### Algorithm: Merge Intervals

Given a collection of intervals, merge all overlapping intervals.

#### Initialize Data Structures:

* Use a list to store the merged intervals.

#### Sort and Merge Intervals:

* Sort the intervals by the start time.
* Iterate through the intervals and merge them if they overlap.

#### Retrieve the Result:

* The list contains the merged intervals.

#### Implementation:

| **def** merge\_intervals(intervals: List[List[int]]) -> List[List[int]]:  **if** **not** intervals:  **return** []  intervals.sort(key=**lambda** x: x[0])  merged = [intervals[0]]  **for** interval **in** intervals[1:]:  **if** interval[0] <= merged[-1][1]:  merged[-1][1] = max(merged[-1][1], interval[1])  **else**:  merged.append(interval)  **return** merged  *# Example usage:*  intervals = [[1, 3], [2, 6], [8, 10], [15, 18]]  print(merge\_intervals(intervals)) *# Output: [[1, 6], [8, 10], [15, 18]]* |
| --- |

#### Explanation:

Initialize:

* merged: A list initialized with the first interval.

Sort and Merge Intervals:

* Sort the intervals by the start time.
* Iterate through the intervals and merge them if they overlap.

Retrieve the Result:

### The list contains the merged intervals.