Understand Sorting Algorithms

Bubble Sort:

- **Algorithm**: Repeatedly compare adjacent elements and swap them if they are in the wrong order. Continue doing this until no more swaps are needed.
- **Time Complexity**: Worst and Average cases: O(n^2)
  Best case: O(n) (when the array is already sorted)

Insertion Sort:

- **Algorithm**: Build the sorted array one item at a time by repeatedly picking the next item and inserting it into the correct position among the previously sorted items.
- **Time Complexity**: Worst and Average cases: O(n^2)
  Best case: O(n) (when the array is already sorted)

Quick Sort:

- **Algorithm**: Choose a pivot element, partition the array into elements less than and greater than the pivot, and recursively apply the same process to the partitions.
- **Time Complexity**: Worst case: O(n^2) (when the pivot is the smallest or largest element every time)
  Average case: O(n log n)

Merge Sort:

- **Algorithm**: Divide the array into halves, recursively sort each half, and then merge the sorted halves back together.
- **Time Complexity**: Worst, Average and Best cases: O(n log n)

Performance Comparison

Why Quick Sort is Preferred:

- **Efficiency**: Quick Sort has better average-case performance (O(n log n)) compared to Bubble Sort's O(n^2).
- **Scalability**: Quick Sort handles larger datasets more efficiently, making it more suitable for performance-critical applications like sorting customer orders on an e-commerce platform.