CHAPTER 14

Algorithm Efficiency and Sorting

PREVIEW

This chapter will show you how to analyze the efficiency of algorithms. The basic mathematical techniques for analyzing algorithms are central to more advanced topics in computer science and give you a way to formalize the notion that one algorithm is significantly more efficient than another.

As examples, you will see analyses of some algorithms that you have studied before, including those that search data. In addition, this chapter examines the important topic of sorting data. You will study some simple algorithms, which you may have seen before, and some more-sophisticated recursive algorithms. Sorting algorithms provide varied and relatively easy examples of the analysis of efficiency.

9.1 Measuring the Efficiency of Algorithms

The comparison of algorithms is a topic that is central to computer science. Measuring an algorithm's efficiency is quite important because your choice of algorithm for a given application often has a great impact. Responsive word processors, automatic teller machines, video games, and life support systems all depend on efficient algorithms.

Suppose two **algorithms** perform the same task, such as searching. What does it mean to compare the algorithms and conclude that one is better? Chapter 1 discussed the several components that contribute to the cost of a computer program. Some of these components involve the cost of human time—the time of the people who develop, maintain, and use the program. The other components involve the cost of program execution—that is, the program's **efficiency**—measured by the amount of computer time and space that the program requires to execute.

Loosely coupled modules are independent

Specify each module's purpose, assumptions, input and output

What Is Problem Solving?

We have, up to this point, emphasized the human cost of a computer program. The early chapters of this book stressed style and readability. They pointed out that well-designed algorithms reduce the human costs of implementing the algorithm with a program, of maintaining the program, and of modifying the program.

> The primary concern has been to develop problem-solving skills and **programming style**. Although we shall continue to concentrate our efforts in that direction, the efficiency of algorithms is also important.

Highly cohesive modules perform one well-defined task

Efficiency is a criterion that you should use when selecting an algorithm and its implementation. The solutions in this book, in addition to illustrating good programming style, are frequently based on relatively efficient algorithms.

Phase 1: Specification. Part I of this book reviewed aspects of problem solving that are closely related to programming issues, presented data abstraction as a technique for solution design that permeates our approach to problem solving, introduced C++ classes as a way to hide a solution's implementation details and to increase its **modularity**, introduced the linked list as a data structure throughout this book.

Specifications as a contract

> The function will receive an array of *num* integers, where *num* > 0.
>
> The function will return the array with the integers stored.

The primary concerns of the remainder of this book are the aspects of problem solving that involve the management of data—that is, the identification and implementation of some of the more common data-management operations.