**Exercise 3: Sorting Customer Orders**

**Bubble Sort**

Bubble Sort is a simple sorting algorithm that repeatedly compares and swaps adjacent elements if they are in the wrong order. It continues to pass through the list until no more swaps are needed. Although easy to understand, it is inefficient for large datasets.

**Insertion Sort**

Insertion Sort builds the final sorted array one item at a time. It is efficient for small datasets or nearly sorted data but performs poorly on large or random datasets.

**Quick Sort**

Quick Sort is a divide-and-conquer algorithm. It picks a pivot element, partitions the array around the pivot, and then recursively sorts the sub-arrays. It is faster and more efficient than Bubble or Insertion Sort for large datasets.

**Merge Sort**

Merge Sort also follows the divide-and-conquer strategy. It splits the array into halves, recursively sorts them, and merges the sorted halves. It guarantees stable and predictable performance.

**Implementation**

**Bubble Sort**

Used to sort orders by total price by comparing and swapping adjacent orders repeatedly until sorted. Simple but inefficient for large inputs.

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

**Quick Sort**

Use a pivot to partition orders into two parts. Recursively sorts each part. Very efficient for large datasets and generally faster than Bubble Sort.

* Best Case: O(n log n)
* Average Case: O(n log n)
* Worst Case: O(n^2) (rare, depends on pivot selection)

Quick Sort is generally preferred over Bubble Sort because:

* It handles large datasets more efficiently.
* It reduces the number of comparisons and swaps.
* It has a better average-case time complexity.
* In practice, it performs well and is widely used in many applications.

Bubble Sort, while easier to understand, is too slow and inefficient for most real-world sorting tasks, especially when performance is important.