

华为OD机试 - 模拟消息队列 (Java & JS & Python)

原创 伏城之外 已于 2023-05-26 22:07:09 修改 1914 收藏 5

版权

分类专栏: 华为OD机试AB (Java & JS & Python)

文章标签: 算法 华为机试 Java JavaScript Python

OD 华为OD机试AB (Ja... 同时被 2 个专栏收录
该专栏为热销专栏榜 第2名

¥59.90 3381 订阅 371 篇文章

已订阅

题目描述



让我们来模拟一个 消息队列 的运作，有一个发布者和若干消费者，发布者会在给定的时刻向消息队列发送消息，

- 若此时消息队列有消费者订阅，这个消息会被发送到订阅的消费者中优先级最高（输入中消费者按优先级升序排列）的一个；
- 若此时没有订阅的消费者，该消息被消息队列丢弃。

消费者则会在给定的时刻订阅消息队列或取消订阅。

- 当消息发送和订阅发生在同一时刻时，先处理订阅操作，即同一时刻订阅的消费者成为消息发送的候选。
- 当消息发送和取消订阅发生在同一时刻时，先处理取消订阅操作，即消息不会被发送到同一时刻取消订阅的消费者。

输入描述

输入为两行。

第一行为 $2N$ 个正整数，代表发布者发送的 N 个消息的时刻和内容（为方便解折，消息内容也用正整数表示）。第一个数字是第一个消息的发送时刻，第二个数字是第一个消息的内容，以此类推。用例保证发送时刻不会重复，但注意消息并没有按照发送时刻排列。

第二行为 $2M$ 个正整数，代表 M 个消费者订阅和取消订阅的时刻。第一个数字是第一个消费者订阅的时刻，第二个数字是第一个消费者取消订阅的时刻，以此类推。用例保证每个消费者的取消订阅时刻大于订阅时刻，消费者按优先级 升序排列。

两行的数字都由空格分隔。 N 不超过100， M 不超过10，每行的长度不超过1000字符。

输出描述

输出为 M 行，依次为 M 个消费者收到的消息内容，消息内容按收到的顺序排列，且由空格分隔；

若某个消费者没有收到任何消息，则对应的行输出-1.

用例

输入	2 22 1 11 4 44 5 55 3 33 1 7 2 3
输出	11 33 44 55 22
说明	消息11在1时刻到达，此时只有第一个消费者订阅，消息发送给它； 消息22在2时刻到达，此时两个消费者都订阅了，消息发给优先级最高的第二个消费者； 消息33在时刻3到达，此时只有第一个消费者订阅，消息发给它； 余下的消息按规则也是发给第一个消费者。

输入	5 64 11 64 9 97 9 11 4 9
输出	97 64
说明	消息64在5时刻到达，此时只有第二个消费者订阅，消息发给它； 消息97在9时刻到达，此时只有第一个消费者订阅（因为第二个消费者刚好在9时刻取消订阅），消息发给它； 11时刻也到达了一个内容为64的消息，不过因为没有消费者订阅，消息被丢弃。

```
1 import java.util.ArrayList;
2 import java.util.Arrays;
3 import java.util.Scanner;
4 import java.util.StringJoiner;
5
6 public class Main {
7     public static void main(String[] args) {
8         Scanner sc = new Scanner(System.in);
9
10        int[] pubArr = Arrays.stream(sc.nextLine().split(
11            "\n")).mapToInt(Integer::parseInt).toArray();
12        int[] subArr = Arrays.stream(sc.nextLine().split(
13            "\n")).mapToInt(Integer::parseInt).toArray();
14
15        getResult(pubArr, subArr);
16    }
17
18    public static void getResult(int[] pubArr, int[] subArr) {
19        int n = pubArr.length;
20        int m = subArr.length;
21
22        // 将第一行输入两个一组，收集到：发布者数组中
23        int[][] publisher = new int[n / 2][];
24        for (int i = 0, k = 0; i < n; i += 2) {
25            publisher[k++] = new int[] {pubArr[i], pubArr[i + 1]}; // [发布时刻, 发布内容]
26        }
27
28        // 将第二行输入两个一组，收集到：订阅者数组中
29        int[][] subscriber = new int[m / 2][];
30        for (int j = 0, k = 0; j < m; j += 2) {
31            subscriber[k++] = new int[] {subArr[j], subArr[j + 1]}; // [订阅时刻, 取消订阅时刻]
32        }
33
34        // 按照发布时刻升序：发布者数组
35        Arrays.sort(publisher, (a, b) -> a[0] - b[0]);
36
37        // 为每一个订阅者构造一个的订阅内容集合
```

```
36     ArrayList<ArrayList<Integer>> subContent = new ArrayList<>();
37     for (int j = 0; j < subscriber.length; j++) subContent.add(new ArrayList<>());
38
39     // 遍历发布者
40     for (int[] pub : publisher) {
41         int pubTime = pub[0]; // 发布时刻
42         int pubContent = pub[1]; // 发布内容
43
44         // 倒序遍历订阅者，因为后面的订阅者优先级更高，因此倒序可以实现高优先级的订阅者先匹配到发
布者
45         for (int j = subscriber.length - 1; j >= 0; j--) {
46             int subTime = subscriber[j][0]; // 订阅时刻
47             int unSubTime = subscriber[j][1]; // 取消订阅时刻
48
49             // 如果 订阅时刻 <= 发布时刻 < 取消订阅时刻
50             if (pubTime >= subTime && pubTime < unSubTime) {
51                 // 那么该订阅者就可以收到发布的内容
52                 subContent.get(j).add(pubContent);
53                 // 题目说，发布内容只会被最高优先级的订阅者收到
54                 break;
55             }
56         }
57     }
58
59     // 打印
60     for (ArrayList<Integer> contents : subContent) {
61         if (contents.size() == 0) {
62             System.out.println("-1");
63         } else {
64             StringJoiner sj = new StringJoiner(" ");
65             for (Integer content : contents) sj.add(content + " ");
66             System.out.println(sj.toString());
67         }
68     }
69 }
70 }
```