第六章 图（三）

最短路径

- 在网络（带权图）中，求两个不同顶点之间的所有路径中，边的权值之和最小的那一条路径
  - 这条路径就是两点之间的最短路径（**Shortest Path**）
- 最短路径问题：从图中某一顶点出发，求其到所有其他顶点的最短路径
  - 最短路径与最小生成树问题主要有三点不同：

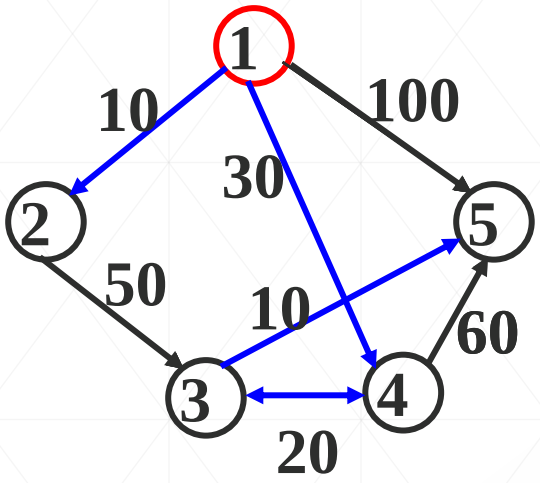
I. 最短路径的操作对象主要是有向图（网），而最小生成树的操作对象是无向图；

II.最短路径有一个始点，最小生成树没有；

III.最短路径关心的是始点到每个顶点的路径最短，而最小生成树关心的是整个树的代价最小

最短路径（cont.）

- 最短路径问题可以采用Dijkstra算法求解
  - Dijkstra提出按路径长度的递增次序，逐步产生最短路径的贪心算法

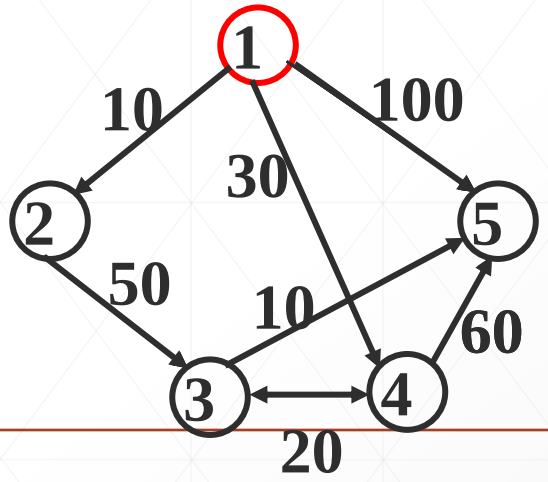


源点S	中间结点	终点	路径长度
1		2	1 0
1		4	3 0
1	4	3	5 0
1	4 3	5	6 0

最短路径（cont.）

- Dijkstra算法
  - 在Dijkstra算法中，引进了一个辅助向量 D；
  - 每个分量$D[i]$表示当前所找到的从始点v_0到每个终点v_i的最短路径长度；
  - 初值为始点v_0到各个终点v_i的直接距离，即若从始点到某终点有（出）弧，则为弧上的权值，否则为∞。

[例]



- 如果1是始点v_0，则$D[i]$的初值为： $D[i]$={10, ∞, 30, 100 }
- 显然， $D[i]$=min {$D[i]|v_i \in V$}是始点出发的长度最短的一条最短路径

最短路径（cont.）

- **Dijkstra算法 (cont.)**
  - 设S为最短路径已经确定的顶点集合；
  - 下一条最短路径（设其终点为v_i）为以下之一：

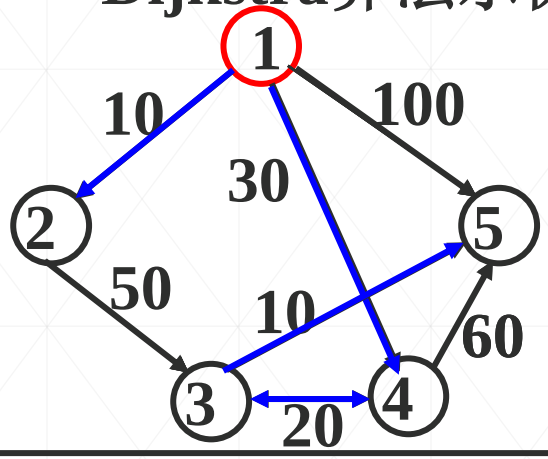
① 中间只经过S中的顶点v_j而后到达顶点v_i的路径

② 弧$< v_0 , v_i >$

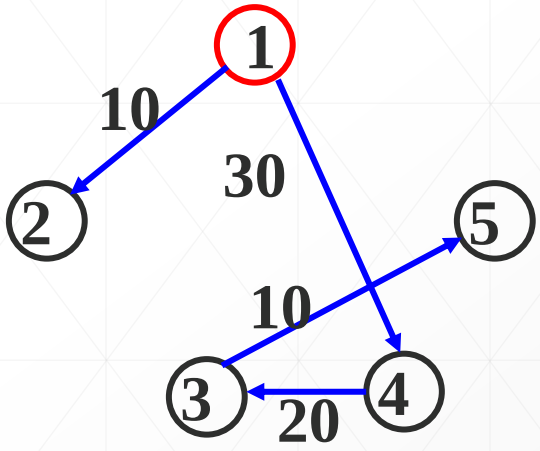
即 $D[i]$=min $\{< v_0 , v_i >$, or $D[j] + < v_j , vi > | v_i \in V - S\}$

最短路径 (cont.)

- Dijkstra算法示例



$$C=\begin{pmatrix} \infty & 10 & \infty & 30 & 100 \\ \infty & \infty & 50 & \infty & \infty \\ \infty & \infty & \infty & 20 & 10 \\ \infty & \infty & 20 & \infty & 60 \\ \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

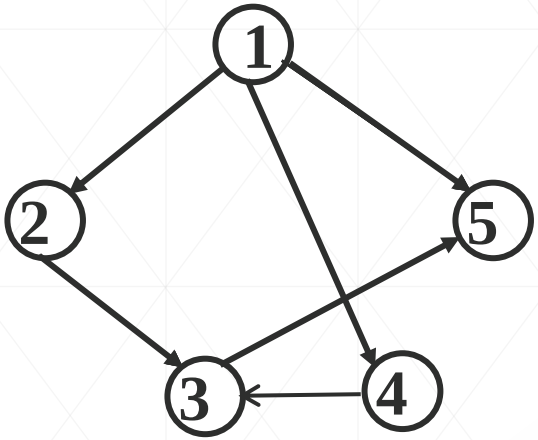


循环	S	D[2]	D[3]	D[4]	D[5]	P[2]	P[3]	P[4]	P[5]
初态	{1}	10	∞	30	100	1	1	1	1
1	{1,2}	10	60	30	100	1	2	1	1
2	{1,2,4}	10	50	30	90	1	4	1	4
3	{1,2,4,3}	10	50	30	60	1	4	1	3
4	{1,2,4,3,5}	10	50	30	60	1	4	1	3

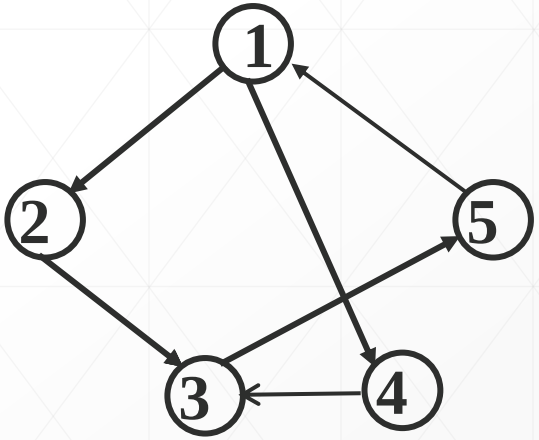
注：P[*i*]表示始点到顶点v_i的当前最短路径上，最后经过的顶点

有向无环图

- 有向无环图(Directed Acycline Graph, DAG)是不存在环路的有向图



DAG



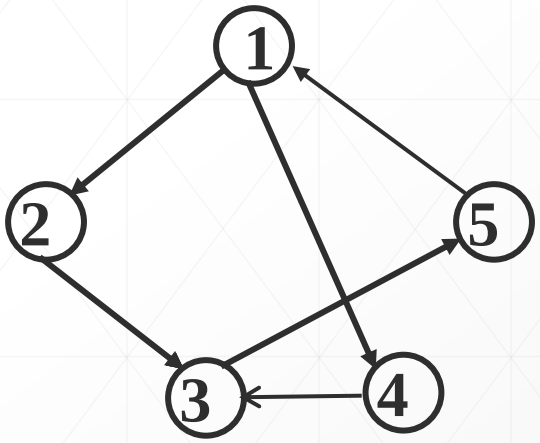
非DAG

注意：无环路的有向图对应的无向图可能存在环路。

DFS: 1, 2, 3, 5, 4

有向无环图 (cont.)

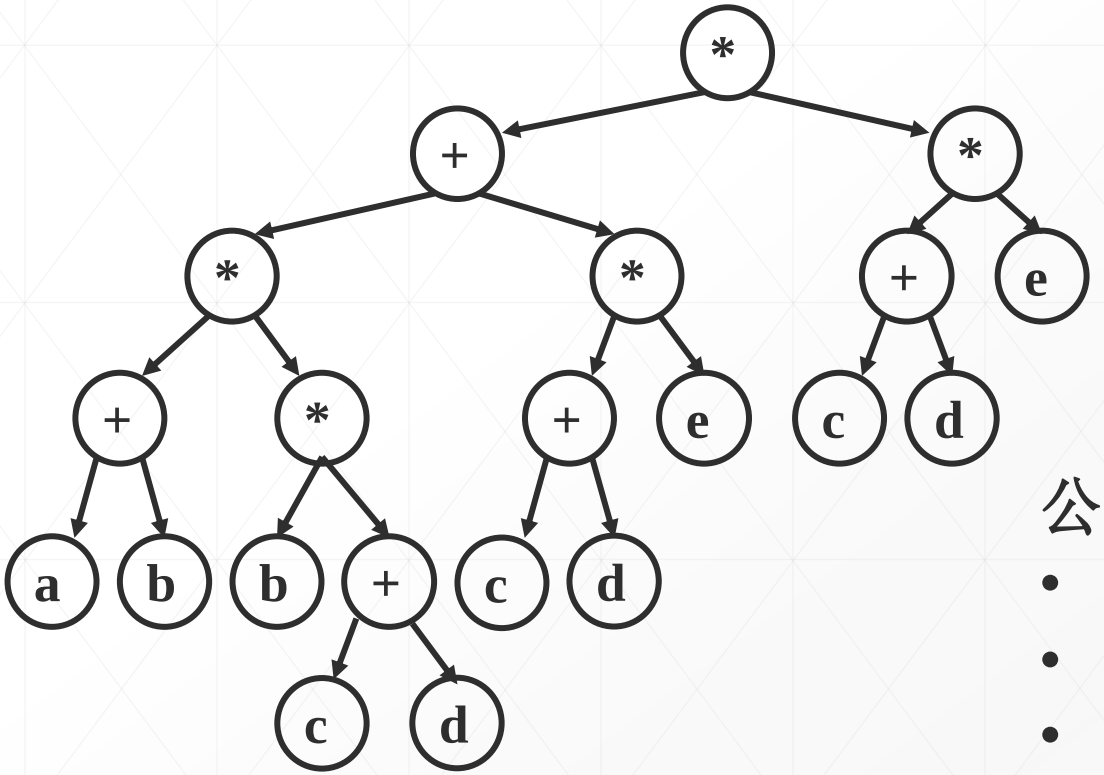
- 有向图中，可以用深度优先搜索(DFS), 找出是否存在环：从某个顶点*v*出发，进行DFS，如果存在一条从顶点*u*到*v*的回边，则有向图中存在环
  - 下图DFS: 1, 2, 3, 5, 4



非DAG

有向无环图 (cont.)

- 有向无环图可以用于表达式的共享
  - 表达式 ((a+b)*(b*(c+d))+(c+d)*e)*((c+d)*e)，用二叉树表示



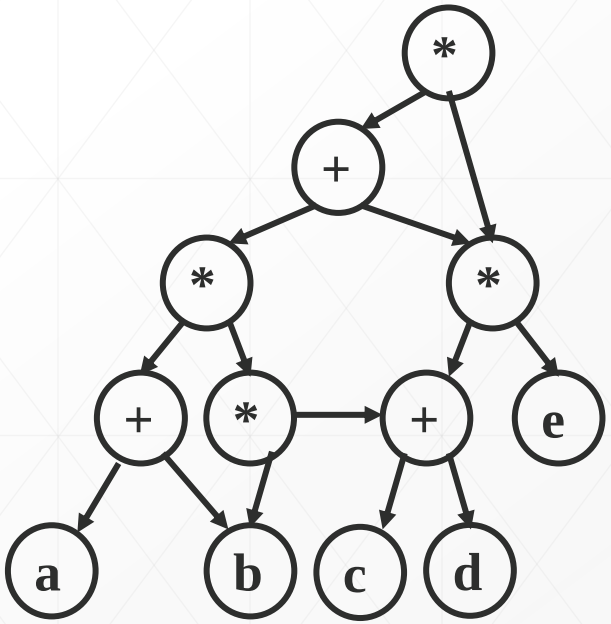
公共子式

- b
- (c+d)
- (c+d)*e

有向无环图 (cont.)

- 有向无环图可以用于表达式的共享
  - 表达式 ((a+b)*(b*(c+d))+(c+d)*e)*((c+d)*e)，用DAG表示
  - 可以节省存储空间

公共子式
- b
- (c+d)
- (c+d)*e



有向无环图 (cont.)

- 有向无环是一类具有代表性的图，主要用于研究工程项目的工序问题、工程时间进度问题等
- 一个工程(project)都可分为若干个称为活动(active)的子工程(或工序)，各个子工程受到一定的条件约束：某个子工程必须开始于另一个子工程完成之后；整个工程有一个开始点(起点)和一个终点
- 一个工程活动可以用有向无环图来描述

拓扑排序

➢偏序关系

- 若集合X上的关系R是：
  - 自反的：$x\ R\ x$
  - 反对称的：$x\ R\ y$ and $y\ R\ x \Longrightarrow x = y$
  - 传递的：$xRy\ \&\ yRz \Longrightarrow xRz$

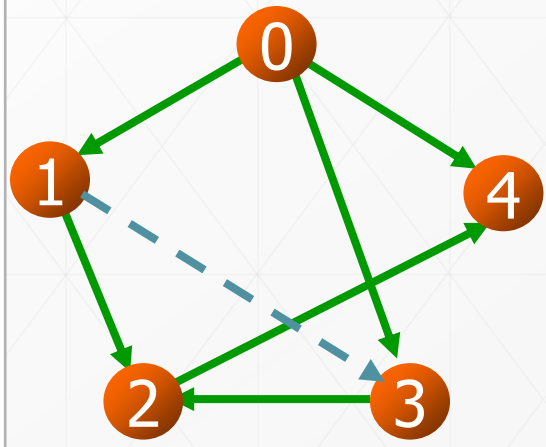
则称R是集合X上的偏序关系, 例如小于等于关系

拓扑排序（cont.）

➢全序关系

- 设关系R是集合X上的偏序，如果对每个$x, y \in X$，必有xRy或者yRx，则称R是集合X上的全序关系
- 全序指集合中全体成员之间均可比较
- 偏序指集合中仅有部分成员之间可比较

- 右图是一个偏序关系，因为 1, 3 没有先后关系
- 如果人为地增加 1, 3 先后关系，如 1 先于 3，则右图变为全序，称为拓扑有序
- 从图来说，全序表示任意两点之间至少存在一条路径



拓扑排序 (cont.)

- 从偏序得到的全序称为拓扑有序
- 由偏序得到拓扑有序的操作称为拓扑排序
- 拓扑排序算法：

  ① 在有向图中选一个没有前驱的顶点且输出之

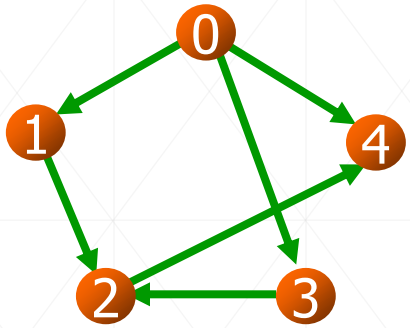
  ② 从图中删除该顶点和所有以它为尾的弧

  ③ 重复(1)(2)两步，直到所有顶点输出为止

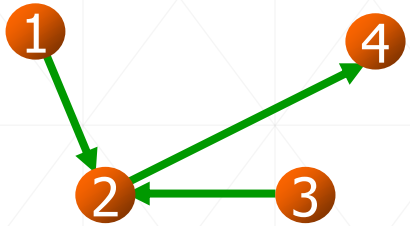
- 关系：通过拓扑排序操作得到的线性序列为拓扑有序序列

拓扑排序（cont.）

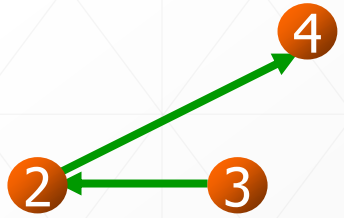
[例]拓扑排序举例



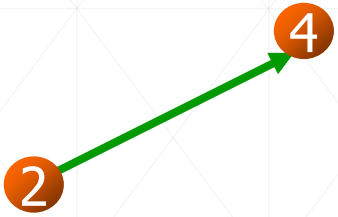
原图



输出0之后



输出0, 1之后



输出0, 1, 3之后

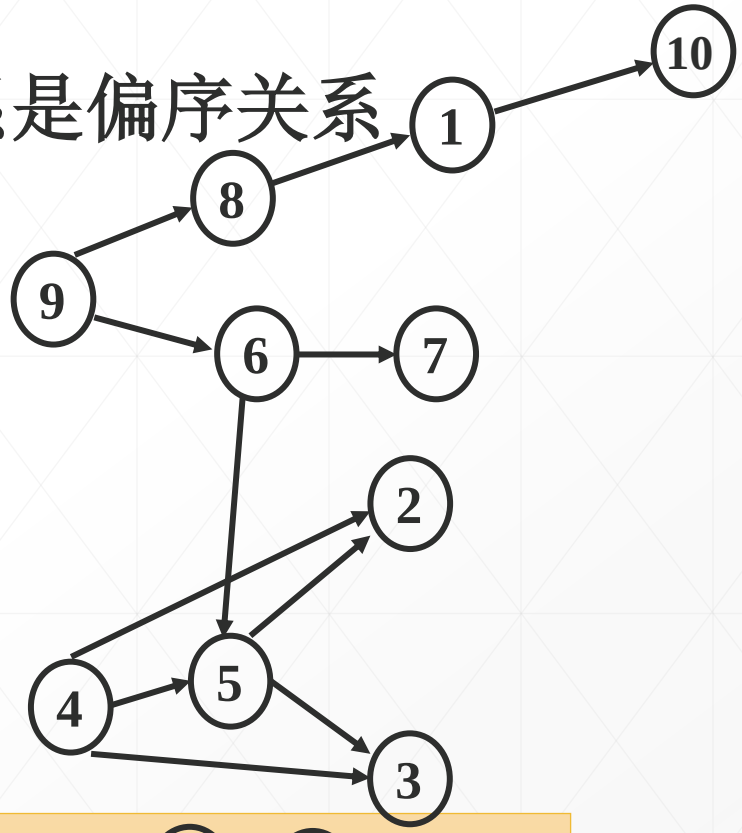
输出0, 1, 3, 2之后

- 最后输出拓扑排序结果：0, 1, 3, 2, 4

拓扑排序 (cont.)

[例] 课程及课程间的先修关系是偏序关系

课程代号	课程名称	先修课代号
1	计算机原理	8
2	编译原理	4, 5
3	操作系统	4, 5
4	程序设计	无
5	数据结构	4, 6
6	离散数学	9
7	形式语言	6
8	电路基础	9
9	高等数学	无
10	计算机网络	1



拓扑序列