**Exercise 3: Sorting Customer Orders**

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

1. Bubble Sort: A simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order.

Time Complexity:

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

1. Insertion Sort: Builds the final sorted array one item at a time. It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort.

Time complexity:

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

1. Quick Sort: A divide-and-conquer algorithm. It works by selecting a 'pivot' element from the array and partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot.

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 in each partition)

1. Merge sort: A divide-and-conquer algorithm that was invented by John von Neumann in 1945. It works by dividing the unsorted list into n sub-lists, each containing one element, and then repeatedly merging sub-lists to produce new sorted sub-lists until there is only one sub-list remaining.

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:

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

Quick Sort:

* **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 in each partition)

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

Quick Sort has an average and best-case time complexity of O(n log n), which is significantly better than Bubble Sort’s O(n^2). Quick Sort performs better in practice compared to Bubble Sort due to its efficient partitioning. Quick Sort can be implemented in-place with O(log n) extra space, whereas Bubble Sort requires O(1) extra space but is much slower.