**Answers**

**1.Bubble Sort**

Bubble Sort is a simple comparison-based sorting 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 no swaps are needed, indicating that the list is sorted.

* **Time Complexity**:
  + Best Case: O(n) (when the array is already sorted)
  + Average Case: O(n²)
  + Worst Case: O(n²)

**Insertion Sort**

Insertion Sort builds a sorted array one element at a time. It takes each element from the unsorted portion and inserts it into the correct position in the sorted portion. It's efficient for small data sets or nearly sorted arrays.

* **Time Complexity**:
  + Best Case: O(n) (when the array is nearly sorted)
  + Average Case: O(n²)
  + Worst Case: O(n²)

**Quick Sort**

Quick Sort is 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**:
  + Best Case: O(n log n)
  + Average Case: O(n log n)
  + Worst Case: O(n²) (when the smallest or largest element is always chosen as the pivot)

**Merge Sort**

Merge Sort is another divide-and-conquer algorithm that divides the array into halves, sorts them, and then merges them back together. It guarantees O(n log n) time complexity.

* **Time Complexity**:
  + Best Case: O(n log n)
  + Average Case: O(n log n)
  + Worst Case: O(n log n)

4. **Analysis**

**Performance Comparison**

* **Bubble Sort**:
  + Best Case: O(n)
  + Average Case: O(n²)
  + Worst Case: O(n²)
* **Quick Sort**:
  + Best Case: O(n log n)
  + Average Case: O(n log n)
  + Worst Case: O(n²)

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

Quick Sort is generally preferred over Bubble Sort for several reasons:

1. **Efficiency**: Quick Sort has better average and worst-case time complexity compared to Bubble Sort, making it faster for larger datasets.
2. **Divide-and-Conquer**: Quick Sort’s divide-and-conquer strategy effectively reduces the problem size at each recursive step, leading to a more efficient sorting process.