Header

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

// Define
using ll = long long;
using ull = unsigned long long;
using ld = long double;
const ll dx[4] = {1, 0, -1, 0};
const ll dy[4] = {0, 1, 0, -1};
const ll MOD = 1e9 + 7;
const ll mod = 998244353;
const ll inf = 1 << 30;
const ll LINF = LONG_MAX;
const ll INF = 1LL << 60;
const ull MAX = ULONG_MAX;
#define mp make_pair
#define pb push_back
#define elif else if
#define endl '\n'
#define space ' '
#define def inline auto
#define func inline constexpr ll
#define run(a) __attribute__((constructor)) def _##a()
#define all(v) begin(v), end(v)
#define rall(v) rbegin(v), rend(v)
#define input(a) scanf("%lld", &(a))
#define print(a) printf("%lld\n", (a))
#define fi first
#define se second
#define ok(a, b) (0 <= (a) && (a) < (b))
template <class T> using vvector = vector<vector<T>>;
template <class T> using pvector = vector<pair<T, T>>;
template <class T>
using rpriority_queue = priority_queue<T, vector<T>, greater<T>>;

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 (a > b) {
        a = b;
        return 1;
    }
    return 0;
}

// Debug
#define debug(...)                                                          \
    {                                                                       \
        cerr << __LINE__ << ": " << #__VA_ARGS__ << " = ";                  \
        for (auto &&X : {__VA_ARGS__}) cerr << "[" << X << "] ";            \
        cerr << endl;                                                       \
    }

#define dump(a, h, w)                                                       \
    {                                                                       \
        cerr << __LINE__ << ": " << #a << " = [" << endl;                   \
        rep(__i, h) {                                                       \
            rep(__j, w) cerr << a[__i][__j] << space;                       \
            cerr << endl;                                                   \
        }                                                                   \
        cerr << "]" << endl;                                                \
    }

#define vdump(a, n)                                                         \
    {                                                                       \
        cerr << __LINE__ << ": " << #a << " = [";                           \
        rep(__i, n) if (__i) cerr << space << a[__i];                       \
        else cerr << a[__i];                                                \
        cerr << "]" << endl;                                                \
    }

// Loop
#define inc(i, a, n) for (ll i = (a), _##i = (n); i <= _##i; ++i)
#define dec(i, a, n) for (ll i = (a), _##i = (n); i >= _##i; --i)
#define rep(i, n) for (ll i = 0, _##i = (n); i < _##i; ++i)
#define each(i, a) for (auto &&i : a)
#define loop() for (;;)
```

```
// Stream
#define fout(n) cout << fixed << setprecision(n)
#define fasten cin.tie(0), ios::sync_with_stdio(0)

// Speed
run(0) { fasten, fout(10); }

// Math
//#define gcd __gcd
func gcd(ll a, ll b) { return b ? gcd(b, a % b) : a; }
func lcm(ll a, ll b) { return a * b / gcd(a, b); }
```

Header-2

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
const ll inf = 1LL << 60;

#define fi first
#define se second
#define pb push_back
#define endl '\n'
#define space ' '
#define inc(i, a, n) for (ll i = (a), _##i = (n); i <= _##i; ++i)
#define dec(i, a, n) for (ll i = (a), _##i = (n); i >= _##i; --i)
#define rep(i, n) for (ll i = 0, _##i = (n); i < _##i; ++i)
#define each(i, a) for (auto &&i : a)
#define debug(...)                                                          \
    {                                                                       \
        cerr << __LINE__ << ": " << #__VA_ARGS__ << " = ";                  \
        for (auto &&X : {__VA_ARGS__}) cerr << "[" << X << "] ";            \
        cerr << endl;                                                       \
    }

#define dump(a, h, w)                                                       \
    {                                                                       \
        cerr << __LINE__ << ": " << #a << " = [" << endl;                   \
        rep(__i, h) {                                                       \
            rep(__j, w) { cerr << a[__i][__j] << space; }                   \
            cerr << endl;                                                   \
        }                                                                   \
        cerr << "]" << endl;                                                \
    }

#define vdump(a, n)                                                         \
    {                                                                       \
        cerr << __LINE__ << ": " << #a << " = [";                           \
        rep(__i, n) if (__i) cerr << space << a[__i];                       \
        else cerr << a[__i];                                                \
        cerr << "]" << endl;                                                \
    }
__attribute__((constructor)) auto a() { cin.tie(0), ios::sync_with_stdio(0); }
```

スライド最小値

```
template <class T> struct slidemin {
#define __SIGSEGV__ "Segmentation fault: 11"
    ll N, l, r;
    vector<ll> data;
    deque<ll> deq;
    slidemin(ll n, ll start) : N(n), data(n), l(start), r(start) {}
    void setdata(ll itr, ll val) { data[itr] = val; }
    void pop_front() {
        while (deq.size() && deq.front() <= l) {
            deq.pop_front();
        }
        l++;
        if (l >= N) cerr << __SIGSEGV__ << endl;
    }
    void push_back() {
        while (deq.size() && data[deq.back()] > data[r]) {
            deq.pop_back();
        }
        deq.push_back(r);
        if (r > N) cerr << __SIGSEGV__ << endl;
        r++;
    }
    ll min() { return data[deq.front()]; }
```

```
};
```

最大長方形

```
struct max_rectangle {
    struct Range {
        ll left, right;
        Range() {}
        Range(ll l, ll r) : left(l), right(r) {}
        ll length() { return right - left; }
        bool operator==(Range A) { return left == A.left && right == A.right; }
        bool operator!=(Range A) { return !(Range(left, right) == A); }
        bool operator>(Range A) { return left < A.left && right > A.right; }
        bool operator<(Range A) { return left > A.left && right < A.right; }
        bool operator>=(Range A) { return left <= A.left && right >= A.right; }
        bool operator<=(Range A) { return left >= A.left && right <= A.right; }
    };
    vector<ll> histgram;
    ll pos_l, pos_r, surface = 0;
    max_rectangle(vector<ll> h) : histgram(h) {}
    void solve() {
        deque<pair<ll, ll>> deq;
        histgram.push_back(0);
        rep(i, histgram.size() + 1) {
            if (deq.empty() || deq.back().fi < histgram[i]) {
                deq.push_back({histgram[i], i});
            } else if (deq.back().fi == histgram[i]) {
                continue;
            } else {
                ll last = -1;
                while (deq.size() && deq.back().fi > histgram[i]) {
                    ll sum = (i - deq.back().se) * deq.back().fi;
                    if (surface <= sum) {
                        surface = sum;
                        pos_l = deq.back().se;
                        pos_r = i + 1;
                    }
                    last = deq.back().se;
                    deq.pop_back();
                }
                if (last != -1) deq.push_back({histgram[i], last});
            }
        }
    }
    Range max_range() { return Range(pos_l, pos_r); }
    ll max_surface() { return surface; }
};
```

CHT

```
struct LinearCHT {
    using F = pair<ll, ll>;
    deque<F> deq;
#define a first
#define b second
#define lines deq.size()
    bool invalid_f1(F f0, F f1, F f2) {
        return (f1.a - f0.a) * (f2.b - f1.b) >= (f1.b - f0.b) * (f2.a - f1.a);
    }
    ll f(F f0, ll x) { return f0.a * x + f0.b; }
    void add(F f0) {
        while (lines >= 2 && invalid_f1(deq[lines - 2], deq[lines - 1], f0)) {
            deq.pop_back();
        }
        deq.push_back(f0);
    }
    ll min(ll x) {
        while (lines >= 2 && f(deq[0], x) >= f(deq[1], x)) {
            deq.pop_front();
        }
        return f(deq[0], x);
    }
#undef a
#undef b
#undef lines
};
```

永続配列

```
template <class T> struct Array {
    struct node {
        ll childl, childr;
        T data;
        node(ll l, ll r, T t) : childl(l), childr(r), data(t) {}
    };
    ll n, depth;
    vector<ll> versions;
    vector<ll> prev_versions;
    vector<node> nodes;
    Array(ll n = 1 << 20, T val = T()) : n(n), depth(0) {
        while (n /= 2) depth++;
        init(val);
    }
    void init(T val) {
        versions.push_back(0);
        prev_versions.push_back(0);
        rep(i, 2 * n - 1) {
            if (i < n - 1) {
                nodes.push_back(node(2 * i + 1, 2 * i + 2, T()));
            } else {
                nodes.push_back(node(0, 0, val));
            }
        }
    }
    void set(ll index, ll val, ll version = -1) {
        ll id, par = nodes.size(), left = 0, right = n;
        if (version == -1) {
            id = versions.back();
            version = versions.size() - 1;
        } else {
            id = versions[version];
        }
        versions.push_back(par);
        prev_versions.push_back(version);
        if (right == -1) right = n;
        rep(i, depth) {
            ll mid = (left + right) / 2;
            if (index < mid) {
                nodes.push_back(node(par + i + 1, nodes[id].childr, T()));
                id = nodes[id].childl;
                right = mid;
            } else {
                nodes.push_back(node(nodes[id].childl, par + i + 1, T()));
                id = nodes[id].childr;
                left = mid;
            }
        }
        nodes.push_back(node(0, 0, val));
    }
    T get(ll index, ll version = -1) {
        ll id, left = 0, right = n;
        if (version == -1) {
            id = versions.back();
        } else {
            id = versions[version];
        }
        rep(i, depth) {
            ll mid = (left + right) / 2;
            if (index < mid) {
                id = nodes[id].childl;
                right = mid;
            } else {
                id = nodes[id].childr;
                left = mid;
            }
        }
        return nodes[id].data;
    }
};
```

UnionFind

```
struct UFS {
    vector<ll> data;
    UFS(ll N) : data(N) { rep(i, N) data[i] = -1; }
    def root(ll x) {
        if (data[x] < 0)
            return x;
        else
            return data[x] = root(data[x]);
    }
```

```
    def unite(ll x, ll y) {
        ll root_x = root(x), root_y = root(y);
        if (root_x != root_y) {
            if (data[root_x] > data[root_y]) swap(root_x, root_y);
            data[root_x] += data[root_y];
            data[root_y] = root_x;
            return true;
        }
        return false;
    }
    def same(ll x, ll y) { return root(x) == root(y); }
    def size(ll x) { return -data[root(x)]; }
};

struct UFR {
    vector<ll> data;
    UFR(ll N) : data(N) { rep(i, N) data[i] = -1; }
    def root(ll x) {
        if (data[x] < 0)
            return x;
        else
            return data[x] = root(data[x]);
    }
    def unite(ll x, ll y) {
        ll root_x = root(x), root_y = root(y);
        if (root_x != root_y) {
            if (data[root_x] > data[root_y]) swap(root_x, root_y);
            data[root_x] -= data[root_x] == data[root_y];
            data[root_y] = root_x;
            return true;
        }
        return false;
    }
    def same(ll x, ll y) { return root(x) == root(y); }
};
```

遅延セグ木加算

```
struct RASQ {
    static const ll n = 1LL << 20;
    vector<ll> node, lazy;
    RASQ() : node(n * 2 - 1), lazy(n * 2 - 1) {}
    inline void eval(ll i, ll l, ll r) {
        node[i] += lazy[i];
        if (r - l > 1) {
            lazy[i * 2 + 1] += lazy[i] / 2;
            lazy[i * 2 + 2] += lazy[i] / 2;
        }
        lazy[i] = 0;
    }
    inline void add(ll a, ll b, ll x, ll i = 0, ll l = 0, ll r = 1LL << 20) {
        eval(i, l, r);
        if (b <= l || r <= a) return;
        if (a <= l && r <= b) {
            lazy[i] += (r - l) * x;
            eval(i, l, r);
        } else {
            add(a, b, x, i * 2 + 1, l, (l + r) / 2);
            add(a, b, x, i * 2 + 2, (l + r) / 2, r);
            node[i] = node[i * 2 + 1] + node[i * 2 + 2];
        }
    }
    inline ll sum(ll a, ll b, ll i = 0, ll l = 0, ll r = 1LL << 20) {
        if (b <= l || r <= a) return 0;
        eval(i, l, r);
        if (a <= l && r <= b) return node[i];
        return sum(a, b, i * 2 + 1, l, (l + r) / 2) +
               sum(a, b, i * 2 + 2, (l + r) / 2, r);
    }
};
```

遅延セグ木max

```
struct RUMQ {
    static const ll n = 1LL << 20;
    vector<ll> node, lazy, flag;
    RUMQ() : node(n * 2 - 1), lazy(n * 2 - 1), flag(n * 2 - 1) {}
    inline void eval(ll i, ll l, ll r) {
        if (flag[i]) {
            node[i] = lazy[i];
```

```
            if (r - l > 1) {
                lazy[i * 2 + 1] = lazy[i * 2 + 2] = lazy[i];
                flag[i * 2 + 1] = flag[i * 2 + 2] = true;
            }
            lazy[i] = 0;
            flag[i] = false;
        }
    }
    inline void update(ll a, ll b, ll x, ll i = 0, ll l = 0, ll r = 1LL << 20) {
        eval(i, l, r);
        if (b <= l || r <= a) return;
        if (a <= l && r <= b) {
            lazy[i] = x;
            flag[i] = true;
            eval(i, l, r);
        } else {
            update(a, b, x, i * 2 + 1, l, (l + r) / 2);
            update(a, b, x, i * 2 + 2, (l + r) / 2, r);
            node[i] = std::min(node[i * 2 + 1], node[i * 2 + 2]);
        }
    }
    inline ll min(ll a, ll b, ll i = 0, ll l = 0, ll r = 1LL << 20) {
        if (b <= l || r <= a) return INF;
        eval(i, l, r);
        if (a <= l && r <= b) return node[i];
        return std::min(min(a, b, i * 2 + 1, l, (l + r) / 2),
                        min(a, b, i * 2 + 2, (l + r) / 2, r));
    }
};
```

StarrySky木

```
struct RUMQ {
    static const ll n = 1LL << 20;
    vector<ll> node, lazy, flag;
    RUMQ() : node(n * 2 - 1), lazy(n * 2 - 1), flag(n * 2 - 1) {}
    inline void eval(ll i, ll l, ll r) {
        if (flag[i]) {
            node[i] = lazy[i];
            if (r - l > 1) {
                lazy[i * 2 + 1] = lazy[i * 2 + 2] = lazy[i];
                flag[i * 2 + 1] = flag[i * 2 + 2] = true;
            }
            lazy[i] = 0;
            flag[i] = false;
        }
    }
    inline void update(ll a, ll b, ll x, ll i = 0, ll l = 0, ll r = 1LL << 20) {
        eval(i, l, r);
        if (b <= l || r <= a) return;
        if (a <= l && r <= b) {
            lazy[i] = x;
            flag[i] = true;
            eval(i, l, r);
        } else {
            update(a, b, x, i * 2 + 1, l, (l + r) / 2);
            update(a, b, x, i * 2 + 2, (l + r) / 2, r);
            node[i] = std::min(node[i * 2 + 1], node[i * 2 + 2]);
        }
    }
    inline ll min(ll a, ll b, ll i = 0, ll l = 0, ll r = 1LL << 20) {
        if (b <= l || r <= a) return INF;
        eval(i, l, r);
        if (a <= l && r <= b) return node[i];
        return std::min(min(a, b, i * 2 + 1, l, (l + r) / 2),
                        min(a, b, i * 2 + 2, (l + r) / 2, r));
    }
};
```

BIT

```
struct BIT {
    vector<ll> data;

    BIT() {}
    BIT(ll N) : data(N + 1) {
        fill(data.begin(), data.end(), 0);
        data[0] = N;
    }
```

```
    void add(ll pos, ll val) {
        while (pos <= data[0]) {
            data[pos] += val;
            pos += pos & -pos;
        }
    }
    ll sum(ll pos) {
        if (pos <= 0) return 0;
        ll res = 0;
        while (pos > 0) {
            res += data[pos];
            pos -= pos & -pos;
        }
        return res;
    }

    ll lower_bound(ll val) {
        if (val <= 0) return 0;
        ll index = 0;
        for (ll d = 1 << ll(log2(data[0])); d > 0; d /= 2) {
            if (index + d <= data[0] && data[index + d] < val) {
                val -= data[index + d];
                index += d;
            }
        }
        return index;
    }
};
```

mod上の数学関数

```
 struct modmath {
    ll mod = 1e9 + 7, max;
    vector<ll> fac, inv;
    modmath() : max(1 << 20), fac(max + 1), inv(max + 1) {
        fac[0] = ll(1);
        rep(i, max) fac[i + 1] = fac[i] * (i + 1) % mod;
        inv[max] = inv(fac[max]);
        dec(i, max - 1, 0) inv[i] = inv[i + 1] * (i + 1) % mod;
    }
    modmath(ll n) : max(n), fac(n + 1), inv(n + 1) {
        fac[0] = 1;
        rep(i, n) fac[i + 1] = fac[i] * (i + 1) % mod;
        inv[n] = inv(fac[n]);
        dec(i, n - 1, 0) inv[i] = inv[i + 1] * (i + 1) % mod;
    }
    modmath(ll n, ll m) : mod(m), max(n), fac(n + 1), inv(n + 1) {
        fac[0] = 1;
        rep(i, n) fac[i + 1] = fac[i] * (i + 1) % mod;
        inv[n] = inv(fac[n]);
        dec(i, n - 1, 0) inv[i] = inv[i + 1] * (i + 1) % mod;
    }

    inline ll fact(ll n) {
        if (n < 0) return 0LL;
        return fac[n];
    }
    inline ll perm(ll n, ll r) {
        if (r < 0 || n < r) return 0LL;
        return fac[n] * inv[n - r] % mod;
    }
    inline ll comb(ll n, ll r) {
        if (r < 0 || n < r) return 0LL;
        return fac[n] * inv[r] % mod * inv[n - r] % mod;
    }
    inline ll nHr(ll n, ll r) { return comb(n + r - 1, n - 1); }
};
```

mod計算用の関数

```
inline ll inv(const ll n, const ll m = MOD) {
    ll a = n, b = m, x = 1, y = 0;
    while (b) {
        ll t = a / b;
        a -= t * b;
        swap(a, b);
        x -= t * y;
        swap(x, y);
    }
    return modulo(x, m);
```

```
}

inline constexpr ll modulo(const ll n, const ll m = MOD) {
    ll k = n % m;
    return k + m * (k < 0);
}

inline constexpr ll chmod(ll &n, const ll m = MOD) {
    n %= m;
    return n += m * (n < 0);
}

inline constexpr ll roundup(const ll a, const ll b) {
    if (a % b == 0)
        return a;
    else
        return a + (b - a % b);
}

inline constexpr ll mpow(ll a, ll n, const ll m = MOD) {
    ll r = 1;
    while (n) {
        if (n & 1) r *= a;
        chmod(r, m);
        a *= a;
        chmod(a, m);
        n >>= 1;
    }
    return r;
}
```

エラトステネスの篩

```
struct fact {
    ll N, A[100000], B[100000], C[100000];
    void factorization() {
        A[0] = 1;
        rep(i, N) {
            if (A[i] == 0) {
                for (ll k = 2 * (i + 1); k < N; k += i + 1) {
                    A[k] = 1;
                    B[k] = i;
                }
            }
        }
        rep(i, N) {
            if (i && A[i] == 1) C[i] = (i + 1) / B[i] + 1;
        }
    }
};
```

中国剰余

```
struct ChugokuJoyo {

    ll mumll(ll p, ll q, ll m) {
        ull x = 0, pp = p;
        for (; q; q /= 2)
            q & 1 ? x += pp, x >= m ? x -= m : 0 : 0, pp *= 2,
                pp >= m ? pp -= m : 0;
        return (ll) x;
    }

    def extgcd(ll a, ll b, ll &x, ll &y) {
        if (b == 0) {
            x = 1, y = 0;
            return a;
        }
        ll d = extgcd(b, a % b, y, x);
        y -= a / b * x;
        return d;
    }

    ll crt(ll a, ll p, ll b, ll q) {
        a %= p;
        if (a < 0) a += p;
        b %= q;
        if (b < 0) b += q;
        if (a > b) {
            ll t;
```

```
            t = a;
            a = b;
            b = t;
            t = p;
            p = q;
            q = t;
        }
        ll x, y, m, d;
        d = extgcd(p, q, x, y); // px + qy = gcd(p, q)
        if ((a - b) % d) return -1;
        m = p / d * q;
        // px + qy = gcd(p, q) mod lcm(p, q)
        if (x < 0) x += m / p;
        if (m < 11LL << 28) return (a + (b - a) / d * p % m * x) % m;
        return (a + (ull) mumll(mumll((b - a) % q / d, p, m), x, m)) % m;
    }
};
```

素数テーブル

```
vector<ll> prime(ll n) {
    vector<ll> Prime(n + 1);
    rep(i, n + 1) Prime[i] = 1;
    Prime[0] = Prime[1] = 0;
    rep(i, n + 1) {
        if (Prime[i]) {
            rep(j, (n + 1) / i - 1) Prime[i * (j + 2)] = 0;
        }
    }
    return Prime;
}
```

BFS

```
struct bfs {
    typedef pair<ll, ll> P;
#define x first
#define y second
#define MAX_N 1000
#define MAX_M 1000
    string maze[MAX_N];
    P start, goal;
    ll N, M;
    ll D[MAX_N][MAX_M], FROM_X[MAX_N][MAX_M], FROM_Y[MAX_N][MAX_M];
    ll BFS() {
        queue<P> Q;
        rep(i, N) rep(j, M) D[i][j] = INF;
        Q.push(start);
        D[start.x][start.y] = 0;
        while (!Q.empty()) {
            P now = Q.front();
            Q.pop();
            if (now.x == goal.x && now.y == goal.y) break;
            rep(i, 4) {
                P next = make_pair(now.x + dx[i], now.y + dy[i]);
                if (0 <= next.x && next.x < N && 0 <= next.y && next.y < M)
                    if (maze[next.x][next.y] != '#' &&
                        D[next.x][next.y] == INF) {
                        Q.push(next);
                        D[next.x][next.y] = D[now.x][now.y] + 1;
                        FROM_X[next.x][next.y] = now.x;
                        FROM_Y[next.x][next.y] = now.y;
                    }
            }
        }
        return D[goal.x][goal.y];
    }
};
```

ベルマンフォード

```
struct BellmanFord {
    struct edge {
        ll to, cost;
        edge(ll a, ll b) : to(a), cost(b) {}
    };
    using graph = vector<vector<edge>>;
```

```
    ll N;
    graph G;
    vector<ll> dist, from;
    vector<bool> error_loop;
    BellmanFord(ll n) : N(n), G(n), dist(n), from(n), error_loop(n) {}
    void setDist(ll from, ll to, ll cost) { G[from].push_back(edge(to, cost)); }
    void culc(ll start = 0) {
        fill(all(dist), INF);
        fill(all(from), -1);
        fill(all(error_loop), false);
        dist[start] = 0;
        rep(i, N - 1) rep(j, N) rep(k, G[j].size()) {
            ll to = G[j][k].to;
            if (dist[j] == INF) continue;
            if (dist[to] > dist[j] + G[j][k].cost) {
                dist[to] = dist[j] + G[j][k].cost;
                from[to] = j;
            }
        }
        rep(i, N) rep(j, N) rep(k, G[j].size()) {
            ll to = G[j][k].to;
            if (dist[j] == INF) continue;
            if (dist[to] > dist[j] + G[j][k].cost) error_loop[to] = true;
            if (error_loop[j]) error_loop[to] = true;
        }
    }
};
```

ダイクストラ

```
template <typename T = ll> struct Dijkstra {
    ll V;
    using P = pair<ll, ll>;
    vector<vector<P>> G;
    vector<T> dist;
    vector<bool> used;
    Dijkstra(ll v) : V(v), G(v), dist(v), used(v) {}
    void setDist(ll a, ll b, ll d) { G[a].push_back(P(d, b)); }
    void culc(ll s = 0) {
        priority_queue<P, vector<P>, greater<P>> Q;
        Q.push(P(0, s));
        fill_n(dist.begin(), V, INF);
        fill_n(used.begin(), V, false);
        while (!Q.empty()) {
            T d;
            ll t;
            tie(d, t) = Q.top(), Q.pop();
            if (used[t]) continue;
            used[t] = true, dist[t] = d;
            for (P e : G[t]) {
                if (dist[e.second] <= d + e.first) continue;
                Q.push(P(d + e.first, e.second));
            }
        }
    }
};
```

MST

```
struct MST {
    struct wedge_t {
        ll src, dst;
        ll weight;
    };

    struct graph {
        ll n;
        vector<wedge_t> edges;
        graph(ll n = 0) : n(n) { p.assign(n, -1); }
        void add_edge(ll src, ll dst, ll weight) {
            n = max(n, max(src, dst) + 1);
            edges.push_back({src, dst, weight});
        }
        vector<ll> p;
        ll root(ll i) { return p[i] < 0 ? i : p[i] = root(p[i]); }
        bool unite(ll i, ll j) {
            if ((i = root(i)) == (j = root(j))) return false;
            if (p[i] > p[j]) swap(i, j);
            p[i] += p[j];
            p[j] = i;
```

```
            return true;
        }
        ll kruskal() {
            sort(all(edges),
                 [](wedge_t x, wedge_t y) { return x.weight < y.weight; });
            ll result = 0;
            for (auto e : edges)
                if (unite(e.src, e.dst)) result += e.weight;
            return result;
        }
    };
};
```

強連結

```
struct SCC {
    vector<vector<ll>> graph;
    vector<vector<ll>> rgraph;
    vector<vector<ll>> new_graph;
    vector<ll> used;
    vector<ll> in_count;
    vector<ll> new_in_count;
    vector<ll> tp_index;
    vector<ll> nodes;
    ll num, new_num;
    SCC(ll n)
        : num(n), graph(n), rgraph(n), in_count(n), tp_index(n), used(n) {}
    void add_edge(ll from, ll to) {
        graph[from].push_back(to);
        rgraph[to].push_back(from);
        in_count[to]++;
    }
    void new_add_edge(ll from, ll to) {
        ll f = tp_index[from], t = tp_index[to];
        if (f == t) return;
        new_graph[f].push_back(t);
        new_in_count[t]++;
    }
    void dfs(ll pos) {
        used[pos] = true;
        each(i, graph[pos]) if (!used[i]) dfs(i);
        nodes.push_back(pos);
    }
    void rdfs(ll pos, ll k) {
        used[pos] = true;
        tp_index[pos] = k;
        each(i, rgraph[pos]) if (!used[i]) rdfs(i, k);
    }
    ll scc() {
        fill(all(used), false);
        nodes.clear();
        rep(i, num) if (!used[i]) dfs(i);
        reverse(all(nodes));
        fill(all(used), false);
        ll k = 0;
        each(i, nodes) if (!used[i]) rdfs(i, k++);
        new_graph.resize(k), new_in_count.resize(k);
        build_new_graph();
        return new_num = k;
    }
    void build_new_graph() { rep(i, num) each(j, graph[i]) new_add_edge(i, j); }
};
```

トポロジカルソート

```
struct DAG {
    ll N;
    vector<list<ll>> graph;
    vector<ll> num, in_count, sorted_nodes;
    DAG(ll n) : N(n), graph(n), num(n, -1), in_count(n) {}
    void setedge(ll from, ll to) { graph[from].push_back(to), in_count[to]++; }
    bool topological_sort() {
        ll pos, nodes = 0;
        vector<list<ll>> g = graph;
        deque<ll> origin;
        rep(i, N) if (!in_count[i]) origin.push_back(i);
        while (origin.size()) {
            pos = origin.front(), origin.pop_front();
            num[pos] = nodes;
            sorted_nodes.push_back(pos);
```

```
            auto iter = g[pos].begin();
            rep(i, g[pos].size()) {
                ll to = *iter;
                iter = g[pos].erase(iter);
                in_count[to]--;
                if (in_count[to] == 0) origin.push_back(to);
            }
            nodes++;
        }
        rep(i, N) if (g[i].size()) return false;
        return true;
    }
    ll getnum(ll at) { return num[at]; }
    vector<ll> sorted() { return sorted_nodes; }
    bool tp_unique() {
        ll count = 0;
        rep(i, N) {
            if (num[i] != N - 1) {
                each(j, graph[i]) {
                    if (num[j] == num[i] + 1) count++;
                }
            }
        }
        return count == N - 1;
    }
};
```

ワーシャルフロイド

```
class WarshallFloyd {
    private:
    const int n;
    vector<vector<ll>> d;

    public:
    WarshallFloyd(int _n) : n(_n), d(_n, vector<ll>(_n)) {
        rep(i, n) rep(j, n) { d[i][j] = (i == j ? 0 : INF); }
    }
    // directed
    void setDist(int i, int j, int c) { d[i][j] = c; }
    int getDist(int i, int j) { return d[i][j]; }
    void calc() {
        rep(k, n) rep(i, n) rep(j, n) {
            d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
        }
    }
};
```

最大マッチング

```
struct BipartiteGraph {
    ll V;
    vector<vector<ll>> G;
    vector<ll> match;
    vector<bool> used;
    BipartiteGraph(ll N) : V(N), G(N), match(N), used(N) {}
    void addEdge(ll i, ll j) {
        G[i].push_back(j);
        G[j].push_back(i);
    }
    bool dfs(ll now) {
        used[now] = true;
        rep(i, G[now].size()) {
            ll next = G[now][i], w = match[next];
            if (w == -1 || (!used[w] && dfs(w))) {
                match[now] = next, match[next] = now;
                return true;
            }
        }
        return false;
    }
    ll matching() {
        ll res = 0;
        fill(all(match), -1);
        rep(i, V) {
            if (match[i] == -1) {
                fill(all(used), false);
                if (dfs(i)) res++;
            }
        }
```

```
        return res;
    }
};
```

フロー

```
struct flow {
    struct edge {
        ll to, capasity, reverse;
        edge() {}
        edge(ll a, ll b, ll c) : to(a), capasity(b), reverse(c) {}
    };
    ll V;
    vector<vector<edge>> G;
    vector<bool> used;
    flow(ll N) : V(N), G(N), used(N) {}
    void addEdge(ll from, ll to, ll capasity) {
        ll index = G[to].size(), rindex = G[from].size();
        G[from].push_back(edge(to, capasity, index));
        G[to].push_back(edge(from, 0, rindex));
    }
    ll dfs(ll now, ll to, ll flow) {
        if (now == to) return flow;
        used[now] = true;
        rep(i, G[now].size()) {
            ll next = G[now][i].to, ri = G[now][i].reverse;
            if (!used[next] && G[now][i].capasity > 0) {
                ll drain = dfs(next, to, min(flow, G[now][i].capasity));
                if (drain > 0) {
                    G[now][i].capasity -= drain;
                    G[next][ri].capasity += drain;
                    return drain;
                }
            }
        }
        return 0;
    }
    ll maxFlow(ll s, ll t) {
        ll flow = 0;
        loop() {
            fill(all(used), 0);
            ll d = dfs(s, t, INF);
            if (d == 0) return flow;
            flow += d;
        }
    }
};
```

素因数分解

```
struct Factor {
    inline vector<ll> factors(ll N) {
        vector<ll> A;
        ll i = 2;
        while (i * i <= N) {
            if (N % i == 0) {
                A.push_back(i);
                N /= i;
            } else {
                i++;
            }
        }
        if (N != 1) A.push_back(N);
        sort(all(A));
        return A;
    }
    inline vector<ll> divisor(ll N) {
        vector<ll> A;
        inc(i, 1, sqrt(N)) {
            if (N % i == 0) {
                A.push_back(i);
                if (i * i != N) A.push_back(N / i);
            }
        }
        sort(all(A));
        return A;
    }
};
```