**Sort Customer Order**

**Bubble Sort:**

Bubble sort is a straightforward sorting algorithm that works by repeatedly stepping through the list, comparing adjacent items, and swapping them if they are in the wrong order. This process continues until a complete pass through the list is made without any swaps, which means the list is sorted. Although it is easy to understand and implement, it becomes inefficient with large datasets. Both the average and worst-case time complexity is O(n^2).

**Insertion Sort:**

Insertion sort processes a list by iterating through it and placing each item into its correct position among the already sorted elements. This method performs well for small or nearly sorted lists. While its average and worst-case time complexity is O(n^2), the best-case scenario is O(n) when the list is already sorted.

**Quick Sort:**

Quick sort chooses a pivot element, then rearranges the list so that elements less than the pivot are on one side and those greater are on the other. It recursively applies the same process to the sublists. This algorithm is efficient for large datasets, achieving an average-case time complexity of O(nlogn). However, in the worst case, such as with poor pivot choices, its complexity can degrade to O(n^2). Quick sort is typically fast but may lack stability.

**Merge Sort:**

Merge sort works by dividing the list into smaller segments, sorting each segment individually, and then merging the sorted segments back together. This algorithm is known for its stability and consistent performance, with a time complexity of O(nlogn) regardless of the input case. It is particularly effective for large datasets and supports parallel processing, but it does require additional memory for merging operations.

**Comparison of Quick Sort and Bubble Sort:**

Quick sort is generally preferred over bubble sort due to its superior efficiency and scalability. With an average time complexity of O(nlogn), quick sort handles large datasets more effectively compared to bubble sort’s O(n^2) complexity, which becomes impractically slow for large lists. Additionally, quick sort's efficient memory use and cache performance contribute to its popularity over bubble sort.