CS263: Design and Analysis of Algorithm Laboratory

Name : Hritik Kumar
Roll No. : 202051088

Task – 1

- **Linear Search-**

  Algorithm

```
void LinearSearch(int arr[], int n, int k)
{
    int flag = 0;
    for(int i=0; i<n; i++){
        if(arr[i] == k){
            cout<<"Found!!!"<<endl;
            flag = 1;
            break;
        }
    }

        if(flag == 0)
            cout<<"Not Found!!!"<<endl;
}
```

Time complexity: -

Initialization of **x** in the loop---------- 1 time

Comparison of **x** in the loop ---------- n + 1 times

Increment of **x** in the loop ---------- n times

Conditional comparison ---------- n times

Return if condition is true ---------- n times

Final return ---------- 1 time

Total steps 4n + 3 so **Time complexity = O(n)**

- **Binary Search-**

Algorithm:-

```
void BinarySearch(int arr[], int l, int r, int x)
{
    int flag = 0;
    if (r >= l) {
        int mid = l + (r - l) / 2;
        if (arr[mid] == x){
            flag = 1;
            cout<<"Found!!!"<<endl;
            return;
        }

        if (arr[mid] > x)
            return BinarySearch(arr, l, mid - 1, x);

        return BinarySearch(arr, mid + 1, r, x);
    }

    if(flag ==0)
        cout<<"Not Found!!!"<<endl;
}
```

Time complexity: -

Initialisation of left ---------- 1 time

Initialisation of right ---------- 1 time

Comparison in loop ---------- log(n) +1 times

Initialisation of mid ---------- log(n) times

Conditional statement 1 comparison ---------- log(n) times

Conditional statement 2 comparison ---------- log(n) times

Conditional statement 3 comparison ---------- log(n) times

Final return ---------- 1 time

Total steps 5log(n) +4 so **Time complexity = O(log(n))**

- **Bubble sort-**

Algorithm

```
void BubbleSort(int arr[], int n)
{
    for(int i=1; i<n; i++){
        for(int j=0; j<n-i; j++){
            if(arr[j]>arr[j+1]){
                arr[j] = arr[j]^arr[j+1];
                arr[j+1] = arr[j]^arr[j+1];
                arr[j] = arr[j]^arr[j+1];
            }
        }

    }
}
```

Time complexity: -

Initialisation of i in the for loop ---------- 1 time

Comparison of i in the for loop ---------- n times

Increment of i in the for loop ---------- n-1 times

Initialization of j in the for loop ----------n time

Comparison of j in the for loop ---------- n^2 times

Increment of j in the for loop ---------- n^2 -n times

Comparisons of arr[j] and arr[j+1] ----------n^2 times

Swapping arr[j] and arr[j+1] ---------- $3*n^2$ times

Final return ---------- 1 time

Total steps $6n^2 + 2n + 1$ so **Time complexity = $O(n^2)$**

- **Selection sort-**

Algorithm

```
void SelectionSort(int arr[], int n)
{
    for(int i=0; i<n-1; i++){
        int mini = i;
        for(int j=i+1; j<n; j++){
            if(arr[mini]>arr[j]){
                mini = j;
            }
        }
        if(mini == i)
            continue;
        arr[i] = arr[i]^arr[mini];
        arr[mini] = arr[i]^arr[mini];
        arr[i] = arr[i]^arr[mini];
    }
}
```

Time complexity: -

Initialization of i in the for loop ---------- 1 time

Comparison of i in the for loop ---------- n + 1 times

Increment of i in the for loop ---------- n times

Initialization of mini ---------- n times

Initialization of j in the for loop ---------- n time

Comparison of j in the for loop ---------- n^2 times

Increment of j in the for loop ---------- n^2 -n times

Comparisons of arr[j] and arr[min] ---------- n^2 times

Assignment of min = j ---------- n^2 times

Swapping arr[i] and arr[min] ---------- 3n times

Final return ---------- 1 time

Total steps $4n^2 + 6n + 3$ so **Time complexity = $O(n^2)$**

- # Insertion sort-

Algorithm

```
void InsertionSort(int arr[], int n)
{
    int temp;
    int j;
    for(int i=1; i<n; i++){
        temp = arr[i];
        for(j= i-1; j>=0 && arr[j]>temp; j--){
            arr[j+1] = arr[j];
        }

        arr[j+1] = temp;
    }
}
```

Time complexity: -

Initialization of i in the for loop ---------- 1 time

Comparison of i in the for loop ---------- n times

Increment of i in the for loop ---------- n-1 times

Initialization of temp ---------- n times

Initialization of h ---------- n times

Comparisons in the while loop ---------- n^2 +n times

Assignment of arr[h+1] =arr[h] ---------- n^2 times

Decrement of h ---------- n^2 times

Assignment of arr[h+1] =temp ---------- n times

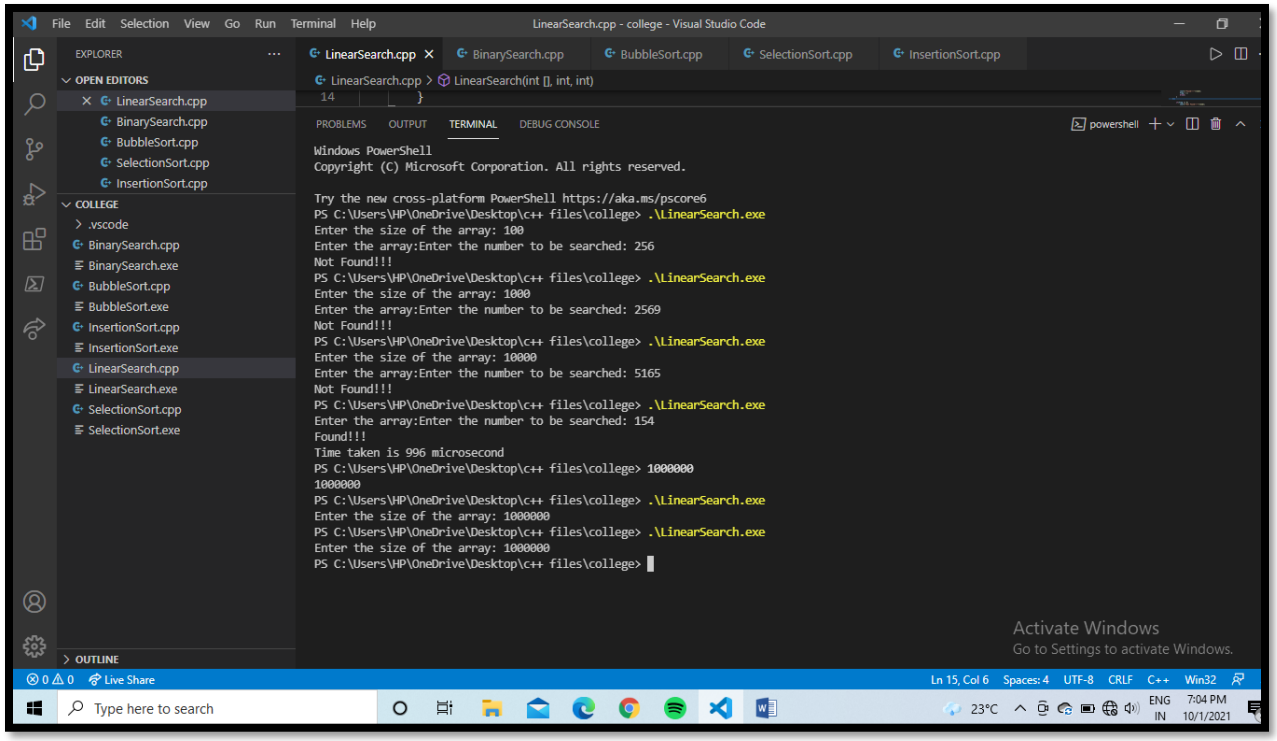
Final return ---------- 1 time

Total steps 3 n^2 + 6n + 1 so **Time complexity = $O(n^2)$**

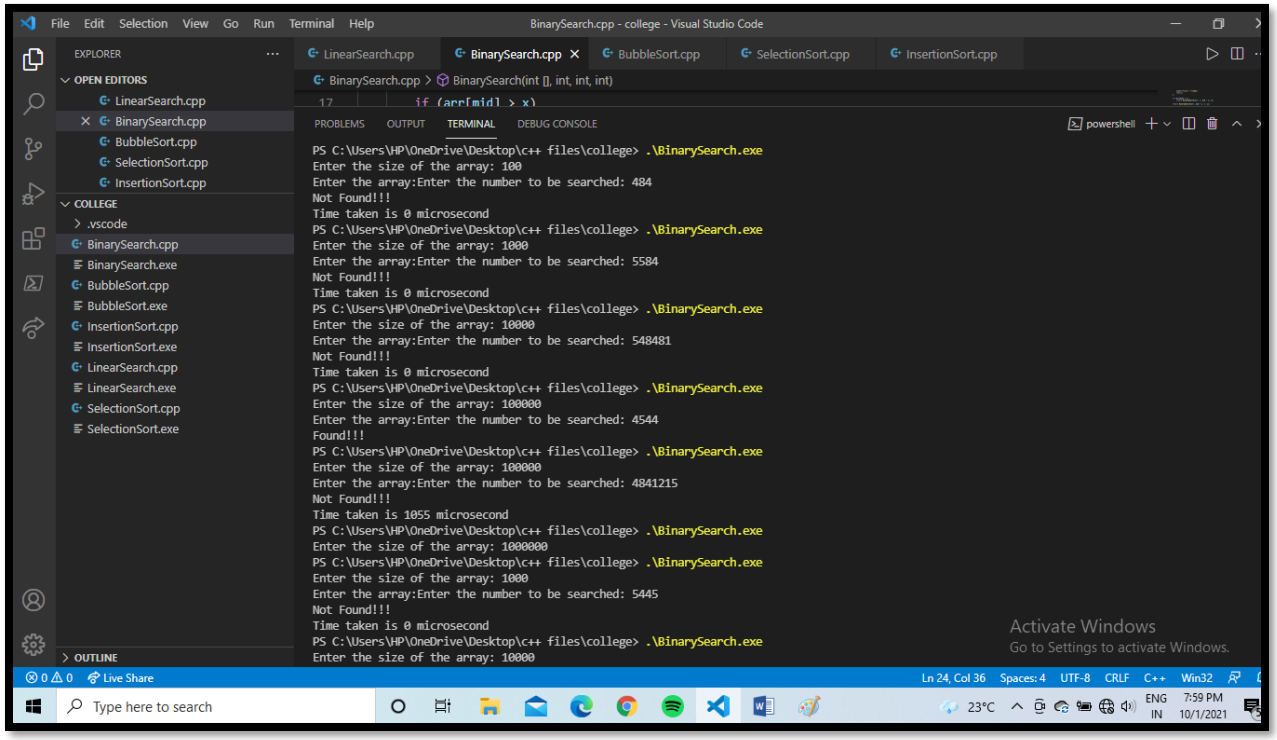
	Best Case	Average Case	Worst Case
Linear Search	O(1)	O(n)	O(n)
Binary Search	O(1)	O(log(n))	O(log(n))
Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	O(n)	$O(n^2)$	$O(n^2)$

Task -2

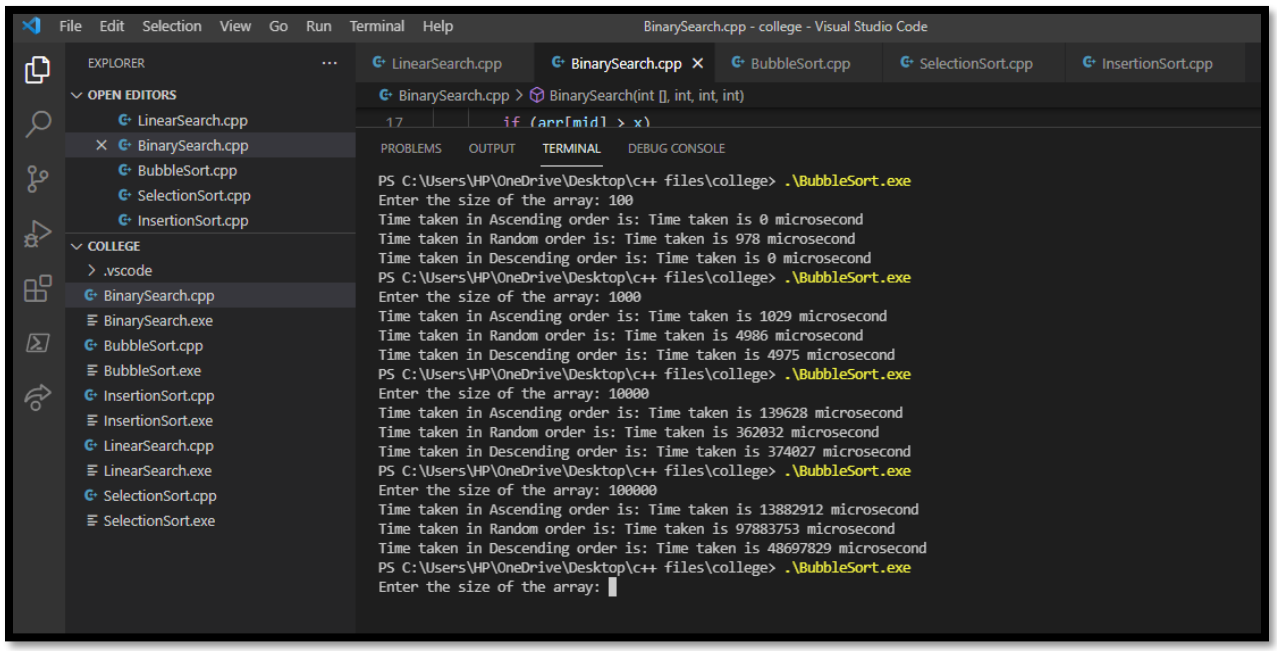
Linear Search:-



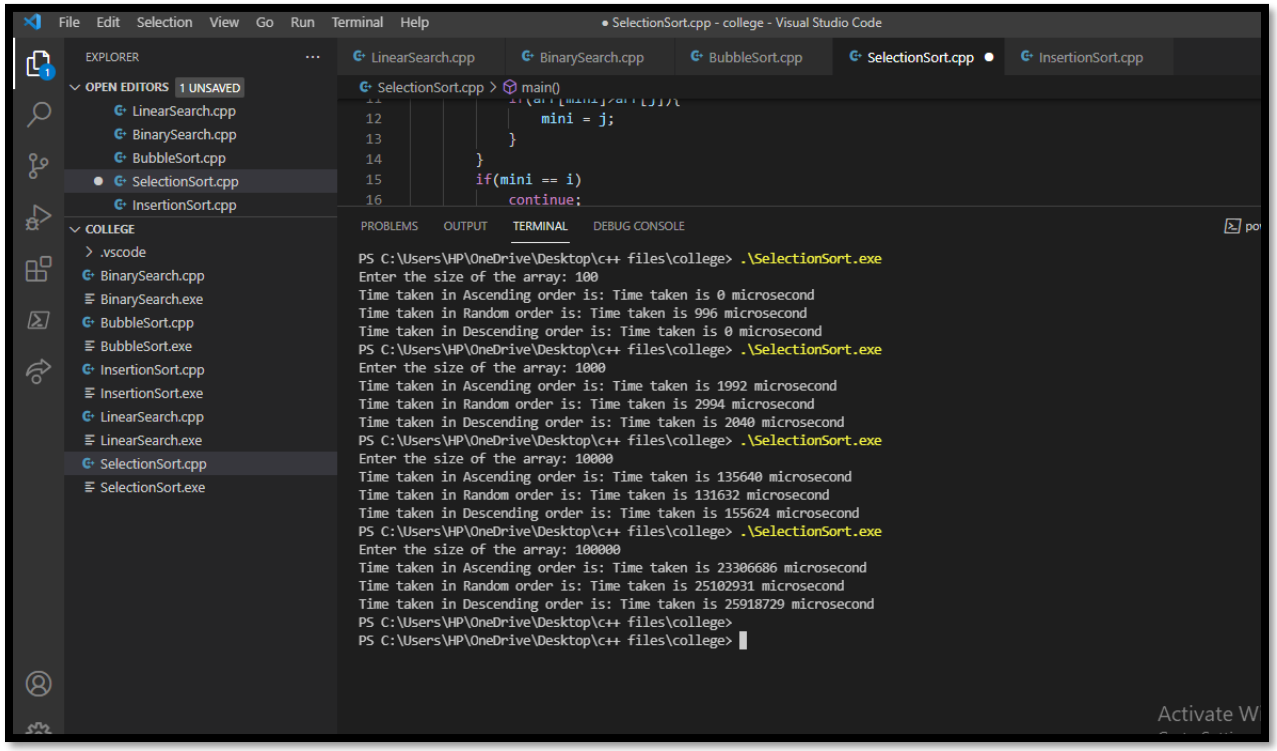
Binary Search:-



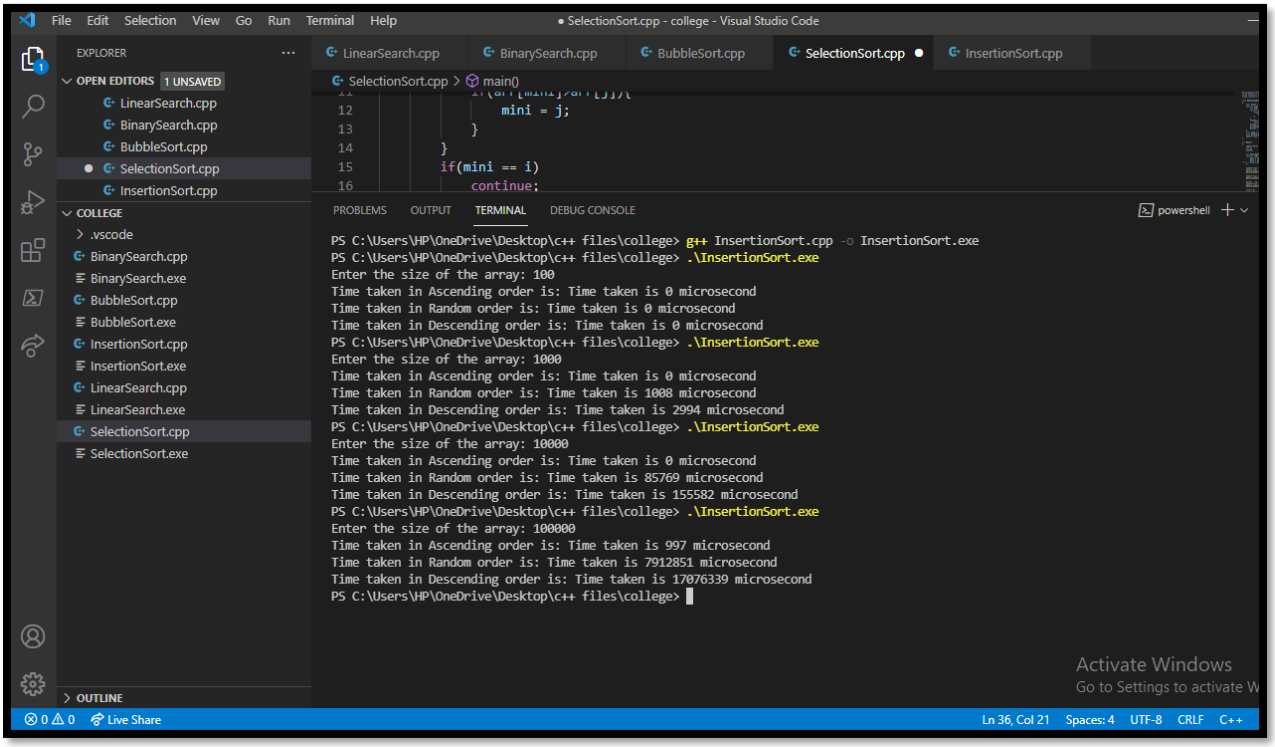
Bubble Sort:-



Selection Sort:-

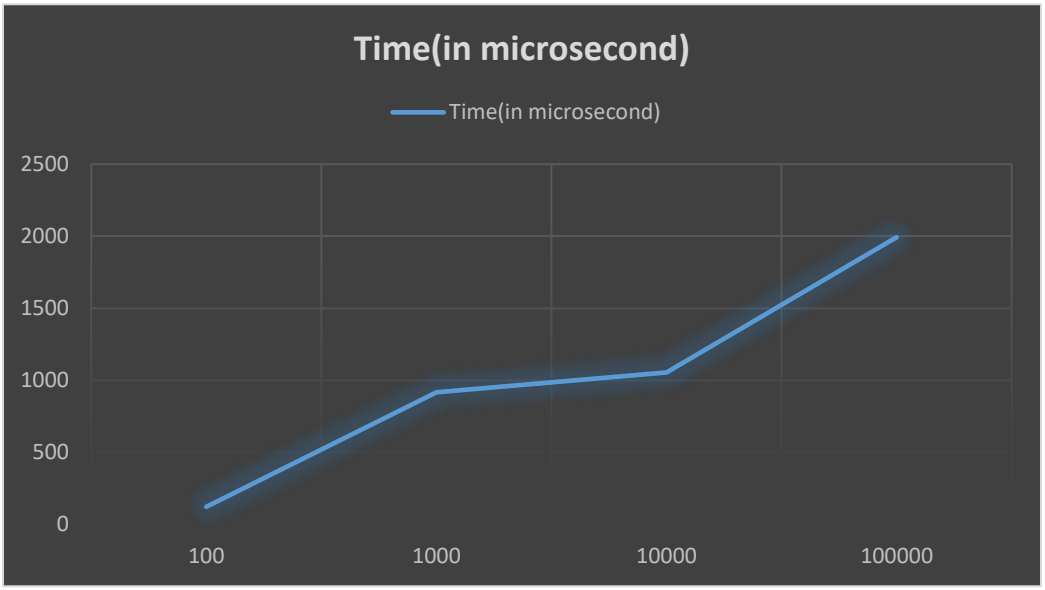


Insertion Sort:-

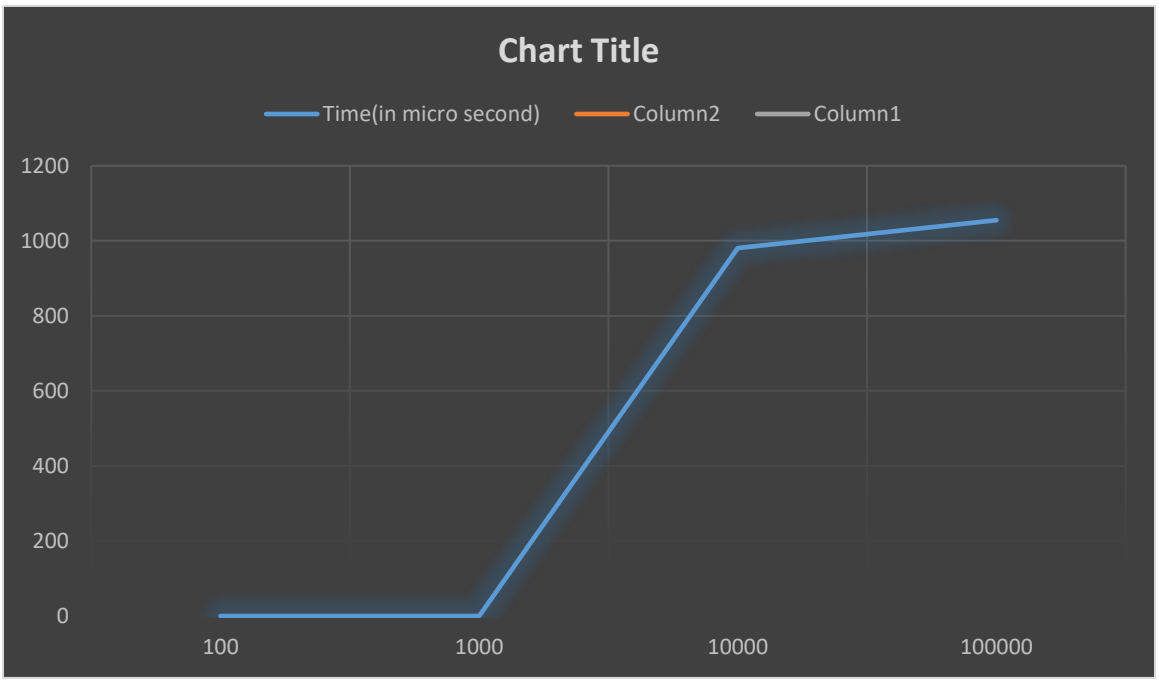


Task – 3

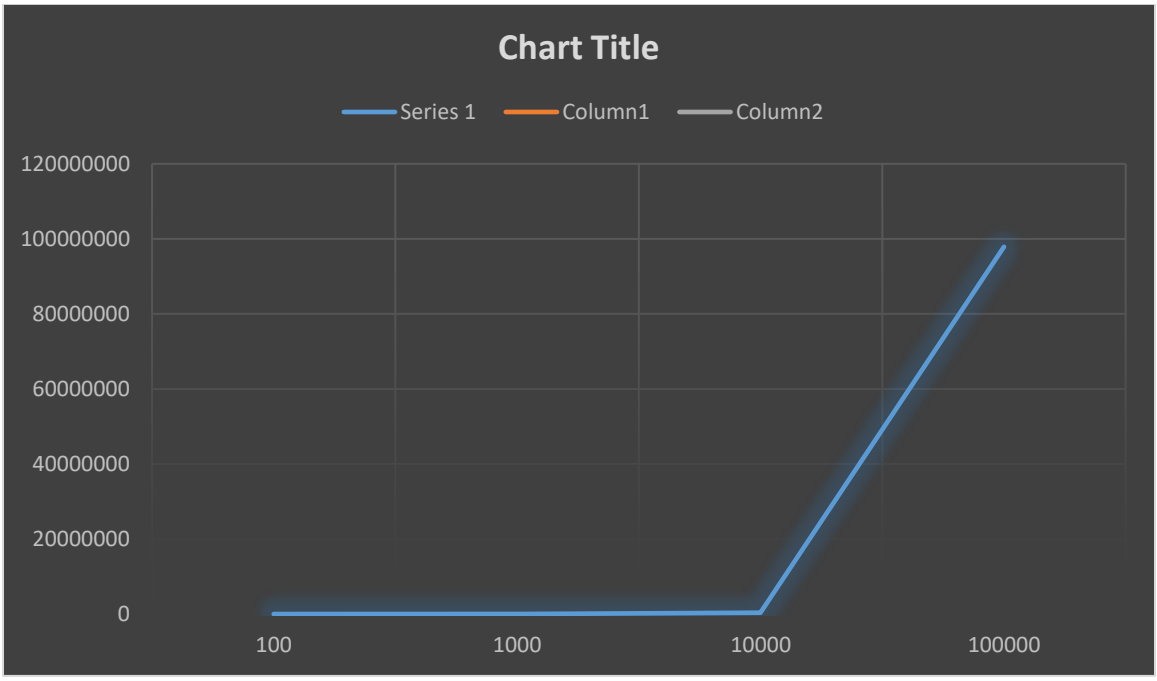
Linear Search:-



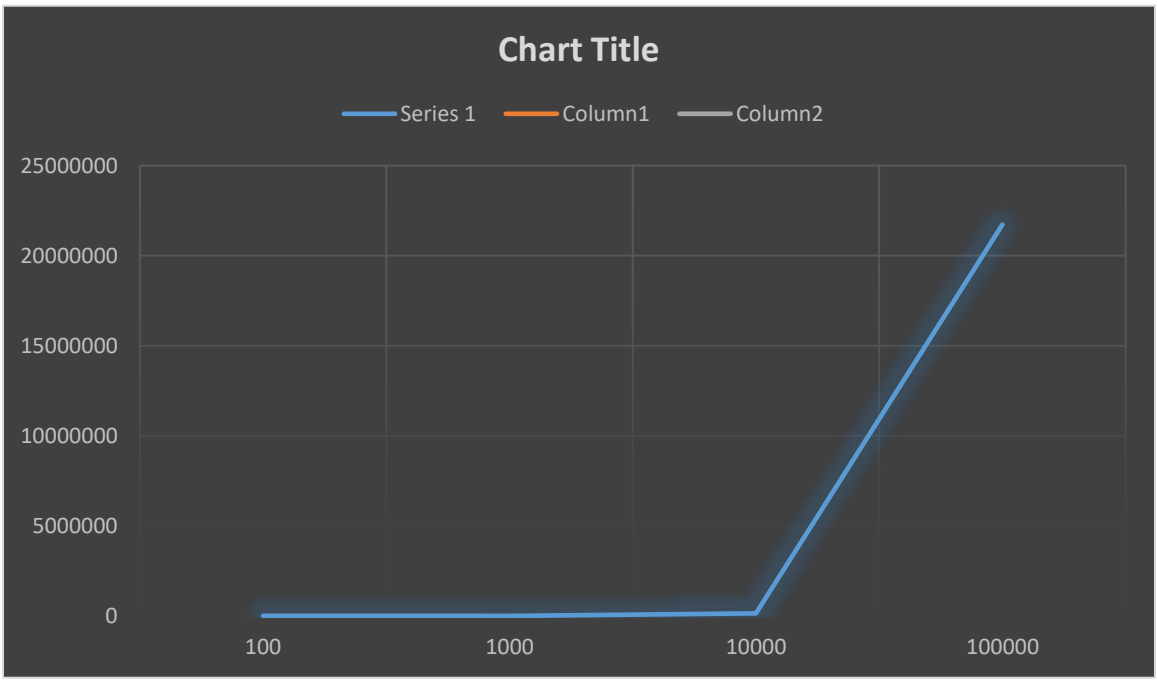
Binary Search:-



Bubble Sort:-



Selection Sort:-



Insertion Sort:-