P4447 分组

题目描述

小可可的学校信息组总共有n 个队员，每个人都有一个实力值a[i]a[i]a[i]。现在，一年一度的编程大赛就要到了，小可可的学校获得了若干个参赛名额，教练决定把学校信息组的nnn 个队员分成若干个小组去参加这场比赛。

但是每个队员都不会愿意与实力跟自己过于悬殊的队员组队，于是要求分成的每个小组的队员实力值连续，同时，一个队不需要两个实力相同的选手。举个例子：[1,2,3,4,5][1, 2, 3, 4, 5][1,2,3,4,5]是合法的分组方案，因为实力值连续；[1,2,3,5][1, 2, 3, 5][1,2,3,5]不是合法的分组方案，因为实力值不连续；[0,1,1,2][0, 1, 1, 2][0,1,1,2]同样不是合法的分组方案，因为出现了两个实力值为1 的选手。

如果有小组内人数太少，就会因为时间不够而无法获得高分，于是小可可想让你给出一个合法的分组方案，满足所有人都恰好分到一个小组，使得人数最少的组人数最多，输出人数最少的组人数的最大值。

注意：实力值可能是负数，分组的数量没有限制。

输入格式

输入有两行：

第一行一个正整数n，表示队员数量。
第二行有n 个整数，第i 个整数a[i]表示第i 个队员的实力。

输出格式

输出一行，包括一个正整数，表示人数最少的组的人数最大值。

输入样例

```
7
4 5 2 3 -4 -3 -5
```

输出样例

```
3
```

解析

简而言之，让最少人数的组所包含的人数尽量多。

本题我们采取一个接龙的思想，把每个小组看成队，队中元素连续递增，每次在已有的队中选取满足条件的最短的队，在该队尾插入一个元素，由于要求相邻两个元素的值只差一，我们用一个结构体来维护一个队，其中las表示队尾元素的值，num表示队的长度。

将读入的元素从小到大排序，每次从已有队伍中倒着扫描找到第一个队尾元素比要插入元素小一的队，插入该元素。

为什么倒着扫就能找到最短的队？

首先，由于我们事先把所有元素从小到大排序，后面的元素一定不小于前面的元素，所以显然第i+1个队的队首元素一定不小于第i个队的队首元素。

由题目限制知若某元素能插入一个队中，则该元素一定比该队的队尾元素大一，所以能插入该元素的队的队尾元素是相同的。

而队中的元素是依次加一的，因为后面的队的队首元素不小于前面的队，所以当队尾元素相等时，后面的队一定不长于前面的队，所以从后往前倒着扫就能找到可以插入元素的最短的队。

对于测试数据则有：

0	1	2	3	4	5	6
-5	-4	-3	2	3	4	5

分组1	-5	-4	-3

分组2	2	3	4	5

编码

```
#include <bits/stdc++.h>

using namespace std;

int n, a[100005], ans = INT_MAX;

//定义接龙组的
struct node {
    //组内总人数
    int num;
    //最后一个人的实力值
    int las;
} s[100005];

int main() {
    //k是我们当前的组数
    int k = 0;
    //是否连接成功的标志
    bool fd;
```

```
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &a[i]);
    }
    //将所有的数据按照从小大大排序
    sort(a + 1, a + n + 1);
    //开始分组
    for (int i = 1; i <= n; i++) {
        //注意每次都要将标记进行重置
        //如果不存在接龙，则需要新开一个分组
        fd = false;
        //遍历全部分好的组
        for (int j = k; j>0; j--) {
            if (s[j].las == a[i] - 1) {
                //标记分组成功
                fd = true;
                //长度增加
                s[j].num++;
                //尾数增加
                s[j].las = a[i];
                //找到合适的分组，就跳出遍历
                break;
            }
        }
        //如果当前数字无法接龙，则需要开辟一个新组
        if (!fd) {
            s[++k].las = a[i];
            s[k].num = 1;
        }
    }
    //遍历所有的分组数据，寻找人数最少的一组
    for (int i = 1; i <= k; i++) {
        if (ans > s[i].num) {
            ans = s[i].num;
        }
    }
    printf("%d", ans);
    return 0;
}
```