

华为OD机试 - 支持优先级的队列 (Java & JS & Python)

原创 伏城之外

已于 2023-07-03 18:49:49 修改 678 收藏

版权

分类专栏:

华为OD机试AB (Java & JS & Python)

文章标签:

华为机试

算法

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

¥99.00

3382 订阅

371 篇文章

已订阅

题目描述

实现一个支持优先级的队列，高优先级先出队列；同优先级时先进先出。

如果两个输入数据和优先级都相同，则后一个数据不入队列被丢弃。

队列存储的数据内容是一个整数。↓

输入描述

一组待存入队列的数据 (包含内容和优先级)

输出描述

队列的数据内容(优先级信息输出时不再体现)

备注

不用考虑输入数据不合法的情况，测试数据不超过100个

用例

输入	(10,1),(20,1),(30,2),(40,3)
输出	40,30,10,20
说明	输入样例中，向队列写入了4个数据，每个数据由数据内容和优先级组成。 输入和输出内容都不含空格。 数据40的优先级最高，所以最先输出，其次是30； 10和20优先级相同，所以按输入顺序输出。

输入	(10,1),(10,1),(30,2),(40,3)
输出	40,30,10
说明	输入样例中，向队列写入了4个数据，每个数据由数据内容和优先级组成。 输入和输出内容都不含空格。 数据40的优先级最高，所以最先输出，其次是30； 两个10和10构成重复数据，被丢弃一个。

题目解析

本题看上去是让我们使用优先队列，但是实际上不是。

本题在维护优先级的同时，也要维护去重特性。其中较难的功能点是：

对于相同优先级的，且不重复的任务，维持插入顺序。

这个功能其实就是在我们实现一个：维护了插入顺序的 Set 集合 Q。

- 对于JS而言，其Set本身就是一个维护了插入顺序的去重集合，因此可以直接复用。
- 对于Java而言，我们可以复用LinkedHashSet。
- 对于Python而言，其set不维护元素插入顺序，因此我们不能复用。为了避免自己实现OrderedSet，我们可以复用Python的object能力，因此Python的object的keys列是维护了插入顺序的去重集合。

我的解题思路如下：

定义一个字典，用于收集相同优先级的数据，即字典的key是优先级，value是一个维护了插入顺序的Set集合。

收集完后，我们可以取出字典的key列进行降序排序（高优先级优先），然后遍历降序后的key列，逐个打印字典对应key下的value内容。

```
1 import java.util.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6
7         String line = sc.nextLine();
8
9         int[][] tasks =
10             Arrays.stream(line.substring(1, line.length() - 1).split("\\\\),\\\\("))
11                 .map(s ->
12                     Arrays.stream(s.split(","))
13                         .mapToInt(Integer::parseInt)
14                         .toArray()
15                         .toArray(int[][]::new));
16
17         System.out.println(getResult(tasks));
18     }
19
20     public static String getResult(int[][] tasks) {
21         HashMap<Integer, LinkedHashSet<Integer>> map = new HashMap<>();
22
23         for (int[] task : tasks) {
24             int num = task[0];
25             int priority = task[1];
26
27             if (!map.containsKey(priority)) {
28                 map.put(priority, new LinkedHashSet<>());
29             }
30
31             map.get(priority).add(num);
32         }
33
34         List<Integer> keys = new ArrayList<>(map.keySet());
35         keys.sort(Comparator.reverseOrder());
36
37         for (Integer key : keys) {
38             System.out.println(key + " " + map.get(key));
39         }
40     }
41 }
```

```
24     map.putIfAbsent(priority, new LinkedHashSet<>());
25     map.get(priority).add(num);
26 }
27
28 StringJoiner sj = new StringJoiner(",");
29 map.keySet().stream()
30     .sorted((a, b) -> b - a)
31     .forEach(
32         p -> {
33             map.get(p).forEach(num -> sj.add(num + ""));
34         });
35
36 return sj.toString();
37 }
38 }
```