**Step 1: Understand Sorting Algorithms: -**

Q1) Explain different sorting algorithms (Bubble Sort, Insertion Sort, Quick Sort, Merge Sort).

Solution: -

->Bubble Sort: This algorithm repeatedly compares adjacent elements and swaps them if they are in the wrong order.

It continues until the array is sorted, but it’s not very efficient for large array.

->Insertion Sort: It builds the sorted array one item at a time by comparing each new item to the already sorted items and inserting it in the correct position.

It’s simple and efficient for small or nearly sorted arrays.

->Quick Sort: This algorithm picks a “pivot” element and partitions the array into two sub-arrays.

I. Elements less than the pivot

II. Elements greater than the pivot.

It then recursively sorts the sub-arrays, making it very efficient for large arrays.

->Merge Sort: It divides the array into two halves, recursively sorts each half, and then merges the sorted halves back together.

This algorithm is efficient and stable, but it requires additional memory for the merging process.

**Step 4: Analysis: -**

Q1) Compare the performance (time complexity) of Bubble Sort and Quick Sort.

Solution: -

Time complexity of Bubble Sort:

->Best Case: O(n)

->Average Case: O(n^2)

->Worst Case: O(n^2)

Time complexity of Quick Sort:

->Best Case: O (n log n)

->Average Case: O (n log n)

->Worst Case: O(n^2)

Q2) Discuss why Quick Sort is generally preferred over Bubble Sort.

Solution: -

Quick sort is generally preferred over Bubble Sort because:

->More efficient for larger datasets.

->Has better time complexity then Bubble Sort.

->It sorts the array in-place.

->Performance can be improved with optimizations, such as choosing a better pivot element.