

ICPC Notebook

template	
hash.sh	1
settings.sh	1
template.hpp	1
data-structure	
BIT.hpp	1
FastSet.hpp	1
binary_trie.hpp	2
dsu.hpp	2
lazy_segtree.hpp	3
segtree.hpp	4
math	
BinaryGCD.hpp	4
ExtGCD.hpp	5
floor_sum.hpp	5
modint	
BarrettReduction.hpp	5
modint.hpp	5
FPS	
FFT.hpp	5
FFT_fast.hpp	5
graph	
low_link.hpp	6
max_flow.hpp	6
min_cost_flow.hpp	7
scc.hpp	8
two_sat.hpp	8
graph/tree	
hld.hpp	8
flow	
燃やす埋める.md	9
string	
KMP.hpp	9
Manacher.hpp	9
RollingHash.hpp	9
SuffixArray.hpp	9
Zalgorithm.hpp	10
algorithm	
geometry	
memo	
Primes.md	10

template

hash.sh

使い方: sh hash.sh -> コピペ -> Ctrl + D
コメント・空白・改行を削除して md5 でハッシュする
g++ -dD -E -P -fpreprocessed - | tr -d '[:space:]' | md5sum | cut -c-6

settings.sh

Clion の設定
Settings → Build → CMake → Reload CMake Project
add_compile_options(-D_GLIBCXX_DEBUG)
Caps Lock を Ctrl に変更
setxkbmap -option ctrl:nocaps

template.hpp

md5: 136d85

#include <bits/stdc++.h>
using namespace std;
using ll = long long;
const ll INF = LLONG_MAX / 4;
#define rep(i, a, b) for(ll i = a; i < (b); i++)
#define all(a) begin(a), end(a)
#define sz(a) ssize(a)
bool chmin(auto& a, auto b) { return a > b ? a = b, 1 : 0; }
bool chmax(auto& a, auto b) { return a < b ? a = b, 1 : 0; }

int main() {
cin.tie(0)->sync_with_stdio(0);
// your code here...
}

data-structure

BIT.hpp

md5: 8133c8

struct BIT {
vector<ll> a;
BIT(ll n) : a(n + 1) {}
void add(ll i, ll x) { // A[i] += x
i++;
while(i < sz(a)) {
a[i] += x;
i += i & -i;
}
}
ll sum(ll r) {
ll s = 0;
while(r) {
s += a[r];
r -= r & -r;
}
return s;
}
ll sum(ll l, ll r) { // sum of A[l, r)
return sum(r) - sum(l);
}
};

FastSet.hpp

md5: 2cb8c9

// using u64 = uint64_t;
const u64 B = 64;
struct FastSet {
u64 n;
vector<vector<u64>> a;
FastSet(u64 n_) : n(n_) {
do a.emplace_back(n_ = (n_ + B - 1) / B);
while(n_ > 1);
}
// bool operator[](ll i) const { return a[0][i / B] >> (i % B) & 1; }
void set(ll i) {
for(auto& v : a) {
v[i / B] |= 1ULL << (i % B);
i /= B;
}

```
    }
}

void reset(ll i) {
    for(auto& v : a) {
        v[i / B] &= ~(1ULL << (i % B));
        if(v[i / B]) break;
        i /= B;
    }
}

ll next(ll i) { // i を超える最小の要素
    rep(h, 0, sz(a)) {
        i++;
        if(i / B >= sz(a[h])) break;
        u64 d = a[h][i / B] >> (i % B);
        if(d) {
            i += countr_zero(d);
            while(h--) i = i * B + countr_zero(a[h][i]);
            return i;
        }
        i /= B;
    }
    return n;
}

ll prev(ll i) { // i より小さい最大の要素
    rep(h, 0, sz(a)) {
        i--;
        if(i < 0) break;
        u64 d = a[h][i / B] << (~i % B);
        if(d) {
            i -= countl_zero(d);
            while(h--) i = i * B + __lg(a[h][i]);
            return i;
        }
        i /= B;
    }
    return -1;
}
};
```

binary_trie.hpp

md5: 264b03

```
// base: 69a9e9
template<typename T, int MAX_LOG = 32> struct BinaryTrie {
    struct Node {
        array<int, 2> next;
        int common;
        T lazy;
        Node() : next{-1, -1}, common(), lazy() {}
    };
    vector<Node> nodes;
    BinaryTrie() { nodes.push_back(Node()); }
    void apply_xor(T val) { nodes[0].lazy ^= val; }
    void push(int cur, int b) {
        if((nodes[cur].lazy >> b) & 1) swap(nodes[cur].next[0],
nodes[cur].next[1]);
        rep(i, 0, 2) {
            if(nodes[cur].next[i] == -1)
nodes[nodes[cur].next[i]].lazy ^= nodes[cur].lazy;
        }
        nodes[cur].lazy = 0;
    }
    void add(T val, int cur = 0, int b = MAX_LOG - 1) {
        nodes[cur].common++;
        if(b == -1) return;
        push(cur, b);
        int nxt = (val >> (T)b) & (T)1;
        if(nodes[cur].next[nxt] == -1) {
            nodes[cur].next[nxt] = sz(nodes);
            nodes.push_back(Node());
        }
        add(val, nodes[cur].next[nxt], b - 1);
    }

    void erase(T val, int cur = 0, int b = MAX_LOG - 1) {
        nodes[cur].common--;
        if(b == -1) return;
        push(cur, b);
        erase(val, nodes[cur].next[(val >> b) & 1], b - 1);
    }

    T min_element(T val = 0, int cur = 0, int b = MAX_LOG - 1) {
        if(b == -1) return 0;
```

```
        push(cur, b);
        T nxt = (val >> b) & 1;
        int ind = nodes[cur].next[nxt];
        if(ind == -1 || nodes[ind].common == 0) nxt ^= 1;
        return min_element(val, nodes[cur].next[nxt], b - 1) |
(nxt << b);
    } // ddf699

    T max_element(T val = 0, int cur = 0, int b = MAX_LOG - 1) {
return min_element(~val); } // 5e1a86

    int lower_bound_rank(T val, int cur = 0, int b = MAX_LOG -
1) {
        if(cur == -1 || b == -1) return 0;
        push(cur, b);
        T nxt = (val >> b) & 1;
        int ret = lower_bound_rank(val, nodes[cur].next[nxt], b -
1);
        if(nxt == 1 && nodes[cur].next[0] != -1) { ret +=
nodes[nodes[cur].next[0]].common; }
        return ret;
    } // 05b14c

    int upper_bound_rank(T val) { return lower_bound_rank(val +
1); } // 70e301

    T kth_smallest(int k, int cur = 0, int b = MAX_LOG - 1) {
        if(b == -1) return 0;
        int lower_ind = nodes[cur].next[0];
        int lower_cnt = 0;
        if(lower_ind != -1) lower_cnt = nodes[lower_ind].common;
        if(k < lower_cnt) return kth_smallest(k,
nodes[cur].next[0], b - 1);
        return kth_smallest(k - lower_cnt, nodes[cur].next[1], b
- 1) | (T)1 << b);
    } // b85845

    T kth_largest(int k) { return kth_smallest(nodes[0].common -
k); } // 1df41b

    int count(T val) {
        int cur = 0;
        for(int b = MAX_LOG - 1; b >= 0; b--) {
            push(cur, b);
            cur = nodes[cur].next[(val >> b) & 1];
            if(cur == -1) return 0;
        }
        return nodes[cur].common;
    } // 2a3342

    int size() { return nodes[0].common; } // 5f9f13
};
```

dsu.hpp

md5: f21c57

```
// base: c45937
struct dsu {
private:
    int _n;
    vector<int> parent_or_size;

public:
    dsu() : _n(0) {}
    explicit dsu(int n) : _n(n), parent_or_size(n, -1) {}

    int merge(int a, int b) {
        // assert(0 <= a && a < _n);
        // assert(0 <= b && b < _n);
        int x = leader(a), y = leader(b);
        if(x == y) return x;
        if(-parent_or_size[x] < -parent_or_size[y]) swap(x, y);
        parent_or_size[x] += parent_or_size[y];
        parent_or_size[y] = x;
        return x;
    }

    bool same(int a, int b) {
        // assert(0 <= a && a < _n);
        // assert(0 <= b && b < _n);
        return leader(a) == leader(b);
    }
};
```

```

int leader(int a) {
    // assert(0 <= a && a < _n);
    if(parent_or_size[a] < 0) return a;
    int x = a;
    while(parent_or_size[x] >= 0) x = parent_or_size[x];
    while(parent_or_size[a] >= 0) {
        int t = parent_or_size[a];
        parent_or_size[a] = x;
        a = t;
    }
    return x;
}

int size(int a) {
    // assert(0 <= a && a < _n);
    return -parent_or_size[leader(a)];
} // 1ff997

vector<vector<int>> groups() {
    vector<int> leader_buf(_n), group_size(_n);
    rep(i, 0, _n) {
        leader_buf[i] = leader(i);
        group_size[leader_buf[i]]++;
    }
    vector<vector<int>> result(_n);
    rep(i, 0, _n) result[i].reserve(group_size[i]);
    rep(i, 0, _n) result[leader_buf[i]].push_back(i);
    result.erase(remove_if(result.begin(), result.end(), [&
(const vector<int>& v) { return v.empty(); }]),
        result.end());
    return result;
} // 45ebf9
};

```

lazy_segtree.hpp

md5: c86cef

```

// base: 918715
unsigned int bit_ceil(unsigned int n) {
    unsigned int x = 1;
    while(x < (unsigned int)(n)) x *= 2;
    return x;
}

int countr_zero(unsigned int n) { return __builtin_ctz(n); }
constexpr int countr_zero_constexpr(unsigned int n) {
    int x = 0;
    while(!(n & (1 << x))) x++;
    return x;
}

template<class S, S (*op)(S, S), S (*e)(), class F, S
(*mapping)(F, S), F (*composition)(F, F), F (*id)()>
struct lazy_segtree {
public:
    lazy_segtree() : lazy_segtree(0) {}
    explicit lazy_segtree(int n) : lazy_segtree(vector<S>(n,
e())) {}
    explicit lazy_segtree(const vector<S>& v) :
_n(int(v.size())) {
        size = (int)bit_ceil((unsigned int)(_n));
        log = countr_zero((unsigned int)size);
        d = vector<S>(2 * size, e());
        lz = vector<F>(size, id());
        for(int i = 0; i < _n; i++) d[size + i] = v[i];
        for(int i = size - 1; i >= 1; i--) { update(i); }
    }
    void set(int p, S x) {
        // assert(0 <= p && p < _n);
        p += size;
        for(int i = log; i >= 1; i--) push(p >> i);
        d[p] = x;
        for(int i = 1; i <= log; i++) update(p >> i);
    }
    S get(int p) {
        // assert(0 <= p && p < _n);
        p += size;
        for(int i = log; i >= 1; i--) push(p >> i);
        return d[p];
    }
    S prod(int l, int r) {
        // assert(0 <= l && l <= r && r <= _n);
        if(l == r) return e();
    }

```

```

l += size;
r += size;

for(int i = log; i >= 1; i--) {
    if(((l >> i) << i) != l) push(l >> i);
    if(((r >> i) << i) != r) push((r - 1) >> i);
}

S sm_l = e(), sm_r = e();
while(l < r) {
    if(l & 1) sm_l = op(sm_l, d[l++]);
    if(r & 1) sm_r = op(d[--r], sm_r);
    l >>= 1;
    r >>= 1;
}

return op(sm_l, sm_r);
}

void apply(int l, int r, F f) {
    assert(0 <= l && l <= r && r <= _n);
    if(l == r) return;

    l += size;
    r += size;

    for(int i = log; i >= 1; i--) {
        if(((l >> i) << i) != l) push(l >> i);
        if(((r >> i) << i) != r) push((r - 1) >> i);
    }

    {
        int l2 = l, r2 = r;
        while(l < r) {
            if(l & 1) all_apply(l++, f);
            if(r & 1) all_apply(--r, f);
            l >>= 1;
            r >>= 1;
        }
        l = l2;
        r = r2;
    }

    for(int i = 1; i <= log; i++) {
        if(((l >> i) << i) != l) update(l >> i);
        if(((r >> i) << i) != r) update((r - 1) >> i);
    }
}

template<class G> int max_right(int l, G g) {
    // assert(0 <= l && l <= _n);
    // assert(g(e()));
    if(l == _n) return _n;
    l += size;
    for(int i = log; i >= 1; i--) push(l >> i);
    S sm = e();
    do {
        while(l % 2 == 0) l >>= 1;
        if(!g(op(sm, d[l]))) {
            while(l < size) {
                push(l);
                l = (2 * l);
                if(g(op(sm, d[l]))) {
                    sm = op(sm, d[l]);
                    l++;
                }
            }
            return l - size;
        }
        sm = op(sm, d[l]);
        l++;
    } while((l & -l) != l);
    return _n;
} // d93691

template<class G> int min_left(int r, G g) {
    // assert(0 <= r && r <= _n);
    // assert(g(e()));
    if(r == 0) return 0;
    r += size;
    for(int i = log; i >= 1; i--) push((r - 1) >> i);
    S sm = e();

```

```

do {
    r--;
    while(r > 1 && (r % 2)) r >>= 1;
    if(!g(op(d[r], sm))) {
        while(r < size) {
            push(r);
            r = (2 * r + 1);
            if(g(op(d[r], sm))) {
                sm = op(d[r], sm);
                r--;
            }
        }
        return r + 1 - size;
    }
    sm = op(d[r], sm);
} while((r & -r) != r);
return 0;
} // c9a7eb

private:
int _n, size, log;
vector<S> d;
vector<F> lz;

void update(int k) { d[k] = op(d[2 * k], d[2 * k + 1]); }
void all_apply(int k, F f) {
    d[k] = mapping(f, d[k]);
    if(k < size) lz[k] = composition(f, lz[k]);
}
void push(int k) {
    all_apply(2 * k, lz[k]);
    all_apply(2 * k + 1, lz[k]);
    lz[k] = id();
}
};

```

segtree.hpp

md5: d32488

```

// base: bafcf8
unsigned int bit_ceil(unsigned int n) {
    unsigned int x = 1;
    while(x < (unsigned int)(n)) x *= 2;
    return x;
}
int countr_zero(unsigned int n) { return __builtin_ctz(n); }
constexpr int countr_zero_constexpr(unsigned int n) {
    int x = 0;
    while(!(n & (1 << x))) x++;
    return x;
}
template<class S, S (*op)(S, S), S (*e)()> struct segtree {
public:
    segtree() : segtree(0) {}
    explicit segtree(int n) : segtree(vector<S>(n, e())) {}
    explicit segtree(const vector<S>& v) : _n(int(v.size())) {
        size = (int)bit_ceil((unsigned int)_n);
        log = countr_zero((unsigned int)size);
        d = vector<S>(2 * size, e());
        for(int i = 0; i < _n; i++) d[size + i] = v[i];
        for(int i = size - 1; i >= 1; i--) { update(i); }
    }

    void set(int p, S x) {
        // assert(0 <= p && p < _n);
        p += size;
        d[p] = x;
        for(int i = 1; i <= log; i++) update(p >> i);
    }

    S get(int p) const {
        // assert(0 <= p && p < _n);
        return d[p + size];
    }

    S prod(int l, int r) const {
        // assert(0 <= l && l <= r && r <= _n);
        S sml = e(), smr = e();
        l += size;
        r += size;

        while(l < r) {
            if(l & 1) sml = op(sml, d[l++]);

```

```

            if(r & 1) smr = op(d[--r], smr);
            l >>= 1;
            r >>= 1;
        }
        return op(sml, smr);
    }

    S all_prod() const { return d[1]; }

    template<class F> int max_right(int l, F f) {
        // assert(0 <= l && l <= _n);
        // assert(f(e()));
        if(l == _n) return _n;
        l += size;
        S sm = e();
        do {
            while(l % 2 == 0) l >>= 1;
            if(!f(op(sm, d[l]))) {
                while(l < size) {
                    l = (2 * l);
                    if(f(op(sm, d[l]))) {
                        sm = op(sm, d[l]);
                        l++;
                    }
                }
                return l - size;
            }
            sm = op(sm, d[l]);
            l++;
        } while((l & -l) != l);
        return _n;
    } // faa03f

    template<class F> int min_left(int r, F f) {
        // assert(0 <= r && r <= _n);
        // assert(f(e()));
        if(r == 0) return 0;
        r += size;
        S sm = e();
        do {
            r--;
            while(r > 1 && (r % 2)) r >>= 1;
            if(!f(op(d[r], sm))) {
                while(r < size) {
                    r = (2 * r + 1);
                    if(f(op(d[r], sm))) {
                        sm = op(d[r], sm);
                        r--;
                    }
                }
                return r + 1 - size;
            }
            sm = op(d[r], sm);
        } while((r & -r) != r);
        return 0;
    } // efa466

private:
int _n, size, log;
vector<S> d;

    void update(int k) { d[k] = op(d[2 * k], d[2 * k + 1]); }
};

```

math

BinaryGCD.hpp

md5: f3ab31

```

u64 ctz(u64 x) { return countr_zero(x); }
u64 binary_gcd(u64 x, u64 y) {
    if(!x || !y) return x | y;
    u64 n = ctz(x), m = ctz(y);
    x >>= n, y >>= m;
    while(x != y) {
        if(x > y) x = (x - y) >> ctz(x - y);
        else y = (y - x) >> ctz(y - x);
    }
    return x << min(n, m);
}

```

ExtGCD.hpp

md5: c3fa9b

// returns gcd(a, b) and assign x, y to integers
// s.t. ax + by = gcd(a, b) and |x| + |y| is minimized
ll extgcd(ll a, ll b, ll& x, ll& y) {
 // assert(a >= 0 && b >= 0);
 if(!b) return x = 1, y = 0, a;
 ll d = extgcd(b, a % b, y, x);
 y -= a / b * x;
 return d;
}

floor_sum.hpp

md5: 0f7242

ll floor_sum(const ll& n, const ll& m, ll a, ll b) {
 ll ret = 0;
 if(a >= m) ret += (n - 1) * n * (a / m) / 2, a %= m;
 if(b >= m) ret += n * (b / m), b %= m;
 ll y = (a * n + b) / m;
 if(y == 0) return ret;
 ll x = y * m - b;
 ret += (n - (x + a - 1) / a) * y;
 ret += floor_sum(y, a, m, (a - x % a) % a);
 return ret;
}

modint

BarrettReduction.hpp

md5: 2ca7f3

// using u64 = uint64_t;
struct Barrett { // mod < 2^32
 u64 m, im;
 Barrett(u64 mod) : m(mod), im(-1ULL / m + 1) {}
 // input: a * b < 2^64, output: a * b % mod
 u64 mul(u64 a, u64 b) const {
 a *= b;
 u64 x = ((__uint128_t)a * im) >> 64;
 a -= x * m;
 if((ll)a < 0) a += m;
 return a;
 }
};

modint.hpp

md5: 81b530

const ll mod = 998244353;
struct mm {
 ll x;
 mm(ll x_ = 0) : x(x_ % mod) {
 if(x < 0) x += mod;
 }
 friend mm operator+(mm a, mm b) { return a.x + b.x; }
 friend mm operator-(mm a, mm b) { return a.x - b.x; }
 friend mm operator*(mm a, mm b) { return a.x * b.x; }
 friend mm operator/(mm a, mm b) { return a * b.inv(); }
 // 4 行コピー Alt + Shift + クリックで複数カーソル
 friend mm& operator+=(mm& a, mm b) { return a = a.x + b.x; }
 friend mm& operator-=(mm& a, mm b) { return a = a.x - b.x; }
 friend mm& operator*=(mm& a, mm b) { return a = a.x * b.x; }
 friend mm& operator/=(mm& a, mm b) { return a = a * b.inv(); }
}

mm inv() const { return pow(mod - 2); }
mm pow(ll b) const {
 mm a = *this, c = 1;
 while(b) {
 if(b & 1) c *= a;
 a *= a;
 b >>= 1;
 }
 return c;
}
};

FPS

FFT.hpp

md5: 3138c7

// {998244353, 3}, {1811939329, 13}, {2013265921, 31}
mm g = 3; // 原始根
void fft(vector<mm>& a) {
 ll n = sz(a), lg = __lg(n);
 assert((1 << lg) == n);
 vector<mm> b(n);
 rep(l, 1, lg + 1) {
 ll w = n >> l;
 mm s = 1, r = g.pow(mod >> l);
 for(ll u = 0; u < n / 2; u += w) {
 rep(d, 0, w) {
 mm x = a[u << 1 | d], y = a[u << 1 | w | d] * s;
 b[u | d] = x + y;
 b[n >> 1 | u | d] = x - y;
 }
 s *= r;
 }
 swap(a, b);
 }
}

vector<mm> conv(vector<mm> a, vector<mm> b) {
 if(a.empty() || b.empty()) return {};
 size_t s = sz(a) + sz(b) - 1, n = bit_ceil(s);
 // if(min(sz(a), sz(b)) <= 60) 愚直に掛け算
 a.resize(n);
 b.resize(n);
 fft(a);
 fft(b);
 mm inv = mm(n).inv();
 rep(i, 0, n) a[i] *= b[i] * inv;
 reverse(1 + all(a));
 fft(a);
 a.resize(s);
 return a;
}

FFT_fast.hpp

md5: c8c567

// modint を u32 にして加減算を真面目にやると速い
mm g = 3; // 原始根
void fft(vector<mm>& a) {
 ll n = sz(a), lg = __lg(n);
 static auto z = [] {
 vector<mm> z(30);
 mm s = 1;
 rep(i, 2, 32) {
 z[i - 2] = s * g.pow(mod >> i);
 s *= g.inv().pow(mod >> i);
 }
 return z;
 }();
 rep(l, 0, lg) {
 ll w = 1 << (lg - l - 1);
 mm s = 1;
 rep(k, 0, 1 << l) {
 ll o = k << (lg - l);
 rep(i, o, o + w) {
 mm x = a[i], y = a[i + w] * s;
 a[i] = x + y;
 a[i + w] = x - y;
 }
 s *= z[countr_zero<uint64_t>(~k)];
 }
 }
}

// コピー
void ifft(vector<mm>& a) {
 ll n = sz(a), lg = __lg(n);
 static auto z = [] {
 vector<mm> z(30);
 mm s = 1;
 rep(i, 2, 32) { // g を逆数に
 z[i - 2] = s * g.inv().pow(mod >> i);
 s *= g.pow(mod >> i);
 }
 return z;
 }();
}

```

    }();
    for(ll l = lg; l--;) { // 逆順に
        ll w = 1 << (lg - l - 1);
        mm s = 1;
        rep(k, 0, 1 << l) {
            ll o = k << (lg - l);
            rep(i, o, o + w) {
                mm x = a[i], y = a[i + w]; // *s を下に移動
                a[i] = x + y;
                a[i + w] = (x - y) * s;
            }
            s *= z[countr_zero<uint64_t>(~k)];
        }
    }
}

vector<mm> conv(vector<mm> a, vector<mm> b) {
    if(a.empty() || b.empty()) return {};
    size_t s = sz(a) + sz(b) - 1, n = bit_ceil(s);
    // if(min(sz(a), sz(b)) <= 60) 愚直に掛け算
    a.resize(n);
    b.resize(n);
    fft(a);
    fft(b);
    mm inv = mm(n).inv();
    rep(i, 0, n) a[i] *= b[i] * inv;
    ifft(a);
    a.resize(s);
    return a;
}

```

graph

low_link.hpp

md5: d5d2f7

```

struct LowLink {
public:
    vector<vector<int>>> g;
    vector<int> ord, low;
    vector<int> articulation;
    vector<bool> visited;
    vector<pair<int, int>>> bridge;

    void dfs(int cur, int pre, int& k) {
        visited[cur] = true;
        ord[cur] = low[cur] = k++;
        bool isArticulation = false;
        int cnt = 0;
        for(auto to : g[cur]) {
            if(!visited[to]) {
                cnt++;
                dfs(to, cur, k);
                chmin(low[cur], low[to]);
                if(pre != -1 && ord[cur] <= low[to]) isArticulation
= true;
                if(ord[cur] < low[to]) bridge.emplace_back(min(cur,
to), max(cur, to));
            } else if(to != pre) chmin(low[cur], ord[to]);
        }
        if(pre == -1 && cnt > 1) isArticulation = true;
        if(isArticulation) articulation.push_back(cur);
    }

    void build(const vector<vector<int>>>& g) {
        int n = g.size();
        this->g = g;
        ord.assign(n, -1);
        low.assign(n, -1);
        visited.assign(n, false);
        int k = 0;
        for(int i = 0; i < n; i++)
            if(!visited[i]) dfs(i, -1, k);
    }

    LowLink(const vector<vector<int>>>& g) { build(g); }

    vector<int>& getArticulations() { return articulation; }
    vector<pair<int, int>>& getBridges() { return bridge; }
    vector<int>& getOrd() { return ord; }
    vector<int>& getLowLink() { return low; }
};

```

max_flow.hpp

md5: dad0c8

```

// base: 89e8d1
template<class Cap> struct mf_graph {
public:
    mf_graph() : _n(0) {}
    mf_graph(int n) : _n(n), g(n) {}

    int add_edge(int from, int to, Cap cap) {
        // assert(0 <= from && from < _n);
        // assert(0 <= to && to < _n);
        // assert(0 <= cap);
        int m = sz(pos);
        pos.push_back({from, sz(g[from])});
        int from_id = sz(g[from]);
        int to_id = sz(g[to]);
        if(from == to) to_id++;
        g[from].push_back(_edge{to, to_id, cap});
        g[to].push_back(_edge{from, from_id, 0});
        return m;
    }

    Cap flow(int s, int t, Cap flow_limit =
numeric_limits<Cap>::max()) {
        // assert(0 <= s && s < _n);
        // assert(0 <= t && t < _n);
        // assert(s != t);

        vector<int> level(_n), iter(_n);
        queue<int> que;
        auto bfs = [&]() {
            fill(all(level), -1);
            level[s] = 0;
            while(!que.empty()) que.pop();
            que.push(s);
            while(!que.empty()) {
                int v = que.front();
                que.pop();
                for(auto e : g[v]) {
                    if(e.cap == 0 || level[e.to] >= 0) continue;
                    level[e.to] = level[v] + 1;
                    if(e.to == t) return;
                    que.push(e.to);
                }
            }
        };
        auto dfs = [&](auto self, int v, Cap up) {
            if(v == s) return up;
            Cap res = 0;
            int level_v = level[v];
            for(int& i = iter[v]; i < sz(g[v]); i++) {
                _edge& e = g[v][i];
                if(level_v <= level[e.to] || g[e.to][e.rev].cap ==
0) continue;
                Cap d = self(self, e.to, min(up - res, g[e.to]
[e.rev].cap));
                if(d <= 0) continue;
                g[v][i].cap -= d;
                g[e.to][e.rev].cap += d;
                res += d;
                if(res == up) break;
            }
            return res;
        };

        Cap flow = 0;
        while(flow < flow_limit) {
            bfs();
            if(level[t] == -1) break;
            fill(all(iter), 0);
            while(flow < flow_limit) {
                Cap f = dfs(dfs, t, flow_limit - flow);
                if(!f) break;
                flow += f;
            }
        }
        return flow;
    }

    vector<bool> min_cut(int s) {
        vector<bool> visited(_n);
    }
};

```

```

    queue<int> que;
    que.push(s);
    visited[s] = true;
    while(!que.empty()) {
        int v = que.front();
        que.pop();
        for(auto e : g[v]) {
            if(e.cap && !visited[e.to]) {
                visited[e.to] = true;
                que.push(e.to);
            }
        }
    }
    return visited;
} // 8735cf

struct edge {
    int from, to;
    Cap cap, flow;
}; // 9fe107

edge get_edge(int i) {
    int m = sz(pos);
    // assert(0 <= i && i < m);
    auto _e = g[pos[i].first][pos[i].second];
    auto _re = g[_e.to][_e.rev];
    return edge{pos[i].first, _e.to, _e.cap + _re.cap,
_re.cap};
} // 8cbb00

vector<edge> edges() {
    int m = sz(pos);
    vector<edge> result;
    rep(i, 0, m) result.push_back(get_edge(i));
    return result;
} // fa2b7d

void change_edge(int i, Cap new_cap, Cap new_flow) {
    int m = int(pos.size());
    // assert(0 <= i && i < m);
    // assert(0 <= new_flow && new_flow <= new_cap);
    auto& _e = g[pos[i].first][pos[i].second];
    auto& _re = g[_e.to][_e.rev];
    _e.cap = new_cap - new_flow;
    _re.cap = new_flow;
} // 025616

private:
int _n;
struct _edge {
    int to, rev;
    Cap cap;
};
vector<pair<int, int>> pos;
vector<vector<_edge>> g;
};

```

min_cost_flow.hpp

md5: 560d2d

```

// base: 4756c7
template<class Cap, class Cost> struct mcf_graph {
public:
    mcf_graph() {}
    mcf_graph(int n) : _n(n), g(n) {}

    int add_edge(int from, int to, Cap cap, Cost cost) {
        // assert(0 <= from && from < _n);
        // assert(0 <= to && to < _n);
        int m = sz(pos);
        pos.push_back({from, sz(g[from])});
        int from_id = sz(g[from]);
        int to_id = sz(g[to]);
        if(from == to) to_id++;
        g[from].push_back(_edge{to, to_id, cap, cost});
        g[to].push_back(_edge{from, from_id, 0, -cost});
        return m;
    }

    pair<Cap, Cost> flow(int s, int t, Cap flow_limit =
numeric_limits<Cap>::max()) {
        return slope(s, t, flow_limit).back();
    }
};

```

```

    vector<pair<Cap, Cost>> slope(int s, int t, Cap flow_limit =
numeric_limits<Cap>::max()) {
        // assert(0 <= s && s < _n);
        // assert(0 <= t && t < _n);
        // assert(s != t);
        vector<Cost> dual(_n, 0), dist(_n);
        vector<int> pv(_n), pe(_n);
        vector<bool> vis(_n);
        auto dual_ref = [&]() {
            fill(all(dist), numeric_limits<Cost>::max());
            fill(all(pv), -1);
            fill(all(pe), -1);
            fill(all(vis), false);
        };
        struct Q {
            Cost key;
            int to;
            bool operator<(const Q& r) const { return key >
r.key; }
        };
        priority_queue<Q> que;
        dist[s] = 0;
        que.push(Q{0, s});
        while(!que.empty()) {
            int v = que.top().to;
            que.pop();
            if(vis[v]) continue;
            vis[v] = true;
            if(v == t) break;
            rep(i, 0, sz(g[v])) {
                auto e = g[v][i];
                if(vis[e.to] || !e.cap) continue;
                Cost cost = e.cost - dual[e.to] + dual[v];
                if(chmin(dist[e.to], dist[v] + cost)) {
                    pv[e.to] = v;
                    pe[e.to] = i;
                    que.push(Q{dist[e.to], e.to});
                }
            }
        }
        if(!vis[t]) return false;
        rep(v, 0, _n) if(vis[v]) dual[v] -= dist[t] - dist[v];
        return true;
    };
    Cap flow = 0;
    Cap cost = 0, prev_cost_per_flow = -1;
    vector<pair<Cap, Cost>> result;
    result.push_back({flow, cost});
    while(flow < flow_limit) {
        if(!dual_ref()) break;
        Cap c = flow_limit - flow;
        for(int v = t; v != s; v = pv[v]) { c = min(c,
g[pv[v]][pe[v]].cap); }
        for(int v = t; v != s; v = pv[v]) {
            auto& e = g[pv[v]][pe[v]];
            e.cap -= c;
            g[v][e.rev].cap += c;
        }
        Cost d = -dual[s];
        flow += c;
        cost += c * d;
        if(prev_cost_per_flow == d) { result.pop_back(); }
        result.push_back({flow, cost});
        prev_cost_per_flow = d;
    }
    return result;
}

struct edge {
    int from, to;
    Cap cap, flow;
}; // 9fe107

edge get_edge(int i) {
    int m = sz(pos);
    // assert(0 <= i && i < m);
    auto _e = g[pos[i].first][pos[i].second];
    auto _re = g[_e.to][_e.rev];
    return edge{pos[i].first, _e.to, _e.cap + _re.cap,
_re.cap};
} // d8c44b

vector<edge> edges() {

```



```

    int m = sz(pos);
    vector<edge> result;
    rep(i, 0, m) result.push_back(get_edge(i));
    return result;
} // fa2b7d

void change_edge(int i, Cap new_cap, Cap new_flow) {
    int m = int(pos.size());
    // assert(0 <= i && i < m);
    // assert(0 <= new_flow && new_flow <= new_cap);

    auto& _e = g[pos[i].first][pos[i].second];
    auto& _re = g[_e.to][_e.rev];
    _e.cap = new_cap - new_flow;
    _re.cap = new_flow;
} // 025616

private:
int _n;
struct _edge {
    int to, rev;
    Cap cap;
    Cost cost;
};

vector<pair<int, int>> pos;
vector<vector<_edge>> g;
};

```

scc.hpp

md5: ec4119

```

// base: 1e9c3f
struct scc_graph {
    public:
    explicit scc_graph(int _n = 0) : n(_n), G(_n), rG(_n),
    comp(_n, -1), visited(_n, 0) {}

    void add_edge(int from, int to) {
        // assert(0 <= from && from < n);
        // assert(0 <= to && to < n);
        G[from].push_back(to);
        rG[to].push_back(from);
    }

    vector<vector<int>> scc() {
        fill(all(visited), 0);
        fill(all(comp), -1);
        order.clear();
        rep(i, 0, n) if(!visited[i]) dfs(i);
        comp_size = 0;
        for(int i = sz(order) - 1; i >= 0; i--) {
            if(comp[order[i]] < 0) rdfs(order[i], comp_size++);
        }
        vector<vector<int>> v(comp_size);
        rep(i, 0, n) v[comp[i]].push_back(i);
        return v;
    }

    vector<int> get_comp() { return comp; } // bdafc0

    vector<vector<int>> dag() {
        vector<vector<int>> res(comp_size);
        rep(i, 0, n) for(auto j : G[i]) {
            if(comp[i] != comp[j])
                res[comp[i]].push_back(comp[j]);
        }
        rep(i, 0, comp_size) {
            sort(all(res[i]));
            res[i].erase(unique(all(res[i])), res[i].end());
        }
        return res;
    } // da3a19

private:
vector<vector<int>> G, rG;
vector<int> order, comp;
vector<bool> visited;
int n, comp_size;

void dfs(int v) {
    visited[v] = true;
    for(auto to : G[v])

```

```

        if(!visited[to]) dfs(to);
        order.push_back(v);
    }

    void rdfs(int v, int k) {
        comp[v] = k;
        for(auto to : rG[v]) {
            if(comp[to] < 0) rdfs(to, k);
        }
    }
};

```

two_sat.hpp

md5: 054791

```

struct two_sat {
    public:
    two_sat() : _n(0), scc(0) {}
    two_sat(int n) : _n(n), scc(2 * n), _answer(n) {}

    void add_clause(int i, bool f, int j, bool g) {
        // assert(0 <= i && i < _n);
        // assert(0 <= j && j < _n);
        scc.add_edge(2 * i + (f ? 0 : 1), 2 * j + (g ? 1 : 0));
        scc.add_edge(2 * j + (g ? 0 : 1), 2 * i + (f ? 1 : 0));
    }

    bool satisfiable() {
        scc.scc();
        auto comp = scc.get_comp();
        rep(i, 0, _n) {
            if(comp[2 * i] == comp[2 * i + 1]) return false;
            _answer[i] = comp[2 * i] < comp[2 * i + 1];
        }
        return true;
    }

    vector<bool> answer() { return _answer; }

private:
int _n;
vector<bool> _answer;
scc_graph scc;
};

```

graph/tree

hld.hpp

md5: 10247f

```

class HLDcomposition {
    private:
    int V;
    vector<vector<int>> G;
    vector<int> stsize, parent, pathtop, in, out;
    int root;
    void build_stsize(int u, int p) {
        stsize[u] = 1, parent[u] = p;
        for(auto&& v : G[u]) {
            if(v == p) {
                if(v == G[u].back()) break;
                else swap(v, G[u].back());
            }
            build_stsize(v, u);
            stsize[u] += stsize[v];
            if(stsize[v] > stsize[G[u][0]]) swap(v, G[u][0]);
        }
    }

    void build_path(int u, int p, int& tm) {
        in[u] = tm++;
        for(auto v : G[u]) {
            if(v == p) continue;
            pathtop[v] = (v == G[u][0] ? pathtop[u] : v);
            build_path(v, u, tm);
        }
        out[u] = tm;
    }

    public:
    void add_edge(int u, int v) {
        G[u].push_back(v);
        G[v].push_back(u);
    }
};

```



```

}

void build(int _root = 0) {
    root = _root;
    int tm = 0;
    build_stsize(root, -1);
    pathtop[root] = root;
    build_path(root, -1, tm);
}

inline int index(int a) { return in[a]; }

int lca(int a, int b) {
    int pa = pathtop[a], pb = pathtop[b];
    while(pa != pb) {
        if(in[pa] > in[pb]) {
            a = parent[pa], pa = pathtop[a];
        } else {
            b = parent[pb], pb = pathtop[b];
        }
    }
    if(in[a] > in[b]) swap(a, b);
    return a;
}

pair<int, int> subtree_query(int a) { return {in[a], out[a]}; }

vector<tuple<int, int, bool>> path_query(int from, int to) {
    int pf = pathtop[from], pt = pathtop[to];
    using T = tuple<int, int, bool>;
    deque<T> front, back;
    while(pf != pt) {
        if(in[pf] > in[pt]) {
            front.push_back({in[pf], in[from] + 1, true});
            from = parent[pf], pf = pathtop[from];
        } else {
            back.push_front({in[pt], in[to] + 1, false});
            to = parent[pt], pt = pathtop[to];
        }
    }
    if(in[from] > in[to]) front.push_back({in[to], in[from] + 1, true});
    else front.push_back({in[from], in[to] + 1, false});
    vector<T> ret;
    while(!front.empty()) {
        ret.push_back(front.front());
        front.pop_front();
    }
    while(!back.empty()) {
        ret.push_back(back.front());
        back.pop_front();
    }
    return ret;
}

HLDcomposition(int node_size)
    : V(node_size), G(V), stsize(V, 0), parent(V, -1),
  pathtop(V, -1), in(V, -1), out(V, -1) {}
};
```

flow

燃やす埋める.md

変形前の制約	変形後の制約
x が 0 のとき z 失う	(x, T, z)
x が 0 のとき z 得る	無条件で z 得る; (S, x, z)
x が 1 のとき z 失う	(S, x, z)
x が 1 のとき z 得る	無条件で z 得る; (x, T, z)
x, y, \dots がすべて 0 のとき z 得る	無条件で z 得る; $(S, w, z), (w, x, \infty), (w, y, \infty)$
x, y, \dots がすべて 1 のとき z 得る	無条件で z 得る; $(w, T, z), (x, w, \infty), (y, w, \infty)$

string

KMP.hppmd5: 886c63

```
// kmp[i] := max{ l ≤ i | s[:l] == s[(i+1)-l:i+1] }
// abacaba -> 0010123
auto KMP(string s) {
    vector<ll> p(sz(s));
    rep(i, 1, sz(s)) {
        ll g = p[i - 1];
        while(g && s[i] != s[g]) g = p[g - 1];
        p[i] = g + (s[i] == s[g]);
    }
    return p;
}
```

Manacher.hppmd5: 5882fb

```
// 各位置での回文半径を求める
// aaabaaa -> 1214121
// 偶数長の回文を含めて直径を知るには、N+1 個の $ を挿入して 1 を引く
// $a$a$a$b$a$a$a$a$ -> 123432181234321
auto manacher(string s) {
    ll n = sz(s), i = 0, j = 0;
    vector<ll> r(n);
    while(i < n) {
        while(i >= j && i + j < n && s[i - j] == s[i + j]) j++;
        r[i] = j;
        ll k = 1;
        while(i >= k && i + k < n && k + r[i - k] < j) {
            r[i + k] = r[i - k];
            k++;
        }
        i += k, j -= k;
    }
    return r;
}
```

RollingHash.hppmd5: adb8d3

```
// using u64 = uint64_t;
const u64 mod = INF;
u64 add(u64 a, u64 b) {
    a += b;
    if(a >= mod) a -= mod;
    return a;
}
u64 mul(u64 a, u64 b) {
    auto c = (__uint128_t)a * b;
    return add(c >> 61, c & mod);
}
random_device rnd;
const u64 r = ((u64)rnd() << 32 | rnd()) % mod;
struct RH {
    ll n;
    vector<u64> hs, pw;
    RH(string s) : n(sz(s)), hs(n + 1), pw(n + 1, 1) {
        rep(i, 0, n) {
            pw[i + 1] = mul(pw[i], r);
            hs[i + 1] = add(mul(hs[i], r), s[i]);
        }
    }
    u64 get(ll l, ll r) const { return add(hs[r], mod - mul(hs[l], pw[r - l])); }
};
```

SuffixArray.hppmd5: 1d70ce

```
// returns pair{sa, lcp}
// sa 長さ n : s[sa[0]:] < s[sa[1]:] < ... < s[sa[n-1]:]
// lcp 長さ n-1 : lcp[i] = LCP(s[sa[i]:], s[sa[i+1]:])
auto SA(string s) {
    ll n = sz(s) + 1, lim = 256;
    // assert(lim > ranges::max(s));
    vector<ll> sa(n), lcp(n), x(all(s) + 1), y(n), ws(max(n, lim)), rk(n);
    iota(all(sa), 0);
    for(ll j = 0, p = 0; p < n; j = max(1LL, j * 2), lim = p) {
        p = j;
```

```
        iota(all(y), n - j);
        rep(i, 0, n) if(sa[i] >= j) y[p++] = sa[i] - j;
        fill(all(ws), 0);
        rep(i, 0, n) ws[x[i]]++;
        rep(i, 1, lim) ws[i] += ws[i - 1];
        for(ll i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
        swap(x, y);
        p = 1;
        x[sa[0]] = 0;
        rep(i, 1, n) {
            ll a = sa[i - 1], b = sa[i];
            x[b] = (y[a] == y[b] && y[a + j] == y[b + j]) ? p - 1
: p++;
        }
    }
    rep(i, 1, n) rk[sa[i]] = i;
    for(ll i = 0, k = 0; i < n - 1; lcp[rk[i++]] = k) {
        if(k) k--;
        while(s[i + k] == s[sa[rk[i] - 1] + k]) k++;
    }
    sa.erase(begin(sa));
    lcp.erase(begin(lcp));
    return pair{sa, lcp};
}
```

Zalgorithm.hpp

md5: b20b04

```
// Z[i] := LCP(s, s[i:])
// abacaba -> 7010301
auto Z(string s) {
    ll n = sz(s), l = -1, r = -1;
    vector<ll> z(n, n);
    rep(i, 1, n) {
        ll& x = z[i] = i < r ? min(r - i, z[i - l]) : 0;
        while(i + x < n && s[i + x] == s[x]) x++;
        if(i + x > r) l = i, r = i + x;
    }
    return z;
}
```

algorithm
geometry
memo

Primes.md

素数の個数

n	10^2	10^3	10^4	10^5	10^6	10^7	10^8	10^9	10^{10}
$\pi(n)$	25	168	1229	9592	78498	664579	5.76e+6	5.08e+7	4.55e+8

高度合成数

$\leq n$	10^3	10^4	10^5	10^6	10^7	10^8	10^9		
x	840	7560	83160	720720	8648640	73513440	735134400		
$d^0(x)$	32	64	128	240	448	768	1344		
$\leq n$	10^{10}	10^{11}	10^{12}	10^{13}	10^{14}	10^{15}	10^{16}	10^{17}	10^{18}
$d^0(x)$	2304	4032	6720	10752	17280	26880	41472	64512	103680

素数階乗

n	2	3	5	7	11	13	17	19	23	29
$n\#$	2	6	30	210	2310	30030	510510	9.70e+6	2.23e+8	6.47e+9

階乗

4!	5!	6!	7!	8!	9!	10!	11!	12!	13!
24	120	720	5040	40320	362880	3.63e+6	3.99e+7	4.79e+8	6.23e+9