Merge Sort

📘 **Definition:**
Merge Sort is a recursive, divide-and-conquer sorting algorithm that **divides** the array into two halves, **sorts** them recursively, and then **merges** the sorted halves to produce a fully sorted array.

💡 **Real-Life Analogy**

Socho tumhare paas 10 students hain jinko **roll number ke order mein lagana hai**.

1. Tum group ko do barabar hisson mein **baant do**.
2. Har part ko alag se **sort karo** (recursively).
3. Fir dono sorted parts ko **merge karo** ek sorted line mein.

Jaise **2 sorted files ko combine karte ho**, waise.

✅ **Merge Sort in C**

```
#include <stdio.h>

// Merge two sorted subarrays: arr[l..m] and arr[m+1..r]

void merge(int arr[], int l, int m, int r) {

    int n1 = m - l + 1;

    int n2 = r - m;

    int L[n1], R[n2]; // Temporary arrays


    // Copy data into temp arrays

    for(int i = 0; i < n1; i++)

        L[i] = arr[l + i];

    for(int j = 0; j < n2; j++)

        R[j] = arr[m + 1 + j];

    // Merge temp arrays back into arr[l..r]

    int i = 0, j = 0, k = l;

    while(i < n1 && j < n2) {

        if(L[i] <= R[j])

            arr[k++] = L[i++];

        else

            arr[k++] = R[j++];

    }
```

```
    // Copy remaining elements
    while(i < n1) arr[k++] = L[i++];
    while(j < n2) arr[k++] = R[j++];
}

// Recursive merge sort function
void mergeSort(int arr[], int l, int r) {
    if(l < r) {
        int m = (l + r) / 2;

        // Sort left and right halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        // Merge sorted halves
        merge(arr, l, m, r);
    }
}

int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int size = sizeof(arr) / sizeof(arr[0]);

    mergeSort(arr, 0, size - 1);

    printf("Sorted array: ");
    for(int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

🧠 Hinglish Breakdown (Line by Line)

- **Divide**: Array ko beech se todte jaao until har subarray mein 1 element bacha ho
- **Conquer**: Subarrays ko recursively sort karo
- **Merge**: 2 sorted parts ko merge karke ek sorted array banao

Example for arr[] = {5, 2, 4, 7, 1, 3, 2, 6}

Split:

[5, 2, 4, 7] and [1, 3, 2, 6]

→ [5, 2] [4, 7] and [1, 3] [2, 6]

→ [5] [2] [4] [7] [1] [3] [2] [6]

→ Merge back in sorted order

📜 Pseudocode

```
mergeSort(arr, left, right):
    if left < right:
        mid = (left + right) / 2
        mergeSort(arr, left, mid)
        mergeSort(arr, mid + 1, right)
        merge(arr, left, mid, right)


merge(arr, l, m, r):
    Create temp arrays L[] and R[]
    Copy data
    Merge while comparing
    Copy leftovers
```

📈 Time & Space Complexity

Case	Time	Space
Best	O(n log n) ✅	O(n) ❗
Average	O(n log n) ✅	O(n)
Worst	O(n log n) ✅	O(n)
Stable	✅	Yes

✅ **When to Use Merge Sort:**

- Data is huge and performance must be **guaranteed**
- You need a **stable** sort
- Sorting linked lists or **external files**