Experiment-1.2

Aim of the Experiment :

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

1. Problem Description :

Bubble Sort: Bubble sort is a sorting algorithm that works by repeatedly comparing adjacent elements in an array and swapping them if they are in the wrong order. The algorithm starts at the beginning of the array and compares the first two elements. If the first element is greater than the second element, the two elements are swapped. The algorithm then moves on to the next two elements and repeats the process. This continues until the algorithm reaches the end of the array.

2. Algorithm :

```
Step 1: Take size of array as input from user.
Step 2: Enter the elements of array from user.
Step 3: Traverse through all the array elements.
Step 4: Traverse the array from 0 to n-i-1
Step 5: Compare the element at the index i with the element at index i + 1.
Step 6: if(arr[j]>arr[j+1]){
                swap(arr[j],arr[j+1]);
        }
Step 7: Print all the elements of array.
Step 8: Exit.
```

3. Complexity Analysis:

Time Complexity:

In Bubble Sort, (n-1) comparisons will be done in the 1st pass, (n-2) comparisons in 2nd pass, (n-3) comparisons in 3rd pass and so on. So the total number of comparisons will be,

= (n-1) + (n-2) + (n-3) + ... + 3 + 2 + 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 bubbleSort( list : array of items )
    loop = list.count;
    for i = 0 to loop-1 do:
        swapped = false
        for j = 0 to loop-1 do:
            /* compare the adjacent elements */
            if list[j] > list[j+1] then
                /* swap them */
```

```
                    swap( list[j], list[j+1] )
                     swapped = true
                end if
            end for
            /*if no number was swapped that means
            array is sorted now, break the loop.*/
            if(not swapped) then
                break
            end if
        end for
end procedure return list
```

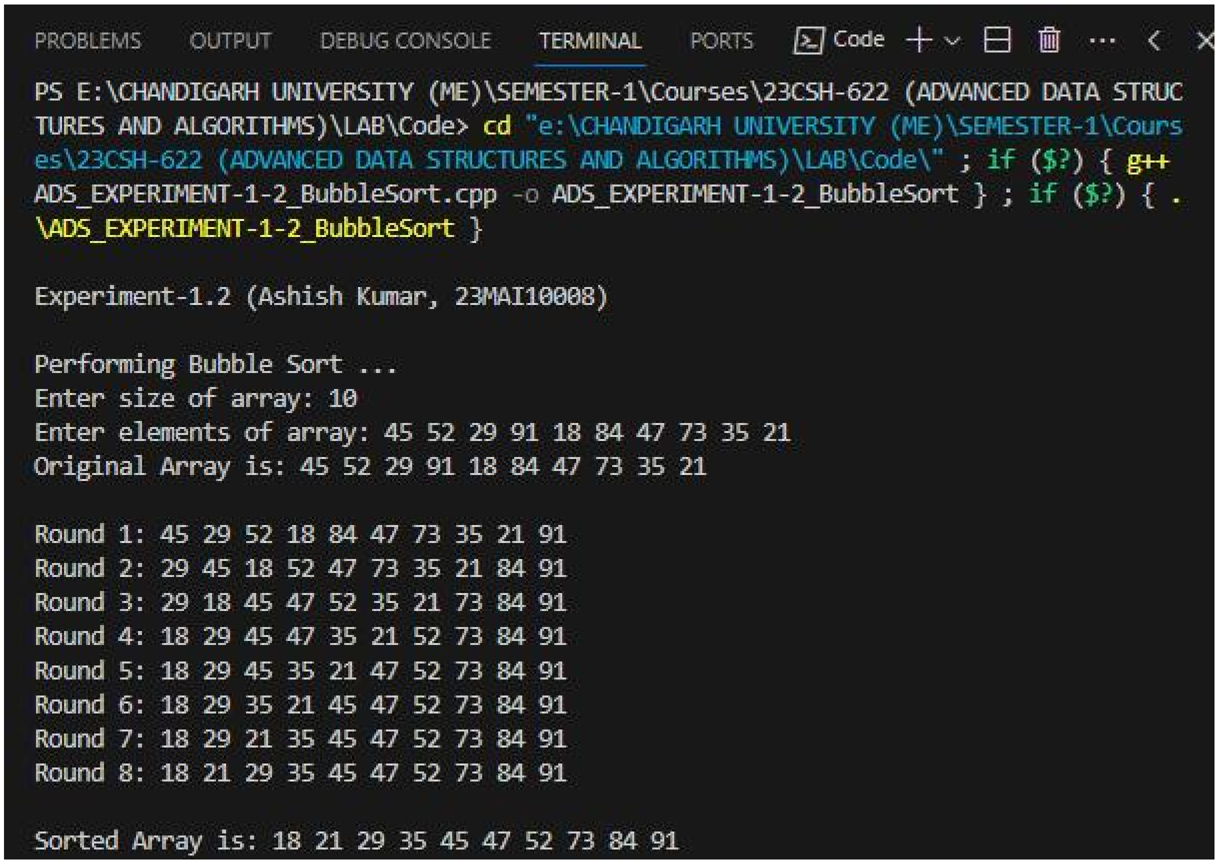
5. Source Code for Experiment :

```
#include<iostream>
using namespace std;

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

```
void bubbleSort(int arr[], int n){
    cout<<endl;
    for(int i=1; i<n; i++){
        bool swapped=false;
        for(int j=0; j<n-i; j++){
            if(arr[j]>arr[j+1]){
                swap(arr[j],arr[j+1]);
                swapped=true;
            }
        }
        if(swapped==false){
            break;
        }

        cout<<"Round "<<i<<": ";
        printArray(arr,n);
        cout<<endl;
    }
    cout<<endl;

}
```

```
int main(){
    cout<<"\nExperiment-1.2 (Ashish Kumar, 23MAI10008)"<<endl<<endl;
    cout<<"Performing Bubble 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;

    // Bubble Sort Function
    bubbleSort(arr,n);

    cout<<"Sorted Array is: ";
    printArray(arr,n);
}
```

6. Result/Output :



```
PS E:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Courses\23CSH-622 (ADVANCED DATA STRUC
TURES AND ALGORITHMS)\LAB\Code> cd "e:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Cours
es\23CSH-622 (ADVANCED DATA STRUCTURES AND ALGORITHMS)\LAB\Code\" ; if ($?) { g++
ADS_EXPERIMENT-1-2_BubbleSort.cpp -o ADS_EXPERIMENT-1-2_BubbleSort } ; if ($?) { .
\ADS_EXPERIMENT-1-2_BubbleSort }

Experiment-1.2 (Ashish Kumar, 23MAI10008)

Performing Bubble 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 29 52 18 84 47 73 35 21 91
Round 2: 29 45 18 52 47 73 35 21 84 91
Round 3: 29 18 45 47 52 35 21 73 84 91
Round 4: 18 29 45 47 35 21 52 73 84 91
Round 5: 18 29 45 35 21 47 52 73 84 91
Round 6: 18 29 35 21 45 47 52 73 84 91
Round 7: 18 29 21 35 45 47 52 73 84 91
Round 8: 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 bubble sort on an array.

3. I learnt about how to optimize the bubble sort.

4. I learnt about how to traverse the array elements.

5. I learnt about time and space complexity of bubble sort.