**Sorting Customer Orders**

**Sorting Algorithms**

1. Bubble Sort: A simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted.

- Time Complexity:

- Best Case: O(n)

- Average Case: O(n²)

- Worst Case: O(n²)

- Space Complexity: O(1)

2. Insertion Sort : Builds the final sorted array one item at a time. It is much less efficient on large lists than more advanced algorithms such as quick sort, merge sort, or heap sort.

- Time Complexity:

- Best Case: O(n)

- Average Case: O(n²)

- Worst Case: O(n²)

- Space Complexity: O(1)

3. Quick Sort : An efficient, comparison-based, divide-and-conquer sorting algorithm. It picks an element as a pivot and partitions the array around the picked pivot.

- Time Complexity:

- Best Case: O(n log n)

- Average Case: O(n log n)

- Worst Case: O(n²)

- Space Complexity : O(log n)

4. Merge Sort : An efficient, stable, comparison-based, divide-and-conquer sorting algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves.

- Time Complexity: O(n log n) for all cases

- Space Complexity: O(n)

**Analysis**

1. Bubble Sort:

- Best Case: O(n) - This occurs when the array is already sorted.

- Average Case: O(n²) - On average, each element is compared with every other element.

- Worst Case: O(n²) - This occurs when the array is sorted in reverse order.

2. Quick Sort:

- Best Case: O(n log n) - This occurs when the pivot splits the array into two equal halves.

- Average Case: O(n log n) - On average, the pivot splits the array into reasonably balanced halves.

- Worst Case: O(n²) - This occurs when the pivot is the smallest or largest element, leading to unbalanced partitions.

**Quick Sort is generally preferred over Bubble Sort because:**

Quick Sort has a better average-case time complexity (O(n log n)) compared to Bubble Sort's average-case time complexity (O(n²)). And is faster for large datasets due to its divide-and-conquer approach, making it more scalable and efficient. Quick Sort is widely used in practice and optimized in many programming libraries and frameworks due to its performance benefits. While Bubble Sort is simpler to implement and understand, its inefficiency for large datasets makes it less suitable for most practical applications compared to Quick Sort.