

In fact, a much better sorted array searching algorithm is available, which we shall study in the next subsection.

3.4.2 Binary search

Think about how you would search a dictionary for a target word. Would you look at each page in turn, starting at page 1, until you find the page containing the target word? Of course not, but that is exactly what linear search implies. Instead you would probably open the dictionary somewhere in the middle. If by good fortune the target word is on the open page, you are finished. If the target word is less (greater) than the words on the open page, you would focus your attention on the part of the dictionary before (after) that page. Whichever part of the dictionary you have focussed on, you would then search it in the same way, repeatedly dividing the focus of the search until you eventually find the page you are looking for. All this works because the words in a dictionary are sorted.

This idea is the basis of the *binary search* algorithm for a sorted array: see Algorithm 3.18.

Figure 3.19 shows the situation between iterations of the binary search algorithm. All components of $a[\text{left} \dots l - 1]$ are known to be less than *target*, and all components of $a[r + 1 \dots \text{right}]$ are known to be greater than *target*. The remaining components $a[l \dots r]$ are still to be searched, and the matching component (if any) must be among them. Initially all components of $a[\text{left} \dots \text{right}]$ remain to be searched, so the algorithm initializes l to *left* and r to *right*.

Figure 3.20(a) illustrates binary search of a sorted array of words. The components known to be less than or greater than *target* are highlighted (compare Figure 3.19).

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

1. Set $l = \text{left}$, and set $r = \text{right}$.
2. While $l \leq r$, repeat:
 - 2.1. Let m be an integer about midway between l and r .
 - 2.2. If *target* is equal to $a[m]$, terminate with answer m .
 - 2.3. If *target* is less than $a[m]$, set $r = m - 1$.
 - 2.4. If *target* is greater than $a[m]$, set $l = m + 1$.
3. Terminate with answer *none*.

Algorithm 3.18 Array binary search algorithm.

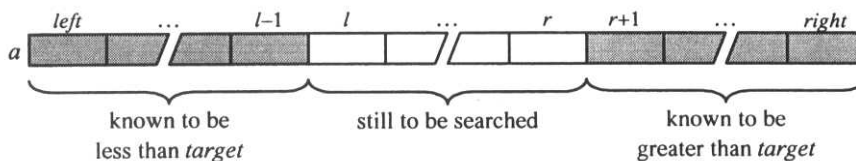


Figure 3.19 Loop invariant for the array binary search algorithm.