XCPC Templates

Khoray

April 4, 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 $f(val^{power})$

```cpp
// 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

```cpp
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) {
```

```
            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 $d_1 \nmid b$，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 $d_2 \nmid \frac{b}{d_1}$，则方程无解；否则同时除以 d_2 得到

$$\frac{a^2}{d_1d_2}\cdot a^{x-2}\frac{b}{d_1d_2}\pmod{\frac{p}{d_1d_2}}$$

同理，这样不停的判断下去。直到 $a\perp\frac{p}{d_1d_2\cdots d_k}$。

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

```cpp
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 线性基

- $O(\log x)$ insert
- $O(\log^2 x)$ get-kth
- $O(\log x)$ get-max
- 如果问能否通过选一些数（不能不选）异或得到 0，必须特判。

```
struct linear_basis {
    vector<int> base, kth;
    int size, max_size, builded;
    linear_basis(int n) : base(n), size(0), max_size(n), builded(0) {}

    void insert(int x) {
        builded = 0;
        for(int i = max_size - 1; i >= 0; i--) {
            if((x >> i) & 1) {
                if(!base[i]) {
                    base[i] = x, size++;
                    break;
                }
                else x ^= base[i];
            }
        }
    }

    int get_max() {
        int ret = 0;
        for(int i = max_size - 1; i >= 0; i--) {
            if(!((ret >> i) & 1) && base[i]) ret ^= base[i];
        }
        return ret;
    }

    bool can_eq(int x) {
        int now = 0;
        for(int i = max_size - 1; i >= 0; i--) {
            if(((now >> i) & 1) != ((x >> i) & 1)) {
                if(!base[i]) return false;
                else now ^= base[i];
            }
        }
        return true;
    }


    int get_kth(int k) {
        if(k >= 1ll << size) return -1;
        if(!builded) buildk();
        int ret = 0;
        for(int i = size - 1; ~i; i--) {
            if(k >> i & 1) {
```

```
                ret ^= kth[i];
            }
        }
        return ret;
    }

    int get_rank(int x) { // return the number of values less than x TODO
        int tmpsz = size, ret = 0, now = 0;
        for(int i = max_size - 1; i >= 0; i--) {
            if(base[i]) tmpsz--;
            if((x >> i) & 1) {
                if(!((now >> i) & 1)) {
                    ret += tmpsz * tmpsz;
                    if(!base[i]) break;
                    else now ^= base[i];
                } else {
                    if(base[i]) ret += tmpsz * tmpsz;
                }
            } else {
                if((now >> i) & 1) {
                    if(!base[i]) break;
                    else now ^= base[i];
                }
            }
        }
        return ret;
    }
private:
    void buildk() {
        builded = 1;
        kth.resize(size);
        int cnt = size;
        for(int i = max_size - 1; ~i; i--) {
            if(base[i]) {
                for(int j = i - 1; ~j; j--) {
                    if(base[i] >> j & 1) {
                        base[i] ^= base[j];
                    }
                }
            }
        }
        for(int i = max_size - 1; ~i; i--) {
            if(base[i]) {
                kth[--cnt] = base[i];
            }
        }
    }
};
```

1.11 高斯消元

- equ 是方程个数，n 是变元个数，答案存在 ans。

- return：无解 (-1)，自由变元个数。

1.11.1 解异或线性方程组

```cpp
const int N = 1005;
template<int N>
struct Matrix {
    bitset<N> mat[N];
    Matrix() { }
    bitset<N> &operator [] (int idx) { return mat[idx]; }
};

template<int N>
int guass(int n, int equ, Matrix<N> a, vector<int> b, vector<int> &ans) {
    fill(ans.begin(), ans.end(), 0);
    vector<int> fre(n + 1);
    int row, col;
    for(row = 1, col = 1; col <= n; col++) {
        if(!a[row][col]) {
            int sw = 0;
            for(int i = row + 1; i <= equ; i++) {
                if(a[i][col]) {
                    swap(a[row], a[i]);
                    swap(b[row], b[i]);
                    sw = 1;
                    break;
                }
            }
            if(!sw) {
                fre[col] = 1;
                continue;
            }
        }
        for(int i = row + 1; i <= equ; i++) {
            if(a[i][col]) {
                a[i] ^= a[row];
                b[i] ^= b[row];
            }
        }
        row++;
    }
    if(row <= equ) {
        for(int i = row; i <= equ; i++) {
            if(b[i]) return -1;
        }
    }
    int all = 0;
    for(col = n; col >= 1; col--) {
        if(fre[col]) {
            ans[col] = 1;
            all++;
        } else {
            row--;
```

```
        ans[col] = b[row];
        for(int i = col + 1; i <= n; i++) {
          if(a[row][i]) ans[col] ^= ans[i];
        }
      }
    }
  return all;
}
```

1.11.2 解 double 线性方程组

```
const int N = 1005;
template<int N>
struct Matrix {
  bitset<N> mat[N];
  Matrix() { }
  bitset<N> &operator [] (int idx) { return mat[idx]; }
};
template<int N>
int guass(int n, int equ, Matrix<N> a, vector<double> b, vector<double> &ans) {
  fill(ans.begin(), ans.end(), 0);
  vector<int> fre(n + 1);
  int row, col;
  for(row = 1, col = 1; col <= n; col++) {
    double mx = fabs(a[row][col]);
    int mxp = row;
    for(int i = row + 1; i <= equ; i++) {
      if(fabs(a[row][col]) > mx) {
        mx = fabs(a[row][col]);
        mxp = i;
      }
    }
    if(mxp != row) {
      for(int i = col; i <= n; i++) {
        swap(a[row][i], a[mxp][i]);
      }
      swap(b[row], b[mxp]);
    }
    if(fabs(a[row][col]) < eps) {
      fre[col] = 1;
    }
    for(int i = row + 1; i <= equ; i++) {
      if(fabs(a[i][col]) > eps) {
        double k = a[i][col] / a[row][col];
        for(int j = col; j <= n; j++) {
          a[i][j] -= a[row][j] * k;
        }
        b[i] -= b[row] * k;
      }
    }
    row++;
  }
```

```
    // 判断解是否存在
    if(row <= equ) {
        for(int i = row; i <= equ; i++) {
            if(fabs(b[i]) > eps) return -1;
        }
    }

    // 回代求解
    int all = 0;
    for(col = n; col >= 1; col--) {
        if(fre[col]) {
            ans[col] = 0;
            all++;
        } else {
            row--;
            ans[col] = b[row];
            for(int i = col + 1; i <= n; i++) {
                ans[col] -= ans[i] * a[row][i];
            }
            ans[col] /= a[row][col];
        }
    }
    return all;
}
```

1.11.3 解模意义线性方程组

- 时间复杂度 $O(n^3 \log mod)$

```
#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

const int N = 1005;
const int mod = 1e9 + 7;
template<int N>
struct Matrix {
    int mat[N][N];
    Matrix() { }
    int* operator [] (int idx) { return mat[idx]; }
};
template<int N>
int guass(int n, int equ, Matrix<N> a, vector<int> b, vector<int> &ans) {
    fill(ans.begin(), ans.end(), 0);
    vector<int> fre(n + 1);
    int row, col;
    for(row = 1, col = 1; col <= n; col++) {
        if(!a[row][col]) {
            int sw = 0;
            for(int i = row + 1; i <= equ; i++) {
                if(a[i][col]) {
                    for(int j = col; j <= n; j++) {
```

```
                    swap(a[row][j], a[i][j]);
                 }
                 swap(b[row], b[i]);
                 sw = 1;
                 break;
              }
           }
           if(!sw) {
              fre[col] = 1;
              continue;
           }
        }
        for(int i = row + 1; i <= equ; i++) {
           if(a[i][col]) {
              int k = a[i][col] * ksm(a[row][col]) % mod;
              for(int j = col; j <= n; j++) {
                 dec(a[i][j], a[row][j] * k % mod);
              }
              dec(b[i], b[row] * k % mod);
           }
        }
        row++;
     }
     if(row <= equ) {
        for(int i = row; i <= equ; i++) {
           if(b[i]) return -1;
        }
     }
     int all = 0;
     for(col = n; col >= 1; col--) {
        if(fre[col]) {
           ans[col] = 0;
           all++;
        } else {
           row--;
           ans[col] = b[row];
           for(int i = col + 1; i <= n; i++) {
              dec(ans[col], ans[i] * a[row][i] % mod);
           }
           mul(ans[col], ksm(a[row][col]));
        }
     }
     return all;
}
```

1.12 Miller-Rabin

- 前置：快速幂 (___int128!!!)
- int 范围：2, 7, 61
- long long 范围：2, 325, 9375, 28178, 450775, 9780504, 1795265022
- 4E13：2, 2570940, 211991001, 3749873356

- 3E15: 2, 2570940, 880937, 610386380, 4130785767
- 注意看判断范围是 int 还是 long long

```
bool is_prime(int n) {
   if(n < 2 || n % 6 % 4 != 1) return (n | 1) == 3;
   int A[] = {2, 325, 9375, 28178, 450775, 9780504, 1795265022},
            s = __builtin_ctzll(n - 1), d = n >> s;
   for(int a : A) { // ^ count t ra i l in g zeroes
      int p = ksm(a % n, d, n), i = s;
      while(p != 1 && p != n - 1 && a % n && i--)
         p = (__int128) p * p % n;
      if(p != n - 1 && i != s) return 0;
   }
   return 1;
}
```

1.13 Pollard-Rho

- 前置：Miller-Rabin

```
inline int pollard_rho(int x) {
   auto f = [&](int x, int c, int n) {
      return ((__int128) x * x + c) % n;
   };
   int s = 0, t = 0, c = 1ll * rand() % (x - 1) + 1;
   int stp = 0, goal = 1;
   int val = 1;
   for(goal = 1;; goal <<= 1, s = t, val = 1) {
      for(stp = 1; stp <= goal; ++stp) {
         t = f(t, c, x);
         val = (__int128)val * abs(t - s) % x;
         if((stp % 127) == 0) {
            int d = __gcd(val, x);
            if(d > 1)
               return d;
         }
      }
      int d = __gcd(val, x);
      if(d > 1)
         return d;
   }
}

inline void get_factor_a(int x, vector<int> &fac) {
   if(x < 2) return;
   if(is_prime(x)) {
      fac.push_back(x);
      return;
   }
   int p = x;
   while(p >= x) p = pollard_rho(x);
```

```
    while((x % p) == 0) x /= p;
    get_factor_a(x, fac), get_factor_a(p, fac);
}
```

1.14 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
```

```cpp
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--------------------------------------*/
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;
```

1.15 数学公式

1.15.1 多项式牛顿迭代

1.15.2 生成函数/形式幂级数

1.15.3 莫比乌斯反演

1.15.4 分配问题

1. n 个球放到 k 个盒子，每个盒子只有一种形态。

n 个球	k 个盒子	盒子可以为空	每个盒子内至少有一个球
有标号	有标号	k^n	$k!S_2(n,k)$
有标号	无标号	$\sum_{i=1}^{k} S_2(n,i)$	$S_2(n,k)$
无标号	有标号	$C(n+k-1,k-1)$	$C(n-1,k-1)$
无标号	无标号	$p(n+k,k)$	$p(n,k)$

其中 $S_2(n,k)$ 为第二类 'Stirling 数 '，$p(n,k)$ 为 '分拆数 '

2. n 个球放到 k 个盒子, 每个盒子至少一个球，装有 i 个球的盒子有 f_i $(i \geq 1)$ 种形态。$F(x)$ 是 f_i 的 $o.g.f.$，$E(x)$ 是 f_i 的 $e.g.f.$

n 个球	k 个盒子	关于 n 方案的生成函数
有标号	有标号	$e.g.f = E(x)^k$
有标号	无标号	$e.g.f = \frac{1}{k!}E(x)^k$
无标号	有标号	$o.g.f = F(x)^k$
无标号	无标号	不会

3. n 个球放到若干盒子，每个盒子至少一个球，装有 i 个球的盒子有 f_i $(i \geq 1)$ 种形态.

n 个球	盒子	方案的生成函数
有标号	有标号	$e.g.f = \frac{1}{1-E(x)}$
有标号	无标号	$e.g.f = \exp(E(x))$
无标号	有标号	$o.g.f = \frac{1}{1-F(x)}$
无标号	无标号	$o.g.f = \prod_{i\geq 1}\left(\frac{1}{1-x^i}\right)^{f_i} = \exp\left(\sum_{j\geq 1}\frac{1}{j}F\left(x^j\right)\right)$

1.15.5 第二类斯特林数

性质：

1. ${n \brace k} = {n-1 \brace k-1} + k{n-1 \brace k}$ 边界条件 ${x\neq 1 \brace 0} = 0, {0 \brace 0} = 1$
2. ${n \brace k} = \frac{1}{k!}\sum\limits_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^n$ （用来求第二类斯特林数 · 行，考虑容斥）
3. $k^n = \sum\limits_{i=0}^{k}{n \brace i}k^{\underline{i}}$ （性质 2 的反演）
4. 重要公式：$x^n = \sum\limits_{k=0}^{n}{n \brace k}x^{\underline{k}}$ （基于 3，x 个球 n 个盒子）
5. 关于 n 的 $e.g.f. = \dfrac{(e^x-1)^k}{k!}$ （考虑将 n 个物品染成 k 种颜色，每一种物品的指数生成函数为 $e^x - 1$ ）（用于求第二类斯特林数 · 列）

1.15.6 第一类斯特林数

性质：

1. ${n \brack k} = {n-1 \brack k-1} + (n-1){n-1 \brack k}$ 边界条件 ${x\neq 1 \brack 0} = 0, {0 \brack 0} = 1$
2. 重要公式：$x^{\overline{n}} = \sum\limits_{k=0}^{n}{n \brack k}x^n$ （x 个球 n 个盒子）（用于计算第一类斯特林数 · 行）
3. 有符号的第一类斯特林数：$S_1(n,k) = (-1)^{n+k}{n \brack k}$

   $x^{\underline{n}} = \sum\limits_{k=0}^{n}S_1(n,k)x^n$
4. 关于 n 的 $e.g.f. = \dfrac{(-\ln(1-x))^k}{k!}$. (列)

1.15.7 分拆数

性质：

1. $p(n,k) = p(n-1,k-1) + p(n-k,k)$
2. $o.g.f. = x^k\prod\limits_{i=1}^{k}\frac{1}{1-x^i}$ （考虑 Ferrers 图中长度为 i 的一列有多少个）

1.15.8 五边形数

性质：

1. $p(n) = \sum\limits_{k=1}^{n}p(n,k)$
2. $o.g.f. = \prod\limits_{i\geq 1}\frac{1}{1-x^i}$

   考虑求 ln，$o.g.f. = \exp(\sum_{n\geq 1}\sum_{d|n}\frac{x^n}{d})$ $O(n\log n)$
3. 递推公式：

   (a) 考虑 $Q(x) = \prod_{i\geq 1}(1-x^i)$

   (b) $Q(x) = \sum_{n\geq 0}(q_{even}(n) - q_{odd}(n))x^n$ $\quad q_{even/odd}$ 表示有偶数/奇数行，每一行的个数都不相同，大部分 $q_{even}(n)$ 和 $q_{odd}(n)$ 抵消，小部分也就是有 b 行，$n = \frac{b(3b-1)}{2}$ 和 $n = \frac{b(3b+1)}{2}$ 无法抵消。其系数都是 $(-1)^b$

(c) 则 $Q(x)=1+\sum_{i\geq 1}(-1)^i x^{\frac{(3i\pm 1)i}{2}}$

(d) $P(x)=Q^{-1}(x)$ 则, $p(n)=\sum_{k\geq 1}(-1)^{k-1}\left(p\left(n-\frac{(3k-1)k}{2}\right)+p\left(n-\frac{(3k+1)k}{2}\right)\right)$

1.15.9 Polya

置换群

G 是置换的集合，$\circ$ 是置换的复合，且 $(G,\circ)$ 为一个群时，称 $(G,\circ)$ 为一个置换群。

旋转群:

设 n 元环的 n 个结点分别为 $a_1,a_2,...,a_n$，旋转操作可以看成 $A=a_1,...,a_n$ 上的 n 个置换，其中第 i 个置换为 $g_i=[i+1,i+2,...,n,1,...,i]$。

设集合 $G=\{g_0,g_1,...,g_{n-1}\}$，则 $(G,\circ)$ 是一个置换群，称为正 n 边形的旋转群。

群对集合的作用

一个操作会将一个对象改变为另一个对象，形式化地:

设 $(G,\circ)$ 是一个群, 其单位元为 e,X 是一个集合, 群 G 对集合 X 的一个作用是一个 $G\times X$ 到 X 的映射 f, 满足:

- $\forall x\in X.f(e,x)=x$
- $\forall g,h\in G.f(h\circ g,x)=f(h,f(g,x))$ 我们把 $f(g,x)$ 简记成 $g_f(x)$ 或（在没有歧义的情况下记成）$g(x)$ 。
- 设 n 元环的 n 个结点分别为 $a_1,a_2,\ldots,a_n$, 令 $A=\{a_1,\ldots,a_n\}$ 。设颜色集合为 $B=\{b_1,b_2,\ldots,b_m\}$ 。给 n 元环染色可以看成 A 到 B 的一个映射 $x:A\to B$, 令 X 是所有这些映射的集合, 即 $X=\{x\mid x:A\to B\}$ 。
- 令 $G=\{g_0,g_1,\ldots,g_{n-1}\}$ 是正 n 边形的旋转群, 定义 $G\times X$ 到集合 X 的映射 f, 其中 $\forall i=0..n-1,x\in X.y=f(g_i,x)$ 满足 $\forall j=1..n.y(a_j)=x(g_i(a_j))$, 可以证明 f 是群 G 到集合 X 的一个作用, 因此我们简记 $y=f(g_i,x)$ 为 $y=g_i(x)$ 。
- X 上的 G 关系为 $R_G=\{(x,y)\mid x,y\in X\wedge(\exists g\in G.y=g(x))\},xR_Gy$ 当且仅当染色方案 x 能通过旋转得到 y 。

要求所有不同的染色方案, 即是求 X 上的 G-轨道的数量。

Burnside 引理

设有限群 $(G,\circ)$ 作用在有限集 X 上, 则 X 上的 G-轨道数量为

$$N=\frac{1}{|G|}\sum_{g\in G}\Psi(g)$$

其中 $\Psi(g)$ 表示 $g(x)=x$ 的 x 的数量。

轮换指标

设 $(G,\circ)$ 是一个 n 元置换的置换群, 它的轮换指标为

$$P_G\left(x_1,x_2,\ldots,x_n\right)=\frac{1}{|G|}\sum_{g\in G}x_1^{b_1}x_2^{b_2}\ldots x_n^{b_n}$$

$x_i^{b_i}$ 表示 g 这个置换的长度为 i 的环有 b_i 个。

正 n 边形旋转群轮换指标:

$$P_G=\frac{1}{n}\sum_{d\mid n}\varphi(d)x_d^{n/d}$$

正 n 边形二面体群轮换指标 (即可对称):

$$P_G=\frac{1}{2n}\sum_{d\mid n}\varphi(d)x_d^{n/d}+\begin{cases}\frac{1}{2}x_1x_2^{\frac{n-1}{2}}, & n \text{ 为奇数}\\ \frac{1}{4}\left(x_2^{\frac{n}{2}}+x_1^2x_2^{\frac{n-2}{2}}\right), & n \text{ 为偶数}\end{cases}$$

正方体置换群:

顶点置换群:

$$P_G=\frac{1}{24}\left(x_1^8+8x_1^2x_3^2+9x_2^4+6x_4^2\right)$$

边置换群:

$$P_G=\frac{1}{24}\left(x_1^{12}+8x_3^4+6x_1^2x_2^5+3x_2^6+6x_4^3\right)$$

面置换群:

$$P_G=\frac{1}{24}\left(x_1^6+8x_3^2+6x_2^3+3x_1^2x_2^2+6x_1^2x_4\right)$$

Polya 定理

集合 X 可以看成是给集合 $A=\{a_1,a_2,\ldots,a_n\}$ 的每个元素赋予式样（颜色, 种类等）的映射的集合

引入表示式样的集合 B, 令 $X=\{x\mid x:A\to B\}$, 记为 B^A

式样清单: G 作用在 B^A 上的 G-轨道的集合称为 B^A 关于 G 的**式样清单**, 记为 F

种类的权值: 假设 B 上的每个元素 b 都赋予了权值 $w(b)$

$f\in B^A$ 的权值: 定义 $w(f):=\prod_{a\in A}w(f(a))$

G-轨道的权值: $w(F):=w(f)$, 任选一个 $f\in F$

定理:

B^A 关于 G 的**式样清单**记为 $\mathcal{F}$, 则

$$\sum_{F\in\mathcal{F}}w(F)=P_G\left(\sum_{b\in B}w(b),\sum_{b\in B}w(b)^2,\ldots,\sum_{b\in B}w(b)^n\right)$$

2 杂项

2.1 debuger

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

#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;
}

```

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

template<typename A, size_t N>
ostream& operator << (ostream &os, const array<A, N> &a) {
    os << "{";
    int f = 0;
    for(int i = 0; i < N; i++) {
        os << (f++ ? ", " : "") << a[i];
    }
    os << "}";
    return os;
}
```