秒懂力扣区间题

重叠区间、合并区间、插入区间

Original Sweetiee姨 甜姨的奇妙冒险 2020-11-04 00:23
今天的力扣打卡题是 57. 插入区间 ，我们再顺便练习两道类似的简单区间题目，比如：判断区间是否重叠（252. 会议室 ）、56. 合并区间。这类面试题目还挺讨巧的，因为不需要掌握什么数据结构与算法的先验知识，看懂题目之后模拟一遍即可，很容易考察出应聘者到底会不会写代码。

一、判断区间是否重叠

题目描述

力扣 252. 会议室
难度：Easy

给定一个会议时间安排的数组 intervals ，每个会议时间都会包括开始和结束的时间 intervals[i] = [starti, endi]，请你判断一个人是否能够参加这里面的全部会议。

示例 1::
输入: intervals = [[0,30],[5,10],[15,20]]
输出: false
解释: 存在重叠区间，一个人在同一时刻只能参加一个会议。

示例 2::
输入: intervals = [[7,10],[2,4]]
输出: true
解释: 不存在重叠区间。

思路分析

因为一个人在同一时刻只能参加一个会议，因此题目实质是判断是否存在重叠区间，这个简单，将区间按照会议开始时间进行排序，然后遍历一遍判断即可。

代码实现

```
class Solution {
    public boolean canAttendMeetings(int[][] intervals) {
        // 将区间按照会议开始实现升序排序
        Arrays.sort(intervals, (v1, v2) -> v1[0] - v2[0]);
        // 遍历会议，如果下一个会议在前一个会议结束之前就开始了，返回 false。
        for (int i = 1; i < intervals.length; i++) {
```

```
            if (intervals[i][0] < intervals[i - 1][1]) {
                return false;
            }
        }
        return true;
    }
}
```

二、合并区间

题目描述
力扣 56. 合并区间
难度：Medium

给出一个区间的集合，请合并所有重叠的区间。

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。
思路分析
和上一题一样，首先对区间按照起始端点进行升序排序，然后逐个判断当前区间是否与前一个区间重叠，如果不重叠的话将当前区间直接加入结果集，反之如果重叠的话，就将当前区间与前一个区间进行合并。
代码实现

```
class Solution {
    public int[][] merge(int[][] intervals) {
        // 先按照区间起始位置排序
        Arrays.sort(intervals, (v1, v2) -> v1[0] - v2[0]);
        // 遍历区间
        int[][] res = new int[intervals.length][2];
        int idx = -1;
        for (int[] interval: intervals) {
            // 如果结果数组是空的，或者当前区间的起始位置 > 结果数组中最后区间的终止位置，说明不重叠。
```

```
            // 则不合并，直接将当前区间加入结果数组。
            if (idx == -1 || interval[0] > res[idx][1]) {
                res[++idx] = interval;
            } else {
                // 反之说明重叠，则将当前区间合并至结果数组的最后区间
                res[idx][1] = Math.max(res[idx][1], interval[1]);
            }
        }
        return Arrays.copyOf(res, idx + 1);
    }
}
```

三、插入区间

题目描述
57. 插入区间
难度：Medium

给出一个无重叠的 ，按照区间起始端点排序的区间列表。

在列表中插入一个新的区间，你需要确保列表中的区间仍然 有序且不重叠（如果有必要的话，可以 合并区间）。

示例 1:：
输入: intervals = [[1,3],[6,9]], newInterval = [2,5]
输出: [[1,5],[6,9]]
解释: 新区间[2,5] 与 [1,3]重叠，因此合并成为 [1,5]。

示例 2:：
输入: intervals = [[1,2],[3,5],[6,7],[8,10],[12,16]], newInterval = [4,8]
输出: [[1,2],[3,10],[12,16]]
解释: 新区间 [4,8] 与 [3,5],[6,7],[8,10] 重叠，因此合并成为 [3,10]。
思路分析

本题中的区间已经按照起始端点升序排列，因此我们直接遍历区间列表，寻找新区间的插入位置即可。具体步骤如下：
首先将新区间左边且相离的区间加入结果集（遍历时，如果当前区间的结束位置小于新区间的开始位置，说明当前区间在新区间的左边且相离）；
接着判断当前区间是否与新区间重叠，重叠的话就进行合并，直到遍历到当前区间在新区间的右边且相离，将最终合并后的新区间加入结果集；
最后将新区间右边且相离的区间加入结果集。
代码实现

```java
class Solution {
    public int[][] insert(int[][] intervals, int[] newInterval) {
        int[][] res = new int[intervals.length + 1][2];
        int idx = 0;
        // 遍历区间列表：
        // 首先将新区间左边且相离的区间加入结果集
        int i = 0;
        while (i < intervals.length && intervals[i][1] < newInterval[0]) {
            res[idx++] = intervals[i++];
        }
        // 接着判断当前区间是否与新区间重叠，重叠的话就进行合并，直到遍历到当前区间在新区间的右边且相离，
        // 将最终合并后的新区间加入结果集
        while (i < intervals.length && intervals[i][0] <= newInterval[1]) {
            newInterval[0] = Math.min(intervals[i][0], newInterval[0]);
            newInterval[1] = Math.max(intervals[i][1], newInterval[1]);
            i++;
        }
        res[idx++] = newInterval;
        // 最后将新区间右边且相离的区间加入结果集
        while (i < intervals.length) {
            res[idx++] = intervals[i++];
        }

        return Arrays.copyOf(res, idx);
    }
}
```

四、题目拓展

力扣**1288.** 删除被覆盖区间
难度：Easy
给你一个区间列表，请你删除列表中被其他区间所覆盖的区间。在完成所有删除操作后，请你返回列表中剩余区间的数目。（对于区间 [a,b) 和区间 [c,d)，若 c <= a 且 d >= b ，则区间 [a,b) 被区间 [c,d) 覆盖）
示例：
输入：intervals = [[1,4],[3,6],[2,8]]
输出：2
解释：区间 [3,6] 被区间 [2,8] 覆盖，所以它被删除了。
思路分析：本题和本文第一题的做法一样~首先按照区间起始端点进行排序，然后遍历区间，将第一题的判断区间是否重叠改为判断区间是否覆盖即可 。

力扣**228.** 汇总区间
难度：Medium
给定一个无重复元素的有序整数数组 nums，返回 恰好覆盖数组中所有数字 的 最小有序 区间范围列表。也就是说 nums 的每个元素都恰好被某个区间范围所覆盖，并且不存在属于某个范围但不属于 nums 的数字。
示例:
输入：nums = [0,1,2,4,5,7]
输出：["0->2","4->5","7"]
思路分析:本题是在有序数组中，找出连续递增的区间，是双指针/滑动窗口的经典题目，和本文中的其他区间题目不是一种类型，我们在之后的文章中再介绍双指针和滑动窗口的模板哈 ~

力扣**354.** 俄罗斯套娃信封问题
难度：Hard
给定一些标记了宽度和高度的信封，宽度和高度以整数对形式 (w, h) 出现。当另一个信封的宽度和高度都比这个信封大的时候，这个信封就可以放进另一个信封里，如同俄罗斯套娃一样。
请计算最多能有多少个信封能组成一组“俄罗斯套娃”信封（即可以把一个信封放到另一个信封里面）。
示例:
输入：envelopes = [[5,4],[6,4],[6,7],[2,3]]
解释：最多信封的个数为 3, 组合为: [2,3] => [5,4] => [6,7]。
思路分析:本题实质是求最长嵌套区间的长度，是经典题目「最长上升子序列」的变种解法也一模一样，有动态规划 和 二分 两种经典解法。由于本题和本文中的其他区间题目不是一种类型，所以我们在后续的章节里再「秒懂最长上升子序列问题及其系列变种」吧！