Day 17: Insertion Sort

Gati Goyal

"The sooner you start to code, the longer the program will take."
— Roy Carlson

---

1 Introduction

Insertion sort is a simple and efficient sorting algorithm that builds the sorted array one element at a time by repeatedly taking the next element from the unsorted part and inserting it into its correct position in the sorted part.

2 Problem Statement

Problem: Sort an array of integers using the insertion sort algorithm. **Hint:** Iterate over the array and insert each element into the sorted portion. **Edge Case:** Handle cases with arrays of size 1 or already sorted arrays.

3 Algorithm

1. Iterate through the array starting from the second element (index 1).
2. Compare the current element with elements in the sorted portion of the array (left side).
3. Shift larger elements one position to the right to make space for the current element.
4. Insert the current element into its correct position.

4 Code

```c
#include <stdio.h>

// Insertion Sort function
void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
```

```
        // Move elements of arr[0..i-1] that are greater than key
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

int main() {
    int n;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter the elements: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    insertionSort(arr, n);

    printf("Sorted array after insertion sort: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

5 Complexity Analysis

- **Time Complexity:**
  - Best Case: $O(n)$ (when the array is already sorted).
  - Average Case: $O(n^2)$.
  - Worst Case: $O(n^2)$ (when the array is sorted in reverse order).
- **Space Complexity:** $O(1)$ (in-place sorting with no additional memory).

6 Examples and Edge Cases

Input Array	Output Array	Steps Required
{12, 11, 13, 5, 6}	{5, 6, 11, 12, 13}	5 Passes
{1, 2, 3, 4, 5}	{1, 2, 3, 4, 5}	1 Pass (Already Sorted)
{5, 4, 3, 2, 1}	{1, 2, 3, 4, 5}	5 Passes

7 Output

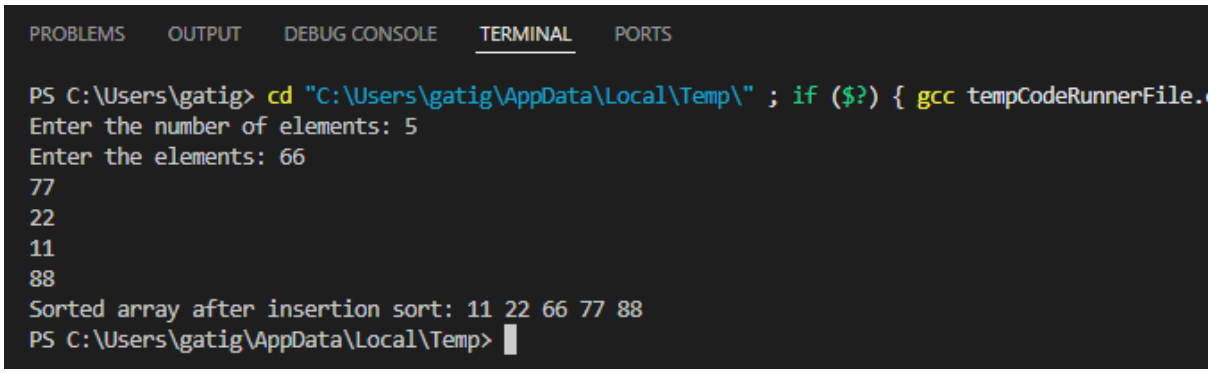

Figure 1: Program Output Screenshot

8 Conclusion

Insertion sort is an efficient algorithm for small or nearly sorted data sets due to its $O(n)$ performance in the best case. However, its $O(n^2)$ complexity in the average and worst cases makes it less suitable for larger data sets. Its simplicity and ability to sort in place make it an excellent learning tool.