8.7 拓扑排序

1、什么是拓扑排序

设G=(V，E)是一个具有n个顶点的有向图，V中顶点序列v_1，v_2，…，v_n称为一个拓扑序列，当且仅当该顶点序列满足下列条件：

若<i，j>是图中的边（或从顶点i⇨j有一条路径）：

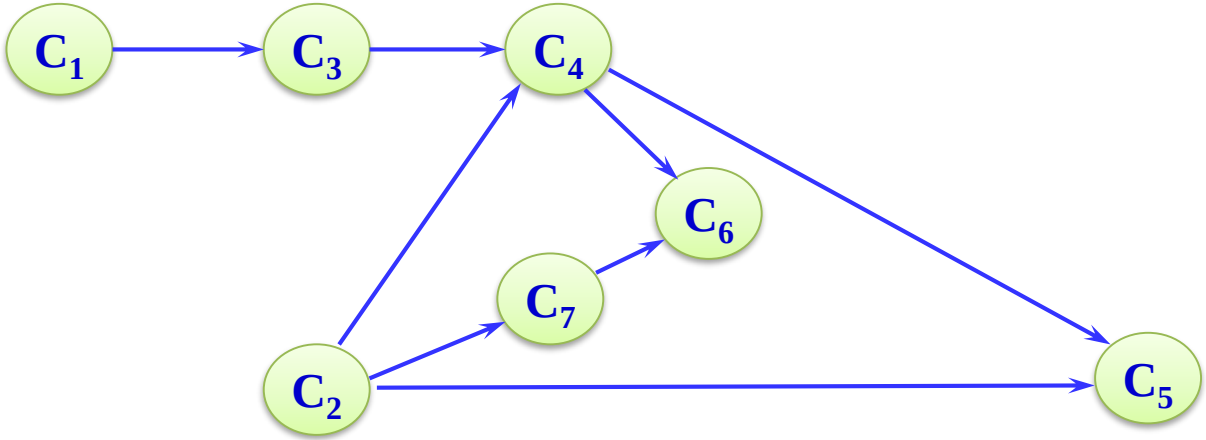
则在拓扑序列中顶点i必须排在顶点j之前。

在一个有向图中找一个拓扑序列的过程称为拓扑排序。

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，假设这些课程的名称与相应代号有如下关系：

课程代号	课程名称	先修课程
C_1	高等数学	无
C_2	程序设计	无
C_3	离散数学	C_1
C_4	数据结构	C_2，C_3
C_5	编译原理	C_2，C_4
C_6	操作系统	C_4，C_7
C_7	计算机组成原理	C_2

课程之间的先后关系可用有向图表示：



可以这样排课：

- $C_1 - C_3 - C_2 - C_4 - C_7 - C_6 - C_5$
- $C_2 - C_7 - C_1 - C_3 - C_4 - C_5 - C_6$

第1学期　　第2学期

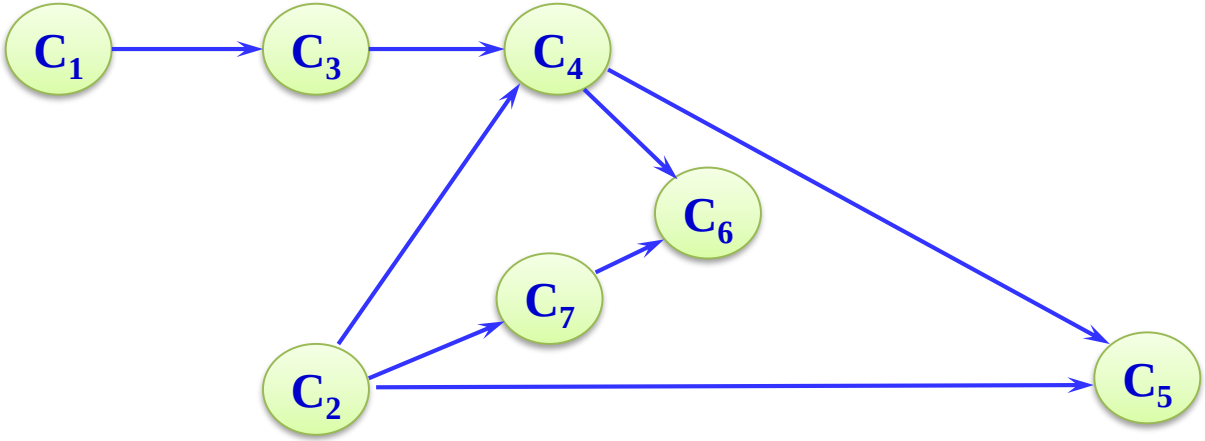
2、拓扑排序步骤

（1）从有向图中选择一个没有前驱（即入度为0）的顶点并且输出它。

（2）从图中删去该顶点，并且删去从该顶点发出的全部有向边。

（3）重复上述两步，直到剩余的图中不再存在没有前驱的顶点为止。

拓扑排序演示



产生一个拓扑序列：

C_1 C_3 C_2 C_7 C_4 C_6 C_5

排序完成

3、拓扑排序算法设计

将邻接表定义中的VNode类型修改如下：

```
typedef struct                  //表头节点类型
{    Vertex data;               //顶点信息
     int count;                 //存放顶点入度
    ArcNode *firstarc;          //指向第一条边
} VNode;
```

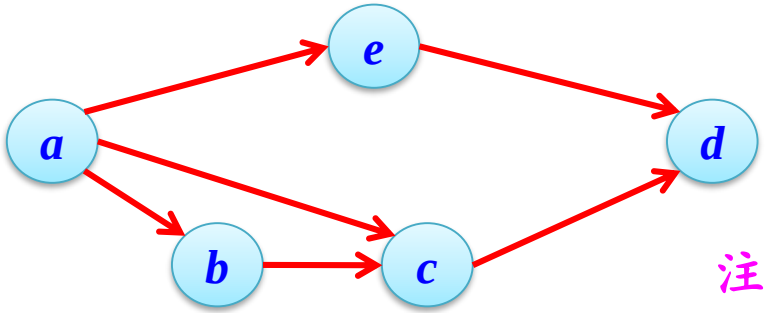
用于查找入度为0的顶点

拓扑排序算法如下：　　修改后的含*n*个顶点的邻接表

```
void  TopSort(VNode adj[],  int n)
{     int i,  j;int St[MAXV],  top=-1;  //栈St的指针为top
      ArcNode *p;
      for (i=0;i<n;i++)
          if (adj[i].count==0)                  //入度为0的顶点进栈
          {    top++; St[top]=i;    }
     while (top>-1)                             //栈不为空时循环
     {    i=St[top];top--;                      //出栈
          printf("%d ",  i);
          p=adj[i].firstarc;
          while (p!=NULL)
          {     j=p->adjvex; adj[j].count--;
                if (adj[j].count==0)
                {     top++;  St[top]=j;   }
                p=p->nextarc;                   //找下一个相邻顶点
          }
     }
}
```

【例8-8】对如图所示的图进行拓扑排序，可以得到不同的拓扑序列个数是_____。

A. 4　　B. 3　　C. 2　　D. 1



注：2010年全国考研题

不同的拓扑序列有：*aebcd*、*abced*、*abecd*。答案为B。

思考题：

❶ 一个有向图中存在回路，能否进行拓扑排序？

❷ 一个有向图能够产生所有顶点的拓扑序列，该图一定是有向无环图吗？

——本讲完——