**Sorting algorithms**

Bubble Sort: Compares adjacent elements and swaps them if they are in the wrong order. Repeatedly passes through the list until no swaps occur.

**Time complexity:** O(n^2) in average and worst cases, O(n) in best case (already sorted)

Insertion Sort: Builds the sorted part of the array one element at a time by inserting elements into their correct position.

**Time complexity:** O(n^2) in average and worst cases, O(n) in best case (already sorted).

Quick Sort: A divide-and-conquer algorithm. Picks a pivot element, partitions the array around the pivot, and recursively sorts the sub-arrays.

**Time complexity:** O(n log n) on average, O(n^2) in worst case (already sorted or reverse sorted).

Merge Sort: A divide-and-conquer algorithm. Divides the array into two halves, recursively sorts them, and merges the sorted halves.

**Time complexity:** O(n log n) in all cases.

**Analysis:**

Quick Sort is generally preferred over Bubble Sort due to its significantly better average-case time complexity (O(nlogn) compared to O(n^2)) and efficiency in handling larger datasets