# ACM_高次同余方程 - xiaotan1314 - CSDN博客

[cpp]view plaincopy

```
/*poj 3243
 *解决高次同余方程的应用，已知 X^Y = K mod Z, 及X,Z,K的值，求 Y 的值
*/
#include
#include
#include
usingnamespace std;
#define lint __int64
#define MAXN 131071
struct HashNode { lint data, id, next; };
HashNode hash[MAXN<<1 span>
bool flag[MAXN<<1 span>
lint top;
void Insert ( lint a, lint b )
{
    lint k = b & MAXN;
if ( flag[k] == false )
    {
        flag[k] = true;
        hash[k].next = -1;
        hash[k].id = a;
        hash[k].data = b;
return;
    }
while( hash[k].next != -1 )
    {
if( hash[k].data == b ) return;
        k = hash[k].next;
    }
if ( hash[k].data == b ) return;
    hash[k].next = ++top;
    hash[top].next = -1;
    hash[top].id = a;
    hash[top].data = b;
}
lint Find ( lint b )
{
    lint k = b & MAXN;
if( flag[k] == false ) return -1;
while ( k != -1 )
    {
if( hash[k].data == b ) return hash[k].id;
        k = hash[k].next;
    }
return -1;
}
lint gcd ( lint a, lint b )
{
return b ? gcd ( b, a % b ) : a;
}
lint ext_gcd (lint a, lint b, lint& x, lint& y )
{
    lint t, ret;
if ( b == 0 )
    {
        x = 1, y = 0;
return a;
    }
    ret = ext_gcd ( b, a % b, x, y );
    t = x, x = y, y = t - a / b * y;
return ret;
}
lint mod_exp ( lint a, lint b, lint n )
```

```
{
    lint ret = 1;
    a = a % n;
while ( b >= 1 )
    {
if( b & 1 )
            ret = ret * a % n;
        a = a * a % n;
        b >>= 1;
    }
return ret;
}
lint BabyStep_GiantStep ( lint A, lint B, lint C )
{
    top = MAXN;  B %= C;
    lint tmp = 1, i;
for ( i = 0; i < tmp=" tmp * A % C, i++ )
if ( tmp == B % C ) return i;
    lint D = 1, cnt = 0;
while( (tmp = gcd(A,C)) !=1 )
    {
if( B % tmp ) return -1;
        C /= tmp;
        B /= tmp;
        D = D * A / tmp % C;
        cnt++;
    }
    lint M = (lint)ceil(sqrt(C+0.0));
for ( tmp = 1, i = 0; i < Mtmp=" tmp * A % C, i++ )
        Insert ( i, tmp );
    lint x, y, K = mod_exp( A, M, C );
for ( i = 0; i < Mispan>
    {
        ext_gcd ( D, C, x, y ); // D * X = 1 ( mod C )
        tmp = ((B * x) % C + C) % C;
if( (y = Find(tmp)) != -1 )
return i * M + y + cnt;
        D = D * K % C;
    }
return -1;
}
int main()
{
    lint A, B, C;
while( scanf("%I64d%I64d%I64d",&A,&C,&B ) !=EOF )
    {
if ( !A && !B && !C ) break;
        memset(flag,0,sizeof(flag));
        lint tmp = BabyStep_GiantStep ( A, B, C );
if ( tmp == -1 )puts("No Solution");
else printf("%I64d\n",tmp);
    }
return 0;
}
```