**ANSWER**

**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:**
   * **Explain different sorting algorithms (Bubble Sort, Insertion Sort, Quick Sort, Merge Sort).**

**Bubble Sort**:

* **Description**: A simple comparison-based sorting algorithm. It 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.
* **Time Complexity**:
  + Best Case: O(n)
  + Average Case: O(n^2)
  + Worst Case: O(n^2)

**Insertion Sort**:

* **Description**: Builds the final sorted array one item at a time. It takes each element from the input data and finds the appropriate position within the sorted part of the array.
* **Time Complexity**:
  + Best Case: O(n)
  + Average Case: O(n^2)
  + Worst Case: O(n^2)

**Quick Sort**:

* **Description**: A divide-and-conquer algorithm. It picks a pivot element and partitions the array into two halves, placing elements smaller than the pivot on one side and larger elements on the other. The process is recursively applied to the two halves.
* **Time Complexity**:
  + Best Case: O(n log n)
  + Average Case: O(n log n)
  + Worst Case: O(n^2)

**Merge Sort**:

* **Description**: Another divide-and-conquer algorithm. It divides the array into two halves, recursively sorts them, and then merges the sorted halves.
* **Time Complexity**:
  + Best Case: O(n log n)
  + Average Case: O(n log n)
  + Worst Case: O(n log n)

1. **Analysis:**
   * Compare the performance (time complexity) of Bubble Sort and Quick Sort.

**Bubble Sort**:

* Best Case: O(n)
* Average Case: O(n^2)
* Worst Case: O(n^2)
* Bubble Sort is simple but inefficient for large datasets due to its quadratic time complexity.

**Quick Sort**:

* Best Case: O(n log n)
* Average Case: O(n log n)
* Worst Case: O(n^2) (rare, typically mitigated by good pivot selection)
* Quick Sort is generally efficient for large datasets due to its logarithmic complexity, though it can degrade to O(n^2) in the worst case (e.g., with a poor pivot choice).
  + Discuss why Quick Sort is generally preferred over Bubble Sort.

Quick Sort is generally preferred over Bubble Sort because:

* **Efficiency**: Quick Sort has an average time complexity of O(n log n), making it much faster than Bubble Sort's O(n^2) for large datasets.
* **Scalability**: Quick Sort handles large datasets more effectively, making it suitable for real-world applications.
* **Divide-and-Conquer**: The divide-and-conquer approach of Quick Sort makes it adaptable and efficient, often outperforming other O(n log n) algorithms like Merge Sort in practice due to better cache performance and lower constant factors.