**Sorting Customer Orders**

**Understand Sorting Algorithms**

**Bubble Sort**:

* Bubble Sort 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**: O(n^2) in the average and worst cases, where n is the number of items being sorted.
* **Space Complexity**: O(1) because it is an in-place sorting algorithm.

**Insertion Sort**:

* Insertion Sort builds the final sorted array one item at a time. It takes each element from the list and finds the correct position within the sorted part of the array.
* **Time Complexity**: O(n^2) in the average and worst cases.
* **Space Complexity**: O(1) as it is an in-place sorting algorithm.

**Quick Sort**:

* Quick Sort is a divide-and-conquer algorithm. It works by selecting a 'pivot' element from the array 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.
* **Space Complexity**: O(log n) due to the recursive stack space.

**Merge Sort**:

* Merge Sort is also a divide-and-conquer algorithm. It divides the unsorted list into n sublists until each sublist contains a single element, then merges the sublists to produce new sorted sublists until there is only one sorted sublist remaining.
* **Time Complexity**: O(n log n) in all cases.
* **Space Complexity**: O(n) due to the additional space used for merging.

Analysis

**Performance Comparison:**

Bubble Sort has a time complexity of O(n^2) for both the average and worst cases. This means that for larger datasets, it becomes very inefficient.

Quick Sort has an average-case time complexity of O(n log n) and a worst-case time complexity of O(n^2). However, with good pivot selection (e.g., using a random pivot), the worst case can be avoided, making Quick Sort generally much faster for larger datasets.

**Why Quick Sort is Preferred Over Bubble Sort:**

**Efficiency**: Quick Sort is more efficient with its average-case time complexity of O(n log n), which is significantly better than Bubble Sort's O(n^2).

**Scalability**: Quick Sort handles larger datasets more effectively. In practical scenarios, Quick Sort is one of the fastest sorting algorithms.

**Versatility**: Quick Sort can be implemented in-place with O(log n) additional space, making it memory efficient compared to other algorithms like Merge Sort, which requires O(n) additional space.

In summary, while Bubble Sort is simpler to implement and understand, Quick Sort is generally preferred for its efficiency and scalability in real-world applications.