# 问题 A: 密码锁

**题目描述**


Scrooge McDuck 将他最珍贵的积蓄放在带组合锁的家庭保险箱里。每次他想把他赚到的宝藏放在那里时，他都必须打开锁。

组合锁由 n 个旋转磁盘表示，上面写着从 0 到 9 的数字。Scrooge McDuck 必须转动一些磁盘，以便磁盘上数字的组合形成一个秘密组合。在一次移动中，他可以向前或向后旋转一个磁盘。特别是，一次移动，他就可以从数字 0 到数字 9，反之亦然。他至少需要多少行动？

**输入格式**

第一行包含单个整数 n（1≤n≤1000）组合锁上的磁盘数量。

第二行包含一串 n 位数字-磁盘的原始状态。

第三行包含一串 n 位数字——密码。

**输出格式**

打印单个整数-Scrooge McDuck 打开锁所需的最小动作数。

Examples

Input

5

82195

64723

Output

13

Note

样例中共进行 13 次移动

磁盘 1: 8->7->6

磁盘 2: 2->3->4

磁盘 3: 1->0->9->8->7

磁盘 4: 9->0->1->2

磁盘 5: 5->4->3

题解：

按题意直接暴力转即可

代码如下：

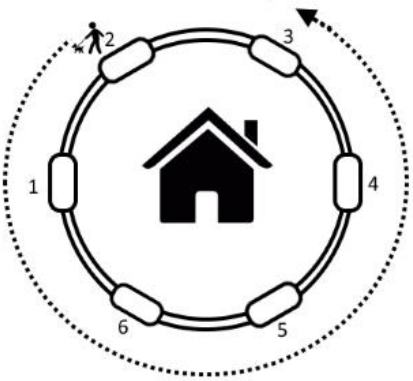
```
#include <bits/stdc++.h>
using namespace std;
int ans;
int main()
{
    string s1, s2;
    int n;
    cin >> n;
    cin >> s1 >> s2;
    long long ans = 0;
    for (int i = 0; i < n; i++)
    {
        int k = s1[i] - '0';
        int m = s2[i] - '0';
        int ans1 = 0;
        int ans2 = 0;
        while (k != m)
        {
            k++;
            k %= 10;
            ans1++;
        }
        k = s1[i] - '0';
        while (k != m)
        {
            k--;
            if (k < 0)
                k += 10;
            ans2++;
        }
        ans += min(ans1, ans2);
    }
    cout << ans << endl;
}
```

# 问题 B: 圆形房子

**题目描述**

废龙小朋友居住在一间圆形的房屋里。这所房屋的所有门都被用数字 1~n 标号了。门口 n 与门口 1 是连起来的。

现在给你 3 个数 n,a,b，废龙的房子有 n 个门口，他打算从门口 a 开始走过 b 个门口，问你最后废龙会站在哪个门口前面。若 b 为负数，就是指废龙要倒车，即，他要倒着走。




一共有 6 个门口，从 2 号门口开始，走-5 个门口，最后停在了 3 号门口

**输入格式**

输入一行包含三个空格分隔的整数 n、a 和 b（1≤n≤100，1≤a≤n，-100≤b≤100）——分别是废龙所在地的入口数量、入口数量和行走长度。

**输出格式**

输入行走结束时将要到达的入口的编号。

**input**

6 2 -5

**output**

3

**input**

5 1 3

**output**

4

**input**

3 2 7

**output**

3

题解：

用一个变量 x 记录答案即可，但是需要注意一些特殊情况。

代码如下：

```
#include <iostream>
using namespace std;
int main()
{
    int n, a, b;
    cin >> n >> a >> b;
    int x = (a + b) % n; // x 是当前位置
    // 在不考虑圆圈的情况下，a+b 是位置，在圆圈内时模 n 即可
    if (x < 0)  // 负数是要特判的，因为当前 x 为负数，显然不能作为答案输出
        x += n; // 因为是模 n 的结果，所以不会超过-n，加 n 即可变为正
    if (x == 0) // 由于余数可能为 0，所以在此题中应直接输出 n
        cout << n << endl;
    else // 否则正常输出 x
        cout << x << endl;
    return 0;
}
```

```
#include<bits/stdc++.h>
using namespace std;
int n,a,b;
int main(){
    cin>>n>>a>>b;
    while(b<0||b>n)//当 b 不在 0-n 的范围里时，持续操作直到范围内
    {
        //超过 n 的所有都是多走的
        if(b<0)b+=n;//比 n 小就加
        else b-=n;//比 n 大就减
    }
    a+=b;//往前走 b 步
    if(a>n)a-=n;//因为加上 b 也可能超过 n，所以减去
    cout<<a;
    return 0;
}
```

# 问题 C: : zz 的游戏

题目描述
zz 和 kk 正在玩一款游戏, 初始，zz 可以设定一个长度为 n 字符串 AB 串（仅有 A 和 B），kk 可以选择某个前缀或后缀进行一次翻转（指原本的 A 变为 B，B 变为 A），当然也可以不选。最后每个玩家所获得的力量：
若字符串中第 i 个位置为 A，则给予 zz Pi 点能量；若为 B，则给予 kk Pi 点能量。求 kk 进行翻转操作后所能得到的最大能量。
输入格式
第一行包含整数 n （1≤n≤5·10^5） –那个字符。
第二行包含 n 个整数 pi （1≤pi≤10^9） – 第 i 字符的强度。
第三行包含 n 个字符 A 或 B 。
输出格式
打印唯一的整数 a – kk 可以达到的最大强度。
Examples
Input
5
1 2 3 4 5
ABABA
Output
11
Input
5
1 2 3 4 5
AAAAA
Output
15
Input
1
1
B
Output
1
Note
在第一个示例中，kk 应该翻转长度 1 的后缀。
在第二个示例中，kk 应该翻转长度为 5 的前缀或后缀（此处相同）。
在第三个示例中，kk 应该什么都不做。

先计算原本数组（未改动）可以获得的能量之和。
然后用两个不同的变量储存。
接着分别枚举前缀和后缀，
最后计算改动后的最大值并输出。

```
#include<iostream>
using namespace std;
long long a[500005],b[500005],x,y,n,ans;
char s;
int main(){
```

```
    cin>>n;//输入 n
    for(int i=0;i<n;i++)cin>>a[i];//输入 a[i]
    for(int i=0;i<n;i++){
        cin>>s;//一个一个读取字符
        b[i]=s=='A'?1:0;//判断是否为'A'
        ans+=b[i]*a[i];//读取总和
    }
    x=y=ans;//赋值
    for(int i=0;i<n;i++){//枚举前缀
        if(!(b[i]-1))x-=a[i];// 如果 b[i]=1，说明当前字符为 A，让 x 去减这一位的权值
        else x+=a[i];//如果 b[i]=0，说明当前字符为 B，让 x 去加这一位的权值
        if(x>ans)ans=x;//轮换，修改最大值
    }
    for(int i=n-1;i>0;i--){//枚举后缀
        if(!(b[i]-1))y-=a[i];//如果 b[i]=1，说明当前字符为 A，让 y 去减这一位的权值
        else y+=a[i];//如果 b[i]=0，说明当前字符为 B，让 y 去加这一位的权值
        if(y>ans)ans=y;//轮换，修改最大值
    }
    cout<<ans<<endl;//输出进行翻转操作后所能得到的最大能量。
    return 0;//byebye
}
```

# 问题 D: 电子表格

题目描述

在课程中，zz 使用一个著名的电子表格 excel 学习如何编辑表格。

现在 zz 有一个填满整数的表格，该表由 n 行和 m 列组成。用 $a_{i,j}$，表示位于第 i 行和第 j 列的整数。我们说，对于从 1 到 n-1 的所有 i，如果 $a_{i,j} \le a_{i+1,j}$，则表在 j 列中按非降序排序。

老师给 zz 布置了 k 个 k 次询问。对问如果只保留矩阵的第 L~R 行，矩阵中是否存在不下降的一列。

即是否存在这样的 j， 对于从 l 到 r-1 的所有 i，使 $a_{i,j} \le a_{i+1,j}$。

请你帮忙！

输入格式

第一行为两个数 n,m($1 \le n*m \le 1000000$)

下面 n 行，每行 m 个数($1 \le a_{i,j} \le 109$).

下一行为 k 的值($1 \le k \le 100000$)

接下来 k 行，每行两个数 l,r( ($1 \le l_i \le r_i \le n$))

输出格式

如果第 l 行到 r 行至少有一列是非递减的，则输出 Yes，否则输出 No.

输入样例：

Input

5 4

1 2 3 5
3 1 3 2
4 5 2 3
5 5 3 2
4 4 3 4
6
1 1
2 5
4 5
3 5
1 3
1 5
Output
Yes
No
Yes
Yes
Yes
No

注意： 在示例中，整个表没有在任何列中进行排序。但是，第 1-3 行在第 1 列中排序，而第 4-5 行在第 3 列中排序。
输入样例 复制
5 4
1 2 3 5
3 1 3 2
4 5 2 3
5 5 3 2
4 4 3 4
6
1 1
2 5
4 5
3 5
1 3
1 5
输出样例 复制
Yes
No
Yes
Yes
Yes
No

我们要求出 L 至 R 行中的不下降序列。那我们只需要记录每个数在每一列中上一个能与其构成逆序对的数的位置，通俗点说，就是记录上一个比这个数大的数的位置，然后获得这一行最大延伸长度（定义最大延伸长度为：能构成的最长不下降序列长度），复杂度为 O(mn)。对于每组询问，我们只需要判断 R 这行的延伸长度是否能达到 L 这一行，即可进行 O(1) 查询。
最大复杂度为 O(mn)，在 $10^5$ 内合法。
还要注意一下，不能开二维数组，会爆。
设计数组 reach[]，记录上一个比这个数大的数；
设计数组 lastreach[]，记录上一行的数的 reach[]；
设计数组 last[]，记录上一行的数；
设计数组 linereach[]，记录该行最大延伸长度能到达的点；
不要忘记用 memset 初始化 lastreach[] 和 last[]；

```
#include<iostream>
#include<cstring>
using namespace std;
#define MAXN 100005
int n,m,k,lastreach[MAXN],reach[MAXN],linereach[MAXN],last[MAXN];
int main(){
    cin>>n>>m;
    memset(linereach,0x3f,sizeof(linereach));//初始化
    memset(last,0x3f,sizeof(last));
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            int x;
            cin>>x;
            if(x<last[j]){//如果比上一行的数小，不能与上面一行的数构成不下降
序列
                reach[j]=i;
            }
            else{//否则，可以与上一行的数构成不下降序列，延续上一行的信息
                reach[j]=lastreach[j];
            }
            lastreach[j]=reach[j];//传递信息
            last[j]=x;
            linereach[i]=min(linereach[i],reach[j]);//记录 linereach，即能
到达的最小的行
        }
    }
    cin>>k;
    while(k--){
        int l,r;
        cin>>l>>r;
        cout<<(linereach[r]<=l?"Yes":"No")<<endl;
    }
    return 0;
}
```