

Java-数组排序哈希表-找出两个整数数组中同时出现的整数

题目描述：现有两个整数数组，需要你找出两个数组中同时出现的整数，并按照如下要求输出： 1、有同时出现的整数时，先按照同时出现次数（整数在两个数组中都出现并且出现次数较少的那个）进行归类，然后按照出现次数从小到大依次按行输出。 2、没有同时出现的整数时，输出 NULL。

输入描述：第一行为第一个整数数组，第二行为第二个整数数组，每行数据中整数与整数之间以英文逗号分隔，整数的取值范围为 $[-200, 200]$ ，数组长度的范围为 $[1, 10000]$ 之间的整数。

输出描述：按照出现次数从小到大依次按行输出，每行输出的格式为：出现次数：该出现次数下的整数升序排序的结果。 格式中的“:”为英文冒号，整数间以英文逗号分隔。

示例 展开

示例1
输入：5,3,6,-8,0,11
2,8,8,8,-1,15
输出：NULL
说明：两个整数数组没有同时出现的整数，输出NULL。

示例2
输入：5,8,11,3,6,8,8,-1,11,2,11,11
11,2,11,8,6,8,8,-1,8,15,3,-9,11
输出：1:-1,2,3,6
3:8,11
说明：两个整数数组中同时出现的整数为-1、2、3、6、8、11，其中同时出现次数为1的整数为-1,2,3,6(升序排序)，同时出现次数为3的整数为8,11(升序排序)，先升序输出出现次数为1的整数，再升序输出出现次数为3的整数。

```

1  import java.util.*;
2
3  // 注意类名必须为 Main, 不要有任何 package xxx 信息
4  public class Main {
5      public static void main(String[] args) {
6          Scanner in = new Scanner(System.in);
7          // 注意 hasNext 和 hasNextLine 的区别
8          while (in.hasNextLine()) { // 注意 while 处理多个 case
9              String[] first = in.nextLine().split(",");
10             String[] second = in.nextLine().split(",");
11
12             Map<Integer, Integer> map1 = new HashMap<>();
13             Map<Integer, Integer> map2 = new HashMap<>();
14             Map<Integer, List<Integer>> result = new TreeMap<>();
15
16             for (int i = 0; i < first.length; i++) {
17                 Integer n = Integer.parseInt(first[i]);
18                 if (map1.get(n) == null) {
19                     map1.put(n, 1);
20                 } else {
21                     map1.put(n, map1.get(n) + 1);
22                 }
23             }
24
25             for (int i = 0; i < second.length; i++) {
26                 Integer n = Integer.parseInt(second[i]);
27                 if (map2.get(n) == null) {
28                     map2.put(n, 1);
29                 } else {
30                     map2.put(n, map2.get(n) + 1);
31                 }
32             }
33
34             boolean contains = false;
35             for (Map.Entry<Integer, Integer> entry : map1.entrySet()) {
36                 if (map2.containsKey(entry.getKey())) {
37                     // 比较谁的value小, 也就是谁的count小
38                     // 把count, list.add

```

```

38 //TCOUNT,ISTADD
39 contains = true;
40 Integer map2Count = map2.get(entry.getKey());
41 if (entry.getValue() < map2Count) {
42     if (result.containsKey(entry.getValue())) {
43         result.get(entry.getValue()).add(entry.getKey());
44     } else {
45         List<Integer> list = new ArrayList<>();
46         list.add(entry.getKey());
47         result.put(entry.getValue(), list);
48     }
49 } else {
50     if (result.containsKey(map2Count)) {
51         result.get(map2Count).add(entry.getKey());
52     } else {
53         List<Integer> list = new ArrayList<>();
54         list.add(entry.getKey());
55
56         result.put(map2Count, list);
57     }
58 }
59 }
60
61 if (!contains) {
62     System.out.println("NULL");
63     return;
64 }
65
66 for (Map.Entry<Integer, List<Integer>> entry : result.entrySet()) {
67     entry.getValue().sort(new Comparator<Integer>() {
68         @Override
69         public int compare(Integer o1, Integer o2) {
70             return o1 - o2;
71         }
72     });
73 }
74
75 for (Map.Entry<Integer, List<Integer>> entry : result.entrySet()) {
76     System.out.println(entry.getKey() + ":" + printArray(entry.getValue()));
77 }
78 }
79 }
80
81 private static String printArray(List<Integer> list) {
82     StringBuilder sb = new StringBuilder();
83     for (int i = 0; i < list.size(); i++) {
84         sb.append(list.get(i)).append(",");
85     }
86     return sb.toString().substring(0, sb.length() - 1);
87 }
88 }

```

