P10778 BZOJ3569 DZY Loves Chinese II

题解：考虑先求出任意一棵生成树，然后对于所有非生成树上的边随机赋一个哈希权值，对于生成树上的边定义其边权是所有非树边所在路径覆盖该边的非树边的哈希异或值。这样构造可以保证对于任意一条树边，所有经过该树边的边的异或值为 0。

此时删除一个集合的边是合法的，当且仅当该集合不存在一个子集使得其边权的异或值是 0，线性基维护即可。

```cpp
// #pragma GCC optimize(3, "Ofast", "inline", "unroll-loops")
#include <bits/stdc++.h>
#define int long long
using namespace std;

const int N = 500010;
const int inf = 1e18;
const int mod = 1e9 + 7;

struct Edge
{
    int u, v;
} edge[N];
int xor_val[N], blk[N], dep[N], up[N], to[N], xor_up[N];
vector<int> adj[N], adj2[N];

class Basis
{
    int basis[64];
public:
    inline Basis() { memset(basis, 0, sizeof basis); }
    inline int insert(int x)
    {
        for (int i = 62; ~i; --i)
            if (x >> i & 1)
            {
                if (!basis[i])
                    return basis[i] = x, 1;
                x ^= basis[i];
            }
        return 0;
    }
} basis;

class DSU
{
```

```
public:
    int fa[N];
    inline DSU() { iota(fa, fa + N, 0); }
    inline int find(int x) { return x == fa[x] ? x : fa[x] =
find(fa[x]); }
    inline int merge(int x, int y)
    {
        int _x = x, _y = y;
        x = find(x), y = find(y);
        if (x != y)
            return fa[x] = y, 1;
        return 0;
    }
} dsu;

inline void dfs(int u, int fa)
{
    for (int &v : adj2[u])
        if (v != fa)
        {
            dfs(v, u);
            xor_up[u] ^= xor_up[v];
        }
    if (u != 1)
        xor_val[to[u]] = xor_up[u];
}

inline void dfs_dep(int u, int fa)
{
    dep[u] = dep[fa] + 1, up[u] = fa;
    for (int &v : adj2[u])
        if (v != fa)
            dfs_dep(v, u);
}

signed main()
{
    cin.tie(0)->sync_with_stdio(false);
    srand(time(0));
    int n, m;
    cin >> n >> m;
    for (int i = 0; i < m; ++i)
        cin >> edge[i].u >> edge[i].v;
    int q;
    cin >> q;
    mt19937_64 mt(time(0));
```

```
    uniform_int_distribution<int> dist(1, (1ll << 62) - 1);
    for (int i = 0; i < m; ++i)
    {
        auto [u, v] = edge[i];
        adj[u].emplace_back(v);
        adj[v].emplace_back(u);
    }
    for (int i = 0; i < m; ++i)
    {
        auto [u, v] = edge[i];
        if (dsu.merge(u, v))
        {
            adj2[u].emplace_back(v);
            adj2[v].emplace_back(u);
        }
    }
    dfs_dep(1, 0);
    for (int i = 0; i < N; ++i)
        dsu.fa[i] = i;
    for (int i = 0; i < m; ++i)
    {
        auto [u, v] = edge[i];
        if (dep[u] > dep[v])
            u ^= v ^= u ^= v;
        if (dsu.merge(u, v))
            e_id.emplace_back(i), blk[i] = 1, to[v] = i;
        else
            xor_val[i] = dist(mt), xor_up[u] ^= xor_val[i],
xor_up[v] ^= xor_val[i];
    }
    dfs(1, 0);
    int bef = 0;
    while (q--)
    {
        int k;
        cin >> k;
        basis = Basis();
        int ok = 1;
        while (k--)
        {
            int x;
            cin >> x;
            ok &= basis.insert(xor_val[(x ^ bef) - 1]);
        }
        bef += ok;
        cout << (ok ? "Connected" : "Disconnected") << '\n';
```

```
127     }
128     return 0;
129 }
130
```