56-合并区间

题述

56. 合并区间

难度 中等 1386

以数组 `intervals` 表示若干个区间的集合，其中单个区间为 `intervals[i] = [start_i, end_i]` 。请你合并所有重叠的区间，并返回 *一个不重叠的区间数组，该数组需恰好覆盖输入中的所有区间* 。

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。
```

提示：

- $1 <= intervals.length <= 10^4$
- `intervals[i].length == 2`
- $0 <= start_i <= end_i <= 10^4$

通过次数 400,747 | 提交次数 830,643

思路

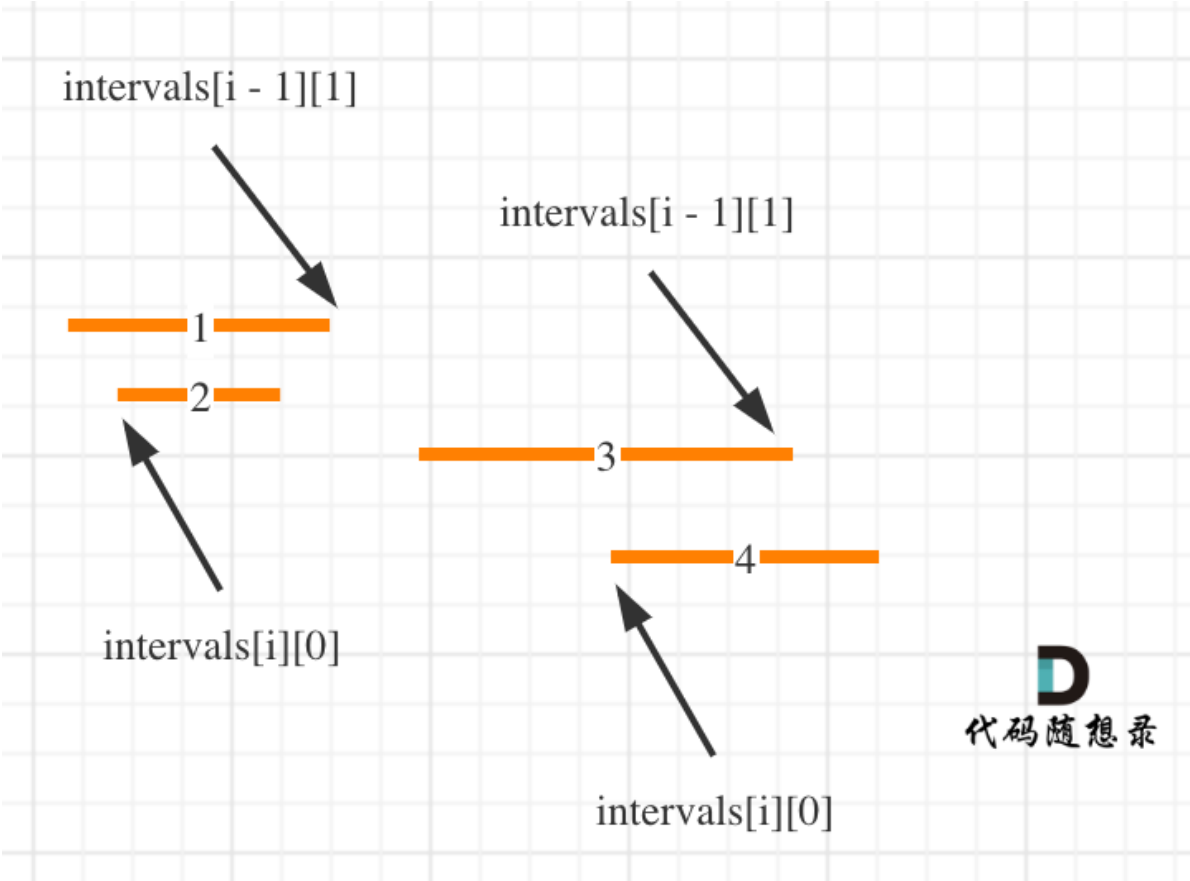
先以左/右边界为参考进行排序

按照左边界排序，排序之后局部最优：每次合并都取最大的右边界，这样就可以合并更多的区间了，整体最优：合并所有重叠的区间

局部最优可以推出全局最优

按照左边界从小到大排序之后，如果 `intervals[i][0] < intervals[i - 1][1]` 即intervals[i]左边界 < intervals[i - 1]右边界，则一定有重复，因为intervals[i]的左边界一定是大于等于intervals[i - 1]的左边界。

intervals[i]的左边界在intervals[i - 1]左边界和右边界的范围内，那么一定有重复！



其实就是用合并区间后左边界和右边界，作为一个新的区间，加入到result数组里就可以了。如果没有合并就把原区间加入到result数组。

题解

C++

```
class Solution {
public:
    vector<vector<int>> merge(vector<vector<int>>& intervals)
    {
        //合并区间
        //先选定左/右边界排序
        //以左边界进行排序，合并时取大的右边界
        //intervals[i][0] < intervals[i - 1][1]
        //intervals[i]左边界 < intervals[i - 1]右边界，则一定有重复
        vector<vector<int>> result;
        if(intervals.size() == 0)
        {
            return result;
        }

        //使用lambda表达式
        sort(intervals.begin(),intervals.end(),[](const vector<int>& a,const
vector<int>& b){return a[0] < b[0];});

        result.push_back(intervals[0]);
        for(int i = 1; i < intervals.size(); i++)
        {
            if(result.back()[1] >= intervals[i][0])
```

```
            {
                //合并
                result.back()[1] = max(result.back()[1],intervals[i][1]);
            }
            else
            {
                result.push_back(intervals[i]);
            }
        }
        return result;
    }
};
```

提交结果	执行用时	内存消耗	语言	提交时间	备注
通过	28 ms	18.4 MB	C++	2022/03/23 10:16	添加备注
通过	12 ms	14.2 MB	C++	2021/09/09 06:17	添加备注
执行出错	N/A	N/A	C++	2021/09/09 06:17	添加备注
编译出错	N/A	N/A	C++	2021/09/09 06:13	添加备注

Python

```python
class Solution:
    def merge(self, intervals: List[List[int]]) -> List[List[int]]:
        if len(intervals) == 0: return intervals
        intervals.sort(key = lambda x : x[0])
        result = []
        result.append(intervals[0])
        for i in range(1,len(intervals)):
            last = result[-1]
            if last[1] >= intervals[i][0]:
                result[-1] = [last[0],max(last[1],intervals[i][1])]
            else:
                result.append(intervals[i])
        return result
```

提交结果	执行用时	内存消耗	语言	提交时间	备
通过	44 ms	17.9 MB	Python3	2022/03/23 10:27	
通过	28 ms	18.4 MB	C++	2022/03/23 10:16	
通过	12 ms	14.2 MB	C++	2021/09/09 06:17	
执行出错	N/A	N/A	C++	2021/09/09 06:17	
编译出错	N/A	N/A	C++	2021/09/09 06:13	

思考

贪心算法是一种比较模糊的思想，就是局部最优推出全局最优