

ICPC Notebook

template	
hash.sh	1
settings.sh	1
template.hpp	1
data-structure	
BIT.hpp	1
FastSet.hpp	1
SparseTable.hpp	2
waveletmatrix.hpp	2
math	
BinaryGCD.hpp	3
CHT.hpp	3
Combination.hpp	3
Eratosthenes.hpp	4
ExtGCD.hpp	4
Matrix.hpp	4
floorsum.hpp	4
miller_rabin.hpp	5
xor_set.hpp	5
modint	
modint.hpp	5
FPS	
FFT.hpp	5
FFT_fast.hpp	6
graph	
2SAT.hpp	6
scc.hpp	6
graph/tree	
lca.hpp	7
flow	
dinic.hpp	7
mincostflow.hpp	8
燃やす埋める.md	8
string	
KMP.hpp	8
Manacher.hpp	8
RollingHash.hpp	8
SuffixArray.hpp	9
Zalgorithm.hpp	9
algorithm	
mo.hpp	9
geometry	
geometry.hpp	9
memo	
Primes.md	11

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: 9df547

```
#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 rrep(i, a, n) for(ll i = a; i >= n; 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;
}
};
```

SparseTable.hpp md5: acd1f4

```
template<typename T> struct SparseTable {
    vector<vector<T>> > st;
    vector<int> lookup;

    SparseTable(const vector<T>& v) {
        int b = 0;
        while((1 << b) <= v.size()) ++b;
        st.assign(b, vector<T>(1 << b));
        for(int i = 0; i < v.size(); i++) { st[0][i] = v[i]; }
        for(int i = 1; i < b; i++) {
            for(int j = 0; j + (1 << i) <= (1 << b); j++) { st[i][
[j] = min(st[i - 1][j], st[i - 1][j + (1 << (i - 1))]); }
            }
            lookup.resize(v.size() + 1);
            for(int i = 2; i < lookup.size(); i++) { lookup[i] =
lookup[i >> 1] + 1; }
        }

        inline T rmq(int l, int r) {
            int b = lookup[r - l];
            return min(st[b][l], st[b][r - (1 << b)]);
        }
    }
};
```

waveletmatrix.hpp md5: 3b1bf0

```
// i桁目のビットが1かどうか
bool has_bit(ll x, int i) { return (x >> i) & 1; }

// 長さnの静的なビット列に対して累積和ができるデータ構造
class BitCumulativeSum {
    // 64桁ごとに累積和を作る
    inline static constexpr int w = 64;

    vector<uint64_t> block; // ビット列をwごとに保持
    vector<int> sum; // 累積和

public:
    BitCumulativeSum() = default;
```

```
    BitCumulativeSum(int n) : block(n / w + 1, 0), sum(1, 0) {
        sum.reserve(block.size() + 1); // 事前に要素数分のメモリを確保しておく（このサイズになっているわけではない）
    }

    // i桁目のビットを立てる
    void set(int i) { block[i / w] |= 1LL << (i % w); }

    // 累積和を作成
    void build() {
        for(const auto& b : block) {
            // popcount : 2進数表記で1の数を数える
            sum.push_back(sum.back() + popcount(b));
        }
    }

    // [0, r) 桁までの1の個数
    int rank1(int r) const { return sum[r / w] +
popcount(block[r / w] & ((1LL << (r % w)) - 1)); }

    // [0, r) 桁まででの0の個数
    int rank0(int r) const { return r - rank1(r); }
};

// 本題
class WaveletMatrix {
    int n, sigma;
    vector<BitCumulativeSum> bv;

public:
    WaveletMatrix(vector<int> v) : n((int)v.size()) {
        // sigmaを決定する
        int mx = 0;
        for(auto& x : v) {
            assert(x >= 0);
            mx = max(mx, x);
        }
        sigma = 0;
        while((1LL << sigma) - 1 < mx) sigma++;

        // 行列の構築
        bv.assign(sigma, n);
        vector<int> nxt_v(n);
        // 上位の桁から構築していく
        for(int h = sigma - 1; h >= 0; h--) {
            auto& B = bv[h]; // h桁目に対応するビットの累積和（ただし0
の個数を数える）

            // vでh桁目が0の要素を左に、1の要素を右に寄せる
            int l = 0, r = n - 1;
            // 1の方を寄せる
            for(int i = n - 1; i >= 0; i--) {
                if(has_bit(v[i], h)) {
                    B.set(i);
                    nxt_v[r--] = v[i];
                }
            }

            B.build(); // 累積和を構築

            // 0の方も寄せる
            for(int i = 0; i < n; i++) {
                if(!has_bit(v[i], h)) { nxt_v[l++] = v[i]; }
            }
            swap(v, nxt_v);
        }
    }

    // [l, r) でk番目に小さい数 (0-indexed)
    int kth_smallest(int l, int r, int k) const {
        assert(0 <= k && k < n);
        uint32_t res = 0;
        // 上位の桁から0か1を決定していく
        for(int h = sigma - 1; h >= 0; h--) {
            const auto& B = bv[h];
            int zero_cnt = B.rank0(r) - B.rank0(l); // 区間のビット
0の個数

            if(k >= zero_cnt) {
                // h桁目が1の場合
```

```
        res |= 1 << h;
        k -= zero_cnt;
        // 区間の更新
        // h桁目が0の要素が左、1の要素が右によっていることを利用する
        l = B.rank0(n) + B.rank1(l);
        r = B.rank0(n) + B.rank1(r);
    } else {
        // h桁目が0の場合
        l = B.rank0(l);
        r = B.rank0(r);
    }
}
return res;
}

// [l, r) でk番目に小さい数 (0-indexed)
int kth_largest(int l, int r, int k) const {
    assert(0 <= r - l - k + 1 && r - l - k + 1 < n);
    return kth_smallest(l, r, r - l - k + 1);
}

// [0, r) でu未満の値の個数
int range_freq(int r, int u) {
    assert(u >= 0);
    if(u >= (1LL << sigma)) return r;

    int l = 0, ret = 0;
    for(int h = sigma - 1; h >= 0; --h) {
        auto& B = bv[h];
        if(has_bit(u, h)) {
            // h桁目が1の場合
            ret += B.rank0(r) - B.rank0(l); // 区間に属しているh
桁目が0の要素はu未満
            l = B.rank0(n) + B.rank1(l);
            r = B.rank0(n) + B.rank1(r);
        } else {
            // h桁目が0の場合
            l = B.rank0(l);
            r = B.rank0(r);
        }
    }

    return ret;
}

// [l, r) でu未満の値の個数
int range_freq(int l, int r, int u) {
    assert(u >= 0);
    return range_freq(r, u) - range_freq(l, u);
}

// [l, r) でd以上u未満の値の個数
int range_freq(int l, int r, int d, int u) {
    assert(d >= 0 && u >= 0);
    return range_freq(l, r, u) - range_freq(l, r, d);
}
};
```

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);
}

CHT.hpp

md5: b42f8f

template<typename T> class CHT {
private:
 struct node {
 node *left, *right;

```
        static const T inf = numeric_limits<T>::max();
        T a, b;
        node() : node(0, inf) {}
        node(const T _a, const T _b) : left(nullptr),
right(nullptr), a(_a), b(_b) {}
        T f(const T x) const { return a * x + b; }

    };
    static void swap(node* x, node* y) { std::swap(x->a, y->a),
std::swap(x->b, y->b); }
    void _add_line(node* cur, node* nw, T l, T r) {
        while(true) {
            if(nw->f(l) < cur->f(l)) swap(cur, nw);
            if(cur->f(r - 1) <= nw->f(r - 1)) break;
            const T mid = (l + r) / 2;
            if(cur->f(mid) <= nw->f(mid)) {
                if(!cur->right) {
                    cur->right = new node(*nw);
                    break;
                } else {
                    cur = cur->right, l = mid;
                }
            } else {
                swap(cur, nw);
                if(!cur->left) {
                    cur->left = new node(*nw);
                    break;
                } else {
                    cur = cur->left, r = mid;
                }
            }
        }
    }
}
T query(node* cur, const T k, T l, T r) const {
    T ans = numeric_limits<T>::max();
    while(cur) {
        ans = min(ans, cur->f(k));
        const T mid = (l + r) / 2;
        if(k < mid) {
            cur = cur->left, r = mid;
        } else {
            cur = cur->right, l = mid;
        }
    }
    return ans;
}

void clear(node* cur) {
    if(cur->left) clear(cur->left);
    if(cur->right) clear(cur->right);
    delete cur;
}

const T lpos, rpos;
node* root;

public:
    CHT(const T _lpos, const T _rpos) : lpos(_lpos),
rpos(_rpos), root(new node()) { assert(lpos < rpos); }
    // ~CHT(){ clear(root); }
    // f(x) = a * x + b を挿入
    void add_line(const T a, const T b) {
        node nw(a, b);
        return _add_line(root, &nw, lpos, rpos);
    }

    // x = k での最小値
    T query(const T k) const { return query(root, k, lpos,
rpos); }
};
```

Combination.hpp

md5: a88ecc

int maxnum = 200005;
vector<ll> fac(maxnum), inv(maxnum), finv(maxnum);
void init_fac() {
 fac[0] = fac[1] = 1;
 inv[1] = 1;
 finv[0] = finv[1] = 1;
 rep(i, 2, maxnum) {
 fac[i] = fac[i - 1] * i % MOD;
 inv[i] = MOD - MOD / i * inv[MOD % i] % MOD;
 finv[i] = finv[i - 1] * inv[i] % MOD;
 }
}

ll nCr(ll n, ll r) {

```
if(n < 0 or n - r < 0 or r < 0) return 0;
return fac[n] * (finv[n - r] * finv[r] % MOD) % MOD;
}
```

Eratosthenes.hpp

md5: 880810

```
int max_num = 1000005;
vector<int> erat(max_num);
void init_e() {
    rep(i, 2, sqrt(max_num - 1)) {
        if(erat[i] == 0) {
            for(int j = i * i; j <= max_num - 1; j += i) {
                if(erat[j] == 0) erat[j] = i;
            }
        }
    }
}
```

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;
}
```

Matrix.hpp

md5: 25fbc4

```
template<class T> struct Matrix {
    vector<vector<T>>> A;

    Matrix() {}

    Matrix(size_t n, size_t m) : A(n, vector<T>(m, 0)) {}

    Matrix(size_t n) : A(n, vector<T>(n, 0)){};

    size_t height() const { return A.size(); }

    size_t width() const { return A[0].size(); }

    inline const vector<T>& operator[](ll k) const { return A.at(k); }

    inline vector<T>& operator[](ll k) { return A.at(k); }

    static Matrix I(size_t n) {
        Matrix mat(n);
        for(ll i = 0; i < n; i++) mat[i][i] = 1;
        return mat;
    }

    Matrix& operator+=(const Matrix& B) {
        size_t n = height(), m = width();
        assert(n == B.height() && m == B.width());
        for(ll i = 0; i < n; i++)
            for(ll j = 0; j < m; j++) (*this)[i][j] += B[i][j];
        return (*this);
    }

    Matrix& operator-=(const Matrix& B) {
        size_t n = height(), m = width();
        assert(n == B.height() && m == B.width());
        for(ll i = 0; i < n; i++)
            for(ll j = 0; j < m; j++) (*this)[i][j] -= B[i][j];
        return (*this);
    }

    Matrix& operator*=(const Matrix& B) {
        size_t n = height(), m = B.width(), p = width();
        assert(p == B.height());
        vector<vector<T>>> C(n, vector<T>(m, 0));
        for(ll i = 0; i < n; i++)
            for(ll j = 0; j < m; j++)
                for(ll k = 0; k < p; k++) C[i][j] = (C[i][j] +
(*this)[i][k] * B[k][j]);
        A.swap(C);
    }
}
```

```
return (*this);
}

Matrix& operator^=(long long k) {
    Matrix B = Matrix::I(height());
    while(k > 0) {
        if(k & 1) B *= *this;
        *this *= *this;
        k >>= 1LL;
    }
    A.swap(B.A);
    return (*this);
}

Matrix operator+(const Matrix& B) const { return
(Matrix(*this) += B); }

Matrix operator-(const Matrix& B) const { return
(Matrix(*this) -= B); }

Matrix operator*(const Matrix& B) const { return
(Matrix(*this) *= B); }

Matrix operator^(const long long k) const { return
(Matrix(*this) ^= k); }

// 行列式
T determinant() {
    Matrix B(*this);
    assert(width() == height());
    T ret = 1;
    for(ll i = 0; i < width(); i++) {
        ll idx = -1;
        for(ll j = i; j < width(); j++) {
            if(B[j][i] != 0) idx = j;
        }
        if(idx == -1) return (0);
        if(i != idx) {
            ret *= -1;
            swap(B[i], B[idx]);
        }
        ret *= B[i][i];
        T vv = B[i][i];
        for(ll j = 0; j < width(); j++) { B[i][j] /= vv; }
        for(ll j = i + 1; j < width(); j++) {
            T a = B[j][i];
            for(ll k = 0; k < width(); k++) { B[j][k] -= B[i]
[k] * a; }
        }
        return (ret);
    }
    ll rank() {
        vector<vector<T>>> B = A;
        ll n = B.size();
        ll m = B[0].size();
        rep(i, 0, n) {
            ll id = i;
            rep(j, i + 1, n) if(B[id] < B[j]) id = j;
            swap(B[i], B[id]);
            ll r = -1;
            rep(j, i, m) if(B[i][j]) {
                r = j;
                break;
            }
            if(r == -1) return i;
            rep(j, 0, n) {
                if(i == j) continue;
                if(B[j][r]) { rep(k, i, m) B[j][k] ^= B[i][k]; }
            }
        }
        return n;
    }
};
```

floorsum.hpp

md5: 76b016

```
long long floor_sum(long long n, long long m, long long a, long
long b) {
    long long ans = 0;
    if(a >= m) {
        ans += (n - 1) * n * (a / m) / 2;
```

```
        a %= m;
    }
    if(b >= m) {
        ans += n * (b / m);
        b %= m;
    }

    long long y_max = (a * n + b) / m, x_max = (y_max * m - b);
    if(y_max == 0) return ans;
    ans += (n - (x_max + a - 1) / a) * y_max;
    ans += floor_sum(y_max, a, m, (a - x_max % a) % a);
    return ans;
}
```

millerrabin.hpp

md5: b6e3d4

```
ll mod_pow(ll a, ll b, ll m) {
    ll res = 1;
    a %= m;
    while(b > 0) {
        if(b & 1) res = __uint128_t(res) * a % m;
        a = __uint128_t(a) * a % m;
        b >>= 1;
    }
    return res;
}

// num が素数なら true, そうでなければ false (計算量 O(log{num}^3))
bool miller_rabin(ll num) {
    const vector<ll> A = {2, 325, 9375, 28178, 450775, 9780504,
1795265022};

    // これらは例外的に判定が必要 (なぜかは分からん)
    if(num == 2 || num == 3 || num == 5 || num == 13 || num ==
19 || num == 73 || num == 193 || num == 407521
        || num == 299210837)
        return true;
    // 1 か 2 以外の偶数は素数でない
    if(num == 1 || !(num & 1)) return false;

    // num-1 = 2^s d (d は奇数) を満たす s, d を求める
    ll s = 0, d = num - 1;
    while(!(d & 1)) { s = s + 1, d >>= 1; }

    // 各 a について, 条件をチェックする
    for(auto a : A) {
        ll powa = mod_pow(a, d, num);
        if(powa == 1 || powa == num - 1) continue;

        bool may_prime = false;
        for(int i = 0; i < s - 1; i++) {
            powa = __uint128_t(powa) * powa % num;
            if(powa == 1) return false;
            if(powa == num - 1) {
                may_prime = true;
                break;
            }
        }
        if(!may_prime) return false;
    }

    return true;
}
```

xor_set.hpp

md5: 70bacd

```
class xor_set {
private:
    vector<ll> w;

public:
    xor_set() {}
    void insert(ll x) {
        for(ll v : w)
            if(v & -v & x) x ^= v;
        if(x == 0) return;
        for(ll& v : w)
            if(x & -x & v) v ^= x;
        w.push_back(x);
    }
    // 独立か判定
```

```
    ll count(ll x) {
        for(ll v : w)
            if(v & -v & x) x ^= v;
        if(x == 0) return 1;
        else return 0;
    }
    vector<ll> get() { return w; }
};
```

modint

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

2SAT.hpp

md5: fde404

```
struct TwoSat {
    int N;
    vector<vector<int>> gr;
    vector<int> values; // 0 = false, 1 = true

    TwoSat(int n = 0): N(n), gr(2 * n) {}
```

```
int addVar() { // (optional)
    gr.emplace_back();
    gr.emplace_back();
    return N++;
}
// (f v j)
void either(int f, int j) {
    f = max(2 * f, -1 - 2 * f);
    j = max(2 * j, -1 - 2 * j);
    gr[f].push_back(j ^ 1);
    gr[j].push_back(f ^ 1);
}
// x は真
void setValue(int x) { either(x, x); }
// li の中から高々1つ真
void atMostOne(const vector<int>& li) { // (optional)
    if(sz(li) <= 1) return;
    int cur = ~li[0];
    rep(i, 2, sz(li)) {
        int next = addVar();
        either(cur, ~li[i]);
        either(cur, next);
        either(~li[i], next);
        cur = ~next;
    }
    either(cur, ~li[1]);
}

vector<int> val, comp, z;
int time = 0;
int dfs(int i) {
    int low = val[i] = ++time, x;
    z.push_back(i);
    for(int e : gr[i])
        if(!comp[e]) low = min(low, val[e] ?: dfs(e));
    if(low == val[i]) do {
        x = z.back();
        z.pop_back();
        comp[x] = low;
        if(values[x >> 1] == -1) values[x >> 1] = x & 1;
    } while(x != i);
    return val[i] = low;
}

bool solve() {
    values.assign(N, -1);
    val.assign(2 * N, 0);
    comp = val;
    rep(i, 0, 2 * N) if(!comp[i]) dfs(i);
    rep(i, 0, N) if(comp[2 * i] == comp[2 * i + 1]) return 0;
    return 1;
}
};
```

scc.hpp

md5: d48735

```
struct StronglyConnectedComponents {
    ll n;
    vector<vector<ll>> G, rG;
    vector<ll> order, component;
    vector<bool> used;
    void dfs(ll v) {
        used[v] = 1;
        for(auto nv : G[v]) {
            if(!used[nv]) dfs(nv);
        }
        order.push_back(v);
    }
    void rdfs(ll v, ll k) {
        component[v] = k;
        for(auto nv : rG[v]) {
            if(component[nv] < 0) rdfs(nv, k);
        }
    }

    StronglyConnectedComponents(vector<vector<ll>>& _G) {
        n = _G.size();
        G = _G;
        rG.resize(n);
        component.assign(n, -1);
        used.resize(n);
```

```

    for(ll v = 0; v < n; v++) {
        for(auto nv : G[v]) rG[nv].push_back(v);
    }
    for(ll v = 0; v < n; v++)
        if(!used[v]) dfs(v);
    ll k = 0;
    reverse(order.begin(), order.end());
    for(auto v : order)
        if(component[v] == -1) rdfs(v, k), k++;
}

/// 頂点(u, v)が同じ強連結成分に含まれるか
bool is_same(ll u, ll v) { return component[u] == component[v]; }

/// 強連結成分を1つのノードに潰したグラフを再構築する
vector<vector<ll>> rebuild() {
    ll N = *max_element(component.begin(), component.end()) + 1;

    vector<vector<ll>> rebuilddG(N);
    set<pair<ll, ll>> connected;
    for(ll v = 0; v < n; v++) {
        for(auto nv : G[v]) {
            if(component[v] != component[nv] and
!connected.count(pair(v, nv))) {
                connected.insert(pair(v, nv));
            }
        }
    }
    rebuilddG[component[v]].push_back(component[nv]);
}

return rebuilddG;
}

vector<vector<ll>> scc() {
    unordered_map<ll, vector<ll>> mp;
    rep(i, 0, n) mp[component[i]].push_back(i);
    vector<vector<ll>> res;
    for(auto [key, val] : mp) res.push_back(val);
    return res;
}
};

```

graph/tree

lca.hpp

md5: 281c40

```

template<class graph> struct lca {
    const graph& G;
    vector<vector<int>> parent;
    vector<int> dep;
    int log = 1;
    lca(const graph& G, int root = 0) : G(G) { init(root); }
    void init(int root = 0) {
        const int n = G.size();
        dep.assign(n, -1);
        parent.assign(1, vector<int>(n, -1));
        queue<int> que({root});
        dep[root] = 0;
        int max_dep = 0;
        while(!que.empty()) {
            const int pos = que.front();
            que.pop();
            max_dep = max(max_dep, dep[pos]);
            for(const auto& x : G[pos]) {
                if(dep[x] == -1) {
                    dep[x] = dep[pos] + 1;
                    parent[0][x] = pos;
                    que.push(x);
                }
            }
        }
        while((1 << log) <= max_dep) log++;
        parent.resize(log, vector<int>(n, -1));
        for(int k = 0; k < log - 1; k++) {
            for(int v = 0; v < n; v++) {
                if(parent[k][v] < 0) parent[k + 1][v] = -1;
                else parent[k + 1][v] = parent[k][parent[k][v]];
            }
        }
    }
};

```

```

int query(int u, int v) const {
    if(dep[u] < dep[v]) swap(u, v);
    const int sub = dep[u] - dep[v];
    for(int k = 0; k < log; k++) {
        if(!(sub >> k)) break;
        if(sub >> k & 1) u = parent[k][u];
    }
    if(u == v) return u;
    for(int k = __lg(dep[u]); k >= 0; k--) {
        if(parent[k][u] != parent[k][v]) {
            u = parent[k][u];
            v = parent[k][v];
        }
    }
    return parent[0][u];
}

int dist(const int u, const int v) const { return dep[u] + dep[v] - dep[query(u, v)] * 2; }

int jump(int u, int v, int d) const {
    const int lc = query(u, v);
    const int l = dep[u] - dep[lc];
    const int r = dep[v] - dep[lc];
    if(d < 0 || d > l + r) return -1;
    if(l < d) {
        d = l + r - d;
        swap(u, v);
    }
    for(int k = 0; k < log; k++) {
        if(!(d >> k)) break;
        if(d >> k & 1) u = parent[k][u];
    }
    return u;
}
};

```

flow

dinic.hpp

md5: fd2103

```

struct Dinic {
    int V; // 頂点数
    vector<vector<vector<ll>>> graph; // グラフ
    vector<int> dis; // 始点からの距離
    vector<int> next; // 次に処理する頂点のメモ
    Dinic(int v) : V(v) { graph.resize(V); }
    void add_edge(int from, int to, ll capacity) {
        graph[from].push_back({to, capacity,
(int)graph[to].size()});
        graph[to].push_back({from, 0, (int)graph[from].size() - 1});
    }

    void bfs(int s) {
        dis.assign(V, -1);
        dis[s] = 0;
        deque<int> pos = {s};
        while(pos.size()) {
            int now = pos[0];
            pos.pop_front();
            for(auto& to : graph[now]) {
                if(dis[to[0]] < 0 and to[1] > 0) {
                    dis[to[0]] = dis[now] + 1;
                    pos.emplace_back(to[0]);
                }
            }
        }
    }

    ll dfs(int v, int t, ll f) {
        if(v == t) return f;
        for(int& i = next[v]; i < graph[v].size(); i++) {
            int to = graph[v][i][0];
            ll& cap = graph[v][i][1];
            int rev = graph[v][i][2];
            if(cap > 0 and dis[v] < dis[to]) {
                ll d = dfs(to, t, min(f, cap));
                if(d > 0) {
                    cap -= d;
                    graph[to][rev][1] += d;
                    return d;
                }
            }
        }
        return 0;
    }
};

```



```
        return 0;
    }
    ll max_flow(int s, int t) {
        ll flow = 0;
        while(1) {
            bfs(s);
            if(dis[t] < 0) return flow;
            next.assign(V, 0);
            ll f;
            while((f = dfs(s, t, INF)) > 0) flow += f;
        }
    }
};
```

mincostflow.hpp

md5: e93128

```
struct MinCostFlow {
    int V;
    vector<vector<vector<ll>>> g; // g[from] = {{to, 容量, コスト, 逆辺のindex} ... }
    vector<ll> h, dis; // ポテンシャル, 最短距離
    vector<int> prevv, preve; // 直前の頂点, 辺

    MinCostFlow(int v) : V(v), g(v), dis(v), prevv(v), preve(v) {}

    void add_edge(int u, int v, ll c, ll d) {
        g[u].push_back({v, c, d, (int)g[v].size()});
        g[v].push_back({u, 0, -d, (int)g[u].size() - 1});
    }

    ll min_cost_flow(int s, int t, ll f) {
        ll res = 0;
        h.assign(V, 0);
        using Q = pair<ll, int>;
        while(f > 0) {
            priority_queue<Q, vector<Q>, greater<Q>> que;
            dis.assign(V, LLONG_MAX);
            dis[s] = 0;
            que.push({0, s});
            while(que.size()) {
                Q q = que.top();
                int v = q.second;
                que.pop();
                if(dis[v] < q.first) continue;
                for(int i = 0; i < g[v].size(); i++) {
                    auto edge = g[v][i];
                    int to = edge[0];
                    ll cap = edge[1], cost = edge[2];
                    if(cap > 0 and dis[to] > dis[v] + cost + h[v] -
h[to]) {
                        dis[to] = dis[v] + cost + h[v] - h[to];
                        prevv[to] = v;
                        preve[to] = i;
                        que.push({dis[to], to});
                    }
                }
            }
            if(dis[t] == INF) return -1;
            for(int i = 0; i < V; i++) h[i] += dis[i];
            ll d = f;
            for(int i = t; i != s; i = prevv[i]) d = min(d,
g[prevv[i]][preve[i]][1]);
            f -= d;
            res += d * h[t];
            for(int i = t; i != s; i = prevv[i]) {
                auto& edge = g[prevv[i]][preve[i]];
                edge[1] -= d;
                g[i][edge[3]][1] += d;
            }
        }
        return res;
    }
};
```

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)

変形前の制約	変形後の制約
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.hpp

md5: 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.hpp

md5: 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$a$ -> 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.hpp

md5: 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.hppmd5: 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

mo.hppmd5: 934d7d

struct Mo {
 int n;
 vector<pair<int, int> > lr;

 explicit Mo(int n) : n(n) {}

 void add(int l, int r) { /* [l, r) */
 lr.emplace_back(l, r);
 }

 template<typename AL, typename AR, typename EL, typename ER, typename O>
 void build(const AL& add_left, const AR& add_right, const EL& erase_left, const ER& erase_right, const O& out) {
 int q = (int)lr.size();
 int bs = n / min<int>(n, sqrt((double)q));
 vector<int> ord(q);
 iota(begin(ord), end(ord), 0);
 sort(begin(ord), end(ord), [&](int a, int b) {
 int ablock = lr[a].first / bs, bbblock = lr[b].first /
bs;

 if(ablock != bbblock) return ablock < bbblock;
 return (ablock & 1) ? lr[a].second > lr[b].second :

lr[a].second < lr[b].second;
 });
 int l = 0, r = 0;
 for(auto idx : ord) {
 while(l > lr[idx].first) add_left(--l);
 while(r < lr[idx].second) add_right(r++);
 while(l < lr[idx].first) erase_left(l++);
 while(r > lr[idx].second) erase_right(--r);
 out(idx);
 }
 }

 template<typename A, typename E, typename O> void
 build(const A& add, const E& erase, const O& out) {
 build(add, add, erase, erase, out);
 }
};

int main() {
 int N;
 cin >> N;
 vector<int> A(N);
 for(auto& a : A) cin >> a;
 int Q;
 cin >> Q;
 Mo mo(N);
 for(int i = 0; i < Q; i++) {
 int a, b;
 cin >> a >> b;
 mo.add(a - 1, b);
 }
 vector<int> cnt(1000001), ans(Q);
 int sum = 0;
 auto add = [&](int i) {
 if(cnt[A[i]]++ == 0) ++sum;
 };
 auto erase = [&](int i) {
 if(--cnt[A[i]] == 0) --sum;
 };
 auto out = [&](int q) { ans[q] = sum; };
 mo.build(add, erase, out);
 for(auto& p : ans) cout << p << "\n";
}

geometry

geometry.hppmd5: f4e0fc

/*
前提
- 点(位置ベクトル)を複素数型で扱う
- 実軸(real)をx軸、虚軸(imag)をy軸として見る
- 比較するときは、計算誤差を意識して EPS で判定 (equal関数)
*/

namespace geometry {
using D = long double;
using Point = std::complex<D>;
using Polygon = vector<Point>;
const D EPS = 1e-8;
const D PI = M_PI;

// 入出力ストリーム
istream& operator>>(istream& is, Point& p) {
 D a, b;
 is >> a >> b;
 p = Point(a, b);
 return is;
}

ostream& operator<<(ostream& os, Point& p) { return os << fixed
<< setprecision(20) << p.real() << ' ' << p.imag(); }

// d 倍する
Point operator*(Point p, D d) { return Point(p.real() * d,
p.imag() * d); }

// 偏角 (0 <= θ < 2π)
D argument(Point p) {
 D res = arg(p);

```

    if(res < 0.0) res += 2.0 * PI; // [-π, π] -> [0, 2π)
    return res;
}

// 等しいかどうか (誤差で判定)
inline bool equal(D a, D b) { return fabs(a - b) < EPS; }

// 単位ベクトル
Point unit_vector(Point a) { return a / abs(a); };

// 法線ベクトル (逆向きがよければ (0, -1) をかける)
Point normal_vector(Point a, D dir = 1) { return a * Point(0, dir); }

// 内積:  $a \cdot b = |a||b|\cos\theta$ 
D dot(Point a, Point b) { return (a.real() * b.real() + a.imag() * b.imag()); }

// 外積:  $a \times b = |a||b|\sin\theta$  (外積の大きさではないか?)
D cross(Point a, Point b) { return (a.real() * b.imag() - a.imag() * b.real()); }

// 反時計回りに theta 回転
Point rotate(Point a, D theta) {
    D c = cos(theta), s = sin(theta);
    return Point(c * a.real() - s * a.imag(), s * a.real() + c * a.imag());
}

// 直線
struct Line {
    Point a, b;
    Line() = default;
    Line(Point a_, Point b_) : a(a_), b(b_) { assert(a_ != b_); }
};

// Ax+By=C
Line(D A, D B, D C) {
    if(equal(A, 0)) {
        a = Point(0, C / B), b = Point(1, C / B);
    } else if(equal(B, 0)) {
        b = Point(C / A, 0), a = Point(C / A, 1);
    } else {
        a = Point(0, C / B), b = Point(C / A, 0);
    }
}

// 線分 (Line と同じ)
struct Segment : Line {
    Segment() = default;
    Segment(Point a_, Point b_) : Line(a_, b_){};
};

// 円 (中心と半径)
struct Circle {
    Point p;
    D r;
    Circle(Point p_, D r_) : p(p_), r(r_) {}
};

// 射影: 直線 (線分) に 点p から引いた垂線の足を求める
Point projection(Line l, Point p) {
    D t = dot(p - l.a, l.a - l.b) / norm(l.a - l.b);
    return l.a + (l.a - l.b) * t;
}

Point projection(Segment l, Point p) {
    D t = dot(p - l.a, l.a - l.b) / norm(l.a - l.b);
    return l.a + (l.a - l.b) * t;
}

// 反射: 直線を対象軸として 点p と線対称の位置にある点を求める
Point reflection(Line l, Point p) { return p + (projection(l, p) - p) * 2.0; }

// 3点 a, b, c の位置関係
int ccw(Point a, Point b, Point c) {
    b -= a, c -= a;
    // 点 a, b, c が
    if(cross(b, c) > EPS) return 1; // 反時計回りのとき
    if(cross(b, c) < -EPS) return -1; // 時計回りのとき
}

```

```

// 同一直線上にある場合
if(dot(b, c) < 0) return 2; // c, a, b の順
if(norm(b) < norm(c)) return -2; // a, b, c の順
return 0; // a, c, b の順
}

// 垂直 (内積 == 0)
bool is_vertical(Line a, Line b) { return equal(dot(a.b - a.a, b.b - b.a), 0); }

// 平行 (外積 == 0)
bool is_parallel(Line a, Line b) { return equal(cross(a.b - a.a, b.b - b.a), 0); }

// 線分の交差判定 (線分 s に対して, 線分 t の端点が反対側にあればよい)
bool is_intersect(Segment s, Segment t) {
    return (ccw(s.a, s.b, t.a) * ccw(s.a, s.b, t.b) <= 0) &&
    (ccw(t.a, t.b, s.a) * ccw(t.a, t.b, s.b) <= 0);
}

// 交点 (交差する前提)
Point cross_point(Line s, Line t) {
    D d1 = cross(s.b - s.a, t.b - t.a);
    D d2 = cross(s.b - s.a, s.b - t.a);
    // s, t が一致する場合 (適当な1点を返す)
    if(equal(abs(d1), 0) && equal(abs(d2), 0)) return t.a;

    return t.a + (t.b - t.a) * (d2 / d1);
}

Point cross_point(Segment s, Segment t) {
    assert(is_intersect(s, t)); // 交差する前提
    return cross_point(Line(s), Line(t));
}

// 点の間の距離
D dist(Point a, Point b) { return abs(a - b); }

// 点と直線の距離 (垂線の足との距離)
D dist_line_point(Line l, Point p) { return abs(p - projection(l, p)); }

// 線分と点の距離 (点p から線分のどこかへの最短距離)
D dist_segment_point(Segment l, Point p) {
    if(dot(l.b - l.a, p - l.a) < EPS) return abs(p - l.a);
    if(dot(l.a - l.b, p - l.b) < EPS) return abs(p - l.b);
    return abs(cross(l.b - l.a, p - l.a)) / abs(l.b - l.a);
}

// 線分と線分の距離
D dist_segment_segment(Segment s, Segment t) {
    if(is_intersect(s, t)) return 0.0;
    D res = min({
        dist_segment_point(s, t.a),
        dist_segment_point(s, t.b),
        dist_segment_point(t, s.a),
        dist_segment_point(t, s.b),
    });
    return res;
}

// 2つの円の交点
pair<Point, Point> crosspoint(const Circle& c1, const Circle& c2) {
    D d = abs(c1.p - c2.p);
    D a = acos((c1.r * c1.r + d * d - c2.r * c2.r) / (2 * c1.r * d));
    D t = atan2(c2.p.imag() - c1.p.imag(), c2.p.real() - c1.p.real());
    Point p1 = c1.p + Point(cos(t + a) * c1.r, sin(t + a) * c1.r);
    Point p2 = c1.p + Point(cos(t - a) * c1.r, sin(t - a) * c1.r);
    return {p1, p2};
}

ll cross_cht(Point o, Point a, Point b) {
    return (a.real() - o.real()) * (b.imag() - o.imag()) -
    (a.imag() - o.imag()) * (b.real() - o.real());
}

```

```
// 凸包
Polygon convex_hull(Polygon ps) {
    int n = ps.size(), k = 0;
    if(n <= 2) return ps;
    sort(ps.begin(), ps.end(),
        [](const Point& a, const Point& b) { return real(a) !=
real(b) ? real(a) < real(b) : imag(a) < imag(b); });
    Polygon res;
    for(auto p : ps) {
        while((int)res.size() >= 2 && cross_cht(res[k - 1], res[k
- 2], p) >= 0) {
            res.pop_back();
            k--;
        }
        res.push_back(p);
        k++;
    }
    int t = res.size();
    rrep(i, n - 2, 0) {
        while((int)res.size() > t && cross_cht(res[k - 1], res[k
- 2], ps[i]) >= 0) {
            res.pop_back();
            k--;
        }
        res.push_back(ps[i]);
        k++;
    }
    return res;
}
}; // namespace geometry
using namespace 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