**Exercise 3: Sorting Customer Orders**

**Sorting Algorithms Overview (Bubble Sort, Insertion Sort, Quick Sort, Merge Sort):**

* **Bubble Sort:**
  + Description: A simple comparison-based algorithm. It repeatedly compares and swaps adjacent elements if they are in the wrong order until the list is sorted.
  + Complexity: O(n^2) time complexity, making it inefficient for large datasets. Space complexity is O(1), as it sorts in place.
* **Insertion Sort:**
  + Description: Builds the sorted array one item at a time by picking the next item and inserting it into the correct position.
  + Complexity: O(n^2) time complexity, but O(n) for nearly sorted data. Space complexity is O(1), efficient for small or nearly sorted datasets.
* **Quick Sort:**
  + Description: A divide-and-conquer algorithm that selects a pivot, partitions the array around the pivot, and recursively sorts the sub-arrays.
  + Complexity: Average time complexity of O(n log n), worst case O(n^2) (rare). Space complexity is O(log n) due to the recursive stack. Very efficient for large datasets.
* **Merge Sort:**
  + Description: Divides the array into halves, sorts them, and merges the sorted halves.
  + Complexity: Consistent O(n log n) time complexity. Space complexity is O(n) due to the need for a temporary array. Stable and ideal for large datasets.

**Analysis:**

**Performance Comparison: Bubble Sort vs. Quick Sort**

**Bubble Sort:**

* **Time Complexity:**
  + **Worst Case:** O(n^2)
  + **Average Case:** O(n^2)
  + **Best Case:** O(n) (when the array is already sorted)
* **Space Complexity:** O(1) (in-place sorting)

**Quick Sort:**

* **Time Complexity:**
  + **Worst Case:** O(n^2) (when the smallest or largest element is always chosen as the pivot)
  + **Average Case:** O(n log n)
  + **Best Case:** O(n log n)
* **Space Complexity:** O(log n) (due to the recursive stack)
* **Why Quick Sort is Generally Preferred Over Bubble Sort?**

Quick Sort is generally preferred over Bubble Sort due to its significantly better average-case time complexity of O(n log n), compared to Bubble Sort's O(n^2). Quick Sort efficiently handles large datasets and is adaptable with optimizations like better pivot selection. Despite its worst-case scenario of O(n^2), this is rare with good pivot choices. Bubble Sort, on the other hand, performs poorly for larger datasets and offers limited optimization potential. Its inherent design of comparing and swapping adjacent elements makes it impractical for anything beyond small or nearly sorted arrays.