**Experiment No. – 1.3**

**Aim**: Write a program to implement Selection sort along with its complexity analysis.

1. **Problem Description:**

We have given an unsorted array of numbers, generate a sorted array of numbers by applying Selection Sort. Demonstrate knowledge of time complexity of Selection Sort by counting the number of operations involved in each iteration and find space complexity. Also compare between selection and bubble sort.

1. **Algorithm:**

Algorithm: (selection\_sort) SELECTION(A,N)

1. Repeat steps 2 & 3 for K = 1,2…N-1:
2. Call MIN(A,K,N,LOC)
3. Interchange A[K] & A[LOC].]

Set TEMP=A[K], A[K]=A[LOC] & A[LOC]=TEMP

[End of step1 loop]

1. Exit.

Algorithm: MIN(A,K,N,LOC)

1. Set MIN=A[k] & LOC=K. [Initializes pointers]
2. Repeat for J=K+1, K+2,..N:

If MIN>A[J], then: Set MIN=A[J] & LOC=J

[End of loop]

3. Return

1. **Complexity Analysis**

**Time complexity of Selection Sort**

In the selection sort algorithm, (N-1) comparisons required during pass1 to find the smallest element, n-2 in 2nd pass and so on. So the total number of comparisons will be,

=f(n) =(n-1)+(n-2)+…..2+1= n(n-1)/2 = O(n2 )

Worst Case: O(n2). When the array is sorted.

Best Case: O(n2). When the elements of the array are in decreasing order.

Average: O(n2)

**Space Complexity:** O(1).As no extra space is used.

1. **Pseudo Code**

procedure selection sort

list : array of items

n : size of list

for i = 1 to n - 1

min = i /\* set current element as minimum\*/

for j = i+1 to n /\* check the element to be minimum \*/

if list[j] < list[min] then

min = j;

end if

end for

if indexMin != i then /\* swap the minimum element with the current element\*/

swap list[min] and list[i]

end if

end for

end procedure

1. **Source Code (C/C++):**

#include <bits/stdc++.h>

using namespace std;

void swapping(int &a, int &b)

{

    int temp;

    temp = a;

    a = b;

    b = temp;

}

void bubbleSort(int a[], int n)

{

    int temp, min;

    for (int i = 0; i < n; i++)

    {

        min = a[i];

        for (int j = i; j < n; j++)

        {

            if (a[j] < min)

            {

                temp = min;

                min = a[j];

                a[j] = temp;

            }

        }

        a[i] = min;

    }

}

int main()

{

    cout << "Name: Saurabh Kumar \nUID: 23MAI10004\n";

    int n;

    cout << "Enter the Size: ";

    cin >> n;

    int arr[n];

    cout << "Enter elements:" << endl;

    for (int i = 0; i < n; i++)

    {

        cin >> arr[i];

    }

    bubbleSort(arr, n);

    cout << "Array after Sorting: ";

    for (int i = 0; i < n; i++)

    {

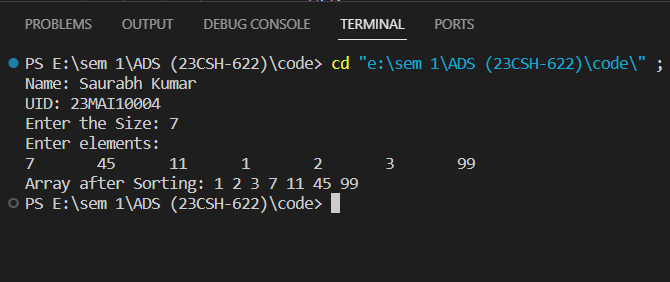
        cout << arr[i] << " ";

    }

    return 0;

}

1. **Screenshot of Outputs:**

****

1. **Learning Outcomes**
2. Learnt about how to sort an array.
3. Learnt about how to implement selection sort.
4. Learnt about how to find time and space complexity
5. Learnt about how to code in c++ and take array as an input.
6. Learnt about how to traverse an array.
7. Learnt about the comparison between selection and bubble sort.

**Comparison between Selection sort and Bubble sort**

|  |  |
| --- | --- |
| Bubble Sort | Selection Sort |
| Bubble sort compares the adjacent elements and move accordingly. | Selection sort selects the smallest element from the unsorted list and moves it at the next position of the sorted list. |
| 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 is not efficient | Selection sort is efficient |
| Bubble sort is stable. | Selection sort is not stable. |
| Bubble sort is relatively slower. | Selection sort is faster as compared to bubble sort. |