**历届试题 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) ) ;

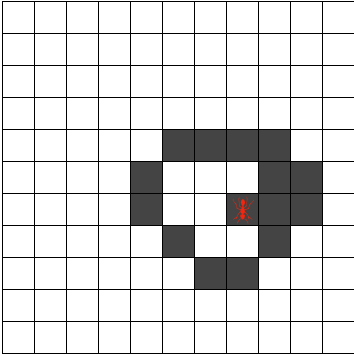
}

}

}

**历届试题 兰顿蚂蚁**

问题描述

  
  
　　兰顿蚂蚁，是于1986年，由克里斯·兰顿提出来的，属于细胞自动机的一种。  
  
　　平面上的正方形格子被填上黑色或白色。在其中一格正方形内有一只“蚂蚁”。  
　　蚂蚁的头部朝向为：上下左右其中一方。  
  
　　蚂蚁的移动规则十分简单：  
　　若蚂蚁在黑格，右转90度，将该格改为白格，并向前移一格；  
　　若蚂蚁在白格，左转90度，将该格改为黑格，并向前移一格。  
  
　　规则虽然简单，蚂蚁的行为却十分复杂。刚刚开始时留下的路线都会有接近对称，像是会重复，但不论起始状态如何，蚂蚁经过漫长的混乱活动后，会开辟出一条规则的“高速公路”。  
  
　　蚂蚁的路线是很难事先预测的。  
  
　　你的任务是根据初始状态，用计算机模拟兰顿蚂蚁在第n步行走后所处的位置。

输入格式

　　输入数据的第一行是 m n 两个整数（3 < m, n < 100），表示正方形格子的行数和列数。  
　　接下来是 m 行数据。  
　　每行数据为 n 个被空格分开的数字。0 表示白格，1 表示黑格。  
  
　　接下来是一行数据：x y s k, 其中x y为整数，表示蚂蚁所在行号和列号（行号从上到下增长，列号从左到右增长，都是从0开始编号）。s 是一个大写字母，表示蚂蚁头的朝向，我们约定：上下左右分别用：UDLR表示。k 表示蚂蚁走的步数。

输出格式

　　输出数据为两个空格分开的整数 p q, 分别表示蚂蚁在k步后，所处格子的行号和列号。

样例输入

5 6  
0 0 0 0 0 0  
0 0 0 0 0 0  
0 0 1 0 0 0  
0 0 0 0 0 0  
0 0 0 0 0 0  
2 3 L 5

样例输出

1 3

样例输入

3 3  
0 0 0  
1 1 1  
1 1 1  
1 1 U 6

样例输出

0 0

本题的C++参考代码如下：

//天农计算机许晓华老师出品

#include<iostream>

using namespace std;

#define N 100

int dx[]={-1,0,1,0};

int dy[]={0,1,0,-1};

char d[]={'U','R','D','L'};//上右下左

int main()

{

int a[N][N],m,n,i,j;

cin>>m>>n;

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

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

cin>>a[i][j];

int x,y,k;

char s;

cin>>x>>y>>s>>k;

for(i=0;i<4;i++)

{

if(d[i]==s) break;

}

while(k--)

{

if(a[x][y])//黑格

i=(i+1)%4;//右转

else//白格

i=(i+3)%4;//左转

a[x][y]^=1;//黑白互换

//if(x>0&&y>0&&x<m-1&&y<n-1)

if(!x&&!i) continue;//不能上

if(y==n-1&&i==1) continue;//不能右

if(x==m-1&&i==2) continue;//不能下

if(!y&&i==3) continue;//不能左

x=x+dx[i],y=y+dy[i];

}

cout<<x<<" "<<y;

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

int a[1000][1000]={

0

};

char f(int i,char now)

{

if(i==0&&now=='U'||i==1&&now=='D')

return 'R';

else if(i==0&&now=='R'||i==1&&now=='L')

return 'D';

else if(i==0&&now=='D'||i==1&&now=='U')

return 'L';

else if(i==0&&now=='L'||i==1&&now=='R')

return 'U';

}

int main()

{

int n,m,i1,i2,x,y,k;

char s;

scanf("%d%d",&n,&m);

for(i1=0;i1<n;i1++)

for(i2=0;i2<m;i2++)

scanf("%d",&a[i1][i2]);

scanf("%d%d",&x,&y);

getchar();

scanf("%c%d",&s,&k);

char now=s;

while(k--)

{ if(a[x][y]==0)

a[x][y]=1;

else

a[x][y]=0;

now=f(a[x][y],now);

if(now=='U')

x--;

else if(now=='D')

x++;

else if(now=='L')

y--;

else

y++;

}

printf("%d %d\n",x,y);

return 0;

}

**历届试题 分糖果**

问题描述

　　有n个小朋友围坐成一圈。老师给每个小朋友随机发偶数个糖果，然后进行下面的游戏：  
  
　　每个小朋友都把自己的糖果分一半给左手边的孩子。  
  
　　一轮分糖后，拥有奇数颗糖的孩子由老师补给1个糖果，从而变成偶数。  
  
　　反复进行这个游戏，直到所有小朋友的糖果数都相同为止。  
  
　　你的任务是预测在已知的初始糖果情形下，老师一共需要补发多少个糖果。

输入格式

　　程序首先读入一个整数N(2<N<100)，表示小朋友的人数。  
　　接着是一行用空格分开的N个偶数（每个偶数不大于1000，不小于2）

输出格式

　　要求程序输出一个整数，表示老师需要补发的糖果数。

样例输入

3  
2 2 4

样例输出

4

本题的C++参考代码如下：

#include <iostream>

using namespace std;

int a[100]; // 小朋友人数<100

bool notEqual(int n)

{

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

if (a[0] != a[i])

return true;

return false;

}

int main()

{

int i, n, t, s = 0;

cin >> n;

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

cin >> a[i];

while ( notEqual(n) )

{

a[0] /= 2;

t = a[0];

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

{

a[i] /= 2;

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

}

a[n-1] += t;

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

if (a[i] % 2) // 奇数个颗糖

++a[i], ++s;

}

cout << s << '\n'; // 补发糖果总数数

return 0;

}

#include <iostream>

using namespace std;

int a[100]; // 小朋友人数<100

bool notEqual(int n)

{

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

if (a[0] != a[i])

return true;

return false;

}

int main()

{

int i, n, t, s = 0;

cin >> n;

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

cin >> a[i];

while ( notEqual(n) )

{

a[0] /= 2;

t = a[0];

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

{

a[i] /= 2;

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

}

a[n-1] += t;

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

if (a[i] % 2) // 奇数个颗糖

++a[i], ++s;

}

cout << s << '\n'; // 补发糖果总数数

return 0;

}

**历届试题 小朋友排队**

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。

本题的C++参考代码如下：

#include<algorithm>

#include <iostream>

#include <cstdio>

#include <cstring>

#include<queue>

#include<set>

#include<stack>

#include<map>

#include<cmath>

#include<vector>

#include<string>

#define \_\_LL64 long long

#define clr(a) memset(a,0,sizeof a)

using namespace std;

const double pi = acos(-1.0);

const double eps = 1e-9;

const long long M = 1000000007;

const long long INF = 0x3f3f3f3f;

typedef long long LL;

typedef unsigned long long ULL;

const LL dir[4][2] = {0,1,-1,0,0,-1,1,0};

using namespace std;

LL \_buf[20];

inline long long read()

{

char c = getchar();

LL isMinus = 1;

long long x = 0;

while ( c!='-' && ( c<'0' || c>'9' )) c = getchar();

if (c=='-') {isMinus = -1;c= getchar();}

while ( c>='0' &&c<='9' ) {x = x\*10+c-'0';c=getchar();}

return x \* isMinus;

}

inline void write(long long num,LL flag = 1)

{

memset(\_buf,0,sizeof(\_buf));

if (num<0) {putchar('-');num=-num;}

LL p = 0;

if (num==0) p++;

else while (num){\_buf[p++]=num%10;num/=10;}

for (LL i=p-1;i>=0;--i) putchar(\_buf[i]+'0');

if (flag==1) puts("");

if (flag==0) printf(" ");

}

const LL Max = 111111;

int n,C[Max],a[Max],b[Max];

LL table[Max];

int lowbit(int x) {return x&(-x);}

void add(int x) { while(x<=n) { C[x]++;x+=lowbit(x); } }

int sum(int x)

{

int ans = 0;

while (x>0) {ans+=C[x];

x-=lowbit(x);}

return ans;

}

void hs()

{

sort(b,b+n);

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

a[i] = lower\_bound(b,b+n,a[i]) - b + 1;

}

int main(){

for (int i=1;i<Max;++i)

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

while (scanf("%d",&n)!=EOF)

{

for (int i=0;i<n;++i) b[i] = a[i] = read();

hs();

clr (b);

clr (C);

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

{

add(a[i]);

b[i] += i + 1- sum(a[i]);

}

clr(C);

for (int i=n-1;i>=0;i--)

{

add(a[i]);

b[i] += sum(a[i]-1);

}

LL ans = 0;

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

ans += table[ b[i] ];

write(ans);

}

return 0;

}

题的C参考代码如下：

#include<stdio.h>

int h[100100];

int un[100100];

int b[1000100];

int reb[1000100];

int Lowbit(int x){

return x&(x^(x-1));

}

int sum(int bit[], int idx){

int ret = 0;

while(idx > 0){

ret += bit[idx];

idx -= Lowbit(idx);

}

return ret;

}

void add(int bit[], int idx, int val){

while(idx < 1000100){

bit[idx] += val;

idx += Lowbit(idx);

}

}

long long uVal[100100];

int main(void){

int n, i;

scanf("%d", &n);

uVal[0] = 0;

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

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

h[i]++;

uVal[i + 1] = uVal[i] + i + 1;

un[i] += i - sum(b, h[i]);

add(b, h[i], 1);

}

long long ans = 0;

for(i = n - 1; i >= 0; i--){

un[i] += sum(reb, h[i] - 1);

add(reb, h[i], 1);

ans += uVal[un[i]];

}

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

return 0;

}

**历届试题 波动数列**

问题描述

　　观察这个数列：  
　　1 3 0 2 -1 1 -2 ...  
  
　　这个数列中后一项总是比前一项增加2或者减少3。  
  
　　栋栋对这种数列很好奇，他想知道长度为 n 和为 s 而且后一项总是比前一项增加a或者减少b的整数数列可能有多少种呢？

输入格式

　　输入的第一行包含四个整数 n s a b，含义如前面说述。

输出格式

　　输出一行，包含一个整数，表示满足条件的方案数。由于这个数很大，请输出方案数除以100000007的余数。

样例输入

4 10 2 3

样例输出

2

样例说明

　　这两个数列分别是2 4 1 3和7 4 1 -2。

数据规模和约定

　　对于10%的数据，1<=n<=5，0<=s<=5，1<=a,b<=5；  
　　对于30%的数据，1<=n<=30，0<=s<=30，1<=a,b<=30；  
　　对于50%的数据，1<=n<=50，0<=s<=50，1<=a,b<=50；  
　　对于70%的数据，1<=n<=100，0<=s<=500，1<=a, b<=50；  
　　对于100%的数据，1<=n<=1000，-1,000,000,000<=s<=1,000,000,000，1<=a, b<=1,000,000。

本题的C++参考代码如下：

#include<stdio.h>

#include<iostream>

using namespace std;

const int mod=100000007;

int a,b,n,s,dp[1005][1005],i,j,jt;

void cut(int &a)

{

while(a>=mod){

a-=mod;

}

return;

}

int main(){

scanf("%d%d%d%d",&n,&s,&a,&b);

b%=n;

b\*=-1;

while(b<0){

b+=n;

}

a%=n;

s%=n;

while(s<0){

s+=n;

}

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

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

dp[i][j]=0;

}

}

dp[1][a]=dp[1][b]=1;

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

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

jt=(j+a\*(i+1))%n;

dp[i+1][jt]+=dp[i][j];

cut(dp[i+1][jt]);

jt=(j+b\*(i+1))%n;

if(jt>=n){

jt-=n;

}

dp[i+1][jt]+=dp[i][j];

cut(dp[i+1][jt]);

}

}

printf("%d\n",dp[n-1][s]);

return 0;

}

本题的C参考代码如下：

#define mod 100000007

int n,s,a,b,x[1001][1001],i,j,t;

fun()

{

while(x[i+1][t]>=mod)x[i+1][t]-=mod;

}

int main()

{

scanf("%d%d%d%d",&n,&s,&a,&b);

b%=n;

b\*=-1;

while(b<0)b+=n;

a%=n;

s%=n;

while(s<0)s+=n;

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

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

x[i][j]=0;

x[1][a]=x[1][b]=1;

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

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

{

t=(j+a\*(i+1))%n;

x[i+1][t]+=x[i][j];

fun();

t=(j+b\*(i+1))%n;

if(t>=n)

t-=n;

x[i+1][t]+=x[i][j];

fun();

}

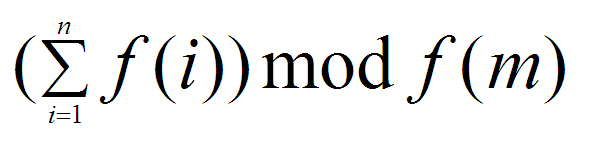
printf("%d\n",x[n-1][s]);

return 0;

}

**历届试题 斐波那契**

问题描述

　　斐波那契数列大家都非常熟悉。它的定义是：  
  
　　f(x) = 1 .... (x=1,2)  
　　f(x) = f(x-1) + f(x-2) .... (x>2)  
  
　　对于给定的整数 n 和 m，我们希望求出：  
　　f(1) + f(2) + ... + f(n) 的值。但这个值可能非常大，所以我们把它对 f(m) 取模。  
　　公式如下  
  
  
　　但这个数字依然很大，所以需要再对 p 求模。

输入格式

　　输入为一行用空格分开的整数 n m p (0 < n, m, p < 10^18)

输出格式

　　输出为1个整数，表示答案

样例输入

2 3 5

样例输出

0

样例输入

15 11 29

样例输出

25

本题的C++参考代码如下：

#include <map>

#include <set>

#include <list>

#include <cmath>

#include <ctime>

#include <deque>

#include <queue>

#include <stack>

#include <bitset>

#include <cctype>

#include <cstdio>

#include <string>

#include <vector>

#include <cstdlib>

#include <cstring>

#include <iomanip>

#include <sstream>

#include <iostream>

#include <algorithm>

using namespace std;

#define PB push\_back

#define MP make\_pair

#define AA first

#define BB second

#define OP begin()

#define ED end()

#define SZ size()

#define SORT(x) sort(x.OP,x.ED)

#define SQ(x) ((x)\*(x))

#define SSP system("pause")

#define cmin(x,y) x=min(x,y)

#define cmax(x,y) x=max(x,y)

typedef long long LL;

typedef pair<int, int> PII;

const double eps=1e-8;

const double INF=1e20;

const double PI=acos( -1. );

const int MXN = 50;

const LL MOD = 1000000007;

LL llmul( LL a,LL b,LL mod ) {

a%=mod;a+=mod;a%=mod;

b%=mod;b+=mod;b%=mod;

if ( a<b )swap( a,b );

LL ret=0;

while ( b ) {

if ( b&1 )ret=( ret+a )%mod;

a=( a<<1 )%mod;

b/=2;

}

return ret;

}

struct matrix {

LL x[3][3];

matrix() {memset( x,0,sizeof x );}

};

matrix mmul( matrix &A,matrix &B,LL mod ) {

matrix ret;

for ( int i=1; i<=2; i++ )

for ( int j=1; j<=2; j++ )

for ( int k=1; k<=2; k++ )

ret.x[i][j]=( ret.x[i][j]+llmul( A.x[i][k],B.x[k][j],mod ) )%mod;

return ret;

}

本题的C参考代码如下：

该题暂时没有人完全正确，暂时没有该语言的参考程序。

**历届试题 地宫取宝**

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

本题的C++参考代码如下：

#pragma comment(linker,"/STACK:102400000,102400000")

#include <algorithm>

#include <iostream>

//#include <fstream>

#include <climits>

#include <cstdlib>

#include <cstring>

#include <string>

#include <vector>

#include <cstdio>

#include <queue>

#include <stack>

#include <cmath>

#include <list>

#include <set>

#include <map>

#define sf scanf

#define pf printf

#define fst first

#define scd second

#define pb push\_back

#define mkp make\_pair

#define cls(a,x) memset(a,x,sizeof a)

#define dt(x) cout<<#x<<"="<<x<<" ";

#define dte(x) cout<<#x<<"="<<x<<endl;

#if(defined(\_WIN32)||defined(\_\_WIN32\_\_))

typedef \_\_int64 LL;

typedef unsigned \_\_int64 ULL;

#define ll I64

#else

typedef long long LL;

typedef unsigned long long ULL;

#define I64 ll

#endif

using namespace std;

template<class T>inline void scaf(T &v)

{

char ch;

while(ch=getchar())

if(ch<='9' && ch>='0') break;

v=ch-'0';

while(ch=getchar())

if(ch<='9' && ch>='0') v=(v<<1)+(v<<3)+ch-'0';

else break;

}

typedef pair<int,int > PII;

const int MX=50100;

const int mod=1000000007;

int sp[51][51];

int g[51][51];

LL dp[51][51][13];

inline void run()

{

int n,m,k,i,j,ii,jj,kk;

scanf("%d%d%d",&n,&m,&k);

for(i=1;i<=50;++i)

sp[i][1]=sp[1][i]=1;

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

for(j=2;j<=m;++j)

sp[i][j]=sp[i-1][j]+sp[i][j-1];

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

{

for(j=1;j<=m;++j)

{

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

dp[i][j][1]=sp[i][j];

}

}

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

{

for(j=1;j<=m;++j)

{

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

{

for(jj=1;jj<=j;++jj)

{

if(g[ii][jj]<g[i][j])

{

for(kk=2;kk<=k;++kk)

{

dp[i][j][kk]+=dp[ii][jj][kk-1]\*sp[i-ii+1][j-jj+1];

dp[i][j][kk]%=mod;

}

}

}

}

}

}

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

// {

// for(j=1;j<=m;++j)

// {

// pf("%d ",dp[i][j][k]);

// }

// putchar(10);

// }

LL Ans=0;

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

{

for(j=1;j<=m;++j)

{

Ans+=dp[i][j][k]\*sp[n-i+1][m-j+1];

Ans%=mod;

}

}

pf("%I64d\n",Ans);

}

int main()

{

run();

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<string.h>

#define N 55

#define MOD 1000000007

int map[55][55];

int dp[55][55][15][15];

int main(void)

{

int n, m, k;

int i, j, c, val, aMax;

scanf("%d%d%d", &n, &m, &k);

aMax = 0;

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

{

for(j = 1; j <= m; j++)

{

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

// map[i][j]++;

if(aMax < map[i][j])

{

aMax = map[i][j];

}

}

}

memset(dp, 0, sizeof(dp));

dp[1][1][0][0] = 1;

dp[1][1][1][map[1][1]] = 1;

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

{

for(j = 1; j <= m; j++)

{

dp[i][j][0][0] += dp[i][j - 1][0][0] + dp[i - 1][j][0][0];

dp[i][j][0][0] %= MOD;

for(c = 1; c <= k; c++)

{

for(val = 0; val <= aMax; val++)

{

dp[i][j][c][val] += dp[i][j - 1][c][val] + dp[i - 1][j][c][val];

dp[i][j][c][val] %= MOD;

}

if(c == 1)

{

dp[i][j][1][map[i][j]] += dp[i][j - 1][0][0];

dp[i][j][1][map[i][j]] %= MOD;

dp[i][j][1][map[i][j]] += dp[i - 1][j][0][0];

dp[i][j][1][map[i][j]] %= MOD;

}

else

{

for(val = 0; val < map[i][j]; val++)

{

dp[i][j][c][map[i][j]] += dp[i][j - 1][c - 1][val];

dp[i][j][c][map[i][j]] %= MOD;

dp[i][j][c][map[i][j]] += dp[i - 1][j][c - 1][val];

dp[i][j][c][map[i][j]] %= MOD;

}

}

}

}

}

int sum = 0;

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

{

sum += dp[n][m][k][i];

sum %= MOD;

}

printf("%d", sum);

return 0;

}

**历届试题 蚂蚁感冒**

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

本题的C++参考代码如下：

#include<stdio.h>

int n,a[50],b[110];

void work()

{

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

{

if(a[i]>0)

{

b[a[i]] = 1;

}

else

{

b[-a[i]] = -1;

}

}

int ans=1,v=a[0];

bool flag=false;

if(v>0)

{

for(int i=v+1 ; i<110 ; i++)

{

if(b[i]<0)

{

ans++;

flag = true;

}

}

}

else

{

for(int i=-v-1 ; i>0 ; i--)

{

if(b[i]>0)

{

ans++;

flag = true;

}

}

}

if(flag)

{

if(v<0)

{

for(int i=-v+1 ; i<110 ; i++)

{

if(b[i]<0)

{

ans++;

flag = true;

}

}

}

else

{

for(int i=v-1 ; i>0 ; i--)

{

if(b[i]>0)

{

ans++;

flag = true;

}

}

}

}

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

}

void init()

{

scanf("%d",&n);

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

{

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

}

}

int main()

{

init();

work();

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

struct mayi

{

int direct; //0为左，1为右

int dist; //距离左端点距离

int cold; //0为正常，1为感冒

} ;

int main()

{

int n,i,sign,j,num=0;

scanf("%d",&n);

struct mayi a[n];

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

{

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

a[i].dist\*=2;

a[i].direct=1;

a[i].cold=0;

if(a[i].dist<0)

{

a[i].dist\*=-1;

a[i].direct=0;

}

a[0].cold=1;

}

for(;;)

{

sign=0;

for(i=0;i<n;i++) //所有蚂蚁走路

{

if(a[i].direct==0) a[i].dist--;

else a[i].dist++;

}

for(i=0;i<n-1;i++)

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

{

if(a[i].dist==a[j].dist)

{

if(a[i].direct==0)

{a[i].direct=1; }

else a[i].direct=0;

if(a[j].direct==0)

{ a[j].direct=1;}

else a[j].direct=0;

if(a[i].cold==1 ) a[j].cold=1;

if(a[j].cold==1 ) a[i].cold=1;

}

}

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

{

if(a[i].dist>=0 && a[i].dist<=200)

{

sign=1;

break;

}

}

if(sign==0) break;

}

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

{

if(a[i].cold==1) num++;

}

printf("%d\n",num);

return 0;

}

**历届试题 最大子阵**

问题描述

　　给定一个n\*m的矩阵A，求A中的一个非空子矩阵，使这个子矩阵中的元素和最大。  
  
　　其中，A的子矩阵指在A中行和列均连续的一块。

输入格式

　　输入的第一行包含两个整数n, m，分别表示矩阵A的行数和列数。  
　　接下来n行，每行m个整数，表示矩阵A。

输出格式

　　输出一行，包含一个整数，表示A中最大的子矩阵中的元素和。

样例输入

3 3  
-1 -4 3  
3 4 -1  
-5 -2 8

样例输出

10

样例说明

　　取最后一列，和为10。

数据规模和约定

　　对于50%的数据，1<=n, m<=50；  
　　对于100%的数据，1<=n, m<=500，A中每个元素的绝对值不超过5000。

本题的C++参考代码如下：

#include "stdio.h"

#include "string.h"

int a[500][500],b[500];

int f(int n,int m)

{int i,j,k,t,max=-999999;

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

{memset(b,0,m\*sizeof(int));

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

{t=-999999;

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

{b[k]+=a[j][k];

t+=b[k];

if(t<b[k])t=b[k];

if(max<t)max=t;

}

}

}

return max;

}

int main()

{

int i,j,n,m;

scanf("%d%d",&n,&m);

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

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

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

printf("%d",f(n,m));

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int a[500][500],b[500];

int maxsub(int n,int m){

int i,j,k,t,max=-999999;

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

{

memset(b,0,m\*sizeof(int));

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

{

t=-999999;

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

{

b[k]+=a[j][k];

t+=b[k];

if(t<b[k])t=b[k];

if(max<t)max=t;

}

}

}

return max;

}

int main()

{

int i,j,n,m;

scanf("%d%d",&n,&m);

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

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

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

printf("%d",maxsub(n,m));

system("pause");

return 0;

}

**历届试题 城市建设**

问题描述

　　栋栋居住在一个繁华的C市中，然而，这个城市的道路大都年久失修。市长准备重新修一些路以方便市民，于是找到了栋栋，希望栋栋能帮助他。  
  
　　C市中有n个比较重要的地点，市长希望这些地点重点被考虑。现在可以修一些道路来连接其中的一些地点，每条道路可以连接其中的两个地点。另外由于C市有一条河从中穿过，也可以在其中的一些地点建设码头，所有建了码头的地点可以通过河道连接。  
  
　　栋栋拿到了允许建设的道路的信息，包括每条可以建设的道路的花费，以及哪些地点可以建设码头和建设码头的花费。  
  
　　市长希望栋栋给出一个方案，使得任意两个地点能只通过新修的路或者河道互达，同时花费尽量小。

输入格式

　　输入的第一行包含两个整数n, m，分别表示C市中重要地点的个数和可以建设的道路条数。所有地点从1到n依次编号。  
　　接下来m行，每行三个整数a, b, c，表示可以建设一条从地点a到地点b的道路，花费为c。若c为正，表示建设是花钱的，如果c为负，则表示建设了道路后还可以赚钱（比如建设收费道路）。  
　　接下来一行，包含n个整数w\_1, w\_2, …, w\_n。如果w\_i为正数，则表示在地点i建设码头的花费，如果w\_i为-1，则表示地点i无法建设码头。  
　　输入保证至少存在一个方法使得任意两个地点能只通过新修的路或者河道互达。

输出格式

　　输出一行，包含一个整数，表示使得所有地点通过新修道路或者码头连接的最小花费。如果满足条件的情况下还能赚钱，那么你应该输出一个负数。

样例输入

5 5  
1 2 4  
1 3 -1  
2 3 3  
2 4 5  
4 5 10  
-1 10 10 1 1

样例输出

9

样例说明

　　建设第2、3、4条道路，在地点4、5建设码头，总的花费为9。

数据规模和约定

　　对于20%的数据，1<=n<=10，1<=m<=20，0<=c<=20，w\_i<=20；  
　　对于50%的数据，1<=n<=100，1<=m<=1000，-50<=c<=50，w\_i<=50；  
　　对于70%的数据，1<=n<=1000；  
　　对于100%的数据，1 <= n <= 10000，1 <= m <= 100000，-1000<=c<=1000，-1<=w\_i<=1000，w\_i≠0。

本题的C++参考代码如下：

#include <iostream>

#include <cstdio>

#include <algorithm>

using namespace std;

const int MAX = 11111;

int parent[MAX];

struct Edge

{

int st;

int ed;

int cost;

bool operator<(const Edge &rhs)const

{

return cost < rhs.cost;

}

};

Edge E[222222];

int find(int r)

{

if(parent[r] == r)

return r;

return parent[r] = find(parent[r]);

}

void input(int &x)

{

char ch = getchar();

bool flag = ch == '-';

while(ch<'0' || ch>'9')

{

ch = getchar();

if(ch == '-')

flag = true;

}

x = 0;

while(ch>='0' && ch<='9')

{

x = x \* 10 + ch - '0';

ch = getchar();

}

x = flag?-x:x;

}

int weight;

int num;

int tmp\_cost;

int cnt ;

void kruskal(int n, int m)

{

int i = 0;

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

parent[i] = i;

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

{

int x = E[i].st;

int y = E[i].ed;

int rx = find(x);

int ry = find(y);

if(rx != ry)

{

parent[rx] = ry;

cnt++;

if(x==0)

{

num++;

tmp\_cost = E[i].cost;

}

weight+=E[i].cost;

}

else if(E[i].cost<0)

weight += E[i].cost;

}

if(num==1)

weight -= tmp\_cost;

}

int main()

{

//freopen("in.txt","r",stdin);

int n,m;

int cost;

input(n);

input(m);

//scanf("%d%d",&n,&m);

int i;

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

{

input(E[i].st);

input(E[i].ed);

input(E[i].cost);

//scanf("%d%d%d",&E[i].st,&E[i].ed,&E[i].cost);

//if(E[i].cost<0)

//weight+=E[i].cost;

}

sort(E,E+m);

kruskal(n,m);

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

{

input(cost);

//scanf("%d",&cost);

if(cost!=-1)

{

E[m].st = 0;

E[m].ed = i;

E[m++].cost = cost;

}

}

if(cnt==n-1)

{

int minWeight = weight;

weight = 0;

sort(E,E+m);

kruskal(n,m);

minWeight = minWeight < weight?minWeight:weight;

cout<<minWeight<<endl;

}

else

{

weight = 0;

sort(E,E+m);

kruskal(n,m);

cout<<weight<<endl;

}

}

本题的C参考代码如下：

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define INF 0xfffffff

int n,m,k;

int pre[10005];

struct s { int u,v,w; }edge[1000005];

int cmp(const void \*a,const void \*b) { return (\*(struct s \*)a).w-(\*(struct s \*)b).w; }

int find(int x) {

if(x==pre[x]) return x;

return pre[x]=find(pre[x]);

}

void init(int n) {

int i;

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

pre[i]=i;

}

}

int ku(int k) {

int sum=0,i;

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

int u=edge[i].u;

int v=edge[i].v;

int w=edge[i].w;

int fa=find(u);

int fb=find(v);

if(fa!=fb||w<0) {

sum+=w;

pre[fa]=fb;

}

}

return sum;

}

int main() {

//while(scanf("%d%d",&n,&m)!=EOF) {

int i,j;

scanf("%d%d",&n,&m);

init(n);

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

int u,v,w;

scanf("%d%d%d",&u,&v,&w);

edge[i].u=u;

edge[i].v=v;

edge[i].w=w;

}

k=m;

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

int cost;

scanf("%d",&cost);

if(cost!=-1) {

edge[k].u=0;

edge[k].v=i;

edge[k++].w=cost;

}

}

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

int fa=find(edge[i].u);

int fb=find(edge[i].v);

if(fa!=fb) pre[fa]=fb;

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

if(find(1)!=find(i)) { break; } }

int temp;

if(i==n+1) {

init(n);

qsort(edge,m,sizeof(edge[0]),cmp);

temp=ku(m);

init(n);

qsort(edge,k,sizeof(edge[0]),cmp);

int ans=ku(k);

if(ans>temp) printf("%d\n",temp);

else printf("%d\n",ans);

//continue;

return 0;

}

init(n);

qsort(edge,k,sizeof(edge[0]),cmp);

int ans=ku(k);

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

//}

return 0;

}

**历届试题 邮局**

问题描述

　　C村住着n户村民，由于交通闭塞，C村的村民只能通过信件与外界交流。为了方便村民们发信，C村打算在C村建设k个邮局，这样每户村民可以去离自己家最近的邮局发信。  
  
　　现在给出了m个备选的邮局，请从中选出k个来，使得村民到自己家最近的邮局的距离和最小。其中两点之间的距离定义为两点之间的直线距离。

输入格式

　　输入的第一行包含三个整数n, m, k，分别表示村民的户数、备选的邮局数和要建的邮局数。  
　　接下来n行，每行两个整数x, y，依次表示每户村民家的坐标。  
　　接下来m行，每行包含两个整数x, y，依次表示每个备选邮局的坐标。  
　　在输入中，村民和村民、村民和邮局、邮局和邮局的坐标可能相同，但你应把它们看成不同的村民或邮局。

输出格式

　　输出一行，包含k个整数，从小到大依次表示你选择的备选邮局编号。（备选邮局按输入顺序由1到m编号）

样例输入

5 4 2  
0 0  
2 0  
3 1  
3 3  
1 1  
0 1  
1 0  
2 1  
3 2

样例输出

2 4

数据规模和约定

　　对于30%的数据，1<=n<=10，1<=m<=10，1<=k<=5；  
　　对于60%的数据，1<=m<=20；  
　　对于100%的数据，1<=n<=50，1<=m<=25，1<=k<=10。

本题的C++参考代码如下：

#include <iostream>

#include <stdio.h>

#include <math.h>

using namespace std;

struct Point {

int no;

int x, y;

double getDist(Point p) {

return sqrt((x - p.x) \* (x - p.x) + (y - p.y) \* (y - p.y));

}

};

Point ans[10];

int n, m, k;

Point person[50];

Point fire[25];

Point result[10];

bool repeat[50], ban[50];

double ansDist = 1000000000;

double minDist[50], sum = 0;

double G[50][25];

void dfs(int deep, int index)

{

if(deep == k) {

if(sum < ansDist) {

ansDist = sum;

for(int i = 0; i < k; i++) {

ans[i] = result[i];

}

}

} else {

double tmpDist[50] = {0}, tsum = sum;

bool flag2 = false;

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

tmpDist[i] = minDist[i];

for(int i = m-k+deep; i >= index; i--) {

if(repeat[i]) continue;

if(deep > 0 && ban[i]) continue;

bool flag = false;

if(deep == 0) {

sum = 0;

for(int j = 0; j < n; j++) {

minDist[j] = G[j][i];

sum += G[j][i];

}

flag = true;

}

else {

sum = tsum;

for(int j = 0; j < n; j++) {

if(G[j][i] < tmpDist[j]) {

sum -= tmpDist[j] - G[j][i];

minDist[j] = G[j][i];

flag = true;

} else minDist[j] = tmpDist[j];

}

}

if(flag) {

flag2 = true;

result[deep] = fire[i];

dfs(deep+1, result[deep].no + 1);

} else {

ban[i] = true;

}

}

if(flag2 == false) {

result[deep] = fire[result[deep-1].no+1];

dfs(deep+1, m-k+deep+1);

} else {

sum = tsum;

for(int j = 0; j < n; j++) {

minDist[j] = tmpDist[j];

}

}

}

}

int main(int argc, char \*argv[]) {

scanf("%d%d%d", &n, &m, &k);

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

scanf("%d%d", &person[i].x, &person[i].y);

minDist[i] = ansDist;

}

for(int i = 0; i < m; i++) {

scanf("%d%d", &fire[i].x, &fire[i].y);

fire[i].no = i;

}

for(int i = 0; i < m; i++) {

if(!repeat[i]) {

for(int j = i+1; j < m; j++)

if(fire[i].x == fire[j].x && fire[i].y == fire[j].y)

repeat[j] = true;

}

}

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

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

G[i][j] = person[i].getDist(fire[j]);

dfs(0, 0);

for(int i = 0; i < k; i++) {

printf("%d ", ans[i].no+1);

}

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<math.h>

struct

{

int x, y;

}pos[50];

float dis[25][50];

int res[10];

float minn = 100000000;

int n, m, k, i, j, f[50];

void dfs(int step,int cnt,float sum,int temp[],float las\_dis[])

{

if(step == m || cnt == k)

{

if(cnt == k && minn > sum)

{

minn = sum;

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

res[i] = temp[i];

}

return;

}

else if(cnt == 0)

{

float w[50];

dfs(step+1,cnt,sum,temp,w);

temp[cnt] = step+1;

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

{

sum += dis[step][i];

w[i] = dis[step][i];

}

dfs(step+1,cnt+1,sum,temp,w);

}

else if(m - step == k - cnt)

{

float w[50];

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

w[i] = las\_dis[i];

temp[cnt] = step+1;

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

{

if(w[i] > dis[step][i])

{

sum = sum - w[i] + dis[step][i];

w[i] = dis[step][i];

}

}

dfs(step+1,cnt+1,sum,temp,w);

}

else

{

float w[50];

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

w[i] = las\_dis[i];

dfs(step+1,cnt,sum,temp,w);

if(!f[step])

{

temp[cnt] = step+1;

int flag = 0;

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

{

if(w[i] > dis[step][i])

{

sum = sum - w[i] + dis[step][i];

w[i] = dis[step][i];

flag = 1;

}

}

if(flag)

dfs(step+1,cnt+1,sum,temp,w);

else

f[step] = 1;

}

}

}

int main()

{

int x, y;

scanf("%d%d%d", &n, &m, &k);

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

{

scanf("%d%d",&pos[i].x,&pos[i].y);

}

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

{

scanf("%d%d", &x, &y);

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

{

dis[i][j] = sqrt(pow(x - pos[j].x,2) + pow(y - pos[j].y,2));

}

}

int temp[25];

float las\_dis[50];

dfs(0,0,0,temp,las\_dis);

printf("%d",res[0]);

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

{

printf(" %d",res[i]);

}

printf("\n");

return 0;

}

**历届试题 数字游戏**

问题描述

　　栋栋正在和同学们玩一个数字游戏。  
  
　　游戏的规则是这样的：栋栋和同学们一共n个人围坐在一圈。栋栋首先说出数字1。接下来，坐在栋栋左手边的同学要说下一个数字2。再下面的一个同学要从上一个同学说的数字往下数两个数说出来，也就是说4。下一个同学要往下数三个数，说7。依次类推。  
  
　　为了使数字不至于太大，栋栋和同学们约定，当在心中数到 k-1 时，下一个数字从0开始数。例如，当k=13时，栋栋和同学们报出的前几个数依次为：  
　　1, 2, 4, 7, 11, 3, 9, 3, 11, 7。  
  
　　游戏进行了一会儿，栋栋想知道，到目前为止，他所有说出的数字的总和是多少。

输入格式

　　输入的第一行包含三个整数 n,k,T，其中 n 和 k 的意义如上面所述，T 表示到目前为止栋栋一共说出的数字个数。

输出格式

　　输出一行，包含一个整数，表示栋栋说出所有数的和。

样例输入

3 13 3

样例输出

17

样例说明

　　栋栋说出的数依次为1, 7, 9，和为17。

数据规模和约定

　　1 < n,k,T < 1,000,000；

本题的C++参考代码如下：

#include<iostream>

#include<string>

using namespace std;

int main()

{

int a,b,c;

cin>>a>>b>>c;

if(a==5&&b==6&&c==41)

cout<<111;

if(a==24&&b==31&&c==100)

cout<<1419;

if(a==384&&b==534&&c==999)

cout<<240141;

if(a==1341&&b==5141&&c==7000)

cout<<17622171;

if(a==4325&&b==54232&&c==88888)

cout<<2409223620;

if(a==121233&&b==88888&&c==999999)

cout<<44423750111;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>/\*

#include <windows.h>3 9\*/

int s[1000000];

// 3 13 9

// 1 2 4 7 11 3 9 3 11 7 4 2 1 1 2 4 7 11 3 9 3 11 7 4 2 1

int main()

{

\_\_int64 n,k,t,i,j=1,g=1,z,sum=0;

scanf("%I64d%I64d%I64d",&n,&k,&t);

if(k%2==0) z=k\*2;

else z=k;

s[0]=1;

do //开始制表

{

j+=g;

if(j>=k) j%=k;

s[g]=j;

g++;

}

while(g<=z) ;

k=0;

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

{

sum+=s[k];

k+=n;

if(k>g-1) k-=g-1;

}

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

//system("pause");

return 0;

}

**历届试题 国王的烦恼**

问题描述

　　C国由n个小岛组成，为了方便小岛之间联络，C国在小岛间建立了m座大桥，每座大桥连接两座小岛。两个小岛间可能存在多座桥连接。然而，由于海水冲刷，有一些大桥面临着不能使用的危险。  
  
　　如果两个小岛间的所有大桥都不能使用，则这两座小岛就不能直接到达了。然而，只要这两座小岛的居民能通过其他的桥或者其他的小岛互相到达，他们就会安然无事。但是，如果前一天两个小岛之间还有方法可以到达，后一天却不能到达了，居民们就会一起抗议。  
  
　　现在C国的国王已经知道了每座桥能使用的天数，超过这个天数就不能使用了。现在他想知道居民们会有多少天进行抗议。

输入格式

　　输入的第一行包含两个整数n, m，分别表示小岛的个数和桥的数量。  
　　接下来m行，每行三个整数a, b, t，分别表示该座桥连接a号和b号两个小岛，能使用t天。小岛的编号从1开始递增。

输出格式

　　输出一个整数，表示居民们会抗议的天数。

样例输入

4 4  
1 2 2  
1 3 2  
2 3 1  
3 4 3

样例输出

2

样例说明

　　第一天后2和3之间的桥不能使用，不影响。  
　　第二天后1和2之间，以及1和3之间的桥不能使用，居民们会抗议。  
　　第三天后3和4之间的桥不能使用，居民们会抗议。

数据规模和约定

　　对于30%的数据，1<=n<=20，1<=m<=100；  
　　对于50%的数据，1<=n<=500，1<=m<=10000；  
　　对于100%的数据，1<=n<=10000，1<=m<=100000，1<=a, b<=n， 1<=t<=100000。

本题的C++参考代码如下：

#include <iostream>

#include <vector>

#include <algorithm>

#include <cstring>

using namespace std;

int fa[10005];

struct Edge

{

int x,y;

int t;

friend bool operator <(Edge e1,Edge e2)

{

return e1.t>e2.t;

}

}e[100005];

int vis[100005];

int findfa(int x)

{

if (fa[x]==x) return x;

else return fa[x]=findfa(fa[x]);

}

int merges(int a,int b)

{

if (findfa(a)!=findfa(b)){

fa[findfa(a)]=findfa(b);

return 1;

}

else return 0;

}

int main()

{

memset(vis,0,sizeof(vis));

ios::sync\_with\_stdio(false);

int n,m;

cin >>n>>m;

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

{

fa[i]=i;

}

int a,b,c;

for (int i=0;i<m;i++)

{

cin >>a>>b>>c;

e[i].x=a;e[i].y=b;e[i].t=c;

}

sort(e,e+m);

for (int i=0;i<m;i++)

{

//cout <<endl;

//cout <<e[i].x<<" "<<e[i].y<<" "<<e[i].t<<" "<<fa[e[i].x]<<" "<<fa[e[i].y]<<endl;

int flag=merges(e[i].x,e[i].y);

//cout <<flag<<endl;

if (!vis[e[i].t]) vis[e[i].t]=flag;

}

int ans=0;

for (int i=1;i<=100000;i++)

{

ans+=vis[i];

}

cout <<ans<<endl;

}

本题的C参考代码如下：

#include <stdio.h>

#include <stdlib.h>

struct NODE

{

int u,v,w;}edge[100000];

int father[10001];

int cmp(const void \*a,const void \*b)

{

return (\*(struct NODE \*)b).w-(\*(struct NODE \*)a).w;

}

int find(int num)

{

int root=num,tmp;

while(father[root]!=root)

root=father[root];

while(root!=num)

{

tmp=father[num];

father[num]=root;

num=tmp;

}

return root;

}

int merge(int n1,int n2)

{

int root1=find(n1),root2=find(n2);

if(root1!=root2)

{

father[root1]=root2;

return 1;

}

return 0;

}

int main()

{

int i,n,m,cnt=0,tmp=0,ans=0;

scanf("%d%d",&n,&m);

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

scanf("%d%d%d",&edge[i].u,&edge[i].v,&edge[i].w);

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

father[i]=i;

qsort(edge,m,sizeof(edge[0]),cmp);

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

{

if(merge(edge[i].u,edge[i].v))

{

if(tmp!=edge[i].w)

ans++;

tmp=edge[i].w;

cnt++;

}

if(cnt==n-1)

break;

}

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

return 0;

}

**历届试题 回文数字**

问题描述

　　观察数字：12321，123321 都有一个共同的特征，无论从左到右读还是从右向左读，都是相同的。这样的数字叫做：回文数字。  
  
　　本题要求你找到一些5位或6位的十进制数字。满足如下要求：  
　　该数字的各个数位之和等于输入的整数。

输入格式

　　一个正整数 n (10<n<100), 表示要求满足的数位和。

输出格式

　　若干行，每行包含一个满足要求的5位或6位整数。  
　　数字按从小到大的顺序排列。  
　　如果没有满足条件的，输出：-1

样例输入

44

样例输出

99899  
499994  
589985  
598895  
679976  
688886  
697796  
769967  
778877  
787787  
796697  
859958  
868868  
877778  
886688  
895598  
949949  
958859  
967769  
976679  
985589  
994499

样例输入

60

样例输出

-1

本题的C++参考代码如下：

#include<stdio.h>

int main()

{

int n;

int flag=0;

scanf("%d",&n);

int i,j,k;

for(i=1;i<=9;i++)

for(j=0;j<=9;j++)

for(k=0;k<=9;k++)

{

if(2\*i+2\*j+k==n)

{

printf("%d%d%d%d%d\n",i,j,k,j,i);

flag=1;

}

}

for(i=1;i<=9;i++)

for(j=0;j<=9;j++)

for(k=0;k<=9;k++)

{

if(2\*i+2\*j+2\*k==n)

{

printf("%d%d%d%d%d%d\n",i,j,k,k,j,i);

flag=1;

}

}

if(flag==0)

printf("-1");

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

int main()

{

int a,b,c;

int n;

int flag=-1; //标志位

scanf("%d",&n);

for(a=1;a<10;a++)

{

for(b=0;b<10;b++)

{

for(c=0;c<10;c++)

{

if(a+b+c+b+a==n)

{

flag=1;

printf("%d%d%d%d%d\n",a,b,c,b,a);

}

}

}

}

for(a=1;a<10;a++)

{

for(b=0;b<10;b++)

{

for(c=0;c<10;c++)

{

if(a+b+c+c+b+a==n)

{

printf("%d%d%d%d%d%d\n",a,b,c,c,b,a);

flag=1;

}

}

}

}

if(flag==-1)

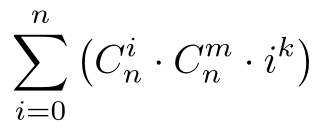
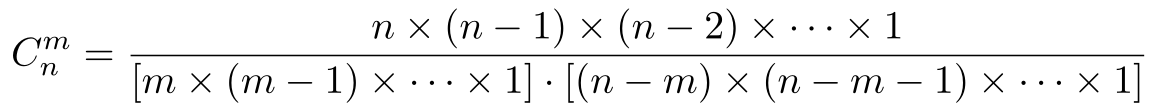
printf("%d\n",flag);

return 0;

}

**历届试题 公式求值**

问题描述

　　输入n, m, k，输出下面公式的值。  
  
　　其中C\_n^m是组合数，表示在n个人的集合中选出m个人组成一个集合的方案数。组合数的计算公式如下。  


输入格式

　　输入的第一行包含一个整数n；第二行包含一个整数m，第三行包含一个整数k。

输出格式

　　计算上面公式的值，由于答案非常大，请输出这个值除以999101的余数。

样例输入

3  
1  
3

样例输出

162

样例输入

20  
10  
10

样例输出

359316

数据规模和约定

　　对于10%的数据，n≤10，k≤3；  
　　对于20%的数据，n≤20，k≤3；  
　　对于30%的数据，n≤1000，k≤5；  
　　对于40%的数据，n≤10^7，k≤10；  
　　对于60%的数据，n≤10^15，k ≤100；  
　　对于70%的数据，n≤10^100，k≤200；  
　　对于80%的数据，n≤10^500，k ≤500；  
　　对于100%的数据，n在十进制下不超过1000位，即1≤n<10^1000，1≤k≤1000，同时0≤m≤n，k≤n。

提示

　　999101是一个质数；  
　　当n位数比较多时，绝大多数情况下答案都是0，但评测的时候会选取一些答案不是0的数据；

本题的C++参考代码如下：

#include<iostream>

#include<string>

using namespace std;

int main()

{

string a,b,c;

cin>>a>>b>>c;

if(a=="9"&&b=="7"&&c=="2")

cout<<"414720";

if(a=="18"&&b=="10"&&c=="3")

cout<<"561400";

if(a=="990"&&b=="300"&&c=="5")

cout<<"941446";

if(a=="7349813"&&b=="3590741"&&c=="9")

cout<<"591101";

if(a=="441083274867"&&b=="394298332530"&&c=="60")

cout<<"418738";

if(a=="823420901930351"&&b=="81400162550286"&&c=="99")

cout<<"438500";

if(a=="28586055549436065778332471275445878099330744807897345804763499678366766114841030555243753890626877"&&b=="11828816409476215599330457923431667883616871345906026967072680965603447821962604789224750212075171"&&c=="197")

cout<<"252024";

if(a=="6967546168712455427206351828581082751153963107895261276464667787541395093433324713375806465754024302965084866409517610925429697112151710074431216079815357281232747660569120335785527305135760198613452314526369984645974029453000492157554313399724268627531965350725519662295308728380841974543566559231291065760780914028073774417120779346331600419267942518139033717853760526555032055734465653700838350021329969006191329401918940942836088231970801471781046385005331757133731210926916058256859281084143650"&&b=="3976961805945090797718410564018307197945529573923025443483602098359292472809768594269622648535562627135860905542073964564038981015839729109769504808805882978172643380101599436174401406095018822974034463505066923133348317623126279639780163426340291960500026368254050732787695382706799849391084264287001915257084246714206939205839071844812312314353544424740073657183316198539565043130590539028823935224266341753408209408956480189752492279868197875853121620827727188567493411989169865366604547669993975"&&c=="495")

cout<<"730993";

if(a=="2389429278534381811327749560919669603592396580433315752747729523203738991362949824943565314867190837505707187815301659506316551565483652431618188326886477500908640362878301663353782425982527026795501645245874075744263522419021504821133861444383162351426548866722526712970341897549636516189106450753573429379413855700684106406139998095859111734063463080335700440240966323152369063830839424581970138856977655051485832150324921752474751379670303317986936281742730494091134564298175942425406454442230948921595893035482810043708749819464414241705425897709854195693499773688858444983151744918149073310748987622482250480529397916937267551975892233326094520856151131540378703988558794735825464976979442596335924983999823484428753498201178452037032551061597841709597162797633620727866260406115363675392044949773119897855895573106438266583573254267517077472206404536092911613906366350060957813504067660720535467361158797512442279414026088850197680703714757940617916815309806615368444584693343934381355012907717"&&b=="397203656601430500452808338326695321007702241461915966105472464487034697457724872917144244449106456715251125785418090393224577029254317580556888519986021589175195745278395490598870323607366855962969636669200395505340135135113190979840210893569627806651941874017584949965180192684138912210174433096128914645315703620435445647722205906010981167565684125583968766253732991913036234412307191946243399873704491431841463156299339105868811947033503933741493618635237699371839742424150088578648002978612344064595843253545736321334008240095859561603104724425239278435089098941783039199158943881462907150604365146962123263434384792590313790113293869802308130284269888619642036745136645053001568906320813379048672815063255621818976238413558486368737180250713745110716049493657662528749538659169534179273429620258016171076883999352192592335984612215519653388973583335167637066987898716953417980608640425374576783297751206784687064284872766112016850337021590767120767444835827382793697115965639472291520761849061"&&c=="999")

cout<<"956287";

if(a=="1003283446431054987505802649995884177219308770169901460850674246314798518964125270752421055744834063959954174446397016113284358638581184979596711022810983881122043907170037887692185774459933111185351432856179125538193623559722368375773838779299084814173901198824826552952836456013228152284007005719620029343419065287735365513014535568656672320990631228394640835909770541164600544479652585185280830559560336804328238482211109099939057563299084716845717929708226575946898499894510787965667383071751003306314659952193405532461030855728057304584848269612504402929681432185255044692702094847763645916057564782578702418166655636493158273383517791698974260749286829959952032926344331318809180755529492965717195044251297165399026960024518916551662497097652783168510860772675066293779455426398575832244975845240247457021667224614786163787556755610216579711593066958547465561952469699069027355081088506737852649323245969014512805307723248838707440038118221254190371268604253808016528934155858666907697039071696"&&b=="484436460467493554222005034570300843560490651792941136661448586684359337589523226010511123940521861457463403579851176856299497721453200901226561896117182831288185210056170097875776875638119278314487860768189436122005147039368066022274028956388358869596217890746207827153044877502387975654723193177162382350468479279186072406888505574788372576901975798615300155900778081578802715935481064874749442387264508841031385186059195878416125482191679740099864744921377699161312915311949724352516547475094190250815609101884233433747504530400280760624115174252298679051843930676894705514561141673340518620527801156707312288231313151139693826713103848420208025670435436070343771646827050283571647402846173000811523705121567114703675664537057877700028568587275342143936033916776932509369698264153669959971354691768156690205343717181337713142677768016340209722551953952373346834678114136360678367771124581820089651785180526447978505924116335338727658367700493795114608640495817088877282858705160742307021788729118"&&c=="971")

cout<<"228000";

return 0;

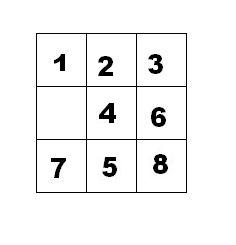
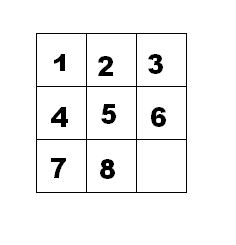
}

本题的C参考代码如下：

该题暂时没有人完全正确，暂时没有该语言的参考程序。

**历届试题 九宫重排**

问题描述

　　如下面第一个图的九宫格中，放着 1~8 的数字卡片，还有一个格子空着。与空格子相邻的格子中的卡片可以移动到空格中。经过若干次移动，可以形成第二个图所示的局面。  
  
　　我们把第一个图的局面记为：12345678.  
　　把第二个图的局面记为：123.46758  
　　显然是按从上到下，从左到右的顺序记录数字，空格记为句点。  
　　本题目的任务是已知九宫的初态和终态，求最少经过多少步的移动可以到达。如果无论多少步都无法到达，则输出-1。

输入格式

　　输入第一行包含九宫的初态，第二行包含九宫的终态。

输出格式

　　输出最少的步数，如果不存在方案，则输出-1。

样例输入

12345678.  
123.46758

样例输出

3

样例输入

13524678.  
46758123.

样例输出

22

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=="12345678."&&b=="152743.86")

cout<<6;

if(a=="12345678."&&b=="12356.784")

cout<<13;

if(a=="2315.6784"&&b=="8235164.7")

cout<<17;

if(a=="12345678."&&b=="87654321.")

cout<<30;

if(a==".87654321"&&b=="12345678.")

cout<<28;

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct \_Node

{

char tab[3][3];

int x,y;

int no;

}Node,\*pNode;

int vx[4]={-1,1,0,0};

int vy[4]={0,0,-1,1};

Node res[400000];

int front=0,rear=0;

int vis[4000000],fact[9];

void input(pNode start);

void bfs(pNode start,pNode end);

void init\_lookup\_table();

int try\_to\_insert(int s);

int main()

{

Node start,end;

input(&start);

input(&end);

bfs(&start,&end);

printf("-1\n");

system("pause");

return 0;

}

void input(pNode start)

{

int i,j;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

scanf("%c",&( (start->tab)[i][j] ));

if((start->tab)[i][j]=='.')

{

start->x = i;

start->y = j;

}

}

}

start->no = 0;

getchar();

}

void bfs(pNode start,pNode end)

{

int i,j;

char ch;

pNode tmp;

init\_lookup\_table();

memcpy(&res[rear],start,sizeof(res[rear]));

try\_to\_insert(rear);

rear++;

while(front!=rear)

{

//printf("%d ",rear);

tmp = &res[front];

if(memcmp(tmp->tab,end->tab,sizeof(end->tab))==0)

{

printf("%d\n",tmp->no);

exit(0);

}

int no = tmp->no;

for(i=0;i<4;i++)

{

int xx = tmp->x+vx[i];

int yy = tmp->y+vy[i];

if(xx>=0 && xx<3 && yy>=0 && yy<3)

{

pNode p = &res[rear];

memcpy(p,tmp,sizeof(res[front]));

p->tab[tmp->x][tmp->y] = p->tab[xx][yy];

p->tab[xx][yy] = tmp->tab[tmp->x][tmp->y];

p->no = no+1;

p->x = xx;

p->y = yy;

if(try\_to\_insert(rear))

{

rear++;

}

}

}

front++;

//printf("%d ",rear);

}

}

void init\_lookup\_table()

{

int i;

fact[0] = 1;

for(i=1;i<9;i++)

{

fact[i] = fact[i-1]\*i;

}

}

int try\_to\_insert(int s)

{

int i,j;

int code = 0;

for(i=0;i<9;i++)

{

int cnt = 0;

for(j=i+1;j<9;j++)

{

if(res[s].tab[j/3][j%3] < res[s].tab[i/3][i%3])

{

cnt++;

}

code += fact[8-i]\*cnt;

}

}

if(vis[code])

{

return 0;

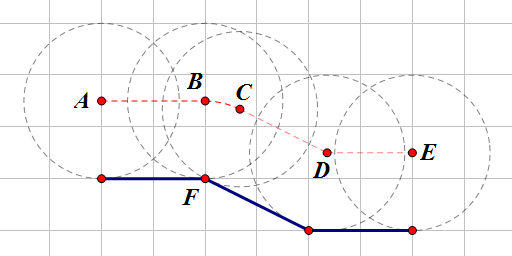
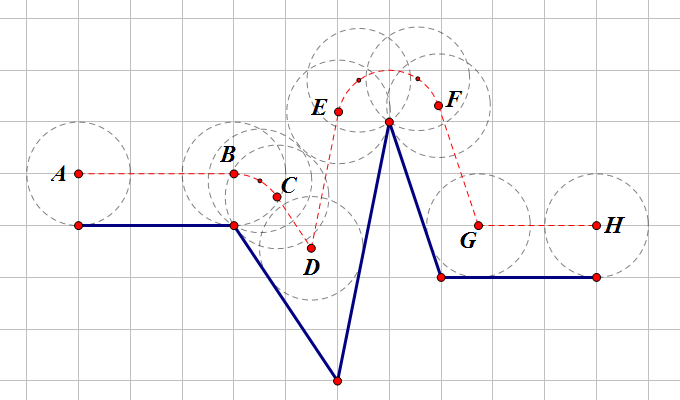
}

return vis[code] = 1;

}

**历届试题 车轮轴迹**

问题描述

　　栋栋每天骑自行车回家需要经过一条狭长的林荫道。道路由于年久失修，变得非常不平整。虽然栋栋每次都很颠簸，但他仍把骑车经过林荫道当成一种乐趣。  
　　由于颠簸，栋栋骑车回家的路径是一条上下起伏的曲线，栋栋想知道，他回家的这条曲线的长度究竟是多长呢？更准确的，栋栋想知道从林荫道的起点到林荫道的终点，他的车前轮的轴（圆心）经过的路径的长度。  
　　栋栋对路面进行了测量。他把道路简化成一条条长短不等的直线段，这些直线段首尾相连，且位于同一平面内。并在该平面内建立了一个直角坐标系，把所有线段的端点坐标都计算好。  
　　假设栋栋的自行车在行进的过程中前轮一直是贴着路面前进的。  
  
  
　　上图给出了一个简单的路面的例子，其中蓝色实线为路面，红色虚线为车轮轴经过的路径。在这个例子中，栋栋的前轮轴从A点出发，水平走到B点，然后绕着地面的F点到C点（绕出一个圆弧），再沿直线下坡到D点，最后水平走到E点，在这个图中地面的坐标依次为：(0, 0), (2, 0), (4, -1), (6, -1)，前轮半径为1.50，前轮轴前进的距离依次为：  
　　AB=2.0000；弧长BC=0.6955；CD=1.8820；DE=1.6459。  
　　总长度为6.2233。  
  
　　下图给出了一个较为复杂的路面的例子，在这个例子中，车轮在第一个下坡还没下完时（D点）就开始上坡了，之后在坡的顶点要从E绕一个较大的圆弧到F点。这个图中前轮的半径为1，每一段的长度依次为：  
　　AB=3.0000；弧长BC=0.9828；CD=1.1913；DE=2.6848；弧长EF=2.6224； FG=2.4415；GH=2.2792。  
　　总长度为15.2021。  
  
　　现在给出了车轮的半径和路面的描述，请求出车轮轴轨迹的总长度。

输入格式

　　输入的第一行包含一个整数n和一个实数r，用一个空格分隔，表示描述路面的坐标点数和车轮的半径。  
　　接下来n行，每个包含两个实数，其中第i行的两个实数x[i], y[i]表示描述路面的第i个点的坐标。  
　　路面定义为所有路面坐标点顺次连接起来的折线。给定的路面的一定满足以下性质：  
  
　　\*第一个坐标点一定是(0, 0)；  
　　\*第一个点和第二个点的纵坐标相同；  
　　\*倒数第一个点和倒数第二个点的纵坐标相同；  
　　\*第一个点和第二个点的距离不少于车轮半径；  
　　\*倒数第一个点和倒数第二个点的的距离不少于车轮半径；  
　　\*后一个坐标点的横坐标大于前一个坐标点的横坐标，即对于所有的i，x[i+1]>x[i]。

输出格式

　　输出一个实数，四舍五入保留两个小数，表示车轮轴经过的总长度。  
　　你的结果必须和参考答案一模一样才能得分。数据保证答案精确值的小数点后第三位不是4或5。

样例输入

4 1.50  
0.00 0.00  
2.00 0.00  
4.00 -1.00  
6.00 -1.00

样例输出

6.22

样例说明

　　这个样例对应第一个图。

样例输入

6 1.00  
0.00 0.00  
3.00 0.00  
5.00 -3.00  
6.00 2.00  
7.00 -1.00  
10.00 -1.00

样例输出

15.20

样例说明

　　这个样例对应第二个图

数据规模和约定

　　对于20%的数据，n=4；  
　　对于40%的数据，n≤10；  
　　对于100%的数据，4≤n≤100，0.5≤r≤20.0，x[i] ≤2000.0，-2000.0≤y[i] ≤2000.0。

本题的C++参考代码如下：

#include <iostream>

#include <cstdlib>

#include <cstdio>

#include <cstring>

#include <vector>

#include <cmath>

#include <algorithm>

using namespace std;

const int MAXN = 10000;

const double PI = atan(1.0) \* 4;

const double EPS = 1e-10;

class Point {

public:

double x, y;

Point() {}

Point(double x, double y) : x(x), y(y) {}

Point operator - (const Point &r) const { return Point(x-r.x, y-r.y); }

Point operator + (const Point &r) const { return Point(x+r.x, y+r.y); }

Point &operator += (const Point &r) { x += r.x; y += r.y; return \*this; }

Point &operator \*= (double m) { x \*= m; y \*= m; return \*this; }

Point pOfRotate(double angle) const {

double cosA = cos(angle);

double sinA = sin(angle);

return Point(cosA\*x-sinA\*y, sinA\*x+cosA\*y);

}

Point pOfRotate90() const { return Point(-y, x); }

double length() const { return sqrt(x\*x+y\*y); }

Point pOfNormal() const {

double len = length();

return Point(x/len, y/len);

}

double angle() const { return atan2(y, x); }

};

ostream & operator <<(ostream &os, const Point &v)

{

os << "(" << v.x << "," << v.y << ")";

return os;

}

class Segment;

class Circle;

class Seg {

public:

virtual double getLeft() const = 0;

virtual double getRight() const = 0;

virtual double getY(double x) const = 0;

virtual double getLength(double x1, double x2) const = 0;

virtual void intersect(Seg \*r) const = 0;

virtual void intersect(const Segment &v) const = 0;

virtual void intersect(const Circle &v) const = 0;

bool contains(double x) const { return x>=getLeft() && x<=getRight(); }

virtual void acceptPrint(ostream &os) const = 0;

};

ostream & operator <<(ostream &os, const Seg &v)

{

v.acceptPrint(os);

return os;

}

Point intersectRet[4];

int tIntersectRet;

class Segment : public Seg {

public:

Point a, b;

Segment &moveLeft(double dis)

{

Point tmp = ((b-a).pOfRotate90().pOfNormal() \*= dis);

a += tmp;

b += tmp;

return \*this;

}

virtual double getLeft() const { return a.x; }

virtual double getRight() const { return b.x; }

virtual double getY(double x) const {

return (x-a.x)\*(b.y-a.y)/(b.x-a.x)+a.y;

}

virtual double getLength(double x1, double x2) const {

return (x2-x1) \* (b-a).length() / (b.x-a.x);

}

virtual void intersect(Seg \*r) const {

r->intersect(\*this);

}

virtual void intersect(const Segment &v) const {

tIntersectRet = 0;

double ang = (b-a).angle();

Point c = (v.a-a).pOfRotate(-ang);

Point d = (v.b-a).pOfRotate(-ang);

// Bug

//double di = b.length();

double di = (b-a).length();

if (!((c.y>0&&d.y<0) || (c.y<0&&d.y>0)))

return ;

double x = (d.x-c.x) \* (-c.y) / (d.y-c.y) + c.x;

if (x<0 || x>di)

return ;

Point ret = Point(x,0).pOfRotate(ang)+a;

intersectRet[tIntersectRet++] = ret;

}

virtual void intersect(const Circle &v) const;

virtual void acceptPrint(ostream &os) const {

os << a << "-" << b;

}

};

class Circle : public Seg {

public:

Point c;

double r;

virtual double getLeft() const { return c.x - r; }

virtual double getRight() const { return c.x + r; }

virtual double getY(double x) const {

double y2 = r \* r - (c.x - x) \* (c.x - x);

if (y2<0) y2 = 0;

return c.y + sqrt(y2);

}

virtual double getLength(double x1, double x2) const {

x1 -= c.x; x2 -= c.x;

double a1 = Point(x1, sqrt(abs(r\*r-x1\*x1))).angle(), a2 = Point(x2, sqrt(abs(r\*r-x2\*x2))).angle();

return (a1-a2) \* r;

}

virtual void intersect(Seg \*r) const {

r->intersect(\*this);

}

virtual void intersect(const Segment &v) const {

tIntersectRet = 0;

Point a = v.a - c;

Point b = v.b - c;

double ang = (b-a).angle();

Point nA = a.pOfRotate(-ang);

Point nB = b.pOfRotate(-ang);

double y = nA.y;

if (y>r || y<-r)

return ;

double x = sqrt(r\*r - y\*y);

if (x>=nA.x && x<=nB.x)

intersectRet[tIntersectRet++] = Point(x, y).pOfRotate(ang) + c;

if (-x>=nA.x && -x<=nB.x)

intersectRet[tIntersectRet++] = Point(-x, y).pOfRotate(ang) + c;

}

virtual void intersect(const Circle &v) const {

tIntersectRet = 0;

Point p = v.c - c;

double d = p.length();

if (d > r + v.r || d==0)

return ;

double x = (r\*r - v.r\*v.r + d\*d) / (2\*d);

if (x <= r)

{

double y = sqrt(abs(r\*r - x\*x));

double ang = p.angle();

intersectRet[tIntersectRet++] = Point(x,y).pOfRotate(ang) + c;

intersectRet[tIntersectRet++] = Point(x,-y).pOfRotate(ang) + c;

}

}

virtual void acceptPrint(ostream &os) const {

os << c << "," << r;

}

};

void Segment::intersect(const Circle &v) const {

v.intersect(\*this);

}

int n;

Point inps[MAXN];

vector<Seg \*> segs;

vector<double> spes;

double radius = 1;

void input()

{

scanf("%d%lf", &n, &radius);

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

{

double x, y;

scanf("%lf%lf", &x, &y);

inps[i] = Point(x, y);

}

}

void process()

{

segs.clear();

spes.clear();

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

{

Circle \*tmp = new Circle;

tmp->c = inps[i];

tmp->r = radius;

segs.push\_back(tmp);

}

for (int i = 0; i + 1 < n; ++i)

{

Segment \*tmp = new Segment;

tmp->a = inps[i];

tmp->b = inps[i+1];

tmp->moveLeft(radius);

segs.push\_back(tmp);

}

for (int i = 0; i < (int)segs.size(); ++i)

{

spes.push\_back(segs[i]->getLeft());

spes.push\_back(segs[i]->getRight());

}

for (int i = 0; i < (int)segs.size(); ++i)

{

for (int j = i+1; j < (int)segs.size(); ++j)

{

segs[i]->intersect(segs[j]);

if (tIntersectRet > 0)

{

for (int id = 0; id < tIntersectRet; ++id)

{

//cout << \*segs[i] << " " << \*segs[j] << " : " << intersectRet[id] << endl;

spes.push\_back(intersectRet[id].x);

}

}

}

}

sort(spes.begin(), spes.end());

double pre = spes[0];

const double NONE = 1e30;

double preEnd = NONE;

double totalLen = 0;

for (int i = 1; i < (int)spes.size(); ++i)

{

if (spes[i]-pre < EPS)

continue;

double cur = (pre+spes[i]) / 2;

//cout << "Processing " << cur << " from " << pre << " to " << spes[i] << endl;

if (cur>=inps[0].x && cur<=inps[n-1].x)

{

double MY = -NONE;

int who;

for (int j = 0; j < (int)segs.size(); ++j)

{

if (!segs[j]->contains(cur))

continue;

double y = segs[j]->getY(cur);

if (y > MY)

{

MY = y;

who = j;

}

}

if (preEnd != NONE)

{

double LY = segs[who]->getY(pre);

//cout << "Drop info " << \*segs[who] << " " << "[" << pre << "]" << endl;

totalLen += abs(preEnd-LY);

//cout << "Pre drop = " << abs(preEnd-LY) << " from " << preEnd << " to " << LY << endl;

}

double len = segs[who]->getLength(pre, spes[i]);

if (len < 0)

printf("Error!\n");

//cout << "Curlen = " << len << " from " << pre << " to " << spes[i] << endl;

totalLen += len;

preEnd = segs[who]->getY(spes[i]);

}

pre = spes[i];

}

printf("%0.2lf\n", totalLen);

for (int i = 0; i < (int)segs.size(); ++i)

delete segs[i];

segs.clear();

}

int main()

{

input();

process();

return 0;

}

本题的C参考代码如下：

该题暂时没有人完全正确，暂时没有该语言的参考程序。

**历届试题 约数倍数选卡片**

问题描述

　　闲暇时，福尔摩斯和华生玩一个游戏：  
　　在N张卡片上写有N个整数。两人轮流拿走一张卡片。要求下一个人拿的数字一定是前一个人拿的数字的约数或倍数。例如，某次福尔摩斯拿走的卡片上写着数字“6”，则接下来华生可以拿的数字包括：  
　　1，2，3, 6，12，18，24 ....  
　　当轮到某一方拿卡片时，没有满足要求的卡片可选，则该方为输方。  
　　请你利用计算机的优势计算一下，在已知所有卡片上的数字和可选哪些数字的条件下，怎样选择才能保证必胜！  
　　当选多个数字都可以必胜时，输出其中最小的数字。如果无论如何都会输，则输出-1。

输入格式

　　输入数据为2行。第一行是若干空格分开的整数（每个整数介于1~100间），表示当前剩余的所有卡片。  
　　第二行也是若干空格分开的整数，表示可以选的数字。当然，第二行的数字必须完全包含在第一行的数字中。

输出格式

　　程序则输出必胜的招法！！

样例输入

2 3 6  
3 6

样例输出

3

样例输入

1 2 2 3 3 4 5  
3 4 5

样例输出

4

本题的C++参考代码如下：

#include <cstdio>

#include <vector>

#include <algorithm>

using namespace std;

const int MAXN = 101;

int cnt[MAXN] = {0};

int to[102];

bool appeared[MAXN] = {0};

vector <int> tbl[MAXN];

vector <int> choice;

bool check(int tar)

{

for(int i = tbl[tar].size()-1; i >= 0; i--)

{

int &j = tbl[tar][i];

if(cnt[j] > 0)

{

cnt[j]--;

if(check(j))

{

cnt[j]++;

return false;

}

cnt[j]++;

}

}

return true;

}

int main()

{

int num;

do

{

scanf("%d", &num);

cnt[num]++;

}

while(getchar()!='\n');

do

{

scanf("%d", &num);

if (!appeared[num])

{

choice.push\_back(num);

appeared[num] = true;

}

}

while(getchar()!='\n');

for(int i = 1; i < MAXN; i++)

{

if(cnt[i] > 0)

{

for(int j = 1; j < MAXN; j++)

{

if( cnt[j]>0 && (i%j==0 || j%i==0) )

{

tbl[i].push\_back(j);

}

}

}

}

sort(choice.begin(), choice.end());

int rec = -1;

for (int i = 0; i < choice.size(); i++)

{

int &j = choice[i];

cnt[j]--;

int k;

for(k=1;k<=100;k++)

to[k]=cnt[k];

for(k=1;k<=100;k++)

{

if(cnt[k]%2)cnt[k]=1;

else cnt[k]=0;

}

if(check(j))

{

rec = j;

break;

}

for(k=1;k<=100;k++)

cnt[k]=to[k];

cnt[j]++;

}

if(rec != -1)

printf("%d",rec);

else

puts("-1");

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int cnt[101];

int table[101][100];

int end[101];

int win(int hold)

{

int i,j;

for(i=end[hold];i>=0;i--)

{

j=table[hold][i];

if(cnt[j]>0)

{

cnt[j]--;

if(win(j))

{

cnt[j]++;

return 0;

}

cnt[j]++;

}

}

return 1;

}

int main()

{

int temp,i,j,t;

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

scanf("%d",&temp);

cnt[temp]++;

while(getchar()!='\n')

{

scanf("%d",&temp);

cnt[temp]++;

}

for(i=1;i<101;i++)

{

if(cnt[i]>0)

{

t=0;

for(j=1;j<=100;j++)

{

if(cnt[j]>0&&(i%j==0||j%i==0))

{

table[i][t]=j;

t++;

}

}

end[i]=t-1;

}

}

int save[101];

i=0;

do

{

scanf("%d",&temp);

save[i]=temp;

i++;

}while(getchar()!='\n');

i--;

int flag=0;

int tempi=i;

i=0;

while(i<=tempi)

{

cnt[save[i]]--;

if(win(save[i]))

{

flag=1;

break;

}

cnt[save[i]]++;

i++;

}

if(flag==1)

printf("%d",save[i]);

else

puts("-1");

system("pause");

return 0;

}

**历届试题 农场阳光**

问题描述

　　X星球十分特殊，它的自转速度与公转速度相同，所以阳光总是以固定的角度照射。  
　　最近，X星球为发展星际旅游业，把空间位置出租给Y国游客来晒太阳。每个租位是漂浮在空中的圆盘形彩云（圆盘与地面平行）。当然，这会遮挡住部分阳光，被遮挡的土地植物无法生长。  
　　本题的任务是计算某个农场宜于作物生长的土地面积有多大。

输入格式

　　输入数据的第一行包含两个整数a, b，表示某农场的长和宽分别是a和b，此时，该农场的范围是由坐标(0, 0, 0), (a, 0, 0), (a, b, 0), (0, b, 0)围成的矩形区域。  
　　第二行包含一个实数g，表示阳光照射的角度。简单起见，我们假设阳光光线是垂直于农场的宽的，此时正好和农场的长的夹角是g度，此时，空间中的一点(x, y, z)在地面的投影点应该是(x + z \* ctg(g度), y, 0)，其中ctg(g度)表示g度对应的余切值。  
　　第三行包含一个非负整数n，表示空中租位个数。  
　　接下来 n 行，描述每个租位。其中第i行包含4个整数xi, yi, zi, ri，表示第i个租位彩云的圆心在(xi, yi, zi)位置，圆半径为ri。

输出格式

　　要求输出一个实数，四舍五入保留两位有效数字，表示农场里能长庄稼的土地的面积。

样例输入

10 10  
90.0  
1  
5 5 10 5

样例输出

21.46

样例输入

8 8  
90.0  
1  
4 4 10 5

样例输出

1.81

样例输入

20 10  
45.0  
2  
5 0 5 5  
8 6 14 6

样例输出

130.15

本题的C++参考代码如下：

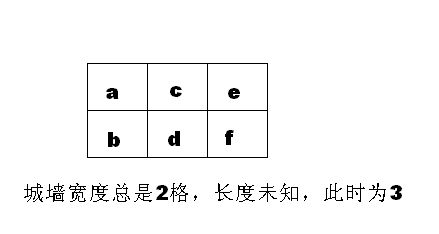
该题暂时没有人完全正确，暂时没有该语言的参考程序

本题的C参考代码如下：

该题暂时没有人完全正确，暂时没有该语言的参考程序。

**历届试题 格子刷油漆**

问题描述

　　X国的一段古城墙的顶端可以看成 2\*N个格子组成的矩形（如下图所示），现需要把这些格子刷上保护漆。  
  
  
　　你可以从任意一个格子刷起，刷完一格，可以移动到和它相邻的格子（对角相邻也算数），但不能移动到较远的格子（因为油漆未干不能踩！）  
　　比如：a d b c e f 就是合格的刷漆顺序。  
　　c e f d a b 是另一种合适的方案。  
　　当已知 N 时，求总的方案数。当N较大时，结果会迅速增大，请把结果对 1000000007 (十亿零七) 取模。

输入格式

　　输入数据为一个正整数（不大于1000）

输出格式

　　输出数据为一个正整数。

样例输入

2

样例输出

24

样例输入

3

样例输出

96

样例输入

22

样例输出

359635897

本题的C++参考代码如下：

#include<cstdio>

#define MAXN 1010

#define MOD 1000000007

int main()

{

long long DP[5][MAXN],ans=2;

int n;

scanf("%d",&n);

if(n>1){

DP[0][2]=DP[2][2]=DP[3][2]=DP[4][2]=4;

DP[1][2]=8;

for(int i=3;i<=n;i++){

DP[0][i]=(DP[0][i-1]\*2+DP[1][i-1]\*2)%MOD;

DP[1][i]=(DP[1][i-1]\*2+DP[2][i-1]\*4+DP[3][i-1]\*2+DP[4][i-1]\*2)%MOD;

DP[2][i]=DP[2][i-1]\*2%MOD;

DP[3][i]=DP[1][i-1];

DP[4][i]=DP[1][i-1];

}

ans=0;

for(int i=0;i<5;i++)ans+=DP[i][n];

ans%=MOD;

}

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

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

long long a[1001],b[1001],sum;

#define NUM 1000000007

int main()

{

int i,n;

scanf("%d",&n);

b[1]=1;

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

b[i]=(b[i-1]\*2%NUM);

a[1]=1;a[2]=6;

for (i=3;i<=n;i++)

a[i]=(2\*a[i-1]+b[i]+4\*a[i-2])%NUM;

sum=4\*a[n];

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

sum=((sum+8\*b[n-i]\*a[i-1]%NUM)%NUM+(8\*a[n-i]\*b[i-1])%NUM)%NUM;

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

return 0;

}**历届试题 高僧斗法**

问题描述

　　古时丧葬活动中经常请高僧做法事。仪式结束后，有时会有“高僧斗法”的趣味节目，以舒缓压抑的气氛。  
　　节目大略步骤为：先用粮食（一般是稻米）在地上“画”出若干级台阶（表示N级浮屠）。又有若干小和尚随机地“站”在某个台阶上。最高一级台阶必须站人，其它任意。(如图1所示)  
　　两位参加游戏的法师分别指挥某个小和尚向上走任意多级的台阶，但会被站在高级台阶上的小和尚阻挡，不能越过。两个小和尚也不能站在同一台阶，也不能向低级台阶移动。  
　　两法师轮流发出指令，最后所有小和尚必然会都挤在高段台阶，再也不能向上移动。轮到哪个法师指挥时无法继续移动，则游戏结束，该法师认输。  
　　对于已知的台阶数和小和尚的分布位置，请你计算先发指令的法师该如何决策才能保证胜出。

输入格式

　　输入数据为一行用空格分开的N个整数，表示小和尚的位置。台阶序号从1算起，所以最后一个小和尚的位置即是台阶的总数。（N<100, 台阶总数<1000）

输出格式

　　输出为一行用空格分开的两个整数: A B, 表示把A位置的小和尚移动到B位置。若有多个解，输出A值较小的解，若无解则输出-1。

样例输入

1 5 9

样例输出

1 4

样例输入

1 5 8 10

样例输出

1 3

本题的C++参考代码如下：

//2014年1月12日12:29:37

#include <stdio.h>

#include <stdlib.h>

#define N 1050

int a[N]; //存放输入数据

int s[N]; //虚拟石子

int total = 0;

int flag= 0;

int i,j;

int main()

{

do

{

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

if(total%2==0)

s[total/2] = a[total-1]-a[total-2]-1;

}while(getchar()!='\n');

for(i=1;i<=total/2;i++)

{

flag^=s[i];

}

//一开始就处于N态

if(!flag) printf("-1");

else

{

for(i=1;i<=total/2;i++)//从最小的开始遍历

{

int temp = s[i];

//左边上移减少数量

while(s[i]--)

{

flag = 0;

for(j=1;j<=total/2;j++) //再检测

flag^=s[j];

//把对手变成p态!

if(flag==0)

{

//第i个移动i步(移动到哪里)

//应该是从哪个位置移到哪个位置 跟哪个小和尚没关系

printf("%d %d\n",a[i\*2-2],a[i\*2-2]+temp-s[i]);

exit(0);

}

}

s[i] = temp;

//右边上移增加数量

while(s[i]-temp <a[i\*2]-a[2\*i-1]-1)

{

s[i]++;

flag = 0;

for(j=1;j<=total/2;j++) //再检测

flag^=s[j];

if(flag==0)

{

printf("%d %d\n",a[i\*2-1],a[i\*2-1]+s[i]-temp);

exit(0);

}

}

s[i]=temp;

}

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h> //参考蓝桥杯贴吧 dezhonger

int main()

{

int a[105],b[105],i=0,j,k,count,sum;

char c;

while(1)

{

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

if(c=='\n')

break;

}

count=i;

for(i = 0;i < count-1;i++)

b[i]=a[i+1]-a[i]-1;

b[count-1]=0;

sum=b[0];

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

sum^=b[i];

if(sum == 0)

printf("-1\n");

else

{

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

for(j = 1;j <= b[i];j++)

{

b[i] -= j;

if(i!=0)

b[i-1]+=j;

sum = b[0];

for(k = 2;k < count;k = k+2)

sum ^= b[k];

if(sum == 0)

{

printf("%d %d\n",a[i],a[i]+j);

break;

}

b[i] += j;

if(i != 0)

b[i-1] -= j;

}

}

return 0;

}

/\*

1 3 5 7 12 14 17 26 38 45 66 100

66 84

\*/

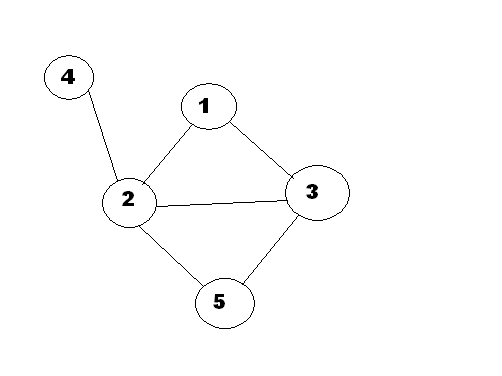
**历届试题 网络寻路**

问题描述

X 国的一个网络使用若干条线路连接若干个节点。节点间的通信是双向的。某重要数据包，为了安全起见，必须恰好被转发两次到达目的地。该包可能在任意一个节点产生，我们需要知道该网络中一共有多少种不同的转发路径。

源地址和目标地址可以相同，但中间节点必须不同。

如下图所示的网络。



1 -> 2 -> 3 -> 1 是允许的

1 -> 2 -> 1 -> 2 或者 1 -> 2 -> 3 -> 2 都是非法的。

输入格式

输入数据的第一行为两个整数N M，分别表示节点个数和连接线路的条数(1<=N<=10000; 0<=M<=100000)。

接下去有M行，每行为两个整数 u 和 v，表示节点u 和 v 联通(1<=u,v<=N , u!=v)。

输入数据保证任意两点最多只有一条边连接，并且没有自己连自己的边，即不存在重边和自环。

输出格式

输出一个整数，表示满足要求的路径条数。

样例输入1

3 3  
1 2  
2 3  
1 3

样例输出1

6

样例输入2

4 4  
1 2  
2 3  
3 1  
1 4

样例输出2

10

本题的C++参考代码如下：

#include<cstdio>

#include<cstring>

#define MAXN 10010

#define MAXM 100010

int Du[MAXN],U[MAXM],V[MAXM];

int main()

{

int n,m;

long long ans=0;

scanf("%d%d",&n,&m);

memset(Du,0,sizeof(Du));

for(int i=0;i<m;i++){

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

Du[U[i]]++;

Du[V[i]]++;

}

for(int i=0;i<m;i++)if(Du[U[i]]>1&&Du[V[i]]>1)ans+=(Du[U[i]]-1)\*(Du[V[i]]-1)\*2;

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

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#define MAXN 10001

#define MAXM 100001

int main()

{

int i, j;

int n, m;

int edge[MAXN]={0}, u[MAXM]={0}, v[MAXM]={0};

int ans = 0;

scanf("%d%d", &n, &m);

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

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

edge[u[i]] ++;

edge[v[i]] ++;

}

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

if(edge[u[i]]>1 && edge[v[i]]>1){

ans += (edge[u[i]]-1) \* (edge[v[i]]-1) \* 2;

}

}

printf("%d", ans);

return 0;

}

**历届试题 危险系数**

问题描述

抗日战争时期，冀中平原的地道战曾发挥重要作用。

地道的多个站点间有通道连接，形成了庞大的网络。但也有隐患，当敌人发现了某个站点后，其它站点间可能因此会失去联系。

我们来定义一个危险系数DF(x,y)：

对于两个站点x和y (x != y), 如果能找到一个站点z，当z被敌人破坏后，x和y不连通，那么我们称z为关于x,y的关键点。相应的，对于任意一对站点x和y，危险系数DF(x,y)就表示为这两点之间的关键点个数。

本题的任务是：已知网络结构，求两站点之间的危险系数。

输入格式

输入数据第一行包含2个整数n(2 <= n <= 1000), m(0 <= m <= 2000),分别代表站点数，通道数；

接下来m行，每行两个整数 u,v (1 <= u, v <= n; u != v)代表一条通道；

最后1行，两个数u,v，代表询问两点之间的危险系数DF(u, v)。

输出格式

一个整数，如果询问的两点不连通则输出-1.

样例输入

7 6  
1 3  
2 3  
3 4  
3 5  
4 5  
5 6  
1 6

样例输出

2

本题的C++参考代码如下：

#include <iostream>

#include <cmath>

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <map>

#include <set>

#include <list>

#include <queue>

#include <stack>

#include <string>

#include <vector>

#include <iomanip>

#include <algorithm>

using namespace std;

typedef long long LL;

#define clr(x, y) memset(x, y, sizeof(x))

#define sf scanf

#define pf printf

const double pi = 4.0 \* atan(1.0);

const double eps = 1e-8;

const int INF = 1 << 29;

const int maxn = 1005;

int N, M, roadNum, vis[maxn], mark[maxn];

vector<int> edge[maxn];

void init() {

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

edge[i].clear();

}

roadNum = 0;

clr(vis, 0), clr(mark, 0);

}

void dfs(int u, int tar) {

vis[u] = 1;

if (u == tar) {

roadNum++;

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

if (vis[i]) mark[i]++;

}

return ;

}

for (int i = 0; i < edge[u].size(); i++) {

int v = edge[u][i];

if (vis[v]) continue;

dfs(v, tar);

vis[v] = false;

}

}

int solve() {

init();

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

int u, v; sf("%d%d",&u,&v);

edge[u].push\_back(v);

edge[v].push\_back(u);

}

int start, end;

sf("%d%d",&start,&end);

dfs(start, end);

int ans = 0;

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

if (mark[i] == roadNum && i != start && i != end) {

ans++;

}

}

return roadNum == 0 ? -1 : ans;

}

int main() {

while (~sf("%d%d",&N,&M)) {

pf("%d\n", solve());

}

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*pNext;

};

struct Node tab[1001];

int visit[1001]={0};

int way[1001]={0};

int count[1001]={0};

int cnt=0;

void Insert(int n,int x);

void Init(int n);

void dfs(int x,int y,int n);

int fun(int n);

int main()

{

int x,y,n,m,u,v;

scanf("%d%d",&n,&m);

Init(n);

while(m--)

{

scanf("%d%d",&u,&v);

Insert(u,v);

Insert(v,u);

}

scanf("%d%d",&x,&y);

dfs(x,y,0);

int ret=fun(n);

printf("%d\n",ret);

return 0;

}

int fun(int n)

{

int i;

int ret=0;

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

{

if(count[i]==cnt)

{

ret++;

}

}

return (ret-2);

}

void dfs(int x,int y,int n)

{

visit[x]=1;

way[n]=x;

struct Node \*p=&tab[x];

if(x==y)

{

int i;

cnt++;

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

{

count[way[i]]++;

}

return ;

}

while((p=p->pNext)!=NULL)

{

if(visit[p->data]!=1)

{

dfs(p->data,y,n+1);

visit[p->data]=0;

}

}

}

void Init(int n)

{

int i;

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

{

tab[i].data=i;

tab[i].pNext=NULL;

}

}

void Insert(int n,int x)

{

struct Node \*p=&tab[n];

while(p->pNext!=NULL)

{

p=p->pNext;

}

struct Node \*new=(struct Node \*)malloc(sizeof(struct Node));

p->pNext=new;

new->data=x;

new->pNext=NULL;

}

**历届试题 横向打印二叉树**

问题描述

二叉树可以用于排序。其原理很简单：对于一个排序二叉树添加新节点时，先与根节点比较，若小则交给左子树继续处理，否则交给右子树。

当遇到空子树时，则把该节点放入那个位置。

比如，10 8 5 7 12 4 的输入顺序，应该建成二叉树如下图所示，其中.表示空白。

...|-12  
10-|  
...|-8-|  
.......|...|-7  
.......|-5-|  
...........|-4

本题目要求：根据已知的数字，建立排序二叉树，并在标准输出中横向打印该二叉树。

输入格式

输入数据为一行空格分开的N个整数。 N<100，每个数字不超过10000。

输入数据中没有重复的数字。

输出格式

输出该排序二叉树的横向表示。为了便于评卷程序比对空格的数目，请把空格用句点代替：

样例输入1

10 5 20

样例输出1

...|-20  
10-|  
...|-5

样例输入2

5 10 20 8 4 7

样例输出2

.......|-20  
..|-10-|  
..|....|-8-|  
..|........|-7  
5-|  
..|-4

 本题的C++参考代码如下：

#include <iostream>

using namespace std;

#include <stdio.h>

#include <string.h>

#include <malloc.h>

#include <vector>

#include <algorithm>

#define N 100

typedef struct BiTNode//二叉树结点

{

int data;//数据

struct BiTNode \*lchild,\*rchild;//左右孩子指针

}BiTNode,\*BiTree;

void CreateBalanceTree(BiTree &T,int x)//x 插入的数据

{

if(T == NULL)//若当前树为空

{

T = (BiTree)malloc(sizeof(BiTNode));

T->data = x;

T->lchild = NULL;

T->rchild = NULL;

}

else if(x < T->data)//如果比当前结点小，插入左子树

{

CreateBalanceTree(T->lchild,x);

}

else if(x > T->data)//如果比当前结点大，插入右子树

{

CreateBalanceTree(T->rchild,x);

}

}

vector<string> InOrder(BiTree T,int level,int lr)//右根左的顺序进行遍历,同时计算层次

{

int d,i,len;//

vector<string> v,vr,vl;

string s="";

string pre="";

string t="";

char c[5];

sprintf(c, "%d", T->data);

s=c;

len=s.length()+2;

if(level==1) len-=2;

if(level>1) s="|-"+s;//除了根结点外,左边都加|-

if(T->lchild||T->rchild) s+="-|";//如果结点有孩子

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

{

pre+='.';

}

if(T->rchild != NULL)

{

vr=InOrder(T->rchild,level+1,1);

}

if(T->lchild != NULL)

{

vl=InOrder(T->lchild,level+1,-1);

}

if(lr<0)

{

for(i=0;i<vr.size();i++)

{

vr[i]="|"+pre+vr[i];

}

for(i=0;i<vl.size();i++)

{

vl[i]="."+pre+vl[i];

}

}

if(lr>0)

{

for(i=0;i<vr.size();i++)

{

vr[i]="."+pre+vr[i];

}

for(i=0;i<vl.size();i++)

{

vl[i]="|"+pre+vl[i];

}

}

if(lr==0)

{

for(i=0;i<vr.size();i++)

{

vr[i]="."+pre+vr[i];

}

for(i=0;i<vl.size();i++)

{

vl[i]="."+pre+vl[i];

}

}

v.insert(v.end(),vr.begin(),vr.end());//右

v.insert(v.end(),s);//根

v.insert(v.end(),vl.begin(),vl.end());//左

return v;

}

void print(string s)

{

cout << s<< endl;

}

int main()

{

int n;

char str[500];//输入的字符串

char \*p=NULL;//读入的每个数字

BiTree T = NULL;

vector<string> v;

gets(str);

p= strtok(str, " ");

while ( p != NULL ) //创建二叉树

{

CreateBalanceTree(T,atoi(p));

p = strtok( NULL, " " );

}

v=InOrder(T,1,0);//根的层次为1

for\_each(v.begin(), v.end(), print);

printf("\n");

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct TNode

{

int key;

struct TNode \*left;

struct TNode \*right;

}TNode, \*Tree;

Tree insert(Tree root, Tree src)

{

if(root == NULL)

{

root = src;

}

else if(src->key > root->key)

{

root->left = insert(root->left, src);

}

else

{

root->right = insert(root->right, src);

}

return root;

}

char l[1000];

#define U 1

#define D 2

#define S ('.')

void print(Tree root, int s, int dir)

{

if(root != NULL)

{

int i;

char buf[10];

sprintf(buf, "|-%d-", root->key);

int len = strlen(buf);

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

{

l[s + i] = S;

}

if(dir == D)

{

l[s] = '|';

}

print(root->left, s + len, U);

l[s] = '\0';

if(root->left == NULL && root->right == NULL)

{

buf[len - 1] = '\0';

printf("%s%s\n", l, buf);

}

else

{

printf("%s%s|\n", l, buf);

}

l[s] = S;

if(dir == U)

{

l[s] = '|';

}

print(root->right, s + len, D);

l[s] = S;

}

}

void printPre(Tree root, int s)

{

if(root != NULL)

{

int i;

char buf[10];

sprintf(buf, "%d-", root->key);

int len = strlen(buf);

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

{

l[s + i] = S;

}

print(root->left, s + len, U);

printf("%s|\n", buf);

print(root->right, s + len, D);

}

}

int main(void)

{

int n;

Tree tree = NULL;

while(scanf("%d", &n) > 0)

{

Tree neo = malloc(sizeof(TNode));

neo->key = n;

neo->left = neo->right = NULL;

tree = insert(tree, neo);

}

printPre(tree, 0);

return 0;

}

**历届试题 幸运数**

问题描述

幸运数是波兰数学家乌拉姆命名的。它采用与生成素数类似的“筛法”生成

。

首先从1开始写出自然数1,2,3,4,5,6,....

1 就是第一个幸运数。

我们从2这个数开始。把所有序号能被2整除的项删除，变为：

1 \_ 3 \_ 5 \_ 7 \_ 9 ....

把它们缩紧，重新记序，为：

1 3 5 7 9 .... 。这时，3为第2个幸运数，然后把所有能被3整除的序号位置的数删去。注意，是序号位置，不是那个数本身能否被3整除!! 删除的应该是5，11, 17, ...

此时7为第3个幸运数，然后再删去序号位置能被7整除的(19,39,...)

最后剩下的序列类似：

1, 3, 7, 9, 13, 15, 21, 25, 31, 33, 37, 43, 49, 51, 63, 67, 69, 73, 75, 79, ...

输入格式

输入两个正整数m n, 用空格分开 (m < n < 1000\*1000)

输出格式

程序输出 位于m和n之间的幸运数的个数（不包含m和n）。

样例输入1

1 20

样例输出1

5

样例输入2

30 69

样例输出2

8

本题的C++参考代码如下：

#include <cstdio>

using namespace std;

int a[] = {1, 3, 7, 9, 13, 15, 21, 25, 31, 33, 37, 43, 49, 51, 63, 67, 69, 73, 75, 79, 87, 93, 99, 105, 111, 115, 127, 129, 133, 135, 141, 151, 159, 163, 169, 171, 189, 193, 195, 201, 205, 211, 219, 223, 231, 235, 237, 241, 259, 261, 267, 273, 283, 285, 289, 297, 303, 307, 319, 321, 327, 331, 339, 349, 357, 361, 367, 385, 391, 393, 399, 409, 415, 421, 427, 429, 433, 451, 463, 475, 477, 483, 487, 489, 495, 511, 517, 519, 529, 535, 537, 541, 553, 559, 577, 579, 583, 591, 601, 613, 615, 619, 621, 631, 639, 643, 645, 651, 655, 673, 679, 685, 693, 699, 717, 723, 727, 729, 735, 739, 741, 745, 769, 777, 781, 787, 801, 805, 819, 823, 831, 841, 855, 867, 873, 883, 885, 895, 897, 903, 925, 927, 931, 933, 937, 957, 961, 975, 979, 981, 991, 993, 997, 1009, 1011, 1021, 1023, 1029, 1039, 1041, 1053, 1057, 1087, 1093, 1095, 1101, 1105, 1107, 1117, 1123, 1147, 1155, 1167, 1179, 1183, 1189, 1197, 1201, 1203, 1209, 1219, 1231, 1233, 1245, 1249, 1251, 1261, 1263, 1275, 1281, 1285, 1291, 1303, 1309, 1323, 1329, 1339, 1357, 1365, 1369, 1387, 1389, 1395, 1401, 1417, 1419, 1435, 1441, 1455, 1459, 1471, 1473, 1485, 1491, 1495, 1497, 1501, 1503, 1519, 1533, 1543, 1545, 1563, 1567, 1575, 1579, 1585, 1587, 1597, 1599, 1611, 1639, 1641, 1645, 1659, 1663, 1675, 1693, 1701, 1705, 1711, 1723, 1731, 1737, 1749, 1765, 1767, 1771, 1773, 1777, 1797, 1801, 1809, 1819, 1827, 1831, 1833, 1839, 1857, 1869, 1879, 1893, 1899, 1915, 1921, 1923, 1933, 1941, 1945, 1959, 1963, 1965, 1977, 1983, 1987, 1995, 2001, 2019, 2023, 2031, 2053, 2059, 2065, 2067, 2079, 2083, 2085, 2095, 2113, 2115, 2121, 2125, 2127, 2133, 2163, 2173, 2187, 2209, 2211, 2215, 2217, 2221, 2239, 2251, 2253, 2257, 2271, 2277, 2281, 2283, 2301, 2311, 2317, 2323, 2335, 2343, 2355, 2365, 2379, 2395, 2403, 2407, 2409, 2415, 2419, 2427, 2439, 2445, 2461, 2467, 2473, 2479, 2491, 2493, 2505, 2511, 2523, 2527, 2545, 2557, 2563, 2571, 2575, 2587, 2589, 2593, 2599, 2607, 2625, 2635, 2647, 2649, 2653, 2661, 2667, 2671, 2689, 2697, 2715, 2725, 2755, 2763, 2773, 2781, 2785, 2787, 2797, 2815, 2817, 2821, 2823, 2827, 2835, 2841, 2845, 2851, 2877, 2887, 2899, 2901, 2905, 2913, 2923, 2943, 2953, 2961, 2971, 2973, 2977, 2983, 3003, 3007, 3027, 3031, 3037, 3039, 3049, 3055, 3073, 3075, 3091, 3097, 3099, 3109, 3111, 3121, 3123, 3133, 3153, 3163, 3171, 3175, 3183, 3187, 3199, 3213, 3223, 3229, 3235, 3243, 3259, 3261, 3289, 3297, 3301, 3307, 3313, 3325, 3339, 3351, 3355, 3363, 3381, 3403, 3405, 3409, 3411, 3427, 3433, 3439, 3451, 3453, 3465, 3477, 3481, 3487, 3489, 3495, 3499, 3507, 3559, 3565, 3571, 3579, 3595, 3597, 3603, 3607, 3613, 3621, 3625, 3633, 3655, 3661, 3663, 3669, 3675, 3685, 3687, 3697, 3709, 3717, 3721, 3727, 3747, 3753, 3763, 3771, 3781, 3789, 3793, 3795, 3811, 3843, 3847, 3849, 3865, 3873, 3879, 3889, 3891, 3897, 3909, 3915, 3931, 3943, 3951, 3955, 3969, 3975, 3981, 3991, 3999, 4003, 4015, 4023, 4033, 4035, 4041, 4045, 4063, 4069, 4081, 4095, 4105, 4107, 4129, 4131, 4161, 4165, 4173, 4179, 4189, 4195, 4201, 4203, 4207, 4227, 4237, 4251, 4255, 4257, 4263, 4269, 4285, 4287, 4315, 4321, 4329, 4335, 4363, 4377, 4383, 4389, 4399, 4413, 4431, 4441, 4443, 4455, 4461, 4465, 4483, 4485, 4495, 4509, 4519, 4521, 4539, 4551, 4561, 4567, 4569, 4573, 4587, 4609, 4611, 4621, 4623, 4629, 4645, 4647, 4653, 4663, 4671, 4675, 4695, 4699, 4713, 4717, 4725, 4741, 4761, 4767, 4773, 4797, 4801, 4809, 4813, 4819, 4833, 4837, 4839, 4843, 4851, 4863, 4867, 4881, 4887, 4893, 4929, 4951, 4963, 4965, 4969, 4977, 4987, 4989, 4993, 4999, 5001, 5007, 5019, 5029, 5041, 5043, 5049, 5053, 5089, 5103, 5127, 5137, 5139, 5149, 5151, 5157, 5169, 5179, 5181, 5191, 5211, 5217, 5229, 5233, 5235, 5253, 5259, 5277, 5283, 5293, 5295, 5299, 5325, 5335, 5341, 5343, 5371, 5377, 5379, 5385, 5409, 5419, 5427, 5433, 5449, 5455, 5463, 5473, 5487, 5491, 5503, 5515, 5527, 5547, 5551, 5559, 5569, 5577, 5587, 5589, 5593, 5599, 5613, 5617, 5637, 5641, 5649, 5655, 5661, 5671, 5673, 5679, 5691, 5701, 5707, 5713, 5719, 5737, 5755, 5763, 5767, 5769, 5803, 5809, 5817, 5827, 5833, 5839, 5851, 5869, 5883, 5889, 5893, 5901, 5905, 5911, 5913, 5923, 5959, 5965, 5967, 5971, 5973, 5977, 5991, 5997, 6009, 6019, 6031, 6049, 6055, 6061, 6079, 6093, 6111, 6115, 6123, 6141, 6147, 6159, 6163, 6175, 6177, 6195, 6211, 6229, 6237, 6243, 6249, 6253, 6271, 6273, 6279, 6301, 6309, 6331, 6345, 6351, 6355, 6363, 6367, 6369, 6373, 6379, 6385, 6399, 6411, 6415, 6427, 6433, 6435, 6447, 6463, 6471, 6475, 6477, 6501, 6505, 6523, 6531, 6535, 6541, 6553, 6559, 6567, 6573, 6601, 6621, 6625, 6631, 6661, 6663, 6667, 6669, 6679, 6687, 6693, 6715, 6723, 6733, 6741, 6745, 6747, 6753, 6757, 6763, 6765, 6783, 6787, 6789, 6841, 6849, 6867, 6871, 6883, 6891, 6909, 6915, 6921, 6931, 6933, 6937, 6951, 6981, 6985, 6999, 7003, 7009, 7035, 7041, 7045, 7047, 7069, 7077, 7081, 7087, 7101, 7111, 7129, 7135, 7153, 7167, 7171, 7173, 7183, 7191, 7195, 7197, 7207, 7215, 7231, 7233, 7237, 7245, 7249, 7279, 7293, 7299, 7311, 7321, 7333, 7339, 7341, 7345, 7357, 7359, 7371, 7377, 7395, 7401, 7405, 7419, 7435, 7437, 7443, 7447, 7459, 7471, 7489, 7501, 7503, 7507, 7531, 7533, 7549, 7551, 7563, 7567, 7575, 7585, 7591, 7593, 7603, 7609, 7629, 7633, 7639, 7645, 7677, 7687, 7689, 7701, 7711, 7717, 7737, 7755, 7779, 7791, 7795, 7801, 7803, 7813, 7827, 7833, 7837, 7855, 7881, 7885, 7897, 7899, 7909, 7917, 7921, 7929, 7939, 7947, 7951, 7963, 7969, 7971, 8001, 8005, 8007, 8013, 8037, 8047, 8071, 8073, 8085, 8089, 8107, 8109, 8127, 8131, 8139, 8151, 8161, 8169, 8173, 8175, 8191, 8193, 8221, 8223, 8227, 8233, 8253, 8257, 8263, 8269, 8283, 8289, 8299, 8325, 8331, 8347, 8359, 8365, 8367, 8379, 8409, 8413, 8421, 8445, 8449, 8467, 8473, 8475, 8479, 8487, 8493, 8515, 8535, 8539, 8551, 8553, 8575, 8577, 8583, 8589, 8601, 8605, 8611, 8617, 8635, 8637, 8641, 8647, 8655, 8661, 8673, 8683, 8719, 8757, 8769, 8773, 8787, 8793, 8805, 8809, 8827, 8829, 8833, 8835, 8841, 8869, 8871, 8895, 8913, 8931, 8937, 8947, 8955, 8961, 8977, 8979, 8989, 8995, 8997, 9009, 9031, 9033, 9061, 9063, 9073, 9075, 9081, 9085, 9115, 9117, 9121, 9123, 9135, 9139, 9145, 9151, 9177, 9181, 9211, 9213, 9223, 9231, 9235, 9249, 9253, 9267, 9273, 9277, 9285, 9291, 9303, 9315, 9331, 9339, 9349, 9373, 9387, 9399, 9403, 9409, 9421, 9423, 9429, 9441, 9451, 9457, 9471, 9475, 9481, 9483, 9487, 9501, 9535, 9537, 9543, 9547, 9549, 9555, 9561, 9567, 9613, 9621, 9625, 9631, 9643, 9649, 9661, 9663, 9675, 9687, 9691, 9703, 9727, 9733, 9751, 9753, 9775, 9787, 9789, 9795, 9801, 9807, 9811, 9813, 9837, 9841, 9883, 9895, 9915, 9927, 9937, 9957, 9961, 9979, 9985, 9987, 9997, 9999, 10003, 10009, 10017, 10041, 10051, 10059, 10069, 10071, 10083, 10087, 10089, 10095, 10111, 10117, 10125, 10131, 10143, 10147, 10149, 10153, 10173, 10191, 10195, 10197, 10221, 10239, 10255, 10261, 10275, 10279, 10291, 10311, 10315, 10317, 10321, 10341, 10363, 10365, 10387, 10399, 10411, 10417, 10431, 10441, 10443, 10459, 10461, 10471, 10501, 10507, 10509, 10525, 10531, 10533, 10555, 10563, 10569, 10575, 10585, 10587, 10597, 10599, 10605, 10609, 10635, 10651, 10653, 10659, 10671, 10683, 10723, 10725, 10731, 10747, 10753, 10755, 10759, 10773, 10785, 10789, 10797, 10809, 10843, 10849, 10873, 10881, 10891, 10903, 10909, 10911, 10921, 10923, 10945, 10953, 10965, 10975, 10977, 11005, 11011, 11017, 11047, 11055, 11059, 11073, 11077, 11089, 11091, 11097, 11113, 11137, 11157, 11163, 11173, 11181, 11193, 11197, 11205, 11217, 11223, 11227, 11239, 11245, 11247, 11259, 11263, 11269, 11283, 11289, 11301, 11323, 11341, 11343, 11371, 11373, 11377, 11389, 11391, 11395, 11407, 11419, 11425, 11427, 11437, 11439, 11475, 11479, 11487, 11491, 11521, 11529, 11535, 11539, 11541, 11577, 11581, 11583, 11599, 11617, 11623, 11629, 11641, 11643, 11659, 11667, 11677, 11679, 11707, 11721, 11725, 11731, 11769, 11781, 11797, 11803, 11811, 11815, 11823, 11827, 11833, 11835, 11845, 11847, 11853, 11865, 11877, 11887, 11911, 11923, 11929, 11943, 11953, 11959, 11961, 11991, 11995, 12007, 12015, 12025, 12049, 12057, 12063, 12069, 12079, 12097, 12103, 12117, 12121, 12133, 12139, 12141, 12151, 12159, 12163, 12165, 12193, 12201, 12207, 12223, 12237, 12243, 12265, 12267, 12285, 12295, 12301, 12307, 12315, 12319, 12321, 12333, 12337, 12357, 12363, 12373, 12379, 12391, 12399, 12411, 12417, 12427, 12435, 12457, 12481, 12487, 12495, 12499, 12519, 12543, 12547, 12553, 12567, 12579, 12589, 12607, 12609, 12621, 12631, 12645, 12649, 12651, 12673, 12679, 12699, 12709, 12711, 12729, 12735, 12741, 12759, 12763, 12777, 12781, 12799, 12801, 12813, 12819, 12861, 12867, 12877, 12885, 12903, 12907, 12925, 12937, 12951, 12957, 12961, 12969, 12991, 12999, 13009, 13011, 13021, 13027, 13029, 13053, 13057, 13063, 13077, 13083, 13113, 13117, 13119, 13125, 13129, 13135, 13141, 13167, 13177, 13179, 13203, 13209, 13215, 13219, 13243, 13245, 13267, 13273, 13279, 13281, 13285, 13293, 13309, 13317, 13323, 13327, 13335, 13345, 13347, 13359, 13363, 13371, 13387, 13419, 13441, 13443, 13465, 13473, 13485, 13489, 13513, 13515, 13519, 13527, 13531, 13533, 13537, 13551, 13567, 13575, 13581, 13587, 13599, 13609, 13633, 13657, 13677, 13681, 13693, 13695, 13707, 13723, 13725, 13737, 13741, 13743, 13759, 13801, 13803, 13821, 13831, 13833, 13849, 13851, 13861, 13873, 13893, 13897, 13929, 13947, 13953, 13957, 13995, 14007, 14017, 14031, 14035, 14037, 14041, 14049, 14053, 14059, 14073, 14079, 14091, 14095, 14113, 14115, 14119, 14137, 14163, 14179, 14181, 14185, 14199, 14205, 14209, 14211, 14217, 14229, 14245, 14247, 14253, 14281, 14287, 14293, 14307, 14313, 14317, 14331, 14347, 14349, 14353, 14367, 14395, 14407, 14419, 14427, 14437, 14439, 14443, 14449, 14457, 14461, 14473, 14491, 14493, 14505, 14521, 14523, 14539, 14541, 14565, 14587, 14595, 14599, 14601, 14607, 14625, 14661, 14667, 14689, 14691, 14713, 14715, 14721, 14731, 14749, 14751, 14755, 14775, 14779, 14785, 14793, 14797, 14817, 14841, 14875, 14881, 14883, 14899, 14905, 14919, 14923, 14931, 14935, 14943, 14947, 14973, 14977, 14997, 15003, 15007, 15009, 15025, 15031, 15049, 15057, 15063, 15069, 15073, 15081, 15103, 15109, 15121, 15129, 15133, 15141, 15165, 15169, 15175, 15183, 15187, 15189, 15193, 15205, 15213, 15217, 15229, 15231, 15235, 15237, 15247, 15255, 15259, 15261, 15271, 15291, 15301, 15313, 15333, 15355, 15379, 15381, 15385, 15397, 15417, 15445, 15457, 15469, 15471, 15481, 15487, 15489, 15511, 15519, 15523, 15531, 15543, 15547, 15567, 15573, 15577, 15585, 15597, 15609, 15645, 15655, 15661, 15673, 15675, 15693, 15697, 15717, 15735, 15763, 15771, 15783, 15793, 15805, 15817, 15819, 15835, 15843, 15847, 15855, 15879, 15883, 15891, 15901, 15907, 15919, 15927, 15939, 15955, 15969, 15975, 15981, 15993, 16003, 16009, 16023, 16045, 16047, 16053, 16057, 16077, 16081, 16095, 16101, 16135, 16153, 16191, 16195, 16197, 16203, 16213, 16227, 16237, 16245, 16255, 16267, 16275, 16279, 16287, 16297, 16309, 16329, 16333, 16341, 16353, 16359, 16365, 16381, 16387, 16389, 16393, 16395, 16413, 16417, 16429, 16431, 16447, 16449, 16467, 16473, 16479, 16491, 16507, 16521, 16539, 16557, 16573, 16591, 16611, 16615, 16617, 16621, 16635, 16639, 16657, 16665, 16677, 16701, 16707, 16725, 16747, 16765, 16771, 16773, 16791, 16795, 16825, 16827, 16845, 16863, 16867, 16893, 16915, 16917, 16921, 16933, 16939, 16947, 16953, 16957, 16959, 16969, 16971, 16981, 16995, 17001, 17011, 17017, 17019, 17035, 17047, 17053, 17061, 17083, 17085, 17103, 17115, 17119, 17125, 17127, 17151, 17157, 17173, 17181, 17209, 17211, 17215, 17221, 17223, 17229, 17251, 17253, 17283, 17287, 17305, 17313, 17329, 17335, 17341, 17355, 17359, 17367, 17389, 17391, 17401, 17409, 17431, 17433, 17455, 17463, 17467, 17473, 17485, 17497, 17517, 17527, 17535, 17545, 17547, 17563, 17565, 17581, 17589, 17619, 17625, 17671, 17677, 17689, 17695, 17703, 17713, 17725, 17733, 17737, 17749, 17767, 17769, 17775, 17781, 17787, 17797, 17811, 17829, 17851, 17875, 17883, 17893, 17913, 17925, 17935, 17943, 17947, 17965, 17977, 17991, 17997, 18001, 18007, 18019, 18025, 18043, 18049, 18055, 18067, 18085, 18087, 18123, 18127, 18145, 18147, 18151, 18153, 18169, 18175, 18181, 18187, 18189, 18195, 18207, 18211, 18237, 18253, 18261, 18279, 18283, 18285, 18295, 18303, 18315, 18333, 18375, 18381, 18397, 18399, 18403, 18409, 18441, 18445, 18451, 18459, 18465, 18469, 18475, 18493, 18495, 18501, 18513, 18523, 18535, 18543, 18589, 18591, 18597, 18615, 18619, 18621, 18633, 18639, 18655, 18661, 18681, 18693, 18715, 18717, 18723, 18745, 18747, 18757, 18759, 18765, 18775, 18777, 18783, 18787, 18817, 18819, 18825, 18837, 18849, 18853, 18871, 18873, 18879, 18915, 18937, 18943, 18951, 18967, 18969, 18975, 18979, 18987, 19009, 19015, 19017, 19033, 19051, 19081, 19113, 19119, 19125, 19131, 19153, 19155, 19159, 19161, 19173, 19195, 19197, 19207, 19225, 19237, 19261, 19263, 19269, 19285, 19309, 19323, 19327, 19329, 19333, 19341, 19347, 19351, 19357, 19363, 19365, 19371, 19389, 19395, 19405, 19411, 19417, 19435, 19455, 19467, 19477, 19479, 19503, 19521, 19533, 19543, 19551, 19579, 19581, 19603, 19609, 19615, 19627, 19629, 19657, 19659, 19687, 19693, 19705, 19707, 19711, 19719, 19723, 19725, 19731, 19735, 19755, 19767, 19771, 19773, 19783, 19791, 19807, 19825, 19851, 19861, 19869, 19875, 19893, 19917, 19921, 19929, 19939, 19963, 19981, 19993, 20005, 20013, 20017, 20043, 20067, 20071, 20077, 20085, 20089, 20101, 20103, 20121, 20133, 20149, 20151, 20169, 20181, 20185, 20197, 20203, 20215, 20227, 20253, 20257, 20271, 20295, 20307, 20319, 20341, 20349, 20353, 20355, 20359, 20371, 20379, 20385, 20397, 20403, 20425, 20445, 20449, 20455, 20461, 20475, 20485, 20487, 20509, 20511, 20539, 20551, 20563, 20581, 20583, 20587, 20605, 20611, 20613, 20649, 20665, 20679, 20697, 20713, 20731, 20733, 20739, 20757, 20763, 20769, 20773, 20793, 20797, 20805, 20811, 20821, 20823, 20833, 20835, 20839, 20845, 20859, 20875, 20887, 20889, 20899, 20905, 20923, 20937, 20941, 20953, 20959, 20965, 20991, 21013, 21015, 21021, 21027, 21043, 21049, 21051, 21055, 21073, 21075, 21111, 21115, 21129, 21139, 21147, 21153, 21169, 21175, 21177, 21189, 21193, 21199, 21205, 21211, 21223, 21231, 21253, 21255, 21307, 21309, 21315, 21327, 21331, 21337, 21345, 21373, 21381, 21405, 21409, 21411, 21421, 21433, 21445, 21451, 21453, 21469, 21471, 21475, 21499, 21505, 21529, 21531, 21537, 21549, 21567, 21571, 21591, 21597, 21609, 21613, 21615, 21621, 21655, 21663, 21687, 21693, 21703, 21709, 21717, 21721, 21735, 21739, 21751, 21757, 21759, 21771, 21787, 21819, 21841, 21847, 21867, 21885, 21891, 21897, 21913, 21915, 21925, 21927, 21931, 21933, 21957, 21967, 21987, 21993, 22009, 22017, 22035, 22039, 22075, 22087, 22093, 22095, 22099, 22113, 22119, 22125, 22147, 22155, 22165, 22167, 22177, 22185, 22197, 22225, 22227, 22231, 22269, 22281, 22305, 22327, 22333, 22339, 22369, 22371, 22377, 22381, 22399, 22401, 22411, 22417, 22431, 22437, 22461, 22471, 22473, 22479, 22503, 22507, 22521, 22525, 22527, 22533, 22555, 22569, 22585, 22587, 22591, 22597, 22603, 22617, 22627, 22629, 22651, 22683, 22687, 22689, 22701, 22711, 22743, 22779, 22789, 22791, 22795, 22807, 22827, 22839, 22843, 22851, 22855, 22869, 22879, 22881, 22885, 22893, 22911, 22921, 22941, 22947, 22953, 22957, 22965, 22969, 22975, 22981, 22987, 22999, 23007, 23031, 23037, 23041, 23061, 23067, 23083, 23089, 23095, 23101, 23109, 23137, 23145, 23163, 23169, 23173, 23191, 23193, 23199, 23205, 23269, 23271, 23293, 23295, 23311, 23323, 23341, 23343, 23353, 23355, 23361, 23365, 23373, 23377, 23397, 23419, 23421, 23427, 23445, 23449, 23451, 23457, 23469, 23479, 23499, 23515, 23523, 23529, 23533, 23551, 23563, 23565, 23569, 23571, 23577, 23607, 23613, 23649, 23671, 23677, 23689, 23701, 23709, 23721, 23767, 23775, 23781, 23799, 23805, 23815, 23829, 23835, 23851, 23857, 23881, 23889, 23893, 23913, 23919, 23925, 23929, 23943, 23953, 23955, 23961, 23965, 23971, 23973, 23991, 24013, 24015, 24025, 24033, 24037, 24069, 24073, 24117, 24121, 24129, 24133, 24145, 24151, 24181, 24183, 24213, 24217, 24237, 24247, 24255, 24261, 24277, 24285, 24289, 24301, 24309, 24321, 24327, 24331, 24333, 24343, 24363, 24381, 24387, 24397, 24403, 24405, 24423, 24429, 24447, 24451, 24481, 24499, 24523, 24531, 24571, 24573, 24585, 24591, 24603, 24619, 24633, 24637, 24639, 24643, 24645, 24649, 24663, 24669, 24709, 24711, 24727, 24729, 24733, 24739, 24741, 24759, 24763, 24771, 24783, 24789, 24805, 24811, 24829, 24831, 24843, 24855, 24865, 24873, 24877, 24895, 24907, 24909, 24933, 24951, 24957, 24963, 24985, 24991, 25015, 25017, 25021, 25033, 25057, 25063, 25095, 25099, 25107, 25117, 25119, 25129, 25147, 25153, 25159, 25167, 25171, 25185, 25189, 25201, 25213, 25215, 25225, 25237, 25255, 25263, 25285, 25287, 25293, 25309, 25317, 25335, 25341, 25347, 25377, 25389, 25413, 25425, 25431, 25441, 25455, 25461, 25477, 25483, 25489, 25497, 25503, 25525, 25549, 25561, 25567, 25569, 25581, 25591, 25593, 25611, 25621, 25623, 25645, 25647, 25657, 25671, 25677, 25687, 25737, 25753, 25759, 25771, 25773, 25803, 25815, 25819, 25821, 25833, 25837, 25845, 25855, 25861, 25875, 25893, 25909, 25917, 25923, 25929, 25939, 25959, 25963, 25971, 25977, 25987, 26005, 26007, 26011, 26025, 26035, 26041, 26071, 26085, 26103, 26107, 26115, 26119, 26149, 26155, 26167, 26169, 26179, 26209, 26215, 26217, 26221, 26229, 26251, 26263, 26281, 26293, 26295, 26301, 26313, 26319, 26343, 26347, 26359, 26365, 26385, 26397, 26409, 26413, 26431, 26445, 26467, 26505, 26523, 26529, 26533, 26547, 26553, 26565, 26595, 26599, 26617, 26629, 26641, 26649, 26655, 26659, 26665, 26683, 26695, 26697, 26703, 26719, 26721, 26727, 26733, 26749, 26755, 26757, 26781, 26797, 26805, 26841, 26845, 26847, 26881, 26883, 26893, 26905, 26907, 26913, 26923, 26925, 26935, 26943, 26997, 27001, 27007, 27009, 27013, 27027, 27031, 27033, 27037, 27039, 27061, 27063, 27075, 27079, 27091, 27093, 27097, 27105, 27141, 27157, 27189, 27195, 27199, 27201, 27205, 27207, 27217, 27237, 27265, 27271, 27283, 27301, 27309, 27315, 27321, 27325, 27331, 27333, 27349, 27357, 27363, 27387, 27393, 27409, 27415, 27417, 27427, 27435, 27453, 27457, 27469, 27475, 27477, 27483, 27489, 27493, 27501, 27511, 27531, 27541, 27561, 27585, 27595, 27597, 27603, 27639, 27663, 27667, 27691, 27693, 27705, 27711, 27741, 27751, 27757, 27765, 27783, 27793, 27805, 27807, 27813, 27829, 27831, 27847, 27855, 27859, 27871, 27879, 27913, 27915, 27951, 27961, 27963, 27975, 28021, 28023, 28027, 28039, 28069, 28071, 28083, 28087, 28089, 28101, 28105, 28119, 28123, 28129, 28131, 28141, 28161, 28167, 28191, 28195, 28203, 28233, 28239, 28245, 28293, 28311, 28321, 28329, 28339, 28351, 28359, 28365, 28375, 28393, 28405, 28417, 28429, 28435, 28437, 28459, 28467, 28483, 28509, 28513, 28525, 28563, 28569, 28581, 28593, 28605, 28633, 28639, 28677, 28689, 28701, 28707, 28735, 28759, 28765, 28771, 28791, 28797, 28801, 28803, 28815, 28825, 28827, 28837, 28843, 28855, 28857, 28867, 28869, 28887, 28891, 28905, 28921, 28923, 28933, 28951, 28963, 28969, 28987, 28989, 28993, 28995, 29005, 29025, 29031, 29035, 29049, 29055, 29065, 29089, 29091, 29095, 29113, 29121, 29127, 29137, 29143, 29149, 29157, 29175, 29185, 29193, 29203, 29205, 29239, 29253, 29281, 29283, 29287, 29295, 29311, 29317, 29319, 29329, 29341, 29349, 29371, 29391, 29413, 29437, 29451, 29457, 29467, 29473, 29485, 29487, 29493, 29497, 29515, 29517, 29521, 29533, 29535, 29563, 29577, 29589, 29595, 29599, 29611, 29613, 29623, 29625, 29653, 29659, 29661, 29665, 29677, 29689, 29697, 29709, 29715, 29721, 29751, 29757, 29779, 29781, 29787, 29791, 29805, 29815, 29829, 29845, 29847, 29851, 29865, 29877, 29893, 29905, 29913, 29925, 29949, 29959, 29971, 29973, 29989, 29995, 30013, 30021, 30061, 30067, 30073, 30075, 30087, 30103, 30115, 30121, 30159, 30163, 30181, 30207, 30225, 30229, 30231, 30241, 30243, 30253, 30261, 30265, 30273, 30289, 30291, 30303, 30313, 30327, 30337, 30367, 30369, 30373, 30387, 30397, 30409, 30411, 30429, 30439, 30465, 30471, 30475, 30477, 30495, 30499, 30505, 30523, 30529, 30537, 30541, 30543, 30547, 30559, 30565, 30567, 30589, 30601, 30643, 30651, 30681, 30687, 30727, 30747, 30753, 30765, 30775, 30777, 30789, 30793, 30807, 30817, 30837, 30849, 30861, 30871, 30873, 30877, 30885, 30901, 30915, 30919, 30943, 30945, 30949, 30969, 30997, 30999, 31003, 31021, 31029, 31033, 31039, 31051, 31069, 31071, 31081, 31093, 31095, 31107, 31135, 31137, 31143, 31147, 31189, 31191, 31207, 31221, 31231, 31237, 31239, 31249, 31251, 31255, 31261, 31263, 31293, 31303, 31311, 31315, 31335, 31341, 31363, 31381, 31383, 31387, 31389, 31395, 31405, 31411, 31425, 31429, 31437, 31449, 31453, 31489, 31491, 31515, 31531, 31545, 31573, 31611, 31617, 31633, 31639, 31641, 31681, 31699, 31701, 31713, 31723, 31737, 31755, 31765, 31773, 31789, 31801, 31807, 31815, 31839, 31849, 31863, 31869, 31879, 31887, 31893, 31903, 31927, 31945, 31953, 31965, 31971, 31975, 31977, 31989, 32005, 32011, 32013, 32019, 32041, 32053, 32055, 32077, 32079, 32097, 32101, 32115, 32119, 32121, 32133, 32145, 32173, 32179, 32197, 32223, 32229, 32245, 32247, 32265, 32269, 32271, 32277, 32287, 32311, 32325, 32329, 32373, 32385, 32391, 32397, 32403, 32407, 32419, 32433, 32455, 32457, 32467, 32481, 32509, 32533, 32539, 32545, 32563, 32571, 32577, 32593, 32595, 32605, 32607, 32613, 32635, 32637, 32641, 32643, 32665, 32683, 32713, 32727, 32731, 32745, 32761, 32769, 32775, 32793, 32803, 32827, 32829, 32833, 32845, 32853, 32865, 32869, 32875, 32889, 32917, 32919, 32923, 32937, 32961, 32973, 32985, 32997, 33019, 33021, 33027, 33033, 33045, 33067, 33079, 33091, 33105, 33109, 33117, 33121, 33141, 33151, 33153, 33169, 33181, 33183, 33189, 33205, 33211, 33217, 33237, 33247, 33249, 33273, 33289, 33297, 33309, 33313, 33327, 33333, 33337, 33349, 33363, 33375, 33379, 33381, 33411, 33423, 33433, 33435, 33439, 33445, 33463, 33465, 33475, 33495, 33525, 33537, 33541, 33549, 33561, 33567, 33595, 33601, 33603, 33615, 33633, 33645, 33657, 33673, 33687, 33705, 33717, 33721, 33757, 33769, 33781, 33805, 33813, 33835, 33837, 33843, 33855, 33861, 33885, 33895, 33897, 33907, 33909, 33925, 33943, 33949, 33961, 33973, 33979, 33987, 33991, 33993, 34003, 34005, 34021, 34027, 34053, 34065, 34093, 34099, 34105, 34107, 34113, 34125, 34129, 34131, 34161, 34171, 34183, 34191, 34197, 34201, 34209, 34213, 34233, 34257, 34261, 34281, 34293, 34305, 34321, 34323, 34357, 34359, 34365, 34377, 34387, 34399, 34401, 34423, 34435, 34449, 34453, 34461, 34473, 34485, 34491, 34497, 34503, 34507, 34513, 34557, 34569, 34573, 34599, 34603, 34617, 34621, 34623, 34639, 34641, 34653, 34687, 34713, 34717, 34719, 34723, 34725, 34737, 34755, 34761, 34767, 34777, 34785, 34807, 34809, 34813, 34821, 34873, 34875, 34885, 34887, 34891, 34909, 34915, 34917, 34951, 34957, 34965, 34971, 34995, 35011, 35035, 35041, 35061, 35071, 35077, 35091, 35103, 35113, 35127, 35137, 35145, 35155, 35167, 35169, 35175, 35197, 35203, 35205, 35221, 35223, 35233, 35287, 35301, 35317, 35329, 35331, 35343, 35349, 35391, 35395, 35409, 35413, 35419, 35421, 35427, 35443, 35449, 35451, 35457, 35461, 35481, 35485, 35511, 35539, 35547, 35557, 35563, 35577, 35587, 35601, 35607, 35611, 35625, 35643, 35673, 35683, 35701, 35703, 35707, 35709, 35713, 35721, 35743, 35799, 35817, 35833, 35847, 35859, 35871, 35881, 35913, 35917, 35919, 35941, 35943, 35959, 35965, 35973, 36003, 36025, 36037, 36039, 36043, 36051, 36067, 36085, 36087, 36099, 36115, 36129, 36135, 36147, 36151, 36165, 36169, 36175, 36187, 36205, 36213, 36231, 36249, 36255, 36261, 36273, 36277, 36297, 36309, 36319, 36325, 36331, 36351, 36361, 36373, 36375, 36381, 36405, 36415, 36417, 36423, 36427, 36429, 36451, 36469, 36481, 36487, 36499, 36501, 36513, 36529, 36541, 36547, 36549, 36561, 36573, 36583, 36603, 36609, 36613, 36615, 36619, 36633, 36669, 36673, 36675, 36679, 36681, 36687, 36691, 36715, 36721, 36733, 36741, 36759, 36763, 36805, 36823, 36825, 36829, 36837, 36855, 36877, 36891, 36897, 36903, 36907, 36943, 36955, 36963, 36987, 36993, 36997, 37011, 37015, 37023, 37047, 37051, 37069, 37075, 37077, 37081, 37087, 37089, 37095, 37117, 37129, 37161, 37185, 37195, 37203, 37215, 37219, 37239, 37255, 37263, 37267, 37279, 37285, 37287, 37317, 37339, 37345, 37365, 37371, 37383, 37389, 37393, 37395, 37435, 37437, 37447, 37455, 37467, 37491, 37495, 37507, 37509, 37521, 37527, 37531, 37533, 37549, 37569, 37585, 37597, 37599, 37621, 37635, 37663, 37683, 37689, 37695, 37705, 37723, 37729, 37737, 37747, 37749, 37753, 37789, 37801, 37809, 37813, 37815, 37833, 37845, 37851, 37855, 37893, 37905, 37909, 37939, 37941, 37957, 37959, 37963, 37969, 37989, 38001, 38031, 38035, 38037, 38041, 38059, 38067, 38097, 38119, 38121, 38131, 38145, 38149, 38167, 38185, 38209, 38221, 38245, 38247, 38265, 38271, 38305, 38307, 38313, 38319, 38325, 38341, 38353, 38355, 38389, 38391, 38397, 38401, 38409, 38421, 38433, 38439, 38455, 38461, 38467, 38473, 38481, 38503, 38527, 38535, 38559, 38563, 38581, 38599, 38611, 38623, 38625, 38629, 38653, 38661, 38685, 38695, 38707, 38713, 38731, 38737, 38749, 38751, 38757, 38769, 38775, 38779, 38787, 38791, 38797, 38799, 38811, 38815, 38821, 38839, 38851, 38853, 38871, 38875, 38913, 38917, 38923, 38925, 38943, 38949, 38967, 38985, 38989, 39007, 39043, 39045, 39061, 39067, 39073, 39093, 39115, 39123, 39135, 39139, 39159, 39171, 39177, 39193, 39235, 39249, 39265, 39271, 39283, 39285, 39295, 39301, 39315, 39321, 39345, 39355, 39361, 39367, 39387, 39411, 39417, 39441, 39451, 39463, 39475, 39481, 39507, 39513, 39553, 39567, 39571, 39573, 39579, 39585, 39589, 39607, 39609, 39615, 39627, 39633, 39637, 39643, 39649, 39673, 39693, 39705, 39711, 39721, 39733, 39741, 39753, 39769, 39783, 39795, 39801, 39825, 39831, 39837, 39841, 39849, 39861, 39865, 39883, 39889, 39891, 39903, 39909, 39931, 39945, 39955, 39957, 39967, 39979, 39987, 40005, 40029, 40041, 40059, 40069, 40083, 40101, 40117, 40141, 40147, 40153, 40161, 40165, 40167, 40195, 40197, 40201, 40207, 40215, 40225, 40231, 40249, 40257, 40273, 40287, 40309, 40321, 40323, 40333, 40335, 40351, 40353, 40369, 40389, 40393, 40399, 40405, 40407, 40447, 40449, 40453, 40455, 40459, 40471, 40477, 40479, 40515, 40521, 40555, 40561, 40593, 40597, 40603, 40609, 40665, 40671, 40701, 40707, 40713, 40729, 40743, 40747, 40753, 40761, 40765, 40767, 40797, 40803, 40815, 40825, 40831, 40837, 40849, 40855, 40861, 40879, 40911, 40921, 40923, 40929, 40933, 40963, 40981, 40983, 40993, 40999, 41013, 41023, 41025, 41029, 41043, 41055, 41089, 41101, 41107, 41119, 41121, 41151, 41161, 41181, 41185, 41191, 41235, 41245, 41247, 41277, 41281, 41287, 41295, 41307, 41313, 41319, 41331, 41341, 41349, 41365, 41373, 41383, 41395, 41401, 41421, 41425, 41439, 41463, 41469, 41475, 41485, 41503, 41527, 41529, 41539, 41541, 41551, 41563, 41595, 41613, 41631, 41635, 41653, 41655, 41667, 41677, 41691, 41697, 41707, 41715, 41721, 41727, 41731, 41743, 41779, 41781, 41791, 41793, 41799, 41803, 41805, 41823, 41845, 41847, 41853, 41869, 41905, 41925, 41937, 41943, 41947, 41949, 41961, 41989, 42003, 42009, 42025, 42033, 42051, 42055, 42063, 42073, 42091, 42109, 42117, 42129, 42153, 42163, 42169, 42189, 42193, 42201, 42211, 42231, 42235, 42243, 42279, 42303, 42309, 42361, 42367, 42373, 42381, 42391, 42403, 42405, 42421, 42435, 42447, 42451, 42469, 42471, 42495, 42499, 42511, 42517, 42525, 42547, 42555, 42559, 42561, 42567, 42577, 42591, 42609, 42619, 42625, 42643, 42651, 42657, 42661, 42663, 42675, 42693, 42705, 42715, 42727, 42739, 42751, 42759, 42781, 42789, 42813, 42829, 42847, 42849, 42853, 42865, 42873, 42883, 42885, 42895, 42903, 42907, 42913, 42933, 42949, 42955, 42967, 42975, 42979, 43015, 43021, 43029, 43035, 43039, 43071, 43077, 43129, 43135, 43159, 43167, 43185, 43191, 43201, 43207, 43209, 43225, 43227, 43231, 43233, 43239, 43251, 43255, 43269, 43285, 43303, 43315, 43329, 43333, 43345, 43347, 43381, 43407, 43413, 43419, 43423, 43429, 43441, 43443, 43459, 43461, 43479, 43483, 43491, 43495, 43507, 43513, 43515, 43519, 43533, 43543, 43545, 43549, 43557, 43579, 43597, 43621, 43633, 43639, 43669, 43681, 43695, 43711, 43729, 43735, 43743, 43747, 43771, 43789, 43797, 43815, 43819, 43831, 43833, 43839, 43857, 43863, 43873, 43879, 43891, 43899, 43923, 43933, 43953, 43957, 43965, 43975, 43977, 43981, 43983, 43995, 44007, 44023, 44025, 44037, 44049, 44059, 44109, 44145, 44149, 44155, 44163, 44175, 44187, 44197, 44229, 44239, 44241, 44259, 44293, 44295, 44311, 44313, 44319, 44341, 44353, 44355, 44361, 44383, 44385, 44389, 44395, 44401, 44415, 44419, 44421, 44451, 44485, 44487, 44499, 44503, 44511, 44515, 44541, 44545, 44547, 44551, 44563, 44571, 44595, 44625, 44635, 44637, 44641, 44649, 44655, 44659, 44673, 44691, 44697, 44703, 44721, 44731, 44751, 44761, 44775, 44805, 44863, 44869, 44889, 44893, 44911, 44923, 44929, 44931, 44953, 44955, 44961, 44983, 44995, 45013, 45051, 45057, 45069, 45081, 45109, 45111, 45121, 45133, 45139, 45145, 45159, 45171, 45175, 45181, 45183, 45201, 45235, 45241, 45267, 45271, 45277, 45279, 45303, 45307, 45327, 45331, 45343, 45349, 45351, 45363, 45373, 45375, 45381, 45393, 45403, 45429, 45435, 45459, 45469, 45489, 45507, 45519, 45529, 45535, 45541, 45555, 45591, 45603, 45613, 45619, 45621, 45625, 45681, 45685, 45709, 45717, 45721, 45723, 45739, 45745, 45763, 45769, 45789, 45801, 45811, 45813, 45831, 45843, 45877, 45897, 45907, 45919, 45927, 45939, 45951, 45957, 45973, 45975, 45981, 45991, 46015, 46021, 46023, 46027, 46035, 46065, 46069, 46087, 46095, 46101, 46129, 46131, 46137, 46141, 46167, 46195, 46201, 46203, 46213, 46221, 46227, 46231, 46243, 46245, 46255, 46257, 46263, 46279, 46309, 46311, 46327, 46329, 46339, 46341, 46369, 46377, 46389, 46399, 46401, 46405, 46461, 46465, 46497, 46503, 46519, 46527, 46531, 46543, 46549, 46557, 46587, 46599, 46629, 46641, 46651, 46653, 46671, 46683, 46687, 46695, 46705, 46707, 46713, 46717, 46735, 46737, 46749, 46755, 46759, 46791, 46795, 46797, 46801, 46813, 46815, 46839, 46845, 46855, 46861, 46875, 46879, 46887, 46915, 46923, 46939, 46947, 46951, 46965, 46987, 46999, 47001, 47005, 47029, 47047, 47065, 47071, 47073, 47103, 47107, 47109, 47115, 47125, 47127, 47155, 47157, 47167, 47179, 47193, 47221, 47229, 47233, 47235, 47265, 47271, 47281, 47287, 47313, 47319, 47329, 47335, 47349, 47361, 47379, 47383, 47385, 47419, 47421, 47425, 47427, 47439, 47443, 47449, 47461, 47463, 47487, 47505, 47523, 47565, 47571, 47575, 47581, 47607, 47611, 47613, 47629, 47631, 47665, 47677, 47683, 47697, 47703, 47715, 47727, 47739, 47743, 47745, 47757, 47775, 47799, 47823, 47833, 47841, 47847, 47865, 47889, 47895, 47901, 47911, 47913, 47923, 47947, 47949, 47959, 47965, 47989, 48007, 48015, 48019, 48027, 48049, 48061, 48069, 48081, 48091, 48099, 48103, 48115, 48117, 48133, 48135, 48157, 48177, 48187, 48195, 48205, 48229, 48231, 48243, 48249, 48259, 48279, 48307, 48343, 48345, 48351, 48355, 48369, 48373, 48375, 48385, 48399, 48415, 48439, 48447, 48451, 48481, 48499, 48501, 48535, 48541, 48559, 48583, 48585, 48615, 48639, 48645, 48657, 48669, 48679, 48681, 48687, 48691, 48703, 48721, 48729, 48741, 48751, 48753, 48777, 48799, 48817, 48831, 48837, 48847, 48849, 48879, 48895, 48897, 48913, 48919, 48921, 48925, 48943, 48955, 48961, 48963, 48967, 48999, 49015, 49021, 49045, 49059, 49081, 49099, 49101, 49111, 49113, 49119, 49123, 49141, 49147, 49149, 49155, 49165, 49203, 49207, 49209, 49213, 49219, 49249, 49275, 49279, 49309, 49315, 49329, 49341, 49359, 49363, 49371, 49375, 49377, 49395, 49399, 49435, 49437, 49447, 49471, 49479, 49489, 49497, 49503, 49507, 49525, 49527, 49561, 49563, 49567, 49585, 49615, 49617, 49623, 49627, 49633, 49635, 49675, 49681, 49689, 49713, 49717, 49719, 49723, 49731, 49743, 49753, 49759, 49779, 49795, 49807, 49819, 49821, 49825, 49833, 49839, 49845, 49855, 49857, 49863, 49869, 49917, 49921, 49927, 49959, 49971, 49981, 50001, 50007, 50023, 50025, 50053, 50059, 50085, 50095, 50101, 50109, 50119, 50139, 50151, 50161, 50169, 50179, 50181, 50203, 50217, 50227, 50247, 50257, 50275, 50287, 50317, 50319, 50323, 50325, 50341, 50367, 50371, 50383, 50407, 50415, 50421, 50431, 50445, 50449, 50463, 50469, 50499, 50517, 50527, 50529, 50551, 50559, 50563, 50575, 50595, 50599, 50625, 50631, 50637, 50643, 50653, 50673, 50677, 50685, 50695, 50697, 50719, 50725, 50739, 50751, 50757, 50763, 50767, 50779, 50845, 50871, 50877, 50887, 50889, 50893, 50905, 50907, 50917, 50929, 50935, 50937, 50983, 50991, 51001, 51009, 51013, 51019, 51039, 51067, 51073, 51085, 51093, 51099, 51105, 51127, 51135, 51139, 51141, 51157, 51159, 51163, 51165, 51169, 51187, 51189, 51207, 51229, 51253, 51255, 51271, 51283, 51289, 51303, 51319, 51325, 51349, 51355, 51379, 51393, 51397, 51399, 51411, 51417, 51421, 51453, 51459, 51475, 51543, 51547, 51549, 51559, 51567, 51571, 51579, 51583, 51597, 51601, 51603, 51609, 51649, 51651, 51673, 51675, 51681, 51703, 51729, 51733, 51747, 51753, 51759, 51777, 51793, 51835, 51837, 51855, 51859, 51861, 51865, 51871, 51903, 51913, 51921, 51937, 51955, 51963, 51979, 52039, 52041, 52045, 52059, 52063, 52071, 52081, 52083, 52089, 52101, 52113, 52149, 52165, 52167, 52179, 52185, 52189, 52219, 52227, 52243, 52249, 52251, 52261, 52275, 52279, 52281, 52291, 52321, 52327, 52339, 52353, 52357, 52365, 52369, 52387, 52425, 52429, 52437, 52441, 52449, 52465, 52467, 52479, 52485, 52491, 52503, 52531, 52545, 52549, 52555, 52567, 52573, 52575, 52611, 52617, 52621, 52627, 52635, 52641, 52659, 52671, 52677, 52683, 52689, 52701, 52719, 52731, 52737, 52753, 52755, 52765, 52785, 52807, 52831, 52845, 52849, 52857, 52861, 52863, 52867, 52879, 52887, 52891, 52903, 52921, 52935, 52951, 52953, 52957, 52963, 52975, 52993, 53005, 53053, 53059, 53067, 53079, 53113, 53145, 53155, 53205, 53235, 53239, 53241, 53245, 53265, 53271, 53287, 53301, 53305, 53307, 53319, 53323, 53329, 53341, 53343, 53361, 53385, 53391, 53407, 53409, 53455, 53457, 53467, 53473, 53475, 53491, 53493, 53499, 53517, 53521, 53533, 53557, 53559, 53563, 53575, 53581, 53593, 53595, 53619, 53623, 53629, 53659, 53697, 53701, 53719, 53745, 53749, 53751, 53773, 53781, 53811, 53815, 53817, 53823, 53827, 53839, 53851, 53865, 53871, 53875, 53881, 53901, 53907, 53941, 53949, 53953, 53971, 53991, 54001, 54007, 54043, 54063, 54069, 54075, 54087, 54103, 54129, 54133, 54141, 54147, 54153, 54163, 54169, 54189, 54223, 54229, 54231, 54247, 54259, 54265, 54273, 54277, 54279, 54285, 54309, 54319, 54327, 54339, 54369, 54373, 54403, 54423, 54445, 54447, 54463, 54465, 54469, 54475, 54481, 54483, 54487, 54517, 54525, 54537, 54543, 54547, 54583, 54589, 54591, 54603, 54621, 54625, 54627, 54637, 54645, 54655, 54673, 54687, 54717, 54721, 54729, 54733, 54735, 54747, 54753, 54777, 54781, 54783, 54801, 54819, 54823, 54841, 54853, 54855, 54879, 54909, 54915, 54921, 54939, 54943, 54949, 54961, 54991, 55005, 55009, 55023, 55029, 55033, 55035, 55045, 55047, 55053, 55063, 55075, 55087, 55093, 55095, 55107, 55129, 55131, 55141, 55159, 55167, 55179, 55191, 55195, 55197, 55219, 55237, 55243, 55255, 55281, 55293, 55303, 55317, 55335, 55351, 55369, 55377, 55387, 55423, 55425, 55477, 55485, 55489, 55503, 55507, 55509, 55515, 55539, 55555, 55569, 55573, 55603, 55609, 55617, 55621, 55633, 55639, 55641, 55665, 55671, 55681, 55693, 55705, 55707, 55713, 55729, 55735, 55741, 55747, 55765, 55801, 55803, 55809, 55819, 55825, 55831, 55863, 55867, 55869, 55885, 55903, 55911, 55929, 55935, 55951, 55953, 55975, 56011, 56029, 56059, 56071, 56077, 56083, 56085, 56091, 56103, 56115, 56119, 56125, 56139, 56181, 56185, 56187, 56203, 56205, 56209, 56251, 56263, 56265, 56269, 56271, 56275, 56283, 56289, 56307, 56311, 56313, 56323, 56347, 56359, 56377, 56385, 56397, 56409, 56437, 56439, 56449, 56461, 56463, 56469, 56479, 56497, 56499, 56517, 56523, 56527, 56533, 56535, 56547, 56563, 56565, 56589, 56637, 56649, 56661, 56673, 56679, 56685, 56703, 56721, 56725, 56731, 56751, 56755, 56775, 56779, 56787, 56797, 56799, 56817, 56839, 56841, 56847, 56869, 56895, 56899, 56931, 56935, 56943, 56955, 56959, 56989, 57021, 57025, 57027, 57045, 57049, 57061, 57081, 57109, 57111, 57115, 57145, 57153, 57217, 57225, 57229, 57241, 57249, 57259, 57267, 57277, 57291, 57301, 57303, 57309, 57313, 57333, 57343, 57345, 57355, 57361, 57397, 57405, 57417, 57429, 57441, 57489, 57499, 57505, 57507, 57523, 57525, 57541, 57561, 57591, 57603, 57607, 57613, 57619, 57625, 57633, 57657, 57667, 57679, 57687, 57691, 57699, 57715, 57717, 57729, 57739, 57757, 57777, 57801, 57805, 57819, 57823, 57825, 57841, 57877, 57879, 57889, 57897, 57907, 57945, 57949, 57951, 57967, 57997, 58003, 58009, 58011, 58023, 58047, 58059, 58075, 58077, 58087, 58117, 58135, 58137, 58159, 58161, 58171, 58183, 58197, 58203, 58227, 58233, 58237, 58243, 58261, 58263, 58287, 58311, 58317, 58323, 58327, 58339, 58351, 58383, 58387, 58389, 58393, 58417, 58423, 58435, 58443, 58447, 58455, 58465, 58467, 58479, 58507, 58509, 58519, 58527, 58549, 58557, 58561, 58605, 58615, 58635, 58639, 58641, 58645, 58663, 58687, 58689, 58695, 58699, 58701, 58719, 58737, 58741, 58753, 58779, 58785, 58795, 58815, 58833, 58845, 58849, 58851, 58867, 58875, 58893, 58897, 58905, 58915, 58969, 58971, 58977, 58989, 58999, 59005, 59019, 59037, 59041, 59043, 59055, 59079, 59083, 59095, 59097, 59125, 59127, 59143, 59145, 59149, 59191, 59233, 59235, 59245, 59257, 59263, 59271, 59283, 59287, 59325, 59329, 59353, 59383, 59391, 59401, 59409, 59413, 59439, 59443, 59445, 59457, 59473, 59479, 59481, 59503, 59517, 59521, 59535, 59547, 59581, 59583, 59587, 59599, 59601, 59605, 59631, 59635, 59649, 59653, 59661, 59665, 59695, 59697, 59707, 59725, 59727, 59737, 59739, 59745, 59761, 59791, 59803, 59823, 59863, 59865, 59871, 59893, 59913, 59917, 59919, 59923, 59943, 59997, 60001, 60007, 60031, 60043, 60045, 60051, 60055, 60061, 60063, 60075, 60085, 60105, 60109, 60117, 60133, 60145, 60147, 60151, 60157, 60175, 60177, 60187, 60217, 60219, 60229, 60231, 60259, 60301, 60307, 60313, 60325, 60337, 60339, 60355, 60369, 60385, 60403, 60433, 60465, 60469, 60483, 60489, 60505, 60513, 60529, 60531, 60535, 60547, 60555, 60559, 60567, 60589, 60595, 60613, 60615, 60619, 60631, 60637, 60639, 60655, 60657, 60675, 60705, 60711, 60745, 60747, 60757, 60763, 60783, 60787, 60801, 60817, 60825, 60843, 60861, 60867, 60889, 60891, 60895, 60909, 60913, 60921, 60925, 60927, 60937, 60955, 60973, 60975, 60993, 60997, 61015, 61017, 61057, 61069, 61077, 61089, 61101, 61119, 61123, 61131, 61153, 61155, 61165, 61173, 61183, 61189, 61197, 61207, 61237, 61243, 61257, 61269, 61299, 61305, 61309, 61329, 61335, 61363, 61377, 61383, 61393, 61399, 61405, 61425, 61437, 61455, 61477, 61503, 61513, 61525, 61551, 61557, 61563, 61581, 61585, 61597, 61615, 61617, 61635, 61663, 61683, 61687, 61699, 61707, 61719, 61723, 61773, 61785, 61791, 61809, 61813, 61825, 61833, 61857, 61875, 61881, 61911, 61915, 61917, 61935, 61953, 61971, 61977, 61999, 62005, 62017, 62025, 62037, 62047, 62067, 62071, 62077, 62079, 62097, 62107, 62109, 62143, 62149, 62151, 62155, 62163, 62167, 62173, 62191, 62193, 62197, 62215, 62227, 62245, 62265, 62287, 62293, 62295, 62299, 62323, 62349, 62359, 62379, 62403, 62421, 62425, 62449, 62467, 62475, 62485, 62499, 62503, 62509, 62517, 62527, 62529, 62533, 62547, 62551, 62583, 62605, 62607, 62613, 62623, 62625, 62629, 62635, 62643, 62659, 62671, 62673, 62701, 62719, 62721, 62733, 62769, 62781, 62797, 62803, 62811, 62815, 62817, 62821, 62863, 62877, 62907, 62911, 62919, 62923, 62929, 62941, 62943, 62961, 62979, 62983, 62985, 62991, 63025, 63033, 63037, 63063, 63069, 63085, 63087, 63111, 63115, 63127, 63133, 63151, 63163, 63171, 63177, 63181, 63193, 63195, 63219, 63223, 63231, 63243, 63255, 63265, 63267, 63283, 63303, 63307, 63319, 63321, 63325, 63339, 63379, 63387, 63409, 63411, 63447, 63457, 63471, 63475, 63483, 63487, 63489, 63495, 63505, 63535, 63547, 63549, 63553, 63583, 63609, 63613, 63637, 63645, 63651, 63661, 63681, 63699, 63703, 63727, 63729, 63741, 63747, 63763, 63765, 63777, 63781, 63789, 63801, 63829, 63835, 63843, 63855, 63865, 63889, 63895, 63903, 63915, 63925, 63945, 63957, 63997, 64015, 64021, 64023, 64039, 64051, 64059, 64071, 64123, 64137, 64147, 64159, 64179, 64185, 64189, 64197, 64203, 64207, 64209, 64213, 64221, 64239, 64251, 64267, 64275, 64327, 64329, 64333, 64345, 64347, 64357, 64369, 64375, 64377, 64387, 64435, 64441, 64449, 64455, 64495, 64497, 64501, 64503, 64521, 64533, 64537, 64545, 64561, 64575, 64579, 64597, 64599, 64609, 64623, 64629, 64647, 64659, 64671, 64705, 64711, 64723, 64785, 64789, 64801, 64813, 64819, 64831, 64833, 64837, 64843, 64851, 64857, 64861, 64881, 64891, 64897, 64903, 64905, 64921, 64923, 64933, 64959, 64963, 64969, 64977, 64995, 65025, 65029, 65037, 65049, 65091, 65101, 65109, 65121, 65125, 65143, 65145, 65163, 65173, 65175, 65187, 65197, 65217, 65235, 65241, 65259, 65281, 65289, 65299, 65341, 65343, 65353, 65367, 65385, 65397, 65427, 65431, 65439, 65457, 65479, 65481, 65493, 65499, 65503, 65521, 65529, 65551, 65553, 65563, 65595, 65599, 65629, 65653, 65661, 65671, 65677, 65683, 65691, 65695, 65713, 65725, 65739, 65745, 65755, 65757, 65779, 65793, 65797, 65803, 65815, 65839, 65857, 65865, 65881, 65883, 65901, 65907, 65913, 65935, 65941, 65977, 65985, 65991, 66003, 66031, 66037, 66039, 66049, 66055, 66057, 66061, 66069, 66073, 66103, 66109, 66123, 66141, 66159, 66163, 66171, 66187, 66193, 66201, 66219, 66235, 66247, 66249, 66255, 66259, 66261, 66267, 66291, 66297, 66325, 66339, 66349, 66355, 66369, 66387, 66427, 66433, 66445, 66447, 66469, 66477, 66489, 66495, 66517, 66529, 66543, 66549, 66553, 66559, 66561, 66565, 66573, 66577, 66579, 66607, 66613, 66637, 66657, 66661, 66667, 66669, 66675, 66699, 66703, 66705, 66723, 66729, 66739, 66747, 66781, 66811, 66823, 66829, 66865, 66873, 66897, 66907, 66915, 66921, 66939, 66943, 66957, 66975, 66985, 67003, 67011, 67015, 67017, 67023, 67057, 67063, 67065, 67075, 67105, 67107, 67117, 67125, 67129, 67131, 67143, 67159, 67189, 67195, 67207, 67209, 67225, 67231, 67243, 67245, 67275, 67285, 67287, 67293, 67329, 67333, 67335, 67339, 67351, 67359, 67417, 67419, 67425, 67443, 67447, 67453, 67473, 67477, 67485, 67503, 67569, 67573, 67579, 67581, 67587, 67591, 67599, 67605, 67641, 67671, 67693, 67695, 67711, 67725, 67731, 67755, 67759, 67791, 67797, 67803, 67837, 67839, 67851, 67861, 67873, 67875, 67887, 67899, 67903, 67915, 67917, 67923, 67927, 67929, 67947, 67963, 67965, 67993, 68025, 68029, 68031, 68043, 68047, 68053, 68061, 68065, 68077, 68085, 68095, 68107, 68113, 68115, 68145, 68181, 68187, 68209, 68217, 68221, 68241, 68253, 68293, 68299, 68313, 68335, 68355, 68365, 68377, 68385, 68397, 68407, 68409, 68425, 68439, 68449, 68461, 68463, 68467, 68469, 68497, 68511, 68527, 68529, 68535, 68547, 68557, 68559, 68577, 68589, 68593, 68595, 68599, 68607, 68643, 68653, 68661, 68671, 68683, 68691, 68755, 68767, 68769, 68779, 68799, 68803, 68809, 68811, 68827, 68829, 68845, 68865, 68869, 68875, 68883, 68895, 68913, 68925, 68929, 68931, 68947, 68965, 68989, 68991, 69009, 69015, 69027, 69033, 69039, 69051, 69073, 69099, 69115, 69117, 69123, 69135, 69141, 69153, 69163, 69165, 69195, 69205, 69219, 69225, 69229, 69237, 69243, 69259, 69271, 69291, 69301, 69309, 69333, 69343, 69367, 69373, 69385, 69393, 69397, 69417, 69435, 69439, 69441, 69451, 69481, 69495, 69511, 69513, 69519, 69523, 69535, 69543, 69565, 69577, 69583, 69595, 69615, 69625, 69627, 69637, 69645, 69649, 69679, 69681, 69687, 69703, 69709, 69723, 69727, 69729, 69747, 69757, 69775, 69787, 69789, 69793, 69817, 69819, 69825, 69841, 69871, 69891, 69919, 69921, 69939, 69951, 69985, 69999, 70035, 70063, 70071, 70077, 70081, 70089, 70105, 70107, 70125, 70131, 70165, 70173, 70207, 70215, 70219, 70227, 70261, 70267, 70269, 70287, 70299, 70317, 70333, 70351, 70353, 70357, 70359, 70371, 70405, 70417, 70423, 70435, 70447, 70449, 70455, 70459, 70465, 70471, 70485, 70501, 70509, 70519, 70539, 70545, 70573, 70591, 70597, 70623, 70635, 70639, 70657, 70665, 70669, 70671, 70677, 70699, 70701, 70707, 70717, 70741, 70755, 70761, 70765, 70773, 70783, 70791, 70795, 70803, 70813, 70815, 70845, 70849, 70857, 70867, 70875, 70899, 70923, 70945, 70983, 71005, 71017, 71025, 71037, 71047, 71049, 71055, 71073, 71089, 71101, 71115, 71157, 71161, 71169, 71173, 71179, 71191, 71199, 71215, 71221, 71223, 71235, 71239, 71275, 71277, 71283, 71305, 71329, 71341, 71347, 71359, 71371, 71379, 71403, 71421, 71431, 71433, 71445, 71455, 71479, 71493, 71517, 71535, 71539, 71547, 71557, 71569, 71577, 71593, 71599, 71601, 71617, 71619, 71623, 71631, 71641, 71673, 71679, 71697, 71725, 71739, 71743, 71749, 71769, 71793, 71803, 71821, 71829, 71833, 71869, 71871, 71899, 71929, 71931, 71949, 71953, 71959, 71989, 71991, 71997, 72001, 72019, 72021, 72025, 72055, 72061, 72075, 72123, 72127, 72135, 72139, 72141, 72151, 72157, 72171, 72183, 72187, 72211, 72223, 72235, 72243, 72249, 72265, 72273, 72277, 72303, 72313, 72327, 72333, 72379, 72391, 72393, 72403, 72411, 72439, 72459, 72463, 72493, 72495, 72499, 72501, 72519, 72535, 72565, 72577, 72585, 72589, 72591, 72601, 72607, 72609, 72649, 72663, 72669, 72687, 72693, 72715, 72727, 72735, 72751, 72753, 72765, 72787, 72799, 72819, 72865, 72891, 72895, 72897, 72903, 72907, 72921, 72939, 72955, 72975, 72979, 72985, 72991, 72997, 73003, 73009, 73017, 73021, 73023, 73053, 73065, 73081, 73087, 73101, 73113, 73117, 73131, 73149, 73155, 73165, 73177, 73179, 73185, 73197, 73213, 73221, 73227, 73231, 73249, 73273, 73291, 73305, 73335, 73341, 73363, 73377, 73387, 73405, 73417, 73419, 73431, 73441, 73449, 73461, 73465, 73467, 73473, 73495, 73537, 73543, 73569, 73585, 73587, 73617, 73627, 73629, 73633, 73639, 73669, 73671, 73677, 73683, 73717, 73723, 73731, 73735, 73777, 73785, 73809, 73821, 73843, 73851, 73873, 73899, 73911, 73915, 73933, 73935, 73969, 73977, 73995, 74013, 74031, 74035, 74041, 74049, 74059, 74071, 74101, 74103, 74109, 74121, 74139, 74143, 74155, 74161, 74175, 74187, 74193, 74197, 74199, 74221, 74227, 74257, 74259, 74289, 74293, 74307, 74311, 74325, 74329, 74331, 74347, 74349, 74353, 74355, 74385, 74391, 74415, 74425, 74433, 74439, 74445, 74449, 74455, 74457, 74487, 74497, 74503, 74509, 74515, 74541, 74559, 74571, 74577, 74581, 74583, 74595, 74613, 74635, 74643, 74647, 74697, 74701, 74721, 74733, 74739, 74749, 74761, 74767, 74773, 74787, 74793, 74797, 74815, 74817, 74857, 74859, 74865, 74869, 74893, 74895, 74907, 74911, 74913, 74919, 74941, 74955, 74959, 74973, 75019, 75025, 75055, 75067, 75099, 75103, 75111, 75117, 75147, 75163, 75181, 75193, 75195, 75201, 75211, 75213, 75223, 75229, 75247, 75255, 75265, 75271, 75289, 75297, 75307, 75319, 75357, 75391, 75399, 75421, 75433, 75435, 75441, 75475, 75481, 75483, 75505, 75511, 75517, 75525, 75529, 75537, 75549, 75561, 75567, 75583, 75585, 75589, 75591, 75603, 75625, 75631, 75637, 75645, 75655, 75675, 75679, 75693, 75709, 75717, 75733, 75735, 75741, 75759, 75781, 75789, 75793, 75799, 75823, 75825, 75835, 75841, 75853, 75859, 75867, 75873, 75885, 75895, 75901, 75925, 75937, 75945, 75967, 75985, 75991, 75993, 76015, 76029, 76045, 76065, 76077, 76083, 76089, 76105, 76107, 76113, 76125, 76129, 76149, 76179, 76183, 76189, 76203, 76209, 76219, 76233, 76239, 76275, 76303, 76317, 76339, 76341, 76357, 76359, 76371, 76381, 76393, 76405, 76423, 76455, 76461, 76473, 76497, 76503, 76525, 76533, 76551, 76555, 76567, 76587, 76609, 76615, 76617, 76623, 76633, 76639, 76663, 76671, 76705, 76713, 76717, 76735, 76737, 76749, 76759, 76779, 76783, 76785, 76797, 76801, 76803, 76831, 76849, 76851, 76873, 76885, 76893, 76905, 76939, 76947, 76969, 76975, 76989, 76995, 77001, 77017, 77037, 77053, 77061, 77065, 77091, 77115, 77127, 77137, 77149, 77167, 77209, 77211, 77217, 77223, 77245, 77253, 77263, 77269, 77275, 77281, 77289, 77293, 77301, 77307, 77353, 77355, 77365, 77377, 77379, 77395, 77397, 77407, 77431, 77457, 77473, 77481, 77503, 77505, 77515, 77521, 77527, 77533, 77553, 77565, 77577, 77599, 77601, 77617, 77619, 77625, 77653, 77661, 77671, 77679, 77689, 77721, 77733, 77745, 77757, 77763, 77767, 77785, 77787, 77791, 77805, 77811, 77821, 77835, 77841, 77853, 77857, 77869, 77877, 77893, 77911, 77913, 77937, 77955, 77961, 77983, 78015, 78025, 78067, 78069, 78079, 78093, 78099, 78109, 78129, 78133, 78141, 78145, 78165, 78187, 78193, 78195, 78199, 78253, 78261, 78271, 78277, 78279, 78283, 78291, 78319, 78321, 78325, 78339, 78355, 78357, 78373, 78379, 78385, 78397, 78409, 78469, 78487, 78511, 78529, 78531, 78553, 78591, 78597, 78603, 78613, 78627, 78645, 78649, 78657, 78661, 78673, 78675, 78679, 78711, 78723, 78729, 78733, 78739, 78751, 78763, 78765, 78775, 78825, 78829, 78837, 78849, 78855, 78861, 78867, 78879, 78901, 78913, 78919, 78945, 78969, 78973, 78985, 78987, 78991, 79003, 79033, 79045, 79057, 79099, 79113, 79117, 79129, 79135, 79137, 79153, 79159, 79161, 79177, 79183, 79207, 79215, 79221, 79255, 79261, 79269, 79275, 79327, 79341, 79353, 79365, 79369, 79371, 79381, 79401, 79413, 79423, 79429, 79449, 79453, 79465, 79467, 79473, 79477, 79491, 79497, 79509, 79513, 79531, 79537, 79551, 79557, 79561, 79569, 79593, 79615, 79623, 79657, 79663, 79669, 79675, 79683, 79701, 79705, 79707, 79725, 79737, 79761, 79773, 79825, 79851, 79855, 79885, 79891, 79899, 79905, 79915, 79917, 79921, 79933, 79959, 79981, 79983, 79989, 79993, 80001, 80023, 80041, 80047, 80059, 80073, 80077, 80107, 80137, 80157, 80179, 80185, 80199, 80203, 80209, 80211, 80241, 80269, 80277, 80287, 80295, 80305, 80311, 80313, 80317, 80335, 80337, 80341, 80347, 80377, 80395, 80413, 80425, 80433, 80437, 80443, 80473, 80475, 80481, 80503, 80527, 80545, 80557, 80559, 80569, 80577, 80581, 80583, 80593, 80625, 80641, 80649, 80655, 80671, 80683, 80685, 80703, 80733, 80737, 80749, 80767, 80773, 80775, 80797, 80815, 80835, 80839, 80851, 80853, 80875, 80881, 80901, 80907, 80913, 80925, 80937, 80941, 80955, 80967, 81001, 81007, 81009, 81019, 81039, 81081, 81093, 81103, 81111, 81115, 81133, 81145, 81147, 81157, 81165, 81169, 81195, 81207, 81213, 81217, 81241, 81261, 81273, 81277, 81295, 81301, 81307, 81313, 81319, 81337, 81349, 81355, 81357, 81363, 81403, 81421, 81439, 81451, 81459, 81465, 81469, 81475, 81481, 81511, 81525, 81529, 81537, 81553, 81573, 81585, 81601, 81609, 81615, 81621, 81649, 81655, 81661, 81711, 81715, 81721, 81723, 81727, 81733, 81745, 81747, 81757, 81781, 81783, 81811, 81817, 81841, 81849, 81861, 81871, 81883, 81913, 81931, 81933, 81937, 81943, 81949, 81969, 81985, 82035, 82039, 82057, 82059, 82063, 82095, 82101, 82105, 82111, 82119, 82123, 82135, 82155, 82165, 82177, 82185, 82189, 82197, 82227, 82231, 82237, 82257, 82269, 82285, 82293, 82299, 82311, 82321, 82345, 82363, 82383, 82393, 82419, 82429, 82437, 82447, 82449, 82473, 82479, 82501, 82503, 82513, 82519, 82543, 82567, 82581, 82585, 82593, 82635, 82677, 82681, 82687, 82689, 82693, 82705, 82707, 82719, 82731, 82743, 82749, 82761, 82771, 82785, 82791, 82795, 82797, 82803, 82825, 82831, 82833, 82855, 82857, 82861, 82879, 82881, 82893, 82909, 82939, 82957, 82959, 82977, 82983, 82995, 83025, 83035, 83055, 83059, 83067, 83077, 83083, 83097, 83133, 83161, 83167, 83181, 83191, 83209, 83223, 83233, 83271, 83277, 83293, 83307, 83317, 83319, 83337, 83355, 83391, 83401, 83419, 83421, 83443, 83445, 83449, 83455, 83479, 83499, 83521, 83523, 83559, 83563, 83569, 83571, 83593, 83601, 83605, 83611, 83625, 83631, 83643, 83647, 83655, 83667, 83671, 83697, 83707, 83709, 83719, 83733, 83751, 83763, 83769, 83773, 83799, 83815, 83841, 83857, 83865, 83887, 83895, 83901, 83905, 83907, 83917, 83919, 83923, 83929, 83953, 83959, 83961, 83985, 83995, 84009, 84049, 84075, 84085, 84091, 84097, 84105, 84115, 84121, 84135, 84141, 84153, 84157, 84189, 84201, 84213, 84223, 84235, 84241, 84273, 84309, 84319, 84331, 84345, 84381, 84393, 84405, 84411, 84421, 84423, 84429, 84435, 84457, 84463, 84465, 84471, 84483, 84487, 84495, 84499, 84519, 84549, 84553, 84591, 84619, 84631, 84633, 84645, 84661, 84663, 84673, 84675, 84679, 84687, 84693, 84721, 84745, 84769, 84781, 84787, 84789, 84813, 84819, 84829, 84841, 84847, 84849, 84861, 84883, 84895, 84925, 84937, 84939, 84949, 84955, 84957, 84975, 84991, 84999, 85003, 85009, 85033, 85035, 85053, 85075, 85087, 85095, 85105, 85129, 85143, 85147, 85149, 85159, 85177, 85179, 85197, 85239, 85245, 85249, 85255, 85275, 85303, 85305, 85327, 85333, 85335, 85339, 85351, 85353, 85371, 85387, 85419, 85441, 85449, 85461, 85465, 85491, 85495, 85507, 85555, 85563, 85569, 85587, 85591, 85597, 85605, 85617, 85621, 85669, 85671, 85693, 85701, 85705, 85735, 85755, 85773, 85777, 85779, 85807, 85813, 85815, 85819, 85821, 85843, 85851, 85861, 85869, 85873, 85875, 85917, 85923, 85933, 85947, 85963, 85965, 85999, 86005, 86025, 86031, 86037, 86071, 86089, 86095, 86101, 86103, 86125, 86155, 86163, 86167, 86185, 86187, 86199, 86205, 86221, 86227, 86257, 86259, 86269, 86281, 86293, 86323, 86331, 86335, 86341, 86385, 86409, 86415, 86421, 86425, 86427, 86439, 86445, 86469, 86481, 86491, 86505, 86515, 86545, 86571, 86593, 86595, 86617, 86631, 86637, 86647, 86661, 86671, 86677, 86679, 86719, 86721, 86743, 86761, 86763, 86775, 86803, 86815, 86829, 86857, 86865, 86877, 86901, 86925, 86929, 86941, 86949, 86953, 86965, 86973, 86983, 86989, 87003, 87007, 87015, 87019, 87025, 87027, 87049, 87051, 87055, 87097, 87103, 87109, 87111, 87121, 87129, 87133, 87151, 87153, 87171, 87175, 87183, 87199, 87207, 87241, 87255, 87261, 87289, 87297, 87301, 87303, 87319, 87339, 87343, 87361, 87387, 87391, 87397, 87403, 87411, 87415, 87433, 87435, 87451, 87453, 87459, 87469, 87481, 87493, 87499, 87529, 87549, 87559, 87571, 87579, 87583, 87595, 87607, 87633, 87639, 87645, 87667, 87669, 87703, 87705, 87717, 87721, 87733, 87741, 87747, 87751, 87783, 87793, 87795, 87823, 87835, 87837, 87867, 87871, 87873, 87897, 87931, 87937, 87957, 87973, 87979, 87981, 87991, 88011, 88027, 88041, 88045, 88057, 88065, 88075, 88083, 88089, 88117, 88119, 88125, 88129, 88147, 88159, 88171, 88173, 88183, 88185, 88221, 88237, 88249, 88267, 88299, 88305, 88329, 88333, 88335, 88347, 88363, 88371, 88401, 88413, 88423, 88435, 88441, 88459, 88467, 88495, 88503, 88519, 88521, 88527, 88537, 88539, 88551, 88561, 88587, 88591, 88593, 88603, 88615, 88629, 88653, 88663, 88665, 88693, 88707, 88713, 88735, 88747, 88755, 88773, 88783, 88789, 88813, 88815, 88819, 88821, 88837, 88839, 88851, 88861, 88881, 88893, 88897, 88903, 88915, 88927, 88939, 88945, 88947, 88971, 88987, 88989, 89001, 89023, 89025, 89029, 89041, 89043, 89065, 89071, 89085, 89115, 89119, 89131, 89167, 89169, 89175, 89191, 89209, 89217, 89223, 89233, 89257, 89283, 89287, 89293, 89319, 89325, 89335, 89337, 89349, 89355, 89359, 89377, 89383, 89385, 89397, 89403, 89421, 89427, 89433, 89445, 89467, 89481, 89493, 89497, 89503, 89509, 89527, 89533, 89545, 89557, 89575, 89577, 89587, 89599, 89637, 89653, 89695, 89715, 89743, 89755, 89781, 89787, 89799, 89805, 89829, 89841, 89845, 89869, 89871, 89883, 89901, 89905, 89911, 89913, 89923, 89947, 89977, 89989, 90009, 90039, 90063, 90069, 90093, 90097, 90099, 90105, 90121, 90127, 90133, 90159, 90175, 90183, 90199, 90229, 90231, 90241, 90259, 90261, 90265, 90279, 90291, 90295, 90301, 90313, 90349, 90351, 90355, 90357, 90373, 90379, 90393, 90409, 90435, 90457, 90489, 90517, 90535, 90537, 90547, 90561, 90579, 90601, 90603, 90619, 90625, 90649, 90663, 90679, 90691, 90703, 90711, 90727, 90745, 90757, 90765, 90769, 90771, 90813, 90817, 90819, 90825, 90847, 90861, 90871, 90891, 90897, 90913, 90919, 90921, 90939, 90951, 90961, 90963, 90973, 90993, 91015, 91021, 91023, 91039, 91047, 91051, 91059, 91087, 91089, 91111, 91113, 91129, 91131, 91141, 91143, 91177, 91191, 91195, 91213, 91225, 91227, 91237, 91239, 91257, 91269, 91279, 91299, 91303, 91329, 91333, 91339, 91387, 91399, 91405, 91423, 91429, 91447, 91479, 91483, 91491, 91497, 91509, 91527, 91531, 91543, 91561, 91563, 91575, 91581, 91585, 91587, 91593, 91605, 91633, 91639, 91651, 91665, 91701, 91711, 91729, 91731, 91761, 91779, 91783, 91795, 91813, 91815, 91833, 91837, 91843, 91861, 91867, 91885, 91891, 91903, 91917, 91921, 91947, 91965, 91995, 92011, 92043, 92053, 92059, 92073, 92077, 92085, 92095, 92113, 92143, 92155, 92179, 92199, 92205, 92211, 92223, 92233, 92239, 92247, 92275, 92277, 92299, 92301, 92341, 92347, 92367, 92373, 92379, 92395, 92407, 92409, 92425, 92427, 92433, 92443, 92445, 92457, 92521, 92527, 92557, 92559, 92569, 92571, 92589, 92599, 92635, 92647, 92659, 92665, 92677, 92685, 92689, 92697, 92709, 92727, 92739, 92743, 92749, 92757, 92761, 92779, 92781, 92785, 92823, 92829, 92841, 92851, 92869, 92887, 92905, 92911, 92913, 92917, 92925, 92929, 92937, 92941, 92973, 92977, 93003, 93021, 93031, 93057, 93063, 93093, 93099, 93105, 93117, 93123, 93129, 93135, 93139, 93147, 93169, 93189, 93223, 93229, 93241, 93247, 93261, 93265, 93285, 93289, 93295, 93303, 93339, 93355, 93381, 93397, 93403, 93417, 93433, 93451, 93459, 93465, 93483, 93501, 93505, 93507, 93517, 93523, 93525, 93543, 93547, 93555, 93577, 93591, 93597, 93609, 93627, 93633, 93669, 93691, 93703, 93705, 93715, 93753, 93759, 93781, 93787, 93799, 93813, 93829, 93831, 93849, 93871, 93877, 93883, 93885, 93895, 93913, 93915, 93945, 93957, 93963, 93969, 93979, 93987, 94003, 94005, 94009, 94017, 94021, 94029, 94039, 94065, 94069, 94089, 94093, 94105, 94135, 94159, 94189, 94195, 94197, 94209, 94215, 94219, 94221, 94255, 94257, 94273, 94291, 94293, 94299, 94317, 94321, 94323, 94327, 94345, 94353, 94363, 94381, 94389, 94407, 94419, 94423, 94443, 94453, 94461, 94471, 94473, 94479, 94489, 94513, 94531, 94551, 94555, 94567, 94573, 94575, 94593, 94605, 94629, 94641, 94651, 94659, 94663, 94677, 94701, 94711, 94735, 94761, 94767, 94785, 94789, 94797, 94801, 94803, 94821, 94831, 94849, 94857, 94867, 94879, 94891, 94909, 94923, 94933, 94957, 94963, 94965, 94971, 94987, 95025, 95029, 95035, 95037, 95055, 95059, 95089, 95113, 95143, 95145, 95163, 95179, 95185, 95209, 95223, 95229, 95239, 95245, 95259, 95271, 95281, 95301, 95335, 95343, 95373, 95383, 95385, 95403, 95413, 95415, 95431, 95445, 95449, 95451, 95461, 95467, 95469, 95479, 95497, 95509, 95517, 95539, 95557, 95575, 95577, 95593, 95601, 95607, 95619, 95635, 95637, 95671, 95677, 95683, 95697, 95707, 95731, 95739, 95745, 95751, 95761, 95791, 95793, 95803, 95809, 95811, 95823, 95853, 95859, 95865, 95875, 95877, 95887, 95923, 95929, 95935, 95955, 95995, 96001, 96003, 96013, 96015, 96019, 96027, 96037, 96043, 96055, 96079, 96099, 96129, 96139, 96141, 96147, 96151, 96181, 96187, 96201, 96211, 96231, 96235, 96243, 96267, 96273, 96309, 96313, 96315, 96333, 96343, 96363, 96375, 96391, 96393, 96399, 96403, 96411, 96423, 96427, 96459, 96469, 96475, 96495, 96529, 96549, 96553, 96571, 96579, 96589, 96591, 96601, 96621, 96625, 96631, 96649, 96657, 96663, 96675, 96709, 96715, 96747, 96751, 96769, 96777, 96783, 96813, 96817, 96823, 96835, 96843, 96847, 96861, 96867, 96877, 96879, 96909, 96957, 96963, 96973, 96979, 96981, 97005, 97009, 97053, 97063, 97069, 97083, 97089, 97093, 97105, 97107, 97113, 97213, 97219, 97231, 97233, 97243, 97255, 97261, 97275, 97279, 97323, 97335, 97347, 97359, 97365, 97381, 97387, 97389, 97411, 97413, 97419, 97423, 97443, 97449, 97465, 97467, 97471, 97491, 97495, 97525, 97531, 97545, 97555, 97561, 97575, 97587, 97599, 97603, 97617, 97623, 97629, 97639, 97671, 97681, 97683, 97687, 97699, 97701, 97729, 97743, 97755, 97779, 97807, 97809, 97819, 97821, 97849, 97861, 97875, 97891, 97911, 97917, 97927, 97947, 97957, 97971, 97975, 97987, 97999, 98007, 98041, 98049, 98053, 98065, 98091, 98107, 98127, 98137, 98155, 98161, 98167, 98179, 98185, 98187, 98199, 98217, 98253, 98283, 98289, 98293, 98305, 98313, 98317, 98329, 98349, 98353, 98391, 98395, 98413, 98443, 98451, 98455, 98469, 98473, 98481, 98491, 98511, 98547, 98553, 98563, 98569, 98583, 98587, 98599, 98611, 98619, 98625, 98629, 98637, 98661, 98679, 98689, 98701, 98707, 98709, 98713, 98731, 98755, 98757, 98763, 98799, 98805, 98817, 98827, 98869, 98871, 98877, 98893, 98895, 98935, 98947, 98953, 98979, 98997, 99003, 99015, 99045, 99057, 99069, 99087, 99103, 99109, 99121, 99141, 99147, 99151, 99153, 99165, 99171, 99175, 99207, 99211, 99217, 99225, 99235, 99267, 99271, 99279, 99289, 99301, 99319, 99333, 99337, 99367, 99375, 99381, 99403, 99417, 99459, 99477, 99483, 99493, 99507, 99525, 99531, 99541, 99543, 99547, 99555, 99571, 99573, 99577, 99583, 99585, 99589, 99591, 99595, 99603, 99637, 99649, 99657, 99667, 99673, 99675, 99691, 99703, 99717, 99735, 99745, 99753, 99781, 99795, 99813, 99817, 99823, 99825, 99843, 99855, 99859, 99865, 99877, 99879, 99885, 99889, 99897, 99907, 99919, 99921, 99949, 99955, 99973, 99987};

int find(int n) {

int l = 0, r = 8772;

while (r - l > 1) {

int mid = (l+r) / 2;

if (a[mid] > n) {

r = mid;

} else {

l = mid;

}

}

return l;

}

int main() {

int n, m;

scanf("%d %d", &n, &m);

printf("%d\n", find(m) - find(n));

}

本题的C参考代码如下：

#include<stdio.h>

#define MAXN 1000010

int flag[MAXN];

int m,n,a[MAXN],s[MAXN],size=0;

int fa(int k)

{

if(flag[k])

return a[k];

return fa(k-1);

}

int main()

{

int i, p, k, j;

scanf("%d%d",&m,&n);

for(i=1;i<=n;i+=2)

{

s[++size]=i;

flag[i]=1;

a[i]=size;

}

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

{

int Mod=s[i],d=s[i]-1;

if(Mod>size)

break;

for(p=1,j=Mod;j<=size;j+=Mod,p++)

{

flag[s[j]]=0;

for(k=1;k<Mod&&k+j<=size;k++)

{

s[++d]=s[j+k];

a[s[j+k]]-=p;

}

}

size=d;

}

printf("%d\n",fa(n-1)-fa(m));

return 0;

}

**历届试题 大臣的旅费**

问题描述

很久以前，T王国空前繁荣。为了更好地管理国家，王国修建了大量的快速路，用于连接首都和王国内的各大城市。

为节省经费，T国的大臣们经过思考，制定了一套优秀的修建方案，使得任何一个大城市都能从首都直接或者通过其他大城市间接到达。同时，如果不重复经过大城市，从首都到达每个大城市的方案都是唯一的。

J是T国重要大臣，他巡查于各大城市之间，体察民情。所以，从一个城市马不停蹄地到另一个城市成了J最常做的事情。他有一个钱袋，用于存放往来城市间的路费。

聪明的J发现，如果不在某个城市停下来修整，在连续行进过程中，他所花的路费与他已走过的距离有关，在走第x千米到第x+1千米这一千米中（x是整数），他花费的路费是x+10这么多。也就是说走1千米花费11，走2千米要花费23。

J大臣想知道：他从某一个城市出发，中间不休息，到达另一个城市，所有可能花费的路费中最多是多少呢？

输入格式

输入的第一行包含一个整数n，表示包括首都在内的T王国的城市数

城市从1开始依次编号，1号城市为首都。

接下来n-1行，描述T国的高速路（T国的高速路一定是n-1条）

每行三个整数Pi, Qi, Di，表示城市Pi和城市Qi之间有一条高速路，长度为Di千米。

输出格式

输出一个整数，表示大臣J最多花费的路费是多少。

样例输入1

5  
1 2 2  
1 3 1  
2 4 5  
2 5 4

样例输出1

135

输出格式

大臣J从城市4到城市5要花费135的路费。

本题的C++参考代码如下：

#include <iostream>

#include <vector>

#include <cstdio>

using namespace std;

struct edge

{

int Number;

int Length;

edge\* next;

edge(int number, int length)

{

Number = number;

Length = length;

next = NULL;

}

};

struct vert

{

int Number;

int PathLen;

bool Isvisit;

edge\* First;

vert(int number)

{

Number = number;

Isvisit = false;

First = NULL;

PathLen = 0;

}

};

void dfs(vector<vert>& GList, int num)

{

edge\* p= GList[num].First;

GList[num].Isvisit = true;

for (; p!=NULL; p=p->next)

{

if(GList[p->Number].Isvisit == false)

{

GList[p->Number].PathLen =GList[num].PathLen + p->Length;

dfs(GList, p->Number);

}

}

}

int main()

{

vector<vert> GList;

int n, i, j;

int fir, sed, len;

cin>>n;

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

{

GList.push\_back(vert(i));

}

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

{

edge\* p1, \*p2;

scanf("%d%d%d",&fir,&sed,&len);

p1 = GList[fir].First;

p2 = GList[sed].First;

//无向图

if (p1==NULL)

{

GList[fir].First = new edge(sed, len);

}

else

{

while(p1->next!=NULL) //将边链接到顶点

{

p1 = p1->next;

}

p1->next = new edge(sed, len);

}

if (p2==NULL)

{

GList[sed].First = new edge(fir, len);

}

else

{

while(p2->next!=NULL) //将边链接到顶点

{

p2 = p2->next;

}

p2->next = new edge(fir, len);

}

}

int start, end, length = 0;

dfs(GList, 1);

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

{

if (GList[j].PathLen > length)//更新最长距离

{

start = i;

end = j;

length = GList[j].PathLen;

}

GList[j].PathLen = 0;

GList[j].Isvisit = false;

}

dfs(GList, end);

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

{

if (GList[j].PathLen > length)//更新最长距离

{

start = i;

end = j;

length = GList[j].PathLen;

}

GList[j].PathLen = 0;

GList[j].Isvisit = false;

}

int cost = length\*10;

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

{

cost+=i;

}

cout<<cost<<endl;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

struct node;

typedef struct node Node;

typedef Node \*PtrToNode;

typedef PtrToNode List;

typedef PtrToNode Position;

struct node

{

int n;

int val;

Position next;

};

int count=0;

int max=0;

int x;

int \*visit; //是否已遍历

Position Last(List l); //找出最后项

void Insert(int x,int q,List l,Position p); //在p后插入含x的项

void Dfs(int a,List l[]); //深度优先搜索

int Num(List l);

int main(void)

{

int n,u,v,q,a,b;

int i,j,k;

Node \*head;

List \*l,tmp;

Position p;

fscanf(stdin,"%d",&n);

head=(Node \*)malloc(sizeof(Node)\*(n+1));

l=(List \*)malloc(sizeof(List)\*(n+1));

visit=(int \*)malloc(sizeof(int)\*(n+1));

for(i=0;i<=n;i++) //初始化表头及链表

{

head[i].next=NULL;

l[i]=&head[i];

}

for(i=1;i<=n-1;i++) //建立无向图

{

fscanf(stdin,"%d%d%d",&u,&v,&q);

Insert(v,q,l[u],Last(l[u]));

Insert(u,q,l[v],Last(l[v]));

}

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

visit[j]=0;

Dfs(1,l); //第一次遍历，找到点a，用全局变量x保存

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

visit[j]=0;

count=0;

max=0;

Dfs(x,l); //第二次遍历，找到点b，用全局变量x保存，此时max为最大距离

printf("%d",max\*10+(max+1)\*max/2);

return 0;

}

Position Last(List l)

{

Position p;

for(p=l;p->next!=NULL;p=p->next);

return p;

}

void Insert(int x,int q,List l,Position p)

{

Position tmp;

tmp=(Position) malloc(sizeof(Node));

tmp->n=x;

tmp->val=q;

tmp->next=p->next;

p->next=tmp;

}

void Dfs(int a,List l[])

{

Position p;

visit[a]=1;

for(p=l[a]->next;p!=NULL;p=p->next)

if(!(visit[p->n]))

{

count+=p->val;

if(count>max)

{

max=count;

x=p->n;

}

Dfs(p->n,l);

count-=p->val;

}

}

int Num(List l)

{

int n=0;

Position p;

for(p=l->next;p!=NULL;p=p->next)

n++;

return n;

}

**历届试题 买不到的数目**

问题描述

小明开了一家糖果店。他别出心裁：把水果糖包成4颗一包和7颗一包的两种。糖果不能拆包卖。

小朋友来买糖的时候，他就用这两种包装来组合。当然有些糖果数目是无法组合出来的，比如要买 10 颗糖。

你可以用计算机测试一下，在这种包装情况下，最大不能买到的数量是17。大于17的任何数字都可以用4和7组合出来。

本题的要求就是在已知两个包装的数量时，求最大不能组合出的数字。

输入格式

两个正整数，表示每种包装中糖的颗数(都不多于1000)

输出格式

一个正整数，表示最大不能买到的糖数

样例输入1

4 7

样例输出1

17

样例输入2

3 5

样例输出2

7

本题的C++参考代码如下：

//2013第四届蓝桥杯全国软件大赛省级选拔赛C/C++A组第8题

//天农计算机系许晓华老师出品

#include<iostream>

using namespace std;

int main()

{

int a,b;

cin>>a>>b;

cout<<a\*b-a-b<<endl;

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#define MAXSIZE 1000

int main() {

int flag[MAXSIZE] = {0};

int maxunuse[MAXSIZE] = {0};

int num1, num2, minNum, maxNum, temp, i, count = 0;

scanf("%d%d", &num1, &num2);

minNum = ((num1 < num2) ? num1 : num2);

maxNum = num1 + num2 - minNum;

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

temp = i \* maxNum % minNum;

if((temp) && flag[temp] == 0){

flag[temp] = 1;

maxunuse[temp] = i \* maxNum -minNum;

count++;

if(count == minNum - 1){

break;

}

}

}

printf("%d\n", maxunuse[temp]);

return 0;

}

**历届试题 连号区间数**

问题描述

小明这些天一直在思考这样一个奇怪而有趣的问题：

在1~N的某个全排列中有多少个连号区间呢？这里所说的连号区间的定义是：

如果区间[L, R] 里的所有元素（即此排列的第L个到第R个元素）递增排序后能得到一个长度为R-L+1的“连续”数列，则称这个区间连号区间。

当N很小的时候，小明可以很快地算出答案，但是当N变大的时候，问题就不是那么简单了，现在小明需要你的帮助。

输入格式

第一行是一个正整数N (1 <= N <= 50000), 表示全排列的规模。

第二行是N个不同的数字Pi(1 <= Pi <= N)， 表示这N个数字的某一全排列。

输出格式

输出一个整数，表示不同连号区间的数目。

样例输入1

4  
3 2 4 1

样例输出1

7

样例输入2

5  
3 4 2 5 1

样例输出2

9

本题的C++参考代码如下：

#include <stdio.h>

#include <stdlib.h>

int main()

{

int a[50010] , n ,sum = 0 , max , min ;

scanf("%d",&n) ;

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

{

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

}

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

{

min = max =a[i] ;

for(int j = i ; j < n ; ++j)

{

if(a[j]>max)

{

max = a[j];

}

else if(a[j]<min)

{

min = a[j];

}

if(max-min == j-i)

{

++sum ;

}

}

}

printf("%d\n",sum) ;

return 0 ;

}

本题的C参考代码如下：

#include<stdio.h>

int main()

{

int s[50005],a,i,min,max,count=0,j;

scanf("%d",&a);

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

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

}

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

min=s[i];

max=s[i];

for( j = i; j <a; j++) {

if(min>s[j]){min =s[j];}

if(max<s[j]){max =s[j];}

if((max-min)==(j-i)){

count++;

}

}

}

printf("%d",count);

return 0;

}

**历届试题 翻硬币**

问题描述

小明正在玩一个“翻硬币”的游戏。

桌上放着排成一排的若干硬币。我们用 \* 表示正面，用 o 表示反面（是小写字母，不是零）。

比如，可能情形是：\*\*oo\*\*\*oooo

如果同时翻转左边的两个硬币，则变为：oooo\*\*\*oooo

现在小明的问题是：如果已知了初始状态和要达到的目标状态，每次只能同时翻转相邻的两个硬币,那么对特定的局面，最少要翻动多少次呢？

我们约定：把翻动相邻的两个硬币叫做一步操作，那么要求：

输入格式

两行等长的字符串，分别表示初始状态和要达到的目标状态。每行的长度<1000

输出格式

一个整数，表示最小操作步数。

样例输入1

\*\*\*\*\*\*\*\*\*\*  
o\*\*\*\*o\*\*\*\*

样例输出1

5

样例输入2

\*o\*\*o\*\*\*o\*\*\*  
\*o\*\*\*o\*\*o\*\*\*

样例输出2

1

本题的C++参考代码如下：

#include <iostream>

using namespace std;

char reverse(char c)

{

if(c=='\*')

c='o';

else

c='\*';

return c;

}

int main()

{

char a[1000],b[1000];

cin>>a>>b;

int move=0;

for(int i=0;a[i]!='\0';i++)

{

if(a[i]!=b[i])

{

a[i]=reverse(a[i]);

a[i+1]=reverse(a[i+1]);

move++;

}

}

cout<<move<<endl;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <stdlib.h>

int turn(char a[], char b[])

{

int i;

int n = 0;

for(i = 0;a[i]!='\0';i++)

{

if(a[i] == b[i])

{

continue;

}

else

{

b[i+1]=(b[i+1]=='\*'?'o':'\*');

n++;

}

}

return n;

}

int main(void)

{

char a[1000];

char b[1000];

gets(a);

gets(b);

printf("%d\n", turn(a, b));

return 0;

}

**历届试题 错误票据**

问题描述

某涉密单位下发了某种票据，并要在年终全部收回。

每张票据有唯一的ID号。全年所有票据的ID号是连续的，但ID的开始数码是随机选定的。

因为工作人员疏忽，在录入ID号的时候发生了一处错误，造成了某个ID断号，另外一个ID重号。

你的任务是通过编程，找出断号的ID和重号的ID。

假设断号不可能发生在最大和最小号。

输入格式

要求程序首先输入一个整数N(N<100)表示后面数据行数。

接着读入N行数据。

每行数据长度不等，是用空格分开的若干个（不大于100个）正整数（不大于100000），请注意行内和行末可能有多余的空格，你的程序需要能处理这些空格。

每个整数代表一个ID号。

输出格式

要求程序输出1行，含两个整数m n，用空格分隔。

其中，m表示断号ID，n表示重号ID

样例输入1

2  
5 6 8 11 9   
10 12 9

样例输出1

7 9

样例输入2

6  
164 178 108 109 180 155 141 159 104 182 179 118 137 184 115 124 125 129 168 196  
172 189 127 107 112 192 103 131 133 169 158   
128 102 110 148 139 157 140 195 197  
185 152 135 106 123 173 122 136 174 191 145 116 151 143 175 120 161 134 162 190  
149 138 142 146 199 126 165 156 153 193 144 166 170 121 171 132 101 194 187 188  
113 130 176 154 177 120 117 150 114 183 186 181 100 163 160 167 147 198 111 119

样例输出2

105 120

本题的C++参考代码如下：

#include<cstdio>

#include<cstring>

#include<cmath>

#include<cstdlib>

#include<algorithm>

#include<iostream>

#include<string>

using namespace std;

int a[100005];

int main(){

int n;

while(cin>>n){

char c[1005];

int b[105],t=0;

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

getchar();

while(n--){

gets(c);

int tc=strlen(c);

c[tc]=' ',c[tc+1]=0;

int tem=0;

for(int i=0;i<=tc;i++){

if(c[i]!=' '){

tem=tem\*10+c[i]-'0';

}

else{

a[tem]++;

b[t++]=tem;

tem=0;

}

}

}

int dh,ch;

sort(b,b+t);

for(int i=b[0];i<b[t-1];i++)

if(a[i]==0)

dh=i;

else if(a[i]==2)

ch=i;

cout<<dh<<' '<<ch<<endl;

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

int main()

{

int a[10001]={0};

long m,min=100000,max=0,i,n;

char c;

scanf("%d",&n);

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

while(1)

{

scanf("%ld",&m);

if(m>max) max=m;

if(m<min) min=m;

a[m]++;

c=getchar();

if(c!=' ') break;

}

for(i=min;i<=max;i++)

{

if(a[i]==0) printf("%ld ",i);

if(a[i]==2) m=i;

}

printf("%ld",m);

return 0;

}

**历届试题 剪格子**

问题描述

如下图所示，3 x 3 的格子中填写了一些整数。

+--\*--+--+  
|10\* 1|52|  
+--\*\*\*\*--+  
|20|30\* 1|  
\*\*\*\*\*\*\*--+  
| 1| 2| 3|  
+--+--+--+

我们沿着图中的星号线剪开，得到两个部分，每个部分的数字和都是60。

本题的要求就是请你编程判定：对给定的m x n 的格子中的整数，是否可以分割为两个部分，使得这两个区域的数字和相等。

如果存在多种解答，请输出包含左上角格子的那个区域包含的格子的最小数目。

如果无法分割，则输出 0。

输入格式

程序先读入两个整数 m n 用空格分割 (m,n<10)。

表示表格的宽度和高度。

接下来是n行，每行m个正整数，用空格分开。每个整数不大于10000。

输出格式

输出一个整数，表示在所有解中，包含左上角的分割区可能包含的最小的格子数目。

样例输入1

3 3  
10 1 52  
20 30 1  
1 2 3

样例输出1

3

样例输入2

4 3  
1 1 1 1  
1 30 80 2  
1 1 1 100

样例输出2

10

本题的C++参考代码如下：

#include<cstdio>

#include<cstring>

#include<cmath>

#include<cstdlib>

#include<algorithm>

#include<iostream>

#include<string>

#include<queue>

using namespace std;

int a[15][15];

int m,n;

int mx,sum;

int mi[][2]={{0,1},{0,-1},{1,0},{-1,0}};

struct ki{

int x,y;

int sum;

int tep;

};

void dfs(int x,int y,int s,int tep){

if(s>sum/(2.0))

return ;

else if(s==sum/(2.0)){

if(tep<mx)

mx=tep;

// printf("s=%d\n",s);

}

else{

for(int i=0;i<4;i++){

int tx=x+mi[i][0];

int ty=y+mi[i][1];

if(tx<0||ty<0||tx>=n||ty>=m)

continue;

if(a[tx][ty]!=-1){

int tem=a[tx][ty];

a[tx][ty]=-1;

// printf("tx=%d ty=%d s=%d tep=%d\n",tx,ty,s,tep);

dfs(tx,ty,s+tem,tep+1);

a[tx][ty]=tem;

}

}

}

}

int main(){

while(cin>>m>>n){

sum=0;

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

for(int k=0;k<m;k++){

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

sum+=a[i][k];

}

mx=10000;

int tem=a[0][0];

a[0][0]=-1;

dfs(0,0,tem,1);

printf("%d\n",mx==10000?0:mx);

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#define N 10

int num[N][N];

int tag[N][N] = {0};

int m, n;

int r = 100;

int find(int i, int j, int t, int ntag[][N])

{

int count = 0;

if (i < 0 || i >= n || j < 0 || j >= m || ntag[i][j] == 1)

return 0;

ntag[i][j] = 1;

if (tag[i][j] != t)

return 0;

count++;

count += find(i - 1, j, t, ntag);

count += find(i + 1, j, t, ntag);

count += find(i, j - 1, t, ntag);

count += find(i, j + 1, t, ntag);

return count;

}

int isbad()

{

int i, j, k = 0,ge2;

int t = tag[0][0];

int ntag1[N][N] = {0};

int ntag2[N][N] = {0};

int ge1 = find(0, 0, t, ntag1);

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

{

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

{

if (tag[i][j] != t)

{

k = 1;

break;

}

}

if (k == 1)

break;

}

if (i == n && j == m)

return 0;

ge2 = find(i, j, tag[i][j], ntag2);

return ge1 + ge2 != m \* n;

}

int bad(int i, int j)

{

int b;

if (i < 0 || i >= n || j < 0 || j >= m || tag[i][j] == 1)

return 1;

tag[i][j] = 1;

b = isbad();

tag[i][j] = 0;

return b;

}

void go(int i, int j, int k, int count)

{

if (bad(i, j) || count < num[i][j])

return;

k++;

if (count == num[i][j])

{

if (r > k)

r = k;

return;

}

tag[i][j] = 1;

count -= num[i][j];

go(i - 1, j, k, count);

go(i + 1, j, k, count);

go(i, j - 1, k, count);

go(i, j + 1, k, count);

tag[i][j] = 0;

}

int main()

{

int i, j;

int half = 0;

scanf("%d %d", &m, &n);

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

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

{

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

half += num[i][j];

}

if (half % 2 == 0 && half >= num[0][0] \* 2)

{

half /= 2;

go(0, 0, 0, half);

}

if (r == 100)

r = 0;

printf("%d", r);

return 0;

}

**历届试题 带分数**

问题描述

100 可以表示为带分数的形式：100 = 3 + 69258 / 714。

还可以表示为：100 = 82 + 3546 / 197。

注意特征：带分数中，数字1~9分别出现且只出现一次（不包含0）。

类似这样的带分数，100 有 11 种表示法。

输入格式

从标准输入读入一个正整数N (N<1000\*1000)

输出格式

程序输出该数字用数码1~9不重复不遗漏地组成带分数表示的全部种数。

注意：不要求输出每个表示，只统计有多少表示法！

样例输入1

100

样例输出1

11

样例输入2

105

样例输出2

6

本题的C++参考代码如下：

#include<iostream>

#include<cstring>

using namespace std;

int N,ans,digit,flag,full[9];

void div(int m)

{

while(m)

{

if(m%10!=0)

full[m%10-1]=1;

m/=10;

digit++;

}

}

bool check(int \*f)

{

int j;

for(j=0;j<9;j++)

if(!f[j])

return false;

return true;

}

void DFS(int a,int b,int c)

{

digit=0;

flag=0;

div(a);

div(b);

div(c);

if(digit>9){memset(full,0,sizeof(full));return ;}

if(check(full)){ans++;memset(full,0,sizeof(full));}

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

DFS(a,(c+1)\*b/c,c+1);

return ;

}

int main()

{

int i;

while(cin>>N)

{

ans=0;

memset(full,0,sizeof(full));

for(i=2;i<N-1;i++)

DFS(i,N-i,1);

cout<<ans<<endl;

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <stdlib.h>

typedef struct Interval

{

int pre;

int rear;

int satisfy;

}Interval;

Interval interval[7][5];

int count=0;

//初始化

void Init()

{

int i,j;

int value;

for(i=1;i<7;i++)

{

value=i;

for(j=1;j<5;j++)

{

interval[i][j].pre=value++;

interval[i][j].rear=value;

}

}

}

//数组初始化为0

void InitZero(int \*sign)

{

int i;

sign[0]=1;

for(i=1;i<10;i++)

{

sign[i]=0;

}

}

//将一个数的各个位上拆分，并在相应的位上赋值1

int Split(int \*sign,int value)

{

int index;

while(value)

{

index=value%10;

if(sign[index]==0) sign[index]=1;

else return 1;

value/=10;

}

return 0;

}

//计算一个数的位数

int CountBit(int value)

{

int n=0;

while(value)

{

n++;

value/=10;

}

return n;

}

//将一个整型数组转换成一个整数

int CreateInteger(int \*data,int n)

{

int i;

int value=0;

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

{

value=value\*10+data[i];

}

return value;

}

//检查是否每个数都用到

int Check(int \*sign)

{

int i;

for(i=1;i<10;i++)

{

if(sign[i]==0) return 0;

}

return 1;

}

//复制

void Copy(int \*sign,int \*temp\_sign)

{

int i;

for(i=0;i<10;i++)

{

temp\_sign[i]=sign[i];

}

}

//创建一个n位数的整数

void CreateNBitNumber(int \*sign,int \*data,int n,int m,int value,int value3)

{

if(n==m)

{

int value1=CreateInteger(data,n);

int value2=value1\*value;

int temp\_sign[10];

Copy(sign,temp\_sign);

if(!Split(temp\_sign,value2) && Check(temp\_sign))

{

count++;

}

}

else

{

int i;

for(i=1;i<10;i++)

{

if(sign[i]==0)

{

sign[i]=1;

data[m]=i;

CreateNBitNumber(sign,data,n,m+1,value,value3);

sign[i]=0;

}

}

}

}

//求出解

void Create(int value)

{

int i,j;

int sign[10];

int result;

int result\_n;

int n;

for(i=3;i<value;i++)

{

InitZero(sign);

if(Split(sign,i)) continue;

result=value-i;

result\_n=CountBit(result);

n=CountBit(i);

for(j=1;j<5;j++)

{

if( ((interval[result\_n][j].pre+j)==(9-n)) || ((interval[result\_n][j].rear+j)==(9-n)))

{

int data[5];

CreateNBitNumber(sign,data,j,0,result,i);

}

}

}

}

int main()

{

int value;

scanf("%d",&value);

Init();

Create(value);

printf("%d\n",count);

return 0;

}

**历届试题 打印十字图**

问题描述

小明为某机构设计了一个十字型的徽标（并非红十字会啊），如下所示：

..$$$$$$$$$$$$$..  
..$...........$..  
$$$.$$$$$$$$$.$$$  
$...$.......$...$  
$.$$$.$$$$$.$$$.$  
$.$...$...$...$.$  
$.$.$$$.$.$$$.$.$  
$.$.$...$...$.$.$  
$.$.$.$$$$$.$.$.$  
$.$.$...$...$.$.$  
$.$.$$$.$.$$$.$.$  
$.$...$...$...$.$  
$.$$$.$$$$$.$$$.$  
$...$.......$...$  
$$$.$$$$$$$$$.$$$  
..$...........$..  
..$$$$$$$$$$$$$..

对方同时也需要在电脑dos窗口中以字符的形式输出该标志，并能任意控制层数。

输入格式

一个正整数 n (n<30) 表示要求打印图形的层数。

输出格式

对应包围层数的该标志。

样例输入1

1

样例输出1

..$$$$$..  
..$...$..  
$$$.$.$$$  
$...$...$  
$.$$$$$.$  
$...$...$  
$$$.$.$$$  
..$...$..  
..$$$$$..

样例输入2

3

样例输出2

..$$$$$$$$$$$$$..  
..$...........$..  
$$$.$$$$$$$$$.$$$  
$...$.......$...$  
$.$$$.$$$$$.$$$.$  
$.$...$...$...$.$  
$.$.$$$.$.$$$.$.$  
$.$.$...$...$.$.$  
$.$.$.$$$$$.$.$.$  
$.$.$...$...$.$.$  
$.$.$$$.$.$$$.$.$  
$.$...$...$...$.$  
$.$$$.$$$$$.$$$.$  
$...$.......$...$  
$$$.$$$$$$$$$.$$$  
..$...........$..  
..$$$$$$$$$$$$$..

提示

请仔细观察样例，尤其要注意句点的数量和输出位置。

本题的C++参考代码如下：

#include <stdio.h>

#include <malloc.h>

#include <string.h>

int main()

{

int n,w,h,l,i,j,x,y,m;

char \*arry;

scanf("%d",&n);

w=h=5+n\*4;

arry=(char \*)malloc(w\*h);

memset(arry,'.',w\*h);

/\*

for(i=n\*2;i<w-n\*2;i++)//画基础图形

{

arry[i+((n+1)\*2)\*w]='$';

arry[i\*w+(n+1)\*2]='$';

}\*/

for(m=0;m<=n;m++)//画没一层

{

for(i=(m+1)\*2;i<w-(m+1)\*2;i++)//四边

{

x=m\*2;

y=i;

arry[x+y\*w]='$';

x=w-m\*2-1;

y=i;

arry[x+y\*w]='$';

x=i;

y=m\*2;

arry[x+y\*w]='$';

x=i;

y=w-m\*2-1;

arry[x+y\*w]='$';

}

for(i=m\*2;i<=(m+1)\*2;i++)//角

{

x=i;

y=(m+1)\*2;

arry[x+y\*w]='$';

x=(m+1)\*2;

y=i;

arry[x+y\*w]='$';

x=w-i-1;

y=(m+1)\*2;

arry[x+y\*w]='$';

x=w-(m+1)\*2-1;

y=i;

arry[x+y\*w]='$';

x=i;

y=h-(m+1)\*2-1;

arry[x+y\*w]='$';

x=(m+1)\*2;

y=h-i-1;

arry[x+y\*w]='$';

x=w-i-1;

y=h-(m+1)\*2-1;

arry[x+y\*w]='$';

x=w-(m+1)\*2-1;

y=h-i-1;

arry[x+y\*w]='$';

}

}

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

{

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

{

printf("%c",arry[i\*w+j]);

}

printf("\n");

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <malloc.h>

#include <string.h>

int main()

{

int n,w,h,l,i,j,x,y,m;

char \*arry;

scanf("%d",&n);

w=h=5+n\*4;

arry=(char \*)malloc(w\*h);

memset(arry,'.',w\*h);

/\*

for(i=n\*2;i<w-n\*2;i++)//画基础图形

{

arry[i+((n+1)\*2)\*w]='$';

arry[i\*w+(n+1)\*2]='$';

}\*/

for(m=0;m<=n;m++)//画没一层

{

for(i=(m+1)\*2;i<w-(m+1)\*2;i++)//四边

{

x=m\*2;

y=i;

arry[x+y\*w]='$';

x=w-m\*2-1;

y=i;

arry[x+y\*w]='$';

x=i;

y=m\*2;

arry[x+y\*w]='$';

x=i;

y=w-m\*2-1;

arry[x+y\*w]='$';

}

for(i=m\*2;i<=(m+1)\*2;i++)//角

{

x=i;

y=(m+1)\*2;

arry[x+y\*w]='$';

x=(m+1)\*2;

y=i;

arry[x+y\*w]='$';

x=w-i-1;

y=(m+1)\*2;

arry[x+y\*w]='$';

x=w-(m+1)\*2-1;

y=i;

arry[x+y\*w]='$';

x=i;

y=h-(m+1)\*2-1;

arry[x+y\*w]='$';

x=(m+1)\*2;

y=h-i-1;

arry[x+y\*w]='$';

x=w-i-1;

y=h-(m+1)\*2-1;

arry[x+y\*w]='$';

x=w-(m+1)\*2-1;

y=h-i-1;

arry[x+y\*w]='$';

}

}

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

{

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

{

printf("%c",arry[i\*w+j]);

}

printf("\n");

}

return 0;

}

**历届试题 核桃的数量**

问题描述

小张是软件项目经理，他带领3个开发组。工期紧，今天都在加班呢。为鼓舞士气，小张打算给每个组发一袋核桃（据传言能补脑）。他的要求是：

1. 各组的核桃数量必须相同

2. 各组内必须能平分核桃（当然是不能打碎的）

3. 尽量提供满足1,2条件的最小数量（节约闹革命嘛）

输入格式

输入包含三个正整数a, b, c，表示每个组正在加班的人数，用空格分开（a,b,c<30）

输出格式

输出一个正整数，表示每袋核桃的数量。

样例输入1

2 4 5

样例输出1

20

样例输入2

3 1 1

样例输出2

3

本题的C++参考代码如下：

#include <stdio.h>

int main(int argc, char \*argv[]) {

int a,b,c,i;

int all;

scanf("%d %d %d",&a,&b,&c);

all=a\*b\*c;

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

{

if(i%a==0&&i%b==0&&i%c==0)

{

printf("%d",i);

break;

}

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <stdlib.h>

//最小公倍数

int LCM(int num1,int num2,int num3)

{

int value=num1;

while(value%num1!=0||value%num2!=0||value%num3!=0)

{

value+=num1;

}

return value;

}

int main()

{

int num1,num2,num3;

scanf("%d%d%d",&num1,&num2,&num3);

printf("%d\n",LCM(num1,num2,num3));

return 0;

}