数论算法

**//gcd.cpp**

#include<iostream>

using namespace std;

int gcd(int a,int b) {

int c;

if(a ==0) return b;

while(b!=0) c=b,b=a%b,a=c;

return a;

}

**//gcd.cpp**

long long gcd(long long a, long long b){

if ((b==0)&(a==0)) //表示无最大公因数

return -1;

return b == 0?a:gcd(b, a%b);

}

**//extend\_gcd.cpp（非递归）**

long long extend\_gcd(long long a, long long b, long long &x, long long &y){

long long x0,x1,x2, y0,y1,y2;

long long r0,r1,r2, q;

if((a== 0)&&(b==0)){

x=0; y=0;

return -1;

}

if((a== 0)&&(b!=0)){

x = 0; y = 1 ;

return b;

}

if((b == 0) &&(a!=0)) {

x = 1; y = 0 ;

return a;

}

if((a!= 0) &&(b!=0)){

x0=0; x1=1; r0= a;

y0 = 1; r1 = b;

r2=r0 % r1;y1 =0- r0/r1;

x2=1; y2=y1;

while((r1%r2) != 0) {

r0=r1; r1=r2;

q= r0/r1;

x2=x0-x1\*q; y2=y0-y1\*q;

x0=x1; x1=x2; y0=y1; y1=y2; r2=r0 % r1;

}

x = x2; y = y2;

return r2;

}

}

**//extend\_gcd.cpp（递归）**

long long extend\_gcd(long long a, long long b, long long &x,long long &y){

long long t,m;

if ((b==0)&(a==0)) return -1;

if (b==0){

x=1;y=0;

return a;

}

else {

m=extend\_gcd(b,a % b,x,y);

t=x;x=y;y=t-(a/b)\*y;

}

return m;

}

int main()

{

long long a,b,x,y;

while(cin>>a>>b){

long long t1 = gcd(a,b);

long long t2 = extend\_gcd(a,b,x,y);

cout<<t1<<" "<<x<<" "<<y<<endl;

}

}

**//ax+by=c.cpp**

/\*ax+by=c的整数解算法 \*/

/\*设t=gcd(a, b)，记a=mt，b=nt

求特解：求整系数方程ax+by=t的一个整数解x0，y0，

求一般解：若 t不是c的因数，则整系数方程ax+by=c无整数解；

若 t是c的因数，记c=gt，则整系数方程ax+by=c一般解为：

x=gx0+bk, y=gy0-ak, k为任何整数

如果x1<=x<=x2&&y1<=y<=y2;

存在这样的x,y输出“have”反之“nothing”

\*/

#include<iostream>

using namespace std;

int main()

{

long long a,b,c,x,y,x0,y0,x1,x2,y1,y2,kx1,kx2,ky1,ky2;

while(cin>>a>>b>>c>>x1>>x2>>y1>>y2){

long long t = extend\_gcd(a,b,x0,y0);//拓展欧几里得

if(c%t == 0){

long long g = c/t;

long long m = a/t;

long long n = b/t;

kx1 = (x1- g\*x0-1)/n;

kx2 = (x2- g\*x0+1)/n;

ky2 = (y1- g\*y0+1)/(-m);

ky1 = (y2- g\*y0-1)/(-m);

for(int k = kx1+1;k<kx2;k++)

if(k>ky1&&k<ky2){

cout<<"have"<<endl;

x = g\*x0+n\*k;

y = g\*y0-m\*k;

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

}

}

else cout<<"nothing"<<endl;

}

}

**//ax === b mod m.cpp**

/\*ax === b(mod m)相当于ax +my = b

根据前面求 ax+my = b的步骤：

（1）求x,y,t = gcd(a,m)

（2）如果b%d!=0无解

（3） 由ax+my = t,得a(x\*b/t)+m(y\*b/t) = b

于是ax===b(mod m)的一个解为x = x\*b/t

（4）ax === b(mod m)的所有解为：

x = (x\*b/t+ i\*m/t)mod m,i = 0,1,2,...d-1

\*/

#include<iostream>

using namespace std;

int main(){

long long a,b,x,m,y;

while(cin>>a>>b>>m){

long long t = extend\_gcd(a,m,x,y);//拓展欧几里得

if(b%t == 0){

x =(x\*b/t)%m;

while(x<0)x = (x+m)%m;

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

cout<<(x+i\*(m/t))%m<<endl;

}

}

else cout<<"no solutions"<<endl;

}

}

**//逆元.cpp**

/\*

ax === 1 mod m

1.费马小定理(m 为质数 由a^(m-1) === 1(mod m)) 所以 a 的逆元为 a^(m-2)//fast\_mod

2.欧拉定理:设a和m 是整数，（m>0）。记b(m)是1到m的整数中与m互素的整数的个数，则a^(b(m)) ≡1 (mod m) a 的逆元为 a^(b(m)-1)//fast\_mod）

3.a m 互质 等价于ax +my = 1

ax+my = 1的步骤：

（1）求x,y,t = gcd(a,m)（2）如果t != 1无解（3）于是ax===1(mod m)的一个解为x = x

\*/

#include<iostream>

using namespace std;

//extend\_gcd(a, m, &x, &y); x为所求解 when gcd(a,m) == 1

long mod\_reverse(long long a, long long m){

long long y=0,x=1,r=a%m, q, t, mm=m;//初始化

if(r<0)r=r+m;

while((m%r) != 0) {

a=m; m=r;

q= a/m; r=a % m;

t=x; x=y-x\*q; y=t;

}

if(r!=1) return 0;

if(x<0) x=x+mm;

return x;

}

int main(){

long long a,m;

while(cin>>a>>m){

cout<<mod\_reverse(a,m)<<endl;

}

}

**//中国剩余定理.cpp**

/\*设 m1，m2,m3...mn是两两互素的n个正整数，

记m = m1m2m3...mn ，Mi = n/mi

，1≤i≤n，那么，下列模线性方程组

x === b1(mod m1)

x === b2(mod m2)

...

x === bn(mod mn)

的解为x0 = (b1m1y1+b2m2y2+...+bnmnyn)，

其中yi 是Miyi ===1(mod mi) 的解，i=1，2，3，…，n

\*/

#include<iostream>

using namespace std;

const int maxn = 1000;

long long m[maxn];

long long b[maxn];

long long ChinaRemainder(int n,long long m0[], long long b[]){

long long d,x,y,t,m=1,a=0,Mi;

int i;

for (i=0; i< n ; i++) m=m\*m0[i];

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

Mi=m / m0[i];

extend\_gcd(Mi, m0[i], x, y); //求逆元拓展欧几里得

//或x=mod\_reverse(MM, m0[i]);

a=(a+Mi\*x\*b[i]) % m;

}

while(a<0)a = a+m;

return a;

}

int main(){

int n;

while(cin>>n){

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

cin>>b[i]>>m[i];

cout<<ChinaRemainder(n,m,b)<<endl;

}

}

**//fast\_mod.cpp**

/\*n^m(mod c)

A^13 = A^8\*A^4\*A^1用二进制表示A^(1101) = A^(1000)\*A^(100)\*A^(1)

例如：：N = 13 = 1101（2），N>>1 = 110（2） ，

\*/

long long fast\_mod(long long n,long long m,long long c){ //求快速幂

long long ret=1;

while(m){

if(m&1)ret = ret\*n%c;

n=n\*n%c;//这里保证m的移位和n的幂数对等，右移x位就为n的2^x次幂

m>>=1;

}

return ret;

}

**RSA加密**

/\*两个大素数p,q n = p\*q

n 的欧拉数 b(n) = (p-1)(q-1)

e 与 b(n)互质且0<=e<=b(n)-1

解密密匙d 满足d\*e===1(mod b(n)) (n d互质)

e n 为公钥 d为私钥

加密信息 m

加密 ci = mi^e(mod n)

解密 mi = ci^d(mod n)

\*/