MeloN's XCPC Library - ICPC2024 - Kunming

Library

目录

Z_H

MeloN_H.hpp

```cpp
using   std::array, std::bitset, std::deque, std::greater, std::less, std::map,
        std::multiset, std::pair, std::priority_queue, std::set, std::stack,
        std::string, std::vector;

using NAME = void;       using uint = unsigned;   using ll = long long;      using ull = unsigned long long;
using ld = long double;  using i128 = __int128_t; using u128 = __uint128_t;  using f128 = __float128;

#define meion     auto
#define iroha     return
```

MeloN_IO.hpp

```cpp
namespace MeIoN_IO {
    std::istream& operator>>(std::istream& is, i128& n) {
        string s;
        is >> s;
        int f = s[0] == '-';
        n = 0;
        for (int i = f; i < s.length(); ++i) {
            n = n * 10 + s[i] - '0';
        }
        if (f) n = -n;
        return is;
    }
    std::ostream& operator<<(std::ostream& os, i128 n) {
        string s;
        bool f = n < 0;
        if (f) n = -n;
        while (n) s += '0' + n % 10, n /= 10;
        if (s.empty()) s += '0';
        if (f) s += '-';
        std::reverse(s.begin(), s.end());
        return os << s;
    }
    std::istream& operator>>(std::istream& is, f128& n) {
        string s;
        is >> s;
        n = std::stold(s);
        return is;
    }
```

```
} using namespace MeIoN_IO;
```

MeIoN_PRET.hpp

```
namespace MeIoN_Pre_Things {
    int T = 1;
    std::mt19937 rng(std::chrono::steady_clock::now().time_since_epoch().count());
    std::mt19937_64 rng_64(std::chrono::steady_clock::now().time_since_epoch().count());
    constexpr int mod99 = 998244353, mod17 = 1000000007;
    constexpr ld eps = 1E-8L, pi = 3.14159265358979323846264338327950L;
    template <class T>
    constexpr T inf = 0;
    template <>
    constexpr int inf<int> = 2147483647;
    template <>
    constexpr uint inf<uint> = 4294967294U;
    template <>
    constexpr ll inf<ll> = 9223372036854775807LL;
    template <>
    constexpr ull inf<ull> = 18446744073709551614ULL;
    template <>
    constexpr i128 inf<i128> = i128(inf<ll>) * 2'000'000'000'000'000'000;
    template <>
    constexpr double inf<double> = inf<ll>;
    template <>
    constexpr long double inf<long double> = inf<ll>;
    template <typename T>
    inline T lowbit(T x) { return x & -x; }
    template <typename T>
    inline int popcount(T n) { return std::__popcount(n); }
    template <typename T>
    inline int clz(T n) { return std::__countl_zero(n); }
    template <class T, class S>
    inline bool chmax(T &a, const S &b) {
        return (a < b ? a = b, 1 : 0);
    }
    template <class T, class S>
    inline bool chmin(T &a, const S &b) {
        return (a > b ? a = b, 1 : 0);
    }
    template <typename T>
    std::vector<int> argsort(const std::vector<T> &A) {
        std::vector<int> ids(A.size());
        std::iota(ids.begin(), ids.end(), 0);
        std::sort(ids.begin(), ids.end(), [&](int i, int j) { return A[i] < A[j] or (A[i] == A[j] and i < j); }
        return ids;
    }
    template <typename T>
    vector<T> rearrange(const vector<T> &A, const vector<int> &I) {
        vector<T> B(I.size());
        for (int i = 0, ed = I.size(); i < ed; ++i)
            B[i] = A[I[i]];
        return B;
    }
    // (0, 1, 2, 3, 4) -> (-1, 0, 1, 1, 2)
    int topbit(int x) { return (x == 0 ? -1 : 31 - __builtin_clz(x)); }
    int topbit(uint x) { return (x == 0 ? -1 : 31 - __builtin_clz(x)); }
    int topbit(ll x) { return (x == 0 ? -1 : 63 - __builtin_clzll(x)); }
    int topbit(ull x) { return (x == 0 ? -1 : 63 - __builtin_clzll(x)); }
    template <typename T, typename U>
    inline T ceil(T x, U y) { return(x > 0 ? (x + y - 1) / y : x / y); }
    template <typename T, typename U>
```

```
    inline T floor(T x, U y) { return (x > 0 ? x / y : (x - y + 1) / y); }
    template <typename F>
    ll binary_search(F check, ll ok, ll ng, bool check_ok = true) {
        if (check_ok) assert(check(ok));
        while (std::abs(ok - ng) > 1) {
            auto x = (ng + ok) / 2;
            (check(x) ? ok : ng) = x;
        }
        return ok;
    }
    template <class T>
    struct MeIoN_Que {
        vector<T> q;
        int pos = 0;
        void reserve(int n) { q.reserve(n); }
        int size() const { return int(q.size()) - pos; }
        bool empty() const { return pos == int(q.size()); }
        T& front() { return q[pos]; }
        T& back() { return q.back(); }
        template <typename... Args>
        void emplace_back(Args&&... args) {
            q.emplace_back(std::forward<Args>(args)...);
        }
        void push_back(const T& v) { q.push_back(v); }
        void pop() { ++pos; }
        void pop_back() { q.pop_back(); }
        void clear() {
            q.clear();
            pos = 0;
        }
    };
} using namespace MeIoN_Pre_Things;
```

MeIoN_debug.hpp

```
// copy from https://github.com/Heltion/debug.h
template <class T, size_t size = std::tuple_size<T>::value>
std::string to_debug(T, std::string s = "")
    requires(not std::ranges::range<T>);
std::string to_debug(auto x)
    requires requires(std::ostream& os) { os << x; }
{
    return static_cast<std::ostringstream>(std::ostringstream() << x).str();
}
std::string to_debug(std::ranges::range auto x, std::string s = "")
    requires(not std::is_same_v<decltype(x), std::string>)
{
    for (auto xi : x) {
        s += ", " + to_debug(xi);
    }
    return "[" + s.substr(s.empty() ? 0 : 2) + "]";
}
template <class T, size_t size>
std::string to_debug(T x, std::string s)
    requires(not std::ranges::range<T>)
{
    [&]<size_t... I>(std::index_sequence<I...>) {
        ((s += ", " + to_debug(std::get<I>(x))), ...);
    }(std::make_index_sequence<size>());
    return "(" + s.substr(s.empty() ? 0 : 2) + ")";
}
```

```
#ifdef MeIoN
#define debug(...)  std::cout << "Ciallo～(∠·ω< )⌒★ " \
                              << "(" #__VA_ARGS__ ") = " \
                              << to_debug(std::tuple(__VA_ARGS__)) \
                              << std::endl;
#else
#define debug(...) void(0721)
#endif
```

fast_io.hpp

```
namespace fast_io {
    static constexpr uint32_t SZ = 1 << 17;
    char ibuf[SZ];
    char obuf[SZ];
    char out[100];
    // pointer of ibuf, obuf
    uint32_t pil = 0, pir = 0, por = 0;

    struct Pre {
        char num[10000][4];
        constexpr Pre() : num() {
            for (int i = 0; i < 10000; i++) {
                int n = i;
                for (int j = 3; j >= 0; j--) {
                    num[i][j] = n % 10 | '0';
                    n /= 10;
                }
            }
        }
    } constexpr pre;

    inline void load() {
        memcpy(ibuf, ibuf + pil, pir - pil);
        pir = pir - pil + fread(ibuf + pir - pil, 1, SZ - pir + pil, stdin);
        pil = 0;
        if (pir < SZ) ibuf[pir++] = '\n';
    }

    inline void flush() {
        fwrite(obuf, 1, por, stdout);
        por = 0;
    }
    void rd(char &c) {
        do {
            if (pil + 1 > pir) load();
            c = ibuf[pil++];
        } while (isspace(c));
    }

    void rd(string &x) {
        x.clear();
        char c;
        do {
            if (pil + 1 > pir) load();
            c = ibuf[pil++];
        } while (isspace(c));
        do {
            x += c;
            if (pil == pir) load();
            c = ibuf[pil++];
        } while (!isspace(c));
```

```
    }

    template <typename T>
    void rd_real(T &x) {
        string s;
        rd(s);
        x = stod(s);
    }

    template <typename T>
    void rd_integer(T &x) {
        if (pil + 100 > pir) load();
        char c;
        do c = ibuf[pil++];
        while (c < '-');
        bool minus = 0;
        if constexpr (std::is_signed<T>::value || std::is_same_v<T, i128>) {
            if (c == '-') {
                minus = 1, c = ibuf[pil++];
            }
        }
        x = 0;
        while ('0' <= c) {
            x = x * 10 + (c & 15), c = ibuf[pil++];
        }
        if constexpr (std::is_signed<T>::value || std::is_same_v<T, i128>) {
            if (minus) x = -x;
        }
    }

    void rd(int &x) { rd_integer(x); }
    void rd(ll &x) { rd_integer(x); }
    void rd(i128 &x) { rd_integer(x); }
    void rd(uint &x) { rd_integer(x); }
    void rd(ull &x) { rd_integer(x); }
    void rd(u128 &x) { rd_integer(x); }
    void rd(double &x) { rd_real(x); }
    void rd(long double &x) { rd_real(x); }
    void rd(f128 &x) { rd_real(x); }

    void read() {}
    template <class H, class... T>
    void read(H &h, T &...t) {
        rd(h), read(t...);
    }

    void wt(const char c) {
        if (por == SZ) flush();
        obuf[por++] = c;
    }
    void wt(const string s) {
        for (char c : s) wt(c);
    }
    void wt(const char *s) {
        size_t len = strlen(s);
        for (size_t i = 0; i < len; i++) wt(s[i]);
    }

    template <typename T>
    void wt_integer(T x) {
        if (por > SZ - 100) flush();
        if (x < 0) {
```

```
            obuf[por++] = '-', x = -x;
        }
        int outi;
        for (outi = 96; x >= 10000; outi -= 4) {
            memcpy(out + outi, pre.num[x % 10000], 4);
            x /= 10000;
        }
        if (x >= 1000) {
            memcpy(obuf + por, pre.num[x], 4);
            por += 4;
        } else if (x >= 100) {
            memcpy(obuf + por, pre.num[x] + 1, 3);
            por += 3;
        } else if (x >= 10) {
            int q = (x * 103) >> 10;
            obuf[por] = q | '0';
            obuf[por + 1] = (x - q * 10) | '0';
            por += 2;
        } else
            obuf[por++] = x | '0';
        memcpy(obuf + por, out + outi + 4, 96 - outi);
        por += 96 - outi;
    }

    template <typename T>
    void wt_real(T x) {
        std::ostringstream oss;
        oss << std::fixed << std::setprecision(15) << double(x);
        std::string s = oss.str();
        wt(s);
    }

    void wt(int x) { wt_integer(x); }
    void wt(ll x) { wt_integer(x); }
    void wt(i128 x) { wt_integer(x); }
    void wt(uint x) { wt_integer(x); }
    void wt(ull x) { wt_integer(x); }
    void wt(u128 x) { wt_integer(x); }
    void wt(double x) { wt_real(x); }
    void wt(long double x) { wt_real(x); }
    void wt(f128 x) { wt_real(x); }
    void wt(std::ios_base &(*__pf)(std::ios_base &)) {}
    void wt(const std::_Setprecision &x) {}

    // gcc expansion. called automaticall after main.
    void __attribute__((destructor)) _d() { flush(); }

    struct io_auxiliary {
        void sync_with_stdio(bool ok) { }
    } *auxiliary_io;

    struct meion_fast_io {
        template<typename T>
        friend meion_fast_io& operator>>(meion_fast_io &in, T &c) {
            rd(c);
            return in;
        }
        template<typename T>
        friend meion_fast_io& operator<<(meion_fast_io &out, const T &c) {
            wt(c);
            return out;
        }
```

```
        io_auxiliary *tie(std::nullptr_t x) {
            return auxiliary_io;
        }
    } fin, fout;
}
#define fast
#define cin fin
#define cout fout
namespace std {
    using fast_io::cin, fast_io::cout;
}
```

ds

LinearBasis.hpp

```cpp
struct LinearBasis {
    static const int B = 30;
    LinearBasis() { memset(basis, -1, sizeof(basis)); }
    void add(int v) {
        v = ask(v);
        if (v) {
            int pivot = 30 - clz(v);
            for (int i = 0; i < B; ++i) {
                if (~basis[i] && (basis[i] >> pivot & 1)) {
                    basis[i] ^= v;
                }
            }
            basis[pivot] = v;
        }
    }
    int ask(int v) const {
        for (int i = B; i--;) {
            if ((v >> i & 1) and ~basis[i]) {
                v ^= basis[i];
            }
        }
        return v;
    }
    int basis[B];
};
struct LinearBasis_64 {
    static const int B = 63;
    LinearBasis_64() { memset(basis, -1, sizeof(basis)); }
    void add(ll v) {
        v = ask(v);
        if (v) {
            int pivot = 62 - clz(v);
            for (int i = 0; i < B; ++i) {
                if (~basis[i] && (basis[i] >> pivot & 1)) {
                    basis[i] ^= v;
                }
            }
            basis[pivot] = v;
        }
    }
    ll ask(ll v) const {
        for (int i = B; i--;) {
            if ((v >> i & 1) and ~basis[i]) {
                v ^= basis[i];
            }
        }
        return v;
    }
    ll quis(ll v = 0ll) {
        for (int i = B; i--; ) {
            if (not (v >> i & 1) and ~basis[i]) {
                v ^= basis[i];
            }
        }
        return v;
    }
    ll basis[B];
```

```
};
```

Wavelet_Matrix.hpp

```
struct Bit_Vector {
    vector<pair<unsigned, unsigned>> dat;
    Bit_Vector(int n) { dat.assign((n + 63) >> 5, {0, 0}); }
    void set(int i) { dat[i >> 5].first |= unsigned(1) << (i & 31); }
    void build() {
        for (int i = 0, ed = int(dat.size()) - 1; i < ed; ++i)
            dat[i + 1].second = dat[i].second + std::popcount(dat[i].first);
    }
    // [0, k) 内の 1 の個数
    int rank(int k, bool f = 1) {
        auto[a, b] = dat[k >> 5];
        int ret = b + std::popcount(a & ((unsigned(1) << (k & 31)) - 1));
        return (f ? ret : k - ret);
    }
};
// 座圧するかどうかを COMPRESS で指定する
// xor 的な使い方をする場合には、コンストラクタで log を渡すこと
template <typename T = int, bool COMPRESS = false>
struct Wavelet_Matrix {
    int N, lg;
    vector<int> mid;
    vector<Bit_Vector> bv;
    vector<T> key;
    bool set_log;
    Wavelet_Matrix(vector<T> A, int log = -1)
        : N(A.size()), lg(log), set_log(log != -1) {
        if (COMPRESS) {
            assert(!set_log);
            key.reserve(N);
            vector<int> I = argsort(A);
            for (auto&& i : I) {
                if (key.empty() || key.back() != A[i]) key.emplace_back(A[i]);
                A[i] = (int)key.size() - 1;
            }
            key.shrink_to_fit();
        }
        if (lg == -1) lg = std::__lg(std::max<ll>(qmax(A), 1)) + 1;
        mid.resize(lg);
        bv.assign(lg, Bit_Vector(N));
        vector<T> A0(N), A1(N);
        for (ll d = (lg)-1; d >= ll(0); --d) {
            int p0 = 0, p1 = 0;
            for (ll i = 0; i < ll(N); ++i) {
                bool f = (A[i] >> d & 1);
                if (!f) A0[p0++] = A[i];
                if (f) bv[d].set(i), A1[p1++] = A[i];
            }
            mid[d] = p0;
            bv[d].build();
            std::swap(A, A0);
            for (ll i = 0; i < ll(p1); ++i) A[p0 + i] = A1[i];
        }
    }
    // xor した結果で [a, b) に収まるものを数える
    int count(int L, int R, T a, T b, T xor_val = 0) {
        return prefix_count(L, R, b, xor_val) - prefix_count(L, R, a, xor_val);
    }
    // xor した結果で [0, x) に収まるものを数える
```

```
    int prefix_count(int L, int R, T x, T xor_val = 0) {
        if (xor_val != 0) assert(set_log);
        x = (COMPRESS
                ? std::distance((key).begin(),
                                std::lower_bound(key.begin(), key.end(), (x)))
                : x);
        if (x >= (1 << lg)) return R - L;
        int ret = 0;
        for (int d = lg - 1; d >= 0; --d) {
            bool add = (x >> d) & 1;
            bool f = ((x ^ xor_val) >> d) & 1;
            if (add) ret += bv[d].rank(R, !f) - bv[d].rank(L, !f);
            L = bv[d].rank(L, f) + (f ? mid[d] : 0);
            R = bv[d].rank(R, f) + (f ? mid[d] : 0);
        }
        return ret;
    }
    T kth(int L, int R, int k, T xor_val = 0) {  // k : 0 index
        if (xor_val != 0) assert(set_log);
        assert(0 <= k && k < R - L);
        T ret = 0;
        for (int d = lg - 1; d >= 0; --d) {
            bool f = (xor_val >> d) & 1;
            int l0 = bv[d].rank(L, 0), r0 = bv[d].rank(R, 0);
            int kf = (f ? (R - L) - (r0 - l0) : (r0 - l0));
            if (k < kf) {
                if (!f) L = l0, R = r0;
                if (f) L += mid[d] - l0, R += mid[d] - r0;
            } else {
                k -= kf, ret |= T(1) << d;
                if (!f) L += mid[d] - l0, R += mid[d] - r0;
                if (f) L = l0, R = r0;
            }
        }
        return(COMPRESS ? key[ret] : ret);
    }
};
```

bit_vec.hpp

```
template <const int N>
struct bitarray {
    static constexpr int sz = ((N + 127) >> 6);
    array<ull, sz> v;
    void set(int i) {
        v[i >> 6] |= 1ull << (i & 63);
    }
    void reset(int i) {
        v[i >> 6] &= ~(1ull << (i & 63));
    }
    void reset() {
        std::ranges::fill(v, 0ull);
    }
    bool operator[](int i) {
        return v[i >> 6] >> (i & 63) & 1;
    }
    bitarray operator &=(const bitarray &b) {
        for (int i = 0, ed = sz; i < ed; ++i) {
            v[i] &= b.v[i];
        }
        return *this;
    }
```

```
    bitarray operator |=(const bitarray &b) {
        for (int i = 0, ed = sz; i < ed; ++i) {
            v[i] |= b.v[i];
        }
        return *this;
    }
    bitarray operator ^=(const bitarray &b) {
        for (int i = 0, ed = sz; i < ed; ++i) {
            v[i] ^= b.v[i];
        }
        return *this;
    }
    bitarray operator &(const bitarray &b) {
        return bitarray(*this) &= b;
    }
    bitarray operator |(const bitarray &b) {
        return bitarray(*this) |= b;
    }
    bitarray operator ^(const bitarray &b) {
        return bitarray(*this) ^= b;
    }
    bitarray operator ~() const {
        bitarray ret(*this);
        for (int i = 0, ed = sz; i < ed; ++i) {
            ret.v[i] = ~ret.v[i];
        }
        return ret;
    }
    bitarray operator <<=(const int t) {
        bitarray ret;
        ret.v.fill(0ull);
        ull last = 0;
        int high = t >> 6, low = t & 63;
        for(int i = 0; i + high < sz; ++i) {
            ret.v[i + high] = last | (v[i] << low);
            if (low) last = v[i] >> (64 - low);
        }
        return (*this) = ret;
    }
    bitarray operator >>=(const int t) {
        bitarray ret;
        ret.v.fill(0ull);
        ull last = 0;
        int high = t >> 6, low = t & 63;
        for(int i = int(v.size() - 1); i > high - 1; --i) {
            ret.v[i - high] = last | (v[i] >> low);
            if (low) last = v[i] << (64 - low);
        }
        return (*this) = ret;
    }
    bitarray operator <<(const int t) {
        return bitarray(*this) <<= t;
    }
    bitarray operator >>(const int t) {
        return bitarray(*this) >>= t;
    }
    std::string to_string() {
        std::string ans;
        for (int i = 0; i < N; ++i) {
            ans += '0' + (*this)[i];
        }
        return ans;
```

```
    }
};
struct bitvector {
    int n;
    vector<ull> v;
    bitvector(int n) : n(n), v(n + 127 >> 6, 0ull) {}
    void set(int i) {
        v[i >> 6] |= 1ull << (i & 63);
    }
    void reset(int i) {
        v[i >> 6] &= ~(1ull << (i & 63));
    }
    void reset() {
        std::ranges::fill(v, 0ull);
    }
    bool operator[](int i) {
        return v[i >> 6] >> (i & 63) & 1;
    }
    bitvector operator &=(const bitvector &b) {
        for (int i = 0, ed = int(v.size()); i < ed; ++i) {
            v[i] &= b.v[i];
        }
        return *this;
    }
    bitvector operator |=(const bitvector &b) {
        for (int i = 0, ed = int(v.size()); i < ed; ++i) {
            v[i] |= b.v[i];
        }
        return *this;
    }
    bitvector operator ^=(const bitvector &b) {
        for (int i = 0, ed = int(v.size()); i < ed; ++i) {
            v[i] ^= b.v[i];
        }
        return *this;
    }
    bitvector operator &(const bitvector &b) {
        return bitvector(*this) &= b;
    }
    bitvector operator |(const bitvector &b) {
        return bitvector(*this) |= b;
    }
    bitvector operator ^(const bitvector &b) {
        return bitvector(*this) ^= b;
    }
    bitvector operator ~() const {
        bitvector ret(*this);
        for (int i = 0, ed = int(v.size()); i < ed; ++i) {
            ret.v[i] = ~ret.v[i];
        }
        return ret;
    }
    bitvector operator <<=(const int t) {
        bitvector ret(n);
        ull last = 0;
        int high = t >> 6, low = t & 63;
        for(int i = 0; i + high < int(v.size()); ++i) {
            ret.v[i + high] = last | (v[i] << low);
            if (low) last = v[i] >> (64 - low);
        }
        return (*this) = ret;
    }
```

```
    bitvector operator >>=(const int t) {
        bitvector ret(n);
        ull last = 0;
        int high = t >> 6, low = t & 63;
        for(int i = int(v.size() - 1); i > high - 1; --i) {
            ret.v[i - high] = last | (v[i] >> low);
            if (low) last = v[i] << (64 - low);
        }
        return (*this) = ret;
    }
    bitvector operator <<(const int t) {
        return bitvector(*this) <<= t;
    }
    bitvector operator >>(const int t) {
        return bitvector(*this) >>= t;
    }
    std::string to_string() {
        std::string ans;
        for (int i = 0; i < n; ++i) {
            ans += '0' + (*this)[i];
        }
        return ans;
    }
};
```

chothlly.hpp

```
template <typename DAT>
struct coler_seg {
    int l, r;
    mutable DAT val;
    coler_seg(int a = -1, int b = -1, DAT c = 0) : l(a), r(b), val(c) {}
    bool operator<(const coler_seg&a) const { return l < a.l; }
};
template <typename DAT = int>
struct Chtholly : std::set<coler_seg<DAT>> {
    using iterator = typename std::set<coler_seg<DAT>>::iterator;
    void add(int l, int r, DAT val) {
        iterator itr = split(r + 1), itl = split(l);
        for (iterator it = itl; it != itr; ++it) {
            it->val += val;
        }
    }
    void assign(int l, int r, DAT val){
        iterator itr = split(r + 1), itl = split(l);
        erase(itl, itr);
        emplace(l, r, val);
    }
    ll kth(int l, int r, int rk) {
        iterator itr = split(r + 1), itl = split(l);
        vector<pair<ll, int>> v;
        for (auto it = itl; it != itr; ++it) {
            v.emplace_back(it->val, it->r - it->l + 1);
        }
        sort(v);
        for (const auto &[val, sz] : v) {
            if (rk <= sz) return val;
            rk -= sz;
        }
        return inf<ll>;
    }
    ll quis(int l, int r, int T, int mod) {
```

```
        iterator itr = split(r + 1), itl = split(l);
        ll res = 0;
        for (iterator it = itl; it != itr; ++it) {
            res = (res + (it->r - it->l + 1ll) * ksm((it->val) % mod, T, mod)) % mod;
        }
        return res;
    }

   private:
    ll ksm(int a, int b, int mod) {
        ll res = 1;
        while (b) {
            if (b & 1) res = (res * a) % mod;
            a = 1ll * a * a % mod;
            b >>= 1;
        }
        return res % mod;
    }
    iterator split(int pos) {
        iterator it = lower_bound(coler_seg<DAT>(pos));
        if (it != this->end() and it->l == pos) return it;
        coler_seg<DAT> tmp = *--it;
        erase(it);
        emplace(tmp.l, pos - 1, tmp.val);
        return emplace(pos, tmp.r, tmp.val).first;
    }
};
```

dsu.hpp

```
struct dsu{     //MeIoNのdsu
public:
    dsu(int _n) : n(_n), comp(_n), fa(_n), sz(_n, 1) {
        std::iota(fa.begin(), fa.end(), 0);
    }
    int operator[](int x) { return ff(x); }
    int size(int x) { return sz[ff(x)]; }
    bool merge(int x, int y) {
        x = ff(x), y = ff(y);
        if (x == y) return false;
        if (sz[x] < sz[y]) std::swap(x, y);
        --comp;
        sz[x] += sz[y], sz[y] = 0; fa[y] = x;
        return true;
    }
    void rebuild() {
        std::iota(fa.begin(), fa.end(), 0);
        fill(sz, 1);
    }
private:
    int n, comp;
    std::vector<int> fa, sz;
    int ff(int x) {
        while (x != fa[x]) x = fa[x] = fa[fa[x]];
        return x;
    }
};
```

fenw.hpp

```cpp
template <class T = ll>
struct Fenw {
    int n;
    T total;
    vector<T> dat;
    Fenw() {}
    Fenw(int n) { build(n); }
    template <typename F>
    Fenw(int n, F f) {
        build(n, f);
    }
    Fenw(const vector<T> &v) { build(v); }

    void build(int m) {
        n = m;
        dat.assign(m, T(0));
        total = T(0);
    }
    void build(const vector<T> &v) {
        build(v.size(), [&](int i) -> T { return v[i]; });
    }
    template <typename F>
    void build(int m, F f) {
        n = m;
        dat.clear();
        dat.reserve(n);
        total = T(0);
        for (int i = 0; i < n; ++i) dat.emplace_back(f(i));
        for (int i = 1; i < n + 1; ++i) {
            int j = i + (i & -i);
            if (j < n + 1) {
                dat[j - 1] += dat[i - 1];
            }
        }
        total = pre_sum(m);
    }

    void add(int k, T x) {
        total += x;
        for (++k; k < n + 1; k += k & -k) {
            dat[k - 1] += x;
        }
    }

    T sum_all() { return total; }
    T prod(int k) { return pre_sum(k); }
    T pre_sum(int k) {
        chmin(k, n);
        T res(0);
        for (; k > 0; k -= k & -k) {
            res += dat[k - 1];
        }
        return res;
    }
    T prod(int l, int r) {
        chmax(l, 0);
        chmin(r, n);
        if (l == 0) return pre_sum(r);
        T pos = T(0), neg = T(0);
        while (l < r) {
            pos += dat[r - 1];
```

```
            r -= r & -r;
        }
        while (r < l) {
            neg += dat[l - 1];
            l -= l & -l;
        }
        return pos - neg;
    }
    vector<T> get_all() {
        vector<T> res(n);
        for (int i = 0; i < n; ++i) {
            res[i] = prod(i, i + 1);
        }
        return res;
    }
};
struct Fenw01 {
    int N, n;
    vector<ull> dat;
    Fenw<int> bit;
    Fenw01() {}
    Fenw01(int n) { build(n); }

    void build(int m) {
        N = m;
        n = ceil(N + 1, 64);
        dat.assign(n, ull(0));
        bit.build(n);
    }

    void add(int k, int x) {
        if (x == 1) add(k);
        if (x == -1) remove(k);
    }

    void add(int k) {
        dat[k / 64] |= 1ull << (k % 64);
        bit.add(k / 64, 1);
    }
    void remove(int k) {
        dat[k / 64] &= ~(1ull << (k % 64));
        bit.add(k / 64, -1);
    }

    int sum_all() { return bit.sum_all(); }
    int pre_sum(int k) {
        int ans = bit.prod(k / 64);
        ans += popcount(dat[k / 64] & ((1ull << (k % 64)) - 1));
        return ans;
    }
    int prod(int k) { return pre_sum(k); }
    int prod(int l, int r) {
        if (l == 0) return pre_sum(r);
        int ans = 0;
        ans -= popcount(dat[l / 64] & ((1ull << (l % 64)) - 1));
        ans += popcount(dat[r / 64] & ((1ull << (r % 64)) - 1));
        ans += bit.prod(l / 64, r / 64);
        return ans;
    }
};
```

hashmap.hpp

```cpp
template <typename Val>
struct hash_map {
    hash_map(uint n = 0) { build(n); }
    void build(uint n) {
        uint k = 8;
        while (k < (n << 1)) k <<= 1;
        cap = k >> 1, msk = k - 1;
        key.resize(k), val.resize(k), used.assign(k, 0);
    }
    void clear() {
        used.assign(used.size(), 0);
        cap = msk + 1 >> 1;
    }
    int size() { return used.size() / 2 - cap; }
    int index(const ull &k) {
        int i = 0;
        for (i = hash(k); used[i] and key[i] != k; i = (i + 1) & msk) {}
        return i;
    }

    Val& operator[](const ull &k) {
        if (cap == 0) extend();
        int i = index(k);
        if (not used[i]) { used[i] = 1, key[i] = k, val[i] = Val{}, --cap; }
        return val[i];
    }

    Val get(const ull &k, Val default_value) {
        int i = index(k);
        return (used[i] ? val[i] : default_value);
    }

    bool count(const ull &k) {
        int i = index(k);
        return used[i] and key[i] == k;
    }

    // f(key, val);
    template <typename F>
    void enumerate_all(F f) {
        for (int i = 0, ed = used.size(); i < ed; ++i) {
            if (used[i]) f(key[i], val[i]);
        }
    }
private :
    uint cap, msk;
    vector<ull> key;
    vector<Val> val;
    vector<bool> used;

    ull hash(ull x) {
        static const ull FIXED_RANDOM = std::chrono::steady_clock::now().time_since_epoch().count();
        x += FIXED_RANDOM;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return (x ^ (x >> 31)) & msk;
    }

    void extend() {
```

```
        vector<pair<ull, Val>> dat;
        dat.reserve(used.size() / 2 - cap);
        for (int i = 0, ed = used.size(); i < ed; ++i) {
            if (used[i]) dat.emplace_back(key[i], val[i]);
        }
        build(dat.size() << 1);
        for (auto &[a, b] : dat) (*this)[a] = b;
    }
};
```

heap.hpp

```
template <typename T>
struct heap {
    priority_queue<T> p, q;

    void push(const T &x) {
        if (!q.empty() and q.top() == x) {
            q.pop();
            while (!q.empty() and q.top() == p.top()) {
                p.pop(), q.pop();
            }
        } else {
            p.push(x);
        }
    }
    template <typename... Args>
    void emplace(Args &&...args) {
        if (!q.empty() and q.top() == T {std::forward<Args>(args)...}) {
            q.pop();
            while (!q.empty() and q.top() == p.top()) {
                p.pop(), q.pop();
            }
        } else {
            p.emplace(std::forward<Args>(args)...);
        }
    }
    void pop() {
        p.pop();
        while (!q.empty() and p.top() == q.top()) {
            p.pop(), q.pop();
        }
    }
    void pop(const T &x) {
        if (p.top() == x) {
            p.pop();
            while (!q.empty() and p.top() == q.top()) {
                p.pop(), q.pop();
            }
        } else {
            q.push(x);
        }
    }
    T top() { return p.top(); }
    bool empty() { return p.empty(); }
};
```

rollback_array.hpp

```
template <typename T>
struct RollbackArray {
```

```
    int N;
    std::vector<T> dat;
    std::vector<std::pair<int, T>> history;
    RollbackArray(std::vector<T> x) : N(x.size()), dat(x) {}
    template <typename F>
    RollbackArray(int N, F f) : N(N) {
        dat.reserve(N);
        for (int i = 0; i < N; ++i) {
            dat.emplace_back(f(i));
        }
    }
    int time() {
        return history.size();
    }
    void rollback(int t) {
        for (int i = time() - 1; i >= t; --i) {
            auto& [idx, v] = history[i]; dat[idx] = v;
        }
        history.resize(t);
    }
    T get(int idx) {
        return dat[idx];
    }
    void set(int idx, T x) {
        history.emplace_back(idx, dat[idx]);
        dat[idx] = x;
    }
    std::vector<T> get_all() {
        std::vector<T> res(N);
        for (int i = 0; i < N; ++i) {
            res[i] = get(i);
        }
        return res; }
    T operator[](int idx) {
        return dat[idx];
    }
};
```

rollback_dsu.hpp

```
#include "rollback_array.hpp"
struct rollback_dsu {
    RollbackArray<int> dat;
    rollback_dsu(int n) : dat(std::vector<int>(n, -1)) {}
    int operator[](int v) {
        while (dat.get(v) >= 0) {
            v = dat.get(v);
        }
        return v;
    }
    int size(int v) {
        return -dat.get((*this)[v]);
    }
    int time() {
        return dat.time();
    }
    void rollback(int t) {
        dat.rollback(t);
    }
    bool merge(int a, int b) {
        a = (*this)[a], b = (*this)[b];
        if (a == b) {
```

```
            return false;
        }
        if (dat.get(a) > dat.get(b)) {
            std::swap(a, b);
        }
        dat.set(a, dat.get(a) + dat.get(b));
        dat.set(b, a);
        return true;
    }
};
```

splay.hpp

```
constexpr int N = 1'000'000 + 10 << 2;
struct MeIoN_Splay {
    int fa[N], ch[N][2], num[N], siz[N], val[N], cnt, root;
    int newnode(int n) {
        val[++cnt] = n;
        ch[cnt][0] = ch[cnt][1] = 0;
        num[cnt] = siz[cnt] = 1;
        return cnt;
    }
    int dir(int x) { return ch[fa[x]][1] == x; }
    void upd(int x) { siz[x] = num[x] + siz[ch[x][0]] + siz[ch[x][1]]; }
    void rotate(int x) {
        int d = dir(x), f = fa[x];
        if ((ch[f][d] = ch[x][d ^ 1]) != 0) fa[ch[f][d]] = f;
        if ((fa[x] = fa[f]) != 0) ch[fa[x]][dir(f)] = x;
        else root = x;
        upd(ch[x][d ^ 1] = f);
        upd(fa[f] = x);
    }
    void splay(int x) {
        for (int f; (f = fa[x]) != 0; rotate(x))
            if (fa[f]) rotate(dir(f) == dir(x) ? f : x);
    }
    void insert(int x) {
        if (root == 0) {
            root = newnode(x);
            return;
        }
        int o = root;
        while (1) {
            int d = (val[o] < x);
            if (val[o] == x) {
                ++num[o];
                ++siz[o];
                break;
            } if (ch[o][d] == 0) {
                ch[o][d] = newnode(x);
                fa[ch[o][d]] = o;
                o = ch[o][d];
                break;
            } else {
                o = ch[o][d];
            }
        }
        splay(o);
        upd(o);
    }
    void find(int x) {
        // find the max v in tree that v <= x
```

```
        int o = root, res = 0;
        while (o != 0) {
            if (val[o] == x) res = o, o = 0;
            else if (val[o] < x) res = o, o = ch[o][1];
            else if (ch[o][0] == 0 and res == 0) res = o, o = 0;
            else o = ch[o][0];
        }
        splay(res);
    }
    void del(int x) {
        find(x);
        if (num[root] > 1) {
            --num[root];
            upd(root);
        } else if (ch[root][0] == 0) {
            fa[root = ch[root][1]] = 0;
        } else if (ch[root][1] == 0) {
            fa[root = ch[root][0]] = 0;
        } else {
            int _tmp = root, o = ch[root][0];
            while (ch[o][1] != 0) o = ch[o][1];
            splay(o);
            ch[o][1] = ch[_tmp][1];
            upd(fa[ch[_tmp][1]] = o);
        }
    }
    int kth(int x) {
        int o = root;
        while (1) {
            if (x <= siz[ch[o][0]]) o = ch[o][0];
            else if (x <= siz[ch[o][0]] + num[o]) return val[o];
            else x -= siz[ch[o][0]] + num[o], o = ch[o][1];
        }
    }
} splay;
/*
    int n, m, ans;
    std::cin >> n >> m;
    for (int i = 0, op, x; i < m; ++i) {
        std::cin >> op >> x;
        if (op == 1) {
            // insert
            g.insert(x);
        } else if (op == 2) {
            // del
            g.del(x);
        } else if (op == 3) {
            // rank of x ( may no x
            g.insert(x);
            g.find(x);
            ans = g.siz[g.ch[g.root][0]] + 1;
            g.del(x);
        } else if (op == 4) {
            // the num ranks x
            ans = g.kth(x);
        } else if (op == 5) {
            // the num lower than x;
            g.find(x - 1);
            ans = g.val[g.root];
        } else {
            // the num greater than x
            g.find(x);
```

```
            if (g.val[g.root] > x) {
                ans = g.val[g.root];
            } else {
                int o = g.ch[g.root][1];
                while (g.ch[o][0] != 0) o = g.ch[o][0];
                ans = g.val[o];
            }
        }
    }
    std::cout << ans << '\n';
*/
```

sqrt_tree.hpp

```
template <typename Monoid>
struct sqrt_tree {   // nlog^2 预处理 O1查询区间信息 满足结合律
    using MX = Monoid;
    using X = typename MX::value_type;

    static constexpr int K = 3;
    static constexpr uint SZ[] = {8, 64, 4096};
    static constexpr uint MASK[] = {7, 63, 4095};

    int N;
    // 元となる静的な列
    vector<X> A;
    // 各階層に対して，ブロック先頭からある要素まで [s,i]
    // 各階層に対して，ある要素からブロック末尾まで [i,t]
    vector<vector<X>> PREF, SUFF;
    // 各階層に対して，あるブロックからあるブロックまで
    vector<vector<X>> BETWEEN;

    sqrt_tree() {}
    template <typename F>
    sqrt_tree(int n, F f) {
        build(n, f);
    }
    sqrt_tree(const vector<X>& v) {
        build(v.size(), [&](int i) -> X { return v[i]; });
    }

    template <typename F>
    void build(int n_, F f) {
        N = n_;
        assert(N <= (1 << 24));
        A.reserve(N);
        for (int i = 0; i < N; ++i) A.emplace_back(f(i));
        // まず prefix, suffix の構築
        PREF.assign(K, A), SUFF.assign(K, A);
        for (int k = 0; k < K; ++k) {
            for (int i = 0; i < N; ++i) {
                if (i & MASK[k]) PREF[k][i] = MX::op(PREF[k][i - 1], A[i]);
            }
            for (int i = N; --i; ) {
                if (i & MASK[k]) SUFF[k][i - 1] = MX::op(A[i - 1], SUFF[k][i]);
            }
        }
        // between の構築
        BETWEEN.resize(K);
        for (int k = 0; k < K; ++k) {
            // n : 全体の小ブロックの個数
            auto get = [&](int i) -> X { return SUFF[k][SZ[k] * i]; };
```

```
            int n = N / SZ[k];
            int s = 0;
            for (int r = 0; r < n; ++r) {
                if (r % SZ[k] == 0) s = r;
                BETWEEN[k].emplace_back(get(r));
                for (int l = r - 1; l >= s; --l) {
                    BETWEEN[k].emplace_back(MX::op(get(l), BETWEEN[k].back()));
                }
            }
        }
    }

    static constexpr int BIT_TO_LAYER[] = {0, 0, 0, 1, 1, 1, 2, 2, 2, 2, 2, 2,
                                           3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3};

    X prod(int L, int R) {
        assert(0 <= L && L <= R && R <= N);
        if (L == R) return MX::unit();
        if (L + 1 == R) return A[L];
        --R;
        int k = BIT_TO_LAYER[topbit(L ^ R)];
        if (k == 0) {
            // 長さ SZ[0] のブロックにクエリが収まっている. 愚直に.
            X x = A[L];
            for (int i = L + 1; i < R + 1; ++i) x = MX::op(x, A[i]);
            return x;
        }
        --k;
        // 同じ長さ SZ[k+1] のブロック内にある. 違う SZ[k] ブロック内にある.
        uint a = L / SZ[k], b = R / SZ[k];
        assert(a < b);
        X &x1 = SUFF[k][L], &x2 = PREF[k][R];
        if (a + 1 == b) return MX::op(x1, x2);
        ++a, --b;
        // [a,b] 番目の SZ[k]-block の間を取得する
        // BETWEEN のどこにデータが置いてあるか調べる
        uint m = a / SZ[k];
        a &= MASK[k], b &= MASK[k];
        uint idx = m * (SZ[k] / 2) * (SZ[k] + 1);
        idx += (b + 1) * (b + 2) / 2 - 1 - a;
        return MX::op(x1, MX::op(BETWEEN[k][idx], x2));
    }
};
```

st_table.hpp

```
namespace RMQ {
    vector<int> lg(2);
    template <typename T> struct maxtable {
        vector<T> a;
        vector<vector<T>> st;
        static int x;
        maxtable(const vector<T> &b):a(b) {
            int n = a.size(), i, j, k, r;
            while (lg.size()<=n) lg.emplace_back(lg[lg.size() >> 1] + 1);
            st.assign(lg[n] + 1,vector<T>(n));
            std::iota(st[0].begin(), st[0].end(), 0);
            for (j = 1; j <= lg[n]; j++) {
                r = n - (1 << j);
                k = 1 << j - 1;
                for (i = 0; i <= r; i++)
                    st[j][i] = a[st[j-1][i]] < a[st[j-1][i+k]] ? st[j-1][i+k] : st[j-1][i];
```

```
            }
        }
        T quis(int l, int r) const { // [l, r]
            assert(0 <= l and l <= r and r < a.size());
            int z = lg[r - l + 1];
            return std::max(a[st[z][l]], a[st[z][r - (1 << z) + 1]]);
        }
        int rmp(int l,int r) const {
            assert(0 <= l and l <= r and r < a.size());
            int z = lg[r-l+1];
            return a[st[z][l]] < a[st[z][r - (1 << z) + 1]] ? st[z][r - (1 << z) + 1] : st[z][l];
        }
    };
} using RMQ::maxtable;
```

rollback mo

```
NAME MeIoN_is_UMP45(){
    int n;
    std::cin >> n;
    vector<int> a(n);
    std::cin >> a;
    auto b = a;
    unique(b);
    for (auto &x : a)
        x = MEION::lower_bound(b, x) - b.begin();
    const int sz = b.size();
    std::cin >> m;
    const int B = std::ceil(std::sqrt(n)); assert(B);
    using aa = array<int, 3>;
    vector<aa> q(m);
    vector<vector<aa>> Q(B);
    for (int i = 0; auto &[l, r, id] : q) {
        std::cin >> l >> r, --l, --r, id = i++;
    }
    MEION::sort(q, [&](const aa &a, const aa &b){
        if (a[0] / B == b[0] / B) return a[1] < b[1]; return a[0] < b[0];
    });
    vector<int> pl(sz), pr(sz);
    auto quis = [&] (int l, int r) {
        int ret = 0;
        for (int i = l; i <= r; ++i) {
            if (~pl[a[i]]) MAX(ret, i - pl[a[i]]);
            else pl[a[i]] = i;
        }
        for (int i = l; i <= r; ++i) pl[a[i]] = -1;
        return ret;
    };
    vector<int> res(m);
    vector<int> del;
    del.reserve(n << 2);
    int pla(-1);
    for (int i = 0, ie = (n - 1) / B + 1; i <= ie; ++i) {
        int maxr = std::min(i * B + B - 1, n - 1), nr = maxr - 1, ret = 0;
        MEION::fill(pl, -1), MEION::fill(pr, -1);
        while (pla + 1 < m and q[pla + 1][0] / B == i) {
            ++pla;
            const auto &[L, R, id] = q[pla];
            if (R / B == i) {
                res[id] = quis(L, R);
                continue;
            }
```

```
            while (nr < R) {
                ++nr;
                pr[a[nr]] = nr;
                if (~pl[a[nr]]) MAX(ret, pr[a[nr]] - pl[a[nr]]);
                else pl[a[nr]] = nr;
            }
            int nl = maxr, ans = ret;
            while (nl > L) {
                --nl;
                if (~pr[a[nl]]) {
                    MAX(ans, pr[a[nl]] - nl);
                } else {
                    pr[a[nl]] = nl;
                    del.emplace_back(a[nl]);
                }
            }
            res[id] = ans;
            while (del.size()) {
                pr[del.back()] = -1; del.pop_back();
            }
        }
    }
    for (const int i : res) std::cout << i << '\n';
}
```

block

```
struct block {
    int n, off;
    int a[0721], b[0721];
    block(const vector<int> &x, int l, int r)
        : n(r - l), off(l) {
            for (int i = l; i < r; ++i) {
                a[i - l] = x[i];
            }
            memcpy(b, a, sizeof a);
            radix_sort(n, b);
        }
    void upd(int pla, int x) {
        pla -= off;
        if (pla < 0 or pla > n - 1) return;
        a[pla] = x;
        memcpy(b, a, sizeof a);
        radix_sort(n, b);
    }
    int quis(int l, int r, int L, int R) {
        l -= off, r -= off;
        chmax(l, 0);
        chmin(r, n);
        if (l > r - 1) return 0;
        if (r - l == n) {
            return int(std::lower_bound(b, b + n, R) -
                       std::lower_bound(b, b + n, L));
        } else {
            int ans = 0;
            for (int i = l; i < r; ++i) {
                ans += a[i] > L - 1 and a[i] < R;
            }
            return ans;
        }
    }
```

```
35  };
```

flow

max_flow.hpp

```cpp
namespace FL {
    using flowt = long long;
    constexpr int M = 3000000, N = 40000 + 10;
    int y[M], nxt[M],
        gap[N], fst[N], c[N], pre[N], q[N], dis[N];
    pair<int, int> e[M];
    flowt f[M];
    int S, T, tot, Tn;
    void III(int s, int t, int tn) {
        tot = 1;
        assert(tn < N);
        for (int i = 0, iE = tn; i < iE; ++i) fst[i] = 0;
        S = s, T = t, Tn = tn;
    }
    void add(int u, int v, flowt c1, flowt c2 = 0) {
        tot++, y[tot] = v, f[tot] = c1, nxt[tot] = fst[u], fst[u] = tot;
        e[tot] = {u, v};
        tot++, y[tot] = u, f[tot] = c2, nxt[tot] = fst[v], fst[v] = tot;
        e[tot] = {v, u};
    }
    flowt sap() {
        int u = S, t = 1;
        flowt flow = 0;
        for (int i = 0; i < Tn; ++i) c[i] = fst[i], dis[i] = Tn, gap[i] = 0;
        q[0] = T, dis[T] = 0, pre[S] = 0;
        for (int i = 0; i < t; ++i) {
            int u = q[i];
            for (int j = fst[u]; j; j = nxt[j])
                if (dis[y[j]] > dis[u] + 1 and f[j ^ 1])
                    q[t++] = y[j], dis[y[j]] = dis[u] + 1;
        }
        for (int i = 0; i < Tn; ++i) gap[dis[i]]++;
        while (dis[S] <= Tn) {
            while (c[u] and (not f[c[u]] or dis[y[c[u]]] + 1 != dis[u]))
                c[u] = nxt[c[u]];
            if (c[u]) {
                pre[y[c[u]]] = c[u] ^ 1;
                u = y[c[u]];
                if (u == T) {
                    flowt minf = inf<flowt>;
                    for (int p = pre[T]; p; p = pre[y[p]])
                        minf = std::min(minf, f[p ^ 1]);
                    for (int p = pre[T]; p; p = pre[y[p]])
                        f[p ^ 1] -= minf, f[p] += minf;
                    flow += minf, u = S;
                }
            } else {
                if (not(--gap[dis[u]])) break;
                int mind = Tn;
                c[u] = fst[u];
                for (int j = fst[u]; j; j = nxt[j])
                    if (f[j] and dis[y[j]] < mind) mind = dis[y[j]], c[u] = j;
                dis[u] = mind + 1;
                gap[dis[u]]++;
                if (u != S) u = y[pre[u]];
            }
        }
```

```
        return flow;
    }
};  // namespace FL
```

max_flow_min_cost.hpp

```
namespace internal {
    template <class E>
    struct csr {
        std::vector<int> start;
        std::vector<E> elist;
        explicit csr(int n, const std::vector<std::pair<int, E>>& edges)
            : start(n + 1), elist(edges.size()) {
            for (auto e: edges) { start[e.first + 1]++; }
            for (int i = 1; i <= n; i++) { start[i] += start[i - 1]; }
            auto counter = start;
            for (auto e: edges) { elist[counter[e.first]++] = e.second; }
        }
    };
    template <class T>
    struct simple_queue {
        std::vector<T> payload;
        int pos = 0;
        void reserve(int n) { payload.reserve(n); }
        int size() const { return int(payload.size()) - pos; }
        bool empty() const { return pos == int(payload.size()); }
        void push(const T& t) { payload.push_back(t); }
        T& front() { return payload[pos]; }
        void clear() {
            payload.clear();
            pos = 0;
        }
        void pop() { pos++; }
    };
} // namespace internal
/*
 ·atcoder library をすこし改変したもの
 ·DAG = true であれば、負辺 OK （1 回目の最短路を dp で行う）
ただし、頂点番号は toposort されていることを仮定している。
*/
template <class Cap = int, class Cost = ll, bool DAG = false>
struct mcf_graph {
public:
    mcf_graph() {}
    explicit mcf_graph(int n) : _n(n) {}
    // frm, to, cap, cost
    int add(int frm, int to, Cap cap, Cost cost) {
        assert(0 <= frm && frm < _n), assert(0 <= to && to < _n), assert(0 <= cap), assert(DAG || 0 <= cost);
        if (DAG) assert(frm < to);
        int m = int(_edges.size());
        _edges.push_back({frm, to, cap, 0, cost});
        return m;
    }
    void DBEUG() {
        std::cout << "flow graph\n";
        std::cout << "frm, to, cap, cost\n";
        for (auto&& [frm, to, cap, flow, cost]: _edges) U(frm, ' ', to, ' ', cap, ' ', cost, '\n');
    }
    struct edge {
        int frm, to;
        Cap cap, flow;
        Cost cost;
```

```
    };
    edge get_edge(int i) {
        int m = int(_edges.size());
        assert(0 <= i && i < m);
        return _edges[i];
    }
    std::vector<edge> edges() { return _edges; }

    // (流量, 費用)
    std::pair<Cap, Cost> flow(int s, int t) {
        return flow(s, t, std::numeric_limits<Cap>::max());
    }
    // (流量, 費用)
    std::pair<Cap, Cost> flow(int s, int t, Cap flow_limit) {
        return slope(s, t, flow_limit).back();
    }
    // 返回流量和费用之间的关系曲线
    std::vector<std::pair<Cap, Cost>> slope(int s, int t) {
        return slope(s, t, std::numeric_limits<Cap>::max());
    }
    std::vector<std::pair<Cap, Cost>> slope(int s, int t, Cap flow_limit) {
        assert(0 <= s && s < _n), assert(0 <= t && t < _n), assert(s != t);
        int m = int(_edges.size());
        std::vector<int> edge_idx(m);
        auto g = [&]() {
            std::vector<int> degree(_n), redge_idx(m);
            std::vector<std::pair<int, _edge>> elist;
            elist.reserve(2 * m);
            for (int i = 0; i < m; i++) {
                auto e = _edges[i];
                edge_idx[i] = degree[e.frm]++;
                redge_idx[i] = degree[e.to]++;
                elist.push_back({e.frm, {e.to, -1, e.cap - e.flow, e.cost}});
                elist.push_back({e.to, {e.frm, -1, e.flow, -e.cost}});
            }
            auto _g = internal::csr<_edge>(_n, elist);
            for (int i = 0; i < m; i++) {
                auto e = _edges[i];
                edge_idx[i] += _g.start[e.frm];
                redge_idx[i] += _g.start[e.to];
                _g.elist[edge_idx[i]].rev = redge_idx[i];
                _g.elist[redge_idx[i]].rev = edge_idx[i];
            }
            return _g;
        }();
        auto result = slope(g, s, t, flow_limit);
        for (int i = 0; i < m; i++) {
            auto e = g.elist[edge_idx[i]];
            _edges[i].flow = _edges[i].cap - e.cap;
        }
        return result;
    }

private:
    int _n;
    std::vector<edge> _edges;
    // inside edge
    struct _edge {
        int to, rev;
        Cap cap;
        Cost cost;
    };
```

```
    std::vector<std::pair<Cap, Cost>> slope(internal::csr<_edge>& g, int s, int t, Cap flow_limit) {
        if (DAG) assert(s == 0 && t == _n - 1);
        std::vector<std::pair<Cost, Cost>> dual_dist(_n);
        std::vector<int> prev_e(_n);
        std::vector<bool> vis(_n);
        struct Q {
            Cost key;
            int to;
            bool operator<(Q r) const { return key > r.key; }
        };
        std::vector<int> que_min;
        std::vector<Q> que;
        auto dual_ref = [&]() {
            for (int i = 0; i < _n; i++) {
                dual_dist[i].second = std::numeric_limits<Cost>::max();
            }
            std::fill(vis.begin(), vis.end(), false);
            que_min.clear();
            que.clear();
            size_t heap_r = 0;
            dual_dist[s].second = 0;
            que_min.push_back(s);
            while (!que_min.empty() || !que.empty()) {
                int v;
                if (!que_min.empty()) {
                    v = que_min.back();
                    que_min.pop_back();
                } else {
                    while (heap_r < que.size()) {
                        heap_r++;
                        std::push_heap(que.begin(), que.begin() + heap_r);
                    }
                    v = que.front().to;
                    std::pop_heap(que.begin(), que.end());
                    que.pop_back();
                    heap_r--;
                }
                if (vis[v]) continue;
                vis[v] = true;
                if (v == t) break;
                Cost dual_v = dual_dist[v].first, dist_v = dual_dist[v].second;
                for (int i = g.start[v]; i < g.start[v + 1]; i++) {
                auto e = g.elist[i];
                if (!e.cap) continue;
                Cost cost = e.cost - dual_dist[e.to].first + dual_v;
                if (dual_dist[e.to].second > dist_v + cost) {
                    Cost dist_to = dist_v + cost;
                    dual_dist[e.to].second = dist_to;
                    prev_e[e.to] = e.rev;
                    if (dist_to == dist_v) {
                        que_min.push_back(e.to);
                    } else {
                        que.push_back(Q{dist_to, e.to});
                    }
                }
                }
            }
            if (!vis[t]) { return false; }

            for (int v = 0; v < _n; v++) {
                if (!vis[v]) continue;
                dual_dist[v].first -= dual_dist[t].second - dual_dist[v].second;
```

```
        }
        return true;
    };

    auto dual_ref_dag = [&]() {
        for (int i = 0; i < _n; i++) {
            dual_dist[i].second = std::numeric_limits<Cost>::max();
        }
        dual_dist[s].second = 0;
        std::fill(vis.begin(), vis.end(), false);
        vis[0] = true;

        for (int v = 0; v < _n; ++v) {
            if (!vis[v]) continue;
            Cost dual_v = dual_dist[v].first, dist_v = dual_dist[v].second;
            for (int i = g.start[v]; i < g.start[v + 1]; i++) {
                auto e = g.elist[i];
                if (!e.cap) continue;
                Cost cost = e.cost - dual_dist[e.to].first + dual_v;
                if (dual_dist[e.to].second > dist_v + cost) {
                    vis[e.to] = true;
                    Cost dist_to = dist_v + cost;
                    dual_dist[e.to].second = dist_to;
                    prev_e[e.to] = e.rev;
                }
            }
        }
        if (!vis[t]) { return false; }

        for (int v = 0; v < _n; v++) {
            if (!vis[v]) continue;
            dual_dist[v].first -= dual_dist[t].second - dual_dist[v].second;
        }
        return true;
    };

    Cap flow = 0;
    Cost cost = 0, prev_cost_per_flow = -1;
    std::vector<std::pair<Cap, Cost>> result = {{Cap(0), Cost(0)}};
    while (flow < flow_limit) {
    if (DAG && flow == 0) {
        if (!dual_ref_dag()) break;
    } else {
        if (!dual_ref()) break;
    }
    Cap c = flow_limit - flow;
    for (int v = t; v != s; v = g.elist[prev_e[v]].to) {
        c = std::min(c, g.elist[g.elist[prev_e[v]].rev].cap);
    }
    for (int v = t; v != s; v = g.elist[prev_e[v]].to) {
        auto& e = g.elist[prev_e[v]];
        e.cap += c;
        g.elist[e.rev].cap -= c;
    }
    Cost d = -dual_dist[s].first;
    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;
```

```
243         }
244     };
```

geo

两圆面积覆盖

```
ld dis = length(p1 - p2);
ld al = 2.0l * std::acos((r * r + dis * dis - rr * rr) / (2.0l * r * dis));
ld R2 = 0.5l * al * r * r - 0.5l * r * r * sin(al);
ld beta =
    2.0l * std::acos((rr * rr + dis * dis - r * r) / (2.0l * rr * dis));
ld R1 = 0.5l * beta * rr * rr - 0.5l * rr * rr * sin(beta);
ld R = R1 + R2;
std::cout << R;
```

正n角形面积

```
ll n;               // 角数
ll r;               // 外接圆面积
ld S(ll n, ll r) {  // 菱形小块S
    ld S;
    S = (r * r * std::sin(pi / n) * std::sin(pi / (2 * n))) /
        (2 * std::sin(pi - pi * 3 / 2 / n));
    return S;
}
ld SS(ll n, ll r) {  // n角形面积
    ld res = S(n, r) * n * 2;
    return res;
}
```

正n锥体体积

```
ld V(ll n, ll a) {
    ld res(0);
    res = a * a * a * n / (12 * std::tan(pi / n)) *
          std::sqrt(1 - 1 / (4 * std::sin(pi / n) * std::sin(pi / n)));
    return res;
}
```

1-base.hpp

```
using RE = long double;
template <typename T = int>
struct point {
    T x, y;
    point() : x(0), y(0) {}

    template <typename A, typename B>
    point(A x, B y) : x(x), y(y) {}

    template <typename A, typename B>
    point(pair<A, B> p) : x(p.first), y(p.second) {}

    point operator+=(const point p) {
        x += p.x, y += p.y;
        return *this;
    }
    point operator-=(const point p) {
        x -= p.x, y -= p.y;
        return *this;
    }
    point operator+(point p) const {
        return {x + p.x, y + p.y};
```

```
    }
    point operator-(point p) const {
        return {x - p.x, y - p.y};
    }
    bool operator==(point p) const {
        return x == p.x and y == p.y;
    }
    bool operator!=(point p) const {
        return x != p.x or y != p.y;
    }
    point operator-() const {
        return {-x, -y};
    }
    point operator*(T t) const {
        return {x * t, y * t};
    }
    point operator/(T t) const {
        return {x / t, y / t};
    }

    bool operator<(point p) const {
        if (x != p.x) return x < p.x;
        return y < p.y;
    }
    bool operator>(point p) const {
        if (x != p.x) return x > p.x;
        return y > p.y;
    }
    T dot(const point &other) const {
        return x * other.x + y * other.y;
    }
    T det(const point &other) const {
        return x * other.y - y * other.x;
    }
    T square() const {
        return x * x + y * y;
    }

    RE length() { return sqrtl(x * x + y * y); }
    RE angle() { return std::atan2(y, x); }

    point rotate(double theta) {
        static_assert(not std::is_integral<T>::value);
        RE c = std::cos(theta), s = std::sin(theta);
        return point{c * x - s * y, s * x + c * y};
    }
    point rot90(bool ccw = 1) {
        return (ccw ? point{-y, x} : point{y, -x});
    }
};

template <typename T>
std::istream& operator>>(std::istream& is, point<T>& any) {
    is >> any.x >> any.y;
    return is;
}
template <typename T>
std::ostream& operator<<(std::ostream& os, const point<T>& any) {
    os << any.x << ' ' << any.y;
    return os;
}

```

```cpp
// A -> B -> Cと進むときに，左转为 +1，右转为 -1。
template<typename T>
int ccw(point<T> a, point<T> b, point<T> c) {
    T x = (b - a).det(c - a);
    if (x > 0) return 1;
    if (x < 0) return -1;
    return 0;
}

template <typename REAL = long double, typename T, typename U>
REAL dist(point<T> a, point<U> b) {
    REAL dx = REAL(a.x) - REAL(b.x);
    REAL dy = REAL(a.y) - REAL(b.y);
    return std::sqrt(dx * dx + dy * dy);
}

// ax+by+c
template <typename T>
struct line {
    T a, b, c;
    line(T a, T b, T c) : a(a), b(b), c(c) {}
    line(point<T> A, point<T> B) {
        a = A.y - B.y;
        b = B.x - A.x;
        c = A.x * B.y - A.y * B.x;
    }
    line(T x1, T y1, T x2, T y2) : line(point<T>(x1, y1), point<T>(x2, y2)) {}

    template <typename U>
    U eval(point<U> p) const {
        return a * p.y + b * p.y + c;
    }

    template <typename U>
    T eval(U x, U y) const {
        return a + x + b * y + c;
    }

    void normalize() {
        static_assert(std::is_same_v<T, int> or std::is_same_v<T, long long>);
        T gcd = std::gcd(std::gcd(std::abs(a), std::abs(b)), std::abs(c));
        a /= gcd, b /= gcd, c /= gcd;
    }

    bool parallel(line other) const {
        return a * other.b - b * other.a == 0;
    }
    bool is_orthoginal(line other) const {
        return a * other.a + b * other.b == 0;
    }
};

template <typename T>
struct segment {
    point<T> a, b;

    segment(point<T> a, point<T> b) : a(a), b(b) {}
    segment(T x1, T y1, T x2, T y2) : segment(point<T>(x1, y1), point<T>(x2, y2)) {}

    bool contain(point<T> c) const {
        T det = (c - a).det(b - a);
        if (det != 0) return 0;
```

```
        return (c - a).dot(b - a) >= 0 and (c - b).dot(a - b) >= 0;
    }

    line<T> to_line() { return line(a, b); }
};

template <typename REAL>
struct circle {
    point<REAL> O;
    REAL r;
    circle() : O(0, 0), r(0) {}
    circle(point<REAL> O, REAL r) : O(O), r(r) {}
    circle(REAL x, REAL y, REAL r) : O(x, y), r(r) {}
    template <typename T>
    bool contain(point<T> p){
        REAL dx = p.x - O.x, dy = p.y - O.y;
        return dx * dx + dy * dy <= r * r;
    }
};

// 反射
template <typename T, typename U>
point<RE> reflection(point<T> p, line<U> l) {
    RE t = RE(l.eval(p)) / (l.a * l.a + l.b * l.b);
    RE x = p.x - 2 * t * l.a;
    RE y = p.y - 2 * t * l.b;
    return point<RE>(x, y);
}

// 不平行假定
template <typename REAL = long double, typename T>
point<REAL> cross_point(const line<T> l1, const line<T> l2) {
    T det = l1.a * l2.b - l1.b * l2.a;
    assert(det != 0);
    REAL x = -REAL(l1.c) * l2.b + REAL(l1.b) * l2.c;
    REAL y = -REAL(l1.a) * l2.c + REAL(l1.c) * l2.a;
    return point<REAL>(x / det, y / det);
}
template <typename REAL = long double, typename T>
point<REAL> line_x_line(const line<T> l1, const line<T> l2) {
    return cross_point<REAL, T>(l1, l2);
}

// 0: 0交点
// 1: 1交点
// 2: 无数交点
template <typename T>
int count_cross(segment<T> s1, segment<T> s2, bool include_ends) {
    static_assert(!std::is_floating_point<T>::value);
    line<T> l1 = s1.to_line();
    line<T> l2 = s2.to_line();
    if (l1.parallel(l2)) {
        if (l1.eval(s2.a) != 0) return 0;
        // 4 点在同一直線上
        T a1 = s1.a.x, b1 = s1.b.x;
        T a2 = s2.a.x, b2 = s2.b.x;
        if (a1 == b1) {
            a1 = s1.a.y, b1 = s1.b.y;
            a2 = s2.a.y, b2 = s2.b.y;
        }
        if (a1 > b1) std::swap(a1, b1);
        if (a2 > b2) std::swap(a2, b2);
```

```
        T a = std::max(a1, a2);
        T b = std::min(b1, b2);
        if (a < b) return 2;
        if (a > b) return 0;
        return (include_ends ? 1 : 0);
    }
    // 不平行場合
    T a1 = l2.eval(s1.a), b1 = l2.eval(s1.b);
    T a2 = l1.eval(s2.a), b2 = l1.eval(s2.b);
    if (a1 > b1) std::swap(a1, b1);
    if (a2 > b2) std::swap(a2, b2);
    bool ok1 = 0, ok2 = 0;
    if (include_ends) {
        ok1 = ((a1 <= T(0)) and (T(0) <= b1));
        ok2 = ((a2 <= T(0)) and (T(0) <= b2));
    } else {
        ok1 = ((a1 < T(0)) and (T(0) < b1));
        ok2 = ((a2 < T(0)) and (T(0) < b2));
    }
    return (ok1 and ok2 ? 1 : 0);
}

template <typename REAL, typename T>
vector<point<REAL>> cross_point(const circle<T> C, const line<T> L) {
    T a = L.a, b = L.b, c = L.a * (C.O.x) + L.b * (C.O.y) + L.c;
    T r = C.r;
    bool sw = 0;
    if (std::abs(a) < std::abs(b)) {
        std::swap(a, b);
        sw = 1;
    }
    // ax + by + c = 0, x ^ 2 + y ^ 2 = r ^ 2
    T D = 4 * c * c * b * b - 4 * (a * a + b * b) * (c * c - a * a * r * r);
    if (D < 0) return {};
    REAL sqD = sqrtl(D);
    REAL y1 = (-2 * b * c + sqD) / (2 * (a * a + b * b));
    REAL y2 = (-2 * b * c - sqD) / (2 * (a * a + b * b));
    REAL x1 = (-b * y1 - c) / a;
    REAL x2 = (-b * y2 - c) / a;
    if (sw) std::swap(x1, y1), std::swap(x2, y2);
    x1 += C.O.x, x2 += C.O.x;
    y1 += C.O.y, y2 += C.O.y;
    if (D == 0) {
        return {point<REAL>(x1, y1)};
    }
    return {point<REAL>(x1, y1), point<REAL>(x2, y2)};
}

template <typename REAL, typename T>
std::tuple<bool, point<T>, point<T>> cross_point_circle(circle<T> C1,
                                                        circle<T> C2) {
    using P = point<T>;
    P O {0, 0};
    P A = C1.O, B = C2.O;
    if (A == B) return {false, O, O};
    T d = (B - A).length();
    REAL cos_val = (C1.r * C1.r + d * d - C2.r * C2.r) / (2 * C1.r * d);
    if (cos_val < -1 || 1 < cos_val) return {false, O, O};
    REAL t = std::acos(cos_val);
    REAL u = (B - A).angle();
    P X = A + P {C1.r * std::cos(u + t), C1.r * std::sin(u + t)};
    P Y = A + P {C1.r * std::cos(u - t), C1.r * std::sin(u - t)};
```

```
271     return {true, X, Y};
272 }
```

2-apollonian_circle.hpp

```cpp
#include "1-base.hpp"

// https://codeforces.com/contest/2/problem/C
template <typename REAL, typename T>
circle<REAL> apollonian_circle(point<T> A, point<T> B, T a, T b) {
    assert(a != b);
    point<REAL> X = (A * b + B * a) / (a + b);
    point<REAL> Y = (A * b - B * a) / (b - a);
    point<REAL> O = (X + Y) / 2.L;
    REAL r = dist<REAL>(X, O);
    return circle<REAL>(O.x, O.y, r);
}
```

3-angle_sort.hpp

```cpp
#include "1-base.hpp"

template <typename T>
int lower_or_upper(const point<T> &p) {
    if (p.y != 0) return (p.y > 0 ? 1 : -1);
    if (p.x > 0) return -1;
    if (p.x < 0) return 1;
    return 0;
}

template <typename T>
int angle_cmp_3(const point<T> &L, const point<T> &R) {
    int a = lower_or_upper(L), b = lower_or_upper(R);
    if (a != b) return (a < b ? -1 : +1);
    T det = L.det(R);
    if (det > 0) return -1;
    if (det < 0) return 1;
    return 0;
}

template <typename T>
vector<int> angle_sort(const vector<point<T>> &v) {
    vector<int> rk(v.size());
    std::iota(rk.begin(), rk.end(), 0);
    sort(rk, [&](auto &L, auto &R) -> bool{
        return (angle_cmp_3(v[L], v[R]) == -1);
    });
    return rk;
}

template <typename T>
vector<int> angle_sort(const vector<pair<T, T>> &v) {
    vector<point<T>> tmp(v.size());
    for (int i = 0, ed = v.size(); i < ed; ++i) {
        tmp[i] = point<T>(v[i]);
    }
    return angle_sort(tmp);
}
```

4-closest_pair.hpp

```cpp
#include "1-base.hpp"
#include "3-angle_sort.hpp"
#include "../random/random.hpp"
#include "../ds/hashmap.hpp"
#include "8-distance.hpp"

using MeIoN_random_hash::shuffle, MeIoN_random_hash::hash_pair;

template <typename T>
pair<int, int> closest_pair(vector<point<T>> points) {
    int n = points.size();
    assert(n > 1);
    hash_map<int> ma(n);
    vector<int> id(n);
    std::iota(id.begin(), id.end(), 0);
    shuffle(id);
    points = rearrange(points, id);

    auto square = [&] (int i, int j) -> T {
        return (points[j] - points[i]).square();
    };

    T best = square(0, 1);
    pair<int, int> res(0, 1);
    T w =  sqrtl(best);

    vector<int> nxt(n, -1);

    auto ins = [&] (int i) -> void {
        ull k = hash_pair(pair{points[i].x / w, points[i].y / w});
        nxt[i] = ma.get(k, -1);
        ma[k] = i;
    };

    auto quis = [&] (int i) -> bool {
        assert(w);
        ll a = points[i].x / w;
        ll b = points[i].y / w;
        bool upd = false;
        for (int dx = -1; dx < 2; ++dx) {
            for (int dy = -1; dy < 2; ++dy) {
                ull k = hash_pair<ll>({a + dx, b + dy});
                int j = ma.get(k, -1);
                while (j != -1) {
                    if (chmin(best, square(i, j))) {
                        upd = true;
                        res = {i, j};
                        w = std::sqrt(best);
                    }
                    j = nxt[j];
                }
            }
        }
        return upd;
    };

    if (best == T(0)) {
        res.first = id[res.first], res.second = id[res.second];
        return res;
    }
    ins(0), ins(1);
```

```
    for (int i = 2; i < n; ++i) {
        if (quis(i)) {
            if (best == T(0)) break;
            ma.build(n);
            for (int k = 0; k < i; ++k) {
                ins(k);
            }
        }
        ins(i);
    }
    res.first = id[res.first], res.second = id[res.second];
    return res;
}

template <typename T>
pair<int, int> closest_pair2(vector<point<T>> points) {
    using RE = long double;
    const int n = points.size();
    if (n == 1) return pair(0, 0);
    ld rd = MeIoN_random_hash::rng(114514) % 360 * 0.114514;
    ld SIN = std::cos(rd), COS = std::sin(rd);
    vector<int> id(n);
    for (int i = 0; i < n; ++i) id[i] = i;
    sort(id, [&](auto &a, auto &b) -> bool {
        return points[a].x * COS - points[a].y * SIN <
               points[b].x * COS - points[b].y * SIN;
    });
    ld best = distance<RE>(points[id[0]], points[id[1]]);
    pair<int, int> ans = pair(id[0], id[1]);
    for (int i = 0; i < n; ++i) {
        for (int k = 1; k < 6 and i + k < n; ++k) {
            if (chmin(best, distance<RE>(points[id[i]], points[id[i + k]]))) {
                ans = pair(id[i], id[i + k]);
            }
        }
    }
    return ans;
}
```

5-hull.hpp

```
#include "1-base.hpp"
// https://qoj.ac/problem/218
template <typename T, bool allow_180 = false>
vector<int> convex_hull(vector<point<T>> &p, string mode = "full",
                        bool sorted = false) {
    assert(mode == "full" or mode == "lower" or mode == "upper");
    int n = p.size();
    if (n == 1) return {0};
    if (n == 2) {
        if (p[0] < p[1]) return {0, 1};
        if (p[0] > p[1]) return {1, 0};
        return {0};
    }
    vector<int> id(n);
    if (sorted) {
        std::iota(id.begin(), id.end(), 0);
    } else {
        id = argsort(p);
    }
    if constexpr (allow_180) {
        for (int i = 0; i < n - 1; ++i) {
```

```
            assert(p[i] != p[i + 1]);
        }
    }
    auto check = [&](int i, int j, int k) -> bool {
        T det = (p[j] - p[i]).det(p[k] - p[i]);
        if constexpr (allow_180) {
            return det >= 0;
        }
        return det > T(0);
    };
    auto cal = [&]() {
        vector<int> res;
        for (const auto &k : id) {
            while (res.size() > 1) {
                auto i = res.end()[-2];
                auto j = res.end()[-1];
                if (check(i, j, k)) break;
                res.pop_back();
            }
            res.emplace_back(k);
        }
        return res;
    };
    vector<int> res;
    if (mode == "full" or mode == "lower") {
        vector<int> Q = cal();
        res.insert(res.end(), Q.begin(), Q.end());
    }
    if (mode == "full" or mode == "upper") {
        if (not res.empty()) res.pop_back();
        rev(id);
        vector<int> Q = cal();
        res.insert(res.end(), Q.begin(), Q.end());
    }
    if (mode == "upper") rev(res);
    while (res.size() > 1 and p[res[0]] == p[res.back()]) res.pop_back();
    return res;
}
```

6-convex_polygon.hpp

```
#include "5-hull.hpp"

// https://codeforces.com/contest/1906/problem/D

template <typename T>
struct convex_polygon {
    using P = point<T>;
    int n;
    vector<P> points;
    T area2;

    // 需要传入一个凸包
    convex_polygon(vector<P> points_) : n((int)points_.size()), points(points_) {
        assert(n > 2);
        area2 = 0;
        for (int i = 0; i < n; ++i) {
            int j = nxt_idx(i), k = nxt_idx(j);
            assert((points[j] - points[i]).det(points[k] - points[i]) >= 0);
            area2 += points[i].det(points[j]);
        }
    }
```

```cpp

    // comp(i, k)
    template <typename F>
    int periodic_min_comp(F comp) {
        int l = 0, m = n, r = n << 1;
        while (true) {
            if (r - l == 2) break;
            int lpos = (l + m) >> 1, rpos = (m + r + 1) >> 1;
            if (comp(lpos % n, m % n)) {
                r = m, m = lpos;
            } else if (comp(rpos % n, m % n)) {
                l = m, m = rpos;
            } else {
                l = lpos, r = rpos;
            }
        }
        return m % n;
    }

    int nxt_idx(int i) { return (i + 1 == n ? 0 : i + 1); }
    int pre_idx(int i) { return (i == 0 ? n - 1 : i - 1); }

    // 中：1, 境界：0, 外：-1. test した.
    int side(P p) {
        int l = 1, r = n - 1;
        T a = (points[l] - points[0]).det(p - points[0]);
        T b = (points[r] - points[0]).det(p - points[0]);
        if (a < 0 or b > 0) return -1;
        // 从点 0 看，点 p 位于 [L, R] 的方向
        while (r - l > 1) {
            int m = l + r >> 1;
            T c = (points[m] - points[0]).det(p - points[0]);
            if (c < 0) {
                r = m, b = c;
            } else {
                l = m, a = c;
            }
        }
        T c = (points[r] - points[l]).det(p - points[l]);
        T x = std::min({a, -b, c});
        if (x < 0) return -1;
        if (x > 0) return 1;
        // on triangle p[0] p[L] p[R]
        if (p == points[0]) return 0;
        if (c != 0 and a == 0 and l != 1) return 1;
        if (c != 0 and b == 0 and r != n - 1) return 1;
        return 0;
    }

    // return {min, idx} 点积最小值 O(log)
    pair<T, int> min_dot(P p) {
        int idx = periodic_min_comp([&](int i, int k) -> bool {
            return points[i].dot(p) < points[k].dot(p);
        });
        return {points[idx].dot(p), idx};
    }

    // return {max, idx} 点积最大值 O(log)
    pair<T, int> max_dot(P p) {
        int idx = periodic_min_comp([&](int i, int k) -> bool {
            return points[i].dot(p) > points[k].dot(p);
        });
```

```
        return {points[idx].dot(p), idx};
    }

    // 计算从一个点 p 观看多边形时，可以看到的多边形的范围
    // 该函数返回两个索引，表示可以看到的范围（考虑反向偏角）
    pair<int, int> visible_range(P p) {
        int a = periodic_min_comp([&](int i, int k) -> bool {
            return ((points[i] - p).det(points[k] - p) < 0);
        });
        int b = periodic_min_comp([&](int i, int k) -> bool {
            return ((points[i] - p).det(points[k] - p) > 0);
        });
        if ((p - points[a]).det(p - points[pre_idx(a)]) == T(0)) a = pre_idx(a);
        if ((p - points[b]).det(p - points[nxt_idx(b)]) == T(0)) b = nxt_idx(b);
        return {a, b};
    }

    // 线段是否与凸多边形相交
    bool check_cross(P pa, P pb) {
        for (int i = 0; i < 2; ++i) {
            std::swap(pa, pb);
            auto [a, b] = visible_range(pa);
            if ((points[a] - pa).det(pb - pa) >= 0) return false;
            if ((points[b] - pa).det(pb - pa) <= 0) return false;
        }
        return true;
    }

    vector<T> AREA;

    // point[i,...,j] (inclusive) 面积 * 2
    T area_between(int i, int k) {
        assert(i <= k and k <= n + i);
        if (k == i + n) return area2;
        i %= n, k %= n;
        if (i > k) k += n;
        if (AREA.empty()) build_AREA();
        return AREA[k] - AREA[i] + (points[k % n].det(points[i]));
    }

    void build_AREA() {
        AREA.resize(n << 1);
        for (int i = 0; i < n; ++i) {
            AREA[n + i] = AREA[i] = points[i].det(points[nxt_idx(i)]);
        }
        AREA.insert(AREA.begin(), T(0));
        for (int i = 0; i < n * 2; ++i) {
            AREA[i + 1] += AREA[i];
        }
    }
};
```

7-points_in_triangles.hpp

```
#include "../ds/fenw.hpp"
#include "3-angle_sort.hpp"
#include "../random/random.hpp"

// 输入点群A和B （Point<ll>）
// query(i,j,k)：返回三角形 Ai Aj Ak 内的 Bl 数量（非负数）
// 预处理 O(NMlogM)，查询 O(1)
// https://codeforces.com/contest/13/problem/D
```

```cpp
struct count_points_in_triangles {
    using P = point<ll>;
    static constexpr int limit = 1'000'000'000 + 10;
    vector<P> A, B;
    vector<int> new_idx;      // 从 O 看到的极角序
    vector<int> points;       // A[i] 与 B[k] 的匹配
    vector<vector<int>> seg;  // 线段 A[i] A[j] 内的 B[k]
    vector<vector<int>> tri;  // OA[i]A[j] 中的 B[k] 的数量
    count_points_in_triangles(const vector<P> &a, const vector<P> &b)
        : A(a), B(b) {
        for (const auto p : A)
            assert(std::max(std::abs(p.x), std::abs(p.y)) < limit);
        for (const auto p : B)
            assert(std::max(std::abs(p.x), std::abs(p.y)) < limit);
        build();
    }

    int count3(int i, int j, int k) {
        i = new_idx[i], j = new_idx[j], k = new_idx[k];
        if (i > j) std::swap(i, j);
        if (j > k) std::swap(j, k);
        if (i > j) std::swap(i, j);
        assert(i < j + 1 and j < k + 1);
        ll d = (A[j] - A[i]).det(A[k] - A[i]);
        if (d == 0) return 0;
        if (d > 0) {
            return tri[i][j] + tri[j][k] - tri[i][k] - seg[i][k];
        }
        int x = tri[i][k] - tri[i][j] - tri[j][k];
        return x - seg[i][j] - seg[j][k] - points[j];
    }

    int count2(int i, int j) {
        i = new_idx[i], j = new_idx[j];
        if (i > j) std::swap(i, j);
        return seg[i][j];
    }

   private:
    P take_origin() {
        // 不要让OAiAj和OAiBj在同一直线上
        // fail prob: at most N(N+M)/LIM
        // return P {-limit, MeIoN_random_hash::rng_64(-limit, limit)};
        return P {-limit, MeIoN_random_hash::rng(-limit, limit)};
    }

    void build() {
        P O = take_origin();
        for (auto &p : A) {
            p = p - O;
        }
        for (auto &p : B) {
            p = p - O;
        }
        int N = A.size(), M = B.size();
        vector<int> id = angle_sort(A);
        A = rearrange(A, id);
        new_idx.resize(N);
        for (int i = 0; i < N; ++i) {
            new_idx[id[i]] = i;
        }

```

```
        id = angle_sort(B);
        B = rearrange(B, id);

        points.assign(N, 0);
        seg.assign(N, vector<int>(N));
        tri.assign(N, vector<int>(N));

        // points
        for (int i = 0; i < N; ++i) {
            for (int k = 0; k < M; ++k) {
                if (A[i] == B[k]) {
                    ++points[i];
                }
            }
        }
        /*
        ll binary_search(F check, ll ok, ll ng, bool check_ok = true) {
            if (check_ok) assert(check(ok));
            while (abs(ok - ng) > 1) {
                auto x = (ng + ok) >> 1;
                (check(x) ? ok : ng) = x;
            }
            return ok;
        }
        */
        int m = 0;
        for (int j = 0; j < N; ++j) {
            while (m < M and A[j].det(B[m]) < 0) ++m;
            vector<P> C(m);
            for (int k = 0; k < m; ++k) {
                C[k] = B[k] - A[j];
            }
            vector<int> id(m);
            for (int i = 0; i < m; ++i) id[i] = i;
            sort(id,
                 [&](auto &a, auto &b) -> bool { return C[a].det(C[b]) > 0; });
            C = rearrange(C, id);
            vector<int> rk(m);
            for (int k = 0; k < m; ++k) {
                rk[id[k]] = k;
            }
            Fenw01 bit(m);

            int k = m;
            for (int i = j; i--;) {
                while (k > 0 and A[i].det(B[k - 1]) > 0) {
                    bit.add(rk[--k], 1);
                }
                P p = A[i] - A[j];
                int lb = binary_search(
                    [&](int n) -> bool {
                        return(n == 0 ? true : C[n - 1].det(p) > 0);
                    }, 0, m + 1);
                int ub = binary_search(
                    [&](int n) -> bool {
                        return(n == 0 ? true : C[n - 1].det(p) >= 0);
                    }, 0, m + 1);
                seg[i][j] += bit.sum(lb, ub), tri[i][j] += bit.sum(lb);
            }
        }
    }
```

```
};
```

8-distance.hpp

```cpp
#include "1-base.hpp"

template <typename REAL = ld, typename T, typename U>
REAL distance(point<T> S, point<U> P) {
    REAL dx = P.x - S.x;
    REAL dy = P.y - S.y;
    return std::sqrt(dx * dx + dy * dy);
}

template <typename REAL = ld, typename T, typename U>
REAL distance(segment<T> S, point<U> P) {
    point<T> A = S.a, B = S.b;
    bool b1 = (B - A).dot(P - A) >= 0;
    bool b2 = (A - B).dot(P - B) >= 0;
    if (b1 and not b2) {
        return distance<REAL, T, T>(B, P);
    }
    if (not b1 and b2) {
        return distance<REAL, T, T>(A, P);
    }
    line<T> L = S.to_line();
    return REAL(std::abs(L.eval(P))) / std::sqrt(REAL(L.a) * L.a + REAL(L.b) * L.b);
}

template <typename REAL, typename T>
REAL distance(segment<T> s1, segment<T> s2) {
    if (count_cross<T>(s1, s2, true)) return REAL(0);
    REAL res = distance<REAL, T, T>(s1, s2.a);
    chmin(res, distance<REAL, T, T>(s1, s2.b));
    chmin(res, distance<REAL, T, T>(s2, s1.a));
    chmin(res, distance<REAL, T, T>(s2, s1.b));
    return res;
}
```

9-furthest_pair.hpp

```cpp
#include "1-base.hpp"
#include "5-hull.hpp"
#include "4-closest_pair.hpp"
#include "8-distance.hpp"

// https://www.luogu.com.cn/problem/P6247
template <typename T>
pair<int, int> furthest_pair(vector<point<T>> points) {
    T best = -1;
    pair<int, int> ans = {-1, -1};

    auto upd = [&](int i, int j) -> void {
        point<T> p = points[i] - points[j];
        ll d = p.dot(p);
        if (chmax(best, d)) ans = pair(i, j);
    };
    upd(0, 1);

    vector<int> id = convex_hull(points);
    int n = id.size();
    if (n == 1) return ans;
```

```
    if (n == 2) return pair(id[0], id[1]);
    /*
    用两条与直径垂直的平行线夹住凸包
    两条平行线夹住的两点是候补点
    将p[i]p[i+1]的相对侧设为候选即可
    */
    for (int i = 0; i < n; ++i) {
        id.emplace_back(id[i]);
    }

    vector<point<T>> C = rearrange(points, id);
    int j = 1;
    for (int i = 0; i < n; ++i) {
        chmax(j, i);
        while (j < 2 * n and (C[i + 1] - C[i]).det(C[j + 1] - C[j]) > 0) ++j;
        upd(id[i], id[j]);
    }
    return ans;
}
```

10-triangle_area.hpp

```
#include "1-base.hpp"

template <typename RE = long double, typename T>
RE triangle_area(point<T> a, point<T> b, point<T> c) {
    return std::abs((b - a).det(c - a)) * 0.5L;
}
template <typename RE = ll, typename T = ll>
RE triangle_area_2(point<T> a, point<T> b, point<T> c) {
    return std::abs((b - a).det(c - a));
}
```

11-in_circle.hpp

```
#include "1-base.hpp"
#include "10-triangle_area.hpp"

template <typename REAL, typename T>
circle<REAL> in_circle(point<T> A, point<T> B, point<T> C) { // 内接圆
    REAL a = distance<REAL, T, T>(B, C);
    REAL b = distance<REAL, T, T>(C, A);
    REAL c = distance<REAL, T, T>(A, B);
    REAL x = (a * A.x + b * B.x + c * C.x) / (a + b + c);
    REAL y = (a * A.y + b * B.y + c * C.y) / (a + b + c);
    REAL r = 2 * triangle_area<REAL>(A, B, C) / (a + b + c);
    return Circle<REAL>(x, y, r);
}
```

12-line_inside_polygon.hpp

```
#include "1-base.hpp"
template <typename T>
bool inside_polygon(const vector<point<T>> &polygon, segment<T> s) { // 判断线段是否在多边形内部
    using P = Point<T>;
    int n = polygon.size();
    int cnt = 0;
    P A = s.A, B = s.B;
    auto prev = [&](int i) -> int { return i == 0 ? n - 1 : i - 1; };
    auto next = [&](int i) -> int { return i == n - 1 ? 0 : i + 1; };
    for (int i = 0; i < n; ++i) {
```

```
        P p = polygon[i], q = polygon[next(i)], r = polygon[prev(i)];
        int a = ccw(A, B, p);
        int b = ccw(A, B, q);
        int c = ccw(A, B, r);
        if (a * b == -1) {
            segment pq(p, q);
            auto L = pq.to_Line();
            T x = L.eval(A), y = L.eval(B);
            if (x < y) {
                x = -x, y = -y;
            }
            if (x <= 0) {
                ++cnt;
            }
            if (0 < x and x < x - y) {
                return false;
            }
        }
        if (a == 0) {
            if (b != c and (b * c < 0 or ccw(p, q, r) > 0)) {
                T t = (p - a).dot(B - A), x = (B - A).dot(B - A);
                if (t <= 0) ++cnt;
                if (0 < t and t < x) {
                    return false;
                }
            }
        }
        return (cnt % 2 == 1);
    }
}
```

13-manhattan_mst.hpp

```
#include "1-base.hpp"
#include "../ds/dsu.hpp"

template <typename T>
vector<vector<pair<int, T>>> manhattan_mst(vector<point<T>> &points) {
    int n = points.size();
    vector<std::tuple<T, int, int>> dat;
    dat.reserve(n << 2);
    vector<int> rk(n);
    std::iota(rk.begin(), rk.end(), 0);

    for (int a = 0; a < 2; ++a) {
        for (auto && [ x, y ] : points) {
            x = -x;
        }
        for (int b = 0; b < 2; ++b) {
            for (auto && [ x, y ] : points) {
                std::swap(x, y);
            }
            sort(rk, [&](const int &i, const int &j) -> bool {
                return points[i].x + points[i].y <
                    points[j].x + points[j].y;
            });

            map<T, int> mp;
            for (const int i : rk) {
                auto & [ x, y ] = points[i];
                for (auto it = mp.lower_bound(-y); it != mp.end();
                     it = mp.erase(it)) {
```

```
                    const int j = it->second;
                    auto &[xj, yj] = points[j];
                    const int dx = x - xj;
                    const int dy = y - yj;
                    if (dy > dx) break;
                    dat.emplace_back(dx + dy, i, j);
                }
                mp[-y] =i;
            }
        }
    }

    sort(dat);
    dsu g(n);
    vector<vector<pair<int, T>>> v(n);
    for (auto &&[cost, i, j] : dat) {
        if (g.merge(i, j)) {
            v[i].emplace_back(j, cost);
            v[j].emplace_back(i, cost);
        }
    }
    return v;
}
```

14-max_norm_sum.hpp

```
#include "1-base.hpp"
#include "3-angle_sort.hpp"

template <typename VAL, typename T>
VAL max_norm_sum(const vector<point<T>> &points) { // 一堆向量选一部分最大模长
    const int n = points.size();
    vector<point<T>> v(points);
    point<T> c{0, 0};
    for (const auto &[x, y] : points) {
        if (y > 0 or (y == 0 and x < 0)) {
            c.x += x, c.y += y;
        }
        v.emplace_back(-x, -y);
    }
    vector<int> rk = angle_sort(v);
    v = rearrange(v, rk);

    auto eval = [&]() -> VAL {
        return VAL(c.x) * c.x + VAL(c.y) * c.y;
    };

    VAL ans = eval();
    for (int i = 0; i < (n << 1); ++i) {
        c = c + v[i];
        chmax(ans, eval());
    }
    return ans;
}
```

15-minkowski_sum.hpp

```
#include "1-base.hpp"
#include "3-angle_sort.hpp"
#include "5-hull.hpp"
#include "6-convex_polygon.hpp"
```

```cpp

template <typename T>
vector<point<T>> minkowski_sum(vector<point<T>> A,
                               vector<point<T>> B) {
    using P = point<T>;
    vector<P> F;
    P p(0, 0);
    for (int i = 0; i < 2; ++i) {
        std::swap(A, B);
        vector<P> points = A;
        int n = points.size();
        for (int i = 0; i < n; ++i) {
            int k = (i + 1) % n;
            F.emplace_back(points[k] - points[i]);
        }
        p = p + *min_element(points.begin(), points.end());
    }
    auto rk = angle_sort(F);
    int n = rk.size();
    F = rearrange(F, rk);
    vector<P> points(n);
    for (int i = 0; i < n - 1; ++i) {
        points[i + 1] = points[i] + F[i];
    }
    P add = p - *min_element(points.begin(), points.end());
    for (auto &x : points) {
        x += add;
    }
    rk = convex_hull(points);
    points = rearrange(points, rk);
    return points;
}
```

16-out_circle.hpp

```cpp
#include "1-base.hpp"

template <typename RE, typename T>
circle<RE> out_circle(point<T> A, point<T> B, point<T> C) {
    RE b1 = B.x - A.x, b2 = B.y - A.y;
    RE c1 = C.x - A.x, c2 = C.y - A.y;
    RE bb = (b1 * b1 + b2 * b2) / 2;
    RE cc = (c1 * c1 + c2 * c2) / 2;

    RE det = b1 * c2 - b2 * c1;
    RE x = (bb * c2 - b2 * cc) / det;
    RE y = (b1 * cc - bb * c1) / det;
    RE r = std::sqrt(x * x + y * y);
    x += A.x, y += A.y;
    return circle<RE>(x, y, r);
}

template <typename T>
int out_circle_side(point<T> A, point<T> B, point<T> C, point<T> p) {
    T d = (B - A).det(C - A);
    assert(d != 0);
    if (d < 0) std::swap(B, C);
    array<point<T>, 3> pts = {A, B, C};
    array<array<T, 3>, 3> mat;
    for (int i = 0; i < 3; ++i) {
        T dx = pts[i].x - p.x, dy = pts[i].y - p.y;
        mat[i][0] = dx, mat[i][1] = dy, mat[i][2] = dx * dx + dy * dy;
```

```
    }
    T det = 0;
    det += mat[0][0] * (mat[1][1] * mat[2][2] - mat[1][2] * mat[2][1]);
    det += mat[0][1] * (mat[1][2] * mat[2][0] - mat[1][0] * mat[2][2]);
    det += mat[0][2] * (mat[1][0] * mat[2][1] - mat[1][1] * mat[2][0]);
    if (det == 0) return 0;
    return (det > 0 ? 1 : -1);
}
```

17-minimum_enclosing_circle.hpp

```
#include "1-base.hpp"
#include "16-out_circle.hpp"
#include "3-angle_sort.hpp"
#include "5-hull.hpp"
#include "15-minkowski_sum.hpp"

template <typename RE, typename T>
std::tuple<circle<RE>, int, int, int> minimum_enclosing_circle( // 一组点的最小包围圆 (某三个点的外接圆
    vector<point<T>> points) {
    const int n = points.size();
    assert(n != 0);
    if (n == 1) {
        circle<RE> C(points[0].x, points[0].y, 0);
        return {C, 0, -1, -1};
    }
    vector<int> rk(n);
    std::iota(rk.begin(), rk.end(), 0);
    for (int i = 0, k; i < n; ++i) {
        k = rng() % (i + 1);
        if (i != k) {
            std::swap(rk[i], rk[k]);
        }
    }

    points = rearrange(points, rk);

    std::tuple<int, int, int> c = {0, -1, -1};
    auto contain = [&](point<T> p) -> bool {
        auto [i, k, j] = c;
        if (k == -1) {
            return p == points[i];
        }
        if (j == -1) {
            return (points[i] - p).dot(points[k] - p) <= 0;
        }
        return out_circle_side(points[i], points[k], points[j], p) >= 0;
    };
    for (int i = 1; i < n; ++i) {
        if (contain(points[i])) continue;
        c = {0, i, -1};
        for (int k = 1; k < i; ++k) {
            if (contain(points[k])) continue;
            c = {i, k, -1};
            for (int j = 0; j < k; ++j) {
                if (contain(points[j])) continue;
                c = {i, k, j};
            }
        }
    }
    auto [i, k, j] = c;
    if (j == -1) {
```

```
52          RE x1 = points[i].x;
53          RE y1 = points[i].y;
54          RE x2 = points[k].x;
55          RE y2 = points[k].y;
56          point<RE> O = {0.5 * (x1 + x2), 0.5 * (y1 + y2)};
57          RE r = sqrtl((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2)) / 2;
58          circle<RE> C(O, r);
59          return {C, rk[i], rk[k], -1};
60      }
61      circle<RE> C = out_circle<RE>(points[i], points[k], points[j]);
62      return {C, rk[i], rk[k], rk[j]};
63  }
```

graph

2_sat.hpp

```cpp
struct TwoSat {  // MeIoNの2-sat
private:
    int n, tot, cnt;
    vector<vector<int>> v;
    vector<bool> ans, vis;
    vector<int> dfn, low, id, s;
    void add(int x, int y) {
        v[x].emplace_back(y);
        v[y ^ 1].emplace_back(x ^ 1);
    }
    void tarjan(int n) {
        dfn[n] = low[n] = ++tot;
        vis[n] = 1;
        s.emplace_back(n);
        for (const int i : v[n]) {
            if (not dfn[i]) {
                tarjan(i);
                chmin(low[n], low[i]);
            } else if (vis[i]) {
                chmin(low[n], dfn[i]);
            }
        }
        if (dfn[n] == low[n]) {
            while (1) {
                int i = s.back();
                s.pop_back();
                id[i] = cnt, vis[i] = 0;
                if (i == n) break;
            } ++cnt;
        }
    }
public:
    TwoSat(int n) : n(n), v(n << 1), ans(n), dfn(n << 1), low(n << 1), vis(n << 1), id(n << 1) {}
    void ban(int x, int y, int val_X, int val_Y) {
        val_X ^= 1;
        add(x << 1 | val_X, y << 1 | val_Y);
    }
    void either(int x, int y, int val_X, int val_Y) {
        ban(x, y, not val_X, not val_Y);
    }
    void either(int x, int y) { ban(x, y, 0, 0); }
    void to(int x, int y) { ban(x, y, 1, 0); }
    void both(int x, int y) {
        ban(x, y, 0, 0);
        ban(x, y, 0, 1);
        ban(x, y, 1, 0);
    }
    bool solve() {
        s.clear();
        std::fill(dfn.begin(), dfn.end(), 0);
        std::fill(low.begin(), low.end(), 0);
        std::fill(vis.begin(), vis.end(), false);
        tot = 0, cnt = 0;
        for (int i = 0; i < n << 1; ++i) if (not dfn[i]) tarjan(i);
        for (int i = 0; i < n; ++i) {
            if (id[i << 1] == id[i << 1 | 1]) return false;
            if (id[i << 1] < id[i << 1 | 1]) {
```

```
                ans[i] = 1;
            }
        }
        return true;
    }
    vector<bool> answer() { return ans; }
};
```

dijkstra.hpp

```
template <typename T, typename VAL>
pair<vector<T>, vector<int>> dijkstra(const vector<vector<pair<int, VAL>>> &v, int s) {
    const int n = v.size();
    vector<T> dis(n, inf<T>);
    vector<int> fa(n, -1);

    using P = pair<T, int>;
    priority_queue<P, vector<P>, greater<P>> q;

    dis[s] = 0;
    q.emplace(0, s);
    while (not q.empty()) {
        auto [dv, n] = q.top();
        q.pop();
        if (dv > dis[n]) continue;
        for (auto [i, w] : v[n]) {
            if (chmin(dis[i], dis[n] + w)) {
                fa[i] = n;
                q.emplace(dis[i], i);
            }
        }
    }
    return {dis, fa};
}
```

三元环计数

```
NAME MeIoN_is_UMP45() {
    int n, m;
    std::cin >> n >> m;
    vector<vector<int>> v(n);
    vector<int> d(n);
    vector<pair<int, int>> e;
    for (int i = 0, l, r; i < m; ++i) {
        std::cin >> l >> r, --l, --r;
        ++d[l], ++d[r];
        e.emplace_back(l, r);
    }
    for (const auto &[l, r] : e) {
        if (d[l] < d[r]) {
            v[l].emplace_back(r);
        } else if (d[l] > d[r]) {
            v[r].emplace_back(l);
        } else {
            v[std::min(l, r)].emplace_back(std::max(l, r));
        }
    }
    ll ans = 0;
    vector<bool> tag(n);
    for (int i = 0; i < n; ++i) {
        for (auto k : v[i]) {
```

```
            tag[k] = true;
        }
        for (auto k : v[i]) {
            for (auto j : v[k]) {
                if (tag[j]) {
                    ++ans;
                }
            }
        }
        for (auto k : v[i]) {
            tag[k] = false;
        }
    }
    std::cout << ans << '\n';
}
```

最大团

```
ll n, ans, anss;

namespace max_t_t{
    vector<vector<int>> v;
    vector<int> cnt, vis, res;
    int dfs(int x,int now){
        for (int i = x + 1, iE = n; i <= iE; ++i) {
            if (cnt[i] + now <= ans) return 0;
            if (not v[x][i]) continue;
            int k;
            for (k = 1; k < now;k++){
                if (not v[i][vis[k]]) break;
            }
            if (k == now){
                vis[now] = i;
                if (dfs(i, now + 1))
                    return 1;
            }
        }
        if (now > ans + 1){
            ans = now - 1;
            for (int i = 1; i < ans; ++i) {
                res[i] = vis[i];
            }
            return 1;
        }
        return 0;
    }
    void work(){
        ans = -1;
        for (int i = n; i; i--){
            vis[1] = i;
            dfs(i, 2);
            cnt[i] = ans;
        }
    }
    void build(int n){
        v.assign(n + 1, vector<int>(n + 1, 0));
        cnt = vector<int>(n + 1, 0);
        vis = res = cnt;
        for (int i = 1, iE = n - 1; i <= iE; ++i) {
            int x, y;
            std::cin >> x >> y;
            v[x][y] = v[y][x] = 1;
```

```
        }
    }
}using namespace max_t_t;
int main(){
    int n;
    build(n);
    work();
    std::cout << ans << '\n';
    for (int i = 1; i < ans; ++i) {
        std::cout << res[i] << " ";
    }
    return 0;
}
```

math

exgcd.hpp

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

mat.hpp

```cpp
template <class mint, ull n>
struct MT : array<array<mint, n>, n> {
    MT(int x = 0, int y = 0) {
        for (int i = 0; i < n; ++i) {
            for (int k = 0; k < n; ++k) {
                (*this)[i][k] = y;
            }
        }
        for (int i = 0; i < n; ++i) {
            (*this)[i][i] = x;
        }
    }
    template <typename T, ull N>
    MT(array<array<T, N>, N> &base) {
        assert(N <= n);
        for (int i = 0; i < N; ++i) {
            for (int k = 0; k < N; ++k) {
                (*this)[i][k] = base[i][k];
            }
        }
    }
    template <typename T> MT(vector<vector<T>>& base) {
        assert(base.size() <= n and base[0].size() <= n);
        for (int i = 0; i < base.size(); ++i) {
            for (int k = 0; k < base[0].size(); ++k) {
                (*this)[i][k] = base[i][k];
            }
        }
    }
    MT& operator*=(const MT& p) {
        MT res;
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                for (int k = 0; k < n; ++k) {
                    res[i][j] += (*this)[i][k] * p[k][j];
                }
            }
        }
        return *this = res;
    }
    MT operator*(const MT& p) {
        return MT(*this) *= p;
    }
    MT ksm(int k, bool ok = false) {
        MT res(1);
```

```
        for (; k; k >>= 1) {
            if (k & 1) {
                res *= (*this);
            }
            (*this) *= (*this);
        }
        if (ok) {
            (*this) = res;
        }
        return res;
    }
    MT ksm(ll k, bool ok = false) {
        MT res(1);
        for (; k; k >>= 1) {
            if (k & 1) {
                res *= (*this);
            }
            (*this) *= (*this);
        }
        if (ok) {
            (*this) = res;
        }
        return res;
    }
};
```

prims_set.hpp

```
struct m64 {
    using i64 = long long;
    using u64 = unsigned long long;
    inline static u64 m, r, n2; // r * m = -1 (mod 1<<64), n2 = 1<<128 (mod m)
    static void set_mod(u64 m) {
        assert(m < (1ull << 62));
        assert((m & 1) == 1);
        m64::m = m;
        n2 = -u128(m) % m;
        r = m;
        for (ll _ = 0; _ < ll(5); ++_) r *= 2 - m * r;
        r = -r;
        assert(r * m == -1ull);
    }
    static u64 reduce(u128 b) { return (b + u128(u64(b) * r) * m) >> 64; }
    u64 x;
    m64() : x(0) {}
    m64(u64 x) : x(reduce(u128(x) * n2)){};
    u64 val() const { u64 y = reduce(x); return y >= m ? y - m : y; }
    m64 &operator+=(m64 y) { x += y.x - (m << 1); x = (i64(x) < 0 ? x + (m << 1) : x); return *this; }
    m64 &operator-=(m64 y) { x -= y.x; x = (i64(x) < 0 ? x + (m << 1) : x); return *this; }
    m64 &operator*=(m64 y) { x = reduce(u128(x) * y.x); return *this; }
    m64 operator+(m64 y) const { return m64(*this) += y; }
    m64 operator-(m64 y) const { return m64(*this) -= y; }
    m64 operator*(m64 y) const { return m64(*this) *= y; }
    bool operator==(m64 y) const { return (x >= m ? x - m : x) == (y.x >= m ? y.x - m : y.x); }
    bool operator!=(m64 y) const { return not operator==(y); }
    m64 pow(u64 n) const { m64 y = 1, z = *this; for (; n; n >>= 1, z *= z) if (n & 1) y *= z; return y; }
};

bool primetest(const uint64_t x) {
    using u64 = uint64_t;
    if (x == 2 or x == 3 or x == 5 or x == 7) return true;
    if (x % 2 == 0 or x % 3 == 0 or x % 5 == 0 or x % 7 == 0) return false;
```

```
    if (x < 121) return x > 1;
    const u64 d = (x - 1) >> __builtin_ctzll(x - 1);
    m64::set_mod(x);
    const m64 one(1), minus_one(x - 1);
    auto ok = [&](u64 a) {
        auto y = m64(a).pow(d);
        u64 t = d;
        while (y != one and y != minus_one and t != x - 1) y *= y, t <<= 1;
        if (y != minus_one and t % 2 == 0) return false;
        return true;
    };
    if (x < (1ull << 32)) {
        for (u64 a: {2, 7, 61}) if (not ok(a)) return false;
    } else {
        for (u64 a: {2, 325, 9375, 28178, 450775, 9780504, 1795265022}) {
            if (x <= a) return true; if (not ok(a)) return false;
        }
    }
    return true;
}
ll rho(ll n, ll c) {
    m64::set_mod(n);
    assert(n > 1);
    const m64 cc(c);
    auto f = [&](m64 x) { return x * x + cc; };
    m64 x = 1, y = 2, z = 1, q = 1;
    ll g = 1;
    const ll m = 1LL << (std::__lg(n) / 5); // ?
    for (ll r = 1; g == 1; r <<= 1) {
        x = y;
        for (ll _ = 0; _ < ll(r); ++_) y = f(y);
        for (ll k = 0; k < r and g == 1; k += m) {
            z = y;
            for (ll _ = 0; _ < ll(std::min(m, r - k)); ++_) y = f(y), q *= x - y;
                g = std::gcd(q.val(), n);
        }
    }
    if (g == n) do {
        z = f(z);
        g = std::gcd((x - z).val(), n);
    } while (g == 1);
    return g;
}
std::mt19937_64 rng_mt{std::random_device{}()};
ll rnd(ll n) { return std::uniform_int_distribution<ll>(0, n - 1)(rng_mt); }
ll find_prime_factor(ll n) {
    assert(n > 1);
    if (primetest(n)) return n;
    for (ll _ = 0; _ < 100ll; ++_) {
        ll m = rho(n, rnd(n));
        if (primetest(m)) return m;
        n = m;
    }
    std::cerr << "failed" << std::endl;
    assert(false);
    return -1;
}

//分解因数
vector<pair<ll, int>> factor(ll n) {
    assert(n >= 1);
    vector<pair<ll, int>> pf;
```

```
    for (int p = 2; p < 100; ++p) {
        if (p * p > n) break;
        if (n % p == 0) {
        int e = 0;
        do { n /= p, e += 1; } while (n % p == 0);
            pf.emplace_back(p, e);
        }
    }
    while (n > 1) {
        ll p = find_prime_factor(n);
        ll e = 0;
        do { n /= p, e += 1; } while (n % p == 0);
        pf.emplace_back(p, e);
    }
    std::ranges::sort(pf);
    return pf;
}
// 通过质因子分解因数
vector<pair<ll, int>> factor_by_lpf(ll n, vector<int>& lpf) {
    vector<pair<ll, int>> res;
    while (n > 1) {
        int p = lpf[n];
        int e = 0;
        while (n % p == 0) {
            n /= p;
            ++e;
        }
        res.emplace_back(p, e);
    }
    return res;
}
```

radix_sort.hpp

```
template <const int N>
void radix_sort(int n, int a[]) {
    static int b[N];
    static int cnt[1 << 8];
    memset(b, 0, sizeof b);
    memset(cnt, 0, sizeof cnt);
    static constexpr int mask = (1 << 8) - 1;
    int *x = a, *y = b;
    for (int i = 0; i < 32; i += 8) {
        for (int j = 0; j != (1 << 8); ++j) cnt[j] = 0;
        for (int j = 0; j != n; ++j) ++cnt[x[j] >> i & mask];
        for (int sum = 0, j = 0; j != (1 << 8); ++j) {
            sum += cnt[j], cnt[j] = sum - cnt[j];
        }
        for (int j = 0; j != n; ++j) y[cnt[x[j] >> i & mask]++] = x[j];
        std::swap(x, y);
    }
}
```

sieve.hpp

```
vector<int> minp, primes;
void sieve(int n) {
    minp.assign(n + 1, 0);
    primes.clear();
    for (int i = 2; i < n + 1; i++) {
        if (minp[i] == 0) {
```

```
            minp[i] = i;
            primes.emplace_back(i);
        }
        for (auto p : primes) {
            if (i * p > n) {
                break;
            }
            minp[i * p] = p;
            if (p == minp[i]) {
                break;
            }
        }
    }
}
```

random

random.hpp

```cpp
#include "../math/mod/modint.hpp"
namespace MeIoN_random_hash {
    std::mt19937 RNG(std::chrono::steady_clock::now().time_since_epoch().count());
    uint rng(uint limit) { return RNG() % limit; }
    int rng(int l, int r) { return l + RNG() % (r - l); }
    std::mt19937_64 RNG_64(std::chrono::steady_clock::now().time_since_epoch().count());
    ull rng_64(ull limit) { return RNG_64() % limit; }
    ll rng_64(ll l, ll r) { return l + RNG_64() % (r - l); }

    using m1 = modint<998244353>;
    using m2 = modint<1000000007>;

    namespace get_prim {

        constexpr ull md = (1ull << 61) - 1;

        static inline constexpr ull modmul(const ull &a, const ull &b) {
            u128 d = u128(a) * b;
            ull ret = (ull(d) & md) + ull(d >> 61);
            return ret >= md ? ret - md : ret;
        }

        static ull modpow(ull a, ull b) {
            ull r = 1;
            for (a %= md; b; a = modmul(a, a), b >>= 1) r = modmul(r, a);
            return r;
        }

        static bool is_primitive(ull x) {
            for (auto &d :
                vector<ull> {2, 3, 5, 7, 11, 13, 31, 41, 61, 151, 331, 1321})
                if (modpow(x, (md - 1) / d) <= 1) return false;
            return true;
        }

        static ull get_basis() {
            static auto rand_time =
                std::chrono::duration_cast<std::chrono::nanoseconds>(
                    std::chrono::high_resolution_clock::now().time_since_epoch())
                    .count();
            static std::mt19937_64 rng(rand_time);
            ull ret;
            while (is_primitive(ret = rng() % (md - 1) + 1) == false);
            return ret;
        }
    } using get_prim::get_basis;

    template <typename T>
    void shuffle(vector<T> &v) {
        int n = v.size();
        for (int i = 0; i < n; ++i) {
            int j = rng(0, i + 1);
            if (i != j) std::swap(v[i], v[j]);
        }
    }

    void random_relabel(int n, vector<pair<int, int>> &v) {
```

```
        shuffle(v);
        vector<int> a(n);
        std::iota(a.begin(), a.end(), 0);
        shuffle(a);
        for (auto &[x, y] : v) {
            x = a[x], y = a[y];
        }
    }

    template <int DIRECTED>
    vector<pair<int, int>> random_graph(int n, bool simple) {
        vector<pair<int, int>> v, cand;
        for (int i = 0; i < n; ++i) {
            for (int k = 0; k < n; ++k) {
                if (simple and i == k) continue;
                if (not DIRECTED and i > k) continue;
                cand.emplace_back(i, k);
            }
        }
        int m = rng(0, (int)cand.size() + 1);
        set<int> se;
        for (int i = 0; i < n; ++m) {
            while (true) {
                int i = rng(0, (int)cand.size());
                if (simple and se.count(i)) continue;
                se.emplace(i);
                auto [a, b] = cand[i];
                v.emplace_back(a, b);
                break;
            }
        }
        random_relabel(n, v);
        return v;
    }

    template <typename T>
    ull hash_pair(const pair<T, T> &X) {
        static ll hash_base = RNG_64();
        if (hash_base == 0) hash_base = RNG_64();
        return hash_base * X.first + X.second;
    }

    template <typename T>
    pair<uint, uint> hash_vector(const vector<T> &v) {
        static vector<pair<m1, m2>> hash_base;
        int n = v.size();
        while (hash_base.size() < n + 1) {
            hash_base.emplace_back(rng(m1::get_mod()), rng(m2::get_mod()));
        }
        m1 h1;
        m2 h2;
        for (int i = 0; i < n; ++i) {
            h1 += hash_base[i].first * m1(v[i]);
            h2 += hash_base[i].second * m2(v[i]);
        }
        h1 += hash_base[n].first, h2 += hash_base[n].second;
        return pair(h1.val, h2.val);
    }

    template <typename T, int K>
    pair<uint, uint> hash_array(const array<T, K> &v) {
        static array<pair<m1, m2>, K> hash_base;
```

```
        if (hash_base[0] == pair(m1(0), m2(0))) {
            for (int i = 0; i < K; ++i) {
                hash_base[i] = {rng(m1::get_mod()), rng(m2::get_mod())};
            }
        }
        m1 h1;
        m2 h2;
        for (int i = 0; i < K; ++i) {
            h1 += hash_base[i].first * m1(v[i]);
            h2 += hash_base[i].second * m2(v[i]);
        }
        return pair(h1.val, h2.val);
    }

    // https://uoj.ac/problem/763
    struct rooted_tree_hash {
        vector<vector<int>> v;
        int n;
        vector<ull> hash;
        vector<int> dis;

        static vector<ull> &xs() {
            static vector<ull> _xs;
            return _xs;
        }

        rooted_tree_hash(const vector<vector<int>> &_v, int root = 0)
            : v(_v), n(_v.size()) {
            hash.resize(n);
            dis.resize(n);
            while ((int)xs().size() <= n) xs().emplace_back(get_basis());
            dfs(root, -1);
        }

       private:
        int dfs(int n, int fa) {
            int dp = 0;
            for (const int &i : v[n]) {
                if (i == fa) continue;
                chmax(dp, dfs(i, n) + 1);
            }
            ull x = xs()[dp], h = 1;
            for (const int &i : v[n]) {
                if (i == fa) continue;
                h = get_prim::modmul(h, (x + hash[i]) % get_prim::md);
            }
            hash[n] = h;
            return dis[n] = dp;
        }
    };
}
```

string

SA.hpp

```cpp
struct MeIoN_SA {
    std::vector<int> p, rank;
    MeIoN_SA(const std::vector<int> &s) : p(s.size()), rank(s.size()) {
        const int n = s.size();
        int k = 0;
        std::iota(p.begin(), p.end(), 0);
        std::ranges::sort(p, {}, [&](int i) { return s[i]; });
        for (int i = 0; i < n; ++i) {
            rank[p[i]] = i and s[p[i]] == s[p[i - 1]] ? rank[p[i - 1]] : k++;
        }
        std::vector<int> q, count, new_rank(n);
        for (int m = 1; m < n; m <<= 1) {
            q.resize(m);
            std::iota(q.begin(), q.end(), n - m);
            for (int i : p) {
                if (i >= m) {
                    q.push_back(i - m);
                }
            }
            count.assign(k, 0);
            for (int i : rank) {
                ++count[i];
            }
            std::partial_sum(count.begin(), count.end(), count.begin());
            for (int i = n - 1; ~i; --i) {
                p[--count[rank[q[i]]]] = q[i];
            }
            auto cmp = [&] (int i, int k) {
                int rk_i = i + m < n ? rank[i + m] : -1;
                int rk_k = k + m < n ? rank[k + m] : -1;
                return rank[i] == rank[k] and rk_i == rk_k;
            };
            k = 0;
            for (int i = 0; i < n; ++i) {
                new_rank[p[i]] =
                    i and cmp(p[i], p[i - 1]) ? new_rank[p[i - 1]] : k++;
            }
            rank.swap(new_rank);
        }
    }
};
```

SAM.hpp

```cpp
namespace MeIoN_SAM {
    static constexpr int ALPHABET = 26;
    struct Node : std::array<int, ALPHABET> {
        int link, len;
        Node() : link(-1), len(0) { fill(-1); }
    };
    struct SAM : std::vector<Node> {
        SAM() : std::vector<Node> (1) {};
        SAM(const int n) : std::vector<Node> (1) { reserve(n); };
        int ext(int p, int c) {
            int pla = size();
            emplace_back();
            back().len = at(p).len + 1;
```

```
            while (~p and at(p)[c] == -1) {
                at(p)[c] = pla;
                p = at(p).link;
            }
            if (~p) {
                int q = at(p)[c];
                if (at(p).len + 1 == at(q).len) {
                    back().link = q;
                } else {
                    int cp = size();
                    push_back(at(q));
                    back().len = at(p).len + 1;
                    while (~p and at(p)[c] == q) {
                        at(p)[c] = cp;
                        p = at(p).link;
                    }
                    at(q).link = at(pla).link = cp;
                }
            } else {
                back().link = 0;
            }
            return pla;
        }
        std::tuple<vector<int>, vector<vector<int>>> build(const string &s) {
            const int n = s.length();
            vector<int> sz(n << 1);
            for (int pla = 0; const char c : s) {
                pla = ext(pla, c - 'a');
                sz[pla] = 1;
            }
            vector<vector<int>> v(n << 1);
            for (int i = 1; i < size(); ++i) {
                v[at(i).link].emplace_back(i);
            }
            auto dfs = [&](auto &&se, int n) -> void {
                for (int i : v[n]) {
                    se(se, i);
                    sz[n] += sz[i];
                }
            };
            dfs(dfs, 0);
            return {sz, v};
        }
    };
} using MeIoN_SAM::SAM;
```

acam.hpp

```
struct MeIoN_ACAM {
    static constexpr int ALPHABET = 26;
    struct Node {
        int len, fail;
        std::array<int, ALPHABET> next;
        Node() : len { 0 } , fail { 0 } , next {} {}
    };
    std::vector<Node> t;
    MeIoN_ACAM() {
        init();
    }
    void init() {
        t.assign(2, Node());
        t[0].next.fill(1);
```

```
        t[0].len = -1;
    }
    int newNode() {
        t.emplace_back();
        return t.size() - 1;
    }
    int add(const std::string& a) {
        int p = 1;
        for (auto c : a) {
            int x = c - 'a';
            if (t[p].next[x] == 0) {
                t[p].next[x] = newNode();
                t[t[p].next[x]].len = t[p].len + 1;
            }
            p = t[p].next[x];
        }
        return p;
    }
    void work() {
        std::queue<int> q;
        q.push(1);
        while (!q.empty()) {
            int x = q.front();
            q.pop();

            for (int i = 0; i < ALPHABET; i++) {
                if (t[x].next[i] == 0) {
                    t[x].next[i] = t[t[x].fail].next[i];
                } else {
                    t[t[x].next[i]].fail = t[t[x].fail].next[i];
                    q.push(t[x].next[i]);
                }
            }
        }
    }
    int next(int p, int x) { return t[p].next[x]; }
    int fail(int p)        { return t[p].fail; }
    int len(int p)         { return t[p].len; }
    int size()             { return t.size(); }
};
using AC = MeIoN_ACAM;
```

hash.hpp

```
namespace getmod {
    bool guidingstar_ckpr(int n) {
        if (n < 1) return false;
        for (int i = 2, ed = n; i * i <= ed; ++i) {
            if (n % i == 0) return false;
        }
        return true;
    }
    int guidingstar_find_pr(int n) {
        while (not guidingstar_ckpr(n)) ++n;
        return n;
    }
    const int m1 = guidingstar_find_pr(rng() % 900000000 + 100000000),
              m2 = guidingstar_find_pr(rng() % 900000000 + 100000000);
    constexpr int M1 = 1000000123, M2 = 1000000181;
}
struct rolling_HASH {
    int n;
```

```
    vector<pair<int, int>> h, p;
    rolling_HASH(const string &s = "") : n(s.length()), h(n + 1), p(n + 1) {
        for (int i = 0; i < n; ++i) {
            h[i + 1].first = (131ll * h[i].first + s[i] - '0') % getmod::m1;
            h[i + 1].second = (131ll * h[i].second + s[i] - '0') % getmod::m2;
        }
        p[0] = {1, 1};
        for (int i = 0; i < n; ++i) {
            p[i + 1].first = 131ll * p[i].first % getmod::m1;
            p[i + 1].second = 131ll * p[i].second % getmod::m2;
        }
    }
    pair<ll, ll> get(int l, int r) const {
        return {
            (h[r].first + 1ll * (getmod::m1 - h[l].first) * p[r - l].first) %
                getmod::m1,
            (h[r].second + 1ll * (getmod::m2 - h[l].second) * p[r - l].second) %
                getmod::m2};
    }
};
struct HASH {
    int n;
    vector<pair<int, int>> h, p;
    HASH(const string &s = "") : n(s.length()), h(n + 1), p(n + 1) {
        for (int i = 0; i < n; ++i) {
            h[i + 1].first = (131ll * h[i].first + s[i] - '0') % getmod::M1;
            h[i + 1].second = (131ll * h[i].second + s[i] - '0') % getmod::M2;
        }
        p[0] = {1, 1};
        for (int i = 0; i < n; ++i) {
            p[i + 1].first = 131ll * p[i].first % getmod::M1;
            p[i + 1].second = 131ll * p[i].second % getmod::M2;
        }
    }
    pair<ll, ll> get(int l, int r) const {
        return {
            (h[r].first + 1ll * (getmod::M1 - h[l].first) * p[r - l].first) %
                getmod::M1,
            (h[r].second + 1ll * (getmod::M2 - h[l].second) * p[r - l].second) %
                getmod::M2};
    }
};
template<typename HASH>
int get_lcp(const HASH &h1, int l1, int r1, const HASH &h2, int l2, int r2) {
    int sz = std::min(r1 - l1, r2 - l2);
    int l = 0, r = sz + 1;
    while (r - l > 1) {
        int m = l + r >> 1;
        if (h1.get(l1, l1 + m) == h2.get(l2, l2 + m)) {
            l = m;
        } else {
            r = m;
        }
    }
    return l;
};
template <typename HASH>
bool hash_same(const HASH &h1, int l1, const HASH &h2, int l2, int sz) {
    return(l1 + sz <= h1.n and l2 + sz <= h2.n) and
        h1.get(l1, l1 + sz) == h2.get(l2, l2 + sz);
}
```

manache.hpp

```cpp
namespace MeIoN_namache{
    ll n;
    constexpr int working_sz = 1145141;
    int p[working_sz];
    string ss;
    void namache(string &s) {
        s = " " + s;
        ss = "";
        ss += '&';
        for (int i = 1; i <= n; i++) {
            ss += '#';
            ss += s[i];
        }
        ss += '#';
        ss += '*';
        for (int i = 1; i < ss.size(); i++) p[i] = 1;
        for (int i = 1, l = 1, r = 0; i + 1 < ss.size(); i++) {
            if (i <= r) p[i] = std::min(r - i + 1, p[l + r - i]);
            while (ss[i - p[i]] == ss[i + p[i]]) p[i] ++;
            if (i + p[i] - 1 > r) {
                l = i - p[i] + 1;
                r = i + p[i] - 1;
            }
        }
    }
    // [l, r]
    bool check(int l, int r) {
        int len = r - l + 1;
        l *= 2, r *= 2;
        return p[l + r >> 1] - 1 >= len;
    }
}using namespace MeIoN_namache;
```

trie

```cpp
template <int W>
struct trie {
    struct node {
        array<int, W> ch,
                      nxt;
        int fa;
        int link;
        node() : fa(-1), link(-1) {
            ch.fill(-1);
            nxt.fill(-1);
        }
    };
    int n_node;
    vector<node> nodes;
    vector<int> words;
    vector<int> bfs;

    trie() :n_node(0) {
        new_node();
    }

    node &operator[](int i) { return nodes[i]; }

    template <typename container>
```

```
    int add(container s, int off) {
        int pla = 0;
        for (auto &&c : s) {
            pla = add_single(pla, c, off);
        }
        words.emplace_back(pla);
        return pla;
    }

    int add_single(int pla, int c, int off) {
        c -= off;
        assert(-1 < c and c < W);
        if (nodes[pla].ch[c] != -1) return nodes[pla].ch[c];
        nodes[pla].ch[c] = new_node();
        nodes.back().fa = pla;
        return nodes[pla].ch[c];
    }

    void calc_suffix_link() {
        bfs.resize(n_node);
        int p = 0, q = 0;
        bfs[q++] = 0;
        nodes[0].nxt.fill(0);
        while (p < q) {
            int v = bfs[p++];
            if (v) nodes[v].nxt = nodes[nodes[v].link].nxt;
            for (int i = 0; i < W; ++i) {
                int w = nodes[v].ch[i];
                if (w == -1) continue;
                nodes[w].link = nodes[v].nxt[i];
                nodes[v].nxt[i] = w;
                bfs[q++] = w;
            }
        }
    }

    vector<int> calc_count() {
        vector<int> count(n_node);
        for (int i : words) {
            ++count[i];
        }
        for (int i : bfs) {
            if (i) {
                count[i] += count[nodes[i].link];
            }
        }
        return count;
    }

  private:
    int new_node() {
        node c;
        nodes.emplace_back(c);
        return n_node++;
    }
};
```

tree

LCA.hpp

```cpp
template <const int N> struct LCA {
public:
    LCA (const vector<vector<int>> &v, int rt) :
    sz(v.size()), root(rt), up(sz), dis(sz), lg(0) {
        for (auto &i : up) i.fill(0);
        while ((1 << lg) <= sz) lg++;
        assert(lg <= N);
        auto dfs = [&](auto &&se, int n, int fa, int dp) -> void {
            dis[n] = dp;
            up[n][0] = fa;
            for (int i = 1; i <= lg - 1; i++) up[n][i] = up[up[n][i - 1]][i - 1];
            for (const auto &x : v[n]) {
                if (x == fa) continue;
                se(se, x, n, dp + 1);
            }
        };
        dfs(dfs, rt, rt, 0);
    }
    int &operator[](const int &x) { return up[x]; }
    int jump(int x, int tp) {
        chmin(tp, dis[x] + 1);
        for (int i = 0; i < lg; i++) {
            if (tp >> i & 1) {
                x = up[x][i];
            }
        }
        return up[x][0];
    }
    int lca(int x,int y){
        if (dis[x] < dis[y])
            std::swap(x, y);
        int z = dis[x] - dis[y];
        for (int i = 0; i < lg; i++) {
            if (z >> i & 1) {
                x = up[x][i];
            }
        }
        if (x == y) return x;
        for (int i = lg; i--; ) {
            int X = up[x][i], Y = up[y][i];
            if (X != Y) x = X, y = Y;
        }
        return up[x][0];
    }
    int dist(int x,int y){
        return dis[x] + dis[y] - 2 * dis[lca(x, y)];
    }
private:
    int root, sz, lg;
    std::vector<std::array<int, N>> up;
    std::vector<int> dis;
};
```

LTT.hpp

```cpp
vector<int> get_fa(const vector<vector<int>> &v, int s) {
    int n = v.size();
```

```
    vector<int> pos(n, -1), p, label(n), dom(n), sdom(n), dsu(n), par(n);
    vector<vector<int>> rg(n), bucket(n);
    auto dfs = [&] (auto &&se, int n)->void {
        int t = p.size();
        p.emplace_back(n);
        label[t] = sdom[t] = dsu[t] = pos[n] = t;
        for (const int i : v[n]) {
            if (pos[i] == -1) {
                se(se, i);
                par[pos[i]] = t;
            }
            rg[pos[i]].emplace_back(t);
        }
    };
    auto find = [&] (auto &&se, int n, int x) {
        if (n == dsu[n]) return x ? -1 : n;
        int v = se(se, dsu[n], x + 1);
        if (v < 0) return n;
        if (sdom[label[dsu[n]]] < sdom[label[n]]) {
            label[n] = label[dsu[n]];
        }
        dsu[n] = v;
        return x ? v : label[n];
    };
    dfs(dfs, s);
    std::iota(dom.begin(), dom.end(), 0);
    for (int i = (int)p.size() - 1; ~i; --i) {
        for (int k : rg[i]) {
            chmin(sdom[i], sdom[find(find, k, 0)]);
        }
        if (i) {
            bucket[sdom[i]].emplace_back(i);
        }
        for (int k : bucket[i]) {
            int j = find(find, k, 0);
            dom[k] = sdom[j] == sdom[k] ? sdom[j] : j;
        }
        if (i > 1) {
            dsu[i] = par[i];
        }
    }
    for (int i = 1; i < (int)p.size(); ++i) {
        if (dom[i] != sdom[i]) {
            dom[i] = dom[dom[i]];
        }
    }
    vector<int> res(n, -1);
    res[s] = s;
    for (int i = 1; i < (int)p.size(); ++i) {
        res[p[i]] = p[dom[i]];
    }
    return res;
}
```

centroid.hpp

```
vector<int> centroid(const vector<vector<int>> &v) {
    const int n = (int)v.size();
    vector<pair<int, int>> st;
    vector<int> sz(n), ff(n);

    st.reserve(n);
```

```
    st.emplace_back(0, -1);
    while (not st.empty()) {
        const auto [n, fa] = st.back();
        if (sz[n] == 0) {
            sz[n] = 1;
            for (const int i : v[n]) {
                if (i == fa) continue;
                st.emplace_back(i, n);
            }
        } else {
            for (const int i : v[n]) {
                if (i == fa) continue;
                sz[n] += sz[i];
            }
            ff[n] = fa;
            st.pop_back();
        }
    }

    vector<int> ret;
    ret.reserve(8);
    int size = n;
    for (int i = 0; i < n; ++i) {
        int val = n - sz[i];
        for (const int x : v[i]) {
            if (x == ff[i]) continue;
            chmax(val, sz[i]);
        }
        if (chmin(size, val)) {
            ret.clear();
        }
        if (val == size) {
            ret.emplace_back(i);
        }
    }
    return ret;
}
```

unrooted_tree_hash.hpp

```

#include "centroid.hpp"
#include "../random/random.hpp"

using MeIoN_random_hash::rooted_tree_hash;

vector<ull> unrooted_tree_hash(const vector<vector<int>> &v) {
    vector root = centroid(v);
    if (root.size() == 1) root.emplace_back(root[0]);
    vector<ull> res;
    for (const int x : root) {
        res.emplace_back(rooted_tree_hash(v, x).hash[x]);
    }
    sort(res);
    return res;
}
```

最小斯坦纳树

```
NAME MeIoN_is_UMP45() {
    std::cin >> n >> m >> k;
```

```
    vector<vector<pair<int, int>>> v(n + 1);
    for (int i = 0, l, r, w; i < m; ++i) {
        std::cin >> l >> r >> w;
        v[l].emplace_back(r, w);
        v[r].emplace_back(l, w);
    }
    A<A<int, 101>, 1 << 11> dp;
    for (auto &v : dp) v.fill(__INT_MAX__);
    vector<int> S(k);
    for (int c = 0; auto & i : S) {
        std::cin >> i;
        dp[1 << c++][i] = 0;
    }
    rpq<pair<int, int>> q;
    auto dij = [&](int BE) {
        while (not q.empty()) {
            int n = q.top().second;
            q.pop();
            for (const auto & [ i, w ] : v[n]) {
                if (dp[BE][i] > dp[BE][n] + w) {
                    dp[BE][i] = dp[BE][n] + w;
                    q.emplace(dp[BE][i], i);
                }
            }
            while (not q.empty() and q.top().first != dp[BE][q.top().second])
                q.pop();
        }
    };
    for (int i = 1; i < 1 << k; ++i) {
        for (int k = 1; k <= n; ++k) {
            for (int j = i & (i - 1); j; j = i & (j - 1)) {
                dp[i][k] = std::min(dp[i][k], dp[j][k] + dp[i ^ j][k]);
            }
            if (dp[i][k] < __INT_MAX__) q.emplace(dp[i][k], k);
        }
        dij(i);
    }
    int ans = __INT_MAX__;
    for (int i = 1; i <= n; ++i) {
        ans = std::min(ans, dp[(1 << k) - 1][i]);
    }
    std::cout << ans << '\n';
}
```

a_monoid

max_add.hpp

```cpp
#include "../monoid/add.hpp"
#include "../monoid/max.hpp"

template <typename E>
struct a_monoid_max_add {
    using Monoid_X = monoid_max<E>;
    using Monoid_A = monoid_add<E>;
    using X = typename Monoid_X::value_type;
    using A = typename Monoid_A::value_type;
    static constexpr X act(const X &x, const A &a, const ll &size) {
        if (x == inf<E>) return x;
        return x + a;
    }
};
```

min_add.hpp

```cpp
#include "../monoid/add.hpp"
#include "../monoid/min.hpp"

template <typename E>
struct a_monoid_min_add {
    using Monoid_X = monoid_min<E>;
    using Monoid_A = monoid_add<E>;
    using X = typename Monoid_X::value_type;
    using A = typename Monoid_A::value_type;
    static constexpr X act(const X &x, const A &a, const ll &size) {
        if (x == inf<E>) return x;
        return x + a;
    }
};
```

minidx_add.hpp

```cpp
#include "../monoid/add.hpp"
#include "../monoid/min_idx.hpp"

template <typename E, bool tie_is_left = true>
struct a_monoid_min_idx_add {
    using Monoid_X = monoid_min_idx<E, tie_is_left>;
    using Monoid_A = monoid_add<E>;
    using X = typename Monoid_X::value_type;
    using A = typename Monoid_A::value_type;
    static constexpr X act(const X &x, const A &a, const ll &size) {
        if (x.first == inf<E>) return x;
        return {x.first + a, x.second};
    }
};
```

sum_add.hpp

```cpp
#include "../monoid/add.hpp"

template <typename E>
struct a_monoid_sum_add {
    using Monoid_X = monoid_add<E>;
    using Monoid_A = monoid_add<E>;
```

```
    using X = typename Monoid_X::value_type;
    using A = typename Monoid_A::value_type;
    static constexpr X act(const X &x, const A &a, const ll &size) {
        return x + a * E(size);
    }
};
```

monoid

add.hpp

```

template <typename E>
struct monoid_add {
    using X = E;
    using value_type = X;
    static constexpr X op(const X &x, const X &y) noexcept { return x + y; }
    static constexpr X inverse(const X &x) noexcept { return -x; }
    static constexpr X power(const X &x, ll n) noexcept { return X(n) * x; }
    static constexpr X unit() { return X(0); }
    static constexpr bool commute = true;
};
```

add_array.hpp

```

template <typename E, int K>
struct monoid_add_array {
    using value_type = array<E, K>;
    using X = value_type;
    static X op(X x, X y) {
        for (int i = 0; i < K; ++i) x[i] += y[i];
        return x;
    }
    static constexpr X unit() { return X {}; }
    static constexpr X inverse(X x) {
        for (auto& v : x) v = -v;
        return x;
    }
    static constexpr X power(X x, ll n) {
        for (auto& v : x) v *= E(n);
        return x;
    }
    static constexpr bool commute = 1;
};
```

add_pair.hpp

```

template <typename E>
struct monoid_add_pair {
    using value_type = pair<E, E>;
    using X = value_type;
    static constexpr X op(const X &x, const X &y) {
        return {x.fi + y.fi, x.se + y.se};
    }
    static constexpr X inverse(const X &x) { return {-x.fi, -x.se}; }
    static constexpr X unit() { return {0, 0}; }
    static constexpr bool commute = true;
};
```

gcd.hpp

```

template <class X>
struct monoid_gcd {
    using value_type = X;
    static constexpr X op(const X & a, const X &b) noexcept { return std::gcd(a, b); }
```

```
    static constexpr X unit() { return 0; }
    static constexpr bool commute = true;
};
```

max.hpp

```

template <class X>
struct monoid_max {
    using value_type = X;
    static constexpr X op(const X & a, const X &b) noexcept { return std::max(a, b); }
    static constexpr X unit() { return -std::numeric_limits<X>::max(); }
    static constexpr bool commute = true;
};
```

max_idx.hpp

```

template <typename T, bool tie_is_left = true>
struct monoid_max_idx {
    using value_type = pair<T, int>;
    using X = value_type;
    static X op(X x, X y) {
        if (x.first > y.first) return x;
        if (x.first < y.first) return y;
        if (x.second > y.second) std::swap(x, y);
        return (tie_is_left ? x : y);
    }
    static constexpr X unit() { return {-INTMAX, -1}; }
    static constexpr bool commute = true;
};
```

min.hpp

```

template <class X>
struct monoid_min {
    using value_type = X;
    static constexpr X op(const X & a, const X &b) noexcept { return std::min(a, b); }
    static constexpr X unit() { return std::numeric_limits<X>::max(); }
    static constexpr bool commute = true;
};
```

min_idx.hpp

```

template <typename T, bool tie_is_left = true>
struct monoid_min_idx {
    using value_type = pair<T, int>;
    using X = value_type;
    static constexpr bool is_small(const X &x, const X &y) {
        if (x.first < y.first) return true;
        if (x.first > y.first) return false;
        return (tie_is_left ? (x.second < y.second) : (x.second >= y.second));
    }
    static X op(X x, X y) { return (is_small(x, y) ? x : y); }
    static constexpr X unit() { return {INTMAX, -1}; }
    static constexpr bool commute = true;
};
```

sum.hpp

```

template <class X>
struct monoid_sum {
    using value_type = X;
    static constexpr X op(const X & a, const X &b) noexcept { return a + b; }
    static constexpr X unit() { return 0; }
    static constexpr bool commute = true;
};
```

xor.hpp

```

template <typename X>
struct monoid_xor {
    using value_type = X;
    static X op(X x, X y) { return x ^ y; };
    static constexpr X inverse(const X &x) noexcept { return x; }
    static constexpr X power(const X &x, ll n) noexcept {
        return (n & 1 ? x : 0);
    }
    static constexpr X unit() { return X(0); };
    static constexpr bool commute = true;
};
```

seg

lazy_seg_base.hpp

```cpp
template <typename a_monoid>
struct lazy_seg {
    using AM = a_monoid;
    using MX = typename AM::Monoid_X;
    using MA = typename AM::Monoid_A;
    using X = typename MX::value_type;
    using A = typename MA::value_type;
    int n, log, size;
    vector<X> dat;
    vector<A> tag;

    lazy_seg() {}
    lazy_seg(int n) { build(n); }
    template <typename F>
    lazy_seg(int n, F f) {
        build(n, f);
    }
    lazy_seg(const vector<X> &v) { build(v); }

    void build(int m) {
        build(m, [](int i) -> X { return MX::unit(); });
    }
    void build(const vector<X> &v) {
        build(v.size(), [&](int i) -> X { return v[i]; });
    }
    template <typename F>
    void build(int m, F f) {
        n = m, log = 1;
        while ((1 << log) < n) ++log;
        size = 1 << log;
        dat.assign(size << 1, MX::unit());
        tag.assign(size, MA::unit());
        for (int i = 0; i < n; ++i) dat[size + i] = f(i);
        for (int i = size - 1; i > 0; --i) update(i);
    }

    void update(int k) { dat[k] = MX::op(dat[2 * k], dat[2 * k + 1]); }
    void set(int p, X x) {
        assert(-1 < p and p < n);
        p += size;
        for (int i = log; i > 0; --i) push(p >> i);
        dat[p] = x;
        for (int i = 1; i < log + 1; ++i) update(p >> i);
    }
    void multiply(int p, const X &x) {
        assert(-1 < p and p < n);
        p += size;
        for (int i = log; i > 0; --i) push(p >> i);
        dat[p] = MX::op(dat[p], x);
        for (int i = 1; i < log + 1; ++i) update(p >> i);
    }

    X get(int p) {
        assert(p > -1 and p < n);
        p += size;
        for (int i = log; i > 0; --i) push(p >> i);
        return dat[p];
```

```
    }

    vector<X> get_all() {
        for (int i = 1; i < size; ++i) push(i);
        return {dat.begin() + size, dat.begin() + size + n};
    }

    X prod(int l, int r) {
        assert(-1 < l and l < r + 1 and r < n + 1);
        if (l == r) return MX::unit();
        l += size, r += size;
        for (int i = log; i > 0; --i) {
            if (((l >> i) << i) != l) push(l >> i);
            if (((r >> i) << i) != r) push((r - 1) >> i);
        }
        X xl = MX::unit(), xr = MX::unit();
        while (l < r) {
            if (l & 1) xl = MX::op(xl, dat[l++]);
            if (r & 1) xr = MX::op(dat[--r], xr);
            l >>= 1, r >>= 1;
        }
        return MX::op(xl, xr);
    }

    X prod_all() { return dat[1]; }

    void apply(int l, int r, A a) {
        assert(-1 < l and l < r + 1 and r < n + 1);
        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) apply_at(l++, a);
            if (r & 1) apply_at(--r, a);
            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 <typename F>
    int max_right(const F check, int l) {
        assert(0 <= l && l <= n);
        assert(check(MX::unit()));
        if (l == n) return n;
        l += size;
        for (int i = log; i >= 1; i--) push(l >> i);
        X sm = MX::unit();
        do {
            while (l % 2 == 0) l >>= 1;
            if (not check(MX::op(sm, dat[l]))) {
                while (l < size) {
                    push(l);
                    l = (2 * l);
                    if (check(MX::op(sm, dat[l]))) {
```

```
                        sm = MX::op(sm, dat[l++]);
                    }
                }
                return l - size;
            }
            sm = MX::op(sm, dat[l++]);
        } while ((l & -l) != l);
        return n;
    }

    template <typename F>
    int min_left(const F check, int r) {
        assert(0 <= r && r <= n);
        assert(check(MX::unit()));
        if (r == 0) return 0;
        r += size;
        for (int i = log; i >= 1; i--) push((r - 1) >> i);
        X sm = MX::unit();
        do {
            r--;
            while (r > 1 && (r % 2)) r >>= 1;
            if (!check(MX::op(dat[r], sm))) {
                while (r < size) {
                    push(r);
                    r = (2 * r + 1);
                    if (check(MX::op(dat[r], sm))) {
                        sm = MX::op(dat[r--], sm);
                    }
                }
                return r + 1 - size;
            }
            sm = MX::op(dat[r], sm);
        } while ((r & -r) != r);
        return 0;
    }

   private:
    void apply_at(int k, A a) {
        int sz = 1 << (log - topbit(k));
        dat[k] = AM::act(dat[k], a, sz);
        if (k < size) tag[k] = MA::op(tag[k], a);
    }
    void push(int k) {
        if (tag[k] == MA::unit()) return;
        apply_at(2 * k, tag[k]), apply_at(2 * k + 1, tag[k]);
        tag[k] = MA::unit();
    }
};
```

seg_base.hpp

```
template <class monoid>
struct Seg {
    using MX = monoid;
    using X = typename MX::value_type;
    using value_type = X;
    vector<X> dat;
    int n, log, sz;
    Seg() {}
    Seg(int n) { build(n); }
    template <typename F>
    Seg(int n, F f) { build(n, f); }
```

```
    Seg(const vector<X> &v) { build(v); }
    void build(int m) { build(m, [](int i) ->X { return MX::unit(); }); }
    void build(const vector<X> &v) { build(int(v.size()), [&](int i) -> X { return v[i]; }); }
    template <typename F>
    void build(int N, F f) {
        n = N;
        while ((1 << log) < n) ++log;
        sz = 1 << log;
        dat.assign(sz << 1, MX::unit());
        for (int i = 0; i < n; ++i) dat[sz + i] = f(i);
        for (int i = sz - 1; i >= 1; --i) update(i);
    }
    X get(int i) { return dat[sz + i]; }
    vector<X> get_all() { return {dat.begin() + sz, dat.begin() + sz + n}; }
    void update(int i) { dat[i] = monoid::op(dat[2 * i], dat[2 * i + 1]); }
    void set(int i, const X &x) {
        dat[i += sz] = x;
        while (i >>= 1) update(i);
    }
    void multiply(int i, const X &x) {
        i += sz;
        dat[i] = monoid::op(dat[i], x);
        while (i >>= 1) update(i);
    }
    X prod(int l, int r) {
        X vl = monoid::unit(), vr = monoid::unit();
        l += sz, r += sz;
        while (l < r) {
            if (l & 1) vl = monoid::op(vl, dat[l++]);
            if (r & 1) vr = monoid::op(dat[--r], vr);
            l >>= 1, r >>= 1;
        }
        return monoid::op(vl, vr);
    }
    X prod_all() { return dat[1]; }
    template <class F>
    int max_right(F check, int l) {
        if (l == n) return n;
        l += sz;
        X sm = monoid::unit();
        do {
            while (l % 2 == 0) l >>= 1;
            if (not check(monoid::op(sm, dat[l]))) {
                while (l < sz) {
                    l = 2 * l;
                    if (check(monoid::op(sm, dat[l]))) { sm = monoid::op(sm, dat[l++]); }
                }
                return l - sz;
            }
            sm = monoid::op(sm, dat[l++]);
        } while ((l & -l) != l);
        return n;
    }
    template <class F>
    int min_left(F check, int r) {
        if (r == 0) return 0;
        r += sz;
        X sm = monoid::unit();
        do {
            --r;
            while (r > 1 and (r % 2)) r >>= 1;
            if (not check(monoid::op(dat[r], sm))) {
```

```
                while (r < sz) {
                    r = 2 * r + 1;
                    if (check(monoid::op(dat[r], sm))) { sm = monoid::op(dat[r--], sm); }
                }
                return r + 1 - sz;
            }
            sm = monoid::op(dat[r], sm);
        } while ((r & -r) != r);
        return 0;
    }
    X xor_prod(int l, int r, int xor_val) {
        static_assert(monoid::commute);
        X x = monoid::unit();
        for (int k = 0; k < log + 1; ++k) {
            if (l >= r) break;
            if (l & 1) { x = monoid::op(x, dat[(sz >> k) + ((l++) ^ xor_val)]); }
            if (r & 1) { x = monoid::op(x, dat[(sz >> k) + ((--r) ^ xor_val)]); }
            l /= 2, r /= r, xor_val /= 2;
        }
        return x;
    }
};
```

mod

modint.hpp

```cpp
struct has_mod_impl {
    template <class T>
    static auto check(T&& x) -> decltype(x.get_mod(), std::true_type {});
    template <class T>
    static auto check(...) -> std::false_type;
};
template <class T>
class has_mod : public decltype(has_mod_impl::check<T>(std::declval<T>())) { };
template <int mod>
struct modint {
    static constexpr bool is_mod_int = true;
    static constexpr unsigned umod = unsigned(mod);
    static_assert(umod < unsigned(1) << 31);
    int val;
    static modint raw(unsigned v) {
        modint x;
        x.val = v;
        return x;
    }
    constexpr modint(const ll val = 0) noexcept : val(val >= 0 ? val % mod : (mod - (-val) % mod) % mod) { }
    bool operator<(const modint& other) const { return val < other.val; }
    modint& operator+=(const modint& p) {
        if ((val += p.val) >= mod)
            val -= mod;
        return* this;
    }
    modint& operator-=(const modint& p) {
        if ((val += mod - p.val) >= mod)
            val -= mod;
        return* this;
    }
    modint& operator*=(const modint& p) {
        val = (int)(1LL * val * p.val % mod);
        return* this;
    }
    modint& operator/=(const modint& p) {
        *this *= p.inv();
        return* this;
    }
    modint operator-() const { return modint::raw(val ? mod - val : unsigned(0)); }
    modint operator+(const modint& p) const { return modint(*this) += p; }
    modint operator-(const modint& p) const { return modint(*this) -= p; }
    modint operator*(const modint& p) const { return modint(*this) *= p; }
    modint operator/(const modint& p) const { return modint(*this) /= p; }
    bool operator==(const modint& p) const { return val == p.val; }
    bool operator!=(const modint& p) const { return val != p.val; }
    friend std::istream& operator>>(std::istream& is, modint& p) {
        ll x;
        is >> x;
        p = x;
        return is;
    }
    friend std::ostream& operator<<(std::ostream& os, modint p) { return os << p.val; }
    modint inv() const {
        int a = val, b = mod, u = 1, v = 0, t;
        while (b > 0)
            t = a / b, std::swap(a -= t * b, b), std::swap(u -= t * v, v);
```

```
        return modint(u);
    }
    modint ksm(ll n) const {
        modint ret(1), mul(val);
        while (n > 0) {
            if (n & 1)
                ret *= mul;
            mul *= mul;
            n >>= 1;
        }
        return ret;
    }
    static constexpr int get_mod() { return mod; }
    static constexpr pair<int, int> ntt_info() {
        if (mod == 120586241) return {20, 74066978};
        if (mod == 167772161) return {25, 17};
        if (mod == 469762049) return {26, 30};
        if (mod == 754974721) return {24, 362};
        if (mod == 880803841) return {23, 211};
        if (mod == 943718401) return {22, 663003469};
        if (mod == 998244353) return {23, 31};
        if (mod == 1004535809) return {21, 836905998};
        if (mod == 1045430273) return {20, 363};
        if (mod == 1051721729) return {20, 330};
        if (mod == 1053818881) return {20, 2789};
        return { -1, -1 };
    }
    static constexpr bool can_ntt() { return ntt_info().first != -1; }
};
ll mod_inv(ll val, ll mod) {
    if (mod == 0) return 0;
    mod = std::abs(mod);
    val %= mod;
    if (val < 0) val += mod;
    ll a = val, b = mod, u = 1, v = 0, t;
    while (b > 0) {
        t = a / b;
        std::swap(a -= t * b, b), std::swap(u -= t * v, v);
    }
    if (u < 0) u += mod;
    return u;
}
constexpr unsigned mod_pow_constexpr(ull a, ull n, unsigned mod) {
    a %= mod;
    ull res = 1;
    for (int _ = 0; _ < 32; ++_) {
        if (n & 1) res = res * a % mod;
        a = a * a % mod, n /= 2;
    }
    return res;
}
unsigned mod_pow(ull a, ull n, unsigned mod) {
    a %= mod;
    ull res = 1;
    for (int _ = 0; _ < 32; ++_) {
        if (n & 1) res = res * a % mod;
        a = a * a % mod, n /= 2;
    }
    return res;
}
ull mod_pow_64(ull a, ull n, ull mod) {
    a %= mod;
```

```
    ull res = 1;
    while (n) {
        if (n & 1) res = u128(res * a) % mod;
        a = u128(a * a) % mod, n >>= 1;
    }
    return res;
}

template <typename T, unsigned p0, unsigned p1, unsigned p2>
T CRT3(ull a0, ull a1, ull a2) {
    static_assert(p0 < p1 && p1 < p2);
    static constexpr ull x0_1 = mod_pow_constexpr(p0, p1 - 2, p1);
    static constexpr ull x01_2 = mod_pow_constexpr(ull(p0) * p1 % p2, p2 - 2, p2);
    ull c = (a1 - a0 + p1) * x0_1 % p1;
    ull a = a0 + c * p0;
    c = (a2 - a % p2 + p2) * x01_2 % p2;
    return T(a) + T(c) * T(p0) * T(p1);
}

template <typename mint>
mint inv(int n) {
    static const int mod = mint::get_mod();
    static vector<mint> dat = {0, 1};
    assert(0 <= n);
    if (n >= mod) n %= mod;
    while (int(dat.size()) <= n) {
        int k = dat.size();
        auto q = (mod + k - 1) / k;
        int r = k * q - mod;
        dat.emplace_back(dat[r] * mint(q));
    }
    return dat[n];
}
template <typename mint>
mint fact(int n) {
    static const int mod = mint::get_mod();
    static vector<mint> dat = {1, 1};
    assert(0 <= n);
    if (n >= mod) return 0;
    while (int(dat.size()) <= n) {
        int k = dat.size();
        dat.emplace_back(dat[k - 1] * mint(k));
    }
    return dat[n];
}
template <typename mint>
mint fact_inv(int n) {
    static const int mod = mint::get_mod();
    static vector<mint> dat = {1, 1};
    assert(-1 <= n && n < mod);
    if (n == -1) return mint(0);
    while (int(dat.size()) <= n) {
        int k = dat.size();
        dat.emplace_back(dat[k - 1] * inv<mint>(k));
    }
    return dat[n];
}
template <typename mint>
mint C(ll n, ll m) {
    if (m < 0 or m > n) return 0ll;
    return fact<mint>(n) * fact_inv<mint>(m) * fact_inv<mint>(n - m);
```

```
181 }
```

others

快速取模

```cpp
inline unsigned long long calc(const unsigned long long &x) {
    return x - (__uint128_t(x) * 9920937979283557439ull >> 93) * 998244353;
}
```

date time

```cpp
struct DateTime {
    static constexpr int month_days[13]
        = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    int year, month, day;
    DateTime(int y, int m, int d) : year(y), month(m), day(d) {}

  // 1年1月1日が 0 となるように変換
    int to_int() {
        int y = (month <= 2 ? year - 1 : year);
        int m = (month <= 2 ? month + 12 : month);
        int d = day;
        return 365 * y + y / 4 - y / 100 + y / 400 + 306 * (m + 1) / 10 + d - 429;
    }

  // to_int() の逆関数
    static DateTime from_int(int x) {
        int y = x * 400 / 146097 + 1;
        int d = x - DateTime(y, 1, 1).to_int();
        int m = 1;
        while (d >= 28) {
            int k = month_days[m] + (m == 2 && is_leap_year(y) ? 1 : 0);
            if (d < k) break;
            ++m;
            d -= k;
        }
        if (m == 13) {
            ++y;
            m = 1;
        }
        ++d;
        return DateTime(y, m, d);
    }

  // 日曜日が 0 として、曜日を [0, 7) で返す
    int weekday() { return (to_int() + 1) % 7; }

    DateTime& operator++() {
        ++day;
        int lim = month_days[month];
        if (is_leap_year(year) && month == 2) lim = 29;
        if (day <= lim) return (*this);
        day = 1;
        ++month;
        if (month == 13) {
            ++year;
            month = 1;
        }
        return (*this);
    }
    DateTime operator++(int) {
        DateTime tmp = *this;
```

```
        ++*this;
        return tmp;
    }

    bool operator==(DateTime const& rhs) const {
        return to_tuple() == rhs.to_tuple();
    }
    bool operator!=(DateTime const& rhs) const {
        return to_tuple() != rhs.to_tuple();
    }
    bool operator<(DateTime const& rhs) const {
        return to_tuple() < rhs.to_tuple();
    }
    bool operator<=(DateTime const& rhs) const {
        return to_tuple() <= rhs.to_tuple();
    }
    bool operator>(DateTime const& rhs) const {
        return to_tuple() > rhs.to_tuple();
    }
    bool operator>=(DateTime const& rhs) const {
        return to_tuple() >= rhs.to_tuple();
    }

  // yyyy[sep]mm[sep]dd
    string to_string(string sep = "-") {
        string y = std::to_string(year);
        string m = std::to_string(month);
        string d = std::to_string(day);
        while (len(y) < 4) y = "0" + y;
        while (len(m) < 2) m = "0" + m;
        while (len(d) < 2) d = "0" + d;
        return y + sep + m + sep + d;
    }

    tuple<int, int, int> to_tuple() const { return {year, month, day}; }

    static bool is_leap_year(int y) {
        if (y % 400 == 0) return true;
        return (y % 4 == 0 && y % 100 != 0);
    }

    static bool is_valid_date(int y, int m, int d) {
        if (!(1 <= m && m <= 12)) return 0;
        int mx = month_days[m];
        if (m == 2 && is_leap_year(y)) ++mx;
        return (1 <= d && d <= mx);
    }
};
```