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

**Bubble Sort**  
Bubble Sort is a simple comparison-based algorithm. It repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process continues until the list is sorted. Although easy to understand and implement, Bubble Sort is inefficient for large datasets due to its average and worst case time complexity of O(n²).

**Insertion Sort**  
Insertion Sort builds the final sorted array one element at a time. It picks each element from the unsorted portion and places it in the correct position in the sorted portion by comparing it with already sorted elements. It performs well for small datasets or nearly sorted data, with an average and worst-case time complexity of O(n²).

**Quick Sort**  
Quick Sort is an efficient divide-and-conquer algorithm. It selects a ‘pivot’ element and partitions the array such that elements less than the pivot go to the left and greater ones go to the right. This process is recursively applied to the subarrays. Quick Sort generally performs very well with an average time complexity of O(nlogn), although its worst case is O(n²) when poorly balanced pivots are chosen.

**Merge Sort**  
Merge Sort is a stable and consistent divide-and-conquer algorithm. It recursively divides the list into halves until individual elements remain, and then merges the sorted halves to form the final sorted array. It guarantees O(n log n) time complexity in all cases but uses additional memory for merging, making its space complexity O(n).

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

**Bubble Sort:**  
Best Case: O(n) - when the array is already sorted.  
Average Case: O(n²)  
Worst Case: O(n²)  
Bubble Sort compares and swaps adjacent elements, which becomes inefficient as the input size increases.

**Quick Sort:**  
Best Case: O(nlogn) - when the pivot divides the array evenly.  
Average Case: O(nlogn)  
Worst Case: O(n²) - occurs when the pivot is always the smallest or largest element.  
Quick Sort uses a divide-and-conquer approach, which generally leads to much faster sorting than Bubble Sort.

**Discuss why Quick Sort is generally preferred over Bubble Sort**

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

Quick Sort is generally preferred because it is significantly faster for large datasets. While Bubble Sort performs repeated passes over the array and makes many unnecessary comparisons and swaps, Quick Sort efficiently partitions the array and reduces the problem size at each step. Its average case time complexity of O(nlogn) makes it suitable for real world applications, whereas Bubble Sort is mostly used for educational purposes due to its simplicity.