🟦 Mergesort

1️⃣ Grundidee

Mergesort ist ein **Divide-and-Conquer-Algorithmus**. Die Liste wird rekursiv in **immer kleinere Teillisten** zerlegt, bis diese trivial sortiert sind, und anschliessend **geordnet wieder zusammengeführt (merge)**.

- Aufteilen → Sortieren → Zusammenführen
- Rekursive Struktur
- Sehr gut für grosse Datenmengen

2️⃣ Voraussetzungen

- ❌ keine
- funktioniert auf **beliebigen vergleichbaren Elementen**

3️⃣ Laufzeiten & Eigenschaften

Eigenschaft	Wert
Best Case	O(n log n)
Average Case	O(n log n)
Worst Case	O(n log n)
Speicherbedarf	O(n)
In-place	nein
Stabil	ja

Hinweis: Die Laufzeit ist **immer O(n log n)**, unabhängig von der Eingabereihenfolge.

4️⃣ Schritt-für-Schritt-Beispiel

Ausgangsliste:

```
[5, 3, 4, 1]
```

Aufteilen

```
[5, 3]   [4, 1]
[5] [3]  [4] [1]
```

Zusammenführen

```
[3, 5]   [1, 4]
[1, 3, 4, 5]
```

Ergebnis:

```
[1, 3, 4, 5]
```

5️⃣ Besonderheiten / Prüfungsrelevante Hinweise

- Garantierte Laufzeit O(n log n)
- Sehr gut für **grosse Datenmengen**
- Wird oft als **Referenzalgorithmus** verwendet
- Nachteil: zusätzlicher Speicherbedarf

6️⃣ Vor- und Nachteile

Vorteile

- stabil
- vorhersehbare Laufzeit
- gut parallelisierbar

Nachteile

- nicht in-place
- zusätzlicher Speicherbedarf

🧠 Merksatz für die Prüfung

Mergesort teilt die Liste rekursiv und fügt sie sortiert zusammen – stabil mit O(n log n).

7️⃣ Python-Implementierung

In [1]:

```python
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left = merge_sort(arr[:mid])
    right = merge_sort(arr[mid:])

    return merge(left, right)

def merge(left, right):
    result = []
    i = j = 0

    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    result.extend(left[i:])
    result.extend(right[j:])
    return result
```