**1.Understanding sorting algorithms**

**Bubble Sort**

It is a simple comparison based sorting algorithm, which iteratively compares and swaps

elements in an array. The repetitive comparison and swapping are done until the entire array

elements are sorted.

Best Case: O(n) – when the list is already sorted.

Average Case: O(n^2)

Worst Case: O(n^2)

Insertion Sort

It takes each element on each iteration and places it in appropriate sorted position.This

is repeated until all the elements got placed in sorted positions.

Best Case: O(n) – when the list is already sorted.

Average Case: O(n^2)

Worst Case: O(n^2)

**Quick Sort:**

A highly efficient approach which uses divide and-conquer approach to sort. It chooses

A pivot element, partitions the array into sub-arrays, checks and positions the subarrays

properly by using recursion. At last we arrive at a sorted array

**Time Complexity**:

* Best Case: O(n log n)
* Average Case: O(n log n)
* Worst Case: O(n^2) – when the smallest or largest element is always chosen as the pivot.

**Space Complexity**: O(log n) – due to the recursive calls.

**Merge Sort**

It is an another efficient that uses divide and conquer to sort the array.It halves the arrays,

Recursively sorts them and at last all are merged to get into the original size.

**Time Complexity**:

Best Case: O(n log n)

Average Case: O(n log n)

Worst Case: O(n log n)

**Space Complexity**: O(n) – requires additional space for merging.

**4.Analysis:**

Bubble sort:

Best Case: O(n) – when the list is already sorted.

Average Case: O(n^2)

Worst Case: O(n^2)

But not suitable for large data

Quick sort:

Best Case: O(n log n)

Average Case: O(n log n)

Worst Case: O(n^2) – when the smallest or largest element is always chosen as the pivot.

Best for large data, comparatively better than bubble sort when average case is concerned