前情提要：

本题解适用比赛：

2021-2022学年春季-每周编程两小时学习成果检验考试 http://scpc.fun/contest/1056

比赛及题解归属：

西南科技大学-计算机科学与技术学院-每周编程两小时课程组

本次比赛出题组成员：

黄小诚、纪佳宇、王弦德、朱杰、高源 (均为西南科技大学 ACM 竞赛团队成员及西南科技大学-计算机科学与技术学院-每周编程两小时课程组成员)

本次比赛题面二创者：

黄小诚

本次比赛组题者：

黄小诚

本题解编写者：

黄小诚

本次比赛及本次课程负责老师：

宋丽丽老师

注意：

本次比赛为 IOI 赛制，实时返回分数，取最高分，允许多次提交，按点给分。

本次比赛题目中出现的诗句均为原创诗句，**未经允许，请勿转载** (不止来自出题组中的成员)

Problem A. 期年

题意：

给定一个长度为 3 的字符串，将它翻转后输出

题解：

都告诉你是三位数 (长度为 3) 了，那么对症下药极其简单

1. 采用 stl 库中 $string$ 来记录字符串，倒序输出

2. 采用 $char$ 数组来记录字符串，倒序输出

3. 开 3 个字符型变量，顺着输入逆着输出

4. 模拟翻转三位数 (麻烦)

... (以下省略无数种做法)

易错点：

无

骗分技巧：

直接输出样例拿 10 分

示例代码：

A.期年.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
signed main()
{
    string w;cin>>w;
    cout<<w[2]<<w[1]<<w[0]<<endl;
}
```

Problem B. 终章

题意：

确定给出的字符串中是否存在子序列 "scpc" 。如果有输出第一个 's' 的位置，否则输出缺失的第一个字母

题解：

直接遍历字符串，使用一个变量 pos 记录第一个 's' 的位置。使用标记变量 now ，每次找到一个字符，now 就 $++$

最后通过 now 的值的大小决定输出什么。

易错点：

子序列和子串定义易混淆。不可以直接使用 strstr 函数

子序列不可以互换顺序，所以不可以通过只查询是否存在所需的三种字符来决定输出结果。(易错字符串：“cspc")

骗分技巧：

这种题，一看数字不好猜，可是如果要输出字母的话，答案只可能有 's' 'c' 'p' 三种情况

分别尝试单独输出这三个字符，你可以获得 30 分

示例代码：

B.终章.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
signed main()
{
    int len,pos=0,flag=1,now=1;
    string ss;cin>>len>>ss;
    for(int i=0;i<len;i++)
    {
        if(flag&&ss[i]=='s') pos=i+1,flag=0;
        if(now==1&&ss[i]=='s') now++;
        if(now==2&&ss[i]=='c') now++;
        if(now==3&&ss[i]=='p') now++;
        if(now==4&&ss[i]=='c') now++;
    }
    if(now==5) cout<<pos<<endl;
    else if(now==4) cout<<'c'<<endl;
    else if(now==3) cout<<'p'<<endl;
    else if(now==2) cout<<'c'<<endl;
    else cout<<'s'<<endl;
}
```

Problem C. 隔世

题意：

给定两个数 a, b ，对 a 做若干次操作，每次操作可以使 a 添加或减少 $[1, 10]$ 的数值。求最少要几次才能使 a 与 b 相等

题解：

贪心。每次能变化越多越好，显然最多的是 10 。故可以证明，答案是两数作差取绝对值向上取整。(向上取整是因为小数点需要额外走一次)

可以使用 $ceil$ 函数或是 if 判断作差绝对值求余 10 是否为 0 ，分别输出

易错点：

1. 没有向上取整

2. 没有取绝对值

3. $ceil$ 函数是双精度函数，必须将里面的值乘 1.0 ，否则只能拿 10 分

骗分技巧：

1. 尝试输出各个样例答案，发现输出 3 可以拿最高 12 分

2. 猜测后台可能会有小数据，从 0 开始尝试输出小数字，发现输出 1 可以拿最高 13 分

示例代码：

C.隔世.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
signed main()
{
    int n,m;cin>>n>>m;
    int sum=abs(n-m);
    int ans=ceil(sum*1.0/10);
    cout<<ans<<endl;
}
```

Problem D. 念水

题意：

凑数字， 可使用的数字有 a 个 n ， b 个 1 ，能否准确凑到 S

题解：

分类讨论，理清所有的可能情况，问题迎刃而解

1. a 个 n 的数值之和大于等于 S ，则判断 S 对 n 求余的值是否大于等于 b ，如果为真，显然可以，反之凑不到

2. a 个 n 的数值之和小于 S ，则判断 $S-a\times n$ 的值是否大于等于 b ，如果为真，显然可以，反之总数都达不到

易错点：

分类讨论情况不足或错误

骗分技巧：

尝试使用随机数来输出 t 次 "YES" 或 "NO" 。

或者直接输出 t 次 "YES" 或 t 次"NO" 。

思路是对的！但是这题数据没那么弱，所以成功率几乎为 0 ，可以不用尝试了。你将得到以下结果

状态	分数	运行时间	运行内存
Wrong Answer	0	3ms	412 KB
Wrong Answer	0	3ms	432 KB
Wrong Answer	0	2ms	504 KB
Wrong Answer	0	3ms	436 KB
Wrong Answer	0	3ms	504 KB
Wrong Answer	0	3ms	436 KB
Wrong Answer	0	3ms	440 KB

示例代码：

D.念水.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
signed main()
{
    int t;cin >> t;
    while(t--)
    {
        int x,y,n,z;cin>>x>>y>>n>>z;
        if(x*n>z) z%=n;
        else z-=x*n;
        if(y>=z) cout<<"YES"<<endl;
        else cout<<"NO"<<endl;
    }
}
```

Problem E. 离殇

题意：

给定 n 个数，每次可以删除任意两个作差绝对值不超过 1 的数中较少的那一个。问能不能删的只剩下一个数

题解：

显然不管删除的是哪一个，最后留下的数字一定是最大值。(这个结论也没用)

关键在于要想删除数，必须能满足删除条件

所以答案呼之欲出了，找出给定序列中的最小值和最大值。只要这个闭区间内所有的数字都在序列中出现，则显然可以一直删除。否则就会被分成若干个块

我们可以一边输入一边用迭代的方法先寻找到最大最小值，并将值入桶。最后从最小值到最大值遍历桶，若其中一个桶中数为 0 (不存在这个数)，显然答案就会变成 "NO" 。若一直有，则 "YES"

易错点：

迭代的易错点就是初始化，最小值要最大化，最大值要最小化

骗分技巧：

看到只需要输出一个 "YES" 或者 "NO" 的题，你应该狂喜。因为，不管输出哪一个，你一定可以获得最少 50 分。所以无脑先输出一下拿大头分

示例代码：

E.离殇.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
int a[1000005],b[1000005];
signed main()
{
    int n,minn=1e9,maxx=0;cin>>n;
    for(int i=1;i<=n;i++)
    {
        cin>>a[i];
        b[a[i]]++;
        minn=min(minn,a[i]),maxx=max(maxx,a[i]);
    }
    bool flag=true;
    for(int i=minn;i<=maxx;i++)
    {
        if(b[i]==0) flag=false;
        if(!flag) break;
    }
    cout<<(flag?"YES":"NO")<<endl;
}
```

Problem F. 以陌 (Easy Version)

题意：

给定一个立体矩阵，矩阵从左到右每一列的方块颜色是红绿蓝依次循环。求右视图能看见三种不同颜色方块各几个。

题解：

hxc 出的都是良心阳间题，下次还填非常简单！

先想想什么方块能被看见。显然是同一行中从右往左看，该方块的高度大于已有最大高度。才有可能被看见。那么搞清楚这个，实现就很简单了

最简单的思路就是建立一个二维数组 $maze$ ，$maze[i][j]$ 表示第 i 行第 j 列的方块高度为 $maze[i][j]$ 。则存图以后，从上到下，从右到左依次遍历每一个值。每一次遍历新的一行，都要将最大值清 0 。至于看到的到底是哪一种颜色的方块，由于循环次数为 3 个颜色一轮，故只需要对当前列数求余 3 ，便可以知道当前列的颜色

分别使用三个变量存储答案，最后输出即可

易错点：

1. 高度有全部都是 0 的，那么啥也看不见

2. 只使用每行最高高度，不管从右到左的迭代中的方块数目

3. 十年做题一场空，不开 $longlong$ 见祖宗

冷静思考，理性分析， int 最大值是 $2^{31}-1$ ，大约是比 2×10^9 多一点，而本题最大值显然是 $n\times a_{ij}$ ，可以达到 5×10^9 。只得到了 30 分左右的同学，你看看你 $longlong$ 开了吗?

骗分技巧：

尝试输出样例答案，失败

猜测后台可能有不存在方块的可能性，输出三行，每行一个 0 ，成功骗到 3 分

示例代码：

F.以陌(Easy Version).cpp

```cpp
#include <bits/stdc++.h>
#define int long long
using namespace std;
int maze [1007][1007];
signed main()
{
    int n,m;cin>>n>>m;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            cin>>maze[i][j];
    int red=0,green=0,blue=0;
    for(int i=1;i<=n;i++)
    {
        int maxx=0;
        for(int j=m;j>=1;j--)
        {
            if(maze[i][j]>maxx)
            {
                if(j%3==1) red+=maze[i][j]-maxx;
                else if(j%3==2) green+=maze[i][j]-maxx;
                else blue+=maze[i][j]-maxx;
                maxx=maze[i][j];
            }
        }
    }
    cout<<red<<endl<<green<<endl<<blue<<endl;
}
```

Problem G. 以陌 (Hard Version)

题意：

题意同上，也是非常简单的题，只是询问改为 q 次，并且视野距离不同

题解：

显然，如果你暴力，时间复杂度是 $O(nm^2)$ 。当然 hxc 太善良了，所以你仍然可以获得一半的分数

所以我们需要在处理的时候，存下每一个不同视野的答案进入答案数组。这是预处理的思想，也是弱化版前缀和 (此处运用前缀和思想则， l 恒等于 1 ， $r=k$ ，答案则为 $sum[r]-sum[l-1]=sum[r]-sum[0]=sum[k]-0=sum[k]$ ，所以你只需要处理好三种颜色的前缀和数组，每次询问到的时候，输出即可)

易错点：

1. 处理答案数组的时候，没有累加前面已有的个数 (见示例代码第 21 行)

2. 十年做题一场空，不开 $longlong$ 见祖宗 ...

... (其他同上 F 题)

骗分技巧：

同 D 题，看到多组输出，基本可以不考虑骗分了

但是， hxc 实在是太善良了。你输出 q 组 $0\ 0\ 0$ ，可以通过两个测试点，获得 2 分呢！

```
while(q--)
{
    int k;cin>>k;
    cout<<0<<" "<<0<<" "<<0<<endl;
}
```

当然，暴力解也是要尝试的，改一改前面一题，提交，你可以喜提 25 分

示例代码：

G.以陌(Hard Version).cpp

```cpp
#include <bits/stdc++.h>
#define int long long
using namespace std;
int red[1005],green[1005],blue[1005],maxx[1005],maze[1007][1007];
signed main()
{
    int n,m;cin>>n>>m;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            cin>>maze[i][j];
    for(int j=m;j>=1;j--)
    {
        for(int i=1;i<=n;i++)
            if(maze[i][j]>maxx[i])
            {
                if(j%3==1) red[m-j+1]+=maze[i][j]-maxx[i];
                else if(j%3==2) green[m-j+1]+=maze[i][j]-maxx[i];
                else blue[m-j+1]+=maze[i][j]-maxx[i];
                maxx[i]=maze[i][j];
            }
        red[m-j+1]+=red[m-j],green[m-j+1]+=green[m-j],blue[m-j+1]+=blue[m-j];
    }
    int q;cin>>q;
    while(q--)
    {
        int k;cin>>k;
        cout<<red[k]<<" "<<green[k]<<" "<<blue[k]<<endl;
    }
}
```

Problem H. 远昼

题意：

给定一个序列，每次可以选择一个区间，使区间里所有元素 -1 或 $+1$ 。求最少要多少次使得区间里所有数相同，并求在最少次数的情况下，序列最终可能的情况有多少种

题解：

简略版：

非常简单，一个技巧贪心小差分

多次区间内修改，时间复杂度会 hin 大~ 故考虑使用我们已学过的算法：差分

了解过差分数组的同学都知道，同时同方法修改某区间内的数字，只需要修改差分数组的 $l, r+1$ 位置的数值即可。 在本题中，即任意选取差分数组两个下标，使其一个 +=1 ，一个 -=1 。

那么第一问的答案呼之欲出，先获得原数组的差分数组，然后分别计算差分数组中负数的和的绝对值以及正数的和的绝对值，将它们分别记作 $sum1, sum2$ 。那么，他们俩的最大值，就是第一问的答案。(两两抵消，但不管差分数组的首项)

而第二问需要一点思维技巧。但是也非常简单。易证得，操作 $min(sum1, sum2)$ 次后，差分数组所有元素中除 0 之外，全部同号。所以共有 $max(sum1, sum2) - min(sum1, sum2)$ （总次数减去已操作次数的闭区间）。由于是闭区间种，所以求差之后要 $+1$ ，最终答案为 $max(sum1, sum2) - min(sum1, sum2) + 1$ 。是不是很简单呢?

详细版：

主要思路：差分

差分的定义：对于一个数列 A，那么它的差分数列 B 定义为：

$$B_i = A_i (2 \le i \le n) - A_{i-1}, B_1 = A_1$$

举个例子：

$$A = \{5, 4, 6, 7, 3\}$$

$$B = \{5, -1, 2, 1, -4\}$$

把原数列的“区间修改”操作改为差分数组的“单点修改”：

若把 A 的区间 $[l, r]$ 加上 d，那么我们只需要对差分数组进行如下两个操作：

- $B_l + = d$
- $B_{r+1} - = d$

对于 l 到 r 之间的其他数，因为 $A_i+=d(l<i<r)$ 且 $A_{i+1}+=d(l<i<r)$，所以 $A_{i+1}-A_i$ 的值不变，即 B_i 不变。有了这个性质，我们稍加分析即可解决此题。

首先，我们令 $B_{n+1}=0$，这是为了利用上述性质时 $r=n$ 的情况。那么，我们的目标就是将所有的 B_i（$i\le 2\le n$）变成 0。由于题目中要求我们每次可以使一个区间加或减去 1，所以每次操作，我们可以任意将 B 数组中的两个元素一个加 1 一个减 1（将 B_i 与 B_j 一个加、一个减 1，相当于在 A 中区间 $[l,r]$ 同时加或减 1）。

此时，问题被我们简化为： 已知一个差分数组 B，每次可以任意将其中两个值一个加 1，一个减 1，直到所有的 B_i（$i\le 2\le n$）为 0，求最优方案的操作次数，以及在操作次数最小的前提下，有多少种操作的方案。

显然，对于每次操作，我们选择 B 中的 2 个元素共可以分为以下 4 中情况（以下的 i，j 都满足 $1<i,j\le n$）：

1. 选择 B_i 和 B_j，即 A 数列中的区间 $[i,j-1]$。
2. 选择 B_1 和 B_j，即 A 数列中的区间 $[1,j-1]$。
3. 选择 B_i 和 B_{n+1}，即 A 数列中的区间 $[i,n]$。
4. 选择 B_1 和 B_{n+1}，即 A 数列中的区间 $[1,n]$。

而第四种操作不会对 B_i（$i\le 2\le n$）产生任何影响，所以是无意义的，不做讨论（更直观的理解是整个 A 数组全部加或减 1，显然是浪费次数的）。

此时，不难想到具体的操作方案：

1. 令 p 为差分序列里正数绝对值的总和，令 q 为差分数列中负数绝对值总和。
2. 此时，不断重复进行操作 1，进行了 $\min(p,q)$ 次后，p 和 q 必定其中有一个是 0，而另一个是 $|p-q|$。
3. 剩下的 $|p-q|$ 就不断进行操作 2 或操作 3 即可。
4. 根据步骤 2 和步骤 3 可得：次数一共是 $\min(p,q)+|p-q|$，化简得 $\max(p,q)$。

第一个问题解决了，那么第二个问题也变得很容易。既然完成所有操作后只有 B_1 不为 0，那么我们只需要讨论 B_1 的可能的数量就是第二个问题的答案。那么 B_1 会有哪些情况呢？很显然是 $B_1\pm 0,B_1\pm 1,B_1\pm 2,\dots,B_1\pm|p-q|$ 共 $|p-q|+1$ 个。

如果觉得文字说明过于抽象，那么我们可以举一个例子试试（数列 A 与 B 就以文章开头的为例）：

前三步 B 的变化：

```
5 0 1 1 -4
5 0 0 1 -3
5 0 0 0 -2
```

此时显然还差两步就完成操作了（除 B_1 外全为 0），因此 B_1 的值可能为 5，4 或 3，即共有 3 种方案。

易错点：

1. 第二问没有注意是闭区间，没有 $+1$

2. 差分数组处理失误，没有从第二项开始处理

骗分技巧：

尝试输出样例，gg

想到有可能出现已经排好的情况，那么答案分别是 0 和 1 。尝试输出。喜提 7 分

胡乱尝试各种情况，发现答案是 $n-1$ 和 n 的情况不少。尝试输出。喜提 20 分

发现输出 0 和 1 的时候，能过前三个点，输出 $n-1$ 和 n 的时候，过 $1,2,14,15,16$ 测试点。

综上可以推断，第一第二个测试点 $n=1$ ，而第三个不是，又看到了这个：

$$\text{对于 } 10\% \text{ 的数据，} 1 \le n \le 10, 1 \le a_i \le 2^3\text{。}$$

胡乱猜测一手第三个是小数据

想办法把第三个拿下！最终写出了离奇的骗分代码：

```cpp
#include <bits/stdc++.h>
#define int long long
using namespace std;
int a[1000005],b[1000005];
signed main(){
    int n;cin>>n;
    if(n<=10) cout<<0<<endl<<1<<endl;
    else cout<<n-1<<endl<<n<<endl;
}
```

最后喜提 22 分

示例代码：

H.远昼.cpp

```cpp
#include <bits/stdc++.h>
#define int long long
using namespace std;
int a[1000005],b[1000005];
signed main()
{
    int n,maxx=-3e9,minn=3e9,sum1=0,sum2=0;cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    for(int i=2;i<=n;i++)
    {
        b[i]=a[i]-a[i-1];
        if(b[i]<0) sum1-=b[i];
        else sum2+=b[i];
    }
    cout<<max(sum1,sum2)<<endl<<max(sum1,sum2)-min(sum1,sum2)+1<<endl;
}
```

当然，你可以稍微缩短一点点行数，如下：

H.远昼(压).cpp

```cpp
#include <bits/stdc++.h>
#define int long long
using namespace std;
int a[1000005],b[1000005];
signed main(){
    int n,maxx=-3e9,minn=3e9,sum1=0,sum2=0;cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    for(int i=2;i<=n;i++) ((a[i]-a[i-1]<0)?sum1-=(a[i]-a[i-1]):sum2+=(a[i]-a[i-1]));
    cout<<max(sum1,sum2)<<endl<<max(sum1,sum2)-min(sum1,sum2)+1<<endl;
}
```

主函数只有 4 行，一眼顶针鉴定为**题

总结

整体难度：

整体难度不难，涉及算法思想的仅有 150 分。而且你可以暴力 $+$ 骗，获得其中的 44 分

分档：

A-D ： 简单题

E-G：中等题

H：中难题

建议：

周一班上的同学补题最后总分不低于 500

周二周三班上的同学补题最后总分不低于 600

周四班上的同学补题最后做出所有题，总分达到 700

骗分：

综上，除了D每道题都能骗分，而D本身是简单题。所以有存在某题零分的同学，应该累计骗分经验哦~

补题方法：

进入 OJ 网址：www.scpc.fun

选择上方导航栏：题目

在右侧标签筛选，点击最下面的 tag ： 每周编程两小时 (黑色tag)

即可做题。

你也可以直接搜索对应题号：$3606 - 3613$ 。

祝大家拥有一个愉快的暑假，也恭喜大家顺利完成本学期的学习任务。希望老师和其他助教能对你的思维能力、代码水平有所帮助和提升。

(hxc 也希望大家能够不要暴打出题人，享受题目中的诗句和故事，ლ(´ ·ᴗ· `)比心)

下次见~