**Understanding Sorting Algorithms:**

Sorting algorithms arrange elements in a specific order (e.g., numerical or lexicographical). Here are the ones we’ll discuss:

**Bubble Sort:**

Time complexity:

Worst and average cases: O(n2)

Best case (when input is already sorted): O(n)

Basic idea: Repeatedly compare adjacent pairs of elements and swap them if they’re in the wrong order. Repeat until the array is fully sorted.

**Quick Sort:**

Time complexity:

Average and best cases: O(nlogn)

Worst case: O(n2)(rare, but possible)

Basic idea: Divide the array into two subarrays (those less than a pivot and those greater than the pivot). Recursively sort the subarrays.

**Merge Sort:**

Time complexity: Always O(nlogn)

Basic idea: Divide the array into halves, sort each half, and then merge the sorted halves.

**Analysis:**

**Performance Comparison:**

Bubble Sort worst-case time complexity: O(n2)

Quick Sort average-case time complexity: O(nlogn)

**Why Quick Sort?**

Quick Sort is generally preferred because:

It’s faster (average-case performance).

It’s more memory-efficient (in-place sorting).

Bubble Sort is inefficient for large datasets.

Quick Sort can handle partially sorted data well.