widsnoy's template

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;
    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;
}
// 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 $\frac{n!}{n以内所有p倍数的乘积}$ 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

```cpp
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 $\gcd(n,m) = \sum_{i \mid n \wedge i \mid m} \varphi(i)$，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 $f(n)=\sum_{n|d} g(d)$, 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。

1.12. 整除分块

1. 下取整

```cpp
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

2. 动态规划

2.1. 缺 1 背包

3. 图论

3.1. 找环

```cpp
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