问题概述

拓扑排序是一种一般在有向无环图(DAG)上实现的算法，其主要目的是把这一有向无环图的顶点排列成一个线性的序列，并且对于图中的任意一条单向边$<u,v>$□点u在点v之前。

算法实现

拓扑排序的流程如下。

我们准备一个空的队列Q□首先我们把所有入度为0的点放入队列，然后我们一次处理队列中的点。每处理一个点，我们把这个点能够到达的所有点的入度减少一，如果经过这一处理后又有点的入度变为0，那么我们就把它放入队列。这样一来，我们从队列中取出点的序列就是问题概述中所要求得的序列。

正确性分析

对于正确定，我们发现由于拓扑排序在DAG上运行，所以一定不会有环，这也就保证了一定能够得到一个答案。而算法的操作保证了如果图中存在边$<u,v>$□点u一定比点v先进入队列，又因为队列先进先出的特性，我们可以知道算法得到的答案一定是正确的。

延申

实际上，拓扑排序不一定要在DAG上运行，在有环的有向图中，拓扑排序也可以正常运行到结束。只不过，在这种情况下，当算法结束时，图中的环以及从环上出来的链无法被处理。当然，在特定情况下，这样的结果反而是一种优势。

复杂度分析及算法应用

很容易发现，第一步将所有点遍历找出入度为0的点的复杂度是$O(n)$□随后的操作中，每个点最多入队一次，出对一次，所以复杂度也是$O(n)$□因此拓扑排序的总时间复杂度就是$O(n)$□

拓扑排序的一个显然的应用就是对有约束的一系列事件进行排序，而当情况变得更复杂的时候，比如图中出现环等情况，也可以在拓扑排序的基础上加以调整，使算法适合特殊的情况。

例题

Uva10305——Ordering Tasks

题目大意

经典拓扑排序题目。

解题思路

按照标准实现方法即可。

代码实现

```
#include <cstring>
#include <deque>
#include <iostream>

using namespace std;

bool go[105][105], flag[105];
int went[105], a, b;
int a1, b1;
deque<int> he;

int main()
{
    int i;

    while(true)
    {
        cin>>a>>b;
        if(a == 0 && b == 0)
            break;
        memset(go, 0, sizeof(go));
        memset(went, 0, sizeof(went));
        memset(flag, 0, sizeof(flag));
        for(i = 0;i < b;i++)
        {
            cin>>a1>>b1;
            go[a1][b1] = 1;
            went[b1]++;
        }
        for(i = 1;i <= a;i++)
            flag[i] = 1;
        int sum = a;
        while(sum)
        {
            for(i = 1;i <= a;i++)
            {
                if(went[i] == 0 && flag[i] == 1)
                {
                    he.push_back(i);
                    flag[i] = 0;
                }
            }
```

```
            while(!he.empty())
            {
                int c = he.front();
                for(i = 1;i <= a;i++)
                {
                    if(go[c][i] == 1)
                        went[i]--;
                }
                cout<<he.front()<<" ";
                he.pop_front();
                sum--;
            }
        }
        cout<<endl;
    }
    return 0;
}
```

by MisakaTao 2020.5.6