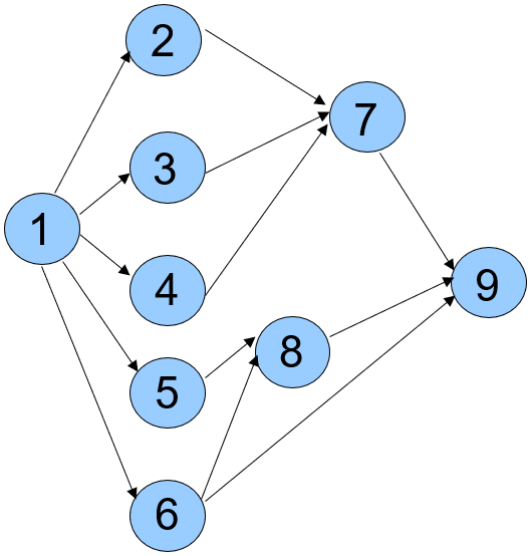
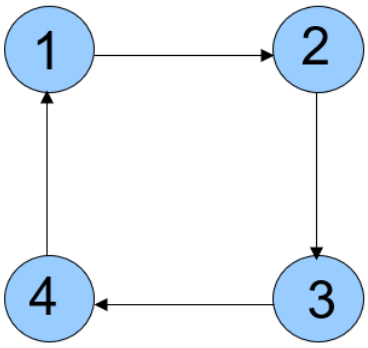
AOV网

在日常生活中，一项大的工程可以看作是由若干个子工程（这些子工程称为`活动`）组成的集合，这些子工程（活动）之间必定存在一些先后关系，即某些子工程（活动）必须在其它一些子工程（活动）完成之后才能开始，我们可以用有向图来形象地表示这些子工程（活动）之间的先后关系，子工程（活动）为顶点，子工程（活动）之间的先后关系为有向边，这种有向图称为“顶点活动网络”，又称`AOV网`（Activity On Vertex Network）。



在AOV网中，有向边代表子工程（活动）的先后关系，我们把一条有向边起点的活动成为终点活动的前驱活动，同理终点的活动称为起点活动的后继活动。而只有当一个活动全部的前驱全部都完成之后，这个活动才能进行。例如在上图中，只有当工程1完成之后，工程2、3、4、5、6才能开始进行。只有当2、3、4全部完成之后，7才能开始进行。

一个AOV网必定是一个`有向无环图`，即不应该带有回路。否则，会出现先后关系的自相矛盾。



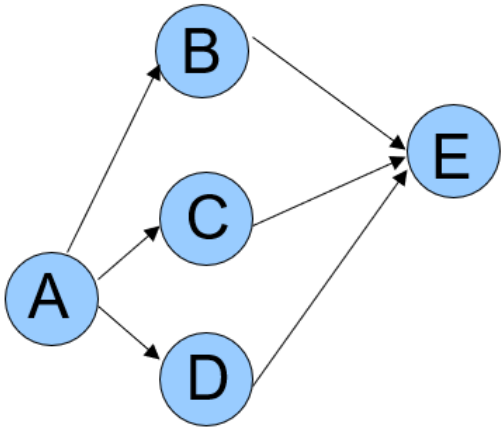
上图就是一个出现环产生自相矛盾的情况。4是1的前驱，想完成1，必须先完成4。3是4的前驱，而2是3的前驱，1又是2的前驱。最后造成想完成1，必须先完成1本身，这显然出现了矛盾。

拓扑排序算法

拓扑排序算法，只适用于AOV网（有向无环图）。

把AOV网中的所有活动排成一个序列， 使得每个活动的所有前驱活动都排在该活动的前面，这个过程称为“拓扑排序”，所得到的活动序列称为“拓扑序列”。

一个AOV网的`拓扑序列是不唯一的`，例如下面的这张图，它的拓扑序列可以是：ABCDE，也可以是ACBDE，或是ADBCE。在下图所示的AOV网中，工程B和工程C显然可以同时进行，先后无所谓；但工程E却要等工程B、C、D都完成以后才能进行。



构造拓扑序列可以帮助我们合理安排一个工程的进度，所以AOV网构造拓扑序列具有很高的实际应用价值。

算法思想

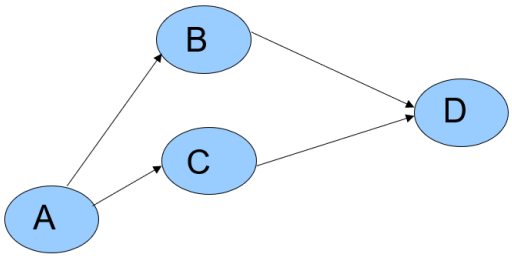
> 这个程序使用队列来找出入度为0的点

```
que ← 所有入度为0的点入队

while(!que.empty()){

    获取队首元素 t ← 队首

    枚举t的所有出边 t → j
        删掉t → j，即du[j]--
        if(du[j] == 0) j入队
}
```

```
void toposort(){

    int hh = 0, tt = -1;

    for(int i = 1; i <= n; i++)
        if(!du[i]) q[++tt] = i;

    while(hh <= tt){
        int t = q[hh++];
        cout << t << " ";
        for(int i = h[t]; ~i; i = ne[i]){
            int j = to[i];
            if(--du[j] == 0) q[++tt] = j;
        }
    }
```

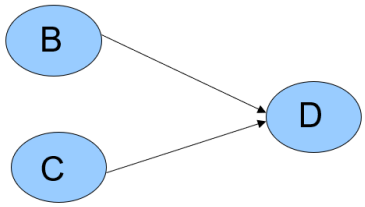
```
16 }
```

算法图解



开始时，只有A入度为0，A入队。

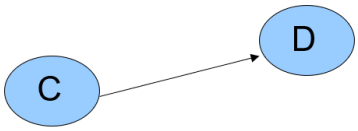
队列：A



队首元素A出队，枚举A的后继B、C入度减1，相当于删除A的所有关联边，B、C入队

拓扑序列：A B C

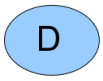
队列：B C



队首元素B出队，B的后继D入度减1

拓扑序列：A B C

队列：C



队首元素C出队，C的后继D入度再减1。这时D入度为0，入队。

拓扑序列：A B C D

队列：D

算法时间复杂度:

如果AOV网络有n个顶点，e条边，在拓扑排序的过程中，搜索入度为零的顶点所需的时间是O(n)。每个顶点入度减1的运算共执行了e次。所以总的时间复杂为`O(n+e)`。

快乐刷题

- P138 家谱树
- P350 奖金