**Exercise 3: Sorting Customer Orders**

**1. Understand Sorting Algorithms**

Sorting algorithms arrange data in a particular order, typically ascending or descending. Each sorting algorithm has its advantages and trade-offs in terms of complexity and performance. Let's explore four popular sorting algorithms: Bubble Sort, Insertion Sort, Quick Sort, and Merge Sort.

**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. The pass through the list is repeated until the list is sorted.

* **Algorithm**:
  1. Compare each pair of adjacent elements.
  2. Swap them if they are in the wrong order.
  3. Repeat the process for each element until no more swaps are needed.
* **Characteristics**:
  1. **Time Complexity**:
     + Best Case: O(n)
     + Average Case: O(n²)
     + Worst Case: O(n²)
  2. **Space Complexity**: O(1)
  3. **Stability**: Yes
* **Pros**: Easy to understand and implement.
* **Cons**: Inefficient on large datasets; performs poorly compared to other algorithms.

**Insertion Sort**

**Insertion Sort** is a simple and intuitive sorting algorithm that 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.

* **Algorithm**:
  1. Start with the second element .
  2. Compare it with the elements before it and insert it into its correct position.
  3. Repeat the process for all elements.
* **Characteristics**:
  1. **Time Complexity**:
     + Best Case: O(n) (when the list is already sorted)
     + Average Case: O(n²)
     + Worst Case: O(n²)
  2. **Space Complexity**: O(1) (in-place sorting)
  3. **Stability**: Yes
* **Pros**: Efficient for small datasets; works well with partially sorted data.
* **Cons**: Not suitable for large datasets due to its O(n²) time complexity.

**Quick Sort**

**Quick Sort** is a highly efficient sorting algorithm and is based on partitioning an array into sub-arrays. It is a divide-and-conquer algorithm that recursively sorts sub-arrays.

* **Algorithm**:
  1. Choose a "pivot" element from the array.
  2. Partition the array into two halves around the pivot.
  3. Recursively apply the same process to the sub-arrays.
* **Characteristics**:
  1. **Time Complexity**:
     + Best Case: O(n log n)
     + Average Case: O(n log n)
     + Worst Case: O(n
  2. **Space Complexity**: O(log n)
* **Pros**: Fast and efficient on average; suitable for large datasets.
* **Cons**: Worst-case time complexity is O(n²) but can be mitigated with good pivot selection strategies.

**Merge Sort**

**Merge Sort** is an efficient, stable, comparison-based, divide-and-conquer sorting algorithm. Most implementations produce a stable sort, meaning that the order of equal elements is preserved.

* **Algorithm**:
  1. Divide the unsorted list into n sublists, each containing one element.
  2. Repeatedly merge sublists to produce new sorted sublists until there is only one sublist remaining.
* **Characteristics**:
  1. **Time Complexity**:
     + Best Case: O(n log n)
     + Average Case: O(n log n)
     + Worst Case: O(n log n)
  2. **Space Complexity**: O(n)
* **Pros**: Always O(n log n) time complexity; stable and works well with large datasets.
* **Cons**: Requires additional space proportional to the input size.

**2. Analysis**

**Time Complexity Comparison**

* **Bubble Sort**:
  + Best Case: O(n) (when the list is already sorted)
  + Average Case: O(n²)
  + Worst Case: O(n²)
* **Quick Sort**:
  + Best Case: O(n log n)
  + Average Case: O(n log n)
  + Worst Case: O(n²) (poor pivot choice)

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

1. **Efficiency**: Quick Sort is generally more efficient with an average time complexity of O(n log n), making it suitable for large datasets. In contrast, Bubble Sort is inefficient on large datasets due to its O(n²) average and worst-case complexity.
2. **Performance**: Quick Sort's divide-and-conquer approach allows it to efficiently handle large datasets, while Bubble Sort performs unnecessary comparisons and swaps, making it slower.
3. **Practical Use**: Quick Sort is widely used in practical applications where efficiency is crucial, whereas Bubble Sort is mainly used for educational purposes to illustrate basic sorting concepts.