Experiment - 3

AIM;

To implement the following search algorithm and analyze its time complexity

→ Binary Search

→ Linear Search

Software Used: VS Code

Theory

→ Linear Search

Algorithm

Linear Search [A, X]

i = 1

if i > n

return n

if A[i] == x

return the position of x

End

Time Complexity Analysis

$$\frac{\sum_{i=1}^{n+1} O(i)}{n+1}$$

$$\Rightarrow \frac{(O(n+1) * (n+2)/2)}{n+1}$$

$$= O(n)$$

Case 1 : Best Case

The number of operations in the Best Case is constant

→ Time complexity is $O(1)$

Case 2 : Worst Case

For linear search, the worst case happens when the element to be searched is not present in the array.

→ Time complexity is $O(n)$

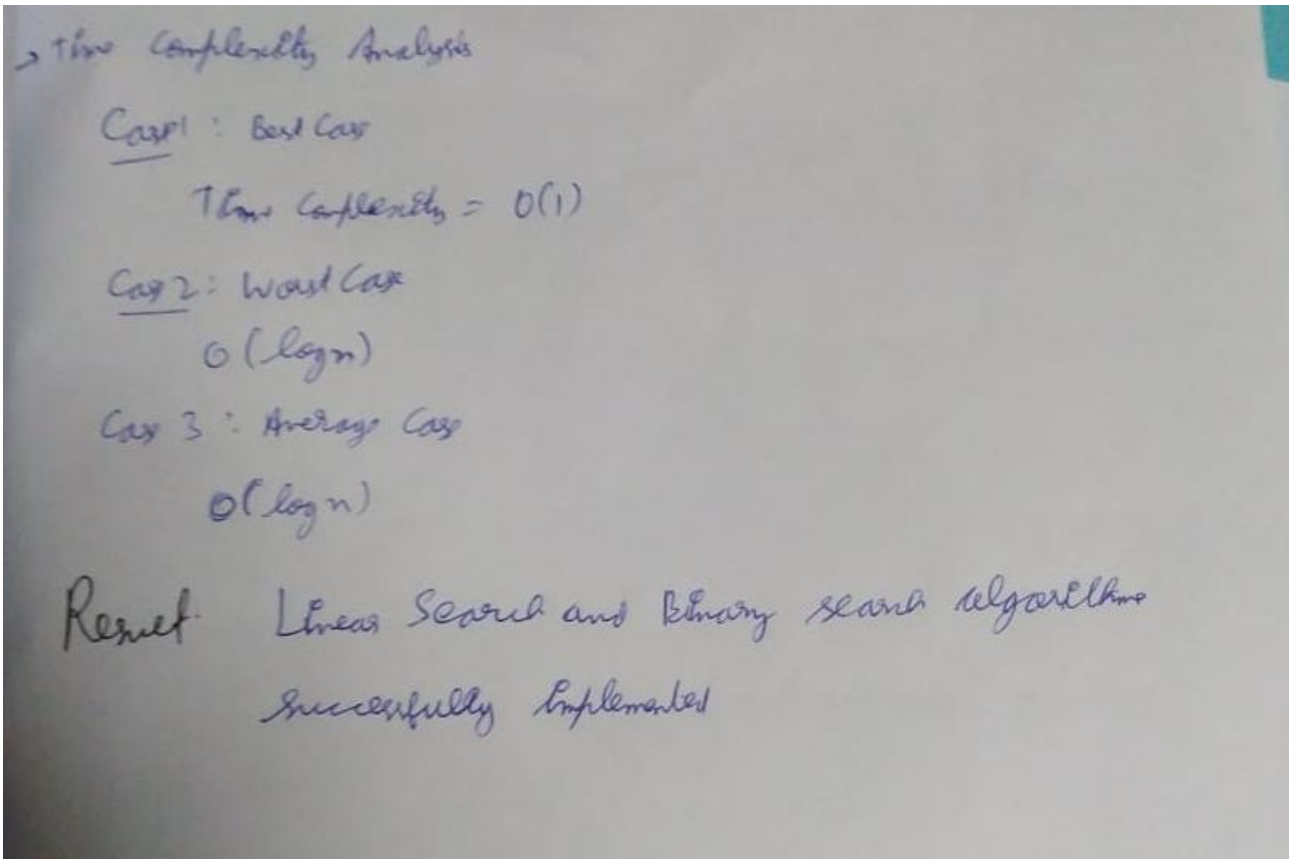
Case 3: Average Case

Taking all possible input the time complexity is $O(n)$

→ Binary Search

Algorithm

```
Binary Search ( A, n, x)
  lower bound = 1
  upper bound = n
  while x not found or upperbound , lower bound
        if upperbound > lowerbound
        set mid point = lowerbound + ( upperbound - lowerbound )/2
  if A[mid Point] < x
     set lowerbound = mid point +1
  if A[mid point] > x
     set upperbound = midpoint - 1
  if A[mid point] = = x
     return
```



Code (Linear Search)

```
#include <iostream>
#include <bits/stdc++.h>
#include <chrono>
using namespace std;
using namespace std::chrono;
int main()
{
    int key, n, flag = 0;
    cout << "Enter Number of elements: ";
    cin >> n;
    int arr[n];
    cout << "Enter the Elements: ";
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
    }
    cout << "Enter Key: ";
```

```
    cin >> key;
    auto start = steady_clock::now();
    for (int i = 0; i < n; i++)
    {
        if (key == arr[i])
        {
            cout << "Key found at index: " << i << endl;
            flag = 1;
            break;
        }
    }
    if (flag == 0)
    {
        cout << "key not found" << endl;
    }
    auto stop = steady_clock::now();
    auto duration = duration_cast<nanoseconds>(stop - start);
    cout << "\nTime taken by function: " << duration.count() << " nanoseconds" <<
 endl;
}
```

Output (Linear Search)

```
Enter Number of elements: 5
Enter the Elements: 1 2 3 4 5
Enter Key: 4
Key found at index: 3

Time taken by function: 1916300 nanoseconds
```

Code (Binary Search)

```cpp
#include <bits/stdc++.h>
#include <chrono>
using namespace std;
using namespace std::chrono;
int binary_search(int arr[], int first, int last, int key)
{
    if (last >= first)
    {
        int mid = first + (last - first) / 2;
        if (arr[mid] == key)
            return mid;
        if (arr[mid] > key)
            return binary_search(arr, first, mid - 1, key);
        return binary_search(arr, mid + 1, last, key);
    }
    return -1;
}
int main()
{
    int n, mid, last, first = 0, key, flag = 0, result;
    cout << "Enter Number of Elements: ";
    cin >> last;
    int arr[last];
    cout << "Enter elements in Sorted order: ";
    for (int i = 0; i < last; i++)
    {
        cin >> arr[i];
    }
    cout << "Enter Key: ";
    cin >> key;
    auto start = steady_clock::now();
    result = binary_search(arr, first, last, key);
    if (result != -1)
    {
        cout << "Key Found at index: " << result << endl;
    }
    else
    {
        cout << "Key not Found" << endl;
    }
    auto stop = steady_clock::now();
    auto duration = duration_cast<nanoseconds>(stop - start);
```

```
    cout << "\nTime taken by function: " << duration.count() << " nanoseconds" <<
 endl;
}
```

Output (Binary Search)

```
Enter Number of Elements: 5
Enter elements in Sorted order: 1 2 3 4 5
Enter Key: 4
Key Found at index: 3

Time taken by function: 1874900 nanoseconds
```