NAME: GAURAV YADAV

ROLL NUMBER: 21075034

E-MAIL: gaurav.yadav.cse21@itbhu.ac.in

Q1 Given an array of n distinct elements. Find the number of swaps required to sort the array in strictly increasing order using

a) **Insertion Sort**

```cpp
#include <bits\stdc++.h>
using namespace std;

int insertionSort(int a[], int n)
{
    int num_swaps = 0;
    int i, j, temp;
    for (int i = 1; i < n; i++)
    {
        temp = a[i];
        j = i - 1;
        while (j >= 0 && temp <= a[j])
        {
            a[j + 1] = a[j];
            j--;
            num_swaps++;
        }
        a[j + 1] = temp;
    }
    return num_swaps;
}

int main()
{
    int n;
    cout << "Enter the number of elments in the array\n";
    cin >> n;
    int arr[n];
    cout << "Now enter the elements of the array\n";
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
    }
    int ans = insertionSort(arr, n);
    cout << "The sorted array looks like: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << "\n";
    cout << "The number of swaps required is: " << ans;
    return 0;
}
```

GAURAV YADAV

21075034

Output:

```
PS C:\Users\Gaurav\Programming\practice> cd "d:\Gaurav\Documents\Alg
Enter the number of elments in the array
5
Now enter the elements of the array
5 1 2 4 3
The sorted array looks like: 1 2 3 4 5
The number of swaps required is: 5
PS D:\Gaurav\Documents\Algorithm_lab_questions> _
```

Time Complexity: The time complexity will be equal to the time complexity of the insertion sorting which is $O(n^2)$.

b) Bubble Sort

```cpp
#include <bits\stdc++.h>
using namespace std;

int main()
{
    int n;
    cout << "Enter the number of elements: ";
    cin >> n;
    int arr[n];
    cout << "Enter the elements of the array:\n";
    for (int i = 0; i < n; i++)
        cin >> arr[i];
    int num_swaps = 0;
    int flag = 0;
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = i + 1; j < n; j++)
        {
            if (arr[i] > arr[j])
            {
                swap(arr[i], arr[j]);
                num_swaps++;
                flag = -1;
            }
        }
        if (flag == 0)
            break;
        else
            flag = 0;
    }
    cout << "The sorted array looks like: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << "\n";
    cout << "the number of swaps done is: " << num_swaps;
    return 0;
}
```

GAURAV YADAV

21075034

Output:

```
PS D:\Gaurav\Documents\Algorithm_lab_questions> cd "d:\Gaurav\Documents\Algorith
Enter the number of elements: 6
Enter the elements of the array:
5 2 4 6 1 3
The sorted array looks like: 1 2 3 4 5 6
the number of swaps done is: 9
PS D:\Gaurav\Documents\Algorithm_lab_questions> _
```

Time Complexity: The time complexity of this algorithm is based on the time complexity of bubble sort algorithm which is $O(n^2)$.

Q2) Given arrival and departure times of all trains that reach a railway station. Find the minimum number of platforms required for the railway station so that no train is kept waiting. Consider that all the trains arrive on the same day and leave on the same day. Arrival and departure time can never be the same for a train but we can have arrival time of one train equal to departure time of the other. At any given instance of time, same platform cannot be used for both departure of a train and arrival of another train. In such cases, we need different platforms.

```cpp
#include <bits/stdc++.h>
using namespace std;

int f(int arr[], int dep[], int n)
{
    int i = 0, j = 0, ans = 0, cnt = 0;
    while (i < n && j < n)
    {
        if (arr[i] <= dep[j])
        {
            i++;
            cnt++;
        }
        else
        {
            j++;
            cnt--;
        }
        ans = max(ans, cnt);
    }
    return ans;
}

int main()
{
    int n;
    cin >> n;
    int arr[n], dep[n];
    for (int i = 0; i < n; i++)
        cin >> arr[i];
    for (int i = 0; i < n; i++)
        cin >> dep[i];
    sort(arr, arr + n);
    sort(dep, dep + n);
    cout << "The required number of stations are: " << f(arr, dep, n);
    return 0;
}
```

GAURAV YADAV

21075034

Output:

```
PS D:\Gaurav\Documents\Algorithm_lab_questions> cd "d:\Gaurav\Documents\Alg
6
0900 0940 0950 1100 1500 1800
0910 1200 1120 1130 1900 2000
The required number of stations are: 3
PS D:\Gaurav\Documents\Algorithm_lab_questions> _
```

Time Complexity: Since we sorted the array which has $O(nlogn)$ and we also traversed the array once which has O(n). Hence our final complexity is O(nlogn).

Q3) Given an array of integers. Find the Inversion Count in the array.

```cpp
#include<bits/stdc++.h>
using namespace std;

int merge(int arr[], int left, int mid, int right) {
  int i = left, j = mid, k = 0;
  int count = 0;
  int temp[(right - left + 1)];

  while ((i < mid) && (j <= right)) {
    if (arr[i] <= arr[j]) {
      temp[k] = arr[i];
      ++k;
      ++i;
    } else {
      temp[k] = arr[j];
      count += (mid - i);
      ++k;
      ++j;
    }
  }

  while (i < mid) {
    temp[k] = arr[i];
    ++k;
    ++i;
  }

  while (j <= right) {
    temp[k] = arr[j];
    ++k;
    ++j;
  }

  for (i = left, k = 0; i <= right; i++, k++) {
    arr[i] = temp[k];
  }

  return count;
}
int mergeSort(int arr[], int left, int right) {
  int count = 0;
```

```
42
43      if (right > left) {
44        int mid = (right + left) / 2;
45        count = mergeSort(arr, left, mid);
46        count += mergeSort(arr, mid + 1, right);
47        count += merge(arr, left, mid + 1, right);
48      }
49      return count;
50    }
51
52    int main(){
53
54    int n;
55    cin>>n;
56    int arr[n];
57    for(int i=0;i<n;i++)
58    cin>>arr[i];
59    int ans=mergeSort(arr,0,n-1);
60    cout<<ans;
61    return 0;
62
63    }
64
```

Output:

```
PS D:\Gaurav\Documents\Algorithm_lab_questions> cd "d:\Gaurav\Documents\Algori
5
2 4 1 3 5
3
PS D:\Gaurav\Documents\Algorithm_lab_questions> _
```

Time Complexity: The time complexity is totally dependent on the merge sorting algorithm which has the time complexity of O(nlogn).