Sorting

Sorting algorithms are fundamental to computer science, arranging data in a specific order (ascending or descending). In C programming, several efficient sorting techniques are commonly employed:

1. Bubble Sort:

- **Simple:** Compares adjacent elements and swaps them if they are in the wrong order.
- **Time complexity:** O(n^2) in the worst case.
- **Space complexity:** O(1).
- **Inefficient for large datasets.**

2. Selection Sort:

- **Finds the minimum element:** In each iteration, finds the minimum element and swaps it with the first unsorted element.
- **Time complexity:** O(n^2) in all cases.
- **Space complexity:** O(1).

3. Insertion Sort:

- **Iterative:** Builds the sorted array one element at a time.
- **Time complexity:** O(n^2) in the worst case, O(n) in the best case.
- **Space complexity:** O(1).
- **Efficient for small datasets and partially sorted arrays.**

4. Merge Sort:

- **Divide and conquer:** Divides the array into two halves, recursively sorts each half, and then merges the sorted halves.
- **Time complexity:** O(n log n) in all cases.
- **Space complexity:** O(n) for the auxiliary array.
- **Efficient for large datasets.**

5. Quick Sort:

- **Divide and conquer:** Picks a pivot element, partitions the array into two subarrays based on the pivot, and recursively sorts the subarrays.
- **Time complexity:** O(n log n) average case, O(n^2) worst case.
- **Space complexity:** O(log n) for the recursion stack.
- **Efficient for large datasets and often considered the fastest sorting algorithm.**