**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.

**Steps:**

**1. Understand Sorting Algorithms**

**Bubble Sort**

Bubble Sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process is repeated until the list is sorted. The algorithm gets its name because smaller elements "bubble" to the top of the list.

* **Time Complexity:**
  + Worst-case: O(n^2)
  + Average-case: O(n^2)
  + Best-case: O(n) (when the list is already sorted)

**Quick Sort**

Quick Sort is 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. The sub-arrays are then sorted recursively.

* **Time Complexity:**
  + Worst-case: O(n^2) (when the smallest or largest element is always chosen as the pivot)
  + Average-case: O(n log n)
  + Best-case: O(n log n)

**Insertion Sort**

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

**Merge Sort**

Merge Sort is a divide-and-conquer algorithm that was invented by John von Neumann in 1945. It divides the unsorted list into n sub-lists, each containing one element, then repeatedly merges sub-lists to produce new sorted sub-lists until there is only one sub-list remaining.

* **Time Complexity:**
  + Worst-case: O(n log n)
  + Average-case: O(n log n)
  + Best-case: O(n log n)

**Analysis**

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

* **Bubble Sort:**
  + Simple to implement and understand.
  + Inefficient for large datasets due to O(n^2) time complexity.
  + Best used for small datasets or nearly sorted arrays.
* **Quick Sort:**
  + More complex to implement but much more efficient.
  + Average-case and best-case time complexity is O(n log n).
  + In practice, Quick Sort is faster than Bubble Sort for large datasets.

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

* **Efficiency:** Quick Sort is generally more efficient than Bubble Sort, especially for large datasets, due to its O(n log n) average-case time complexity compared to Bubble Sort's O(n^2).
* **Scalability:** Quick Sort scales better with the size of the input array. While Bubble Sort may be useful for small or nearly sorted arrays, its performance significantly degrades with larger datasets.
* **Practical Use:** Quick Sort is widely used in practice due to its efficiency and is implemented in various standard libraries and frameworks.