

목차

헤더 (1)

Math (1-6)

Geometry (6-8)

Flow (8-11)

Graph (11-14)

String (14-18)

Tree (18-19)

Data Structure (19~25)

민규헤더

```
#include<bits/stdc++.h>
#include <bits/extc++.h>
#include<ext/rope>
using namespace std;
using namespace __gnu_cxx;
using namespace __gnu_pbds;
#define fi first
#define se second
#define fastio
ios_base::sync_with_stdio(false);cin.tie(0)
#define fopen freopen("input.txt", "r", stdin)
#define eb emplace_back
#define em emplace
#define prec(a) cout<<fixed;cout.precision(a);
#define all(a) (a).begin(), (a).end()
typedef long long ll;typedef long double ld;typedef
unsigned long long ul;typedef unsigned int
ui;typedef pair<int,int> pii;typedef pair<ll,ll>
pll;typedef complex<double> cpx;
typedef tuple<int,int,int> tiii;
template<class T>
using ordered_set = tree<T, null_type, less<T>,
rb_tree_tag,tree_order_statistics_node_update>;
template<class T>
void pr(T t) {cerr << t << " ";}
template<class T, class ...Args>
void pr(T a, Args ...args) {cerr << a << "
";pr(args...);}
template<class ...Args>
void prl(Args ...args) {pr(args...);cerr << endl;}
const ll INF = 2e18;
const int inf = 2e9;
const double PI = acos(-1);
```

```
pii operator + (pii &a,pii &b){return
pii(a.fi+b.fi, a.se+b.se);}
pii operator - (pii &a,pii &b){return pii(a.fi-
b.fi, a.se-b.se);}
pll operator + (pll &a,pll &b){return
pll(a.fi+b.fi, a.se+b.se);}
pll operator - (pll &a,pll &b){return pll(a.fi-
b.fi, a.se-b.se);}

int main(){
    fastio;

}
```

서용헤더

```
# pragma GCC optimize ("O3")
# pragma GCC optimize ("Ofast")
# pragma GCC optimize ("unroll-loops")
# pragma GCC
target("sse,sse2,sse3,ssse3,sse4,avx,avx2")
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef long double ld;
const ll mod = 1e9+7;
const int MxN = 2210;
const int inf = INT_MAX;
#define REP0(i,n) for(int i=0; i<n; i++)
#define REP1(i,n) for(int i=1; i<=n; i++)
#define REP(i,a,b) for(int i=a; i<=b; i++)
#define sz(v) ((int)(v).size())
#define all(v) (v).begin(), (v).end()
#define compress(v) sort(all(v));
v.erase( unique(all(v)) , v.end() )
#define reset(X) memset(X, 0, sizeof(X))
#define pb push_back
#define fi first
#define se second
#define pii pair<int, int>
#define pll pair<ll, ll>
#define vint vector<int>
#define vll vector<ll>
#define vpai vector<pair<int, int>>
#define vpai vector<pair<ll,ll>>
#define endl "\n"
#define LOG 18
```

희승헤더

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

```
#define eb emplace_back
#define em emplace
#define all(v) v.begin(), v.end()
#define fi first
#define se second
```

```
using namespace std;
```

```
typedef long long ll;
typedef pair <int, int> pii;
typedef pair <ll, ll> pll;
```

```
const int MAX = 2020;
const int INF = 1e9;
const ll LINF = 9e18;
```

Math-1. Fastest FFT

```
const double PI = acos(-1);
typedef complex<double> cpx;
```

```
void FFT(vector<cpx> &a, bool inv) {
    int n = a.size();
    // bit reversal
    for (int i = 1, j = 0; i<n; ++i) {
        int bit = n >> 1;
        while (!(j ^= bit) & bit) bit >>= 1;
        if (i<j) swap(a[i], a[j]);
    }
    for (int i = 1; i<n; i <= 1) {
        double x = inv ? PI / i : -PI / i;
        cpx w = cpx(cos(x), sin(x));
        for (int j = 0; j<n; j += i < 1) {
            cpx p = cpx(1, 0);
            for (int k = 0; k<i; ++k) {
                cpx tmp = a[i + j + k] * p;
                a[i + j + k] = a[j + k] - tmp;
                a[j + k] += tmp;
                p *= w;
            }
        }
    }
    if (inv) {
        for (int i = 0; i<n; ++i) a[i] /= n;
    }
}
```

```
vector<ll> multiply(vector<ll> const& a, vector<ll>
const& b) {
```

```

vector<cpx> fa(a.begin(), a.end()),
fb(b.begin(), b.end());

// 가장 가까운 2^k 꼴의 n 으로 크기를 정해줌.
int n = 1;
while (n < a.size() + b.size())
    n <= 1;
fa.resize(n);
fb.resize(n);

// fft : coefficient rep to value rep :
O(nlogn)
FFT(fa, 0);
FFT(fb, 0);
// value rep 에서 곱하기 : O(n)
for (int i = 0; i < n; i++)
    fa[i] *= fb[i];
//Inverse FFT : value rep to coefficeint
rep :O(nlogn)
FFT(fa, 1);

vector<ll> result(n);
for (int i = 0; i < n; i++)
    result[i] = (ll)round(fa[i].real());
return result;
}

```

Math-2. High-precision FFT

```

const long double PI = acos(-1);
typedef complex<long double> cpx;

void FFT(vector<cpx> &a, bool inv) {
    int n = a.size();
    // bit reversal
    for (int i = 1, j = 0; i < n; ++i) {
        int bit = n >> 1;
        while (!(j ^= bit) & bit) bit >>= 1;
        if (i < j) swap(a[i], a[j]);
    }
    for (int i = 1; i < n; i <= 1) {
        long double x = inv ? PI / i : -PI / i;
        cpx w = cpx(cos(x), sin(x));
        for (int j = 0; j < n; j += i << 1) {
            cpx p = cpx(1, 0);
            for (int k = 0; k < i; ++k) {
                cpx tmp = a[j + j + k] * p;
                a[j + j + k] = a[j + k] - tmp;
            }
        }
    }
}

```

```

        a[j + k] += tmp;
        p *= w;
    }
}
if (inv) {
    for (int i = 0; i < n; ++i) a[i] /= n;
}
}

vector<ll> multiply_high_precision(vector<ll>
const& a, vector<ll> const& b) {

    // 1023 = 2^10-1 : select this value to be near
sqrt(Max possible coefficient)

    // 가장 가까운 2^k 꼴의 n 으로 크기를 정해줌.
    int n = 1; while (n < a.size() + b.size()) {n
<= 1;}
    // a(i) = amsb(i) * cut + alsb(i)
    vector<cpx> amsb(n), alsb(n), bmsb(n), blsb(n);
    for(int i=0 ; i<sz(a);i++){amsb[i] =
a[i]>>10;alsb[i]=a[i]&1023;}
    for(int i=0 ; i<sz(b);i++){bmsb[i] =
b[i]>>10;blsb[i]=b[i]&1023;}

    // fft : coefficient rep to value rep :
O(nlogn)
FFT(amsb,0); FFT(alsb,0); FFT(bmsb,0);
FFT(blsb,0);
vector<cpx> high(n), middle(n), low(n);

    // value rep 에서 곱하기 : O(n) amsb*bmsb*2^20 +
(amsb*blsb+alsb*bmsb)*2^10 + alsb*blsb
    for (int i = 0; i < n; i++)
    {
        high[i] = amsb[i]*bmsb[i];
        middle[i] =
amsb[i]*blsb[i]+alsb[i]*bmsb[i];
        low[i] = alsb[i]*blsb[i];
    }
    //Inverse FFT : value rep to coefficeint
rep :O(nlogn)
FFT(high, 1);FFT(middle, 1);FFT(low, 1);
//Summation ->Result
vector<ll> result(n);
for (int i = 0; i < n; i++)
{
    ll h = (ll)round(high[i].real());
    ll m = (ll)round(middle[i].real());
    ll l = (ll)round(low[i].real());
    result[i] = (h<<20) + (m<<10)+l;
}
}

```

```

}

return result;
}

```

Math-3. Miller-Rabin

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod = 1e9+7;
const int MxN = 1e5+1;
const int MxK = 1e9;
const int inf = INT_MAX;
#define REP0(i,n) for(int i=0; i<n; i++)
#define REP1(i,n) for(int i=1; i<=n; i++)
#define REP(i,a,b) for(int i=a; i<=b; i++)
#define pb push_back
#define fi first
#define se second
#define reset(X) memset(X, 0, sizeof(X))
#define pii pair<int, int>
#define endl "\n"
#define vint vector<int>
#define pll pair<ll, ll>
#define vll vector<ll>
#define LOG 18 //enough for 10^5

//O(numrep * (Logn)^3 ).
typedef __int128 lll;
ll modpow(int x, ll y, ll mod)
{
    if (y == 0) return 1;
    lll ans = modpow(x,y/2,mod);
    ans = (ans*ans)%mod;
    if (y%2 == 1) ans = (ans*x)%mod;
    return ans;
}

bool miillerTest(ll d, ll n, int a)
{
    // Compute a^d % n
    lll x = modpow(a, d, n);

    if (x == 1 || x == n-1)
        return true;

    while (d != n-1)
    {

```

```

        x = (x * x) % n;
        d *= 2;

        if (x == 1)    return false;
        if (x == n-1)  return true;
    }
    return false;
}

bool Miller_Rabin(ll n, int numrep)
{
    if (n <= 1 || n == 4) return false;
    if (n <= 3) return true;

    ll d = n - 1;
    while (d % 2 == 0) d /= 2;

    // 하나라도 false 가 뜨면 바로 합성수. 합성수들을
    // 걸러내는 작업으로 보이면됨.
    int a[]={2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
31, 37}; //n 이 작은 Longlong 일때 결정적으로
// 판별함.
    for (int i = 0; i < numrep; i++)
    {
        if(n==a[i])return true;
        if (!miillerTest(d, n,a[i]))return false;
    }
    return true;
}

int main()
{
    for(int i=1; i<100; i++)
    {
        if(Miller_Rabin(i,12))cout<<i<<endl;
    }
}

```

Math-4. NTT

```

/*
민규 헤더 생략
*/

```

```

class NTT{
    //2^23크기까지 가능
    const ll mod = 998244353, w = 3;

public:

```

```

    ll Pow(ll x, ll y) {
        ll r = 1;
        while (y) {
            if (y & 1) r = r * x % mod;
            x = x * x % mod;
            y >>= 1;
        }
        return r;
    }

    void FFT(vector<ll> &a, bool f) {
        int N = a.size();
        ll i, j=0, k, x, y, z;
        for (i = 1; i < N; i++) {
            for (k = N >> 1; j >= k; k >>= 1) j -=
k;

            j += k;
            if (i < j) swap(a[i], a[j]);
        }
        for (i = 1; i < N; i <= 1) {
            x = Pow(f ? Pow(w, mod - 2) : w, mod /
i >> 1);

            for (j = 0; j < N; j += i <= 1) {
                y = 1;
                for (k = 0; k < i; k++) {
                    z = a[i | j | k] * y % mod;
                    a[i | j | k] = a[j | k] - z;
                    if (a[i | j | k] < 0) a[i | j |
k] += mod;

                    a[j | k] += z;
                    if (a[j | k] >= mod) a[j | k] -=
mod;

                    y = y * x % mod;
                }
            }
            if (f) {
                j = Pow(a.size(), mod - 2);
                for (i = 0; i < N; i++) a[i] = a[i] *
j % mod;
            }
        }
        vector<ll> mult(vector<ll> a, vector<ll> b){
            int n = 1;
            while(n < a.size() + b.size() - 1) n<<=1;
            a.resize(n);
            b.resize(n);
            vector<ll> c(n);
            FFT(a, false);
            FFT(b, false);
            for(int i=0; i<n; i++) c[i] = a[i] * b[i] %
mod;

```

```

        FFT(c, true);
        return c;
    }
}Ntt;

int main() {
    vector<ll> a(3,1), b(3,1);
    vector<ll> c = Ntt.mult(a,b);
    for(auto i:c) cout<<i<<" "; //1 2 3 2 1 0 0 0
}

```

Math-5. Pollard's rho algorithm

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod = 1e9+7;
const int MxN = 1e5;
const int MxK = 1e9;
const ll inf = INT_MAX;
#define REP0(i,n) for(int i=0; i<n; i++)
#define REP1(i,n) for(int i=1; i<=n; i++)
#define REP(i,a,b) for(int i=a; i<=b; i++)
#define pb push_back
#define fi first
#define se second
#define reset(X) memset(X, 0, sizeof(X))
#define pii pair<int, int>
#define endl "\n"
#define vint vector<int>
#define pll pair<ll, ll>
#define vll vector<ll>
#define LOG 18 //enough for 10^5
typedef __int128 lll;

//Find a divisor that divides n in under than
O( sqrt(minP) ) <= O( N^1/4 )
//code works for Long Long n
long long gcd(long long a, long long b)
{
    if(b==0)return a;
    return gcd(b, a%b);
}
long long f(long long x, long long c, long long
mod)
{
    return ((lll)x*x%mod+c)%mod;
}

```

```

}
long long PollardRho(long long n)
{
    /* initialize random seed */
    //srand ((unsigned int)time(NULL));

    /* no prime divisor for 1 */
    if (n==1) return n;

    /* even number means one of the divisors is 2
*/
    if (n % 2 == 0) return 2;

    long long x = (rand()%(n-2))+2;
    long long y = x;
    long long c = (rand()%20)+1;

    /* Initialize candidate divisor (or result) */
    long long d = 1;

    /* until the prime factor isn't obtained.
    If n is prime, return n */
    while (d==1)
    {
        x=f(x,c,n);
        y=f(y,c,n);
        y=f(y,c,n);

        /* check gcd of |x-y| and n */
        d = gcd(abs(x-y),n);

        if (d==n)
        {
            x = (rand()%(n-2))+2;
            y = x;
            c = (rand()%20)+1;
            d = 1;
        }
    }

    return d;
}

int main()
{
    ll x= mod*mod;
    cout<<x<<endl;
    cout<<PollardRho(x)<<endl;
}

```

Math-6. Basic Math

```

/*
-density of primes :  $O(N/\ln N)$ 
-goldbach's conjecture : even number = sum of two primes
-twin prime conjecture :  $p, p+2$  prime
-legendre's conjecture : prime exists between  $n^2, (n+1)^2$ 
-Lagrange's theorem: positive int = sum of four squares
-Zeckendorf's theorem : positive int : unique sum of Fibonacci where no numbers are same and no numbers are consecutive Fibonacci
-Pythagorean triple :  $0 < m < n, (n, m) = 1, n$  or  $m$  is even :  $(n^2 - m^2, 2nm, n^2 + m^2)$  produces all pythagorean triple.
-Fermat's theorem : if  $p$ : prime,  $(x, p) = 1$ , then  $x^{(p-1)} \% p = 1$ 
-Euler's theorem : if  $(x, m) = 1, x^{(\phi(m))} \% m = 1$ 
-wilson's theorem :  $p$  is prime  $\leftrightarrow (p-1)! \% p = p-1$ 
-Burnside Lemma :
-The number of orbits are equal to the average number of fixed points int the group action(taken over all the group elements)
*/

//modular inverse : if  $(x, m) = 1$ , inverse of  $x$  modulo  $m = x^{(\phi(m)-1)}$ 
// $O(\sqrt{\text{mod}})$ 
int inverse(int x, int mod)
{
    if(gcd(x, mod) != 1) return 0; //doesn't exist.
    return modpow(x, phi(mod)-1, mod);
    // return modpow(x, mod-2, mod) if mod is prime.
}

//general eqn = particular sol +  $\{kb/g, -ka/g\}$ ,
particular sol = Extended *  $c/g$ .
//Extended euclidean algorithm : solve  $ax+by = \text{gcd}(a, b)$ ;
// $O(\log(\min(a, b)))$ 
pii Extended(int a, int b)
{
    if (a == 0) return {0, 1};

    int x, y;

```

```

    pii p = Extended(b%a, a);
    x=p.first; y=p.second;

    return {y - (b/a) * x, x};
}
//find modular inverse in  $O(\log \text{mod})$ 
int inverse_Euclidean(int x, int mod)
{
    return Extended(x, mod).fi;
}

//Chinese remainder theorem :  $O(N \log M)$  if we use extended euclidean algorithm for calculating inverse,  $O(\sqrt{M})$  if we use phi function for inverse.
// find  $x$  st.  $a_1(\text{mod } m_1) \& a_2(\text{mod } m_2) \& \dots \& a_n(\text{mod } mn)$ .  $m_i$ 's are coprime

int Chinese(vector<int> modular, vector<int> remainder)
{
    // Compute product of all numbers
    int total_product = 1;
    int n = (int)modular.size();

    for (int i = 0; i < n; i++)
        total_product *= modular[i];

    // Initialize result
    int result = 0;

    // Apply formula
    for (int i = 0; i < n; i++) {
        int pp = total_product / modular[i];
        result += remainder[i] * Extended(pp, modular[i]).fi * pp;
        result %= total_product;
    }

    return result % total_product; //result + total_product*k 끝은 모두 해임
}

/* nCr calculation */

//Solution 1. <just dp :  $O(n^2)$ >
long long nCr(int n, int r, int mod)
{
    if(n < r) return 0;
    if(n == 0 && r == 0) return 1;

```

```

    long long dp[n+1][n+1];
    for(int i=1; i<=n; i++){dp[i][0]=1;
dp[i][i]=1;}
    for(int i=2; i<=n; i++)
    {
        for(int j=1; j<i; j++)
        {
            dp[i][j]= dp[i-1][j-1]+dp[i-1][j];
            dp[i][j]%mod;
        }
    }
    return dp[n][r];
}

//Solution 2. <페르마 소정리. inverse.O(n)>
long long modpow(int x, int y, int mod)
{
    if (y == 0) return 1;
    long long ans = modpow(x,y/2,mod);
    ans = (ans*ans)%mod;
    if (y%2 == 1) ans = (ans*x)%mod;
    return ans;
}
long long inverse(int x, int mod) {return modpow(x,
mod-2, mod);}

long long nCr_Fermat(int n, int r, int mod) //mod
is prime!!
{
    if (n < r) return 0;
    if (r == 0)return 1;

    long long fac[n + 1]; fac[0] = 1;
    for (int i = 1; i <= n; i++) fac[i] = (fac[i -
1] * i) % mod;

    return (fac[n] * inverse((int)fac[r], mod) %
mod * inverse((int)fac[n - r], mod) % mod) % mod;
}

//Solution 3. <Lucas Theorem. 나누는 수가 작을때
사용.O(p^2logp) >
// nCr%p 는 n 과 r 을 p 진법으로 바꾸고 각각의
niCri 들의 곱과 같다.

long long nCr_Lucas(int n, int r, int p)
{
    if (r==0) return 1;
    // Compute last digits of n and r in base p
    int ni = n%p, ri = r%p;

```

```

    return nCr_Lucas(n/p, r/p, p) * nCr(ni, ri, p)%
p;
}

```

Math-7. More Math

```

typedef long long C;
typedef complex<C> Point;
#define X real()
#define Y imag()

C cross_product(Point a, Point b)
{
    return (conj(a)*b).Y;
}

int orientation(Point l1, Point l2, Point p)
{
    C val = cross_product(p-l1, p-l2); //양수면 l1-
>l2 선분에 대해 왼쪽, 음수면 오른쪽, 0이면 선분위
if(val>0)return 1;
if(val==0)return 0;
return -1;
}

bool online(Point l1, Point l2, Point
p) //collinear 한 세점이 주어졌을때 p 는 l1, l2
사이에 있는가?
{
    C left = l1.X; C right=l2.X;
    C down =l1.Y; C up =l2.Y;
    if(left>right)swap(left, right);
    if(down>up)swap(down, up);
    return (left<=p.X && p.X<=right && down<=p.Y &&
p.Y<=up);
}

bool intersect(Point p1, Point q1, Point p2, Point
q2) //does line segments intersect with each
other??
{
    if(orientation(p1,q1,p2)!=orientation(p1,q1,q2)
&&
orientation(p2,q2,p1)!=orientation(p2,q2,q1))return
true;
    else{
        if(orientation(p1,q1,p2)==0 &&
orientation(p1,q1,q2)==0)//네점 collinear
        {

```

```

            if (online(p1, q1, p2) ||
online(p1,q1,q2) || online(p2,q2,p1) ||
online(p2,q2,q1))return true; //무한히 많은 교점
        }
        return false;
    }
}

////////////////////////////////////
////////////////////////////////////
// Finding inverse = making RREF = Solving Linear
eqn ...

//int -> double & remove p& modulo op for ordinary
cases
void GaussJordan(vector<vint> & A, vector<vint> &
invA, int n, int p, vint & inverse)
{
    int curr=0;
    for(int j=0;j<n;j++)
    {
        //find first row with nonzero component in
jth column
        int index=-1;
        for(int i=curr;
i<n;i++)if(A[i][j]!=0){index = i;break;}
        if(index==-1)continue;
        //swap that row
        swap(A[curr], A[index]);
        swap(invA[curr] , invA[index]);
        //make leading 1
        int divide = inverse[A[curr][j]];
        for(int k=j;k<n;k++)A[curr][k] = A[curr][k]
* divide % p;
        for(int k=0; k<n;k++)invA[curr][k] =
invA[curr][k] * divide % p;
        //update the upper & lower rows
        for(int i=0; i<n;i++)
        {
            if(i!=curr){
                int mult = A[i][j];
                for(int k =j;k<n;k++)A[i][k] =
((A[i][k] - A[curr][k] *mult)%p+p)%p;
                for(int k=0;k<n;k++)invA[i][k] =
((invA[i][k] - invA[curr][k] *mult) %p+p)%p;
            }
            curr++;
        }
    }
}

```

```
//finding inverse in modulo p
vector<vint> inverse(vector<vint> & A , int p)
{
    int n = sz(A);
    // identity matrix invA
    vector<vint> invA;
    for(int rep=0; rep<n;rep++)
    {
        vint temp(n);
        invA.pb(temp);
    }
    REP0(i,n)invA[i][i]=1;
    //preprocess inverse in moudo p
    vint inverse(p);
    inverse[1] =1;
    for(int i=2; i<p;i++)inverse[i] = p -
(p/i)*inverse[p%i]%p;

    //Gauss Jordan elimination
    GaussJordan(A, invA,n,p, inverse);
    return invA;
}
```

Geometry-1. Bulldozer

```
/*
희승 헤더 생략
*/

struct Point {
    int x, y;
    ll val;
    bool operator < (const Point &p) const {
        return pii(x, y) < pii(p.x, p.y);
    }
};

struct Evt{
    int dx, dy, idx1, idx2;
    Evt(int dx, int dy, int idx1, int idx2) :
    dx(dx), dy(dy), idx1(idx1), idx2(idx2) {}
    bool operator < (const Evt &s)const{
        ll k = 1ll * dx * s.dy - 1ll * dy * s.dx;
        if(k == 0) return pii(idx1, idx2) < pii(s.idx1,
s.idx2);
        return k > 0;
    }
};

ll ccw(Evt a, Evt b) {
    return (1ll)a.dx * b.dy - (1ll)b.dx * a.dy;
```

```
}

int main() {
    ios::sync_with_stdio(false); cin.tie(nullptr);

    int n;
    cin >> n;
    vector <Point> a(n);
    vector <int> pos(n);

    for(int i = 0; i < n; i++) {
        cin >> a[i].x >> a[i].y >> a[i].val;
    }

    if(n == 1) {
        // 예외처리
        return 0;
    }
    sort(all(a));
    for(int i = 0; i < n; i++) {
        pos[i] = i;
    }

    vector <Evt> line;
    for(int i = 0; i < n; i++) {
        for(int j = i + 1; j < n; j++) {
            line.eb(a[j].x - a[i].x, a[j].y -
a[i].y, i, j);
        }
    }
    sort(all(line));

    // modify data structures
    ll ans = 0;
    for(int i = 0; i < line.size(); i++) {
        int rx = pos[line[i].idx1], ry =
pos[line[i].idx2];
        swap(a[rx], a[ry]);
        swap(pos[line[i].idx1], pos[line[i].idx2]);
        // modify data structures
        if(i + 1 < line.size() && ccw(line[i],
line[i+1]) == 0) continue;
        //modify ans
    }
    cout << ans << endl;
}
```

Geometry-2. ConvexHull

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

```
using namespace std;
#define fi first
#define se second
typedef long long ll;
typedef pair<ll,ll> pll;

//coding by red1108
ll cross(pll a, pll b){return a.fi*b.se-a.se*b.fi;}
vector<pll> convex_hull(vector<pll> feed){
    sort(feed.begin(), feed.end());
    sort(feed.begin()+1, feed.end(), [&](pll a, pll
b){
        ll t = cross({a.fi-feed[0].fi,a.se-
feed[0].se},{b.fi-feed[0].fi,b.se-feed[0].se});
        if(t==0) return a.fi<b.fi;
        return t>0;
    });
    vector<pll> ret;
    for(pll i:feed){
        while(ret.size()>1&&cross({ret[ret.size()-
1].fi-ret[ret.size()-2].fi,ret[ret.size()-1].se-
ret[ret.size()-2].se},{i.fi-ret[ret.size()-
1].fi,i.se-ret[ret.size()-1].se})<=0)
ret.pop_back();
        ret.push_back(i);
    }
    pll i=feed[0];
    while(ret.size()>2&&cross({ret[ret.size()-
1].fi-ret[ret.size()-2].fi,ret[ret.size()-1].se-
ret[ret.size()-2].se},{i.fi-ret[ret.size()-
1].fi,i.se-ret[ret.size()-1].se})<=0)
ret.pop_back();
    return ret;
}
// 민규야... 컨벡스헐 이렇게 드럽게 짜니 내 코드 보셈

int main(){
    vector<pll> test;
    test.emplace_back(1,1);
    test.emplace_back(0,0);
    test.emplace_back(0,2);
    test.emplace_back(2,0);
    test.emplace_back(2,2);
    test = convex_hull(test);
    for(auto i:test) cout<<i.first<<
"<<i.second<<endl;
}
```


Geometry-3. Rotating_Callipus

```
/*
회승 헤더 생략
*/

const int MAX = 101010;
const int INF = 1e9;
const ll LINF = 9e18;

ll ccw pll a, pll b, pll c {
    return (b.fi - a.fi) * (c.se - a.se) - (b.se - a.se) * (c.fi - a.fi);
}

ll ccw2(pll a, pll b, pll c, pll d) {
    return (b.fi - a.fi) * (d.se - c.se) - (b.se - a.se) * (d.fi - c.fi);
}

void make_convex_hull(vector<pll> &A, vector<pll> &CH) {
    sort(all(A), [&](pll a, pll b) {
        return a < b;
    });
    sort(next(A.begin()), A.end(), [&A](pll a, pll b) {
        return ccw(A[0], a, b) > 0;
    });
    for(auto i : A) {
        while(CH.size() >= 2 && ccw(CH[CH.size()-2], CH[CH.size()-1], i) <= 0) CH.pop_back();
        CH.eb(i);
    }
}

double dist(pll a, pll b) {
    return sqrt((a.fi - b.fi) * (a.fi - b.fi) + (a.se - b.se) * (a.se - b.se));
}

double ldist(pll a, pll b1, pll b2) {
    return abs((ll)(a.fi - b1.fi) * (b2.se - b1.se) - (ll)(a.se - b1.se) * (b2.fi - b1.fi)) / dist(b1, b2);
}

double farthest(vector<pll> CH) {
    int idx = 0;
    double ret = 0.0;
    for(int i = 0; i < CH.size(); i++) {
```

```
        while(idx + 1 < CH.size() && ccw2(CH[i], CH[i+1], CH[idx], CH[idx+1]) >= 0) {
            ret = max(ret, dist(CH[i], CH[idx]));
            idx++;
        }
        ret = max(ret, dist(CH[i], CH[idx]));
    }
    return ret;
}

int main() {
    ios::sync_with_stdio(false); cin.tie(nullptr);
    cout.precision(9); cout << fixed;
    int n;
    cin >> n;
    vector<pll> A(n);
    for(int i = 0; i < n; i++) {
        cin >> A[i].fi >> A[i].se;
    }
    vector<pll> CH;
    make_convex_hull(A, CH);
    cout << farthest(CH) << endl;
}
```

Flow-0. Flow summary

Dinic : $\min(EF, V^2E)$ //very pessimistic
// 바깥쪽 bfs 가 $O(V)$ 번 돌고, 주어진 level graph 에서 $O(E)$ 번 dfs 가 돌면서 각 level graph 는 $O(VE)$ 안에 처리됨.
//in unit capacity : $\min(V^{2/3} * E, E^{3/2})$
//in unit network : $O(EV^{1/2}) == \text{Hopcroft-Karp}$

//adj 는 양방향으로 연결, residual 은 진짜 간선있는 곳에만 update, adj/residual testcase 마다 초기화, $M \times N$ 값 vertex 보다 크게 설정.
// source sink 가 아닌 두 점 사이에서 flow 를 새로 흘리면 우회 경로를 찾을 수 있음

//Maxflow = Mincut // residual 이 0 인 놈들 기준으로 node 를 나누면 됨, 이후 실제로 있는 간선들 중에서 도달가능-도달 불가능 페어를 찾으면 됨.
//disjoint path(#of edge-disjoint path = maxflow, vertex->vertex 선분들을 만든 후, #of vertex-disjoint path = maxflow)

-Bipartite matching
// Find maximum matching in bipartite graph

// Hall's theorem : perfect matching \leftrightarrow for arbitrary set X , $|X| \leq |f(X)|$
//Konig's theorem : Minimum node cover = Maximum matching
// Maximum independent set + Minimum node cover = V
//find minimum Node-disjoint path cover & find minimum general path cover

-Mincost Maxflow : find the minimum cost among the maxflows.
//Maxcost Maxflow
//cost 는 양방향으로(양수, 음수) 입력, adj,residual,cost 초기화.
//양방향 mcmf(두 노드 사이의 edge 가 cost 다른 경우) : 정점분할 필수

-Global Mincut, all pairs maximum flow.

-Circulation problem
//Flow with demands : 각 정점이 demand(공급 - 방출량)을 가지는 경우.
//1. 모든 vertex 에 대해 demand 의 합은 0 임을 확인
//2. demand 가 음수인 놈 에 newsource 가 공급, demand 가 양수인 놈은 newsink 에 공급 후 maxflow 값이 demand 합과 같은지 확인

//유량의 하한이 존재하는 경우
//sink->source INF 간선 만들고 진행.
//u->v [L,R] : 1.change capacity to R-L, demand(u) +=L, demand(v) -=L
//복원시에는 L 값을 다시 더해줘야함.
//maxflow 구하고 싶으면 flow with demands 풀고 원래 source 에서 sink 로 찾을 수 있을만큼 찾기.

Flow-1. Dinic

```
/*
서용 헤더 생략
MxN = 2210
*/
int residual[MxN][MxN];
vint adj[MxN];

int level[MxN];
int savepoint[MxN];
```

```

bool bfs(int n, int source, int sink)    //flow
더 보내기 가능?
{
    REP1(i,n)level[i]=-1;
    level[source]=0;

    queue<int> q;
    q.push(source);

    while(!q.empty())
    {
        int curr = q.front(); q.pop();
        for (auto next : adj[curr])
        {
            if (residual[curr][next]>0 &&
level[next]==-1)
            {
                level[next]=level[curr]+1;
                q.push(next);
            }
        }
    }
    return (level[sink]!=-1); // is it possible to
reach the sink??
}

int dfs(int curr, int destination, int
flow)    //curr -> destination 까지 flow 를 넘기는
하나의 path 찾기. 그 과정에서 residual update
{
    if(curr==destination)return flow;

    for (int &i= savepoint[curr];
i<adj[curr].size(); i++)    //look for all adjacent
edges
    {
        int next = adj[curr][i];
        if(level[next]==level[curr]+1 &&
residual[curr][next]>0)    //Level 이 1증가하고
이동가능(residual>0) 하면. 즉 level graph 상에서
edge 가 있으면 가본다.
        {
            int possibleflow = dfs(next,
destination, min(flow, residual[curr][next]));

            if(possibleflow>0)    //경로를
찾았다!!!!!!
            {

```

```

                residual[curr][next] -=
possibleflow;    //residual graph update
                residual[next][curr] +=
possibleflow;
                return possibleflow;    // 한번
dfs(source, sink) 부를때마다 하나의 path 를 따라서
flow 를 보낸다. 이과정을 하나의 level graph 에 대해
불가능할때까지 반복.
            }
        }
    }
    return 0; //no possible flow from source to
sink in current level graph.
}

int Dinic(int n, int source, int sink)
{
    int maxflow=0;

    while(bfs(n, source, sink))    //there exits
path that can flow
    {
        reset(savepoint);    //we are going to
continuously work on a single level graph, so we
need to store the edges that we have traveled
before.

        while(true)    //for a single level graph
        {
            int flow = dfs(source, sink, inf);
            maxflow += flow;
            if(flow==0)break;
        }
    }
    return maxflow;
}

int demand[MxN];

void solve()
{
    int m,n,p, pp; cin>>m>>n>>p>>pp;
    pii q[n+1];
    REP1(i,n)
    {
        int x,y; cin>>x>>y; q[i] = {x,y};
    }
    vector<pair<int, pii> > r[m+1];
    int totalk =0;
    REP1(i,m)

```

```

{
    int k; cin>>k;
    totalk += k;
    while(k--)
    {
        int d,rp,rpp;cin>>d>>rp>>rpp;
        r[i].pb({d,{rp,rpp}});
    }
}

int source = n+m+totalk+1;
int sink = n+m+totalk+2;
int newsource = n+m+totalk+3;
int newsink =n+m+totalk+4;

//sink -> source : inf
adj[sink].pb(source);
adj[source].pb(sink);
residual[sink][source]=inf;
//source -> members
REP1(i,m)
{
    adj[source].pb(i);adj[i].pb(source);residua
l[source][i]=pp-p;
    demand[source] += n-pp;
    demand[i] -= (n-pp);
}
//days -> sink
REP(i, m+1, m+n)
{
    adj[i].pb(sink); adj[sink].pb(i);
    residual[i][sink]=q[i-m].se-q[i-m].fi;
    demand[i] += m-q[i-m].se;
    demand[sink] -= m-q[i-m].se;
}
//middle nodes
int curr = m+n+1;
REP1(i,m)
{
    for(auto x : r[i])
    {
        adj[i].pb(curr);
        adj[curr].pb(curr);residual[i][curr] = x.se.se-
x.se.fi+1-x.fi;
        demand[i] += x.fi;
        demand[curr] -=x.fi;

        REP(point, x.se.fi,x.se.se)
        {
            adj[curr].pb(point+m);adj[point+m].
pb(curr);residual[curr][point+m]=1;
        }
    }
}

```



```

        curr++;
    }
}
int sumofdemand =0;
REP1(i,n+m+totalk+2)
{
    if(demand[i]>0)
    {
        sumofdemand += demand[i];
        adj[i].pb(newsink);adj[newsink].pb(i);r
        esidual[i][newsink]=demand[i];
    }
    else{
        adj[newsource].pb(i);adj[i].pb(newsourc
        e);residual[newsource][i]=-demand[i];
    }
}

curr=n+m+1;

if( sumofdemand == Dinic(n+m+totalk+4,
newsource, newsink) )
{
    cout<<1<<endl;
    REP1(i,m)
    {
        vint v;
        for(auto x : r[i])
        {
            REP(point, x.se.fi, x.se.se)
            {
                if(residual[curr][m+point]==0)
                {
                    v.pb(point);
                }
            }
            curr++;
        }
        cout<<sz(v)<<" ";
        for(auto x: v)cout<<x<<" ";
        cout<<endl;
    }
}
else cout<<-1<<endl;
}

```

```

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t; t=1;
    while(t-->0)
    {
        solve();
    }
}

```

Flow-2. MCMF

```

/*
서용 헤더 생략
MxN = 2020
*/
int residual[MxN][MxN];
vint adj[MxN];
int cost[MxN][MxN];

int dist[MxN];
int parent[MxN];
bool inQ[MxN];

bool SPFA(int n, int source, int sink)
{
    REP1(i,n)
    {
        parent[i]=-1;
        dist[i]=inf;
        inQ[i]=false;
    }
    dist[source]=0;
    parent[source]=0;
    queue<int> q;
    q.push(source);
    inQ[source]=true;

    while(!q.empty())
    {
        int curr = q.front(); q.pop();
        inQ[curr]=false;
        for (auto next : adj[curr])
        {
            if (residual[curr][next]>0 &&
            dist[next]>dist[curr]+cost[curr][next]) //if

```

```

there is path(in adj& residual>0) and distance can
be updated
        {
            dist[next]=dist[curr]+cost[curr][ne
            xt]; //update dist
            parent[next]=curr;
            //update parent

            if(!inQ[next]) //if next is not
            in Q we add it.
            {
                q.push(next);
                inQ[next]=true;
            }
        }
    }
    return (parent[sink]!=-1); //경로가 있는지
    없는지 return
}

pii MCMF(int n, int source, int sink)
{
    int maxflow=0;
    int mincost=0;

    while(SPFA(n,source, sink)) //there exists
    path that can flow
    {
        //calculate flowofpath
        int flowofpath=INT_MAX;
        int curr = sink;
        while(curr!=source)
        {
            flowofpath = min(flowofpath,
            residual[parent[curr]][curr]);
            curr = parent[curr];
        }
        //update residual graph, maxflow, mincost
        curr=sink;
        while(curr!=source)
        {
            mincost +=
            cost[parent[curr]][curr]*flowofpath;
            residual[parent[curr]][curr] -=
            flowofpath;
            residual[curr][parent[curr]] +=
            flowofpath;
            curr = parent[curr];
        }
        //update maxflow
        maxflow += flowofpath;
    }
}

```

```

    }
    return {mincost, maxflow}; //pair of mincost,
maxflow
}

void solve()
{
    int e,v;
    while(cin>>v>>e)
    {

        int source = 2*v+1;
        int sink = 2*v+2;
        //source to 1
        adj[source].pb(1);
        adj[1].pb(source);
        residual[source][1] =2;
        cost[source][1] =0;
        cost[1][source]=0;
        // 2*v to sink
        adj[2*v].pb(sink);
        adj[sink].pb(2*v);
        residual[2*v][sink]=2;
        cost[2*v][sink]=0;
        cost[sink][2*v] =0;

        //given edges

        REP1(i,e){
            int from,to, w; cin>>from>>to>>w;

            adj[from+v].pb(to);
            adj[to].pb(from+v);
            residual[from+v][to]=1;
            cost[from+v][to] = w;
            cost[to][from+v] = -w;

        }

        //node to node
        REP(i,1, v)
        {
            adj[i].pb(i+v);
            adj[i+v].pb(i);
            if(i==1 || i==v)residual[i][i+v]=2;
            else residual[i][i+v]=1;
            cost[i][i+v]= 0;
            cost[i+v][i]=0;
        }
        cout<<MCMF(2*v+2, source ,sink).fi<<endl;
    }
}

```

```

        REP1(i, 2*v+2)
        {
            adj[i].clear();
            REP1(j, 2*v+2)
            {
                residual[i][j] = 0;
                cost[i][j]=0;
            }
        }
    }

}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t; t=1;
    while(t--)
    {
        solve();
    }
}

```

Flow-3. Mincut

```

/*
<Maxflow Mincut Theorem>

Max-flow = Min-cut
*/

int residual[MxN][MxN];
vint adj[MxN];
vint realadj[MxN]; // undirected 이면 상관없음,
directed 이면 실제 간선만을 적어줘야함.

void dfs_findcut(int curr, bool canvisit[])
{
    canvisit[curr]=true;
    for(auto next : adj[curr])
    {

```

```

        if (residual[curr][next]>0 &&
canvisit[next]==false)
        {
            dfs_findcut(next, canvisit);
        }
    }
}

vector<pii> MinCut(int n, int source)
{
    vector<pii> cuts;

    bool canvisit[n+1];
    reset(canvisit);

    dfs_findcut(source, canvisit);

    REP1(startvertex,n){
        for (auto endvertex :
realadj[startvertex]) //실제 간선들 중에서
print 해야함.
        {
            if
(canvisit[startvertex]&& !canvisit[endvertex])
cuts.pb({startvertex, endvertex});
        }
    }
    return cuts;
}

```

Flow-4. Edmonds-Karp algo

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod = 1e9+7;
const int MxN = 410;
const int MxK = 1e9;
const ll inf = INT_MAX;
#define REP0(i,n) for(int i=0; i<n; i++)
#define REP1(i,n) for(int i=1; i<=n; i++)
#define REP(i,a,b) for(int i=a; i<=b; i++)
#define pb push_back
#define fi first
#define se second
#define reset(X) memset(X, 0, sizeof(X))
#define pii pair<int, int>
#define endl "\n"
#define vint vector<int>

```

```
#define pll pair<ll, ll>
#define vll vector<ll>
#define LOG 18 //enough for 10^5
//Edmonds Karp algorithm : O(min(E^2 V),
min(EF)) : a improved version of ford-fulkerson
```

```
int residual[MxN][MxN];
vint adj[MxN];
```

```
bool visited[MxN];
int parent[MxN];
```

```
bool bfs(int source, int sink)
{
    reset(visited);
    visited[source]=true;
    parent[source]=0;
    queue<int> q;
    q.push(source);

    while(!q.empty())
    {
        int curr = q.front(); q.pop();
        for (auto next : adj[curr])
        {
            if (residual[curr][next]>0 &&
visited[next]==false)
            {
                visited[next]=true;
                parent[next]=curr;
                q.push(next);
                if(next==sink) return
true; //possible to reach sink;
            }
        }
    }
    return false; //no path to reach sink
}
```

```
int Edmonds_Karp(int source, int sink)
{
    int maxflow=0;

    while(bfs(source, sink)) //there exists path
that can flow
    {
        //calculate flowofpath
        int flowofpath=INT_MAX;
        int curr = sink;
        while(curr!=source)
        {
```

```
            flowofpath = min(flowofpath,
residual[parent[curr]][curr]);
            curr = parent[curr];
        }
        //update residual graph
        curr=sink;
        while(curr!=source)
        {
            residual[parent[curr]][curr] -=
flowofpath;
            residual[curr][parent[curr]] +=
flowofpath;
            curr = parent[curr];
        }
        //update maxflow
        maxflow += flowofpath;
    }
    return maxflow;
}

void solve()
{
    //initialize adj, and residual graph.
    int n,m; cin>>n>>m;

    int source=1; int sink=2;
    reset(residual);

    REP0(i,m)
    {
        int u, v; cin>>u>>v;
        adj[u].pb(v);
        adj[v].pb(u); //even if it is directed
graph!!!
        residual[u][v]++;
    }

    cout<<Edmonds_Karp(source, sink)<<endl;
}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t; t=1;
    while(t--)
    {
        solve();
    }
}
```

Graph-1. BCC

```
#include <bits/stdc++.h>
#define eb emplace_back
#define em emplace
#define all(v) v.begin(), v.end()
#define fi first
#define se second
```

```
using namespace std;
```

```
typedef long long ll;
typedef pair <int, int> pii;
typedef pair <ll, ll> pll;
```

```
const int MAX = 101010;
const int INF = (int)1e9;
const ll LINF = (ll)1e18;
```

```
vector <int> g[MAX], stk, ng[MAX];
int ord[MAX], low[MAX], cnt, c, BCC[MAX];
```

```
void dfs(int x, int pa) {
    low[x] = ord[x] = ++cnt;
    stk.eb(x);
    bool flag = false;
    for(auto i : g[x]) {
        if(i == pa && !flag) {
            flag = true;
            continue;
        }
        if(ord[i]) low[x] = min(low[x], ord[i]);
        else {
            dfs(i, x);
            low[x] = min(low[x], low[i]);
        }
    }
    if(low[x] == ord[x]) {
        c++;
        while(stk.back() != x) {
            BCC[stk.back()] = c;
            stk.pop_back();
        }
        BCC[x] = c;
        stk.pop_back();
    }
}
```

```

int main() {
    ios::sync_with_stdio(false); cin.tie(nullptr);

    int n, m;
    cin >> n >> m;
    for(int i = 1; i <= m; i++) {
        int u, v;
        cin >> u >> v;
        g[u].eb(v), g[v].eb(u);
    }
    dfs(1, 0);
    for(int i = 1; i <= n; i++) {
        for(auto j : g[i]) {
            if(BCC[i] != BCC[j])
ng[BCC[i]].eb(BCC[j]);
        }
    }
}

```

Graph-2. SCC

```

#include <bits/stdc++.h>
#define eb emplace_back
#define em emplace
#define all(v) v.begin(), v.end()
#define fi first
#define se second

```

```
using namespace std;
```

```

typedef long long ll;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

```

```

const int MAX = 101010;
const int INF = (int)1e9;
const ll LINF = (ll)1e63;

```

```

int sz, low[505050], num[505050], scc[505050];
stack<int> stk;
vector<int> g[505050];
bool chk[505050];

```

```

int cnt;
void dfs(int x) {
    chk[x] = true;
    low[x] = num[x] = ++cnt;
    stk.push(x);
    for(auto i : g[x]) {

```

```

        if(num[i] == 0) {
            dfs(i);
            low[x] = min(low[x], low[i]);
        }
        else if(chk[i]) {
            low[x] = min(low[x], num[i]);
        }
    }
    if(low[x] == num[x]) {
        sz++;
        while(!stk.empty()) {
            int u = stk.top();
            stk.pop();
            chk[u] = false;
            scc[u] = sz;
            if(x == u) break;
        }
    }
}

int main() {
    ios::sync_with_stdio(false); cin.tie(0);

    int n, m;
    cin >> n >> m;
    for(int i = 0; i < m; i++) {
        int u, v; cin >> u >> v;
        g[u].eb(v);
    }

    for(int i = 1; i <= n; i++) {
        if(num[i] == 0)dfs(i);
    }
}

```

Graph-3. 2-SAT

```

/*
서용 헤더 생략
*/

```

```

vint adj[MxN+1];
vint rev[MxN+1];
vint component[MxN+1];
vint family[MxN+1];
vint topology;
int value[MxN+1];
bool visited[MxN+1];
int parent[MxN+1];
int n,m;

```

```
int NOT(int x){return (x>n)?(x-n):(x+n);}
```

```

void dfs_visit(int curr)
{
    visited[curr]=true;
    for(auto
next :adj[curr])if(visited[next]==false)dfs_visit(n
ext);
    topology.pb(curr);
}
void dfs_rev(int curr, int p)
{
    family[p].pb(curr);
    visited[curr]=true;
    parent[curr]=p;
    for(auto next :
rev[curr])if(visited[next]==false)dfs_rev(next,p);
}
void build_scc()
{
    REP1(i,2*n)
    {
        if(visited[i]==false)dfs_visit(i);
    }
    reverse(all(topology));
    reset(visited);
    for(auto i : topology)
    {
        if(visited[i]==false)dfs_rev(i,i);
    }
}

void dfs_comp(int curr)
{
    visited[curr]=true;

    for(auto next : component[curr])
    {
        if(visited[next]==false)dfs_comp(next);
    }

    topology.pb(curr);
}

void sort_component()
{

```

```

topology.clear();
reset(visited);
REP1(i, 2*n)
{
    if(parent[i]==i && visited[i]==false)
    {
        dfs_comp(i);
    }
}
reverse(all(topology));
}

void solve()
{
    cin>>n>>m;
    while(m--)
    {
        int a,b; cin>>a>>b;
        if(a<0)a = NOT(-a);
        if(b<0)b= NOT(-b);
        adj[NOT(a)].pb(b);
        adj[NOT(b)].pb(a);
        rev[b].pb(NOT(a));
        rev[a].pb(NOT(b));
    }

    build_scc();
    for(int i=1; i<=2*n; i++)
    {
        for(auto x: adj[i])
        {
            if(parent[i]!=parent[x])
            {
                component[parent[i]].pb(parent[x]);
            }
        }
    }
    sort_component();
    reverse(all(topology));
    memset(value, -1, sizeof(value));

    REP0(i,topology.size())
    {
        for(auto x : family[topology[i]])
        {
            if(value[x]!=-1)continue;
            if(value[NOT(x)]!=-1)
            {
                value[x] = (value[NOT(x)]^1);
                continue;
            }

```

```

                value[x] =1;
                for(auto next : adj[x])
                {
                    if(value[next]==0)value[x]=0;
                }
            }
        }

        bool pos=true;
        REP1(i,n)
        {
            if(parent[i] == parent[NOT(i)])pos=false;
        }

        cout<<pos<<endl;
        if(pos)REP1(i,n)cout<<value[i]<<" ";
    }

    int main()
    {
        ios::sync_with_stdio(0);
        cin.tie(0);

        int t; t=1;
        while(t--)
        {
            solve();
        }
    }
}

```

Graph-4. Offline_DynamicConnectivity

```

#include <bits/stdc++.h>
#define eb emplace_back
#define em emplace
#define all(v) v.begin(), v.end()
#define fi first
#define se second

using namespace std;

```

```

typedef long long ll;
typedef pair <int, int> pii;
typedef pair <ll, ll> pll;

const int MAX = 101010;
const int INF = (int)1e9;
const ll LINF = (ll)1e63;

struct Query {
    int t, a, b;
};

Query q[MAX];
vector <pii> E[4*MAX];
vector <int> ans;

void update(int node, int s, int e, int l, int r,
pii k) {
    if(s > r || e < l) return;
    if(s >= l && e <= r) {
        E[node].eb(k);
        return;
    }
    int m = (s + e) / 2;
    update(node*2, s, m, l, r, k);
    update(node*2+1, m+1, e, l, r, k);
}

struct UFT {
    int p[MAX], rnk[MAX];
    vector <int> bp[4*MAX], br[4*MAX];

    void init(int n) {
        for(int i = 1; i <= n; i++) {
            p[i] = i;
        }
    }

    int find(int x) {
        if(x == p[x]) return x;
        return find(p[x]);
    }

    void uni(int x, int y, int node) {
        x = find(x), y = find(y);
        if(x == y) return;
        if(rnk[x] < rnk[y]) swap(x, y);
        p[y] = x;
        bp[node].eb(y);
        if(rnk[x] == rnk[y]) {
            rnk[x]++;
            br[node].eb(x);
        }
    }
}

```

```

    }
}

void rollback(int node) {
    for(auto i : br[node]) rnk[i]--;
    for(auto i : bp[node]) p[i] = i;
}
} tree;

void dnc(int node, int s, int e) {
    for(auto i : E[node]) {
        tree.uni(i.fi, i.se, node);
    }
    if(s == e) {
        if(q[s].t == 3) {
            ans.eb(tree.find(q[s].a) ==
tree.find(q[s].b));
        }
    }
    else {
        int m = (s + e) / 2;
        dnc(node*2, s, m);
        dnc(node*2+1, m+1, e);
    }
    tree.rollback(node);
}

int main() {
    ios::sync_with_stdio(false); cin.tie(0);

    int n, m;
    cin >> n >> m;
    tree.init(n);
    map<pii, int> mp;
    for(int i = 1; i <= m; i++) {
        cin >> q[i].t >> q[i].a >> q[i].b;
        if(q[i].a > q[i].b) swap(q[i].a, q[i].b);
        if(q[i].t == 1) {
            mp[{q[i].a, q[i].b}] = i;
        }
        if(q[i].t == 2) {
            int val = mp[{q[i].a, q[i].b}];
            update(1, 1, m, val, i, {q[i].a,
q[i].b});
            mp.erase({q[i].a, q[i].b});
        }
    }
    for(auto i : mp) {
        update(1, 1, m, i.se, m, i.fi);
    }

    dnc(1, 1, m);
}

```

```

for(int i : ans) {
    cout << i << '\n';
}
}

```

String-1. KMP

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod = 1e9+7;
const int MxN = 1e5;
const int MxK = 1e9;
const ll inf = INT_MAX;
#define REP0(i,n) for(int i=0; i<n; i++)
#define REP1(i,n) for(int i=1; i<=n; i++)
#define REP(i,a,b) for(int i=a; i<=b; i++)
#define pb push_back
#define fi first
#define se second
#define reset(X) memset(X, 0, sizeof(X))
#define pii pair<int, int>
#define endl "\n"
#define vint vector<int>
//vector<int> adj[MxN+1];
//ll dp[MxN];

```

//given a text and a pattern we want to print out all the occurrences!
// Time complexity $O(n)$ in the worst case (preprocessing -failure ftn takes $O(m)$ time)

```

vector<int> failure_ftn(const string& pat)
{
    int m = (int)pat.size();
    vector<int> f(m, 0);

    int begin=1; int j=0;

    while(begin+j<m)
    {
        if (pat[begin+j]==pat[j])
        {
            j++;
            f[begin+j-1]=j;
        }
        else{

```

```

            if (j==0)
            {
                begin++;
            }
            else{
                begin += j-f[j-1];
                j=f[j-1];
            }
        }
    }
    return f;
}

```

```

vector<int> KMP(const string& text, const string& pat)
{

```

```

    int n = (int)text.size();
    int m = (int)pat.size();
    vector<int> f =
failure_ftn(pat); //prerprocessed vector
    vector<int> find; //where we will store the
starting positions

```

```

    int begin=0; int j=0; //begin indicates the
starting position of text, j indicates the position
of pat

```

```

    while(begin <= n-m) // length condition
should be satisfied
    {
        if (j<m && text[begin+j]==pat[j]){
            j++; // move on to the next character
            if
(j==m)find.push_back(begin); //found the
matching position!
        }
        else{ //mismatched

            if(j==0) begin++;
            else{
                begin = begin+j-f[j-1];
                j=f[j-1];
            }
        }
    }

    return find;
}

```



```

void solve()
{
    string text, pat; cin>>text>>pat;

    vint v;
    v = KMP(text, pat);

    REP0(i,v.size()) cout<< v[i]<<" ";    //zero
based indexing
}

```

```

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t; t=1;
    while(t--)
    {
        solve();
    }
}

```

String-2. LCP

```

/*
서용 헤더 생략
*/

```

```

void counting_sort(vint & p, vint & c)
{
    int n = sz(p);
    vint cnt(n);
    vint pos(n);

    for(auto x : c)cnt[x]++;
    pos[0]=0;
    for(int i=1; i<n; i++)pos[i] = pos[i-1]+cnt[i-1];

    vint p_update(n);
    for(auto x: p)
    {
        p_update[pos[c[x]]] = x;
    }
}

```

```

        pos[c[x]]++;
    }
    p = p_update;
}

pair<vint, vint> suffix_lcp_array(string
s) //build suffix array in O(nlogn), lcp in O(n)
{
    s += "$";
    int n = sz(s);
    vint p(n), c(n);

    vector<pair<char, int>> temp(n);
    REP0(i,n)temp[i] = {s[i], i};
    sort(all(temp));
    REP0(i,n)p[i]=temp[i].se;
    c[p[0]]=0;
    REP1(i,n-1)
    {
        if(temp[i].fi!=temp[i-1].fi)c[p[i]]=c[p[i-1]]+1;
        else c[p[i]]=c[p[i-1]];
    }

    int k=0;
    while((1<<k) < n)
    {
        REP0(i,n)p[i] = (p[i]-(1<<k)+n)%n;

        counting_sort(p,c);

        vint c_update(n);
        c_update[p[0]]=0;
        REP1(i,n-1)
        {
            pii prev = {c[p[i-1]], c[(p[i-1] +
(1<<k))%n]};
            pii curr = {c[p[i]], c[(p[i] +
(1<<k))%n]};
            if(prev!=curr)c_update[p[i]]=c_update[p
[i-1]]+1;
            else c_update[p[i]]=c_update[p[i-1]];
        }
        c = c_update;
        k ++;
    }

    vint lcp(n);
    int skip = 0;
}

```

```

for(int i=0; i<n-1; i++)
{
    int pi = c[i];
    int j = p[pi - 1];
    while(s[i+skip]==s[j+skip])skip++;

    lcp[pi] = skip;
    skip=max(skip-1,0);
}

return {p, lcp};
// suffix array : p
//lcp array : lcp
}

void solve()
{
    string s; cin>>s;

    vint sa,lcp;
    pair<vint, vint> ans = suffix_lcp_array(s);
    sa = ans.fi; lcp = ans.se;

    //printing out the suffix array and lcp array.
    REP1(i, sz(sa)-1)cout<<sa[i]+1<<" "; //problem
uses 1 based indexing
    cout<<endl;
    cout<<"x"<<" ";
    REP(i,2, sz(lcp)-1)cout<<lcp[i]<<" ";

}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t; t=1;
    while(t--)
    {
        solve();
    }
}

```

String-3. TRIE

```
/*
민규 헤더 생략
알파벳 개수 = 26
대문자 기준 -> 'A' (기본값)
소문자 기준 -> 'a'
string 의 경우 c_str()로 바꿔서 함수호출.
*/

struct Node{
    Node* childs[26] = {NULL};
    bool finished=false;
    Node(){}
```

```
};
struct Trie{
    Node* root = new Node();
    void insert(const char* str){
        Node* cur = root;
        for (int i=0;i<strlen(str);i++){
            if (cur->childs[str[i] - 'A']==
NULL)cur->childs[str[i] - 'A'] = new Node();
            cur = cur->childs[str[i] - 'A'];
        }
        cur->finished = true;
    }
    bool find(const char* str){
        Node* cur = root;
        for(int i=0;i<strlen(str);i++){
            if(cur->childs[str[i]-'A'] == NULL)
return false;
            cur = cur -> childs[str[i]-'A'];
        }
        return cur->finished;
    }
}trie;

int main()
{
    trie.insert("ABC");
    trie.insert("ABCDE");
    trie.insert("ABCD");
    cout<<trie.find("ABCD");
}
```

String-4. Manacher

```
/*
서용 헤더 생략
*/

//find palindrome in O(n)

string manipulate(string s)
{
    string changed="#";
    for(int i=0; i<(int)s.size(); i++)
    {
        changed+=s[i];
        changed+="#";
    }
    return changed;
}

vector<int> Manacher(string s)    //returns longest
palindrome substring lps array.
{
    vector<int> lps((int)s.size()); //original
string size:N -> manipulated string size:2N+1;
    int r=0; int c=0;    // [2*c-r ,r] is palindrome
and r is the maximum untill now

    for(int i=0; i<(int)s.size(); i++)
    {
        if(i<=r)lps[i] = min(r-i, lps[2*c-i]);
        else lps[i]=0;

        while( (i-lps[i]-
1)>=0 && (i+lps[i]+1)<s.size() && s[i+lps[i]+1]
==s[i-lps[i]-1] )lps[i]++;

        if(r<lps[i]+i)
        {
            r=lps[i]+i;
            c=i;
        }
    }

    return lps;
}

void solve()
{
    vint v= Manacher(manipulate("aaaaa"));
    REP0(i,v.size())cout<<v[i]<<endl;
```

```
}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t; t=1;
    while(t--){
        solve();
    }
}

String-5. Aho corasick

/*
서용 헤더 생략
*/

//KMP : O(nk + Sigma(mi) )
//Ahocorasick : O(n+ Sigma(mi) )
//given string s, multiple patterns m1,
m2,m3, ...,mk
//we decide if the pattern exists inside the string
or not
const int MAXN = 100005, MAXC = 26;
struct aho_corasick{
    int trie[MAXN][MAXC], piv; // trie
    int fail[MAXN]; // failure link
    int term[MAXN]; // output check
    void init(vector<string> &v){
        memset(trie, 0, sizeof(trie));
        memset(fail, 0, sizeof(fail));
        memset(term, 0, sizeof(term));
        piv = 0;
        for(auto &i : v){
            int p = 0;
            for(auto &j : i){
                if(!trie[p][j-'a']) trie[p][j-'a']
= ++piv;

                p = trie[p][j-'a'];
            }
            term[p] = 1;
        }
        queue<int> que;
        for(int i=0; i<MAXC; i++){
            if(trie[0][i]) que.push(trie[0][i]);
        }
        while(!que.empty()){
            int x = que.front();
```



```

const int INF = (int)1e9;
const ll LINF = (ll)1e63;

vector<pii> g[MAX];
int n;
int sz[MAX], in[MAX], nxt[MAX], p[MAX], dep[MAX],
val[MAX], po[MAX], cnt;
int tree[4*MAX];

void dfs(int x, int pa) {
    sz[x] = 1;
    p[x] = pa;
    dep[x] = dep[pa] + 1;
    for(int i = 0; i < g[x].size(); i++) {
        int y = g[x][i].fi;
        if(y == pa) continue;
        dfs(y, x);
        sz[x] += sz[y];
        po[g[x][i].se] = y;
        if(sz[y] > sz[g[x][0].fi] || g[x][0].fi ==
pa) {
            swap(g[x][i], g[x][0]);
        }
    }
}

void hld(int x) {
    in[x] = ++cnt;
    for(int i = 0; i < g[x].size(); i++) {
        int y = g[x][i].fi;
        if(y == p[x]) continue;
        if(i == 0) nxt[y] = nxt[x];
        else nxt[y] = y;
        hld(y);
    }
}

void update(int node, int s, int e, int k, int x) {
    if(s == e) {
        tree[node] = x;
        return;
    }
    int m = (s + e) / 2;
    if(k <= m) update(node * 2, s, m, k, x);
    else update(node * 2 + 1, m + 1, e, k, x);
    tree[node] = max(tree[node * 2], tree[node * 2
+ 1]);
}

int cal(int node, int s, int e, int l, int r) {
    if(s > r || e < l) return 0;
    if(s >= l && e <= r) return tree[node];

```

```

    int m = (s + e) / 2;
    int ret = max(cal(node * 2, s, m, l, r),
cal(node * 2 + 1, m + 1, e, l, r));
    return ret;
}

int maxquery(int u, int v) {
    int ret = 0;
    while(nxt[u] != nxt[v]) {
        if(dep[nxt[u]] > dep[nxt[v]]) swap(u, v);
        ret = max(ret, cal(1, 1, n, in[nxt[v]],
in[v]));
        v = p[nxt[v]];
    }
    if(dep[u] > dep[v]) swap(u, v);
    if(u != v) ret = max(ret, cal(1, 1, n, in[u] +
1, in[v]));
    return ret;
}

int main() {
    ios::sync_with_stdio(false); cin.tie(0);

    cin >> n;
    for(int i = 1; i < n; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        g[u].eb(v, i);
        g[v].eb(u, i);
        val[i] = w;
    }
    dfs(1, 0);
    nxt[1] = 1;
    hld(1);
    for(int i = 1; i < n; i++) {
        update(1, 1, n, in[po[i]], val[i]);
    }

    int q;
    cin >> q;
    while(q--) {
        int t, u, v;
        cin >> t >> u >> v;
        if(t == 1) {
            val[u] = v;
            update(1, 1, n, in[po[u]], v);
        }
        if(t == 2) {
            cout << maxquery(u, v) << '\n';
        }
    }
}

```

DataStructure-1. LichaoTree_in_SegmentTree

```

#include <bits/stdc++.h>
#define eb emplace_back
#define em emplace
#define all(v) v.begin(), v.end()
#define fi first
#define se second

using namespace std;

typedef long long ll;
typedef pair<int, ll> pii;
typedef pair<ll, ll> pll;

const int MAX = 303030;
const int INF = (int)1e9;
const ll LINF = (ll)9e18;

class LiChaoSegmentTree {
    struct Node {
        Node *l = nullptr, *r = nullptr;
        pii val;
    } *tree;

public :
    ll cal(pii l, int x) {
        return (ll)l.fi * x + l.se;
    }
    void add_line(Node *node, pii val, int s = -
INF, int e = INF + 1) {
        int m = s + e >> 1;
        bool left = cal(val, s) > cal(node->val,
s);
        bool mid = cal(val, m) > cal(node->val, m);
        if(mid) swap(val, node->val);
        if(e == s + 1) return;
        if(left != mid) {
            if(!node->l) {
                node->l = new Node;
                node->l->val = val;
            }
            else add_line(node->l, val, s, m);
        }
        else {
            if(!node->r) {
                node->r = new Node;

```

```

        node->r->val = val;
    }
    else add_line(node->r, val, m, e);
}

void Add_line(pii val) {
    if(!tree) {
        tree = new Node;
        tree->val = val;
        return;
    }
    else add_line(tree, val);
}

ll cal_max(Node *node, int x, int s = -INF, int e = INF + 1) {
    int m = s + e >> 1;
    if(e == s + 1) return cal(node->val, x);
    if(x < m) {
        if(node->l) return max(cal(node->val, x), cal_max(node->l, x, s, m));
        else return cal(node->val, x);
    }
    else {
        if(node->r) return max(cal(node->val, x), cal_max(node->r, x, m, e));
        else return cal(node->val, x);
    }
}

ll Cal_max(int x) {
    if(!tree) return -LINF;
    return cal_max(tree, x);
}
};

```

```

pii line[MAX];
bool chk[MAX];
vector <pii> pt;

```

```

int main() {
    ios::sync_with_stdio(false); cin.tie(nullptr);

    int n;
    cin >> n;
    ST.Init(n);
    for(int i = 1; i <= n; i++) {
        int t;
        cin >> t;
        if(t == 1) {

```

```

        cin >> line[i].fi >> line[i].se;
        chk[i] = true;
    }
    if(t == 2) {
        int x;
        cin >> x;
        ST.Update(x, i, line[i]);
        chk[x] = false;
    }
    if(t == 3) {
        int x;
        cin >> x;
        pt.eb(i, x);
    }
}

for(int i = 1; i <= n; i++) {
    if(chk[i]) ST.Update(i, n, line[i]);
}

for(auto i : pt) {
    ll ans = ST.Cal_max(i.fi, i.se);
    if(ans == -LINF) cout << "EMPTY" << '\n';
    else cout << ans << '\n';
}
}

```

DataStructure-2. LineContainer

```

/**
 * Author: Simon Lindholm
 * Date: 2017-04-20
 * License: CC0
 * Source: own work
 * Description: Container where you can add lines
of the form  $kx+m$ , and query maximum values at
points  $x$ .
 * Useful for dynamic programming (``convex hull
trick'').
 * Time:  $O(\log N)$ 
 * Status: stress-tested
 */
#pragma once

struct Line {
    mutable ll k, m, p;
    bool operator<(const Line& o) const { return k
< o.k; } // >로 바꾸면 min query
    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; //<로 바꾸면 min query
        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;
    }
};

```

DataStructure-3. Mo's algorithm

```

/*
서용 헤더 생략
*/

/* Mo's Algo
1.offline 2.static 3. processing
insertion/deletion in active range takes  $f(n)$ .
Mo's algorithm Time complexity :
 $O((Q+N)\sqrt{N}*F + Q\log Q)$ 
 $i \sim j$  에 존재하는 서로 다른 수의 개수 (online :
Mergesorttree or pst)
 $i \sim j$  에 가장 많이 등장한 수
 $i \sim j$  에 가장 많이 등장한 수가 등장한 횟수
*/

```

```
//      query      //
int q;
int sqrtQ;
struct Query
{
    int index, left, right;
    bool operator< (const Query & q1)
    {
        if (left/sqrtQ!=q1.left/sqrtQ)return
left/sqrtQ<q1.left/sqrtQ;
        return right<q1.right;
    }
};
vector<Query> queries;
vint ansforquery;
void init()
{
    sqrtQ = sqrt(q);
    ansforquery.resize(q);
    sort(all(queries));
}
//              //

int ar[MxN+1];
vint store;
int occurrence[MxN];
int pos[MxN+1];

int getindex(int x) {return
lower_bound(all(store),x) - store.begin();}

void solve()
{
    int n; cin>>n;
    REP1(i,n){cin>>ar[i]; store.pb(ar[i]);}
    compress(store);
    REP1(i,n)pos[i] = getindex(ar[i]);
    cin>>q;
    REP0(i,q)
    {
        int s,e; cin>>s>>e;
        queries.pb({i,s,e});
    }
    init();
```

```
    int activelleft = queries[0].left; int
activeright = queries[0].left-1; int ans=0;
    REP0(i,q)
    {
        int left = queries[i].left; int right =
queries[i].right; int index = queries[i].index;

        while(activeright < right)
        {
            activeright++;
            if(occurrence[pos[activeright]]
==0 )ans++;
            occurrence[pos[activeright]] ++;
        }
        while(left < activelleft)
        {
            activelleft --;
            if(occurrence[pos[activelleft]]==0)ans
++;
            occurrence[pos[activelleft]]++;
        }
        while(activeright>right)
        {
            occurrence[pos[activeright]] -- ;
            if(occurrence[pos[activeright]]==0)ans
--;
            activeright--;
        }
        while(activelleft<left)
        {
            occurrence[pos[activelleft]]--;
            if(occurrence[pos[activelleft]]==0)ans-
-;
            activelleft++;
        }

        ansforquery[index] = ans;
    }

    REP0(i, q)cout<<ansforquery[i]<<endl;
}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t; t=1;
```

```
    while(t--)
    {
        solve();
    }
}
```

DataStructure-4. Pbds

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

template<class T>
using ordered_set = tree<T, null_type, less<T>,
rb_tree_tag,tree_order_statistics_node_update>;

ordered_set<int> X;
int main()
{
    X.insert(5);
    X.insert(3);
    X.insert(10);
    X.insert(15);

    cout<<*X.find_by_order(1)<<endl; //10 (0-based)
    X.erase(X.find(10));
    cout<<*X.find_by_order(2)<<endl; //15 (0-based)

    for(auto i:X){ // 3 5 15
        cout<<i<<" ";
    }
}
```

DataStructure-5. SegmentTreeBeats

```
#include <bits/stdc++.h>
#define eb emplace_back
#define em emplace
#define all(v) v.begin(), v.end()
#define fi first
#define se second

using namespace std;

typedef long long ll;
```



```

typedef pair <int, int> pii;
typedef pair <ll, ll> pll;

const int MAX = 101010;
const int INF = 1e9;
const ll LINF = 1e18;

// 수열과 쿼리 19 정답 코드임. 어차피 세그 비츠는 어떤
// 특성값 C를 정해서 세그먼트 트리 노드 전체의 C 값에 대한
// 곱셈이 필요함.
// 단지 이런 느낌으로 코딩하면 된다는 참고용
class SegmentTree {
    struct Node {
        ll sum, mn, mx;
        Node(ll sum = 0, ll mn = LINF, ll mx = -
LINF) : sum(sum), mn(mn), mx(mx) {}
        Node operator + (Node b) {
            Node ret;
            ret.sum = sum + b.sum;
            ret.mx = max(mx, b.mx);
            ret.mn = min(mn, b.mn);
            return ret;
        }
    };
    vector <Node> tree;
    vector <pll> lazy;
    int n;

public:
    ll div(ll a, ll d) {
        if(a >= 0) return a / d;
        else return (a - (d - 1)) / d;
    }

    void update_lazy(int node, int s, int e) {
        tree[node].sum = tree[node].sum *
lazy[node].fi + lazy[node].se * (e-s+1);
        tree[node].mx = tree[node].mx *
lazy[node].fi + lazy[node].se;
        tree[node].mn = tree[node].mn *
lazy[node].fi + lazy[node].se;
        if(s != e) {
            lazy[node<<1].fi = lazy[node<<1].fi *
lazy[node].fi;
            lazy[node<<1].se = lazy[node<<1].se *
lazy[node].fi + lazy[node].se;
            lazy[node<<1|1].fi = lazy[node<<1|1].fi
* lazy[node].fi;
            lazy[node<<1|1].se = lazy[node<<1|1].se
* lazy[node].fi + lazy[node].se;

```

```

        }
        lazy[node] = {1, 0};
    }

    void Init(int N, vector <ll> &A) {
        n = N;
        tree.resize(4*n+10);
        lazy.resize(4*n+10);
        init(1, 0, n-1, A);
    }

    void init(int node, int s, int e, vector <ll>
&A) {
        lazy[node] = {1, 0};
        if(s == e) {
            tree[node].sum = tree[node].mn =
tree[node].mx = A[s];
            return;
        }
        int m = s + e >> 1;
        init(node<<1, s, m, A);
        init(node<<1|1, m+1, e, A);
        tree[node] = tree[node<<1] +
tree[node<<1|1];
    }

    void divide(int l, int r, ll d) {
        update_d(1, 0, n-1, l, r, d);
    }

    void update_d(int node, int s, int e, int l,
int r, ll d) {
        update_lazy(node, s, e);
        if(s > r || e < l) return;
        if(s >= l && e <= r && div(tree[node].mx,
d) == div(tree[node].mn, d)) {
            lazy[node].fi = 0;
            lazy[node].se = div(tree[node].mx, d);
            update_lazy(node, s, e);
            return;
        }
        if(s >= l && e <= r && tree[node].mx -
tree[node].mn == 1) {
            lazy[node].se = div(tree[node].mn, d) -
tree[node].mn;
            update_lazy(node, s, e);
            return;
        }
        int m = s + e >> 1;
        update_d(node<<1, s, m, l, r, d);
        update_d(node<<1|1, m+1, e, l, r, d);

```

```

        tree[node] = tree[node<<1] +
tree[node<<1|1];
    }

    void add(int l, int r, ll s) {
        update_s(1, 0, n-1, l, r, s);
    }

    void update_s(int node, int s, int e, int l,
int r, ll c) {
        update_lazy(node, s, e);
        if(s > r || e < l) return;
        if(s >= l && e <= r) {
            lazy[node].se = c;
            update_lazy(node, s, e);
            return;
        }
        int m = s + e >> 1;
        update_s(node<<1, s, m, l, r, c);
        update_s(node<<1|1, m+1, e, l, r, c);
        tree[node] = tree[node<<1] +
tree[node<<1|1];
    }

    ll sum(int l, int r) {
        return cal(1, 0, n-1, l, r).sum;
    }

    ll mn(int l, int r) {
        return cal(1, 0, n-1, l, r).mn;
    }

    Node cal(int node, int s, int e, int l, int r)
{
        update_lazy(node, s, e);
        if(s > r || e < l) return Node();
        if(s >= l && e <= r) return tree[node];
        int m = s + e >> 1;
        return cal(node<<1, s, m, l, r) +
cal(node<<1|1, m+1, e, l, r);
    }
} ST;

int main() {
    ios::sync_with_stdio(false); cin.tie(0);

    int n, q;
    cin >> n >> q;
    vector <ll> A(n);
    for(int i = 0; i < n; i++) {
        cin >> A[i];
    }

```

```

ST.Init(n, A);

while(q--){
    int t, l, r;
    cin >> t >> l >> r;
    if(t == 1){
        ll c;
        cin >> c;
        ST.add(l, r, c);
    }
    if(t == 2){
        ll d;
        cin >> d;
        ST.divide(l, r, d);
    }
    if(t == 3){
        cout << ST.mn(l, r) << '\n';
    }
    if(t == 4){
        cout << ST.sum(l, r) << '\n';
    }
}
}

```

DataStructure-6. SplayTree

```

/*
회승 헤더 생략
*/

template <
class T,
class Container = vector <T>,
class Compare = less <T>
> class SplayTree {
    struct Node{
        Node *l, *r, *p;
        int c;
        T val;
    } *tree;

public :
    void upd(Node *x) {
        x->c = 1;
        if(x->l) x->c += x->l->c;
        if(x->r) x->c += x->r->c;
    }
}

```

```

void rotate(Node *x) {
    Node *p = x->p;
    Node *m;
    if(x == p->l) {
        p->l = m = x->r;
        x->r = p;
    }
    else {
        p->r = m = x->l;
        x->l = p;
    }
    if(m) m->p = p;
    x->p = p->p;
    p->p = x;
    if(x->p) {
        if(p == x->p->l) x->p->l = x;
        else x->p->r = x;
    }
    else tree = x;
    upd(p);
    upd(x);
}

void splay(Node *x) {
    while(x->p) {
        Node *p = x->p, *pp = p->p;
        if(pp) {
            if((pp->l == p) == (p->l == x))
                rotate(p);
            else rotate(x);
        }
        rotate(x);
    }
}

void splay_k(int k) {
    Node *x = tree;
    while(1) {
        while(x->l && x->l->c > k) x = x->l;
        if(x->l) k -= x->l->c;
        if(!k) break;
        k--;
        x = x->r;
        splay(x);
    }
}

void splay_itv(int s, int e) {
    splay_k(s - 1);
    Node *tmp = tree;
    tree = tmp->r;
}

```

```

tree->p = nullptr;
splay_k(e - s + 1);
tmp->r = tree;
tree->p = tmp;
tree = tmp;
}

void insert(T val) {
    Node *p = tree;
    Node **pp;
    if(!p) {
        Node *x = new Node;
        tree = x;
        x->l = x->r = x->p = NULL;
        x->val = val;
        return;
    }
    while(1) {
        //if(val == p->val) return; //중복 포함
        if(Compare(val, p->val)) {
            if(!p->l) {
                pp = &p->l;
                break;
            }
            p = p->l;
        }
        else {
            if(!p->r) {
                pp = &p->r;
                break;
            }
            p = p->r;
        }
    }
    Node *x = new Node;
    *pp = x;
    x->l = x->r = NULL;
    x->p = p, x->val = val;
    splay(x);
}

bool find(T val) {
    Node *p = tree;
    if(!p) return false;
    while(p) {
        if(val == p->val) break;
        if(compare(val, p->val)) {
            if(!p->l) break;
            p = p->l;
        }
        else {

```

```

        if(!p->r) break;
        p = p->r;
    }
}
splay(p);
return val == p->val;
}

void delete(int key){
    if(!find(key)) return;
    node* p = tree;
    if(p->l && p->r) {
        tree = p->l; tree->p = NULL;
        node* x = tree;
        while(x->r) x = x->r;
        x->r = p->r; p->r->p = x;
        delete p;
        return;
    }
    if(p->l) {
        tree = p->l; tree->p = NULL;
        delete p;
        return;
    }
    if(p->r) {
        tree = p->r; tree->p = NULL;
        delete p;
        return;
    }

    delete p; tree = NULL;
}

void init(Container V) {
    int n = V.size();
    v.push_back((T)0);
    Node *x = new Node;
    tree = x;
    x->l = x->r = x->p = nullptr;
    x->c = n + 2;
    x->val = (T)0;
    for(int i = 0; i <= n; i++) {
        x->r = new Node;
        x->r->p = x;
        x = x->r;
        x->l = x->r = nullptr;
        x->c = n + 1 - i;
        x->val = V[i];
    }
}
};

```

```

int main() {
    ios::sync_with_stdio(false);

    string s;
    cin >> s;
    SplayTree <int> ST;
}

```

DataStructure-7. Dominator tree

```

/*
서용 헤더 생략
*/

struct dominator{
    vint dtree[MxN+2]; //dominator tree
    vint adj[MxN+2]; //original graph
    vint reverse[MxN+2]; //reverse graph (in
index. everything below is in index)
    vint nodes_having_semidom[MxN+2];
    int dom[MxN+2],
    sdом[MxN+2],parent[MxN+2]; //idom, sdом,
parent in dfs tree
    int link[MxN+2],label[MxN+2]; //about dsu ,
label[i] stores vertex with min sdом lying on
i~root
    int indexof[MxN+2], valueof[MxN+2];
//mapping (i <-> index in dfs tree )
    int T=0; //used for counting index of dfstree

    dominator()
    {
        REP0(i,MxN+2 )
        {
            dom[i] =
0;sdом[i]=0;parent[i]=0;link[i]=0;
            label[i]=0; indexof[i]=0;valueof[i]=0;
        }

        void build()
        {
            initialize(0); //build dfstree from
source(node 0)
            calcsdom();
            calcidom();
        }
    }
}

```

```

int Find(int u,int x=0) //return minimum
sdом lying on path v to r
{
    if(u==link[u])return x?-1:u;
    int v = Find(link[u],x+1);
    if(v<0)return u;
    if(sdом[label[link[u]]]<sdом[label[u]])
        label[u] = label[link[u]];
    link[u] = v;
    return x?v:label[u];
}

void Union(int u,int v){ //Merge dsu tree where
u and v is placed
    link[v]=u;
}

void initialize(int node)
{
    T++;
    indexof[node] =T; valueof[T] = node;
    label[T]=T;
    sdом[T] =T;
    link[T] =T;
    for (auto next : adj[node])
    {
        if(indexof[next]==0)
        {
            initialize(next);
            parent[indexof[next]] =
indexof[node];
        }
        reverse[indexof[next]].pb(indexof[node]
);
    }
}

void calcsdom()
{
    int n=T;
    for(int i=n;i>=1;i--)
    {
        for(int
j=0;j<sz(reverse[i]);j++)sdом[i] =
min(sdом[i],sdом[Find(reverse[i][j])]);
        if(i>1)nodes_having_semidom[sdom[i]].pb
(i);
        for(int
j=0;j<sz(nodes_having_semidom[i]);j++)
        {
            int w =
nodes_having_semidom[i][j],v = Find(w);

```

```

        if(sdom[v]==sdom[w])
dom[w]=sdom[w];
        else dom[w] = v;
    }
    if(i>1)Union(parent[i],i);
}

void calcidom()
{
    int n=T;
    REP(i,2,n)
    {
        if(dom[i] !=sdom[i])dom[i] =
dom[dom[i]];
        //make dfstree
        dtree[valueof[i]].pb(valueof[dom[i]]);
        dtree[valueof[dom[i]]].pb(valueof[i]);
    }
};

void solve()
{
    int n,m,k; cin>>n>>m>>k;
    //build graph. don't forget source node
    dominator D;
    REP0(i,k){int u,v; cin>>u>>v; D.adj[u].pb(v);}
    int source =0;

    REP1(i,m)
    {
        D.adj[source].pb(i);
    }
    D.build();

    cout<<sz(D.dtree[source])<<endl;
}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t; t=1;
    while(t--)
    {

```

```

        solve();
    }
}

```

Optimization-1. DnC Opt

```

/*
서용헤더 생략
Knuth Optimization
Recurrence : dp[i][j] =
min[i<=k<j](dp[i][k]+dp[k+1][j]+C[i][j])
Condition. #1. C[i][j] is Monge array
#2.C[a][d]>=C[b][c] (a<=b<=c<=d)
Following Property holds. opt[i][j-
1]<=opt[i][j]<=opt[i+1][j].
Using the property, we calculate dp[i][j] in the
order of increasing j-i, and we limit the range of
k.
This Leads to naive O(n^3) -> O(n^2)
*/

/*
Divide and Conquer Optimization : O(mn^2) -> O(m
nlogn)
Recursion : dp[i][j] = Min[k] ( dp[i-1][k]
+C[k][j])
Condition : (고정된 i 에서) j 일때 최소값을 주는 k 를
opt(j) 라 할때 optj 가 monotonic.
각각의 dp[i][1~n]를 O(nlogn)에 calculate.
만약 k<=j & C[k][j]가 Monge array 라면 optj 가
monotonic 임을 보일 수 있음.
*/

ll val[8001];
ll sum[8001];
ll dp[801][8001];

long long C(int k, int j)    //C[k][j] 값을 계산
{
    return (j-k)*(sum[j]-sum[k]);
}

void compute(int l, int r, int optl, int optr, int
i) //compute dp[i][l] ~ dp[i][r]을 찾는다. 가능한
k 는 optl~optr
{
    if(l>r)return;

```

```

    int mid = (l+r)/2;
    pair<ll, int> best = {LLONG_MAX, -
1};    //min 을 구하는 것이므로 LLONG_MAX

    for (int k = optl; k <=min(mid-1,optr); k++)
    {
        best = min(best, {dp[i-1][k] + C(k, mid),
k});
    }

    dp[i][mid] = best.first;
    int opt = best.second;

    compute(l, mid - 1, optl, opt,i);
    compute(mid + 1, r, opt, optr,i);
}

void solve()
{
    int n,m; cin>>n>>m;
    REP1(i,n)cin>>val[i];
    REP1(i,n)sum[i] = sum[i-1]+val[i];

    //initialize;
    REP0(i,m+1)dp[i][0]=0;
    REP0(i,n+1)dp[1][i]=i*sum[i];
    //fill the dp array

    for(int i=2; i<=m; i++)
    {
        for(int j=1; j<i; j++)
        {
            dp[i][j] = sum[j];
        }
        compute(i,n,1, n-1,i); //fill dp[i][1~n]
    }

    cout<<dp[m][n]<<endl;
}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t; t=1;
    while(t--)
    {
        solve();
    }
}

```