**Sorting Customer Orders**

**Understanding Sorting Algorithms**

**Bubble Sort**:

* **Description**: A simple comparison-based algorithm where each pair of adjacent elements is compared and swapped if they are in the wrong order. This process is repeated until the list is sorted.
* **Time Complexity**: O(n^2) in the average and worst-case scenarios. O(n) in the best-case scenario (when the list is already sorted).
* **Space Complexity**: O(1) (in-place sort).

**Insertion Sort**:

* **Description**: Builds the sorted array one item at a time, inserting each new item into its correct position within the already sorted section of the array.
* **Time Complexity**: O(n^2) in the average and worst-case scenarios. O(n) in the best-case scenario (when the list is already sorted).
* **Space Complexity**: O(1) (in-place sort).

**Quick Sort**:

* **Description**: A divide-and-conquer algorithm. It works by selecting a 'pivot' element and partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot. The sub-arrays are then sorted recursively.
* **Time Complexity**: O(n log n) on average. O(n^2) in the worst-case scenario (rare, usually when the pivot selection is poor).
* **Space Complexity**: O(log n) due to the recursive stack.

**Merge Sort**:

* **Description**: Another divide-and-conquer algorithm that divides the array into halves, sorts each half, and then merges the sorted halves back together.
* **Time Complexity**: O(n log n) in all cases (average, best, and worst).
* **Space Complexity**: O(n) due to the need for a temporary array to hold the merged result.