**Sorting Customers Orders**

1. **Different Sorting algorithms.**

Sorting is essential in organizing data for faster processing, improved user experience, and efficient decision-making (e.g., prioritizing high-value orders).

* Bubble Sort:
  + Repeatedly swaps adjacent elements if they are in the wrong order.
  + Time Complexity: O(n^2) in worst and average cases.
  + Easy to implement but inefficient for large datasets.
* Insertion Sort:
  + Builds the sorted array one item at a time.
  + Time Complexity: O(n^2)
  + Efficient for small or nearly sorted datasets.
* Quick Sort:
  + Divides the array into partitions based on a pivot.
  + Average Time Complexity: O(n log n)
  + One of the fastest comparison-based sorts in practice.
* Merge Sort:
  + Recursively splits the array and merges sorted parts.
  + Time Complexity: O(n log n)
  + Good for linked lists and stable sorting.

1. **Compare the performance of Bubble sort and Quick sort.**

#### **Bubble Sort**

* Best Case: O(n) — when the array is already sorted.
* Average Case: O(n²) — due to nested comparisons.
* Worst Case: O(n²) — when the array is sorted in reverse.
* Space Complexity: O(1) — it’s an in-place sorting algorithm.
* Stability: Yes — it maintains the relative order of equal elements.
* Use Case: Suitable only for small datasets or educational purposes.

#### **Quick Sort**

* Best Case: O(n log n) — happens when the pivot divides the array evenly.
* Average Case: O(n log n) — very efficient in most real-world scenarios.
* Worst Case: O(n²) — occurs when the smallest or largest element is always chosen as pivot.
* Space Complexity: O(log n) — due to recursive calls on the call stack.
* Stability: No — equal elements may not preserve original order unless modified.
* Use Case: Ideal for large datasets and is widely used in practice due to fast performance.

**3. Why Quick sort is preferred over Bubble sort?**

* Quick Sort outperforms Bubble Sort in terms of performance, especially for large datasets. Its average-case time complexity of O(n log n) allows it to sort data much faster compared to Bubble Sort's O(n²) complexity. In terms of efficiency, Quick Sort minimizes the number of comparisons and swaps, making it more optimal for real-world scenarios.
* When it comes to scalability, Quick Sort is well-suited for handling large-scale orders and big data, whereas Bubble Sort becomes impractical due to its quadratic time growth. As a result, Quick Sort is widely adopted in many standard libraries and real-time systems where fast sorting is required, making it a preferred choice in production-level applications.