

Experiment-1.4

Aim of the Experiment :

Write a program to implement the Insertion Sort along with its complexity analysis.

1. Problem Description :

Insertion Sort: Insertion sort is a sorting algorithm that places an unsorted element at its suitable place in each iteration. We assume that the first element is already sorted then, we select an unsorted element. If the unsorted element is greater than the element in hand, it is placed on the right otherwise, to the left. In the same way, other unsorted elements are taken and put in their right place.

2. Algorithm :

Step 1: Take size of array as input from user.

Step 2: Enter the elements of array from user.

Step 3: Assume first element as already sorted and pick the next element

Step 4: Compare with all elements in the sorted sub-list

Step 5: Shift all the elements in the sorted sub-list that is greater than the selected element to be sorted

Step 6: Insert the selected element at right place.

Step 7: Repeat until list is sorted

Step 8: Exit.

3. Complexity Analysis:

Time Complexity:

In Selection Sort, 1 comparison will be done in the 1st pass, 2 comparisons in 2nd pass, till (n-1) comparisons in (n-1)th pass. So the total number of comparisons will be,

$$= 1 + 2 + 3 + \dots + (n-3) + (n-2) + (n-1)$$

$$= n(n-1)/2$$

i.e. $O(n^2)$

1) Best Case: $O(N)$. When the elements of the array are already sorted.

2) Worst Case: $O(N^2)$. When the elements of the array are in decreasing order.

3) Average Case: $O(N^2)$

Space Complexity: $O(1)$. As no extra space is used while sorting.

4. Pseudo Code :

procedure insertionSort(A : array of items)

 int holePosition

 int valueToInsert

 for i = 1 to length(A) inclusive do:

 /* select value to be inserted */

 valueToInsert = A[i]

```
holePosition = i

/*locate hole position for the element to be inserted */

while holePosition > 0 and A[holePosition-1] > valueToInsert do:

    A[holePosition] = A[holePosition-1]

    holePosition = holePosition - 1

end while

/* insert the number at hole position */

A[holePosition] = valueToInsert

end for

end procedure
```

5. Source Code for Experiment :

```
#include<iostream>

using namespace std;

void printArray(int arr[], int n){
    for(int i=0; i<n; i++){
        cout<<arr[i]<<" ";
    }
}
```



```
void insertionSort(int arr[], int n){  
    cout<<endl;  
    for(int i=1; i<n; i++){  
        int temp=arr[i];  
        int j=i-1;  
        for(; j>=0; j--){  
            if(arr[j]>temp){  
                arr[j+1]=arr[j];  
            }  
            else{  
                break;  
            }  
        }  
        arr[j+1]=temp;  
  
        cout<<"Round "<<i<<": ";  
        printArray(arr,n);  
        cout<<endl;  
    }  
    cout<<endl;  
}
```

```
int main(){

    cout<<"\nExperiment-1.4 (Ashish Kumar, 23MAI10008)"<<endl<<endl;

    cout<<"Performing Insertion Sort ..."<<endl;


    int n;

    int arr[100];

    cout<<"Enter size of array: ";

    cin>>n;


    cout<<"Enter elements of array: ";

    for(int i=0; i<n; i++){

        cin>>arr[i];

    }


    cout<<"Original Array is: ";

    printArray(arr,n);

    cout<<endl;


    // Insertion Sort Function

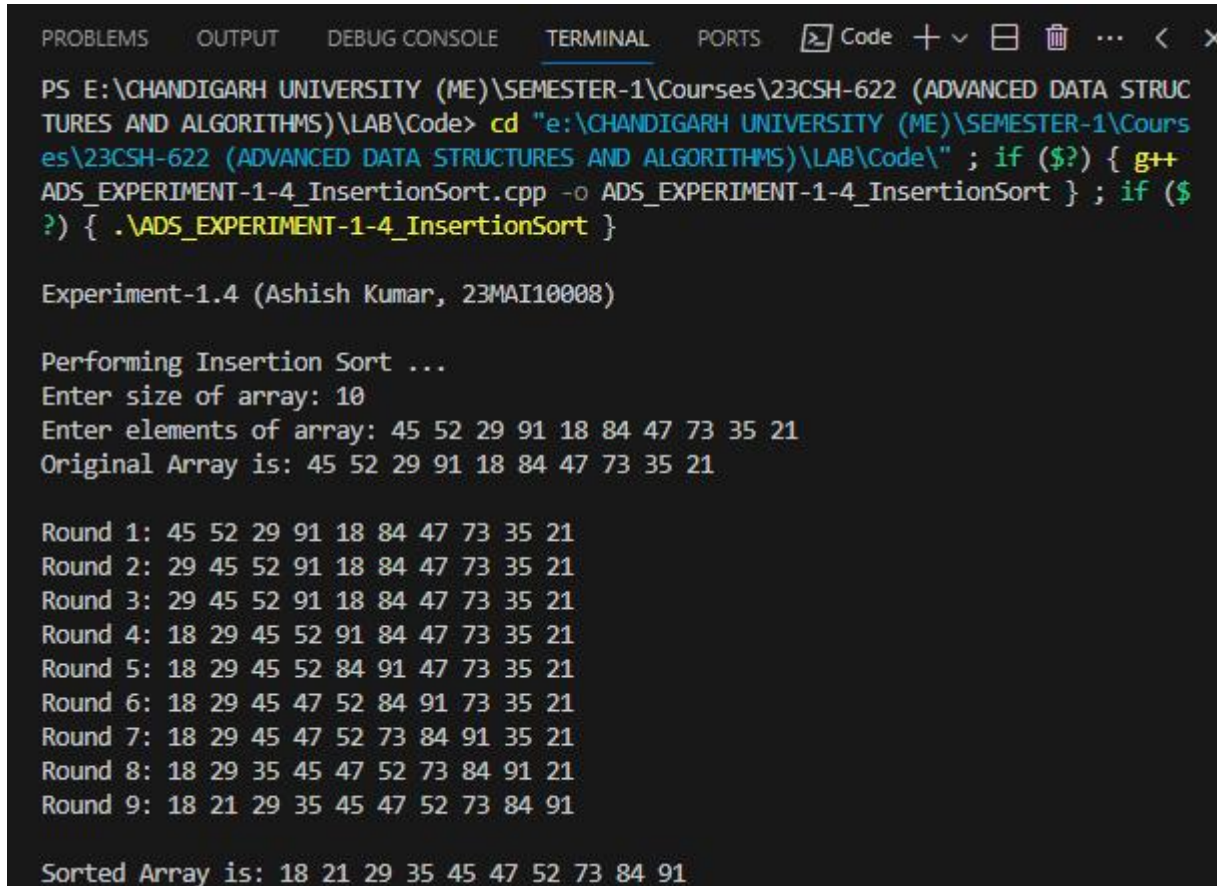
    insertionSort(arr,n);


    cout<<"Sorted Array is: ";

    printArray(arr,n);

}
```

6. Result/Output :



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code + v [ ] [X] ... < X
PS E:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Courses\23CSH-622 (ADVANCED DATA STRUCTURES AND ALGORITHMS)\LAB\Code> cd "e:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Courses\23CSH-622 (ADVANCED DATA STRUCTURES AND ALGORITHMS)\LAB\Code\" ; if ($?) { g++ ADS_EXPERIMENT-1-4_InsertionSort.cpp -o ADS_EXPERIMENT-1-4_InsertionSort } ; if ($?) { .\ADS_EXPERIMENT-1-4_InsertionSort }

Experiment-1.4 (Ashish Kumar, 23MAI10008)

Performing Insertion Sort ...
Enter size of array: 10
Enter elements of array: 45 52 29 91 18 84 47 73 35 21
Original Array is: 45 52 29 91 18 84 47 73 35 21

Round 1: 45 52 29 91 18 84 47 73 35 21
Round 2: 29 45 52 91 18 84 47 73 35 21
Round 3: 29 45 52 91 18 84 47 73 35 21
Round 4: 18 29 45 52 91 84 47 73 35 21
Round 5: 18 29 45 52 84 91 47 73 35 21
Round 6: 18 29 45 47 52 84 91 73 35 21
Round 7: 18 29 45 47 52 73 84 91 35 21
Round 8: 18 29 35 45 47 52 73 84 91 21
Round 9: 18 21 29 35 45 47 52 73 84 91

Sorted Array is: 18 21 29 35 45 47 52 73 84 91
```

Learning outcomes (What I have learnt):

1. I learnt about how to sort the elements of an array.
2. I learnt about how to perform insertion sort on an array.
3. I learnt about difference between selection and insertion sort.
4. I learnt about how to traverse the array elements.
5. I learnt about time and space complexity of insertion sort.