UNDERSTANDING

Sorting Algorithms

Sorting algorithms are essential for organizing data in a particular order, such as ascending or descending by price, name, or ID. In the context of e-commerce, sorting orders by totalPrice helps in prioritizing high-value orders.

There are various sorting algorithms, each with different time complexities and use cases.

* **Bubble Sort** is a simple algorithm that repeatedly compares and swaps adjacent elements if they are in the wrong order. It is easy to understand but inefficient for large datasets, with a worst-case time complexity of **O(n²)**.
* **Insertion Sort** builds the final sorted array one item at a time, placing each element in its correct position. Like Bubble Sort, it performs well on small or nearly sorted datasets but is also **O(n²)** in the worst case.
* **Quick Sort**, on the other hand, is a highly efficient **divide-and-conquer algorithm** that selects a pivot element and partitions the array into two halves , elements less than the pivot and those greater , then recursively sorts the partitions. It has an average-case time complexity of **O(n log n)**.
* **Merge Sort** also follows the divide-and-conquer approach by dividing the array into halves, sorting them recursively, and merging the results. Though **Merge Sort** guarantees **O(n log n)** performance even in the worst case, it requires extra space, unlike Quick Sort, which is generally **more space-efficient** and faster in practice.

ANALYSIS

Comparison of Bubble Sort and Quick Sort

**Bubble Sort** is easy to implement and understand but becomes highly inefficient as the number of elements increases, with both its average and worst-case time complexity being **O(n²)**. This makes it unsuitable for sorting large datasets like customer orders on an e-commerce platform.

**Quick Sort**, in contrast, is much more efficient with an average-case time complexity of **O(n log n)**, thanks to its divide-and-conquer approach. While the worst-case performance of Quick Sort is also **O(n²)** (when the pivot is poorly chosen), this can be minimized using techniques like random pivot selection or choosing the median.

Additionally, Quick Sort is an **in-place algorithm**, meaning it requires very little additional memory compared to Merge Sort.

Due to its superior efficiency on large datasets and better real-world performance, **Quick Sort is generally preferred over Bubble Sort** for sorting operations in platforms where speed and scalability are essential.