1. 题目
2. 题干

给出一个区间的集合，请合并所有重叠的区间。

1. 示例

示例 1:

输入: intervals = [[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: intervals = [[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。

1. 题解
2. 思路

给出多个区间，要将区间进行合并，合并的条件是什么呢？就是a区间的右端点大于或者等于b区间的左端点，那么a区间就可以和b区间合并，将a区间的右端点替换为一个新的数，该数为a区间，b区间的右端点中更大的那个。

合并的问题解决了，那么我们怎样进行有效地合并呢？因为我们比较的是前面一个区间的右端点和后面一个区间的左端点，那么排列的有序性一定能够帮助我们。首先，将区间都按照左端点排序，用数组merged存储最终的区间输出，对每个区间循环，判断merged数组中的最后一个区间的右端点和当前区间的左端点的大小，以此来决定是否合并。

1. 代码实现

JAVA：

class Solution {

    public int[][] merge(int[][] intervals) {

        if(intervals.length == 0){

            return new int[0][2];

        }

        Arrays.sort(intervals, new Comparator<int[]>(){

            public int compare(int[] interval1, int[] interval2){

                return interval1[0] - interval2[0];

            }

        });

        List<int[]> merged = new ArrayList<int[]>();

        for(int i = 0; i < intervals.length; ++i){

            int l = intervals[i][0];

            int r = intervals[i][1];

            if(merged.size()==0 || merged.get(merged.size()-1)[1] < l){

                merged.add(new int[]{l,r});

            }else{

                merged.get(merged.size()-1)[1] = Math.max(merged.get(merged.size()-1)[1], r);

            }

        }

        return merged.toArray(new int[merged.size()][]);

    }

}

C++：

class Solution {

public:

vector<vector<int>> merge(vector<vector<int>>& intervals) {

if (intervals.size() == 0) {

return {};

}

sort(intervals.begin(), intervals.end());

vector<vector<int>> merged;

for (int i = 0; i < intervals.size(); ++i) {

int L = intervals[i][0], R = intervals[i][1];

if (!merged.size() || merged.back()[1] < L) {

merged.push\_back({L, R});

}

else {

merged.back()[1] = max(merged.back()[1], R);

}

}

return merged;

}

};