Experiment No.11
Application of Binary Search Technique
Name: Sharvari Anand Bhondekar
Roll No: 06
Date of Performance:
Date of Submission:
Marks:
Sign:

Experiment No. 11: Application of Binary Search Technique

Aim: Application of Binary Search Technique.

Objective:

1) Understand the optimal search algorithm in terms of time.
2) Understand the method searching technique.

Theory:

Binary search is the search technique that works efficiently on sorted lists. Hence, to search an element into some list using the binary search technique. Binary search follows the divide and conquer approach in which the list is divided into two halves, and the item is compared with the middle element of the list. If the match is found then, the location of the middle element is returned. Otherwise, we search into either of the halves depending upon the result produced through the match.

Binary search is much faster than linear search for large datasets, with a time complexity of O(log n). The iterative version of binary search uses O(1) additional space, making it memory efficient.

Algorithm:

Binary_Search(a, lower_bound, upper_bound, val) // 'a' is the given array, 'lower_bound' is the index of the first array element, 'upper_bound' is the index of the last array element, 'val' is the value to search

Step 1: set beg = lower_bound, end = upper_bound, pos = - 1

Step 2: repeat steps 3 and 4 while beg <=end

Step 3: set mid = (beg + end)/2

Step 4: if a[mid] = val

set pos = mid

print pos

go to step 6

else if a[mid] > val

set end = mid - 1

else

set beg = mid + 1

[end of if]

[end of loop]

Step 5: if pos = -1

print "value is not present in the array"

[end of if]

Step 6: exit

Code:

```
#include <stdio.h>


int binarySearch(int arr[], int size, int target) {

    int left = 0;

    int right = size - 1;


    while (left <= right) {

    int mid = left + (right - left) / 2;


    if (arr[mid] == target)

    return mid;

    if (arr[mid] < target)
```

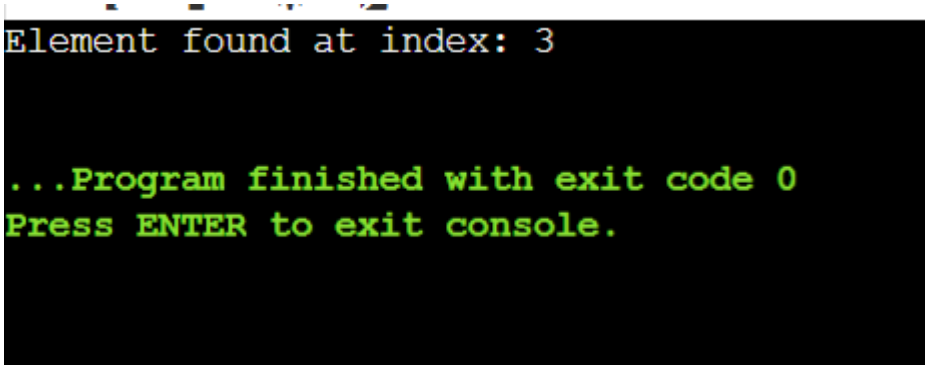
```
        left = mid + 1;
        else
        right = mid - 1;
        }
        return -1;
}

int main() {
        int arr[] = {2, 3, 4, 10, 40};
        int size = sizeof(arr) / sizeof(arr[0]);
        int target = 10;
        int result = binarySearch(arr, size, target);

        if (result != -1)
        printf("Element found at index: %d\n", result);
        else
        printf("Element not found in array.\n");

        return 0;
}
```

Output:



Conclusion:

1. What is searching?

Searching is a fundamental computational operation that involves locating a specific element or a set of elements within a data structure, such as an array, list, or database. The objective of searching is to determine whether a particular value exists in the data structure and, if so, to identify its location (usually its index or key).

2. Differentiate between binary search and linear search.

Base of comparison	Linear search	Binary search
Time complexity	O(N)	O (log2 *N*)
Best case time	O (1) first element	O (1) center element
Prerequisite of an array	No prerequisite	Array must be sorted in order
Input data	No need to be sorted	Need to be sorted
Access	Sequential	random