


★



ALGORITHMS, 4TH EDITION

1. Fundamentals

2. Sorting

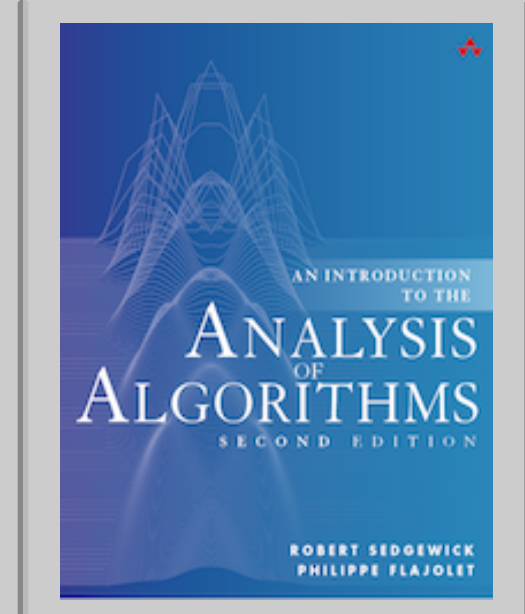
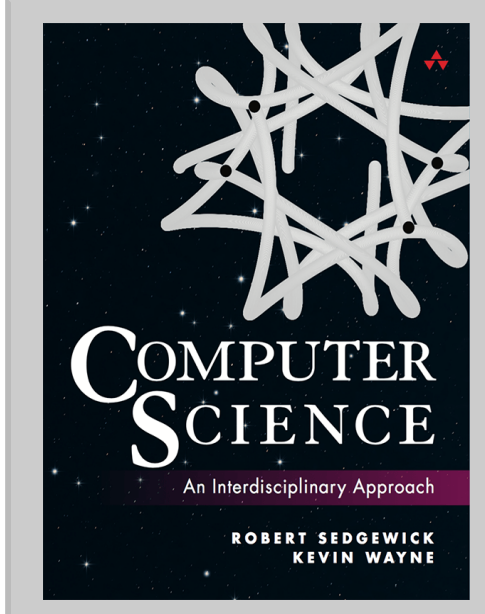
3. Searching

4. Graphs

5. Strings

6. Context

RELATED BOOKSITES



WEB RESOURCES

FAQ

Data

Code

Errata

Lectures

Cheatsheet

References

Online Course

Programming Assignments

ENHANCED BY










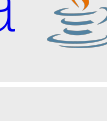






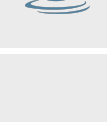
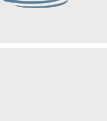

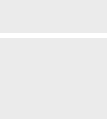
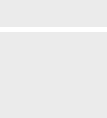

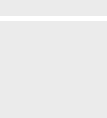
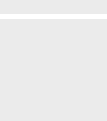



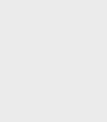

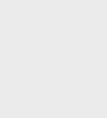




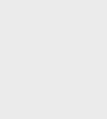

Google

## 1. FUNDAMENTALS

**Overview.** The objective of this book is to study a broad variety of important and useful *algorithms*—methods for solving problems that are suited for computer implementations. Algorithms go hand in hand with *data structures*—schemes for organizing data. This chapter introduces the basic tools that we need to study algorithms and data structures.

- [1.1 Programming Model](#) introduces our *basic programming model*. All of our programs are implemented using a small subset of the Java programming language plus a few of our own libraries for input and output.
- [1.2 Data Abstraction](#) emphasizes *data abstraction*, where we define *abstract data types* (ADTs). We specify an *applications programming interface* (API) and then use the Java class mechanism to develop an implementation for use in client code.
- [1.3 Bags, Queues, and Stacks](#) considers three fundamental ADTs: the *bag*, the *queue*, and the *stack*. We describe APIs and implementations using resizing arrays and linked lists.
- [1.4 Analysis of Algorithms](#) describes our approach to analyzing algorithm performance. The basis of our approach is the *scientific method*: we develop hypotheses about performance, create mathematical models, and run experiments to test them.
- [1.5 Case Study: Union-Find](#) is a case study where we consider solutions to a *connectivity* problem that uses algorithms and data structures that implement the classic *union-find* ADT.

**Java programs in this chapter.** Below is a list of Java programs in this chapter. Click on the program name to access the Java code; click on the reference number for a brief description; read the textbook for a full discussion.

REF	PROGRAM	DESCRIPTION / JAVADOC
-	<a href="#">BinarySearch.java</a> 	binary search
-	<a href="#">RandomSeq.java</a> 	random numbers in a given range
-	<a href="#">Average.java</a> 	average of a sequence of numbers
-	<a href="#">Cat.java</a> 	concatenate files
-	<a href="#">Knuth.java</a> 	Knuth shuffle
-	<a href="#">Counter.java</a> 	counter
-	<a href="#">StaticSETofInts.java</a> 	set of integers
-	<a href="#">Allowlist.java</a> 	allowlist client
-	<a href="#">Vector.java</a> 	Euclidean vector
-	<a href="#">Date.java</a> 	date
-	<a href="#">Transaction.java</a> 	transaction
-	<a href="#">Point2D.java</a> 	point
-	<a href="#">RectHV.java</a> 	axis-aligned rectangle
-	<a href="#">Interval1D.java</a> 	1d interval
-	<a href="#">Interval2D.java</a> 	2d interval
-	<a href="#">Accumulator.java</a> 	running average and stddev
1.1	<a href="#">ResizingArrayStack.java</a> 	LIFO stack (resizing array)
1.2	<a href="#">LinkedStack.java</a> 	LIFO stack (linked list)
-	<a href="#">Stack.java</a> 	LIFO stack
-	<a href="#">ResizingArrayQueue.java</a> 	FIFO queue (resizing array)
1.3	<a href="#">LinkedQueue.java</a> 	FIFO queue (linked list)
-	<a href="#">Queue.java</a> 	FIFO queue
-	<a href="#">ResizingArrayBag.java</a> 	multiset (resizing array)
1.4	<a href="#">LinkedBag.java</a> 	multiset (linked list)
-	<a href="#">Bag.java</a> 	multiset
-	<a href="#">Stopwatch.java</a> 	timer (wall time)
-	<a href="#">StopwatchCPU.java</a> 	timer (CPU time)
-	<a href="#">LinearRegression.java</a> 	simple linear regression
-	<a href="#">ThreeSum.java</a> 	brute-force three sum
-	<a href="#">ThreeSumFast.java</a> 	faster three sum
-	<a href="#">DoublingTest.java</a> 	doubling test
-	<a href="#">DoublingRatio.java</a> 	doubling ratio
-	<a href="#">QuickFindUF.java</a> 	quick find
-	<a href="#">QuickUnionUF.java</a> 	quick union
1.5	<a href="#">WeightedQuickUnionUF.java</a> 	weighted quick union
-	<a href="#">UF.java</a> 	union-by-rank with path halving

Last modified on August 26, 2016.

Copyright © 2000–2019 [Robert Sedgwick](#) and [Kevin Wayne](#). All rights reserved.