**Exercise 3: Sorting Customer Orders**

**Scenario:**

You are tasked with sorting customer orders by their total price on an e-commerce platform. This helps in prioritizing high-value orders.

**Different Sorting Algorithms:**

1. **Bubble Sort:**
   * Repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order.
   * Time Complexity: O(n^2)
2. **Insertion Sort:**
   * Builds the final sorted array one item at a time, picking the next element and inserting it into its correct position.
   * Time Complexity: O(n^2)
3. **Quick Sort:**
   * Selects a 'pivot' element and partitions the array into elements less than and greater than the pivot, then recursively sorts the subarrays.
   * Time Complexity: O(nlogn) on average
4. **Merge Sort:**
   * Divides the array into halves, recursively sorts them, and then merges the sorted halves.
   * Time Complexity: O(nlogn)

**Performance Comparison (Bubble Sort vs. Quick Sort):**

* **Bubble Sort:**
  + Time Complexity: O(n^2)
  + Simple but inefficient for large datasets due to the high number of comparisons and swaps.
* **Quick Sort:**
  + Time Complexity: O(nlogn) on average, O(n^2) in the worst case
  + Efficient for large datasets, fewer comparisons and swaps, and better use of memory hierarchy.

**Why Quick Sort is Generally Preferred Over Bubble Sort:**

* Quick Sort is generally preferred because it is much more efficient for large datasets, with an average time complexity of O(n log n)compared to Bubble Sort's O(n^2). This results in significantly faster sorting for larger arrays.