1)

Sorting algorithms are fundamental techniques in computer science used to reorder elements in a list based on specific criteria.

* 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 is repeated until the list is sorted.

Time complexity:- O(n^2)(Worst)

* Insertion Sort :- Insertion Sort builds the final sorted array one item at a time. It takes each element from the input data and finds the appropriate place within the sorted list, then shifts the remaining elements to make space.
* Time complexity:- O(n^2)(Worst)
* Quick Sort:- Quick Sort is a divide-and-conquer algorithm. It selects a 'pivot' element and partitions the array into two halves such that elements less than the pivot are on the left, and elements greater than the pivot are on the right. It then recursively sorts the sub-arrays.
* Time complexity:- O(n^2)(Worst)

Merge Sort:- Merge Sort is another divide-and-conquer algorithm. It splits the array into halves, recursively sorts each half, and then merges the sorted halves back together.

Time complexity:-O(nlogn) (Worst)

4)

**Bubble Sort**

* **Best Case**: O(n)
  + This occurs when the array is already sorted. Only one pass through the array is needed to confirm that it is sorted.
* **Average Case**: O(n^2)
  + On average, each element needs to be compared with every other element, leading to quadratic time complexity.
* **Worst Case**: O(n^2)
  + This occurs when the array is sorted in reverse order. Each element will need to be compared and swapped with every other element.

**Quick Sort**

* **Best Case**: O(n log n)
  + The best case occurs when the pivot selection divides the array into two nearly equal halves at each step.
* **Average Case**: O(n log n)
  + On average, the pivot selection tends to split the array into reasonably balanced partitions, leading to logarithmic recursive depth with linear work per level.
* **Worst Case**: O(n^2)
  + The worst case occurs when the pivot selection results in the most unbalanced partitions possible.

Quick Sort is Generally Preferred Over Bubble Sort because:-

a)Efficiency :- Quick Sort is significantly more efficient than Bubble Sort for large datasets.

b)Memory Usage :- they do not require additional memory proportional to the input size. However, Quick Sort’s divide-and-conquer strategy is more efficient in terms of reducing the problem size quickly.