原根

Primitive Root

阶 Order

定义

若 $(a,m)=1$，称使得 $a^l \equiv 1 \pmod m$ 成立的**最小的** l 为 a 模 m 的阶，记作：$\mathrm{ord}_m a$.

换句话说，定义 a 模 m 的阶是 $a^x \equiv 1 \pmod m$ 的**最小正整数解**。

理解：根据欧拉定理，$a^{\varphi(m)} \equiv 1 \pmod m$，所以 $\{a^1, a^2, \cdots\}$ 这个数列在模 m 下有一个长度为 $\varphi(m)$ 的循环节，但是这不一定是最小循环节，最小循环节长度是 $\mathrm{ord}_m a$.

性质

性质1：

$$\mathrm{ord}_m a \mid \varphi(m)$$

由上述理解易知。

性质2：

$$a^n \equiv 1 \pmod m \implies \mathrm{ord}_m a \mid n$$

$a^n \equiv 1 \pmod m$ 说明 $\{a^1, a^2, \cdots, a^n\}$ 是一个循环节，长度自然是最小循环节的整数倍。

性质3：设 $m \in \mathbb{N}^*$，$a, b \in \mathbb{Z}$，$\gcd(a,m)=\gcd(b,m)=1$，则：

$$\mathrm{ord}_m(ab)=\mathrm{ord}_m a \cdot \mathrm{ord}_m b \iff \gcd(\mathrm{ord}_m a, \mathrm{ord}_m b)=1$$

证：必要性：由 $a^{\mathrm{ord}_m a} \equiv 1 \pmod m$ 和 $b^{\mathrm{ord}_m b} \equiv 1 \pmod m$ 可知：

$$(ab)^{\mathrm{lcm}(\mathrm{ord}_m a, \mathrm{ord}_m b)} \equiv 1 \pmod m$$

根据前述性质可得：

$$\mathrm{ord}_m(ab) \mid \mathrm{lcm}(\mathrm{ord}_m a, \mathrm{ord}_m b)$$

又有条件：$\mathrm{ord}_m(ab)=\mathrm{ord}_m a \cdot \mathrm{ord}_m b$，于是：

$$\mathrm{ord}_m a \cdot \mathrm{ord}_m b \mid \mathrm{lcm}(\mathrm{ord}_m a, \mathrm{ord}_m b)$$

即得：$\gcd(\mathrm{ord}_m a, \mathrm{ord}_m b)=1$.

充分性：由 $(ab)^{\mathrm{ord}_m(ab)} \equiv 1 \pmod m$ 可知：

$$(ab)^{\mathrm{ord}_m(ab) \cdot \mathrm{ord}_m b} \equiv a^{\mathrm{ord}_m(ab) \cdot \mathrm{ord}_m b} \equiv 1 \pmod m$$

根据前述性质可得：$\mathrm{ord}_m a \mid \mathrm{ord}_m(ab) \cdot \mathrm{ord}_m b$，又有条件：$\gcd(\mathrm{ord}_m a, \mathrm{ord}_m b)=1$，所以：

$$\mathrm{ord}_m a \mid \mathrm{ord}_m(ab)$$

同理，

$$\mathrm{ord}_m b \mid \mathrm{ord}_m(ab)$$

因为 $\gcd(\mathrm{ord}_m a, \mathrm{ord}_m b) = 1$，所以：

$$\mathrm{ord}_m a \cdot \mathrm{ord}_m b \mid \mathrm{ord}_m(ab)$$

另一方面，

$$(ab)^{\mathrm{ord}_m a \cdot \mathrm{ord}_m b} \equiv 1 \pmod m$$

根据前述性质可得：

$$\mathrm{ord}_m(ab) \mid \mathrm{ord}_m a \cdot \mathrm{ord}_m b$$

综上，

$$\mathrm{ord}_m(ab) = \mathrm{ord}_m a \cdot \mathrm{ord}_m b$$

证毕。

性质4： 设 $k \in \mathbb{N}$，$m \in \mathbb{N}^*$，$a \in \mathbb{Z}$，$\gcd(a, m) = 1$，则：

$$\mathrm{ord}_m\left(a^k\right) = \frac{\mathrm{ord}_m a}{\gcd(\mathrm{ord}_m a, k)}$$

证：由于

$$\left(a^k\right)^{\mathrm{ord}_m\left(a^k\right)} \equiv a^{k \cdot \mathrm{ord}_m\left(a^k\right)} \equiv 1 \pmod m$$

根据前述性质可得：

$$\mathrm{ord}_m a \mid k \cdot \mathrm{ord}_m\left(a^k\right)$$

于是：

$$\frac{\mathrm{ord}_m a}{\gcd(\mathrm{ord}_m a, k)} \mid \mathrm{ord}_m\left(a^k\right)$$

另一方面，

$$\left(a^k\right)^{\frac{\mathrm{ord}_m a}{\gcd(\mathrm{ord}_m a, k)}} \equiv \left(a^{\mathrm{ord}_m a}\right)^{\frac{k}{\gcd(\mathrm{ord}_m a, k)}} \equiv 1 \pmod m$$

根据前述性质可得：

$$\mathrm{ord}_m\left(a^k\right) \mid \frac{\mathrm{ord}_m a}{\gcd(\mathrm{ord}_m a, k)}$$

综上，

$$\mathrm{ord}_m\left(a^k\right) = \frac{\mathrm{ord}_m a}{\gcd(\mathrm{ord}_m a, k)}$$

证毕。

原根 Primitive Root

设 $(g, m) = 1$，若 $\mathrm{ord}_m g = \varphi(m)$，则称 g 为 m 的一个原根。

换句话说，g 是 m 的一个原根当且仅当 $g^1, g^2, \cdots, g^{\varphi(m)}$ 互不相同，也即 $\{g^1, g^2, \cdots, g^{\varphi(m)}\}$ 是 m 的一个缩剩余系。

存在性

不是每个数都有原根，我们有定理：

若 m 有原根，则 m 必为以下几种形式之一：$1, 2, 4, p^\alpha, 2p^\alpha$，其中 p 是奇质数。

实现：一个数是否有原根可以在线性筛的同时运用 $\textbf{Eratosthenes}$ 筛的思想预处理。

数量

设 a 是 m 的一个原根，那么对于任意 $\leqslant \varphi(m)$ 且和 $\varphi(m)$ 互质的正整数 s，a^s 也是 m 的原根。**它们是 m 的所有原根。**

证明：由于 $\{s, 2s, \cdots, \varphi(m)s\}$ 在模 m 下互不相同，所以 $\{a^s, a^{2s}, \cdots, a^{\varphi(m)s}\}$ 在模 m 下互不相同，故 a^s 是 m 的一个原根。"它们是所有原根"待证。

m 的原根的数量就是 s 的数量，即 $\varphi(\varphi(m))$.

求出最小原根

已经证明了，若数 m 存在原根，则最小原根是 $O(m^{0.25})$ 级别的。这意味着，我们可以直接暴力枚举，判断枚举的数是否是原根。

如何判断呢？设枚举的数为 g，根据阶与原根的性质（循环节性质），如果存在一个 $j \mid \varphi(m), j < \varphi(m)$ 使得 $g^j \equiv 1 \pmod m$，则 g 不是原根。于是我们有了一个 $O(d(\varphi(m)))$ 的判断方法，其中 d 是约数个数函数，大概是 $d(n) = O(n^{1.066/\ln\ln n})$。

这个过程还可以优化，设 $\varphi(m) = \prod {p_i}^{a_i}$，那么如果存在一个 $\varphi(m)/p_i$ 使得 $g^{\varphi(m)/p_i} \equiv 1 \pmod m$，则 g 不是原根。这是因为，如果 g 不是原根，那么一定存在一个 $d \mid \varphi(m), d < \varphi(m)$ 使得 $g^d \equiv 1 \pmod m$。又这个 d 显然是某一个 $\varphi(m)/p_i$ 的因子，设 $\varphi(m)/p_i = d \cdot k$，于是有：

$$g^{\frac{\varphi(m)}{p_i}} \equiv g^{d \cdot k} \equiv 1^k \equiv 1 \pmod m$$

所以判断 $\varphi(m)/p_i$ 就够了。这样我们判断的复杂度是 $O(\omega(\varphi(m)))$，其中 ω 是不计算重数的素因子个数函数，假装是 $\omega(n) = O(\lg n)$ 好了。

综上，我们可以在 $O(m^{0.25} \lg \varphi(m) \lg m + \sqrt{\varphi(m)})$ 内找到 m 的最小原根。

求出所有原根

找到最小原根 g 后，找到所有 $\leqslant \varphi(m)$ 与 $\varphi(m)$ 互质的正整数 s，g^s 也是一个原根。

Code

```cpp
int phi[N], pList[N], pID;
bool notP[N];
bool existRoot[N];
void Euler(int n){
    notP[0] = notP[1] = 1, phi[1] = 1;
    existRoot[1] = existRoot[2] = existRoot[4] = true;
    for(int i = 1; i <= n; i++){
        if(notP[i] == 0){
            pList[++pID] = i, phi[i] = i - 1;
            if(i != 2){
                for(long long j = i; j <= n; j *= i){
                    existRoot[j] = true;
                    if(j * 2 <= n)  existRoot[j*2] = true;
                }
            }
        }
```

```
        for(int j = 1; j <= pID; j++){
            if(1ll * i * pList[j] > n)  break;
            notP[i * pList[j]] = 1;
            if(i % pList[j] == 0){
                phi[i * pList[j]] = phi[i] * pList[j];
                break;
            }
            else    phi[i * pList[j]] = phi[i] * (pList[j] - 1);
        }
    }
}

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;
}

vector<int> getPrimitiveRoot(int m){
    vector<int> res;
    if(!existRoot[m])   return res;

    vector<int> factors; // get PRIME factors of phi(m)
    int phim = phi[m];
    for(int i = 2; i * i <= phim; i++){
        if(phim % i)    continue;
        factors.emplace_back(i);
        while(phim % i == 0)    phim /= i;
    } if(phim > 1)  factors.emplace_back(phim);

    int g = 0; // smallest primitive root
    for(g = 2; g <= m; g++){
        if(gcd(g, m) != 1)  continue;
        bool ok = true;
        for(auto &factor : factors){
            if(fpow(g, phi[m] / factor, m) == 1){
                ok = false; break;
            }
        }
        if(ok)  break;
    }
    for(int s = 1, cur = 1; s <= phi[m]; s++){
        cur = 1ll * cur * g % m;
        if(gcd(s, phi[m]) == 1)
            res.emplace_back(cur);
    }
    sort(res.begin(), res.end());
    return res;
}
```