原创 【SGU 261】BSGS算法详解

2019-09-05 20:49:21 我是一只计算鸡 阅读数 12 更多

BSGS(baby-step gaint-step)，大步小步算法，该算法可以在$O(\sqrt{p})$时间复杂度内求解

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

$gcd(a,p)=1, 0 \le x < p$

算法描述

我们令$x = A\lceil\sqrt{p}\ \rceil - B$，$0 \le A, B <= \lceil\sqrt{p}\ \rceil$，这样我们可以保证 $0 \le x < p$

$a^{A\lceil\sqrt{p}\ \rceil - B} \equiv b\ mod\ p$

$\Leftrightarrow\ a^{A\lceil\sqrt{p}\ \rceil} \equiv ba^B\ mod\ p$

由于我们知道a, b，我们可以先枚举B算出右边 $ba^B\ mod\ p$，存在一个hash表里面，然后再枚举A计算左边的$a^{A\lceil\sqrt{p}\ \rceil}$的值，然后去判断hash表里面是
如果有，我们就得到了 $x = A\lceil\sqrt{p}\ \rceil - B$。

原根

$gcd(g,m)=1, g^{\varphi(m)} \equiv 1\ mod\ m$，$g$ 为 m 的一个原根。

所有原根

若g为m的一个原根，则集合$S=\{g^s|1 \le s \le \varphi(m)\ , gcd(s,m)=1\}$包含所有原根，由此如果$m$有原根，则$m$一共有$\varphi(\varphi(m))$个原根关于模$m$两两

一个原根

$gcd(g,m)=1$,$p_1,p_2,p_3,...,p_k$为$\varphi(m)$的不同素因子，当且仅当 对于任意的 $1 \le i \le k, g^{\frac{\varphi(m)}{p_i}} \not\equiv 1\ mod\ m$ 成立，g为m的一个原根

了解了原根以后，我们就可以在仅当p是素数时，就可以求解方程 $x^a \equiv b\ mod\ p$

$x^a \equiv b\ mod\ p$，由于p是一个素数，则p一定存在一个原根g，因此对于模p下的任意$x(0 \le x < p)$，存在唯一一个$i(0 \le i < p-1)$满足 $x = g^i$。

于是我们令$x = g^c$，所以$(g^c)^a \equiv b\ mod\ p \Leftrightarrow (g^a)^c \equiv b\ mod\ p$，于是这就转换成了一个BSGS模型，可以算出$c$，

$x_0 \equiv g^c\ mod\ p$

求出一个解以后，如何得到所有解

$x_0 \equiv g^c\ mod\ n, g^{\varphi(n)} \equiv 1\ mod\ n$

$\forall t \in z, x^k \equiv g^{ck} \equiv g^{ck+t\varphi(n)}\ mod\ n$

$\forall t \in z, k|t\cdot\varphi(n), x \equiv g^{c+\frac{t\cdot\varphi(n)}{k}}\ mod\ n$

既然，$k|t\cdot\varphi(n)$，那么 $\frac{k}{gcd(k,\varphi(n))}|t$，我们令 $t = i * \frac{k}{gcd(k,\varphi(n))}$

于是，$\forall\ i \in z, x \equiv g^{c+\frac{\varphi(n)}{gcd(k,\varphi(n))}*i}\ mod\ n$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll gcd(ll a, ll b)  // 求最大公约数
{
        if(b == 0) return a;
        while(b) {
                ll t = a;
                a = b;
                b = t % b;
```

```
	}
		return a;
}
ll power(ll a, ll b, ll p) // 快速幂
{
	ll ans = 1;
	while(b) {
		if(b & 1) ans = (ans * a) % p;
		a = (a * a) % p;
		b >>= 1;
	}
	return ans;
}
ll generator(ll p)  // 求模p的原根
{
	vector<ll> fact;
	ll phi = p - 1, n = phi;  // phi(p) = p - 1, p 为素数
	for(ll i = 2; i * i <= n; i++) {  // 素因子分解
		if(n % i == 0) {
			fact.push_back(i);
			while(n % i == 0) n /= i;
		}
	}
	if(n > 1) fact.push_back(n);
	for(ll res = 2; res <= p; res++) {  // 枚举每一可能的值
		bool ok = true;
		for(vector<ll>::iterator it = fact.begin(); it != fact.end(); it++) {
			if(power(res, phi / (*it), p) == 1) {  // 对于每一个素因子p_i, a^{phi(p)/p_i} != 1 mod p 才为原根
				ok = false;
				break;
			}
		}
		if(ok) return res;
	}
	return -1;
}
void BSGS(ll k, ll a, ll n)  // x^k = a mod n
{
	if(a == 0) {
		printf("1\n0\n");
		return ;
	}
	ll g = generator(n); // 原根
	ll sq = (ll)sqrt(n + .0) + 1;  // sqrt(n) 向上取整
	vector<pair<ll, int> > dec(sq);
	// 枚举 A
	for(ll i = 1; i <= sq; i++) dec[i-1] = make_pair(power(g, i * sq * k % (n - 1), n), i);
	sort(dec.begin(), dec.end());
	ll res = -1;
	// 枚举 B
	for(int i = 1; i <= sq; i++) {
		ll my = power(g, i * k % (n - 1), n) * a % n;
		vector<pair<ll, int> >::iterator it = lower_bound(dec.begin(), dec.end(), make_pair(my, 0));
		if(it != dec.end() && it->first == my) {
			res = it->second * sq - i;  // A sqrt(n) - B
			break;
		}
	}
```


展开阅读全文