

扫描线技巧：安排会议室

 Stars 108k  B站 @labuladong 配套PDF和插件 下载 打卡挑战 报名 精品课程 查看





微信搜一搜

Q labuladong公众号

通知： 数据结构精品课 V1.7 持续更新中；B 站可查看 核心算法框架系列视频。

读完本文，你不仅学会了算法套路，还可以顺便解决如下题目：

牛客	LeetCode	力扣	难度
-	253. Meeting Rooms II 	253. 会议室 II 	

之前面试，被问到一道非常经典且非常实用的算法题目：会议室安排问题。

力扣上类似的问题是会员题目，你可能没办法做，但对于这种经典的算法题，掌握思路还是必要的。

先说下题目，力扣第 253 题「会议室 II」：

给你输入若干形如 `[begin, end]` 的区间，代表若干会议的开始时间和结束时间，请你计算至少需要申请多少间会议室。

函数签名如下：

```
// 返回需要申请的会议室数量
int minMeetingRooms(int[][] meetings);
```

个会议时间是冲突的，至少申请两个会议室才能让所有会议顺利进行。

如果会议之间的时间有重叠，那就得额外申请会议室来开会，想求至少需要多少间会议室，就是让你计算同一时刻最多有多少会议在同时进行。

换句话说，**如果把每个会议的起始时间看做一个线段区间，那么题目就是让你求最多有几个重叠区间**，仅此而已。

对于这种时间安排的问题，本质上讲就是区间调度问题，十有八九得排序，然后找规律来解决。

题目延伸

我们之前写过很多区间调度相关的文章，这里就顺便帮大家梳理一下这类问题的思路：

第一个场景，假设现在只有一个会议室，还有若干会议，你如何将尽可能多的会议安排到这个会议室里？

这个问题需要将这些会议（区间）按结束时间（右端点）排序，然后进行处理，详见前文 [贪心算法做时间管理](#)。

第二个场景，给你若干较短的视频片段，和一个较长的视频片段，请你从较短的片段中尽可能少地挑出一些片段，拼接出较长的这个片段。

这个问题需要将这些视频片段（区间）按开始时间（左端点）排序，然后进行处理，详见后文 [剪视频剪出一个贪心算法](#)。

第三个场景，给你若干区间，其中可能有些区间比较短，被其他区间完全覆盖住了，请你删除这些被覆盖的区间。

这个问题需要将这些区间按左端点排序，然后就能找到并删除那些被完全覆盖的区间了，详见后文 [删除覆盖区间](#)。

第四个场景，给你若干区间，请你将所有有重叠部分的区间进行合并。

这个问题需要将这些区间按左端点排序，方便找出存在重叠的区间，详见后文 [合并重叠区间](#)。

第五个场景，有两个部门同时预约了同一个会议室的若干时间段，请你计算会议室的冲突时段。

这个问题就是给你两组区间列表，请你找出这两组区间的交集，这需要你将这些区间按左端点排序，详见后文 [区间交集问题](#)。

间最少？

这个问题需要动动脑筋，说白了这就是个 0-1 背包问题的变形：

会议室可以看做一个背包，每个会议可以看做一个物品，物品的价值就是会议的时长，请问你如何选择物品（会议）才能最大化背包中的价值（会议室的使用时长）？

当然，这里背包的约束不是一个最大重量，而是各个物品（会议）不能互相冲突。把各个会议按照结束时间进行排序，然后参考前文 [0-1 背包问题详解](#) 的思路即可解决，等我以后有机会可以写一写这个问题。

第七个场景，就是本文想讲的场景，给你若干会议，让你合理申请会议室。

好了，举例了这么多，来看看今天的这个问题如何解决。

题目分析

重复一下题目的本质：

给你输入若干时间区间，让你计算同一时刻「最多」有几个区间重叠。

题目的关键点在于，给你任意一个时刻，你是否能够说出这个时刻有几个会议？

如果可以做到，那我遍历所有的时刻，找个最大值，就是需要申请的会议室数量。

有没有一种数据结构或者算法，给我输入若干区间，我能知道每个位置有多少个区间重叠？

老读者肯定可以联想到之前说过的一个算法技巧：[差分数组技巧](#)。

把时间线想象成一个初始值为 0 的数组，每个时间区间 `[i, j]` 就相当于一个子数组，这个时间区间有一个会议，那我就把这个子数组中的元素都加一。

最后，每个时刻有几个会议我不就知道了吗？我遍历整个数组，不就知道至少需要几间会议室了吗？

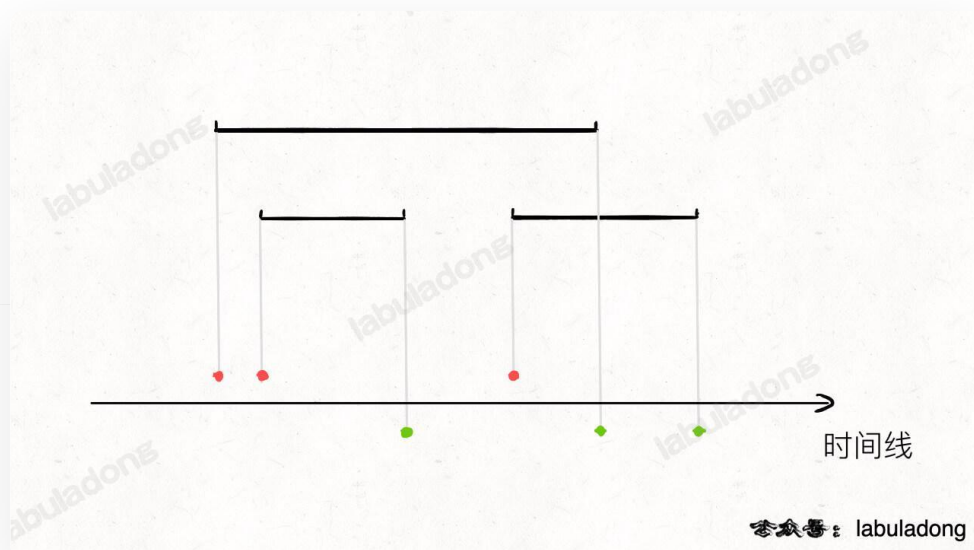
举例来说，如果输入 `meetings = [[0,30],[5,10],[15,20]]`，那么我们就给数组中 `[0,30]`，`[5,10]`，`[15,20]` 这几个索引区间分别加一，最后遍历数组，求个最大值就行了。

还记得吗，差分数组技巧可以在 $O(1)$ 时间对整个区间的元素进行加减，所以可以拿来解决这道题。

原理，有兴趣的读者可以自己尝试去实现。

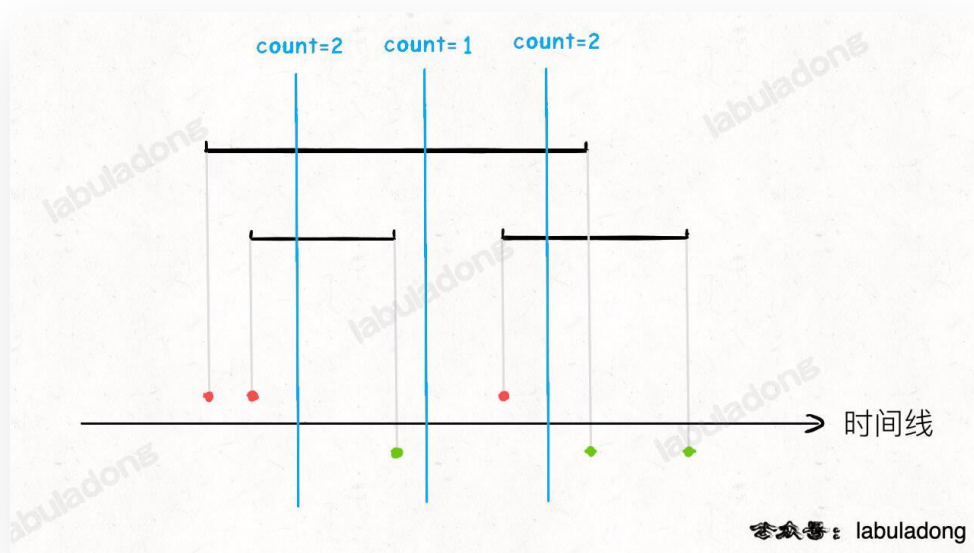
基于差分数组的思路，我们可以推导出一种更高效，更优雅的解法。

我们首先把这些会议的时间区间进行投影：



红色的点代表每个会议的开始时间点，绿色的点代表每个会议的结束时间点。

现在假想有一条带着计数器的线，在时间线上从左至右进行扫描，每遇到红色的点，计数器 `count` 加一，每遇到绿色的点，计数器 `count` 减一：





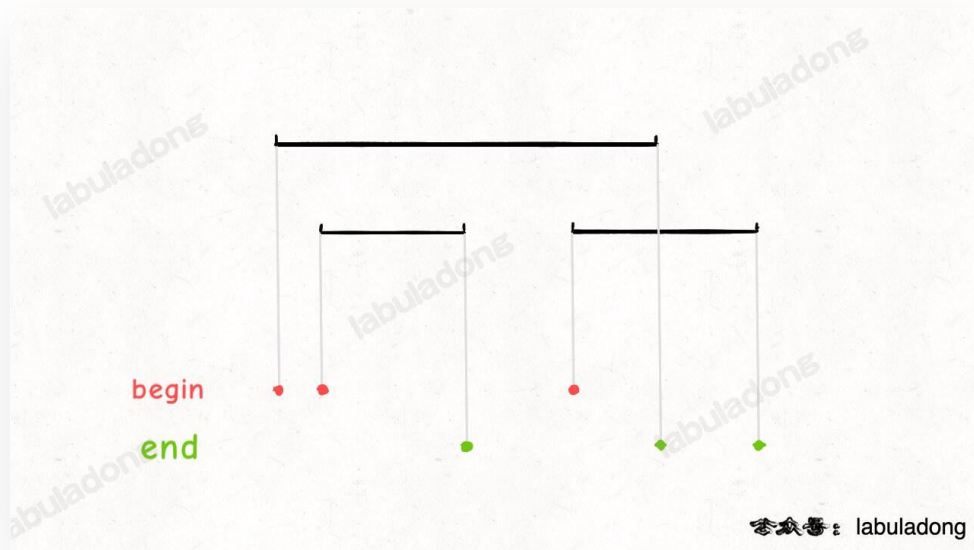
需要申请的会议室数量。

对差分数组技巧熟悉的读者一眼就能看出来了，这个扫描线其实就是差分数组的遍历过程，所以我们说这是差分数组技巧衍生出来的解法。

代码实现

那么，如何写代码实现这个扫描的过程呢？

首先，对区间进行投影，就相当于对每个区间的起点和终点分别进行排序：



```
int minMeetingRooms(int[][] meetings) {  
    int n = meetings.length;  
    int[] begin = new int[n];  
    int[] end = new int[n];  
    // 把左端点和右端点单独拿出来  
    for(int i = 0; i < n; i++) {  
        begin[i] = meetings[i][0];  
        end[i] = meetings[i][1];  
    }  
    // 排序后就是图中的红点  
    Arrays.sort(begin);  
    // 排序后就是图中的绿点  
    Arrays.sort(end);
```



然后就简单了，扫描线从左向右前进，遇到红点就对计数器加一，遇到绿点就对计数器减一，计数器 `count` 的最大值就是答案：

```
int minMeetingRooms(int[][] meetings) {
    int n = meetings.length;
    int[] begin = new int[n];
    int[] end = new int[n];
    for(int i = 0; i < n; i++) {
        begin[i] = meetings[i][0];
        end[i] = meetings[i][1];
    }
    Arrays.sort(begin);
    Arrays.sort(end);

    // 扫描过程中的计数器
    int count = 0;
    // 双指针技巧
    int res = 0, i = 0, j = 0;
    while (i < n && j < n) {💡
        if (begin[i] < end[j]) {
            // 扫描到一个红点
            count++;
            i++;
        } else {
            // 扫描到一个绿点
            count--;
            j++;
        }
        // 记录扫描过程中的最大值
        res = Math.max(res, count);
    }

    return res;
}
```

这里使用的是 **双指针技巧**，根据 `i, j` 的相对位置模拟扫描线前进的过程。

至此，这道题就做完了。当然，这个题目也可以变形，比如给你若干会议，问你 `k` 个会议室够不够用，其实你套用本文的解法代码，也可以很轻松解决。