## PREV-52 小数第n位

### 1 原题目

资源限制 <http://lx.lanqiao.cn/problem.page?gpid=T456>

时间限制：1.0s   内存限制：256.0MB

问题描述

　　我们知道，整数做除法时，有时得到有限小数，有时得到无限循环小数。  
　　如果我们把有限小数的末尾加上无限多个0，它们就有了统一的形式。  
　　本题的任务是：在上面的约定下，求整数除法小数点后的第n位开始的3位数。

输入格式

　　一行三个整数：a b n，用空格分开。a是被除数，b是除数，n是所求的小数后位置（0<a,b,n<1000000000）

输出格式

　　一行3位数字，表示：a除以b，小数后第n位开始的3位数字。

样例输入

1 8 1

样例输出

125

样例输入

1 8 3

样例输出

500

样例输入

282866 999000 6

样例输出

914

### 2 基础知识

#### 2.1数据结构

无。

#### 2.2算法

快速幂算法（解法一用到）

### 3 解法

#### 3.1 解法1（C++已过）

参考链接：<https://blog.csdn.net/A_ACM/article/details/88304399>

本题主要是此公式的运用：x/d%m = x%(d\*m)/d

本人对此公式推到的理解：

9 % 7 = 2； 9与7同时乘上相同的倍数 9\*2 % 7\*2 = 4

所以如果想让9\*2 % 7\*2的余数等于9 % 7的余数，就需要再除以那个倍数，

即9\*2 % 7\*2 / 2 = 2，所以可推导出x/d%m = x%(d\*m)/d

本人代码：

#include <iostream>

#include <cstring>

#include <cstdio>

#include <cstdlib>

#include <cmath>

#define ll long long

using namespace std;

ll qpow(ll a,ll n,ll mod)//利用快速幂求a^n % mod

{

ll res = 1;

while(n > 0) {

if(n % 2 == 1)

res = (res \* a) % mod;

a = (a \* a) % mod;

n = n / 2;

}

return res % mod;

}

//res = [a/b\*10^(n+2)]%1000

//因为公式：x/d%m = x%(d\*m)/d

//所以res = [a\*10^(n+2)]%(b\*1000)/b

int main() {

ll a,b,n;

cin>>a>>b>>n;

ll b1 = b \* 1000;

ll temp = qpow(10,n + 2,b1);

ll res = ((a % b1 \* temp) % b1) / b;

cout<<res<<endl;

return 0;

}

## PREV-36 包子凑数

### 1 原题目

资源限制 <http://lx.lanqiao.cn/problem.page?gpid=T440>

时间限制：1.0s   内存限制：256.0MB

问题描述

　　小明几乎每天早晨都会在一家包子铺吃早餐。他发现这家包子铺有N种蒸笼，其中第i种蒸笼恰好能放Ai个包子。每种蒸笼都有非常多笼，可以认为是无限笼。  
　　每当有顾客想买X个包子，卖包子的大叔就会迅速选出若干笼包子来，使得这若干笼中恰好一共有X个包子。比如一共有3种蒸笼，分别能放3、4和5个包子。当顾客想买11个包子时，大叔就会选2笼3个的再加1笼5个的（也可能选出1笼3个的再加2笼4个的）。  
　　当然有时包子大叔无论如何也凑不出顾客想买的数量。比如一共有3种蒸笼，分别能放4、5和6个包子。而顾客想买7个包子时，大叔就凑不出来了。  
　　小明想知道一共有多少种数目是包子大叔凑不出来的。

输入格式

　　第一行包含一个整数N。(1 <= N <= 100)  
　　以下N行每行包含一个整数Ai。(1 <= Ai <= 100)

输出格式

　　一个整数代表答案。如果凑不出的数目有无限多个，输出INF。

样例输入

2  
4  
5

样例输出

6

样例输入

2  
4  
6

样例输出

INF

样例说明

　　对于样例1，凑不出的数目包括：1, 2, 3, 6, 7, 11。  
　　对于样例2，所有奇数都凑不出来，所以有无限多个。

数据规模和约定

　　峰值内存消耗（含虚拟机） < 256M  
　　CPU消耗 < 1000ms  
　　请严格按要求输出，不要画蛇添足地打印类似：“请您输入...” 的多余内容。  
　　注意：  
　　main函数需要返回0;  
　　只使用ANSI C/ANSI C++ 标准;  
　　不要调用依赖于编译环境或操作系统的特殊函数。  
　　所有依赖的函数必须明确地在源文件中 #include <xxx>  
　　不能通过工程设置而省略常用头文件。  
　　提交程序时，注意选择所期望的语言类型和编译器类型。

### 2 基础知识

#### 2.1数据结构

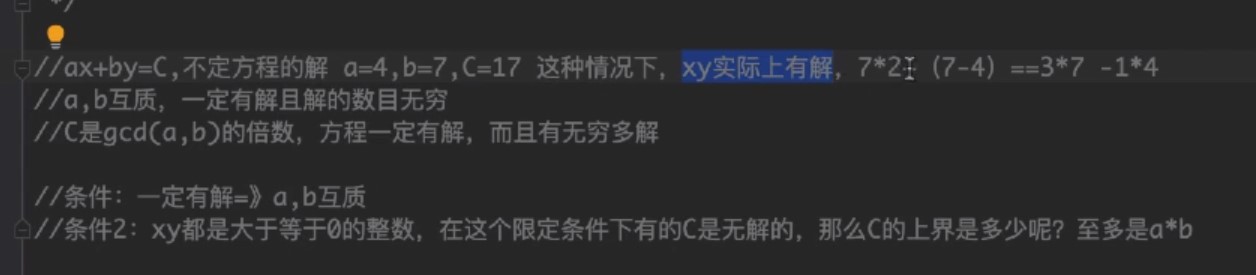
无。

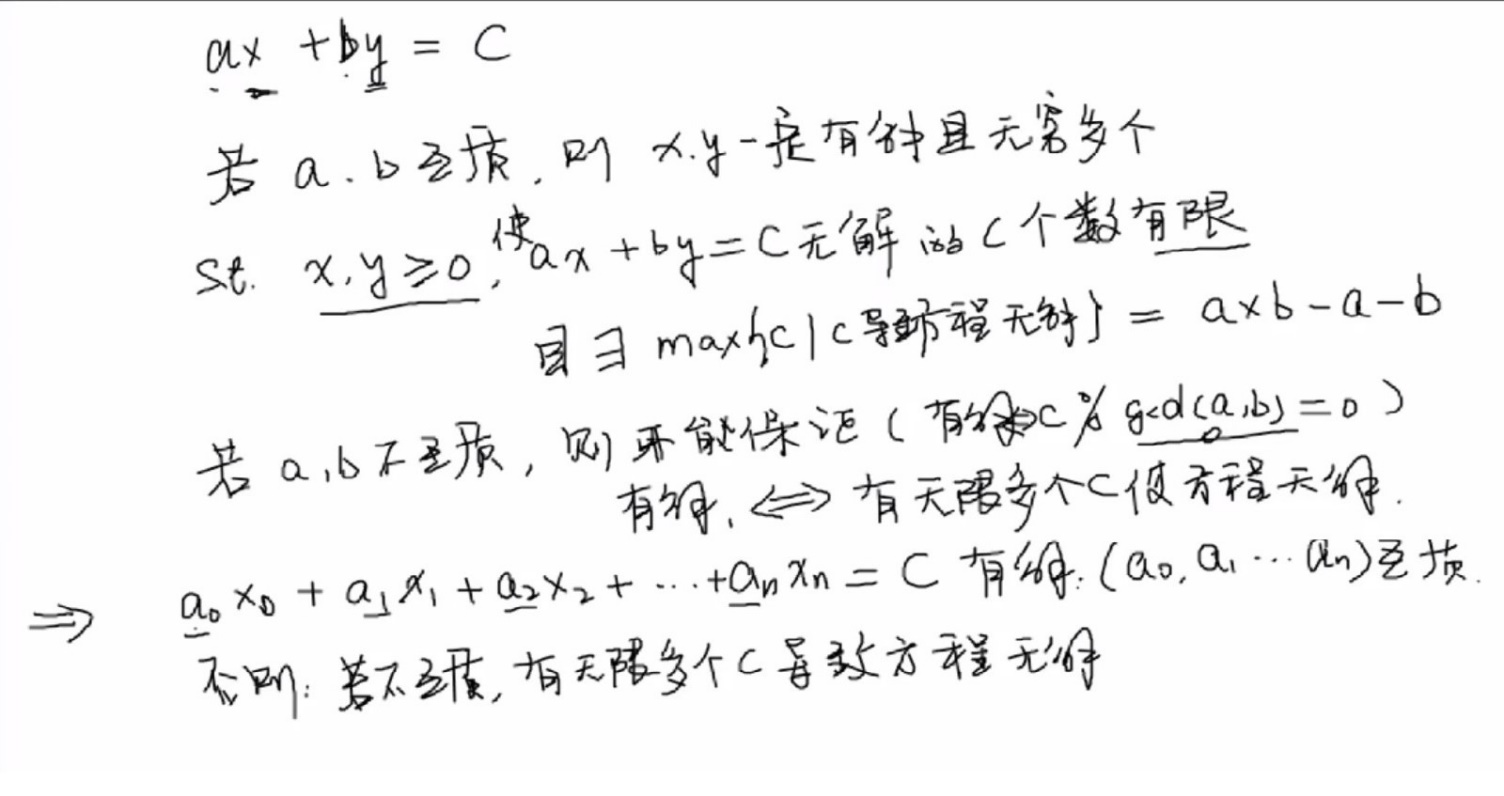
#### 2.2算法

欧几里得算法（解法一用到）

### 3 解法

#### 3.1 解法1（C++已过）





当凑不出来的数是已知的数目时，我们可以利用标记数组来求出来有几种情况是我们凑不出来的。包子数0肯定是可以凑出来的，所以map[0]标记为1，我们遍历包子的笼数n，每一笼的包子加上当前可以凑出的包子数都是可以凑出来的。

本人代码：

#include <iostream>

#include<algorithm>

#include <cstring>

#include <cstdio>

#include <cstdlib>

#include <cmath>

using namespace std;

int gcd(int n,int m)//辗转相除法求最大公约数

{

int temp;

while(m > 0)

{

temp = n % m;

n = m;

m = temp;

}

return n;

}

int main() {

int arr[105] = {0},map[10005] = {0},n,i,j;

cin>>n;

for(i = 0; i < n; i++)

cin>>arr[i];

int g = gcd(arr[0],arr[1]);

for(i = 2; i < n; i++)

g = gcd(g,arr[i]);//求这n个数的最大公约数

if(g != 1)//最大公约数不是1

cout<<"INF"<<endl;//说明有无限个凑不出来

else {

map[0] = 1;//标记为1代表可以凑出来

for(i = 0; i < n; i++) {

for(j = 0; j + arr[i] < 10005; j++) {

if(map[j] == 1)//如果当前包子数可以凑出来

map[j + arr[i]] = 1;//那么再加上每种蒸笼里的包子数也可以凑出来

}

}

int num = 0;

for(i = 0; i < 10000; i++)//统计凑不出来的个数（最多100种蒸笼，每笼最多100个包子）

if(map[i] == 0)

num++;

cout<<num<<endl;

}

return 0;

}

## PREV-37 分巧克力

### 1 原题目

资源限制 <http://lx.lanqiao.cn/problem.page?gpid=T441>

时间限制：1.0s   内存限制：256.0MB

问题描述

　　儿童节那天有K位小朋友到小明家做客。小明拿出了珍藏的巧克力招待小朋友们。  
　　小明一共有N块巧克力，其中第i块是Hi x Wi的方格组成的长方形。  
　　为了公平起见，小明需要从这 N 块巧克力中切出K块巧克力分给小朋友们。切出的巧克力需要满足：  
　　1. 形状是正方形，边长是整数  
　　2. 大小相同  
　　例如一块6x5的巧克力可以切出6块2x2的巧克力或者2块3x3的巧克力。  
　　当然小朋友们都希望得到的巧克力尽可能大，你能帮小Hi计算出最大的边长是多少么？

输入格式

　　第一行包含两个整数N和K。(1 <= N, K <= 100000)  
　　以下N行每行包含两个整数Hi和Wi。(1 <= Hi, Wi <= 100000)  
　　输入保证每位小朋友至少能获得一块1x1的巧克力。

输出格式

　　输出切出的正方形巧克力最大可能的边长。

样例输入

2 10  
6 5  
5 6

样例输出

2

数据规模和约定

　　峰值内存消耗（含虚拟机） < 256M  
　　CPU消耗 < 1000ms  
　　请严格按要求输出，不要画蛇添足地打印类似：“请您输入...” 的多余内容。  
　　注意：  
　　main函数需要返回0;  
　　只使用ANSI C/ANSI C++ 标准;  
　　不要调用依赖于编译环境或操作系统的特殊函数。  
　　所有依赖的函数必须明确地在源文件中 #include <xxx>  
　　不能通过工程设置而省略常用头文件。  
　　提交程序时，注意选择所期望的语言类型和编译器类型。

### 2 基础知识

#### 2.1数据结构

无。

#### 2.2算法

二分法（解法二用到）

### 3 解法

#### 3.1 解法1（C++已过）

暴力遍历的思想，一共n块大巧克力，k个小朋友，巧克力边长为1到100000。

所以我们就让所有大巧克力分别分成边长为i的小巧克力（i的范围是1到100000）

遍历的终止条件就是首次分成边长为i的小巧克力不够小朋友分，就跳出遍历。（此时的i不够分，那分成更大的边长的小巧克力就更不够分）。此时的i-1就是答案（题目求最大边长的小巧克力）

本人代码：

#include <iostream>

#include<algorithm>

#include <cstring>

#include <cstdio>

#include <cstdlib>

#include <cmath>

using namespace std;

int n,k,H[100005],W[100005];

int main() {

cin>>n>>k;

int i,j;

for(i = 1; i <= n; i++)

cin>>H[i]>>W[i];

int num;//记录巧克力被分成的块数

for(i = 1; i <= 100000; i++) {//每块巧克力分成的大小的边界是1到100000

num = 0;

for(j = 1; j <= n; j++) {//一共n块巧克力

num += (int)(H[j] / i) \* (int)(W[j] / i);//计算每块巧克力如果被分成边长为i的巧克力所分成的块数并加进去

if(num >= k)//如果当前的块数大于小朋友数就跳出

break;

}

if(num < k)//如果块数小于了小朋友数就找到了答案，因为每块巧克力分成当前i的大小的巧克力不够小朋友分，那么分成更大边长的一定也不够分

break;

}

cout<<i - 1<<endl;//输出结果

return 0;

}

#### 3.2 解法2（C++已过）

二分法，左边界为1（代表一定够小朋友分），右边界为100001（超出巧克力上限，代表一定不够分），然后我们只需一直判断中间值mid（mid够分就给左，不够分给右）直到left < right – 1（因为左右边界很明确一个够分一个不够分，所以当左右边界相邻时就是循环终止的条件），因为题目求最大的小巧克力边长，所以最后的left就是结果。

本人代码：

#include <iostream>

#include<algorithm>

#include <cstring>

#include <cstdio>

#include <cstdlib>

#include <cmath>

using namespace std;

int n,k,H[100005],W[100005];

bool judge(int m) {

int num = 0;//记录巧克力被分成的块数

for(int j = 1; j <= n; j++) {//一共n大块巧克力

num += (int)(H[j] / m) \* (int)(W[j] / m);//计算每块巧克力如果被分成边长为m的巧克力所分成的块数并加进去

if(num >= k)//如果当前的块数大于小朋友数就返回真

return true;

}

return false;

}

int main() {

cin>>n>>k;

int i;

for(i = 1; i <= n; i++)

cin>>H[i]>>W[i];

int left = 1,right = 100001,mid;//二分的左右边界和中间值（默认左边界是可以分到的，右边界是无法分到的边长）

while(left < right - 1) {

mid = left + (right - left) / 2;//中间值

if(judge(mid))//mid可以分到

left = mid;//给左边界

else//mid分不到

right = mid;//给右边界

}

cout<<left<<endl;//输出左边界

return 0;

}

## PREV-40 k倍区间

### 1 原题目

资源限制 <http://lx.lanqiao.cn/problem.page?gpid=T444>

时间限制：2.0s   内存限制：256.0MB

问题描述

　　给定一个长度为N的数列，A1, A2, ... AN，如果其中一段连续的子序列Ai, Ai+1, ... Aj(i <= j)之和是K的倍数，我们就称这个区间[i, j]是K倍区间。  
　　你能求出数列中总共有多少个K倍区间吗？

输入格式

　　第一行包含两个整数N和K。(1 <= N, K <= 100000)  
　　以下N行每行包含一个整数Ai。(1 <= Ai <= 100000)

输出格式

　　输出一个整数，代表K倍区间的数目。

样例输入

5 2  
1  
2  
3  
4  
5

样例输出

6

数据规模和约定

　　峰值内存消耗（含虚拟机） < 256M  
　　CPU消耗 < 2000ms  
　　请严格按要求输出，不要画蛇添足地打印类似：“请您输入...” 的多余内容。  
　　注意：  
　　main函数需要返回0;  
　　只使用ANSI C/ANSI C++ 标准;  
　　不要调用依赖于编译环境或操作系统的特殊函数。  
　　所有依赖的函数必须明确地在源文件中 #include <xxx>  
　　不能通过工程设置而省略常用头文件。  
　　提交程序时，注意选择所期望的语言类型和编译器类型。

### 2 基础知识

#### 2.1数据结构

无。

#### 2.2算法

无。

### 3 解法

#### 3.1 解法1（C++已过）

做法：首先统计前缀和sum[i] 表示A1+A2+…+Ai.所以对于任意一段区间[l,r]的和就是sum[r]-sum[l-1].如果要保证这个区间和为K倍数就是：(sum[r]-sum[l-1])%k == 0.变形后就是：sum[r]%k==sum[l-1]%k，所以我们计算前缀和的时候顺带模K，然后统计前缀和中相同的数据就行了。

参考链接：<https://blog.csdn.net/qq_34594236/article/details/70845223>

本人代码：

#include <iostream>

#include<algorithm>

#include <cstring>

#include <cstdio>

#include <cstdlib>

#include <cmath>

typedef long long ll;

using namespace std;

int main() {

ll arr[100005] = {0};//存储题目给的数列

ll sum[100005] = {0};//存储前i个元素的和对k求余后的结果

ll cnt[100005] = {0};//记录不同余数的个数

ll n,k,res = 0;

cin>>n>>k;

//首先统计前缀和sum[i] 表示A1+A2+…+Ai.

//对于任意一段区间[l,r]的和就是sum[r]-sum[l-1].

//如果要保证这个区间和为K倍数就是：(sum[r]-sum[l-1])%k == 0.变形后就是：sum[r]%k==sum[l-1]%k

for(int i = 1; i <= n; i++) {

cin>>arr[i];

sum[i] = (sum[i - 1] + arr[i]) % k;

res = res + cnt[sum[i]];

cnt[sum[i]]++;

}

cout<<res + cnt[0]<<endl;//最后结果要加上余数为0的情况（自己本身就是一个区间）

return 0;

}

## PREV-27 蚂蚁感冒

### 1 原题目

资源限制

时间限制：1.0s   内存限制：256.0MB

问题描述

　　长100厘米的细长直杆子上有n只蚂蚁。它们的头有的朝左，有的朝右。  
　　每只蚂蚁都只能沿着杆子向前爬，速度是1厘米/秒。  
　　当两只蚂蚁碰面时，它们会同时掉头往相反的方向爬行。  
　　这些蚂蚁中，有1只蚂蚁感冒了。并且在和其它蚂蚁碰面时，会把感冒传染给碰到的蚂蚁。  
　　请你计算，当所有蚂蚁都爬离杆子时，有多少只蚂蚁患上了感冒。

输入格式

　　第一行输入一个整数n (1 < n < 50), 表示蚂蚁的总数。  
　　接着的一行是n个用空格分开的整数 Xi (-100 < Xi < 100), Xi的绝对值，表示蚂蚁离开杆子左边端点的距离。正值表示头朝右，负值表示头朝左，数据中不会出现0值，也不会出现两只蚂蚁占用同一位置。其中，第一个数据代表的蚂蚁感冒了。

输出格式

　　要求输出1个整数，表示最后感冒蚂蚁的数目。

样例输入

3  
5 -2 8

样例输出

1

样例输入

5  
-10 8 -20 12 25

样例输出

3

### 2 基础知识

#### 2.1数据结构

无。

#### 2.2算法

无。

### 3 解法

#### 3.1 解法1（C++已过）

蚂蚁相遇并转向前进可以直接看成蚂蚁穿过直接继续前进。

问题解为1的情况是：得病蚂蚁所朝向得那个方向没有和他对着走的（假如朝左走，那么在它左边没有蚂蚁朝右走，假如朝右走则亦然）。

其余问题解的求法：因为可以看成直接穿过，所以第一只蚂蚁不管方向朝哪它右边的蚂蚁只要向左走就可能碰撞感染(解为1的特殊情况除外)，同样，第一只蚂蚁左边的蚂蚁只要朝右边走也可能被感染，这样就很容易得到这样的公式

ans = 左边蚂蚁向右走的数量+右边蚂蚁向左走的数量+第一只蚂蚁本身=right+left+1

参考链接：<https://blog.csdn.net/computer_liuyun/article/details/23350077>

本人代码：

#include <iostream>

#include<algorithm>

#include <cstring>

#include <cstdio>

#include <cstdlib>

#include <cmath>

using namespace std;

int main() {

int arr[55] = {0};//存储蚂蚁初始位置信息

int n,i;

cin>>n;

for(i = 0; i < n; i++)

cin>>arr[i];

int left = 0,right = 0;

for(i = 1; i < n; i++) {

if(arr[i] > 0 && abs(arr[i]) < abs(arr[0]))//找到该蚂蚁左边且向右走的

right++;

if(arr[i] < 0 && abs(arr[i]) > abs(arr[0]))//找到该蚂蚁右边且向左走的

left++;

}

if((arr[0] < 0 && right == 0) || (arr[i] > 0 && left == 0))//不会有其他蚂蚁染病的情况

cout<<1<<endl;

else

cout<<left + right + 1<<endl;

return 0;

}

## PREV-28 地宫取宝

### 1 原题目

资源限制 <http://lx.lanqiao.cn/problem.page?gpid=T120>

时间限制：1.0s   内存限制：256.0MB

问题描述

　　X 国王有一个地宫宝库。是 n x m 个格子的矩阵。每个格子放一件宝贝。每个宝贝贴着价值标签。  
　　地宫的入口在左上角，出口在右下角。  
　　小明被带到地宫的入口，国王要求他只能向右或向下行走。  
　　走过某个格子时，如果那个格子中的宝贝价值比小明手中任意宝贝价值都大，小明就可以拿起它（当然，也可以不拿）。  
　　当小明走到出口时，如果他手中的宝贝恰好是k件，则这些宝贝就可以送给小明。  
　　请你帮小明算一算，在给定的局面下，他有多少种不同的行动方案能获得这k件宝贝。

输入格式

　　输入一行3个整数，用空格分开：n m k (1<=n,m<=50, 1<=k<=12)  
　　接下来有 n 行数据，每行有 m 个整数 Ci (0<=Ci<=12)代表这个格子上的宝物的价值

输出格式

　　要求输出一个整数，表示正好取k个宝贝的行动方案数。该数字可能很大，输出它对 1000000007 取模的结果。

样例输入

2 2 2  
1 2  
2 1

样例输出

2

样例输入

2 3 2  
1 2 3  
2 1 5

样例输出

14

### 2 基础知识

#### 2.1数据结构

无。

#### 2.2算法

记忆化搜索（解法一用到）

### 3 解法

#### 3.1 解法1（C++已过）

详见注释和参考链接。

参考链接：<https://blog.csdn.net/u014665013/article/details/45201479>

本人代码：

#include <iostream>

#include<algorithm>

#include <cstring>

#include <cstdio>

#include <cstdlib>

#include <cmath>

using namespace std;

#define N 1000000007

int n,m,k;

int map[55][55];

int dp[55][55][20][20];

//x，y表示当前坐标，num表示当前拿到的宝物数，max表示当前所有宝物中的最大价值

//如dp[3][4][5][6]=7 即当在map[3][4]且身上有5件宝物 宝物的最大值是6 到达终点有7种路径

int dfs(int x,int y,int num,int max) {

//判断这个状态是否已经走过，如果走过就直接用记录的数值计算，因为宝物有可能为0所以定义max时用最小值-1 这就导致无法作为下标使用

//实际上如果测试数据中宝物价值没有0，将所有的+1去掉也是可以的 这里的话如果去掉肯定是有些数据不对的，不信可以提交试一下，根本过不了

if(dp[x][y][num][max + 1] != -1)

return dp[x][y][num][max + 1];

long long t = 0;

if(x == n - 1 && y == m - 1) {//到达出口

if(num == k || (num == k - 1 && map[x][y] > max))//到达右下角，（1）可能不算右下角的也正好，（2）如果右下角的比当前的大，算上这一个如果正好也可以增加路径

t++;

dp[x][y][num][max + 1] = t;

return dp[x][y][num][max + 1];

}

if(x + 1 < n) {//向下走

if(map[x][y] > max) {

t += dfs(x + 1,y,num + 1,map[x][y]);

t %= N;

}

t += dfs(x + 1,y,num,max);

t %= N;

}

if(y + 1 < m) {//向右走

if(map[x][y] > max) {

t += dfs(x,y + 1,num + 1,map[x][y]);

t %= N;

}

t += dfs(x,y + 1,num,max);

t %= N;

}

dp[x][y][num][max + 1] = t;

return dp[x][y][num][max + 1];

}

int main() {

cin>>n>>m>>k;

int i,j;

for(i = 0; i < n; i++)

for(j = 0; j < m; j++)

cin>>map[i][j];

memset(dp,-1,sizeof(dp));

dfs(0,0,0,-1);

cout<<dp[0][0][0][0]<<endl;

return 0;

}

## PREV-31 小朋友排队

### 1 原题目

资源限制

时间限制：1.0s   内存限制：256.0MB

问题描述

　　n 个小朋友站成一排。现在要把他们按身高从低到高的顺序排列，但是每次只能交换位置相邻的两个小朋友。  
　　每个小朋友都有一个不高兴的程度。开始的时候，所有小朋友的不高兴程度都是0。  
　　如果某个小朋友第一次被要求交换，则他的不高兴程度增加1，如果第二次要求他交换，则他的不高兴程度增加2（即不高兴程度为3），依次类推。当要求某个小朋友第k次交换时，他的不高兴程度增加k。  
　　请问，要让所有小朋友按从低到高排队，他们的不高兴程度之和最小是多少。  
　　如果有两个小朋友身高一样，则他们谁站在谁前面是没有关系的。

输入格式

　　输入的第一行包含一个整数n，表示小朋友的个数。  
　　第二行包含 n 个整数 H1 H2 … Hn，分别表示每个小朋友的身高。

输出格式

　　输出一行，包含一个整数，表示小朋友的不高兴程度和的最小值。

样例输入

3  
3 2 1

样例输出

9

样例说明

　　首先交换身高为3和2的小朋友，再交换身高为3和1的小朋友，再交换身高为2和1的小朋友，每个小朋友的不高兴程度都是3，总和为9。

数据规模和约定

　　对于10%的数据， 1<=n<=10；  
　　对于30%的数据， 1<=n<=1000；  
　　对于50%的数据， 1<=n<=10000；  
　　对于100%的数据，1<=n<=100000，0<=Hi<=1000000。

### 2 基础知识

#### 2.1数据结构

树状数组—单点更新区间查询

#### 2.2算法

无。

### 3 解法

#### 3.1 解法1（C++已过）

逆序对加树状数组的应用。

参考链接：<http://www.aiuxian.com/article/p-2288507.html>

本人代码：

#include <iostream>

#include<algorithm>

#include <cstring>

#include <cstdio>

#include <cstdlib>

#include <cmath>

using namespace std;

int a[100005],N,b[1000005];//a记录小朋友身高，b记录小朋友交换次数

int C1[1000005],C2[1000005];//两个树状数组（通过对小朋友身高计数来建立树状数组，C1是从头到尾计数，C2为从尾到头计数）

long long total[100005];//交换n次的总共不开心值

int lowbit(int x)

{

return x&(-x);

}

int getsum(int i,int \*c)//求a[1]到a[i]的和(包含a[i])

{

int res = 0;

while(i > 0)

{

res += c[i];

i -= lowbit(i);

}

return res;

}

void updata(int i,int k,int \*c)//在i位置加上k（此函数只更改树状数组的值，若要更改原数组的值需要在主函数中实现）

{

while(i <= 1000005)//小朋友最高为1000000，所以小于1000005

{

c[i] += k;

i += lowbit(i);

}

}

int main() {

scanf("%d",&N);

int i;

for(i = 0; i < N; i++) {//从头到尾计数（C1）

scanf("%d",&a[i]);

updata(a[i] + 1,1,C1);//因为小朋友身高值可以为0，而树状数组下标不能用0，所以要都加1

b[i] = i - getsum(a[i],C1);//计算在当前数之前且比它大的元素个数

b[i] = b[i] - (getsum(a[i] + 1,C1) - getsum(a[i],C1) - 1);//去重（身高相等的情况）

}

for(i = N - 1; i >= 0; i--) {//从尾到头计数（C2）

updata(a[i] + 1,1,C2);

b[i] = b[i] + getsum(a[i],C2);//计算在当前数之后且比它小的元素个数

}

for(i = 1; i <= N; i++) {//计算交换n次的总共不开心值

total[i] = total[i - 1] + i;

}

long long ans = 0;

for(i = 0; i < N; i++) {//计算结果

ans += total[b[i]];

}

printf("%I64d\n",ans);

return 0;

}

### 1 原题目

### 2 基础知识

#### 2.1数据结构

无。

#### 2.2算法

无。

### 3 解法

#### 3.1 解法1（C++已过）

本人代码：

|  |
| --- |
|  |

### 1 原题目

### 2 基础知识

#### 2.1数据结构

无。

#### 2.2算法

无。

### 3 解法

#### 3.1 解法1（C++已过）

本人代码：

|  |
| --- |
|  |

### 1 原题目

### 2 基础知识

#### 2.1数据结构

无。

#### 2.2算法

无。

### 3 解法

#### 3.1 解法1（C++已过）

本人代码：

|  |
| --- |
|  |

### 1 原题目

### 2 基础知识

#### 2.1数据结构

无。

#### 2.2算法

无。

### 3 解法

#### 3.1 解法1（C++已过）

本人代码：

|  |
| --- |
|  |