不开心的小朋友

题目描述：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

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）

```
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);

        int carNum = Integer.parseInt(in.nextLine());

        int[] childArr = Arrays.stream(in.nextLine().split(" ")).mapToInt(Integer::parseInt).toArray();

        System.out.println(solution(carNum,childArr));

    }

    private static int solution(int carNum, int[] childArr) {
        //玩游戏的孩子队列
        HashSet<Integer> playing = new HashSet<>();
        //正在等的孩子队列，如果有空摇摇车，则按照队列进行玩耍
        PriorityQueue<Integer> waiting = new PriorityQueue<>();

        //不开心的小朋友数量
        int res = 0;

        //遍历每个元素时，对应孩子可能有三种状态，刚到达，正在玩，正在等；
        //当遍历到的元素处于正在等状态时，对应孩子要走掉了，世界上又多了一个伤心
```

```
的小朋友
        for (int child : childArr) {
            //如果当前孩子已经在玩了，则此孩子要走了，玩到了摇摇车，很开心
            if(playing.contains(child)) {
                //从玩耍队列中移出
                playing.remove(child);
                //如果此时有在排队的孩子，则队首的孩子开始玩，否则空闲摇摇车+1
                if(!waiting.isEmpty()){
                    playing.add(waiting.poll());
                }else{
                    carNum++;
                }
            }
            //如果当前孩子之前在等，则此孩子要走了，没有玩到，世界上又多了一个伤
心的小朋友
            else if(waiting.contains(child)){
                res++;
                waiting.remove(child);
            }
            //如果当前孩子刚刚到达，有空闲摇摇车则直接玩耍，摇摇车-1；没有空闲摇
摇车则进行排队
            else{
                if(carNum >0){
                    playing.add(child);
                    carNum--;
                }else{
                    waiting.add(child);
                }
            }
        }
        return res;
    }
}
```