

06F1 拓扑排序-零入度排序

#数据结构邓神

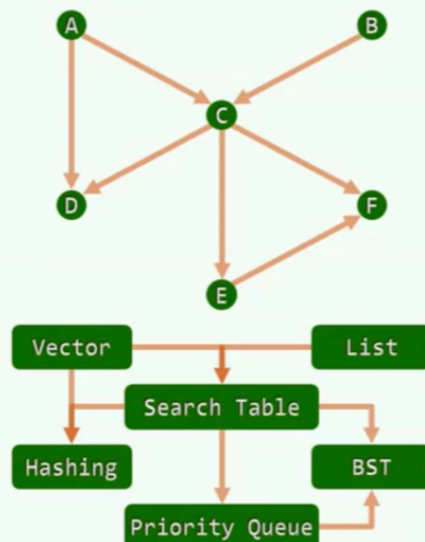
拓扑排序

有向无环图

❖ Directed Acyclic Graph

❖ 应用

- 类派生和继承关系图中，是否存在循环定义
- 操作系统中相互等待的一组线程，如何调度
- 给定一组相互依赖的课程，设计可行的培养方案
- 给定一组相互依赖的知识点，设计可行的教学进度方案
- 项目工程图中，设计可串行施工的方案
- email系统中，是否存在自动转发或回复的回路



拓扑排序



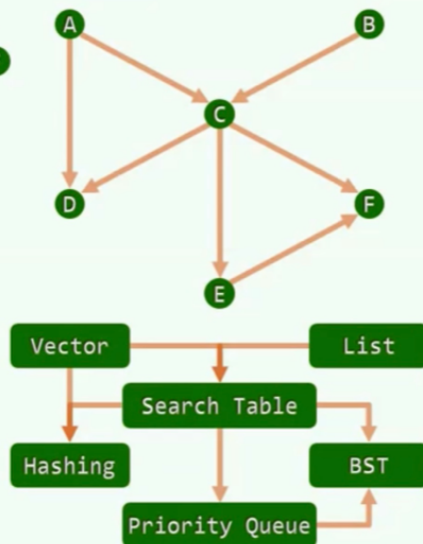
❖ 任给有向图G（不一定是DAG），尝试

将所有顶点排成一个**线性序列**，使其次序须与原图**相容**

（每一顶点都不会通过边指向**前驱**顶点）

❖ 接口要求

- 若原图存在回路（即并非DAG），检
- 否则，给出一个相容的线性序列



算法

- 这种排序不能碰到环

偏序 ~ 极值



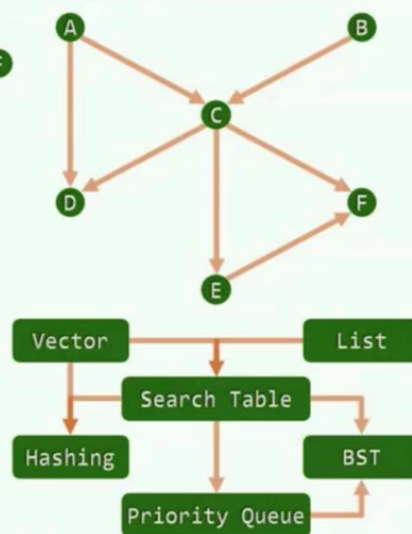
❖ 每个DAG对应于一个**偏序集**；拓扑排序对应于一个**全序集**
所谓的拓扑排序，即构造一个与指定偏序集**相容**的全序集

❖ 可以拓扑排序的有向图，必定无环 //反之...

任何DAG，都存在（至少）一种拓扑排序？是的！

❖ 有限偏序集必有**极值**元素...

❖ 可归纳证明，并直接导出一个算法



存在性



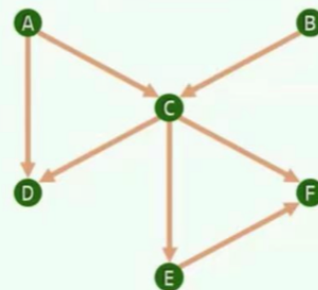
❖ 任何有向无环图 G 中

必有一个**零入度**的顶点 m

❖ 若 $G \setminus \{m\}$ 存在拓扑排序 $S = \{u_{k_1}, u_{k_2}, u_{k_3}, \dots, u_{k_{n-1}}\}$ //subtraction

则 $S' = \{m, u_{k_1}, u_{k_2}, u_{k_3}, \dots, u_{k_{n-1}}\}$ 即为 G 的拓扑排序 //DAG子图亦为DAG

当然，只要 m 不唯一，拓扑排序也



在算法开始必须要找到一个零入度的点（可以认为你在大学里上的第一门课，没有任何的先验知识的）

而且只要是无环图且图不是无限的就一定可以找到零入度的点

策略：顺序输出零入度顶点



将所有入度为零的顶点存入栈 S ，取空队列 Q $//O(n)$

```
while ( ! S.empty() ) {  $//O(n)$ 
```

```
     $Q.enqueue( v = S.pop() );$   $//$ 栈顶 $v$ 转入队列
```

```
    for each edge(  $v, u$  )  $//v$ 的邻接顶点 $u$ 若入度仅为1
```

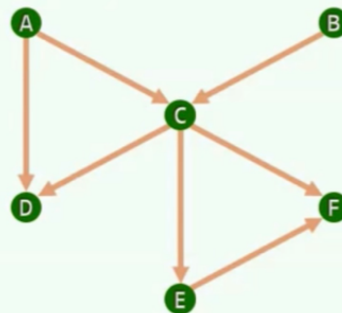
```
        if (  $u.inDegree < 2$  )  $S.push( u );$   $//$ 则入栈
```

```
     $G = G \setminus \{ v \};$   $//$ 删除 $v$ 及其关联边 ( 邻接顶点入度减1 )
```

```
}  $//$ 总体 $O(n + e)$ 
```

```
return  $|G| ?$  "NOT_A_DAG" :
```

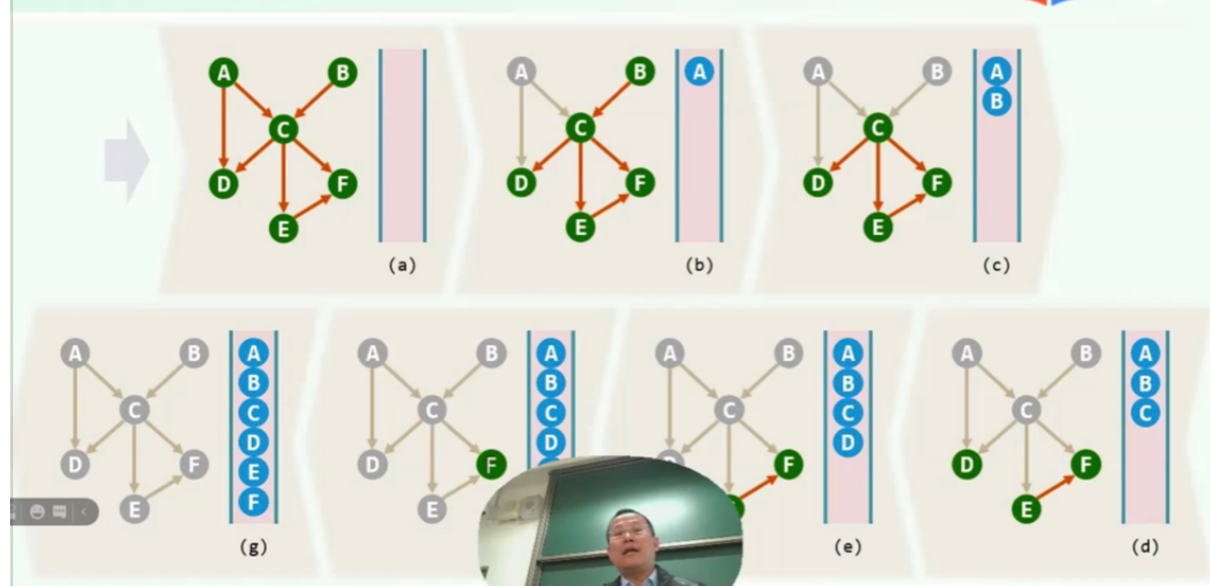
仅当原图可拓扑排序



实例

我们可以从开始不断找到一些零入度的不断去掉最终图会被清空

实例



这个方法虽然看起来很好看，但是找0入度的点是每次都要 $O(n)$ 的时间，而且如果不复制图占用大量空间的话，原本的图可能还会受到伤害