**Sorting Customer Orders**

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

**Bubble Sort:**

* Bubble Sort is a simple sorting algorithm that compares each pair of adjacent elements and swaps them if they are in the wrong order. This process repeats until the entire list is sorted.
* Time Complexity:
  + Best-case: O(n) - When the list is already sorted.
  + Average-case: O(n^2) - When the list is randomly ordered.
  + Worst-case: O(n^2) - When the list is sorted in reverse order.

**Insertion Sort:**

* Insertion Sort builds the sorted list one item at a time by repeatedly picking the next item and inserting it into its correct position among the already sorted items.
* Time Complexity:
  + Best-case: O(n) - When the list is already sorted.
  + Average-case: O(n^2) - When the list is randomly ordered.
  + Worst-case: O(n^2) - When the list is sorted in reverse order.

**Quick Sort:**

* Quick Sort is a divide-and-conquer algorithm that works by selecting a 'pivot' element and partitioning the array 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) - When the pivot divides the list into two nearly equal parts.
  + Average-case: O (n log n) - On average, due to good pivots.
  + Worst-case: O(n^2) - When the pivot is the smallest or largest element every time (for example already sorted list).

**Merge Sort:**

* Merge Sort is a divide-and-conquer algorithm that divides the list into halves, sorts each half, and then merges the sorted halves back together.
* Time Complexity:

For a given size input, it always does the same sequence of comparisons. That’s why it has same time complexity O(n log n) for best-case, average-case and worst-case.

**Time Complexity Analysis:**

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

**Bubble Sort** is slow. It checks each number with the next one and swaps them if they are in the wrong order. It does this many times. As the list gets bigger, it takes much longer [O(n^2) time complexity].

**Quick Sort** is much faster. It divides the list into smaller parts and sorts those parts. Then it combines them [O(n log n) average time complexity].

So, for small lists **Bubble Sort** might be okay, but for big lists **Quick Sort** is way better.

* 1. **Discuss why Quick Sort is generally preferred over Bubble Sort.**

Though **Bubble Sort** is easy to understand and implement but inefficient for large lists due to its O(n^2) time complexity. Whereas, **Quick Sort** is more complex but generally much faster due to its O(n log n) average time complexity. That’s why it is preferred over Bubble Sort for larger datasets.