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。

```java
import java.util.*;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        while (in.hasNextLine()) { // 注意 while 处理多个 case
            String[] fisrt = in.nextLine().split(",");
            String[] second = in.nextLine().split(",");

            Map<Integer, Integer> map1 = new HashMap<>();
            Map<Integer, Integer> map2 = new HashMap<>();
            Map<Integer, List<Integer>> result = new TreeMap<>();

            for (int i = 0; i < fisrt.length; i++) {
                Integer n = Integer.parseInt(fisrt[i]);
                if (map1.get(n) == null) {
                    map1.put(n, 1);
                } else {
                    map1.put(n, map1.get(n) + 1);
                }
            }

            for (int i = 0; i < second.length; i++) {
                Integer n = Integer.parseInt(second[i]);
                if (map2.get(n) == null) {
                    map2.put(n, 1);
                } else {
                    map2.put(n, map2.get(n) + 1);
                }
            }

            boolean contains  = false;
            for (Map.Entry<Integer, Integer> entry : map1.entrySet()) {
                if (map2.containsKey(entry.getKey())) {
                    //比较谁的value小，也就是谁的count小
                    //把count,list.add
```

```
                //把count,list add
                contains = true;
                Integer map2Count = map2.get(entry.getKey());
                if (entry.getValue() < map2Count) {
                    if (result.containsKey(entry.getValue())) {
                        result.get(entry.getValue()).add(entry.getKey());
                    } else {
                        List<Integer> list = new ArrayList<>();
                        list.add(entry.getKey());
                        result.put(entry.getValue(), list);
                    }
                } else {
                    if (result.containsKey(map2Count)) {
                        result.get(map2Count).add(entry.getKey());
                    } else {
                        List<Integer> list = new ArrayList<>();
                        list.add(entry.getKey());
                        result.put(map2Count, list);
                    }
                }
            }
        }

        if (!contains) {
            System.out.println("NULL");
            return;
        }

        for (Map.Entry<Integer, List<Integer>> entry : result.entrySet()) {
            entry.getValue().sort(new Comparator<Integer>() {
                @Override
                public int compare(Integer o1, Integer o2) {
                    return o1 - o2;
                }
            });
        }

        for (Map.Entry<Integer, List<Integer>> entry : result.entrySet()) {
            System.out.println(entry.getKey() + ":" + printArray(entry.getValue()));
        }
    }
}

private static String printArray(List<Integer> list) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < list.size(); i++) {
        sb.append(list.get(i)).append(",");
    }
    return sb.toString().substring(0, sb.length() - 1);
}
}
```