原创 # 【SPOJ】Power Modulo Inverted 扩展BSGS

2019-09-10 20:20:00 我是一只计算鸡 阅读数 11 更多

$a^x \equiv b \ mod \ n$

当$gcd(a, n) \neq 1$时，我们就不能用BSGS算法求解。既然不互质时，我们无法求解，那我们就想办法让$gcd(a, n) = 1$

$a * a^{x-1} \equiv b \ mod \ n$

$\frac{a}{g} * a^{x-1} \equiv \frac{b}{g} \ mod \ \frac{n}{g}, g = gcd(a, n)$

$a^{x-1} \equiv b * a^{-1} \ mod \ \frac{n}{g}$

如果$gcd(a, \frac{n}{g}) = 1$，就成了我们熟悉的BSGS算法了。

如果$gcd(a, \frac{n}{g}) \neq 1$，$a * a^{x-2} = b * a^{-1} \ mod \ \frac{n}{g}$

$\frac{a}{g'} * a^{x-2} \equiv \frac{b}{g'} * a^{-1} \ mod \ \frac{n}{g * g'}$，$g' = gcd(a, \frac{n}{g})$

如果$b \% g \neq 1$，则方程无解

否者一直除到互质为止

假设进行了t次除法，$a^{x-t} \equiv b * inv(\frac{a}{\prod_{i=1}^{t} g}) \ mod \ \frac{n}{\prod_{x=1}^{t} g}$

```
#include<bits/stdc++.h>
using namespace std;
const int MOD = 76543;
typedef long long ll;
ll hs[MOD], id[MOD], _next[MOD], head[MOD], top;
// hs hash表 记录存的值
// id 值所所对应的id
// _next 相同hash值的上一个序号
// head hash 值相同的最后一个元素的序号
// 其实就是用链式前向星实现的hash表
ll gcd(ll a, ll b)  //最大公约数
{
        if(b == 0) return a;
        while(b) {
                ll t = a;
                a = b;
                b = t % b;
        }
        return a;
}
ll exgcd(ll a, ll b, ll &x, ll &y)  // 扩展欧几里得
{
        if(b == 0) {
                x = 1;
                y = 0;
                return a;
        }
        ll d = exgcd(b, a % b, x, y);
        ll t = x;
        x = y;
        y = t - a / b * y;
        return d;
}
ll inv(ll a, ll p)  // 求解逆元
{
```

```
        ll x, y;
        exgcd(a, p, x, y);
        return (x % p + p) % p;  // 不能直接 return x, 因为 x 可能为负数
}
void insert(ll x, ll y)  // 插入hash表
{
        ll k = x % MOD;
        hs[top] = x;
        id[top] = y;
        _next[top] = head[k];
        head[k] = top ++;
        return ;
}
ll find(ll x)  // 查找
{
        ll k = x % MOD;
        for(ll i = head[k]; i != -1; i = _next[i]) if(hs[i] == x) return id[i];
        return -1;
}
ll EXBSGS(ll a, ll b, ll n) // 扩展大步小步算法, gcd(a, n) != 1
{
        memset(head, -1, sizeof(head)); // hash表初始化
        top = 1;
        if(b == 1 ) return 0;  // a ^ 0 = 1 mod n
        ll cnt = 0, tmp = 1, d;
        while((d = gcd(a, n)) != 1) {  // a, n 不互质
                if(b % d != 0) return -1;  // 无解
                b /= d;
                n /= d;
                tmp = tmp * (a / d) % n;
                cnt ++;
                if(b == tmp) return cnt;  // a^{x - cnt} = 1 mod n
        }
        b = b * inv(tmp, n) % n;
        ll m = sqrt(n * 1.0), j;
        ll x = 1, p = 1;
        for(ll i = 0; i < m; i++, p = p * a % n) insert(b * p % n, i); // 枚举 右边 1 ~ sqrt(n)
        for(ll i = m; ; i += m) { // 枚举左边sqrt(n) * {1 ~ sqrt(n)}
                if((j = find( x = x * p % n)) != -1) return i - j + cnt;
                if(i > n) break;
        }
        return -1;
}
int main()
{
        ll a, b, n;
        while(scanf("%lld %lld %lld", &a, &n, &b) == 3 && (a || n || b)) {
                ll ans = EXBSGS(a, b, n);
                if(ans == -1) printf("No Solution\n");
                else printf("%lld\n", ans);
        }
        return 0;
}
```