competitive-programing

UnionFind

▶

最初n個の無関係な点を関係性ごとにつないでいける

友達の友達は友達

https://atcoder.jp/contests/arc065/tasks/arc065_b

UnionFind 2個用意して、道路と鉄道の情報を管理して、

n個の点それぞれの、道路のrootと鉄道のrootを出して、mapでその組み合わせの

出現回数を数える

BFS

▶

幅優先探索

全ての点から点が等距離の場合は、最短距離を求めることができる

https://atcoder.jp/contests/arc005/tasks/arc005_3

'.'は通れて、'#'は通れないが、二回までなら'#'を通っても良い

'.'の時、cost=0,'#'の時、cost=1として、それぞれのますに行くためのcostの最小を更新していく。

cost(gx,gy)<=2ならok

▶ コード

```
#include <bits/stdc++.h>
#define rep(i,n)for(int i=0;i<(n);i++)
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const long long INF = 1LL<<60;
#define rev(s) (string((s).rbegin(), (s).rend()))

int h,w,sy,sx,gy,gx;;
const int inf = 300000;
vector<string> field(510);
int dx[] = {1,0,-1,0};
int dy[] = {0,1,0,-1};
vector<vector<int>> cost(510,vector<int>(510,inf));
```

```
void bfs() {
    int cs;
    queue<P> q;
    q.push(P(sy,sx));
    cost[sy][sx]=0;
    while (q.size()) {
        P pa = q.front();q.pop();
        rep(i,4) {
            int ny = pa.first+dy[i];
            int nx = pa.second+dx[i];
            if (ny>=0&&ny<h&&nx>=0&&nx<w) {
                if (field[ny][nx]=='#') {
                    cs=1;
                }
                else {
                    cs=0;
                }
                if (cost[ny][nx]>cost[pa.first][pa.second]+cs) {
                    cost[ny][nx]=cost[pa.first][pa.second]+cs;
                    q.push(P(ny,nx));
                }
            }
        }
    }
}


int main() {
    cin>>h>>w;
    rep(i,h) {
        cin>>field[i];
    }
    rep(i,h) {
        rep(j,w) {
            if (field[i][j]=='s') {
                sy=i;sx=j;
            }if (field[i][j]=='g') {
                gy=i;gx=j;
            }
        }
    }
    bfs();
    if (cost[gy][gx]<=2) {
        cout<<"YES"<<endl;
    }
    else {
        cout<<"NO"<<endl;
    }

}
```

https://atcoder.jp/contests/agc043/tasks/agc043_a

上の問題の応用

多項式

▶

dpとかでやるような数え上げを多項式で解く

ナップザックみたいな、重さと価値の２変数はキツそう

例題

https://atcoder.jp/contests/abc159/tasks/abc159_f

$$f_i = (1+x^{a_i})$$ とし、

$$\begin{aligned} F_0 &= 0, \ F_n &= (1+F_{n-1})*f_n \end{aligned}$$ とおくと、求めるものは、

$$[x^{s}]\sum_i F_i$$

となるので、順次F_iを求めながら、ansに$p[s]$を足していく。

▶ コード

```
#include <bits/stdc++.h>
#define rep(i,n)for(int i=0;i<(n);i++)
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
typedef tuple<ll,ll,ll> T;
const long long INF = 1LL<<60;
const int MOD = 1000000000+7;
#define rev(s) (string((s).rbegin(), (s).rend()))
// cout << fixed << setprecision(10) << ans << endl; 有効桁数指定
// *min_element(c + l, c + r) *max_element(c + l, c + r) 配列の中のmin-max
// int dx[8]={1,1,0,-1,-1,-1,0,1};
// int dy[8]={0,1,1,1,0,-1,-1,-1};
// int dx[4]={1,0,-1,0};
// int dy[4]={0,1,0,-1};
// ~ は、-1の時だけfalse

int a[3010];
vector<ll> p(3010);

const int mod = 998244353;

int main () {

    int n,s;cin>>n>>s;

    rep(i,n) cin>>a[i];

    ll ans=0;
```

```
    rep(i,n) {
        vector<ll> q(3010);
        p[0]++;
        rep(j,s+1) {
            q[j]=(q[j]+p[j])%mod;
            if (j+a[i]<=s) {
                q[j+a[i]]=(q[j+a[i]]+p[j])%mod;
            }
        }
        ans=(ans+q[s])%mod;
        p=q;
    }

    cout<<ans<<endl;


}
```

https://atcoder.jp/contests/arc028/tasks/arc028_4

赤diff

戻すdp

全部欠けた後に、個別に割っていくタイプ

rep(i,q)だと間に合わないから、 rep(i,n)でやる そのために、メモっていく配列を用意する

▶ コード

```
#include<bits/stdc++.h>
#define rep(i,a,b) for(int i=a;i<b;i++)
#define rrep(i,a,b) for(int i=a;i>=b;i--)
#define fore(i,a) for(auto &i:a)
#define all(x) (x).begin(),(x).end()
//#pragma GCC optimize ("-O3")
using namespace std; void _main(); int main() { cin.tie(0);
ios::sync_with_stdio(false); _main(); }
typedef long long ll; const int inf = INT_MAX / 2; const ll infl = 1LL <<
60;
template<class T>bool chmax(T& a, const T& b) { if (a < b) { a = b; return
1; } return 0; }
template<class T>bool chmin(T& a, const T& b) { if (b < a) { a = b; return
1; } return 0; }
//------------------------------------------------------------------------
--------------------------
#ifdef _MSC_VER
#pragma push_macro("long")
#undef long
#ifdef _WIN32
```

```
inline unsigned int __builtin_ctz(unsigned int x) { unsigned long r;
_BitScanForward(&r, x); return r; }
inline unsigned int __builtin_clz(unsigned int x) { unsigned long r;
_BitScanReverse(&r, x); return 31 - r; }
inline unsigned int __builtin_ffs(unsigned int x) { unsigned long r;
return _BitScanForward(&r, x) ? r + 1 : 0; }
inline unsigned int __builtin_popcount(unsigned int x) { return
__popcnt(x); }
#ifdef _WIN64
inline unsigned long long __builtin_ctzll(unsigned long long x) { unsigned
long r; _BitScanForward64(&r, x); return r; }
inline unsigned long long __builtin_clzll(unsigned long long x) { unsigned
long r; _BitScanReverse64(&r, x); return 63 - r; }
inline unsigned long long __builtin_ffsll(unsigned long long x) { unsigned
long r; return _BitScanForward64(&r, x) ? r + 1 : 0; }
inline unsigned long long __builtin_popcountll(unsigned long long x) {
return __popcnt64(x); }
#else
inline unsigned int hidword(unsigned long long x) { return
static_cast<unsigned int>(x >> 32); }
inline unsigned int lodword(unsigned long long x) { return
static_cast<unsigned int>(x & 0xFFFFFFFF); }
inline unsigned long long __builtin_ctzll(unsigned long long x) { return
lodword(x) ? __builtin_ctz(lodword(x)) : __builtin_ctz(hidword(x)) + 32; }
inline unsigned long long __builtin_clzll(unsigned long long x) { return
hidword(x) ? __builtin_clz(hidword(x)) : __builtin_clz(lodword(x)) + 32; }
inline unsigned long long __builtin_ffsll(unsigned long long x) { return
lodword(x) ? __builtin_ffs(lodword(x)) : hidword(x) ?
__builtin_ffs(hidword(x)) + 32 : 0; }
inline unsigned long long __builtin_popcountll(unsigned long long x) {
return __builtin_popcount(lodword(x)) + __builtin_popcount(hidword(x)); }
#endif // _WIN64
#endif // _WIN32
#pragma pop_macro("long")
#endif // _MSC_VER
template<int MOD> struct ModInt {
    static const int Mod = MOD; unsigned x; ModInt() : x(0) { }
    ModInt(signed sig) { x = sig < 0 ? sig % MOD + MOD : sig % MOD; }
    ModInt(signed long long sig) { x = sig < 0 ? sig % MOD + MOD : sig %
MOD; }
    int get() const { return (int)x; }
    ModInt& operator+=(ModInt that) { if ((x += that.x) >= MOD) x -= MOD;
return *this; }
    ModInt& operator-=(ModInt that) { if ((x += MOD - that.x) >= MOD) x -=
MOD; return *this; }
    ModInt& operator*=(ModInt that) { x = (unsigned long long)x * that.x %
MOD; return *this; }
    ModInt& operator/=(ModInt that) { return *this *= that.inverse(); }
    ModInt operator+(ModInt that) const { return ModInt(*this) += that; }
    ModInt operator-(ModInt that) const { return ModInt(*this) -= that; }
    ModInt operator*(ModInt that) const { return ModInt(*this) *= that; }
    ModInt operator/(ModInt that) const { return ModInt(*this) /= that; }
    ModInt inverse() const {
        long long a = x, b = MOD, u = 1, v = 0;
```

```
        while (b) { long long t = a / b; a -= t * b; std::swap(a, b); u -=
t * v; std::swap(u, v); }
        return ModInt(u);
    }
    bool operator==(ModInt that) const { return x == that.x; }
    bool operator!=(ModInt that) const { return x != that.x; }
    ModInt operator-() const { ModInt t; t.x = x == 0 ? 0 : Mod - x;
return t; }
};
template<int MOD> ostream& operator<<(ostream& st, const ModInt<MOD> a) {
st << a.get(); return st; };
template<int MOD> ModInt<MOD> operator^(ModInt<MOD> a, unsigned long long
k) {
    ModInt<MOD> r = 1; while (k) { if (k & 1) r *= a; a *= a; k >>= 1; }
return r;
}
template<typename T, int FAC_MAX> struct Comb {
    vector<T> fac, ifac;
    Comb() {
        fac.resize(FAC_MAX, 1); ifac.resize(FAC_MAX, 1); rep(i, 1,
FAC_MAX)fac[i] = fac[i - 1] * i;
        ifac[FAC_MAX - 1] = T(1) / fac[FAC_MAX - 1]; rrep(i, FAC_MAX - 2,
1)ifac[i] = ifac[i + 1] * T(i + 1);
    }
    T aPb(int a, int b) { if (b < 0 || a < b) return T(0); return fac[a] *
ifac[a - b]; }
    T aCb(int a, int b) { if (b < 0 || a < b) return T(0); return fac[a] *
ifac[a - b] * ifac[b]; }
    T nHk(int n, int k) {
        if (n == 0 && k == 0) return T(1); if (n <= 0 || k < 0) return 0;
        return aCb(n + k - 1, k);
    } // nHk = (n+k-1)Ck : n is separator
    T pairCombination(int n) { if (n % 2 == 1)return T(0); return fac[n] *
ifac[n / 2] / (T(2) ^ (n / 2)); }
    // combination of paris for n
};
typedef ModInt<1000000007> mint;
Comb<mint, 1010101> com;
template<typename T>
struct FormalPowerSeries {
    using Poly = vector<T>;
    using Conv = function<Poly(Poly, Poly)>;
    Conv conv;
    FormalPowerSeries(Conv conv) :conv(conv) {}

    Poly pre(const Poly& as, int deg) {
        return Poly(as.begin(), as.begin() + min((int)as.size(), deg));
    }

    Poly add(Poly as, Poly bs) {
        int sz = max(as.size(), bs.size());
        Poly cs(sz, T(0));
        for (int i = 0; i < (int)as.size(); i++) cs[i] += as[i];
        for (int i = 0; i < (int)bs.size(); i++) cs[i] += bs[i];
```

```
        return cs;
    }

    Poly sub(Poly as, Poly bs) {
        int sz = max(as.size(), bs.size());
        Poly cs(sz, T(0));
        for (int i = 0; i < (int)as.size(); i++) cs[i] += as[i];
        for (int i = 0; i < (int)bs.size(); i++) cs[i] -= bs[i];
        return cs;
    }

    Poly mul(Poly as, Poly bs) {
        return conv(as, bs);
    }

    Poly mul(Poly as, T k) {
        Poly res(all(as));
        for (auto& a : res) a *= k;
        return res;
    }

    // F(0) must not be 0
    Poly inv(Poly as, int deg) {
        assert(as[0] != T(0));
        Poly rs({ T(1) / as[0] });
        for (int i = 1; i < deg; i <<= 1)
            rs = pre(sub(add(rs, rs), mul(mul(rs, rs), pre(as, i << 1))),
i << 1);
        return rs;
    }

    // not zero
    Poly div(Poly as, Poly bs) {
        while (as.back() == T(0)) as.pop_back();
        while (bs.back() == T(0)) bs.pop_back();
        if (bs.size() > as.size()) return Poly();
        reverse(as.begin(), as.end());
        reverse(bs.begin(), bs.end());
        int need = as.size() - bs.size() + 1;
        Poly ds = pre(mul(as, inv(bs, need)), need);
        reverse(ds.begin(), ds.end());
        return ds;
    }

    // F(0) must be 1
    Poly sqrt(Poly as, int deg) {
        assert(as[0] == T(1));
        T inv2 = T(1) / T(2);
        Poly ss({ T(1) });
        for (int i = 1; i < deg; i <<= 1) {
            ss = pre(add(ss, mul(pre(as, i << 1), inv(ss, i << 1))), i <<
1);
            for (T& x : ss) x *= inv2;
        }
```

```
        return ss;
    }

    Poly diff(Poly as) {
        int n = as.size();
        Poly res(n - 1);
        for (int i = 1; i < n; i++) res[i - 1] = as[i] * T(i);
        return res;
    }

    Poly integral(Poly as) {
        int n = as.size();
        Poly res(n + 1);
        res[0] = T(0);
        for (int i = 0; i < n; i++) res[i + 1] = as[i] / T(i + 1);
        return res;
    }

    // F(0) must be 1
    Poly log(Poly as, int deg) {
        return pre(integral(mul(diff(as), inv(as, deg))), deg);
    }

    // F(0) must be 0
    Poly exp(Poly as, int deg) {
        Poly f({ T(1) });
        as[0] += T(1);
        for (int i = 1; i < deg; i <<= 1)
            f = pre(mul(f, sub(pre(as, i << 1), log(f, i << 1))), i << 1);
        return f;
    }

    Poly partition(int n) {
        Poly rs(n + 1);
        rs[0] = T(1);
        for (int k = 1; k <= n; k++) {
            if (1LL * k * (3 * k + 1) / 2 <= n) rs[k * (3 * k + 1) / 2] +=
T(k % 2 ? -1LL : 1LL);
            if (1LL * k * (3 * k - 1) / 2 <= n) rs[k * (3 * k - 1) / 2] +=
T(k % 2 ? -1LL : 1LL);
        }
        return inv(rs, n + 1);
    }

    Poly catalan(int n) {
        Poly rs(n + 1);
        rs[0] = 1;
        rep(i, 1, n + 1) rs[i] = com.aCb(2 * i, i) - com.aCb(2 * i, i -
1);
        return rs;
    }

    // *(1-x^n)
    Poly mul_1_minus_x_n(Poly as, int n) {
```

```
    Poly res(all(as));
    int m = res.size();
    rrep(i, m - 1, n) res[i] -= res[i - n];
    return res;
}

// /(1-x^n)
Poly div_1_minus_x_n(Poly as, int n) {
    Poly res(all(as));
    int m = res.size();
    rep(i, n, m) res[i] += res[i - n];
    return res;
}

// *(1+x+...+x^n)=*(1-x^(n+1))/(1-x)
Poly mul_1_plus_x_n(Poly as, int n) {
    Poly p1 = mul_1_minus_x_n(as, n + 1);
    return div_1_minus_x_n(p1, 1);
}

// /(1+x+...+x^n)=*(1-x)/(1-x^(n+1))
Poly div_1_plus_x_n(Poly as, int n) {
    Poly p1 = mul_1_minus_x_n(as, 1);
    return div_1_minus_x_n(p1, n + 1);
}









int getrandmax() {
    static uint32_t y = time(NULL);
    y ^= (y << 13); y ^= (y >> 17);
    y ^= (y << 5);
    return abs((int)y);
}

template<typename T2>
int jacobi(T2 a, T2 mod) {
    int s = 1;
    if (a < 0) a = a % mod + mod;
    while (mod > 1) {
        a %= mod;
        if (a == 0) return 0;
        int r = __builtin_ctz(a);
        if ((r & 1) && ((mod + 2) & 4)) s = -s;
        a >>= r;
        if (a & mod & 2) s = -s;
        swap(a, mod);
```

```
        }
        return s;
    }

    template<typename T2>
    vector<T2> mod_sqrt(T2 a, T2 mod) {
        if (mod == 2) return { a & 1 };
        int j = jacobi(a, mod);
        if (j == 0) return { 0 };
        if (j == -1) return {};
        ll b, d;
        while (1) {
            b = getrandmax() % mod;
            d = (b * b - a) % mod;
            if (d < 0) d += mod;
            if (jacobi<ll>(d, mod) == -1) break;
        }

        ll f0 = b, f1 = 1, g0 = 1, g1 = 0;
        for (ll e = (mod + 1) >> 1; e; e >>= 1) {
            if (e & 1) {
                ll tmp = (g0 * f0 + d * ((g1 * f1) % mod)) % mod;
                g1 = (g0 * f1 + g1 * f0) % mod;
                g0 = tmp;
            }
            ll tmp = (f0 * f0 + d * ((f1 * f1) % mod)) % mod;
            f1 = (2 * f0 * f1) % mod;
            f0 = tmp;
        }
        if (g0 > mod - g0) g0 = mod - g0;
        return { T2(g0),T2(mod - g0) };
    }


    Poly super_sqrt(Poly from, int deg) {
        deque<int> as(deg);
        for (int i = 0; i < deg; i++) as[i] = from[i].get();

        while (!as.empty() && as.front() == 0) as.pop_front();

        if (as.empty()) {
            Poly res(deg, 0);
            return res;
        }

        int m = as.size();
        if ((deg - m) & 1) {
            return Poly();
        }

        auto ss = mod_sqrt(as[0], 998244353);
        if (ss.empty()) return Poly();

        vector<T> ps(deg, T(0));
```

```
        for (int i = 0; i < m; i++) ps[i] = T(as[i]) / T(as[0]);

        auto bs = sqrt(ps, deg);
        bs.insert(bs.begin(), (deg - m) / 2, T(0));
        Poly res(deg);
        for (int i = 0; i < deg; i++) {
            res[i] = bs[i] * ss[0];
        }
        return res;
    }









};
#define FOR(i,n) for(int i = 0; i < (n); i++)
#define sz(c) ((int)(c).size())
#define ten(x) ((int)1e##x)
template<class T> T extgcd(T a, T b, T& x, T& y) { for (T u = y = 1, v = x
= 0; a;) { T q = b / a; swap(x -= q * u, u); swap(y -= q * v, v); swap(b -
= q * a, a); } return b; }
template<class T> T mod_inv(T a, T m) { T x, y; extgcd(a, m, x, y); return
(m + x % m) % m; }
ll mod_pow(ll a, ll n, ll mod) { ll ret = 1; ll p = a % mod; while (n) {
if (n & 1) ret = ret * p % mod; p = p * p % mod; n >>= 1; } return ret; }
struct MathsNTTModAny {
    template<int mod, int primitive_root>
    class NTT {
    public:
        int get_mod() const { return mod; }
        void _ntt(vector<ll>& a, int sign) {
            const int n = sz(a);
            assert((n ^ (n & -n)) == 0); //n = 2^k

            const int g = 3; //g is primitive root of mod
            int h = (int)mod_pow(g, (mod - 1) / n, mod); // h^n = 1
            if (sign == -1) h = (int)mod_inv(h, mod); //h = h^-1 % mod

            //bit reverse
            int i = 0;
            for (int j = 1; j < n - 1; ++j) {
                for (int k = n >> 1; k > (i ^= k); k >>= 1);
                if (j < i) swap(a[i], a[j]);
            }

            for (int m = 1; m < n; m *= 2) {
                const int m2 = 2 * m;
```

```
                const ll base = mod_pow(h, n / m2, mod);
                ll w = 1;
                FOR(x, m) {
                    for (int s = x; s < n; s += m2) {
                        ll u = a[s];
                        ll d = a[s + m] * w % mod;
                        a[s] = u + d;
                        if (a[s] >= mod) a[s] -= mod;
                        a[s + m] = u - d;
                        if (a[s + m] < 0) a[s + m] += mod;
                    }
                    w = w * base % mod;
                }
            }

            for (auto& x : a) if (x < 0) x += mod;
        }
        void ntt(vector<ll>& input) {
            _ntt(input, 1);
        }
        void intt(vector<ll>& input) {
            _ntt(input, -1);
            const int n_inv = mod_inv(sz(input), mod);
            for (auto& x : input) x = x * n_inv % mod;
        }

        vector<ll> convolution(const vector<ll>& a, const vector<ll>& b) {
            int ntt_size = 1;
            while (ntt_size < sz(a) + sz(b)) ntt_size *= 2;

            vector<ll> _a = a, _b = b;
            _a.resize(ntt_size); _b.resize(ntt_size);

            ntt(_a);
            ntt(_b);

            FOR(i, ntt_size) {
                (_a[i] *= _b[i]) %= mod;
            }

            intt(_a);
            return _a;
        }
    };

    ll garner(vector<pair<int, int>> mr, int mod) {
        mr.emplace_back(mod, 0);

        vector<ll> coffs(sz(mr), 1);
        vector<ll> constants(sz(mr), 0);
        FOR(i, sz(mr) - 1) {
            // coffs[i] * v + constants[i] == mr[i].second (mod
mr[i].first)
            ll v = (mr[i].second - constants[i]) * mod_inv<ll>(coffs[i],
```

```
mr[i].first) % mr[i].first;
            if (v < 0) v += mr[i].first;

            for (int j = i + 1; j < sz(mr); j++) {
                (constants[j] += coffs[j] * v) %= mr[j].first;
                (coffs[j] *= mr[i].first) %= mr[j].first;
            }
        }

        return constants[sz(mr) - 1];
    }

    typedef NTT<167772161, 3> NTT_1;
    typedef NTT<469762049, 3> NTT_2;
    typedef NTT<1224736769, 3> NTT_3;

    vector<ll> solve(vector<ll> a, vector<ll> b, int mod = 1000000007) {
        for (auto& x : a) x %= mod;
        for (auto& x : b) x %= mod;

        NTT_1 ntt1; NTT_2 ntt2; NTT_3 ntt3;
        assert(ntt1.get_mod() < ntt2.get_mod() && ntt2.get_mod() <
ntt3.get_mod());
        auto x = ntt1.convolution(a, b);
        auto y = ntt2.convolution(a, b);
        auto z = ntt3.convolution(a, b);

        const ll m1 = ntt1.get_mod(), m2 = ntt2.get_mod(), m3 =
ntt3.get_mod();
        const ll m1_inv_m2 = mod_inv<ll>(m1, m2);
        const ll m12_inv_m3 = mod_inv<ll>(m1 * m2, m3);
        const ll m12_mod = m1 * m2 % mod;
        vector<ll> ret(sz(x));
        FOR(i, sz(x)) {
            ll v1 = (y[i] - x[i]) * m1_inv_m2 % m2;
            if (v1 < 0) v1 += m2;
            ll v2 = (z[i] - (x[i] + m1 * v1) % m3) * m12_inv_m3 % m3;
            if (v2 < 0) v2 += m3;
            ll constants3 = (x[i] + m1 * v1 + m12_mod * v2) % mod;
            if (constants3 < 0) constants3 += mod;
            ret[i] = constants3;
        }

        return ret;
    }

    vector<int> solve(vector<int> a, vector<int> b, int mod = 1000000007)
{
        vector<ll> x(all(a));
        vector<ll> y(all(b));

        auto z = solve(x, y, mod);
        vector<int> res;
        fore(aa, z) res.push_back(aa % mod);
```

```
        return res;
    }

    vector<mint> solve(vector<mint> a, vector<mint> b, int mod =
1000000007) {
        int n = a.size();
        vector<ll> x(n);
        rep(i, 0, n) x[i] = a[i].get();
        n = b.size();
        vector<ll> y(n);
        rep(i, 0, n) y[i] = b[i].get();

        auto z = solve(x, y, mod);
        vector<mint> res;
        fore(aa, z) res.push_back(aa);

        return res;
    }
};
/*-------------------------------------------------------------------------
--------------------------
                    ∧__∧
         ∧__∧    (´<_` )   Welcome to My Coding Space!
        ( ´_ゝ`)  /   ⌒i     @hamayanhamayan
      /     \      |  |
     /    /       /   |
   __(__ニつ/      __/ .| .|_______
       \/_______/   (u ⊃
-------------------------------------------------------------------------
------------------------*/









int a[2010],k[500010],x[500010];
vector<int> kx[2010];
mint ans[500010];

void _main() {
    using T = mint;
    FormalPowerSeries<T> FPS([&](auto a, auto b) {
        MathsNTTModAny ntt;
        return ntt.solve(a, b);
        });
```

```
    int n,m,q;cin>>n>>m>>q;

    vector<T> p(m + 10);

    rep(i,0,n) cin>>a[i];

    rep(i,0,q) {
        cin>>k[i]>>x[i];
        k[i]--;
        kx[k[i]].push_back(i);
    }

    p[0] = 1;
    rep(i, 0, n) {
        p = FPS.mul_1_plus_x_n(p,a[i]);
    }

    rep(i,0,n) {
        vector<T> vec(m + 10);
        vec = FPS.div_1_plus_x_n(p,a[i]);
        fore(c,kx[i])  ans[c]=vec[m-x[c]];
    }

    rep(i,0,q) cout<<ans[i]<<endl;

}
```

一番下