天然蓄水库

知识点 双指针

时间限制：1s 空间限制：256MB 限定语言：不限

题目描述：

描述：
公元2919年，人类终于发现了一颗宜居星球——X星。现想在X星一片连绵起伏的山脉间建一个天热蓄水库，如何选取水库边界，使蓄水量最大？

要求：

山脉用正整数数组s表示，每个元素代表山脉的高度。

选取山脉上两个点作为蓄水库的边界，则边界内的区域可以蓄水，蓄水量需排除山脉占用的空间

蓄水量的高度为两边界的最小值。

如果出现多个满足条件的边界，应选取距离最近的一组边界。

输出边界下标（从0开始）和最大蓄水量；如果无法蓄水，则返回0，此时不返回边界。

例如，当山脉为s=[3,1,2]时，则选取s[0]和s[2]作为水库边界，则蓄水量为1，此时输出：0 2:1

当山脉s=[3,2,1]时，不存在合理的边界，此时输出：0。

输入描述：

一行正整数，用空格隔开，例如输入
1 2 3
表示s=[1,2,3]

输出描述：

当存在合理的水库边界时，输出左边界、空格、右边界、英文冒号、蓄水量；例如
0 2:1
当不存在合理的书库边界时，输出0；例如
0

补充说明：

数组s满足：
1 <= length(s) <= 10000
0 <= s[i] <= 10000

示例1

输入:

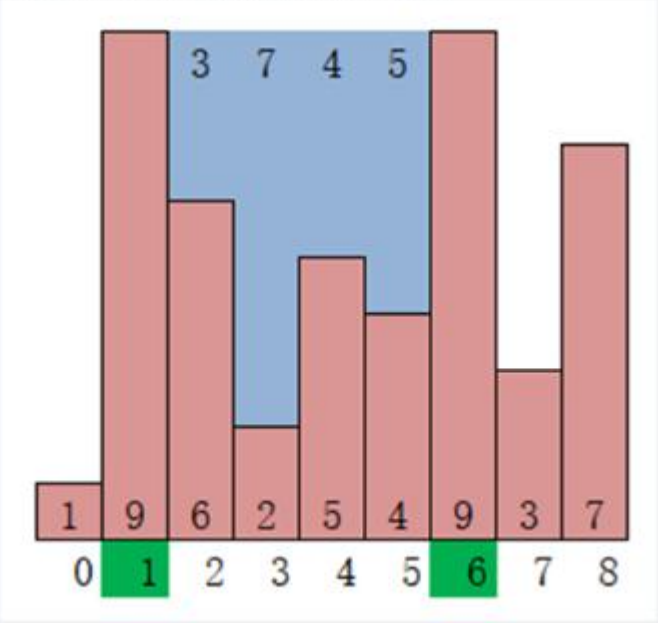
1 9 6 2 5 4 9 3 7

输出:

1 6:19

说明:

经过分析，选取s[1]和s[6]时，水库蓄水量为19（3+7+4+5）



示例2

输入：

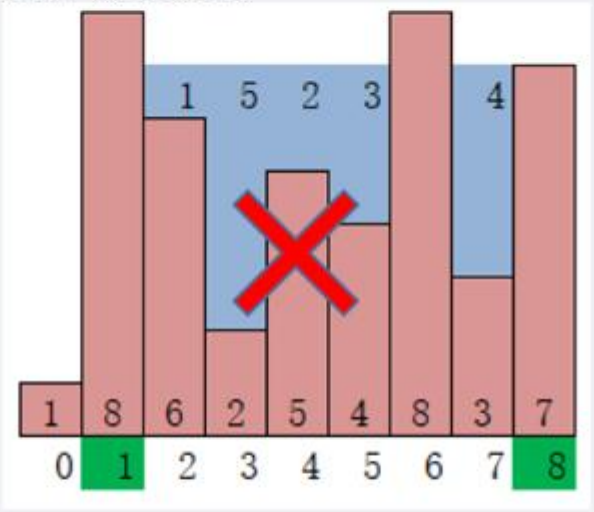
1 8 6 2 5 4 8 3 7

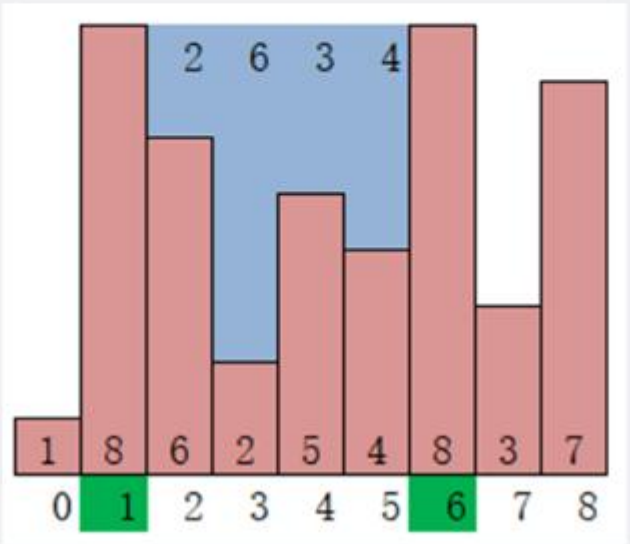
输出：

1 6:15

说明：

经过分析，选取s[1]和s[8]时，水库蓄水量为15；同样选取s[1]和s[6]时，水库蓄水量也为15。由于后者下标距离小（为5），故应选取后者。





示例3

输入:

```
1 2 3
```

输出:

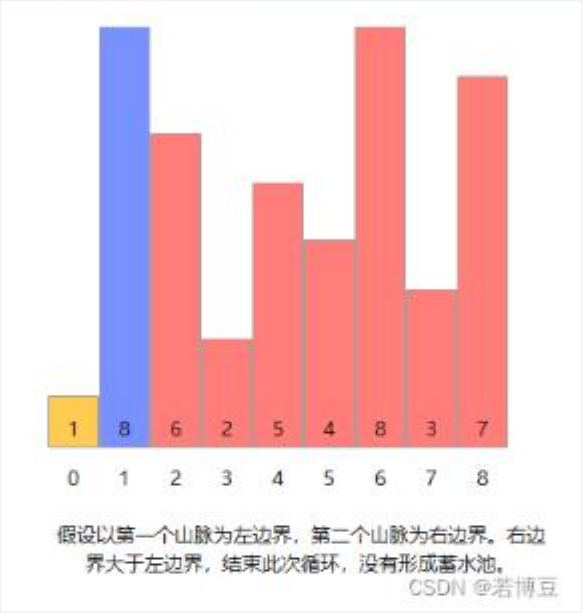
```
0
```

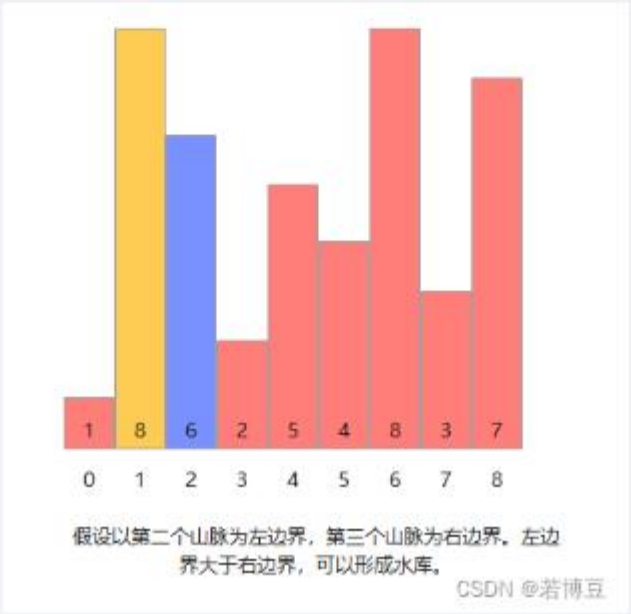
说明:

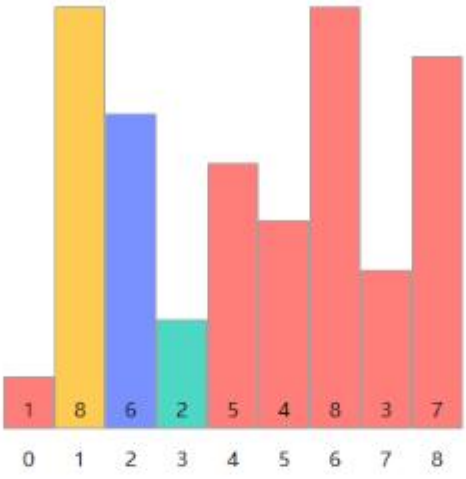
不存在合理的水库边界。

解题思路：

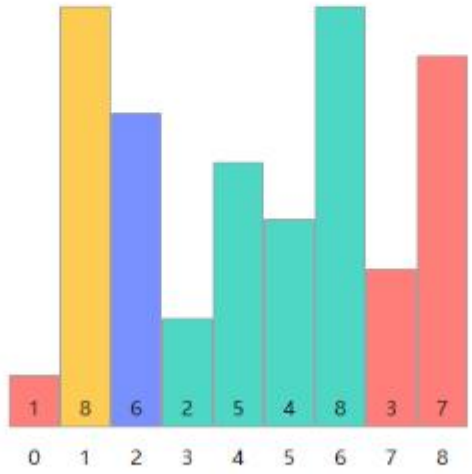
这里的右边界相当于水库中第二高的山脉。



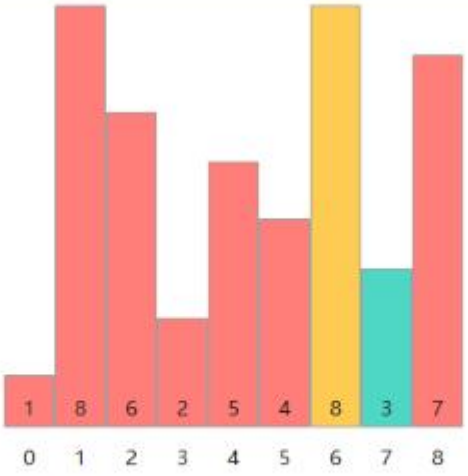




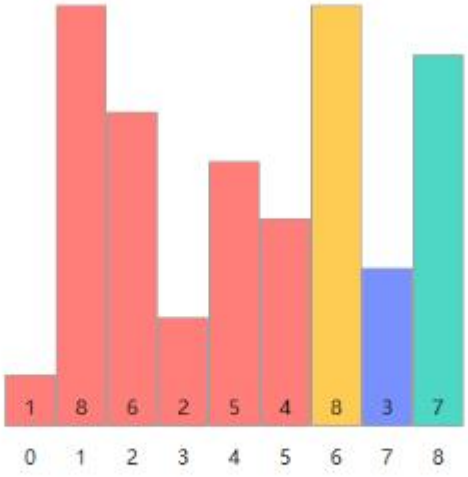
第四个山脉高度为2，满足山脉要求。



当遍历到第七山脉高度为8，等左边界，这样无论后面的山脉多高都不起作用，形成水库，退出循环。水库各山脉集合={8,6,2,5,4,8}，求出水库高度=8（集合首末两值的最小值），遍历集合，通过计算山脉高度与水库高度的差值来求出水库容量=2+6+3+4=15。



因为第七个山脉已经与之前的山脉形成了水库，所以我们可以从第七个山脉开始继续寻找水库。将第七山脉作为左边界，第八山脉为右边界。



第九山脉大于右边界，形成新的右边界。同时说明可以形成水库。水库山脉集合={8,3,7}；height=7，水库大小=7-3=4。

```
public class Main{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        String[] strs = sc.nextLine().split(" ");

        int res = 0;
        int indexLeft = 0;   //最大蓄水池左边界
        int indexRight = 0;        //最大蓄水池右边界
        for(int i=indexRight; i<strs.length-2; i++){

            int left = Integer.valueOf(strs[i]);       //左侧边界
            int right = Integer.valueOf(strs[i+1]);       //右侧边界
            if(left <= right){        //左边界小于等于右边界，不能形成蓄水池
                continue;
            }

            List<Integer> list = new ArrayList<>();        //山脉集合
            list.add(left);
            list.add(right);
            boolean isSuccess = false;   //是否形成蓄水池
            int index = 0;   //蓄水池右边界索引

            for(int j = i+2; j<strs.length; j++){
                int height = Integer.valueOf(strs[j]);   //此时山脉高度
                list.add(height);
                if(height > right){     //大于右边界时，形成新的右边界
                    right = height;
                    index = list.size() - 1;
                    isSuccess = true;
                }
                if(height >= left){       //右边界大于左边界直接跳出
                    break;
                }
            }

            if(isSuccess){
                int height = Math.min( list.get(0), list.get(index));          //求出蓄水池边界
                int count = 0;          //蓄水池面积
                for(int k=1; k<index; k++){
                    count += (height - list.get(k));
                }
                res = Math.max( res, count);       //求出蓄水池面积最大值
```

```
                    if(res == count){
                        indexLeft = i;          //左边界索引
                        indexRight = index + i;         //右边界索引
                    }
                }
            }

            if(res == 0){
                System.out.println(0);
            }else {
                System.out.println(indexLeft + " " + indexRight + ":" + res);
            }

        }

}
```