**Exercise 3 : Sorting Customer Orders**

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

1. Bubble Sort:

Bubble sort repeatedly steps through the list, compares adjacent items, and swaps them if they are in the wrong order. This process is repeated until the list is sorted.

Time Complexity:

* Best Case: O(n) (when the list is already sorted)
* Average Case: O(n^2)
* Worst Case: O(n^2)

1. Insertion Sort:

Insertion sort builds the final sorted array one item at a time. It picks elements from the unsorted portion and inserts them into their correct position in the sorted portion.

Time Complexity:

* Best Case: O(n) (when the list is already sorted)
* Average Case: O(n^2)
* Worst Case: O(n^2)

1. Quick Sort:

Divides the array into two smaller sub-arrays based on a pivot element. Elements less than the pivot go to one sub-array, and elements greater go to the other. 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) (when the pivot is the smallest or largest element)

1. Merge Sort:

Divides the array into halves, sorts each half, and then merges the sorted halves to produce the sorted array.

Time Complexity:

* Best Case: O(n log n)
* Average Case: O(n log n)
* Worst Case: O(n log n)

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

**Bubble Sort:**

Bubble sort is simple to implement but it is inefficient for large datasets due to its quadratic time complexity.

The best case is O(n) but the average and worst cases is O(n^2).

**Quick Sort:**

Quick sort is much better than bubble sort for large datasets as it uses divide and Conquer approach.

The average and best cases are O(n log n) where as the worst case is O(n^2).

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

Quick Sort generally performs better than Bubble Sort, especially for large datasets, due to its average time complexity of O(n log n) compared to Bubble Sort’s O(n^2).

The divide-and-conquer strategy of Quick Sort reduces the problem size with each recursive call, leading to more efficient sorting.