**Exercise 3: Sorting Customer Orders**

**Understand Sorting Algorithms**

**Bubble Sort**

* Compares adjacent elements and swaps them if needed.
* Repeats until the array is sorted.
* **Time Complexity**:
  + Worst: O(n²)
  + Best: O(n) (already sorted)
  + Avg: O(n²)

**Insertion Sort** (not required here, but good to know)

* Builds the sorted array one element at a time.
* Good for small or nearly sorted datasets.

**Quick Sort**

* Divide & Conquer algorithm.
* Picks a pivot, partitions the array, and recursively sorts parts.
* **Time Complexity**:
  + Worst: O(n²) (rare)
  + Avg/Best: O(n log n)
* Faster than Bubble Sort for large data.

**Analysis**

| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** | **Space** | **Stable?** |
| --- | --- | --- | --- | --- | --- |
| Bubble Sort | O(n) | O(n²) | O(n²) | O(1) | ✅ Yes |
| Quick Sort | O(n log n) | O(n log n) | O(n²) (rare) | O(log n) (stack) | ❌ No |

**Why Quick Sort is Preferred:**

* Much faster for large datasets.
* Average case is O(n log n) vs. O(n²) for Bubble Sort.
* Widely used in real-world systems (e.g., Arrays.sort() uses Dual-Pivot Quick Sort in Java).