Fundamentals of
Data Structures using C

Binary Search

B.Bhuvaneswaran, AP (SG) / CSE

9791519152

bhuvaneswaran@rajalakshmi.edu.in

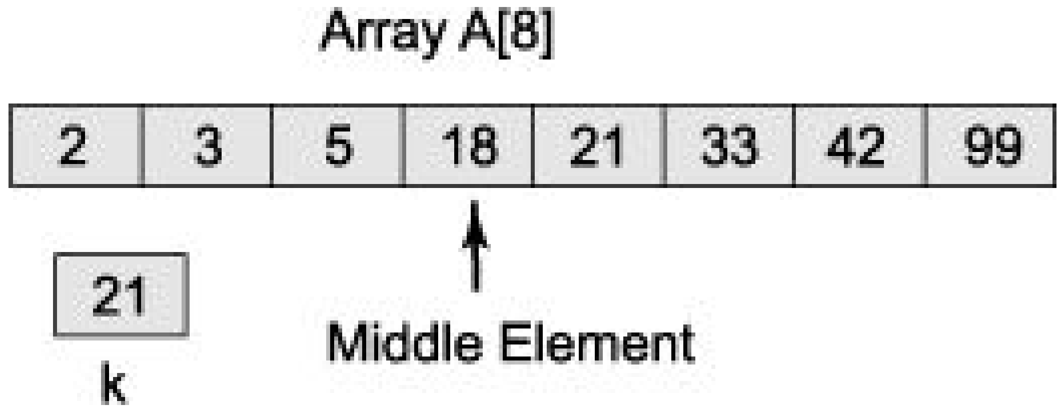
RAJALAKSHMI ENGINEERING COLLEGE

Introduction

- Binary search technique has a prerequisite – it requires the elements of a data structure (list) to be already arranged in a sorted manner before search can be performed in it.
- It begins by comparing the element that is present at the middle of the list.
- If there is a match, then the search ends immediately and the location of the middle element is returned.
- However, if there is a mismatch then it focuses the search either in the left or the right sub list depending on whether the target element is lesser than or greater than middle element.
- The same methodology is repeatedly followed until the target element is found.

Introduction

- Consider an array of integers A containing eight elements, as shown in Fig. Let k = 21 be the value that needs to be searched.



Introduction

- As we can see in Fig., the array A on which binary search is to be performed is already sorted.
- The following steps describe how binary search is performed on array A to search for value k:
    - First of all, the middle element in the array A is identified, which is 18.
    - Now, k is compared with 18. Since k is greater than 18, the search is focused on the right sub list.
    - The middle element in the right sub list is 33. Since k is less than 33, the search is focused on the left sub list, which is {21, 33}.
    - Now, again k is compared with the middle element of {21, 33}, which is 21. Thus, it matches with k.
    - The index value of 21, i.e., 4 is returned and the search is considered as successful.

Routine

```
int BinarySearch(int a[], int n, int key)
{
        int first, last, mid;
        first = 0;
        last = n - 1;
        while (first <= last)
        {
                mid = (first + last) / 2;
                if (a[mid] == key)
                        return mid;
                else if (a[mid] < key)
                        first = mid + 1;
                else
                        last = mid - 1;
        }
        return -1;
}
```

Efficiency of Binary Search-Best Case

- The best case for a binary search algorithm occurs when the element to be searched is present at the middle of the list.
- In this case, only one comparison is made to perform the search operation.
- Thus, efficiency = O(1)

Efficiency of Binary Search-Worst Case

- The worst case for a binary search algorithm occurs when the element to be searched is not present in the list.
- In this case, the list is continuously divided until only one element is left for comparison.
- Let n be the number of list elements and c be the total number of comparisons made in the worst case.
- Now, after every single comparison, the number of list elements left to be searched is reduced by 2.
- Thus, $c = \log_2 n$
- Hence, efficiency = $O(\log_2 n)$

Advantages of Binary Search

- It requires lesser number of iterations.
- It is a lot faster than linear search.

Limitations of Linear Search

- Unlike linear search, it requires the list to be sorted before search can be performed.
- In comparison to linear search, the binary search technique may seem to be a little difficult to implement.

Queries?

Thank You!