区间连接器

知识点 数组排序 滑窗
时间限制：1s 空间限制：256MB 限定语言：不限

题目描述：

有一组区间 [a0, b0], [a1, b1], ... (a, b 表示起点, 终点)，区间有可能重叠、相邻，重叠或相邻则可以合并为更大的区间；给定一组连接器[x1, x2, x3, ...]（x 表示连接器的最大可连接长度，即 x>=gap），可用于将分离的区间连接起来，但两个分离区间之间只能使用1个连接器；请编程实现使用连接器后，最少的区间数结果。

区间数量 <10000；a,b 均 <=10000

连接器梳理 <10000; x <=10000

输入描述：

区间组：[1,10],[15,20],[18,30],[33,40]
连接器组：[5,4,3,2]

输出描述：

1
说明：合并后：[1,10], [15,30], [33,40]，使用 5, 3 两个连接器连接后只剩下 [1,40]

示例1

输入：

[1,10],[15,20],[18,30],[33,40]
[5,4,3,2]

输出：

1

说明：

合并后：[1,10], [15,30], [33,40]，使用 5, 3 两个连接器连接后只剩下 [1,40]

示例2

输入：

[1,2],[3,5],[7,10],[15,20],[30,100]
[5,4,3,2,1]

输出：

2

说明：

无重叠和相邻，使用 1，2， 5 三个连接器连接后只剩下 [1, 20], [30, 100]

解题思路：

1、对区间进行升序排序；
2、将相邻和存在交集的区间进行合并。
3、求得步骤2中各区间的距离集合
4、将步骤3的集合与连接器集合进行比较
注：使用过的连接器不能继续使用；连接器取最接近的。

```
public class Main{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        String[] regionsStr = sc.nextLine().replaceAll("\\[","")
                                           .replaceAll("]","")
                                           .split(",");
        String[] linksStr = sc.nextLine().replaceAll("\\[","")
                                         .replaceAll("]","")
                                         .split(",");;
```

```
List<int[]> regions = new ArrayList<>();        //区间集合
for( int i=0; i<regionsStr.length; i+=2){
        int left = Integer.valueOf(regionsStr[i]);
        int right = Integer.valueOf(regionsStr[i+1]);
        regions.add(new int[]{ left, right});
}

List<Integer> links = new ArrayList<>();        //连接器集合
for(String s : linksStr){
        links.add(Integer.valueOf(s));
}

regions.sort((a, b) -> {        //区间进行升序排序
        if(b[0] == a[0]){
                return a[1] - b[1];
        }
        return a[0] - b[0];
});

int[] region = null;
Iterator<int[]> iter = regions.iterator();
while(iter.hasNext()) {
        int[] next = iter.next();
        if(region == null) {
                region = next;
        } else if(region[1] >= next[0]) {
                if(region[1] < next[1]) {
                        region[1] = next[1];
                }
                iter.remove();
        } else {
                region = next;
        }
}

List<Integer> gaps = new ArrayList<>();         //各区间所需连接器的长度集合
iter = regions.iterator();
region = null;
while(iter.hasNext()) {
        int[] next = iter.next();
        if(region != null) {
                int gap = next[0] - region[1];
                gaps.add(gap);
        }
```

```
            region = next;
        }

        Collections.sort(gaps);
        Collections.sort(links);

        int i = 0; // gaps index
        int j = 0; // links index
        while(i < gaps.size() && j < links.size()) {
            if(links.get(j) >= gaps.get(i)) {     //连接器长度大于等于所需连接器长度，符合要
求
                gaps.set(i, 0);         //可以连接的两个区间距离设置为 0
                i++;
                j++;       //使用过的连接器不再使用
            } else {
                j++;
            }
        }

        int noneZoreNum = 0;
        for (int g :    gaps) {
            if(g > 0) {         //大于 0，说明两个区间无法进行连接
                noneZoreNum++;
            }
        };
        System.out.println(noneZoreNum + 1);
    }
}
```