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

Bubble Sort-

Description: A simple comparison-based algorithm. It repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The process is repeated until the list is sorted.

Time Complexity:

Worst-case: O(n²)

Average-case: O(n²)

Best-case: O(n) (when the list is already sorted)

Space Complexity: O(1) (in-place sorting)

Stability: Yes

Insertion Sort-

Description: Builds the sorted list one item at a time. It takes each element from the input and finds its appropriate position within the sorted list.

Time Complexity:

Worst-case: O(n²)

Average-case: O(n²)

Best-case: O(n) (when the list is already sorted)

Space Complexity: O(1) (in-place sorting)

Stability: Yes

Quick Sort-

Description: A divide-and-conquer algorithm. It selects a 'pivot' element from the array and partitions the other elements into two sub-arrays, according to whether they are less than or greater than the pivot. The sub-arrays are then sorted recursively.

Time Complexity:

Worst-case: O(n²) (rare, happens when the smallest or largest element is always chosen as the pivot)

Average-case: O(n log n)

Best-case: O(n log n)

Space Complexity: O(log n) (due to recursive stack space)

Stability: No (depends on implementation)

Merge Sort-

Description: Another divide-and-conquer algorithm. It divides the array into two halves, sorts them, and then merges the sorted halves.

Time Complexity:

Worst-case: O(n log n)

Average-case: O(n log n)

Best-case: O(n log n)

Space Complexity: O(n) (requires additional memory for merging)

Stability: Yes.

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

Bubble Sort: O(n^2) in both the worst and average cases. This is because Bubble Sort repeatedly traverses the list to swap elements.

Quick Sort: O(n log n) on average and O(n^2) in the worst case. However, with good pivot selection (e.g., median-of-three or randomized pivot), the worst case is rare .So quick sort is generally preffered.

**II) Discuss why Quick Sort is generally preferred over Bubble Sort.**

Quick Sort generally performs better than Bubble Sort for large datasets due to its average-case time complexity of O(n log n).

Memory Usage: Although Quick Sort is not an in-place sort due to recursive calls, its space complexity of O(log n) is much better than the O(n) of Merge Sort .Quick Sort handles large datasets more efficiently, making it more suitable for practical use in systems where performance is critical.

In conclusion, while Bubble Sort is easy to understand and implement, Quick Sort is generally preferred for sorting large datasets due to its better average-case performance and efficiency.