Sorting Customer Orders

1. **Explain different sorting algorithms (Bubble Sort, Insertion Sort, Quick Sort, Merge Sort)**

   1. Bubble Sort

- Compares and swaps adjacent elements repeatedly to "bubble" the largest to the end.
- Easy to implement but inefficient for large lists.
- Time complexity: Best O(n), Average/Worst O(n²)

2. Insertion Sort

- Builds sorted list one item at a time by inserting each into the correct position.
- Works well for small or nearly sorted arrays.
- Time complexity: Best O(n), Average/Worst O(n²)

3. Quick Sort

- Picks a pivot, partitions array into smaller and greater elements, and sorts recursively.
- Very efficient on average but worst-case can occur with bad pivots.
- Time complexity: Best/Average O(n log n), Worst O(n²)

4. Merge Sort

- Divides array into halves, sorts them recursively, then merges the sorted halves.
- Stable and consistent performance, but uses extra memory.
- Time complexity: Best/Average/Worst O(n log n)

2. **Compare the performance (time complexity) of Bubble Sort and Quick Sort.**

   Bubble sort:
   Best case : O(n)
   Average Case : O(n²)
   Worst Case : O(n²)

   Quick sort:
   Best case : O(nlog n)
   Average Case : O(n log n)
   Worst Case : O(n²)

3. **Discuss why Quick Sort is generally preferred over Bubble Sort.**
   - Much faster for large datasets due to divide-and-conquer.
   - In-place (doesn't require additional arrays).
   - Widely used in real-world libraries (Java’s Arrays.sort() uses dual-pivot quicksort for primitives).