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

Ans)

1. **Bubble Sort**

Bubble Sort is a simple comparison-based algorithm where each pair of adjacent elements is compared, and the elements are swapped if they are in the wrong order. This process is repeated until the list is sorted.

**Working Principle:**

1. Compare each pair of adjacent elements.
2. Swap the elements if they are in the wrong order.
3. Repeat the process for all elements until no swaps are needed.

**Time Complexity:**

* Best Case: O(n) (when the array is already sorted)
* Average Case: O(n^2)
* Worst Case: O(n^2)

**Use Case:**

* Bubble Sort is rarely used in practice for large datasets due to its inefficiency. However, it's useful for educational purposes to understand basic sorting concepts.

1. **Insertion Sort**

Insertion Sort builds the final sorted array one element at a time. It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort.

**Working Principle:**

1. Start with the first element, considering it as a sorted portion.
2. Take the next element and insert it into the sorted portion at the correct position.
3. Repeat the process for all elements.

**Time Complexity:**

* Best Case: O(n) (when the array is already sorted)
* Average Case: O(n^2)
* Worst Case: O(n^2)

**Use Case:**

* Insertion Sort is efficient for small datasets or partially sorted arrays. It's also used in practice for sorting small arrays in more complex algorithms like quicksort and merge sort.

**3. Quick Sort**

Quick Sort is a highly efficient sorting algorithm and is based on the divide-and-conquer approach. It works by selecting a 'pivot' element from the array and partitioning the other elements into two sub-arrays according to whether they are less than or greater than the pivot.

**Working Principle:**

1. Choose a pivot element.
2. Partition the array into two sub-arrays: elements less than the pivot and elements greater than the pivot.
3. Recursively apply the above steps to the sub-arrays.
4. Combine the sub-arrays to get the sorted array.

**Time Complexity:**

* Best Case: O(n log n)
* Average Case: O(n log n)
* Worst Case: O(n^2) (when the pivot selection is poor)

**Use Case:**

* Quick Sort is used in various applications due to its efficiency and simplicity. It's particularly effective for large datasets.

**4. Merge Sort**

Merge Sort is a stable, comparison-based sorting algorithm. It is based on the divide-and-conquer approach and works by dividing the unsorted list into n sublists, each containing one element, and then merging them to produce a sorted list.

**Working Principle:**

1. Divide the unsorted array into two roughly equal sub-arrays.
2. Recursively sort the sub-arrays.
3. Merge the two sorted sub-arrays to produce the final sorted array.

**Time Complexity:**

* Best Case: O(n log n)
* Average Case: O(n log n)
* Worst Case: O(n log n)

**Use Case:**

* Merge Sort is used in applications where stable sorting is required. It's also used for sorting linked lists and external sorting (where data is too large to fit into memory).

**Summary of Sorting Algorithms:**

* **Bubble Sort**: Simple but inefficient for large datasets (O(n^2)).
* **Insertion Sort**: Efficient for small or nearly sorted datasets (O(n^2)).
* **Quick Sort**: Efficient for large datasets with average case O(n log n), but worst case O(n^2).
* **Merge Sort**: Efficient and stable, suitable for large datasets and external sorting (O(n log n)).

Each of these algorithms has its strengths and weaknesses, making them suitable for different types of datasets and scenarios.

2) Compare the performance (time complexity) of Bubble Sort and Quick Sort?

Ans)

**Time Complexity Comparison**

* **Bubble Sort:**
  + Best Case: O(n)
  + Average Case: O(n^2)
  + Worst Case: O(n^2)
  + **Suitability:** Bubble Sort is simple but inefficient for large datasets due to its quadratic time complexity.
* **Quick Sort:**
  + Best Case: O(n log n)
  + Average Case: O(n log n)
  + Worst Case: O(n^2)
  + **Suitability:** Quick Sort is generally more efficient and preferred for large datasets due to its average-case performance of O(n log n). The worst case can be mitigated with techniques like random pivot selection.

3) Discuss why Quick Sort is generally preferred over Bubble Sort?

Ans)

Quick Sort is generally more efficient and preferred for large datasets due to its average-case performance of O(n log n). The worst case can be mitigated with techniques like random pivot selection.