Challestend's ACM Templates

Challestend Rehtorbegnaro

Last Update: 2023 年 8 月 3 日

目录

1 数论

1.1 扩展欧几里得算法

当且仅当 $(a,b) \mid c$ 时，不定方程

$$ax+by=c$$

有整数解。其通解为

$$x = x_0 + \frac{b}{(a,b)} \cdot t$$

$$y = y_0 - \frac{a}{(a,b)} \cdot t$$

```
inline void exgcd(int a, int b, int &x, int &y)
{
    if (b == 0)
    {
        x = 1;
        y = 0;
        return;
    }
    else
    {
        exgcd(b, a % b, y, x);
        y -= a / b * x;
    }
}
```

1.2 CRT

一元线性同余方程组

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

若 m_i 两两互质，则通解为

$$\sum_{k=1}^{n} a_i t_i M_i \pmod{M}$$

其中

$$M = \prod_{i=1}^{n} m_i$$

$$M_i = \frac{M}{m_i}$$

$$t_i = M_i^{-1} \pmod{m_i}$$

若 m_i 不保证两两互质，设有两个方程形如

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \end{cases}$$

设

$$x = pm_1 + a_1 = qm_2 + a_2$$

解不定方程

$$m_1 p - m_2 q = a_2 - a_1$$

即可得到原来的两个方程的解

$$x \equiv pm_1 + a_1 \pmod{[m_1, m_2]}$$

1.3 Lucas 定理

如果 p 是质数，有

$$\binom{n}{m} \equiv \binom{\lfloor \frac{n}{p} \rfloor}{\lfloor \frac{m}{p} \rfloor} \binom{n \bmod p}{m \bmod p} \pmod{p}$$

如果模数不保证是质数，将其唯一分解后对于每个 p^k 分别求解再用中国剩余定理合并。令

$$\varphi(l, r, x) = \prod_{i=l}^{r} [x \nmid i] i$$

$$f(n) = f\left(\left\lfloor \frac{n}{p} \right\rfloor\right) \varphi^{\left\lfloor \frac{n}{p^k} \right\rfloor}(1, p^k, p) \varphi\left(p^k \left\lfloor \frac{n}{p^k} \right\rfloor + 1, n, p\right)$$

$$g(n) = \left\lfloor \frac{n}{p} \right\rfloor + g\left(\left\lfloor \frac{n}{p} \right\rfloor\right)$$

可以得出

$$\binom{n}{m} \equiv \frac{f(n)}{f(m) f(n-m)} \cdot p^{g(n) - g(m) - g(n-m)} \pmod{p^k}$$

1.4 康托展开

$$A = \sum_{i=1}^{n} \mathrm{cnt}(i) \cdot (n-i)!$$

其中 $\mathrm{cnt}(i)$ 表示 i 右侧小于 a_i 的数有多少个。

1.5 BSGS

求解方程

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

要求 $a \perp p$。

令 $x = us - v$，其中 $s = \lceil \sqrt{p} \rceil$，$0 \leqslant u, v \leqslant s$。移项得到

$$a^{us} \equiv ba^v \pmod p$$

1.6 高斯消元

```
int rank = 0;
double det = 1;
for (int i = 1; i <= n; ++i)
{
    int cur = 0;
    for (int j = 1; j <= n; ++j)
        if (a[i][j] & !vis[j])
        {
            cur = j;
            break;
        }
    if (cur == 0)
    {
        det = 0;
        break;
    }
    else
    {
        ++rank;
        det *= a[i][cur];
    }
    for (int j = 1; j <= n; ++j)
        if (j != cur)
```

```
            a[i][j] /= a[i][cur];
    b[i] /= a[i][cur];
    a[i][cur] = 1;
    for (int k = 1; k <= n; ++k)
        if (k != i)
        {
            for (int j = 1; j <= n; ++j)
                if (j != cur)
                    a[k][j] -= a[k][cur] * a[i][j];
            b[k] -= a[k][cur] * b[i];
            a[k][cur] = 0;
        }
}
```

1.7 Miller-Rabin 算法

要求 n 是否为质数。

多次选取底数 a。若某个 a 不满足费马小定理或二次探测定理（的逆否命题），则 n 为合数。全部满足则 n 可能为质数。

代码见下一小节。

1.8 Pollard-Rho 算法

不是很懂。

下面的代码是 Miller-Rabin 和 Pollard-Rho，还顺便附赠了快速加，快速乘，快速幂和 gcd。

效率：Luogu P4718 11/14。

```
inline long long add(long long x, long long y, long long m)
{
    return x + y < m ? x + y : x + y - m;
}

inline long long mul(long long x, long long n, long long m)
{
    long long res = 0;
    for (; n; n >>= 1, x = add(x, x, m))
        if (n & 1)
            res = add(res, x, m);
    return res;
}

inline long long pow(long long x, long long n, long long m)
```

```
{
    long long res = 1;
    for (; n; n >>= 1, x = mul(x, x, m))
        if (n & 1)
            res = mul(res, x, m);
    return res;
}

inline long long gcd(long long a, long long b)
{
    for (; b %= a ^= b ^= a ^= b; );
    return a;
}

inline bool IsNotPrime(long long n, int a)
{
    long long u = n - 1;
    int t = 0;
    for (; (u & 1) == 0; u >>= 1, ++t);
    long long x = pow(a, u, n);
    for (int i = 1; i <= t; ++i)
    {
        long long y = mul(x, x, n);
        if (y == 1)
            if (x == n - 1)
                return false;
            else if (x != 1)
                return true;
        x = y;
    }
    return x != 1;
}

inline bool MillerRabin(long long n)
{
    const int test[]={2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
    for (int i = 0; i < 12; ++i)
        if (n == test[i])
            return true;
    if (n <= 40)
        return false;
```

```
    for (int i = 0; i < 12; ++i)
        if (IsNotPrime(n, test[i]))
            return false;
    return true;
}

inline long long Rho(long long n)
{
    long long c = rand() % (n - 3) + 1;
    long long s = rand() % n, t = add(mul(s, s, n), c, n);
    for (int lim = 1; s != t; lim = std::min(128, lim << 1))
    {
        long long val = 1;
        for (int i = 1; i <= lim; ++i)
        {
            long long tmp = mul(val, std::abs(t - s), n);
            if (tmp)
            {
                val = tmp;
                s = add(mul(s, s, n), c, n);
                t = add(mul(t, t, n), c, n);
                t = add(mul(t, t, n), c, n);
            }
            else
                break;
        }
        long long d = gcd(val, n);
        if (d > 1)
            return d;
    }
    return n;
}
```

1.9 杜教筛

令 $f(n)$ 为一积性函数，现要计算其前缀和 $S(n)$。
另取一积性函数 $g(n)$，使得 $(f \times g)(n)$ 能够快速求前缀和，有

$$S(n) = \sum_{k=1}^{n} (f \times g)(k) - \sum_{k=2}^{n} g(k) f\left(\left\lfloor \frac{n}{k} \right\rfloor\right)$$

使用线性筛预处理出前 $n^{2/3}$ 项，则总时间复杂度取到最小值。

1.10 min_25 筛

minp(n) 表示 n 的最小质因子。p_k 表示第 k 小的质数。
第一部分。定义

$$F(n,k)=\sum_{i=1}^{n}[i\in\mathbb{P}\vee \mathrm{minp}(i)>p_k]f(i)$$

要求

$$F(n,+\infty)=\sum_{i=1}^{n}[i\in\mathbb{P}]f(i)$$

$p_k^2\leqslant n$ 时

$$F(n,k)=F(n,k-1)-f(p_k)\left(F\left(\left\lfloor\frac{n}{p_k}\right\rfloor,k-1\right)-\sum_{i=1}^{k-1}f(p_i)\right)$$

$p_k^2>n$ 时

$$F(n,k)=F(n,k-1)$$

话说回来，代码块的注释不能写中文？不过懒得调了，反正应该能看懂。

```
for (int i = 2; i <= sq; ++i)
{
    if (f[i] == 0)
    {
        g[++g[0]] = i;
        // fill in the blank below with f(i)
        fsum[g[0]] = fsum[g[0] - 1] + /* */;
    }
    for (int j = 1; j <= g[0] && i * g[j] <= sq; ++j)
    {
        f[i * g[j]] = 1;
        if (i % g[j] == 0)
            break;
    }
}
for (int l = 1, r; l <= n; r = n / (n / l), l = r + 1)
{
    w[++m] = n / l;
    // fill in the blank below with \sum_{t = 2}^{w[m]} f(t)
    F[m] = /* */;
    if (w[m] <= sq)
        id1[w[m]] = m;
```

```
        else
            id2[n / w[m]] = m;
}
for (int j = 1; j <= g[0]; ++j)
    for (int i = 1; i <= m && w[i] >= g[j] * g[j]; ++i)
    {
        int id;
        if (w[i] / g[j] <= sq)
            id = id1[w[i] / g[j]];
        else
            id = id2[n / (w[i] / g[j])];
        // fill in the blank below with f(g[j])
        F[i] -= /* */ * (F[id] - fsum[j - 1]);
    }
```

第二部分。定义

$$S(n,k)=\sum_{i=1}^{n}[\mathrm{minp}(i)\geqslant p_k]f(i)$$

要求

$$S(n,1)+f(1)$$

有

$$S(n,k)=F(n,+\infty)-\sum_{i=1}^{k-1}f(p_i)+\sum_{i\geqslant k\wedge p_i^2\leqslant n}\sum_{j\geqslant 1\wedge p_i^{j+1}\leqslant n}\left(f\left(p_i^j\right)S\left(\left\lfloor\frac{n}{p_i^j}\right\rfloor,i+1\right)+f\left(p_i^{j+1}\right)\right)$$

```
int S(int x, int y)
{
    if (x <= 1 || g[y] > x)
        return 0;
    else
    {
        int id = x <= sq ? id1[x] : id2[n / x];
        int res = F[id] - fsum[y - 1];
        for (int i = y; i <= g[0] && g[i] * g[i] <= x; ++i)
            for (int p = g[i]; p * g[i] <= x; p *= g[i])
                // fill in the blanks below with f(p) and f(p * g[i]),
                    correspondingly
                res += /* */ * S(x / p, i + 1) + /* */;
        return res;
```

```
    }
}
```

2 数论：特殊无穷数列

2.1 组合数

递推式

$$\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m}$$

$$\binom{n}{0} = \binom{n}{n} = 1$$

通项

$$\binom{n}{m} = \frac{n^{\underline{m}}}{m!}$$

2.2 第二类斯特林数

递推式

$$\left\{ {n \atop m} \right\} = \left\{ {n-1 \atop m-1} \right\} + m \left\{ {n-1 \atop m} \right\}$$

$$\left\{ {n \atop 0} \right\} = [n=0]$$

2.3 第一类斯特林数

递推式

$$\begin{bmatrix} n \\ m \end{bmatrix} = \begin{bmatrix} n-1 \\ m-1 \end{bmatrix} + (n-1) \begin{bmatrix} n-1 \\ m \end{bmatrix}$$

$$\begin{bmatrix} n \\ 0 \end{bmatrix} = [n=0]$$

2.4 斐波那契数

递推式

$$F_n = F_{n-1} + F_{n-2}$$

$$F_0 = 0 \quad F_1 = 1$$

一些性质

$$F_{n+m} = F_{n+1}F_m + F_nF_{m-1}$$

$$\sum_{k=0}^{n} F_k^2 = F_nF_{n+1}$$

$$F_{(n,m)} = (F_n, F_m)$$

$$\pi(p^k) = p^{k-1}\pi(p)$$

$$\pi(nm) = [\pi(n), \pi(m)] \quad (n \perp m)$$

$$\pi(2) = 3 \quad \pi(5) = 20$$

$$p \equiv \pm 1 \pmod{10} \Rightarrow \pi(p) \mid p-1$$

$$p \equiv \pm 2 \pmod{5} \Rightarrow \pi(p) \mid 2p+2$$

2.5 卡特兰数

递推式

$$C_{n+1} = \sum_{k=0}^{n} C_kC_{n-k}$$

$$C_0 = 1$$

通项

$$C_n = \binom{2n}{n} - \binom{2n}{n-1} = \frac{1}{n+1}\binom{2n}{n}$$

OGF

$$C(x) = \frac{1-\sqrt{1-4x}}{2x}$$

2.6 贝尔数

递推式

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$$

$$B_0 = 1$$

通项

$$B_n = \sum_{k=0}^{n} \begin{Bmatrix} n \\ k \end{Bmatrix}$$

EGF

$$B(x) = \exp(e^x - 1)$$

其他性质

$$B_{n+p} \equiv B_n + B_{n+1} \pmod p$$

2.7 伯努利数

递推式

$$\sum_{k=0}^{n} \binom{n+1}{k} B_k = 0$$

$$B_0 = 1$$

EGF

$$B(z) = \frac{z}{e^z - 1}$$

其他性质

$$\sum_{k=0}^{n} k^m = \frac{1}{m+1} \sum_{k=0}^{m} \binom{m+1}{k} B_k (n+1)^{m-k+1}$$

2.8 默慈金数

递推式

$$M_{n+1} = M_n + \sum_{k=0}^{n-1} M_k M_{n-k-1} = \frac{(2n+3)M_n + 3nM_{n-1}}{n+3}$$

$$M_0 = 0 \quad M_1 = 1$$

通项

$$M_n = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{2k} C_k$$

2.9 错排数

递推式

$$D_n = (n-1)(D_{n-1} + D_{n-2})$$

$$D_1 = 0 \quad D_2 = 1$$

通项

$$D_n = n! \sum_{k=2}^{n} \frac{(-1)^k}{k!} = \left\lfloor \frac{n!}{e} + \frac{1}{2} \right\rfloor$$

EGF

$$D(x) = \frac{1}{\exp(x + \ln(1-x))}$$

2.10 分拆数

OGF

$$F(x) = \prod_{k \geqslant 1} \frac{1}{1-x^k}$$

$$\frac{1}{F(x)} = \sum_{k=-\infty}^{+\infty} (-1)^k x^{k(3k-1)/2}$$

3 多项式

3.1 FFT

都 2022 年了谁还写 FFT 啊（暴论

```
inline void FFT(std::complex<double> F[], int N, int tp)
{
    for (int i = 0; i < N; ++i)
        if (i < (rev[i] = (rev[i >> 1] >> 1) | ((i & 1) ? (N >> 1) : 0))
            )
            std::swap(F[i], F[rev[i]]);
    for (int len = 2, p = 1; len <= N; len <<= 1, p <<= 1)
    {
        std::complex<double> unit = {cos(Pi / p), tp * sin(Pi / p)};
        for (int i = 0; i < N; i += len)
        {
            std::complex<double> cur = {1.0, 0.0};
            for (int j = i; j < i + p; ++j)
```

```
            {
                std::complex<double> x = F[j], y = F[j + p] * cur;
                F[j] = x + y;
                F[j + p] = x - y;
                cur = cur * unit;
            }
        }
    }
}
```

3.2 NTT

包含：

1. 多项式与常数的四则运算
2. 多项式快速幂，求导，不定积分，DFT/IDFT，求逆，ln 函数，exp 函数
3. 多项式相加，相减，卷积，点乘，转置乘
4. 多项式多点求值，快速插值
5. 波斯坦-茉莉算法
6. 常系数齐次线性递推第 n 项的求解

封装程度较大，可能会因为常数不够优秀而 TLE。

```
#include <cstdio>
#include <algorithm>
#include <set>
#define maxn 524288
#define maxlog 20
#define mod 998244353

int unit[2][24];
int inv[maxn], fac[maxn], fav[maxn];
int rev[maxn];

inline int add(int a, int b)
{
    return a + b < mod ? a + b : a + b - mod;
}

inline int sub(int a, int b)
```

```
{
    return a - b >= 0 ? a - b : a - b + mod;
}

inline int pow(int a, int n)
{
    int res = 1;
    for (; n; n >>= 1, a = 1LL * a * a % mod)
        if (n & 1)
            res = 1LL * res * a % mod;
    return res;
}

inline int C(int n, int m)
{
    if (m >= 0 && m <= n)
        return 1LL * fac[n] * fav[m] % mod * fav[n - m] % mod;
    else
        return 0;
}

inline void InitIntUnit()
{
    unit[0][23] = pow(3, 119);
    unit[1][23] = pow(332748118, 119);
    for(int i = 0; i < 2; ++i)
        for(int j = 22; j >= 0; --j)
            unit[i][j] = 1LL * unit[i][j + 1] * unit[i][j + 1] % mod;
}

inline void InitFac()
{
    inv[1] = 1;
    for (int i = 2; i < maxn; ++i)
        inv[i] = sub(0, 1LL * (mod / i) * inv[mod % i] % mod);
    fac[0] = fav[0] = 1;
    for (int i = 1; i < maxn; ++i)
    {
        fac[i] = 1LL * fac[i - 1] * i % mod;
        fav[i] = 1LL * fav[i - 1] * inv[i] % mod;
    }
```

```
}

inline void Derivative(int F[], int n, int G[])
{
    for (int i = 0; i < n; ++i)
        G[i] = 1LL * (i + 1) * F[i + 1] % mod;
    G[n] = 0;
}

inline void Integration(int F[], int n, int G[], int C)
{
    for (int i = n + 1; i > 0; --i)
        G[i] = 1LL * inv[i] * F[i - 1] % mod;
    G[0] = C;
}

inline void NTT(int F[], int N, int tp, int A[])
{
    for (int i = 0; i < N; ++i)
        A[i] = F[rev[i] = (rev[i >> 1] >> 1) | ((i & 1) ? (N >> 1) : 0)
            ];
    for (int k = 1, p = 1; p < N; ++k, p <<= 1)
        for (int i = 0; i < N; i += p << 1)
            for (int j = i, tmp = 1; j < i + p; ++j, tmp = 1LL * tmp *
                unit[tp][k] % mod)
            {
                int x = A[j], y = 1LL * A[j + p] * tmp % mod;
                A[j] = add(x, y);
                A[j + p] = sub(x, y);
            }
    if (tp == 1)
    {
        int v = pow(N, mod - 2);
        for (int i = 0; i < N; ++i)
            A[i] = 1LL * A[i] * v % mod;
    }
}

inline void Prod(int F[], int n, int G[], int m, int H[])
{
    static int A[maxn], B[maxn], C[maxn];
```

```
    int N = 1;
    for (; N < n + m + 1; N <<= 1);
    for (int i = n + 1; i < N; ++i)
        F[i] = 0;
    for (int i = m + 1; i < N; ++i)
        G[i] = 0;
    NTT(F, N, 0, A);
    NTT(G, N, 0, B);
    for (int i = 0; i < N; ++i)
        C[i] = 1LL * A[i] * B[i] % mod;
    NTT(C, N, 1, H);
    for (int i = n + m + 1; i < N; ++i)
        H[i] = 0;
}

inline void TransProd(int F[], int n, int G[], int m, int H[])
{
    static int tmp[maxn];
    for (int i = 0; i <= m; ++i)
        tmp[i] = G[m - i];
    Prod(F, n, tmp, m, H);
    for (int i = m; i <= n; ++i)
        H[i - m] = H[i];
    for (int i = n - m + 1; i <= n + m; ++i)
        H[i] = 0;
}

inline void Inv(int F[], int n, int G[])
{
    static int tmp[maxn], res[maxn];
    res[0] = pow(F[0], mod - 2);
    for (int i = 1; i - 1 < n; i <<= 1)
    {
        for (int j = 0; j < 2 * i; ++j)
            tmp[j] = F[j];
        Prod(tmp, 2 * i - 1, res, i - 1, tmp);
        for (int j = 0; j < 2 * i; ++j)
            tmp[j] = sub(0, tmp[j]);
        tmp[0] = add(tmp[0], 2);
        Prod(tmp, 2 * i - 1, res, i - 1, res);
    }
```

```
    for (int i = 0; i <= n; ++i)
        G[i] = res[i];
}

inline void Ln(int F[], int n, int G[])
{
    static int tmp0[maxn], tmp1[maxn];
    Derivative(F, n, tmp0);
    Inv(F, n, tmp1);
    Prod(tmp0, n - 1, tmp1, n, G);
    Integration(G, n - 1, G, 0);
}

inline void Exp(int F[], int n, int G[])
{
    static int tmp[maxn], res[maxn];
    res[0] = 1;
    for (int i = 1; i - 1 < 2 * n; i <<= 1)
    {
        for (int j = 0; j < i; ++j)
            tmp[j] = res[j];
        Ln(tmp, i - 1, tmp);
        for (int j = 0; j < i; ++j)
            tmp[j] = sub(F[j], tmp[j]);
        for (int j = i; j < 2 * i; ++j)
            tmp[j] = F[j];
        tmp[0] = add(tmp[0], 1);
        Prod(tmp, 2 * i - 1, res, i - 1, res);
    }
    for (int i = 0; i <= n; ++i)
        G[i] = res[i];
}

int _Q[maxlog][maxn], _B[maxlog][maxn];

void dfsM1(int A[], int cur, int dep, int l, int r)
{
    static int tmp0[maxn], tmp1[maxn], tmp2[maxn];
    if (l == r)
        _Q[dep][l] = sub(0, A[l]);
    else
```

```
    {
        int mid = (l + r) >> 1;
        dfsM1(A, cur << 1, dep + 1, l, mid);
        dfsM1(A, cur << 1 | 1, dep + 1, mid + 1, r);
        for (int i = l; i <= mid; ++i)
            tmp0[i - l + 1] = _Q[dep + 1][i];
        tmp0[0] = 1;
        for (int i = mid + 1; i <= r; ++i)
            tmp1[i - mid] = _Q[dep + 1][i];
        tmp1[0] = 1;
        Prod(tmp0, mid - l + 1, tmp1, r - mid, tmp2);
        for (int i = l; i <= r; ++i)
            _Q[dep][i] = tmp2[i - l + 1];
    }
}

void dfsM2(int B[], int cur, int dep, int l, int r)
{
    static int tmp0[maxn], tmp1[maxn], tmp2[maxn];
    if (l == r)
        B[l] = _B[dep][l];
    else
    {
        int mid = (l + r) >> 1;
        for (int i = l; i <= r; ++i)
            tmp0[i - l] = _B[dep][i];
        for (int i = l; i <= mid; ++i)
            tmp1[i - l + 1] = _Q[dep + 1][i];
        tmp1[0] = 1;
        TransProd(tmp0, r - l, tmp1, mid - l + 1, tmp2);
        for (int i = mid + 1; i <= r; ++i)
            _B[dep + 1][i] = tmp2[i - (mid + 1)];
        for (int i = mid + 1; i <= r; ++i)
            tmp1[i - mid] = _Q[dep + 1][i];
        tmp1[0] = 1;
        TransProd(tmp0, r - l, tmp1, r - mid, tmp2);
        for (int i = l; i <= mid; ++i)
            _B[dep + 1][i] = tmp2[i - l];
        dfsM2(B, cur << 1, dep + 1, l, mid);
        dfsM2(B, cur << 1 | 1, dep + 1, mid + 1, r);
    }
```

```cpp
}

inline void Multipoint(int F[], int A[], int n, int B[])
{
    static int tmp[maxn];
    dfsM1(A, 1, 0, 0, n);
    for (int i = 1; i <= n + 1; ++i)
        tmp[i] = _Q[0][i - 1];
    tmp[0] = 1;
    Inv(tmp, n + 1, tmp);
    TransProd(F, 2 * n + 1, tmp, n + 1, _B[0]);
    dfsM2(B, 1, 0, 0, n);
}

int _P[maxlog][maxn], _F[maxlog][maxn];

void dfsI1(int A[], int cur, int dep, int l, int r)
{
    static int tmp0[maxn], tmp1[maxn], tmp2[maxn];
    if (l == r)
        _P[dep][l] = sub(0, A[l]);
    else
    {
        int mid = (l + r) >> 1;
        dfsI1(A, cur << 1, dep + 1, l, mid);
        dfsI1(A, cur << 1 | 1, dep + 1, mid + 1, r);
        for (int i = l; i <= mid; ++i)
            tmp0[i - l] = _P[dep + 1][i];
        tmp0[mid - l + 1] = 1;
        for (int i = mid + 1; i <= r; ++i)
            tmp1[i - mid - 1] = _P[dep + 1][i];
        tmp1[r - mid] = 1;
        Prod(tmp0, mid - l + 1, tmp1, r - mid, tmp2);
        for (int i = l; i <= r; ++i)
            _P[dep][i] = tmp2[i - l];
    }
}

void dfsI2(int B[], int C[], int cur, int dep, int l, int r)
{
    static int tmp0[maxn], tmp1[maxn], tmp2[maxn], tmp3[maxn];
```

```
    if (l == r)
        _F[dep][l] = 1LL * B[l] * pow(C[l], mod - 2) % mod;
    else
    {
        int mid = (l + r) >> 1;
        dfsI2(B, C, cur << 1, dep + 1, l, mid);
        dfsI2(B, C, cur << 1 | 1, dep + 1, mid + 1, r);
        for (int i = l; i <= mid; ++i)
            tmp0[i - l] = _F[dep + 1][i];
        for (int i = mid + 1; i <= r; ++i)
            tmp1[i - mid - 1] = _P[dep + 1][i];
        tmp1[r - mid] = 1;
        Prod(tmp0, mid - l, tmp1, r - mid, tmp2);
        for (int i = mid + 1; i <= r; ++i)
            tmp0[i - mid - 1] = _F[dep + 1][i];
        for (int i = l; i <= mid; ++i)
            tmp1[i - l] = _P[dep + 1][i];
        tmp1[mid - l + 1] = 1;
        Prod(tmp0, r - mid - 1, tmp1, mid - l + 1, tmp3);
        for (int i = l; i <= r; ++i)
            _F[dep][i] = add(tmp2[i - l], tmp3[i - l]);
    }
}

inline int Unique(int A[], int B[], int n)
{
    std::set<int> S;
    S.clear();
    for (int i = 0; i <= n; ++i)
        if (!S.count(A[i]))
        {
            S.insert(A[i]);
            A[S.size() - 1] = A[i];
            B[S.size() - 1] = B[i];
        }
    return S.size() - 1;
}

inline void Interpolation(int A[], int B[], int n, int F[])
{
    static int tmp[maxn];
```

```
    n = Unique(A, B, n);
    dfsI1(A, 1, 0, 0, n);
    for (int i = 0; i <= n; ++i)
        tmp[i] = _P[0][i];
    tmp[n + 1] = 1;
    Derivative(tmp, n + 1, tmp);
    Multipoint(tmp, A, n, tmp);
    dfsI2(B, tmp, 1, 0, 0, n);
    for (int i = 0; i <= n; ++i)
        F[i] = _F[0][i];
}

inline int BostanMori(int F[], int G[], int k, int n)
{
    static int tmp0[maxn], tmp1[maxn];
    for (; n; n >>= 1)
    {
        for (int i = 0; i <= k; ++i)
            tmp0[i] = (i & 1) ? G[i] : sub(0, G[i]);
        Prod(F, k, tmp0, k, tmp1);
        for (int i = (n & 1), j = 0; i <= 2 * k; i += 2, ++j)
            F[j] = tmp1[i];
        Prod(G, k, tmp0, k, tmp1);
        for (int i = 0, j = 0; i <= 2 * k; i += 2, ++j)
            G[j] = tmp1[i];
    }
    return 1LL * F[0] * pow(G[0], mod - 2) % mod;
}

inline int Recurrence(int F[], int A[], int k, int n)
{
    static int tmp[maxn];
    F[0] = 1;
    for (int i = 1; i <= k; ++i)
        F[i] = sub(0, F[i]);
    Prod(F, k, A, k, tmp);
    tmp[k] = 0;
    return BostanMori(tmp, F, k, n);
}
```

3.3 FWT

FFT，但是是位运算。

按位或

$$\tilde{F}=\begin{cases}(\tilde{F_0},\tilde{F_1}+\tilde{F_0}) & (t>0)\\ F & (t=0)\end{cases}$$

$$\epsilon=(1,0,\cdots,0,0)$$

$$\left(F^{-1}\right)_0={F_0}^{-1}$$

$$\left(F^{-1}\right)_1=-F_1\vee {F_0}^{-1}\vee (F_0+F_1)^{-1}$$

按位与

$$\tilde{F}=\begin{cases}(\tilde{F_0}+\tilde{F_1},\tilde{F_1}) & (t>0)\\ F & (t=0)\end{cases}$$

$$\epsilon=(0,0,\cdots,0,1)$$

$$\left(F^{-1}\right)_1={F_1}^{-1}$$

$$\left(F^{-1}\right)_0=-F_0\wedge {F_1}^{-1}\wedge (F_0+F_1)^{-1}$$

按位异或

$$\tilde{F}=\begin{cases}(\tilde{F_0}+\tilde{F_1},\tilde{F_0}-\tilde{F_1}) & (t>0)\\ F & (t=0)\end{cases}$$

$$\epsilon=(1,0,\cdots,0,0)$$

$$\left(F^{-1}\right)_0=\frac{(F_0+F_1)^{-1}+(F_0-F_1)^{-1}}{2}$$

$$\left(F^{-1}\right)_1=\frac{(F_0+F_1)^{-1}-(F_0-F_1)^{-1}}{2}$$

4 其他数学

4.1 牛顿迭代

给定形式幂级数 $G(x)$。设

$$G(F_0(x)) \equiv 0 \pmod{x^n}$$

令

$$F(x) \equiv F_0(x) - \frac{G(F_0(x))}{G'(F_0(x))} \pmod{x^{2n}}$$

则

$$G(F(x)) \equiv 0 \pmod{x^{2n}}$$

4.2 组合对象符号化公式

无标号集合的 Sequence 构造

$$B(x) = \frac{1}{1 - A(x)}$$

无标号集合的 Multiset 构造（完全背包）

$$B(x) = \mathrm{Exp}(A(x)) = \exp \sum_{n \geqslant 1} \frac{A(x^n)}{n}$$

无标号集合的 Powerset 构造（01 背包）

$$B(x) = \overline{\mathrm{Exp}}(A(x)) = \exp \sum_{n \geqslant 1} \frac{(-1)^{n-1} A(x^n)}{n}$$

有标号集合的 Sequence 构造

$$B(x) = \frac{1}{1 - A(x)}$$

有标号集合的 Set 构造

$$B(x) = \exp A(x)$$

4.3 拉格朗日反演

设形式幂级数 $F(x), G(x)$ 互为复合逆，有

$$[x^n]G(x) = \frac{1}{n}[x^{n-1}] \left(\frac{x}{F(x)} \right)^n$$

设另有一个形式幂级数 $H(x)$，有

$$[x^n]H(G(x)) = \frac{1}{n}[x^{n-1}]H'(x)\left(\frac{x}{F(x)}\right)^n$$

4.4 Burnside 引理 & Polya 定理

有如下 Burnside 引理

$$|X/H| = \frac{1}{|H|}\sum_{g\in H}|X^g|$$

人话版本：集合 X 在群 $\langle H, \times\rangle$ 作用下的等价类数量等于 H 中所有元素作用在集合 X 上时的不动点数量的算术平均值。

如果 X 是使用 m 中颜色对一个大小为 n 的集合进行染色的方案数，H 是 n 阶置换群。有如下 Polya 定理

$$|X/H| = \frac{1}{|H|}\sum_{g\in H}m^{\#(g)}$$

其中 $\#(g)$ 表示 g 的轮换数。

5 数据结构

5.1 可持久化 WBLT

写的非常丑陋。估计常数巨大。

空间开销至少 20 倍 $n\log n$。

```
const double alpha = 0.293, beta = 0.707;

class WBLT
{
    public:
    struct node
    {
        node *lc, *rc;
        int min, max, size;
    } mempool[80 * maxn + 5], *memtop, *root[maxn + 5];

    WBLT()
    {
        memtop = mempool;
        root[0] = NewNode(NewNode(-2147483647), NewNode(2147483647));
    }

    inline node *NewNode(int val)
```

```
    {
        node *p = memtop++;
        p->lc = NULL;
        p->rc = NULL;
        p->min = val;
        p->max = val;
        p->size = 1;
        return p;
    }

    inline node *NewNode(node *p, node *q)
    {
        node *r = memtop++;
        r->lc = p;
        r->rc = q;
        r->min = p->min;
        r->max = q->max;
        r->size = p->size + q->size;
        return r;
    }

    std::pair<node *, node *> SplitByRank(node *p, int rank)
    {
        if (p == NULL)
            return std::make_pair((node *)NULL, (node *)NULL);
        else if (rank == 0)
            return std::make_pair((node *)NULL, p);
        else if (rank == p->size)
            return std::make_pair(p, (node *)NULL);
        else if (rank < p->lc->size)
        {
            std::pair<node *, node *> ans = SplitByRank(p->lc, rank);
            ans.second = Merge(ans.second, p->rc);
            return ans;
        }
        else
        {
            std::pair<node *, node *> ans = SplitByRank(p->rc, rank - p
               ->lc->size);
            ans.first = Merge(p->lc, ans.first);
            return ans;
```

```
        }
    }

    std::pair<node *, node *> SplitByVal(node *p, int val)
    {
        if (p == NULL)
            return std::make_pair((node *)NULL, (node *)NULL);
        else if (val < p->min)
            return std::make_pair((node *)NULL, p);
        else if (val >= p->max)
            return std::make_pair(p, (node *)NULL);
        else if (val < p->lc->max)
        {
            std::pair<node *, node *> ans = SplitByVal(p->lc, val);
            ans.second = Merge(ans.second, p->rc);
            return ans;
        }
        else
        {
            std::pair<node *, node *> ans = SplitByVal(p->rc, val);
            ans.first = Merge(p->lc, ans.first);
            return ans;
        }
    }

    node *Merge(node *x, node *y)
    {
        if (x == NULL)
            return y;
        else if (y == NULL)
            return x;
        else if (std::max(x->size, y->size) <= beta * (x->size + y->size
            ))
            return NewNode(x, y);
        else if (x->size > y->size)
        {
            if (x->lc->size > alpha * (x->size + y->size))
                return NewNode(x->lc, Merge(x->rc, y));
            else
                return Merge(Merge(x->lc, x->rc->lc), Merge(x->rc->rc, y
                    ));
```

```cpp
        }
        else
        {
            if (y->rc->size > alpha * (x->size + y->size))
                return NewNode(Merge(x, y->lc), y->rc);
            else
                return Merge(Merge(x, y->lc->lc), Merge(y->lc->rc, y->rc
                    ));
        }
    }

    inline void Insert(int id, int val, int cur)
    {
        std::pair<node *, node *> pair = SplitByVal(root[id], val);
        root[cur] = Merge(Merge(pair.first, NewNode(val)), pair.second);
    }

    inline void Erase(int id, int val, int cur)
    {
        std::pair<node *, node *> pair1 = SplitByVal(root[id], val);
        std::pair<node *, node *> pair2 = SplitByRank(pair1.first, pair1
            .first->size - 1);
        pair1.first = val == pair2.second->max ? pair2.first : Merge(
            pair2.first, pair2.second);
        root[cur] = Merge(pair1.first, pair1.second);
    }

    inline int QueryRank(int id, int val)
    {
        node *x = SplitByVal(root[id], val - 1).first;
        return x->size + 1;
    }

    inline int QueryKth(int id, int rank)
    {
        node *x = SplitByRank(root[id], rank).first;
        node *y = SplitByRank(x, rank - 1).second;
        return y->max;
    }
} wblt;
```

6 图论

6.1 矩阵树定理

设图 G 无自环。令图 G 的度数/出度/入度矩阵分别为 $D(G), D^{\text{out}}(G), D^{\text{in}}(G)$，邻接矩阵为 $A(G)$，生成树数量为 $t(G)$，以点 k 为根的根向/叶向树形图数量分别为 $t^{\text{root}}(G,k), t^{\text{leaf}}(G,k)$。

$$t(G) = \det\left[D(G) - A(G)\right]\begin{pmatrix}1,2,\dots,i-1,i+1,\dots,n\\1,2,\dots,i-1,i+1,\dots,n\end{pmatrix}$$

$$t^{\text{root}}(G,k) = \det\left[D^{\text{out}}(G) - A(G)\right]\begin{pmatrix}1,2,\dots,k-1,k+1,\dots,n\\1,2,\dots,k-1,k+1,\dots,n\end{pmatrix}$$

$$t^{\text{leaf}}(G,k) = \det\left[D^{\text{in}}(G) - A(G)\right]\begin{pmatrix}1,2,\dots,k-1,k+1,\dots,n\\1,2,\dots,k-1,k+1,\dots,n\end{pmatrix}$$

6.2 最大流

```
int n, m, s, t, ec = 1, maxHeight, curHeight;
int des[2 * MAXM + 5], suc[2 * MAXM + 5], cap[2 * MAXM + 5], last[MAXN +
    5];
int que[MAXN + 5], height[MAXN + 5], cnt[MAXN + 5], extra[MAXN + 5];
std::list<int> list[MAXN + 5];
std::list<int>::iterator pos[MAXN + 5];
std::vector<int> vector[MAXN + 5];

inline void connect(int u, int v, int c)
{
    des[++ec] = v;
    suc[ec] = last[u];
    cap[ec] = c;
    last[u] = ec;
}

inline void PushEdge(int u, int e)
{
    int flow = std::min(extra[u], cap[e]);
    extra[u] -= flow;
    extra[des[e]] += flow;
    cap[e] -= flow;
    cap[e ^ 1] += flow;
    if (extra[des[e]] > 0 && extra[des[e]] <= flow)
        vector[height[des[e]]].push_back(des[e]);
}
```

```

inline void BFS()
{
    for (int i = 1; i <= n; ++i)
        height[i] = n + 1;
    height[t] = 0;
    int head = 0, tail = 0;
    que[++tail] = t;
    for (; head < tail; )
    {
        int u = que[++head];
        for (int i = last[u]; i; i = suc[i])
            if (height[des[i]] == n + 1 && cap[i ^ 1] > 0)
            {
                height[des[i]] = height[u] + 1;
                ++cnt[height[des[i]]];
                que[++tail] = des[i];
            }
    }
    for (int i = 0; i <= n; ++i)
    {
        vector[i].clear();
        list[i].clear();
    }
    for (int i = 1; i <= n; ++i)
        if (height[i] < n + 1)
        {
            pos[i] = list[height[i]].insert(list[height[i]].begin(), i);
            if (extra[i] > 0) vector[height[i]].push_back(i);
        }
    maxHeight = height[que[tail - 1]];
    curHeight = maxHeight;
}

inline void Push(int u)
{
    int nextHeight = n + 1;
    for (int i = last[u]; i; i = suc[i])
        if (cap[i] > 0)
        {
            if (height[des[i]] + 1 == height[u])
```

```
            {
                PushEdge(u, i);
                if (extra[u] == 0) return;
            }
            else
                nextHeight = std::min(nextHeight, height[des[i]] + 1);
        }
    if (cnt[height[u]] == 1)
    {
        for (; maxHeight >= height[u]; list[maxHeight].clear(), --maxHeight)
            for (int v : list[maxHeight])
            {
                --cnt[height[v]];
                height[v] = n + 1;
            }
    }
    else
    {
        --cnt[height[u]];
        list[height[u]].erase(pos[u]);
        height[u] = nextHeight;
        if (height[u] == n + 1) return;
        ++cnt[height[u]];
        pos[u] = list[height[u]].insert(list[height[u]].begin(), u);
        curHeight = height[u];
        maxHeight = std::max(maxHeight, curHeight);
        vector[height[u]].push_back(u);
    }
}

inline int HLPP()
{
    // height[s] = n;
    extra[s] = INF;
    extra[t] = -INF;
    for (int i = last[s]; i; i = suc[i])
        PushEdge(s, i);
    BFS();
    for (; curHeight >= 0; )
        if (vector[curHeight].empty())
```

```
            --curHeight;
        else
        {
            int u = vector[curHeight].back();
            vector[curHeight].pop_back();
            Push(u);
        }
    return extra[t] + INF;
}
```

6.3 费用流

```
inline bool spfa()
{
    for (int i = 1; i <= n; ++i)
        dist[i] = 2E9;
    dist[s] = 0;
    flow[s] = 2E9;
    ex[h[1] = s] = true;
    for (int head = 0, tail = 1; head < tail; )
    {
        int x = h[++head];
        ex[x] = false;
        for (int i = last[x]; i >= 0; i = suc[i])
            if (cap[i] > 0 && dist[des[i]] > dist[x] + cost[i])
            {
                dist[des[i]] = dist[x] + cost[i];
                flow[des[i]] = min(flow[x], e[i].cap);
                pre[des[i]] = i;
                if (!ex[des[i]])
                    ex[h[++tail] = des[i]] = true;
            }
    }
    return dist[t] < 2E9;
}

inline void addflow()
{
    for (int cur = t; cur != s; cur = des[pre[cur] ^ 1])
    {
        cap[pre[cur]] -= flow[t];
```

```
        cap[pre[cur] ^ 1] += flow[t];
    }
    maxflow += flow[t];
    mincost += dist[t] * flow[t];
}
```

6.4 带下界可行流

令

$$\mathrm{cap}(e) = \mathrm{upper}(e) - \mathrm{lower}(e)$$

$$w(u) = \sum_{(v,u)\in E} \mathrm{lower}(v,u) - \sum_{(u,v)\in E} \mathrm{lower}(u,v)$$

如果 $w(u) > 0$，连附加边 $(S, u, w(u))$；如果 $w(u) < 0$，连附加边 $(u, T, -w(u))$。
如果这些附加边没有满载则无解。

6.5 带下界最大流

在可行流的基础上连边 $(T, S, +\infty)$。

6.6 六元组模型

连边 $(S, i, a_i), (i, T, b_i), (S, x_j, D_j), (S, y_j, E_j), (x_j, T, F_j), (y_j, T, G_j), (x_j, y_j, P_j), (y_j, x_j, Q_j)$
满足

$$\begin{cases} D_j + E_j & = (c_{1,1})_j \\ F_j + G_j & = (c_{2,2})_j \\ D_j + G_j + Q_j & = (c_{1,2})_j \\ E_j + F_j + P_j & = (c_{2,1})_j \end{cases}$$

D_j 和 F_j，E_j 和 G_j 可以同时加上一个数调整成非负数。

$$P_j = Q_j = \frac{(c_{1,2})_j + (c_{2,1})_j - (c_{1,1})_j - (c_{2,2})_j}{2}$$

这个东西非负则可做。

7 字符串

7.1 KMP

```
for (int i = 2, p = 0; i <= len2; ++i)
{
    for (; p > 0 && s2[p + 1] != s2[i]; p = kmp[p]);
    if (s2[p + 1] == s2[i])
        ++p;
    kmp[i] = p;
}
for (int i = 1, p = 0; i <= len1; ++i)
{
    for (; p > 0 && s2[p + 1] != s1[i]; p = kmp[p]);
    if (s2[p + 1] == s1[i])
        ++p;
    if (p == len2)
        p = kmp[p];
}
```

7.2 AC 自动机

```
class AC
{
    public:
    int size;
    int ch[maxn + 5][26], fail[maxn + 5];

    inline void init()
    {
        size = 0;
    }

    inline int newNode()
    {
        int u = ++size;
        for (int i = 0; i < 26; ++i)
            ch[u][i] = 0;
        fail[u] = 0;
        return u;
    }

    inline void insert(char s[])
    {
        int len = strlen(s + 1);
```

```
        for (int i = 1, u = 0; i <= len; ++i, u = ch[u][s[i] - 'a'])
            if (ch[u][s[i] - 'a'] == 0)
                ch[u][s[i] - 'a'] = newNode();
    }

    inline void prework()
    {
        std::queue<int> que;
        for (; !que.empty(); que.pop());
        for (int i = 0; i < 26; ++i)
            if (ch[0][i])
                que.push(ch[0][i]);
        for (; !que.empty(); )
        {
            int x = que.front();
            que.pop();
            for (int c = 0; c < 26; ++c)
                if (ch[x][c] > 0)
                {
                    fail[ch[x][c]] = ch[fail[x]][c];
                    que.push(ch[x][c]);
                }
                else
                    ch[x][c] = ch[fail[x]][c];
        }
    }
};
```

7.3 倍增 SA

```
for (int i = 1; i <= n; ++i)
    ++tnk[fst[i] = a[i]];
for (int i = 1; i <= max; ++i)
    tnk[i] += tnk[i - 1];
for (int i = n; i >= 1; --i)
    sa[tnk[fst[i]]--] = i;
for (int k = 1; ; k <<= 1)
{
    int cnt = 0;
    for (int i = n - k + 1; i <= n; ++i)
        snd[++cnt] = i;
```

```
    for (int i = 1; i <= n; ++i)
        if (sa[i] > k)
            snd[++cnt] = sa[i] - k;
    for (int i = 1; i <= max; ++i)
        tnk[i] = 0;
    for (int i = 1; i <= n; ++i)
        ++tnk[fst[i]];
    for (int i = 1; i <= max; ++i)
        tnk[i] += tnk[i - 1];
    for (int i = n; i >= 1; --i)
        sa[tnk[fst[snd[i]]]--] = snd[i];
    for (int i = 1; i <= n; ++i)
        snd[i] = fst[i];
    fst[sa[1]] = cnt = 1;
    for (int i = 2; i <= n; ++i)
        fst[sa[i]] = snd[sa[i]] == snd[sa[i - 1]] && snd[sa[i] + k] == snd[sa[i - 1] + k] ? cnt : ++cnt;
    if (cnt == n)
        break;
    else
        max = cnt;
}
for (int i = 1; i <= n; ++i)
    rnk[sa[i]] = i;
for (int i = 1, j = 0; i <= n; ++i)
    if (rnk[i] > 1)
    {
        j && (--j);
        int pos = sa[rnk[i] - 1];
        for (; i + j <= n && pos + j <= n && a[i + j] == a[pos + j]; ++j);
        het[rnk[i]][0] = j;
    }
for (int j = 1, p = 2; p <= n; ++j, p <<= 1)
    for (int i = 1; i + p - 1 <= n; ++i)
        het[i][j] = std::min(het[i][j - 1], het[i + (p >> 1)][j - 1]);
for (int i = 2; i <= n; ++i)
    lg[i] = lg[i >> 1] + 1;
```

注意事项：

1. sa[i] 的含义是排名为 i 的后缀的位置

2. rnk[i] 的含义是位置为 i 的后缀的排名
3. het[i] 的含义是 LCP($sa[i], sa[i-1]$)
4. rnk 数组需要开到 2 倍大小

7.4 SAM

```
class SAM
{
    public:
    int size, last;
    int ch[2 * maxn + 5][26], fail[2 * maxn + 5], len[2 * maxn + 5];

    inline void init()
    {
        size = last = 1;
        for (int i = 0; i < 26; ++i)
            ch[1][i] = 1;
        fail[1] = len[1] = 0;
    }

    inline int newNode()
    {
        int u = ++size;
        for (int i = 0; i < 26; ++i)
            ch[u][i] = 1;
        fail[u] = 1, len[u] = 0;
        return u;
    }

    inline int append(int c)
    {
        int cur = newNode(), u = last;
        len[cur] = len[u] + 1;
        for (; u > 0 && ch[u][c] == 1; ch[u][c] = cur, u = fail[u]);
        if (u == 0)
            fail[cur] = 1;
        else
        {
            int v = ch[u][c];
            if (len[v] == len[u] + 1)
```

```
                fail[cur] = v;
            else
            {
                int clone = newNode();
                for (int i = 0; i < 26; ++i)
                    ch[clone][i] = ch[v][i];
                fail[clone] = fail[v];
                len[clone] = len[u] + 1;
                fail[v] = fail[cur] = clone;
                for (; u > 0 && ch[u][c] == v; ch[u][c] = clone, u =
                    fail[u]);
            }
        }
        return last = cur;
    }
};
```

7.5 manacher

```
for (int i = n; i >= 0; --i)
{
    S[i << 1] = S[i];
    S[i << 1 | 1] = '#';
}
int m = n << 1 | 1;
for (int i = 1, maxRight = 0, mid = 0; i <= m; ++i)
{
    ans[i] = std::max(std::min(ans[(mid << 1) - i], maxRight - i + 1),
        1);
    for (; i - ans[i] >= 1 && i + ans[i] <= m && S[i - ans[i]] == S[i +
        ans[i]]; ++ans[i]);
    if (i + ans[i] - 1 > maxRight)
    {
        maxRight = i + ans[i] - 1;
        mid = i;
    }
}
```