



Experiment No. : 08

Title: Implementation of searching algorithm



Batch: SY_IT(B3)

Roll No.:16010423076

Experiment No 8

Aim: Demonstrate the use of Sorting in binary searching 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 given value position in a list of values.

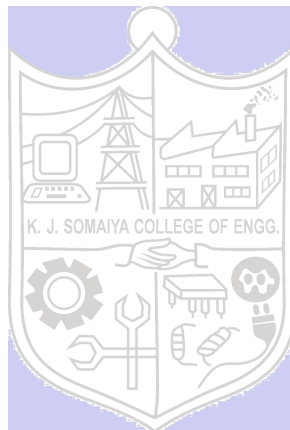
The **binary search algorithm** can be used with only sorted list of element. That means, binary search can be used only with list of element which are already arranged in a order. The binary search cannot be used for list of element which are in random order. This search process starts comparing of 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 left sub-list of the middle element. If the search element is larger, then we repeat the same process for right sub-list of the middle element. We repeat this process until we find the search element in the list or until we 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 C program to demonstrate the use of Sorting in binary search algorithm.

Code :

```
#include <stdio.h>
```

```
void bubblesort(int array[], int n);
```

```
int binarySearch(int array[], int low, int high, int x) {
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (array[mid] == x)
            return mid;
        if (array[mid] < x)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return -1;
}
```

```
void bubblesort(int array[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (array[j] > array[j + 1]) {
                // Swap
                int temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }
}
```

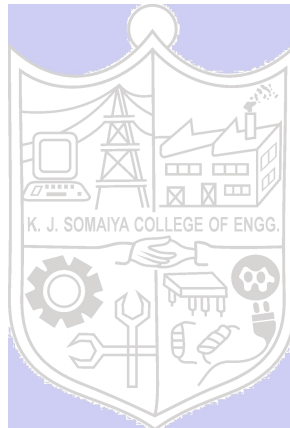
```
int main() {
    int n, x, i;

    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);

    int array[n];

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &array[i]);
    }

    printf("Enter the element to search for: ");
```



```

scanf("%d", &x);

printf("\nOriginal array: ");
for (i = 0; i < n; i++) {
    printf("%d ", array[i]);
}

bubblesort(array, n);

printf("\nSorted array: ");
for (i = 0; i < n; i++) {
    printf("%d ", array[i]);
}

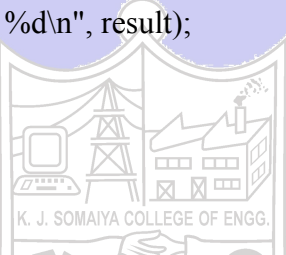
int result = binarySearch(array, 0, n - 1, x);

if (result == -1)
    printf("\nElement not found\n");
else
    printf("\nElement found at index %d\n", result);

return 0;
}

```

Output :



```

Output
/tmp/xlwIiSL8GU.o
Enter the number of elements in the array: 4
Enter 4 elements:
3
86
1
55
Enter the element to search for: 86

Original array: 3 86 1 55
Sorted array: 1 3 55 86
Element found at index 3

=== Code Execution Successful ===

```

2. Demonstrate sorting and searching algorithms in Virtual Lab (screen shots of simulation result and quiz)

<https://www.vlab.co.in/broad-area-computer-science-and-engineering>

Bubble Sort :

Practice:

Instructions



Observations

Correct!

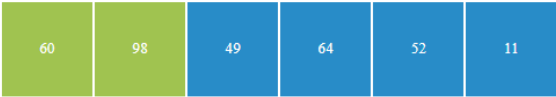
Next

Swap

Reset

Exercise :

Instructions



Observations

Number of iterations: 0

Submit

Next

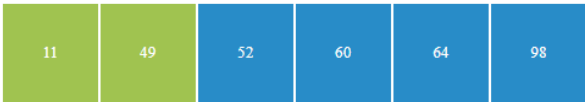
Swap

Undo

Reset



Instructions



Observations

Number of iterations: 4

Submit

Next

Swap

Undo

Reset

Instructions



Observations

CORRECT ANSWER

Submit

Next

Swap

Undo

Reset

**Quiz :**

Computer Science and Engineering > Data Structures - 1 > Experiments

Aim
Overview
Recap
Pretest
Bubble Sort ▾
Aim
Concept
Algorithm
Demo
Practice
Exercise
Quiz
Optimized Bubble Sort ▾
Code Assessment
Analysis ▾
Posttest
Further Readings/References
Feedback

Bubble Sort

Choose difficulty:

☒ Beginner☒ Intermediate☒ Advanced

1. Which of the following statements is true (assume ascending sort order)?

- ☐ a. After T iterations, atleast T of the smallest elements will be in their correct positions.
☐ b. After T iterations, atleast T random elements will be in their correct positions.
☐ c. After T iterations, a random number of elements will be in their correct positions.
☒ d. After T iterations, atleast T of the largest elements will be in their correct positions.

2. To sort an array in descending order, when will we swap two adjacent elements under consideration?

- ☒ a. When the ith element is lesser than the (i + 1)th element.
☐ b. When the ith element is equal to the (i + 1)th element.
☐ c. When the ith element is greater than the (i + 1)th element.
☐ d. None of the above.

3. Consider the following array:

A = [9, -1, -10, 9', 2]

Note that the "'" is used to mark a distinction between the two 9's in order to keep track of their order while sorting. Which of the following represents the steps in sorting the above array (assume ascending order)?

- ☒ a. [9, -1, -10, 9', 2] → [-1, -10, 9, 2, 9'] → [-10, -1, 2, 9, 9'] → [-10, -1, 2, 9, 9'] → [-10, -1, 2, 9, 9']
☐ b. [9, -1, -10, 9', 2] → [-1, -10, 9', 2, 9] → [-10, -1, 2, 9, 9'] → [-10, -1, 2, 9', 9] → [-10, -1, 2, 9, 9']
☐ c. [9, -1, -10, 9', 2] → [-1, -10, 2, 9, 9'] → [-10, -1, 2, 9, 9'] → [-10, -1, 2, 9, 9'] → [-10, -1, 2, 9, 9']
☐ d. [9, -1, -10, 9', 2] → [9, -1, 9', 2, -10] → [9, 9', 2, -1, -10] → [9, 9', 2, -1, -10] → [9, 9', 2, -1, -10]

Submit Quiz

Score: 3 out of 3

**Binary Search :
Practice :**

14	31	40	52	73	77	77	82	88	90
----	----	----	----	----	----	----	----	----	----

Observations

Correct Answer, Great Job!

Number To be Searched: Binary Search Order:

Check Answer

Exercise :

Instructions

13	14	27	31	41	41	42	75	82	91
----	----	----	----	----	----	----	----	----	----

Observations

31 was found in the array at position 3. Binary Search Complete. Great Work!

Min. Speed Max. Speed

Quiz ::

Unsorted Arrays vs Binary Search

Choose difficulty: ☒ Beginner ☐ Intermediate ☐ Advanced

1. What is the prerequisite to perform Binary Search?

☐ a. The values in the array have a maximum bound value they can take. [Explanation](#)

☒ b. Array must be sorted either in ascending or descending order. [Explanation](#)

☐ c. Array must be broken into sub-arrays

☐ d. None of the above

2. Binary Search is an example of _____ algorithm.

☐ a. Greedy

☐ b. Dynamic Programming

☐ c. Backtracking

☒ d. None of the above. [Explanation](#)

3. Let us assume an array {1,2,3,145,178,3203}. How many iterations are needed to find 23? (Assuming we are considering floor of values for floating point values, and index starting from 1)

☒ a. 1 [Explanation](#)

☐ b. 2

☐ c. 3

☐ d. 4

4. Let us assume an array {1, 33, 145, 1294, 1365, 1450, 3300, 4500, 6000, 8000, 9000}. Let us search for 4500 using binary search. What would be the mid values at the second and third iteration respectively? (Assuming we are considering floor of values for floating point values, and index starting from 1)

☐ a. 1450 and 6000

☒ b. 6000 and 2300 [Explanation](#)

☐ c. 8000 and 4500

☐ d. 1450 and 4500

5. What is the space complexity of binary search implemented using recursion?

☐ a. $O(1)$

☐ b. $O(N)$

☐ c. $O(N \log N)$

☒ d. $O(\log N)$ [Explanation](#)

[Submit Quiz](#)

Score: 5 out of 5

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.