widsnoy's template

hdu-t05: widsnoy, WQhuanm, xu826281112

1. 数论

1.1. 取模还原分数

1.2. 原根

- 阶：$\mathrm{ord}_m(a)$ 是最小的正整数 n 使 $a^n \equiv 1 (\mathrm{mod}\, m)$
- 原根：若 g 满足 $(g,m)=1$ 且 $\mathrm{ord}_m(g)=\varphi(m)$ 则 g 是 m 的原根。若 m 是质数，有 $g^i \bmod m, 0<i<m$ 的取值各不相同。

原根的应用：m 是质数时，若求$a_k=\sum_{i*j \bmod m=k} f_i * g_j$ 可以通过原根转化为卷积形式(要求 0 处无取值)。具体而言，$[1,m-1]$ 可以映射到 $g^{[1,m-1]}$，原式变为 $a_{g^k}=\sum_{g^{i+j \bmod (m-1)}=g^k} f_{g^i} * g_{g^j}$，令 $f_i=f_{g^i}$ 则 $a_k=\sum_{(i+j) \bmod (m-1)=k} f_i * g_j$

```cpp
int q[10005];
int getG(int n) {
    int i, j, t = 0;
    for (i = 2; (ll)(i * i) < n - 1; i++) {
        if ((n - 1) % i == 0) q[t++] = i, q[t++] = (n - 1) / i;
    }
    for (i = 2; ;i++) {
        for (j = 0; j < t;  j++) if (fpow(i, q[j], n) == 1) break;
        if (j == t) return i;
    }
    return -1;
}

vector<int> fpow(int kth) {
    if (kth == 0) return e;
    auto r = fpow(kth - 1);
    r = multiply(r, r);
    for (int i = p - 1; i < r.size(); i++) r[i % (p - 1)] = (r[i % (p - 1)] + r[i]) % mod;
    r.resize(p - 1);
    if (kk[kth] == '1') {
        r = multiply(r, e);
        for (int i = p - 1; i < r.size(); i++) r[i % (p - 1)] = (r[i % (p - 1)] + r[i]) % mod;
        r.resize(p - 1);
    }
    return r;
}
void MAIN() {
    g = getG(p);
    int tmp = 1;
    for (int i = 1; i < p; i++) {
        tmp = tmp * 1ll * g % p;
        mp[tmp] = i % (p - 1);
    }
    e.resize(p - 1);
    for (int i = 0; i < p - 1; i++) e[i] = 0;
```

```
    for (int i = 0; i < p; i++) {
        for (int j = 0; j <= i; j++) {
            if (binom[i][j] == 0) continue;
            e[mp[binom[i][j]]]++;
        }
    }
}
```

1.3. 解不定方程

给出 a,b,c,x1,x2,y1,y2，求满足 ax+by+c=0，且 x∈[x1,x2],y∈[y1,y2]的整数解有多少对?

输入格式

第一行包含 7 个整数，a,b,c,x1,x2,y1,y2，整数间用空格隔开。

a,b,c,x1,x2,y1,y2 的绝对值不超过10^8。

```
#define y1 miku

ll a, b, c, x1, x2, y1, y2;
ll exgcd(ll a, ll b, ll &x, ll &y) {
    if (b) {
        ll d = exgcd(b, a % b, y, x);
        return y -= a / b * x, d;
    } return x = 1, y = 0, a;
}

pll get_up(ll a, ll b, ll x1, ll x2) {
    //x2>=ax+b>=x1
    if (a == 0) return (b >= x1 && b <= x2) ? (pll){-1e18, 1e18} : (pll)
{1, 0};
    ll L, R;
    ll l = (x1 - b) / a - 3;
    for (L = l; L * a + b < x1; L++);
    ll r = (x2 - b) / a + 3;
    for (R = r; R * a + b > x2; R--);
    return {L, R};
}
pll get_dn(ll a, ll b, ll x1, ll x2) {
    //x2>=ax+b>=x1
    if (a == 0) return (b >= x1 && b <= x2) ? (pll){-1e18, 1e18} : (pll)
{1, 0};
    ll L, R;
    ll l = (x2 - b) / a - 3;
    for (L = l; L * a + b > x2; L++);
    ll r = (x1 - b) / a + 3;
    for (R = r; R * a + b < x1; R--);
    return {L, R};
}

void MAIN() {
```

```
    cin >> a >> b >> c >> x1 >> x2 >> y1 >> y2;
    if (a == 0 && b == 0) return cout << (c == 0) * (y2 - y1 + 1) * (x2 - x1 + 1) << '\n', void();
    ll x, y, d = exgcd(a, b, x, y);
    c = -c;
    if (c % d != 0) return cout << "0\n", void();
    x *= c / d, y *= c / d;
    ll sx = b / d, sy = -a / d;
    //x + k * sx  y + k * sy
    // 0<= 3 - k <= 4 [-1,3] [0,4]
    auto A = (sx > 0 ? get_up(sx, x, x1, x2) : get_dn(sx, x, x1, x2));
    auto B = (sy > 0 ? get_up(sy, y, y1, y2) : get_dn(sy, y, y1, y2));
    A.fi = max(A.fi, B.fi), A.se = min(A.se, B.se);
    cout << max(0ll, A.se - A.fi + 1) << '\n';
}
```

1.4. 中国剩余定理

考虑合并两个同余方程

$$\begin{cases} x \equiv a_1 (\bmod m_1) \\ x \equiv a_2 (\bmod m_2) \end{cases}$$

改写为不定方程形式

$$\begin{cases} x + m_1 y = a_1 \\ x + m_2 y = a_2 \end{cases}$$

取解集公共部分 $x = a_1 - m_1 y_1 = a_2 - m_2 y_2$, 若$\gcd(m_1, m_2) | (a_1 - a_2)$ 有解，可以得到$x = k\operatorname{lcm}(m_1, m_2) + a_2 - m_2 y_2$ 化为同余方程的形式：$x \equiv a_2 - m_2 y_2 (\bmod \operatorname{lcm}(m_1, m_2))$

```
ll n, m, a;
ll exgcd(ll a, ll b, ll &x, ll &y) {
    if (b != 0) {
        ll g = exgcd(b, a % b, y, x);
        return y -= a / b * x, g;
    } return x = 1, y = 0, a;
}
ll getinv(ll a, ll mod) {
    ll x, y;
    exgcd(a, mod, x, y);
    x = (x % mod + mod) % mod;
    return x;
}
int get(ll x) {
    return x < 0 ? -1 : 1;
}
ll mul(ll a, ll b, ll mod) {
    ll res = 0;
```

```
19      if (a == 0 || b == 0) return 0;
20      ll f = get(a) * get(b);
21      a = abs(a), b = abs(b);
22      for (; b; b >>= 1, a = (a + a) % mod) if (b & 1) res = (res + a) %
    mod;
23      res *= f;
24      if (res < 0) res += mod;
25      return res;
26  }
27  // m 互质
28  // int main() {
29  //     cin >> n;
30  //     ll phi = 1;
31  //     for (int i = 1; i <= n; i++) {
32  //         cin >> m[i] >> a[i];
33  //         phi *= m[i];
34  //     }
35  //     ll ans = 0;
36  //     for (int i = 1; i <= n; i++) {
37  //         ll p = phi / m[i], q = getinv(p, m[i]);
38  //         ans += mul(p, mul(q, a[i], phi), phi);
39  //         ans %= phi;
40  //     }
41  //     cout << ans << '\n';
42  // }
43  int main() {
44      cin >> n;
45      cin >> m >> a;
46      for (int i = 2; i <= n; i++) {
47          ll nm, na;
48          cin >> nm >> na;
49          ll x, y;
50          ll g = exgcd(m, -nm, x, y), d = (na - a) / g, md = abs(nm / g);
51          if ((na - a) % g) return -1;
52          x = mul(x, d, md);
53          ll lc = abs(m / g);
54          lc *= nm;
55          a = (a + mul(m, x, lc)) % lc;
56          m = lc;
57      }
58      cout << a << '\n';
59  }
```

1.5. 卢卡斯定理

- p 为质数

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

- p 不为质数

其中 calc(n, x, p) 计算 $\frac{n!}{x^y} \bmod p$ 的结果，其中 y 是 $n!$ 含有 x 的个数

如果 p 是质数，利用 Wilson 定理 $(p-1)! \equiv -1 (\bmod p)$ 可以$O(\log P)$ 的计算 calc。其他情况可以通过预处理 $\frac{n!}{n以内所有p倍数的乘积}$ 达到同样的效果。

```cpp
ll exgcd(ll a, ll b, ll &x, ll &y) {
    if (b) {
        ll d = exgcd(b, a % b, y, x);
        return y -= a / b * x, d;
    } else return x = 1, y = 0, a;
}
int getinv(ll v, ll mod) {
    ll x, y;
    exgcd(v, mod, x, y);
    return (x % mod + mod) % mod;
}
ll fpow(ll a, ll b, ll p) {
    ll res = 1;
    for (; b; b >>= 1, a = a * 1ll * a % p) if (b & 1) res = res * 1ll * a % p;
    return res;
}
ll calc(ll n, ll x, ll p) {
    if (n == 0) return 1;
    ll s = 1;
    for (ll i = 1; i <= p; i++) if (i % x) s = s * i % p;
    s = fpow(s, n / p, p);
    for (ll i = n / p * p + 1; i <= n; i++) if (i % x) s = i % p * s % p;
    return calc(n / x, x, p) * 1ll * s % p;
}
int get(ll x) {
    return x < 0 ? -1 : 1;
}
ll mul(ll a, ll b, ll mod) {
    ll res = 0;
    if (a == 0 || b == 0) return 0;
    ll f = get(a) * get(b);
    a = abs(a), b = abs(b);
    for (; b; b >>= 1, a = (a + a) % mod) if (b & 1) res = (res + a) % mod;
    res *= f;
    if (res < 0) res += mod;
    return res;
}
ll sublucas(ll n, ll m, ll x, ll p) {
    ll cnt = 0;
    for (ll i = n; i; ) cnt += (i = i / x);
    for (ll i = m; i; ) cnt -= (i = i / x);
    for (ll i = n - m; i; ) cnt -= (i = i / x);
    return fpow(x, cnt, p) * calc(n, x, p) % p * getinv(calc(m, x, p), p) % p * getinv(calc(n - m, x, p), p) % p;
```

```
}
ll lucas(ll n, ll m, ll p) {
    int cnt = 0;
    ll a[21], mo[21];
    for (ll i = 2; i * i <= p; i++) if (p % i == 0) {
        mo[++cnt] = 1;
        while (p % i == 0) mo[cnt] *= i, p /= i;
        a[cnt] = sublucas(n, m, i, mo[cnt]);
    }
    if (p != 1) mo[++cnt] = p, a[cnt] = sublucas(n, m, p, mo[cnt]);
    ll phi = 1;
    for (int i = 1; i <= cnt; i++) phi *= mo[i];
    ll ans = 0;
    for (int i = 1; i <= cnt; i++) {
        ll p = phi / mo[i], q = getinv(p, mo[i]);
        ans += mul(p, mul(q, a[i], phi), phi);
        ans %= phi;
    }
    return ans;
}
```

1.6. BSGS

求解 $a^x \equiv n (\mathrm{mod}\, p), a, p$ 不一定互质

```
int fpow(int a, int b, int p) {
    int res = 1;
    for (; b; b >>= 1, a = a * 1ll * a % p) if (b & 1) res = res * 1ll * a % p;
    return res;
}
ll exgcd(ll a, ll b, ll &x, ll &y) {
    if (b == 0) return x = 1, y = 0, a;
    ll d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}
int inv(int a, int p) {
    ll x, y;
    ll g = exgcd(a, p, x, y);
    if (g != 1) return -1;
    return (x % p + p) % p;
}
int BSGS(int a, int b, int p) {
    if (p == 1) return 1;
    unordered_map<int, int> x;
    int m = sqrt(p + 0.5) + 1;
    int v = inv(fpow(a, m, p), p);
    int e = 1;
    for(int i = 1; i <= m; i++) {
        e = e * 1ll * a % p;
```

```
26          if(!x.count(e)) x[e] = i;
27      }
28      for(int i = 0; i <= m; i++) {
29          if(x.count(b)) return i * m + x[b];
30          b = b * 1ll * v % p;
31      }
32      return -1;
33  }
34  pii exBSGS(int a, int n, int p) {
35      int d, q = 0, sum = 1;
36      if (n == 1) return {0, gcd(a, p) == 1 ? BSGS(a, 1, p) : 0};
37      a %= p, n %= p;
38      while((d = gcd(a, p)) != 1) {
39          if(n % d) return {-1, -1};
40          q++; n /= d; p /= d;
41          sum = (sum * 1ll * a / d) % p;
42          if(sum == n) return {q, gcd(a, p) == 1 ? BSGS(a, 1, p) : 0};
43      }
44      int v = inv(sum, p);
45      n = n * 1ll * v % p;
46      int ans = BSGS(a, n, p);
47      if(ans == -1) return {-1, -1};
48      return {ans + q, BSGS(a, 1, p)};
49  }
```

1.7. 数论函数

1. $\varphi(n) = n\prod\left(1-\frac{1}{p}\right)$
2. $\mu(n) = \begin{cases} 1, n=1 \\ (-1)^{\text{质因子个数}}, n \text{ 无平方因子} \\ 0, n \text{ 有平方因子} \end{cases}$
3. $\mu * \mathrm{id} = \varphi, \mu * 1 = \varepsilon, \varphi * 1 = \mathrm{id}$

- 有一个表格，$a_{i,j} = \gcd(i,j)$, 支持某一列一行乘一个数，查询整个表格的和。

因为 $\gcd(n,m) = \sum_{i|n \wedge i|m} \varphi(i)$，对每个 $\varphi(i)$ 维护一个大小为 $\lfloor \frac{n}{i} \rfloor$ 的表格，初始值全是 $\varphi(i)$, (x,y) 对应 $(x*i, y*i)$。对大表格的修改可以转化为对小表格的修改，只需要对每行每列维护一个懒标记就行。

1.8. 莫比乌斯反演

1. 若 $f(n) = \sum_{d|n} g(d)$, 则 $g(n) = \sum_{d|n} \mu(\frac{n}{d}) f(d)$

$$\begin{aligned} \sum_{d|n} \mu\left(\frac{n}{d}\right) f(d) &= \sum_{d|n} \mu\left(\frac{n}{d}\right) \sum_{k|d} g(k) \\ &= \sum_{k|n} g(k) \sum_{d|\frac{n}{k}} \mu(d) \\ &= \sum_{k|n} g(k)[\frac{n}{k} = 1] = g(n) \end{aligned}$$

2. 若 $f(n)=\sum_{n|d}g(d)$, 则 $g(n)=\sum_{n|d}\mu\left(\frac{d}{n}\right)f(d)$
3. $d(nm)=\sum_{i|n}\sum_{j|m}[\gcd(i,j)=1]$

常见的一些推式子套路：

1. 证明是否积性函数，只需要观察是否满足 $f(p^i)f(q^j)=f(p^iq^j)$ 即可，用线性筛积性函数也是同理。
2. 形如 $\sum_{d|n}\mu(d)\sum_{k|\frac{n}{d}}\varphi(k)\left\lfloor\frac{n}{dk}\right\rfloor$ 的式子，这时候令 $T=dk$，枚举 T 就能得到 d,k 一个卷积的形式。如果是底数和指数，这时候不能线性筛，但是可以调和级数暴力算函数值。

1.9. 整除分块

1. 下取整

```
for (int i = 1, j; i <= min(n, m); i = j + 1) {
    j = min(n / (n / i), m / (m / i));
    // n / {i,...,j} = n / i
}
```

1. 上取整

$\left\lceil\frac{n}{i}\right\rceil=\left\lfloor\frac{n+i-1}{i}\right\rfloor=\left\lfloor\frac{n-1}{i}\right\rfloor+1$

1.10. 区间筛

- 求解一个区间内的素数

如果是合数那么一定不大于 $\sqrt{x}$ 的约数，使用这个范围内的数埃氏筛即可。

1.11. 杜教筛

1.12. Min25 筛

能在 $O\left(\frac{n^{\frac{3}{4}}}{\log(n)}\right)$ 时间求出 $F(n)=\sum_{i=1}^{n}f(i)$ 的值，要求积性函数能快速求出 $f\left(p^k\right)$ 处的点值。

- 定义 $R(i)$ 表示 i 的最小质因子

$$G(n,j)=\sum_{i=1}^{n}f(i)\left[i\in\text{prime}\vee R(i)>P_j\right]$$

考虑递推

$$G(n,j)=\begin{cases}G(n,j-1)\text{ IF }p_j\times p_j>n\\G(n,j-1)-f\left(p_j\right)\left(G\left(\frac{n}{p_j},j-1\right)-\sum_{i=1}^{j-1}f(p_i)\right)\text{ IF }p_j\times p_j\le n\end{cases}$$

根据整除分块，G 函数的第一维只用 $\sqrt{n}$ 种取值，将其存在 $w[]$ 中，且用 id1[] 和 id2[] 分别存数字对应的下标位置。因为最后只需要知道 $G(x,\text{pcnt})$ 所以第二维可以滚掉。

- 定义 $S(n,j)=\sum_{i=1}^{n}f(i)[R(i)\geq p_j]$

质数部分答案显然为 $G(n,\text{pcnt})-\sum_{i=1}^{j-1}f(p_i)$, 合数部分考虑提出最小的质因子 p^k，得到 $S(n,j)$ 的递推式

$$S(n,j)=G(n,\text{pcnt})-\sum_{i=1}^{j-1}f(p_i)+\sum_{i=j}^{\text{pcnt}}\sum_{k=1}^{p_i^{k+1}\leq n}f(p^k)S\left(\frac{n}{p^k},j+1\right)+f(p^{k+1})$$

递归边界是 $n=1\vee p_j>n, S(n,j)=0$

$\sum_{i=1}^{n}f(i)=S(n,1)+f(1)$

```cpp
#include <cstdio>
#include <cmath>

typedef long long ll;
const int N = 4e6 + 5, MOD = 1e9 + 7;
const ll i6 = 166666668, i2 = 500000004;
ll n, id1[N], id2[N], su1[N], su2[N], p[N], sqr, w[N], g[N], h[N];
int cnt, m;
bool vis[N];

ll add(ll a, ll b) {a %= MOD, b %= MOD; return (a + b >= MOD) ? a + b - MOD : a + b;}
ll mul(ll a, ll b) {a %= MOD, b %= MOD; return a * b % MOD;}
ll dec(ll a, ll b) {a %= MOD, b %= MOD; return ((a - b) % MOD + MOD) % MOD;}

void init(int m) {
  for (ll i = 2; i <= m; i++) {
    if (!vis[i]) p[++cnt] = i, su1[cnt] = add(su1[cnt - 1], i), su2[cnt] = add(su2[cnt - 1], mul(i, i));
    for (int j = 1; j <= cnt && i * p[j] <= m; j++) {
      vis[p[j] * i] = 1;
      if (i % p[j] == 0) break;
    }
  }
}

ll S(ll x, int y) {
  if (p[y] > x || x <= 1) return 0;
  int k = (x <= sqr) ? id1[x] : id2[n / x];
  ll res = dec(dec(g[k], h[k]), dec(su2[y - 1], su1[y - 1]));
  for (int i = y; i <= cnt && p[i] * p[i] <= x; i++) {
    ll pow1 = p[i], pow2 = p[i] * p[i];
    for (int e = 1; pow2 <= x; pow1 = pow2, pow2 *= p[i], e++) {
      ll tmp = mul(mul(pow1, dec(pow1, 1)), S(x / pow1, i + 1));
      tmp = add(tmp, mul(pow2, dec(pow2, 1)));
      res = add(res, tmp);
    }
  }
  return res;
```

```
}

int main() {
    scanf("%lld", &n);
  sqr = sqrt(n + 0.5) + 1;
  init(sqr);
  for (ll l = 1, r; l <= n; l = r + 1) {
        r = n / (n / l);
    w[++m] = n / l;
    g[m] = mul(w[m] % MOD, (w[m] + 1) % MOD);
    g[m] = mul(g[m], (2 * w[m] + 1) % MOD);
    g[m] = mul(g[m], i6);
        g[m] = dec(g[m], 1);
    h[m] = mul(w[m] % MOD, (w[m] + 1) % MOD);;
      h[m] = mul(h[m], i2);
    h[m] = dec(h[m], 1);
      (w[m] <= sqr) ? id1[w[m]] = m : id2[r] = m;
  }
  for (int j = 1; j <= cnt; j++)
    for (int i = 1; i <= m && p[j] * p[j] <= w[i]; i++) {
      int k = (w[i] / p[j] <= sqr) ? id1[w[i] / p[j]] : id2[n / (w[i] /
p[j])];
        g[i] = dec(g[i], mul(mul(p[j], p[j]), dec(g[k], su2[j - 1])));
      h[i] = dec(h[i], mul(p[j], dec(h[k], su1[j - 1])));
    }
  //printf("%lld\n", g[1] - h[1]);
  printf("%lld\n", add(S(n, 1), 1));
  return 0;
}
```

2. 动态规划

2.1. 缺 1 背包

背包和放物品先后顺序无关，所以可以直接倒退删物品。

```
memcpy(g, f, sizeof f);
for(int j = w[i]; j <= m; ++j)
    g[j] -= g[j - w[i]];
```

3. 图论

3.1. 找环

```
const int N = 5e5 + 5;
int n, m, col[N], pre[N], pre_edg[N];
vector<pii> G[N];
vector<vector<int>> resp, rese;
//point
```

```cpp
void get_cyc(int u, int v) {
    if (!resp.empty()) return;
    vector<int> cyc;
    cyc.push_back(v);
    while (true) {
        v = pre[v];
        if (v == 0) break;
        cyc.push_back(v);
        if (v == u) break;
    }
    reverse(cyc.begin(), cyc.end());
    resp.push_back(cyc);
}
// edge
void get_cyc(int u, int v, int id) {
    if (!rese.empty()) return;
    vector<int> cyc;
    cyc.push_back(id);
    while (true) {
        if (pre[v] == 0) break;
        cyc.push_back(pre_edg[v]);
        v = pre[v];
        if (v == u) break;
    }
    reverse(cyc.begin(), cyc.end());
    rese.push_back(cyc);
}
void dfs(int u, int edg) {
    col[u] = 1;
    for (auto [v, id] : G[u]) if (id != edg) {
        if (col[v] == 1) {
            get_cyc(v, u);
            get_cyc(v, u, id);
        } else if (col[v] == 0) {
            pre[v] = u;
            pre_edg[v] = id;
            dfs(v, id);
        }
    }
    col[u] = 2;
}
void MAIN() {
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {
        int u, v; cin >> u >> v;
        // G[u].push_back({v, i});
        // G[v].push_back({u, i});
    }
    for (int i = 1; i <= n; i++) if (!col[i]) dfs(i, -1);
}
```

3.2. 差分约束

3.2.1. SPFA 乱搞

```cpp
mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());

const int mod = 998244353;
const int N = 5e5 + 5;
const ll inf = 1e17;
int n, m, s, t, q[N], ql, qr;
int vis[N], fr[N];
ll dis[N];
vector<pii> G[N];
void MAIN() {
    cin >> n >> m >> s >> t;
    for (int i = 1; i <= m; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        G[u].push_back({v, w});
    }
    for (int i = 0; i <= n; i++) dis[i] = inf;
    dis[s] = 0; q[qr] = s; vis[s] = 1;
    while (ql <= qr) {
        if (rng() % (qr - ql + 1) == 0) sort(q + ql, q + qr + 1, [](int x, int y) {
            return dis[x] < dis[y];
        });
        int u = q[ql++];
        vis[u] = 0;
        for (auto [v, w] : G[u]) {
            if (dis[u] + w < dis[v]) {
                dis[v] = dis[u] + w;
                fr[v] = u;
                if (!vis[v]) {
                    if (ql > 0) q[--ql] = v;
                    else q[++qr] = v;
                    vis[v] = 1;
                }
            }
        }
    }
    if (dis[t] == inf) {
        cout << "-1\n";
        return;
    }
    cout << dis[t] << ' ';
    vector<pii> stk;
    while (t != s) {
        stk.push_back({fr[t], t});
        t = fr[t];
    }
    reverse(stk.begin(), stk.end());
    cout << stk.size() << '\n';
    for (auto [u, v] : stk) cout << u << ' ' << v << '\n';
}
```

3.3. 竞赛图

3.4. 连通分量

3.4.1. 有向图强连通分量

```cpp
const int N = 5e5 + 5;
int n, m, dfc, dfn[N], low[N], stk[N], top, idx[N], in_stk[N], scc_cnt;
vector<int> G[N];

void tarjan(int u) {
    low[u] = dfn[u] = ++dfc;
    stk[++top] = u;
    in_stk[u] = 1;
    for (int v : G[u]) {
        if (!dfn[v]) {
            tarjan(v);
            low[u] = min(low[u], low[v]);
        } else if (in_stk[v]) low[u] = min(dfn[v], low[u]);
    }
    if (low[u] == dfn[u]) {
        int x;
        scc_cnt++;
        do {
            x = stk[top--];
            idx[x] = scc_cnt;
            in_stk[x] = 0;
        } while (x != u);
    }
}

void MAIN() {
    for (int i = 1; i <= n; i++) low[i] = dfn[i] = idx[i] = in_stk[i] =
0;
    dfc = scc_cnt = top = 0;
    cin >> n >> m;
    for (int i = 1; i <= n; i++) if (!dfn[i]) tarjan(i);
}
```

3.4.2. 强连通分量(incremental)

edge[3] 保存了每条边的两个点在同一个强连通分量的时间。调用的时候右端点时间要大一位，因为可能有些边到最后也不能在一个强连通分量中。

```cpp
int n, m, Q, s[N];
vector<array<int, 4>> edge;
vector<int> G[N];
struct DSU {
    int fa[N], dep[N], top;
    pii stk[N];
    void init(int n) {
        top = 0;
```

```
 9          iota(fa, fa + n + 1, 0);
10          fill(dep, dep + n + 1, 1);
11      }
12      int find(int u) {
13          return u == fa[u] ? u : find(fa[u]);
14      }
15      void merge(int u, int v) {
16          u = find(u), v = find(v);
17          if (u == v) return;
18          if (dep[u] > dep[v]) swap(u, v);
19          stk[++top] = {u, (dep[u] == dep[v] ? v : -1)};
20          fa[u] = v;
21          dep[v] += (dep[u] == dep[v]);
22      }
23      void rev(int tim) {
24          while (tim < top) {
25              auto [u, v] = stk[top--];
26              fa[u] = u;
27              if (v != -1) dep[v]--;
28          }
29      }
30  } D;
31  int stk[N], top, dfc, dfn[N], low[N], in_stk[N];
32  void tarjan(int u) {
33      low[u] = dfn[u] = ++dfc;
34      stk[++top] = u;
35      in_stk[u] = 1;
36      for (int v : G[u]) {
37          if (!dfn[v]) {
38              tarjan(v);
39              low[u] = min(low[u], low[v]);
40          } else if (in_stk[v]) low[u] = min(dfn[v], low[u]);
41      }
42      if (low[u] == dfn[u]) {
43          int x;
44          do {
45              x = stk[top--];
46              D.merge(x, u);
47              in_stk[x] = 0;
48          } while (x != u);
49      }
50  }
51
52  void solve(int l, int r, int a, int b) {
53      if (l == r) {
54          for (int i = a; i <= b; i++) edge[i][3] = l;
55          return;
56      }
57      int mid = (l + r) >> 1;
58      vector<int> node;
59      for (int i = a; i <= b; i++) if (edge[i][0] <= mid) {
60          int u = D.find(edge[i][1]), v = D.find(edge[i][2]);
```

```
61          if (u != v) node.push_back(u), node.push_back(v),
    G[u].push_back(v);
62      }
63      int otp = D.top;
64      for (int x : node) if (!dfn[x]) tarjan(x);
65      vector<array<int, 4>> e1, e2;
66      for (int i = a; i <= b; i++) {
67          int u = D.find(edge[i][1]), v = D.find(edge[i][2]);
68          if (edge[i][0] > mid || u != v) e2.push_back(edge[i]);
69          else e1.push_back(edge[i]);
70      }
71      int s1 = e1.size(), s2 = e2.size();
72      for (int i = a; i < a + s1; i++) edge[i] = e1[i - a];
73      for (int i = a + s1; i <= b; i++) edge[i] = e2[i - a - s1];
74      dfc = 0;
75      for (int x : node) dfn[x] = low[x] = 0, vector<int>().swap(G[x]);
76      vector<int>().swap(node);
77      vector<array<int, 4>>().swap(e1);
78      vector<array<int, 4>>().swap(e2);
79      solve(mid + 1, r, a + s1, b);
80      D.rev(otp);
81      solve(l, mid, a, a + s1 - 1);
82  }
```

3.4.3. 割点和桥

```
1  int dfn[N], low[N], dfs_clock;
2  bool iscut[N], vis[N];
3  void dfs(int u, int fa) {
4      dfn[u] = low[u] = ++dfs_clock;
5      vis[u] = 1;
6      int child = 0;
7      for (int v : e[u]) {
8          if (v == fa) continue;
9          if (!dfn[v]) {
10             dfs(v, u);
11             low[u] = min(low[u], low[v]);
12             child++;
13             if (low[v] >= dfn[u]) iscut[u] = 1;
14         } else if (dfn[u] > dfn[v] && v != fa) low[u] = min(low[u],
    dfn[v]);
15         if (fa == 0 && child == 1) iscut[u] = 0;
16     }
17 }
```

3.4.4. 点双

```
1 #include <cstdio>
2 #include <vector>
3 using namespace std;
```

```cpp
const int N = 5e5 + 5, M = 2e6 + 5;
int n, m;

struct edge {
  int to, nt;
} e[M << 1];

int hd[N], tot = 1;

void add(int u, int v) { e[++tot] = (edge){v, hd[u]}, hd[u] = tot; }

void uadd(int u, int v) { add(u, v), add(v, u); }

int ans;
int dfn[N], low[N], bcc_cnt;
int sta[N], top, cnt;
bool cut[N];
vector<int> dcc[N];
int root;

void tarjan(int u) {
  dfn[u] = low[u] = ++bcc_cnt, sta[++top] = u;
  if (u == root && hd[u] == 0) {
    dcc[++cnt].push_back(u);
    return;
  }
  int f = 0;
  for (int i = hd[u]; i; i = e[i].nt) {
    int v = e[i].to;
    if (!dfn[v]) {
      tarjan(v);
      low[u] = min(low[u], low[v]);
      if (low[v] >= dfn[u]) {
        if (++f > 1 || u != root) cut[u] = true;
        cnt++;
        do dcc[cnt].push_back(sta[top--]);
        while (sta[top + 1] != v);
        dcc[cnt].push_back(u);
      }
    } else
      low[u] = min(low[u], dfn[v]);
  }
}

int main() {
  scanf("%d%d", &n, &m);
  int u, v;
  for (int i = 1; i <= m; i++) {
    scanf("%d%d", &u, &v);
    if (u != v) uadd(u, v);
  }
  for (int i = 1; i <= n; i++)
```

```
    if (!dfn[i]) root = i, tarjan(i);
  printf("%d\n", cnt);
  for (int i = 1; i <= cnt; i++) {
    printf("%llu ", dcc[i].size());
    for (int j = 0; j < dcc[i].size(); j++) printf("%d ", dcc[i][j]);
    printf("\n");
  }
  return 0;
}
```

3.4.5. 边双

```
#include <algorithm>
#include <cstdio>
#include <vector>

using namespace std;
const int N = 5e5 + 5, M = 2e6 + 5;
int n, m, ans;
int tot = 1, hd[N];

struct edge {
  int to, nt;
} e[M << 1];

void add(int u, int v) { e[++tot].to = v, e[tot].nt = hd[u], hd[u] =
tot; }

void uadd(int u, int v) { add(u, v), add(v, u); }

bool bz[M << 1];
int bcc_cnt, dfn[N], low[N], vis_bcc[N];
vector<vector<int>> bcc;

void tarjan(int x, int in) {
  dfn[x] = low[x] = ++bcc_cnt;
  for (int i = hd[x]; i; i = e[i].nt) {
    int v = e[i].to;
    if (dfn[v] == 0) {
      tarjan(v, i);
      if (dfn[x] < low[v]) bz[i] = bz[i ^ 1] = 1;
      low[x] = min(low[x], low[v]);
    } else if (i != (in ^ 1))
      low[x] = min(low[x], dfn[v]);
  }
}

void dfs(int x, int id) {
  vis_bcc[x] = id, bcc[id - 1].push_back(x);
  for (int i = hd[x]; i; i = e[i].nt) {
    int v = e[i].to;
```

```
    if (vis_bcc[v] || bz[i]) continue;
    dfs(v, id);
  }
}

int main() {
  scanf("%d%d", &n, &m);
  int u, v;
  for (int i = 1; i <= m; i++) {
    scanf("%d%d", &u, &v);
    if (u == v) continue;
    uadd(u, v);
  }
  for (int i = 1; i <= n; i++)
    if (dfn[i] == 0) tarjan(i, 0);
  for (int i = 1; i <= n; i++)
    if (vis_bcc[i] == 0) {
      bcc.push_back(vector<int>());
      dfs(i, ++ans);
    }
  printf("%d\n", ans);
  for (int i = 0; i < ans; i++) {
    printf("%llu", bcc[i].size());
    for (int j = 0; j < bcc[i].size(); j++) printf(" %d", bcc[i][j]);
    printf("\n");
  }
  return 0;
}
```

3.5. 二分图匹配

3.5.1. 匈牙利算法

3.5.2. KM

3.6. 网络流

3.6.1. 网络最大流

```
int head[N], cur[N], ecnt, d[N];
struct Edge {
    int nxt, v, flow, cap;
}e[];
void add_edge(int u, int v, int flow, int cap) {
    e[ecnt] = {head[u], v, flow, cap}; head[u] = ecnt++;
    e[ecnt] = {head[v], u, flow, 0}; head[v] = ecnt++;
}
bool bfs() {
    memset(vis, 0, sizeof vis);
    std::queue<int> q;
    q.push(s);
```

```
    vis[s] = 1;
    d[s] = 0;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        for (int i = head[u]; i != -1; i = e[i].nxt) {
            int v = e[i].v;
            if (vis[v] || e[i].flow >= e[i].cap) continue;
            d[v] = d[u] + 1;
            vis[v] = 1;
            q.push(v);
        }
    }
    return vis[t];
}
int dfs(int u, int a) {
    if (u == t || !a) return a;
    int flow = 0, f;
    for (int& i = cur[u]; i != -1; i = e[i].nxt) {
        int v = e[i].v;
        if (d[u] + 1 == d[v] && (f = dfs(v, std::min(a, e[i].cap -
e[i].flow))) > 0) {
            e[i].flow += f;
            e[i ^ 1].flow -= f;
            flow += f;
            a -= f;
            if (!a) break;
        }
    }
    return flow;
}

```

3.6.2. 最小费用最大流

```
const int inf = 1e9;
int head[N], cur[N], ecnt, dis[N], s, t, n, m, mincost;
bool vis[N];
struct Edge {
    int nxt, v, flow, cap, w;
}e[100002];
void add_edge(int u, int v, int flow, int cap, int w) {
    e[ecnt] = {head[u], v, flow, cap, w}; head[u] = ecnt++;
    e[ecnt] = {head[v], u, flow, 0, -w}; head[v] = ecnt++;
}
bool spfa(int s, int t) {
    std::fill(vis + s, vis + t + 1, 0);
    std::fill(dis + s, dis + t + 1, inf);
    std::queue<int> q;
    q.push(s);
    dis[s] = 0;
```

```
    vis[s] = 1;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        vis[u] = 0;
        for (int i = head[u]; i != -1; i = e[i].nxt) {
            int v = e[i].v;
            if (e[i].flow < e[i].cap && dis[u] + e[i].w < dis[v]) {
                dis[v] = dis[u] + e[i].w;
                if (!vis[v]) vis[v] = 1, q.push(v);
            }
        }
    }
    return dis[t] != inf;
}
int dfs(int u, int a) {
    if (vis[u]) return 0;
    if (u == t || !a) return a;
    vis[u] = 1;
    int flow = 0, f;
    for (int& i = cur[u]; i != -1; i = e[i].nxt) {
        int v = e[i].v;
        if (dis[u] + e[i].w == dis[v] && (f = dfs(v, std::min(a,
e[i].cap - e[i].flow))) > 0) {
            e[i].flow += f;
            e[i ^ 1].flow -= f;
            flow += f;
            mincost += e[i].w * f;
            a -= f;
            if (!a) break;
        }
    }
    vis[u] = 0;
    return flow;
}
```

3.7. 2-SAT

$2 * u$ 代表不选择，$2 * u + 1$ 代表选择。

3.7.1. 搜索 (最小字典序)

```
vector<int> G[N * 2];
bool mark[N * 2];
int stk[N], top;
void build_G() {
    for (int i = 1; i <= n; i++) {
        int u, v;
        G[2 * u + 1].push_back(2 * v);
        G[2 * v + 1].push_back(2 * u);
    }
}
```

```
bool dfs(int u) {
    if (mark[u ^ 1]) return false;
    if (mark[u]) return true;
    mark[u] = 1;
    stk[++top] = u;
    for (int v : G[u]) {
        if (!dfs(v)) return false;
    }
    return true;
}
bool 2_sat() {
    for (int i = 1; i <= n; i++) {
        if (!mark[i * 2] && !mark[i * 2 + 1]) {
            top = 0;
            if (!dfs(2 * i)) {
                while (top) mark[stk[top--]] = 0;
                if (!dfs(2 * i + 1)) return 0;
            }
        }
    }
    return 1;
}
```

3.7.2. tarjan

如果对于一个 $\mathbf{x}$ sccno 比它的反状态 $\mathbf{x}\wedge 1$ 的 sccno 要小，那么我们用 $\mathbf{x}$ 这个状态当做答案，否则用它的反状态当做答案。

3.7.3. 前后缀优化

3.7.4. 线段树优化

3.8. 生成树

3.8.1. Prime

```
int n, m;
vector<pii> G[N];
ll dis[N];
int vis[N];
void MAIN() {
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        G[u].push_back({v, w});
        G[v].push_back({u, w});
    }
    for (int i = 1; i <= n; i++) dis[i] = 1e18, vis[i] = 0;
    priority_queue<pair<ll, int>> q;
    dis[1] = 0;
    q.push({-dis[1], 1});
```

```
    ll ans = 0;
    while (!q.empty()) {
        auto [val, u] = q.top(); q.pop();
        if (vis[u]) continue;
        vis[u] = 1;
        ans -= val;
        for (auto [v, w] : G[u]) if (dis[v] > w) {
            dis[v] = w;
            q.push({-w, v});
        }
    }
    cout << ans << '\n';
}
```

3.9. 圆方树

记得开两倍空间。

```
void tarjan(int u) {
    stk[++top] = u;
    low[u] = dfn[u] = ++dfc;
    for (int v : G[u]) {
        if (!dfn[v]) {
            tarjan(v);
            low[u] = min(low[u], low[v]);
            if (low[v] == dfn[u]) {
                cnt++;
                for (int x = 0; x != v; --top) {
                    x = stk[top];
                    T[cnt].push_back(x);
                    T[x].push_back(cnt);
                    val[cnt]++;
                }
                T[cnt].push_back(u);
                T[u].push_back(cnt);
                val[cnt]++;
            }
        } else low[u] = min(low[u], dfn[v]);
    }
}
// 调用
cnt = n;
for (int i = 1; i <= n; i++) if (!dfn[i]) {
    tarjan(i);
    --top;
}
```

- 静态仙人掌最短路。边权设置为到点双顶点的最短距离。

```
void tarjan(int u) {
    stk[++top] = u;
```

```
    dfn[u] = low[u] = ++dfc;
    for (auto [v, w] : G[u]) if (!dfn[v]) {
        dis[v] = dis[u] + w;
        tarjan(v);
        low[u] = min(low[u], low[v]);
        if (low[v] == dfn[u]) {
            ++cnt;
            val[cnt] = cyc[stk[top]] + dis[stk[top]] - dis[u];
            for (int x = 0; x != v; --top) {
                x = stk[top];
                //assert(val[cnt] >= (dis[x] - dis[u]));
                int w = min(dis[x] - dis[u], val[cnt] - (dis[x] -
dis[u]));
                T[cnt].push_back({x, w});
                T[x].push_back({cnt, w});
            }
            T[cnt].push_back({u, 0});
            T[u].push_back({cnt, 0});
        }
    } else if (dfn[v] < dfn[u]) {
        cyc[u] = w;
        low[u] = min(low[u], dfn[v]);
    }
}

void dfs(int u, int fa) {
    faz[0][u] = fa;
    for (int k = 1; k < M; k++) faz[k][u] = faz[k - 1][faz[k - 1][u]];
    for (auto [v, w] : T[u]) if (v != fa) {
        dep[v] = dep[u] + 1;
        ff[v] = ff[u] + w;
        dfs(v, u);
    }
}
int dist(int u, int v) {
    int tu = u, tv = v;
    if (dep[u] < dep[v]) swap(u, v);
    int det = dep[u] - dep[v];
    for (int k = 0; k < M; k++) if ((det >> k) & 1) u = faz[k][u];
    int lca;
    if (u == v) lca = u;
    else {
        for (int k = M - 1; k >= 0; k--) if (faz[k][u] != faz[k][v]) {
            u = faz[k][u]; v = faz[k][v];
        }
        lca = faz[0][u];
    }
    if (lca <= n) return ff[tu] + ff[tv] - ff[lca] * 2;
    int tm = min(abs(dis[u] - dis[v]), val[lca] - abs(dis[u] - dis[v]));
    return ff[tu] - ff[u] + ff[tv] - ff[v] + tm;
}
```

- 圆方树上 dp

以单源最短路为例，原点记录该点出发是否返回的最长路，方点记录顶点出发经过环上所能走到的最长路。

```cpp
void dfs(int u, int fa) {
    for (int v : T[u]) if (v != fa) dfs(v, u);
    if (u <= n) {
        int mx = 0;
        /*
        这里必须设为 0 而不是 -inf, 或者在平凡方点转移的时候要 max(dp[0], dp[1])
        hack: 4 4
        1 2
        2 3
        3 4
        4 2
        */
        for (int v : T[u]) if (v != fa) {
            dp[u][1] += dp[v][1];
            mx = max(mx, dp[v][0] - dp[v][1]);
            dp[u][0] += dp[v][1];
        }
       dp[u][0] += mx;
    } else {
        int sum = 1;
        dp[u][1] = 1;
        for (int v : T[u]) if (v != fa) {
            dp[u][1] += dp[v][1] + 1;
            dp[u][0] = max(dp[u][0], sum + dp[v][0]);
            sum += dp[v][1] + 1;
        }
        sum = 1;
        reverse(T[u].begin(), T[u].end());
        for (int v : T[u]) if (v != fa) {
            dp[u][0] = max(dp[u][0], sum + dp[v][0]);
            sum += dp[v][1] + 1;
        }
        if (val[u] == 2) dp[u][1] = 0;
    }
}
```

3.10. 欧拉回路

- 有向图

```cpp
void dfs(int u) {
    for (int &i = hd[u]; i < G[u].size(); ) dfs(G[u][i++]);
    stk.push_back(u);
}
int check() {
    int mo = 0, le = 0, st = 1;
```

```
    for (int i = 1; i <= n; i++) {
        if (abs(in[i] - out[i]) > 1) return -1;
        if (in[i] > out[i]) le++;
        if (in[i] < out[i]) mo++, st = i;
    }
    if (mo > 1 || le > 1 || mo + le == 1) return -1;
    return st;
}

void MAIN() {
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {
        int u, v;
        cin >> u >> v;
        in[v]++; out[u]++;
        G[u].push_back(v);
    }
    for (int i = 1; i <= n; i++) sort(G[i].begin(), G[i].end());
    int tmp = check();
    if (tmp == -1) cout << "No\n";
    else {
        dfs(tmp);
        copy(stk.rbegin(), stk.rend(), ostream_iterator<int>(cout, "
"));
        cout << '\n';
    }
}
```

- 无向图

```
void dfs(int u) {
    for (int &i = hd[u]; i < G[u].size(); ) {
        while (i < G[u].size() && cnt[u][G[u][i]] == 0) ++i;
        if (i == G[u].size()) break;
        cnt[u][G[u][i]]--;
        cnt[G[u][i]][u]--;
        dfs(G[u][i++]);
    }
    stk.push_back(u);
}
int check() {
    int odd = 0, st = -1;
    for (int i = 1; i <= n; i++) {
        if (deg[i] == 0) continue;
        if (st == -1) st = i;
        if (deg[i] & 1) {
            ++odd;
            if (odd == 1) st = i;
        }
    }
    if (odd > 2) return -1;
```

```
    return st;
}

void MAIN() {
    n = 500;
    cin >> m;
    for (int i = 1; i <= m; i++) {
        int u, v;
        cin >> u >> v;
        ++deg[u]; ++deg[v];
        G[u].push_back(v);
        G[v].push_back(u);
        ++cnt[u][v];
        ++cnt[v][u];
    }
    for (int i = 1; i <= n; i++) sort(G[i].begin(), G[i].end());
    int tmp = check();
    if (tmp == -1) cout << "No\n";
    else {
        dfs(tmp);
        copy(stk.rbegin(), stk.rend(), ostream_iterator<int>(cout,
"\n"));
    }
}
```

3.11. 曼哈顿路

3.12. 无向图三/四元环计数

- 三元环

```
int vis[N];
vector<int> G[N];
ll main() {
    ll cnt = 0;
    for (int i = 0; i < m; i++) {
        if (deg[ed[i].fi] == deg[ed[i].se] && ed[i].fi > ed[i].se)
swap(ed[i].fi, ed[i].se);
        if (deg[ed[i].fi] > deg[ed[i].se]) swap(ed[i].fi, ed[i].se);
        G[ed[i].fi].push_back(ed[i].se);
    }
    for (int u = 1; u <= n; u++) {
        for (int v : G[u]) vis[v] = 1;
        for (int v : G[u]) for (int w : G[v]) if (vis[w]) ++cnt;
        for (int v : G[u]) vis[v] = 0;
    }
    return cnt;
}
```

- 四元环

统计 $c?\, b \to a \leftarrow d?\, c$ 的数目，因为最大度数点 a 不同，所以不会算重。

```cpp
int n, m, deg[N], cnt[N];
bool bigger(int a, int b) {
    return deg[a] > deg[b] || (deg[a] == deg[b] && a > b);
}
void MAIN() {
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {
        int u, v;
        cin >> u >> v;
        ed.push_back({u, v});
        G[u].push_back(v);
        G[v].push_back(u);
        ++deg[u]; ++deg[v];
    }
    for (auto [u, v] : ed) {
        if (bigger(v, u)) swap(u, v);
        T[u].push_back(v);
    }
    ll ans = 0;
    for (int a = 1; a <= n; a++) {
        for (int b : T[a]) {
            for (int c : G[b]) {
                if (c == a || bigger(c, a)) continue;
                ans += cnt[c];
                ++cnt[c];
            }
        }
        for (int b : T[a]) for (int c : G[b]) cnt[c] = 0;
    }
    cout << ans << '\n';
}
```

4. 树论

4.1. prufer

4.2. 虚树

需要保证 $\mathrm{LCA}(0,u)=0$

```cpp
int solve(vector<int>po) {
    sort(po.begin(), po.end(), [](int x, int y) {
        return dfn[x] < dfn[y];
    });
    int ans = 0;
    top = 0;
    stk[++top] = 0;
    for (int u : po) {
        int lca = LCA(u, stk[top]);
        if (lca == stk[top]) stk[++top] = u;
        else {
```

```
12              for (int i = top; i >= 2 && dep[stk[i - 1]] >= dep[lca];
    i--) {
13                //  ans += ff[stk[i]] - ff[stk[i - 1]] - (vis[stk[i]] ?
    val[stk[i]]: 0);
14                //  cout << stk[i] << ' ' << stk[i - 1] << ' ' <<
    ff[stk[i]] - ff[stk[i - 1]] - (vis[stk[i]] ? val[stk[i]]: 0) << '\n';
15                  add_edge(stk[i], stk[i - 1]);
16                  --top;
17              }
18              if (stk[top] != lca) {
19                //  cout << lca << ' ' << stk[top] << ' ' << ff[stk[top]]
    - ff[lca] - (vis[stk[top]] ? val[stk[top]] : 0) << '\n';
20                //  ans += ff[stk[top]] - ff[lca] - (vis[stk[top]] ?
    val[stk[top]] : 0);
21                  add_edge(stk[top], lca);
22                  stk[top] = lca;
23              }
24              stk[++top] = u;
25          }
26      }
27      for (int i = 2; i < top; i++) {
28        //  cout << stk[i + 1] << ' ' << stk[i] << ' ' << ff[stk[i + 1]] -
    ff[stk[i]] - (vis[stk[i + 1]] ? val[stk[i + 1]] : 0) << '\n';
29         // ans += ff[stk[i + 1]] - ff[stk[i]] - (vis[stk[i + 1]] ?
    val[stk[i + 1]] : 0);
30          add_edge(stk[i + 1], stk[i]);
31      }
32      //ans += (vis[stk[2]] ? 0 : val[stk[2]]);
33      return ans;
34  }
```

4.3. 最近公共祖先

4.4. 树分治

4.4.1. 点分治

4.4.2. 点分树

4.5. 链分治

4.5.1. 重链分治

4.5.2. 长链分治

4.6. dsu on tree

5. 数学

5.1. 组合恒等式

5.2. min-max 容斥

5.3. 序列容斥

5.4. 二项式反演

5.5. 斯特林数

5.6. 高维前缀和

5.7. 线性基

5.8. 行列式

5.9. 高斯消元

```cpp
namespace Gauss {
    bitset<258> a[256 + 256 + 5];
    int n;
    void push(const bitset<258>& x) {
        a[++n] = x;
    }
    bool solve(int m) {
        int k = 1;
        for (int i = 1; i <= m; i++) {
            if (k > n) break;
            for (int j = k + 1; j <= n; j++) if (a[j][i] > 0) {
                swap(a[k], a[j]);
                break;
            }
            if (a[k][i] == 0) break;
            for (int j = 1; j <= n; j++) if (j != k && a[j][i]) {
                a[j] ^= a[k];
            }
            ++k;
        }
        for (int i = k; i <= n; i++) if (a[i][m + 1]) return false;
        return true;
    }
}
```

6. 多项式

6.1. NTT

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

typedef vector<int> poly;
const int mod = 998244353;
```

```cpp
const int N = 4000000 + 5;

int rf[32][N];
int fpow(int a, int b) {
    int res = 1;
    for (; b; b >>= 1, a = a * 1ll * a % mod) if (b & 1)
        res = res * 1ll * a % mod;
    return res;
}
void init(int n) {
    assert(n < N);
    int lg = __lg(n);
    static vector<bool> bt(32, 0);
    if (bt[lg] == 1) return;
    bt[lg] = 1;
    for (int i = 0; i < n; i++) rf[lg][i] = (rf[lg][i >> 1] >> 1) + ((i & 1) ? (n >> 1) : 0);
}
void ntt(poly &x, int lim, int op) {
    int lg = __lg(lim), gn, g, tmp;;
    for (int i = 0; i < lim; i++) if (i < rf[lg][i]) swap(x[i], x[rf[lg][i]]);
    for (int len = 2; len <= lim; len <<= 1) {
        int k = (len >> 1);
        gn = fpow(3, (mod - 1) / len);
        for (int i = 0; i < lim; i += len) {
            g = 1;
            for (int j = 0; j < k; j++, g = gn * 1ll * g % mod) {
                tmp = x[i + j + k] * 1ll * g % mod;
                x[i + j + k] = (x[i + j] - tmp + mod) % mod;
                x[i + j] = (x[i + j] + tmp) % mod;
            }
        }
    }
    if (op == -1) {
        reverse(x.begin() + 1, x.begin() + lim);
        int inv = fpow(lim, mod - 2);
        for (int i = 0; i < lim; i++) x[i] = x[i] * 1ll * inv % mod;
    }
}
poly multiply(const poly &a, const poly &b) {
    assert(!a.empty() && !b.empty());
    int lim = 1;
    while (lim + 1 < int(a.size() + b.size())) lim <<= 1;
    init(lim);
    poly pa = a, pb = b;
    while (pa.size() < lim) pa.push_back(0);
    while (pb.size() < lim) pb.push_back(0);
    ntt(pa, lim, 1); ntt(pb, lim, 1);
    for (int i = 0; i < lim; i++) pa[i] = pa[i] * 1ll * pb[i] % mod;
    ntt(pa, lim, -1);
    while (int(pa.size()) + 1 > int(a.size() + b.size())) pa.pop_back();
```

```
    return pa;
}
poly prod_poly(const vector<poly>& vec) { // init vector, too slow
    int n = vec.size();
    auto calc = [&](const auto &self, int l, int r) -> poly {
        if (l == r) return vec[l];
        int mid = (l + r) >> 1;
        return multiply(self(self, l, mid), self(self, mid + 1, r));
    };
    return calc(calc, 0, n - 1);
}

// Semi-Online-Convolution
poly semi_online_convolution(const poly& g, int n, int op = 0) {
    assert(n == g.size());
    poly f(n, 0);
    f[0] = 1;
    auto CDQ = [&](const auto &self, int l, int r) -> void {
        if (l == r) {
            // exp
            if (op == 1 && l > 0) f[l] = f[l] * 1ll * fpow(l, mod - 2) %
mod;
            return;
        }
        int mid = (l + r) >> 1;
        self(self, l, mid);
        poly a, b;
        for (int i = l; i <= mid; i++) a.push_back(f[i]);
        for (int i = 0; i <= r - l - 1; i++) b.push_back(g[i + 1]);
        a = multiply(a, b);
        for (int i = mid + 1; i <= r; i++) f[i] = (f[i] + a[i - l - 1])
% mod;
        self(self, mid + 1, r);
    };
    CDQ(CDQ, 0, n - 1);
    return f;
}

poly getinv(const poly &a) {
    assert(!a.empty());
    poly res = {fpow(a[0], mod - 2)}, na = {a[0]};
    int lim = 1;
    while (lim < int(a.size())) lim <<= 1;
    for (int len = 2; len <= lim; len <<= 1) {
        while (na.size() < len) {
            int tmp = na.size();
            if (tmp < a.size()) na.push_back(a[tmp]);
            else na.push_back(0);
        }
        auto tmp = multiply(na, res);
        for (auto &x : tmp) x = (x > 0 ? mod - x : x);
        tmp[0] = ((tmp[0] + 2) >= mod) && (tmp[0] -= mod);
```

```
106         tmp = multiply(res, tmp);
107         while (tmp.size() > len) tmp.pop_back();
108         res = tmp;
109     }
110     while (res.size() > a.size()) res.pop_back();
111     return res;
112 }
113 poly exp(const poly &g) {
114     int n = g.size();
115     poly b(n, 0);
116     for (int i = 1; i < n; i++) b[i] = i * 1ll * g[i] % mod;
117     return semi_online_convolution(b, n, 1);
118 }
119 poly ln(const poly &A) {
120     int n = A.size();
121     auto C = getinv(A);
122     poly A1(n, 0);
123     for (int i = 0; i < n - 1; i++) A1[i] = (i + 1) * 1ll * A[i + 1] %
    mod;
124     C = multiply(C, A1);
125     for (int i = n - 1; i > 0; i--) C[i] = C[i - 1] * 1ll * fpow(i, mod
    - 2) % mod;
126     C[0] = 0;
127     while (C.size() > n) C.pop_back();
128     return C;
129 }
130 poly quick_pow(poly &a, int k, int k_mod_phi, bool is_k_bigger_than_mod
    = false) {
131     assert(!a.empty());
132     int n = a.size(), t = -1, b;
133     for (int i = 0; i < n; i++) if (a[i]) {
134         t = i, b = a[i];
135         break;
136     }
137     if (t == -1 || t && is_k_bigger_than_mod || k * 1ll * t >= n) return
    poly(n, 0);
138     poly f;
139     for (int i = 0; i < n; i++) {
140         if (i + t < n) f.push_back(a[i + t] * 1ll * fpow(b, mod - 2) %
    mod);
141         else f.push_back(0);
142     }
143     f = ln(f);
144     for (auto &x : f) x = x * 1ll * k % mod;
145     f = exp(f);
146     poly res;
147     for (int i = 0; i < k * t; i++) res.push_back(0);
148     int fb = fpow(b, k_mod_phi);
149     for (int i = k * t; i < n; i++) res.push_back(f[i - k * t] * 1ll *
    fb % mod);
150     return res;
151 }
```

```
152
153 int main() {
154     ios::sync_with_stdio(0); cin.tie(0);
155     int n, k = 0, k_mod_phi = 0, isb = 0;
156     string s;
157     cin >> n >> s;
158     for (auto ch : s) {
159         if ((ch - '0') + k * 10ll >= mod) isb = 1;
160         k = ((ch - '0') + k * 10ll) % mod;
161         k_mod_phi = ((ch - '0') + k_mod_phi * 10ll) % 998244352;
162     }
163     poly a(n);
164     for (auto &x : a) cin >> x;
165     a = quick_pow(a, k, k_mod_phi, isb);
166     while (a.size() > n) a.pop_back();
167     for (auto x : a) cout << x << ' ';
168     return 0;
169 }
```

6.2. 任意模数 NTT

模数小于 10^9

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  typedef complex<double> cp;
5  typedef vector<cp> poly;
6  typedef long long ll;
7
8  const int N = 4000000 + 5;
9  const double pi = acos(-1);
10
11 int rf[26][N];
12 void init(int n) {
13     assert(n < N);
14     int lg = __lg(n);
15     static vector<bool> bt(26, 0);
16     if (bt[lg] == 1) return;
17     bt[lg] = 1;
18     for (int i = 0; i < n; i++) rf[lg][i] = (rf[lg][i >> 1] >> 1) + ((i
   & 1) ? (n >> 1) : 0);
19 }
20 void fft(poly &x, int lim, int op) {
21     int lg = __lg(lim);
22     for (int i = 0; i < lim; i++) if (i < rf[lg][i]) swap(x[i], x[rf[lg]
   [i]]);
23     for (int len = 2; len <= lim; len <<= 1) {
24         int k = (len >> 1);
25         for (int i = 0; i < lim; i += len) {
26             for (int j = 0; j < k; j++) {
```

```
                cp w(cos(pi * j / k), op * sin(pi * j / k));
                cp tmp = w * x[i + j + k];
                x[i + j + k] = x[i + j] - tmp;
                x[i + j] = x[i + j] + tmp;
            }
        }
    }
    if (op == -1) for (int i = 0; i < lim; i++) x[i] /= lim;
}
poly multiply(const poly &a, const poly &b) {
    assert(!a.empty() && !b.empty());
    int lim = 1;
    while (lim + 1 < int(a.size() + b.size())) lim <<= 1;
    init(lim);
    poly pa = a, pb = b;
    pa.resize(lim);
    pb.resize(lim);
    for (int i = 0; i < lim; i++) pa[i] = (cp){pa[i].real(),
pb[i].real()};
    fft(pa, lim, 1);
    pb[0] = conj(pa[0]);
    for (int i = 1; i < lim; i++) pb[lim - i] = conj(pa[i]);
    for (int i = 0; i < lim; i++) {
        pa[i] = (pa[i] + pb[i]) * (pa[i] - pb[i]) / cp({0, 4});
    }
    fft(pa, lim, -1);
    pa.resize(int(a.size() + b.size()) - 1);
    return pa;
}
vector<int> MTT(const vector<int> &a, const vector<int> &b, const int
mod) {
    const int B = (1 << 15) - 1, M = (1 << 15);
    int lim = 1;
    while (lim + 1 < int(a.size() + b.size())) lim <<= 1;
    init(lim);
    poly pa(lim), pb(lim);
    auto get = [](const vector<int>& v, int pos) -> int {
        if (pos >= v.size()) return 0;
        else return v[pos];
    };
    for (int i = 0; i < lim; i++) pa[i] = (cp){get(a, i) >> 15, get(a,
i) & B};
    fft(pa, lim, 1);
    pb[0] = conj(pa[0]);
    for (int i = 1; i < lim; i++) pb[lim - i] = conj(pa[i]);
    poly A0(lim), A1(lim);
    for (int i = 0; i < lim; i++) {
        A0[i] = (pa[i] + pb[i]) / (cp){2, 0};
        A1[i] = (pa[i] - pb[i]) / (cp){0, 2};
    }
    for (int i = 0; i < lim; i++) pa[i] = (cp){get(b, i) >> 15, get(b,
i) & B};
```

```
75      fft(pa, lim, 1);
76      pb[0] = conj(pa[0]);
77      for (int i = 1; i < lim; i++) pb[lim - i] = conj(pa[i]);
78      poly B0(lim), B1(lim);
79      for (int i = 0; i < lim; i++) {
80          B0[i] = (pa[i] + pb[i]) / (cp){2, 0};
81          B1[i] = (pa[i] - pb[i]) / (cp){0, 2};
82      }
83      for (int i = 0; i < lim; i++) {
84          pa[i] = A0[i] * B0[i];
85          pb[i] = A0[i] * B1[i];
86          A0[i] = pa[i];
87          pa[i] = A1[i] * B1[i];
88          B1[i] = pb[i];
89          B0[i] = A1[i] * B0[i];
90          A1[i] = pa[i];
91          pa[i] = A0[i] + (cp){0, 1} * A1[i];
92          pb[i] = B0[i] + (cp){0, 1} * B1[i];
93      }
94      fft(pa, lim, -1); fft(pb, lim, -1);
95      vector<int> res(int(a.size() + b.size()) - 1);
96      const int M2 = M * 1ll * M % mod;
97      for (int i = 0; i < res.size(); i++) {
98          ll a0 = round(pa[i].real()), a1 = round(pa[i].imag()), b0 =
    round(pb[i].real()), b1 = round(pb[i].imag());
99          a0 %= mod; a1 %= mod; b0 %= mod; b1 %= mod;
100         res[i] = (a0 * 1ll * M2 % mod + a1 + (b0 + b1) % mod * 1ll * M %
    mod) % mod;
101     }
102     return res;
103 }
104
105 int main() {
106 #ifdef LOCAL
107     freopen("miku.in", "r", stdin);
108     freopen("miku.out", "w", stdout);
109 #endif
110     ios::sync_with_stdio(0); cin.tie(0);
111     int n, m, p;
112     cin >> n >> m >> p;
113     vector<int> a(n + 1), b(m + 1);
114     for (auto &x : a) cin >> x;
115     for (auto &x : b) cin >> x;
116     auto res = MTT(a, b, p);
117     for (auto x : res) cout << x << ' ';
118 }
```

6.3. 自然数幂和

6.4. 快速沃尔什变换

6.5. 子集卷积

7. 数据结构

7.1. 线段树

7.1.1. 李超树 (最大，次大，第三大)

7.1.2. 合并分裂

7.1.3. 线段树二分

7.1.4. 兔队线段树

7.2. 平衡树

7.2.1. 文艺平衡树

7.3. 历史版本信息线段树

7.4. 树状数组二分

7.5. 二维树状数组

7.6. ODT

7.7. KDT

7.8. 手写堆

8. 字符串

8.1. KMP

```
int n = strlen(s + 1);
for (int i = 2; i <= n; i++) {
    int j = k[i - 1];
    while (j != 0 && s[i] != s[j + 1]) j = k[j];
    if (s[i] == s[j + 1]) k[i] = j + 1;
    else k[i] = 0;
}
```

8.2. Z function

```
 for (int i = 2, l = 0, r = 0; i <= n; i++) {
     if (r >= i && r - i + 1 > z[i - l + 1]) {
      z[i] = z[i - l + 1];
     } else {
      z[i] = max(0, r - i + 1);
      while (z[i] < n - i + 1 && s[z[i] + 1] == s[i + z[i]]) ++z[i];
     }
     if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
```

```
}
```

8.3. SA

```
int sa[N], ork[N], rk[N], cnt[N], id[N], h[N], M, n;
char s[N];
int mn[22][N];
int lcp(int a, int b) {
    if (a == b) return n - a + 1;
    if (rk[a] > rk[b]) swap(a, b);
    int l = rk[a] + 1, r = rk[b];
    int len = r - l + 1, k = __lg(len);
    return min(mn[k][l], mn[k][r - (1 << k) + 1]);
}
void MAIN() {
    scanf("%s", s + 1);
    n = strlen(s + 1);
    for (int i = 1; i <= n; i++) M = max(M, (int)s[i]);
    for (int i = 1; i <= n; i++) if ((int)(s[i]) > M) M = (int)(s[i]);
    for (int i = 1; i <= n; i++) cnt[rk[i] = s[i]]++;
    for (int i = 0; i <= M; i++) cnt[i] += cnt[i - 1];
    for (int i = n; i; i--) sa[cnt[rk[i]]--] = i;
    for (int w = 1, p; w < n; w <<= 1, M = p) {
        p = 0;
        for (int i = n; i > n - w; i--) id[++p] = i;
        for (int i = 1; i <= n; i++) if (sa[i] > w) id[++p] = sa[i] - w;
        for (int i = 0; i <= M; i++) cnt[i] = 0;
        for (int i = 1; i <= n; i++) cnt[rk[i]]++;
        for (int i = 1; i <= M; i++) cnt[i] += cnt[i - 1];
        for (int i = n; i; i--) sa[cnt[rk[id[i]]]--] = id[i];
        p = 0;
        for (int i = 0; i <= n; i++) ork[i] = rk[i];
        for (int i = 1; i <= n; i++) {
            if (ork[sa[i]] == ork[sa[i - 1]] && ork[sa[i] + w] ==
ork[sa[i - 1] + w]) rk[sa[i]] = p;
            else rk[sa[i]] = ++p;
        }
        if (p == n) break;
    }
    for (int i = 1, k = 0; i <= n; i++) {
        if (rk[i] == 1) continue;
        if (k) k--;
        while (s[i + k] == s[sa[rk[i] - 1] + k]) k++;
        h[rk[i]] = k;
    }
    for (int i = 1; i <= n; i++) mn[0][i] = h[i];
    for (int j = 1; j < 22; j++) {
        for (int i = 1; i <= n; i++) {
            mn[j][i] = min(mn[j - 1][i], mn[j - 1][min(n, i + (1 << (j -
1)))]);
        }
```

```
    }
}
```

8.4. AC 自动机

```
int ch[N][26], tot, fail[N], e[N];
void insert(const char *s) {
  int u = 0, n = strlen(s + 1);
  for (int i = 1; i <= n; i++) {
        if (!ch[u][s[i] - 'a']) ch[u][s[i] - 'a'] = ++tot;
        u = ch[u][s[i] - 'a'];
  }
  e[u] += 1;
}
void build() {
  queue<int> q;
  for (int i = 0; i <= 25; i++) if (ch[0][i]) q.push(ch[0][i]);
  while (!q.empty()) {
    int now = q.front(); q.pop();
    for (int i = 0; i < 26; i++) {
      if (ch[now][i]) fail[ch[now][i]] = ch[fail[now]][i],
q.push(ch[now][i]);
      else ch[now][i] = ch[fail[now]][i];
    }
  }
}
int query(const char *s) {
    int u = 0, n = strlen(s + 1), res = 0;
    for (int i = 1; i <= n; i++){
      u = ch[u][s[i] - 'a'];
      for (int j = u; j && e[j] != -1; j = fail[j]) {
        res += e[j];
        e[j] = -1;
      }
    }
    return res;
}
```

8.5. Manacher

对于第 i 个字符为对称轴:

1. 如果回文串长为奇数, $\frac{d[2*i]}{2}$ 是半径加上自己的长度
2. 如果长为偶数, $\frac{d[2*i-1]}{2}$ 是半径的长度, 方向向右.

```
int n, d[N * 2];
char s[N];

for (int i = 1; i <= n; i++) t[i * 2] = s[i], t[i * 2 - 1] = '#';
t[n * 2 + 1] = '#';
```

```
m = n * 2 + 1;
for (int i = 1, l = 0, r = 0; i <= m; i++) {
    int k = i <= r ? min(d[r - i + l], r - i + 1) : 1;
    while (i + k <= m && i - k >= 1 && t[i + k] == t[i - k]) k++;
    d[i] = k--;
    if (i + k > r) r = i + k, l = i - k;
}
```

9. 杂项

9.1. fastio

来自 oiwiki

```
// #define DEBUG 1  // 调试开关
struct IO {
#define MAXSIZE (1 << 20)
#define isdigit(x) (x >= '0' && x <= '9')
  char buf[MAXSIZE], *p1, *p2;
  char pbuf[MAXSIZE], *pp;
#if DEBUG
#else
  IO() : p1(buf), p2(buf), pp(pbuf) {}

  ~IO() { fwrite(pbuf, 1, pp - pbuf, stdout); }
#endif
  char gc() {
#if DEBUG  // 调试，可显示字符
    return getchar();
#endif
    if (p1 == p2) p2 = (p1 = buf) + fread(buf, 1, MAXSIZE, stdin);
    return p1 == p2 ? ' ' : *p1++;
  }

  bool blank(char ch) {
    return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t';
  }

  template <class T>
  void read(T &x) {
    double tmp = 1;
    bool sign = false;
    x = 0;
    char ch = gc();
    for (; !isdigit(ch); ch = gc())
      if (ch == '-') sign = 1;
    for (; isdigit(ch); ch = gc()) x = x * 10 + (ch - '0');
    if (ch == '.')
      for (ch = gc(); isdigit(ch); ch = gc())
        tmp /= 10.0, x += tmp * (ch - '0');
    if (sign) x = -x;
```

```
38   }
39
40   void read(char *s) {
41     char ch = gc();
42     for (; blank(ch); ch = gc());
43     for (; !blank(ch); ch = gc()) *s++ = ch;
44     *s = 0;
45   }
46
47   void read(char &c) { for (c = gc(); blank(c); c = gc()); }
48
49   void push(const char &c) {
50 #if DEBUG  // 调试，可显示字符
51     putchar(c);
52 #else
53     if (pp - pbuf == MAXSIZE) fwrite(pbuf, 1, MAXSIZE, stdout), pp =
   pbuf;
54     *pp++ = c;
55 #endif
56   }
57
58   template <class T>
59   void write(T x) {
60     if (x < 0) x = -x, push('-');  // 负数输出
61     static T sta[35];
62     T top = 0;
63     do {
64       sta[top++] = x % 10, x /= 10;
65     } while (x);
66     while (top) push(sta[--top] + '0');
67   }
68
69   template <class T>
70   void write(T x, char lastChar) {
71     write(x), push(lastChar);
72   }
73 } io;
74
```

9.2. 高精度

来自 oiwiki

```
1 constexpr int MAXN = 9999;
2 // MAXN 是一位中最大的数字
3 constexpr int MAXSIZE = 10024;
4 // MAXSIZE 是位数
5 constexpr int DLEN = 4;
6
7 // DLEN 记录压几位
8 struct Big {
```

```
  int a[MAXSIZE], len;
  bool flag;  // 标记符号'-'

  Big() {
    len = 1;
    memset(a, 0, sizeof a);
    flag = false;
  }

  Big(const int);
  Big(const char*);
  Big(const Big&);
  Big& operator=(const Big&);
  Big operator+(const Big&) const;
  Big operator-(const Big&) const;
  Big operator*(const Big&) const;
  Big operator/(const int&) const;
  // TODO: Big / Big;
  Big operator^(const int&) const;
  // TODO: Big ^ Big;

  // TODO: Big 位运算;

  int operator%(const int&) const;
  // TODO: Big ^ Big;
  bool operator<(const Big&) const;
  bool operator<(const int& t) const;
  void print() const;
};

Big::Big(const int b) {
  int c, d = b;
  len = 0;
  // memset(a,0,sizeof a);
  CLR(a);
  while (d > MAXN) {
    c = d - (d / (MAXN + 1) * (MAXN + 1));
    d = d / (MAXN + 1);
    a[len++] = c;
  }
  a[len++] = d;
}

Big::Big(const char* s) {
  int t, k, index, l;
  CLR(a);
  l = strlen(s);
  len = l / DLEN;
  if (l % DLEN) ++len;
  index = 0;
  for (int i = l - 1; i >= 0; i -= DLEN) {
    t = 0;
```

```
61      k = i - DLEN + 1;
62      if (k < 0) k = 0;
63      g(j, k, i) t = t * 10 + s[j] - '0';
64      a[index++] = t;
65    }
66  }
67
68  Big::Big(const Big& T) : len(T.len) {
69    CLR(a);
70    f(i, 0, len) a[i] = T.a[i];
71    // TODO:重载此处？
72  }
73
74  Big& Big::operator=(const Big& T) {
75    CLR(a);
76    len = T.len;
77    f(i, 0, len) a[i] = T.a[i];
78    return *this;
79  }
80
81  Big Big::operator+(const Big& T) const {
82    Big t(*this);
83    int big = len;
84    if (T.len > len) big = T.len;
85    f(i, 0, big) {
86      t.a[i] += T.a[i];
87      if (t.a[i] > MAXN) {
88        ++t.a[i + 1];
89        t.a[i] -= MAXN + 1;
90      }
91    }
92    if (t.a[big])
93      t.len = big + 1;
94    else
95      t.len = big;
96    return t;
97  }
98
99  Big Big::operator-(const Big& T) const {
100   int big;
101   bool ctf;
102   Big t1, t2;
103   if (*this < T) {
104     t1 = T;
105     t2 = *this;
106     ctf = true;
107   } else {
108     t1 = *this;
109     t2 = T;
110     ctf = false;
111   }
112   big = t1.len;
```

```
113   int j = 0;
114   f(i, 0, big) {
115     if (t1.a[i] < t2.a[i]) {
116       j = i + 1;
117       while (t1.a[j] == 0) ++j;
118       --t1.a[j--];
119       // WTF?
120       while (j > i) t1.a[j--] += MAXN;
121       t1.a[i] += MAXN + 1 - t2.a[i];
122     } else
123       t1.a[i] -= t2.a[i];
124   }
125   t1.len = big;
126   while (t1.len > 1 && t1.a[t1.len - 1] == 0) {
127     --t1.len;
128     --big;
129   }
130   if (ctf) t1.a[big - 1] = -t1.a[big - 1];
131   return t1;
132 }
133
134 Big Big::operator*(const Big& T) const {
135   Big res;
136   int up;
137   int te, tee;
138   f(i, 0, len) {
139     up = 0;
140     f(j, 0, T.len) {
141       te = a[i] * T.a[j] + res.a[i + j] + up;
142       if (te > MAXN) {
143         tee = te - te / (MAXN + 1) * (MAXN + 1);
144         up = te / (MAXN + 1);
145         res.a[i + j] = tee;
146       } else {
147         up = 0;
148         res.a[i + j] = te;
149       }
150     }
151     if (up) res.a[i + T.len] = up;
152   }
153   res.len = len + T.len;
154   while (res.len > 1 && res.a[res.len - 1] == 0) --res.len;
155   return res;
156 }
157
158 Big Big::operator/(const int& b) const {
159   Big res;
160   int down = 0;
161   gd(i, len - 1, 0) {
162     res.a[i] = (a[i] + down * (MAXN + 1)) / b;
163     down = a[i] + down * (MAXN + 1) - res.a[i] * b;
164   }
```

```
165   res.len = len;
166   while (res.len > 1 && res.a[res.len - 1] == 0) --res.len;
167   return res;
168 }
169
170 int Big::operator%(const int& b) const {
171   int d = 0;
172   gd(i, len - 1, 0) d = (d * (MAXN + 1) % b + a[i]) % b;
173   return d;
174 }
175
176 Big Big::operator^(const int& n) const {
177   Big t(n), res(1);
178   int y = n;
179   while (y) {
180     if (y & 1) res = res * t;
181     t = t * t;
182     y >>= 1;
183   }
184   return res;
185 }
186
187 bool Big::operator<(const Big& T) const {
188   int ln;
189   if (len < T.len) return true;
190   if (len == T.len) {
191     ln = len - 1;
192     while (ln >= 0 && a[ln] == T.a[ln]) --ln;
193     if (ln >= 0 && a[ln] < T.a[ln]) return true;
194     return false;
195   }
196   return false;
197 }
198
199 bool Big::operator<(const int& t) const {
200   Big tee(t);
201   return *this < tee;
202 }
203
204 void Big::print() const {
205   printf("%d", a[len - 1]);
206   gd(i, len - 2, 0) { printf("%04d", a[i]); }
207 }
208
209 void print(const Big& s) {
210   int len = s.len;
211   printf("%d", s.a[len - 1]);
212   gd(i, len - 2, 0) { printf("%04d", s.a[i]); }
213 }
```

9.3. 手写 bitset

```cpp
struct Bitset {
    #define For(i,a,b) for(int i=a,i##end=b; i<=i##end; i++)
    #define foR(i,a,b) for(int i=a,i##end=b; i>=i##end; i--)
    using uint = unsigned int;
    using ull = unsigned long long;
    vector < ull > bit; int len;
    Bitset(int x = n) {x = (x >> 6) + 1; bit.resize(x); len = x;}
    void resize(int x) {bit.resize((x >> 6) + 1); len = (x >> 6) + 1;For(i, 0, len-1) bit[i] = 0;}
    void set1(int x) {bit[x>>6] |= (1ull<<(x&63));}
    void set0(int x) {bit[x>>6] &= (~(1ull<<(x&63)));}
    void flip(int x) {bit[x>>6] ^= (1ull<<(x&63));}
    bool operator [] (int x) {return (bit[x>>6] >> (x&63)) & 1;}
    bool any() {For(i, 0, len-1) if(bit[i]) return 1;return 0;}
    Bitset operator ~ () const {Bitset res(len);For(i, 0, len-1) res.bit[i] = ~bit[i];return res;}
    Bitset operator | (const Bitset &b) const {Bitset res(len); For(i, 0, len-1) res.bit[i] = bit[i] | b.bit[i];return res;}
    Bitset operator & (const Bitset &b) const {Bitset res(len); For(i, 0, len-1) res.bit[i] = bit[i] & b.bit[i];return res;}
    Bitset operator ^ (const Bitset &b) const {Bitset res(len); For(i, 0, len-1) res.bit[i] = bit[i] ^ b.bit[i];return res;}
    void operator &= (const Bitset &b) {For(i, 0, len-1) bit[i] &= b.bit[i];}
    void operator |= (const Bitset &b) {For(i, 0, len-1) bit[i] |= b.bit[i];}
    void operator ^= (const Bitset &b) {For(i, 0, len-1) bit[i] ^= b.bit[i];}
    Bitset operator << (const int t) const {
        Bitset res(len); int high = t >> 6, low = t & 63; ull lst = 0;
        for(int i = 0; i + high < len; i++) {
            res.bit[i + high] = (lst | (bit[i] << low));
            if(low) lst = (bit[i] >> (64 - low));
        }
        return res;
    }
    Bitset operator >> (const int t) const {
        Bitset res(len); int high = t >> 6, low = t & 63; ull lst = 0;
        for(int i = len - 1; i >= high; i--) {
            res.bit[i - high] = (lst | (bit[i] >> low));
            if(low) lst = (bit[i] << (64 - low));
        }
        return res;
    }
    void operator <<= (const int t) {
        int high = t >> 6, low = t & 63;
        for(int i = len - high - 1; ~i; i--) {
            bit[i + high] = (bit[i] << low);
            if(low && i) bit[i + high] |= (bit[i - 1] >> (64 - low));
        }
        for(int i = 0; i < min(high, len - 1); i++) bit[i] = 0;
    }
```

```
45      void operator >>= (const int t) {
46          int high = t >> 6, low = t & 63;
47          for(int i = high; i < len; i++) {
48              bit[i - high] = (bit[i] >> low);
49              if(low && i != len) bit[i - high] |= (bit[i + 1] << (64 -
    low));
50          }
51          for(int i = max(len - high, 0); i < len; i++) bit[i] = 0;
52      }
53      ull get(int x) {
54          int t = x >> 6, q = x & 63;
55          if (q == 63) return bit[t];
56          return bit[t] & ((1ull << (q + 1)) - 1);
57      }
58      ull get(int l, int r) {
59          int lt = (l >> 6), rt = (r >> 6);
60          if (lt == rt) {
61              if ((l & 63) == 0) return get(r);
62              return (get(r) - get(l - 1)) >> ((l & 63));
63          }
64           ull a = (l & 63) == 0 ? (bit[lt]) : ((bit[lt] - get(l - 1)) >>
    ((l & 63)));
65           return a + (get(r) << (64 - (l & 63)));
66       }
67  }
```

9.4. 对拍

```
1  #!/usr/bin/bash
2  g++ ./my.cpp -o my -std=c++17 -fsanitize=undefined
3  g++ ./std.cpp -o std -std=c++17 -fsanitize=undefined
4  g++ ./data.cpp -o data -std=c++17 -fsanitize=undefined
5  cnt=0;
6  while true; do
7    ./data > data.in
8    ./my < data.in > my.out
9    ./std < data.in > std.out
10   if diff my.out std.out; then
11     let cnt++;
12     echo "# $cnt AC";
13   else
14     echo "WA";
15     break;
16   fi
17 done
```