Day 16: Selection Sort

Gati Goyal

"Programs must be written for people to read, and only incidentally for machines to execute."
— Harold Abelson

---

1 Introduction

Selection sort is a simple comparison-based sorting algorithm. It divides the input list into two parts: a sorted subarray which is built up from left to right and a subarray of the remaining unsorted elements. At each step, the smallest (or largest) element is selected from the unsorted subarray and swapped with the leftmost unsorted element.

2 Problem Statement

Problem: Sort an array of integers using the selection sort algorithm. **Hint:** Find the minimum element in each iteration and place it in the correct position. **Edge Case:** Handle arrays of size 1 or empty arrays.

3 Algorithm

1. Iterate through the array from left to right.
2. For each element, find the smallest element in the remaining unsorted portion of the array.
3. Swap the smallest element with the current element.
4. Repeat the process for the next position until the entire array is sorted.

4 Code

```c
#include <stdio.h>

// Selection Sort function
void selectionSort(int arr[], int n) {
    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 the minimum element with the current element
        int temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}

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]);
    }

    selectionSort(arr, n);

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

    return 0;
}
```

5 Complexity Analysis

- **Time Complexity:**
  - Best Case: $O(n^2)$.
  - Average Case: $O(n^2)$.
  - Worst Case: $O(n^2)$.
- **Space Complexity:** $O(1)$ (in-place sorting with no additional memory).

6 Examples and Edge Cases

Input Array	Output Array	Steps Required
{64, 25, 12, 22, 11}	{11, 12, 22, 25, 64}	4 Passes
{1, 2, 3, 4, 5}	{1, 2, 3, 4, 5}	4 Passes (Already Sorted)
{5, 4, 3, 2, 1}	{1, 2, 3, 4, 5}	4 Passes

7 Output

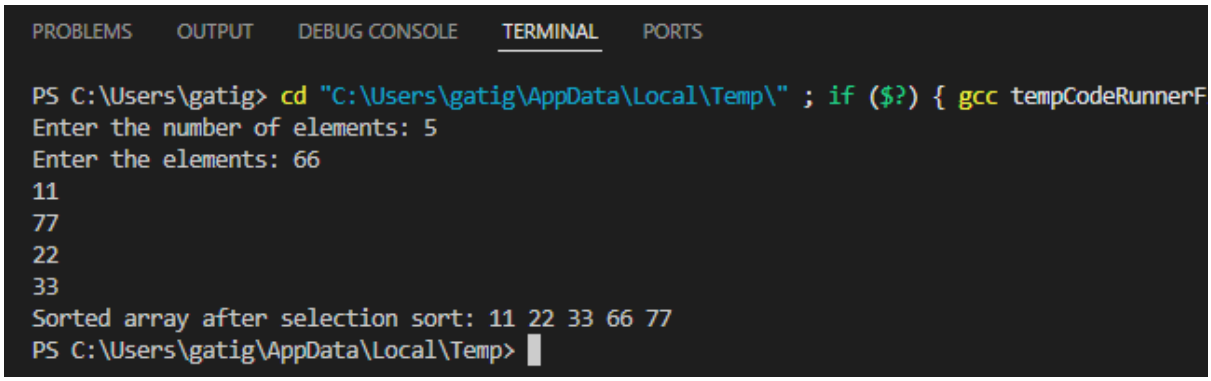

Figure 1: Program Output Screenshot

8 Conclusion

Selection sort is a simple and intuitive sorting algorithm suitable for small data sets. Although it is less efficient than algorithms like merge sort or quicksort for larger data sets, its deterministic $O(n^2)$ complexity and simplicity make it ideal for teaching sorting concepts.