

(For an explanation of this summation, see Appendix A.3.) The fastest-growing term is  $n^2/2$ , so the selection sort algorithm has time complexity  $O(n^2)$ .

It is also possible to analyze a sorting algorithm in terms of copies. Step 1.2 of the selection sort algorithm performs two copies. The main loop is iterated  $n - 1$  times, so we get:

$$\text{No. of copies} = 2(n - 1) \quad (3.9)$$

In terms of copies alone, selection sort is  $O(n)$ . This does not affect its overall time complexity, which is  $O(n^2)$ .

Program 3.30 shows a Java implementation of the selection sort algorithm.

### 3.6.2 Insertion sort

Let us briefly consider a slightly different problem: reading a file of values into an array, where the file is unsorted, but the array must be sorted. Of course, we could just copy all the values from the file into the array, and then sort the array. Alternatively, we could keep the array sorted at all times: read one value at a time, and insert it into the array in such a way as to keep the array sorted. This turns out to be a better algorithm (see Exercise 3.15).

We can exploit the same idea to sort values already stored in an array. This is called *insertion sort*, and is shown as Algorithm 3.31. Step 1.2 is a variant of the array insertion algorithm (see Exercise 3.10(c)).

Figure 3.32 shows the loop invariant for the insertion sort algorithm. The left subarray,  $a[\text{left}..r - 1]$ , contains the same values that were originally in this subarray, but by now

```

static void selectionSort
    (Comparable[] a, int left, int right) {
// Sort a[left...right].
    for (int l = left; l < right; l++) {
        int p = l;
        Comparable least = a[p];
        // ... least will always contain the value of a[p].
        for (int k = l+1; k <= right; k++) {
            int comp = a[k].compareTo(least);
            if (comp < 0) {
                p = k;  least = a[p];
            }
        }
        if (p != l) {
            a[p] = a[l];  a[l] = least;
        }
    }
}

```

**Program 3.30** Implementation of the array selection sort algorithm as a Java method.