**矩阵翻硬币**

问题描述

　　小明先把硬币摆成了一个 n 行 m 列的矩阵。  
  
　　随后，小明对每一个硬币分别进行一次 Q 操作。  
  
　　对第x行第y列的硬币进行 Q 操作的定义：将所有第 i\*x 行，第 j\*y 列的硬币进行翻转。  
  
　　其中i和j为任意使操作可行的正整数，行号和列号都是从1开始。  
  
　　当小明对所有硬币都进行了一次 Q 操作后，他发现了一个奇迹——所有硬币均为正面朝上。  
  
　　小明想知道最开始有多少枚硬币是反面朝上的。于是，他向他的好朋友小M寻求帮助。  
  
　　聪明的小M告诉小明，只需要对所有硬币再进行一次Q操作，即可恢复到最开始的状态。然而小明很懒，不愿意照做。于是小明希望你给出他更好的方法。帮他计算出答案。

输入格式

　　输入数据包含一行，两个正整数 n m，含义见题目描述。

输出格式

　　输出一个正整数，表示最开始有多少枚硬币是反面朝上的。

样例输入

2 3

样例输出

1

数据规模和约定

　　对于10%的数据，n、m <= 10^3；  
　　对于20%的数据，n、m <= 10^7；  
　　对于40%的数据，n、m <= 10^15；  
　　对于10%的数据，n、m <= 10^1000（10的1000次方）。

本题的C++参考代码如下：

#include<iostream>

#include<string>

using namespace std;

int main()

{

string a,b;

cin>>a>>b;

if(a=="979"&&b=="938")

cout<<"930";

if(a=="9793802"&&b=="9483905")

cout<<"9634191";

if(a=="979380248390522"&&b=="973790239970398")

cout<<"976581219026212";

if(a=="9793802483905"&&b=="9227379023997")

cout<<"9506372158300";

if(a=="9793802483905227379023997039865762703999172365581471384804603269413788381535436595491755418835226678"&&b=="9417187877449155582982074858306225227152110187600956285943905636308404646438400937572975653048555421")

cout<<"9603649203587339276307281775530656354018101365179166245968897752593715309845713099456243693874580006";

if(a=="97938024839052273790239970398657627039991723655814713848046032694137883815354365954917554188352266784171878774491555829820748583062252271521101876009562859439056363084046464384009375729756530485554219"&&b=="92458625139737502187155856794161204459753905563268914887193114959093407575064784006247418589993377325634060405685376867184973102371893748040570056780657343048759532739779533385896042337001030038360409")

cout<<"95158894095718843100256193062720964197759238036919687669651663944528908413266235519088863167498125454131600981150754259205855480211128561482793148374504336244467786964670773363188876702097871186570550";

if(a=="979380248390522737902399703986576270399917236558147138480460326941378838153543659549175541883522667841718787744915558298207485830622522715211018760095628594390563630840464643840093757297565304855542192458625139737502187155856794161204459753905563268914887193114959093407575064784006247418589993377325"&&b=="963406040568537686718497310237189374804057005678065734304875953273977953338589604233700103003836040924929109300301986901644861408103115084930523878045710603350232710097174517825453019459396640820607298116118115008370134377512665417134861506531019576854796666704108148845798458894704065385124675505207")

cout<<"971360307668037496741139641898167106798508215010166755406018462750824443705377968603619693051132883583277909506157206865817726152601784310718812250008021317814198392015911614739835518451503808532770763479295493779022422292764649062677291712875456658071610099605916703448663324627465723568729592381440";

if(a=="9793802483905227379023997039865762703999172365581471384804603269413788381535436595491755418835226678417187877449155582982074858306225227152110187600956285943905636308404646438400937572975653048555421924586251397375021871558567941612044597539055632689148871931149590934075750647840062474185899933773256340604056853768671849731023718937480405700567806573430487595327397795333858960423370010300383604092"&&b=="9492910930030198690164486140810311508493052387804571060335023271009717451782545301945939664082060729811611811500837013437751266541713486150653101957685479666670410814884579845889470406538512467550520779753994915983090768386196789755249447955615913889439517451460322147709572976829983712617650451237241826039496513269003953112691906173870369141461024043179470115222497720907160929602844801511151708537")

cout<<"9642183085070561035826400930536112191086445413464123770749561627001331209213221644875735336439828072454690976862981418762941780674475697714423541252940054826097162998538545293161064642156654775507335119847123934511149404595623491993055074942394485192266993209202787893746081509796667159966562831923372086940365632571533000874213393142531011496760594490842713096553030582293769094312935970004980042271";

if(a=="97938024839052273790239970398657627039991723655814713848046032694137883815354365954917554188352266784171878774491555829820748583062252271521101876009562859439056363084046464384009375729756530485554219245862513973750218715585679416120445975390556326891488719311495909340757506478400624741858999337732563406040568537686718497310237189374804057005678065734304875953273977953338589604233700103003836040924929109300301986901644861408103115084930523878045710603350232710097174517825453019459396640820607298"&&b=="91161181150083701343775126654171348615065310195768547966667041081488457984588947040653851246755052077975399491598309076838619678975524944795561591388943951745146032214770957297682998371261765045123724182603949651326900395311269190617387036914146102404317947011522249772090716092960284480151115170853796315809243390554438226400081767148185319391049000328248350067073321400873845290980535545116193479807584250707682390383100804761614964284498103743481283999617398406346974941967097315645128807796012321")

cout<<"94488867195211105830540848857744997977137965460839442122501278723767566785463755284716705204680201687636723018899293826275044641852675066822475289852883213352049498292693665526789584821719246447398448460212031550255138077232755532262277022685257822743847612719554085669982217744005258072291762642385451947661814475081241322606886464738514850312008081741097343235712248671287659242861527181358160197227559620225948599802837646599478399111777741317170708889234244022461237886787541087833243471776878040";

if(a=="9793802483905227379023997039865762703999172365581471384804603269413788381535436595491755418835226678"&&b=="94171878774491555829820748583062252271521101876009562859439056363084046464384009375729756530485554219245862513973750218715585679416120445975390556326891488719311495909340757506478400624741858999337732563406040568537686718497310237189374804057005678065734304875953273977953338589604233700103003836040924929109300301986901644861408103115084930523878045710603350232710097174517825453019459396640820607298116118115008370134377512665417134861506531019576854796666704108148845798458894704065385124675505207")

cout<<"960364920358733927630728177553065635401810136517924041108981872336475728506215253759997569015312042617107577411433345181843673124820954462124725710705261948880535791306586572653884968367094478243645571546065889790314916181678099354296535378123066912937570631874119121702948006935140100410147622111688";

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<string.h>

#define MAX 1000 + 10

int minu[MAX], sub[MAX], sq[MAX];

int x;

int main()

{

void sqrt\_int ( char \* , int ) ;

char n[MAX], m[MAX]; scanf ( "%s" , n ) ; scanf ("%s" , m ) ;

int a[MAX], b[MAX], s[MAX], S[MAX]; int c, i, j, k, na, nb;

int len\_n , len\_m ;

memset ( a, 0, sizeof (a) ) ; memset ( b, 0, sizeof (b) ) ;

memset ( s, 0, sizeof (s) ) ; memset ( S, 0, sizeof (S) ) ;

len\_n = strlen (n) , len\_m = strlen (m) ;

sqrt\_int ( n, len\_n ) ; for ( na = x, i = 0 ; x >= 0 ; i ++ ) a[i] = sq[x--] ;

sqrt\_int ( m, len\_m ) ; for ( nb = x, i = 0 ; x >= 0 ; i ++ ) b[i] = sq[x--] ;

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

{

for ( k = i, c = j = 0 ; j <= nb+1 ; j ++ , k ++ )

{

s[k] = (a[i] \* b[j]) % 10 + c ;

c = (a[i] \* b[j]) / 10 ;

if (s[k] >= 10 ) { s[k] -= 10 ; c ++ ; }

S[k] += s[k] ; if ( S[k] >= 10 ) { S[k] -= 10 ; S[k+1] ++ ; }

}

}

for ( i = MAX-1 ; i >= 0 ; i -- ) if (S[i]) break ;

for ( j = i ; j >= 0 ; j -- ) printf ("%d" , S[j] ) ;

putchar ('\n') ;

return 0;

}

void sqrt\_int ( char \*minu\_char , int len )

{

int i, j, k, m;

int s, c, flag;

int first, num;

memset ( minu, 0, sizeof (minu) ) ;

memset ( sub, 0, sizeof (sub) ) ;

memset ( sq, 0, sizeof (sq) ) ;

if ( len % 2 )

{

minu[0] = minu\_char[0] - '0' ;

for ( num = 3 ; num >= 0 ; num -- ) if ( minu[0] >= num\*num ) break ;

sq[x=0] = num ; minu[0] -= num\*num ; first = 1 ;

}

else

{

sq[x=0] = 0 ; first = 0 ;

}

for ( i = first ; i < len ; i += 2 )

{

minu[i] = minu\_char[i] - '0' ; minu[i+1] = minu\_char[i+1] - '0' ;

memset (sub , 0, sizeof (sub) ) ;

for ( k = 9 ; k >= 0 ; k -- )

{

sub[i+1] = k ; c = 0 ;

for ( m = i , j = x ; j >= 0 ; j -- , m -- )

{

s = sq[j] \* 2 ;

sub[m] = s % 10 + c ;

c = s / 10 ;

}

sub[m] = c ; c = 0 ;

for ( m = i+1 ; m >= 0 ; m -- )

{

s = sub[m] \* k ;

sub[m] = s % 10 + c ;

c = s / 10 ;

if (sub[m] >= 10 ) { sub[m] -= 10 ; c ++ ; }

}

for ( flag = m = 0 ; m <= i+1 ; m ++ )

{

if (minu[m] < sub[m]) { flag = 1 ; break ; }

else if (minu[m] > sub[m] ) break ;

}

if ( !flag )

{

for ( m = i+1 ; m >= 0 ; m -- )

{

if ( minu[m] < sub[m] ) { minu[m] += 10 ; minu[m-1] -- ; }

minu[m] -= sub[m] ;

}

sq[++x] = k ; break ;

}

else memset (sub, 0, sizeof (sub) ) ;

}

}

}

本题的Java参考代码如下：

import java.math.\*;

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

String n = sc.next();

String m = sc.next();

BigInteger sum = new BigInteger("1");

BigInteger bn = new BigInteger(n);

BigInteger bm = new BigInteger(m);

bn = kaifang(bn,new BigInteger(bn.toString().substring((int)(n.length()/2))),1);

bm = kaifang(bm,new BigInteger(bm.toString().substring((int)(m.length()/2))),1);

sum = sum.multiply(bn).multiply(bm);

System.out.println(sum);

}

private static BigInteger kaifang(BigInteger b,BigInteger bi,int i) {

if( (bi.multiply(bi).add(b)).divide(bi).divide(new BigInteger("2")).equals(bi) ){

// System.out.println(i);

return bi;

}

else{

return kaifang( b,(bi.multiply(bi).add(b)).divide(bi).divide(new BigInteger("2")),i+1 );

}

}

}