N次剩余

Discrete Root

模数是质数时

设 p 是一个质数，求解：

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

求出一个特解

由于 p 是质数，必存在原根 g，则 $\{g^1, g^2, \cdots, g^{p-1}\}$ 构成了一个模 p 的完全剩余系，所以一定 $\exists c$ 使得 $x \equiv g^c \pmod p$，问题转化为求 c.

代入方程得到：

$$(g^a)^c \equiv b \pmod p$$

使用 **BSGS** 算法求解 c，得到特解：$x_0 \equiv g^c \pmod p$.

得到通解

由于 $g^{p-1} \equiv 1 \pmod p$，故

$$\forall t \in \mathbb{Z},\quad x^a \equiv g^{ac} \equiv g^{ac+t(p-1)} \equiv b \pmod p$$

又由于 g 是原根，所以可以肯定通解为：

$$\forall t \in \mathbb{Z},\ a \mid t(p-1),\quad x \equiv g^{c+\frac{t(p-1)}{a}} \pmod p$$

由于 $a \mid t(p-1) \implies \frac{a}{\gcd(a,p-1)} \mid t$，不妨设 $t = \frac{a}{\gcd(a,p-1)}i$，那么通解写作：

$$\forall i \in \mathbb{Z},\quad x \equiv g^{c+\frac{i\cdot(p-1)}{\gcd(a,p-1)}} \pmod p$$

实现时，遍历 $0 \leqslant i < \gcd(a, p-1)$ 即可，因为要取得不同的 x，要求取 i 使得 $\frac{i\cdot(p-1)}{\gcd(a,p-1)}$ 模 $p-1$ 不同，显然当 $i = \gcd(a, p-1)$ 时又回到了 $i = 0$ 的情况。

Code

```cpp
#include<bits/stdc++.h>

using namespace std;

typedef long long LL;

int gcd(int a, int b){ return b == 0 ? a : gcd(b, a % b); }

inline int fpow(int bs, int idx, int m){
    int res = 1;
    while(idx){
        if(idx & 1) res = 1ll * res * bs % m;
        bs = 1ll * bs * bs % m;
        idx >>= 1;
    }
    return res;
}

int getSPR(int p){
    // get the smallest primitive root of PRIME p
    vector<int> factors; // PRIME factors of phi(p)=p-1
    int phip = p - 1;
    for(int i = 2; i * i <= phip; i++){
```

```
        if(phip % i)    continue;
        factors.emplace_back(i);
        while(phip % i == 0)    phip /= i;
    } if(phip > 1)  factors.emplace_back(phip);

    int g = 0; // smallest primitive root
    for(g = 2; g <= p; g++){
        bool ok = true;
        for(auto &factor : factors){
            if(fpow(g, (p - 1) / factor, p) == 1){
                ok = false; break;
            }
        }
        if(ok)  break;
    }
    return g;
}

int BSGS(int a, int b, int m){
    // solve a^x = b (mod m)
    unordered_map<int, int> val;
    int sq = sqrt(m) + 1;
    LL an = 1;
    for(int i = 1; i <= sq; i++)    an = an * a % m;
    for(LL q = 0, cur = b; q <= sq; cur = cur * a % m, q++)
        val[cur] = q;
    for(LL p = 1, cur = an; p <= sq; cur = cur * an % m, p++)
        if(val.count(cur))
            return sq * p - val[cur];
    return -1;
}

vector<int> DiscreteRoot(int a, int b, int p){
    // solve x^a = b (mod p)
    vector<int> res;
    if(b == 0){ res.emplace_back(0); return res; }
    int g = getSPR(p);
    int c = BSGS(fpow(g, a, p), b, p);
    if(c == -1) return res;
    int d = gcd(a, p-1);
    int delta = (p - 1) / d;
    for(int i = 0; i < d; i++){
        int cur = (c + 1ll * i * delta % (p-1)) % (p-1);
        res.emplace_back(fpow(g, cur, p));
    }
    sort(res.begin(), res.end());
    return res;
}

int main(){
    int p, a, b; scanf("%d%d%d", &p, &a, &b);
    vector<int> ans = DiscreteRoot(a, b, p);
    printf("%d\n", (int)ans.size());
    for(auto &k : ans)  printf("%d ", k);
    return 0;
}
```