**Experiment No. – 1.1**

**Aim**:

Write a program for the implementation of following searching techniques on linear data structures

1. Linear Search
2. Binary Search
3. **Problem Description:**

We have given array of size n we have to implement linear search and understand its time and space complexity. **Linear Search** is defined as a sequential search algorithm that starts at one end and goes through each element of a list until the desired element is found.

1. **Algorithm:**

# **Algorithm Linear Search:**

Step 1**:** Set i to 1

Step 2: if i > n then go to step 7

Step 3: if A[i] = x then go to step 6

Step 4: Set i to i + 1

Step 5: Go to Step 2

Step 6: Print Element x Found at index i and go to step 8

Step 7: Print element not found

Step 8: Exit

1. **Complexity Analysis**

Time complexity

Best Case: O(1)

Worst Case: O(n)

Average Case: O(n)

**Space Complexity**: O(1)

1. **Pseudo Code**

**Pseudo Code for Linear Search**

procedure linear\_search (list, value)

for each item in the list

if match item == value

return the item's location

end if

end for

end procedure

1. **Source Code (C/C++):**

**Code for Linear Search**

#include <bits/stdc++.h>

using namespace std;

int search(int arr[], int n, int x)

{

    for (int i = 0; i < n; i++)

    {

        if (arr[i] == x)

            return i;

    }

    return -1;

}

int main()

{

    cout << "Name: Saurabh Kumar \nUID: 23MAI10004\n";

    cout << "Enter size of array: ";

    int n;

    cin >> n;

    cout << "Enter the values: \n";

    int \*arr = new int[n];

    for (int i = 0; i < n; i++)

    {

        cin >> arr[i];

    }

    int x;

    cout << "Enter value to find: \n";

    cin >> x;

    int ans = search(arr, n, x);

    if (ans == -1)

        cout << "Element not found";

    else

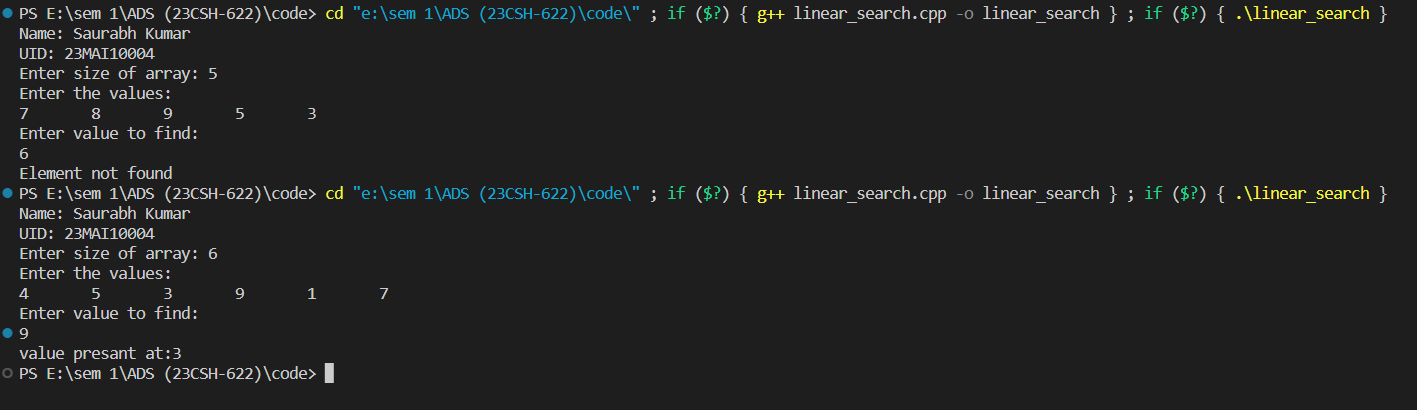
        cout << "value presant at: " << ans;

    return 0;

}

1. **Screenshot of Outputs:**

**Output for linear search:**



1. **Problem Description:**

We have given array of size n we have to implement binary search and understand its time and space complexity, and find which is better searching algorithm in terms of time and space complexity.

1. **Algorithm:**

# **Algorithm Binary Search:**

# BINARY\_SEARCH(A, lower\_bound, upper\_bound, VAL)

Step 1**:** [INITIALIZE] 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

1. **Complexity Analysis**

**Time complexity**

Best Case: O(1)

Worst Case: O(logN)

Average Case: O(logN)

**Space Complexity**: O(1)

1. **Pseudo Code**

**Pseudo Code for Binary Search**

Procedure binary\_search

A ← sorted array

n ← size of array

x ← value to be searched

Set lowerBound = 1

Set upperBound = n

while x not found

if upperBound < lowerBound

EXIT: x does not exists.

set midPoint = lowerBound + ( upperBound - lowerBound ) / 2

if A[midPoint] < x

set lowerBound = midPoint + 1

if A[midPoint] > x

set upperBound = midPoint - 1

if A[midPoint] = x

EXIT: x found at location midPoint

end while

end procedure

1. **Source Code (C/C++):**

**Code for Binary Search**

#include <bits/stdc++.h>

using namespace std;

void binaryser(int arr[], int n, int x)

{

    int low = 0;

    int high = n - 1;

    while (low <= high)

    {

        int mid = (high + low) / 2;

        if (arr[mid] == x)

        {

            cout << "Value present at index: " << mid;

            return;

        }

        else if (arr[mid] < x)

            low = mid + 1;

        else

            high = mid - 1;

    }

    cout << -1;

}

int main()

{

    cout << "Name: Saurabh Kumar \nUID: 23MAI10004\n";

    int n;

    cout << "Enter length of an array: ";

    cin >> n;

    cout << "Enter array elements" << endl;

    int arr[n];

    for (int i = 0; i < n; i++)

    {

        cin >> arr[i];

    }

    int x;

    cout << "Enter value to find: ";

    cin >> x;

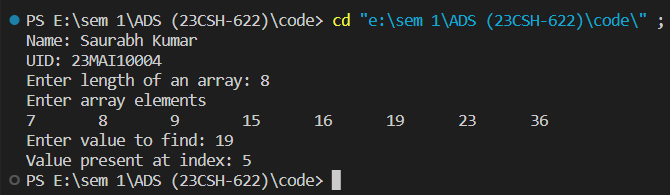
    binaryser(arr, n, x);

    return 0;

}

1. **Screenshot of Outputs:**

**Output for Binary search:**

****

1. **Learning Outcomes**
2. Learnt about how to write code in c++ and take array as input.
3. Learnt about linear search and binary search.
4. Learnt about time and space complexity of linear search.
5. Learnt about time and space complexity of binary search.
6. Learnt which is better searching algorithm in terms of space and time complexity.