



## Experiment-1.3

### Aim of the Experiment :

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

### 1. Problem Description :

Selection Sort: Selection sort is a sorting algorithm that works by repeatedly selecting the smallest element from the unsorted portion of the list and swaps it with the first element of the unsorted part. This process is repeated for the remaining unsorted portion until the entire list is sorted.

### 2. Algorithm :

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

Step 2: Enter the elements of array from user.

Step 3: Set MIN to location 0

Step 4: Search the minimum element in the array

Step 5: Swap the minimum element with value at location MIN

Step 6: Increment MIN to point to next element

Step 7: Repeat until list is sorted

Step 8: Print all the elements of array.

Step 9: Exit.

### 3. Complexity Analysis:

#### Time Complexity:

In Selection 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,

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

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

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

1) Best Case:  $O(N^2)$ . 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 selection sort

list : array of items

n : size of list

for i = 1 to n - 1

/\* set current element as minimum\*/

min = i

/\* check the element to be minimum \*/

```
        for j = i+1 to n
            if list[j] < list[min] then
                min = j;
            end if
        end for

        /* swap the minimum element with the current element*/

        if indexMin != i then
            swap list[min] and list[i]
        end if
    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 selectionSort(int arr[], int n){  
    cout<<endl;  
    for(int i=0; i<n-1; i++){  
        int minindex=i;  
        for(int j=i+1; j<n; j++){  
            if(arr[j]<arr[minindex]){  
                minindex=j;  
            }  
        }  
        swap(arr[i],arr[minindex]);  
  
        cout<<"Round "<<i+1<<": ";  
        printArray(arr,n);  
        cout<<endl;  
    }  
    cout<<endl;  
  
}  
  
int main(){  
  
    cout<<"\nExperiment-1.3 (Ashish Kumar, 23MAI10008)"<<endl<<endl;  
    cout<<"Performing Selection 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;


// Selection Sort Function

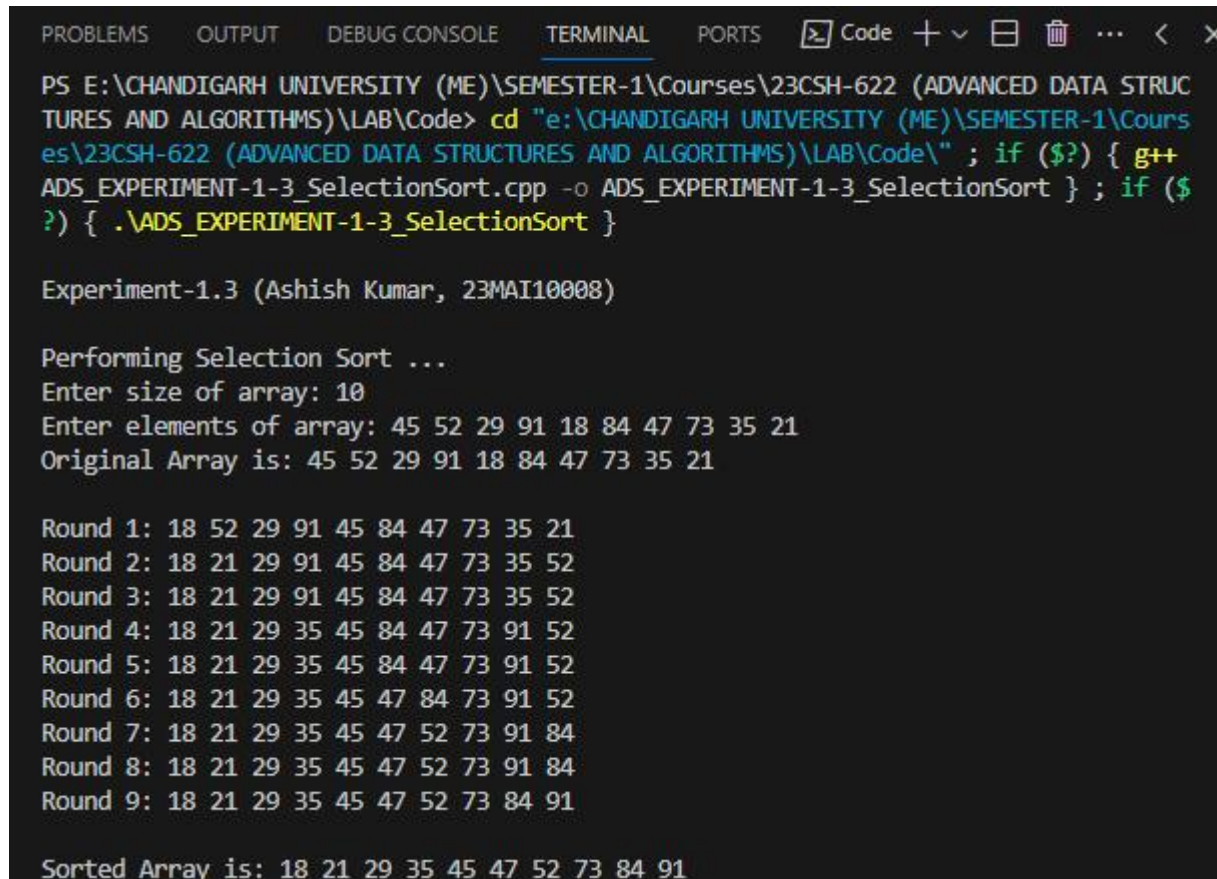
selectionSort(arr,n);


cout<<"Sorted Array is: ";

printArray(arr,n);

}
```

## 6. Result/Output :



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code + - [ ] [X] ... < X
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 ($?) { g++
ADS_EXPERIMENT-1-3_SelectionSort.cpp -o ADS_EXPERIMENT-1-3_SelectionSort } ; if ($
?) { .\ADS_EXPERIMENT-1-3_SelectionSort }

Experiment-1.3 (Ashish Kumar, 23MAI10008)

Performing Selection 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: 18 52 29 91 45 84 47 73 35 21
Round 2: 18 21 29 91 45 84 47 73 35 52
Round 3: 18 21 29 91 45 84 47 73 35 52
Round 4: 18 21 29 35 45 84 47 73 91 52
Round 5: 18 21 29 35 45 84 47 73 91 52
Round 6: 18 21 29 35 45 47 84 73 91 52
Round 7: 18 21 29 35 45 47 52 73 91 84
Round 8: 18 21 29 35 45 47 52 73 91 84
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 selection sort on an array.
3. I learnt about difference between bubble and selection sort.
4. I learnt about how to traverse the array elements.
5. I learnt about time and space complexity of selection sort.

**Comparison between Bubble Sort and Selection Sort:**

<b>Bubble Sort</b>	<b>Selection Sort</b>
Bubble sorting is a sorting algorithm where we check two elements and swap them at their correct positions.	Selection sorting is a sorting algorithm where we select the minimum element from the array and put that at its correct position.
Bubble sort performs a large number of swaps or moves to sort the list.	Selection sort performs comparatively less number of swaps or moves to sort the list.
Bubble sort performs sorting of an array by exchanging elements.	Selection sort performs sorting of a list by the selection of element.
The efficiency of the Bubble sort is less.	The efficiency of the Selection sort is high.
Bubble sort is a stable algorithm.	Selection sort is not a stable algorithm.
Bubble sort is relatively slower.	Selection sort is faster as compared to bubble sort.