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

原创 伏城之外 已于 2023-07-20 22:13:32 修改 969 收藏 4 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 Python JavaScript

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

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，每个箱子上面贴有一个数字。

阿里巴巴念出一个咒语数字，查看宝箱是否存在两个不同箱子，这两个箱子上贴的数字相同，同时这两个箱了的编号之差的绝对值小于等于咒语数字，
如果存在这样的一对宝箱，请返回最先找到的那对宝箱左边箱子的编号，如果不存在则返回-1.

## 输入描述

第一行输入一个数字字串，数字之间使用逗号分隔，例如: 1,2,3,1

- 1 ≤ 字串中数字个数 ≤ 100000
- -100000 ≤ 每个数字值 ≤ 100000

第二行输入咒语数字，例如: 3

- 1 ≤ 咒语数字 ≤ 100000

## 输出描述

存在这样的一对宝箱，请返回最先找到的那对宝箱左边箱子的编号，如果不存在则返回-1

## 用例

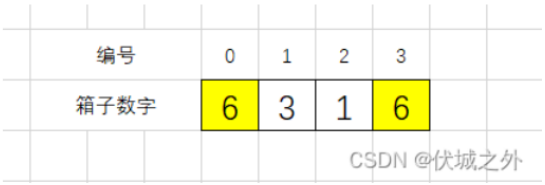
| | |
|---|---|
| 输入 | 6,3,1,6<br>3 |
| 输出 | 1 |
| 说明 | 无 |

| | |
|---|---|
| 输入 | 5,6,7,5,6,7<br>2 |
| 输出 | 0 |
| 说明 | 无 |

### 题目解析

本题的用例似乎有问题。

用例1

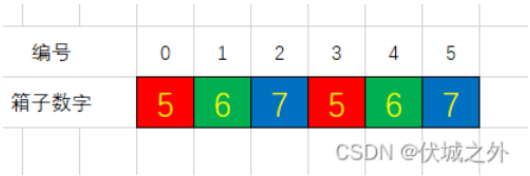



编号0，和编号3的箱子数字相同，编号差的绝对值 = 3。符合<=咒语3 的要求，因此返回这对箱子左边的箱子编号0。

但是用例1输出的是1，难道箱子编号要从1开始？但是题目描述开头就说了

> 藏宝地有编号从0-N的箱子

用例2




编号0和编号3的箱子数字相同，编号差的绝对值 = 3

编号1和编号4的箱子数字相同，编号差的绝对值 = 3

编号2和编号5的箱子数字相同，编号差的绝对值 = 3

以上三对箱子没有编号差的绝对值 <= 咒语2的，因此本用例应该返回-1

但是用例2输出的是0

？？？？

因此，下面解法是不符合用例要求的。实际考试时，需要灵活应对。

我的解题思路如下：

首先定义一个字典lastIdx，用于统计对应数字的上一次出现的箱子编号

再定义一个set集合find，用于统计对应数字是否已经找到了符合咒语要求的箱子对

接下来遍历箱子：

- 如果find有箱子数字，那么说明该数字已经找到符合要求的箱子对，无需找后续的，因为题目只需要找每个数字的第一对符合咒语的
- 如果find没有箱子数字：

1. 如果lastIdx没有该箱子数字，则将当前箱子的编号value，和箱子数字key，录入lastIdx
2. 如果lastIdx有该箱子数字num，则计算当前箱子编号idx，和相同数字的上一次出现的箱子编号lastIdx[num]的差值绝对值，看是否符合咒语要求，若符合，则记录此时lastIdx[num]（左边箱子编号）到ans中，并将num录入find，若不符合，则只更新lastIdx[num] = idx

最后，从ans记录的所有左边箱子编号中，找出最左边的即可。

2023.07.20

本题存在歧义，题目最后要求返回

> 如果存在这样的一对宝箱，请返回最先找到的那对宝箱左边箱子的编号，如果不存在则返回-1.

其中，“最先找到的那对宝箱”，歧义在于“那对”，即一对宝箱有两个

- 先找到这对宝箱的左边的宝箱为“最先”？
- 先找到这对宝箱的右边的宝箱为“最先”？

比如用例

```
1,2,2,1
10
```

如果以先找到这对宝箱的左边的宝箱为“最先”，则这对宝箱贴的数字应该是1

如果以先找到这对宝箱的右边的宝箱为“最先”，则这对宝箱贴的数字应该是2

前面的解析是按照：先找到这对宝箱的左边的宝箱为“最先”，的逻辑来实现的。

鉴于争议比较大，这里我再提供一种：先找到这对宝箱的右边的宝箱为“最先”，的解法，大家考试时可以都尝试一下。

### 先找到这对宝箱的左边的宝箱为“最先”

```java
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int[] boxes = Arrays.stream(sc.nextLine().split(",")).mapToInt(Integer::parseInt).toArray();
    int len = Integer.parseInt(sc.nextLine());

    System.out.println(getResult(boxes, len));
  }

  public static int getResult(int[] boxes, int len) {
    // 统计该数字上一个箱子的编号
    HashMap<Integer, Integer> lastIdx = new HashMap<>();
    // 对应数字的箱子已经找到了，符合咒语要求的箱子对
```

```java
    HashSet<Integer> find = new HashSet<>();

    int ans = -1;

    for (int i = 0; i < boxes.length; i++) {
      // 箱子上贴的数字
      int num = boxes[i];

      // 该数字是否已经找到符合咒语要求的箱子对，如果找到了，则不需要再看后面的，只找第一对即可
      if (find.contains(num)) continue;

      // 检查箱子对是否符合咒语要求
      if (lastIdx.containsKey(num) && i - lastIdx.get(num) <= len) {
        find.add(num);
        ans = ans == -1 ? lastIdx.get(num) : Math.min(ans, lastIdx.get(num));
      } else {
        lastIdx.put(num, i);
      }
    }

    return ans;
  }
}
```