/\*\*

\* @param {number[][]} edges

\* @return {number[]}

\*/

var findRedundantConnection = function(edges) {

const n = edges.length;

const parent = new Array(n + 1); // 节点编号从1到n

// 初始化并查集，每个节点的父节点是自己

for (let i = 1; i <= n; i++) {

parent[i] = i;

}

// 查找根节点（带路径压缩）

const find = (x) => {

while (parent[x] !== x) {

parent[x] = parent[parent[x]]; // 路径压缩

x = parent[x];

}

return x;

};

// 遍历所有边

for (const [u, v] of edges) {

const rootU = find(u);

const rootV = find(v);

// 如果两个节点已经在同一个集合中，说明这条边会形成环

if (rootU === rootV) {

return [u, v];

}

// 否则合并两个集合

parent[rootU] = rootV;

}

return []; // 根据题意，这里不会执行

};