



Experiment No. : 08

Title: Implementation of searching algorithm



Batch: B-1

Roll No.: 16010422234

Name: Chandana Ramesh Galgali

Aim: Demonstrate the use of Sorting in binary search algorithm**Resources Used:** Turbo C/ C++ editor and C compiler.

Theory:**Searching –**

Search is a process of finding a value in a list of values. In other words, searching is the process of locating a given value position in a list of values.

The **binary search algorithm** can be used with only a sorted list of elements. That means, binary search can be used only with a list of elements which are already arranged in an order. The binary search cannot be used for lists of elements which are in random order. This search process starts comparing the search element with the middle element in the list. If both are matched, then the result is "element found". Otherwise, we check whether the search element is smaller or larger than the middle element in the list. If the search element is smaller, then we repeat the same process for the left sub-list of the middle element. If the search element is larger, then we repeat the same process for the right sub-list of the middle element. We repeat this process until we find the search element in the list or until we are left with a sub- list of only one element. And if that element also doesn't match with the search element, then the result is "Element not found in the list".

By implementing and observing the interplay between sorting and binary search, this lab aims to highlight the importance of a sorted dataset for efficient binary search operations.

Algorithm :

1. **PreCondition** - Sort the given input array using any sorting algorithm as counting sort, quick sort or merge sort

2. **Binary Search algorithm -**

BinarySearch(list[], min, max, key)

if max < min **then**

return false

else

 mid = (max+min) / 2

if list[mid] > key **then**

return BinarySearch(list[], min, mid-1, key)

else if list[mid] < key **then**

return BinarySearch(list[], mid+1, max, key)

else

return mid

end if

end if

Activity:

1. Write a C program to demonstrate the use of Sorting in binary search algorithms.
 2. Demonstrate sorting and searching algorithms in Virtual Lab (screenshots of simulation result and quiz)
-

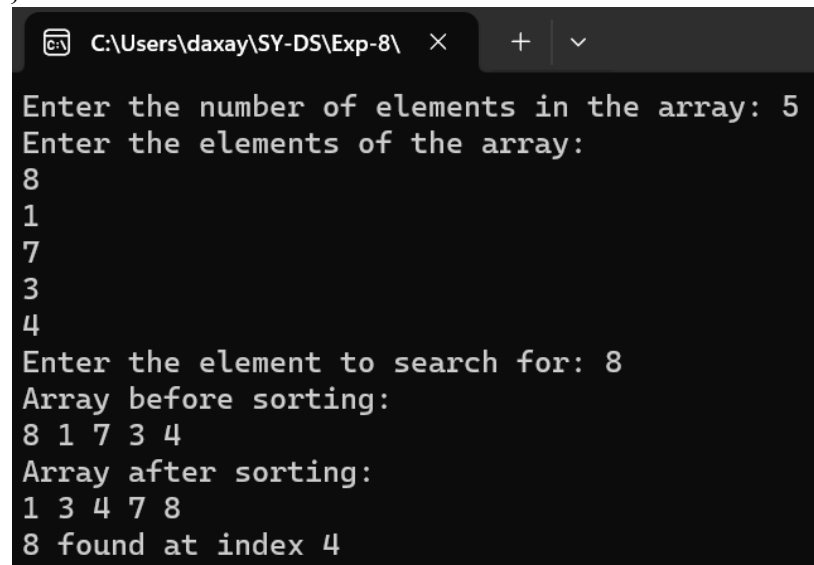
Results:

```
#include <stdio.h>
int binarySearch(int arr[], int low, int high, int key) {
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == key)
            return mid;
        if (arr[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return -1;
}
void bubbleSort(int arr[], int n) {
    int i, j;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
int main() {
    int n, i;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the elements of the array:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int key;
    printf("Enter the element to search for: ");
    scanf("%d", &key);
    printf("Array before sorting:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    bubbleSort(arr, n);
```

```

printf("\nArray after sorting:\n");
for (i = 0; i < n; i++) {
    printf("%d ", arr[i]);
}
int index = binarySearch(arr, 0, n - 1, key);
if (index != -1)
    printf("\n%d found at index %d\n", key, index);
else
    printf("\n%d not found in the array\n", key);
return 0;
}

```



```

C:\Users\daxay\SY-DS\Exp-8\
Enter the number of elements in the array: 5
Enter the elements of the array:
8
1
7
3
4
Enter the element to search for: 8
Array before sorting:
8 1 7 3 4
Array after sorting:
1 3 4 7 8
8 found at index 4

```

Outcomes: Demonstrate sorting and searching methods.

Conclusion:

The experiment showcased the importance of sorting in optimizing the binary search algorithm and highlighted the benefits of using sorting techniques in search algorithms.

References:

Books/ Journals/ Websites:

- Y. Langsam, M. Augenstein and A. Tenenbaum, "Data Structures using C", Pearson Education Asia, 1st Edition, 2002.
- Vlabs on binary search and counting sort.

Link:

<https://ds1-iiith.vlabs.ac.in/exp/unordered-arrays/linear-search/linear-search-algorithm.html>