### **Step 1: Understand Sorting Algorithms**

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

Sorting algorithms are essential for organizing data so that it can be accessed, analyzed, and processed efficiently. In e-commerce, sorting customer orders by price helps prioritize high-value transactions — this supports quicker delivery, risk detection, and better customer satisfaction.

Sorting Algorithms:

**1. Bubble Sort**

* Compares each pair of adjacent elements and swaps them if they are in the wrong order.
* Like repeatedly bubbling the largest value to the end.
* Example: comparing prices of each pair of orders repeatedly until the biggest price settles at the end — like bubbles floating to the surface.

**Advantages**:

* Simple and easy to write
* Good for educational purposes.

**Disadvantages**:

* Very slow for large datasets
* Many unnecessary comparisons

**Time Complexity**:

* Best: O(n) [when already sorted]
* Average/Worst: O(n²)

**2. Insertion Sort**

* Best for small or nearly sorted data.
* Example: sorting playing cards in our hand,we pick one card at a time and place it in its correct position.

**Advantages**:

* Efficient for small or nearly sorted data
* Simple to understand

**Disadvantages**:

* Inefficient for large datasets

**Time Complexity**:

* Best: O(n)
* Average/Worst: O(n²)

**3. Quick Sort**

* Uses a pivot element to divide the array into two partitions: left (smaller) and right (greater).
* Recursively sorts the partitions.

**Advantages**:

* Very fast for large datasets
* Efficient memory usage (in-place sorting)

**Disadvantages**:

* Worst case is O(n²), but happens rarely if pivot is bad
* Not stable (equal elements may swap)

**Time Complexity**:

* Best: O(n log n)
* Average: O(n log n)
* Worst: O(n²)

**4. Merge Sort**

* Recursively divides the array into halves, sorts each half, and merges them.

**Advantages**:

* Very stable and consistent
* Good for very large data (can be external sort too)

**Disadvantages**:

* Requires additional memory
* Slower than Quick Sort in most real-world scenarios

**Time Complexity**:

* Best/Average/Worst: O(n log n)

### **Step 4: Analysis**

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

| Algorithm | Best | Average | Worst |
| --- | --- | --- | --- |
| Bubble Sort | O(n) | O(n²) | O(n²) |
| Quick Sort | O(n log n) | O(n log n) | O(n²) |

1. Discuss why Quick Sort is generally preferred over Bubble Sort.

* **Speed & Efficiency**:
* Quick Sort is exponentially faster on average. For 1,000 orders, Bubble Sort might take 1,000,000 comparisons, while Quick Sort only around 10,000–12,000.
* **Scalability**:
* E-commerce platforms handle **thousands or millions of orders**. Quick Sort handles large lists far better.
* **Fewer Swaps & Comparisons**:
* Quick Sort smartly partitions data instead of brute-force swapping every pair.
* **Memory Efficient**:
* Unlike Merge Sort, Quick Sort sorts in place (less memory used).
* **Real-World Impact**:
* Prioritizing high-value orders quickly can reduce fraud risk, optimize delivery, and enhance user experience.