拓扑排序

TopSort

引例：选课(NKOJ1109)

在CQNK中学，学生们可以自由选择他喜欢的课程，但有的课程必须学完一些其它课程后才能选择。比如要选“编程”课就必须先学“离散数学”和“何老板思想”。有个同学有n门喜欢的课程，下面给出这n门课程间的关系(课程的编号为1到n)，问要学完这n门课程，需要安排一个怎样的学习顺序才是合理的。

输入格式：

第一行 两个整数n(n<=200)和m(m<=500)

接下来m行，每行两个整数x和y，表示要学y课程，必须先学x课程。

输出格式：

一行，n个空格间隔的整数，从左到右表示学完这n门课程的先后顺序。(如果有多种方案，输出字典序最小的方案)

如果无法找到一个合理的学习顺序，输出"impossible"

样例输入：

4 4
1 2
2 4
1 3
3 4

样例输出：

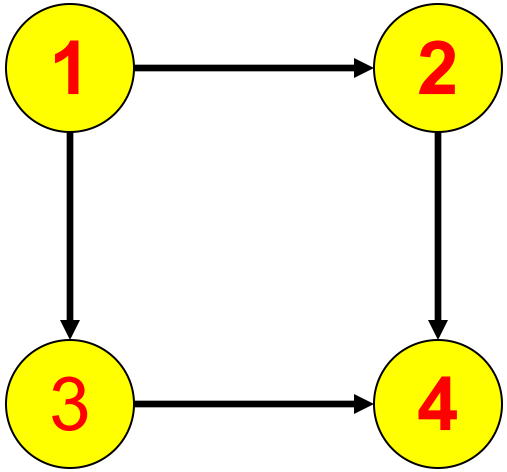
1 2 3 4

设G是具有n个顶点的有向图

S:$V_1,V_2,V_3,..V_n$是G的n个顶点构成的序列

若这个序列满足条件：V_x,V_y是序列S中的任意两个点，若< V_x,V_y >是图G中的一条边，顶点V_x在S中的位置一定在顶点V_y之前。这样的序列称为拓扑序列。

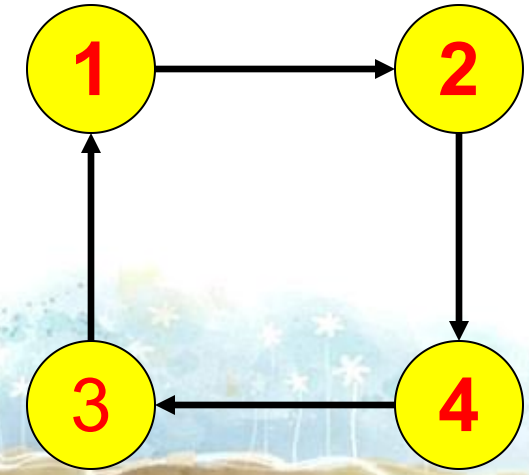
例如：



拓扑序列

V_1,V_2,V_3,V_4

V_1,V_3,V_2,V_4



注意：包含回路的有向图不能构成拓扑序列

将有向图中的顶点排列成一个拓扑序列的过程，称为拓扑排序。

对于给定的有向图，可按以下步骤进行拓扑排序：

① 任选一个入度为0的顶点X;

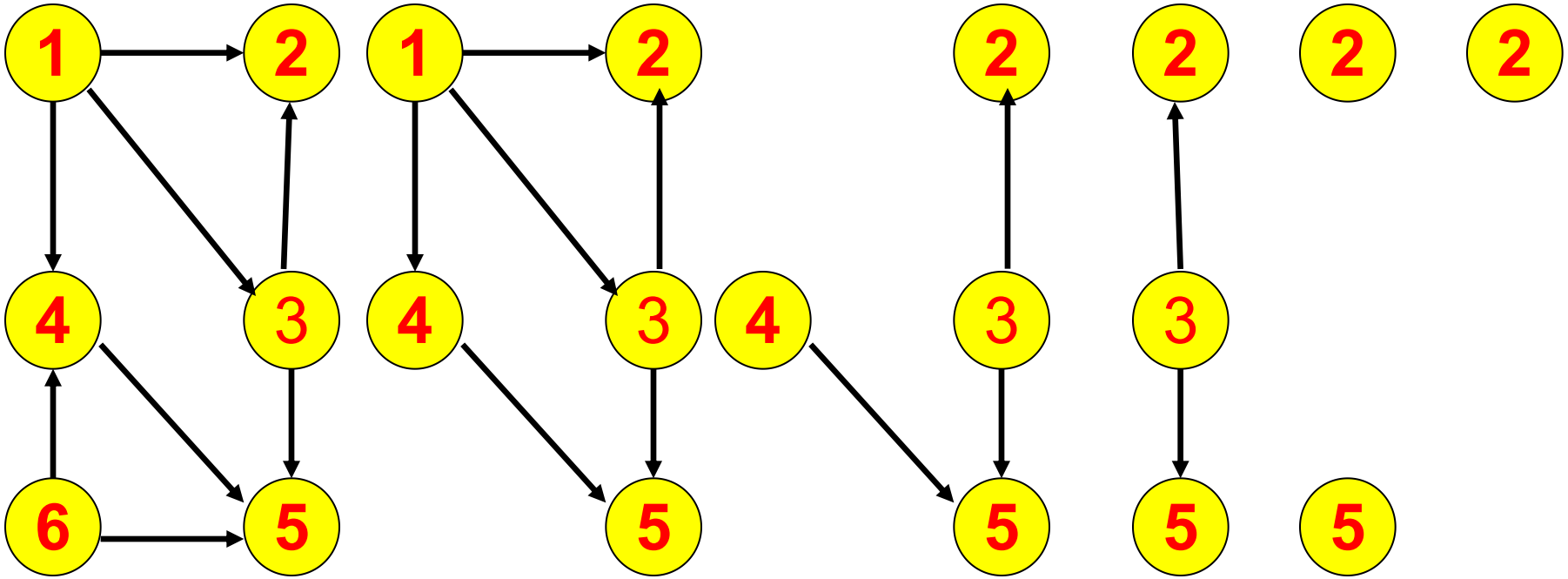
② 输出并删除X；

③ 对于所有邻接于X的点，将它们的入度减1；

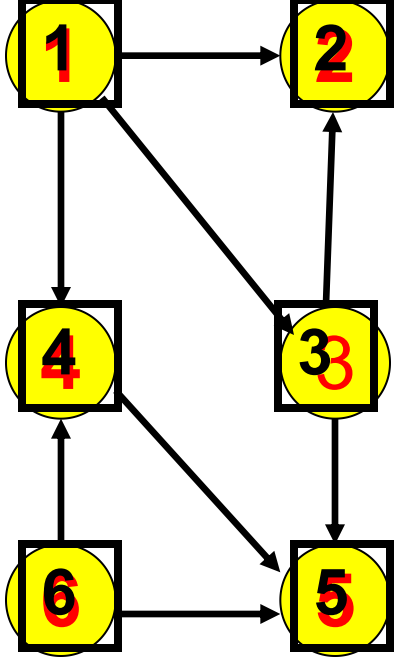
重复执行上诉步骤，直到所有顶点都被输出为止。

注意：如果在执行过程中找不到入度为0的点，但图中还有顶点未被输出，则说明图中存在回路，不能进行拓扑排序。

拓扑排序的例子



6 1 4 3 5 2


1
2
4
3
6
5

拓扑排序程序实现

```
for(i=1;i<=n;i++)
如果顶点i的入度为0，则把该点放入栈s中；

while (栈s不为空 )
{
    取出栈顶元素t，将其输出；  栈顶指针减1；
    for(j=1;j<=n;j++)
    {
        如果顶点j与t邻接，则把j的入度减1；
        如果j的入度为0,栈顶指针加1,把j放入栈s中；  }
}
```

```
//输入
for(i=1;i<=m;i++)
{
    cin>>x>>y>>z;
    Map[x][y]=z;
    d[y]++;  //d[y]记录y点的入度
}

//初始化队列，把入度为0的点入栈
queue<int>Q;
for(i=1;i<=n;i++)
  if(d[i]==0)
  {
      Q.push(i);
  }
```

```
//拓扑排序
while(Q.size())
{
     t=Q.front();
     Q.pop();
     cout<<t;
for(i=1;i<=n;i++)
        if(Map[t][i]!=inf)
        {
              d[i]--;
              if(d[i]==0)
              {
                    Q.push(i);
              }
        }
}
```

没有删边语句
map[t][i]=inf
？？？

引例：选课(NKOJ1109)

在CQNK中学，学生们可以自由选择他喜欢的课程，但有的课程必须学完一些其它课程后才能选择。比如要选“编程”课就必须先学“离散数学”和“何老板思想”。有个同学有n门喜欢的课程，下面给出这n门课程间的关系(课程的编号为1到n)，问要学完这n门课程，需要安排一个怎样的学习顺序才是合理的。

输入格式：

第一行 两个整数n(n<=200)和m(m<=500)

接下来m行，每行两个整数x和y，表示要学y课程，必须先学x课程。

输出格式：

一行，n个空格间隔的整数，从左到右表示学完这n门课程的先后顺序。(如果有多种方案，输出字典序最小的方案)

如果无法找到一个合理的学习顺序，输出"impossible"

样例输入：

4 4
1 2
2 4
1 3
3 4

样例输出：

1 2 3 4

怎样保证字典序最小?

```
int du[300],ans[300];              //记录入度和结果
bool mark,Map[300][300];     //Map存储地图

 //初始化，读入数据
scanf("%d%d",&n,&m);
 for(i=1;i<=m;i++)
 {
  scanf("%d%d",&x,&y);
  Map[x][y]=true;
  du[y]++;
 }
```

```
// “拓扑排序”
for(i=1;i<=n;i++)          //总共要删除n个点
{
     mark=false;           //用于标记是否找到了入度为0的点
     for(j=1;j<=n;j++)     //每次选出入度为0的编号最小的点。
     if(du[j]==0)
     {
          mark=true;
          ans[i]=j;          //记录结果
          du[j]--;           //du[j]==-1,以后就不会再被讨论了
          break;
     }
    if(!mark)
    {
          printf("impossible\n");
          return 0;
    }
    for(k=1;k<=n;k++)     //把与j相邻的点，入度减一
        if(Map[j][k])du[k]--;
}
```

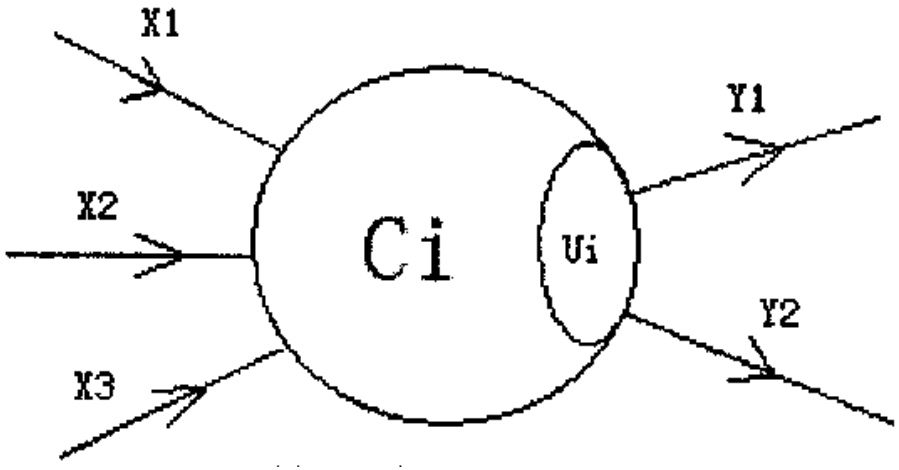
//此题数据规模小，可按上述方式操作。正确方式应该使用小根堆。

拓扑排序习题：

1109,1110，1506

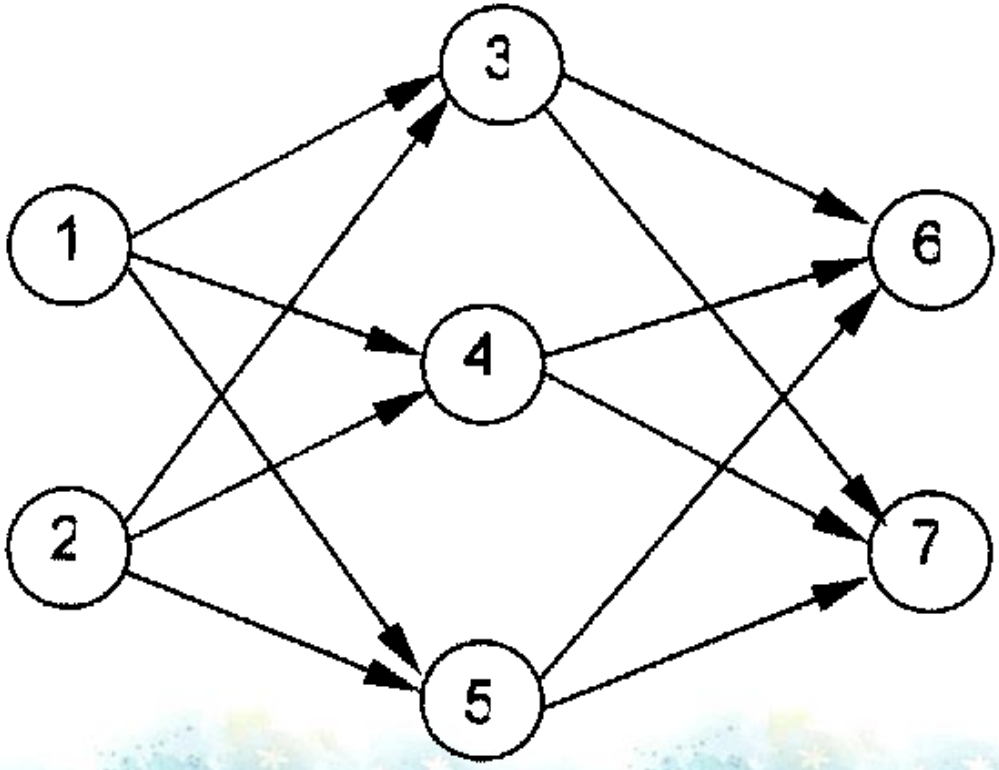
神经网络(NOIP2003 NKOJ1110)

神经网络就是一张有向图，图中的节点称为神经元，而且两个神经元之间至多有一条边相连，下图是一个神经元的例子：



图中，X1—X3是信息输入渠道，Y1－Y2是信息输出渠道，C1表示神经元目前的状态，Ui是阈值，可视为神经元的一个内在参数。

神经元按一定的顺序排列，构成整个神经网络。神经网络中的神经分为几层；称为输入层、输出层，和若干个中间层。每层神经元只向下一层的神经元输出信息，只从上一层神经元接受信息。下图是一个简单的三层神经网络的例子。



规定，C_i服从公式：（其中E是网络中所有神经元的数目）

$$C_i = \sum_{(j,i)\in E} W_{ji} C_j - U_i$$

公式中的W_{ji}（可能为负值）表示连接j号神经元和 i号神经元的边的权值。**当 C_i大于0时**，该神经元处于兴奋状态，否则就处于平静状态。当神经元处于兴奋状态时，下一秒它会向其他神经元传送信号，信号的强度为C_i。如此．在输入层神经元被激发之后，整个网络系统就在信息传输的推动下进行运作。现在，给定一个神经网络，及当前输入层神经元的状态（C_i），要求你的程序运算出最后网络输出层的状态。

∑ 求累加之和

例如 有数组 int a[6]={0,2,4,6,8,10};

$\sum_{i\in n}$a[i]=a[1]+a[2]+a[3]+a[4]+a[5]

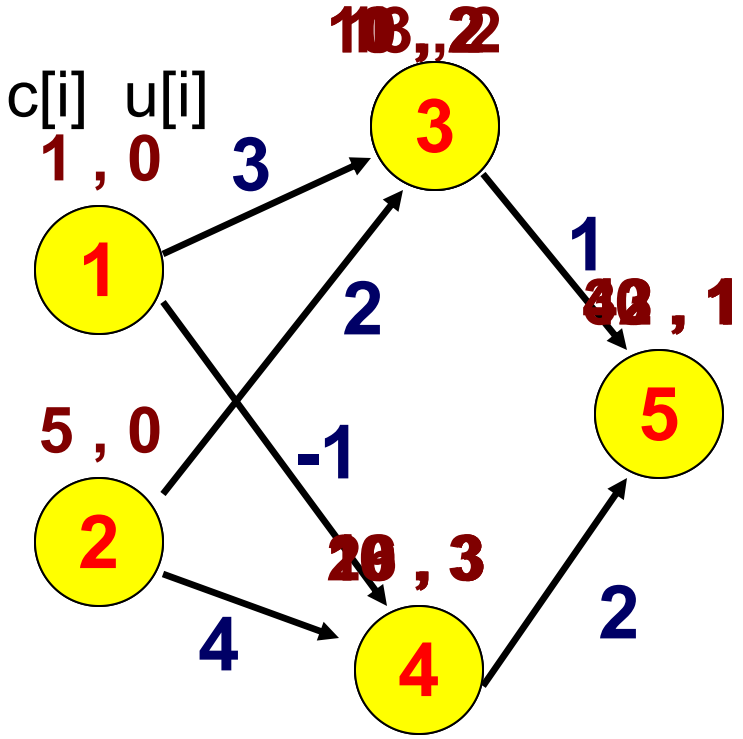
- 【输入格式】
- 输入文件第一行是两个整数n（1≤n≤20）和p。接下来n行，每行两个整数，第i＋1行是神经元i最初状态和其阈值（Ui），非输入层的神经元开始时状态必然为0。再下面P行，每行由两个整数i，j及一个整数Wij，表示连接神经元i、j的边权值为Wij。
- 【输出格式】
- 输出文件包含若干行，每行有两个整数，分别对应一个神经元的编号，及其最后的状态，两个整数间以空格分隔。仅输出最后状态非零的输出层神经元状态，并且按照编号由小到大顺序输出！
- 若输出层的神经元最后状态均为 0，则输出 NULL。

【输入样例】

```
5 6
1 0
1 0
0 1
0 1
0 1
1 3 1
1 4 1
1 5 1
2 3 1
2 4 1
2 5 1
```

【输出样例】

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



$$C_i = \sum_{(j,i)\in E} W_{ji} C_j - U_i$$

c[3] = c[1]*w[1][3]+c[2]*w[2][3]-u[3]

= 1*3 + 5*2 - 2

= 11

每次选一个入度为0且未检查过的点x，将从x出发的所有边[x][y]扩展，所谓扩展就是c[y]=c[y]+c[x]*w[x][y]，然后将点y的入度减1，如果[x][y]是最后一条通往y的边（此时点y的入度为0），则c[y]=c[y]-u[y]。

最后输出出度为0且C值大于0的点。

$$C_i = \sum_{(j,i)\in E} W_{ji} C_j - U_i$$

解法：用托普排序模拟

先记录下每个点的入度，每次选一个入度为0且未检查过的点i，将从i出发的所有边[i][j]扩展，所谓扩展就是c[j]=c[j]+c[i]*w[i][j]，然后将点j的入度减1，如果[i,j]是最后一条通往j的边（点j的入度为0），则c[j]=c[j]-u[j]。输出出度为0且C值大于0的点。

应注意的地方：

1.一旦点i的C[i]值小于等于0 则它不输出信号

2.连接两个节点的边的权值可能是负数也可能是0