华为OD机试 - 阿里巴巴找黄金宝箱(II)（Java & JS & Python）

题目描述

一贫如洗的樵夫阿里巴巴在去砍柴的路上，无意中发现了强盗集团的藏宝地，藏宝地有编号从0-N的箱子，每个箱子上面贴有箱子中藏有金币的数量。

从金币数量中选出一个数字集合，并销毁贴有这些数字的每个箱子，如果能销毁一半及以上的箱子，则返回这个数字集合的最小大小

输入描述

一个数字字串，数字之间使用逗号分隔，例如：6,6,6,6,3,3,3,1,1,5

字串中数字的个数为偶数，并且

- 1 ≤ 字串中数字的个数 ≤ 100000
- 1 ≤ 每个数字 ≤ 100000

输出描述

这个数字集合的最小大小，例如: 2

用例

输入	1,1,1,1,3,3,3,6,6,8
输出	2
说明	选择集合{1,8}，销毁后的结果数组为[3,3,3,6,6]，长度为5，长度为原数组的一半。 大小为 2 的可行集合还有{1,3},{1,6},{3,6}。 选择 {6,8} 集合是不可行的，它销后的结果数组为[1,1,1,1,3,3,3]，新数组长度大于原数组的二分之一。

输入	2,2,2,2
输出	1
说明	我们只能选择集合{2}，销毁后的结果数组为空。

题目解析

本题主要考察 贪心🔍思维。

本题中有几个数字的概念：

- 箱子上贴的数字
- 箱子中金币数量
- 箱子的个数

其中，箱子上贴的数字就是箱子中金币数量，这里我们可以将这两个概念合并为一个箱子的类别

- 箱子的类别（等价于箱子上贴的数字，等价于箱子中金币数量）
- 箱子的个数

我们需要统计 各个类别的出现次数。

然后，对这些类别按照出现次数进行降序排序，即第一个类别的箱子的出现次数最多。

本题想要销毁最多的箱子，但是又要尽可能保留最多类别，因此根据贪心原则，我们应该优先销毁出现次数最多的类别，这样就能保证销毁最少的类别，但是可以销毁更多的箱子，让销毁的箱子数量尽快达到总数的一半及以上。

我们最后返回销毁的类别的数量即可。

```java
import java.util.*;
import java.util.stream.Collectors;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int[] nums = Arrays.stream(sc.nextLine().split(",")).mapToInt(Integer::parseInt).toArray();
    System.out.println(getResult(nums));
  }

  public static int getResult(int[] nums) {
    // 箱子上贴的数字可以看出 类别
    // 统计每一个类别出现的次数，key是类别，value是类别出现次数
    HashMap<Integer, Integer> count = new HashMap<>();

    for (int num : nums) {
      count.put(num, count.getOrDefault(num, 0) + 1);
    }

    // half是箱子数量的一半，注意向上取整，因为比如箱子有7个，则销毁一半及以上的箱子数量至少是4个，而不是3个
    int half = (int) Math.ceil(nums.length / 2.0);
```

```java

    // 按类别出现次数降序 类别
    List<Map.Entry<Integer, Integer>> collect =
        count.entrySet().stream()
            .sorted((a, b) -> b.getValue() - a.getValue())
            .collect(Collectors.toList());

    // remove记录销毁的箱子数量
    int remove = 0;
    // numCount记录销毁的类别数量
    int numCount = 0;
    // 贪心思维,想要销毁最多的箱子数，又要销毁的箱子的类别数最少，则应该尽可能销毁出现次数多的类别，因此前面对按照次数降序了类别
    for (Map.Entry<Integer, Integer> entry : collect) {
      remove += entry.getValue();
      numCount++;
      // 一旦达标，则返回销毁的类别数量
      if (remove >= half) {
        return numCount;
      }
    }

    return -1; // 走不到此行，仅用于代码健壮性
  }
}
```