

**Assignment-2**

**Name: - Mann Kothari**

**Reg. No.: - 22BCB7064**

**Course Title: - Design and Analysis of Algorithm (Embedded Lab)**

**Course Code: - CSE3023**

**Slot: - L21+L22**

**Submitted to: - Prof. Tanikella Divya Naga Pavani**

* **INDEX**
* Linear Search.
* Algorithm
* Code
* Output
* Complexity
* Binary Search
* Algorithm
* Code
* Output
* Complexity
* Complexity Chart
* Implement the following problems using recursion in C/C++.

1. Linear Search.

Algorithm: -

Procedure search(i, j, x: i, j, x integers, 1≤ i ≤ j ≤n)

if a; = x then

return i

else if i j then

return 0

else

return search(i + 1, j, x) {output is the location of x in a1, a2,..., an if it appears; otherwise it is 0)

Time Complexity: -

O(n)

Code: -

#include <stdio.h>

int linear\_search(int a[], int n, int x)

{

    int i, flag = 0, index;

    for (i = 0; i < n; i++)

    {

        if (a[i] == x)

        {

            flag = 1;

            index = i;

        }

    }

    if (flag == 1)

        printf("%d is present in the array at index %d\n", x, index);

    else

        printf("%d is not present in the array \n", x);

    return 0;

}

void main()

{

    int arr[10], i, n = 10, x;

    printf("Enter the array values\n");

    for (i = 0; i < 10; i++)

    {

        scanf("%d", &arr[i]);

    }

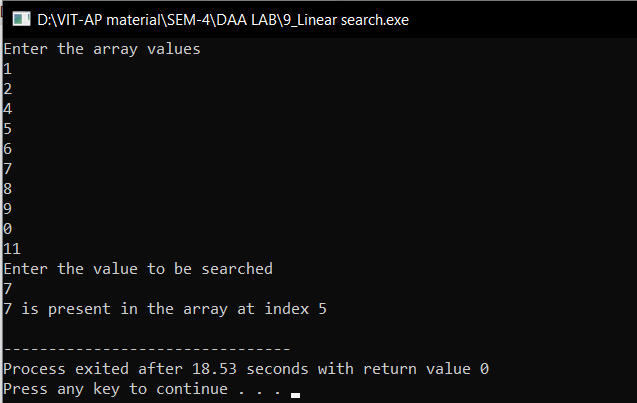
    printf("Enter the value to be searched\n");

    scanf("%d", &x);

    linear\_search(arr, n, x);

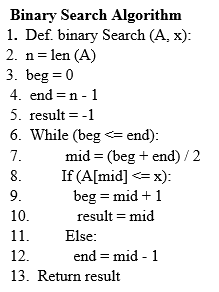
}

* **Output:**

****

1. Binary Search

Algorithm: -



Time Complexity: -

O(n)

Code: -

#include <stdio.h>

int binarySearch(int arr[], int l, int i, int x)

{

    int mid;

    if (l = i)

    {

        if (x = arr[i])

        {

            return i;

        }

        else

            return 0;

    }

    else

    {

        mid = ((i + l) / 2);

        if (x = arr[mid])

            return mid;

        else

        {

            if (x < arr[mid])

                return binarySearch(arr, l, mid - 1, x);

            else

                return binarySearch(arr, mid + 1, i, x);

        }

    }

}

int main()

{

    int arr[] = {2, 3, 4, 40, 10, 5};

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

    int x = 10;

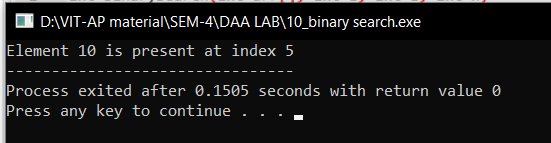
    int result = binarySearch(arr, 0, n - 1, x);

    (result == -1) ? printf("Element is not present in array") : printf("Element %d is present at index %d", x, result);

    return 0;

}

* **Output:**



|  |  |  |
| --- | --- | --- |
| Algorithm | Time Complexity | Space Complexity |
| Linear Search | O(n) | O(n) |
| Binary search | O(logn) | O(n) |