	319
7.1.9	Maps in java.util	325
7.2	Tries	328
7.3	Concluding Remarks	337
7.4	Case Study: Spell Checker	337
7.5	Exercises	348
7.6	Programming Assignments	349

8

GRAPHS

355

8.1	Graph Representation	357
8.2	Graph Traversals	357
8.3	Shortest Paths	362
8.3.1	All-to-All Shortest Path Problem	369
8.4	Cycle Detection	372
8.4.1	Union-Find Problem	372

8.5	Spanning Trees	375
8.5.1	Borůvka's Algorithm	376
8.5.2	Kruskal's Algorithm	377
8.5.3	Jarník-Prim's Algorithm	377
8.5.4	Dijkstra's Method	380
8.6	Connectivity	380
8.6.1	Connectivity in Undirected Graphs	381
8.6.2	Connectivity in Directed Graphs	384
8.7	Topological Sort	386
8.8	Networks	388
8.8.1	Maximum Flows	388
8.8.2	Maximum Flows of Minimum Cost	398
8.9	Matching	401
8.9.1	Assignment Problem	408
8.9.2	Matching in Nonbipartite Graphs	410
8.10	Eulerian and Hamiltonian Graphs	412
8.10.1	Eulerian Graphs	412
8.10.2	Hamiltonian Graphs	413
8.11	Case Study: Distinct Representatives	416
8.12	Exercises	425
8.13	Programming Assignments	428

9

SORTING**433**

9.1	Elementary Sorting Algorithms	434
9.1.1	Insertion Sort	434
9.1.2	Selection Sort	438
9.1.3	Bubble Sort	439
9.2	Decision Trees	441
9.3	Efficient Sorting Algorithms	445
9.3.1	Shell Sort	445
9.3.2	Heap Sort	448
9.3.3	Quicksort	452
9.3.4	Mergesort	458
9.3.5	Radix Sort	461
9.4	Sorting in <code>java.util</code>	466
9.5	Concluding Remarks	469
9.6	Case Study: Adding Polynomials	471

- 9.7 Exercises 478
- 9.8 Programming Assignments 480

10 HASHING

483

- 10.1 Hash Functions 484
 - 10.1.1 Division 484
 - 10.1.2 Folding 484
 - 10.1.3 Mid-Square Function 485
 - 10.1.4 Extraction 485
 - 10.1.5 Radix Transformation 486
- 10.2 Collision Resolution 486
 - 10.2.1 Open Addressing 487
 - 10.2.2 Chaining 492
 - 10.2.3 Bucket Addressing 495
- 10.3 Deletion 496
- 10.4 Perfect Hash Functions 497
 - 10.4.1 Cichelli's Method 498
 - 10.4.2 The FHCD Algorithm 500
- 10.5 Hash Functions for Extendible Files 503
 - 10.5.1 Extendible Hashing 503
 - 10.5.2 Linear Hashing 506
- 10.6 Hashing in `java.util` 509
- 10.7 Case Study: Hashing with Buckets 514
- 10.8 Exercises 522
- 10.9 Programming Assignments 523

11 DATA COMPRESSION

527

- 11.1 Conditions for Data Compression 527
- 11.2 Huffman Coding 529
 - 11.2.1 Adaptive Huffman Coding 539
- 11.3 Shannon-Fano Code 543
- 11.4 Run-Length Encoding 545
- 11.5 Ziv-Lempel Code 546
- 11.6 Case Study: Huffman Method with Run-Length Encoding 549
- 11.7 Exercises 560
- 11.8 Programming Assignments 561

12 MEMORY MANAGEMENT 565

12.1	The Sequential-Fit Methods	566
12.2	The Nonsequential-Fit Methods	567
12.2.1	Buddy Systems	569
12.3	Garbage Collection	576
12.3.1	Mark-and-Sweep	577
12.3.2	Copying Methods	584
12.3.3	Incremental Garbage Collection	586
12.4	Concluding Remarks	594
12.5	Case Study: An In-Place Garbage Collector	595
12.6	Exercises	603
12.7	Programming Assignments	604

APPENDIX

A	Computing Big-O	609
A.1	Harmonic Series	609
A.2	Approximation of the Function $\lg(n!)$	609
A.3	Big-O for Average Case of Quicksort	611
A.4	Average Path Length in a Random Binary Tree	613

Name Index 615

Subject Index 619