拓扑排序

一个无环的有向图被称为**有向无环图(Directed Acycline Graph,DAG)**。有向无环图是描述一个工程、计划、生产、系统等流程的有效工具。一个大工程可分为若干子工程(活动),活动之间通常有一定的约束,例如先做什么活动,在什么活动完成后才可以开始下一个活动。

用节点表示活动,用弧表示活动之间的优先关系的有向图,被称为 AOV 网(Activity On Vertex Network)。

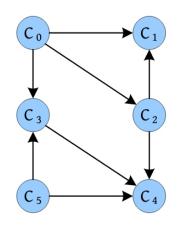
在 AOV 网中,若从节点 i 到节点 j 存在一条有向路径,则称节点 i 是节点 j 的前驱,或者称节点 j 是节点 i 的后继。

若<i,j>是图中的弧,则称节点i是节点j的直接前驱,节点j是节点i的直接后继。

AOV 网中的弧表示了活动之间存在的制约关系。例如,计算机专业的学生必须完成一系列规定的基础课和专业课才能毕业。学生按照怎样的顺序来学习这些课程呢?这个问题可以被看成一个大的工程,其活动就是学习每一门课程。课程的名称与相应编号如下表所示。

课程编号	课程名称	先修课程		
C _o	程序设计基础	无		
C 1	数据结构	C ₀ , C ₂		
C 2	离散数学	C 0		
C ₃	高级程序设计	C_0 , C_5		
C 4	数值分析	C ₂ , C ₃ , C ₅		
C 5	高等数学	无		

如果用节点表示课程,用弧表示先修关系,若课程 i 是课程 j 的先修课程,则用弧 < i,j>表示,课程之间的关系如下图所示。

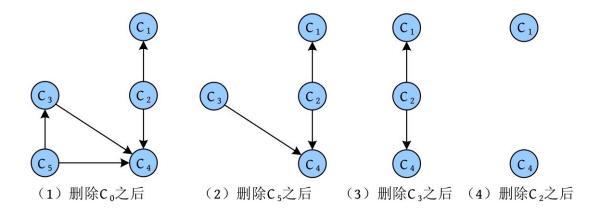


在 **AOV 网中是不允许有环**的,否则会出现自己是自己的前驱的情况,陷入死循环。**怎么判断在 AOV 网中是否有环呢?**一种检测的办法是对有向图中的节点进行拓扑排序。**如果 AOV 网中的所有节点都在拓扑序列中,则在 AOV 网中必定无环**。

拓扑排序指将 AOV 网中的节点排成一个线性序列,该序列必须满足:若从节点 i 到节点 j 有一条路径,则在该序列中节点 i 一定在节点 j 之前。

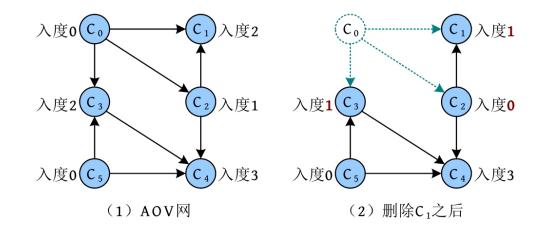
拓扑排序的基本思想:①选择一个无前驱的节点并输出;②从图中删除该节点和该节点的所有发出边;③重复步骤 1、2,直到不存在无前驱的节点;④如果输出的节点数少于 AOV 网中的节点数,则说明网中有环,否则输出的序列即拓扑序列。

拓扑排序并不是唯一的,例如在上图中,节点 C_0 和 C_5 都无前驱,先输出哪一个都可以,如果先输出 C_0 ,则删除 C_0 及 C_0 的所有发出边。此时 C_2 和 C_5 都无前驱,如果输出 C_5 ,则删除 C_5 及 C_5 的所有发出边。此时 C_2 和 C_5 都无前驱,如果输出 C_5 ,则删除 C_5 及 C_5 的所有发出边。此时 C_5 和 C_6 和 C



拓扑序列为 C_0 、 C_5 、 C_3 、 C_2 、 C_1 、 C_4 。

在上述描述过程中有删除节点和边的操作,实际上,**没必要真的删除节点和边。可以将没有前驱的节点(入度为 0)暂存到栈中,输出时出栈 即表示删除。进行边的删除时将其邻接点的入度减 1 即可。**例如在下图中删除 C_0 的所有发出边,相当于将 C_3 、 C_2 、 C_1 节点的入度减 1。

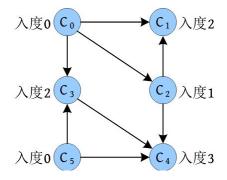


1. 算法步骤

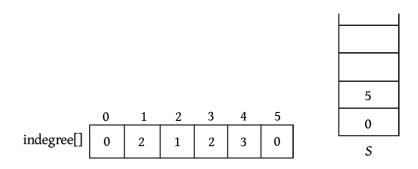
- (1) 求各节点的入度,将其存入数组 indegree[]中,并将入度为0的节点入栈S。
- (2)如果栈不空,则重复执行以下操作:①将栈顶元素 i 出栈并保存到拓扑序列数组 topo[]中;②将节点 i 的所有邻接点入度都减 1,如果减 1 后入度为 0,则立即入栈 S。
 - (3) 如果输出的节点数少于 AOV 网中的节点数,则说明网中有环,否则输出拓扑序列。

2. 图解

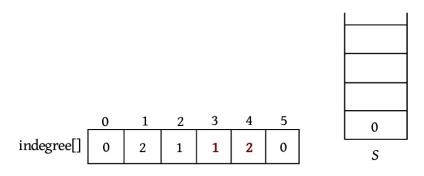
例如,一个AOV网如下图所示,其拓扑排序的过程如下。



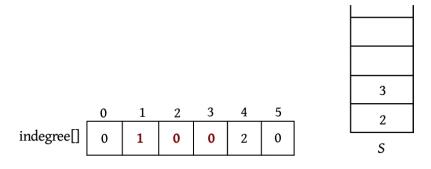
(1)输入边时累加节点的入度并保存到数组 indegree[]中,将入度为 0 的节点入栈 S。



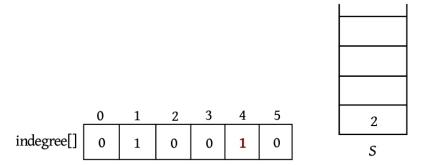
(2)将栈顶元素 5 出栈并保存到拓扑序列数组 topo[]中。将节点 5 的所有邻接点(C₃、C₄)入度都减 1,如果减 1 后入度为 0,则立即入栈 S。



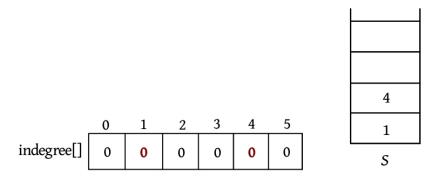
(3)将栈顶元素 0 出栈并保存到拓扑序列数组 topo[]中。将节点 0 的所有邻接点(C_1 、 C_2 、 C_3)入度都减 1,如果减 1 后入度为 0,则立即入栈 S。



(4)将栈顶元素 3 出栈并保存到拓扑序列数组 topo[]中。将节点 3 的邻接点 C4入度减 1,如果减 1 后入度为 0,则立即入栈 S。



(5)将栈顶元素 2 出栈并保存到拓扑序列数组 topo[]中。将节点 2 的所有邻接点(C_1 、 C_4)入度减 1,如果减 1 后入度为 0,则立即入栈 S_0 节点 1 没有邻接点,什么也不做。



- (6)将栈顶元素 4 出栈并保存到拓扑序列数组 topo[]中。节点 4 没有邻接点。
- (7)将栈顶元素1出栈并保存到拓扑序列数组topo[]中。节点1没有邻接点。
- (8) 栈空,算法停止。输出拓扑排序序列。

	0	1	2	3	4	5
topo[]	5	0	3	2	4	1

3. 算法实现

4. 算法分析

时间复杂度: 度数为 0 的节点入栈的时间复杂度为 O(n), 在每个节点出栈后都需要将其邻接点入度减 1, 如果使用邻接矩阵存储,则每次访问邻接点的时间复杂度都为 O(n),总的时间复杂度为 O(n²)。采用邻接表或链式前向星存储访问一个节点的所有邻接点,访问次数为该节点的度,总的时间复杂度为 O(e)。

空间复杂度:辅助空间包括入度数组 indegree[]、拓扑序列数组 topo[]、栈 S , 算法的空间复杂度是 O(n)。