扩展欧几里得

即 $ax+by=1$ 给定 a,b 求 x,y

和欧几里得算法一样，辗转相除，每次更新外层 x,y 对即可。

需要注意的是 $ax+by=c$ 有解当且仅当 $\gcd(a,b) |c$ 需要进行特判。

原根

由欧拉定理，若 $(a,n)=1$ 则 $a^{\varphi(m)}\equiv 1\pmod m$

类似单位复根，数 m 的原根 $g\in[1,m],(g,m)=1$ 满足 $\{g,g^2,\ldots,g^{\varphi(m)}\}$ 构成模 m 的一个既约剩余系。易得，对于质数 m 这个既约剩余系的取值范围为 $[1,m-1]$

原根的另一个定义是 $\forall p<\varphi(m), g^{\varphi(m)}\neq 1$ 即 $\varphi(m)$ 是最小的让 $g^d\equiv 1\pmod m$ 的正整数 d

这两个定义可以互推，桥梁是 $\forall i,j\in[1,\varphi(m)],i\neq j,g^{i}\neq g^j\pmod m$

判断是否有原根

数 n 有原根的充要条件为 n 可表示为 $2,4,p^{a},2p^{a}$ 的形式 p 为奇素数 a 为正整数）。这可以推出：质数必有原根。

求一个原根

可以通过枚举正整数 g 并验证是否满足原根的第二个定义。

设 d_i 为 $\varphi(m)$ 的所有约数，当 $\forall i,g^{d_i}\neq 1$ 时， g 为 m 的一个原根。

可以优化这个过程：设 p_i 为 $\varphi(m)$ 的所有质因数，则当 $\forall i,g^{\frac{\varphi(m)}{p_i}}\neq 1$ 时， g 为 m 的一个原根。

证明很简单：若 $a^{x}\equiv 1\pmod m$ 则 $a^{kx}\equiv 1\pmod m$ $\frac {\varphi(m)}{p_i}$ 是除了含有 $p_i^{k_i}$ 因子之外，所有 d_i 的倍数，故如果 $\exists k\in \mathbb N,k\cdot d_j=\frac{\varphi(m)}{p_i},g^{d_j\cdot k}\equiv g^{\frac{\varphi(m)}{p_i}}\equiv 1 $ 的因子通过其他部分验证。验证一圈下来除了 $\prod_{i}p_i^{k_i}$即$\varphi(m)$ 之外（本身就要求是 1）别的都得到了验证。

这样优化下来，复杂度几乎是一个常数（质因数个数增长极慢）。

求所有原根

假设已经求出 m 的一个原根 g ，显然对于集合 $S_0=\{g^s|1\leq s\leq \varphi(m)\}$ 中的每一个元素 x 都有 $x^{\varphi(m)}\equiv 1$ ，根据第二个定义， $\forall j\in[1,\varphi(m)],(g^s)^{j}\bmod m\ne 1=(g^s)^{0}$ ，即只有 $\forall j\in[1,\varphi(m)),j\cdot s\bmod \varphi(m) \ne 0)$ ，这要求 $(s,\varphi(m))=1$ 。

故集合 $S=\{g^s\bmod m|1\leq s\leq \varphi(m),(\varphi(m),s)=1\}$ 中包含所有关于 m 不同余（由原根 g 的性质得）的原根，个数为 $\varphi(\varphi(m))$ 。

BSGS

即， $a^{x}\equiv b\pmod c$ ，给定 a,b,c ，求出 x 。

设 $x=iB+t$ （或者 $x=iB-t$ 都可），其中 B 为块大小，先设为 $\left\lceil\sqrt c\right\rceil$ 。

那么有 $a^{iB}\equiv b\cdot a^{-t}$ ，把右半部分预处理并扔进 map 里，从小到大枚举 $i\in[0,B]$ ，通过扩欧解出来右半边 a^{-t} 应当取的值，查map判断，即可 $O(\sqrt c)$ 求解。

N次剩余

即， $x^n\equiv a\pmod m$ ，给定 $m\in prime,n,a$ ，求出 x 。

前置芝士：扩欧，原根， BSGS

模板题：HDU3930 Broot（数据范围有误，应为 $1e12$ ，数据较弱，提供几组稍强数据）

解法1

设 m 的一个原根为 g ，由于 $a,x\in[0,m-1]$ ，必有 $x=0$ 或正整数 y 满足 $g^{y}=x$ ， $a=0$ 或正整数 z 满足 $g^{z}=a$ 。

容易通过 BSGS 求出 z ，现在 $g^{yn}\equiv g^{z} \pmod m$ 式中只有 y 未知。

式子等价于， $yn\equiv z\pmod \varphi(m)$ ，可以用扩欧求出 y 的一个解 y_0 。

设 $gcd=\gcd(n,\varphi(m))$ ， y 的解集为 $\{k\in[0, gcd)|y_0+k\cdot \frac{\varphi(m)}{gcd}\}$ ，再根据此求出 x 即可。

```
#include<cstdio>
#include<algorithm>
#include<queue>
#include<map>
```

```
#include<cstring>
#include<cmath>
#include<cstdlib>
#include<set>
#include<unordered_map>
#include<vector>
typedef long long ll;
using namespace std;
#define pb push_back
#define mp make_pair
#define fi first
#define se second
#define N 1238747
#define hash HASH
using namespace std;
typedef long long ll;
ll md(ll x,ll m){return x>=m?x-m:x;}
ll mul(ll a,ll b,ll m){
    ll ans=0;
    a=(a%m+m)%m;b=(b%m+m)%m;
    for(;b;b>>=1,a=md(a+a,m))if(b&1)ans=md(ans+a,m);
    return ans;
}
ll qp(ll a,ll p,ll mod){
    ll ans=1;
    for(;p;p>>=1,a=mul(a,a,mod))if(p&1)ans=mul(ans,a,mod);
    return ans;
}

//---------------BSGS----------------
struct Triple{
    ll x,y,z;
    Triple(ll a,ll b,ll c) :x(a),y(b),z(c) {}
};
Triple exgcd(ll a,ll b){
    if(b==0) return Triple(1,0,a);
    const Triple last=exgcd(b,a%b);
    return Triple(last.y,last.x-a/b*last.y,last.z);
}
int A,B,C;
set<ll>S;
unordered_map<ll,int>ma;
ll bsgs(ll A,ll B,ll C){
    ll sqrtn=ceil(sqrt(C)),base=1;
    ma.clear();
    for(int i=0;i<sqrtn;i++){
        if(base)ma[base]=i;
        base=mul(base,A,C);
    }
    ll i=0,j=-1,D=1;
    S.clear();
```

```
    for(;i<sqrtn;i++){
        Triple res=exgcd(D,C);
        const ll c=C/res.z;
        res.x=(mul(res.x,(B/res.z),c)+c)%c;
        if(ma.count(res.x)){
            j=ma[res.x];
            return (i*sqrtn+j);
        }
        D=mul(D,base,C);
    }
    return -1;
}
//------------sieve----------------
#define M 1000000
#define P 1000000
int isnp[M],pri[P],cnt_prime;
void sieve(){
    int i,j;
    isnp[0]=isnp[1]=1;
    for(i=2;i<M;i++){
        if(!isnp[i])pri[cnt_prime++]=i;
        for(j=0;j<cnt_prime&&1LL*i*pri[j]<M;j++){
            isnp[i*pri[j]]=1;
            if(i%pri[j]==0)break;
        }
    }
}
//-------------pri factor------------
vector<pair<ll,int>>pris;
void get_pf(ll x){
    int i;
    pris.clear();
    for(i=0;i<cnt_prime&&1LL*pri[i]*pri[i]<=x;i++){
        if(x%pri[i]==0){
            int count=0;
            while(x%pri[i]==0)count++,x/=pri[i];
            pris.pb(mp(pri[i],count));
        }
    }
    if(x>1)pris.pb(mp(x,1));
}
//-----------primitive root-------------
int jud_proot(int g,int index,ll d,ll p){
    // optimized
    int i;
    for(i=0;i<pris.size();i++)if(qp(g,(p-1)/pris[i].first,p)==1)return 0;
    return 1;

    // origin
    /*if(index==pris.size())return !(qp(g,d,p)==1&&d!=p-1);
```

```
    ll tmp=1;int i;
    ll pri=pris[index].fi,count=pris[index].se;
    for(i=0;i<=count;i++){
        if(!jud_proot(g,index+1,d*tmp,p))return 0;
        tmp=tmp*pri;
    }
    return 1;*/
}
int get_proot(ll p){
    int g=2;
    while(!jud_proot(g,0,1,p))g++;
    return g;
}
int main(){
    ll n,p,a;
    int cas=0,i;
    sieve();
    while(~scanf("%lld%lld%lld",&n,&p,&a)){
        a%=p;
        printf("case%d:\n",++cas);
        if(!a){
            if(n==0)printf("-1\n");
            else printf("0\n");
            continue;
        }
        get_pf(p-1);
        int g=get_proot(p);
        ll y=bsgs(g,a,p);
        if(y==-1){printf("-1\n");continue;}
        // nX+(p-1)Y=y
        Triple tri=exgcd(n,p-1);
        ll gcd=tri.z;
        ll X=tri.x;
        if(y%gcd){printf("-1\n");continue;}
        vector<ll>ans;ans.clear();
        ll a1=n/gcd,a2=(p-1)/gcd,a3=y/gcd;
        //a1*X+a2*Y=a3 (mod a2), (a1,a2)=1
        ans.pb((mul(X,a3,a2)+a2)%a2);
        for(i=1;i<gcd;i++)ans.pb(ans[i-1]+a2);
        for(i=0;i<gcd;i++)ans[i]=qp(g,ans[i],p);
        sort(ans.begin(),ans.end());
        for(i=0;i<gcd;i++)printf("%lld\n",ans[i]);
    }
    return 0;
}
```

解法2

类似的，设 m 的一个原根为 g ，由于 $a,x\in[0,m-1]$ ，必有 $x=0$ 或正整数 y 满足 $g^{y}=x$ ，

现在 $(g^{n})^{y}\equiv a \pmod m$ 式中只有 y 未知。

通过 BSGS 可以求出所有符合要求的 y 但这里的 BSGS 需要使用 map<int,vector>，因为 g^n 不是原根 a^{t} 部分并非一对一关系。过后根据 y 算出 x 即可。

这两种解法都通过了原根进行转换，第一种方法由于没有使用较为复杂的 map，常数比较小；第二种方法则更加简便、易写。

```
#include<cstdio>
#include<algorithm>
#include<queue>
#include<map>
#include<cstring>
#include<cmath>
#include<cstdlib>
#include<set>
#include<unordered_map>
#include<vector>
typedef long long ll;
using namespace std;
#define pb push_back
#define mp make_pair
#define fi first
#define se second
using namespace std;
typedef long long ll;
ll md(ll x,ll m){return x>=m?x-m:x;}
ll mul(ll a,ll b,ll m){
    ll ans=0;
    a=(a%m+m)%m;b=(b%m+m)%m;
    for(;b;b>>=1,a=md(a+a,m))if(b&1)ans=md(ans+a,m);
    return ans;
}
ll qp(ll a,ll p,ll mod){
    ll ans=1;
    for(;p;p>>=1,a=mul(a,a,mod))if(p&1)ans=mul(ans,a,mod);
    return ans;
}
//---------------BSGS----------------
struct Triple{
    ll x,y,z;
    Triple(ll a,ll b,ll c) :x(a),y(b),z(c) {}
};
Triple exgcd(ll a,ll b){
    if(b==0) return Triple(1,0,a);
    const Triple last=exgcd(b,a%b);
    return Triple(last.y,last.x-a/b*last.y,last.z);
}
set<ll>S;
unordered_map<ll,vector<int>>ma;
```

```
void bsgs(ll A,ll B,ll C,int g){
    ll sqrtn=ceil(sqrt(C)),base=1;
    ma.clear();
    for(int i=0;i<sqrtn;i++){
        if(base)ma[base].pb(i);
        base=mul(base,A,C);
    }
    ll i=0,j=-1,D=1;
    S.clear();
    for(;i<sqrtn;i++){
        Triple res=exgcd(D,C);
        ll c=C/res.z;
        res.x=(mul(res.x,B/res.z,c)+c)%c;
        if(ma.count(res.x)){
            for(auto j:ma[res.x])
                  S.insert(qp(g,(i*sqrtn+j),C));
        }
        D=mul(D,base,C);
     }
}
//------------sieve----------------
#define M 1000005
#define P 1000000
int isnp[M],pri[P],cnt_prime;
void sieve(){
    int i,j;
    isnp[0]=isnp[1]=1;
    for(i=2;i<M;i++){
        if(!isnp[i])pri[cnt_prime++]=i;
        for(j=0;j<cnt_prime&&1LL*i*pri[j]<M;j++){
            isnp[i*pri[j]]=1;
            if(i%pri[j]==0)break;
        }
    }
}
//-------------pri factor------------
vector<pair<ll,int>>pris;
void get_pf(ll x){
    int i;
    pris.clear();
    for(i=0;i<cnt_prime&&1LL*pri[i]*pri[i]<=x;i++){
        if(x%pri[i]==0){
            int count=0;
            while(x%pri[i]==0)count++,x/=pri[i];
            pris.pb(mp(pri[i],count));
        }
    }
    if(x>1)pris.pb(mp(x,1));
}
//-----------primitive root-------------
int jud_proot(int g,int index,ll d,ll p){
```

```
    // optimized
    int i;
    for(i=0;i<pris.size();i++)if(qp(g,(p-1)/pris[i].first,p)==1)return 0;
    return 1;

    // origin
    /*if(index==pris.size())return !(qp(g,d,p)==1&&d!=p-1);
    ll tmp=1;int i;
    ll pri=pris[index].fi,count=pris[index].se;
    for(i=0;i<=count;i++){
        if(!jud_proot(g,index+1,d*tmp,p))return 0;
        tmp=tmp*pri;
    }
    return 1;*/
}
int get_proot(ll p){
    int g=2;
    while(!jud_proot(g,0,1,p))g++;
    return g;
}
int main(){
    ll n,p,a;
    //freopen("in.txt","r",stdin);
    int cas=0;
    sieve();
    while(~scanf("%lld%lld%lld",&n,&p,&a)){
        a%=p;
        printf("case%d:\n",++cas);
        if(!a){
            if(n==0)printf("-1\n");
            else printf("0\n");
            continue;
        }
        get_pf(p-1);
        int g=get_proot(p);
        //printf("%d\n",g);
        ll bas=qp(g,n,p);
        bsgs(bas,a,p,g);
        if(!S.size())printf("-1\n");
        else for(auto x:S)printf("%lld\n",x);
    }
    return 0;
}
```