中山大学计算机学院本科生实验报告

（2021 学年第 1 学期）

课程名称：Data structures and algorithms　　任课教师：张子臻

年级	20 级	专业（方向）	软件工程
学号	20337270	姓名	钟海财
电话	13397996670	Email	2940599563@qq.com
开始日期	2021/12/16	完成日期	2021/12/23

1. 实验题目

1)　DAG?

2)　Ordering Tasks

2.实验目的

1)输入一个有向图，判断该图是否是有向无环图(Directed Acyclic Graph)。如果图是 DAG，输出 1，否则输出 0

2）输出 n 个任务的拓扑排序的结果。如果有多个解决方案，按顺序输出最小的解决方案。

3.程序设计

1)　DAG?

设计思路：

判断是否为有向无环图 DAG，可使用广度优先搜索 BFS。以每个节点为起点，分别进行广度优先搜索，如果该节点的某个新增恰为该节点时，说明该节

点为某个环上的点，即该有向图不是有向无环图 DAG；反之如果所有的节点都不在环上，则该有向图是有向无环图 DAG。

代码：

```
#include<iostream>
#include<queue>
#include <vector>
using namespace std;
vector<int> g[101];//邻接表
bool vis[101];//判断是否为遍历过的节点
bool bfs(int s){ //广度优先搜索
    queue<int> q;
    q.push(s);
     while (!q.empty()){
         int u = q.front();
         for(int i = 0; i < g[u].size(); ++i){
             if (!vis[g[u][i]]){
                 vis[g[u][i]] = 1;
                 q.push(g[u][i]);
                 if (s == g[u][i])//当某新增节点==起点时则成环
                     return false;
             }
         }
         q.pop();
     }
     return true;
}
int main(){
    int n, m, u, v;
    cin >> n >> m;
    for(int i = 0; i < m; ++i){//读入邻接表
        cin >> u >> v;
        g[u].push_back(v);
    }
    int ans = 1;
    for(int i = 1; i < n; ++i){//对每个节点进行 BFS
        if(!bfs(i)){//某个节点成环时令 ans=0,就不是 DAG
            ans = 0;
            break;
        }
    }
    cout << ans << endl;
    return 0;
}
```

2) Ordering Tasks

设计思路：

由于题目要求输出的是 n 个具有依赖关系的任务的排序结果，即为拓扑排序结果，所以使用拓扑排序：

1）从 DAG 图中选择一个 没有前驱（即入度为 0）的顶点并输出。

2）从图中删除该顶点和所有以它为起点的有向边。

3）重复 1 和 2 直到当前的 DAG 图为空或当前图中不存在无前驱的顶点为止。后一种情况说明有向图中必然存在环。

由于题目要求按顺序输出最小的解决方案，所以修改 1），删除最小的入读为 0 的顶点并输出，所以使用优先队列记录当前所有入度为 0 的顶点，方便找到最小的入度为 0 的顶点并删除。

代码：

```
#include <iostream>
#include <vector>
#include <queue>
using namespace std;
int main() {
    int case1, n, m; //测试用例数，点数，边数
    cin >> case1;
    while(case1--) {
        cin >> n >> m;
        int a, b;
        vector<int> result;//记录拓扑序列
        priority_queue<int, vector<int>, greater<int> > readyvis;
        //从小到大的优先队列，记录入度为 0 的节点
        vector<int> vis[n+1];//邻接表
        int indegree[n+1]={0};//入度表
        for(int i = 0; i < m; i++) {
            cin >> a >> b;
            indegree[b]++;
            vis[a].push_back(b);
        }
```

```
        for(int i = 1; i <= n; i++) {
            if(indegree[i] == 0) {
                readyvis.push(i);//记录入度为0的节点
            }
        }
        while(!readyvis.empty()) {//拓扑排序
            int temp = readyvis.top();
            readyvis.pop();//删去入度为0的节点
            result.push_back(temp);//删除的节点记入拓扑序列中
            for(int i=0;i<vis[temp].size(); i++) {
                indegree[vis[temp][i]]--;//删除该节点所有出去的边
                if(indegree[vis[temp][i]] == 0) { //入度为0的儿子节点进入优先队列
                    readyvis.push(vis[temp][i]);
                }
            }
        }
        for(int i = 0; i < result.size(); i++){//输出排序好的拓扑序列
            cout << result[i] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

4.程序运行与测试

1) DAG?

输入 1

```
3 3
1 2
2 3
3 1
```

输出 1（通过）

```
0
```

输入 2

```
4 6
1 2
2 3
3 1
1 3
1 4
4 2
```

输出 2（通过）

```
0
```

输入 3

```
4 5
1 2
2 3
1 3
1 4
4 2
```

输出 3（通过）

```
1
```

2) Ordering Tasks

输入 1

```
1
5 5
3 4
4 1
3 2
2 4
5 3
```

输出 1（通过）

```
5 3 2 4 1
```

输入 2

```
1
7 5
1 2
2 4
3 4
6 1
5 6
```

输出 2（通过）

```
3 5 6 1 2 4 7
```

输入 3

```
1
10 8
2 4
3 4
6 1
5 6
7 1
8 3
10 2
9 8
```

输出 3（通过）

```
5 6 7 1 9 8 3 10 2 4
```

5.实验总结与心得

除了使用广度优先搜索 BFS 判断 DAG，还可以使用拓扑排序来判断，根据拓扑排序的特点：

1）从 DAG 图中选择一个 没有前驱（即入度为 0）的顶点并输出。

2）从图中删除该顶点和所有以它为起点的有向边。

3）重复 1 和 2 直到当前的 DAG 图为空或当前图中不存在无前驱的顶点为

止。后一种情况说明有向图中必然存在环。

如果当所有入度为 0 的顶点都从图中删去，图中还剩下顶点（入度不为 0 的），便说明了该图中存在环，不是 DAG.

附录、提交文件清单