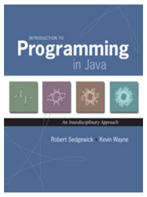
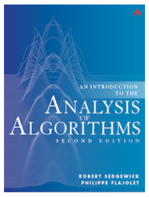
Algorithms
FOURTH EDITION
ROBERT SEDGEWICK | KEVIN WAYNE





Search

Java Algorithms and Clients

Our original goal for this book was to cover *the 50 algorithms that every programmer should know.* We use the word *programmer* to refer to anyone engaged in trying to accomplish something with the help of a computer, including scientists, engineers, and applications developers, not to mention college students in science, engineering, and computer science.

Algorithms and clients in the textbook. The list below includes a few more than 100 Java programs (some are clients, some others are basic infrastructure). Click on the program name to access the Java code; click on the description to access the javadoc; click on the data file names to access the data. You can download all of the programs as algs4.jar and the data as algs4-data.zip.

1	FUNDAMENTALS		DATA
–	BinarySearch.java	binary search	tinyW.txt tinyT.txt largeW.txt largeT.txt
		random numbers	

–	RandomSeq.java	in a given range	–
–	Average.java	average of a sequence of numbers	–
–	Cat.java	concatenate files	in1.txt in2.txt
–	Shuffle.java	Knuth shuffle	cards.txt
–	Counter.java	counter	–
–	StaticSETofInts.java	set of integers	–
–	Whitelist.java	whitelist client	tinyW.txt tinyT.txt largeW.txt largeT.txt
–	Vector.java	mathematical vector	–
–	Date.java	date	–
–	Transaction.java	transaction	–
–	Point2D.java	point	–
–	Interval1D.java	1-d interval	–
–	Interval2D.java	2-d interval	–
1.1	ResizingArrayStack.java	LIFO stack (resizing array)	tobe.txt
1.2	Stack.java	LIFO stack (linked list)	tobe.txt
–	ResizingArrayQueue.java	FIFO queue (resizing array)	tobe.txt
1.3	Queue.java	FIFO queue (linked list)	tobe.txt
1.4	Bag.java	multiset	–
–	Stopwatch.java	timer	–
–	ThreeSum.java	brute-force three sum	1Kints.txt 2Kints.txt 4Kints.txt 8Kints.txt
–	ThreeSumFast.java	faster three sum	1Kints.txt 2Kints.txt 4Kints.txt 8Kints.txt
–	DoublingTest.java	doubling test	–
–	DoublingRatio.java	doubling ratio	–
–	QuickFindUF.java	quick find	tinyUF.txt mediumUF.txt largeUF.txt

–	QuickUnionUF.java	quick union	tinyUF.txt mediumUF.txt largeUF.txt
1.5	WeightedQuickUnionUF.java	weighted quick union	tinyUF.txt mediumUF.txt largeUF.txt
1.5	UF.java	union find	tinyUF.txt mediumUF.txt largeUF.txt
–	QuickUnionPathCompressionUF.java	path compression	tinyUF.txt mediumUF.txt largeUF.txt
2	**SORTING**		**DATA**
2.1	Insertion.java	insertion sort	tiny.txt words3.txt
2.2	Selection.java	selection sort	–
2.3	Shell.java	shellsort	–
2.4	Merge.java	top-down mergesort	–
–	MergeBU.java	bottom-up mergesort	–
2.5	Quick.java	quicksort	–
–	Quick3way.java	quicksort with 3-way partitioning	–
–	TopM.java	priority queue client	tinyBatch.txt
2.6	MaxPQ.java	max heap priority queue	tinyPQ.txt
–	MinPQ.java	min heap priority queue	tinyPQ.txt
–	IndexMinPQ.java	index min heap priority queue	–
–	IndexMaxPQ.java	index max heap priority queue	–
–	Multiway.java	multiway merge	m1.txt m2.txt m3.txt
2.7	Heap.java	heapsort	tiny.txt words3.txt
3	**SEARCHING**		**DATA**
–	FrequencyCounter.java	frequency counter	tinyTale.txt tale.txt leipzig1M.txt
3.1	SequentialSearchST.java	sequential search	tinyST.txt
3.2	BinarySearchST.java	binary search	tinyST.txt

3.3	BST.java	binary search tree	tinyST.txt
3.4	RedBlackBST.java	red-black tree	tinyST.txt
3.5	SeparateChainingHashST.java	separate chaining hash table	–
3.6	LinearProbingHashST.java	linear probing hash table	–
–	ST.java	ordered symbol table	–
–	SET.java	ordered set	–
–	DeDup.java	remove duplicates	tinyTale.txt
–	WhiteFilter.java	whitelist filter	list.txt tinyTale.txt
–	BlackFilter.java	blacklist filter	list.txt tinyTale.txt
–	LookupCSV.java	dictionary lookup	ip.csv DJIA.csv amino.csv UPC.csv
–	LookupIndex.java	index and inverted index	aminoI.csv movies.txt
–	FileIndex.java	file indexing	ex1.txt ex2.txt ex3.txt ex4.txt
–	SparseVector.java	sparse vector	–
4	**GRAPHS**		**DATA**
4.1	Graph.java	undirected graph	tinyG.txt mediumG.txt
–	DepthFirstSearch.java	depth-first search in a graph	tinyG.txt mediumG.txt
4.2	DepthFirstPaths.java	paths in a graph (DFS)	tinyCG.txt
4.3	BreadthFirstPaths.java	paths in a graph (BFS)	tinyCG.txt
4.4	CC.java	connected components of a graph	tinyG.txt mediumG.txt
–	Bipartite.java	bipartite or odd cycle	–
–	Cycle.java	cycle in a graph	–
–	SymbolGraph.java	symbol graph	routes.txt movies.txt
–	DegreesOfSeparation.java	degrees of separation	routes.txt movies.txt
4.5	Digraph.java	directed graph	tinyDG.txt

–	DigraphGenerator.java	generate random digraphs	–
4.4	DirectedDFS.java	depth-first search in a digraph	tinyDG.txt
–	DepthFirstDirectedPaths.java	paths in a digraph (DFS)	–
–	DirectedCycle.java	cycle in a digraph	tinyDG.txt tinyDAG.txt
–	DepthFirstOrder.java	depth-first order in a digraph	tinyDG.txt tinyDAG.txt
4.5	Topological.java	topological order in a DAG	jobs.txt
–	BreadthFirstDirectedPaths.java	paths in a digraph (BFS)	tinyDG.txt
–	TransitiveClosure.java	transitive closure	tinyDG.txt
–	SymbolDigraph.java	symbol digraph	–
4.6	KosarajuSharirSCC.java	strong components in a digraph	tinyDG.txt
4.7	EdgeWeightedGraph.java	edge-weighted graph	–
–	Edge.java	weighted edge	–
4.8	LazyPrimMST.java	MST (lazy Prim)	tinyEWG.txt mediumEWG.txt
–	PrimMST.java	MST (Prim)	tinyEWG.txt mediumEWG.txt
4.9	KruskalMST.java	MST (Kruskal)	tinyEWG.txt mediumEWG.txt
–	BoruvkaMST.java	MST (Boruvka)	tinyEWG.txt mediumEWG.txt
4.10	EdgeWeightedDigraph.java	edge-weighted digraph	tinyEWD.txt
–	DirectedEdge.java	weighted, directed edge	–
4.11	DijkstraSP.java	shortest paths (Dijkstra)	tinyEWD.txt mediumEWD.txt
–	DijkstraAllPairsSP.java	all-pairs shortest paths	tinyEWD.txt mediumEWD.txt
4.12	AcyclicSP.java	shortest paths in a DAG	tinyEWDAG.txt
		longest paths in a	

–	AcyclicLP.java	DAG	tinyEWDAG.txt
–	CPM.java	critical path method	jobsPC.txt
4.13	BellmanFordSP.java	shortest paths (Bellman-Ford)	tinyEWDn.txt tinyEWDnc.txt
–	EdgeWeightedDirectedCycle.java	cycle in an edge-weighted digraph	–
4.15	Arbitrage.java	arbitrage detection	rates.txt
–	FloydWarshall.java	all-pairs shortest paths (dense)	tinyEWD.txt
–	AdjMatrixEdgeWeightedDigraph.java	edge-weighted graph (dense)	tinyEWD.txt
5	**STRINGS**		**DATA**
–	Alphabet.java	alphabet	–
–	Count.java	alphabet client	abra.txt pi.txt
5.1	LSD.java	LSD radix sort	words3.txt
5.2	MSD.java	MSD radix sort	shells.txt
5.3	Quick3string.java	3-way string quicksort	shells.txt
5.4	TrieST.java	multiway trie	shellsST.txt
5.5	TST.java	ternary search trie	shellsST.txt
5.6	KMP.java	Knuth-Morris-Pratt substring search	–
5.7	BoyerMoore.java	Boyer-Moore substring search	–
5.8	RabinKarp.java	Rabin-Karp substring search	–
5.9	NFA.java	NFA for regular expressions	–
–	GREP.java	grep	–
–	BinaryDump.java	binary dump	abra.txt
–	HexDump.java	hex dump	abra.txt
–	PictureDump.java	picture dump	abra.txt

–	Genome.java	genomic code	genomeTiny.txt genomeVirus.txt
–	RunLength.java	run-length coding	4runs.bin q32x48.bin q64x96.bin
5.10	Huffman.java	Huffman coding	tinytinyTale.txt medTale.txt tale.txt
5.11	LZW.java	Lempel-Ziv-Welch coding	abraLZW.txt ababLZW.txt
6	**CONTEXT**		**DATA**
6.1	CollisionSystem.java	collision system	brownian.txt diffusion.txt
–	Particle.java	particle	–
6.2	BTree.java	B-tree	–
6.3	SuffixArray.java	suffix array	abra.txt
–	LRS.java	longest repeated substring	tinyTale.txt mobydick.txt
–	KWIK.java	keyword in context	tale.txt
6.4	FordFulkerson.java	max flow / min cut	tinyFN.txt
–	FlowNetwork.java	capacitated network	–
–	FlowEdge.java	capacitated edge with flow	–
–	BipartiteMatching.java	bipartite matching	–
–	AssignmentProblem.java	weighted bipartite matching	–
–	Simplex.java	simplex method	–
9	**BEYOND**		**DATA**
–	GaussianElimination.java	Gaussian elimination	–
–	FFT.java	Fast Fourier transform	–
–	Complex.java	complex number	–
–	GrahamScan.java	2d convex hull	rs1423.txt
–	FarthestPair.java	2d farthest pair	rs1423.txt

–	ClosestPair.java	2d closest pair	rs1423.txt

Here a few algorithms on our wishlist. If you wish to implement any of these and contribute to algs4.jar, send us an email and we'd be happy to include your code with an appropriate attribution. Be sure to thoroughly test and comment your code; strive for clarity; and use a style consistent with the other programs in the library. We also welcome any Javadoc comment contributions—our main data types have Javadoc comments, but most clients do not yet have Javadoc comments.

Standard input and output libraries.

We use these standard input and output libraries from *Introduction to Programming: An Interdisciplinary Approach*. You can download them all together as stdlib.jar.

§	PROGRAM	DESCRIPTION / JAVADOC
1.5	StdIn.java	read numbers and text from standard input
1.5	StdOut.java	write numbers and text to standard output
1.5	StdDraw.java	draw geometric shapes in a window
1.5	StdAudio.java	create, play, and manipulate sound
2.2	StdRandom.java	generate random numbers
2.2	StdStats.java	compute statistics
2.2	StdArrayIO.java	read and write 1D and 2D arrays
3.1	In.java	read numbers and text from files and URLs
3.1	Out.java	write numbers and text to files
3.1	Draw.java	draw geometric shapes
3.1	Picture.java	process digital images
3.2	Stopwatch.java	measure running time
–	BinaryStdIn.java	read bits from standard input
–	BinaryStdOut.java	write bits to standard output
–	BinaryIn.java	read bits from files and URLs

–	BinaryOut.java	write bits to files

Installing Java.

Here are instructions for installing a Java programming environment on your operating system: Mac OS X, Windows, or Linux.

Installing the textbook libraries.

To use the textbook libraries, download stdlib.jar and algs4.jar and add them to your Java classpath. Do not unjar them. Here is how to accomplish that in a variety of environments:

- *Mac OS X (automatic).* The Mac OS X installer downloads stdlib.jar and algs4.jar to the `/Users/username/introcs` folder; it also adds each jar file to the CLASSPATH environment variable and to the DrJava classpath.

- *Windows (automatic).* The Windows installer downloads stdlib.jar and algs4.jar to the `C:\Users\username\introcs` folder; it also adds each each jar file to the CLASSPATH environment variable and to the DrJava classpath.

- *Windows Command Prompt (manual).* Downloads stdlib.jar and algs4.jar to a folder, say `C:\Users\username\algs4`. Next, add each jar file to the CLASSPATH environment variable.

  - Windows 7: *Start -> Computer -> System Properties -> Advanced system settings -> Environment Variables -> User variables -> CLASSPATH.*

    Vista: *Start -> My Computer -> Properties -> Advanced -> Environment Variables -> User variables -> CLASSPATH.*

    Windows XP: *Start -> Control Panel -> System -> Advanced -> Environment Variables -> User variables -> CLASSPATH.*

  - Prepend the following to the *beginning* of the CLASSPATH variable:

    ```
    C:\Users\username\algs4\stdlib.jar;C:\Users\username\algs4\algs4.jar;
    ```

    The semicolons separate entries in the CLASSPATH.

  - Click OK three times.

  If you don't see a variable named CLASSPATH, click *New* and in the popup window enter CLASSPATH for the variable name. Then, perform the instructions above.

- *Mac OS X Terminal (manual).* Downloads stdlib.jar and algs4.jar to a folder, say `~/algs4`. Depending on your shell, add the following line or lines to the file specified:

  - *Bourne-again shell (bash).* Add the following line to the file `~/.bash_profile` (if it exists); otherwise add it to the file `~/.bash_login` (if it exists); otherwise, add it to the file `~/.profile` (if it doesn't exist, create it first):

    ```
    export CLASSPATH=$CLASSPATH:~/algs4/stdlib.jar:~/algs4/algs4.jar
    ```

    The colons separate entries in the CLASSPATH.

  - *C shell (csh).* Add the following line to the file `~/.cshrc` (if it doesn't exist, create it first):

    ```
    if ( !($?CLASSPATH) ) then
        setenv CLASSPATH .:~/algs4/stdlib.jar:~/algs4/algs4.jar
    else
        setenv CLASSPATH .:~/algs4/stdlib.jar:~/algs4/algs4.jar:${CLASSPATH}
    endif
    ```

  - *Bourne shell (sh).* Add the following line to the file `~/.profile` (if it doesn't exist, create it first):

    ```
    export CLASSPATH=$CLASSPATH:~/algs4/stdlib.jar:~/algs4/algs4.jar
    ```

  - *T shell (tcsh).* Add the following line to the file `~/.tcshrc` (if it exists); otherwise add it to the file `~/.cshrc` (if it doesn't exist, create it first):

    ```
    if ( !($?CLASSPATH) ) then
        setenv CLASSPATH .:~/algs4/stdlib.jar:~/algs4/algs4.jar
    else
        setenv CLASSPATH .:~/algs4/stdlib.jar:~/algs4/algs4.jar:${CLASSPATH}
    endif
    ```

- *Linux Command Line (manual).* Follow the same instructions as for Mac OS X Terminal.

- *DrJava (manual).* Download stdlib.jar and algs4.jar to a folder and add each jar file to the classpath via *Preferences -> Resources -> Extra Classpath -> Add*.

- *Eclipse (manual).* Download stdlib.jar and algs4.jar to a folder and add each jar file to the classpath variable to the build path of a project via *Project -> Properties -> Java Build Path -> Libaries -> Add External JARs*.

Download our test data files.

To use the data, unzip algs4-data.zip. It will create a subdirectory `algs4-data` with all of the data files used in the textbook.

Exercise solutions.

Here is a list of solutions to selected coding exercises.

1	**FUNDAMENTALS**	
1.2.13	Transaction.java	transaction data type
1.2.16	Rational.java	rational number data type
1.2.19	Date.java	date data type
1.3.1	FixedCapacityStackOfStrings.java	add isFull() method to stack
1.3.4	Parentheses.java	balanced parentheses
1.3.7	Stack.java	add peek() method to stack
1.3.10	InfixToPostfix.java	infix-to-postfix conversion
1.3.11	EvaluatePostfix.java	evaluate a postfix expression
1.3.14	ResizingArrayQueue.java	resizing array queue
1.3.37	Josephus.java	Josephus problem
1.4.14	FourSum.java	brute-force 4-sum
1.5.7	QuickUnionUF.java	quick union
1.5.7	QuickFindUF.java	quick find
1.5.17	ErdosRenyi.java	Erdos-Renyi simulation
2	**SORTING**	
2.1.1	TraceSelection.java	trace of selection sort
2.1.4	TraceInsertion.java	trace of insertion sort
2.1.9	TraceShell.java	trace of shellsort
2.1.21	Transaction.java	add natural order to Transaction

2.1.22	SortTransactions.java	sort transactions
2.1.23	InsertionX.java	insertion sort with sentinel
2.1.24	InsertionX.java	insertion sort with half exchanges
2.2.2	TraceMerge.java	mergesort trace
2.2.3	TraceMergeBU.java	bottom-up mergesort trace
2.2.9	Merge.java	mergesort without static array
2.2.11	MergeX.java	improved mergesort
2.2.19	Inversions.java	count number of inversions
2.2.20	Merge.java	index sort
2.3.1	TracePartition.java	partition trace
2.3.2	TraceQuick.java	quicksort trace
2.3.5	Sort2distinct.java	sort array with 2 distinct keys
2.3.12	TraceQuick3way.java	3-way quicksort trace
2.3.16	QuickBest.java	best-case for quicksort
2.3.22	QuickX.java	Bentley-McIlroy 3-way quicksort
2.4.3	OrderedArrayMaxPQ.java	ordered array priority queue
2.4.3	UnorderedArrayMaxPQ.java	unordered array priority queue
2.4.15	MaxPQ.java	check if an array is heap-ordered
2.4.25	CubeSum.java	find a^3 + b^3 = c^3 + d^3
2.4.33	IndexMaxPQ.java	index priority queue
2.5.8	Frequency.java	count word frequencies
2.5.12	SPT.java	shortest processing time first rule
2.5.13	LPT.java	longest processing time first rule
2.5.14	Domain.java	sort by reverse domain name
2.5.16	California.java	2003 California gubernatorial election order
2.5.19	KendallTau.java	Kendall Tau distance

2.5.24	StableMinPQ.java	stable priority queue
2.5.25	Point2D.java	point comparators
2.5.27	Insertion.java	index sort
2.5.28	FileSorter.java	sort files by name
3	**SEARCHING**	
3.1.1	GPA.java	compute GPA
3.1.2	ArrayST.java	unordered-array symbol table
3.1.5	SequentialSearchST.java	add size(), delete(), and keys()
3.1.16	BinarySearchST.java	add delete()
3.1.17	BinarySearchST.java	add floor()
3.1.29	TestBinarySearchST.java	test client
3.1.30	BinarySearchST.java	check internal invariants
3.2.6	BST.java	add height() method
3.2.10	TestBST.java	test client
3.2.13	NonrecursiveBST.java	nonrecursive BST
3.2.24	PerfectBalance.java	perfectly balanced BST
3.2.32	BST.java	order check
3.2.33	BST.java	rank/select check
4	**GRAPHS**	
4.1.3	Graph.java	add copy constructor
4.1.13	BreadthFirstPaths.java	add distTo() method
4.1.23	BaconHistogram.java	histogram of Bacon numbers
4.1.26	DegreesOfSeparationDFS.java	degrees of separation with DFS
4.1.27	MemoryOfGraph.java	memory of Graph data type
4.1.36	Bridge.java	find bridges
4.2.3	Digraph.java	add copy constructor
4.2.21	MemoryOfDigraph.java	memory of Digraph data type
4.2.23	DirectedEulerianCycle.java	directed Eulerian cycle
4.2.31	TopologicalQueue.java	queue-based topologial sort
4.2.39	WebCrawler.java	web crawler

4.3.9	EdgeWeightedGraph.java	edge-weighted graph constructor
4.3.11	MemoryOfEdgeWeightedGraph.java	memory of edge-weighted graph
4.3.17	EdgeWeightedGraph.java	add toString() to EdgeWeightedGraph
4.3.21	PrimMST.java	add edges() to PrimMST
4.3.22	PrimMST.java	minimum spanning forrest
4.3.22	KruskalMST.java	minimum spanning forrest
4.3.33	KruskalMST.java	MST certification
4.3.43	BoruvkaMST.java	Boruvka's algorithm
4.4.2	EdgeWeightedDigraph.java	add toString() method
4.4.11	MemoryOfEdgeWeightedDigraph.java	memory of EdgeWeightedDigraph data type
4.4.12	Topological.java	topological sort in edge-weighted digraphs
4.4.12	EdgeWeightedDirectedCycle.java	directed cycle in edge-weighted digraphs
4.4.28	AcyclicLP.java	longest paths in DAGs
4.4.35	LazyDijkstraSP.java	lazy implementation of Dijkstra

Q + A

Q. Can I use your code in my project?

A. Our libraries `stdlib.jar` and `algs4.jar` are released under the GNU General Public License, version 3 (GPLv3). If you wish to license the code under different terms, please contact our publisher to discuss.

Q. If I use a named package to structure my code, the compiler can no longer access the libraries in `stdlib.jar` or `algs4.jar`. Why not?

A. The libraries in `stdlib.jar` and `algs4.jar` are in the "default" package. In Java, you can't access classes in the default package from a named package. If you need to use our libraries with a named package, you can use these package versions: stdlib-package.jar and algs4-package.jar.

Warning: if you are taking Princeton COS 226 or Coursera, Algorithms, Part I or II, you must use the default package verison of our libraries to facilitate grading.

Q. Why is there Javadoc for only some of the classes in algs4.jar?

A. We documented the most important classes in the library and we hope to slowly do more. To help us along, we welcome crowdsourcing efforts—just send us the Javadoc'd verison of a class (being sure to maintain a consistent style, e.g., please don't use tabs!) and we'll update.

Last modified on April 20, 2013.