Experiment no 11

Bubble, Selection and Insertion Sort

OBJECTIVES:

- To learn about the concept of Bubble, Selection and Insertion Sort.
- To learn different ways for implementing a Bubble, Selection and Insertion Sort.

1. BUBBLE SORT:

Bubble sort repeatedly compares and swaps(if needed) adjacent elements in every pass. In i-th pass of Bubble Sort (ascending order), last (i-1) elements are already sorted, and i-th largest element is placed at (N-i)-th position, i.e. i-th last position.

```
BubbleSort (Arr, N) // Arr is an array of size N.
{
    For ( I:= 1 to (N-1) ) // N elements => (N-1) pass
    {
    // Swap adjacent elements of Arr[1:(N-I)]such that
    // largest among { Arr[1], Arr[2], ..., Arr[N-I] } reaches to Arr[N-I]
        For ( J:= 1 to  (N-I) ) // Execute the pass
        {
            If ( Arr [J] > Arr[J+1] )
                Swap( Arr[j], Arr[J+1] );
        }
    }
}
```

Figure 1: Bubble Sort

Optimization of Algorithm: Check if there happened any swapping operation in the inner loop (pass execution loop) or not. If there is no swapping in any pass, it means the array is now fully sorted, hence no need to continue, stop the sorting operation. So we can optimize the number of passes when the array gets sorted before the completion of all passes. And it can also detect if the given / input array is sorted or not, in the first pass.

2. SELECTION SORT:

Selection sort selects i-th smallest element and places at i-th position. This algorithm divides the array into two parts: sorted (left) and unsorted (right) subarray. It selects the smallest element from unsorted subarray and places in the first position of that subarray (ascending order). It repeatedly selects the next smallest element.

Time Complexity:

- Best Case [O(N2)]. And O(1) swaps.
- Worst Case: Reversely sorted, and when the inner loop makes a maximum comparison. [O(N2)] . Also, O(N) swaps.
- Average Case: [O(N2)] . Also O(N) swaps.

```
SelectionSort (Arr, N) // Arr is an array of size N.
{
    For ( I:= 1 to (N-1) ) // N elements => (N-1) pass
    {
    // I=N is ignored, Arr[N] is already at proper place.
    // Arr[1:(I-1)] is sorted subarray, Arr[I:N] is unsorted subarray
    // smallest among { Arr[I], Arr[I+1], Arr[I+2], ..., Arr[N] } is at place min_index

        min_index = I;
        For ( J:= I+1 to N ) // Search Unsorted Subarray (Right lalf)
        {
            If ( Arr [J] < Arr[min_index] )
                min_index = J; // Current minimum
        }
        // Swap I-th smallest element with current I-th place element
        If (min_Index != I)
              Swap ( Arr[I], Arr[min_index] );

    }
}
```

Figure 2: Selection Sort

3. INSERTION SORT:

Insertion Sort is a simple comparison based sorting algorithm. It inserts every array element into its proper position. In i-th iteration, previous (i-1) elements (i.e. subarray Arr[1:(i-1)]) are already sorted, and the i-th element (Arr[i]) is inserted into its proper place in the previously sorted subarray.
Find more details in this GFG Link.

```
InsertionSort (Arr, N) // Arr is an array of size N.
{
    For ( I:= 2 to N ) // N elements => (N-1) pass
    {
    // Pass 1 is trivially sorted, hence not considered
    // Subarray { Arr[1], Arr[2], ..., Arr[I-I] } is already sorted

        insert_at = I; // Find suitable position insert_at, for Arr[I]
        // Move subarray Arr [ insert_at: I-1 ] to one position right
        item = Arr[I]; J=I-1;
        While ( J ? 1 && item < Arr[J] )
        {
                Arr[J+1] = Arr[J]; // Move to right
                // insert_at = J;
                J--;
            }
            insert_at = J+1; // Insert at proper position
            Arr[insert_at] = item; // Arr[J+1] = item;
        }
    }
}
```

Figure 3: Insertion Sort

Time Complexity:

- **Best Case** Sorted array as input, [**O(N)**]. And **O(1)** swaps.
- **Worst Case**: Reversely sorted, and when inner loop makes maximum comparison, [**O(N2)**] . And **O(N2)** swaps.
- **Average Case**: [**O(N2)**] . And **O(N2)** swaps.

4. ACTIVITIES:

A1) Write a code for Bubble Sort.

A2) Write a code for Selection Sort.

A3) Write a code Insertion Sort.

A4) Write a code in which bubble, selection and insertion sort should be implemented.

IMPORTANT: All the activities` code should be attached to the manual before summary section.

Lab Exercise and Summary

Summary should cover Introduction, Procedure, Data Analysis and Evaluation.

Student's Signature:_____ **Date:**_____

LABORATORY SKILLS ASSESSMENT (Psychomotor)

Total Marks:100

Criteria (Max Marks)	Level 1 0% ≤ S < 50%	Level 2 50% ≤ S< 70%	Level 3 70% ≤ S< 90%	Level 4 90%≤ S ≤100%	Score (S)
Procedural Awareness (30)	Selects inappropriate skills and/or strategies Required by the task.	Selects and applies appropriate skills and/or strategies required by the task with major errors.	Selects and applies the appropriate strategies and/or skills specific to the task without significant errors.	Selects and applies appropriate strategies and/or skills specific to the task without any error.	
Practical Implementation (30)	Makes major critical errors in applying procedural knowledge related to python Bubble, Selection and Insertion Sort	Makes numerous critical errors in applying procedural knowledge related to python Bubble, Selection and Insertion Sort	Makes some non-critical errors in applying procedural knowledge related to python Bubble, Selection and Insertion Sort	Applies the procedural knowledge in optimized ways related to python Lists, Bubble, Selection and Insertion Sort	
Use of Tool/Equipment (30)	Uses tools, equipment and materials with limited competence.	Uses tools, equipment and materials with some competence.	Uses tools, equipment and materials with considerable competence.	Uses tools, equipment and materials with a high degree of competence.	
Safety (10)	Requires constant reminders to follow safety procedures.	Requires some reminders to follow safety procedures.	Follows safety procedures with only minimal reminders.	Routinely follows safety procedures.	
				Marks Obtained	

Instructor Name:_____ **Sign:** _____

LABORATORY SKILLS ASSESSMENT (Affective)

Total Marks: 40

Criteria (Max. Marks)	Level 1 0% ≤ S < 50%	Level 2 50% ≤ S < 70%	Level 3 70% ≤ S < 90%	Level 4 90% ≤ S ≤ 100%	Score
Introduction (5)	Very little background information provided or information is incorrect	Introduction is brief with some minor mistakes	Introduction is nearly complete, missing some minor points	Introduction complete and well-written; provides all necessary background principles for the experiment	
Procedure (5)	Many stages of the procedure are not entered on the lab report.	Many stages of the procedure are entered on the lab report.	The procedure could be more efficiently designed but most stages of the procedure are entered on the lab report.	The procedure is well designed and all stages of the procedure are entered on the lab report.	
Data Record (10)	Data is brief and missing significant pieces of information.	Data provides some significant information and has few critical mistakes.	Data is almost complete but has some minor mistakes.	Data is complete and relevant. Tables with units are provided. Graphs are labeled. All questions are answered correctly.	
Data Analysis (10)	Data is presented in very unclear manner.	Data is presented in ways that are not clear enough.	Data is presented in ways that can be understood and interpreted.	Data is presented in ways that best facilitate understanding and interpretation.	
Report Quality (10)	Report contains many errors.	Report is somewhat organized with some spelling or grammatical errors.	Report is well organized and cohesive but contains some grammatical errors.	Report is well organized and cohesive and contains no grammatical errors. Presentation seems polished.	
				Marks Obtained	

LABORATORY SKILLS ASSESSMENT (Cognitive)

Total Marks: 10

(If any) **Marks Obtained**	

Instructor's Signature:_____ **Date:**_____