```
class ACauto
{
    private:
        int fail[MAXN] = {0}, trie[MAXN][26] = {0}, t = 0;
        int head[MAXN] = {0}, cnt = 0;
        struct Edge { int to,nxt; } e[MAXN];
        inline void addEdge(int,int);
    public:
        int ans[MAXN] = {0}, pos[MAXN] = {0};
        void ins(string&,int);
        void query(string&);
        void fresh();
        void getFail();
        void init();
        void dfs(int);
}a;
void ACauto::ins(string &s, int i)
{
    int p = 0;
    for(int i = 0; s[i]; ++i)
    {
        if(!trie[p][s[i]-'a']) trie[p][s[i]-'a'] = ++t;
        p = trie[p][s[i]-'a'];
    }
    pos[i] = p;
}
void ACauto::query(string &s)
{
    int p = 0;
    for(int i = 0; s[i]; ++i)
        p = trie[p][s[i]-'a'], ++ans[p];
}
void ACauto::getFail()
{
    int v, p;
    queue<int> q;
    for(int i = 0; i < 26; ++i)
        if(trie[0][i])
            q.push(trie[0][i]), fail[trie[0][i]] = 0;
    while(!q.empty())
    {
        p = q.front(); q.pop();
        for(int i = 0; i < 26; ++i)
        {
            v = trie[p][i];
            if(v) fail[v] = trie[fail[p]][i], q.push(v);
            else  trie[p][i] = trie[fail[p]][i];
        }
    }
}
inline void ACauto::addEdge(int u, int v)
{
    e[++cnt].to = v;
    e[cnt].nxt = head[u];
    head[u] = cnt;
}
```

```
void ACauto::init()
{
    for(int i = 1;i <= t; ++i)
        addEdge(fail[i], i);
}
void ACauto::dfs(int u)
{
    for(int i = head[u]; i; i = e[i].nxt)
    {
        dfs(e[i].to);
        ans[u] += ans[e[i].to];
    }
}
```

圆方树

```
struct BlockCutTree
{
    std::vector<std::vector<int>> adj;
    std::vector<int> dfn, low, st;
    int n, t = 0, R;

    BlockCutTree(const std::vector<vector<int>> &g) { init(g); }
    void init(const std::vector<std::vector<int>> &g)
    {
        R = n = g.size()-1; t = 0;
        adj.assign(n+1, std::vector<int>());
        dfn.assign(n+1,0); low.assign(n+1,0); st.clear();
        For(i,1,n) if(!dfn[i]) tarjan(i,g);
    }
    const std::vector<std::vector<int>>& getBCT() { return adj; }

    void tarjan(int u, const std::vector<std::vector<int>> &g)
    {
        dfn[u] = low[u] = ++t; st.push_back(u);
        for(auto v: g[u]) {
            if(!dfn[v]) {
                tarjan(v,g);
                chkmin(low[u],low[v]);
                if(low[v] == dfn[u])
                {
                    ++R; adj.push_back(std::vector<int>{u});
                    adj[u].push_back(R);
                    while(!st.empty())
                    {
                        int x = st.back(); st.pop_back();
                        adj[x].push_back(R); adj[R].push_back(x);
                        if(x == v) break;
                    }
                }
            } else chkmin(low[u],dfn[v]);
        }
    };
    void sort() { For(i, 1, R) std::sort(adj[i].begin(),adj[i].end()); }
};
```

珂朵莉树

```
struct node
{
    int l,r;
    mutable _ll val;
    node(int l = 0, int r = 0, _ll val = 0):l(l),r(r),val(val){}
    bool operator < (const node &rhs) const { return l < rhs.l; }
};
class Chtholly_Tree
{
    private:
        set<node> s;
        typedef set<node>::iterator It;
        It split(int pos)
        {
            It it = s.lower_bound(node(pos));
            if(it != s.end() && it->l == pos) return it;
            --it;
            int l = it->l, r = it->r; _ll v = it->val;
            s.erase(it); s.insert(node(l,pos-1,v));
            return s.insert(node(pos,r,v)).first;
        }
    public:
        void setn(int n, _ll a[]) { s.clear(); For(i,1,n) s.insert(node(i,i,a[i])); }
        void assign(int,int,_ll);
        void add(int,int,_ll);
        _ll kth(int,int,int);
        _ll query(int,int,_ll,_ll);
}s;
void Chtholly_Tree::assign(int l, int r, _ll val)
{
    It itr = split(r+1), itl = split(l);
    s.erase(itl,itr);
    s.insert(node(l,r,val));
}
void Chtholly_Tree::add(int l, int r, _ll val)
{
    It itr = split(r+1), itl = split(l);
    for(It it = itl; it != itr; ++it) it->val += val;
}
_ll Chtholly_Tree::kth(int l, int r, int k)
{
    vector<pll> v;
    It itr = split(r+1), itl = split(l);
    for(It it = itl; it != itr; ++it) v.push_back(make_pair(it->val,it->r-it->l+1));
    sort(v.begin(),v.end());
    for(auto i : v)
    {
        k -= i.second;
        if(k <= 0) return i.first;
    }
    return -1;
}
_ll Chtholly_Tree::query(int l, int r, _ll x, _ll p)
{
    _ll ans = 0;
    It itr = split(r+1), itl = split(l);
    for(It it = itl; it != itr; ++it) ans = (ans + (it->r-it->l+1)*qpow(it->val,x,p)) % p;
    return ans;
}
```

点分树

```
int n,m,u,v,w,rt,Size,T,top,s[MAXN],ans[MAXN];
int query[MAXN];
int sz[MAXN],d[MAXN],mx[MAXN],dis[MAXN];
bool vis[MAXN],jud[MAXK];
void dfs1(int u, int pre)
{
    sz[u] = 1; mx[u] = 0;
    ForE(i,u)
    {
        int v = e[i].to;
        if(vis[v] || v == pre) continue;
        dfs1(v,u);
        sz[u] += sz[v];
        mx[u] = max(mx[u],sz[v]);
    }
    mx[u] = max(mx[u],Size-sz[u]);
    if(mx[u] < mx[rt]) rt = u;
}
void dfs2(int u, int pre)
{
    if(dis[u] >= MAXK) return;
    d[++T] = dis[u];
    ForE(i,u)
    {
        int v = e[i].to;
        if(vis[v] || v == pre) continue;
        dis[v] = dis[u] + e[i].w;
        dfs2(v,u);
    }
}
void solve(int u)
{
    ForE(i,u)
    {
        int v = e[i].to;
        if(vis[v]) continue;
        T = 0; dis[v] = e[i].w;
        dfs2(v,u);
        For(j,1,T) For(k,1,m)
            if(query[k] >= d[j])
                ans[k] |= jud[query[k]-d[j]];
        For(j,1,T)
            jud[s[++top]=d[j]] = 1;
    }
    while(top) jud[s[top--]] = 0;
}
void dsu(int u)
{
    vis[u] = jud[0] = 1;
```

```
    solve(u);
    ForE(i,u)
    {
        int v = e[i].to;
        if(vis[v]) continue;
        Size = sz[v]; mx[rt=0] = n;
        dfs1(v,u);
        dsu(rt);
    }
}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    cin >> n >> m; Size = n;
    For(i,2,n) cin >> u >> v >> w, addEdge(u,v,w), addEdge(v,u,w);
    For(i,1,m) cin >> query[i];
    mx[rt=0] = n;
    dfs1(1,-1);
    dsu(rt);
    For(i,1,m) cout << (ans[i] ? "AYE" : "NAY") << '\n';
    return 0;
}
```

欧拉路

```
void dfs(int u)
{
    for(int i = over[u]; i < e[u].size(); i = over[u])
    {
        over[u] = i+1;
        dfs(e[u][i]);
    }
    st.push(u);
}
int main()
{
    cin >> n >> m;
    For(i,1,m) cin >> x >> y, e[x].push_back(y), ++in[y], ++out[x];
    For(i,1,n)
    {
        sort(e[i].begin(),e[i].end());
        if(in[i]+1 == out[i]) ++ss, s = i;
        else if(in[i] == out[i]+1) ++ts;
        else if(in[i] != out[i]) { cout << "No"; return 0; }
    }
    if(ss != ts) { cout << "No"; return 0; }
    if(!ss && !ts) s = 1;
    dfs(s);
    while(!st.empty()) cout << st.top() << " ", st.pop();
    fclose(stdin);
    return 0;
}
```

DSU

```
struct Dsu
{
    std::vector<int> pre, sz;
    Dsu(int n = 0) { init(n); }
    void init(int n) { pre.resize(n+1); sz.assign(n+1,1); iota(pre.begin(),pre.end(),0); }
    int find(int x)
    {
        while(x != pre[x]) x = pre[x] = pre[pre[x]];
        return x;
    }
    int same(int x, int y) { return find(x) == find(y); }
    bool merge(int x, int y)
    {
        x = find(x); y = find(y);
        if(x == y) return false;
        if(sz[x] < sz[y]) std::swap(x, y);
        pre[y] = x; sz[x] += sz[y];
        return true;
    }
    int size(int x) { return sz[find(x)]; }
};
```

Excrt

```
i64 exgcd(i64 a, i64 b, i64 &x, i64 &y)
{
    x = 1; y = 0;
    i64 x1 = 0, y1 = 1, a1 = a, b1 = b;
    while(b1)
    {
        i64 q = a1 / b1;
        std::tie(x, x1) = std::pair(x1, x - q * x1);
        std::tie(y, y1) = std::pair(y1, y - q * y1);
        std::tie(a1, b1) = std::pair(b1, a1 - q * b1);
    }
    return a1;
}

i64 exgcd(i64 &x, i64 &y, i64 a, i64 b)
{
    if(!b)
    {
        x = 1; y = 0;
        return a;
    }
    i64 g = exgcd(x,y,b,a%b);
    i64 t = x; x=y;
    y = t - a / b * y;
    return g;
}

i64 excrt()
{
    M = bb[1]; ans = aa[1];
    For(i,2,n)
    {
        x = y = 0; a = aa[i]; b = bb[i];
        c = ((a - ans) % b + b) % b;
```

```
        g = exgcd(M,b,x,y);
        // cout << g << ' ' << M << ' ' << b << ' ' << x << ' ' << y << endl;
        p = b / g;
        x = mul(x, c/g, p);
        ans += x * M;
        M *= p;
        ans = (ans % M + M) % M;
    }
}
```

exbsgs

```
_ll exgcd(_ll a, _ll b, _ll &x, _ll &y)
{
    x = 1; y = 0;
    _ll q, x1 = 0, y1 = 1, a1 = a, b1 = b, x2,y2,b2;
    while(b1)
    {
        q = a1 / b1;
        x2 = x1;     x1 = x - q * x1;    x = x2;
        y2 = y1;     y1 = y - q * y1;    y = y2;
        b2 = b1;     b1 = a1 - q * b1;   a1 = b2;
    }
    return a1;
}
_ll inv(_ll a, _ll p)
{
    _ll x = 0, y = 0;
    exgcd(a,p,x,y);
    return (x%p+p)%p;
}



pii p[MAXN];
int bsgs(int a, int b, int P)
{
    int k = ceil(sqrt(P));
    p[0] = pii(b,0);
    For(i,1,k) p[i] = pii(1ll*p[i-1].first*a%P,i);
    sort(p, p+1+k, [](const pii &a, const pii &b) { return a.first==b.first? a.second >
b.second: a.first < b.first; });
    p[k+1] = pii(-1,0);
    int tmp = qpow(a,k,P);
    int x = 1;
    For(i,1,k)
    {
        x = 1ll*x*tmp%P;
        auto it = lower_bound(p,p+1+k,pii(x,0));
        if(x == it->first) return i*k-it->second;
    }
    return INF;
}
int exbsgs(int a, int b, int P)
{
    int g; int tmp = 1, k = 0;
    if(b == 1 || P == 1) return 0;
    while((g = __gcd(a,P)) > 1)
    {
        if(b%g) return INF;
        P /= g; b /= g;
        tmp = 1ll*tmp*(a/g)%P; ++k;
        if(tmp == b%P) return k;
    }
    int ans = bsgs(a,1ll*b*inv(tmp,P)%P,P);
    if(ans < INF) return ans + k;
    else return INF;
}
```

Fenwick

```
template <typename T>
struct Fenwick
{
    int sz, lg;
    std::vector<T> t;
    Fenwick(int n = 0, T tt = T()) { init(std::vector<T>(n+1,tt)); }
    Fenwick(const std::vector<T> &a) { init(a); }
    void init(int n = 0, T tt = T()) { init(std::vector<T>(n+1,tt)); }
    void init(const std::vector<T> &a)
    {
        sz = a.size()-1; lg = log2(sz); t.assign(sz+1,T());
        For(i,1,sz)
        {
            t[i] += a[i];

            int j = i + lowbit(i);
            if (j <= sz) t[j] += t[i];
        }
    }
    int lowbit(int i) { return i & -i; }
    void add(int i, T k) { while(i <= sz) t[i] += k, i += lowbit(i); }
    void add(int l, int r, T k) { add(l,k); add(r+1,-k); }
    T query(int i) { T res = 0; while(i) res += t[i], i -= lowbit(i); return res; }
    T query(int l, int r) { if(r < l) std::swap(l,r); return query(r)-query(l-1); }
    int lower_bound(T k)
    {
        int res = 0;
        foR(i,lg,0)
            if(res + (1<<i) <= sz && t[res+(1<<i)] < k)
                k -= t[res+=(1<<i)];
        return res+1;
    }
    int upper_bound(T k)
    {
        int res = 0;
        foR(i,lg,0)
            if(res + (1<<i) <= sz && t[res+(1<<i)] <= k)
                k -= t[res+=(1<<i)];
        return res+1;
    }
};
```

FHQ_Treap

```cpp
std::random_device rd;
std::mt19937 gen(rd());
std::uniform_int_distribution<> distrib(0, 998244352);
template<class Info, class Tag>
struct Node
{
    array<Node*, 2> ch{};
    Info info = Info();
    Tag tag = Tag();
    int rk = distrib(gen);
    Node(const Info &I = Info()): info(I) { }
};
template<class Info, class Tag>
struct Treap
{
    using pNode = Node<Info, Tag>*;
    pNode rt = nullptr;

    Treap() { }
    ~Treap()
    {
        auto dfs = [&](auto &&self, pNode u) -> void
        {
            if(!u) return;
            if(u->ch[0]) self(self, u->ch[0]);
            if(u->ch[1]) self(self, u->ch[1]);
            delete u;
            u = nullptr;
        };
        dfs(dfs, rt);
    }

    void apply(pNode u, const Tag &t) { u->info.apply(t); u->tag.apply(t); }
    void pushup(pNode u)
    {
        u->info.fresh();
        if(u->ch[0]) u->info = u->info + u->ch[0]->info;
        if(u->ch[1]) u->info = u->info + u->ch[1]->info;
    }
    void pushdown(pNode u)
    {
        if(u->ch[0]) apply(u->ch[0], u->tag);
        if(u->ch[1]) apply(u->ch[1], u->tag);
        if(u->tag.rt) swap(u->ch[0], u->ch[1]);
        u->tag = Tag();
    }

    void split(pNode u, int k, pNode &x, pNode &y, pNode &z)
    {
        if(u == nullptr) { x = y = z = nullptr; return; }
        pushdown(u);
        int lsz = (u->ch[0]? u->ch[0]->info.size: 0);
        if(k <= lsz) z = u, split(u->ch[0], k, x, y, u->ch[0]);
        else if(k > lsz + u->info.cnt) x = u, split(u->ch[1], k-lsz-u->info.cnt, u->ch[1],
y, z);
        else
        {
            x = u->ch[0]; y = u; z = u->ch[1];
            u->ch.fill(nullptr);
        }
        pushup(u);
    }
    pNode merge(pNode x, pNode y)
    {
        if(x == nullptr) return y;
        else if(y == nullptr) return x;
        if(x->rk < y->rk)
        {
            pushdown(x);
            x->ch[1] = merge(x->ch[1], y);
            pushup(x);
            return x;
        }
        pushdown(y);
        y->ch[0] = merge(x, y->ch[0]);
        pushup(y);
        return y;
    }

    int rank(Info val)
    {
        pNode u = rt;
        int rk = 0;
        while(u)
        {
            if(val.key < u->info.key) u = u->ch[0];
            else
            {
                if(u->ch[0]) rk += u->ch[0]->info.size;
                if(val.key == u->info.key) break;
                rk += u->info.cnt;
                u = u->ch[1];
            }
        }
        return rk + 1;
    }

    void insert(Info val)
    {
        pNode x(nullptr), y(nullptr), z(nullptr);
        int rk = rank(val);
        split(rt, rk, x, y, z);
        if(y == nullptr) y = new Node<Info, Tag>(Info(val.key, 1, 1));
        else if(y->info.key == val.key)
        {
            ++y->info.cnt;
            pushup(y);
        }
        else y->ch[y->info.key < val.key] = new Node<Info, Tag>(Info(val.key, 1, 1)),
pushup(y);
        rt = merge(merge(x, y), z);
```

```
    }
    void erase(Info val)
    {
        int rk = rank(val);
        pNode x(nullptr), y(nullptr), z(nullptr);
        split(rt, rk, x, y, z);
        if(y && y->info.key == val.key) delete y, y = nullptr;
        rt = merge(merge(x, y), z);
    }
    void extract(Info val)
    {
        int rk = rank(val);
        pNode x(nullptr), y(nullptr), z(nullptr);
        split(rt, rk, x, y, z);
        if(y == nullptr || y->info.key != val.key);
        else if(y->info.cnt == 1) delete y, y = nullptr;
        else --y->info.cnt, pushup(y);
        rt = merge(merge(x, y), z);
    }
    Info kth(int k, pNode _rt)
    {
        pNode u = _rt;
        while(u)
        {
            if(u->ch[0] && u->ch[0]->info.size >= k) u = u->ch[0];
            else if(u->info.cnt + (u->ch[0]? u->ch[0]->info.size: 0) >= k) return u->info;
            else k -= u->info.cnt + (u->ch[0]? u->ch[0]->info.size: 0), u = u->ch[1];
        }
        return {-1};
    }
    Info kth(int k) { return kth(k, rt); }
    Info pre(Info val)
    {
        int rk = rank(val);
        pNode x(nullptr), y(nullptr), z(nullptr);
        split(rt,rk-1,x,y,z);
        Info ans = y->info;
        rt = merge(merge(x, y), z);
        return ans;
    }
    Info nxt(Info val)
    {
        ++val.key;
        int rk = rank(val);
        pNode x(nullptr), y(nullptr), z(nullptr);
        split(rt,rk,x,y,z);
        Info ans = y->info;
        rt = merge(merge(x, y), z);
        return ans;
    }
};
struct Tag
{
    bool rt = false;
    void apply(const Tag &t)
    {
        rt ^= t.rt;
    }
};
struct Info
{
    int key = 0;
    int cnt = 0, size = 0;
    void fresh() { this->size = cnt; }
    void apply(const Tag &t) { }
    Info operator+(Info o)
    {
        Info res = *this;
        res.size += o.size;
        return res;
    }
};
```

FWT

```
void AND(_ll *f, _ll x = 1)
{
    for(int mid = 2; mid <= n; mid <<= 1)
        for(int j = 0; j < n; j += mid)
            for(int k = j; k < j + (mid>>1); ++k)
                f[k] = (f[k] + f[k+(mid>>1)]*x + P) % P;
}
void IOR(_ll *f, _ll x = 1)
{
    for(int mid = 2; mid <= n; mid <<= 1)
        for(int j = 0; j < n; j += mid)
            for(int k = j; k < j + (mid>>1); ++k)
                f[k+(mid>>1)] = (f[k+(mid>>1)] + f[k]*x + P) % P;
}
void XOR(_ll *f, _ll x = 1)
{
    for(int mid = 2; mid <= n; mid <<= 1)
        for(int j = 0; j < n; j += mid)
            for(int k = j; k < j + (mid>>1); ++k)
            {
                f[k] = (f[k] + f[k+(mid>>1)] + P) % P;
                f[k+(mid>>1)] = (f[k] - f[k+(mid>>1)]*2%P + P) % P;
                f[k] = f[k] * x % P; f[k+(mid>>1)] = f[k+(mid>>1)] * x % P;
            }
}
```

SA

```cpp
std::pair<int, std::vector<f64>> Gauss(std::vector<std::vector<f64>> a)
{
    constexpr f64 eps = 1e-6;
    const int n = a.size() - 1, m = a[1].size() - 1;
    int rows = 1;
    f64 coef = 1;
    For(j, 1, n)
    {
        int mx = rows;
        For(i, rows+1, n)
            if(std::abs(a[i][j]) > std::abs(a[mx][j]))
                mx = i;
        if(std::abs(a[mx][j]) < eps) continue;
        if(mx != rows)
        {
            coef *= -1;
            std::swap(a[rows], a[mx]);
        }
        For(i, 1, n)
        {
            if(i == rows) continue;
            f64 tmp = a[i][j] / a[rows][j];
            For(k, j + 1, m) a[i][k] -= tmp * a[rows][k];
        }
        ++rows;
    }
    if(rows <= n)
    {
        For(i, rows, n)
            if(std::abs(a[i][m]) > eps)
                return {-1, {}};

        return {0, {}};
    }
    std::vector<f64> ans(n+1);
    For(i, 1, n) ans[i] = a[i][m] / a[i][i];
    return {1, ans};
}
```

筛

```cpp
std::vector<int> ps, phi, mu;
void getPrime(int n = 1e6)
{
    phi.resize(n+1); mu.resize(n+1);
    std::vector<bool> vis(n+1, false);
    vis[1] = true; phi[1] = 1; mu[1] = 1;
    for(int i = 2; i <= n; ++i)
    {
        if(!vis[i]) ps.push_back(i), phi[i] = i-1, mu[i] = -1;
        for(auto p: ps)
        {
            if(1ll * i * p > n) break;
            vis[i*p] = true;
            if(!(i%p))
            {
                mu[i*p] = 0;
                phi[i*p] = phi[i] * p;
                break;
            }
            phi[i*p] = phi[i] * phi[p];
            mu[i*p] = -mu[i];
        }
    }
}
```

HLD

```cpp
struct HLD
{
    int n, t = 0;
    std::vector<int> sz, top, dep, pre, dfn, out, seq;
    std::vector<std::vector<int>> adj;
    HLD(int n = 0) { init(n); }
    void init(int _n = 0)
    {
        n = _n; t = 0;
        sz.assign(n+1, 1); seq.assign(n+1, 0);
        dep = pre = top = dfn = out = seq;
        adj.assign(n+1, std::vector<int>());
    }
    void addEdge(int u, int v) { adj[u].push_back(v); adj[v].push_back(u); }

    void work(int rt = 1)
    {
        pre[rt] = 0; dep[rt] = 1; top[rt] = rt;
        dfs1(rt); dfs2(rt);
    }
    void dfs1(int u)
    {
        if(pre[u]) adj[u].erase(find(adj[u].begin(),adj[u].end(),pre[u]));
        sz[u] = 1;
        for(auto &v: adj[u])
        {
            pre[v] = u; dep[v] = dep[u] + 1;
            dfs1(v); sz[u] += sz[v];
            if(sz[v] > sz[adj[u][0]]) std::swap(v,adj[u][0]);
        }
    }
    void dfs2(int u)
    {
        dfn[u] = ++t; seq[t] = u;
        for(auto v: adj[u])
        {
            top[v] = (v == adj[u][0] ? top[u] : v);
            dfs2(v);
        }
        out[u] = t;
    }

    int lca(int u, int v)
    {
```

```
        while(top[u] != top[v])
        {
            if(dep[top[u]] < dep[top[v]]) v = pre[top[v]];
            else u = pre[top[u]];
        }
        return dep[u] < dep[v] ? u : v;
    }
    int dis(int u, int v) { return dep[u] + dep[v] - 2 * dep[lca(u,v)]; }
    int jump(int u, int k)
    {
        if(dep[u] < k) return 0;
        while(dep[u] - dep[top[u]] < k) k -= dep[u] - dep[top[u]] + 1, u = pre[top[u]];
        return seq[dfn[u]-k];
    }
    std::vector<pii> decompose(int u, int v)
    {
        std::vector<pii> res;
        while(top[u] != top[v])
        {
            if(dep[top[u]] < dep[top[v]]) std::swap(u,v);
            res.emplace_back(dfn[top[u]],dfn[u]);
            u = pre[top[u]];
        }
        if(dep[u] > dep[v]) std::swap(u,v);
        res.emplace_back(dfn[u],dfn[v]);
        return res;
    }
    pii subtree(int u) { return {dfn[u],out[u]}; }

    bool isAncestor(int u, int v) { return dfn[u] <= dfn[v] && out[u] >= out[v]; }
    int rootedParent(int u, int v)
    {
        if(u == v) return u;
        std::swap(u,v);
        if(!isAncestor(u,v)) return pre[u];
        return *upper_bound(adj[u].begin(),adj[u].end(),v,[this](int x, int y) { return
dfn[x] < dfn[y]; }) - 1;
    }
    int rootedSize(int u, int v)
    {
        if(u == v) return n;
        if(!isAncestor(v,u)) return sz[v];
        return n - sz[rootedParent(v,u)];
    }
    int rootedLCA(int a, int b, int c) { return lca(a,b)^lca(b,c)^lca(c,a); }
};
```

KM

```
_ll n,m,x,y,res,d,v[MAXN],mat[MAXN],pre[MAXN],ll[MAXN],lr[MAXN],s[MAXN],e[MAXN][MAXN];

void bfs(_ll u)
{
    _ll y1=0; y=0; memset(pre,0,sizeof(pre));
    for(_ll i = 1; i <= n; ++i) s[i] = INF;
    mat[y] = u;
    while(1)
    {
        x = mat[y]; d = INF; v[y] = 1;
        for(_ll i = 1; i <= n; ++i)
        {
            if(v[i]) continue;
            if(s[i] > ll[x] + lr[i] - e[x][i])
                s[i] = ll[x] + lr[i] - e[x][i], pre[i] = y;
            if(s[i] < d) d = s[i], y1 = i;
        }
        for(_ll i = 0; i <= n; ++i)
        {
            if(v[i]) ll[mat[i]] -= d, lr[i] += d;
            else     s[i] -= d;
        }
        y = y1;
        if(mat[y] == -1) break;
    }
    while(y) mat[y] = mat[pre[y]], y = pre[y];
}
_ll KM()
{
    _ll res = 0ll;
    memset(mat,-1,sizeof(mat)); memset(ll,0,sizeof(ll)); memset(lr,0,sizeof(lr));
    for(_ll i = 1; i <= n; ++i)
    {   memset(v,0,sizeof(v)); bfs(i);    }
    for(_ll i = 1; i <= n; ++i)
        if(mat[i]!=-1)
            res += e[mat[i]][i];
    return res;
}

int main()
{
    cin >> n >> m;
    for(int i = 1; i <= n; ++i)
        for(int j = 1; j <= n; ++j)
            e[i][j] = -INF;
    for(int i = 1; i <= m; ++i)
    {   cin >> x >> y; cin >> e[x][y];  }
    cout << KM() << endl;
    for(int i = 1; i <= n; ++i)
        printf("%lld ",mat[i]);
    return 0;
}
```

KMP

```
template<typename T>
std::vector<int> getKmp(const std::vector<T> &a)
{
    std::vector<int> kmp(a.size());
    int n = kmp.size() - 1;
    for(int i = 2, j = 0; i <= n; ++i)
    {
        while(j && a[j + 1] != a[i]) j = kmp[j];
        if(a[j + 1] == a[i]) ++j;
        kmp[i] = j;
```

```
    }
    return kmp;
}
std::vector<int> getKmp(const std::string &a) { return getKmp(std::vector<char>(a.begin(),
a.end())); }
```

Manacher

```
std::vector<int> manacher(const std::string &s)
{
    std::string t = "#";
    for(auto ch: s) t.push_back(ch), t.push_back('#');
    int n = t.length();
    std::vector<int> r(n);
    for(int i = 0, j = 0; i < n; ++i)
    {
        if(2 * j >= i && j + r[j] > i) r[i] = std::min(r[2 * j - i], j + r[j] - i);
        while(i - r[i] >= 0 && i + r[i] < n && t[i-r[i]] == t[i+r[i]]) ++r[i];
        if(i + r[i] > j + r[j]) j = i;
    }
    return r;
}
```

Z

```
template<typename T>
std::vector<int> getZ(const std::vector<T> &a)
{
    int n = a.size() - 1;
    std::vector<int> z(n + 1); z[1] = n;
    for(int i = 2, l = 0, r = 0; i <= n; ++i)
    {
        if(i <= r) z[i] = std::min(z[i-l+1], r-i+1);
        while(i + z[i] <= n && a[i+z[i]] == a[z[i]+1]) ++z[i];
        if(i + z[i] - 1 > r) l = i, r = z[i] + i - 1;
    }
    return z;
}
std::vector<int> getZ(const std::string &a) { return getZ(std::vector<char>(a.begin(),
a.end())); }
```

LazySegmentTree

```
template<class Info, class Tag>
struct SegmentTree
{
    #define ls(i) i<<1
    #define rs(i) i<<1^1

    int n, s = 1;
    std::vector<Info> info; std::vector<Tag> tag;

    SegmentTree(int _n = 0, Info t = Info()) { init(std::vector<Info>(_n+1,t)); }
    SegmentTree(const std::vector<Info> &v) { init(v); }
    void sets(int _s) { s = _s; }
    void init(int _n = 0, Info t = Info()) { init(std::vector<Info>(_n+1,t)); }
    void init(const std::vector<Info> &v)
    {
        n = v.size()-1;
        info.assign(n<<2^1,Info()); tag.assign(n<<2^1,Tag());
        std::function<void(int,int,int)> build = [&](int i, int l, int r)
        {
            if(l >= r) { info[i] = v[l]; return; }
            int mid = (l+r)>>1;
            build(ls(i),l,mid); build(rs(i),mid+1,r);
            pushup(i);
        };
        build(1,s,n);
    }
    void pushup(int i) { info[i] = info[ls(i)] + info[rs(i)]; }
    void pushdown(int i) { apply(ls(i),tag[i]); apply(rs(i),tag[i]); tag[i] = Tag(); }
    void apply(int i, const Tag &t) { info[i].apply(t); tag[i].apply(t); }
    void modify(int i, int l, int r, int x, const Info &I)
    {
        if(l >= r) { info[i] = I; return; }
        int mid = (l+r)>>1; pushdown(i);
        if(x <= mid) modify(ls(i),l,mid,x,I);
        else modify(rs(i),mid+1,r,x,I);
        pushup(i);
    }
    void apply(int i, int l, int r, int x, int y, const Tag &t)
    {
        if(y < l || r < x) return;
        if(x <= l && r <= y) { apply(i,t); return; }
        int mid = (l+r)>>1; pushdown(i);
        apply(ls(i),l,mid,x,y,t); apply(rs(i),mid+1,r,x,y,t);
        pushup(i);
    }
    Info query(int i, int l, int r, int x, int y)
    {
        if(y < l || r < x) return Info();
        if(x <= l && r <= y) return info[i];
        int mid = (l+r)>>1; pushdown(i);
        return query(ls(i),l,mid,x,y) + query(rs(i),mid+1,r,x,y);
    }
    template<class F>
    int findFirst(int i, int l, int r, int x, int y, F k)
    {
        if(y < l || r < x || !k(info[i])) return -1;
        if(l == r) return l;
        int mid = (l+r)>>1; pushdown(i);
        int res = findFirst(ls(i),l,mid,x,y,k);
        if(!~res) res = findFirst(rs(i),mid+1,r,x,y,k);
        return res;
    }
    template<class F>
    int findLast(int i, int l, int r, int x, int y, F k)
    {
        if(y < l || r < x || !k(info[i])) return -1;
        if(l == r) return l;
        int mid = (l+r)>>1; pushdown(i);
        int res = findLast(rs(i),mid+1,r,x,y,k);
        if(!~res) res = findLast(ls(i),l,mid,x,y,k);
```

```
        return res;
    }
    void modify(int x, const Info &I) { modify(1,s,n,x,I); }
    void apply(int l, int r, const Tag &t) { apply(1,s,n,l,r,t); }
    Info query(int l, int r) { return query(1,s,n,l,r); }
    template<class F> int findFirst(int l, int r, F k) { return findFirst(1,s,n,l,r,k); }
    template<class F> int findLast(int l, int r, F k) { return findLast(1,s,n,l,r,k); }

    #undef ls
    #undef rs
};

struct Tag
{
    i64 p,m;
    Tag(i64 _p = 0, i64 _m = 1): p(_p), m(_m) {}
    void apply(const Tag &t)
    {
        if(t.m != 1) p = t.m*p%P, m = t.m*m%P;
        if(t.p) (p += t.p) %= P;
    }
};
struct Info
{
    i64 sum; int len;
    Info(i64 _s = 0, int _l = 0): sum(_s), len(_l) {}
    void apply(const Tag &t) { sum = (sum*t.m + t.p*len)%P; }
    Info operator+(const Info &o) const { return Info((sum+o.sum)%P, len+o.len); }
};
```

LC_TREE

```
const double EPI = 1e-8;

int cmp(double x, double y)
{
    if(fabs(x-y) < EPI) return 0;
    return x < y ? -1 : 1;
}

int cnt,n,op,x1,Y1,x2,y2,last;

struct line
{
    double k,b;
    line(double k = 0, double b = 0):k(k),b(b){}
    double calc(double x) const { return k*x+b; }
}a[MAXN];

int segmax(int u, int v, int x)
{
    int flag = cmp(a[u].calc(x),a[v].calc(x));
    if(flag == 1 || (!flag && u < v)) return u;
    return v;
}
```

```
class LC_Seg_Tree
{
    private:
        int pos[MAXN<<2];
        void update(int,int,int,int);
    public:
        void modify(int,int,int,int,int,int);
        int query(int,int,int,int);
}s;
void LC_Seg_Tree::update(int i, int l, int r, int v)
{
    int &u = pos[i], mid = (l+r)>>1;
    int flag_mid = cmp(a[v].calc(mid),a[u].calc(mid));
    if(flag_mid == 1 || (flag_mid == 0 && v < u)) swap(u,v);
    int flag_l = cmp(a[v].calc(l),a[u].calc(l)), flag_r = cmp(a[v].calc(r),a[u].calc(r));
    if(flag_l == 1 || (flag_l == 0 && v < u)) update(i<<1,l,mid,v);
    if(flag_r == 1 || (flag_r == 0 && v < u)) update(i<<1^1,mid+1,r,v);
}
void LC_Seg_Tree::modify(int i, int l, int r, int x, int y, int v)
{
    if(x <= l && r <= y) { update(i,l,r,v); return; }
    int mid = (l+r)>>1;
    if(x <= mid) modify(i<<1,l,mid,x,y,v);
    if(y >  mid) modify(i<<1^1,mid+1,r,x,y,v);
}
int LC_Seg_Tree::query(int i, int l, int r, int x)
{
    if(x < l || r < x) return 0;
    if(l == r) return pos[i];
    int mid = (l+r)>>1, res = pos[i];
    int resl = query(i<<1,l,mid,x), resr = query(i<<1^1,mid+1,r,x);
    return segmax(segmax(res,resl,x),resr,x);
}

line make_line(int x1, int Y1, int x2, int y2)
{
    if(x1 == x2) return line(0,max(Y1,y2));
    double k = 1.0*(Y1-y2)/(x1-x2), b = Y1-k*x1;
    return line(k,b);
}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    cin >> n;
    while(n--)
    {
        cin >> op;
        if(op)
        {
            cin >> x1 >> Y1 >> x2 >> y2;
            x1 = (x1+last-1)%39989+1; Y1 = (Y1+last-1)%1000000000+1;
            x2 = (x2+last-1)%39989+1; y2 = (y2+last-1)%1000000000+1;
            if(x1 > x2) swap(x1,x2), swap(Y1,y2);
            a[++cnt]=make_line(x1,Y1,x2,y2);
```

```
            s.modify(1,1,39989,x1,x2,cnt);
        }
        else
        {
            cin >> x1;
            x1 = (x1+last-1)%39989+1;
            cout << (last = s.query(1,1,39989,x1)) << endl;
        }
    }
    return 0;
}
```

O1 LCA

```
void dfs(int u, int pre)
{
    dfn[u] = ++T; st[0][T] = pre;
    ForE(i,u)
        if (e[i].to != pre)
            dfs(e[i].to,u);
}

inline int check(int u, int v)
{
    return dfn[u] < dfn[v] ? u : v;
}
void init()
{
    For(i,1,22)
        for (int j = 1; j + (1<<i) - 1 <= T; ++j)
            st[i][j] = check(st[i-1][j],st[i-1][j+(1<<(i-1))]);
}
int query(int l, int r)
{
    int k = log2(r-l);
    return check(st[k][l+1],st[k][r-(1<<k)+1]);
}
```

LCT

```
template<class Info, class Tag>
struct Node
{
    int pre = 0;
    array<int, 2> ch{};
    Info info = Info();
    Tag tag = Tag();
};
template<class Info, class Tag>
struct LinkCutTree
{
    std::vector<Node<Info,Tag>> tr;

    LinkCutTree(int n = 0, const Info I = Info()) { init(n, I); }
    void init(int n, const Info I = Info())
    {
        tr.assign(n+1, {0,0,0,I});
        tr[0].info = Info();
    }

    bool get(int u) { return u == tr[tr[u].pre].ch[1]; }
    void pushup(int u) { tr[u].info.fresh(); tr[u].info = tr[u].info +
tr[tr[u].ch[0]].info + tr[tr[u].ch[1]].info; }
    void apply(int u, const Tag &t) { tr[u].info.apply(t); tr[u].tag.apply(t); }
    void pushdown(int u)
    {
        if(tr[u].ch[0]) apply(tr[u].ch[0], tr[u].tag);
        if(tr[u].ch[1]) apply(tr[u].ch[1], tr[u].tag);
        if(tr[u].tag.rt)
            std::swap(tr[u].ch[0], tr[u].ch[1]);
        tr[u].tag = Tag();
    }
    void rotModify(int x, int y, int chk)
    {
        if(x) tr[x].pre = y;
        if(y) tr[y].ch[chk] = x;
    }
    void rotate(int x)
    {
        int y = tr[x].pre, z = tr[y].pre;
        int chk = get(x);
        if(!isroot(y)) rotModify(x,z,get(y));
        tr[x].pre = z;
        rotModify(tr[x].ch[chk^1],y,chk);
        rotModify(y,x,chk^1);
        pushup(y); pushup(x);
    }
    void splay(int x)
    {
        update(x);
        for(int y = tr[x].pre; y = tr[x].pre, !isroot(x); rotate(x))
            if(!isroot(y))
                rotate(get(y)==get(x)?y:x);
    }

    bool isroot(int x) { return tr[tr[x].pre].ch[0] != x && tr[tr[x].pre].ch[1] != x; }
    void update(int u)
    {
        if(!isroot(u))
            update(tr[u].pre);
        pushdown(u);
    }
    int access(int x)
    {
        int y;
        for(y = 0; x; y = x, x = tr[x].pre)
            splay(x), tr[x].ch[1] = y, pushup(x);
        return y;
    }
    void makeroot(int u) { access(u); splay(u); tr[u].tag.rt ^= 1; }
    void split(int x, int y) { makeroot(x); access(y); splay(y); }
    int find(int x)
```

```
    {
        access(x); splay(x);
        pushdown(x);
        while(tr[x].ch[0])
            x = tr[x].ch[0], pushdown(x);
        splay(x);
        return x;
    }
    void link(int x, int y)
    {
        makeroot(x);
        if(find(y) != x) tr[x].pre = y;
    }
    bool cut(int x, int y)
    {
        makeroot(x);
        if(find(y) != x || tr[x].info.size > 2) return false;
        tr[y].pre = tr[x].ch[1] = 0; pushup(x);
        return true;
    }

    void apply(int u, int v, const Tag &t) { split(u,v); apply(v,t); }
    Info query(int u, int v) { split(u,v); return tr[v].info; }
};

struct Tag
{
    int p = 0, m = 1;
    bool rt = false;
    void apply(const Tag &t)
    {
        m *= t.m; p *= t.m; p += t.p;
        rt ^= t.rt;
    }
};
struct Info
{
    int key = 0, sum = 0;
    int size = 0;
    void fresh() { this->sum = key; this->size = 1; }
    void apply(const Tag &t)
    {
        sum = sum * t.m + t.p * size;
        key = key * t.m + t.p;
    }
    Info operator+(Info o)
    {
        Info res = *this;
        res.sum += o.sum;
        res.size += o.size;
        return res;
    }
};
```

MAX_FLow

```
template<typename T>
struct Max_Flow
{
    struct edge { int to; T f; };

    int n;
    std::vector<edge> e;
    std::vector<std::vector<int>> adj;
    std::vector<int> cur,dis;

    Max_Flow(int _n = 0) { init(_n); }
    void init(int _n = 0)
    {
        n = _n;
        e.clear(); adj.assign(n+1, {});
        cur.resize(n+1); dis.resize(n+1);
    }

    void addEdge(int u, int v, T f)
    {
        adj[u].push_back(e.size());
        e.emplace_back(v, f);
    }
    void add(int u, int v, T f = 0)
    {
        addEdge(u,v,f);
        addEdge(v,u,0);
    }

    int cntu;
    bool bfs(int s, int t)
    {
        dis.assign(n+1,-1);
        std::queue<int> q;
        while(!q.empty()) q.pop();
        q.push(s); dis[s] = 0;
        while(!q.empty())
        {
            int u = q.front(); q.pop();
            for(auto i: adj[u])
            {
                auto [v, f] = e[i];
                if(f > 0 && dis[v] == -1)
                {
                    q.push(v); dis[v] = dis[u] + 1;
                    if(v == t) return true;
                }
            }
        }
        return false;
    }
    T dfs(int u, int t, T sum)
    {
        if(u == t) return sum;
        T rem = sum;
```

```
        for(int &i = cur[u]; i < int(adj[u].size()); ++i)
        {
            const int j = adj[u][i]; auto [v,f] = e[j];
            if(f > 0 && dis[v] == dis[u]+1)
            {
                T k = dfs(v,t,std::min(rem, f));
                e[j].f -= k; e[j^1].f += k;
                if((rem-=k) == 0) return sum;
            }
        }
        return sum-rem;
    }
    T dinic(int s, int t)
    {
        T flow = 0;
        while(bfs(s,t))
            cur.assign(n+1, 0), flow += dfs(s, t, std::numeric_limits<T>::max()>>1);
        return flow;
    }
    std::vector<bool> min_cut()
    {
        std::vector<bool> c(n+1);
        For(i,1,n) c[i] = (dis[i] != -1);
        return c;
    }
};
```

MCF

```
struct MCF
{
    struct edge { int to; int f,c; };

    int n;
    std::vector<edge> e;
    std::vector<std::vector<int>> adj;
    std::vector<int> pre;
    std::vector<i64> h,dis;
    const i64 INF = std::numeric_limits<i64>::max()>>1;

    MCF(int _n = 0) { init(_n); }
    void init(int _n = 0)
    {
        n = _n;
        e.clear(); adj.assign(n+1, {});
    }

    void addEdge(int u, int v, int f, int c)
    {
        adj[u].push_back(e.size());
        e.emplace_back(v,f,c);
    }
    void add(int u, int v, int f = 0, int c = 0)
    {
        addEdge(u,v,f,c);
        addEdge(v,u,0,-c);
    }

    bool dij(int s, int t)
    {
        dis.assign(n+1, INF); pre.assign(n+1, -1);
        using pli = std::pair<i64,int>;
        std::priority_queue<pli,std::vector<pli>,std::greater<>> q;
        q.emplace(dis[s]=0,s);
        while(!q.empty())
        {
            auto [d,u] = q.top(); q.pop();
            if(dis[u] < d) continue;
            for(int i: adj[u])
            {
                auto [v,f,c] = e[i];
                if(f > 0 && dis[v] > d + h[u] - h[v] + c)
                    q.emplace(dis[v]=d+h[u]-h[v]+c, v), pre[v] = i;
            }
        }
        return dis[t] != INF;
    }
    std::pair<int,i64> dinic(int s, int t)
    {
        int flow(0); i64 cost(0); h.assign(n+1,0);
        while(dij(s,t))
        {
            int mn = std::numeric_limits<int>::max()>>1;
            For(i,1,n) h[i] += dis[i];
            for(int i = t; i != s; i = e[pre[i]^1].to)
                chkmin(mn,e[pre[i]].f);
            for(int i = t; i != s; i = e[pre[i]^1].to)
                e[pre[i]].f -= mn, e[pre[i]^1].f += mn;
            flow += mn; cost += mn * h[t];
        }
        return {flow,cost};
    }
};
```

ModInt

```cpp
template<u32 P>
constexpr u32 mul(u32 a, u32 b) {
    return 1llu * a * b % P;
}

template<u64 P>
constexpr u64 mul(u64 a, u64 b) {
    u64 res = a * b - u64(1.l * a * b / P - 0.5l) * P;
    res %= P;
    return res;
}

template<typename U, U P>
requires std::unsigned_integral<U>
struct MIntBase
{
    U x;

    static U mod;
    constexpr static U getMod() { return P? P: mod; }
    constexpr static void setMod(U Mod_) { mod = Mod_; }

    constexpr MIntBase() : x(0) {}
    template<typename T>
    requires std::integral<T>
    constexpr MIntBase(T x_) : x(norm(x_ % T {P})) {}

    constexpr U norm(U x) {
        if ((x >> (8 * sizeof(U) - 1) & 1) == 1) {
            x += getMod();
        }
        if (x >= getMod()) {
            x -= getMod();
        }
        return x;
    }
    constexpr int val() const { return x; }
    explicit constexpr operator U() const { return x; }

    constexpr MIntBase operator+=(const MIntBase o) { return x = norm(x+o.x),
*this; }
    constexpr MIntBase operator-=(const MIntBase o) { return x = norm(x-o.x),
*this; }
    constexpr MIntBase operator*=(const MIntBase o) { return x = mul<P>(x,
o.val()), *this; }
    constexpr MIntBase operator/=(const MIntBase o) { return *this *= o.inv(); }
    friend constexpr MIntBase operator+(const MIntBase a, const MIntBase b)
{ MIntBase res(a); return res += b; }
    friend constexpr MIntBase operator-(const MIntBase a, const MIntBase b)
{ MIntBase res(a); return res -= b; }
    friend constexpr MIntBase operator*(const MIntBase a, const MIntBase b)
{ MIntBase res(a); return res *= b; }
    friend constexpr MIntBase operator/(const MIntBase a, const MIntBase b)
{ MIntBase res(a); return res /= b; }
    friend constexpr MIntBase operator-(const MIntBase a) { MIntBase res(0);
return res -= a; }

    friend constexpr bool operator==(const MIntBase &a, const MIntBase &b)
{ return a.x == b.x; }
    friend constexpr bool operator!=(const MIntBase &a, const MIntBase &b)
{ return a.x != b.x; }
    friend constexpr auto operator<=>(const MIntBase &a, const MIntBase &b)
{ return a.x <=> b.x; }
    friend std::istream &operator>>(std::istream &is, MIntBase &a) { is >> a.x,
a.x = a.norm(a.x); return is; }
    friend std::ostream &operator<<(std::ostream &os, const MIntBase &a) { return
os << a.x; }

    constexpr MIntBase inv() const { return qpow(*this,getMod()-2); }
};

template<u32 P>
using MInt= MIntBase<u32, P>;

template<u64 P>
using MLong = MIntBase<u64, P>;

using Z = MInt<P>;
```

Poly

```cpp
template<int V, int P>
constexpr MInt<P> CInv = MInt<P>(V).inv();

template<int P>
constexpr MInt<P> findG()
{
    MInt<P> i = 2;
    int k = __builtin_ctz(P - 1);
    while(qpow(i, (P-1) >> 1) == 1) ++i;
    return qpow(i, (P-1) >> k);
}

template<int P>
constexpr MInt<P> G = findG<P>();
template<>
constexpr MInt<998244353> G<998244353> {3};

template<int P>
std::vector<MInt<P>> w{0, 1};

template<int P>
void extendw(MInt<P> l)
{
    int limit = l.val();
    if(int(w<P>.size()) < limit)
    {
        int t = __builtin_ctz(w<P>.size());
        w<P>.resize(limit);
        while((1 << t) < limit)
        {
            MInt<P> wn = qpow(G<P>, (P-1)/(1<<(t+1)));
```

```
            w<P>[1 << t] = 1;
            for(int j = (1 << t) + 1; j < (1 << (t+1)); ++j)
                w<P>[j] = w<P>[j-1] * wn;
            ++t;
        }
    }
}
template<int P>
void NTT(std::vector<MInt<P>> &a)
{
    int limit = a.size();
    extendw(MInt<P>(limit));
    for(int mid = limit >> 1; mid >= 1; mid >>= 1)
    {
        for(int j = 0; j < limit; j += mid<<1)
        {
            for(int k = j, s = mid; k < j+mid; ++k, ++s)
            {
                MInt<P> x = a[k], y = a[k+mid];
                a[k] = x + y;
                a[k+mid] = (x - y) * w<P>[s];
            }
        }
    }
}
template<int P>
void INTT(std::vector<MInt<P>> &a)
{
    int limit = a.size();
    extendw(MInt<P>(limit));
    for(int mid = 1; mid < limit; mid <<= 1)
    {
        for(int j = 0; j < limit; j += mid<<1)
        {
            for(int k = j, s = mid; k < j+mid; ++k, ++s)
            {
                MInt<P> x = a[k], y = w<P>[s] * a[k+mid];
                a[k] = x + y;
                a[k+mid] = x - y;
            }
        }
    }
    reverse(a.begin()+1, a.end());
    MInt<P> inv = MInt<P>(limit).inv();
    for(int i = 0; i < limit; ++i) a[i] *= inv;
}

template<int P = 998244353>
struct Poly: std::vector<MInt<P>>
{
    using Z = MInt<P>;
    explicit constexpr Poly(int n = 0) : std::vector<Z>(n) {}
    explicit constexpr Poly(const std::vector<Z> &a) : std::vector<Z>(a) {}
    constexpr Poly(const std::initializer_list<Z> &a) : std::vector<Z>(a) {}
    template<class F> explicit constexpr Poly(int n, F f) : std::vector<Z>(n) { for(int i
= 0; i < n; ++i) (*this)[i] = f(i); }
    template<class InputIt, class = std::_RequireInputIter<InputIt>>
    explicit constexpr Poly(InputIt first, InputIt last) : std::vector<Z>(first, last) {}

    constexpr Poly shift(int k) const
    {
        if(k >= 0)
        {
            auto b = *this;
            b.insert(b.begin(), k, 0);
            return b;
        }
        else if (int(this->size()) <= -k) return Poly();
        else return Poly(this->begin() + (-k), this->end());
    }
    constexpr Poly trunc(int k) const
    {
        Poly f = *this;
        f.resize(k);
        return f;
    }
    constexpr friend Poly operator+(const Poly &a, const Poly &b)
    {
        Poly res(std::max(a.size(), b.size()));
        for(int i = 0; i < int(a.size()); ++i) res[i] = a[i];
        for(int i = 0; i < int(b.size()); ++i) res[i] += b[i];
        return res;
    }
    constexpr friend Poly operator-(const Poly &a, const Poly &b)
    {
        Poly res(std::max(a.size(), b.size()));
        for(int i = 0; i < int(a.size()); ++i) res[i] = a[i];
        for(int i = 0; i < int(b.size()); ++i) res[i] -= b[i];
        return res;
    }
    constexpr friend Poly operator-(const Poly &a)
    {
        std::vector<Z> res(a.size());
        for(int i = 0; i < int(res.size()); ++i) res[i] = -a[i];
        return Poly(res);
    }
    constexpr friend Poly operator*(Poly a, Poly b)
    {
        if(a.size() == 0 || b.size() == 0) return Poly();
        if(a.size() < b.size()) std::swap(a, b);
        int limit = 1, tot = a.size() + b.size() - 1;
        while(limit < tot) limit <<= 1;
        if(((P - 1) & (limit - 1)) != 0 || b.size() < 128)
        {
            Poly c(a.size() + b.size() - 1);
            for(int i = 0; i < int(a.size()); ++i)
                for(int j = 0; j < int(b.size()); ++j)
                    c[i+j] += a[i] * b[j];
            return c;
        }
        a.resize(limit); b.resize(limit);
```

```
    NTT(a); NTT(b);
    for(int i = 0; i < limit; ++i) a[i] *= b[i];
    INTT(a);
    a.resize(tot);
    return a;
}
constexpr friend Poly operator*(Z a, Poly b)
{
    for(int i = 0; i < int(b.size()); ++i) b[i] *= a;
    return b;
}
constexpr friend Poly operator*(Poly a, Z b)
{
    for(int i = 0; i < int(a.size()); ++i) a[i] *= b;
    return a;
}
constexpr friend Poly operator/(Poly a, Z b)
{
    for (int i = 0; i < int(a.size()); ++i) a[i] /= b;
    return a;
}
constexpr Poly &operator+=(Poly b) { return (*this) = (*this) + b; }
constexpr Poly &operator-=(Poly b) { return (*this) = (*this) - b; }
constexpr Poly &operator*=(Poly b) { return (*this) = (*this) * b; }
constexpr Poly &operator*=(Z b) { return (*this) = (*this) * b; }
constexpr Poly &operator/=(Z b) { return (*this) = (*this) / b; }

constexpr Poly deriv() const
{
    if(this->empty()) return Poly();
    Poly res(this->size() - 1);
    for(int i = 0; i < int(res.size()); ++i) res[i] = (i + 1) * (*this)[i + 1];
    return res;
}
inline static std::vector<Z> invi{0};
constexpr Poly integ() const
{
    Poly res(this->size()+1);
    if(invi.size() < this->size()+1)
    {
        int lst = invi.size();
        invi.resize(this->size()+1);
        for(int i = lst; i <= int(this->size()); ++i) invi[i] = Z(i).inv();
    }
    for(int i = 0; i < int(this->size()); ++i) res[i+1] = (*this)[i] * invi[i+1];
    return res;
}
constexpr Poly inv(int n) const
{
    Poly res{(*this)[0].inv()};
    int m = 1;
    while(m < n) m <<= 1, res = (res * (Poly{2}-trunc(m)*res)).trunc(m);
    return res.trunc(n);
}
constexpr Poly ln(int n) const { return (deriv() * inv(n)).integ().trunc(n); }
constexpr Poly exp(int n) const
{
    Poly res{1}; int m = 1;
    while(m < n) m <<= 1, res = (res * (Poly{1}-res.ln(m)+trunc(m))).trunc(m);
    return res.trunc(n);
}
constexpr Poly pow(int n, int k, int k2 = -1, bool flag = false) const
{
    int pos = 0;
    if(k2 < 0) k2 = k;
    while(pos < int(this->size()) && (*this)[pos] == 0) ++pos;
    if(pos == int(this->size()) || (pos && flag) || 1ll*pos*k >= n) return Poly(n);
    Z a0 = (*this)[pos];
    auto f = shift(-pos) * a0.inv();
    return (f.ln(n-pos*k)*k).exp(n-pos*k).shift(pos*k) * qpow(a0, k2);
}
constexpr Poly sqrt(int n) const
{
    Poly res{1};
    int m = 1;
    while(m < n) m <<= 1, res = (res + (trunc(m) * res.inv(m)).trunc(m)) * CInv<2, P>;
    return res.trunc(n);
}
constexpr std::pair<Poly, Poly> div(Poly b) const
{
    if(this->size() < b.size()) return {Poly(0), *this};
    Poly a = *this;
    std::reverse(a.begin(), a.end());
    std::reverse(b.begin(), b.end());
    Poly q = (a * b.inv(this->size() - b.size() + 1)).trunc(this->size() - b.size() +
1);
    std::reverse(q.begin(), q.end());
    std::reverse(b.begin(), b.end());
    return {q, (*this - q * b).trunc(b.size() - 1)};
}
constexpr Poly mulT(Poly b) const
{
    int n = b.size();
    if(n == 0) return Poly();
    std::reverse(b.begin(), b.end());
    return ((*this) * b).shift(1 - n);
}
constexpr std::vector<Z> eval(std::vector<Z> x) const
{
    if(this->empty()) return std::vector<Z>(x.size(), 0);
    const int n = std::max(x.size(), this->size());
    std::vector<Poly> a(n * 4 + 1);
    std::vector<Z> ans(x.size());
    x.resize(n);
    #define ls(i) i * 2
    #define rs(i) i * 2 + 1
    auto build = [&](auto &&self, int i, int l, int r) -> void
    {
        if(l == r)
        {
            a[i] = {1, -x[l]};
            return;
```

```
            }
            int mid = (l + r) / 2;
            self(self, ls(i), l, mid); self(self, rs(i), mid + 1, r);
            a[i] = a[ls(i)] * a[rs(i)];
        };
        auto query = [&](auto &&self, int i, int l, int r, const Poly &t) -> void
        {
            if(l == r)
            {
                if(l < (int)ans.size()) ans[l] = t[0];
                return;
            }
            int mid = (l + r) / 2;
            self(self, ls(i), l, mid, t.mulT(a[rs(i)]).trunc(mid - l + 1));
            self(self, rs(i), mid + 1, r, t.mulT(a[ls(i)]).trunc(r - mid));
        };
        #undef ls
        #undef rs
        build(build, 1, 0, n - 1); query(query, 1, 0, n - 1, mulT(a[1].inv(n)));
        return ans;
    }
};
```

PAM

```
struct Trie
{
    int cnt,len,fail;
    int ch[26];
    Trie() { cnt = len = fail = 0; memset(ch,0,sizeof(ch)); }
};
class PAM
{
    private:
        int cnt,last,cur;
        char s[MAXN];
        Trie a[MAXN];
    public:
        void init();
        int get_fail(int x) { while(cur-a[x].len-1<1 || s[cur-a[x].len-1]!=s[cur]) x =
a[x].fail; return x; }
        int get_ans() { return a[last].cnt; }
        void ins(int);
}pam;
void PAM::init()
{
    cnt = 1; cur = last = 0; a[0].len = 0; a[1].len = -1;
    a[0].fail = 1; a[1].fail = 0;
}
void PAM::ins(int ch)
{
    s[++cur] = ch;
    int p = get_fail(last);
    if(!a[p].ch[ch])
    {
        a[++cnt].len = a[p].len+2;
        a[cnt].fail = a[get_fail(a[p].fail)].ch[ch];
        a[cnt].cnt = a[a[cnt].fail].cnt + 1;
        a[p].ch[ch] = cnt;
    }
    last = a[p].ch[ch];
}
```

Mod_Pers_Seg_Tree

```
struct Mod_Pers_Seg_Tree
{
    int sz, T = 0;
    std::vector<int> rt;
    std::vector<int> cnt,ls,rs;

    Mod_Pers_Seg_Tree(int rts = 0, int _sz = 0) { init(rts,_sz); }
    void init(int rts = 0, int _sz = 0)
    {
        sz = _sz; int lgs = 32 - __builtin_clz(sz), lgr = 32 - __builtin_clz(rts), lg =
lgs * lgr / 2;
        rt.assign(rts+1,0);
        cnt.assign(rts * lg, 0); ls.assign(rts * lg, 0); rs.assign(rts * lg, 0);
    }

    void modify(int &i, int l, int r, int k, int v)
    {
        if(!i) i = ++T;
        if(T >= cnt.size()) cnt.push_back(0), ls.push_back(0), rs.push_back(0);
        cnt[i] += v;
        if(l == r) return;
        int mid = (l+r)>>1;
        if(k <= mid) modify(ls[i],l,mid,k,v);
        else modify(rs[i],mid+1,r,k,v);
    }

    int lowbit(int x) { return x&-x; }
    void add(int i, int k, int v)
    {
        while(i < rt.size())
            modify(rt[i],1,sz,k,v), i += lowbit(i);
    }
    int query(int l, int r, int k, vector<int> &L, vector<int> &R)
    {
        if(l == r) return l;
        int mid = (l+r)>>1, sum = 0;
        for(auto x: L) sum -= cnt[ls[x]];
        for(auto x: R) sum += cnt[ls[x]];
        if(k <= sum)
        {
            for(auto &x: L) x = ls[x];
            for(auto &x: R) x = ls[x];
            return query(l,mid,k,L,R);
        }
        else
        {
            for(auto &x: L) x = rs[x];
            for(auto &x: R) x = rs[x];
```

```
            return query(mid+1,r,k-sum,L,R);
        }
    }
    int query(int l, int r, int k)
    {
        std::vector<int> L,R;
        while(l) L.push_back(rt[l]), l -= lowbit(l);
        while(r) R.push_back(rt[r]), r -= lowbit(r);
        return query(1,sz,k,L,R);
    }
};
```

Pers_Seg_Tree

```
template<class Info>
struct Pers_Seg_Tree
{
    int sz, T = 0, s = 1;
    std::vector<int> rt;
    std::vector<int> ls,rs;
    std::vector<Info> info;
    Pers_Seg_Tree(int n = 0, int _sz = 0) { init(n, _sz); }
    void sets(int _s) { s = _s; }
    void init(int n = 0, int _sz = 0)
    {
        sz = _sz; int lg = 33 - __builtin_clz(sz);
        rt.assign(n+1, 0);
        info.assign((sz << 1) + n * lg, Info()); ls.assign((sz << 1) + n * lg, 0);
rs.assign((sz << 1) + n * lg, 0);
        std::function<void(int&,int,int)> build = [&](int &i, int l, int r)
        {
            i = ++T;
            if(l==r) return;
            int mid = (l+r) >> 1;
            build(ls[i],l,mid); build(rs[i],mid+1,r);
        };
        build(rt[0],s,sz);
    }

    void pushup(int i) { info[i] = info[ls[i]] + info[rs[i]]; }
    void modify(int i, int &j, int l, int r, int k, const Info &I)
    {
        j = ++T;
        while(int(info.size()) <= T) info.push_back(Info()), ls.push_back(0),
rs.push_back(0);
        info[j] = info[i]; ls[j] = ls[i]; rs[j] = rs[i];
        if(l == r) { info[j] = info[j] + I; return; }
        int mid = (l+r)>>1;
        if(k <= mid) modify(ls[i],ls[j],l,mid,k,I);
        else modify(rs[i],rs[j],mid+1,r,k,I);
        pushup(j);
    }
    Info query(int i, int j, int l, int r, int x, int y)
    {
        if(y < l || r < x) return Info();
        if(x <= l && r <= y) return info[j] - info[i];
        int mid = (l+r)>>1;
        return query(ls[i],ls[j],l,mid,x,y) + query(rs[i],rs[j],mid+1,r,x,y);
    }
    template<class F>
    int findFirst(int i, int j, int l, int r, int x, int y, F k)
    {
        if(y < l || r < x || !k(info[j]-info[i])) return -1;
        if(l == r) return l;
        int mid = (l+r)>>1;
        int res = findFirst(ls[i],ls[j],l,mid,x,y,k);
        if(!~res) res = findFirst(rs[i],rs[j],mid+1,r,x,y,k-(info[ls[j]]-info[ls[i]]));
        return res;
    }
    template<class F>
    int findLast(int i, int j, int l, int r, int x, int y, F k)
    {
        if(y < l || r < x || !k(info[j]-info[i])) return -1;
        if(l == r) return l;
        int mid = (l+r)>>1;
        int res = findLast(rs[i],rs[j],mid+1,r,x,y,k);
        if(!~res) res = findLast(ls[i],ls[j],l,mid,x,y,k-(info[rs[j]]-info[rs[i]]));
        return res;
    }
    void modify(int lst, int cur, int k, const Info &I)
{ modify(rt[lst],rt[cur],s,sz,k,I); }
    Info query(int l, int r, int x, int y) { return query(rt[l-1],rt[r],s,sz,x,y); }
    template<class F> int findFirst(int l, int r, int x, int y, F k) { return
findFirst(rt[l-1],rt[r],s,sz,x,y,k); }
    template<class F> int findLast(int l, int r, int x, int y, F k) { return
findLast(rt[l-1],rt[r],s,sz,x,y,k); }
};

struct Info
{
    int cnt;
    Info(int n = 0): cnt(n) {}
    Info operator+(const Info &o) const { return Info(cnt + o.cnt); }
    Info operator-(const Info &o) const { return Info(cnt - o.cnt); }
};
struct Find
{
    int k = 0;
    bool operator()(const Info &o) { return k <= o.cnt; }
    Find operator-(const Info &o) const { return Find{k - o.cnt}; }
};


struct Pers_Trie
{
    std::vector<int> cnt, rt;
    std::vector<std::array<int, 2>> nxt;
    int tot = 0, hb;

    Pers_Trie(int n = 1, int b = 30) { init(n, b); };
    void init(int n = 1, int b = 30)
    {
        hb = b - 1;
```

```
        rt.assign(n + 1, 0);
        cnt.assign(b * n + 2, 0); nxt.assign(b * n + 2, {});
    }

    void insert(int lst, int cur, int x, int d = 1)
    {
        if(!rt[cur]) rt[cur] = ++tot;
        int u = rt[lst], v = rt[cur];
        cnt[v] = cnt[u] + d;
        for(int i = hb; i >= 0; --i) {
            bool p = x >> i & 1;
            nxt[v][p] = ++tot;
            cnt[nxt[v][p]] = cnt[nxt[u][p]];
            nxt[nxt[v][p]] = nxt[nxt[u][p]];
            u = nxt[u][p]; v = nxt[v][p];
            cnt[v] += d;
        }
    }
    int query(int l, int r, int x)
    {
        int u = rt[l - 1], v = rt[r];
        if(cnt[v] <= cnt[u]) return 0;
        int res = 0;
        for(int i = hb; i >= 0; --i) {
            bool p = x >> i & 1;
            int cu = cnt[nxt[u][!p]], cv = cnt[nxt[v][!p]];
            if(nxt[v][!p] && cv > cu) u = nxt[u][!p], v = nxt[v][!p], res ^= (1 << i);
            else u = nxt[u][p], v = nxt[v][p];
        }
        return res;
    }
};
```

点+凸包

```
struct Point
{
    i64 x = 0, y = 0;
    constexpr Point& operator+=(const Point o) { return x += o.x, y += o.y, *this; }
    constexpr Point& operator-=(const Point o) { return x -= o.x, y -= o.y, *this; }
    constexpr Point& operator*=(const int k) { return x *= k, y *= k, *this; }
    constexpr Point operator+(const Point o) const { Point res(*this); return res += o; }
    constexpr Point operator-(const Point o) const { Point res(*this); return res -= o; }
    constexpr Point operator*(const int k) const { Point res(*this); return res *= k; }
    constexpr i64 operator%(const Point o) { return o.y * x - o.y * x; }
    constexpr i64 operator*(const Point o) { return x * o.x + y * o.y; }
    constexpr bool operator<(const Point &o) const { return x == o.x? y < o.y: x < o.x; }
    constexpr bool operator==(const Point &o) const { return x == o.x && y == o.y; }
    constexpr i64 norm2() { return x * x + y * y; }
    friend i64 dist(const Point a, const Point b) { return (b-a).norm2(); }
    friend std::istream& operator>>(std::istream& is, Point &o) { return is >> o.x >>
o.y; }
};
std::vector<Point> getHull(std::vector<Point> p)
{
    std::vector<Point> h, l{{-1, -1}};
    std::sort(p.begin() + 1, p.end());
    p.erase(std::unique(p.begin() + 1, p.end()), p.end());
    if(p.size() <= 2) return p;
    For(i, 1, p.size() - 1) {
        while(h.size() > 1 && (p[i] - h.back()) % (p[i] - h[h.size() - 2]) <= 0)
h.pop_back();
        while(l.size() > 2 && (p[i] - l.back()) % (p[i] - l[l.size() - 2]) >= 0)
l.pop_back();
        l.push_back(p[i]); h.push_back(p[i]);
    }
    l.pop_back();
    std::reverse(h.begin(), h.end());
    h.pop_back();
    l.insert(l.end(), h.begin(), h.end());
    return l;
}
```

PR

```
bool Miller_Rabin(_ll p)
{
    if(p < 2) return 0;
    if(p == 2) return 1;
    if(p == 3) return 1;
    _ll d = p - 1, r = 0;
    while(!(d & 1)) ++r, d >>= 1;
    for(_ll k = 0; k < 10; ++k)
    {
        _ll a = rand() % (p - 2) + 2;
        _ll x = qpow(a, d, p);
        if(x == 1 || x == p - 1) continue;
        for(int i = 0; i < r - 1; ++i)
        {
            x = (__int128)x * x % p;
            if(x == p - 1) break;
        }
            if(x != p - 1) return 0;
    }
    return 1;
}
_ll Pollard_Rho(_ll x)
{
    if(x == 4) return 2;
    _ll s = 0, t = 0;
    _ll c = (_ll)rand() % (x - 1) + 1;
    int step = 0, goal = 1;
    _ll val = 1;
    for (goal = 1;; goal *= 2, s = t, val = 1) {
        for (step = 1; step <= goal; ++step) {
            t = ((__int128)t * t + c) % x;
            val = (__int128)val * abs(t - s) % x;
            if ((step % 127) == 0) {
                _ll d = __gcd(val, x);
                if (d > 1) return d;
            }
        }
        _ll d = __gcd(val, x);
        if (d > 1)
```

```
            return d;
    }
}
void fac(_ll x)
{
    if (x <= max_factor || x < 2)
        return;
    if (Miller_Rabin(x)) {
        max_factor = max(max_factor, x);
        return;
    }
    _ll p = x;
    while (p >= x)
        p = Pollard_Rho(x);
    while (!(x % p))
        x /= p;
    fac(x), fac(p);
}
```

Poly2

```
ui qpow(ui x, ui n)
{
    ui res = 1;
    while(n)
    {
        if(n&1) res = 1ll*res*x%P;
        n >>= 1; x = 1ll*x*x%P;
    }
    return res;
}
ui limit, inv, w[MAXN];
void init(ui n)
{
    limit = 2;
    while(limit < n) limit <<= 1;
    inv = qpow(limit,P-2);
}
void initw()
{
    for(ui i = 1; i < limit; i <<= 1)
    {
        ui wn = qpow(G, (P-1)/(i<<1));
        w[i] = 1;
        for(ui j = i+1; j < (i<<1); ++j)
            w[j] = 1ll*w[j-1]*wn%P;
    }
}
void NTT(ui a[])
{
    for(ui mid = limit>>1; mid >= 1; mid >>= 1) {
        for(ui j = 0; j < limit; j += mid<<1) {
            for(ui k = j, s = mid; k < j+mid; ++k, ++s) {
                ui x = a[k], y = a[k+mid];
                a[k] = x+y>=P?x+y-P:x+y;
                a[k+mid] = 1ll*(x<y?P+x-y:x-y)*w[s]%P;
            }
        }
    }
}
void INTT(ui a[])
{
    for(ui mid = 1; mid < limit; mid <<= 1) {
        for(ui j = 0; j < limit; j += mid<<1) {
            for(ui k = j, s = mid; k < j+mid; ++k, ++s) {
                ui x = a[k], y = (1ll * w[s] * a[k+mid])%P;
                a[k] = x+y>=P?x+y-P:x+y;
                a[k+mid] = x<y?P+x-y:x-y;
            }
        }
    }
    reverse(a+1, a+limit);
    for(ui i = 0; i < limit; ++i)
        a[i] = 1ll*a[i]*inv%P;
}

ui invi[MAXN];
void intg(ui a[], ui n)
{
    foR(i,n-1,1) a[i] = 1ll*a[i-1]*invi[i]%P;
    a[0] = 0;
}
void der(ui a[], ui n)
{
    For(i,1,n-1) a[i-1] = 1ll*a[i]*i%P;
    a[n-1] = 0;
}
ui c[MAXN];
void poly_inv(ui n, ui a[], ui b[])
{
    ui m = 1; b[0] = qpow(a[0], P-2); inv = inv2;
    while(m < n)
    {
        m <<= 1; limit = m<<1; inv = 1ll*inv*inv2%P;
        std::copy(a, a+m, c);
        std::fill(c+m, c+limit, 0);
        NTT(c); NTT(b);
        for(ui i = 0; i < limit; ++i)
            b[i] = (P+2 - 1ll*c[i]*b[i]%P)%P * b[i] % P;
        INTT(b);
        std::fill(b+m, b+limit, 0);
    }
}
ui aa[MAXN],b[MAXN];
void poly_ln(ui n, ui a[])
{
    init(n<<1);
    copy(a,a+n,aa); fill(aa+n,aa+limit,0);
    fill(b,b+limit,0);
    der(a,n);
    poly_inv(n,aa,b);
    init(n<<1), NTT(a), NTT(b);
    For(i,0,limit-1) a[i] = 1ll*a[i]*b[i]%P;
```

```
    INTT(a);
    intg(a,n);
}
ui lnb[MAXN];
void poly_exp(ui n, ui a[], ui b[])
{
    ui m = 1; b[0] = 1;
    while(m < n)
    {
        m <<= 1;
        copy(b, b+m, lnb);
        poly_ln(m,lnb);
        init(m<<1);
        For(i,0,m-1) lnb[i] = a[i]<lnb[i]? P+a[i]-lnb[i]: a[i]-lnb[i];
        fill(lnb+m,lnb+limit,0);


        ++lnb[0];
        NTT(b); NTT(lnb);
        for(ui i = 0; i < limit; ++i)
            b[i] = 1ll*b[i]*lnb[i]%P;
        INTT(b);
        std::fill(b+m, b+limit, 0);
    }
}

void poly_pow(ui n, ui k1, ui k2, ui f, ui a[], ui res[])
{
    ui pos = 0;
    while(pos < n && !a[pos]) ++pos;
    if(pos >= n || (pos && f) || 1ll*pos*k1 >= n) return;

    ui inv0,a0;
    inv0 = qpow(a[pos], P-2); a0 = a[pos];
    for(ui i = pos; i < n; ++i) a[i] = 1ll*a[i]*inv0%P;
    copy(a+pos,a+n,a);

    init(n<<1); initw();
    For(i,0,limit-1) invi[i] = qpow(i,P-2);
    poly_ln(n-pos*k1,a);
    For(i,0,n-pos*k1-1) a[i] = 1ll*a[i]*k1%P;
    poly_exp(n-pos*k1,a,res);

    a0 = qpow(a0,k2);
    For(i,0,n-k1*pos-1) res[i] = 1ll*res[i]*a0%P;
    copy(res,res+n-k1*pos,res+k1*pos);
    fill(res,res+k1*pos,0);
}


int cnt,p[MAXN],phi[MAXN];
bool v[MAXN],b[MAXN<<1];
void get_prime(int n)
{
    v[1] = v[0] = 1; b[2] = b[4] = 1; phi[1] = 1;
    For(i,2,n)
    {
        if(!v[i])
        {
            p[++cnt] = i; phi[i] = i-1;
            if(i > 2) for(_ll j = i; j <= n; j *= i) b[j] = b[j<<1] = 1;
        }
        for(int j = 1; j <= cnt && 1ll*i*p[j] <= n; ++j)
        {
            v[i*p[j]] = 1;
            if(!(i%p[j])) { phi[i*p[j]] = phi[i] * p[j]; break; }
            phi[i*p[j]] = phi[i] * phi[p[j]];
        }
    }
}

int T,n,q,d,m,mm,g;
bool vis[MAXN];
int main()
{
    get_prime(MAXN-6);
    cin >> T;
    while(T--)
    {
        cin >> n >> d; g = 0;
        if(!b[n]) { cout << "0\n\n"; continue; }
        vector<int> ans,fac;
        cout << phi[phi[n]] << '\n';
        int r = pow(n,0.25)+1; mm = m = phi[n];
        if(!v[m]) fac.push_back(m), vis[m] = 1;
        else for(int i = 1; i <= cnt && p[i] <= mm; ++i)
        {
            if(!(mm%p[i]))
            {
                fac.push_back(p[i]);
                while(!(mm%p[i])) mm /= p[i];
                for(_ll j = 1; j*p[i] <= m; ++j) vis[j*p[i]] = 1;
            }
        }
        for(int f = 0; ; ++g, f = 0)
        {
            while(qpow(g,m,n) != 1) ++g;
            for(auto x: fac)
                if(qpow(g,m/x,n) == 1)
                {   f = 1;  break;  }
            if(!f) break;
        }
        For(i,1,m)
        {
            if(vis[i]) vis[i] = 0;
            else ans.push_back(qpow(g,i,n));
        }
        sort(ans.begin(),ans.end());
        for(int i = d-1; i < ans.size(); i += d)
            cout << ans[i] << ' ';
        cout << '\n';
    }
```

```
    return 0;
}
```

RMQ

```
template<class T, class Cmp = std::less<T>>
struct RMQ
{
    const Cmp cmp = Cmp();
    static constexpr unsigned int B = 64;
    int n;
    std::vector<std::vector<T>> a;
    std::vector<T> pre, suf, ini;
    std::vector<u64> st;
    RMQ() {}
    RMQ(const std::vector<T> &v) { init(v); }
    void init(const std::vector<T> &v)
    {
        n = v.size();
        pre = suf = ini = v;
        st.resize(n);
        if(!n) return;
        const int M = (n-1) / B + 1, lg = std::__lg(M);
        a.assign(lg+1, std::vector<T>(M));
        For(i, 0, M-1)
        {
            a[0][i] = v[i*B];
            For(j, 1, std::min(B, n - i * B) - 1)
                a[0][i] = std::min(a[0][i], v[i * B + j], cmp);
        }
        For(i, 1, n-1) if(i % B) pre[i] = std::min(pre[i], pre[i-1], cmp);
        foR(i, n-2, 0) if(i % B != B - 1) suf[i] = std::min(suf[i], suf[i+1], cmp);
        For(j, 0, lg - 1) For(i, 0, M - (2 << j)) a[j+1][i] = std::min(a[j][i], a[j][i +
(1 << j)], cmp);
        For(i, 0, M-1)
        {
            const int l = i * B, r = std::min(1u * n, l + B);
            u64 s = 0;
            For(j, l, r-1)
            {
                while(s && cmp(v[j], v[std::__lg(s) + l])) s ^= 1llu << std::__lg(s);
                s |= 1llu << (j - l);
                st[j] = s;
            }
        }
    }
    T operator()(int l, int r)
    {
        ++r;
        if(l / B == (r - 1) / B)
        {
            int x = B * (l / B);
            return ini[__builtin_ctzll(st[r-1] >> (l-x)) + l];
        }
        T ans = std::min(suf[l], pre[r-1], cmp);
        l /= B; ++l; r /= B;
        if(l < r)
        {
            int k = std::__lg(r - l);
            ans = std::min({ans, a[k][l], a[k][r - (1 << k)]}, cmp);
        }
        return ans;
    }
};
```

SAM

```
struct node
{
    int ch[27],len,pre,cnt;
    node() { memset(ch,0,sizeof(ch)); len = pre = cnt = 0; }
    node(const node &a) { memcpy(ch,a.ch,sizeof(ch)); len = a.len; pre = a.pre; cnt =
a.cnt; }
    node operator= (const node &a) { memcpy(ch,a.ch,sizeof(ch)); len = a.len; pre = a.pre;
cnt = a.cnt; return *this; }
};
class SAM
{
    private:
        node a[MAXN<<1];
        int cnt,last;
    public:
        const int rt = 1;
        SAM() { init(); }
        void init(){ cnt = last = rt; a[rt].len = 0; a[rt].pre = 0; }
        void insert(int ch);
        void build();
        int dfs(int u);
}sam;
void SAM::insert(int ch)
{
    int p = last, now = last = ++cnt;
    a[now].len = a[p].len+1; a[now].cnt = 1;
    while(p && !a[p].ch[ch]) a[p].ch[ch] = now, p = a[p].pre;
    if(!p) { a[now].pre = rt; return; }
    int q = a[p].ch[ch];
    if(a[q].len == a[p].len+1) { a[now].pre = q; return; }
    int clone = ++cnt;
    a[clone] = a[q]; a[clone].cnt = 0;
    a[clone].len = a[p].len+1;
    while(p && a[p].ch[ch] == q) a[p].ch[ch] = clone, p = a[p].pre;
    a[q].pre = a[now].pre = clone;
}
void SAM::build()
{
    For(i,1,cnt)
        addEdge(a[i].pre,i);
}
int SAM::dfs(int u)
{
    ForE(i,u)
        a[u].cnt += dfs(e[i].to);
```

```
    if(a[u].cnt != 1) ans = max(ans, 1ll*a[u].len*a[u].cnt);
    return a[u].cnt;
}



ld a,b,c,d,L,R;
inline ld f(ld x) { return (c*x+d)/(a*x+b); }
inline ld simpson(ld l, ld r) { return (r-l)*(f(l)+f(r)+4*f((l+r)/2))/6; }
ld asr(ld l, ld r, ld epi, ld ans)
{
    ld mid = (l+r)/2, la = simpson(l,mid), ra = simpson(mid,r);
    if(abs(la+ra-ans) < 15.0*epi) return la+ra+(la+ra-ans)/15;
    return asr(l,mid,epi/2,la)+asr(mid,r,epi/2,ra);
}
```

SA

```
int n;
f64 calc(item cur)
{
    f64 x = cur.x, y = cur.y, res = 0;
    For(i,1,n)
    {
        auto [dx,dy,w] = item{x-a[i].x, y-a[i].y, a[i].w};
        res += sqrtl(dx*dx+dy*dy)*w;
    }
    return res;
}

void SimulateAnneal(item &ans)
{
    const f64 t0 = 1e5, tn = 1e-3, d = 0.973;
    const int cnt = 1e3;
    f64 t = t0, res = calc(ans), tmp;
    item cur = ans, nxt;
    while(t > tn)
    {
        nxt = { cur.x + t * (distrib(gen)*2-1),
                cur.y + t * (distrib(gen)*2-1), 0 };
        f64 delta = (tmp = calc(nxt)) - calc(cur);
        if(tmp < res) cur = ans = nxt, res = tmp;
        else if(exp(-delta / t) > distrib(gen)) cur = nxt;
        t *= d;
    }
    For(i,1,cnt)
    {
        nxt = { ans.x + t * (distrib(gen)*2-1),
                ans.y + t * (distrib(gen)*2-1), 0 };
        if((tmp=calc(nxt)) < res) ans = nxt, res = tmp;
    }
}
```

Slpay

```
template<typename T>
struct Node
{
    T val = T();
    Node<T> *pre = nullptr;
    int cnt = 0, size = 0;
    std::array<Node<T>*, 2> ch{};
    void pushup()
    {
        size = cnt;
        For(p,0,1) if(ch[p]) size += ch[p]->size;
    }
};
template<typename T>
struct Splay
{
    Node<T> *rt = nullptr;

    struct iterator
    {
        Node<T> *data;
        T operator*(){ return data->val; }
    };

    Splay() { insert(numeric_limits<T>::max()); insert(numeric_limits<T>::min()); }

    bool get(Node<T> *u) { return u->pre? u == u->pre->ch[1]: 0; }
    void rotModify(Node<T> *x, Node<T> *y, int chk)
    {
        if(x) x->pre = y;
        if(y) y->ch[chk] = x;
        else  rt = x;
    }
    void rotate(Node<T> *x)
    {
        auto *y = x->pre, *z = y->pre;
        int chk = get(x);
        rotModify(x,z,get(y));
        rotModify(x->ch[chk^1],y,chk);
        rotModify(y,x,chk^1);
        y->pushup(); x->pushup();
    }
    void splay(Node<T> *x, Node<T> *tar = nullptr)
    {
        if(!x) return;
        for(auto *y = x->pre; y = x->pre, y != tar; rotate(x))
            if(y->pre != tar)
                rotate(get(y)==get(x)? y: x);
        if(!tar) rt = x;
    }
    void insert(T k)
    {
        Node<T> *cur = rt, *pre = nullptr;
        while(cur && cur->val != k) pre = cur, cur = cur->ch[cur->val < k];
        if(cur) ++cur->cnt;
        else
        {
```

```
        cur = new Node<T>{k,pre,1,1};
        if(pre) pre->ch[pre->val < k] = cur;
    }
    splay(cur);
}
void erase(T k)
{
    auto *pre = prev(k).data, *nxt = next(k).data;
    splay(pre); splay(nxt,pre);
    auto *it = nxt->ch[0];
    delete it;
    nxt->ch[0] = nullptr;
}
void extract(T k)
{
    auto *pre = prev(k).data, *nxt = next(k).data;
    splay(pre); splay(nxt,pre);
    auto *it = nxt->ch[0];
    if(it -> cnt > 1) { --it->cnt; splay(it); return; }
    delete it;
    nxt->ch[0] = nullptr;
}
void find(T k)
{
    auto *cur = rt;
    if(!cur) return;
    while(cur->ch[cur->val < k] && k != cur->val)
        cur = cur->ch[cur->val < k];
    splay(cur);
}
iterator prev(T k)
{
    find(k);
    auto *cur = rt;
    if(cur->val < k) return {cur};
    cur = cur->ch[0];
    while(cur->ch[1]) cur = cur->ch[1];
    splay(cur);
    return {cur};
}
iterator next(T k)
{
    find(k);
    auto *cur = rt;
    if(cur->val > k) return {cur};
    cur = cur->ch[1];
    while(cur->ch[0]) cur = cur->ch[0];
    splay(cur);
    return {cur};
}
int rank(T k)
{
    find(k);
    return (rt->ch[0]? rt->ch[0]->size: 0) + (rt->val < k)  *rt->cnt;
}
T kth(int k)
{
    auto *cur = rt;
    while(1)
    {
        if(cur->ch[0] && k <= cur->ch[0]->size) cur = cur->ch[0];
        else
        {
            k -= (cur->ch[0]? cur->ch[0]->size: 0) + cur->cnt;
            if(k <= 0)
            { splay(cur); return cur->val; }
            cur = cur->ch[1];
        }
    }
}
};
```

Tarjan(无向图)

```
// tarjan for e-dcc
void tarjan(int u, int ed)
{
    dfn[u] = low[u] = ++t;
    for(int i = head[u]; i; i = e[i].nxt)
    {
        int v = e[i].to;
        if(!dfn[v])
        {
            tarjan(v,i);
            if(dfn[u] < low[v])
                b[i] = b[i^1] = 1;
            low[u] = min(low[u],low[v]);
        }
        else if(i^1 != ed) low[u] = min(low[u],dfn[v]);
    }
}
//dfs for e-dcc
void dfs(int u, int cc)
{
    dcc[u] = cc;
    ans[cc-1].push_back(u);
    for(int i = head[u]; i; i = e[i].nxt)
    {
        int v = e[i].to;
        if(dcc[v]||b[i]) continue;
        dfs(v,cc);
    }
}

// tarjan for v-dcc
int s[MAXN],top,rt;
void tarjan(int u)
{
    dfn[u] = low[u] = ++t;
    s[++top] = u;
    if(u==rt && !head[u]) { ans.push_back(vector<int>()); ans[res++].push_back(u);
return; }
```

```
    int c = 0,d;
    for(int i = head[u]; i; i = e[i].nxt) {
        int v = e[i].to;
        if(!dfn[v]) {
            tarjan(v);
            low[u] = min(low[u],low[v]);
            if(low[v] >= dfn[u]) {
                ++c;
                if(x^rt||c>1) b[i] = 1; ans.push_back(vector<int>());
                do { d = s[top--]; ans[res].push_back(d); } while(d != v);
                ans[res++].push_back(u);
            }
        }
        low[u] = min(low[u],dfn[v]);
    }
}
```

树哈希

```
const ull mask = chrono::steady_clock::now().time_since_epoch().count();
ull func(ull x)
{
    x ^= mask;
    x ^= x << 13;
    x ^= x >> 7;
    x ^= x << 17;
    x ^= mask;
    return x;
}

ull hsh[2][MAXN];
ull dfs(int u, int pre)
{
    hsh[0][u] = 1;
    ForE(i,u)
    {
        int v = e[i].to;
        if(v == pre) continue;
        hsh[0][u] += func(dfs(v,u));
    }
    return hsh[0][u];
}
void dp(int u, int pre)
{
    if(pre) hsh[1][u] = hsh[0][u] + func(hsh[1][pre]-func(hsh[0][u]));
    else hsh[1][u] = hsh[0][u];
    ForE(i,u)
    {
        int v = e[i].to;
        if(v == pre) continue;
        dp(v,u);
    }
}
int m,n,p[MAXN];
ull ans[MAXM];
void solve(int i)
{
    cin >> n;
    tot = 0; fill(head+1,head+1+n,0);
    For(i,1,n)
    {
        cin >> p[i];
        if(p[i]) add_edge(i,p[i]), add_edge(p[i],i);
    }
    dfs(1,0);
    dp(1,0);
    For(u,1,n) ans[i] = max(ans[i], hsh[1][u]);
}
```

Trie

```
struct Trie
{
    std::vector<int> cnt;
    std::vector<std::array<int, 2>> nxt;
    static constexpr int rt = 0;
    int tot = rt, hb;

    Trie(int n = 1, int b = 30) { init(n, b); };
    void init(int n = 1, int b = 30)
    {
        hb = b - 1;
        cnt.assign(b * n + 2, 0); nxt.assign(b * n + 2, {});
    }

    void insert(int x, int d = 1)
    {
        int u = rt; ++cnt[rt];
        foR(i, hb, 0)
        {
            bool p = x >> i & 1;
            if(!nxt[u][p]) nxt[u][p] = ++tot;
            u = nxt[u][p];
            cnt[u] += d;
        }
    }
    int query(int x)
    {
        if(!cnt[rt]) return 0;
        int u = rt, res(0);
        foR(i, hb, 0)
        {
            bool p = x >> i & 1;
            if(nxt[u][!p] && cnt[nxt[u][!p]]) u = nxt[u][!p], res ^= (1 << i);
            else u = nxt[u][p];
        }
        return res;
    }
};
```

线性基

```
bool add(_ll x)
{
    foR(i,63,0)
        if(x & (1ll << i))
        {
            if(!p[i]) { p[i] = x; return 1; }
            x ^= p[i];
        }
    return 0;
}
_ll query()
{
    _ll ans = 0ll;
    foR(i,63,0) ans = max(ans,ans ^ p[i]);
    return ans;
}
```

杜教筛

```
auto xudyhSmu = [](auto &&self, int n) -> int
{
    static std::unordered_map<int, int> mp;
    if(n <= N) return sm[n];
    if(mp.contains(n)) return mp[n];
    int res = 1;
    for(int l = 2, r; l <= n; l = r + 1)
    {
        r = n / (n / l);
        res -= self(self, n / l) * (r - l + 1);
    }
    return mp[n] = res;
};
auto xudyhSphi = [](auto &&self, int n) -> i64
{
    static std::unordered_map<int, i64> mp;
    if(n <= N) return sp[n];
    if(mp.contains(n)) return mp[n];
    i64 res = n * (n + 1ll) / 2;
    for(int l = 2, r; l <= n; l = r + 1)
    {
        r = n / (n / l);
        res -= self(self, n / l) * (r - l + 1);
    }
    return mp[n] = res;
};
```

```
cmake_minimum_required(VERSION 3.17)

project(MyProject)

set(CMAKE_CXX_STANDARD 20)
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -O2 -Wall -Wextra -Wl,--stack=536870912 -
finput-charset=UTF-8")
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/bin)

file (GLOB files *.cpp)
foreach (file ${files})
    string(REGEX REPLACE ".+/(.+)\\..*" "\\1" exe ${file})
    add_executable (${exe} ${file})
    message (\ \ \ \ --\ src/${exe}.cpp\ will\ be\ compiled\ to\ bin/${exe})
endforeach ()
```

```
{
    "configurations": [
        {
            "name": "Win32",
            "includePath": [
                "${default}"
            ],
            "defines": [
                "_DEBUG",
                "UNICODE",
                "_UNICODE"
            ],
            "compilerPath": "E:\\mingw64_posix\\mingw64\\bin\\g++.exe",
            "cStandard": "c11",
            "cppStandard": "c++20",
            "intelliSenseMode": "windows-gcc-x64"
        }
    ],
    "version": 4
}
```

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(gdb) Launch",
            "type": "cppdbg",
            "request": "launch",
            "program": "${fileDirname}/${fileBasenameNoExtension}.exe",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${workspaceFolder}",
            "environment": [],
            "externalConsole": true,
            "internalConsoleOptions": "neverOpen",
            "MIMode": "gdb",
            "miDebuggerPath": "E:/mingw64_posix/mingw64/bin/gdb.exe",
            "setupCommands": [
                {
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": false
                }
            ],
            "preLaunchTask": "Compile",
        }
    ]
}
```

```
{
    "tasks": [
        {
            "type": "process",
            "label": "Compile",
            "command": "E:\\mingw64_posix\\mingw64\\bin\\g++.exe",
            "args": [
                "-fdiagnostics-color=always",
                "${file}",
                "-o",
                "${fileDirname}\\${fileBasenameNoExtension}.exe",
                "-O2",
                "-g",
                "-m64",
                "-static-libgcc",
                "-std=c++20",
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": {
                "owner": "cpp",
                "fileLocation": [
                    "relative",
                    "${workspaceFolder}"
                ],
                "pattern": [
                    {
                        "regexp":
"^([^\\s].*)\\:(\\d+,\\d+)\\:\s*(.*)$",
                        "file": 1,
                        "location": 2,
                        "message": 3
                    }
                ]
            },
            "group": "build",
            "presentation": {
                "echo": true,
                "reveal": "always",
                "focus": false,
                "panel": "shared"
            },
            "detail": "调试器生成的任务。"
        }
    ],
    "version": "2.0.0"
}
```