**Divide And Conquer**

**Merge Sort:**

Merge Sort is a divide-and-conquer sorting algorithm that efficiently sorts an array or list by recursively dividing it into smaller subarrays, sorting those subarrays, and then merging them to produce the final sorted array. It is one of the most efficient comparison-based sorting algorithms with consistent performance.

Algorithm Steps:

* Divide: The array is divided into two halves recursively until each subarray contains only one element. This is achieved by finding the middle index of the array and splitting it into left and right subarrays.
* Conquer: Each pair of adjacent subarrays is merged together in sorted order. This process continues recursively until all subarrays are merged, resulting in a sorted array.

Time Complexity:

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

Merge Sort offers consistent performance regardless of the input data and is suitable for sorting large datasets efficiently.

**Quick Sort:**

Quick Sort is another efficient divide-and-conquer sorting algorithm that sorts an array by selecting a pivot element, partitioning the array around the pivot, and recursively sorting the subarrays on either side of the pivot. It is widely used in practice due to its simplicity and high performance.

Algorithm Steps:

* Choose Pivot: Select a pivot element from the array. Commonly, the last element is chosen as the pivot.
* Partition: Rearrange the elements in the array so that all elements less than the pivot are moved to its left, and all elements greater than the pivot are moved to its right. The pivot element is now in its sorted position.
* Recursion: Recursively apply the same process to the subarrays on the left and right of the pivot.

Time Complexity:

* Best Case: O(n log n)
* Average Case: O(n log n)
* Worst Case: O(n^2)

Quick Sort has excellent average-case performance and is often preferred for sorting large datasets or arrays in practice.