



INTERNATIONAL
EDITION

Data Structures and Algorithm Analysis in Java™

Third Edition

Mark Allen Weiss

ALWAYS LEARNING

PEARSON

This page intentionally left blank

Data Structures and Algorithm Analysis in Java

Table of Contents

Cover

Contents

Preface

Chapter 1 Introduction

1.1 What's the Book About?

1.2 Mathematics Review

1.2.1 Exponents

1.2.2 Logarithms

1.2.3 Series

1.2.4 Modular Arithmetic

1.2.5 The P Word

1.3 A Brief Introduction to Recursion

1.4 Implementing Generic Components Pre-Java 5

1.4.1 Using Object for Genericity

1.4.2 Wrappers for Primitive Types

1.4.3 Using Interface Types for Genericity

1.4.4 Compatibility of Array Types

1.5 Implementing Generic Components Using Java 5 Generics

1.5.1 Simple Generic Classes and Interfaces

1.5.2 Autoboxing/Unboxing

1.5.3 The Diamond Operator

1.5.4 Wildcards with Bounds

1.5.5 Generic Static Methods

1.5.6 Type Bounds

1.5.7 Type Erasure

1.5.8 Restrictions on Generics

1.6 Function Objects

Summary

Exercises

References

Chapter 2 Algorithm Analysis

2.1 Mathematical Background

2.2 Model

2.3 What to Analyze

2.4 Running Time Calculations

2.4.1 A Simple Example

2.4.2 General Rules

Table of Contents

2.4.3 Solutions for the Maximum Subsequence Sum Problem

2.4.4 Logarithms in the Running Time

2.4.5 A Grain of Salt

Summary

Exercises

References

Chapter 3 Lists, Stacks, and Queues

3.1 Abstract Data Types (ADTs)

3.2 The List ADT

3.2.1 Simple Array Implementation of Lists

3.2.2 Simple Linked Lists

3.3 Lists in the Java Collections API

3.3.1 Collection Interface

3.3.2 Iterators

3.3.3 The List Interface, ArrayList, and LinkedList

3.3.4 Example: Using remove on a LinkedList

3.3.5 ListIterators

3.4 Implementation of ArrayList

3.4.1 The Basic Class

3.4.2 The Iterator and Java Nested and Inner Classes

3.5 Implementation of LinkedList

3.6 The Stack ADT

3.6.1 Stack Model

3.6.2 Implementation of Stacks

3.6.3 Applications

3.7 The Queue ADT

3.7.1 Queue Model

3.7.2 Array Implementation of Queues

3.7.3 Applications of Queues

Summary

Exercises

Chapter 4 Trees

4.1 Preliminaries

4.1.1 Implementation of Trees

4.1.2 Tree Traversals with an Application

4.2 Binary Trees

4.2.1 Implementation

4.2.2 An Example: Expression Trees

4.3 The Search Tree ADT Binary Search Trees

Table of Contents

- 4.3.1 contains
- 4.3.2 findMin and findMax
- 4.3.3 insert
- 4.3.4 remove
- 4.3.5 Average-Case Analysis

4.4 AVL Trees

- 4.4.1 Single Rotation
- 4.4.2 Double Rotation

4.5 Splay Trees

- 4.5.1 A Simple Idea (That Does Not Work)
- 4.5.2 Splaying

4.6 Tree Traversals (Revisited)

4.7 B-Trees

4.8 Sets and Maps in the Standard Library

- 4.8.1 Sets
- 4.8.2 Maps
- 4.8.3 Implementation of TreeSet and TreeMap
- 4.8.4 An Example That Uses Several Maps

Summary

Exercises

References

Chapter 5 Hashing

5.1 General Idea

5.2 Hash Function

5.3 Separate Chaining

5.4 Hash Tables Without Linked Lists

- 5.4.1 Linear Probing
- 5.4.2 Quadratic Probing
- 5.4.3 Double Hashing

5.5 Rehashing

5.6 Hash Tables in the Standard Library

5.7 Hash Tables with Worst-Case $O(1)$ Access

- 5.7.1 Perfect Hashing
- 5.7.2 Cuckoo Hashing
- 5.7.3 Hopscotch Hashing

5.8 Universal Hashing

5.9 Extendible Hashing

Summary

Exercises

Table of Contents

References

Chapter 6 Priority Queues (Heaps)

6.1 Model

6.2 Simple Implementations

6.3 Binary Heap

6.3.1 Structure Property

6.3.2 Heap-Order Property

6.3.3 Basic Heap Operations

6.3.4 Other Heap Operations

6.4 Applications of Priority Queues

6.4.1 The Selection Problem

6.4.2 Event Simulation

6.5 d-Heaps

6.6 Leftist Heaps

6.6.1 Leftist Heap Property

6.6.2 Leftist Heap Operations

6.7 Skew Heaps

6.8 Binomial Queues

6.8.1 Binomial Queue Structure

6.8.2 Binomial Queue Operations

6.8.3 Implementation of Binomial Queues

6.9 Priority Queues in the Standard Library

Summary

Exercises

References

Chapter 7 Sorting

7.1 Preliminaries

7.2 Insertion Sort

7.2.1 The Algorithm

7.2.2 Analysis of Insertion Sort

7.3 A Lower Bound for Simple Sorting Algorithms

7.4 Shellsort

7.4.1 Worst-Case Analysis of Shellsort

7.5 Heapsort

7.5.1 Analysis of Heapsort

7.6 Mergesort

7.6.1 Analysis of Mergesort

7.7 Quicksort

7.7.1 Picking the Pivot

Table of Contents

- 7.7.2 Partitioning Strategy
- 7.7.3 Small Arrays
- 7.7.4 Actual Quicksort Routines
- 7.7.5 Analysis of Quicksort
- 7.7.6 A Linear-Expected-Time Algorithm for Selection

7.8 A General Lower Bound for Sorting

- 7.8.1 Decision Trees

7.9 Decision-Tree Lower Bounds for Selection Problems

7.10 Adversary Lower Bounds

7.11 Linear-Time Sorts: Bucket Sort and Radix Sort

7.12 External Sorting

- 7.12.1 Why We Need New Algorithms
- 7.12.2 Model for External Sorting
- 7.12.3 The Simple Algorithm
- 7.12.4 Multiway Merge
- 7.12.5 Polyphase Merge
- 7.12.6 Replacement Selection

Summary

Exercises

References

Chapter 8 The Disjoint Set Class

- 8.1 Equivalence Relations
- 8.2 The Dynamic Equivalence Problem
- 8.3 Basic Data Structure
- 8.4 Smart Union Algorithms
- 8.5 Path Compression
- 8.6 Worst Case for Union-by-Rank and Path Compression
 - 8.6.1 Slowly Growing Functions
 - 8.6.2 An Analysis By Recursive Decomposition
 - 8.6.3 An $O(M \log^* N)$ Bound
 - 8.6.4 An $O(M(M, N))$ Bound

8.7 An Application

Summary

Exercises

References

Chapter 9 Graph Algorithms

- 9.1 Definitions
 - 9.1.1 Representation of Graphs
- 9.2 Topological Sort

Table of Contents

9.3 Shortest-Path Algorithms

- 9.3.1 Unweighted Shortest Paths
- 9.3.2 Dijkstra's Algorithm
- 9.3.3 Graphs with Negative Edge Costs
- 9.3.4 Acyclic Graphs
- 9.3.5 All-Pairs Shortest Path
- 9.3.6 Shortest-Path Example

9.4 Network Flow Problems

- 9.4.1 A Simple Maximum-Flow Algorithm

9.5 Minimum Spanning Tree

- 9.5.1 Prim's Algorithm
- 9.5.2 Kruskal's Algorithm

9.6 Applications of Depth-First Search

- 9.6.1 Undirected Graphs
- 9.6.2 Biconnectivity
- 9.6.3 Euler Circuits
- 9.6.4 Directed Graphs
- 9.6.5 Finding Strong Components

9.7 Introduction to NP-Completeness

- 9.7.1 Easy vs. Hard
- 9.7.2 The Class NP
- 9.7.3 NP-Complete Problems

Summary

Exercises

References

Chapter 10 Algorithm Design Techniques

10.1 Greedy Algorithms

- 10.1.1 A Simple Scheduling Problem
- 10.1.2 Huffman Codes
- 10.1.3 Approximate Bin Packing

10.2 Divide and Conquer

- 10.2.1 Running Time of Divide-and-Conquer Algorithms
- 10.2.2 Closest-Points Problem
- 10.2.3 The Selection Problem
- 10.2.4 Theoretical Improvements for Arithmetic Problems

10.3 Dynamic Programming

- 10.3.1 Using a Table Instead of Recursion
- 10.3.2 Ordering Matrix Multiplications
- 10.3.3 Optimal Binary Search Tree
- 10.3.4 All-Pairs Shortest Path

Table of Contents

10.4 Randomized Algorithms

10.4.1 Random Number Generators

10.4.2 Skip Lists

10.4.3 Primality Testing

10.5 Backtracking Algorithms

10.5.1 The Turnpike Reconstruction Problem

10.5.2 Games

Summary

Exercises

References

Chapter 11 Amortized Analysis

11.1 An Unrelated Puzzle

11.2 Binomial Queues

11.3 Skew Heaps

11.4 Fibonacci Heaps

11.4.1 Cutting Nodes in Leftist Heaps

11.4.2 Lazy Merging for Binomial Queues

11.4.3 The Fibonacci Heap Operations

11.4.4 Proof of the Time Bound

11.5 Splay Trees

Summary

Exercises

References

Chapter 12 Advanced Data Structures and Implementation

12.1 Top-Down Splay Trees

12.2 Red-Black Trees

12.2.1 Bottom-Up Insertion

12.2.2 Top-Down Red-Black Trees

12.2.3 Top-Down Deletion

12.3 Treaps

12.4 Suffix Arrays and Suffix Trees

12.4.1 Suffix Arrays

12.4.2 Suffix Trees

12.4.3 Linear-Time Construction of Suffix Arrays and Suffix Trees

12.5 k-d Trees

12.6 Pairing Heaps

Summary

Exercises

References

Table of Contents

Index

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z