Math

```
long long gcd(long long a, long long b)
{
    return b ? a : gcd(b, a % b);
}
long long lcm(long long a, long long b)
{
    return a * b / gcd(a, b);
}

/*
    By using exgcd(a,b,x,y),you can get one solution of the equation:
    ax+by==gcd(a,b)
*/
long long exgcd(long long a, long long b, long long &x, long long &y)
{
    if (!b)
    {
        x = 1;
        y = 0;
        return a;
    }
    long long res = exgcd(b, a % b, y, x);
    y -= (a / b) * x;
    return res;
}

//Get a^{-1} (mod m)
inline long long inv(long long a, long long m)
{
    long long x, y;
    exgcd(a, m, x, y);
    return (x + m) % m;
}

//By using phi(n),you can get the Euler function of n
long long get_phi(long long x)
{
    long long res = x;
    for (long long i = 2; i * i <= x; i++)
    {
        if (x % i == 0)
        {
            res /= i;
            res *= (i - 1);
            while (x % i == 0) x /= i;
        }
    }
    if (x != 1) res /= x, res *= (x - 1);
    return res;
}

/*
```

```
    China Remainer Theory is an algorithm to solve linear congruent
equations.Like:
    x==a1 (mod m1)
    x==a2 (mod m2)
    ...
    x==an (mod mn)
    Each a is coprime.
    If not,you should use exCRT(n).
    By using CRT(n),you can get the minimum solution of the equations.
*/
long long a[MAXN], m[MAXN];
inline long long CRT(int n)
{
    long long M = 1, ans = 0;
    for (int i = 1; i <= n; i++) M *= m[i];
    for (int i = 1; i <= n; i++)
    {
        long long w = M / m[i], x, y;
        exgcd(w, m[i], x, y);
        ans = (ans + x * w * a[i]) % M;
    }
    return (ans + M) % M;
}

inline long long exCRT(int n)
{
    long long M = m[1], ans = a[1], x, y;
    for (int i = 2; i <= n; i++)
    {
        long long g = exgcd(M, m[i], x, y);
        if ((a[i] - ans) % g) return -1LL;
        x = (a[i] - ans) / g * x % (m[i] / g);
        ans += x * M;
        M = M / g * m[i];
        ans %= M;
    }
    return (ans + M) % M;
}

// get x^2 = n (mod p)
long long msqrt(long long n, long long p)
{
    if (!n) return 0;
    long long q = p - 1, s = 0, z = 2;
    //while (~q & 1) q >>= 1, s++;
    q >>= (s = __builtin_ctzll(q));
    if (s == 1) return fpow(n, (p + 1) / 4, p);
    while(fpow(z, (p - 1) / 2, p) == 1) ++z;
    long long c = fpow(z, q, p), t = fpow(n, q, p),
              r = fpow(n, (q + 1) / 2, p), m = s;
    while(t % p != 1)
    {
        long long i = 1; while(fpow(t, 1ll << i, p) != 1) ++i;
        long long b = fpow(c, 1ll << (m - i - 1), p);
        r = r * b % p; c = (b * b) % p;
        t = (t * c) % p; m = i;
    }
    return min(r, p - r); //    r^2=(p-r)^2=n
```

```
}

//By using Linear_Shaker(),you can get prime,Euler's function and Mobius's
function in [1,n].
long long prime[MAXN], phi[MAXN], mu[MAXN];
bool not_prime[MAXN];
int tot;

inline void Linear_Shaker(int n)
{
    phi[1] = 1LL;
    mu[1] = 1LL;
    for (int i = 2; i <= n; i++)
    {
        if (!not_prime[i])
        {
            prime[++tot] = i;
            phi[i] = i - 1;
            mu[i] = -1;
        }
        for (int j = 1; i * prime[j] <= n; j++)
        {
            not_prime[i * prime[j]] = true;
            if (!(i % prime[j]))
            {
                mu[i * prime[j]] = 0LL;
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            }
            mu[i * prime[j]] = -mu[i];
            phi[i * prime[j]] = phi[i] * (prime[j] - 1);
        }
    }
    return ;
}

/*
    By using BSGS(a,b,c),you can get the minimum solution of the equation:
    a^x==b (mod c)
*/

unordered_map <long long, long long> mp;

long long exBSGS(long long a, long long b, long long M)
{
    if (b == 1) return 0;
    long long w = 1; int c = 0;
    for (long long d; (d = gcd(a, M)) != 1;)
    {
        if (b % d) return -1;
        b /= d; M /= d; ++c; w = w * (a / d) % M;
        if (w == b) return c;
    }
    b = b * inv(w, M) % M; mp.clear();
    long long t = 1LL, r = b, x, y, B = ceil(sqrt(M));
    for (long long i = 0; i < B; i++, t = t * a % M) if (!mp.count(t))
mp.insert({t, i});
    t = inv(t, M);
```

```
    for (long long i = 0; i < B; i++, b = b * t % M) if (mp.count(b)) return i * B + mp[b] + c;
    return -1LL;
}

/*
    By using Gauss_Elimination(n),you can solve n equation group.
    equation[1..n] is the coefficient and equation[n+1] is constant term.
    To the i_st equation,the formular is like:
    equation[i][1]x1+equation[i][2]x2+...+equation[i][n]xn=equation[i][n+1]
*/
double equations[MAXN][MAXN];
inline void Gauss_Elimination(int n)
{
    for (int i = 1; i <= n; i++)
    {
        int now = i;
        for (int j = i + 1; j <= n; j++)
            if (fabs(equations[now][i]) < fabs(equations[j][i])) now = j;
        swap(equations[now], equations[i]);
        double t = equations[i][i];
        for (int j = 1; j <= n + 1; j++) equations[i][j] /= t;
        for (int j = 1; j <= n; j++)
            if (j != i)
            {
                t = equations[j][i];
                for (int k = 1; k <= n + 1; k++)
                    equations[j][k] -= t * equations[i][k];
            }
    }
    return ;
}

struct Matrix
{
    long long m[MAXN][MAXN];
}inv;

bool vis[MAXN];
long long det;
int rnk;

inline void getDetRankInv(Matrix a)
{
    memset(vis, 0, sizeof vis);
    det = 1; rnk = n;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            inv.m[i][j] = i == j;
    for (int i = 1; i <= n; i++)
    {
        int now = 0;
        for (int j = 1; j <= n && !now; j++) if (!vis[j] && a.m[j][i]) now = j;
        if (!now){ --rnk; continue; }
        if (now != i) det = -det;
        vis[i] = true;
        for (int j = 1; j <= n; j++)
        {
```

```
            swap(a.m[now][j], a.m[i][j]);
            swap(inv.m[now][j], inv.m[i][j]);
        }
        long long t = inv(a.m[i][i], M);
        for (int j = 1; j <= n; j++)
            if (j != i)
            {
                long long x = a.m[j][i] * t % M;
                for (int k = 1; k <= n; k++)
                {
                    a.m[j][k] -= x * a.m[i][k] % M; (a.m[j][k] += M) %= M;
                    inv.m[j][k] -= x * inv.m[i][k]%M; (inv.m[j][k] += M) %= M;
                }
            }
    }
    det = (det + M) % M;
    for (int i = 1; i <= n; i++)
    {
        (det *= a.m[i][i]) %= M;
        long long t = inv(a.m[i][i], M);
        for (int j = 1; j <= n; j++) (inv.m[i][j] *= t) %= M;
    }
    return ;
}

inline Matrix Gauss_Elimination(Matrix a)
{
    memset(vis, false, sizeof vis);
    for (int i = 1; i <= n; i++)
    {
        int now = 0;
        for (int j = 1; j <= n && !now; j++) if (!vis[j] && a.m[j][i]) now = j;
        if (!now) continue;
        vis[i] = true;
        for (int j = 1; j <= n; j++) swap(a.m[now][j], a.m[i][j]);
        long long t = inv(a.m[i][i], M);
        for (int j = 1; j <= n; j++)
            if (j != i)
            {
                long long x = a.m[j][i] * t % M;
                for (int k = 1; k <= n; k++) a.m[j][k] -= x * a.m[i][k] % M,
(a.m[j][k] += M) %= M;
            }
    }
    return a;
}

//By using Get_Factor_and_Inv(n,MOD),you can get i! and i^(-1) and i^(-1)!.
long long fac[MAXN], inv[MAXN], invfac[MAXN];
inline void Get_Factor_and_Inv(int n, long long MOD)
{
    inv[1] = fac[0] = invfac[0] = 1LL;
    for (int i = 2; i <= n; i++) inv[i] = (MOD - MOD / i) * inv[MOD % i] % MOD;
    for (int i = 2; i <= n; i++) fac[i] = (fac[i - 1] * i) % MOD, invfac[i] =
(invfac[i - 1] * inv[i]) % MOD;
    return ;
}
```

```
inline long long A(int m, int n, long long MOD)
{
    return fac[n] * invfac[n - m] % MOD;
}
inline long long C(int m, int n, long long MOD)
{
    return fac[n] * invfac[m] % MOD * invfac[n - m] % MOD;
}

/*
    By using Lucas_C(n,m,MOD),you can figure out C(m,n)%MOD as well.
    When n,m is huge but MOD is smaller than 1e8,Lucas_C is recommended.
*/
long long Lucas_C(long long m, long long n, long long MOD)
{
    if (n < m) return 0LL;
    if (n < MOD && m < MOD) return C(m, n, MOD);
    return Lucas_C(n / MOD, m / MOD, MOD) * Lucas_C(n % MOD, m % MOD, MOD) %
MOD;
}
```

CG

```
const double eps = 1e-7;

struct point
{
    double x, y;
    point() {}
    point(double _x, double _y):
        x(_x), y(_y) {}
    point operator + (const point &p) const {
        return point(x + p.x, y + p.y);
    }
    point operator - (const point &p) const {
        return point(x - p.x, y - p.y);
    }
    point operator * (double rate) const {
        return point(x * rate, y * rate);
    }
    point operator / (double rate) const {
        return point(x / rate, y / rate);
    }
    bool operator < (const point &p) const {
        if (p.x != x) return p.x > x;
        else return p.y > y;
    }
    int quadrant() const {
        int xs = sgn(x), ys = sgn(y);
        return xs == 0 && ys == 0 ? -1 : ((ys < 0 || ys == 0 && xs > 0) ? 0 :
1);
    }
    double length() const {
        return sqrt(x * x + y * y);
    }
    double operator ^ (const point &p) const {
        return x * p.x + y * p.y;
```

```
    }
    double operator * (const point &p) const {
        return x * p.y - y * p.x;
    }
    double Polar_Angle() const {
        return atan2(y, x);
    }
    point Rotate(double alpha) const {
        return point(x * cos(alpha) - y * sin(alpha), x * sin(alpha) + y *
cos(alpha));
    }
    point norm() const {
        return point(-y, x);
    }
    point unit(double r) const {
        double d = sqrt(x * x + y * y);
        return point(x / d * r, y / d * r);
    }
};

bool cmp(const point &a, const point &b)
{
    int lq = quadrant(), rq = p.quadrant();
    if (lq != rq) return lq < rq;
    int s = sgn(*this * p);
    return s ? s > 0 : sgn(length() - p.length()) < 0;
}

struct line
{
    point p, v;
    line() {}
    line(const point &_p, const point &_v): p(_p), v(_v) {}
    bool OnLeft(const point &A) const {
        return (A - p) * v < -eps;
    }
    bool OnRight(const point &A) const {
        return (A - p) * v > eps;
    }
    point GetIntersection(const line &l) {
        point u = p - l.p;
        return p + v * ((l.v * u) / (v * l.v));
    }
    bool operator < (const line &l) const {
        int r1 = sgn(v * l.v), r2 = sgn(v ^ l.v);
        return !r1 && r2 > 0 ? l.OnLeft(p) : cmp(v, l.v);
    }
};

struct Circle
{
    point c;
    double r;
    Circle(){}
    Circle(point _p,double _r):c(_p),r(_r){}
};

int n;
```

```
point p[MAXN], ch[MAXN];
//Please start at 0 when you input.(It can be convenient for figure the size)

int ConvexHull(point *p, int n)
{
    sort(p, p + n);
    int m = 0;
    for (int i = 0; i < n; i++)
    {
        while (m > 1 && (ch[m - 1] - ch[m - 2]) * (p[i] - ch[m - 2]) <= eps) --
m;
        ch[m++] = p[i];
    } // Lower hull
    int k = m;
    for (int i = n - 2; ~i; i--)
    {
        while (m > k && (ch[m - 1] - ch[m - 2]) * (p[i] - ch[m - 2]) <= eps) --
m;
        ch[m++] = p[i];
    } // Upper hull
    if (n > 1) m--;
    return m;
}

double Polygon_Area(point *p, int n) //Figure out the size of the Polygon
{
    double S = 0.0;
    int siz = ConvexHull(p, n);
    for (int i = 0; i < siz; i++) S += p[i % siz] * p[(i + 1) % siz];
    return S / 2.0;
}

double Rotating_Calipers() //Get the longest distance among points
{
    double res = 0.0;
    int q = 1;
    for (int i = 0; i < siz; i++)
    {
        while ((ch[i + 1] - ch[i]) * (ch[q + 1] - ch[i]) > (ch[i + 1] - ch[i]) *
(ch[q] - ch[i])) q = (q + 1) % siz;
        res = max(res, max(dis(ch[i], ch[q]), dis(ch[i + 1], ch[q + 1])));
    }
    return res;
}

// Is segment ab intersect with segment cd?
inline bool isIntersection(point a, point b, point c, point d)
{
    if (max(a.x, b.x) < min(c.x, d.x)) return false;
    if (max(a.y, b.y) < min(c.y, d.y)) return false;
    if (max(c.x, d.x) < min(a.x, b.x)) return false;
    if (max(c.y, d.y) < min(a.y, b.y)) return false;
    double p = (b - a) * (b - c);
    double q = (b - a) * (b - d);
    double r = (d - c) * (d - a);
    double s = (d - c) * (d - b);
    return (p * q <= eps) && (r * s <= eps);
}
```

```
// PIP Problem: Ray casting algorithm
// Point: (x, y), Polygon size: c
// Return val: 0: out, 1: in, 2: on the polygon
int in(int x, int y, int c)
{
    double k = sqrt(2), b = y - k * x;
    int cnt = 0; Point O = Point(x, y);
    for (int i = 0; i < c; i++)
        if (ch[i].x == x && ch[i].y == y) return 2;
    for (int i = 0; i < c; i++)
    {
        Point oa = ch[i] - O;
        Point ob = ch[(i + 1) % c] - O;
        if (fabs(oa * ob) <= eps) return 2;
    }
    for (int i = 0; i < c; i++)
    {
        double A = ch[(i + 1) % c].y - ch[i].y;
        double B = ch[i].x - ch[(i + 1) % c].x;
        double C = ch[(i + 1) % c] * ch[i];
        double lx = min(ch[i].x, ch[(i + 1)%c].x), rx = max(ch[i].x, ch[(i + 1)
% c].x);
        double ly = min(ch[i].y, ch[(i + 1)%c].y), ry = max(ch[i].y, ch[(i + 1)
% c].y);
        double nx = -(B * b + C) / (A + B * k);
        double ny = k * nx + b;
        if (ny > y) continue;
        if (lx - eps <= nx && nx <= rx + eps && ly - eps <= ny && ny <= ry +
eps) ++cnt;
    }
    return cnt & 1;
}

point r[MAXN];
line l[MAXN], q[MAXN];

inline int halfplaneIntersection(int n)
{
    sort(l, l + n);
    int h = 0, t = 0;
    q[t++] = l[0];
    for (int i = 1; i < n; i++)
    {
        while (h + 1 < t && l[i].OnRight(r[t - 2])) --t;
        while (h + 1 < t && l[i].OnRight(r[h])) ++h;
        q[t++] = l[i];
        if (!sgn(q[t - 2].v * q[t - 1].v))
        {
            t--;
            if (q[t - 1].OnLeft(l[i].p)) q[t - 1] = l[i];
        }
        if (h + 1 < t) r[t - 2] = q[t - 2].GetIntersection(q[t - 1]);
    }
    while (h + 1 < t && q[h].OnRight(r[t - 2])) --t;
    r[t - 1] = q[h].GetIntersection(q[t - 1]);
    int m = t - h;
    for (int i = 0; i < m; i++) r[i] = r[i + h];
```

```
    return m;
}

int dcmp(double x)
{
    if (fabs(x)<eps) return 0;
    return x<0?-1:1;
}

inline double TriangleCircleInsection(Circle C,point A,point B)
{
    point OA=A-C.c,OB=B-C.c;
    point BA=A-B, BC=C.c-B;
    point AB=B-A, AC=C.c-A;
    double DOA=OA.length(),DOB=OB.length(),DAB=AB.length(),r=C.r;
    if (!dcmp(OA*OB)) return 0;
    if (dcmp(DOA-C.r)<0&&dcmp(DOB-C.r)<0) return (OA*OB)*0.5;
    else if (DOB<r&&DOA>=r)
    {
        double x=((BA^BC)+sqrt(r*r*DAB*DAB-(BA*BC)*(BA*BC)))/DAB;
        double TS=(OA*OB)*0.5;
        return asin(TS*(1-x/DAB)*2/r/DOA)*r*r*0.5+TS*x/DAB;
    }
    else if(DOB>=r&&DOA<r)
    {
        double y=((AB^AC)+sqrt(r*r*DAB*DAB-(AB*AC)*(AB*AC)))/DAB;
        double TS=(OA*OB)*0.5;
        return asin(TS*(1-y/DAB)*2/r/DOB)*r*r*0.5+TS*y/DAB;
    }
    else if (fabs(OA*OB)>=r*DAB||(AB^AC)<=0||(BA^BC)<=0)
    {
        if((OA^OB)<0)
        {
            if ((OA*OB)<0) return (-pi-asin((OA*OB)/DOA/DOB))*r*r*0.5;
            else return (pi-asin((OA*OB)/DOA/DOB))*r*r*0.5;
        }
        else return asin((OA*OB)/DOA/DOB)*r*r*0.5;
    }
    else
    {
        double x=((BA^BC)+sqrt(r*r*DAB*DAB-(BA*BC)*(BA*BC)))/DAB;
        double y=((AB^AC)+sqrt(r*r*DAB*DAB-(AB*AC)*(AB*AC)))/DAB;
        double TS=(OA*OB)*0.5;
        return (asin(TS*(1-x/DAB)*2/r/DOA)+asin(TS*(1-
y/DAB)*2/r/DOB))*r*r*0.5+TS*((x+y)/DAB-1);
    }
}

inline double PolygonCircleInsection(point p[], double r)
{
    double ans=0.0;
    Circle C=Circle(point(0,0),r);
    for (int i=0;i<n;i++) ans+=TriangleCircleInsection(C,p[i],p[(i+1)%n]);
    return fabs(ans);
}

double simpson(double l, double r) {
  double mid = (l + r) / 2;
```

```
  return (r - l) * (f(l) + 4 * f(mid) + f(r)) / 6;
}
double asr(double l, double r, double eqs, double ans) {
  double mid = (l + r) / 2;
  double fl = simpson(l, mid), fr = simpson(mid, r);
  if (abs(fl + fr - ans) <= 15 * eqs)
    return fl + fr + (fl + fr - ans) / 15;
  return asr(l, mid, eqs / 2, fl) +
         asr(mid, r, eqs / 2, fr);
}
```

Dinic

```
#include <queue>
#include <cstdio>
#include <cstring>
#define MAXN 100
#define MAXM 20000
using namespace std;

struct link
{
    int to, flow, nxt;
};

link e[MAXM << 1];
int head[MAXN], cnt = 1;
int dpt[MAXN];
int S, T, ans;
queue <int> q;

inline void add(int u, int v, int f) //add edge u->v
{
    e[++cnt] = (link)
    {
        v, f, head[u]
    };
    head[u] = cnt;
    e[++cnt] = (link)
    {
        u, 0, head[v]
    };
    head[v] = cnt;
}

inline bool BFS()
{
    memset(dpt, -1, sizeof dpt);
    dpt[S] = 1;
    q.push(S);
    while (!q.empty())
    {
        int x = q.front();
        q.pop();
        for (int i = head[x]; ~i; i = e[i].nxt)
            if (e[i].flow && !~dpt[e[i].to])
            {
```

```
                dpt[e[i].to] = dpt[x] + 1;
                q.push(e[i].to);
            }
    }
    return dpt[T] != -1;
}

inline int Dinic(int x, int flow)
{
    int left = flow;
    if (x == T) return flow;
    for (int i = head[x]; ~i && left; i = e[i].nxt)
        if (e[i].flow && dpt[e[i].to] == dpt[x] + 1)
        {
            int t = Dinic(e[i].to, min(left, e[i].flow));
            e[i].flow -= t;
            e[i ^ 1].flow += t;
            left -= t;
        }
    if (left) dpt[x] = -1;
    return flow - left;
}

int main()
{
    memset(head, -1, sizeof head);
    //Make_Graph
    while (BFS()) ans += Dinic(S, ~(1 << 31));
    printf("%d\n", ans);
    return 0;
}
```

费用流

```
#include <bits/stdc++.h>
#define MAXN 405
#define MAXM 15005
using namespace std;

struct link
{
    int to, rev, flow, val;
    link(){}
    link(int _to, int _rev, int _flow, int _val):
        to(_to), rev(_rev), flow(_flow), val(_val){}
};

vector <link> g[MAXN];
int n, m, u, v, s, t, S, T, dis[MAXN], flow, ans, ptr[MAXN];
bool vis[MAXN];
queue <int> q;

inline void add(int u, int v, int c, int f)
{
    g[u].push_back(link(v, g[v].size(), f, c));
    g[v].push_back(link(u, int(g[u].size()) - 1, 0, -c));
}
```

```
inline bool SPFA()
{
    memset(dis, 0x7f, sizeof dis);
    memset(vis, false, sizeof vis);
    q.push(S); dis[S] = 0;
    while (!q.empty())
    {
        int u = q.front(); q.pop(); vis[u] = false;
        for (auto& e : g[u])
            if (e.flow && dis[e.to] > dis[u] + e.val)
            {
                dis[e.to] = dis[u] + e.val;
                if (!vis[e.to])
                {
                    vis[e.to] = true;
                    q.push(e.to);
                }
            }
    }
    return dis[T] != 0x7f7f7f7f;
}

inline int Dinic(int x, int flow)
{
    int left = flow;
    if (x == T) return flow;
    vis[x] = true;
    for (int& i = ptr[x]; i < int(g[x].size()) && left && flow; i++)
    {
        link& e = g[x][i];
        if (dis[e.to] == dis[x] + e.val && e.flow && !vis[e.to])
        {
            int t = Dinic(e.to, min(e.flow, left));
            e.flow -= t; g[e.to][e.rev].flow += t; left -= t;
        }
    }
    if (!left) vis[x] = false;
    return flow - left;
}

int main()
{
    scanf("%d %d",&n,&m); S = 1; T = n;
    for (int i=1;i<=m;i++)
    {
        scanf("%d %d %d %d", &u, &v, &s, &t);
        add(u, v, t, s);
    }
    while (SPFA())
    {
        memset(ptr, 0, sizeof ptr);
        int d = Dinic(S, ~(1 << 31));
        flow += d;
        ans += d * dis[T];
    }
    printf("%d %d\n", flow, ans);
    return 0;
```

```
}
```

Tarjan

```
#include <bits/stdc++.h>
#define MAXN 100010
#define MAXM 100010
using namespace std;

namespace Tarjan_SCC
{
    struct link
    {
        int to,val,nxt;
    };

    link e[MAXM];
    int head[MAXN],cnt;
    int n,m,T,u,v,w,scc,top;
    int dfn[MAXN],low[MAXN],bel[MAXN],sta[MAXN];
    bool vis[MAXN];

    inline void add(int u,int v,int w) // One way
    {
        e[cnt]=(link){v,w,head[u]};head[u]=cnt++;
    }

    inline void Tarjan(int x,int fa)
    {
        dfn[x]=low[x]=++T;sta[++top]=x;
        for (int i=head[x];~i;i=e[i].nxt)
        {
            if (vis[e[i].to]||e[i].to==fa) continue;
            if (dfn[e[i].to]) low[x]=min(low[x],dfn[e[i].to]);
            else
            {
                Tarjan(e[i].to,x);
                low[x]=min(low[x],low[e[i].to]);
            }
        }
        if (dfn[x]==low[x])
        {
            int t;scc++;
            do{
                vis[t=sta[top--]]=true;
                bel[t]=scc;
            } while (t!=x);
        }
        return ;
    }
}

namespace Tarjan_EBCC // Must 1-based
{
    vector <pair<int,int> > g[MAXN];
    int n,m,T,u,v,w,bcc,top;
    int dfn[MAXN],low[MAXN],vbel[MAXN],ebel[MAXM],sta[MAXN+MAXM];
```

```
    bool vis[MAXM];

    inline void add(int u,int v,int i) // Two way
    {
        g[u].push_back({v,i});
        g[v].push_back({u,i});
    }

    inline void Tarjan(int x,int fa)
    {
        dfn[x]=low[x]=++T;sta[++top]=x;
        for (auto i:g[x])
        {
            int v=i.first,pt=i.second;
            if (vis[pt]) continue;
            vis[pt]=true;sta[++top]=-pt;
            if (dfn[v]) low[x]=min(low[x],dfn[v]);
            else
            {
                Tarjan(v,pt);
                low[x]=min(low[x],low[v]);
            }
        }
        if (dfn[x]==low[x])
        {
            for (++bcc;sta[top]!=-fa;--top)
                sta[top]>0?vbel[sta[top]]=bcc:ebel[-sta[top]]=bcc;
            if (top) --top;
        }
        return ;
    }
}

namespace Tarjan_VBCC_BCTree
{
    struct link
    {
        int to,nxt;
    };

    link e[MAXN<<1];
    int head[MAXN],cnt,bel[MAXN];
    int n,m,Q,T,R,u,v,bcc,low[MAXN],dfn[MAXN],sta[MAXN],top;
    vector <int> s[MAXN],g[MAXN<<1];
    bool cut[MAXN];

    inline void add(int u,int v)
    {
        e[cnt]=(link){v,head[u]};head[u]=cnt++;
        e[cnt]=(link){u,head[v]};head[v]=cnt++;
    }

    inline void addg(int u,int v)
    {
        g[u].push_back(v);g[v].push_back(u);
        return ;
    }
```

```
    inline void Tarjan(int x,int fa)
    {
        dfn[x]=low[x]=++T;sta[++top]=x;
        for (int i=head[x];~i;i=e[i].nxt)
        {
            if (!dfn[e[i].to])
            {
                Tarjan(e[i].to,x);
                low[x]=min(low[x],low[e[i].to]);
                if (low[e[i].to]>=dfn[x])
                {
                    cut[x]|=(dfn[x]>1||dfn[e[i].to]>2);
                    ++bcc;int y;
                    do{
                        y=sta[top--];s[bcc].push_back(y);
                    }while (y!=e[i].to);
                    s[bcc].push_back(x);
                }
            }
            else
                low[x]=min(low[x],dfn[e[i].to]);
        }
        return ;
    }

    inline void BuildTree()
    {
        int cnt=0;
        for (int i=1;i<=n;i++) if (cut[i]) bel[i]=++cnt;
        for (int i=1;i<=bcc;i++)
        {
            ++cnt;
            for (auto x:s[i])
            {
                if (cut[x]) addg(bel[x],cnt);
                else bel[x]=cnt;
            }
        }
        return ;
    }
}

inline void read(int &x)
{
    x=0;char ch=getchar();
    while (ch<'0'||ch>'9') ch=getchar();
    while (ch>='0'&&ch<='9') x=(x<<3)+(x<<1)+ch-'0',ch=getchar();
    return ;
}

int main()
{
    using namespace Tarjan_SCC;
    // using namespace Tarjan_EBCC;
    // using namespace Tarjan_VBCC_BCTree;
    memset(head,-1,sizeof head);
    // Make Graph
    for (int i=1;i<=n;i++) if (!dfn[i]) Tarjan(i,0);
```

```
    return 0;
}
```

$O(n)$ 求 $\sum i^k$

```
#include <bits/stdc++.h>
#define MAXN 1000005
using namespace std;

const long long M=1e9+7;

int n,k,pri[MAXN],f[MAXN],tot;
bool not_prime[MAXN];

inline void fadd(int &x,int y){x+=y;if (x>=M) x-=M;return ;}

inline int fpow(int a,int b)
{
    int r=1;
    for (;b;b>>=1,a=1LL*a*a%M) if (b&1) r=1LL*r*a%M;
    return r;
}

void Linear_Shaker(int n)
{
    f[1]=1;
    for (int i=2;i<=n;i++)
    {
        if (!not_prime[i]) pri[++tot]=i,f[i]=fpow(i,k);
        for (int j=1;i*pri[j]<=n;j++)
        {
            not_prime[i*pri[j]]=true;
            f[i*pri[j]]=1LL*f[i]*f[pri[j]]%M;
            if (!(i%pri[j])) break;
        }
    }
    //for (int i=1;i<=n;i++) fadd(f[i],f[i-1]);
    return ;
}

int main()
{
    scanf("%d %d",&n,&k);
    Linear_Shaker(n);
    for (int i=1;i<=n;i++) assert(f[i]==fpow(i,k));
    return 0;
}
```

Segment Tree Beats!

```
//If you want to write it right, you'd better not modify ANY CHARACTER in this
code.
#include <bits/stdc++.h>
#define MAXN 500010
using namespace std;
```

```
struct mxinfo
{
    int m,s,t;
    mxinfo(){}
    mxinfo(int _,int __,int ___):m(_),s(__),t(___){}
    mxinfo operator + (const mxinfo &a)const{
        mxinfo res;
        if (m>a.m)
        {
            res.m=m;res.t=t;
            res.s=max(s,a.m);
        }
        else if (m<a.m)
        {
            res.m=a.m;res.t=a.t;
            res.s=max(a.s,m);
        }
        else
        {
            res.m=a.m;res.t=t+a.t;
            res.s=max(s,a.s);
        }
        return res;
    }
};

struct mninfo
{
    int m,s,t;
    mninfo(){}
    mninfo(int _,int __,int ___):m(_),s(__),t(___){}
    mninfo operator + (const mninfo &a)const{
        mninfo res;
        if (m<a.m)
        {
            res.m=m;res.t=t;
            res.s=min(s,a.m);
        }
        else if (m>a.m)
        {
            res.m=a.m;res.t=a.t;
            res.s=min(a.s,m);
        }
        else
        {
            res.m=a.m;res.t=t+a.t;
            res.s=min(s,a.s);
        }
        return res;
    }
};

struct SegmentTreeBeats
{
    int p,r,m;long long x,lazy;
    mxinfo mx;mninfo mn;
};
```

```
SegmentTreeBeats tree[1<<20];
int n,m,o,l,r,x,a[MAXN];

inline void PushUp(int u)
{
    tree[u].x=tree[u<<1].x+tree[u<<1|1].x;
    tree[u].mn=tree[u<<1].mn+tree[u<<1|1].mn;
    tree[u].mx=tree[u<<1].mx+tree[u<<1|1].mx;
    return ;
}

inline void addmax(int u,int v)
{
    tree[u].x+=1LL*tree[u].mn.t*(v-tree[u].mn.m);
    tree[u].mn.m=v;tree[u].mx.m=max(tree[u].mx.m,v);
    if (tree[u].mx.m==tree[u].mn.m)
    {
        tree[u].x=1LL*(tree[u].r-tree[u].p)*v;
        tree[u].mn=mninfo(tree[u].mn.m,~(1<<31),tree[u].r-tree[u].p);
        tree[u].mx=mxinfo(tree[u].mx.m,  1<<31 ,tree[u].r-tree[u].p);
    }
    else tree[u].mx.s=max(tree[u].mx.s,v);
    return ;
}

inline void addmin(int u,int v)
{
    tree[u].x-=1LL*tree[u].mx.t*(tree[u].mx.m-v);
    tree[u].mx.m=v;tree[u].mn.m=min(tree[u].mn.m,v);
    if (tree[u].mx.m==tree[u].mn.m)
    {
        tree[u].x=1LL*(tree[u].r-tree[u].p)*v;
        tree[u].mn=mninfo(tree[u].mn.m,~(1<<31),tree[u].r-tree[u].p);
        tree[u].mx=mxinfo(tree[u].mx.m,  1<<31 ,tree[u].r-tree[u].p);
    }
    else tree[u].mn.s=min(tree[u].mn.s,v);
    return ;
}

inline void Lazy(int u,long long v)
{
    tree[u].lazy+=v;tree[u].x+=(tree[u].r-tree[u].p)*v;
    tree[u].mx.m+=v;if (tree[u].mx.s!=  1<<31 ) tree[u].mx.s+=v;
    tree[u].mn.m+=v;if (tree[u].mn.s!=~(1<<31)) tree[u].mn.s+=v;
    return ;
}

inline void PushDown(int u)
{
    if (tree[u].lazy)
    {
        Lazy(u<<1,tree[u].lazy);
        Lazy(u<<1|1,tree[u].lazy);
        tree[u].lazy=0LL;
    }
    if (tree[u<<1].mx.s<tree[u].mx.m&&tree[u].mx.m<tree[u<<1].mx.m)
addmin(u<<1,tree[u].mx.m);
```

```
    if (tree[u<<1].mn.m<tree[u].mn.m&&tree[u].mn.m<tree[u<<1].mn.s)
addmax(u<<1,tree[u].mn.m);
    if (tree[u<<1|1].mx.s<tree[u].mx.m&&tree[u].mx.m<tree[u<<1|1].mx.m)
addmin(u<<1|1,tree[u].mx.m);
    if (tree[u<<1|1].mn.m<tree[u].mn.m&&tree[u].mn.m<tree[u<<1|1].mn.s)
addmax(u<<1|1,tree[u].mn.m);
    return ;
}

void BuildTree(int u)
{
    if (tree[u].p+1==tree[u].r)
    {
        tree[u].x=a[tree[u].p];
        tree[u].mn=mninfo(a[tree[u].p],~(1<<31),1);
        tree[u].mx=mxinfo(a[tree[u].p],  1<<31 ,1);
        return ;
    }
    tree[u].m=tree[u].p+tree[u].r>>1;
    tree[u<<1].p=tree[u].p;tree[u<<1].r=tree[u].m;BuildTree(u<<1);
    tree[u<<1|1].p=tree[u].m;tree[u<<1|1].r=tree[u].r;BuildTree(u<<1|1);
    PushUp(u);
    return ;
}

inline void add(int u,int l,int r,int v)
{
    if (tree[u].p==l&&tree[u].r==r)
    {
        Lazy(u,v);
        return ;
    }
    PushDown(u);
    if (r<=tree[u].m) add(u<<1,l,r,v);
    else if (tree[u].m<=l) add(u<<1|1,l,r,v);
    else
    {
        add(u<<1,l,tree[u].m,v);
        add(u<<1|1,tree[u].m,r,v);
    }
    PushUp(u);
    return ;
}

inline void changemin(int u,int l,int r,int v)
{
    if (v>=tree[u].mx.m) return ;
    if (tree[u].p==l&&tree[u].r==r&&v>tree[u].mx.s){addmin(u,v);return ;}
    if (tree[u].p+1==tree[u].r) return ;
    PushDown(u);
    if (r<=tree[u].m) changemin(u<<1,l,r,v);
    else if (tree[u].m<=l) changemin(u<<1|1,l,r,v);
    else
    {
        changemin(u<<1,l,tree[u].m,v);
        changemin(u<<1|1,tree[u].m,r,v);
    }
    PushUp(u);
```

```
    return ;
}

inline void changemax(int u,int l,int r,int v)
{
    if (v<=tree[u].mn.m) return ;
    if (tree[u].p==l&&tree[u].r==r&&v<tree[u].mn.s){addmax(u,v);return ;}
    if (tree[u].p+1==tree[u].r) return ;
    PushDown(u);
    if (r<=tree[u].m) changemax(u<<1,l,r,v);
    else if (tree[u].m<=l) changemax(u<<1|1,l,r,v);
    else
    {
        changemax(u<<1,l,tree[u].m,v);
        changemax(u<<1|1,tree[u].m,r,v);
    }
    PushUp(u);
    return ;
}

inline long long query(int u,int l,int r)
{
    if (tree[u].p==l&&tree[u].r==r) return tree[u].x;
    PushDown(u);
    if (r<=tree[u].m) return query(u<<1,l,r);
    else if (tree[u].m<=l) return query(u<<1|1,l,r);
    else return query(u<<1,l,tree[u].m)+query(u<<1|1,tree[u].m,r);
}

inline int qmin(int u,int l,int r)
{
    if (tree[u].p==l&&tree[u].r==r) return tree[u].mn.m;
    PushDown(u);
    if (r<=tree[u].m) return qmin(u<<1,l,r);
    else if (tree[u].m<=l) return qmin(u<<1|1,l,r);
    else return min(qmin(u<<1,l,tree[u].m),qmin(u<<1|1,tree[u].m,r));
}

inline int qmax(int u,int l,int r)
{
    if (tree[u].p==l&&tree[u].r==r) return tree[u].mx.m;
    PushDown(u);
    if (r<=tree[u].m) return qmax(u<<1,l,r);
    else if (tree[u].m<=l) return qmax(u<<1|1,l,r);
    else return max(qmax(u<<1,l,tree[u].m),qmax(u<<1|1,tree[u].m,r));
}

inline void read(int &x)
{
    x=0;bool f=false;char ch=getchar();
    while (ch<'0'||ch>'9'){if (ch=='-') f=true;ch=getchar();}
    while (ch>='0'&&ch<='9') x=(x<<3)+(x<<1)+ch-'0',ch=getchar();
    if (f) x=-x;return ;
}

int main()
{
    read(n);
```

```
    for (int i=1;i<=n;i++) read(a[i]);
    tree[1].p=1;tree[1].r=n+1;BuildTree(1);
    read(m);
    while (m--)
    {
        read(o);read(l);read(r);if (o<=3) read(x);
        if (o==1) add(1,l,r+1,x);
        else if (o==2) changemax(1,l,r+1,x); // ai -> max(ai,x)
        else if (o==3) changemin(1,l,r+1,x); // ai -> min(ai,x)
        else if (o==4) printf("%lld\n",query(1,l,r+1));
        else if (o==5) printf("%d\n",qmax(1,l,r+1));
        else printf("%d\n",qmin(1,l,r+1));
    }
    return 0;
}
```

Splay 区间翻转

```
#include <cstdio>
#include <algorithm>
#define MAXN 100005
#define LC(x) tree[(x)].ch[0]
#define RC(x) tree[(x)].ch[1]
using namespace std;

struct SplayTree
{
    int ch[2],f,siz;bool rev;long long lazy,w,sw;
    SplayTree(){}
    SplayTree(int _f,long long _w):
        f(_f),siz(1),w(_w),rev(false),lazy(0LL),sw(_w)
        {ch[0]=ch[1]=0;}
};

SplayTree tree[MAXN];
int n,T,root,top,o,l,r,x,p[MAXN],sta[MAXN];
long long w[MAXN],all,v;

inline void Push_Up(int x)
{
    tree[x].siz=tree[LC(x)].siz+tree[RC(x)].siz+1;
    tree[x].sw=tree[LC(x)].sw+tree[RC(x)].sw+tree[x].w;
    return ;
}

inline void Push_Down(int x)
{
    if (tree[x].rev)
    {
        tree[LC(x)].rev^=1;tree[RC(x)].rev^=1;tree[x].rev=false;
        swap(LC(x),RC(x));
    }
    if (tree[x].lazy!=0)
    {
        tree[LC(x)].w+=tree[x].lazy;tree[LC(x)].lazy+=tree[x].lazy;
        tree[LC(x)].sw+=tree[LC(x)].siz*tree[x].lazy;
        tree[RC(x)].w+=tree[x].lazy;tree[RC(x)].lazy+=tree[x].lazy;
```

```
        tree[RC(x)].sw+=tree[RC(x)].siz*tree[x].lazy;
        tree[x].lazy=0LL;
    }
    return ;
}

inline void Rotate(int x,int &k)
{
    int y=tree[x].f,z=tree[y].f,l=RC(y)==x,r=l^1;
    if (y==k) k=x;
    else tree[z].ch[RC(z)==y]=x;
    tree[x].f=z;tree[y].f=x;tree[tree[x].ch[r]].f=y;
    tree[y].ch[l]=tree[x].ch[r];tree[x].ch[r]=y;
    Push_Up(y);Push_Up(x);
    return ;
}

inline void Splay(int x,int &k)
{
    int y,z;
    sta[sta[0]=1]=x;
    for (int i=tree[x].f;i;i=tree[i].f) sta[++sta[0]]=i;
    while (sta[0]) Push_Down(sta[sta[0]--]);
    while (x!=k)
    {
        y=tree[x].f;z=tree[y].f;
        if (y!=k)
        {
            if (LC(z)==y^LC(y)==x) Rotate(x,k);
            else Rotate(y,k);
        }
        Rotate(x,k);
    }
    return ;
}

inline int find(int k)
{
    int x=root;
    while (x)
    {
        Push_Down(x);
        if (k==tree[LC(x)].siz) return x;
        if (k<=tree[LC(x)].siz) x=LC(x);
        else k-=tree[LC(x)].siz+1,x=RC(x);
    }
    return -1;
}

inline int BuildTree(int l,int r,int f)
{
    if (l>r) return 0;
    int pos=++top,m=l+r>>1;tree[pos]=SplayTree(f,w[m]);
    if (l==r) return pos;
    LC(pos)=BuildTree(l,m-1,pos);RC(pos)=BuildTree(m+1,r,pos);
    Push_Up(pos);return pos;
}
```

```
inline void add(int l,int r,long long v)
{
    int x=find(l-1),y=find(r+1);
    Splay(x,root);Splay(y,RC(x));
    tree[LC(y)].lazy+=v;tree[LC(y)].w+=v;
    tree[LC(y)].sw+=tree[LC(y)].siz*v;
    return ;
}

inline void reverse(int l,int r)
{
    if (l>=r) return ;
    int x=find(l-1),y=find(r+1);
    Splay(x,root);Splay(y,RC(x));int rt=LC(y);
    sta[sta[0]=1]=rt;
    for (int i=tree[rt].f;i;i=tree[i].f) sta[++sta[0]]=i;
    while (sta[0]) Push_Down(sta[sta[0]--]);
    tree[rt].rev^=1;
    return ;
}

inline int getPos(int x,int rt)
{
    int pos=tree[LC(x)].siz+1;
    for (;x!=rt;x=tree[x].f)
        if (RC(tree[x].f)==x) pos+=tree[LC(tree[x].f)].siz+1;
    return pos;
}

inline int query(long long all)
{
    int x=find(0),y=find(n+1);
    Splay(x,root);Splay(y,RC(x));int rt=LC(y);int r=rt;
    while (true)
    {
        Push_Down(r);
        if (tree[LC(r)].sw>=all) r=LC(r);
        else if (tree[LC(r)].sw+tree[r].w>=all) return getPos(r,rt);
        else all-=tree[LC(r)].sw+tree[r].w,r=RC(r);
    }
    return -1;
}

inline void read(int &x)
{
    x=0;char ch=getchar();bool f=false;
    while (ch<'0'||ch>'9')
    {
        if (ch=='A'){x=0;return ;}
        if (ch=='B'){x=1;return ;}
        if (ch=='-') f=true;
        ch=getchar();
    }
    while (ch>='0'&&ch<='9') x=(x<<3)+(x<<1)+ch-'0',ch=getchar();
    if (f) x=-x;return ;
}

int main()
```

```
{
    read(n);read(T);
    for (int i=1;i<=n;i++) read(x),w[i]=x,all+=w[i],w[i]<<=1;
    for (int i=1;i<=n;i++) read(p[i]);
    root=BuildTree(0,n+1,0);
    printf("%d\n",p[query(all)]);
    while (T--)
    {
        read(o);
        if (!o)
        {
            read(l);read(r);read(x);v=x;all+=(r-l+1)*v;
            add(l,r,v*2);
        }
        else
        {
            read(l);read(r);
            if (l<r){reverse(l,r);reverse(l,r-1);}
            else{reverse(r,l);reverse(r+1,l);}
        }
        printf("%d\n",p[query(all)]);
    }
    return 0;
}
```

动态图连通性（离线）

```
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<algorithm>
#include<cstring>
#include<stack>
using namespace std;
const int N=2000005;

int n,m,q,ans[N];

int t[N];
inline int lowbit(int x){return x&(-x);}
void add(int i,int p) {for (;i<=m;i+=lowbit(i)) t[i]+=p;}
int sum(int i){int ans=0;for (;i;i-=lowbit(i)) ans+=t[i];return ans;}

struct aa
{
    int a,b,id;
    bool operator <(const aa &c) const
    {
        return b<c.b;
    }
}Q[N],bian[N];

int fa[N],ch[N][2],rev[N],val[N],mi[N],id[N],ql[N],qr[N];
stack<int> s;
bool isroot(int x)
{
    return ch[fa[x]][0]!=x&&ch[fa[x]][1]!=x;
```

```
}
void up(int x)
{
    mi[x]=val[x],id[x]=x;
    if (ch[x][0]&&mi[ch[x][0]]<mi[x]) mi[x]=mi[ch[x][0]],id[x]=id[ch[x][0]];
    if (ch[x][1]&&mi[ch[x][1]]<mi[x]) mi[x]=mi[ch[x][1]],id[x]=id[ch[x][1]];
}
void down(int x)
{
    if (rev[x])
    {
        rev[x]^=1,rev[ch[x][0]]^=1,rev[ch[x][1]]^=1;
        swap(ch[x][0],ch[x][1]);
    }
}
void rot(int x)
{
    int y=fa[x],z=fa[y],l,r;
    if (ch[y][0]==x) l=0;else l=1;r=l^1;
    if (!isroot(y))
        if (ch[z][0]==y) ch[z][0]=x;else ch[z][1]=x;
    fa[x]=z,fa[y]=x,fa[ch[x][r]]=y;
    ch[y][l]=ch[x][r],ch[x][r]=y;
    up(y);
}
void splay(int x)
{
    int y=x,z;
    while (!isroot(y)) s.push(y),y=fa[y];s.push(y);
    while (!s.empty()) down(s.top()),s.pop();
    while (!isroot(x))
    {
        y=fa[x],z=fa[y];
        if (!isroot(y))
        if (ch[y][0]==x^ch[z][0]==y) rot(x);else rot(y);
        rot(x);
    }
    up(x);
}
void access(int x)
{
    int t=0;
    while (x)
    {
        splay(x);down(x);
        ch[x][1]=t;up(x);
        t=x,x=fa[x];
    }
}
int find(int x)
{
    access(x),splay(x);
    while (ch[x][0]) x=ch[x][0];
    return x;
}
void to_rt(int x)
{
    access(x),splay(x),rev[x]^=1;
```

```
}
void link(int x,int y)
{
    to_rt(x),fa[x]=y;
}
void cut(int x,int y)
{
    to_rt(x),access(y),splay(y);ch[y][0]=fa[x]=0;up(y);
}
int find_mi(int u,int v)
{
    to_rt(u),access(v),splay(v);
    return id[v];
}
void addedge(int ii)
{
    int u=bian[ii].a,v=bian[ii].b,fu,fv;
    if (u==v) return ;
    fu=find(u),fv=find(v);
    add(ii,1);
    if (fu!=fv)
    {
        link(u,n+ii);
        link(v,n+ii);
        return ;
    }
    int pos=find_mi(u,v)-n;
    add(pos,-1);
    cut(pos+n,bian[pos].a);
    cut(pos+n,bian[pos].b);
    link(ii+n,u);
    link(ii+n,v);
}
void clear()
{
    for (int i=0;i<=m+5;i++) t[i]=0;
    for (int i=0;i<=n+m+5;i++) fa[i]=ch[i][0]=ch[i][1]=rev[i]=id[i]=mi[i]=0;
}
const int SIZ =150000000+3;
char buf1[SIZ];
char *p1=buf1,*p2=buf1;
inline char readchar()
{
    if(p1==p2)p1=buf1,p2=buf1+fread(buf1,1,SIZ,stdin);
    return p1==p2?EOF:*p1++;
}
inline void read(int &ret)
{
    int c;ret=0;
    while((c=readchar())> '9'|c< '0');ret=c-'0';
    while((c=readchar())>='0'&&c<='9')ret=ret*10+c-'0';
    return ;
}
int main()
{
    int T,x,y;
    read(T);
    while (T--)
```

```
    {
        read(n);read(m);read(q);
        for (int i=0;i<=n+5;i++) mi[i]=val[i]=1e9,id[i]=i;
        for (int i=1;i<=m;i++)
        {
            read(bian[i].a);read(bian[i].b);
            mi[i+n]=val[n+i]=i,id[i+n]=i+n;
        }
        for (int i=1,l,r;i<=q;i++)
        {
            read(l);read(r);
            ql[i]=l%m+1;qr[i]=r%m+1;
            if (ql[i]>qr[i]) swap(ql[i],qr[i]);
            Q[i]={ql[i],qr[i],i};
            ql[i+q]=(l^1)%m+1;qr[i+q]=(r^1)%m+1;
            if (ql[i+q]>qr[i+q]) swap(ql[i+q],qr[i+q]);
            Q[i+q]={ql[i+q],qr[i+q],i+q};
        }
        q<<=1;
        sort(Q+1,Q+q+1);
        int j=1;
        for (int i=1;i<=m;i++)
        {
            addedge(i);
            for (;Q[j].b==i;j++)
                ans[Q[j].id]=n-(sum(m)-sum(Q[j].a-1));
        }
        int las=0;
        q>>=1;
        for (int i=1;i<=q;i++)
        {
            int l,r,cnt;
            if (!las)
            {
                l=ql[i];r=qr[i];cnt=ans[i];
            }
            else
            {
                l=ql[i+q];r=qr[i+q];cnt=ans[i+q];
            }
            if (r-l+1<n-cnt+1)
            {
                puts("No");las=0;
            }
            else
            {
                puts("Yes");las=1;
            }
        }
        clear();
    }
    return 0;
}
```

珂朵莉树

```
struct ChthollyTree
```

```
{
    int l,r;
    mutable int v;
    ChthollyTree(int L,int R=-1,int V=0):l(L),r(R),v(V){}
    bool operator < (const ChthollyTree& o) const {
        return l<o.l;
    }
};

set <ChthollyTree> s;
// every node indicates a segment with same value.

int query(int pos)
{
    auto it=s.lower_bound(ChthollyTree(pos));
    if (it!=s.end()&&it->l==pos) return it->v;
    --it;
    return it->v;
}

int query(int l,int r,int v) // query \sum{l to r} [a_i = v]
{
    int res=0;
    auto itr=split(r+1),itl=split(l);
    for (;itl!=itr;++itl) if (itl->v==v) res+=itl->r-itl->l+1;
    return res;
}

void assign_val(int l,int r,int val)
{
    auto itr=split(r+1),itl=split(l);
    s.erase(itl,itr);
    s.insert(ChthollyTree(l,r,val));
    return ;
}

void makeSqrt(int l,int r) // i in [l, r]: ai -> sqrt(ai)
{
    auto itr=split(r+1),itl=split(l);
    int pre=-1;bool same=true;
    for (;itl!=itr;++itl)
    {
        itl->v=(int)sqrt(itl->v);
        if (~pre&&itl->v!=pre) same=false;
        pre=itl->v;
    }
    if (same)
    {
        itl=split(l);
        s.erase(itl,itr);
        s.insert(ChthollyTree(l,r,pre));
    }
    return ;
}
```

DMST

```cpp
// N = 2n

typedef long long ll;

const ll inf = INT_MAX;
struct edge {
    int u, v;
    ll w;
} es[N];
int ls[N], rs[N], dis[N];
ll val[N], tag[N];

void update(int x, int t) {
    val[x] += t;
    tag[x] += t;
}

void push_down(int x) {
    if (ls[x])
        update(ls[x], tag[x]);
    if (rs[x])
        update(rs[x], tag[x]);
    tag[x] = 0;
}

int merge(int x, int y) {
    if (!x || !y)
        return x | y;
    if (val[x] > val[y])
        swap(x, y);
    push_down(x);
    rs[x] = merge(rs[x], y);
    if (dis[ls[x]] < dis[rs[x]])
        swap(ls[x], rs[x]);
    dis[x] = dis[rs[x]] + 1;
    return x;
}

int f[N];
int find(int x) { return f[x] ? f[x] = find(f[x]) : x; }
int top[N], fa[N], ine[N], nc;
vector<int> ch[N];

int gn(int cnt) {
    while (cnt--) {
        int x = ++nc;
        top[x] = fa[x] = ine[x] = f[x] = 0;
        ch[x].clear();
    }
    return nc;
}

void contract(int n, int m) {
    nc = 0;
    gn(n);
    for (int i = 1; i <= n; ++i) es[++m] = { i % n + 1, i, inf };
    for (int i = 1; i <= m; ++i) val[i] = es[i].w;
    fill_n(ls + 1, m, 0);
```

```
    fill_n(rs + 1, m, 0);
    fill_n(tag + 1, m, 0);
    fill_n(dis + 1, m, 1);
    for (int i = 1; i <= m; ++i) top[es[i].v] = merge(top[es[i].v], i);
    int x = 1;
    while (top[x]) {
        int i = top[x], y = find(es[i].u);
        push_down(top[x]);
        top[x] = merge(ls[i], rs[i]);
        if (y != x) {
            ine[x] = i;
            if (!ine[es[i].u])
                x = y;
            else {
                for (int z = gn(1); x != z; x = find(es[ine[x]].u)) {
                    fa[x] = f[find(x)] = z;
                    ch[z].push_back(x);
                    if (top[x]) update(top[x], -val[ine[x]]);
                    top[z] = merge(top[z], top[x]);
                }
            }
        }
    }
}

int fa2[N], ine2[N];
vector<int> expand(int n, int r) {
    copy_n(fa + 1, nc, fa2 + 1);
    copy_n(ine + 1, nc, ine2 + 1);
    vector<int> s, res;
    s.push_back(r);
    while (!s.empty()) {
        int x = s.back();
        s.pop_back();
        int i = ine2[x];
        ine2[es[i].v] = i;
        for (int y = es[i].v; fa2[y]; y = fa2[y]) {
            for (int z : ch[fa2[y]]) {
                if (z == y)
                    continue;
                fa2[z] = 0;
                if (!ch[z].empty())
                    s.push_back(z);
            }
        }
    }
    for (int i = 1; i <= n; ++i)
        if (i != r)
            res.push_back(ine2[i]);
    return res;
}

//  Example:
//  After reading edges into es[1~m]...
contract(n, m);
vector<int> res = expand(n, r); // All the n - 1 edges
ll ans = 0; bool fail = 0;
for (int i : res) {
```

```
    if (es[i].w == inf)
        fail = 1;
    ans += es[i].w;
}

if (fail) cout << -1 << endl;
else cout << ans << endl;
```

k 短路

```
// M = mlogm
typedef long long ll;

namespace kth_shortest_path {

const ll inf = LLONG_MAX;
typedef pair<ll, int> pli;
struct edge { int v, i; ll w; };

//  Persistent Leftist Tree
int ls[M], rs[M], dep[M], nc; pli val[M];
int gn(pli v, int q = 0) {
    int p = ++nc;
    ls[p] = ls[q]; rs[p] = rs[q];
    dep[p] = q ? dep[q] : 1;
    val[p] = q ? val[q] : v;
    return p;
}

int merge(int x, int y) {
    if (!x || !y) return x | y;
    if (val[x] > val[y]) swap(x, y);
    int z = gn({}, x); rs[z] = merge(rs[z], y);
    if (dep[ls[z]] < dep[rs[z]]) swap(ls[z], rs[z]);
    dep[z] = dep[rs[z]] + 1; return z;
}


vector<edge> g[N], gr[N]; vector<int> gt[N];
ll dis[N]; int pre[N], rt[N];

ll solve(int n, int m, int s, int t, int k,
          const vector<pair<pair<int, int>, ll>>& es) {
    for (int i = 1; i <= n; ++i) {
        g[i].resize(0);
        gr[i].resize(0);
        gt[i].resize(0);
    }
    for (int i = 0; i != es.size(); ++i) {
        pair<pair<int, int>, ll> p = es[i];
        int u = p.first.first, v = p.first.second;
        ll w = p.second;
        g[u].push_back({ v, i, w });
        gr[v].push_back({ u, i, w });
    }

    //  Dijkstra
```

```
    fill_n(dis + 1, n, inf); fill_n(pre + 1, n, -1);
    priority_queue<pli, vector<pli>, greater<pli>> pq;
    pq.push({ dis[t] = 0, t });
    while (!pq.empty()) {
        pli p = pq.top(); pq.pop();
        int u = p.second; ll du = p.first;
        if (du > dis[u]) continue;
        for (edge e : gr[u]) if (dis[e.v] > du + e.w)
            pq.push({ dis[e.v] = du + e.w, e.v }), pre[e.v] = e.i;
    }

    if (dis[s] == inf) return -1;

    //  Building heaps
    nc = 0; fill_n(rt + 1, n, 0);
    for (int u = 1; u <= n; ++u) {
        if (dis[u] == inf) continue;
        for (edge e : g[u]) {
            if (e.i == pre[u])
                gt[e.v].push_back(u);
            else if (dis[e.v] != inf)
                rt[u] = merge(rt[u], gn({ dis[e.v] + e.w - dis[u], e.v }));
        }
    }

    //  Merging heaps
    queue<int> q; q.push(t);
    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (int v : gt[u])
            rt[v] = merge(rt[v], rt[u]), q.push(v);
    }

    //  Iterate k times
    if (k == 1) return dis[s]; int cnt = 0;
    if (rt[s]) pq.push({ dis[s] + val[rt[s]].first, rt[s] });
    while (!pq.empty()) {
        pli p = pq.top(); pq.pop();
        int x = p.second, u = val[x].second;
        ll res = p.first; if (++cnt == k - 1) return res;
        if (rt[u]) pq.push({ res + val[rt[u]].first, rt[u] });
        if (ls[x]) pq.push({ res - val[x].first + val[ls[x]].first, ls[x] });
        if (rs[x]) pq.push({ res - val[x].first + val[rs[x]].first, rs[x] });
    }
    return -1;
}

}   //  namespace kth_shortest_path
```

快读板子在动态图里

exKMP

```
inline void GetExtendNext(char S[])
{
    int a = 0, Slen = strlen(S);
    exnxt[0] = Slen;
```

```
    while (a < Slen && S[a] == S[a + 1]) a++;
    exnxt[1] = a;
    a = 1;
    for (int i = 2; i < Slen; i++)
    {
        int p = a + exnxt[a] - 1, l = exnxt[i - a];
        if (i - 1 + l >= p)
        {
            int j = (p - i + 1) > 0 ? p - i + 1 : 0;
            while (i + j < Slen && S[i + j] == S[j]) j++;
            exnxt[i] = j;
            a = i;
        }
        else exnxt[i] = l;
    }
    return ;
}

inline void GetExtend(char S[], char T[])
{
    int a = 0;
    GetExtendNext(T);
    int Slen = strlen(S), Tlen = strlen(T);
    int Minlen = mymin(Slen, Tlen);
    while (a < Minlen && S[a] == T[a]) a++;
    ext[0] = a;
    a = 0;
    for (int i = 1; i < Slen; i++)
    {
        int p = a + ext[a] - 1, l = exnxt[i - a];
        if (i - 1 + l >= p)
        {
            int j = (p - i + 1) > 0 ? p - i + 1 : 0;
            while (i + j < Slen && j < Tlen && S[i + j] == T[j]) j++;
            ext[i] = j;
            a = i;
        }
        else ext[i] = l;
    }
    return ;
}
```

CDQ

```
#define MAXW 1<<12|1
#define MAXQ 1200005
using namespace std;

struct Mokia
{
    int x,y,o,d,p,pt;
    Mokia(){}
    Mokia(int _x,int _y,int _o,int _d,int _p,int _pt):
        x(_x),y(_y),o(_o),d(_d),p(_p),pt(_pt){}
    bool operator < (const Mokia &a)const{
        if (x==a.x&&y==a.y) return o<a.o;
        return x==a.x?y<a.y:x<a.x;
```

```
    }
};

Mokia q[MAXQ],t[MAXQ];
int n,m,T,o,x,y,u,v,k,Q;
long long c[MAXW],ans[MAXQ];

inline long long f(int x,long long v){return x>0?v:-v;}
inline int lowbit(int x){return x&-x;}
inline void modify(int x,long long v){for (;x<=m;x+=lowbit(x)) c[x]+=v;return ;}
inline long long query(int x){long long r=0;for (;x;x-=lowbit(x)) r+=c[x];return
r;}

void CDQ(int l,int r)
{
    if (l==r) return ;
    int mid=l+r>>1,ll=l,rl=mid+1;
    for (int i=l;i<=r;i++)
    {
        if (q[i].p<=mid&&!q[i].o) modify(q[i].y,q[i].d);
        else if (q[i].p>mid&&q[i].o) ans[q[i].pt]+=f(q[i].d,query(q[i].y));
    }
    for (int i=l;i<=r;i++)
        if (q[i].p<=mid&&!q[i].o) modify(q[i].y,-q[i].d);
    for (int i=l;i<=r;i++)
        if (q[i].p<=mid) t[ll++]=q[i];
        else t[rl++]=q[i];
    for (int i=l;i<=r;i++) q[i]=t[i];
    CDQ(l,mid);CDQ(mid+1,r);
    return ;
}

inline void addQuery(int x,int y)
{
    ++Q;++T;q[T]=Mokia(x,y,1,1,T,Q);
    return ;
}

inline void addModify(int x,int y,int u,int v,int k)
{
    ++u;++v;
    ++T;q[T]=Mokia(u,v,0,k,T,0);
    ++T;q[T]=Mokia(x,v,0,-k,T,0);
    ++T;q[T]=Mokia(u,y,0,-k,T,0);
    ++T;q[T]=Mokia(x,y,0,k,T,0);
    return ;
}

inline void addModify(int x,int y,int k)
{
    ++T;q[T]=Mokia(x,y,0,k,T,0);
    return ;
}

inline void addQuery(int x,int y,int u,int v)
{
    ++Q;
    ++T;q[T]=Mokia(u,v,1,1,T,Q);
```

```
    ++T;q[T]=Mokia(x-1,v,1,-1,T,Q);
    ++T;q[T]=Mokia(u,y-1,1,-1,T,Q);
    ++T;q[T]=Mokia(x-1,y-1,1,1,T,Q);
    return ;
}
```

2D Segment Tree

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

int ri() {
    int n;
    scanf("%d", &n);
    return n;
}

struct SegTree {
    size_t n;
    std::vector<int64_t> data;
    std::vector<int> xs;
    void reserve(int i) { xs.push_back(i); }
    void build0() {
        std::sort(xs.begin(), xs.end());
        for (n = 1; n < xs.size(); n <<= 1);
        data.resize(n << 1);
    }
    void build1() {
        for (int i = n; --i; ) data[i] = data[i << 1] + data[i << 1 | 1];
    }
    void add_pre(int i, int val) {
        i = std::lower_bound(xs.begin(), xs.end(), i) - xs.begin();
        data[i + n] += val;
    }
    void add(int i, int val) {
        i = std::lower_bound(xs.begin(), xs.end(), i) - xs.begin();
        for (i += n; i; i >>= 1) data[i] += val;
    }
    int64_t sum(int l, int r) {
        l = std::lower_bound(xs.begin(), xs.end(), l) - xs.begin();
        r = std::lower_bound(xs.begin(), xs.end(), r) - xs.begin();
        int64_t res = 0;
        for (l += n, r += n; l < r; l >>= 1, r >>= 1) {
            if (r & 1) res += data[--r];
            if (l & 1) res += data[l++];
        }
        return res;
    }
};
struct SegTree2D {
    size_t n;
    std::vector<SegTree> trees;
    SegTree2D (size_t n_) {
        for (n = 1; n < n_; n <<= 1);
        trees.resize(n << 1);
    }
    void reserve(int i, int j) {
        for (i += n; i; i >>= 1) trees[i].reserve(j);
```

```
    }
    void build0() {
        for (auto &i : trees) i.build0();
    }
    void build1() {
        for (auto &i : trees) i.build1();
    }
    void add_pre(int i, int j, int val) {
        for (i += n; i; i >>= 1) trees[i].add_pre(j, val);
    }
    void add(int i, int j, int val) {
        for (i += n; i; i >>= 1) trees[i].add(j, val);
    }
    int64_t sum(int l0, int r0, int l1, int r1) {
        int64_t res = 0;
        for (l0 += n, r0 += n; l0 < r0; l0 >>= 1, r0 >>= 1) {
            if (r0 & 1) res += trees[--r0].sum(l1, r1);
            if (l0 & 1) res += trees[l0++].sum(l1, r1);
        }
        return res;
    }
};

int main() {
    int n = ri();
    int q = ri();
    std::vector<int> xs;
    struct Point {
        int x;
        int y;
        int weight;
    };
    std::vector<Point> pts(n);
    for (auto &i : pts) {
        i.x = ri();
        i.y = ri();
        i.weight = ri();
        xs.push_back(i.x);
    }
    std::vector<int> a(q), b(q), c(q), d(q);
    for (int i = 0; i < q; i++) {
        a[i] = ri() - 1;
        if (!a[i]) a[i] = ri();
        b[i] = ri();
        c[i] = ri();
        d[i] = ri();
        if (a[i] == -1) xs.push_back(b[i]);
    }
    std::sort(xs.begin(), xs.end());
    SegTree2D tree(xs.size());
    for (auto &i : pts) {
        i.x = std::lower_bound(xs.begin(), xs.end(), i.x) - xs.begin();
        tree.reserve(i.x, i.y);
    }
    for (int i = 0; i < q; i++) {
        if (a[i] == -1) {
            b[i] = std::lower_bound(xs.begin(), xs.end(), b[i]) - xs.begin();
            tree.reserve(b[i], c[i]);
```

```
        } else {
            a[i] = std::lower_bound(xs.begin(), xs.end(), a[i]) - xs.begin();
            c[i] = std::lower_bound(xs.begin(), xs.end(), c[i]) - xs.begin();
        }
    }
    tree.build0();
    for (auto i : pts) tree.add_pre(i.x, i.y, i.weight);
    tree.build1();
    for (int i = 0; i < q; i++) {
        if (a[i] == -1) tree.add(b[i], c[i], d[i]);
        else printf("%" PRId64 "\n", tree.sum(a[i], c[i], b[i], d[i]));
    }
    return 0;
}
```

KDTree

```
#define MAXN 50010
#define MAXM 10
#define MAXK 5
#define INF ~(1<<31)
#define sqr(x) (x)*(x)
using namespace std;

int K, D;

struct point
{
    int a[MAXK];
    inline int Dis(const point &p)const
    {
        int dis = 0;
        for (int i = 0; i < K; i++) dis += sqr(a[i] - p.a[i]);
        return dis;
    }
};

bool cmp(point x, point y)
{
    return x.a[D] < y.a[D];
}
inline int mymin(int a, int b)
{
    return a < b ? a : b;
}
inline int mymax(int a, int b)
{
    return a > b ? a : b;
}

struct KDTree
{
    int ch[2], mx[MAXK], mn[MAXK];
    point P;
    inline int MinDis(const point &x)const
    {
        int dis = 0;
```

```
        for (int i = 0; i < K; i++)
            if (x.a[i] < mn[i] || x.a[i] > mx[i])
                dis += mymin(sqr(x.a[i] - mn[i]), sqr(mx[i] - x.a[i]));
        return dis;
    }
};

point p[MAXN], t, ans[MAXM];
KDTree tree[MAXN];
int n, T, m, top;
priority_queue <point> q;

bool operator <(point x, point y)
{
    return t.Dis(x) < t.Dis(y);
}

inline void Push_Up(int x, int y)
{
    for (int i = 0; i < K; i++)
    {
        tree[x].mx[i] = mymax(tree[x].mx[i], tree[y].mx[i]);
        tree[x].mn[i] = mymin(tree[x].mn[i], tree[y].mn[i]);
    }
    return ;
}

inline void Build_KDTree(int u, int l, int r, int d)
{
    D = d;
    int mid = l + r >> 1;
    nth_element(p + l, p + mid, p + r + 1, cmp);
    tree[u].P = p[mid];
    tree[u].ch[0] = tree[u].ch[1] = 0;
    for (int i = 0; i < K; i++) tree[u].mx[i] = tree[u].mn[i] = p[mid].a[i];
    if (l <= mid - 1)
    {
        tree[u].ch[0] = ++top;
        Build_KDTree(top, l, mid - 1, (d + 1) % K);
    }
    if (mid + 1 <= r)
    {
        tree[u].ch[1] = ++top;
        Build_KDTree(top, mid + 1, r, (d + 1) % K);
    }
    if (tree[u].ch[0]) Push_Up(u, tree[u].ch[0]);
    if (tree[u].ch[1]) Push_Up(u, tree[u].ch[1]);
    return ;
}

// The cloest M point
inline void query(int u)
{
    if (!u) return ;
    int d0 = t.Dis(tree[u].P), dis[2] = {INF, INF};
    if (q.size() < m) q.push(tree[u].P);
    else if (t.Dis(q.top()) > d0)
    {
```

```
        q.pop();
        q.push(tree[u].P);
    }
    if (tree[u].ch[0]) dis[0] = tree[tree[u].ch[0]].MinDis(t);
    if (tree[u].ch[1]) dis[1] = tree[tree[u].ch[1]].MinDis(t);
    int f = dis[0] > dis[1];
    if (q.size() < m || dis[f] < t.Dis(q.top())) query(tree[u].ch[f]);
    f ^= 1;
    if (q.size() < m || dis[f] < t.Dis(q.top())) query(tree[u].ch[f]);
    return ;
}

// Get the min
inline void query(int u, point P)
{
    long long q = tree[u].P.Dis(P);
    if (q)
    {
        if (q < res)
        {
            res = q;
            resP = tree[u].P;
        }
        else if (q == res && tree[u].P < resP)
        {
            resP = tree[u].P;
        }
    }
    long long ldis = tree[u].lch ? tree[tree[u].lch].MinDis(P) : ~(1LL<<63);
    long long rdis = tree[u].rch ? tree[tree[u].rch].MinDis(P) : ~(1LL<<63);
    if (ldis < rdis)
    {
        if (tree[u].lch && res >= ldis) query(tree[u].lch, P);
        if (tree[u].rch && res >= rdis) query(tree[u].rch, P);
    }
    else
    {
        if (tree[u].rch && res >= rdis) query(tree[u].rch, P);
        if (tree[u].lch && res >= ldis) query(tree[u].lch, P);
    }
    return ;
}
```

RMQLCA

```
namespace RMQLCA
{
    int n,u,v,rt,T,R,lg[MAXN<<1],pt[MAXN],dep[MAXN<<1];
    int fa[18][MAXN<<1],euler[MAXN<<1],dis[MAXN];

    inline void dfs(int x,int f,int deep,int v)
    {
        pt[x]=++R;euler[R]=x;dep[R]=deep;dis[x]=v;
        for (int i=head[x];~i;i=e[i].nxt)
            if (e[i].to!=f)
            {
                dfs(e[i].to,x,deep+1,e[i].val+v);
```

```
                euler[++R]=x;dep[R]=deep;
            }
        return ;
    }

    inline void RMQLCA()
    {
        dfs(1,-1,0,0);
        for (int i=1;i<=R;i++) fa[0][i]=i;
        for (int i=2;i<=R;i++) lg[i]=lg[i>>1]+1;
        for (int j=1;j<=lg[R];j++)
            for (int i=1;i+(1<<j)-1<=R;i++)
            {
                int a=fa[j-1][i],b=fa[j-1][i+(1<<(j-1))];
                fa[j][i]=dep[a]<=dep[b]?a:b;
            }
        return ;
    }

    inline int getLCA(int x,int y)
    {
        x=pt[x];y=pt[y];
        if (x>y) swap(x,y);
        int t=lg[y-x+1];int a=fa[t][x],b=fa[t][y-(1<<t)+1];
        return dep[a]<=dep[b]?euler[a]:euler[b];
    }

    inline int getDis(int x,int y)
    {
        return dis[x]+dis[y]-2*dis[getLCA(x,y)];
    }
}
```

HLD

```
inline void dfs1(int x)
{
    siz[x] = 1; mx[x] = -1;
    for (auto v : g[x])
        if (v != fa[x])
        {
            dep[v] = dep[x] + 1; fa[v] = x;
            dfs1(v);
            siz[x] += siz[v];
            if (!~mx[x] || siz[v] > siz[mx[x]]) mx[x] = v;
        }
    return ;
}

inline void dfs2(int x, int t)
{
    top[x] = t; rev[R] = x; in[x] = R++;
    if (~mx[x]) dfs2(mx[x], t);
    for (auto v : g[x])
        if (v != mx[x] && v != fa[x]) dfs2(v, v);
    out[x] = R;
    return ;
```

```
}

inline void modify_link(int l, int r, int v)
{
    while (top[l] != top[r])
    {
        if (dep[top[l]] < dep[top[r]]) swap(l, r);
        modify(1, in[top[l]], in[l] + 1, v, 1, n + 1);
        l = fa[top[l]];
    }
    if (in[l] > in[r]) swap(l, r);
    modify(1, in[l], in[r] + 1, v, 1, n + 1); // Weight on vertex
    // if (in[l] < in[r]) modify(1, in[l] + 1, in[r] + 1, v, 1, n + 1) // Weight
on edge
    return ;
}

inline int query_link(int l, int r)
{
    int res = 0;
    while (top[l] != top[r])
    {
        if (dep[top[l]] < dep[top[r]]) swap(l, r);
        res += query(1, in[top[l]], in[l] + 1, 1, n + 1);
        l = fa[top[l]];
    }
    if (in[l] > in[r]) swap(l, r);
    res += query(1, in[l], in[r] + 1, 1, n + 1);
    // if (in[l] < in[r]) res += query(1, in[l] + 1, in[r] + 1, 1, n + 1) //
Weight on edge
    return res;
}
```

TreeDC

```
inline void Calculate(int x,int fa,int pre)
{
    pre^=a[x];ans+=query(pre,r)-query(pre,l-1);
    for (auto v:g[x])
        if (!vis[v]&&v!=fa) Calculate(v,x,pre);
    return ;
}

inline void DFS(int x,int fa,int pre)
{
    pre^=a[x];insert(pre);
    for (auto v:g[x])
        if (!vis[v]&&v!=fa) DFS(v,x,pre);
    return ;
}

inline void DFS_size_and_G(int x,int fa)
{
    siz[x]=1;mx[x]=0;
    for (auto v:g[x])
        if (v!=fa&&!vis[v])
        {
```

```
            DFS_size_and_G(v,x);
            siz[x]+=siz[v];
            mx[x]=max(mx[x],siz[v]);
        }
    mx[x]=max(mx[x],SIZE-siz[x]);
    if (mx[x]<mx[G]) G=x;
    return ;
}

inline void TreeDC(int x)
{
    vis[x]=true;newrt();insert(a[x]);
    for (auto v:g[x])
        if (!vis[v])
        {
            Calculate(v,x,0);
            DFS(v,x,a[x]);
        }
    for (auto v:g[x])
        if (!vis[v])
        {
            SIZE=mx[0]=siz[v];G=0;
            DFS_size_and_G(v,x);
            TreeDC(G);
        }
    return ;
}

SIZE=mx[0]=n;G=0;DFS_size_and_G(1,-1);
TreeDC(G);
```