

widsnoy's template

1. 数论	1
1.1. 取模还原分数	1
1.2. 原根	1
1.3. 解不定方程	2
1.4. 中国剩余定理	3
1.5. 卢卡斯定理	4
1.6. BSGS	6
1.7. 数论函数	7
1.8. 莫比乌斯反演	7
1.9. 整除分块	8
1.10. 区间筛	8
1.11. 杜教筛	8
1.12. Min25 筛	8
2. 动态规划	10
2.1. 缺 1 背包	10
3. 图论	10
3.1. 找环	10
3.2. 差分约束	11
3.2.1. SPFA 乱搞	12
3.3. 竞赛图	13
3.4. 连通分量	13
3.4.1. 有向图强连通分量	13
3.4.2. 强连通分量(incremental)	13
3.4.3. 割点和桥	15
3.4.4. 点双	15
3.4.5. 边双	17
3.5. 二分图匹配	18
3.5.1. 匈牙利算法	18
3.5.2. KM	18
3.6. 网络流	18
3.6.1. 网络最大流	18
3.6.2. 最小费用最大流	19
3.7. 2-SAT	20
3.7.1. 搜索(最小字典序)	20
3.7.2. tarjan	21
3.7.3. 前后缀优化	21
3.7.4. 线段树优化	21
3.8. 生成树	21
3.8.1. Prime	21
3.9. 圆方树	22
3.10. 欧拉回路	24

3.11. 曼哈顿路	26
3.12. 无向图三/四元环计数	26
4. 树论	27
4.1. prufer	27
4.2. 虚树	27
4.3. 最近公共祖先	28
4.4. 树分治	28
4.4.1. 点分治	28
4.4.2. 点分树	28
4.5. 链分治	28
4.5.1. 重链分治	28
4.5.2. 长链分治	28
4.6. dsu on tree	28
5. 数学	28
5.1. 组合恒等式	28
5.2. min-max 容斥	29
5.3. 序列容斥	29
5.4. 二项式反演	29
5.5. 斯特林数	29
5.6. 高维前缀和	29
5.7. 线性基	29
5.8. 行列式	29
5.9. 高斯消元	29
6. 多项式	29
6.1. NTT	29
6.2. 任意模数 NTT	32
6.3. 自然数幂和	35
6.4. 快速沃尔什变换	35
6.5. 子集卷积	35
7. 数据结构	35
7.1. 线段树	35
7.1.1. 李超树 (最大, 次大, 第三大)	35
7.1.2. 合并分裂	35
7.1.3. 线段树二分	35
7.1.4. 兔队线段树	35
7.2. 平衡树	35
7.2.1. 文艺平衡树	35
7.3. 历史版本信息线段树	35
7.4. 树状数组二分	35
7.5. 二维树状数组	35
7.6. ODT	35
7.7. KDT	36

7.8. 手写堆	36
8. 字符串	36
8.1. KMP	36
8.2. Z function	36
8.3. SA	36
8.4. AC 自动机	37
8.5. Manacher	38
9. 杂项	38
9.1. fastio	38
9.2. 高精度	40
9.3. 对拍	44

1. 数论

1.1. 取模还原分数

1.2. 原根

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

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

```

1 int q[10005];
2 int getG(int n) {
3     int i, j, t = 0;
4     for (i = 2; (ll)(i * i) < n - 1; i++) {
5         if ((n - 1) % i == 0) q[t++] = i, q[t++] = (n - 1) / i;
6     }
7     for (i = 2; ; i++) {
8         for (j = 0; j < t; j++) if (fpow(i, q[j], n) == 1) break;
9         if (j == t) return i;
10    }
11    return -1;
12 }
13
14 vector<int> fpow(int kth) {
15     if (kth == 0) return e;
16     auto r = fpow(kth - 1);
17     r = multiply(r, r);
18     for (int i = p - 1; i < r.size(); i++) r[i % (p - 1)] = (r[i % (p - 1)] + r[i]) % mod;
19     r.resize(p - 1);
20     if (kk[kth] == '1') {
21         r = multiply(r, e);
22         for (int i = p - 1; i < r.size(); i++) r[i % (p - 1)] = (r[i % (p - 1)] + r[i]) % mod;
23         r.resize(p - 1);
24     }
25     return r;
26 }
27 void MAIN() {
28     g = getG(p);
29     int tmp = 1;
30     for (int i = 1; i < p; i++) {
31         tmp = tmp * 1ll * g % p;
32         mp[tmp] = i % (p - 1);
33     }
34     e.resize(p - 1);
35     for (int i = 0; i < p - 1; i++) e[i] = 0;

```

```
36     for (int i = 0; i < p; i++) {
37         for (int j = 0; j <= i; j++) {
38             if (binom[i][j] == 0) continue;
39             e[mp[binom[i][j]]]++;
40         }
41     }
42 }
```

1.3. 解不定方程

给出 $a, b, c, x_1, x_2, y_1, y_2$, 求满足 $ax+by+c=0$, 且 $x \in [x_1, x_2]$, $y \in [y_1, y_2]$ 的整数解有多少对?

输入格式

第一行包含 7 个整数, $a, b, c, x_1, x_2, y_1, y_2$, 整数间用空格隔开。

$a, b, c, x_1, x_2, y_1, y_2$ 的绝对值不超过 10^8 。

```
1 #define y1 miku
2
3 ll a, b, c, x1, x2, y1, y2;
4 ll exgcd(ll a, ll b, ll &x, ll &y) {
5     if (b) {
6         ll d = exgcd(b, a % b, y, x);
7         return y -= a / b * x, d;
8     } return x = 1, y = 0, a;
9 }
10
11 pll get_up(ll a, ll b, ll x1, ll x2) {
12     //x2>=ax+b>=x1
13     if (a == 0) return (b >= x1 && b <= x2) ? (pll){-1e18, 1e18} : (pll)
{1, 0};
14     ll L, R;
15     ll l = (x1 - b) / a - 3;
16     for (L = l; L * a + b < x1; L++);
17     ll r = (x2 - b) / a + 3;
18     for (R = r; R * a + b > x2; R--);
19     return {L, R};
20 }
21 pll get_dn(ll a, ll b, ll x1, ll x2) {
22     //x2>=ax+b>=x1
23     if (a == 0) return (b >= x1 && b <= x2) ? (pll){-1e18, 1e18} : (pll)
{1, 0};
24     ll L, R;
25     ll l = (x2 - b) / a - 3;
26     for (L = l; L * a + b > x2; L++);
27     ll r = (x1 - b) / a + 3;
28     for (R = r; R * a + b < x1; R--);
29     return {L, R};
30 }
31
32 void MAIN() {
```

```

33     cin >> a >> b >> c >> x1 >> x2 >> y1 >> y2;
34     if (a == 0 && b == 0) return cout << (c == 0) * (y2 - y1 + 1) * (x2
- x1 + 1) << '\n', void();
35     ll x, y, d = exgcd(a, b, x, y);
36     c = -c;
37     if (c % d != 0) return cout << "0\n", void();
38     x *= c / d, y *= c / d;
39     ll sx = b / d, sy = -a / d;
40     //x + k * sx = y + k * sy
41     // 0<= 3 - k <= 4 [-1,3] [0,4]
42     auto A = (sx > 0 ? get_up(sx, x, x1, x2) : get_dn(sx, x, x1, x2));
43     auto B = (sy > 0 ? get_up(sy, y, y1, y2) : get_dn(sy, y, y1, y2));
44     A.fi = max(A.fi, B.fi), A.se = min(A.se, B.se);
45     cout << max(0ll, A.se - A.fi + 1) << '\n';
46 }

```

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

$$\begin{cases} x + m_1y = a_1 \\ x + m_2y = a_2 \end{cases}$$

取解集公共部分 $x = a_1 - m_1y_1 = a_2 - m_2y_2$, 若 $\gcd(m_1, m_2) | (a_1 - a_2)$ 有解, 可以得到 $x = k\text{lcm}(m_1, m_2) + a_2 - m_2y_2$ 化为同余方程的形式: $x \equiv a_2 - m_2y_2 \pmod{\text{lcm}(m_1, m_2)}$

```

1 ll n, m, a;
2 ll exgcd(ll a, ll b, ll &x, ll &y) {
3     if (b != 0) {
4         ll g = exgcd(b, a % b, y, x);
5         return y -= a / b * x, g;
6     } return x = 1, y = 0, a;
7 }
8 ll getinv(ll a, ll mod) {
9     ll x, y;
10    exgcd(a, mod, x, y);
11    x = (x % mod + mod) % mod;
12    return x;
13 }
14 int get(ll x) {
15     return x < 0 ? -1 : 1;
16 }
17 ll mul(ll a, ll b, ll mod) {
18     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 $\text{calc}(n, x, p)$ 计算 $\frac{n!}{x^y} \pmod{p}$ 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 $(p - 1)! \equiv -1 \pmod{p}$ 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。

```

1 ll exgcd(ll a, ll b, ll &x, ll &y) {
2     if (b) {
3         ll d = exgcd(b, a % b, y, x);
4         return y -= a / b * x, d;
5     } else return x = 1, y = 0, a;
6 }
7 int getinv(ll v, ll mod) {
8     ll x, y;
9     exgcd(v, mod, x, y);
10    return (x % mod + mod) % mod;
11 }
12 ll fpow(ll a, ll b, ll p) {
13     ll res = 1;
14     for (; b; b >= 1, a = a * 1ll * a % p) if (b & 1) res = res * 1ll *
15         a % p;
16     return res;
17 }
18 ll calc(ll n, ll x, ll p) {
19     if (n == 0) return 1;
20     ll s = 1;
21     for (ll i = 1; i <= p; i++) if (i % x) s = s * i % p;
22     s = fpow(s, n / p, p);
23     for (ll i = n / p * p + 1; i <= n; i++) if (i % x) s = i % p * s % p;
24     return calc(n / x, x, p) * 1ll * s % p;
25 }
26 int get(ll x) {
27     return x < 0 ? -1 : 1;
28 }
29 ll mul(ll a, ll b, ll mod) {
30     ll res = 0;
31     if (a == 0 || b == 0) return 0;
32     ll f = get(a) * get(b);
33     a = abs(a), b = abs(b);
34     for (; b; b >= 1, a = (a + a) % mod) if (b & 1) res = (res + a) %
35         mod;
36     res *= f;
37     if (res < 0) res += mod;
38     return res;
39 }
40 ll sublucas(ll n, ll m, ll x, ll p) {
41     ll cnt = 0;
42     for (ll i = n; i; ) cnt += (i = i / x);
43     for (ll i = m; i; ) cnt -= (i = i / x);
44     for (ll i = n - m; i; ) cnt -= (i = i / x);
45     return fpow(x, cnt, p) * calc(n, x, p) % p * getinv(calc(m, x, p),
46 p) % p * getinv(calc(n - m, x, p), p) % p;

```

```
44 }
45 ll lucas(ll n, ll m, ll p) {
46     int cnt = 0;
47     ll a[21], mo[21];
48     for (ll i = 2; i * i <= p; i++) if (p % i == 0) {
49         mo[++cnt] = 1;
50         while (p % i == 0) mo[cnt] *= i, p /= i;
51         a[cnt] = sublucas(n, m, i, mo[cnt]);
52     }
53     if (p != 1) mo[++cnt] = p, a[cnt] = sublucas(n, m, p, mo[cnt]);
54     ll phi = 1;
55     for (int i = 1; i <= cnt; i++) phi *= mo[i];
56     ll ans = 0;
57     for (int i = 1; i <= cnt; i++) {
58         ll p = phi / mo[i], q = getinv(p, mo[i]);
59         ans += mul(p, mul(q, a[i], phi), phi);
60         ans %= phi;
61     }
62     return ans;
63 }
```

1.6. BSGS

求解 $a^x \equiv n \pmod{p}$, a, p 不一定互质

```
1 int fpow(int a, int b, int p) {
2     int res = 1;
3     for (; b; b >= 1, a = a * 1ll * a % p) if (b & 1) res = res * 1ll *
4         a % p;
5     return res;
6 }
7 ll exgcd(ll a, ll b, ll &x, ll &y) {
8     if (b == 0) return x = 1, y = 0, a;
9     ll d = exgcd(b, a % b, y, x);
10    y -= a / b * x;
11    return d;
12 }
13 int inv(int a, int p) {
14     ll x, y;
15     ll g = exgcd(a, p, x, y);
16     if (g != 1) return -1;
17     return (x % p + p) % p;
18 }
19 int BSGS(int a, int b, int p) {
20     if (p == 1) return 1;
21     unordered_map<int, int> x;
22     int m = sqrt(p + 0.5) + 1;
23     int v = inv(fpow(a, m, p), p);
24     int e = 1;
25     for (int i = 1; i <= m; i++) {
26         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

$$1. \varphi(n) = n \prod \left(1 - \frac{1}{p}\right)$$

$$2. \mu(n) = \begin{cases} 1, & n=1 \\ (-1)^{\text{质因子个数}}, & n \text{ 无平方因子} \\ 0, & n \text{ 有平方因子} \end{cases}$$

$$3. \mu * \text{id} = \varphi, \mu * 1 = \varepsilon, \varphi * 1 = \text{id}$$

- 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

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

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

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

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 $\sum_{d|n} \mu(d) \sum_{k|\frac{n}{d}} \varphi(k) \lfloor \frac{n}{dk} \rfloor$ 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。

1.9. 整除分块

1. 下取整

```

1 for (int i = 1, j; i <= min(n, m); i = j + 1) {
2     j = min(n / (n / i), m / (m / i));
3     // n / {i, ..., j} = n / i
4 }
```

1. 上取整

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

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 $f(p^k)$ 处的点值。

- 定义 $R(i)$ 表示 i 的最小质因子

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

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 $\text{id1}[]$ 和 $\text{id2}[]$ 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 $n = 1 \vee p_j > n, S(n, j) = 0$

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

```

1 #include <cstdio>
2 #include <cmath>
3
4 typedef long long ll;
5 const int N = 4e6 + 5, MOD = 1e9 + 7;
6 const ll i6 = 166666668, i2 = 500000004;
7 ll n, id1[N], id2[N], su1[N], su2[N], p[N], sqr, w[N], g[N], h[N];
8 int cnt, m;
9 bool vis[N];
10
11 ll add(ll a, ll b) {a %= MOD, b %= MOD; return (a + b >= MOD) ? a + b - MOD : a + b;}
12 ll mul(ll a, ll b) {a %= MOD, b %= MOD; return a * b % MOD;}
13 ll dec(ll a, ll b) {a %= MOD, b %= MOD; return ((a - b) % MOD + MOD) % MOD;}
14
15 void init(int m) {
16     for (ll i = 2; i <= m; i++) {
17         if (!vis[i]) p[++cnt] = i, su1[cnt] = add(su1[cnt - 1], i), su2[cnt] = add(su2[cnt - 1], mul(i, i));
18         for (int j = 1; j <= cnt && i * p[j] <= m; j++) {
19             vis[p[j] * i] = 1;
20             if (i % p[j] == 0) break;
21         }
22     }
23 }
24
25 ll S(ll x, int y) {
26     if (p[y] > x || x <= 1) return 0;
27     int k = (x <= sqr) ? id1[x] : id2[n / x];
28     ll res = dec(dec(g[k], h[k]), dec(su2[y - 1], su1[y - 1]));
29     for (int i = y; i <= cnt && p[i] * p[i] <= x; i++) {
30         ll pow1 = p[i], pow2 = p[i] * p[i];
31         for (int e = 1; pow2 <= x; pow1 = pow2, pow2 *= p[i], e++) {
32             ll tmp = mul(mul(pow1, dec(pow1, 1)), S(x / pow1, i + 1));
33             tmp = add(tmp, mul(pow2, dec(pow2, 1)));
34             res = add(res, tmp);
35         }
36     }
37     return res;
}

```

```
38 }
39
40 int main() {
41     scanf("%lld", &n);
42     sqr = sqrt(n + 0.5) + 1;
43     init(sqr);
44     for (ll l = 1, r; l <= n; l = r + 1) {
45         r = n / (n / l);
46         w[++m] = n / l;
47         g[m] = mul(w[m] % MOD, (w[m] + 1) % MOD);
48         g[m] = mul(g[m], (2 * w[m] + 1) % MOD);
49         g[m] = mul(g[m], i6);
50         g[m] = dec(g[m], 1);
51         h[m] = mul(w[m] % MOD, (w[m] + 1) % MOD);;
52         h[m] = mul(h[m], i2);
53         h[m] = dec(h[m], 1);
54         (w[m] <= sqr) ? id1[w[m]] = m : id2[r] = m;
55     }
56     for (int j = 1; j <= cnt; j++)
57         for (int i = 1; i <= m && p[j] * p[j] <= w[i]; i++) {
58             int k = (w[i] / p[j] <= sqr) ? id1[w[i] / p[j]] : id2[n / (w[i] / p[j])];
59             g[i] = dec(g[i], mul(mul(p[j], p[j]), dec(g[k], su2[j - 1])));
60             h[i] = dec(h[i], mul(p[j], dec(h[k], su1[j - 1])));
61         }
62     //printf("%lld\n", g[1] - h[1]);
63     printf("%lld\n", add(S(n, 1), 1));
64     return 0;
65 }
```

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。

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

3. 图论

3.1. 找环

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

```

6 void get_cyc(int u, int v) {
7     if (!resp.empty()) return;
8     vector<int> cyc;
9     cyc.push_back(v);
10    while (true) {
11        v = pre[v];
12        if (v == 0) break;
13        cyc.push_back(v);
14        if (v == u) break;
15    }
16    reverse(cyc.begin(), cyc.end());
17    resp.push_back(cyc);
18 }
19 // edge
20 void get_cyc(int u, int v, int id) {
21     if (!rese.empty()) return;
22     vector<int> cyc;
23     cyc.push_back(id);
24     while (true) {
25         if (pre[v] == 0) break;
26         cyc.push_back(pre_edg[v]);
27         v = pre[v];
28         if (v == u) break;
29     }
30     reverse(cyc.begin(), cyc.end());
31     rese.push_back(cyc);
32 }
33 void dfs(int u, int edg) {
34     col[u] = 1;
35     for (auto [v, id] : G[u]) if (id != edg) {
36         if (col[v] == 1) {
37             get_cyc(v, u);
38             get_cyc(v, u, id);
39         } else if (col[v] == 0) {
40             pre[v] = u;
41             pre_edg[v] = id;
42             dfs(v, id);
43         }
44     }
45     col[u] = 2;
46 }
47 void MAIN() {
48     cin >> n >> m;
49     for (int i = 1; i <= m; i++) {
50         int u, v; cin >> u >> v;
51         // G[u].push_back({v, i});
52         // G[v].push_back({u, i});
53     }
54     for (int i = 1; i <= n; i++) if (!col[i]) dfs(i, -1);
55 }

```

3.2. 差分约束

3.2.1. SPFA 亂搞

```

1 mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());
2
3 const int mod = 998244353;
4 const int N = 5e5 + 5;
5 const ll inf = 1e17;
6 int n, m, s, t, q[N], ql, qr;
7 int vis[N], fr[N];
8 ll dis[N];
9 vector<pii> G[N];
10 void MAIN() {
11     cin >> n >> m >> s >> t;
12     for (int i = 1; i <= m; i++) {
13         int u, v, w;
14         cin >> u >> v >> w;
15         G[u].push_back({v, w});
16     }
17     for (int i = 0; i <= n; i++) dis[i] = inf;
18     dis[s] = 0; q[qr] = s; vis[s] = 1;
19     while (ql <= qr) {
20         if (rng() % (qr - ql + 1) == 0) sort(q + ql, q + qr + 1, [] (int
21 x, int y) {
22             return dis[x] < dis[y];
23         });
24         int u = q[ql++];
25         vis[u] = 0;
26         for (auto [v, w] : G[u]) {
27             if (dis[u] + w < dis[v]) {
28                 dis[v] = dis[u] + w;
29                 fr[v] = u;
30                 if (!vis[v]) {
31                     if (ql > 0) q[--ql] = v;
32                     else q[++qr] = v;
33                     vis[v] = 1;
34                 }
35             }
36         }
37     }
38     if (dis[t] == inf) {
39         cout << "-1\n";
40         return;
41     }
42     cout << dis[t] << ' ';
43     vector<pii> stk;
44     while (t != s) {
45         stk.push_back({fr[t], t});
46         t = fr[t];
47     }
48     reverse(stk.begin(), stk.end());
49     cout << stk.size() << '\n';
50     for (auto [u, v] : stk) cout << u << ' ' << v << '\n';
51 }
```

3.3. 竞赛图

3.4. 连通分量

3.4.1. 有向图强连通分量

```

1 const int N = 5e5 + 5;
2 int n, m, dfc, dfn[N], low[N], stk[N], top, idx[N], in_stk[N], scc_cnt;
3 vector<int> G[N];
4
5 void tarjan(int u) {
6     low[u] = dfn[u] = ++dfc;
7     stk[++top] = u;
8     in_stk[u] = 1;
9     for (int v : G[u]) {
10         if (!dfn[v]) {
11             tarjan(v);
12             low[u] = min(low[u], low[v]);
13         } else if (in_stk[v]) low[u] = min(dfn[v], low[u]);
14     }
15     if (low[u] == dfn[u]) {
16         int x;
17         scc_cnt++;
18         do {
19             x = stk[top--];
20             idx[x] = scc_cnt;
21             in_stk[x] = 0;
22         } while (x != u);
23     }
24 }
25
26 void MAIN() {
27     for (int i = 1; i <= n; i++) low[i] = dfn[i] = idx[i] = in_stk[i] =
28     0;
29     dfc = scc_cnt = top = 0;
30     cin >> n >> m;
31     for (int i = 1; i <= n; i++) if (!dfn[i]) tarjan(i);
32 }
```

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。

```

1 int n, m, Q, s[N];
2 vector<array<int, 4>> edge;
3 vector<int> G[N];
4 struct DSU {
5     int fa[N], dep[N], top;
6     pii stk[N];
7     void init(int n) {
8         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 void solve(int l, int r, int a, int b) {
52     if (l == r) {
53         for (int i = a; i <= b; i++) edge[i][3] = l;
54         return;
55     }
56     int mid = (l + r) >> 1;
57     vector<int> node;
58     for (int i = a; i <= b; i++) if (edge[i][0] <= mid) {
59         int u = D.find(edge[i][1]), v = D.find(edge[i][2]);
60

```

```
61         if (u != v) node.push_back(u), node.push_back(v),
62         G[u].push_back(v);
63     }
64     int otp = D.top;
65     for (int x : node) if (!dfn[x]) tarjan(x);
66     vector<array<int, 4>> e1, e2;
67     for (int i = a; i <= b; i++) {
68         int u = D.find(edge[i][1]), v = D.find(edge[i][2]);
69         if (edge[i][0] > mid || u != v) e2.push_back(edge[i]);
70         else e1.push_back(edge[i]);
71     }
72     int s1 = e1.size(), s2 = e2.size();
73     for (int i = a; i < a + s1; i++) edge[i] = e1[i - a];
74     for (int i = a + s1; i <= b; i++) edge[i] = e2[i - a - s1];
75     dfc = 0;
76     for (int x : node) dfn[x] = low[x] = 0, vector<int>().swap(G[x]);
77     vector<int>().swap(node);
78     vector<array<int, 4>>().swap(e1);
79     vector<array<int, 4>>().swap(e2);
80     solve(mid + 1, r, a + s1, b);
81     D.rev(otp);
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],
15             dfn[v]);
16         if (fa == 0 && child == 1) iscut[u] = 0;
17     }
18 }
```

3.4.4. 点双

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

```

4 const int N = 5e5 + 5, M = 2e6 + 5;
5 int n, m;
6
7 struct edge {
8     int to, nt;
9 } e[M << 1];
10
11 int hd[N], tot = 1;
12
13 void add(int u, int v) { e[++tot] = (edge){v, hd[u]}, hd[u] = tot; }
14
15 void uadd(int u, int v) { add(u, v), add(v, u); }
16
17 int ans;
18 int dfn[N], low[N], bcc_cnt;
19 int sta[N], top, cnt;
20 bool cut[N];
21 vector<int> dcc[N];
22 int root;
23
24 void tarjan(int u) {
25     dfn[u] = low[u] = ++bcc_cnt, sta[++top] = u;
26     if (u == root && hd[u] == 0) {
27         dcc[++cnt].push_back(u);
28         return;
29     }
30     int f = 0;
31     for (int i = hd[u]; i; i = e[i].nt) {
32         int v = e[i].to;
33         if (!dfn[v]) {
34             tarjan(v);
35             low[u] = min(low[u], low[v]);
36             if (low[v] >= dfn[u]) {
37                 if (++f > 1 || u != root) cut[u] = true;
38                 cnt++;
39                 do dcc[cnt].push_back(sta[top--]);
40                 while (sta[top + 1] != v);
41                 dcc[cnt].push_back(u);
42             }
43         } else
44             low[u] = min(low[u], dfn[v]);
45     }
46 }
47
48 int main() {
49     scanf("%d%d", &n, &m);
50     int u, v;
51     for (int i = 1; i <= m; i++) {
52         scanf("%d%d", &u, &v);
53         if (u != v) uadd(u, v);
54     }
55     for (int i = 1; i <= n; i++)

```

```

56     if (!dfn[i]) root = i, tarjan(i);
57     printf("%d\n", cnt);
58     for (int i = 1; i <= cnt; i++) {
59         printf("%llu ", dcc[i].size());
60         for (int j = 0; j < dcc[i].size(); j++) printf("%d ", dcc[i][j]);
61         printf("\n");
62     }
63     return 0;
64 }
```

3.4.5. 边双

```

1 #include <algorithm>
2 #include <cstdio>
3 #include <vector>
4
5 using namespace std;
6 const int N = 5e5 + 5, M = 2e6 + 5;
7 int n, m, ans;
8 int tot = 1, hd[N];
9
10 struct edge {
11     int to, nt;
12 } e[M << 1];
13
14 void add(int u, int v) { e[++tot].to = v, e[tot].nt = hd[u], hd[u] = tot; }
15
16 void uadd(int u, int v) { add(u, v), add(v, u); }
17
18 bool bz[M << 1];
19 int bcc_cnt, dfn[N], low[N], vis_bcc[N];
20 vector<vector<int>> bcc;
21
22 void tarjan(int x, int in) {
23     dfn[x] = low[x] = ++bcc_cnt;
24     for (int i = hd[x]; i; i = e[i].nt) {
25         int v = e[i].to;
26         if (dfn[v] == 0) {
27             tarjan(v, in);
28             if (dfn[x] < low[v]) bz[i] = bz[i ^ 1] = 1;
29             low[x] = min(low[x], low[v]);
30         } else if (i != (in ^ 1))
31             low[x] = min(low[x], dfn[v]);
32     }
33 }
34
35 void dfs(int x, int id) {
36     vis_bcc[x] = id, bcc[id - 1].push_back(x);
37     for (int i = hd[x]; i; i = e[i].nt) {
38         int v = e[i].to;
```

```
39     if (vis_bcc[v] || bz[i]) continue;
40     dfs(v, id);
41   }
42 }
43
44 int main() {
45   scanf("%d%d", &n, &m);
46   int u, v;
47   for (int i = 1; i <= m; i++) {
48     scanf("%d%d", &u, &v);
49     if (u == v) continue;
50     uadd(u, v);
51   }
52   for (int i = 1; i <= n; i++)
53     if (dfn[i] == 0) tarjan(i, 0);
54   for (int i = 1; i <= n; i++)
55     if (vis_bcc[i] == 0) {
56       bcc.push_back(vector<int>());
57       dfs(i, ++ans);
58     }
59   printf("%d\n", ans);
60   for (int i = 0; i < ans; i++) {
61     printf("%llu", bcc[i].size());
62     for (int j = 0; j < bcc[i].size(); j++) printf(" %d", bcc[i][j]);
63     printf("\n");
64   }
65   return 0;
66 }
```

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

```
1 int head[N], cur[N], ecnt, d[N];
2 struct Edge {
3   int nxt, v, flow, cap;
4 }e[];
5 void add_edge(int u, int v, int flow, int cap) {
6   e[ecnt] = {head[u], v, flow, cap}; head[u] = ecnt++;
7   e[ecnt] = {head[v], u, 0, 0}; head[v] = ecnt++;
8 }
9 bool bfs() {
10   memset(vis, 0, sizeof vis);
11   std::queue<int> q;
12   q.push(s);
```

```

13     vis[s] = 1;
14     d[s] = 0;
15     while (!q.empty()) {
16         int u = q.front();
17         q.pop();
18         for (int i = head[u]; i != -1; i = e[i].nxt) {
19             int v = e[i].v;
20             if (vis[v] || e[i].flow >= e[i].cap) continue;
21             d[v] = d[u] + 1;
22             vis[v] = 1;
23             q.push(v);
24         }
25     }
26     return vis[t];
27 }
28 int dfs(int u, int a) {
29     if (u == t || !a) return a;
30     int flow = 0, f;
31     for (int& i = cur[u]; i != -1; i = e[i].nxt) {
32         int v = e[i].v;
33         if (d[u] + 1 == d[v] && (f = dfs(v, std::min(a, e[i].cap -
e[i].flow))) > 0) {
34             e[i].flow += f;
35             e[i ^ 1].flow -= f;
36             flow += f;
37             a -= f;
38             if (!a) break;
39         }
40     }
41     return flow;
42 }
43

```

3.6.2. 最小费用最大流

```

1 const int inf = 1e9;
2 int head[N], cur[N], ecnt, dis[N], s, t, n, m, mincost;
3 bool vis[N];
4 struct Edge {
5     int nxt, v, flow, cap, w;
6 }e[100002];
7 void add_edge(int u, int v, int flow, int cap, int w) {
8     e[ecnt] = {head[u], v, flow, cap, w}; head[u] = ecnt++;
9     e[ecnt] = {head[v], u, flow, 0, -w}; head[v] = ecnt++;
10 }
11 bool spfa(int s, int t) {
12     std::fill(vis + s, vis + t + 1, 0);
13     std::fill(dis + s, dis + t + 1, inf);
14     std::queue<int> q;
15     q.push(s);
16     dis[s] = 0;

```

```

17     vis[s] = 1;
18     while (!q.empty()) {
19         int u = q.front();
20         q.pop();
21         vis[u] = 0;
22         for (int i = head[u]; i != -1; i = e[i].nxt) {
23             int v = e[i].v;
24             if (e[i].flow < e[i].cap && dis[u] + e[i].w < dis[v]) {
25                 dis[v] = dis[u] + e[i].w;
26                 if (!vis[v]) vis[v] = 1, q.push(v);
27             }
28         }
29     }
30     return dis[t] != inf;
31 }
32 int dfs(int u, int a) {
33     if (vis[u]) return 0;
34     if (u == t || !a) return a;
35     vis[u] = 1;
36     int flow = 0, f;
37     for (int& i = cur[u]; i != -1; i = e[i].nxt) {
38         int v = e[i].v;
39         if (dis[u] + e[i].w == dis[v] && (f = dfs(v, std::min(a,
e[i].cap - e[i].flow))) > 0) {
40             e[i].flow += f;
41             e[i ^ 1].flow -= f;
42             flow += f;
43             mincost += e[i].w * f;
44             a -= f;
45             if (!a) break;
46         }
47     }
48     vis[u] = 0;
49     return flow;
50 }
```

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)

```

1 vector<int> G[N * 2];
2 bool mark[N * 2];
3 int stk[N], top;
4 void build_G() {
5     for (int i = 1; i <= n; i++) {
6         int u, v;
7         G[2 * u + 1].push_back(2 * v);
8         G[2 * v + 1].push_back(2 * u);
9     }
10 }
```

```
11 bool dfs(int u) {
12     if (mark[u ^ 1]) return false;
13     if (mark[u]) return true;
14     mark[u] = 1;
15     stk[++top] = u;
16     for (int v : G[u]) {
17         if (!dfs(v)) return false;
18     }
19     return true;
20 }
21 bool 2_sat() {
22     for (int i = 1; i <= n; i++) {
23         if (!mark[i * 2] && !mark[i * 2 + 1]) {
24             top = 0;
25             if (!dfs(2 * i)) {
26                 while (top) mark[stk[top--]] = 0;
27                 if (!dfs(2 * i + 1)) return 0;
28             }
29         }
30     }
31     return 1;
32 }
```

3.7.2. tarjan

如果对于一个 x $sccno$ 比它的反状态 $x \wedge 1$ 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。

3.7.3. 前后缀优化

3.7.4. 线段树优化

3.8. 生成树

3.8.1. Prime

```
1 int n, m;
2 vector<pii> G[N];
3 ll dis[N];
4 int vis[N];
5 void MAIN() {
6     cin >> n >> m;
7     for (int i = 1; i <= m; i++) {
8         int u, v, w;
9         cin >> u >> v >> w;
10        G[u].push_back({v, w});
11        G[v].push_back({u, w});
12    }
13    for (int i = 1; i <= n; i++) dis[i] = 1e18, vis[i] = 0;
14    priority_queue<pair<ll, int>> q;
15    dis[1] = 0;
16    q.push({-dis[1], 1});
```

```
17     ll ans = 0;
18     while (!q.empty()) {
19         auto [val, u] = q.top(); q.pop();
20         if (vis[u]) continue;
21         vis[u] = 1;
22         ans -= val;
23         for (auto [v, w] : G[u]) if (dis[v] > w) {
24             dis[v] = w;
25             q.push({-w, v});
26         }
27     }
28     cout << ans << '\n';
29 }
```

3.9. 圆方树

记得开两倍空间。

```
1 void tarjan(int u) {
2     stk[++top] = u;
3     low[u] = dfn[u] = ++dfc;
4     for (int v : G[u]) {
5         if (!dfn[v]) {
6             tarjan(v);
7             low[u] = min(low[u], low[v]);
8             if (low[v] == dfn[u]) {
9                 cnt++;
10                for (int x = 0; x != v; --top) {
11                    x = stk[top];
12                    T[cnt].push_back(x);
13                    T[x].push_back(cnt);
14                    val[cnt]++;
15                }
16                T[cnt].push_back(u);
17                T[u].push_back(cnt);
18                val[cnt]++;
19            }
20        } else low[u] = min(low[u], dfn[v]);
21    }
22 }
23 // 调用
24 cnt = n;
25 for (int i = 1; i <= n; i++) if (!dfn[i]) {
26     tarjan(i);
27     --top;
28 }
```

- 静态仙人掌最短路。边权设置为到点双顶点的最短距离。

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

```

3     dfn[u] = low[u] = ++dfc;
4     for (auto [v, w] : G[u]) if (!dfn[v]) {
5         dis[v] = dis[u] + w;
6         tarjan(v);
7         low[u] = min(low[u], low[v]);
8         if (low[v] == dfn[u]) {
9             ++cnt;
10            val[cnt] = cyc[stk[top]] + dis[stk[top]] - dis[u];
11            for (int x = 0; x != v; --top) {
12                x = stk[top];
13                //assert(val[cnt] >= (dis[x] - dis[u]));
14                int w = min(dis[x] - dis[u], val[cnt] - (dis[x] -
15                dis[u]));
16                T[cnt].push_back({x, w});
17                T[x].push_back({cnt, w});
18            }
19            T[cnt].push_back({u, 0});
20            T[u].push_back({cnt, 0});
21        }
22    } else if (dfn[v] < dfn[u]) {
23        cyc[u] = w;
24        low[u] = min(low[u], dfn[v]);
25    }
26
27 void dfs(int u, int fa) {
28     faz[0][u] = fa;
29     for (int k = 1; k < M; k++) faz[k][u] = faz[k - 1][faz[k - 1][u]];
30     for (auto [v, w] : T[u]) if (v != fa) {
31         dep[v] = dep[u] + 1;
32         ff[v] = ff[u] + w;
33         dfs(v, u);
34     }
35 }
36 int dist(int u, int v) {
37     int tu = u, tv = v;
38     if (dep[u] < dep[v]) swap(u, v);
39     int det = dep[u] - dep[v];
40     for (int k = 0; k < M; k++) if ((det >> k) & 1) u = faz[k][u];
41     int lca;
42     if (u == v) lca = u;
43     else {
44         for (int k = M - 1; k >= 0; k--) if (faz[k][u] != faz[k][v]) {
45             u = faz[k][u]; v = faz[k][v];
46         }
47         lca = faz[0][u];
48     }
49     if (lca <= n) return ff[tu] + ff[tv] - ff[lca] * 2;
50     int tm = min(abs(dis[u] - dis[v]), val[lca] - abs(dis[u] - dis[v]));
51     return ff[tu] - ff[u] + ff[tv] - ff[v] + tm;
52 }
```

- 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。

```

1 void dfs(int u, int fa) {
2     for (int v : T[u]) if (v != fa) dfs(v, u);
3     if (u <= n) {
4         int mx = 0;
5         /*
6             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],
7             dp[1])
8             hack: 4 4
9             1 2
10            2 3
11            3 4
12            4 2
13            */
14         for (int v : T[u]) if (v != fa) {
15             dp[u][1] += dp[v][1];
16             mx = max(mx, dp[v][0] - dp[v][1]);
17             dp[u][0] += dp[v][1];
18         }
19     } else {
20         int sum = 1;
21         dp[u][1] = 1;
22         for (int v : T[u]) if (v != fa) {
23             dp[u][1] += dp[v][1] + 1;
24             dp[u][0] = max(dp[u][0], sum + dp[v][0]);
25             sum += dp[v][1] + 1;
26         }
27         sum = 1;
28         reverse(T[u].begin(), T[u].end());
29         for (int v : T[u]) if (v != fa) {
30             dp[u][0] = max(dp[u][0], sum + dp[v][0]);
31             sum += dp[v][1] + 1;
32         }
33         if (val[u] == 2) dp[u][1] = 0;
34     }
35 }
```

3.10. 欧拉回路

- 有向图

```

1 void dfs(int u) {
2     for (int &i = hd[u]; i < G[u].size(); ) dfs(G[u][i++]);
3     stk.push_back(u);
4 }
5 int check() {
6     int mo = 0, le = 0, st = 1;
```

```

7     for (int i = 1; i <= n; i++) {
8         if (abs(in[i] - out[i]) > 1) return -1;
9         if (in[i] > out[i]) le++;
10        if (in[i] < out[i]) mo++, st = i;
11    }
12    if (mo > 1 || le > 1 || mo + le == 1) return -1;
13    return st;
14 }
15
16 void MAIN() {
17     cin >> n >> m;
18     for (int i = 1; i <= m; i++) {
19         int u, v;
20         cin >> u >> v;
21         in[v]++;
22         out[u]++;
23         G[u].push_back(v);
24     }
25     for (int i = 1; i <= n; i++) sort(G[i].begin(), G[i].end());
26     int tmp = check();
27     if (tmp == -1) cout << "No\n";
28     else {
29         dfs(tmp);
30         copy(stk.rbegin(), stk.rend(), ostream_iterator<int>(cout, "
"));
31         cout << '\n';
32     }
}

```

- 无向图

```

1 void dfs(int u) {
2     for (int &i = hd[u]; i < G[u].size(); ) {
3         while (i < G[u].size() && cnt[u][G[u][i]] == 0) ++i;
4         if (i == G[u].size()) break;
5         cnt[u][G[u][i]]--;
6         cnt[G[u][i]][u]--;
7         dfs(G[u][i++]);
8     }
9     stk.push_back(u);
10 }
11 int check() {
12     int odd = 0, st = -1;
13     for (int i = 1; i <= n; i++) {
14         if (deg[i] == 0) continue;
15         if (st == -1) st = i;
16         if (deg[i] & 1) {
17             ++odd;
18             if (odd == 1) st = i;
19         }
20     }
21     if (odd > 2) return -1;
}

```

```
22     return st;
23 }
24
25 void MAIN() {
26     n = 500;
27     cin >> m;
28     for (int i = 1; i <= m; i++) {
29         int u, v;
30         cin >> u >> v;
31         ++deg[u]; ++deg[v];
32         G[u].push_back(v);
33         G[v].push_back(u);
34         ++cnt[u][v];
35         ++cnt[v][u];
36     }
37     for (int i = 1; i <= n; i++) sort(G[i].begin(), G[i].end());
38     int tmp = check();
39     if (tmp == -1) cout << "No\n";
40     else {
41         dfs(tmp);
42         copy(stk.rbegin(), stk.rend(), ostream_iterator<int>(cout,
43             "\n"));
44     }
}
```

3.11. 曼哈顿路

3.12. 无向图三/四元环计数

- 三元环

```
1 int vis[N];
2 vector<int> G[N];
3 ll main() {
4     ll cnt = 0;
5     for (int i = 0; i < m; i++) {
6         if (deg[ed[i].fi] == deg[ed[i].se] && ed[i].fi > ed[i].se)
7             swap(ed[i].fi, ed[i].se);
8         G[ed[i].fi].push_back(ed[i].se);
9     }
10    for (int u = 1; u <= n; u++) {
11        for (int v : G[u]) vis[v] = 1;
12        for (int v : G[u]) for (int w : G[v]) if (vis[w]) ++cnt;
13        for (int v : G[u]) vis[v] = 0;
14    }
15    return cnt;
16 }
```

- 四元环

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

```
1 int n, m, deg[N], cnt[N];
2 bool bigger(int a, int b) {
3     return deg[a] > deg[b] || (deg[a] == deg[b] && a > b);
4 }
5 void MAIN() {
6     cin >> n >> m;
7     for (int i = 1; i <= m; i++) {
8         int u, v;
9         cin >> u >> v;
10        ed.push_back({u, v});
11        G[u].push_back(v);
12        G[v].push_back(u);
13        ++deg[u]; ++deg[v];
14    }
15    for (auto [u, v] : ed) {
16        if (bigger(v, u)) swap(u, v);
17        T[u].push_back(v);
18    }
19    ll ans = 0;
20    for (int a = 1; a <= n; a++) {
21        for (int b : T[a]) {
22            for (int c : G[b]) {
23                if (c == a || bigger(c, a)) continue;
24                ans += cnt[c];
25                ++cnt[c];
26            }
27        }
28        for (int b : T[a]) for (int c : G[b]) cnt[c] = 0;
29    }
30    cout << ans << '\n';
31 }
```

4. 树论

4.1. prufer

4.2. 虚树

需要保证 $\text{LCA}(0, u) = 0$

```
1 int solve(vector<int>po) {
2     sort(po.begin(), po.end(), [](int x, int y) {
3         return dfn[x] < dfn[y];
4     });
5     int ans = 0;
6     top = 0;
7     stk[++top] = 0;
8     for (int u : po) {
9         int lca = LCA(u, stk[top]);
10        if (lca == stk[top]) stk[++top] = u;
11        else {
```

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

6. 多项式

6.1. NTT

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 typedef vector<int> poly;
5 const int mod = 998244353;
6 const int N = 4000000 + 5;
7
8 int rf[32][N];
9 int fpow(int a, int b) {
10     int res = 1;
11     for (; b; b >= 1, a = a * 1ll * a % mod) if (b & 1)
12         res = res * 1ll * a % mod;
13     return res;
14 }
15 void init(int n) {
16     assert(n < N);
17     int lg = __lg(n);
18     static vector<bool> bt(32, 0);
19     if (bt[lg] == 1) return;
20     bt[lg] = 1;
21     for (int i = 0; i < n; i++) rf[lg][i] = (rf[lg][i >> 1] >> 1) + ((i
22     & 1) ? (n >> 1) : 0);
23 }
24 void ntt(poly &x, int lim, int op) {
25     int lg = __lg(lim), gn, g, tmp;;
26     for (int i = 0; i < lim; i++) if (i < rf[lg][i]) swap(x[i], x[rf[lg]
27     [i]]);
28     for (int len = 2; len <= lim; len <= 1) {
29         int k = (len >> 1);
30         gn = fpow(3, (mod - 1) / len);
31         for (int i = 0; i < lim; i += len) {
```

```

30         g = 1;
31         for (int j = 0; j < k; j++, g = gn * 1ll * g % mod) {
32             tmp = x[i + j + k] * 1ll * g % mod;
33             x[i + j + k] = (x[i + j] - tmp + mod) % mod;
34             x[i + j] = (x[i + j] + tmp) % mod;
35         }
36     }
37 }
38 if (op == -1) {
39     reverse(x.begin() + 1, x.begin() + lim);
40     int inv = fpow(lim, mod - 2);
41     for (int i = 0; i < lim; i++) x[i] = x[i] * 1ll * inv % mod;
42 }
43 }
44 poly multiply(const poly &a, const poly &b) {
45     assert(!a.empty() && !b.empty());
46     int lim = 1;
47     while (lim + 1 < int(a.size() + b.size())) lim <= 1;
48     init(lim);
49     poly pa = a, pb = b;
50     while (pa.size() < lim) pa.push_back(0);
51     while (pb.size() < lim) pb.push_back(0);
52     ntt(pa, lim, 1); ntt(pb, lim, 1);
53     for (int i = 0; i < lim; i++) pa[i] = pa[i] * 1ll * pb[i] % mod;
54     ntt(pa, lim, -1);
55     while (int(pa.size()) + 1 > int(a.size() + b.size())) pa.pop_back();
56     return pa;
57 }
58 poly prod_poly(const vector<poly>& vec) { // init vector, too slow
59     int n = vec.size();
60     auto calc = [&](const auto &self, int l, int r) -> poly {
61         if (l == r) return vec[l];
62         int mid = (l + r) >> 1;
63         return multiply(self(self, l, mid), self(self, mid + 1, r));
64     };
65     return calc(calc, 0, n - 1);
66 }
67
68 // Semi-Online-Convolution
69 poly semi_online_convolution(const poly& g, int n, int op = 0) {
70     assert(n == g.size());
71     poly f(n, 0);
72     f[0] = 1;
73     auto CDQ = [&](const auto &self, int l, int r) -> void {
74         if (l == r) {
75             // exp
76             if (op == 1 && l > 0) f[l] = f[l] * 1ll * fpow(l, mod - 2) %
77             mod;
78             return;
79         }
80         int mid = (l + r) >> 1;
81         self(self, l, mid);

```

```

81     poly a, b;
82     for (int i = l; i <= mid; i++) a.push_back(f[i]);
83     for (int i = 0; i <= r - l - 1; i++) b.push_back(g[i + 1]);
84     a = multiply(a, b);
85     for (int i = mid + 1; i <= r; i++) f[i] = (f[i] + a[i - l - 1]) %
86     mod;
87     self(self, mid + 1, r);
88   };
89   CDQ(CDQ, 0, n - 1);
90   return f;
91 }
92
93 poly getinv(const poly &a) {
94   assert(!a.empty());
95   poly res = {fpow(a[0], mod - 2)}, na = {a[0]};
96   int lim = 1;
97   while (lim < int(a.size())) lim <= 1;
98   for (int len = 2; len <= lim; len <= 1) {
99     while (na.size() < len) {
100       int tmp = na.size();
101       if (tmp < a.size()) na.push_back(a[tmp]);
102       else na.push_back(0);
103     }
104     auto tmp = multiply(na, res);
105     for (auto &x : tmp) x = (x > 0 ? mod - x : x);
106     tmp[0] = ((tmp[0] + 2) >= mod) && (tmp[0] -= mod);
107     tmp = multiply(res, tmp);
108     while (tmp.size() > len) tmp.pop_back();
109     res = tmp;
110   }
111   while (res.size() > a.size()) res.pop_back();
112   return res;
113 }
114 poly exp(const poly &g) {
115   int n = g.size();
116   poly b(n, 0);
117   for (int i = 1; i < n; i++) b[i] = i * 1ll * g[i] % mod;
118   return semi_online_convolution(b, n, 1);
119 }
120 poly ln(const poly &A) {
121   int n = A.size();
122   auto C = getinv(A);
123   poly A1(n, 0);
124   for (int i = 0; i < n - 1; i++) A1[i] = (i + 1) * 1ll * A[i + 1] %
125   mod;
126   C = multiply(C, A1);
127   for (int i = n - 1; i > 0; i--) C[i] = C[i - 1] * 1ll * fpow(i, mod -
128   2) % mod;
129   C[0] = 0;
130   while (C.size() > n) C.pop_back();
131   return C;
132 }
```

```

130 poly quick_pow(poly &a, int k, int k_mod_phi, bool is_k_bigger_than_mod
131   = false) {
132     assert(!a.empty());
133     int n = a.size(), t = -1, b;
134     for (int i = 0; i < n; i++) if (a[i]) {
135       t = i, b = a[i];
136       break;
137     }
138     if (t == -1 || t && is_k_bigger_than_mod || k * 1ll * t >= n) return
139     poly(n, 0);
140     poly f;
141     for (int i = 0; i < n; i++) {
142       if (i + t < n) f.push_back(a[i + t] * 1ll * fpow(b, mod - 2) %
143         mod);
144       else f.push_back(0);
145     }
146     f = ln(f);
147     for (auto &x : f) x = x * 1ll * k % mod;
148     f = exp(f);
149     poly res;
150     for (int i = 0; i < k * t; i++) res.push_back(0);
151     int fb = fpow(b, k_mod_phi);
152     for (int i = k * t; i < n; i++) res.push_back(f[i - k * t] * 1ll *
153       fb % mod);
154     return res;
155   }
156
157   int main() {
158     ios::sync_with_stdio(0); cin.tie(0);
159     int n, k = 0, k_mod_phi = 0, isb = 0;
160     string s;
161     cin >> n >> s;
162     for (auto ch : s) {
163       if ((ch - '0') + k * 10ll >= mod) isb = 1;
164       k = ((ch - '0') + k * 10ll) % mod;
165       k_mod_phi = ((ch - '0') + k_mod_phi * 10ll) % 998244352;
166     }
167     poly a(n);
168     for (auto &x : a) cin >> x;
169     a = quick_pow(a, k, k_mod_phi, isb);
170     while (a.size() > n) a.pop_back();
171     for (auto x : a) cout << x << ' ';
172     return 0;
173   }

```

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
19     & 1) ? (n >> 1) : 0);
20 }
21 void fft(poly &x, int lim, int op) {
22     int lg = __lg(lim);
23     for (int i = 0; i < lim; i++) if (i < rf[lg][i]) swap(x[i], x[rf[lg]
24     [i]]);
25     for (int len = 2; len <= lim; len <= 1) {
26         int k = (len >> 1);
27         for (int i = 0; i < lim; i += len) {
28             for (int j = 0; j < k; j++) {
29                 cp w(cos(pi * j / k), op * sin(pi * j / k));
30                 cp tmp = w * x[i + j + k];
31                 x[i + j + k] = x[i + j] - tmp;
32                 x[i + j] = x[i + j] + tmp;
33             }
34         }
35     }
36     if (op == -1) for (int i = 0; i < lim; i++) x[i] /= lim;
37 }
38 poly multiply(const poly &a, const poly &b) {
39     assert(!a.empty() && !b.empty());
40     int lim = 1;
41     while (lim + 1 < int(a.size() + b.size())) lim <= 1;
42     init(lim);
43     poly pa = a, pb = b;
44     pa.resize(lim);
45     pb.resize(lim);
46     for (int i = 0; i < lim; i++) pa[i] = (cp){pa[i].real(),
47     pb[i].real()};
48     fft(pa, lim, 1);
49     pb[0] = conj(pa[0]);
50     for (int i = 1; i < lim; i++) pb[lim - i] = conj(pa[i]);
51     for (int i = 0; i < lim; i++) {
52         pa[i] = (pa[i] + pb[i]) * (pa[i] - pb[i]) / cp({0, 4});
53     }
54     fft(pa, lim, -1);

```

```

52     pa.resize(int(a.size() + b.size()) - 1);
53     return pa;
54 }
55 vector<int> MTT(const vector<int> &a, const vector<int> &b, const int
mod) {
56     const int B = (1 << 15) - 1, M = (1 << 15);
57     int lim = 1;
58     while (lim + 1 < int(a.size() + b.size())) lim <= 1;
59     init(lim);
60     poly pa(lim), pb(lim);
61     auto get = [] (const vector<int> &v, int pos) -> int {
62         if (pos >= v.size()) return 0;
63         else return v[pos];
64     };
65     for (int i = 0; i < lim; i++) pa[i] = (cp){get(a, i) >> 15, get(a,
i) & B};
66     fft(pa, lim, 1);
67     pb[0] = conj(pa[0]);
68     for (int i = 1; i < lim; i++) pb[lim - i] = conj(pa[i]);
69     poly A0(lim), A1(lim);
70     for (int i = 0; i < lim; i++) {
71         A0[i] = (pa[i] + pb[i]) / (cp){2, 0};
72         A1[i] = (pa[i] - pb[i]) / (cp){0, 2};
73     }
74     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 * lll * M2 % mod + a1 + (b0 + b1) % mod * lll * 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

```
1 int n = strlen(s + 1);
2 for (int i = 2; i <= n; i++) {
3     int j = k[i - 1];
4     while (j != 0 && s[i] != s[j + 1]) j = k[j];
5     if (s[i] == s[j + 1]) k[i] = j + 1;
6     else k[i] = 0;
7 }
```

8.2. Z function

```
1 for (int i = 2, l = 0, r = 0; i <= n; i++) {
2     if (r >= i && r - i + 1 > z[i - l + 1]) {
3         z[i] = z[i - l + 1];
4     } else {
5         z[i] = max(0, r - i + 1);
6         while (z[i] < n - i + 1 && s[z[i] + 1] == s[i + z[i]]) ++z[i];
7     }
8     if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
9 }
```

8.3. SA

```
1 int sa[N], ork[N], rk[N], cnt[N], id[N], h[N], M, n;
2 char s[N];
3 int mn[22][N];
4 int lcp(int a, int b) {
5     if (a == b) return n - a + 1;
6     if (rk[a] > rk[b]) swap(a, b);
7     int l = rk[a] + 1, r = rk[b];
8     int len = r - l + 1, k = __lg(len);
9     return min(mn[k][l], mn[k][r - (1 << k) + 1]);
10 }
11 void MAIN() {
12     scanf("%s", s + 1);
13     n = strlen(s + 1);
14     for (int i = 1; i <= n; i++) M = max(M, (int)s[i]);
15     for (int i = 1; i <= n; i++) if ((int)(s[i]) > M) M = (int)(s[i]);
16     for (int i = 1; i <= n; i++) cnt[rk[i]] = s[i]++;
17     for (int i = 0; i <= M; i++) cnt[i] += cnt[i - 1];
18     for (int i = n; i; i--) sa[cnt[rk[i]]--] = i;
19     for (int w = 1, p; w < n; w <= 1, M = p) {
20         p = 0;
```

```

21     for (int i = n; i > n - w; i--) id[++p] = i;
22     for (int i = 1; i <= n; i++) if (sa[i] > w) id[++p] = sa[i] - w;
23     for (int i = 0; i <= M; i++) cnt[i] = 0;
24     for (int i = 1; i <= n; i++) cnt[rk[i]]++;
25     for (int i = 1; i <= M; i++) cnt[i] += cnt[i - 1];
26     for (int i = n; i; i--) sa[cnt[rk[id[i]]]--] = id[i];
27     p = 0;
28     for (int i = 0; i <= n; i++) ork[i] = rk[i];
29     for (int i = 1; i <= n; i++) {
30         if (ork[sa[i]] == ork[sa[i - 1]] && ork[sa[i] + w] ==
31             ork[sa[i - 1] + w]) rk[sa[i]] = p;
32         else rk[sa[i]] = ++p;
33     }
34     if (p == n) break;
35 }
36     for (int i = 1, k = 0; i <= n; i++) {
37         if (rk[i] == 1) continue;
38         if (k) k--;
39         while (s[i + k] == s[sa[rk[i] - 1] + k]) k++;
40         h[rk[i]] = k;
41     }
42     for (int i = 1; i <= n; i++) mn[0][i] = h[i];
43     for (int j = 1; j < 22; j++) {
44         for (int i = 1; i <= n; i++) {
45             mn[j][i] = min(mn[j - 1][i], mn[j - 1][min(n, i + (1 << (j -
1)))]);
46         }
47     }

```

8.4. AC 自动机

```

1 namespace AC {
2     int ch[N][26], tot, fail[N], e[N];
3     void insert(const char *s) {
4         int u = 0, n = strlen(s + 1);
5         for (int i = 1; i <= n; i++) {
6             if (!ch[u][s[i] - 'a']) ch[u][s[i] - 'a'] = ++tot;
7             u = ch[u][s[i] - 'a'];
8         }
9         e[u] += 1;
10    }
11    void build() {
12        queue<int> q;
13        for (int i = 0; i <= 25; i++) if (ch[0][i]) q.push(ch[0][i]);
14        while (!q.empty()) {
15            int now = q.front(); q.pop();
16            for (int i = 0; i < 26; i++) {
17                if (ch[now][i]) fail[ch[now][i]] = ch[fail[now]][i],
18                    q.push(ch[now][i]);
19                else ch[now][i] = ch[fail[now]][i];
20            }
21        }
22    }
23 }

```

```

19     }
20 }
21 }
22 int query(const char *s) {
23     int u = 0, n = strlen(s + 1), res = 0;
24     for (int i = 1; i <= n; i++) {
25         u = ch[u][s[i] - 'a'];
26         for (int j = u; j && e[j] != -1; j = fail[j]) {
27             res += e[j];
28             e[j] = -1;
29         }
30     }
31     return res;
32 }
33 }
```

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.

```

1 int n, d[N * 2];
2 char s[N];
3
4 for (int i = 1; i <= n; i++) t[i * 2] = s[i], t[i * 2 - 1] = '#';
5 t[n * 2 + 1] = '#';
6 m = n * 2 + 1;
7 for (int i = 1, l = 0, r = 0; i <= m; i++) {
8     int k = i <= r ? min(d[r - i + l], r - i + 1) : 1;
9     while (i + k <= m && i - k >= 1 && t[i + k] == t[i - k]) k++;
10    d[i] = k--;
11    if (i + k > r) r = i + k, l = i - k;
12 }
```

9. 杂项

9.1. fastio

来自 oiwiki

```

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

```

9   IO() : p1(buf), p2(buf), pp(pbuf) {}
10
11  ~IO() { fwrite(pbuf, 1, pp - pbuf, stdout); }
12 #endif
13  char gc() {
14 #if DEBUG // 调试, 可显示字符
15     return getchar();
16 #endif
17     if (p1 == p2) p2 = (p1 = buf) + fread(buf, 1, MAXSIZE, stdin);
18     return p1 == p2 ? ' ' : *p1++;
19 }
20
21  bool blank(char ch) {
22     return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t';
23 }
24
25  template <class T>
26  void read(T &x) {
27     double tmp = 1;
28     bool sign = false;
29     x = 0;
30     char ch = gc();
31     for (; !isdigit(ch); ch = gc())
32         if (ch == '-') sign = 1;
33     for (; isdigit(ch); ch = gc()) x = x * 10 + (ch - '0');
34     if (ch == '.')
35         for (ch = gc(); isdigit(ch); ch = gc())
36             tmp /= 10.0, x += tmp * (ch - '0');
37     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 =
54         pbuf;
55     *pp++ = c;
56 #endif
57 }
58
59  template <class T>
60  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 {
9     int a[MAXSIZE], len;
10    bool flag; // 标记符号'-' 
11
12    Big() {
13        len = 1;
14        memset(a, 0, sizeof a);
15        flag = false;
16    }
17
18    Big(const int);
19    Big(const char* );
20    Big(const Big& );
21    Big& operator=(const Big& );
22    Big operator+(const Big& ) const;
23    Big operator-(const Big& ) const;
24    Big operator*(const Big& ) const;
25    Big operator/(const int& ) const;
26    // TODO: Big / Big;
27    Big operator^(const int& ) const;
28    // TODO: Big ^ Big;
29
30    // TODO: Big 位运算;
31
```

```

32     int operator%(const int&) const;
33     // TODO: Big ^ Big;
34     bool operator<(const Big&) const;
35     bool operator<(const int& t) const;
36     void print() const;
37 };
38
39 Big::Big(const int b) {
40     int c, d = b;
41     len = 0;
42     // memset(a, 0, sizeof a);
43     CLR(a);
44     while (d > MAXN) {
45         c = d - (d / (MAXN + 1)) * (MAXN + 1));
46         d = d / (MAXN + 1);
47         a[len++] = c;
48     }
49     a[len++] = d;
50 }
51
52 Big::Big(const char* s) {
53     int t, k, index, l;
54     CLR(a);
55     l = strlen(s);
56     len = l / DLEN;
57     if (l % DLEN) ++len;
58     index = 0;
59     for (int i = l - 1; i >= 0; i -= DLEN) {
60         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 - 1];
119             // WTF?
120             while (j > i) t1.a[j - 1] += 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. 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
```