题目描述

给出一个区间的集合，请合并所有重叠的区间。

示例 1:

输入: [[1,3],[2,6],[8,10],[15,18]]
输出: [[1,6],[8,10],[15,18]]
解释: 区间 [1,3] 和 [2,6] 重叠, 将它们合并为 [1,6].
示例 2:

输入: [[1,4],[4,5]]
输出: [[1,5]]
解释: 区间 [1,4] 和 [4,5] 可被视为重叠区间。

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/merge-intervals

解题思想

将区间根据左端大小进行排序，左端相同直接合并，左端不同的，看当前区间的右端与下一个区间的左端的比较结果

代码实现

```
class Solution {
public:
    vector<vector<int>> merge(vector<vector<int>>& intervals) {
        int m = intervals.size();
        if (m == 0)
            return {};
        sort(intervals.begin(), intervals.end());       //先为所有区间排序（以左端点排序
        vector<vector<int>> result;                     //当出现新区间时就加入
        for (int i = 0; i < m; i++)
        {
            int L = intervals[i][0];
            int R = intervals[i][1];
            if (result.empty() || result.back()[1] < L)     //新区间出现
                result.push_back({ L,R });
            else
            {
                result.back()[1] = max(result.back()[1], R);    // 不是新区间，比较后更新旧区间
            }
        }
        return result;
    }
};
```

Python实现：

```python
# coding=utf-8

class Solution:
    def merge(self, intervals:List[List[int]]) -> List[List[int]]:
        intervals.sort(key=lambda x:x[0])           # 根据x[0]来排序
        merged = []
        for interval in intervals:
            if not merged or merged[-1][1]<interval[0]:     # -1表示列表的倒数第一个，也就是上一个，not merged判断 list是否为空,若为空 if merged 若有数，则返回0
                merged.append(interval)
            else:
                merged[-1][1]=max(merged[-1][1], interval[1])

        return  merged
```