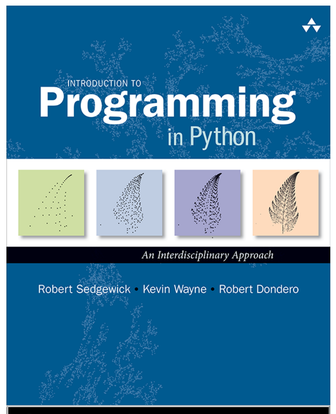

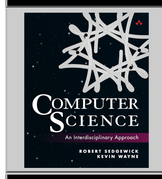

ENHANCED BY Google

4. Algorithms and Data Structures

This chapter presents fundamental data types that are essential building blocks for a broad variety of applications. We present full implementations, even though some of them are built into Python, so that you can have a clear idea of how they work and why they are important.

The algorithms and data structures that we consider in this chapter introduce a body of knowledge developed over the past several decades that constitutes the basis for the efficient use of computers for a broad variety of applications. As the scope of computing applications continues to expand, so grows the impact of these basic approaches.

- *4.1 Performance* outlines a scientific method and powerful theory for understanding the performance and resource consumption of the program that we write.
- *4.2 Sorting and Searching* describes two classical algorithms — mergesort and binary search — along with several applications where their efficiency plays a critical role.
- *4.3 Stacks and Queues* introduces two closely related data structures for manipulating arbitrary large collections of data.
- *4.4 Symbol Tables* considers a quintessential data structure known as the symbol table for storing information, and an efficient implementation known as the binary search tree.
- *4.5 Small World Phenomenon* presents a case study to investigate the small world phenomenon — the principle that we are all linked by short chains of acquaintances.

Python Programs in this Chapter

Below is a list of Python programs in this chapter.

REF	PROGRAM	DESCRIPTION	DATA
4.1.1	threesum.py	3-sum problem	8ints.txt 1kints.txt 2kints.txt 4kints.txt 8kints.txt 16kints.txt 32kints.txt 64kints.txt 128kints.txt
4.1.2	doublingtest.py	validating a doubling hypothesis	–
4.1.3	timeops.py	timing operators and functions	–
4.1.4	bigarray.py	discovering memory capacity	–
4.2.1	questions.py	binary search (20 questions)	–
4.2.2	bisection.py	binary search (inverting a function)	–
4.2.3	binarysearch.py	binary search (sorted array)	emails.txt white.txt
4.2.4	insertion.py	insertion sort	tiny.txt tomsawyer.txt
4.2.5	timesort.py	doubling test for sorting functions	–
4.2.6	merge.py	mergesort	tiny.txt tomsawyer.txt
4.2.7	frequencycount.py	frequency counts	leipzig100k.txt leipzig200k.txt leipzig1m.txt
4.3.1	arraystack.py	stack (resizing array implementation)	tobe.txt
4.3.2	linkedstack.py	stack (linked list implementation)	tobe.txt
4.3.3	evaluate.py	expression evaluation	expression1.txt expression2.txt
4.3.4	linkedqueue.py	queue (linked list implementation)	tobe.txt
4.3.5	mm1queue.py	M/M/1 queue simulation	–
4.3.6	loadbalance.py	load balancing simulation	–
4.4.1	lookup.py	dictionary lookup	amino.csv djia.csv elements.csv ip.csv ip-by-country.csv morse.csv phone-na.csv
4.4.2	index.py	indexing	mobydick.txt tale.txt
4.4.3	hashst.py	hash symbol table data type	–
4.4.4	bst.py	BST symbol table data type	–
4.5.1	graph.py	graph data type	tinygraph.txt
4.5.2	invert.py	using a graph to invert an index	tinygraph.txt movies.txt
4.5.3	separation.py	shortest-paths client	routes.txt movies.txt
4.5.4	pathfinder.py	shortest-paths client	–
4.5.5	smallworld.py	small-world test	tinygraph.txt
4.5.6	performer.py	performer-performer graph	tinymovies.txt moviesg.txt

Copyright © 2000–2015 by Robert Sedgewick, Kevin Wayne, and Robert Dondero. All rights reserved.