BSGS

基础篇

问题：

给出a，b，p，其中$gcd(a,p)=1$，求x满足

$$a^x \equiv b(mod\,p)$$

思路：

设$x=A\sqrt{p}-B$其中$A\in(0,\sqrt{p}],B\in[0,\sqrt{p})$，得到问题的变形

$$a^{A\sqrt{p}-B} \equiv b(mod\,p)$$
$$a^{A\sqrt{p}} \equiv ba^B(mod\,p)$$

我们先枚举B，算出每个$ba^B\ mod\,p$，用$unordered_map$存起来，再枚举A，计算出$a^{A\sqrt{p}}$，在$unordered_map$中找相同的值，这样的A,B就能恰好凑成一对答案。复杂度$O(\sqrt{p})$，如果用map的话，就多一个log。

模板：

[P2485 [SDOI2011]计算器]

[[TJOI2007] 可爱的质数/【模板】BSGS]

进阶篇

问题：

$$x^a \equiv b(mod\,p)$$

解法一：

设g是p的一个原根，根据原根的性质存在c，满足$g^c\equiv x$成立，同理存在t，满足$g^t\equiv b$成立。

$$\therefore (g^c)^a \equiv b(mod\,p)$$
$$(g^a)^c \equiv b(mod\,p)$$

g^a已知，所以我们就能用基础篇直接求解c，也就是一个特解。

解法二：

设g是p的一个原根，根据原根的性质存在c，满足$g^c\equiv x$成立，同理存在t，满足$g^t\equiv b$成立。

$$\therefore g^{ac}\equiv g^t(mod\ p)$$

$$\text{根据阶的性质有}$$

$$\therefore ac\equiv t(mod\ \varphi(p))$$

$$\text{根据}exgcd\text{求出}c\text{，也是一个特解。}$$

求全部的解：

我们在已知一个特解g^c的情况下，我们要得到全部解。

$$\because g^{\varphi(p)}\equiv 1(mod\ p)$$

$$\therefore \forall t\in\mathbb{Z}, x^a\equiv g^{ca+\frac{t\varphi(p)}{a}}(mod\ p)$$

$$\therefore \forall t\in\mathbb{Z}\text{且}\ a|t\varphi(p), x\equiv g^{c+\frac{t\varphi(p)}{a}}(mod\ p)$$

$$\because a|t\varphi(p)$$

$$\therefore \frac{a}{gcd(a,\varphi(p))}|t$$

$$\therefore t=\frac{a}{gcd(a,\varphi(p))}i$$

$$\therefore \text{全部的解为：}\forall i\in\mathbb{Z}, x\equiv g^{c+\frac{\varphi(p)}{gcd(a,\varphi(p))}i}(mod\ p)$$

题目：

[P3306 [SDOI2013] 随机数生成器]

$$x_{i+1}\equiv ax_i+b(mod\ p)$$

$$x_{i+1}+\frac{b}{a-1}\equiv a(x_i+\frac{b}{a-1})(mod\ p)$$

$$x_n\equiv t(mod\ p)$$

$$a^{n-1}(x_1+\frac{b}{a-1})-\frac{b}{a-1}\equiv t(mod\ p)$$

$$a^{n-1}\equiv (t+\frac{b}{a-1})*inv(x_1+\frac{b}{a-1})(mod\ p)$$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll qpow(ll x, ll y, ll mod) {
    ll ans = 1;
    while(y) {
        if(y & 1) ans = ans * x % mod;
        x = x * x % mod;
        y >>= 1;
    }
    return ans;
```

```cpp
}

unordered_map<ll, ll> mp;
ll bsgs(ll a, ll b, ll p) {
    if(a % p == 0) return -1;
    mp.clear();
    ll k = ceil(sqrt(p));
    for(int i=0; i<=k; i++) {
        mp[b] = i;
        b = b * a % p;
    }
    ll aa = qpow(a, k, p), A = aa;
    for(int i=1; i<=k; i++) {
        if(mp[aa]) {
            return 1ll * i * k - mp[aa] + 1;
        }
        aa = aa * A % p;
    }
    return -1;
}
int main() {
#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
#endif
    int t;
    cin >> t;
    while(t--) {
        ll p, a, b, x, t;
        cin >> p >> a >> b >> x >> t;
        if(x == t) {
            cout << 1 << endl;
            continue;
        }
        if(a == 0) {
            if(b == t) cout << 2 << endl;
            else cout << -1 << endl;
            continue;
        }
        if(a == 1) {
            if(b == 0) cout << -1  << endl;
            else {
                ll k = qpow(b, p-2, p);
                cout << ((t-x+p)%p*k)  % p + 1 << endl;
            }
            continue;
        }

        ll tmp = b*qpow(a-1, p-2, p) % p;
        t = (t + tmp) % p;
        t = t * qpow((x+tmp)%p, p-2, p) % p;
        ll ans = bsgs(a, t, p);
        cout << ans << endl;
    }
    return 0;
}
```

261. Discrete Roots模板

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll qpow(ll x, ll y, ll mod) {
    ll ans = 1;
    while(y) {
        if(y & 1) ans = ans * x % mod;
        x = x * x % mod;
        y >>= 1;
    }
    return ans;
}
ll G(ll p) {
    if(p == 2) return 1;
    vector<ll> tmp;
    int phi = p-1, n = phi;
    for(int i=2; 1ll*i*i<=n; i++) {
        if(n % i == 0) {
            tmp.push_back(i);
            while(n % i == 0) n /= i;
        }
    }
    if(n > 1) tmp.push_back(n);
    for(int i=1; i<=p; i++) {
        bool flag = 0;
        for(auto it : tmp) {
            if(qpow(i, phi/it, p) == 1) {
                flag = 1;
                break;
            }
        }
        if(!flag) return i;
    }
    return -1;
}
unordered_map<ll, ll> mp;
ll bsgs(ll a, ll b, ll p) {
    if(a % p == 0 && b != 0) return -1;
    mp.clear();
    int k = ceil(sqrt(p));
    for(int i=0; i<=k; i++) {
        mp[b] = i;
        b = b * a % p;
    }
    ll aa = qpow(a, k, p), A = aa;
    for(int i=1; i<=k; i++) {
        if(mp[A]) return 1ll * i * k - mp[A];
        A = A * aa % p;
    }
    return -1;
```

```
}

int main() {
#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
#endif
    ll p, k, a;
    while(cin >> p >> k >> a) {
        if(a == 0) {
            cout << "1\n0\n";
            continue;
        }
        ll g = G(p), gk = qpow(g, k, p);
        ll x0 = bsgs(gk, a, p), t = bsgs(g, a, p);
        if(x0 == -1) {
            cout << "0\n";
            continue;
        }

        vector<ll> ans;
        ll d = __gcd(p-1, k), mod = (p-1)/d;
        x0 = x0 % mod;
        for(int i=0; i<d; i++) {//最多只有d个不同的解。
            ans.push_back(qpow(g, ( x0+i*mod%(p-1) )%(p-1), p));
        }
        sort(ans.begin(), ans.end());
        cout << ans.size() << endl;
        for(auto it : ans) cout << it << ' ';
        cout << endl;
    }
    return 0;
}
```

F. Lunar New Year and a Recursive Sequence

题意:

给出$f_1 = f_2 = \cdots = f_{k-1} = 1$和$b_1, b_2 \cdots b_k$，还有递推方程

$$f_i = f_{i-1}^{b_1} f_{i-2}^{b_2} \cdots f_{i-k}^{b_k}$$

问是否存在一个f_k使得$f_n \equiv m(mod\ p)$成立;

思路:

矩阵快速幂$+BSGS$

因为n很大，所以不能直接推出f_n是f_k的多少次方，观察递推式全是乘法，我们就想到用矩阵加速。

先推出矩阵。

假设$k = 4$，则可以推出下面的式子。

$$
\begin{aligned}
f_k &= f_k \\
f_{k+1} &= f_k^{b_1} f_{k-1}^{b_2} f_{k-2}^{b_3} f_{k-3}^{b_4} = f_k^{b_1} \\
f_{k+2} &= f_{k+1}^{b_1} f_k^{b_2} f_{k-1}^{b_3} f_{k-2}^{b_4} = (f_k^{b_1})^{b_1} f_k^{b_2} = f_k^{b_1^2+b_2} \\
f_{k+3} &= f_{k+2}^{b_1} f_{k+1}^{b_2} f_k^{b_3} f_{k-1}^{b_4} = f_k^{b_1^3+2b_1*b_2+b_3} \\
f_{k+4} &= f_{k+3}^{b_1} f_{k+2}^{b_2} f_{k+1}^{b_3} f_k^{b_4} = f_k^{b_1^4+3b_1^2b_2+2b_1b_3+b_2^2+b_4} \\
f_{k+5} &= f_{k+4}^{b_1} f_{k+3}^{b_2} f_{k+2}^{b_3} f_{k+1}^{b_1}
\end{aligned}
$$

我们可以看出每个f_i都是由前k个f值推出来的，所以我们设f_i的对应的f_k的系数是g_i

我们就能找到一个一维的矩阵乘法

$$
[\,g_4\,,g_3\,,g_2\,,g_1\,] \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = g_5
$$

$$
[\,g_5\,,g_4\,,g_3\,,g_2\,] \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = g_6
$$

所以我们把矩阵扩展到k维，希望得到一个包含g_i的矩阵乘以另一个矩阵得到包含形式相同且包含g_{i+1}的矩阵，

推出：

$$
\begin{bmatrix} g_7 & g_6 & g_5 & g_4 \\ g_6 & g_5 & g_4 & g_3 \\ g_5 & g_4 & g_3 & g_2 \\ g_4 & g_3 & g_3 & g_1 \end{bmatrix} \times \begin{bmatrix} b_1 & 1 & 0 & 0 \\ b_2 & 0 & 1 & 0 \\ b_3 & 0 & 0 & 1 \\ b_4 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} g_8 & g_7 & g_6 & g_5 \\ g_7 & g_6 & g_5 & g_4 \\ g_6 & g_5 & g_4 & g_3 \\ g_5 & g_4 & g_3 & g_2 \end{bmatrix}
$$

通过$k=4$的特例，我们能够推出k是范围内的任意矩阵。

我们就能推出f_n是f_k的多少次方，设$f_n = f_k^a$。

现在的问题是已知a，求是否存在f_k使$f_k^a \equiv m(mod\ p)$成立。

这个就是之前的$bsgs$的进阶篇：[进阶篇]

$Code:$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e2+10;
const int mod = 998244353;
vector<ll> a;
ll b[N], k;
struct matrix {
    ll s[N][N];
    matrix operator * (const matrix &t) {
        matrix tmp;
        for(int i=1; i<=k; i++) {
            for(int j=1; j<=k; j++) {
                tmp.s[i][j] = 0;
                for(int l=1; l<=k; l++) {
```

```
                    tmp.s[i][j] = (tmp.s[i][j] + s[i][l] * t.s[l][j] % (mod-1)) % (mod-1);
                }
            }
        }
        return tmp;
    }
} A, I, B;

matrix mqpow(matrix t, ll y) {
    matrix ans = I;
    while(y) {
        if(y & 1) ans = ans * t;
        t = t * t;
        y >>= 1;
    }
    return ans;
}

ll qpow(ll x, ll y) {
    ll ans = 1;
    while(y) {
        if(y & 1) ans = ans * x % mod;
        x = x * x % mod;
        y >>= 1;
    }
    return  ans;
}

ll G(ll p) {
    vector<int> v;
    ll phi = p-1, tmp = phi;
    for(int i=2; 1ll*i*i<=tmp; i++) {
        if(tmp % i == 0) {
            v.push_back(i);
            while(tmp % i == 0) tmp /= i;
        }
    }
    if(tmp > 1) v.push_back(tmp);
    for(int i=2; i<=phi; i++) {
        int f = 1;
        for(auto it : v) {
            if(i % it == 0) {
                f = 0;
                break;
            }
            if(f) return i;
        }
    }
    return -1;
}
unordered_map<ll, ll> mp;
ll bsgs(ll aaa, ll b, ll p) {
    mp.clear();
    if(aaa % p == 0 && b % p != 0) return -1;
    int k = ceil(sqrt(p));
    for(int i=0; i<=k; i++) {
        mp[b] = i, b = b * aaa % mod;
```

```
    }
    ll aa = qpow(aaa, k), A = aa;
    // cout << aa<< endl;
    for(int i=1; i<=k; i++) {
        if(mp[A]) return 1ll*i*k - mp[A];
        A = A * aa % mod;
    }
    return -1;
}
int main() {
#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
#endif
    scanf("%lld", &k);
    for(int i=1; i<=k; i++) I.s[i][i] = 1;
    for(int i=1; i<=k; i++) {
        scanf("%lld", &b[i]);
    }
    for(int i=1; i<k; i++) a.push_back(0);
    a.push_back(1);
    for(int i=1; i<=k; i++) {
        ll tmp = 0;
        for(int j=1; j<=k; j++) {
            tmp = (tmp + b[j] * a[a.size()-j] % (mod-1)) % (mod-1);
        }
        a.push_back(tmp);
    }
    ll n, m, aa;
    scanf("%lld%lld", &n, &m);
    if(n <= 2*k) {
        aa = a[n-1] % (mod-1);
    }
    else {
        for(int j=1, i=a.size()-1; j<=k; j++, i--) {
            int l = 1;
            while(l <= k) {
                A.s[j][l] = a[i-l+1];
                // cout << i-l+1 << endl;
                l++;
            }
        }
        for(int i=1; i<=k; i++) {
            for(int j=1; j<=k; j++) {
                if(i == 1) B.s[j][i] = b[j];
                else if(j == i-1) {
                    B.s[j][i] = 1;
                }
            }
        }
        A = A * mqpow(B, n-2*k);
        aa = A.s[1][1]%(mod-1);
    }
    ll g = G(mod);
    ll ans = bsgs(qpow(g, aa), m, mod);
    if(ans != -1) cout << qpow(g, ans);
    else cout << -1 << endl;
    return 0;
```

```
}
```

扩展篇

问题:

当p，a可能不互质时，求

$$a^x \equiv b(mod\, p)$$

方法:

当$d_1 = gcd(a, p) \neq 1$，当$d_1 \nmid b$时，无解，则原式变成:

$$\frac{a}{d_1} a^{x-1} \equiv \frac{b}{d_1} (mod\ \frac{p}{d_1})$$

当设$d_2 = gcd(a, \frac{p}{d_1}) \neq 1$，当$d_2 \nmid \frac{b}{d_1}$时，无解，则原式可变成:

$$\frac{a^2}{d_1 d_2} a^{x-2} \equiv \frac{b}{d_1 d_2} (mod\ \frac{p}{d_1 d_2})$$

重复直到$d_{cnt} = 1$。设$D = \prod_{i=1}^{cnt} d_i$，原式可以写成:

$$\frac{a^{cnt}}{D} a^{x-cnt} \equiv \frac{b}{D} (mod\ \frac{p}{D})$$

剩下的$a^{x-cnt}, \frac{b}{D}, \frac{p}{D}$，可以构成基础篇的$bsgs$，修改的地方就是乘了一个系数$\frac{a^{cnt}}{D}$。

注意：cnt可能大于x这个解，所以我们在做除法的时候就要判断是不是能够相等，相等直接输出当前的cnt。

题目:

P4195 【模板】扩展BSGS

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll qpow(ll x, ll y, ll mod) {
    ll ans = 1;
    while(y) {
        if(y & 1) ans = ans * x % mod;
        x = x * x % mod;
        y >>= 1;
    }
    return ans;
}
unordered_map<ll, ll> mp;
```

```
ll bsgs(ll a, ll b, ll p, ll ad) {
    if(a % p == 0 && b%p != 0) return -1;
    mp.clear();
    int k = ceil(sqrt(p));
    for(int i=0; i<k; i++) {
        mp[b] = i;
        b = b * a % p;
    }
    ll aa = qpow(a, k, p), A = aa*ad % p;
    for(int i=1; i<=k; i++) {
        if(mp[A]) return 1ll*i*k - mp[A];
        A = A * aa % p;
    }
    return -1;
}
ll exgcd(ll &x, ll &y, ll a, ll b) {
    if(b == 0) {
        x = 1, y = 0;
        return a;
    }
    ll d = exgcd(x, y, b, a%b);
    ll tmp = y;
    y = x - a/b * y;
    x = tmp;
    return d;
}
ll inv(ll a, ll b) {
    ll x, y;
    ll d = exgcd(x, y, a, b);
    return x;
}

ll exbsgs(ll a, ll b, ll p) {
    a %= p, b %= p;
    if(b == 1 || p == 1) return 0;
    ll cnt = 0, d, tmp = 1;
    while((d = __gcd(a, p)) ^ 1) {
        if(b % d) return -1;
        cnt++, b /= d, p /= d;
        tmp = tmp * a/d % p;
        if(tmp == b) return cnt;
    }
    ll ans = bsgs(a, b, p, tmp);
    if(ans == -1) return -1;
    else return ans + cnt;
}
int main() {
#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
#endif
    ll a, p, b, d;
    while(cin >> a >> p >> b) {
        if(!a || !p || !b) break;
        ll ans = exbsgs(a, b, p);
        if(ans != -1) cout << ans << endl;
        else cout << "No Solution\n";
    }
```

```
    return 0;
}
```

阶和原根

阶

定义：

对于$m > 1$且$(a, m) = 1$，使$a^n \equiv 1(mod\ m)$成立的最小的n，称为n模m的阶，记作$\delta_m(a)$。

根据欧拉定理，可以证明至少存在一个n使$a^n \equiv 1(mod\ p)$成立。

性质：

1. $a, a^2, \ldots, a^{\delta_m(a)}$在模$m$下互不同余。

证明：

假设存在i，$j \in [1, \delta_m(a)]$)，使$a^i \equiv a^j(mod\ m)$，则可以变形成$a^{|i-j|} \equiv 1(mod\ m)$，易知$|i-j| < \delta_m(a)$，与阶的定义相矛盾，所以假设不成立，故原命题成立。

2. 若$a^n \equiv 1(mod\ m)$，则$\delta_m(a)|n$。

证明：

假设$\delta_m(a) \nmid n$，则$n = k\delta_m(a) + r$（$r > 0$），$a^{k\delta_m(a)}a^r \equiv a^r \equiv 1(mod\ m)$，根据性质1，可知假设不成立，原命题成立。

可以推出：$a^p \equiv a^q(mod\ m)$，则$p \equiv q(mod\ \delta_m(a))$。

3. 设$m > 1$，a，$b \in \mathbb{Z}$，$(a, m) = (b, m) = 1$，则$\delta_m(ab) = \delta_m(a)\delta_m(b)$的充要条件是：$(\delta_m(a), \delta_m(b)) = 1$

证明：

必要性：

$$\because a^{\delta_m(a)} \equiv 1(mod\, p)和 b^{\delta_m(b)} \equiv 1(mod\, m)$$
$$\therefore (ab)^{[\delta_m(a),\delta_m(b)]} \equiv 1(mod\, m)$$
$$\because 性质2$$
$$\therefore \delta_m(ab) \mid [\delta_m(a), \delta_m(b)]$$
$$\because \delta_m(a)\delta_m(b)| = \delta_m(ab)$$
$$\therefore \delta_m(a)\delta_m(b)|[\delta_m(a), \delta_m(b)]$$
$$\therefore (\delta_m(a), \delta_m(b)) = 1$$

充分性:

$$\because (ab)^{\delta_m(ab)} \equiv 1(mod\, m)$$
$$\therefore (ab)^{\delta_m(ab)\delta_m(b)} \equiv a^{\delta_m(ab)} b^{\delta_m(b)} \equiv a^{\delta_m(ab)} \equiv 1(mod\, m)$$
$$\therefore \delta_m(a) \mid \delta_m(ab)$$
$$同理: \delta_m(b) \mid \delta_m(ab)$$
$$\therefore \delta_m(a)\delta_m(b) \mid \delta_m(ab)$$
$$\because a^{\delta_m(a)} b^{\delta_m(b)} \equiv (ab)^{\delta_m(a)\delta_m(b)}$$
$$\therefore \delta_m(ab) \mid \delta_m(a)\delta_m(b)$$
$$综上: \delta_m(ab) = \delta_m(a)\delta_m(b)$$

4. 设$k \in \mathbb{N}, m > 1, a \in \mathbb{Z}, (a, m) = 1$，则: $\delta_m(a^k) = \frac{\delta_m(a)}{(\delta_m(a),k)}$.

证明:

$$\because (a^k)^{\delta_m(a^k)} \equiv a^{k\delta_m(a^k)} \equiv 1(mod\, m)$$
$$\therefore \delta_m(a) \mid k\delta_m(a^k)$$
$$\therefore \frac{\delta_m(a)}{(\delta_m(a), k)} \mid \delta_m(a^k)$$
$$\because (a^k)^{\frac{\delta_m(a)}{(\delta_m(a),k)}} \equiv (a^{\delta_m(a)})^{\frac{k}{(\delta_m(a),k)}} \equiv 1(mod\, m)$$
$$\therefore \delta_m(a^k) \mid \frac{\delta_m(a)}{(\delta_m(a), k)}$$
$$综上: \delta_m(a^k) = \frac{\delta_m(a)}{(\delta_m(a), k)}$$

原根

定义:

设$m \in \mathbb{N}^*$,$a \in \mathbb{Z}$.若$(a, m) = 1$,且$\delta_m(a) = \phi(m)$,则称a为模m的原根（注：a，m互质）.

原根判定定理

若g是模m的一个原根,则对于$\phi(m)$的任何大于1且不为自身的因数p,都有$g^{\frac{\phi(m)}{p}} \equiv 1(mod\, m)$.

证明:

假设存在一个$t < \mu(m)$使得$a^t \equiv 1(mod\, m)$且$\forall i \in [1, k] : a^{\frac{\phi(m)}{d_i}} \equiv 1(mod\, m)$.

由裴蜀定理得，一定存在一组k，x满足$kt = x\phi(m) + (t, \mu(m))$; 由欧拉定理/费马小定理得 $a^{\phi(p)} \equiv 1(mod\, p)$;

$\therefore 1 \equiv a^{kt} \equiv a^{x\phi(m)} a^{(t,\phi(m))} \equiv a^{(t,\phi(m))} (mod\ m)$

$\because t < \phi(m) \ \therefore (t, \phi(m)) \le t < \phi(m)$

$\because (t, \phi(m)) | \phi(m), \ \therefore (t, \phi(m))$一定至少整除$\frac{\phi(m)}{d_i}$中的至少一个。

设$(t, \phi(m)) | \frac{\phi(m)}{d_i}$,则$a^{\frac{\phi(m)}{d_i}} \equiv a^{(t,\phi(m))} \equiv 1 (mod\ m)$。

$\therefore$假设不成立，原命题成立

原根的个数

如果一个数m有原根g，则它的原根个数为$\phi(\phi(m))$。

证明：

如果m存在原根g，则

$$\delta_m(g^k) = \frac{\delta_m(g)}{(\delta_m(g), k)} (\text{阶的性质})$$
$$if\ (k, \delta_m(g)) = 1 \text{且} 1 < k < \phi(m) \text{的} k \text{有} \phi(\phi(m)) \text{个。}$$
$$\therefore \text{原根有} \phi(\phi(m)) \text{个。}$$

原根一定是$\phi(\phi(m))$个，在模m的情况下。

原根的存在定理

一个数m存在原根当且仅当$m = 2, \ 4, \ p^\alpha, \ 2p^\alpha$，其中$p$为奇素数。$\alpha \in \mathbb{N}^*$。

O(1)快速乘

```
inline ll multi(ll x, ll y, ll mod) {
    ll tmp=(x*y-(l)((long double)x/mod*y+1.0e-8)*mod);
    return tmp<0 ? tmp+mod : tmp;
}
```

python进制转化

```python
def C(a, b, c):
    s = ''
    k, I = 0, 1
    for i in reversed(c):
        if '0' <= i <= '9':
            k += (ord(i) - ord('0')) * I
        elif 'A' <= i <= 'Z':
            k += (ord(i) - ord('A') + 10) * I
        else:
            k += (ord(i) - ord('a') + 36) * I
        I = I * a
    if k == 0:
        return '0'
    while k:
        m = k % b
        if 0 <= m <= 9:
            s = s + str(chr(ord('0') + m))
        if 10 <= m <= 35:
            s = s + str(chr(ord('A') + m-10))
        if 36 <= m <= 61:
            s = s + str(chr(ord('a') + m-36))
        k //= b
        # reversed(s)
    return s


x = input()
y = []
tmp = ''
for i in x:
    if i == ' ':
        y.append(tmp)
        tmp = ''
    else:
        tmp += i
y.append(tmp)
a = eval(y[0])
b = eval(y[1])
cc = C(a, b, y[2])
ans = ''
for i in reversed(cc):
    ans += i
    print(i, end='')
# print()
# for i in reversed(C(b, a, ans)):
#     print(i, end='')
```