题面

数论描述

给定 n 个正整数 $m_1, m_2, \cdots, m_n$, 判断每个 m 是不是模 p 意义下的 k 次剩余。多个 m 符合条件按从小到大顺序输出.

通俗描述

给定 n 个正整数 $m_1, m_2, \cdots, m_n$, 判断每个 m 是否满足:

1. $\gcd(m, p) = 1$.
2. 存在一个整数 a, 使得 $a^k \equiv m \pmod p$.

如果一个数 m 满足上述两个条件, 则称 m 是模 p 意义下的 k 次剩余。多个 m 符合条件按从小到大顺序输出.

$x \equiv y \pmod p$ 的定义是 $p \mid (x-y)$, 等价描述是存在整数 b 使得 $x - bp = y$.

题解

显然 $i^k \equiv (i\%p)^k \pmod p$. 所有可能是 k 次剩余的数至多有 $p-1$ 个, 他们在 $1^k, 2^k, \cdots, (p-1)^k \pmod p$ 当中. 将这些数字枚举出来并放入一个 std::set 或 bool 数组进行存储. 对于给定的一个 m, 先判断 $\gcd(m, p)$ 是否为 1. 如果是, 再判断上述 $p-1$ 个数字中有没有 m. 如果两个条件都符合, 那么 m 就是模 p 意义下的 k 次剩余, 否则就不是.

使用快速幂算法可以在 $O(\log k)$ 的时间里算出一个 i^k, 算出所有可能的 $p-1$ 个数的时间复杂度是 $O(p \log k)$. 求 gcd 使用辗转相除法可以在 $O(\log p)$ 的时间内求得 $\gcd(m, p)$. 使用 std::set 进行查询一次的时间复杂度是 $O(\log p)$, 使用 bool 数组进行查询一次的时间复杂度是 $O(1)$. 于是总时间复杂度就是 $O(p \log k + n \log p)$.

代码

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

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

inline int kpow(ll a, int k, int p)
{
    ll ans = 1;
    while (k)
    {
        if (k & 1)
            ans *= a, ans %= p;
        a *= a, a %= p;
        k >>= 1;
```

```cpp
    }
    return ans;
}

int main()
{
    int k, p, n;
    vector<int> que;
    set<int> pop, ans;
    cin >> k >> p >> n;
    for (int i = 0; i < n; ++i)
    {
        int a;
        cin >> a;
        que.push_back(a);
    }
    for (int i = 1; i < p; ++i)
        pop.insert(kpow(i, k, p));
    for (int i = 0; i < n; ++i)
        if (gcd(que[i], p) == 1 && pop.count(que[i]))
            ans.insert(que[i]);
    cout << ans.size() << endl;
    for (int a : ans)
        cout << a << " ";
    return 0;
}
```