

If the search is *unsuccessful*, on the other hand, all the components will be compared:

$$\text{No. of comparisons (unsuccessful search)} = n \quad (3.4)$$

In either case, the linear search algorithm has time complexity $O(n)$.

Program 3.15 shows how the linear search algorithm would be implemented in Java.

Sorted array

Now let us assume that the (sub)array is sorted. This enables us to improve the linear search algorithm slightly. In Figure 3.13, if component $a[p]$ contains a value greater than *target*, the search might as well be terminated immediately, because components $a[p + 1 \dots \text{right}]$ will be greater still. Algorithm 3.16 shows the sorted array linear search algorithm. Whether the search is successful or unsuccessful, this algorithm's average number of comparisons is $(n + 1)/2$. Its time complexity is still $O(n)$, but on average it requires only half as many comparisons as Algorithm 3.12 when the search is unsuccessful (see Exercise 3.8).

Program 3.17 shows how the sorted array linear search algorithm would be implemented in Java. Note that the array components and *target* are assumed to be objects of a class that implements the `Comparable` interface, and the `compareTo` method is used to compare these objects. Steps 1.1 and 1.2 are implemented with a *single* call to the `compareTo` method.

To find which (if any) component of $a[\text{left} \dots \text{right}]$ equals *target*
(where *a* is sorted):

1. For $p = \text{left}, \dots, \text{right}$, repeat:
 - 1.1. If *target* is equal to $a[p]$, terminate with answer *p*.
 - 1.2. If *target* is less than $a[p]$, terminate with answer *none*.
2. Terminate with answer *none*.

Algorithm 3.16 Sorted array linear search algorithm.

```

static int linearSearch (Comparable target,
                        Comparable[] a, int left, int right) {
// Find which (if any) component of a[left...right] equals target
// (where a is sorted).
    for (int p = left; p <= right; p++) {
        int comp = target.compareTo(a[p]);
        if (comp == 0) return p;
        else if (comp < 0) return NONE;
    }
    return NONE;
}

```

Program 3.17 Implementation of the sorted array linear search algorithm as a Java method.