

3.4 Searching

In this section we consider the problem of searching an array or subarray $a[\textit{left} \dots \textit{right}]$ to find which (if any) component contains a value that equals some target value. In general, the (sub)array could be sorted or unsorted.

3.4.1 Linear search

Unsorted array

For the moment, let us assume that the (sub)array is unsorted. The *linear search* (or *sequential search*) algorithm is so called because it compares each array component in turn with the target value: see Algorithm 3.12. As soon as it finds a component that equals the target value, the algorithm terminates and returns that component's index as its answer. If it finds no such component, it returns a special value *none* as its answer.

Figure 3.13 shows the situation between iterations of the linear search algorithm. All components of $a[\textit{left} \dots p - 1]$ have already been searched and are known not to match *target*. The remaining components $a[p \dots \textit{right}]$ are still to be searched, and the matching component (if any) must be among them.

```

static void delete (int del,
                   Object[] a, int left, int right) {
    // Delete the object at position del of a[left...right]
    // (where left ≤ del ≤ right).
    for (int i = del+1; i ≤ right; i++)
        a[i-1] = a[i];
    a[right] = null;
}

```

Program 3.11 Implementation of the array deletion algorithm as a Java method.

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

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

Algorithm 3.12 Unsorted array linear search algorithm.

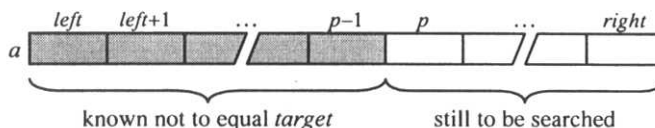


Figure 3.13 Loop invariant for the array linear search algorithms.