



**Batch: A3 Roll No.:16010421075 Experiment :8 Aim: To implement binary search algorithm using array.**



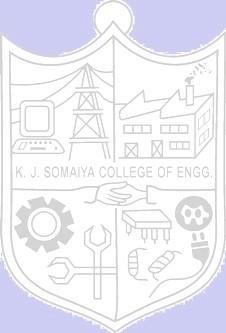
**Resources Used:** Turbo C/ C++ editor and C compiler.



**Theory:**

# Searching –

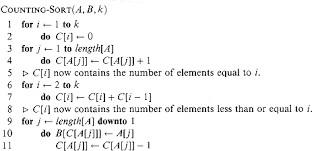
Search is a process of finding a value in a list of values. In other words, searching is the process of locating given value position in a list of values.

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".

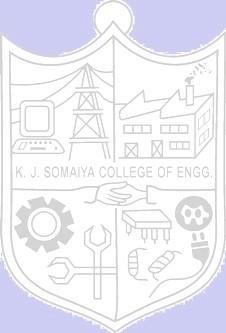


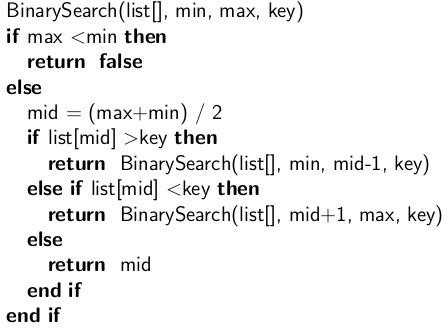
# Algorithm :

1. **PreCondition -** Sort the given input array using counting sort**.**



# Binary Search algorithm -

****



**PROGRAM:-**

#include <stdio.h>

void countingSort(int array[], int size) {

int output[10];

int max = array[0];

for (int i = 1; i < size; i++) {

if (array[i] > max)

max = array[i];

}

int count[10];

for (int i = 0; i <= max; ++i) {

count[i] = 0;

}

for (int i = 0; i < size; i++) {

count[array[i]]++;

}

for (int i = 1; i <= max; i++) {

count[i] += count[i - 1];

}

for (int i = size - 1; i >= 0; i--) {

output[count[array[i]] - 1] = array[i];

count[array[i]]--;

}

for (int i = 0; i < size; i++) {

array[i] = output[i];

}

}

void printArray(int array[], int size) {

for (int i = 0; i < size; ++i) {

printf("%d ", array[i]);

}

printf("\n");

}

int binarySearch(int array[], int x, int low, int high) {

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;

}

int main() {

int array[] = {7,3,4,1,2,6,8};

int loop;

int n = sizeof(array) / sizeof(array[0]);

printf("Orignal array:");

for(loop = 0; loop < n; loop++)

printf("%d ", array[loop]);

printf("\n");

countingSort(array, n);

printf("sorted array :");

printArray(array, n);

int x, location = -1;

printf("Enter the item which you want to search ");

scanf("%d", &x);

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

if (result == -1)

printf("Not found");

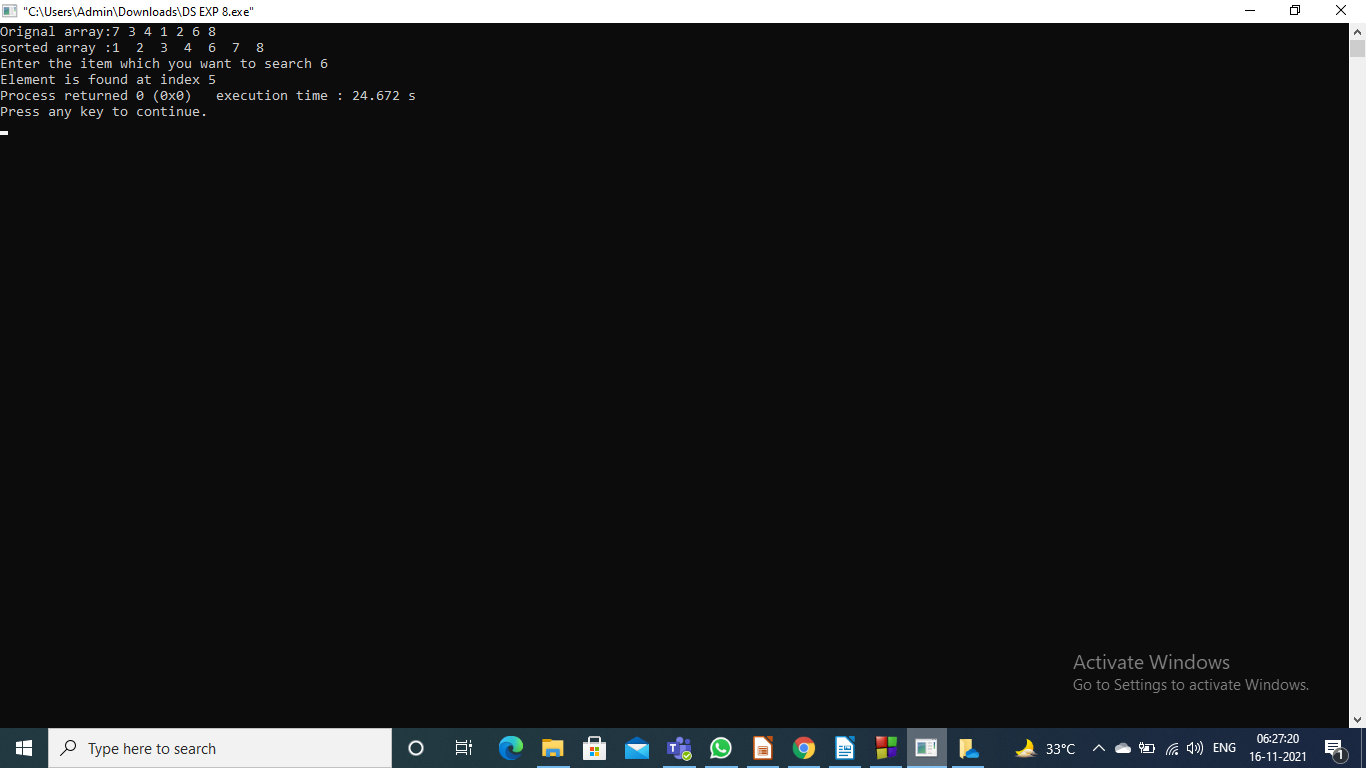
else

printf("Element is found at index %d", result+1);

return 0;

}

**Output:**



# Outcomes:

Apply linear and non-linear data structure in application development



# Conclusion:

**Thus, from the following experiment the concept of Binary Search was understood and implemented in C programming language. The binary search algorithm was implemented and thus the desired objectives were understood and completed**



# References:

**Books/ Journals/ Websites:**

* Y. Langsam, M. Augenstin and A. Tenenbaum, “Data Structures using C”, Pearson Education Asia, 1st Edition, 2002.
* Vlabs on binary search and counting sort.