endless rain: widsnoy, WQhuanm, xu826281112

widsnoy's template

1. 数论 4
1.1. 取模还原分数 4
1.2. 原根 4
1.3. 解不定方程 5
1.4. 中国剩余定理 6
1.5. 卢卡斯定理 7
1.6. BSGS 9
1.7. 二次剩余（待补） 10
1.8. Miller-Rabin（待补） 10
1.9. Pollard-rho（待补） 10
1.10. 数论函数 10
1.11. 莫比乌斯反演 10
1.12. 整除分块 10
1.13. 区间筛 11
1.14. 杜教筛 11
1.15. Min25 筛 11
2. 动态规划 13
2.1. 缺 1 背包 13
3. 图论 13
3.1. 找环 13
3.2. SPFA 乱搞 14
3.3. 差分约束 15
3.4. 竞赛图 15
3.5. 有向图强连通分量 15
3.5.1. Tarjan 15
3.5.2. Kosaraju 16
3.6. 强连通分量(incremental) 16
3.7. 连通分量 18
3.7.1. 割点和桥 18
3.7.2. 点双 18
3.7.3. 边双 19
3.8. 二分图匹配 21
3.8.1. 匈牙利算法 21
3.8.2. KM 21
3.9. 网络流 21
3.9.1. 网络最大流 21
3.9.2. 最小费用最大流 22
3.9.3. 上下界网络流（待学） 23
3.10. 2-SAT 23
3.10.1. 搜索 (最小字典序) 23
3.10.2. tarjan 24
3.11. 生成树 24
3.11.1. Prime 24

3.11.2. 次小生成树 24
3.11.3. 生成树计数 25
3.12. 三元环 25
3.13. 四元环 25
3.14. 欧拉路 25
3.15. 曼哈顿路 25
3.16. 建图优化 25
3.16.1. 前后缀优化 25
3.16.2. 线段树优化 25
4. 树论 25
4.1. prufer 25
4.2. 圆方树 25
4.2.1. 广义 25
4.2.2. 仙人掌 25
4.3. 最近公共祖先 25
4.4. 树分治 25
4.4.1. 点分治 25
4.4.2. 点分树 25
4.5. 链分治 25
4.5.1. 重链分治 25
4.5.2. 长链分治 25
4.6. dsu on tree 25
5. 数学 25
5.1. 组合恒等式 25
5.2. min-max 容斥 25
5.3. 序列容斥 25
5.4. 二项式反演 25
5.5. 斯特林数 26
5.6. 高维前缀和 26
5.7. 线性基 26
5.8. 行列式 26
5.9. 高斯消元 26
6. 多项式 26
6.1. 快速数论变换 26
6.2. 快速傅里叶变换 26
6.3. 任意模数 NTT 26
6.4. 自然数幂和 26
6.5. 快速沃尔什变换 26
6.6. 子集卷积 26
7. 数据结构 26
7.1. 线段树 26
7.1.1. 李超树 (最大，次大，第三大) 26
7.1.2. 合并分裂 26
7.1.3. 线段树二分 26

7.1.4. 兔队线段树 26
7.2. 平衡树 26
7.2.1. 文艺平衡树 26
7.3. 历史版本信息线段树 26
7.4. 树状数组二分 26
7.5. 二维树状数组 26
7.6. ODT 26
7.7. KDT 26
7.8. 手写堆 26
8. 字符串 27
8.1. KMP 27
8.2. exKMP 27
8.3. SA 27
8.4. AC 自动机 27
8.5. 马拉车 27
9. 杂项 27
9.1. gcd, xor, or 分块 27
9.2. 超级钢琴 27
9.3. 平方计数 27
9.4. FFT 字符串匹配 27
9.5. 循环矩阵乘法 27
9.6. 线性逆元 27
9.7. 底数固定快速幂 27
9.8. fastio 27
9.9. 高精度 27
10. 配置相关 27
10.1. 对拍 27
10.2. vscode 配置 27

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 $a^n \equiv 1 (\operatorname{mod} m)$
- 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 $g^i \operatorname{mod} m, 0 < i < m$ 的取值各不相同。

原根的应用：m 是质数时，若求$a_k = \sum_{i*j \operatorname{mod} m=k} f_i * g_j$ 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 $a_{g^k} = \sum_{g^{i+j \operatorname{mod}(m-1)}=g^k} f_{g^i} * g_{g^j}$，令 $f_i = f_{g^i}$ 则 $a_k = \sum_{(i+j) \operatorname{mod}(m-1)=k} f_i * g_j$

```
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

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

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：$x \equiv a_2 - m_2 y_2 \pmod{\operatorname{lcm}(m_1, m_2)}$

```
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;
    if (a == 0 || b == 0) return 0;
```

```cpp
    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;
}
// m 互质
// int main() {
//     cin >> n;
//     ll phi = 1;
//     for (int i = 1; i <= n; i++) {
//         cin >> m[i] >> a[i];
//         phi *= m[i];
//     }
//     ll ans = 0;
//     for (int i = 1; i <= n; i++) {
//         ll p = phi / m[i], q = getinv(p, m[i]);
//         ans += mul(p, mul(q, a[i], phi), phi);
//         ans %= phi;
//     }
//     cout << ans << '\n';
// }
int main() {
    cin >> n;
    cin >> m >> a;
    for (int i = 2; i <= n; i++) {
        ll nm, na;
        cin >> nm >> na;
        ll x, y;
        ll g = exgcd(m, -nm, x, y), d = (na - a) / g, md = abs(nm / g);
        x = mul(x, d, md);
        ll lc = abs(m / g);
        lc *= nm;
        a = (a + mul(m, x, lc)) % lc;
        m = lc;
    }
    cout << a << '\n';
}
```

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 $\frac{n!}{n\text{以内所有}p\text{倍数的乘积}}$ 达到同样的效果。

```
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 $a^x \equiv n \pmod p$, a, p 不一定互质

```
int BSGS(int a, int b, int p) {
  unordered_map<int, int> x;
  int m = sqrt(p + 0.5);
  int v = ni(fpow(a, m), p);
  int e = 1; x[1] = 0;
  for(int i = 1; i < m; i++) {
        e = e * 1ll * a % p;
        if(!x[e]) x[e] = i;
  }
  for(int i = 0; i <= m; i++) {
    if(x[b]) return i * m + x[b];
    b = b * 1ll * v % p;
  }
  return -1;
}
int exBSGS(int a, int n, int p) {
    int d, q = 0, sum = 1;
    a %= p, n %= p;
    if(a == 1 || n == 1) return 0;
    while((d = gcd(a, p)) != 1) {
        if(n % d) return -1;
        q++; n /= d; p /= d;
        sum = (sum * 1ll * a / d) % p;
        if(sum == n) return q;
    }
    int v = ni(sum, p);
    n = n * 1ll * v % p;
    int ans = BSGS(a, n, p);
```

```
    if(ans == -1) return -1;
    return ans + q;
}
```

1.7. 二次剩余（待补）

1.8. Miller-Rabin（待补）

1.9. Pollard-rho（待补）

1.10. 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 $\gcd(n, m) = \sum_{i \mid n \land i \mid m} \varphi(i)$，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。

1.11. 莫比乌斯反演

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

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

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

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 $f(p^i) f(q^j) = f(p^i q^j)$ 即可，用线性筛积性函数也是同理。
2. 形如 $\sum_{d \mid n} \mu(d) \sum_{k \mid \frac{n}{d}} \varphi(k) \lfloor \frac{n}{dk} \rfloor$ 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。

1.12. 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

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

1.13. 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。

1.14. 杜教筛

1.15. 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 $f(p^k)$ 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

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

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

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 $n = 1 \vee p_j > n, S(n, j) = 0$

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

```
#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

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
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. SPFA 乱搞

```
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. 差分约束

3.4. 竞赛图

3.5. 有向图强连通分量

3.5.1. Tarjan

```
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);
}
```

3.5.2. Kosaraju

3.6. 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。

```
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;
        iota(fa, fa + n + 1, 0);
        fill(dep, dep + n + 1, 1);
    }
    int find(int u) {
        return u == fa[u] ? u : find(fa[u]);
    }
    void merge(int u, int v) {
        u = find(u), v = find(v);
        if (u == v) return;
        if (dep[u] > dep[v]) swap(u, v);
        stk[++top] = {u, (dep[u] == dep[v] ? v : -1)};
        fa[u] = v;
        dep[v] += (dep[u] == dep[v]);
```

```
    }
    void rev(int tim) {
        while (tim < top) {
            auto [u, v] = stk[top--];
            fa[u] = u;
            if (v != -1) dep[v]--;
        }
    }
} D;
int stk[N], top, dfc, dfn[N], low[N], in_stk[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;
        do {
            x = stk[top--];
            D.merge(x, u);
            in_stk[x] = 0;
        } while (x != u);
    }
}

void solve(int l, int r, int a, int b) {
    if (l == r) {
        for (int i = a; i <= b; i++) edge[i][3] = l;
        return;
    }
    int mid = (l + r) >> 1;
    vector<int> node;
    for (int i = a; i <= b; i++) if (edge[i][0] <= mid) {
        int u = D.find(edge[i][1]), v = D.find(edge[i][2]);
        if (u != v) node.push_back(u), node.push_back(v), G[u].push_back(v);
    }
    int otp = D.top;
    for (int x : node) if (!dfn[x]) tarjan(x);
    vector<array<int, 4>> e1, e2;
    for (int i = a; i <= b; i++) {
        int u = D.find(edge[i][1]), v = D.find(edge[i][2]);
        if (edge[i][0] > mid || u != v) e2.push_back(edge[i]);
        else e1.push_back(edge[i]);
    }
    int s1 = e1.size(), s2 = e2.size();
    for (int i = a; i < a + s1; i++) edge[i] = e1[i - a];
```

```
    for (int i = a + s1; i <= b; i++) edge[i] = e2[i - a - s1];
    dfc = 0;
    for (int x : node) dfn[x] = low[x] = 0, vector<int>().swap(G[x]);
    vector<int>().swap(node);
    vector<array<int, 4>>().swap(e1);
    vector<array<int, 4>>().swap(e2);
    solve(mid + 1, r, a + s1, b);
    D.rev(otp);
    solve(l, mid, a, a + s1 - 1);
}
```

3.7. 连通分量

3.7.1. 割点和桥

```
int dfn[N], low[N], dfs_clock;
bool iscut[N], vis[N];
void dfs(int u, int fa) {
    dfn[u] = low[u] = ++dfs_clock;
    vis[u] = 1;
    int child = 0;
    for (int v : e[u]) {
        if (v == fa) continue;
        if (!dfn[v]) {
            dfs(v, u);
            low[u] = min(low[u], low[v]);
            child++;
            if (low[v] >= dfn[u]) iscut[u] = 1;
        } else if (dfn[u] > dfn[v] && v != fa) low[u] = min(low[u],
dfn[v]);
        if (fa == 0 && child == 1) iscut[u] = 0;
    }
}
```

3.7.2. 点双

```
#include <cstdio>
#include <vector>
using namespace std;
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; }

```

```cpp
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;
}
```

3.7.3. 边双

```cpp
#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;
}
```

3.8. 二分图匹配

3.8.1. 匈牙利算法

3.8.2. KM

3.9. 网络流

3.9.1. 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);
        }
    }
```

```cpp
    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;
}

```

3.9.2. 最小费用最大流

```cpp
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);
            }
        }
    }
```

```cpp
    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;
}
```

3.9.3. 上下界网络流（待学）

3.10. 2-SAT

$2 * u$ 代表不选择，$2 * u + 1$ 代表选择。

3.10.1. 搜索 (最小字典序)

```cpp
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() {
```

```cpp
    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;
}
```

3.10.2. tarjan

如果对于一个 **x** sccno 比它的反状态 **x**∧1 的 sccno 要小，那么我们用 **x** 这个状态当做答案，否则用它的反状态当做答案。

3.11. 生成树

3.11.1. Prime

```cpp
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';
}
```

3.11.2. 次小生成树

3.11.3. 生成树计数

3.12. 三元环

3.13. 四元环

3.14. 欧拉路

3.15. 曼哈顿路

3.16. 建图优化

3.16.1. 前后缀优化

3.16.2. 线段树优化

4. 树论

4.1. prufer

4.2. 圆方树

4.2.1. 广义

4.2.2. 仙人掌

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

6. 多项式

6.1. 快速数论变换

6.2. 快速傅里叶变换

6.3. 任意模数 NTT

6.4. 自然数幂和

6.5. 快速沃尔什变换

6.6. 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

8.2. exKMP

8.3. SA

8.4. AC 自动机

8.5. 马拉车

9. 杂项

9.1. gcd, xor, or 分块

9.2. 超级钢琴

9.3. 平方计数

9.4. FFT 字符串匹配

9.5. 循环矩阵乘法

9.6. 线性逆元

9.7. 底数固定快速幂

9.8. fastio

9.9. 高精度

10. 配置相关

10.1. 对拍

10.2. vscode 配置