Team Note of ConForza

Compiled on 2025년 4월 19일

차 례

1 Have you...

1.1 tried...

- **Reading the problem once more?**
- doubting "obvious" things?
- writing obivous things?
- radical greedy approach?
- thinking in reverse direction?
- a greedy algorithm?
- network flow when your greedy algorithms stuck?
- a dynamic programming?
- checking the range of answer?
- random algorithm?
- graph modeling using states?
- inverting state only on odd indexes?
- calculating error bound on a real number usage?

1.2 checked...

- **you have read the statement correctly?**
- typo copying the team note?
- initialization on multiple test case problem?
- additional information from the problem?
- undefined behavior?
- overflow?
- function without return value?
- real number error?
- implicit conversion?
- comparison between signed and unsigned integer?

2 Algorithmic Idea Note

I. Complete Search: Backtracking & Pruning
II. Math
    A. Number Theory
        1. Prime Number
            i) Sieve of Eratosthenes, Prime Factorization
            ii) Fast Prime Verdict; Millar-Rabin
            iii) Fast Prime Factorization; Pollad Rho
            iv) Primitive Root
        2. Extended Euclidean Algorithm; Diophantos Equation
        3. Chinese Remainder Theorem
        4. Harmonic Lemma
        5. Floor Sum (Sum of Rational Arithmetic Sequence)
        6. Several Sieves
    B. Linear Programming
        1. Solve (some) LP with Shortest Path
    C. FFT & Polynomials
        1. FFT : Convolution
            i) High precision FFT with modulo 1e9+7
        2. NTT : Number Theoretic Tranform
        3. Quotient Ring (Formal Power Series)
            i) Multiplication
            ii) FPS : Inverse / Division
            iii) Integration / Differentiation
            iv) FPS : Logarithm / Exponentiation
            v) FPS : Power of Polynomial
            vi) Division - Quotient & Remainder
            vii) Polynomial Taylor Shift
            viii) Multipoint Evaluation
    D. Combinatorics
        1. Labeled Combinatorial Target
        2. The Twelvefold Way (12정도)
        3. Generating Function
            i) OGF(Ordinary Generating Function)
            ii) EGF(Exponentional Generating Function)
            iii) DGF?(Dirichlet Genetration Function) : Number Theory
        4. Umbral Calculus
III. Linear Algebra
IV. Geometry
    A. Basic Tools
        1. Outer Product (CCW)
        2. Sorting by Polar
        3. Segment Intersection
        4. Closest Point
        5. Furthest Point
    B. Convex Polygon (Convex Hull)
        1. Convex Hull Construction
        2. Convex Layer
        3. Rotating Calipers
        4. Point Containment
        5. Tangent to convex polygon
        6. Inner and Outer Tangent of two Convex's
    C. General Polygon
    D. Half Plane Intersection
    E. Delaunay Triangulation : Voronoi diagram
V. Greedy
    A. Rearrangement Inequality
VI. DP
    A. DP Optimization
        1. Convex Hull Trick
        2. Alien's Trick (Lagrangian Relaxation)
        3. Slope Trick
VII. String

i) Mo's Algorithm

2.1 astilate

```
#include <bits/stdc++.h>

#define getint(n) int n; scanf("%d%*c", &n)
#define getll(n) long long n; scanf("%lld%*c", &n)
#define getchar(n) char n; scanf("%c%*c", &n);
#define intab getint(a); getint(b)

#define forr(i, n) for(int i=1;i<=(n);i++)
#define fors(i, s, e) for(int i=(s); i<=(e); i++)
#define fore(i, e, s) for(int i=(e); i>=(s); i--)

#define fi first
#define se second
#define all(v) (v).begin(), (v).end()
#define rall(v) (v).rbegin(), (v).rend()
#define pb push_back

using namespace std;
using ll = long long;        using lll = __int128_t;
using pii = pair<int,int>;   using pll = pair<ll,ll>;
using vi = vector<int>;      using vl = vector<ll>;
using vii = vector<pii>;     using vll = vector<pll>;
```

2.2 qwerty

```
#pragma GCC optimize("O3")
#pragma GCC optimize("Ofast")
#pragma GCC optimize("unroll-loops")

#define endl '\n'
#define foreach(i, l, r) for(ll i=(l); i<=(r); i++)
#define forreach(i, l, r) for(ll i=l; i>=(r); i--)
```

2.3 Tips for Inequality with Rational Number

Let A, B, x, n be integer, and operator '/' means floor division (quotient).

$$Ax \leqslant B \Leftrightarrow x \leqslant B/A$$
$$Ax < B \Leftrightarrow x \leqslant (B-1)/A$$
$$Ax \geqslant B \Leftrightarrow x \geqslant (B+A-1)/A$$
$$Ax > B \Leftrightarrow x \geqslant B/A+1$$

$$x < n \Leftrightarrow x+1 \leqslant n$$

3 Math

3.1 Equations

$$ax^2+bx+c=0 \Rightarrow x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

The extremum is given by $x=-b/2a$.

$$\begin{matrix} ax+by=e \\ cx+dy=f \end{matrix} \Rightarrow \begin{matrix} x=\dfrac{ed-bf}{ad-bc} \\ y=\dfrac{af-ec}{ad-bc} \end{matrix}$$

In general, given an equation $Ax=b$, the solution to a variable x_i is given by

$$x_i=\frac{\det A_i'}{\det A}$$

where A_i' is A with the i'th column replaced by b.

3.2 Trigonometry

$$\sin(v+w)=\sin v\cos w+\cos v\sin w$$
$$\cos(v+w)=\cos v\cos w-\sin v\sin w$$
$$\tan(v+w)=\frac{\tan v+\tan w}{1-\tan v\tan w}$$
$$\sin v+\sin w=2\sin\frac{v+w}{2}\cos\frac{v-w}{2}$$
$$\cos v+\cos w=2\cos\frac{v+w}{2}\cos\frac{v-w}{2}$$
$$(V+W)\tan(v-w)/2=(V-W)\tan(v+w)/2$$

where V,W are lengths of sides opposite angles v,w.

$$a\cos x+b\sin x=r\cos(x-\phi)$$
$$a\sin x+b\cos x=r\sin(x+\phi)$$

where $r=\sqrt{a^2+b^2}, \phi=\operatorname{atan2}(b,a)$.

3.3 Geometry

3.3.1 Triangles

Side lengths: a,b,c

Semiperimeter: $p=\dfrac{a+b+c}{2}$

Area: $A=\sqrt{p(p-a)(p-b)(p-c)}$

Circumradius: $R=\dfrac{abc}{4A}$

Inradius: $r=\dfrac{A}{p}$

Length of the median (divides the triangle into two equal area triangles): $m_a=\frac{1}{2}\sqrt{2b^2+2c^2-a^2}$

Length of the bisector (divides angles into two): $s_a=\sqrt{bc\left[1-\left(\dfrac{a}{b+c}\right)^2\right]}$

Law of sines: $\dfrac{\sin\alpha}{a}=\dfrac{\sin\beta}{b}=\dfrac{\sin\gamma}{c}=\dfrac{1}{2R}$

Law of cosines: $a^2=b^2+c^2-2bc\cos\alpha$

Law of tangents: $\dfrac{a+b}{a-b}=\dfrac{\tan\dfrac{\alpha+\beta}{2}}{\tan\dfrac{\alpha-\beta}{2}}$

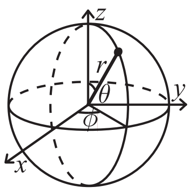
3.3.2 Quadrilaterals

With side lengths a,b,c,d, diagonals e,f, diagonals angle θ, area A and magic flux $F=b^2+d^2-a^2-c^2$:

$$4A=2ef\cdot\sin\theta=F\tan\theta=\sqrt{4e^2f^2-F^2}$$

For cyclic quadrilaterals the sum of opposite angles is $180°$, $ef=ac+bd$, and $A=\sqrt{(p-a)(p-b)(p-c)(p-d)}$.

3.3.3 Spherical coordinates



$$x = r\sin\theta\cos\phi \qquad r = \sqrt{x^2+y^2+z^2}$$
$$y = r\sin\theta\sin\phi \qquad \theta = \mathrm{acos}(z/\sqrt{x^2+y^2+z^2})$$
$$z = r\cos\theta \qquad \phi = \mathrm{atan2}(y,x)$$

3.4 Derivatives/Integrals

$$\frac{d}{dx}\arcsin x = \frac{1}{\sqrt{1-x^2}} \qquad \frac{d}{dx}\arccos x = -\frac{1}{\sqrt{1-x^2}}$$
$$\frac{d}{dx}\tan x = 1+\tan^2 x \qquad \frac{d}{dx}\arctan x = \frac{1}{1+x^2}$$
$$\int \tan ax = -\frac{\ln|\cos ax|}{a} \qquad \int x\sin ax = \frac{\sin ax - ax\cos ax}{a^2}$$
$$\int e^{-x^2} = \frac{\sqrt{\pi}}{2}\mathrm{erf}(x) \qquad \int xe^{ax}dx = \frac{e^{ax}}{a^2}(ax-1)$$

Integration by parts:

$$\int_a^b f(x)g(x)dx = [F(x)g(x)]_a^b - \int_a^b F(x)g'(x)dx$$

3.5 Sums

$$c^a + c^{a+1} + \cdots + c^b = \frac{c^{b+1}-c^a}{c-1}, c \neq 1$$

$$1+2+3+\cdots+n = \frac{n(n+1)}{2}$$
$$1^2+2^2+3^2+\cdots+n^2 = \frac{n(2n+1)(n+1)}{6}$$
$$1^3+2^3+3^3+\cdots+n^3 = \frac{n^2(n+1)^2}{4}$$
$$1^4+2^4+3^4+\cdots+n^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$

3.6 Series

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots, \ (-\infty < x < \infty)$$
$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots, \ (-1 < x \leq 1)$$
$$\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{2x^3}{32} - \frac{5x^4}{128} + \ldots, \ (-1 \leq x \leq 1)$$
$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots, \ (-\infty < x < \infty)$$
$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots, \ (-\infty < x < \infty)$$

3.7 Probability theory

Let X be a discrete random variable with probability $p_X(x)$ of assuming the value x. It will then have an expected value (mean) $\mu = \mathbb{E}(X) = \sum_x xp_X(x)$ and variance $\sigma^2 = V(X) = \mathbb{E}(X^2) - (\mathbb{E}(X))^2 = \sum_x (x-\mathbb{E}(X))^2 p_X(x)$ where σ is the standard deviation. If X is instead continuous it will have a probability density function $f_X(x)$ and the sums above will instead be integrals with $p_X(x)$ replaced by $f_X(x)$.

Expectation is linear:

$$\mathbb{E}(aX+bY) = a\mathbb{E}(X) + b\mathbb{E}(Y)$$

For independent X and Y,

$$V(aX+bY) = a^2V(X) + b^2V(Y).$$

3.7.1 Binomial distribution

The number of successes in n independent yes/no experiments, each which yields success with probability p is $\mathrm{Bin}(n,p)$, $n = 1,2,\ldots$, $0 \leq p \leq 1$.

$$p(k) = \binom{n}{k} p^k (1-p)^{n-k}$$
$$\mu = np, \ \sigma^2 = np(1-p)$$

$\mathrm{Bin}(n,p)$ is approximately $\mathrm{Po}(np)$ for small p.

3.7.2 First success distribution

The number of trials needed to get the first success in independent yes/no experiments, each which yields success with probability p is $\mathrm{Fs}(p)$, $0 \leq p \leq 1$.

$$p(k) = p(1-p)^{k-1}, \ k = 1,2,\ldots$$
$$\mu = \frac{1}{p}, \ \sigma^2 = \frac{1-p}{p^2}$$

3.7.3 Poisson distribution

The number of events occurring in a fixed period of time t if these events occur with a known average rate κ and independently of the time since the last event is $\mathrm{Po}(\lambda)$, $\lambda = t\kappa$.

$$p(k) = e^{-\lambda}\frac{\lambda^k}{k!}, k = 0,1,2,\ldots$$
$$\mu = \lambda, \ \sigma^2 = \lambda$$

3.7.4 Uniform distribution

If the probability density function is constant between a and b and 0 elsewhere it is $\mathrm{U}(a,b)$, $a < b$.

$$f(x) = \begin{cases} \frac{1}{b-a} & a < x < b \\ 0 & \text{otherwise} \end{cases}$$
$$\mu = \frac{a+b}{2}, \ \sigma^2 = \frac{(b-a)^2}{12}$$

3.7.5 Exponential distribution

The time between events in a Poisson process is $\mathrm{Exp}(\lambda)$, $\lambda > 0$.

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$
$$\mu = \frac{1}{\lambda}, \ \sigma^2 = \frac{1}{\lambda^2}$$

3.7.6 Normal distribution

Most real random values with mean μ and variance σ^2 are well described by $\mathcal{N}(\mu, \sigma^2)$, $\sigma > 0$.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

If $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ then

$$aX_1 + bX_2 + c \sim \mathcal{N}(\mu_1 + \mu_2 + c, a^2\sigma_1^2 + b^2\sigma_2^2)$$

3.8 Prime Number

3.8.1 Distribution of Prime Number

1e2	25	1e6	78,498	1e10	<5e8
1e3	168	1e7	664,579	1e11	<5e9
1e4	1,229	1e8	<6e6	1e12	<4e10
1e5	9,592	1e9	<6e7	1e13	<4e11

3.8.2 Prime Gap

$2 \cdot 10^5$ 이하의 소수 간극 ≤ 100

2^{32} 이하의 소수 간극 ≤ 464

2^{64} 이하의 소수 간극 ≤ 1550

3.9 Miller-Rabin Algorithm

Usage: `is_p(X)` : returns true if X is prime, otherwise false.

When $X \leq 2^{32}, D = \{2, 7, 61\}$ is sufficient;

$X \leq 2^{64}, D = \{p | p \text{ is prime}, p \leq 37\}$ is sufficient.

Time Complexity: $\mathcal{O}(\log^3 X)$

```
ll pow(ll a, ll b, ll mod) {
    ll ret = 1;
    for(int st=0; (1LL<<st) <= b; st++) {
        if((1LL<<st) & b) ret=(lll)ret*a%mod;
        a=(lll)a*a%mod;
    }
    return ret;
}
bool miller(ll n, ll a) {
    if(n == a) return true;
    ll x = n-1;
    if(pow(a, x, n) != 1) return false;
    while(x%2==0) {
        x/=2;
        ll t = pow(a, x, n);
        if(t!=1 and t!=n-1) return false;
        if(t==n-1) return true;
    }
    return true;
}
bool is_p(ll n) {
    if(n<=2) return n==2;
    vi D = {2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37};
    for(auto i:D) if(!miller(n, i)) return false;
    return true;
}
```

3.10 Pollad Rho Algorithm

Usage: `po_rho(N)` : returns array of prime factors of X.

Time Complexity: $\mathcal{O}(N^{1/4})$

```
void fact(ll n, vl& ret) {
    if(n == 1) return;
    else if(n%2 == 0) ret.pb(2), fact(n/2, ret);
    else if(is_p(n)) ret.pb(n);
    else {
        ll a, b, c, g = n;
        auto f = [&c, &n](ll x)->ll{return (c+(lll)x*x)%n;};

        do {
            if(g == n) a=b=rand()%(n-2)+2, c=rand()%20+1;
            a=f(a); b=f(f(b));
            g = gcd(a-b, n);
        } while(g == 1);
        fact(g, ret); fact(n/g, ret);
    }
}
vl po_rho(ll n) {
    vl ret;
    fact(n, ret);
    sort(all(ret));
    return ret;
}
```

3.11 Primitive Root

Usage: Calculate one of the primitive roots of given prime.

```
ll primary_root(ll p) {
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<ll> distrib(1, p-1);

    //distrib(gen);
    vl g = po_rho(p-1);

    while(true) {
        ll c = distrib(gen);

        bool ok = true; ll u = p-1;
        ll b = 1;
        for(auto i:g) {
            if(i != b) u = p-1;
            ll x = pow(c, u/i, p);
            if(x == 1) ok = false;
            u /= i; b = i;
        }

        if(ok) return c;
    }
}
```

3.12 Diopantos Equation(Extended Euclidian Algorithm)

Usage: `diophantos(a, b)` : return one integer solution of $ax+by = 1$, satisfying $0 \leq x < b$.

Time Complexity: $\mathcal{O}(\log(\max(a, b)))$

```
pll diophantos(ll a, ll b) {
    assert(a>0 and b>=0);
```

```
    if(b == 0) return {1, 0};
    auto [y, x] = diophantos(b, a%b); y = y-(a/b)*x;
    if(x < 0 or x >= b) {
        ll t = x/b;
        if(x%b < 0) t--;

        x -= b*t; y += a*t;
    }
    return {x, y};
}
```

3.13 Chinese Remainder Theorem

Usage: `crt(pll p, pll q)` : return `pll r`, satisfying follows:

$$\begin{aligned} x &\equiv \texttt{p.fi} \mod \texttt{p.se} \\ \text{and } x &\equiv \texttt{q.fi} \mod \texttt{q.se} \\ \leftrightarrow x &\equiv \texttt{r.fi} \mod \texttt{r.se} \end{aligned}$$

If there's no such `r`, return $\{-1, -1\}$.

Time Complexity: $\mathcal{O}(\log A)$

```
pll crt(pll p, pll q) {
    if(p.fi > q.fi) swap(p, q);
    auto [a, A] = p;
    auto [b, B] = q;

    ll g = gcd(A, B);
    if((b-a)%g != 0) return {-1, -1};

    ll i = A, j = B, k = b-a;
    i/=g; j/=g; k/=g;
    auto [x, y] = diophantos(i, j);
    return {(ll)((a+(lll)A*k*x)%(A*B/g)), A*B/g};
}
```

3.14 Harmonic Lemma

Usage: `f(N)` : return the value

$$\sum_{i=1}^{N} \left\lfloor \frac{N}{i} \right\rfloor = \left\lfloor \frac{N}{1} \right\rfloor + \left\lfloor \frac{N}{2} \right\rfloor + \cdots + \left\lfloor \frac{N}{N} \right\rfloor$$

Using the fact that $\left\lfloor \frac{N}{x} \right\rfloor$ has $\mathcal{O}(\sqrt{N})$ different values.

Time Complexity: $\mathcal{O}(\sqrt{N})$

```
ll harmonic(ll n) {
    ll ans = 0;
    for(ll i = 1; i <= n; i = n/(n/i)+ 1) {
        //for j \in [i, n/(n/i)] : n/j == n/i

        ans += n/i * (n/(n/i) - i + 1);

        // \sum_{i=1}^n {f(n/i)}
        // ans += f(n/i) * (n/(n/i) - i + 1)
    }
    return ans;
}
```

3.15 Floor Sum (Sum of Floor of Rational Arithmetic Sequence)

Usage: `floor_sum(A, B, C, N)` : return the value

$$\sum_{x=0}^{N-1} \left\lfloor \frac{Ax+B}{C} \right\rfloor$$

Time Complexity: $\mathcal{O}(\log N)$

```
ll floor_sum(ll a, ll b, ll c, ll n)
{
    if(a == 0) return b/c*n;
    if(a>=c or b>=c) return n*(n-1)/2 * (a/c) + n * (b/c) +
    floor_sum(a%c, b%c, c, n);
    ll m = (a*(n-1)+b)/c;
    return m*n - floor_sum(c, c-b+a-1, a, m);
}
```

3.16 FFT - Convolution

Time Complexity: $\mathcal{O}(N \log N)$

```
using cpx = complex<double>;
using vcpx = vector<cpx>;
void fft(vcpx &a, bool inv = false) {
    int n = a.size(), j = 0; assert((n&-n) == n);
    for(int i=1; i<n; i++) {
        int bit = (n >> 1);
        while(j >= bit) {
            j -= bit;
            bit >>= 1;
        }
        j += bit;
        if(i < j) swap(a[i], a[j]);
    }

    vcpx roots(n/2);
    prec c = 2 * pi * (inv ? -1 : 1);
    for(int i=0; i<n/2; i++)
        roots[i] = cpx(cosl(c * i / n), sinl(c * i / n));

    for(int i=2; i<=n; i<<=1) {
        int step = n / i;
        for(int j=0; j<n; j+=i) {
            for(int k=0; k<i/2; k++) {
                cpx u = a[j+k], v =
                a[j+k+i/2]*roots[step*k];
                a[j+k] = u+v;
                a[j+k+i/2] = u-v;
            }
        }
    }

    if(inv) for(int i=0; i<n; i++) a[i] /= n;
}

ll mod = 1e9+7;
vl conv(const vl& AA,const vl& BB) {
    const ll G = 1<<15;
    int n = AA.size()+BB.size()-1;
    int m = 1; while(m < n) m<<=1;

    int a = AA.size(), b = BB.size();

    vcpx A(m), B(m), C(m), D(m);
```

```
    fors(i, 0, a-1) A[i] = cpx(AA[i]/G, AA[i]%G);
    fors(i, 0, b-1) B[i] = cpx(BB[i]/G, BB[i]%G);

    fft(A); fft(B);

    fors(i, 0, m-1) {
        int j = i?m-i:0;
        cpx A1 = (A[i]+conj(A[j]))*cpx(0.5, 0);
        cpx A2 = (A[i]-conj(A[j]))*cpx(0, -0.5);

        cpx B1 = (B[i]+conj(B[j]))*cpx(0.5, 0);
        cpx B2 = (B[i]-conj(B[j]))*cpx(0, -0.5);

        C[i] = A1*B1 + A2*B2*cpx(0, 1);
        D[i] = A2*B1 + A1*B2*cpx(0, 1);
    }

    fft(C, true); fft(D, true);



    vl ret(m); ll G1 = G%mod, G2 = (lll)G*G%mod;
    fors(i, 0, m-1) {
        ll p = ll(C[i].real()+0.5);
        ll q = ll(D[i].real()+0.5) + ll(D[i].imag()+0.5);
        ll r = ll(C[i].imag()+0.5);

        p %= mod; q %= mod; r %= mod;
        ret[i] =
        (((lll)p*G2)%mod+((lll)q*G1)%mod+r%mod)%mod;
    }
    ret.resize(n);
    return ret;
}
```

3.17 NTT - Number Theoretic Transform

Usage: helloworld

```
namespace GMS {
    template<ll mod>
    ll pow(ll a, ll b) {
        static_assert(mod <= (ll)2e9, "mod should be less
        than 2e9");
        a %= mod;
        ll ret = 1;
        while(b != 0) {
            if(b&1) ret = ret*a%mod;
            a = a*a%mod; b>>=1;
        }
        return ret;
    }
    template<ll mod, ll w>
    void ntt(vector<ll> &a, bool inv = false) {
        static_assert(mod <= (ll)2e9, "mod should be less
        than 2e9");
        int n = a.size(), j = 0;

        assert((n & -n) == n && (mod-1)%n == 0);

        for(int i=1; i<n; i++) {
            int bit = (n >> 1);
            while(j >= bit) {
                j -= bit;
                bit >>= 1;
            }
            j += bit;
            if(i < j) swap(a[i], a[j]);
        }

        static vector<ll> root[30], iroot[30];
        for(int st=1; (1<<st) <= n; st++) {
            if(root[st].empty()) {
                ll t = pow<mod>(w, (mod-1)/(1<<st));

                root[st].pb(1);
                for(int i=1; i<(1<<(st-1)); i++)
                    root[st].pb(root[st].back()*t%mod);
            }
            if(iroot[st].empty()) {
                ll t = pow<mod>(w, (mod-1)/(1<<st));
                t = pow<mod>(t, mod-2);

                iroot[st].pb(1);
                for(int i=1; i<(1<<(st-1)); i++)
                    iroot[st].pb(iroot[st].back()*t%mod);
            }
        }

        vector<ll>* r = (inv?root:iroot);

        for(int st = 1; (1<<st) <= n; st++) {
            int i = 1<<st; //int step = n / i;
            for(int j=0; j<n; j+=i) {
                for(int k=0; k<i/2; k++) {
                    ll u = a[j+k], v = a[j+k+i/2] *
                    r[st][k]%mod;
                    a[j+k] = (u+v)%mod;
                    a[j+k+i/2] = (mod+u-v)%mod;
                }
            }
        }
        if(inv) {
            ll in = pow<mod>(n, mod-2);
            for(int i=0; i<n; i++) a[i] = a[i]*in%mod;
        }
    }

    template<ll mod, ll w>
    vl conv(vl A, vl B) {
        int n = A.size(), m = B.size();
        int t = 1; while(t < n+m-1) t*=2;
        A.resize(t); B.resize(t);

        ntt<mod, w>(A); ntt<mod, w>(B);

        fors(i, 0, t-1) A[i] = A[i]*B[i]%mod;

        ntt<mod, w>(A, true);
        A.resize(n+m-1);
```

```
        return A;
    }
} // namespace GMS
```

3.17.1 Good prime numbers to run NTT

595 591 169	71«23|1	$\omega = 3$
645 922 817	77«23|1	
897 581 057	107«23|1	
998 244 353	119«23|1	
1 300 234 241	155«23|1	
1 224 736 769	73«24|1	
2 130 706 433	127«24|1	
167 772 161	5«25|1	
469 762 049	7«26|1	

3.18 Polynomial (Formal Power Series)

```
namespace GMS {
    template<ll mod, ll w>
    struct Qring : public vl {
        using poly = Qring<mod, w>;
        Qring()                 : vl(1, 0)      {}
        Qring(ll c)             : vl(1, c%mod)  {}
        Qring(ll c, int n)      : vl(n, c%mod)  {}
        Qring(const vl& cp)     : vl(cp)        {for(auto
        &i:*this) i%=mod;}

        ll& operator[](ll idx) {
            if((unsigned)idx < size()) return
            vl::operator[](idx);
            this->resize(idx+1); return vl::operator[](idx);
        }
        ll operator[](ll idx) const {
            if((unsigned)idx < size()) return
            vl::operator[](idx);
            return 0LL;
        }

        void adjust() {
            while(size() > 1 and back() == 0) pop_back();
        }
        void adjust(int n){resize(n, 0);}

        ll operator()(ll x) {
            x %= mod; ll ret = 0;
            for(auto it=rbegin(); it!=rend(); it++)
                ret = (ret*x+*it)%mod;

            return ret;
        }

        friend poly operator%(const poly& A, int B){ //
        remainder by x^B
            poly ret(A); ret.resize(B, 0);
            return ret;
        }

        friend poly operator+(const poly& A, const poly& B)
        {
            int n = max(A.size(), B.size()); poly ret(0, n);
            fors(i, 0, n-1) ret[i] = A[i]+B[i];
            for(auto&i:ret) if(i >= mod) i -= mod;
            return ret.adjust(), ret;
        }

        friend poly operator-(const poly& A) {
            int n = A.size(); poly ret(0, n);
            fors(i, 0, n-1) ret[i] = A[i]?mod-A[i]:0;
            return ret;
        }

        friend poly operator-(const poly& A, const poly& B)
        {
            int n = max(A.size(), B.size());
            poly ret(0, n);
            fors(i, 0, n-1) ret[i] = (mod+A[i]-B[i])%mod;
            ret.adjust();
            return ret;
        }

        friend poly operator*(ll x, const poly& B) {
            poly ret(B); x %= mod;
            for(auto &i : ret) i = (i*x)%mod;
            ret.adjust();
            return ret;
        }

        friend poly operator*(const poly& A, const poly& B)
        {
            poly ret = conv<mod, w>(A, B);
            // ACL : poly ret = atcoder::convolution<mod>(A,
            B);
            ret.adjust();
            return ret;
        }

        friend poly inv(const poly& A, int t) { assert(A[0]
        != 0);
            poly g = pow<mod>(A[0], mod-2);

            int st=1;
            while(st < t){st*=2; g = (-A%st*g%st+2)*g%st;}

            g.adjust(t);
            return g;
        }

        friend poly diff(const poly& A) {
            int n = A.size(); poly ret(0, n-1);
            fors(i, 0, n-2) ret[i] = (i+1)*A[i+1]%mod;
            return ret;
        }
        friend poly inte(const poly& A) {
```

```
    static vector<ll> inv(1, 1);
    int n = A.size(); poly ret(0, n+1);
    inv.resize(max((int)inv.size(), n+1));

    fors(i, 1, n) if(inv[i] == 0) inv[i] =
    pow<mod>(i, mod-2);
    fors(i, 1, n) ret[i] = inv[i]*A[i-1]%mod;
    return ret;
}

friend poly log(const poly& A, int t) { assert(A[0]
== 1);
    return inte(diff(A) * inv(A, t)%t)%t;
}

friend poly exp(const poly& A, int t) { assert(A[0]
== 0);
    poly g = 1; int st = 1;
    while(st < t) {st*=2; g = (A%st-log(g,
    st)+1)*g%st;}
    return g.adjust(), g;
}

friend poly pow(const poly& A, ll b, ll t) {
    poly ret(A); ret.adjust();
    if(ret.size() == 1) {
        ret[0] = pow<mod>(ret[0], b);
        ret.adjust(t);
        return ret;
    }

    ll idx = 0; while(ret[idx] == 0) idx++;
    if((__int128_t) idx * b >= t) return poly(0, t);

    ll c = ret[idx]; ll ic = pow<mod>(ret[idx],
    mod-2); poly g;
    int n = ret.size();
    fors(i, idx, n-1) g[i-idx] = ret[i]*ic%mod;
    g.resize(t-idx*b);
    g = exp(b * log(g, t-idx*b), t-idx*b);
    c = pow<mod>(c, b);

    ret = poly(0, t); fors(i, idx*b, t-1) ret[i] =
    g[i-idx*b] * c % mod;

    return ret;
}

//Only just Polynomial, not Qring
void rev() {
    int n=size(); poly& F = *this;
    for(int i=0; i<n/2; i++) std::swap(F[i],
    F[n-i-1]);
}
friend poly div_quot(poly F, poly G) {
    F.adjust(); G.adjust();
    ll df = F.size(), dg = G.size();
    if(df < dg) return poly(0);

    F.rev(); G.rev();
    F = F%(df-dg+1)*inv(G, df-dg+1)%(df-dg+1);
    F.rev();
    return F;
}
friend poly div_rem(poly F, poly G) {return
F-G*div_quot(F, G);}

friend poly shift(const poly& F, ll c) {
    ll n = F.size(); c %= mod;
    poly A(0, n); ll fac = 1;
    fors(i, 0, n-1) A[i] = F[i]*fac%mod, fac =
    fac*(i+1)%mod;
    A.rev();

    poly C(1, n); fors(i, 1, n-1) C[i] =
    C[i-1]*c%mod;
    ll facc = fac = pow<mod>(fac, mod-2)*n%mod;
    fore(i, n-1, 0) C[i] = C[i]*fac%mod, fac =
    fac*i%mod;

    poly B = C*A; B.resize(n); B.rev();
    fore(i, n-1, 0) B[i] = B[i]*facc%mod, facc =
    facc*i%mod;
    return B;
}
friend void calcG(vector<poly>& G, int i, int l, int
r, const vl& p) {
    if(l == r){ll g = p[l]?mod-p[l]:0; G[i] = vl({g,
    1}); return;}
    int mid = (l+r)/2;
    calcG(G, i*2, l, mid, p); calcG(G, i*2+1, mid+1,
    r, p);
    G[i] = G[i*2]*G[i*2+1];
}
friend void eval(const vector<poly>& G, int i, int
l, int r, poly&& F, vl& ret) {
    if(l == r){ret[l] = F[0];return;}
    int mid=(l+r)/2;
    eval(G, i*2, l, mid, div_rem(F, G[i*2]), ret);
    eval(G, i*2+1, mid+1, r, div_rem(F, G[i*2+1]),
    ret);
}
friend vl multipoint_eval(const poly& A, const vl&
B) {
    int m = B.size();
    vector<poly> G(4*m); calcG(G, 1, 0, m-1, B);
    vl ret(m, 0); eval(G, 1, 0, m-1, div_rem(A,
    G[1]), ret);
    return ret;
}
};
} // namespace GMS
```

3.19 Combinatorics

3.19.1 Labeled Combinatorial Target

- Permutation, Combination (with, w/o repetition)
  - Permutation ${}_nP_r = \frac{n!}{r!(n-r)!}$
  - Combination ${}_nC_r = \binom{n}{k} = \frac{n!}{r!}$
  - Permutation with repetition ${}_n\Pi_r = n^r$
  - Combination with repetition ${}_nH_r ={}_{n+r-1} C_r$
- Catalan Number C_n : the number of regular bracket string of length $2n$.
  - $C_n = \frac{1}{n+1}\binom{2n}{n}$
  - $C_n : 1, 1, 2, 5, 14, 42, 132, 429, 1430, \cdots$ (from index 0)
  - OGF of C_n : $c(x) = 1 + xc(x)^2, c(c) = \frac{1-\sqrt{1-4x}}{2x}$
  - Catalan's triangle $C(n, k)$: the number of string with n X's, k Y's, where the number of Y's is not greater than the number of X's when written down the string.
    - ∗ i.e. $C(4, 3)$: XXXXYYY, XXYXYYX are OK, but XXYYYXX is not OK.
    - ∗ $C(n, k) = \binom{n+k}{k} - \binom{n+k}{k-1} = \frac{n-k+1}{n+1}\binom{n+k}{k}$
    - ∗ We can give the condition relax : $C_m(n, k)$: the number of string with n X's, k Y's, where the number of Y's is not greater than (the number of X's) + m when written down the string.
    - ∗ $C_m(n, k) = \binom{n+k}{k} - \binom{n+k}{k-m}$ when $m \leqslant k \leqslant n + m - 1$.
    - ∗ $C_1(n, k) = C(n, k)$
- Derangement(교란순열) D_n : the number of permutation of length n which $\forall i\ p_i \neq i$.
  - D_n : $1, 0, 1, 2, 9, 44, 265, 1854, 14833, 133496, 1334961 \cdots$ (from index 0)
  - EGF of D_n : $D(x) = \sum_{n=0} \frac{D_n}{n!}x^n = \frac{e^{-x}}{1-x}$
- Stirling Number of the 1st/2nd kind.
  - Unsigned Stirling Number of the 1st kind $c(n, k) = \left[{n \atop k}\right]$: the number of permutation of length n with k disjoint cycles.
    - ∗ $c(n + 1, k) = n \times c(n, k) + c(n, k - 1)$
    - ∗ with boundary condition $c(0, 0) = 1, c(n, 0) = c(0, k) = 0\ \forall n, k > 0$
    - ∗ $c(n, k)$

$n\backslash k$	0	1	2	3	4	5	6
0	1						
1	0	1					
2	0	1	1				
3	0	2	3	1			
4	0	6	11	6	1		
5	0	24	50	35	10	1	
6	0	120	274	225	85	15	1

    - ∗ OGF of $\left[{n \atop k}\right]$: for fixed n, $\sum_{k=0}^{n}\left[{n \atop k}\right]x^k = x(x + 1)(x + 2)\cdots(x + n - 1)$: we can calculate it by polynomial shift & divide and conquer
    - ∗ EGF of $\left[{n \atop k}\right]$: for fixed k, $\sum_{n=k}^{\infty}\left[{n \atop k}\right]\frac{x^n}{n!} = \frac{(-\log(1-z))^k}{k!}$
    - ∗ Signed Stirling Number of the 1st kind $s(n, k) = (-1)^{n-k}c(n, k)$
    - ∗ EGF of $s(n, k)$: for fixed k, $\sum_{n=k}^{\infty} s(n, k)\frac{x^n}{n!} = \frac{(\log(1+z))^k}{k!}$
  - Stirling Number of the 2nd kind $S(n, k) = \left\{{n \atop k}\right\}$: the number of partition of n labeled objects into k nonempty unlabelled subsets.
    - ∗ $S(n, k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^{k-i}\binom{k}{i}i^n$
    - ∗ $S(n, k)$

$n\backslash k$	0	1	2	3	4	5	6
0	1						
1	0	1					
2	0	1	1				
3	0	1	3	1			
4	0	1	7	6	1		
5	0	1	15	25	10	1	
6	0	1	31	90	65	15	1

    - ∗ OGF of $\left\{{n \atop k}\right\}$: for fixed n, $\sum_{k=0}^{n}\left\{{n \atop k}\right\}x^k = T_n(x)$, where $T_n(x) = e^{-x}\sum_{k=0}^{\infty}\frac{k^n}{k!}x^k$ is Touchard Polynomials.
    - ∗ EGF of $\left\{{n \atop k}\right\}$: for fixed k, $\sum_{n=k}^{\infty}\left\{{n \atop k}\right\}x^n = \frac{(e^x-1)^k}{k!}$
- Bell Number B_n : the number of partition of set with size n.
  - i.e. partitionize $\{a, b, c\}$ is $\{\{a\}, \{b\}, \{c\}\}, \{\{a, b\}, \{c\}\}, \{\{b, c\}, \{a\}\}, \{\{c, a\}, \{b\}\}, \{\{a, b, c\}\}$. Therefore, $B_3 = 5$.
  - $B_n : 1, 1, 2, 5, 15, 52, 203, 877, 4140, \cdots$ (from index 0)
  - $B_n = \sum_{k=0}^{n}\left\{{n \atop k}\right\}$ (2nd kind of stirling number)
  - EGF of B_n : $B(x) = \sum_{n=0}\frac{B_n}{n!}x^n = e^{e^x-1}$
  - Ordered Bell Number(Fubini Number) a_n : the number of distinct weak ordering on a set of n elements.
    - ∗ $a_n : 1, 1, 3, 13, 75, 541, 4683, 47293, 545835, 7087261, \cdots$
    - ∗ $a_n = \sum_{k=0}^{n} k!\left\{{n \atop k}\right\}$ (stirling number of the 2nd kind)
    - ∗ EGF of a_n : $a(x) = \sum_{n=0}^{\infty}\frac{a_n}{n!}x^n = \frac{1}{2-e^x}$
- Integer Partition $P(n, k)$: the number of partitions of n into exactly k parts.
  - i.e. $P(4, 2) = 2$ since $4 = 1 + 3 = 2 + 2$
  - we also say $P(n) = \sum_{k=1}^{n} P(n, k)$
  - $P(n, k)$

$n\backslash k$	1	2	3	4	5	6	7	8	9
1	1								
2	1	1							
3	1	1	1						
4	1	2	1	1					
5	1	2	2	1	1				
6	1	3	3	2	1	1			
7	1	3	4	3	2	1	1		
8	1	4	5	5	3	2	1	1	
9	1	4	7	6	5	3	2	1	1

- 12정도(the Twelvefold Way) : the number of functions (Domain $|N| = n$, Codomain $|X| = x$)

f-class	Any f	Injective	Surjective
Distinct f	${}_n\Pi_x$	${}_xP_n$	$x!\left\{{x \atop n}\right\}$
S_n orbits $f \circ S_n$	${}_xH_n$	${}_xC_n$	${}_{n-1}C_{n-x}$
S_x orbits $S_x \circ f$	$\sum_{k=0}^{x}\left\{{n \atop k}\right\}$	$[n \leqslant x]$	$\left\{{n \atop x}\right\}$
$S_n \times S_x$ orbits $S_x \circ f \circ S_n$	$P(n + x, x)$	$[n \leqslant x]$	$P(n, x)$

3.20 General Lucas Comb

```
vector<ll> euler(1000003, -1), primes;
//Generate primes and also calculate the euler number
void genprime() {
   for(ll i = 2;i<=1000002;i++) {
     if(euler[i]==-1) {
       primes.push_back(i);
       euler[i] = i-1;
       for(ll j = 2*i; j<=1000002; j+=i) {
         if(euler[j]==-1)euler[j] = j;
         euler[j] = (euler[j]/i)*(i-1);
       }
     }
   }
}
//Calculates x raised to the power of p % m
ll powll(ll x, ll p, ll m = 1ll<<62)
// Mod inverse
ll inverse(ll x, ll m)

//finds (n!)_p
ll ff(ll n, ll p, ll q)
{
    ll x = 1, y = powll(p, q);
    for(ll i = 2; i<=n;i++) if(i%p)
        x = (x*i)%y;
    return x%y;
}

//Generalized Lucas Theorem calculates nCm mod p^q
ll lucas_pow_comb(ll n, ll m, ll p, ll q) {
   ll r = n-m, x = powll(p, q);
   ll e0 = 0, eq = 0;
   ll mul = (p==2&&q>=3)? 1: -1;
   ll cr = r, cm = m, carry = 0, cnt = 0;
   while(cr||cm||carry)  {
     cnt++;
     int rr = cr%p, rm = cm%p;
     if(rr + rm + carry >= p) {
       e0++;
       if(cnt>=q)eq++;
     }
     carry = (carry+rr+rm)/p;
     cr/=p; cm/=p;
   }
   mul = powll(p, e0)*powll(mul, eq);
   ll ret = (mul % x + x) % x;
   ll temp = 1;
   for(ll i = 0;;i++)//This is THE line that calculates the
   formula {
     ret = ((ret*ff((n/temp)%x, p,
     q)%x)%x*(inverse(ff((m/temp)%x, p, q),
     x)%x*inverse(ff((r/temp)%x, p, q), x)%x)%x)%x;

     if(temp>n/p && temp>m/p && temp>r/p)
      break;
     temp = temp*p;
   }
   return (ret%x+x)%x;
}
```

4 Linear Algebra

4.1 Matrix

<MINTED>

5 Geometry

5.1 Mindset

```
using pii=pair<int,int>;
pii operator+(pii A, pii B){return {A.fi+B.fi, A.se+B.se};}
pii operator-(pii A, pii B){return {A.fi-B.fi, A.se-B.se};}
ll operator*(pii A, pii B){return
(ll)A.fi*B.fi+(ll)A.se*B.se;} // inner product
ll operator/(pii A, pii B){return
(ll)A.fi*B.se-(ll)A.se*B.fi;} // outer product

// 각도 정렬 (O = pii(0, 0))
sort(P+1, P+1+n, [](pii A, pii B) {
    if(B == O) return false;
    if(A == O) return true;
    return (A<O)!=(B<O)?A>B:A/B<0;
});

// 선분 교차
// Segment : fi에서 시작하는 se 벡터
// fi + k * se, 0 <= k <= 1

using Segment = pair<pii, pii>;
int isJoin(const Segment& A, const Segment& B) {
    if(B.se/A.se != 0) {
        ll p = (A.fi-B.fi)/A.se;
        ll q = B.se/A.se;

        if(q<0) q = -q, p = -p;
        if(p < 0 or p > q) return false;

        p = (B.fi-A.fi)/B.se;
        q = A.se/B.se;

        if(q<0) q = -q, p = -p;
        if(p < 0 or p > q) return false;

        return true;
    }
    else {
        if((A.fi-B.fi)/A.se != 0) return false;

        ll p = A.fi*A.se, q = (A.fi+A.se)*A.se;
        ll r = B.fi*A.se, s = (B.fi+B.se)*A.se;

        if(p>q) swap(p, q); if(r>s) swap(r, s);

        if(max(p, r) > min(q, s)) return false;

        return true;
```

```
    }
}
```

5.2 kactl templete

```
typedef pair<ll, ll> pll;
#define _x first
#define _y second

pll operator-(const pll &a, const pll &b){
    return {a._x - b._x, a._y - b._y};
}

ll cross(const pll &a, const pll &b){
    return a._x * b._y - b._x * a._y;
}

ll dot(const pll &a, const pll &b){
    return a._x * b._x + a._y * b._y;
}

int ccw(const pll &p1, const pll &p2, const pll &p3){
    ll res = cross(p2 - p1, p3 - p1);
    return (res != 0) * (res < 0 ? -1 : 1);
}

// dist of point - point
double dist(const pll &p1, const pll &p2){
    return sqrt((p1._x - p2._x) * (p1._x - p2._x) + (p1._y -
    p2._y) * (p1._y - p2._y));
}

// dist of line - point
double dist(const pll &l1, const pll &l2, const pll &p){
    ll area = abs(cross(l2 - l1, p - l1));
    return area / dist(l1, l2);
}

// dist of seg - point
double segDist(P& s, P& e, P& p) {
  if (s==e) return (p-s).dist();
  auto d = (e-s).dist2(), t = min(d,max(.0,(p-s).dot(e-s)));
  return ((p-s)*d-(e-s)*t).dist()/d;
}

P perp() const { return P(-y, x); } // rotates +90 degrees
int sgn(T x) { return (x > 0) - (x < 0); } // sign of x

// Returns where p is as seen from s towards e. 1/0/-1 <->
left/on line/right.
int sideOf(P s, P e, P p) { return sgn(s.cross(e, p)); }

// Returns a vector of either 0, 1, or 2 intersection
points.
template<class P>
vector<P> circleLine(P c, double r, P a, P b) {
  P ab = b - a, p = a + ab * (c-a).dot(ab) / ab.dist2();
  double s = a.cross(b, c), h2 = r*r - s*s / ab.dist2();
  if (h2 < 0) return {};
  if (h2 == 0) return {p};
  P h = ab.unit() * sqrt(h2);
  return {p - h, p + h};
}

// Computes the pair of points at which two circles
intersect.
typedef Point<double> P;
bool circleInter(P a,P b,double r1,double r2,pair<P, P>*
out) {
  if (a == b) { assert(r1 != r2); return false; }
  P vec = b - a;
  double d2 = vec.dist2(), sum = r1+r2, dif = r1-r2,
         p = (d2 + r1*r1 - r2*r2)/(d2*2), h2 = r1*r1 -
         p*p*d2;
  if (sum*sum < d2 || dif*dif > d2) return false;
  P mid = a + vec*p, per = vec.perp() * sqrt(fmax(0, h2) /
  d2);
  *out = {mid + per, mid - per};
  return true;
}

// Returns true iff p lies on the line segment from s to e.
template<class P> bool onSegment(P s, P e, P p) {
  return p.cross(s, e) == 0 && (s - p).dot(e - p) <= 0;
}

// Circumcircle
typedef Point<double> P;
double ccRadius(const P& A, const P& B, const P& C) {
  return (B-A).dist()*(C-B).dist()*(A-C).dist()/
      abs((B-A).cross(C-A))/2;
}
P ccCenter(const P& A, const P& B, const P& C) {
  P b = C-A, c = B-A;
  return A + (b*c.dist2()-c*b.dist2()).perp()/b.cross(c)/2;
}

// Minimum enclosing circle
pair<P, double> mec(vector<P> ps) {
  shuffle(all(ps), mt19937(time(0)));
  P o = ps[0];
  double r = 0, EPS = 1 + 1e-8;
  rep(i,0,sz(ps)) if ((o - ps[i]).dist() > r * EPS) {
    o = ps[i], r = 0;
    rep(j,0,i) if ((o - ps[j]).dist() > r * EPS) {
      o = (ps[i] + ps[j]) / 2;
      r = (o - ps[i]).dist();
      rep(k,0,j) if ((o - ps[k]).dist() > r * EPS) {
        o = ccCenter(ps[i], ps[j], ps[k]);
        r = (o - ps[i]).dist();
      }
    }
  }
  return {o, r};
}

// point inside convex hull in logN
bool inHull(const vector<P>& l, P p, bool strict = true) {
```

```
  int a = 1, b = sz(l) - 1, r = !strict;
  if (sz(l) < 3) return r && onSegment(l[0], l.back(), p);
  if (sideOf(l[0], l[a], l[b]) > 0) swap(a, b);
  if (sideOf(l[0], l[a], p) >= r || sideOf(l[0], l[b], p)<=
  -r)
    return false;
  while (abs(a - b) > 1) {
    int c = (a + b) / 2;
    (sideOf(l[0], l[c], p) > 0 ? b : a) = c;
  }
  return sgn(l[a].cross(l[b], p)) < r;
}

// convex hull
vector<pll> convex_hull(vector<pii> &arr){
    vector<pll> up, down;
    for(auto p : arr){
        while(up.size() >= 2 && ccw(up[up.size() - 2],
        up[up.size() - 1], p) >= 0) up.pop_back();
        while(down.size() >= 2 && ccw(down[down.size() - 2],
        down[down.size() - 1], p) <= 0) down.pop_back();
        up.push_back(p);
        down.push_back(p);
    }
    up.insert(up.end(), down.rbegin() + 1, down.rend());
    return up;
}

// rotating callipers
typedef Point<ll> P;
array<P, 2> hullDiameter(vector<P> S) {
  int n = sz(S), j = n < 2 ? 0 : 1;
  pair<ll, array<P, 2>> res({0, {S[0], S[0]}});
  rep(i,0,j)
    for (;; j = (j + 1) % n) {
      res = max(res, {(S[i] - S[j]).dist2(), {S[i], S[j]}});
      if ((S[(j + 1) % n] - S[j]).cross(S[i + 1] - S[i]) >=
      0)
        break;
    }
  return res.second;
}

// Closest pair
typedef Point<ll> P;
pair<P, P> closest(vector<P> v) {
  assert(sz(v) > 1);
  set<P> S;
  sort(all(v), [](P a, P b) { return a.y < b.y; });
  pair<ll, pair<P, P>> ret{LLONG_MAX, {P(), P()}};
  int j = 0;
  for (P p : v) {
    P d{1 + (ll)sqrt(ret.first), 0};
    while (v[j].y <= p.y - d.x) S.erase(v[j++]);
    auto lo = S.lower_bound(p - d), hi = S.upper_bound(p +
    d);
    for (; lo != hi; ++lo)
      ret = min(ret, {(*lo - p).dist2(), {*lo, p}});
    S.insert(p);
  }
  return ret.second;
}
```

6 Greedy

6.1 Rearrange Inequality - Extensions

Let A_i, B_i be non-decreasing sequence of length n, and p be some permutation, and let $inc := (1, 2, \ldots, n-1, n)$, $dec := (n, n-1, \ldots, 2, 1)$.

$S(p) = \sum_{i=1}^{n} A_i B_{p_i}$	maximize S	$\Rightarrow p : inc$
	minimize S	$\Rightarrow p : dec$
$P_{max}(p) = \max_i(A_i B_{p_i})$	minimize P_{max}	$\Rightarrow p : dec$
$P_{min}(p) = \min_i(A_i B_{p_i})$	maximize P_{min}	$\Rightarrow p : dec$
$A_{max}(p) = \max_i(A_i + B_{p_i})$	minimize A_{max}	$\Rightarrow p : dec$
$A_{min}(p) = \min_i(A_i + B_{p_i})$	maximize A_{min}	$\Rightarrow p : dec$
$D_{max}(p) = \max_i \lvert A_i - B_{p_i} \rvert$	minimize D_{max}	$\Rightarrow p : inc$
$D_{min}(p) = \min_i \lvert A_i - B_{p_i} \rvert$	maximize D_{min}	$\Rightarrow p : ?$

$$\text{Permutate } A \text{ s.t. maximize } \sum_{i=1}^{n} A_i A_{i+1} \ (\text{Let } A_{n+1} = A_1)$$

$$\Rightarrow \text{ Pendulum Arrangement}$$

7 DP

7.1 LIS

```
template<class I> vi lis(const vector<I>& S) {
  if (S.empty()) return {};
  vi prev(sz(S));
  typedef pair<I, int> p;
  vector<p> res;
  rep(i,0,sz(S)) {
    // change 0 -> i for longest non-decreasing subsequence
    auto it = lower_bound(all(res), p{S[i], 0});
    if (it == res.end()) res.emplace_back(), it =
    res.end()-1;
    *it = {S[i], i};
    prev[i] = it == res.begin() ? 0 : (it-1)->second;
  }
  int L = sz(res), cur = res.back().second;
  vi ans(L);
  while (L--) ans[L] = cur, cur = prev[cur];
  return ans;
```

```
}
```

7.2 DP Optimization

7.2.1 Convex Hull Optimization

$$\text{Recurrence}: D[i] = \min_{j<i}(B[j] \times A[i] + D[j])$$

$$\text{Complexity}: \mathcal{O}(N^2) \to \mathcal{O}(N \log N)$$

7.2.2 Divide and Conquer Optimization

$$\text{Recurrence}: D[i][j] = \min_{k<i}(D[i-1][k] + C[k][j])$$

$$\text{Condition}: C[i][j] \text{ is Monge}$$

$$(\text{ if } a \leqslant b \leqslant c \leqslant d, \text{ then } C[a][c] + C[b][d] \leqslant C[a][d] + C[b][c])$$

$$\text{Complexity}: \mathcal{O}(KN^2) \to \mathcal{O}(KN \log N)$$

```
//D[t][s...e]를 구해야 하고, j의 탐색 범위는 [l, r]
void f(int t, int s, int e, int l, int r){
    if(s > e) return;
    int m = s + e >> 1;
    int opt = l;
    for(int i=l; i<=r; i++){
        if(D[t-1][opt] + C[opt][m] > D[t-1][i] + C[i][m])
        opt = i;
    }
    D[t][m] = D[t-1][opt] + C[opt][m];
    f(t, s, m-1, l, opt);
    f(t, m+1, e, opt, r);
}
```

7.2.3 Monotone Queue Optimization

$$\text{Recurrence}: D[i] = \min_{j<i}(D[j] + C[j][i])$$

$$\text{Condition}: C[i][j] \text{ is Monge}$$

$$\text{Complexity}: \mathcal{O}(N^2) \to \mathcal{O}(N \log N)$$

7.2.4 Knuth's Optimization

$$\text{Recurrence}: D[i][j] = \min_{i\leqslant k<i}(D[i][k] + D[k+1][j]) + C[i][j]$$

$$\text{Condition}: C[i][j] \text{ is Monge } \&$$

$$C[a][d] \geqslant C[b][c] \text{ for } a \leqslant b \leqslant c \leqslant d$$

$$\text{Complexity}: \mathcal{O}(N^3) \to \mathcal{O}(N^2)$$

구간에 대해 동적 계획법(DP)을 수행할 때, 다음과 같은 점화식이 있다고 가정합니다:

$$a[i][j] = \min_{i<k<j}\left(a[i][k] + a[k][j]\right) + f(i,j)$$

여기서 (최소화된) 최적의 k가 i와 j 모두에 대해 증가한다고 하면, 구간의 길이에 따라 DP를 계산하며 $a[i][j]$에 대해 $k = p[i][j]$를 $p[i][j-1]$부터 $p[i+1][j]$ 사이에서만 탐색하면 됩니다.

7.2.5 Aliens Trick (Lagrangian relaxation)

$$\text{Recurrence}: D[k][i] = \min_{j<i}(D[k-1][j] + C[j+1][i])$$

$$\text{Condition}: D[x][N] \text{ is convex (is implied when } C[i][j] \text{ is Monge)}$$

$$(f(x+1) - f(x) \leqslant f(x+2) - f(x+1))$$

$$\text{Complexity}: \mathcal{O}(KN^2) \to \mathcal{O}(N^2 \log |W|)$$

7.2.6 Slope Trick

13323 BOJ 수열 1/2. 수열 A가 주어진다. 증가수열 B에 대해, $\sum_{i=1}^{N} |A_i - B_i|$를 최소화하고, 그 B를 찾아라.

```
const int N = 1e6+7;
int arr[N];
priority_queue<int> pq;
ll ans = 0;
int main()
{
    getint(n);
    forr(i, n) scanf("%d", arr+i);

    pq.push(arr[1]); int t=0; ll val = 0;
    fors(i, 2, n)
    {
        t++;
        int r = t + pq.top();
        if(r <= arr[i]) pq.push(arr[i]-t);
        else
        {
            pq.push(arr[i]-t); pq.push(arr[i]-t); pq.pop();
            ans += r-arr[i];
        }
    }

    printf("%lld", ans);
}


int arr[N];
priority_queue<int> pq;
int ans2[N];
int main()
{
    getint(n);
    forr(i, n) scanf("%d", arr+i);

    pq.push(arr[1]); ll ans = 0;
    ans2[1] = arr[1];
    fors(i, 2, n)
    {
        int r = (i-1) + pq.top();
        if(r <= arr[i]) pq.push(arr[i]-(i-1));
        else
        {
            pq.push(arr[i]-(i-1)); pq.push(arr[i]-(i-1));
            pq.pop();
            ans += r-arr[i];
        }

        ans2[i] = pq.top() + (i-1);
    }
```

```
    fore(i, n-1, 1) ans2[i] = min(ans2[i], ans2[i+1]-1);
    forr(i, n) printf("%d\n", ans2[i]);
}
```

7.3 LineContainer

Time Complexity: $\mathcal{O}(\log N)$

```
struct Line {
  mutable ll k, m, p;
  bool operator<(const Line& o) const { return k < o.k; }
  bool operator<(ll x) const { return p < x; }
};

struct LineContainer : multiset<Line, less<>> {
  // (for doubles, use inf = 1/.0, div(a,b) = a/b)
  static const ll inf = LLONG_MAX;
  ll div(ll a, ll b) { // floored division
    return a / b - ((a ^ b) < 0 && a % b); }
  bool isect(iterator x, iterator y) {
    if (y == end()) return x->p = inf, 0;
    if (x->k == y->k) x->p = x->m > y->m ? inf : -inf;
    else x->p = div(y->m - x->m, x->k - y->k);
    return x->p >= y->p;
  }
  void add(ll k, ll m) {
    auto z = insert({k, m, 0}), y = z++, x = y;
    while (isect(y, z)) z = erase(z);
    if (x != begin() && isect(--x, y)) isect(x, y =
    erase(y));
    while ((y = x) != begin() && (--x)->p >= y->p)
      isect(x, erase(y));
  }
  ll query(ll x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return l.k * x + l.m;
  }
};
```

7.4 SoS

Time Complexity: $\mathcal{O}(N * 2^N)$

```
int n = 20;
vector<int> a(1 << n);

// keeps track of the sum over subsets
// with a certain amount of matching bits in the prefix
vector<vector<int>> dp(1 << n, vector<int>(n));

vector<int> sos(1 << n);
for (int mask = 0; mask < (1 << n); mask++) {
  dp[mask][-1] = a[mask];
  for (int x = 0; x < n; x++) {
    dp[mask][x] = dp[mask][x - 1];
    if (mask & (1 << x)) { dp[mask][x] += dp[mask - (1 <<
    x)][x - 1]; }
  }
  sos[mask] = dp[mask][n - 1];
}

////////////////////////////////////////////
D[i]에 미리 i에 해당하는 값을 넣어둔다
fors(d, 0, 19) fors(i,0,(1<<20)-1)
    if(i & (1<<d)) D[i] += D[i^(1<<d)];
-> D[i] : sum of subset of mask i
```

8 String

8.1 KMP

```
int fail[N]; // N : s의 최대 길이
vi kmp(char obj[], char s[]) {
    vi ret;
    for(int i = 1, j = 0; s[i]; i++) {
        fail[i] = 0;
        while(j > 0 and s[i] != s[j]) j = fail[j-1];
        if(s[i] == s[j]) fail[i] = ++j;
    }
    for(int i = 0, j = 0; obj[i]; i++) {
        while(j > 0 and obj[i] != s[j]) j = fail[j-1];
        if(obj[i] == s[j]) {
            if(s[j+1]) j++;
            else ret.push_back(i-j), j = fail[j];
        }
    }
    return ret;
}
```

8.2 F, Z, M, SA(Suffix Array), LCP(Longest Common Prefix)

Usage: For string s(1-indexed) of length N;

`F[i]`	=	maximum $k < i$	s.t. $s[1\dots k] = s[i-k+1\dots i]$
`Z[i]`	=	maximum k	s.t. $s[1\dots k] = s[i\dots i+k-1]$
`M[i]`	=	maximum k	s.t. $s[i-k+1\dots i+k-1]$ is palindrom.
`SA[i]`	=	k	s.t. $s[k\dots N]$ is the i^{th} smallest of $\{s[1\dots N],\ s[2\dots N],\ \cdots,\ s[N\dots N]\}$
`LCP[i]`	=	maximum k	s.t. $s[SA[i-1]\dots SA[i-1]+k-1] = s[SA[i]\dots SA[i]+k-1]$

Time Complexity: $\mathcal{O}(N), \mathcal{O}(N), \mathcal{O}(N), \mathcal{O}(N\log N), \mathcal{O}(N)$, respectively

```
const int N = 1e5+7;
char s[N];
int F[N], Z[N], M[N];
int sa[N]; int ord[N], tmp[N], cnt[N];
int lcp[N];
int main() {
    scanf("%s", s+1);
    int n = strlen(s+1);

    // KMP - fail function {
        F[1] = 0; int j = 0;
        for(int i=2; i<=n;i++)
        {
            while(j > 0 and s[i] != s[j+1]) j = F[j];
            F[i] = j+=(s[i] == s[j+1]);
        }
```

```
}

//Z - Z array {
    Z[1] = n; int j = 1, r = 0;
    for(int i=2; i<=n; i++)
    {
        Z[i] = i<j+r?min(Z[i-j+1], j+r-i):0;
        while(s[1+Z[i]] == s[i+Z[i]]) Z[i]++;
        if(j+r < i+Z[i]) j = i, r = Z[i];
    }
}

//Manacher - M array {
    M[1] = 0; int j = 1, r = 0;
    for(int i=2; i<=n; i++)
    {
        M[i] = i<j+r?min(M[2*j-i], j+r-i):0;
        while(1 <= i-M[i]-1 && i+M[i]+1 <= n
               && s[i-M[i]-1] == s[i+M[i]+1]) M[i]++;
        if(j+r < i+M[i]) j = i, r = M[i];
    }
}

//Suffix Array - SA {
    int t = 1; ord[n+1] = 0; tmp[0] = 0; sa[0] = 0;

    auto cmp = [&t, &n](int i,int j) {
           return ord[i] == ord[j]
                  ?ord[min(i+t, n+1)]<ord[min(j+t, n+1)]
                  :ord[i]<ord[j];
    };

    forr(i, n) ord[i] = s[i], sa[i] = i;
    sort(sa+1, sa+1+n, [](int i,int j){return
    ord[i]<ord[j];});

    forr(i, n)  tmp[sa[i]] = tmp[sa[i-1]] +
    (ord[sa[i-1]]<ord[sa[i]]);
    swap(tmp, ord);

    while(t < n) {
        fors(i, 0, n) cnt[i] = 0;
        forr(i, n) cnt[ord[min(i+t, n+1)]]++;
        forr(i, n) cnt[i] += cnt[i-1];
        fore(i, n, 1) tmp[cnt[ord[min(i+t, n+1)]]--] =
        i;

        fors(i, 0, n) cnt[i] = 0;
        forr(i, n) cnt[ord[i]]++;
        forr(i, n) cnt[i] += cnt[i-1];
        fore(i, n, 1) sa[cnt[ord[tmp[i]]]--] = tmp[i];

        forr(i, n) tmp[sa[i]] = tmp[sa[i-1]] +
        cmp(sa[i-1], sa[i]);
        swap(ord, tmp);

        t<<=1;
        if(ord[sa[n]] == n) break;
    }
}

//LCP array {
    int k = 0;
    forr(i, n) if(ord[i] != 1) {
        int j = sa[ord[i]-1];
        while(s[i+k] == s[j+k]) k++;
        lcp[ord[i]] = k;

        if(k > 0) k--;
    }
}

printf("\nF : "); forr(i, n) printf("%d ", F[i]);
printf("\nZ : "); forr(i, n) printf("%d ", Z[i]);
printf("\nM : "); forr(i, n) printf("%d ", M[i]);
printf("\nSA : "); forr(i, n) printf("%d ", sa[i]);
printf("\nLCP : x "); fors(i, 2, n) printf("%d ",
lcp[i]);
    printf("\n"); forr(i, n) printf("%s\n", s+sa[i]);
}
```

9 Graph

9.1 SCC - Tarjan Algorithm

Usage: `scn[i]` : SCC number of node i, `nscc` : the number of SCCs

```
vi adj[N];
int in[N], c = 0;
stack<int> p;
bool fin[N]; int scn[N], nscc = 0;
int dfs(int s) {
    in[s] = ++c;
    p.push(s);

    int m = c;
    for(auto i : adj[s]) {
        if(in[i] == 0) m = min(m, dfs(i));
        else if(!fin[i]) m = min(m, in[i]);
    }

    if(m == in[s]) {
        nscc++;
        while(p.top() != s)
        {
            int i = p.top(); p.pop();
            scn[i] = nscc; fin[i] = true;
        }
        p.pop();
        scn[s] = nscc; fin[s] = true;
    }
    return m;
}

forr(i, n) if(!fin[i]) dfs(i);
```

9.2 Bipartite Matching - with DFS

Usage: Let's say that graph is bipartite. And Let's say that one group is A, and the other graph is B. $|A| = N$, $|B| = M$. `matching(c = s)` : add one matching from $s \in A$. If successfully matched, return true; otherwise return false. `selby[i] =` store $s \in A$, s.t. $i \in B$ is matched with s.
(e.g.) `forr(i, n) ans += matching(c=i);`

Time Complexity: $\mathcal{O}(VE)$

```
vector<int> sideadj[N];
int selby[M];
int chk[M], c;
bool matching(int s) {
    for(auto i : sideadj[s]) {
        if(chk[i] == c) continue;
        chk[i] = c;
        if(selby[i] and !matching(selby[i])) continue;
        selby[i] = s;
        return true;
    }
    return false;
}
```

9.2.1 Minimum Vertex Cover on Bipartite Graph(Kőnig's Therorem)

On bipartite graph,

$$|\texttt{Minimum Vertex Cover}| = |\texttt{Maximum Matching}|$$

To find Minimum Vertex Cover, (Should be added.)

9.2.2 Maximum Independent Set on Bipartite Graph

On bipartite graph,

$$|\texttt{Maximum Independent Set}| = |V| - |\texttt{Maximum Matching}|$$

* Note : Complement of the Vertex Cover is the Independent Set.

9.2.3 Minimum Path Cover on DAG

Let's think about the bipartite graph, with vertex set A and B, satisfying follow property:

- If there's edge from node i to node j on DAG, then there's edge connecting i^{th} node of A and j^{th} node of B, and vice versa.

Then following holds:
$|\texttt{Minimum Path Cover of DAG}| = |\texttt{Maximum Matching on Bipartite Graph}|$

9.2.4 Maximum Antichain on DAG(Dilworth's Theorem)

On DAG,

$$|\texttt{Minimum Path Cover}| = |\texttt{Maximum Antichain}|$$

9.3 Network Flow - Dinic

Usage: Construct graph with `connect(from, to, capacity, isDirected);`. Find the flow from S to T with `flow(S, T);`.

Time Complexity: $\mathcal{O}(V^2E)$, but it works like magic.

```
struct Edge {
    int to, cap, now;
    Edge* rev;
    Edge(int to,int cap):to(to), cap(cap), now(0){}
    int left(){return cap - now;}
    void flow(int f){now += f; rev->now -= f;}
    void reset(){now = 0;}
};
vector<Edge*> adj[N];

int lv[N]; bool chk[N];
bool bfs(int S, int T) {
    queue<int> q;
    q.push(S); lv[S] = 0; chk[S] = true;
    while(!q.empty()) {
        int s = q.front(); q.pop();
        for(auto i : adj[s]) {
            if(i->left() and !chk[i->to]) {
                lv[i->to] = lv[s]+1; chk[i->to] = true;
                q.push(i->to);
            }
        }
    }
    return chk[T];
}

Edge* hist[N]; int last[N];
bool dfs(int s, int T) {
    if(s == T) return true;

    for(int &j=last[s]; j < adj[s].size(); j++) {
        int i = adj[s][j]->to;
        if(adj[s][j]->left() == 0 or lv[i] != lv[s]+1)
        continue;
        hist[i] = adj[s][j];

        if(dfs(i, T)) return true;
    }
    return false;
}

ll flow(int S,int T) {
    ll ans = 0;
    while(bfs(S, T)) {
        while(dfs(S, T)) {
            int m = 2e9;
            int now = T;
            while(S != now) {
                m = min(m, hist[now]->left());
                now = hist[now]->rev->to;
            }
            now = T;
            while(S != now)
                hist[now]->flow(m), now =
                hist[now]->rev->to;
            ans += m;
        }
        memset(last, 0, sizeof last);
        memset(chk, 0, sizeof chk);
    }
```

```
    return ans;
}
// isDir : isDirected => 양방향 간선이면 false
void connect(int from, int to, int cap, bool isDir = true) {
    Edge *fw, *bw;
    fw = new Edge(to, cap);
    bw = new Edge(from, !isDir ? cap : 0);
    fw->rev = bw; bw->rev = fw;
    adj[from].push_back(fw);
    adj[to].push_back(bw);
}
```

9.4 MCMF - with SPFA

Usage: Construct graph with `connect(from, to, capacity, cost);`. Find the maximum flow and corresponding minimum cost from S to T with `flow(S, T);`.

Time Complexity: $\mathcal{O}(VEf)$, but it works like magic.

```
struct Edge {
    int to, cap, now;
    ll cost;
    Edge* rev;
    Edge(int to,int cap, ll cost)
        :to(to), cap(cap), now(0), cost(cost){}
    int left(){return cap - now;}
    ll flow(int f)
        {now += f; rev->now -= f; return cost * f;}
    void reset(){now = 0;}
};
vector<Edge*> adj[N];
Edge* hist[N]; ll dist[N]; bool inQueue[N], chk[N];
bool spfa(int s, int t) {
    memset(dist, 0, sizeof(dist));
    memset(chk, 0, sizeof(chk)); chk[s] = true;
    queue<int> q;
    memset(inQueue, 0, sizeof(inQueue));
    q.push(s); inQueue[s] = true;
    while(!q.empty()) {
        int now = q.front();
        q.pop(); inQueue[now] = false;
        for(auto e : adj[now]) {
            int next = e->to;
            if(e->left() > 0 and
                  (chk[next] == false
                          or dist[next] > dist[now] +
                          e->cost)) {
                chk[next] = true;
                dist[next] = dist[now] + e->cost;
                hist[next] = e;
                if(!inQueue[next])
                    q.push(next), inQueue[next] = true;
            }
        }
    }
    return chk[t];
}
// cost가 들어가면 항상 단방향만 가능하다. (양방향 : 2번
connect)
void connect(int from, int to, int cap, ll cost) {
    Edge *fw, *bw;
    fw = new Edge(to, cap, cost);
    bw = new Edge(from, 0, -cost);
    fw->rev = bw; bw->rev = fw;
    adj[from].push_back(fw);
    adj[to].push_back(bw);
}
//maximum matching & minimum cost
pair<ll, ll> flow(int S,int T) {
    ll ans = 0; ll cost = 0;
    while(spfa(S, T)) {
        int m = 2e9;

        int now = T;
        while(S != now) {
            m = min(m, hist[now]->left());
            now = hist[now]->rev->to;
        }
        now = T;
        while(S != now) {
            cost += hist[now]->flow(m);
            now = hist[now]->rev->to;
        }
        ans += m;
    }
    return {ans, cost};
}
```

10 Tree

10.1 HLD(Heavy Light Decomposition)

```
BOJ 트리와 쿼리 1
1 i c: i번 간선의 비용을 c로 바꾼다.
2 u v: u에서 v로 가는 단순 경로에 존재하는 비용 중에서 가장
큰 것을 출력한다.

const int N = 1e5+7;

vi adj[N]; int par[N]; int sz[N]; int d[N];
void dfs1(int s) {
    sz[s] = 1;
    for(int i=0;i<adj[s].size();i++)
        if(adj[s][i] == par[s])
            {adj[s].erase(adj[s].begin() + i); break;}
    for(auto &i : adj[s]) {
        par[i] = s; d[i] = d[s] + 1;
        dfs1(i);
        sz[s] += sz[i];
        if(sz[i] > sz[adj[s][0]]) swap(adj[s][0], i);
    }
}
int in[N], c; int top[N];
void dfs2(int s) {
    in[s] = ++c;
    for(auto i : adj[s]) {
        if(i == adj[s][0])
            top[i] = top[s];
```

```
        else top[i] = i;

        dfs2(i);
    }
}

Node *root; // Segment Tree
int query(int a,int b) {
    int ans = 0;
    while(top[a] != top[b]) {
        if(d[top[a]] > d[top[b]]) swap(a, b);
        ans = max(ans, query(root, in[top[b]], in[b]));
        b = par[top[b]];
    }

    if(d[a] > d[b]) swap(a, b);
    ans = max(ans, query(root, in[a]+1, in[b]));

    return ans;
}
map<pii, int> m;
int arr[N]; pii edge[N];
int main() {
    getint(n);
    forr(i, n-1) {
        intab; adj[a].pb(b); adj[b].pb(a);
        getint(c);
        m[{a,b}] = m[{b, a}] = c;
        edge[i] = {a,b};
    }

    dfs1(1); dfs2(1);
    forr(i, n) arr[in[i]] = m[{par[i], i}];
    root = new Node(1, n); init(root, arr);

    getint(Q);
    while(Q--) {
        getint(q);
        if(q == 1) {
            getint(i); getint(c);
            auto [a, b] = edge[i];
            if(par[b] == a) a = b;

            update(root, in[a], c, true);
        }
        if(q == 2) {
            intab;
            printf("%d\n", query(a, b));
        }
    }
}
```

10.2 Centroid Tree

```
vi adj[N]; bool cent[N];
int sz[N], par[N];

void getSz(int s){
    sz[s] = 1;
    for(auto i:adj[s]){
        if(cent[i]) continue;
        if(par[s] == i) continue;
        par[i] = s; getSz(i); par[i] = 0;

        sz[s] += sz[i];
    }
}
int getCent(int s, int n){
    for(auto i:adj[s])
        if(!cent[i] and sz[i] < sz[s] and sz[i] > n/2)
            return getCent(i, n);
    return s;
}

int cpar[N];
int getCentTree(int s){
    getSz(s);
    int C = getCent(s, sz[s]);
    cent[C] = true;
    for(auto i:adj[C]){
        if(cent[i]) continue;
        int c = getCentTree(i);
        cpar[c] = C;
    }
    return C;
}

int C = getCentTree(1); cpar[C] = -1;
```

11 Data Structure

11.1 PBDS - Policy-Based Data Structure

Time Complexity: Equivalent to std::set

```
#include <bits/stdc++.h>
#include <ext/rope>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace __gnu_cxx;

template<typename T>
using indexed_set = tree<T, null_type, less<T>, rb_tree_tag,
tree_order_statistics_node_update>;

indexed_set<int> s;
s.insert(3); s.insert(2); s.insert(3); s.insert(9);
s.insert(7); //2 3 7 9
s.insert(5); //2 3 5 7 9
s.erase(5); //2 3 7 9

auto x = s.find_by_order(2); // *x : 7

s.order_of_key(6) // 2
s.order_of_key(7) // 2
s.order_of_key(8) // 3

/////////////////////////////////////////////////////////////
```

```
// greater_equal <- ordered_multiset / greater <-
ordered_multiset
#define oset_greater tree<ll, null_type, greater_equal<ll>,
rb_tree_tag, tree_order_statistics_node_update>
#define oset_less tree<ll, null_type, less_equal<ll>,
rb_tree_tag, tree_order_statistics_node_update>

void oset_m_erase(ordered_set_greater &OS, ll val){
    int index = OS.order_of_key(val);
    oset_greater::iterator it = OS.find_by_order(index);
    if(it != OS.end() && *it == val) OS.erase(it);
}


/////////////////////////////////////////////////////////////
rope<ll> r;
r.insert(r.size() - t, i); //r.size()-t번째 자리에 i를 삽입
r.substr(a, b - a + 1) // a부터 (b-a+1)개 만큼을 잘라낸다.
즉, [a, b] 선택
```

11.2 rope

11.3 Union and Find - Queue Undoing

Time Complexity: $\mathcal{O}(\log^2 N)$

```
struct dsu_pb {
    const int N;
    vi par; stack<pair<pii, pii> > s;

    dsu_pb(int N):N(N), par(N) {
        fors(i, 0, N-1) par[i] = -1;
    }
    int root(int i) {
        if(par[i] < 0) return i;
        return root(par[i]);
    }
    bool join(int i, int j) {
        i = root(i); j = root(j);
        s.push({{i, par[i]}, {j, par[j]}});
        if(i == j) return false;
        if(-par[i] < -par[j]) swap(i, j);
        par[i] += par[j]; par[j] = i;
        return true;
    }

protected:
    void unjoin() {
        assert(!s.empty());
        auto [i, j] = s.top(); s.pop();
        par[i.fi] = i.se; par[j.fi] = j.se;
    }
};

struct dsu_pf : public dsu_pb {
    vector<pair<bool, pii> > st;          // fi==0 -> B
    type, fi==1 -> A type
    vector<pair<bool, pii> > tmp[2];
    int A=0, B=0;
    dsu_pf(int N):dsu_pb(N){}

    bool join(int i, int j) {
        st.pb({0, {i, j}}); B++;
        return dsu_pb::join(i, j);
    }
    void pop_front() {
        assert(!st.empty());
        if(A == 0) {
            forr(i, B) unjoin();
            A = B; B = 0; reverse(all(st));
            for(auto &[b, p]:st) b = 1, dsu_pb::join(p.fi,
            p.se);
        }
        else if(st.back().fi == false) {
            tmp[st.back().fi].pb(st.back()); st.pop_back();
            unjoin();
            while(tmp[0].size() != tmp[1].size() and
            (unsigned) A != tmp[1].size()) {
                tmp[st.back().fi].pb(st.back());
                st.pop_back();
                unjoin();
            }
            for(auto i:{0, 1}) reverse(all(tmp[i]));
            for(auto i:{0, 1}) for(auto v:tmp[i])
                st.pb(v), dsu_pb::join(v.se.fi, v.se.se);
            tmp[0].clear(); tmp[1].clear();
        }
        A--; st.pop_back(); unjoin();
    }
};
```

11.4 Fenwick Tree

```
ll tree[N];
void update(int i,ll x) {
    while(i < N) tree[i] += x, i += i&-i;
}
int query(int i) {
    ll s = 0;
    while(i) s += tree[i], i -= i&-i;
    return s;
}
```

11.5 Segment Tree Generalization

Time Complexity: $\mathcal{O}(\log N)$

```
namespace GMS
{
    template<typename D, D (*join)(D,D), D _e>
    class Segtree {
        class Node {
            Node *l, *r;
            int s,e; D v;
        public:
            Node(int s, int e) :l(0), r(0), s(s), e(e),
            v(_e){};
            ~Node(){delete l; delete r;}

            template<typename Dini>
            friend void init(Node* node, Dini arr[] = NULL)
            {
```

```
        int s = node->s, e=node->e, mid=(s+e)/2;
        if(s == e) {
            node->v = D(arr?arr[s]:_e);
            return;
        }

        node->l = new Node(s, mid);
        init(node->l, arr);
        node->r = new Node(mid+1, e);
        init(node->r, arr);

        node->v = join(node->l->v, node->r->v);
    }
    friend D _query(Node* node, int a, int b) {
        int s=node->s, e=node->e;
        if(a <= s and e <= b) return node->v;
        if(b < s or e < a) return _e;

        return join(_query(node->l, a, b),
        _query(node->r, a, b));
    }
    friend void _update
            (Node* node, int i, function<D(D)>
            upd) {
        int s=node->s, e=node->e;
        if(i < s or e < i) return;
        if(s == e)
        {
            node->v = upd(node->v);
            return;
        }

        _update(node->l, i, upd);
        _update(node->r, i, upd);

        node->v = join(node->l->v, node->r->v);
    }
};
```

```
    Node *root;
public:
    template<typename Dini>
    Segtree(int s,int e, Dini arr[] = NULL) {
        root = new Node(s, e);
        init(root, arr);
    }
    ~Segtree(){delete root;}
    D query(int s, int e)
                      {return _query(root, s, e);}
    void update(int i, function<D(D)> upd)
                      {_update(root, i, upd);}
};


template<typename D, D (*join)(D,D), D _e, typename L, D
(*apply)(D, L, int), L (*give)(L, L), L _l>
class LZSegtree {
    class Node {
        Node *l, *r;
        int s,e;
        D v; L lz;

        void prop() {
            v = apply(v, lz, e-s+1);
            if(l) l->lz = give(l->lz, lz);
            if(r) r->lz = give(r->lz, lz);

            lz = _l;
        }

    public:
        Node(int s, int e)
              :l(0), r(0), s(s), e(e), v(_e), lz(_l){};
        ~Node(){delete l; delete r;}

        template<typename Dini>
        friend void init(Node* node, Dini arr[] = NULL)
        {
```

```
            int s = node->s, e=node->e, mid=(s+e)/2;
            if(s == e)
            {
                node->v = D(arr?arr[s]:_e);
                return;
            }

            node->l = new Node(s, mid);
            init(node->l, arr);
            node->r = new Node(mid+1, e);
            init(node->r, arr);

            node->v = join(node->l->v, node->r->v);
        }
        friend D _query(Node* node, int a, int b) {
            node->prop();
            int s=node->s, e=node->e;
            if(a <= s and e <= b) return node->v;
            if(b < s or e < a) return _e;

            return join(_query(node->l, a, b),
            _query(node->r, a, b));
        }
        friend void _update
           (Node* node, int a, int b, function<L(L)>
           upd){
            node->prop();
            int s=node->s, e=node->e;
            if(b < s or e < a) return;
            if(a <= s and e <= b)
            {
                node->lz = upd(node->lz);
                node->prop();
                return;
            }

            _update(node->l, a, b, upd);
            _update(node->r, a, b, upd);
```

```
            node->v = join(node->l->v, node->r->v);
        }
    };

    Node *root;
public:
    template<typename Dini>
    LZSegtree(int s,int e, Dini arr[] = NULL)
    {
        root = new Node(s, e);
        init(root, arr);
    }
    ~LZSegtree(){delete root;}
    D query(int s, int e){return _query(root, s, e);}
    void update(int s, int e, function<L(L)>
    upd){_update(root, s, e, upd);}
};
} // namespace GMS

//////////////////////////////////////////////////////
#define data _data

struct data {
    int m, m_cnt;
    constexpr data(int m):m(m), m_cnt(1){}
    constexpr data(int m, int m_cnt):m(m), m_cnt(m_cnt){}
};
data join(data A, data B) {
    if(A.m == B.m) return data(A.m, A.m_cnt+B.m_cnt);
    if(A.m < B.m) return A;
    else return B;
}
data apply(data A, int lz, int len)
                     {return {A.m+lz, A.m_cnt};}
int give(int a, int b){return a+b;}

using Seg = GMS::LZSegtree<data, join, {(int)1e9, 0}, int,
apply, give, 0>;
```

11.6 Li-Chao Tree

```
struct Line
{
    ll a=0, b=(ll)2e18+7;
    ll operator()(ll x){return a*x+b;}
    Line():a(0),b((ll)2e18+7){}
    Line(ll a, ll b):a(a), b(b){}
};

ll middle(ll s, ll e){return (s+e+(ll)2e18)/2-(ll)1e18;}
struct Node {
    Node *l=0, *r=0;
    Line v;
    Node():l(0), r(0), v(Line()){}
};
void insert(Node* node, Line v, ll l, ll r, ll s, ll e) {
    ll mid=middle(s, e);
    if(e < l or r < s) return;
    if(s == e) {
        node->v = (node->v(s) < v(s))?node->v:v;
        return;
    }
    if(!node->l) node->l = new Node();
    if(!node->r) node->r = new Node();

    if(l <= s and e <= r) {
        if(node->v(s) >= v(s) and node->v(e) >=
        v(e)){node->v = v; return;}
        if(node->v(s) <= v(s) and node->v(e) <= v(e))
        return;
        insert(node->l, v, l, r, s, mid);
        insert(node->r, v, l, r, mid+1, e);
    }
    else {
        insert(node->l, v, l, r, s, mid);
        insert(node->r, v, l, r, mid+1, e);
    }
}
ll query(Node* node, ll x, ll s, ll e)
{
    if(!node) return (ll)2e18+7;
    if(x < s or e < x) return (ll)2e18+7;
    if(s == e) return node->v(x);

    ll mid = middle(s, e);
    return min({query(node->l, x, s,mid), query(node->r, x,
    mid+1, e), node->v(x)});
}
// 전체 구간을 미리 고정해 놓아야 함.
// (const int L = -1e9, R = 1e9);
// Insert : insert(root, Line 객체, l, r, L, R);
// Query : query(root, x, L, R);
int main() {
    Node *root = new Node();
    getint(n); getint(Q);
    forr(i, n) {
        getll(l); getll(r); getll(a); getll(b);
        insert(root, Line(a,b), l,r-1,(ll)-1e9-7,
        (ll)1e9+7);
    }
    while(Q--) {
        getint(q);
        if(q == 0) {
            getll(l); getll(r); getll(a); getll(b);
            insert(root, Line(a,b), l,r-1,(ll)-1e9-7,
            (ll)1e9+7);
        }
        if(q == 1) {
            getll(x);
            ll ans = query(root, x,(ll)-1e9-7, (ll)1e9+7);
            if(ans == (ll)2e18+7) printf("INFINITY\n");
            else printf("%lld\n", ans);
        }
    }
}
```

11.7 Splay Tree

```
struct Node
```

```cpp
{
    Node *p, *l, *r;
    int cnt;
    ll val; ll m, M, sum; ll lazy;
    bool flip; bool dum;

    Node(ll val, bool dum = false):p(0),l(0), r(0), cnt(1),
    val(val), m(val), M(val), sum(val), lazy(0), flip(0),
    dum(dum){}

    void fix(){
        cnt = 1+(l?l->cnt:0)+(r?r->cnt:0);
        sum = val+(l?l->sum:0)+(r?r->sum:0);
        m = min({val, (l?l->m:inf), (r?r->m:inf)});
        M = max({val, (l?l->M:-1), (r?r->M:-1)});
    }
    void prop(){
        if(flip){
            swap(l, r);
            if(l) l->flip = !l->flip;
            if(r) r->flip = !r->flip;

            flip = false;
        }
        if(lazy){
            val += lazy; sum += cnt * lazy;
            if(l) l->lazy += lazy;
            if(r) r->lazy += lazy;

            lazy = 0;
        }
    }

} *root;

// 자기보다 더 높은 노드를 루트로 하는 SplayTree를 조작하는
경우, 하위 SplayTree는 unvalid된다.
struct SplayTree{
    Node *root = NULL, *rp = NULL;

    SplayTree(){}
    SplayTree(Node *rt){
        if(!rt) return;

        root = rt;
        rp = rt->p;
    }

    void mop(Node *node){
        if(node == root) node->prop();
        else mop(node->p);

        if(node->l) node->l->prop();
        if(node->r) node->r->prop();
    }

    void rotate(Node *node){
        if(!root) return;

        if(node->p == rp) return;
        if(node->p->l == node){
            Node *p = node->p, *g = p->p;
            Node *a = node->l, *b = node->r, *c = p->r;

            p->l = b; if(b) b->p = p;
            p->r = c; if(c) c->p = p;

            node->l = a; if(a) a->p = node;
            node->r = p; p->p = node;

            node->p = g; if(g) (g->l == p?g->l:g->r) = node;

            p->fix(); node->fix();

            if(p == root) root = node;
        }
        else{
            Node *p = node->p, *g = p->p;
            Node *a = p->l, *b = node->l, *c = node->r;

            p->l = a; if(a) a->p = p;
            p->r = b; if(b) b->p = p;

            node->l = p; p->p = node;
            node->r = c; if(c) c->p = node;

            node->p = g; if(g) (g->l == p?g->l:g->r) = node;

            p->fix(); node->fix();

            if(p == root) root = node;
        }

    }

    void splay(Node* node){
        if(!root) return;
        assert(node); mop(node);

        while(node->p != rp){
            Node *p, *g;
            p = node->p; g = p->p;

            if(g == rp) rotate(node);
            else if((p->l == node) == (g->l == p))
            rotate(p), rotate(node);
            else rotate(node), rotate(node);
        }

        root = node;
    }

    Node* insert(ll val, bool dum = false){
        if(!root){
            root = new Node(val, dum);
            return root;
        }
```

```
    else{
        Node *now = root;
        while(now->r) now = now->r;
        Node* ret = now->r = new Node(val, dum);
        now->r->p = now;

        splay(ret); return ret;
    }
}

Node* find_kth(int k) { // 0-indexed
    assert(root);
    assert(root->cnt > k);
    Node *now = root; now->prop();
    while(true){
        while(now->l and now->l->cnt > k) now = now->l,
        now->prop();
        k -= now->l?now->l->cnt:0;

        if(k == 0) break;
        k--; now = now->r;
        now->prop();
    }

    splay(now);
    return now;
}
// s-1, e+1번째 노드가 항상 존재해야 한다.
Node* gather(int s, int e){
    find_kth(e+1);
    SplayTree(root->l).find_kth(s-1);

    assert(root->l->r);
    return root->l->r;
}
void update(int i, int j, ll val){
    Node *node = gather(i, j);

    node->lazy += val; node->prop();
        node->p->fix(); node->p->p->fix();
    }
    void reverse(int i, int j){
        Node *node = gather(i, j);
        node->flip = !node->flip;
    }

    void p_vals(){p_vals(root, 0, false);}
    void p_vals(Node* node, ll lz, bool flip){
        lz += node->lazy; flip ^= node->flip;
        if(!flip){
            if(node->l) p_vals(node->l, lz, flip);
            if(!node->dum) printf("%lld ", node->val+lz);
            if(node->r) p_vals(node->r, lz, flip);
        }
        else{
            if(node->r) p_vals(node->r, lz, flip);
            if(!node->dum) printf("%lld ", node->val+lz);
            if(node->l) p_vals(node->l, lz, flip);
        }
    }
};
```

12 Numerical Analysis

13 Technic

14 Misc

14.1 Fast Input

Usage: Fast Input with fread. Do not use with scanf, cin, or other input function. Use `forr(i, n) read(arr[i]);` instead of `forr(i, n) scanf("%d", arr+i);`. Use `read(s+1)` instead of `scanf("%s", s+1);`.

```
#define getint(n) int n; read(n)
#define getll(n) ll n; read(n)
#define inta getint(a)
#define intab getint(a); getint(b)
char get() {
    static char buf[100000], *S=buf, *T=buf;
    if(S == T) {
        S = buf;T = buf + fread(buf, 1, 100000, stdin);
        if(S == T) return EOF;
    }
    return *S++;
}
void read(int& n) {
    n = 0;
    char c; bool neg = false;
    for(c = get(); c < '0'; c=get()) if(c=='-') neg = true;
    for(;c>='0';c=get()) n = n*10+c-'0';
    if(neg) n = -n;
}
void read(ll& n) {
    n = 0;
    char c; bool neg = false;
    for(c = get(); c < '0'; c=get()) if(c=='-') neg = true;
    for(;c>='0';c=get()) n = n*10+c-'0';
    if(neg) n = -n;
}
int read(char s[]) {
    char c; int p = 0;
    while((c = get()) <= ' ');

    s[p++] = c;
    while((c = get()) >= ' ') s[p++] = c;
    s[p] = '\0';

    return p;
}
```

14.2 MT19937 Random Number

```
const long long rand_L = 1;
const long long rand_R = 10;
mt19937_64
rng(chrono::steady_clock::now().time_since_epoch().count());
uniform_int_distribution<int> dist(rand_L, rand_R);
auto gen = bind(dist, rng);
```

```
gen(); gen(); gen();
```

— Document end —