华为OD机考算法题：不开心的小朋友

御剑乐逍遥 已于 2023-09-14 18:38:18 修改

目录

题目部分

题目 不开心的小朋友

难度 难

题目说明 游乐场里增加了一批摇摇车，非常受小朋友欢迎，但是每辆摇摇车同时只能有一个小朋友使用，如果没有空余的摇摇车需要排队等候，或者直接离开，最后没有玩上的小朋友会非常不开心。
请根据今天小朋友的来去情况，统计不开心的小朋友数量。
1.摇摇车数量为N，范围是: 1 <= N < 10;
2.每个小朋友都对应一个编码，编码是不重复的数字，今天小朋友的来去情况，可以使用编码表示为:1 1 2 3 2 3。(若小朋友离去之前有空闲的摇摇车，则代表玩耍后离开;不考虑小朋友多次玩的情况)。小朋友数量≤ 100。
3.题目保证所有输入数据无异常且范围满足上述说明。

输入描述 第一行: 摇摇车数量。
第二行: 小朋友来去情况。

输出描述 返回不开心的小朋友数量。

补充说明 无

示例

示例1

输入　1
1 2 1 2

输出　0

说明　第一行，1个摇摇车第二行，1号来 2号来(排队) 1号走 2号走(1号走后摇摇车已有空闲，所以玩后离开)。

示例2

输入　1
1 2 2 3 1 3

输出　1

说明　第一行，1个摇摇车第二行，1号来 2号来(排队) 2号走(不开心离开) 3号来(排队)1号走 3号走(1号走后摇摇车已有空闲，所以玩后离开)。

解读与分析

题目解读：

第一行输入摇摇车的个数。
第二行输入中的数字为成对出现，一个数字第一次出现时，表示该编号的小朋友来了，此数字第二次出现时，表示该小朋友走了。

当下朋友来的时候，如果有摇摇车处于空闲状态，则玩摇摇车；如果没有空闲则派对；如果玩摇摇车的小朋友离开，则排在队伍最前面的小朋友去玩空出来的摇摇车；如果排队的小朋友离开，则此小朋友不开心。
此题需要计算不开心小朋友的人数。

分析与思路：

在讲解思路之前，申明 4 个变量：
1. shakeToy，set<Integer>，整形数字集合，初始为空。集合中存放正在玩摇摇车的小朋友的编号。集合最大size 为用户输入的摇摇车个数。
2. kidQueue，List<Integer>，用于存放正在排队的小朋友。
3. allKidSet，Set<Integer>，整形数字集合，初始为空。用于存放所有正在排队和正在玩摇摇车的小朋友编号，其数据是shakeToy 和 kidList 的合集。
4. unhappyCnt，整形数字，初始值 0，用于统计不高兴小朋友的个数。

遍历第二行输入字符串中的数字，如果此数字：
1. 首次出现，则数字在 shakeToy 和 kidList 都不存在。判断此时 shakeToy 是否满了（即其 size 是否等于摇摇车个数），如果没有满，把此数字放到 shakeToy 中去；如果 shakeToy 满了，则把把此数字放到 kidList 的末尾。继续遍历下一个数字。
2. 第二次出现，则数字一定在 shakeToy 或 kidList 中。如果在 shakeToy 中，则把他从 shakeToy 中删除，并把 kidList 排在首位（如果 kidList 不为空）的数字放到 shakeToy 中；如果在 kidList 中，从 kidList 中删除，表示此小朋友还在排队中就离开，unhappyCnt 加 1。继续遍历下一个数字。
最后统计的 unhappyCnt 就是期望的结果。

此算法只需要遍历一次字符串，时间复杂度为 O(n)，空间复杂度为 O(n)。

代码实现

Java代码

复制

```java
import java.util.Set;
import java.util.HashSet;
import java.util.List;
import java.util.ArrayList;
import java.util.Scanner;

/**
 * 不开心的小朋友
 * @since 2023.09.12
 * @version 0.1
 * @author Frank
 *
 */
public class UnhappyKids {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNext()) {
            String input = sc.nextLine();
            int count = Integer.parseInt( input );
            input = sc.nextLine();
            String[] numbers = input.split( " " );
            processUnhappyKids( count, numbers );
        }

    }

    private static void processUnhappyKids( int count, String numbers[] )
    {
```

```
        Set<Integer> shakeToy = new HashSet<Integer>();
        List<Integer> kidQueue = new ArrayList<Integer>();
        Set<Integer> allKidSet = new HashSet<Integer>();
        int unhappyCnt = 0;

        for( int i = 0; i < numbers.length; i ++ )
        {
            int currentNum = Integer.parseInt( numbers[i] );

            // 如果首次出现
            if( !allKidSet.contains( currentNum ))
            {
                // 摇摇车满了
                if( shakeToy.size() >= count )
                {
                    kidQueue.add( currentNum );
                }else
                {
                    shakeToy.add( currentNum );
                }
                allKidSet.add( currentNum );
            }else  // 第二次出现，则在排队或者在玩游戏中
            {

                if( shakeToy.contains( currentNum ) )
                {
                    shakeToy.remove( currentNum );
                    if( kidQueue.size() > 0 )
                    {
                        int queueFirst = kidQueue.get( 0 );
                        shakeToy.add( queueFirst );
                        kidQueue.remove( 0 );
                    }
                }else  // 在排队中离开，不高兴
                {
                    // 删除 Object，而不是 index，此处要转换成 Integer。
                    kidQueue.remove( (Integer) currentNum );
                    unhappyCnt ++;
                }
                allKidSet.remove( currentNum );
            }
        }
        System.out.println( unhappyCnt );
```

```
    }
}
```

JavaScript代码

```
const rl = require("readline").createInterface({ input: process.stdin });
var iter = rl[Symbol.asyncIterator]();
const readline = async () => (await iter.next()).value;
void async function() {
    while (line = await readline()) {
        var count = parseInt(line);
        line = await readline();
        var numberArr = line.split(" ");
        processUnhappyKids(count, numberArr);
    }
}();

function processUnhappyKids(count, numbers) {
    var shakeToy = new Set();
    var kidQueue = new Array();
    var allKidSet = new Set();
    var unhappyCnt = 0;

    for (var i = 0; i < numbers.length; i++) {
        var currentNum = parseInt(numbers[i]);

        // 如果首次出现
        if (!allKidSet.has(currentNum)) {
            // 摇摇车满了
            if (shakeToy.size >= count) {
                kidQueue.push(currentNum);
            } else {
                shakeToy.add(currentNum);
            }
            allKidSet.add(currentNum);
        } else // 第二次出现，则在排队或者在玩游戏中
        {

            if (shakeToy.has(currentNum)) {
                shakeToy.delete(currentNum);
                if (kidQueue.length > 0) {
```

```
                var queueFirst = kidQueue.shift();
                shakeToy.add(queueFirst);
            }
        } else // 在排队中离开，不高兴
        {
            var idx = kidQueue.indexOf( currentNum );
            kidQueue.splice( idx, 1 );
            unhappyCnt++;
        }
        allKidSet.delete(currentNum);
    }
}
console.log(unhappyCnt);
}
```

(完)

文章知识点与官方知识档案匹配，可进一步学习相关知识

算法技能树 > 首页 > 概览　51727 人正在系统学习中