J - NEO WINGS

题目描述

The truth is in your hand.
Save the world!

银河星际出现了危机!

银河星际由 n 颗星球以及星球之间的 m 条**单向**虫洞构成。第 i 条单向虫洞允许星际飞船从第 u_i 颗星球前往第 v_i 颗星球。

现在每条虫洞都被神秘组织劫持了。作为拯救银河星际的唯一人选，你需要规划一条路径，从任意点出发，经过每条虫洞**恰好一次**，以扫除劫持每条虫洞的神秘组织。

如果符合条件的路径存在，则输出你所规划的路径依次经过的星球编号。如果这样的路径不存在，输出 `mission impossible`。如果有多条符合条件的路径，输出星球编号**字典序最小**的路径。

欧拉路径：

通过图中所有边的简单路径

图G存在欧拉路径，当且仅当：

- 图G是连通的（弱连通的）
- 无向图：图G的奇点数为0或2，两个奇点其中一个为起点，一个为终点
- 有向图：可以存在2个点，入度不等于出度，其中一个入度比出度大1（起点），一个出度比入度大1（终点），或者不存在这样的点

有向图寻找欧拉路径：

寻找起点S，从S开始dfs，每访问一条边需要删掉这条边，当节点没有边可以访问时，将节点加入栈中，最后倒序输出栈内节点即可。

```
dfs(u):
    对于每条边u->v
        删除u->v
        dfs(v)
    u入栈
```

满足字典序最小：

起点不唯一时，需要在弱联通块中寻找编号最小的节点作为起点。

对每个顶点的出边进行排序，保证最先搜索编号小的节点。

代码：

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

using namespace std;

int t, n, m;
int u, v;
vector<vector<int>> edge;
vector<vector<int>> edgeU;
```

```cpp
int indeg[30004], outdeg[30004];
int vis[30004];

void dfs(int u) {
    vis[u] = 1;
    for (auto v : edgeU[u]) {
        if (vis[v] == 0) {
            dfs(v);
        }
    }
}

vector<int> hierholzer(int s) {
    vector<int> ans, idx(n + 1, 0), st;
    st.emplace_back(s);

    while (!st.empty()) {
        int u = st.back();
        if (idx[u] < edge[u].size()) {
            st.emplace_back(edge[u][idx[u]]);
            ++idx[u];
        } else {
            ans.emplace_back(u);
            st.pop_back();
        }
    }

    reverse(ans.begin(), ans.end());
    return ans;
}

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

    cin >> t;
    while (t--) {
        cin >> n >> m;

        edge.assign(n + 1, vector<int>());
        edgeU.assign(n + 1, vector<int>());

        for (int i = 1; i <= n; ++i) {
            indeg[i] = outdeg[i] = vis[i] = 0;
        }

        for (int i = 0; i < m; ++i) {
            cin >> u >> v;
            edge[u].emplace_back(v);
            edgeU[u].emplace_back(v);
            edgeU[v].emplace_back(u);
            ++indeg[v];
            ++outdeg[u];
        }

        for (int i = 1; i <= n; ++i) {
```

```
            if (indeg[i] || outdeg[i]) {
                dfs(i);
                break;
            }
        }

        int st = 0, ed = 0, tr = 0, num = 0;
        int stnode;
        for (int i = 1; i <= n; ++i) {
            sort(edge[i].begin(), edge[i].end());
            if (vis[i]) ++num;
            else continue;
            if (indeg[i] == outdeg[i]) ++tr;
            else if (outdeg[i] == indeg[i] + 1) ++st, stnode = i;
            else if (indeg[i] == outdeg[i] + 1) ++ed;
        }

        vector<int> ans;
        if (tr == num) {
            for (int i = 1; i <= n; ++i) {
                if (vis[i]) {
                    ans = hierholzer(i);
                    break;
                }
            }
        } else if (tr == num - 2 && st == 1 && ed == 1) {
            ans = hierholzer(stnode);
        }

        if (ans.size() != m + 1) {
            cout << "mission impossible" << endl;
        } else {
            for (int i = 0; i < ans.size(); ++i) {
                cout << ans[i] << " ";
            }
            cout << endl;
        }
    }

    return 0;
}
```