Aim of Experiment

To implement and compare Bubble Sort, Insertion Sort /Selection Sort algorithms in C++ for sorting a list of data items.

Objectives

- To understand the concept of sorting techniques.
- To implement Bubble Sort algorithm using C++.
- To implement Insertion /Selection Sort algorithm using C++.
- To analyze the working principle and efficiency of these algorithms.
- To compare the time complexity of Bubble, Insertion, and Selection Sort algorithms.

Theory

Sorting

Sorting is the process of arranging data in a particular order (ascending or descending). Sorting makes searching and data processing more efficient.

Bubble Sort

Bubble Sort is a simple sorting algorithm that repeatedly compares adjacent elements and swaps them if they are in the wrong order.

Algorithm (Bubble Sort):

Step 1: Start
Step 2: Input array of n elements.
Step 3: Repeat for i = 0 to n-1
    For j = 0 to n-i-1
        If array[j] > array[j+1], swap them
Step 4: End

Time Complexity: $O(n^2)$
Best Case: O(n) (already sorted)
Worst Case: $O(n^2)$
Space Complexity: O(1)

Insertion Sort

Insertion Sort works by building a sorted list one element at a time by repeatedly picking the next element and inserting it into its correct position.

Algorithm (Insertion Sort):

Step 1: Start
Step 2: Input array of n elements.
Step 3: Repeat for i = 1 to n-1
    key = array[i]
    j = i - 1
    While j >= 0 and array[j] > key
        array[j+1] = array[j]
        j = j - 1
    array[j+1] = key
Step 4: End

Time Complexity: $O(n^2)$
Best Case: O(n) (already sorted)
Worst Case: $O(n^2)$
Space Complexity: O(1)

Selection Sort

Selection Sort works by repeatedly selecting the smallest (or largest) element from the unsorted portion and placing it at the correct position.

Algorithm (Selection Sort):

Step 1: Start
Step 2: Input array of n elements.
Step 3: Repeat for i = 0 to n-1
    min_index = i
    For j = i+1 to n-1
        If array[j] < array[min_index]
            min_index = j
    Swap array[min_index] and array[i]
Step 4: End

Time Complexity: $O(n^2)$
Best Case: $O(n^2)$
Worst Case: $O(n^2)$
Space Complexity: O(1)

Applications

Bubble Sort Applications:

- Simple and easy to implement for small datasets.
- Useful when simplicity is more important than efficiency.
- Helpful for teaching basic sorting concepts.

Insertion Sort Applications:

- Efficient for small datasets.
- Used when the array is already nearly sorted.
- Preferred in online algorithms where data is received in a stream.

Selection Sort Applications:

- Useful when memory writes are costly (since it makes fewer swaps).
- Can be used for small datasets where simplicity is preferred.
- Works well when the cost of swapping is higher than comparing.

Questions

Task-1

Manually **trace Bubble Sort** on a small list of integers.

i. Take a list of **05 unsorted integers** (distinct numbers).

ii. Apply the **Bubble Sort algorithm** to sort the list in **ascending order**.

Task-2

Manually **trace Selection Sort** on a small list of integers.

i. Take a list of **08 unsorted integers** (distinct numbers).

ii. Apply the **Selection Sort algorithm** to sort the list in **ascending order**.

Task-3

Manually **trace Insertion Sort** on a small list of integers.

i. Take a list of **08 unsorted integers** (distinct numbers).

ii. Apply the **Insertion Sort algorithm** to sort the list in **ascending order**.

References

1. Yedidyah Langsam, Moshe J Augenstein, Aaron M Tenenbaum – Data structures using C and C++ - PHI Publications (2nd Edition).
2. Ellis Horowitz, Sataraj Sahni- Fundamentals of Data Structures – Galgotia Books source.

SE(ECE)	Data Structures and Algorithms	
Experiment No: 3	Implement a) Bubble Sort, b) Insertion or Selection Sort algorithms using C++	Page: 4/4

Conclusion

A	C	O	T	Sign.