Merge Sort

Recursively divides the list into halves until each sublist has one element, then merges the sorted sublists back together to form a fully sorted list.

```python
def merge(left: list, right: list) -> list:
    i = j = 0
    results = []
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            results.append(left[i])
            i += 1
        else:
            results.append(right[j])
            j += 1
    results.extend(left[i:])
    results.extend(right[j:])
    return results


def merge_sort(arr: list) -> list:
    if len(arr) < 2:
        return arr

    mid = len(arr) // 2

    return merge(merge_sort(arr[:mid]), merge_sort(arr[mid:]))
```

Property	Details
How it works	Recursively splits the list into two halves until each sublist has one element (or is empty). A sublist of length 1 or 0 is already considered sorted. Then, the sorted sublists are merged from left to right to form a fully sorted list. The sorting happens during the merge step, where elements from both sublists are combined in order.
Time Complexity	O(n * log n) in all cases (Best, Average, Worst)
Space Complexity	O(n)
Stable	Yes
Type	creates copies, recursive

- Follow on Bluesky
- Join Discord