XCPC Templates

Khoray

March 24, 2022

Contents

1 数学

1.1 欧拉筛/积性函数筛/线性筛

- 积性函数筛，$f(pq) = f(p)f(q), (p,q) = 1$
- calc_f(val, power) 返回 $f(val^{power})$

```
// all in which f[pq] = f[p] * f[q] (gcd(p, q) = 1)
const int N = 1e7 + 5;
int pri[N / 5], notpri[N], prinum, minpri_cnt[N], f[N];

int calc_f(int val, int power) {

}

void init_pri() {
    for(int i = 2; i < N; i++) {
        if(!notpri[i]) pri[++prinum] = i, minpri_cnt[i] = 1, f[i] = calc_f(i, 1);
        for(int j = 1; j <= prinum && pri[j] * i < N; j++) {
            notpri[pri[j] * i] = pri[j];
            if(i % pri[j] == 0) {
                minpri_cnt[pri[j] * i] = minpri_cnt[i] + 1;
                f[pri[j] * i] = f[i] / calc_f(pri[j], minpri_cnt[i]) * calc_f(pri[j],
                    minpri_cnt[i] + 1);
                break;
            }
            minpri_cnt[pri[j] * i] = 1;
            f[pri[j] * i] = f[i] * calc_f(pri[j], 1);
        }
    }
}
```

1.2 快速幂

```
int ksm(int a, int b = mod - 2, int MOD_KSM = mod) {
    int ret = 1;
    while(b) {
        if(b & 1) {
            ret = ret * a % MOD_KSM;
        }
        a = a * a % MOD_KSM;
        b >>= 1;
    }
    return ret;
}
```

1.3 组合数

1.3.1 暴力

- 暴力求组合数 $\binom{n}{k}$, 时间复杂度 $O(\min(k, n-k))$.

- 前置：快速幂
- 模数必须是质数！

```
int binom(int n, int k) {
    if(n < 0 || k < 0 || k > n) { return 0; }
    k = min(n - k, k);
    int u = 1, v = 1;
    for(int i = 0; i < k; i++) {
        v = v * (i + 1) % mod;
        u = u * (n - i) % mod;
    }
    return u * ksm(v, mod - 2) % mod;
}
```

1.3.2 递推

- $O(n^2)$ 递推求，模数随意。

```
const int N = 31;
int C[N][N];

void init_C() {
    C[0][0] = C[1][0] = C[1][1] = 1;
    for(int i = 2; i < N; i++) {
        C[i][0] = 1;
        for(int j = 1; j <= i; j++)
            C[i][j] = (C[i - 1][j] + C[i - 1][j - 1]) % mod;
    }
}

int binom(int n, int k) {
    if(n < 0 || k < 0 || n < k) return 0;
    return C[n][k];
}
```

1.3.3 逆元

- 模数必须是质数!
- 前置：快速幂

```
const int N = 1e7 + 5;
const int mod = 1e9 + 7;
int facinv[N], fac[N];
void init_fac() {
    fac[0] = fac[1] = 1;
    for(int i = 2; i < N; i++) {
        fac[i] = fac[i - 1] * i % mod;
    }
    facinv[N - 1] = ksm(fac[N - 1], mod - 2);
    for(int i = N - 2; i >= 0; i--) {
```

```
        facinv[i] = facinv[i + 1] * (i + 1) % mod;
    }
}
int binom(int n, int k) {
    if(n < 0 || k < 0 || k > n) { return 0; }
    return fac[n] * facinv[n - k] % mod * facinv[k] % mod;
}

int inv[N];
void init_inv() {
    inv[0] = inv[1] = 1;
    for(int i = 2; i < N; i++) {
        inv[i] = (mod - mod / i) * inv[mod % i] % mod;
    }
}
```

1.4 ExGCD

- 求解 $ax + by = (a, b)$ 的特解 x_0, y_0.
- 通解 $x^* = x_0 + \frac{bk}{(a,b)}, y^* = y_0 - \frac{ak}{(a,b)}$ ($k \in \mathbb{Z}$) .

```
// ax + by = gcd(a, b)
// return gcd(a, b)
// all sol: x = x_0 + k[a, b] / a, y = y_0 - k[a, b] / b;
int exgcd(int &x, int &y, int a, int b) {
    if(!b) {
        x = 1;
        y = 0;
        return a;
    }
    int ret = exgcd(x, y, b, a % b);
    int t = x;
    x = y;
    y = t - a / b * y;
    return ret;
}
```

1.5 拉格朗日插值

- $f(x)$ 是多项式，并且我们知道一系列连续的点值 $f(l), \cdots, f(r)$ ，求解 $f(n)$。$O(r-l)$
- 前置：逆元组合数
- 模数为质数

```
int LagrangeInterpolation(vector<int> &y, int l, int r, int n) {
    if(n <= r && n >= l) return y[n];
    vector<int> lg(r - l + 3), rg(r - l + 3);
    int ret = 0;
    lg[0] = 1;
    rg[r - l + 2] = 1;
```

```
    for(int i = l; i <= r; i++) {
        lg[i - l + 1] = (long long) lg[i - l] * (n - i) % mod;
    }
    for(int i = r; i >= l; i--) {
        rg[i - l + 1] = (long long) rg[i - l + 2] * (n - i) % mod;
    }
    for(int i = l; i <= r; i++) {
        if(r - i & 1) {
            ret = (ret - (long long) y[i] * lg[i - l] % mod * rg[i - l + 2] % mod *
                facinv[i - l] % mod * facinv[r - i] % mod + mod) % mod;
        } else {
            ret = (ret + (long long) y[i] * lg[i - l] % mod * rg[i - l + 2] % mod *
                facinv[i - l] % mod * facinv[r - i] % mod) % mod;
        }
    }
    return ret;
}
```

1.6 原根

若 g 满足:

$$(g, m) = 1$$

$$\delta_m(g) = \phi(m)$$

则 g 为 m 的原根。找所有原根:

1. 找到最小的原根 g。如果一个数 g 是原根，那么 $\forall p | \phi(m) : g^{\frac{\phi(m)}{p}} \neq 1$
2. 找小于 m 与 $\phi(m)$ 互质的数 k ，则 g^k 也是原根（能覆盖所有原根）个数为 $\phi(\phi(m))$ 个。

题目：找出 n 的所有原根，间隔 d 输出。

```
void solve() {
    // d 是间隔输出（对题目无影响
    int n, d; cin >> n >> d;
    vector<int> pf; // 质因数分解
    int pn = phi[n];
    while(notpri[pn]) {
        int now = notpri[pn];
        pf.push_back(now);
        while(pn % now == 0) pn /= now;
    }
    if(pn != 1) {
        pf.push_back(pn);
    }
    int cnt = 0;
    int ming = -1;
    vector<int> ans, vis(n); // 记录答案
    // 找到最小的原根 min_g
    for(int i = 1; i < n; i++) {
        if(__gcd(i, n) != 1) continue;
        int judge = 1;
        for(auto &p : pf) {
```

```cpp
            if(ksm(i, phi[n] / p, n) == 1) {
                judge = 0;
                break;
            }
        }
        if(judge) {
            ming = i;
            break;
        }
    }
    // 还原出所有原根 g
    if(ming > 0) {
        for(int i = 1; i < n; i++) {
            if(__gcd(i, phi[n]) == 1) {
                int cur = ksm(ming, i, n);
                if(!vis[cur]) {
                    vis[cur] = 1;
                    ans.push_back(cur);
                }
            }
        }
    }
    // 排序输出所有原根
    cout << ans.size() << '\n';
    sort(ans.begin(), ans.end());
    for(auto as : ans) {
        cnt++;
        if(cnt % d == 0) {
            cout << as << ' ';
        }
    }
    cout << '\n';
}
```

1.7 Ex-Baby-Step-Giant-Step-Algorithm

BSGS

求解 $a^x = b \pmod p), (0 \le x < p)$

令 $x = A\lceil\sqrt{p}\rceil - B$，其中 $0 \le A, B \le \lceil\sqrt{p}\rceil$，则有 $a^{A\lceil\sqrt{p}\rceil - B} \equiv b \pmod p$，稍加变换，则有 $a^{A\lceil\sqrt{p}\rceil} \equiv ba^B \pmod p$。

我们已知的是 a, b，所以我们可以先算出等式右边的 ba^B 的所有取值，枚举 B，用 'hash'/'map' 存下来，然后逐一计算 $a^{A\lceil\sqrt{p}\rceil}$，枚举 A，寻找是否有与之相等的 ba^B，从而我们可以得到所有的 x，$x = A\lceil\sqrt{p}\rceil - B$。

注意到 A, B 均小于 $\lceil\sqrt{p}\rceil$，所以时间复杂度为 $\Theta\left(\sqrt{p}\right)$，用 'map' 则多一个 log。

exBSGS

其中 a, p 不一定互质。

当 $a \perp p$ 时，在模 p 意义下 a 存在逆元，因此可以使用 BSGS 算法求解。于是我们想办法让他们变得互质。

具体地，设 $d_1 = \gcd(a, p)$。如果 $d_1 \mid b$，则原方程无解。否则我们把方程同时除以 d_1，得到

$$\frac{a}{d_1} \cdot a^{x-1} \equiv \frac{b}{d_1} \pmod{\frac{p}{d_1}}$$

如果 a 和 $\frac{p}{d_1}$ 仍不互质就再除，设 $d_2 = \gcd\left(a, \frac{p}{d_1}\right)$。如果 $d_2 \mid \frac{b}{d_1}$，则方程无解；否则同时除以 d_2 得到

$$\frac{a^2}{d_1 d_2} \cdot a^{x-2} \frac{b}{d_1 d_2} \pmod{\frac{p}{d_1 d_2}}$$

同理，这样不停的判断下去。直到 $a \perp \frac{p}{d_1 d_2 \cdots d_k}$。

记 $D = \prod_{i=1}^{k} d_i$，于是方程就变成了这样：

$$\frac{a^k}{D} \cdot a^{x-k} \equiv \frac{b}{D} \pmod{\frac{p}{D}}$$

由于 $a \perp \frac{p}{D}$，于是推出 $\frac{a^k}{D} \perp \frac{p}{D}$。这样 $\frac{a^k}{D}$ 就有逆元了，于是把它丢到方程右边，这就是一个普通的 BSGS 问题了，于是求解 $x - k$ 后再加上 k 就是原方程的解啦。

注意，不排除解小于等于 k 的情况，所以在消因子之前做一下 $\Theta(k)$ 枚举，直接验证 $a^i \equiv b \pmod{p}$，这样就能避免这种情况。

- 注意，inv 必须由扩欧求！
- 注意开 long long
- 前置：ksm, exgcd 求逆元

```
int bsgs(int a, int b, int p) { //BSGS算法
    unordered_map<int, int> f;
    int m = ceil(sqrt(p));
    b %= p;
    for(int i = 1; i <= m; i++) {
        b = b * a % p;
        f[b] = i;
    }
    int tmp = ksm(a, m, p);
    b = 1;
    for(int i = 1; i <= m; i++) {
        b = b * tmp % p;
        if(f[b]) {
            return (i * m - f[b] + p) % p;
        }
    }
    return -1;
}
int exbsgs(int a, int b, int p) {
    b %= p;
    a %= p;
    if(b == 1 || p == 1) {
        return 0; //特殊情况，x=0时最小解
    }
    int g = __gcd(a, p), k = 0, na = 1;
    while(g > 1) {
        if(b % g != 0) {
            return -1; //无法整除则无解
        }
        k++;
        b /= g;
        p /= g;
```

```
        na = na * (a / g) % p;
        if(na == b) {
            return k; //na=b说明前面的a的次数为0，只需要返回k
        }
        g = __gcd(a, p);
    }
    int f = bsgs(a, b * inv(na, p) % p, p);
    if(f == -1) {
        return -1;
    }
    return f + k;
}
```

1.8 逆元

1.8.1 exgcd 求逆元

- 前置：exgcd
- $(x, p) = 1$

```
int inv(int x, int p) {
    int y, k;
    int gcd = exgcd(y, k, x, p);
    int moder = p / gcd;
    return (y % moder + moder) % moder;
}
```

1.8.2 快速幂求逆元

根据费马小定理：$p \in primes \to a^{-1} \equiv a^{p-2} \pmod p$

1.8.3 整数除法取模

如果 $\frac{a}{b} \in \mathbb{N}$, $b \times p$ 可以在计算机中表示，那么 $\frac{a}{b} \bmod p = \frac{a \bmod (p \times b)}{b}$

1.9 上下取整

- b 必须为正整数。

```
// b must be positive integer

int updiv(int a, int b) {
    return a > 0 ? (a + b - 1) / b : a / b;
}

int downdiv(int a, int b) {
    return a > 0 ? a / b : (a - b + 1) / b;
}
```

1.10 多项式全家桶

- 注意调整原根 g，模数 mod，N 开 3 到 4 倍数据范围，附录 A
- 注意 resize()
- 注意 Inv/Ln 的时候常数项不能为 0
- 注意 Exp 的时候常数项必须是 0
- 注意这里面的 ksm() 第三个参数是初值而不是模数

```
#define fp(i, a, b) for (int i = (a); i <= (b); i++)
#define fd(i, a, b) for (int i = (a); i >= (b); i--)
const int N = 3e5 + 5, mod = 998244353; // (N = 4 * n)

using ll = int64_t;
using Poly = vector<int>;
/*---------------------------------------------------------------------------*/
// 二次剩余
class Cipolla {
   int mod, I2{};
   using pll = pair<ll, ll>;
#define X first
#define Y second
   ll mul(ll a, ll b) const { return a * b % mod; }
   pll mul(pll a, pll b) const { return {(a.X * b.X + I2 * a.Y % mod * b.Y) % mod, (a.X * b.Y + a.Y * b.X) % mod}; }
   template<class T> T ksm(T a, int b, T x) { for (; b; b >>= 1, a = mul(a, a)) if (b & 1) x = mul(x, a); return x; }
public:
   Cipolla(int p = 0) : mod(p) {}
   pair<int, int> sqrt(int n) {
      int a = rand(), x;
      if (!(n %= mod)) return {0, 0};
      if (ksm(n, (mod - 1) >> 1, 1ll) == mod - 1) return {-1, -1};
      while (ksm(I2 = ((ll) a * a - n + mod) % mod, (mod - 1) >> 1, 1ll) == 1) a = rand();
      x = (int) ksm(pll{a, 1}, (mod + 1) >> 1, {1, 0}).X;
      if (2 * x > mod) x = mod - x;
      return {x, mod - x};
   }
#undef X
#undef Y
};
/*-------------------------------------Modular---------------------------------*/
#define mul(a, b) (ll(a) * (b) % mod)
#define add(a, b) (((a) += (b)) >= mod ? (a) -= mod : 0) // (a += b) %= P
#define dec(a, b) (((a) -= (b)) < 0 ? (a) += mod: 0) // ((a -= b) += P) %= P
Poly getInv(int L) { Poly inv(L); inv[1] = 1; fp(i, 2, L - 1) inv[i] = mul((mod - mod / i), inv[mod % i]); return inv; }
int ksm(ll a, int b = mod - 2, ll x = 1) { for (; b; b >>= 1, a = a * a % mod) if (b & 1) x = x * a % mod; return x; }
auto inv = getInv(N); // NOLINT
/*-----------------------------------NTT-------------------------------------*/
```

```cpp
namespace NTT {
    const int g = 3;
    Poly Omega(int L) {
        int wn = ksm(g, mod / L);
        Poly w(L); w[L >> 1] = 1;
        fp(i, L / 2 + 1, L - 1) w[i] = mul(w[i - 1], wn);
        fd(i, L / 2 - 1, 1) w[i] = w[i << 1];
        return w;
    }
    auto W = Omega(1 << 20); // NOLINT
    void DIF(int *a, int n) {
        for (int k = n >> 1; k; k >>= 1)
            for (int i = 0, y; i < n; i += k << 1)
                for (int j = 0; j < k; ++j)
                    y = a[i + j + k], a[i + j + k] = mul(a[i + j] - y + mod, W[k + j]), add(a[i + j], y);
    }
    void IDIT(int *a, int n) {
        for (int k = 1; k < n; k <<= 1)
            for (int i = 0, x, y; i < n; i += k << 1)
                for (int j = 0; j < k; ++j)
                    x = a[i + j], y = mul(a[i + j + k], W[k + j]),
                    a[i + j + k] = x - y < 0 ? x - y + mod : x - y, add(a[i + j], y);
        int Inv = mod - (mod - 1) / n;
        fp(i, 0, n - 1) a[i] = mul(a[i], Inv);
        reverse(a + 1, a + n);
    }
}
/*------------------------------------Polynomial 全家桶 -----------------------------------*/
namespace Polynomial {
    // basic operator
    int norm(int n) { return 1 << (__lg(n - 1) + 1); }
    void norm(Poly &a) { if (!a.empty()) a.resize(norm(a.size()), 0); else a = {0}; }
    void DFT(Poly &a) { NTT::DIF(a.data(), a.size()); }
    void IDFT(Poly &a) { NTT::IDIT(a.data(), a.size()); }
    Poly &dot(Poly &a, Poly &b) { fp(i, 0, a.size() - 1) a[i] = mul(a[i], b[i]); return a; }

    // mul / div int
    Poly &operator*=(Poly &a, int b) { for (auto &x : a) x = mul(x, b); return a; }
    Poly operator*(Poly a, int b) { return a *= b; }
    Poly operator*(int a, Poly b) { return b * a; }
    Poly &operator/=(Poly &a, int b) { return a *= ksm(b); }
    Poly operator/(Poly a, int b) { return a /= b; }

    // Poly add / sub
    Poly &operator+=(Poly &a, Poly b) {
        a.resize(max(a.size(), b.size()));
        fp(i, 0, b.size() - 1) add(a[i], b[i]);
        return a;
    }
```

```
Poly operator+(Poly a, Poly b) { return a += b; }
Poly &operator-=(Poly &a, Poly b) {
   a.resize(max(a.size(), b.size()));
   fp(i, 0, b.size() - 1) dec(a[i], b[i]);
   return a;
}
Poly operator-(Poly a, Poly b) { return a -= b; }

// Poly mul
Poly operator*(Poly a, Poly b) {
   int n = a.size() + b.size() - 1, L = norm(n);
   if (a.size() <= 8 || b.size() <= 8) {
      Poly c(n);
      fp(i, 0, a.size() - 1) fp(j, 0, b.size() - 1)
         c[i + j] = (c[i + j] + (ll) a[i] * b[j]) % mod;
      return c;
   }
   a.resize(L), b.resize(L);
   DFT(a), DFT(b), dot(a, b), IDFT(a);
   return a.resize(n), a;
}

// Poly inv
Poly Inv2k(Poly a) { // |a| = 2 ^ k
   int n = a.size(), m = n >> 1;
   if (n == 1) return {ksm(a[0])};
   Poly b = Inv2k(Poly(a.begin(), a.begin() + m)), c = b;
   b.resize(n), DFT(a), DFT(b), dot(a, b), IDFT(a);
   fp(i, 0, n - 1) a[i] = i < m ? 0 : mod - a[i];
   DFT(a), dot(a, b), IDFT(a);
   return move(c.begin(), c.end(), a.begin()), a;
}
Poly Inv(Poly a) {
   int n = a.size();
   norm(a), a = Inv2k(a);
   return a.resize(n), a;
}

// Poly div / mod
Poly operator/(Poly a,Poly b){
   int k = a.size() - b.size() + 1;
   if (k < 0) return {0};
   reverse(a.begin(), a.end());
   reverse(b.begin(), b.end());
   b.resize(k), a = a * Inv(b);
   a.resize(k), reverse(a.begin(), a.end());
   return a;
}
pair<Poly, Poly> operator%(Poly a, const Poly& b) {
   Poly c = a / b;
   a -= b * c, a.resize(b.size() - 1);
   return {c, a};
}
```

```

// Poly calculus
Poly deriv(Poly a) {
    fp(i, 1, a.size() - 1) a[i - 1] = mul(i, a[i]);
    return a.pop_back(), a;
}
Poly integ(Poly a) {
    a.push_back(0);
    fd(i, a.size() - 1, 1) a[i] = mul(inv[i], a[i - 1]);
    return a[0] = 0, a;
}

// Poly ln
Poly Ln(Poly a) {
    int n = a.size();
    a = deriv(a) * Inv(a);
    return a.resize(n - 1), integ(a);
}

// Poly exp
Poly Exp(Poly a) {
    int n = a.size(), k = norm(n);
    Poly b = {1}, c, d; a.resize(k);
    for (int L = 2; L <= k; L <<= 1) {
        d = b, b.resize(L), c = Ln(b), c.resize(L);
        fp(i, 0, L - 1) c[i] = a[i] - c[i] + (a[i] < c[i] ? mod : 0);
        add(c[0], 1), DFT(b), DFT(c), dot(b, c), IDFT(b);
        move(d.begin(), d.end(), b.begin());
    }
    return b.resize(n), b;
}

// Poly sqrt
Poly Sqrt(Poly a) {
    int n = a.size(), k = norm(n); a.resize(k);
    Poly b = {(new Cipolla(mod))->sqrt(a[0]).first, 0}, c;
    for (int L = 2; L <= k; L <<= 1) {
        b.resize(L), c = Poly(a.begin(), a.begin() + L) * Inv2k(b);
        fp(i, L / 2, L - 1) b[i] = mul(c[i], (mod + 1) / 2);
    }
    return b.resize(n), b;
}

// Poly pow
Poly Pow(Poly &a, int b) { return Exp(Ln(a) * b); } // a[0] = 1
Poly Pow(Poly a, int b1, int b2) { // b1 = b % mod, b2 = b % phi(mod) and b >= n
     iff a[0] > 0
    int n = a.size(), d = 0, k;
    while (d < n && !a[d]) ++d;
    if ((ll) d * b1 >= n) return Poly(n);
    a.erase(a.begin(), a.begin() + d);
    k = ksm(a[0]), norm(a *= k);
    a = Pow(a, b1) * ksm(k, mod - 1 - b2);
```

```
        a.resize(n), d *= b1;
        fd(i, n - 1, 0) a[i] = i >= d ? a[i - d] : 0;
        return a;
    }

}
using namespace Polynomial;
```

1.11 数学公式

1.11.1

2 杂项

2.1 debuger

```cpp
#define out(args...) { cout << "Line " << __LINE__ << ": [" << #args << "] = [";
    debug(args); cout << "]\n"; }

template<typename T> void debug(T a) { cout << a; }

template<typename T, typename...args> void debug(T a, args...b) {
    cout << a << ", ";
    debug(b...);
}

template<typename T>
ostream& operator << (ostream &os, const vector<T> &a) {
    os << "[";
    int f = 0;
    for(auto &x : a) os << (f++ ? ", " : "") << x;
    os << "]";
    return os;
}

template<typename T>
ostream& operator << (ostream &os, const set<T> &a) {
    os << "{";
    int f = 0;
    for(auto &x : a) os << (f++ ? ", " : "") << x;
    os << "}";
    return os;
}

template<typename T>
ostream& operator << (ostream &os, const multiset<T> &a) {
    os << "{";
    int f = 0;
    for(auto &x : a) os << (f++ ? ", " : "") << x;
    os << "}";
    return os;
}

template<typename A, typename B>
ostream& operator << (ostream &os, const map<A, B> &a) {
    os << "{";
    int f = 0;
    for(auto &x : a) os << (f++ ? ", " : "") << x;
    os << "}";
    return os;
}

template<typename A, typename B>
ostream& operator << (ostream &os, const pair<A, B> &a) {
    os << "(" << a.first << ", " << a.second << ")";
```

```
    return os;
}

template<typename A, typename B, typename C>
ostream& operator << (ostream &os, const tuple<A, B, C> &a) {
    os << "(" << get<0>(a) << ", " << get<1>(a) << ", " << get<2>(a) << ")";
    return os;
}

template<typename A, typename B, typename C, typename D>
ostream& operator << (ostream &os, const tuple<A, B, C, D> &a) {
    os << "(" << get<0>(a) << ", " << get<1>(a) << ", " << get<2>(a) << ", " << get
        <3>(a) << ")";
    return os;
}
```