3. Sorting Customers Orders

Understanding the sorting algorithms

1. **Bubble Sort**:

- **Description**: Bubble Sort is a simple comparison-based sorting algorithm. It repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The process is repeated until the list is sorted.
- **Time Complexity**:
  - Best Case: O(n)
  - Average Case: O(n^2)
  - Worst Case: O(n^2)
- **Space Complexity**: O(1)

. 2. **Insertion Sort:**

- **Description:** Insertion Sort builds the sorted array one item at a time. It picks each element from the unsorted portion and inserts it into the correct position in the sorted portion.
- **Time Complexity**:
  - Best Case**: O(n)**
  - **Average Case: O(n^2)**
  - **Worst Case: O(n^2)**

Space Complexity:
  - O(1) (in-place sort)

3. **Quick Sort:**

- **Description**: Quick Sort is a divide-and-conquer algorithm. It selects a 'pivot' element from the array and partitions 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**:
  - Best Case: O(n log n)
  - Average Case: O(n log n)
  - Worst Case: O(n^2) (rare, happens when the smallest or largest element is always chosen as the pivot)
- **Space Complexity**: O(log n) (due to recursive stack space)

4. **Merge Sort**:

- **Description**: Merge Sort is a divide-and-conquer algorithm. It divides the array into halves, recursively sorts each half, and then merges the sorted halves back together.
- **Time Complexity**:
  - Best Case: O(n log n)
  - Average Case: O(n log n)
  - Worst Case: O(n log n)
- **Space Complexity**: O(n)

Analysis

Time Complexity:

- **Bubble Sort**:
    - **Best Case**: O(n) - When the array is already sorted.
    - **Average Case**: O(n^2) - Due to the nested loops.
    - **Worst Case**: O(n^2) - When the array is sorted in reverse order.
- **Quick Sort**:
    - **Best Case**: O(nlogn) - When the pivot divides the array into two nearly equal halves.
    - **Average Case**: O(nlogn) - On average, due to balanced partitions.
    - **Worst Case**: O(n^2) - When the pivot is the smallest or largest element repeatedly (mitigated by good pivot selection strategies like choosing the median).

Why Quick Sort is Generally Preferred Over Bubble Sort

- **Efficiency**: Quick Sort is generally more efficient with an average-case time complexity of O(nlogn), compared to Bubble Sort's O(n^2). This makes Quick Sort more suitable for larger datasets.

- **Performance**: Even in the worst case, with optimisations like choosing a better pivot, Quick Sort can avoid its O(n^2)
  O(n^2) worst-case performance. Techniques like randomised Quick Sort or choosing the median of three can help achieve this.

- **Practical Use**: Quick Sort is often faster in practice due to better cache performance and lower constant factors compared to other O(nlogn) algorithms like Merge Sort.