**Exercise 3: Sorting Customer Orders**

In e-commerce, sorting algorithms are critical to the prioritization of high-value orders. Bubble sort is inefficient for large datasets because it continually swaps neighboring entries until the list is sorted. Ideal for small datasets or lists that are almost sorted, insertion sort creates a sorted array one element at a time. Quick sort is effective in sorting the array recursively by splitting it into smaller sub-arrays, but its effectiveness is dependent on the pivot selection. With consistent performance across all input data, merge sort splits the array in half, sorts each half recursively, and then merges the sorted halves.

Bubble Sort's performance gradually deteriorates as the number of elements increases due to its O(n^2) time complexity. Because of this, sorting huge datasets like client orders is not feasible.

Larger datasets can benefit greatly from Quick Sort's O(n log n) time complexity, which is much faster than Bubble Sort's. In worst-case situations, it can decline to O(n^2), although this is rather uncommon. Quick Sort is the recommended option for sorting customer orders by total price because of its divide-and-conquer strategy, which enables it to handle enormous amounts of data efficiently.

In real-world applications, Quick Sort is frequently the preferred algorithm for sorting big datasets because of its exceptional performance.