Conclusion

Method Name	Best Time Complexity	Worst Time Complexity	Ave Time Complexity	Space Complexity	Is Stable
Merge Sort	O(n*log(n))	O(n*log(n))	O(n*log(n))	O(n)	Yes
Bubble Sort	O(n)	O(n**2)	O(n**2)	O(1)	Yes
Insertion Sort	O(n)	O(n**2)	O(n**2)	O(1)	Yes
Selection Sort	O(n**2)	O(n**2)	O(n**2)	O(1)	No
Shell Sort	O(n)	O(n**1.3)	O(n**2)	O(1)	No
Heap Sort	O(n*log(n))	O(n*log(n))	O(n*log(n))	O(1)	No
Quick Sort	O(n*log(n))	O(n*log(n))	O(n**2)	O(log(n))~O(n)	No

1 Merge Sort

1.1 *Definition*

A divide-and-conquer algorithm that recursively divides the array into halves, sorts each half, and then merges the sorted halves.

1.2 *Easy understanding*

let's imagine you have a bunch of numbered playing cards, and you want to arrange them in order from the smallest to the biggest. But, you can only compare two cards at a time, and it's a bit tricky to do it all at once. So, here's what you do:

1.Divide and Conquer: First, you divide the cards into smaller piles. Then, you do the same thing with each of those smaller piles. Keep doing this until each pile only has one card in it.

2.Sort Each Pile: Now, you start combining the piles back together, but in a sorted way. Imagine you have two sorted piles of cards. To combine them, you look at the top cards of each pile and pick the smaller one. You put that card in a new pile. Then, you look again at the top cards of the two piles and pick the smaller one. You keep doing this until all the cards are in the new pile, and it stays sorted.

3.Combine Everything: You do this combining step for all the pairs of piles, then for the groups of four piles, and so on until you have just one big sorted pile. It's like putting the cards back together, but now they are all in the right order.

Example: Let's say you have the cards 5, 3, 7, 2, 8, 4 .Here's how it might work:
Divide: Split the cards into pairs: (5, 3), (7, 2), (8, 4).
Sort Each Pair: Sort each pair separately: (3, 5), (2, 7), (4, 8).
Combine Pairs: Combine the pairs back together, sorting as you go: (2, 3, 5, 7), (4, 8).
Combine Everything: Now, combine those two sorted piles, making one big sorted pile: (2, 3, 4, 5, 7, 8).

1.3 *Implemented Java Code*

a simple Java *implementation* of the Merge Sort algorithm:

```
public class MergeSort {
   public static void main(String[] args) {
       int[] array = {12, 11, 13, 5, 6, 7};

       System.out.println("Original array:");
       printArray(array);

       mergeSort(array);

       System.out.println("\nSorted array:");
       printArray(array);
   }

   // Merge Sort function
   public static void mergeSort(int[] array) {
       int n = array.length;
       if (n > 1) {
           int mid = n / 2;
           int[] leftArray = new int[mid];
           int[] rightArray = new int[n - mid];

           // Copy data to temporary arrays leftArray[] and rightArray[]
           System.arraycopy(array, 0, leftArray, 0, mid);
           System.arraycopy(array, mid, rightArray, 0, n - mid);

           // Recursively sort the two halves
           mergeSort(leftArray);
           mergeSort(rightArray);

           // Merge the sorted halves
           merge(array, leftArray, rightArray);
       }
   }

   // Merge two subarrays of array[]
   public static void merge(int[] array, int[] leftArray, int[] rightArray) {
       int i = 0, j = 0, k = 0;

       // Merge elements back into the original array in sorted order
       while (i < leftArray.length && j < rightArray.length) {
           if (leftArray[i] <= rightArray[j]) {
               array[k] = leftArray[i];
               i++;
           } else {
               array[k] = rightArray[j];
               j++;
           }
           k++;
       }
```

```
        // Copy the remaining elements of leftArray[], if there are any
        while (i < leftArray.length) {
            array[k] = leftArray[i];
            i++;
            k++;
        }

        // Copy the remaining elements of rightArray[], if there are any
        while (j < rightArray.length) {
            array[k] = rightArray[j];
            j++;
            k++;
        }
    }

    // Utility function to print an array
    public static void printArray(int[] array) {
        for (int value : array) {
            System.out.print(value + " ");
        }
        System.out.println();
    }
}
```

1.4 *A gif to help u better understand*