[**[省选前题目整理][SGU 261]Discrete Roots(扩展欧几里得+中国剩余定理+原根+大步小步算法)**](http://blog.csdn.net/qpswwww/article/details/44652503)

## 题目链接

<http://acm.hust.edu.cn/vjudge/problem/viewProblem.action?id=22207>

## 题目大意

已知k、y、p，求xk≡y(modp)的所有可行解

## 思路

首先，我们求出modp的原根g。   
不妨设gq≡y(modp)，q可以通过大步小步[**算法**](http://lib.csdn.net/base/datastructure)求得。   
由于在1≤x<p时，modp意义下所有的数字均可以用g的幂次表示。因此一定存在等式gi≡k(modp−1)。这个可以通过中国剩余定理找出所有的可行解。

—————–以下来闲扯一下大步小步算法——————-   
大步小步算法看上去非常的暴力，个人认为其思路就是基于分块思想。假设已知x、y、p，求xk≡y(modp)的k，那么首先我们要对于所有的i≤p√,在map中记录下xi对应于i的映射。然后维护一个值xt，初始时t=0，每次t+=p√,若在map中能够查找到yxt(m−2)(这个操作复杂度只是对数),那么就找到了答案，k=t+mp[yxt(m−2)]，mp[yxt(m−2)]=yxt(m−2)在map中映射的幂

## 代码

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <algorithm>

#include <vector>

#include <map>

#include <cmath>

using namespace std;

typedef long long int LL;

vector<LL>divisor,ans;

LL extGCD(LL a,LL b,LL &x,LL &y)

{

if(!b)

{

x=1;

y=0;

return a;

}

LL gcd=extGCD(b,a%b,x,y);

LL tmp=x;

x=y;

y=tmp-a/b\*y;

return gcd;

}

LL fastPow(LL base,LL pow,LL mod)

{

LL ans=1;

while(pow)

{

if(pow&1) ans=(ans\*base)%mod;

base=(base\*base)%mod;

pow>>=1;

}

return ans;

}

bool g\_test(LL g,LL p) //检查g是否是mod p的原根

{

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

if(fastPow(g,(p-1)/divisor[i],p)==1)

return false;

return true;

}

LL PrimitiveRoot(LL p) //求mod p的原根

{

divisor.clear();

LL tmp=p-1;

for(LL i=2;i\*i<=tmp;i++)

{

if(tmp%i==0)

{

divisor.push\_back(i);

while(tmp%i==0) tmp/=i;

}

}

if(tmp!=1) divisor.push\_back(tmp);

LL g=0; //爆枚原根g

while(++g)

if(g\_test(g,p))

return g;

return -1;

}

LL DiscreteLog(LL x,LL n,LL m) //求离散对数,x^y=n(mod m)，求y

{

map<LL,LL>table; //一个有序表

LL sqrtm=(LL)(sqrt(m)+0.5);

LL tmp=1; //x^i=tmp

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

{

table[tmp]=i;

tmp=tmp\*x%m;

}

LL step=tmp; //t=x^sqrt(m)

tmp=1;

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

{

LL d=n\*fastPow(tmp,m-2,m)%m;

if(table.count(d))

return i\*sqrtm+table[d];

tmp=tmp\*step%m;

}

return -1;

}

void CRT(LL a,LL b,LL n) //ax=b(mod n)

{

LL x,y,gcd;

gcd=extGCD(a,n,x,y);

if(b%gcd==0) //有解

{

x=(x%n+n)%n;

ans.push\_back(x\*(b/gcd)%(n/gcd));

for(LL i=1;i<gcd;i++)

ans.push\_back((ans[0]+i\*n/gcd)%n);

}

}

int main()

{

LL a,p,g,q,k;

while(scanf("%lld%lld%lld",&p,&k,&a)!=EOF) //x^k=a(mod p)

{

if(!a) //a=0要特判,x只能为0

{

printf("1\n0\n");

continue;

}

g=PrimitiveRoot(p); //g=mod p的原根

q=DiscreteLog(g,a,p); //g^q=a(mod p)

CRT(k,q,p-1); //问题变为kx=q(mod p-1)的所有正整数解

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

ans[i]=fastPow(g,ans[i],p); //ans[i]=g^x mod p

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

printf("%d\n",(int)ans.size());

for(int i=0;i<(int)ans.size();i++) printf("%lld\n",ans[i]);

}

return 0;

}