19.5 Sorting Algorithms

Sorting data (i.e., placing the data into some particular order, such as ascending or descending) is one of the most important computing applications. A bank sorts all transactions by account number so that it can prepare individual bank statements at the end of each month. Telephone companies sort their lists of accounts by last name and, further, by first name to make it easy to find phone numbers. Virtually every organization must sort some data, and often massive amounts of it. Sorting data is an intriguing, computer-intensive problem that has attracted intense research efforts.

An important item to understand about sorting is that the end result—the sorted array—will be the *same* no matter which algorithm you use to sort the array. The choice of algorithm affects only the run time and memory use of the program. The rest of this chapter introduces three common sorting algorithms. The first two—*selection sort* and *insertion sort*—are easy to program but *inefficient*. The last algorithm—*merge sort*—is much *faster* than selection sort and insertion sort but *harder to program*. We focus on sorting arrays of primitive-type data, namely `int`s. It's possible to sort arrays of class objects as well, as we demonstrated in Section 16.7.1 by using the built-in sorting capabilities of the Collections API.