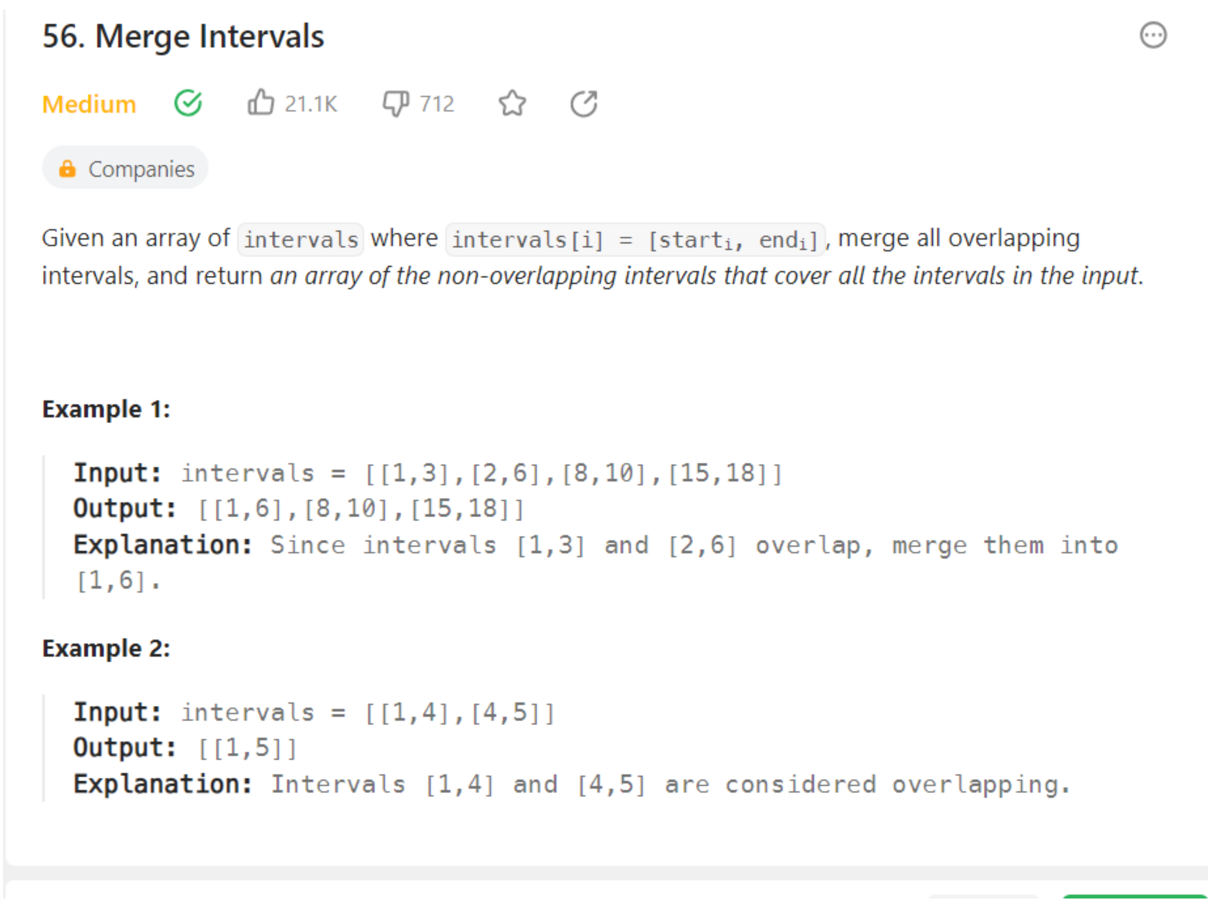

56. Merge Intervals

Given an array of intervals where intervals[i] = [$start_i$, end_i], merge all overlapping intervals, and return *an array of the non-overlapping intervals that cover all the intervals in the input.*

Example 1:

```
Input: intervals = [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]
Explanation: Since intervals [1,3] and [2,6] overlap, merge them into
[1,6].
```

Example 2:

```
Input: intervals = [[1,4],[4,5]]
Output: [[1,5]]
Explanation: Intervals [1,4] and [4,5] are considered overlapping.
```

Leran:
https://medium.com/@timpark0807/leetcode-is-easy-the-interval-pattern-d68a7c1c841

```
def merge(intervals):
    intervals.sort()
    result = [intervals[0]]
    # Iterate over the input intervals
    for interval in intervals[1:]:
        interval_2 = result[-1]
        # If they overlap, merge them.
        if do_overlap(interval, interval_2):
            merged_front = min(interval[0], interval_2[0])
            merged_back = max(interval[1], interval_2[1])
            result[-1] = [merged_front, merged_back]
        # If they don't overlap, check the next interval.
        else:
            result.append(interval)
    return result
def do_overlap(interval_1, interval_2):
    front = max(interval_1[0], interval_2[0])
```

```
   back = min(interval_1[1], interval_2[1])
   return back - front >= 0
```

Code

```
class Solution {
    func merge(_ intt: [[Int]]) -> [[Int]] {

    let intervals = intt.sorted(by: {$0[0] < $1[0]})
    var res:[[Int]] = [intervals[0]]
    for i in 1..<intervals.count {
        let temp = res.last!
        if checkIfOverlap(temp,intervals[i]) {
            let b = intervals[i]
            let front = min(temp[0],b[0])
            let back = max(temp[1],b[1])
            res[res.count-1] = [front,back]
        } else {
            res.append(intervals[i])
        }
    }
    return res
    }

    func checkIfOverlap(_ a:[Int], _ b:[Int]) -> Bool {
        let front = max(a[0],b[0])
        let back = min(a[1],b[1])
        return back-front >= 0 ? true : false
    }
}
```