# 第13章 图

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

# 第13章 图的基本概念

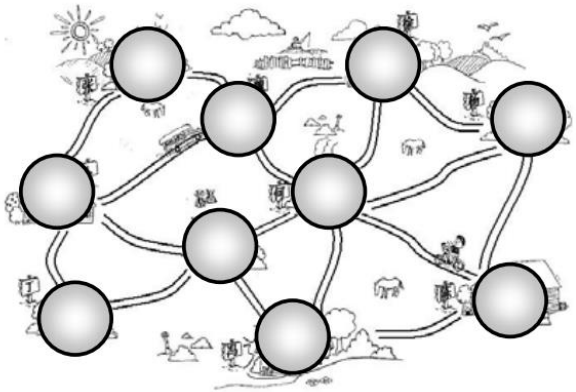
- 图的定义
- 图的术语
- 图的运算
- 图的存储
- 图的遍历
- 图遍历的应用

- 图的基本术语和性质
- 图的存储方式：邻接矩阵和邻接表
- 图的遍历：深度、广度遍历
- 欧拉回路是否存在的条件，如何寻找欧拉回路
- 拓扑排序、关键路径

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

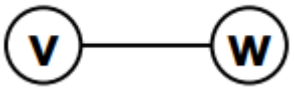
# 图的定义

- 图：表示“多对多”的关系；

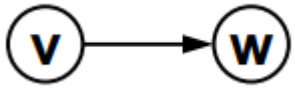


- 图可以用G=(V, E)表示。其中，V是顶点的集合，E是连接顶点的边(弧)的集合。

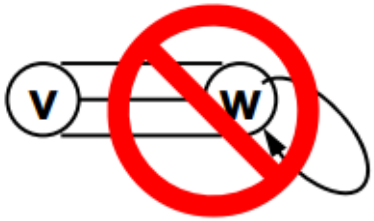
边是顶点对：(v, w) ∈ E ， 其中 v, w ∈ V




有向边 < v, w> 表示从v指向w的边（单行线）




不考虑重边和自回路



# 图的基本术语

- 邻接：如(Vi，Vj)是图中的一条边，则称Vi和Vj是邻接的。如<Vi，Vj>是图中的一条边，则称Vi邻接到Vj，或Vj和Vi邻接。
- 度：无向图中邻接于某一结点的边的总数。
- 入度：有向图中进入某一结点的边数，称为该结点的入度
- 出度：有向图中离开某一结点的边数，称为该结点的出度

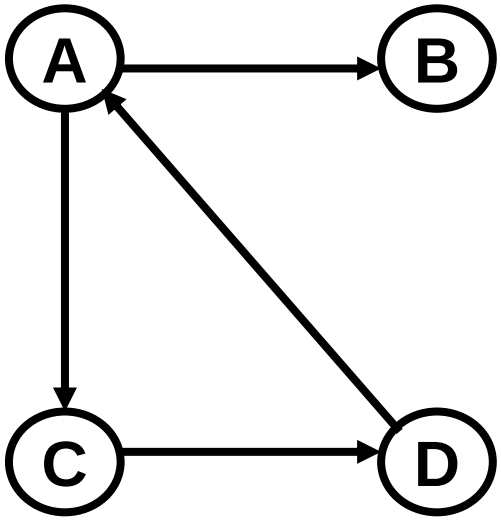
上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 子图

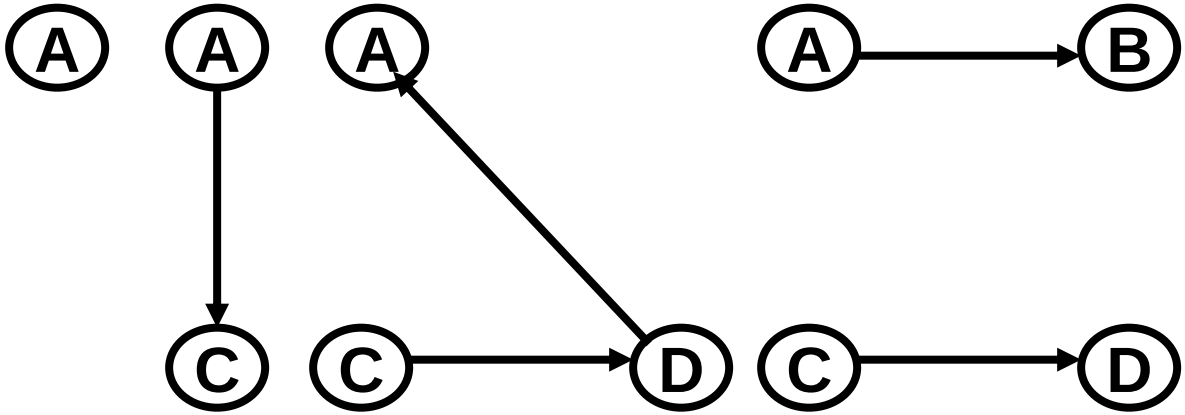
设有两个图G=(V，E)和G‘=(V’，E’)，如果

$V' \subseteq V, E' \subseteq E$ 则称G’是G的子图

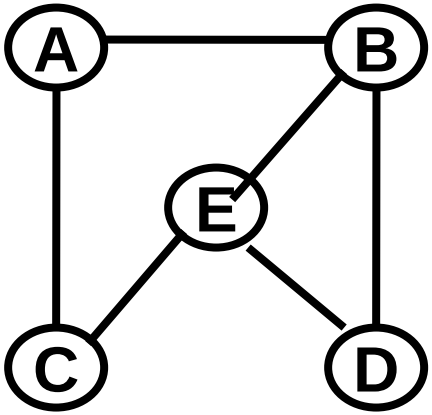
有向图 G1



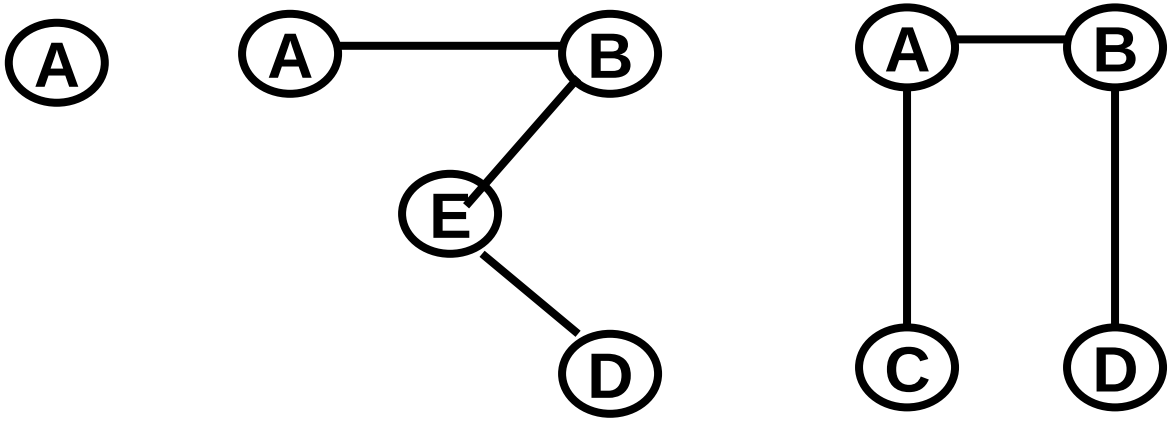

有向图G1的子图

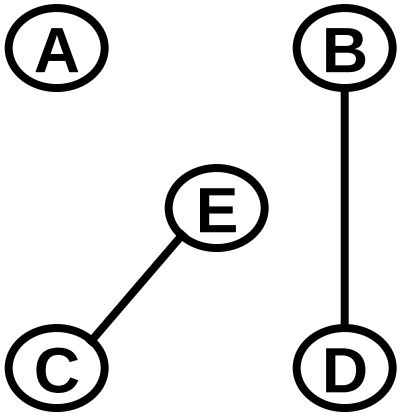


无向图 G2




无向图G2的子图

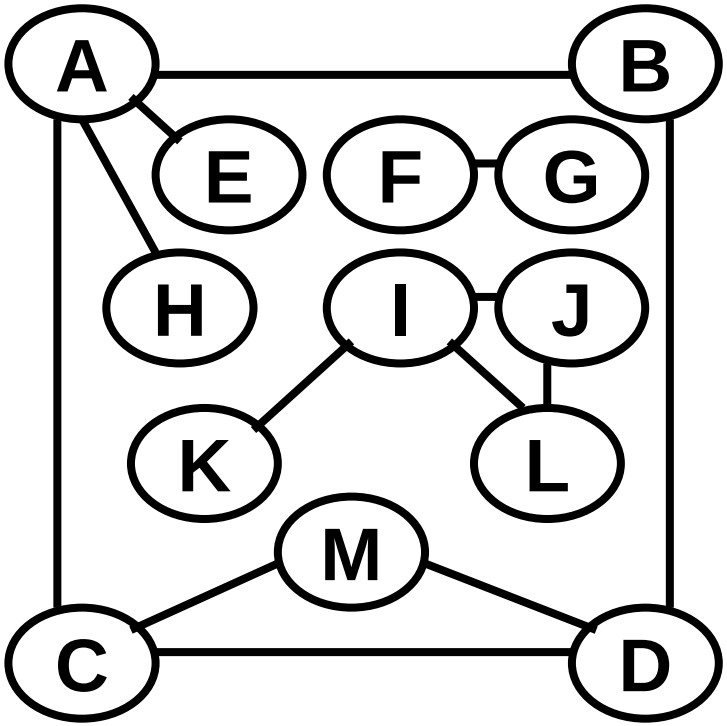


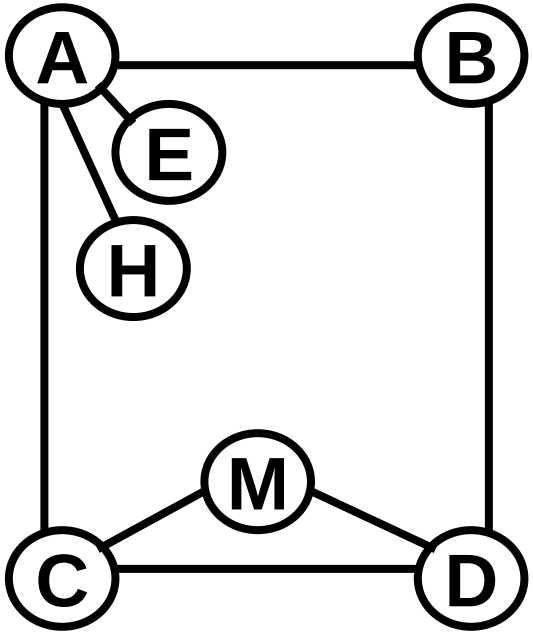
# 无向图的连通性

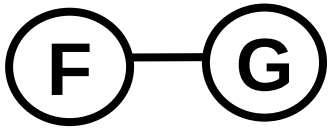
- 连通：顶点v至v’之间有路径存在
- 连通图：无向图 G 的任意两点之间都是连通的，则称 G 是连通图。
- 连通分量：非连通图中的极大连通子图

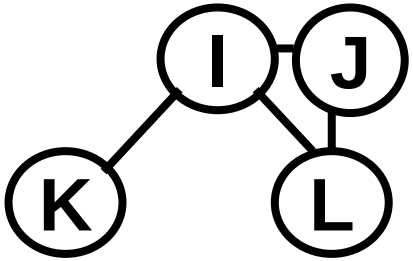
无向图G




无向图G的三个连通分量



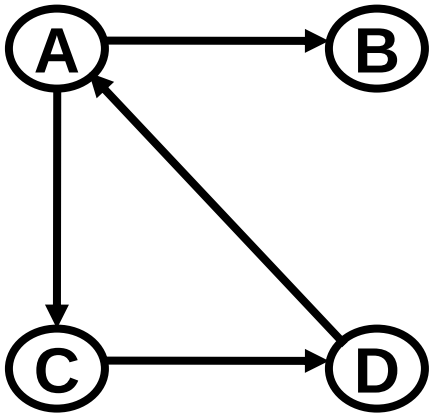
# 有向图的连通性

- 强连通图：有向图 G 的任意两点之间都是连通的，则称 G 是强连通图。
- 强连通分量：极大连通子图
- 弱连通图：如有向图G不是强连通的，但如果把它看成是无向图时是连通的，则称该图是弱连通的

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

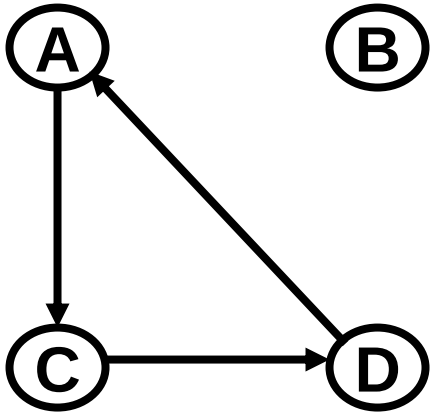
# 有向图的连通性

不是强连通图。因为B到其他结点都没有路径。但此图是弱连通的。




有向图G

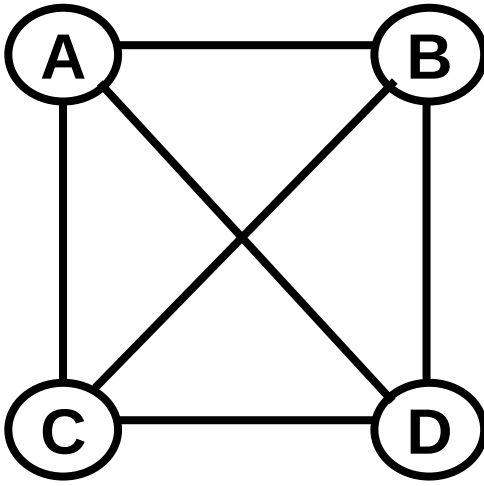
有向图G的两个强连通分量



上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 完全图

- 无向完全图：每两个节点之间都有边的无向图称为完全图。完全图有 n(n-1)/2 条边的无向图，其中 n 是结点个数。即 $C_n^2$
- 有向完全图：每两个节点之间都有两条弧的有向图称为有向完全图。有向完全图有 n(n-1) 条边，其中 n 是结点个数。即 $2C_n^2$
- 如果一个有向图中没有环，则称为有向无环图，简写为DAG

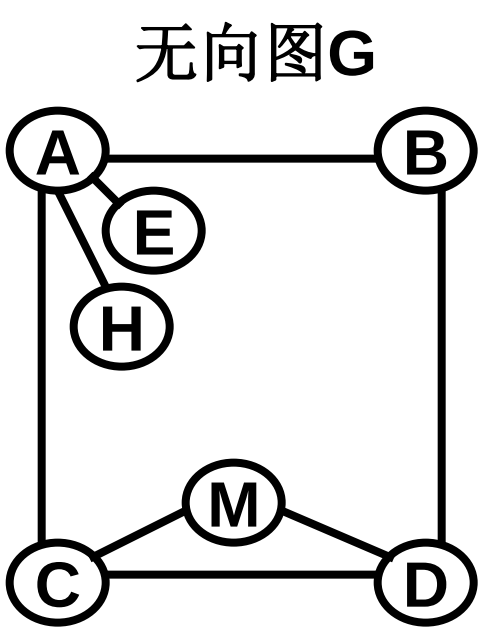


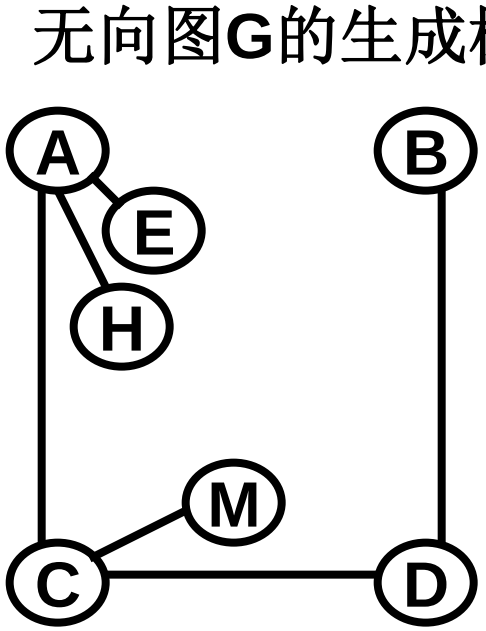

无向完全图

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 生成树

- 生成树是连通图的极小连通子图。包含图的所有 n 个结点，但只含图的 n-1 条边。在生成树中添加一条边之后，必定会形成回路或环。



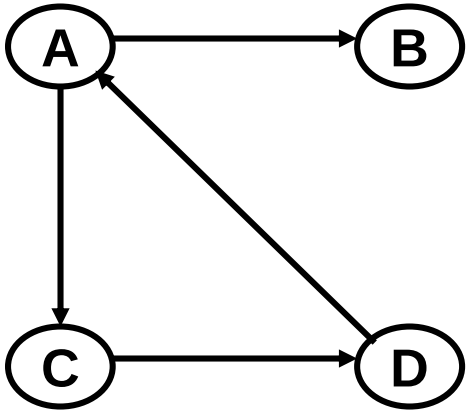
# 图的存储

- 邻接矩阵和加权邻接矩阵
- 邻接表

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 邻接矩阵—有向图

设有向图具有 n 个结点，则用 n 行 n 列的布尔矩阵 A 表示该有向图如果i 至 j 有一条有向边, A[i,j] = 1 ,如果 i 至 j 没有一条有向边,A[i,j] = 0




表示成右图矩阵

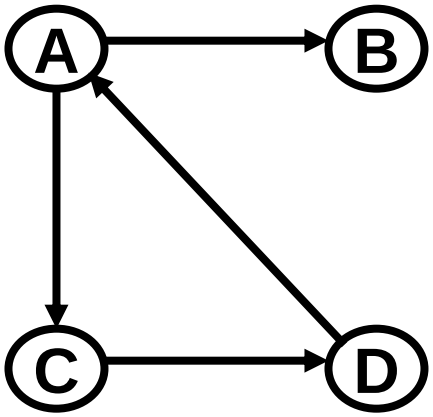
$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

**注意： 出度：i行之和。入度：j列之和。**

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 邻接矩阵—有向图

•在物理实现时的考虑：分别用 0、1、2、3 分别标识结点A、B、C、D。而将真正的数据字段之值放入一个一维数组之中。




表示成右图矩阵

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 |
| 3 | 1 | 0 | 0 | 0 |

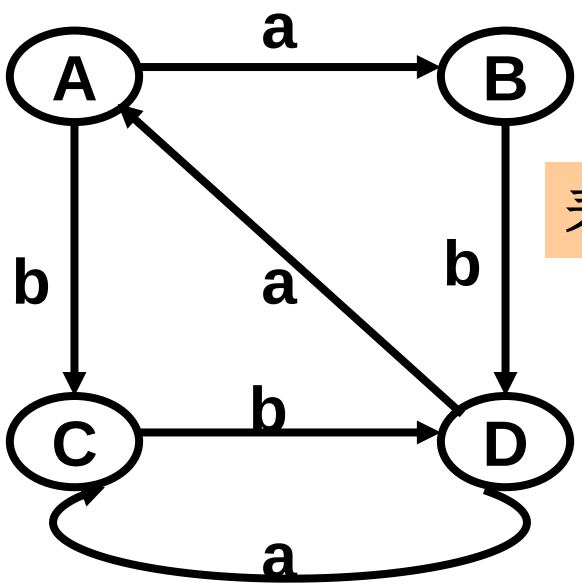
| 0 | 1 | 2 | 3 |
|---|---|---|---|
| A | B | C | D |

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 加权的邻接矩阵—有向图

设有向图具有 n 个结点，则用 n 行 n 列的矩阵 A 表示该有向图； 如果i 至 j 有一条有向边且它的权值为a ，则A[i,j] = a 。如果 i 至 j 没有一条有向边。则

A[i,j] = 空 或其它标志，如 ∞



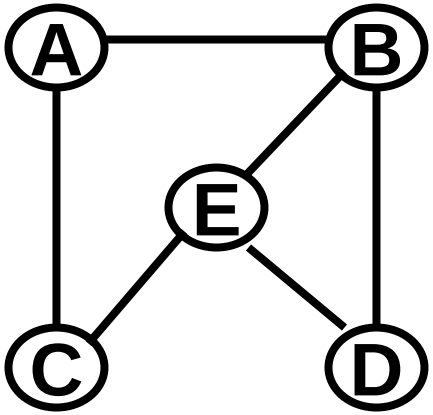

表示成右图矩阵

$$\begin{pmatrix} 0 & a & b & \infty \\ \infty & 0 & \infty & b \\ \infty & \infty & 0 & b \\ a & \infty & a & 0 \end{pmatrix}$$

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 邻接矩阵—无向图

设无向图具有 n 个结点，则用 n 行 n 列的布尔矩阵 A 表示该无向图；并且 A[i,j] = 1，如果i 至 j 有一条无向边；A[i,j] = 0如果 i 至 j 没有一条无向边




表示成右图矩阵

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

注意： 无向图的邻接矩阵是一个对称矩阵

i 结点的度：i 行或 i 列之和。

在物理实现时的考虑，和前一页的无向图类似。

# 邻接矩阵的特点

- 优点：判断任意两点之间是否有边方便，仅耗费 O(1) 时间。
- 缺点：即使 << $n^2$ 条边，也需内存 $n^2$ 单元，太多；仅读入数据耗费 O( $n^2$ ) 时间，太长。而大多数的图的边数远远小于$n^2$。

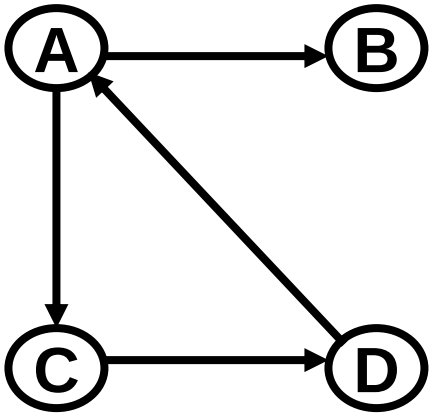
上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

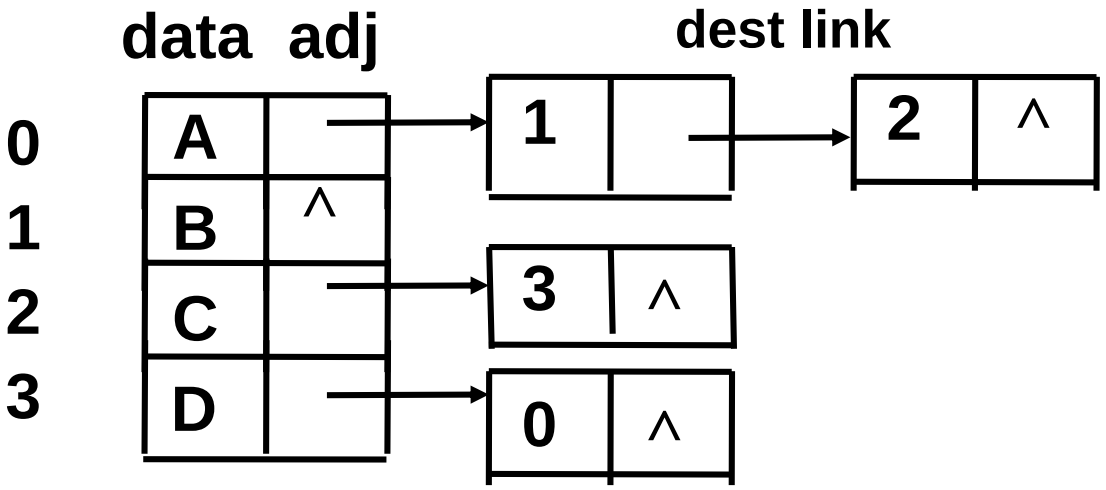
# 邻接表

- 设有向图或无向图具有 n 个结点，则用结点表、边表表示该有向图或无向图。
- 结点表：用数组或单链表的形式存放所有的结点值。如果结点数n已知，则采用数组形式，否则应采用单链表的形式。
- 边表(边结点表)：每条边用一个结点进行表示。同一个结点出发的所有的边形成它的边结点单链表。

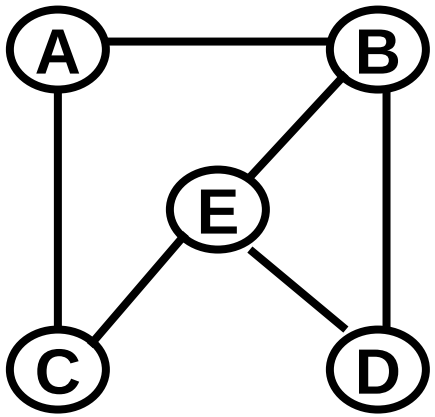
上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

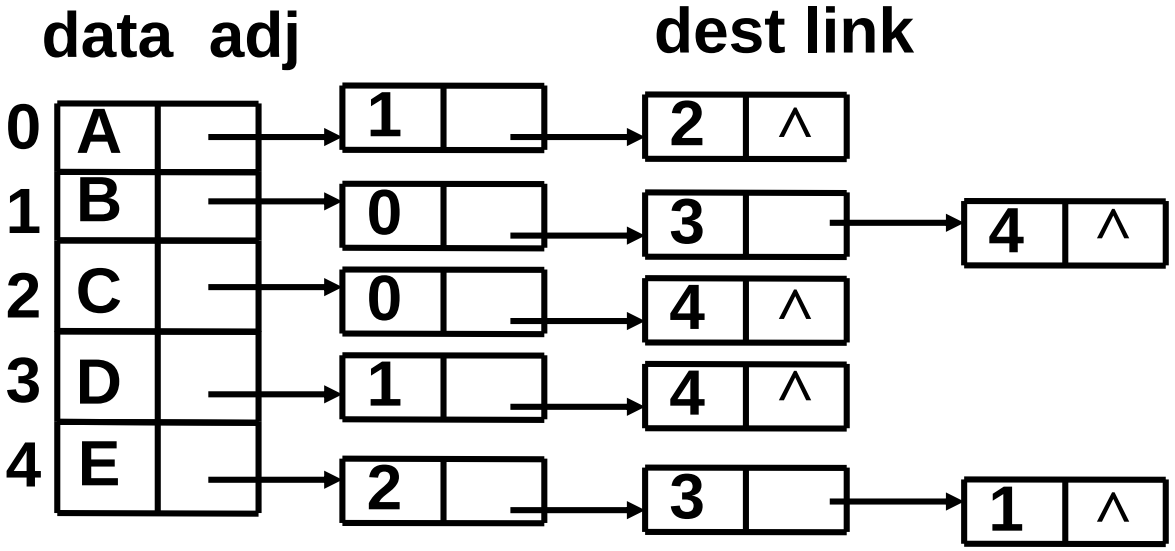
有向图 G1




无向图 G2



上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 邻接表的特点

- 邻接表是图的标准存储方式
- 优点：内存 ＝ 结点数 ＋ 边数，处理时间也是结点数 ＋ 边数，即为O(|V|+|E|)。
- 当谈及图的线性算法时，一般指的是O(|V|+|E|)
- 缺点：
  - 确定 i --> j 是否有边，最坏需耗费 O(n) 时间。
  - 无向图同一条边表示两次，边表空间浪费一倍。
  - 有向图中寻找进入某结点的边，非常困难。

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 图的遍历

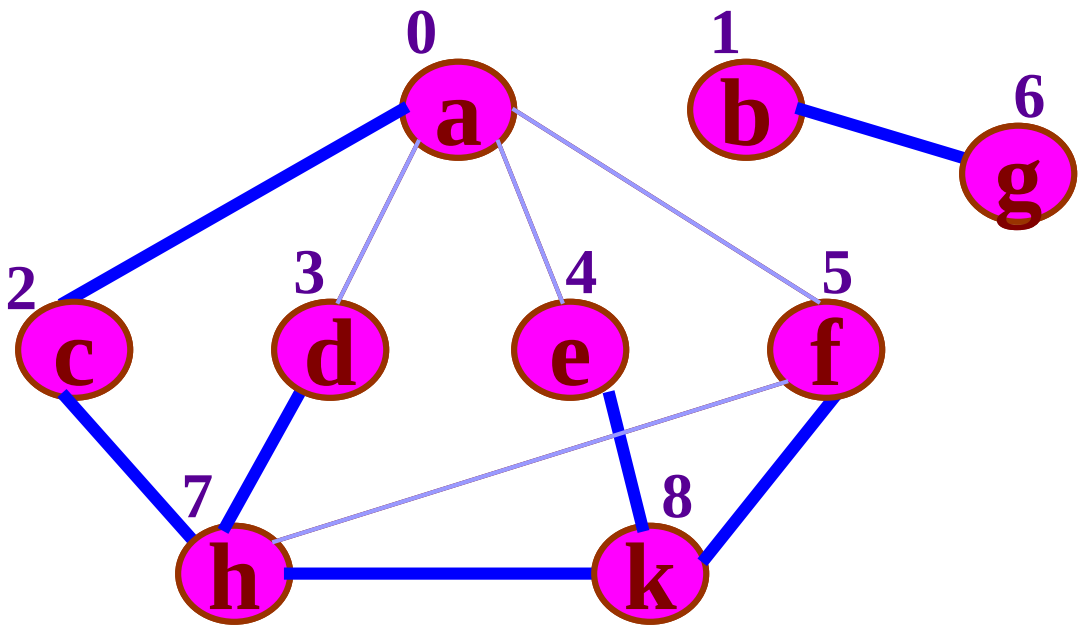
对有向图和无向图进行遍历是按照某种次序系统地访问图中的所有顶点，并且使得每个顶点只能被访问一次。在图中某个顶点可能和图中的多个顶点邻接并且存在回路，因此在图中访问一个顶点u之后，在以后的访问过程中，又可能再次返回到顶点u，所以图的遍历要比树的遍历更复杂

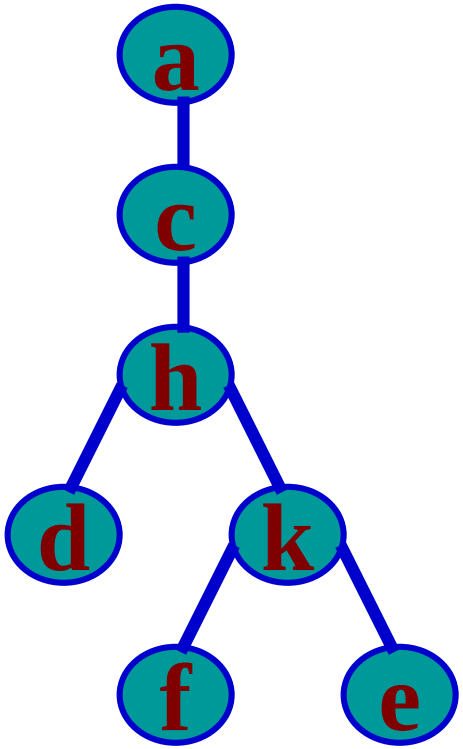
- 深度优先搜索
- 广度优先搜索

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 深度优先搜索（DFS）

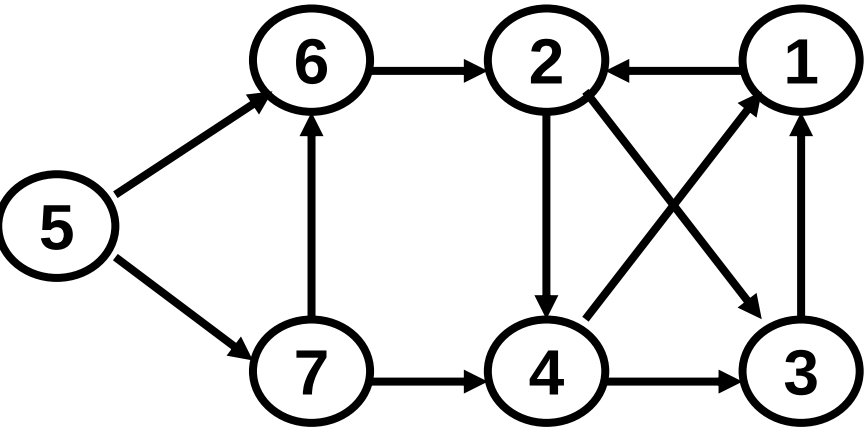
例



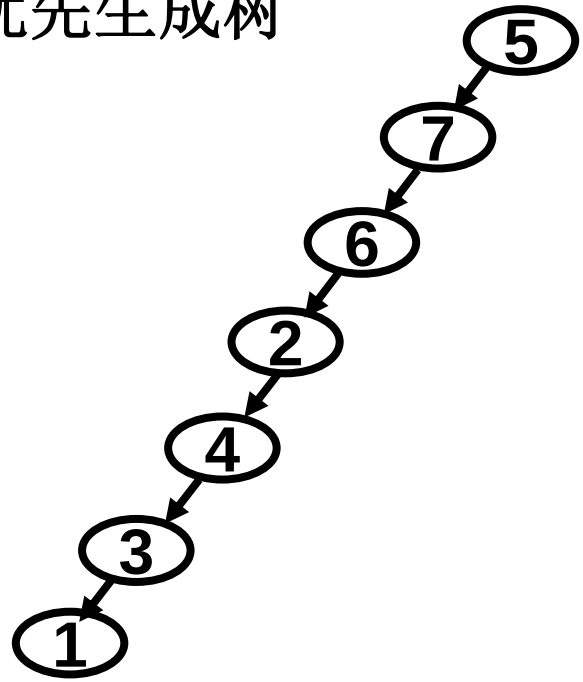

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 访问标志： | T | T | T | T | T | T | T | T | T |

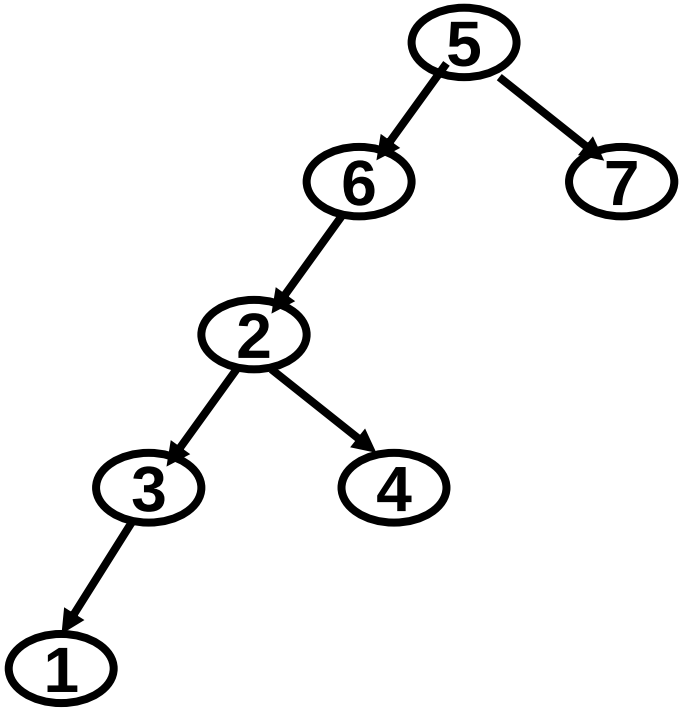
| 访问次序： | a | c | h | d | k | f | e | b | g |
|---|---|---|---|---|---|---|---|---|---|




从结点5开始进行深度优先的搜索，则遍历序列可以为：5，7，6，2，4，3，1，
也可以为：
5，6，2，3，1，4，7。

深度优先生成树

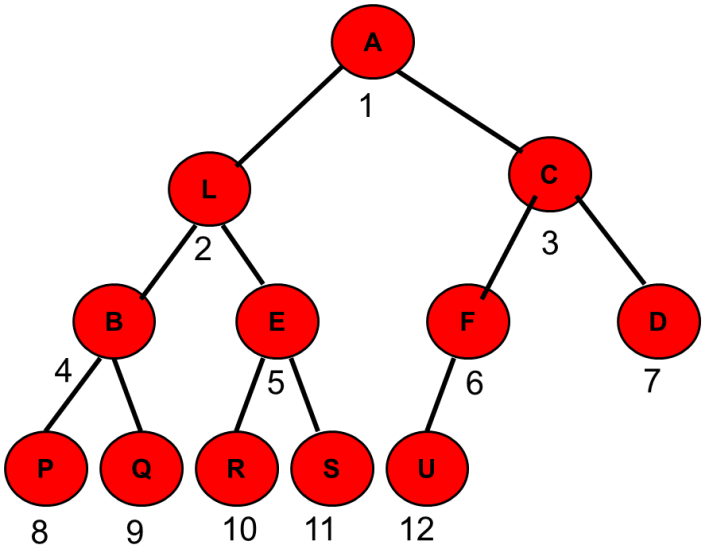


上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 深度优先搜索的实现

- 深度优先搜索DFS的实现方法和树的**前序遍历**算法类似，但必须对访问过的顶点加以标记
- DFS函数不需要参数，也没有返回值。它从编号最小的结点出发开始搜索，并将对当前对象的深度优先搜索的序列显示在显示器上。

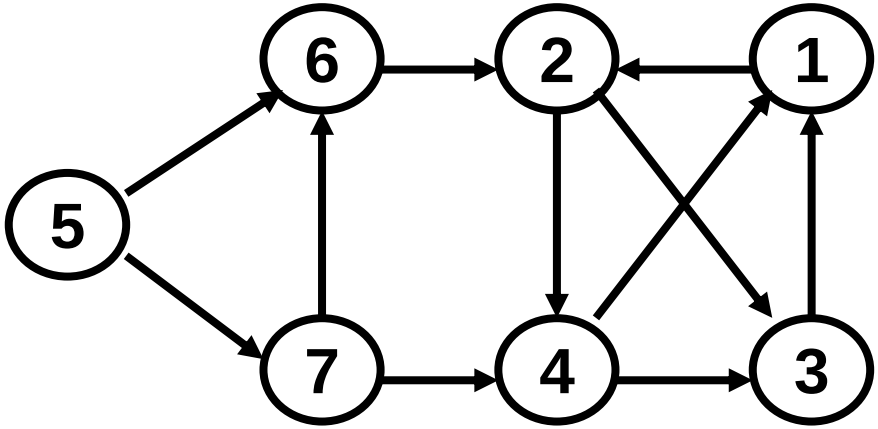


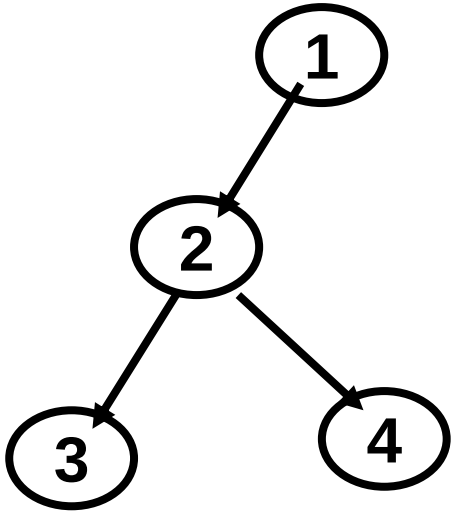
# 时间性能分析

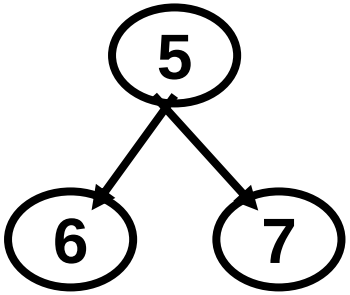
- dfs函数将对所有的顶点和边进行访问，因此它的时间代价和顶点数 |V| 及边数 |E| 是相关的，即是O(|V|+|E|)。
- 如果图是用邻接矩阵来表示，则所需要的时间是$O(|V|^2)$。

# 广度优先搜索 BFS

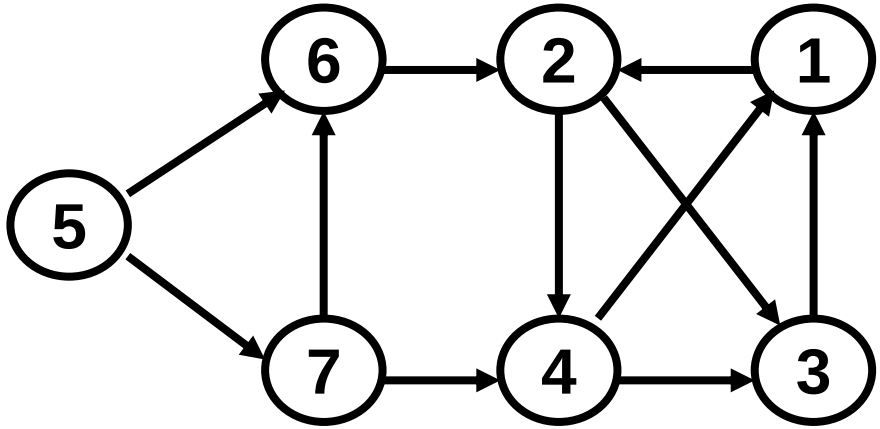
按照顶点序号小的先访问，序号大的后访问的原则，则它的广度优先访问序列为：1，2，4，3，5，6，7。对应的广度优先生成森林为

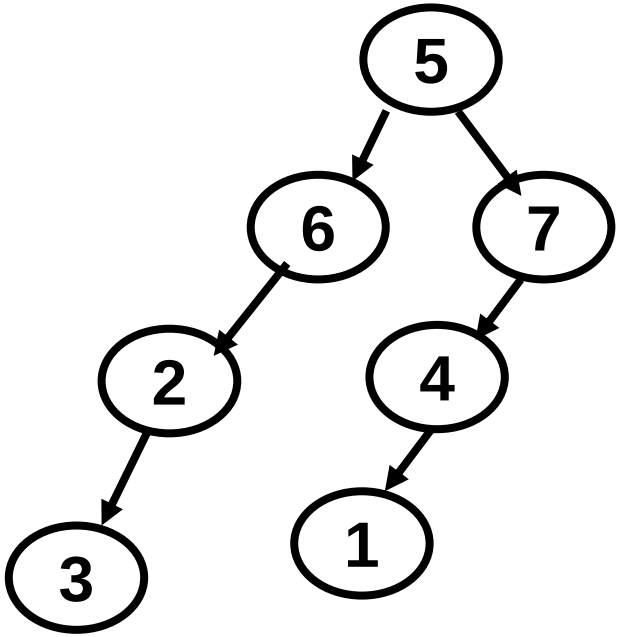


从不同的结点开始可以得到不同的搜索序列。例如，从5开始广度优先搜索这个图，得到的遍历序列为：5，6，7，2，4，3，1。



# 广度优先搜索 BFS

广度优先搜索类似于树的从树根出发的按照层次的遍历。它的访问方式如下：

1、选中第一个被访问的顶点；

2、对顶点作已访问过的标志；

3、依次访问已访问顶点的未被访问过的第一个、第二个、第三个.....第 m 个邻接顶点 W1 、W2、W3..... Wm ，进行访问且进行标记，转向3；

4、如果还有顶点未被访问，则选中一个起始顶点，转向2；

5、所有的顶点都被访问到，则结束。

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 广度优先搜索的实现

- 需要记录每个结点是否已被访问
- 需要记住每个已被访问的结点的后继结点，然后依次访问这些后继结点。这可以用一个队列来实现
- 过程（☹举例子）
  - 将序号最小的顶点放入队列
  - 重复取队列的队头元素进行处理，直到队列为空。对出队的每个元素，首先检查该元素是否已被访问。如果没有被访问过，则访问该元素，并将它的所有的没有被访问过的后继入队
  - 检查是否还有结点未被访问。如果有，重复上述两个步骤

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 时间性能分析

- 在广度优先搜索中，每个顶点都入队一次。
- 对于每个出队的结点，需要查看它的所有邻接结点。
- 假如图是用邻接表存储，查看所有的邻接结点需要O(|E|)的时间，每个结点入队一次需要O(|V|)的时间，因此广度优先遍历的时间复杂度为O(|E|+|V|)。
- 如果图是用邻接矩阵存储，查看某个结点所有的边需要O(|V|)的时间，因此广度优先搜索的时间复杂度为$O(|V|^2)$

1. 已知邻接矩阵如图所示，从结点O出发，不能按深度优先遍历得到的结点序列是_____

A. 0, 2, 4, 3, 1, 5, 6

B. 0, 1, 3, 5, 6, 4, 2

C. 0, 4, 2, 3, 1, 6, 5

D. 0, 1, 3, 4, 2, 5, 6

答案：A

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 3 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

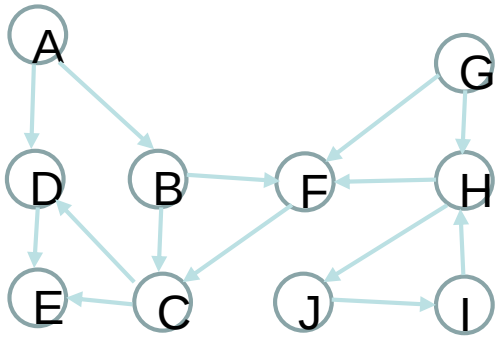
2. 已知有向图有6个顶点，边的输入序列如下：

<1, 2>, <1, 3>, <3, 2>, <3, 0>, <4, 5>, <5, 3>, <0, 1>

求该图的邻接表，强连通分量的个数

3. 按照字母从小到大的顺序（从A开始）， 写出该有向图的深度优先搜索和广度优先搜索序列结果

深度优先搜索：A,B,C,D,E,F,G,H,J,I
广度优先搜索：A, B,D, C,F, E,G,H,J,I



4. 如果图用邻接表结构存储，则常见操作的算法时间复杂度（ ）

A. 只和顶点的个数有关

B. 只和边的条数有关

C. 和顶点个数，边的条数都可能有关

D. 和两者都无关

答案：C

5. 在含n个顶点，e条边的无向图的邻接矩阵中，空元素的个数为_____

A. e

B. 2e

C. $n^2$

D. $n^2$-2e

答案：D

6. 无向图G=(V, E), 对该图进行深度优先遍历，得到的顶点序列正确的是（ ），其中 V={ a, b, c, d, e, f }, E={(a, b), (a, e), (a, c), (b, e), (c, f), (f, d), (e, d)}

A. a, b, e, c, d, f

B. a, c, f, e, b, d

C. a, e, c, b, f, d

D. a, e, d, f, c, b

答案：D

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

# 图遍历的应用

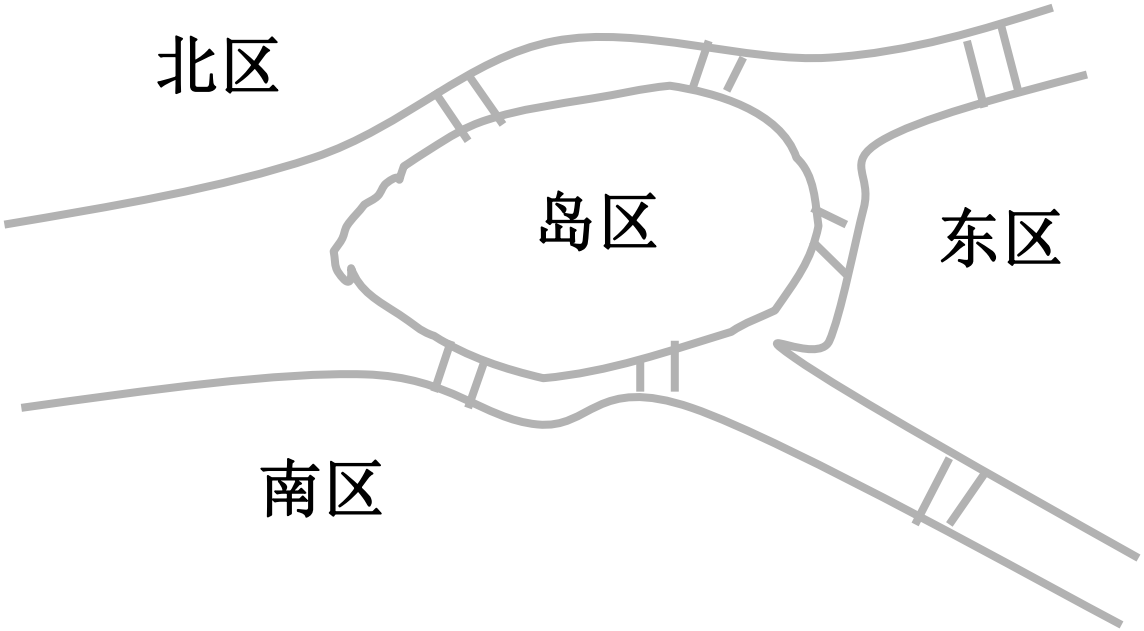
- 无向图的连通性
- 欧拉回路
- 有向图的连通性
- 拓扑排序
- 关键路径

# 无向图的连通性

- 深度优先搜索和广度优先搜索都可以用来测试无向图的连通性。
- 如果无向图是连通的，则从无向图中的任意结点出发进行深度优先搜索或广度优先搜索都可以访问到每一个结点。访问的次序是一棵深度/广度优先生成树。
- 如果图是非连通的，深度/广度优先搜索可以找到一片深度/广度优先生成森林。每棵树就是一个连通分量。对无向图来说，深度/广度优先搜索可以找到了它的所有连通分量。
- 前面介绍的讨论的深度优先和广度优先遍历中，都已实现了这个功能。在这两个函数的输出中，每一行代表一个连通分量。

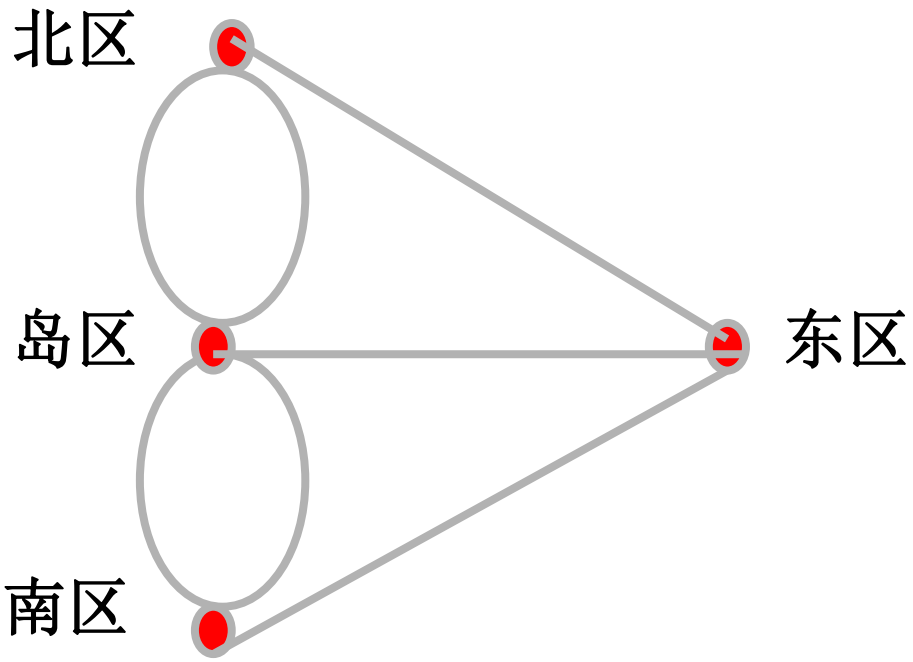
# 欧拉回路

- 哥尼斯堡七桥问题就是：能否找到一条走遍这七座桥，而且每座桥只经过一次，最后又回到原出发点的路径。



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

# 七桥问题的抽象



上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

# 欧拉的证明

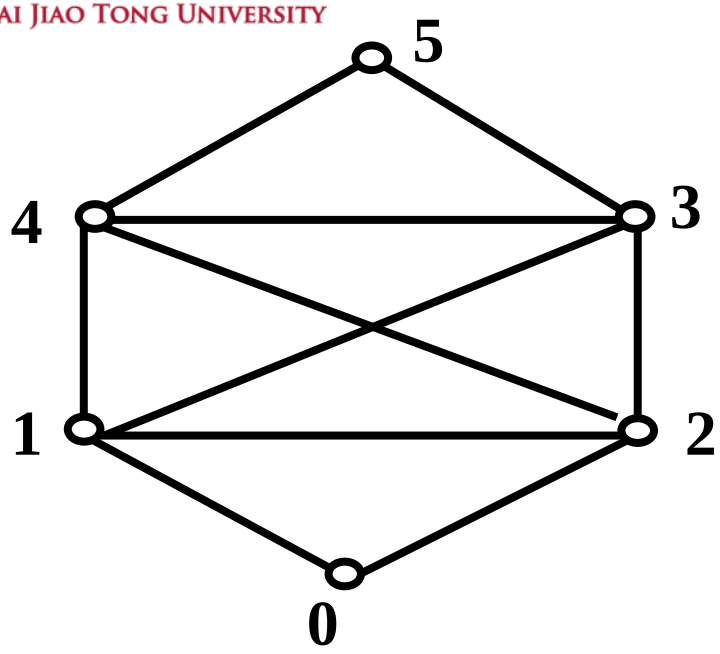
- 如果有奇数桥的地方不止两个，满足要求的路径是找不到的。
- 如果只有两个地方有奇数桥，可以从这两个地方之一出发，经过所有的桥一次，再回到另一个地方。（**欧拉路径**）
- 如果都是偶数桥，从任意地方出发都能回到原点。

# 欧拉回路和欧拉路径

- 如果能够在一个图中找到一条路径，使得该路径对图的每一条边正好经过一次，这条路径被称为欧拉路径。
- 如果再增加一个附加条件，即起点和终点是相同的，这条路径被称为欧拉回路。

# 基本想法

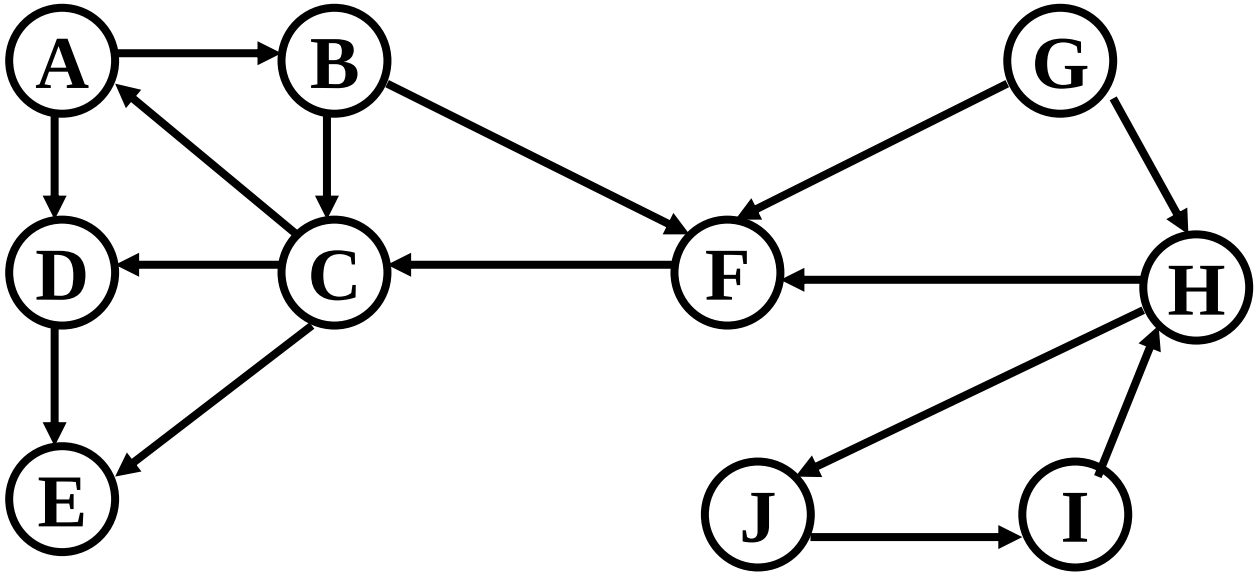
- 执行一次深度优先的搜索。从起始结点开始，沿着这条路一直往下走，直到无路可走。而且在此过程中不允许回溯。因此欧拉回路问题也被称为一笔画问题。
- 但是有很多的搜索方案是行不通的。



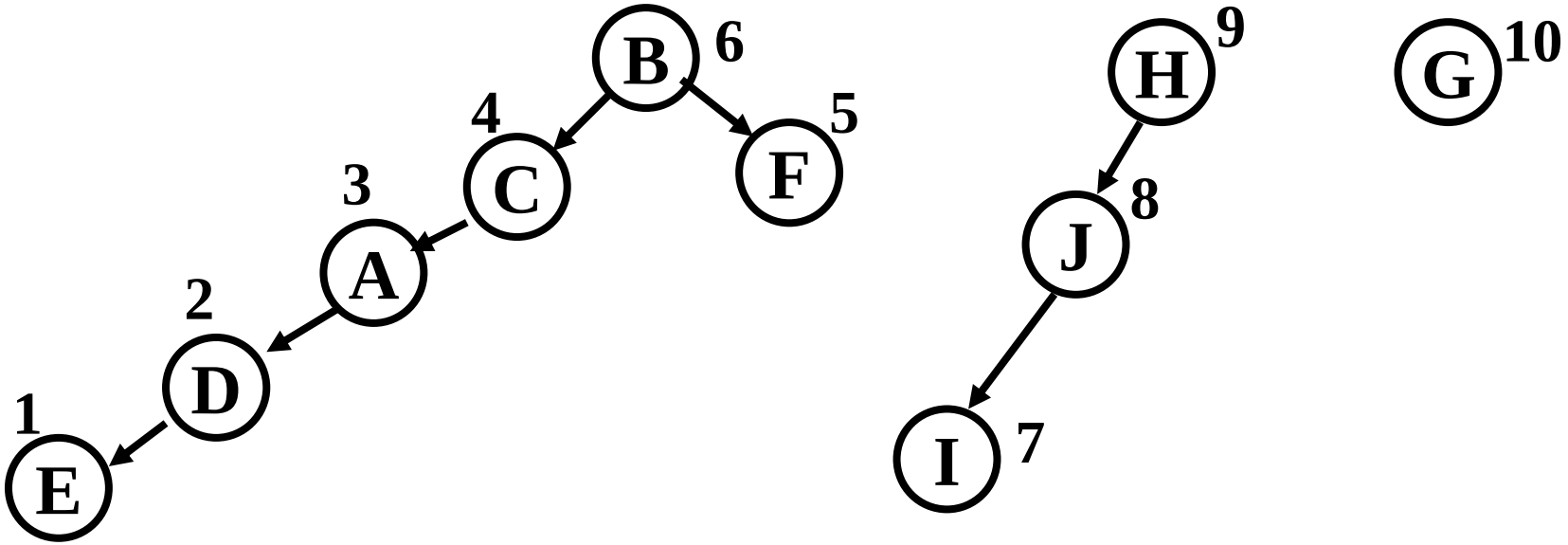

对于上图，它的所有结点的度均为偶数，应该存在欧拉回路。但如果从结点5出发开始深度优先的访问，选择的路径为5->4->3->5，则此时，就无法访问其他结点了,因为5没有其他的尚未被访问的边了。（深度优先？？）

# 有向图的连通性

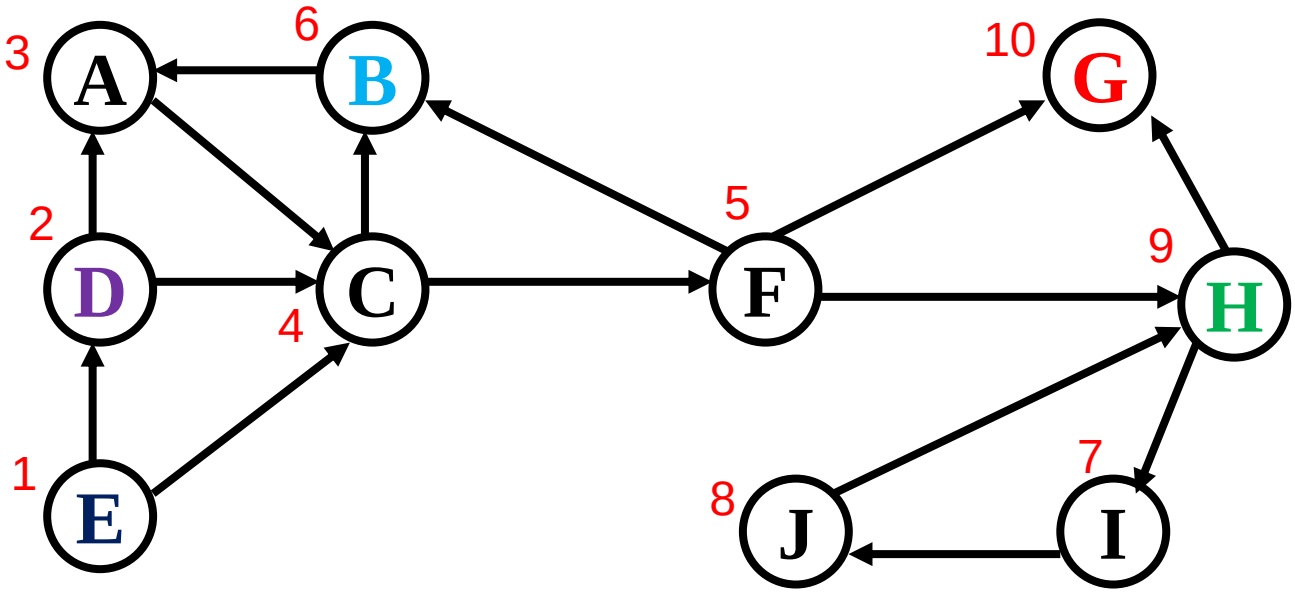
- 对有向图，深度优先搜索可以测试是否强连通，并找出所有强连通分量
- 找强连通分量的方法
  - 从任意节点开始深度优先遍历G。对森林中的每棵树进行深度优先遍历，并按后序遍历的顺序给每个节点编号
  - 将G的每条边逆向，形成Gr。从编号最大的节点开始深度优先遍历Gr。得到的深度优先遍历森林的每棵树就是G的强连通分量。

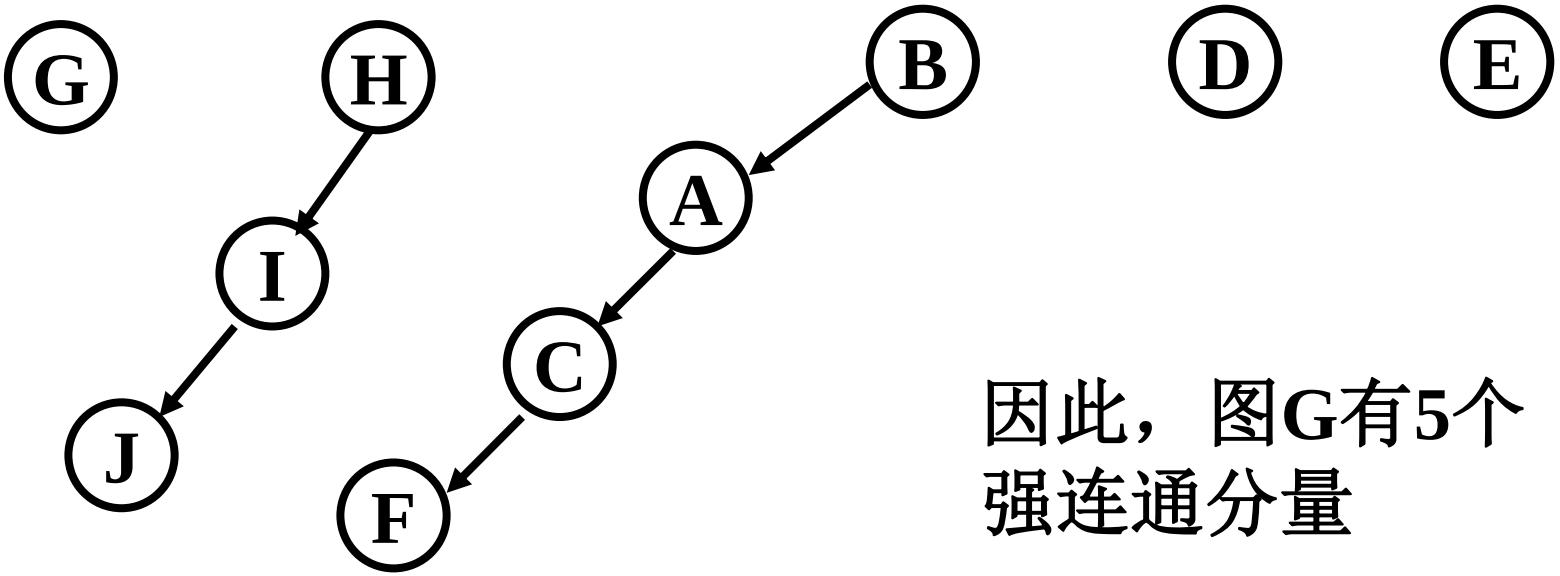



从B开始深度优先搜索



Gr




因此，图G有5个强连通分量

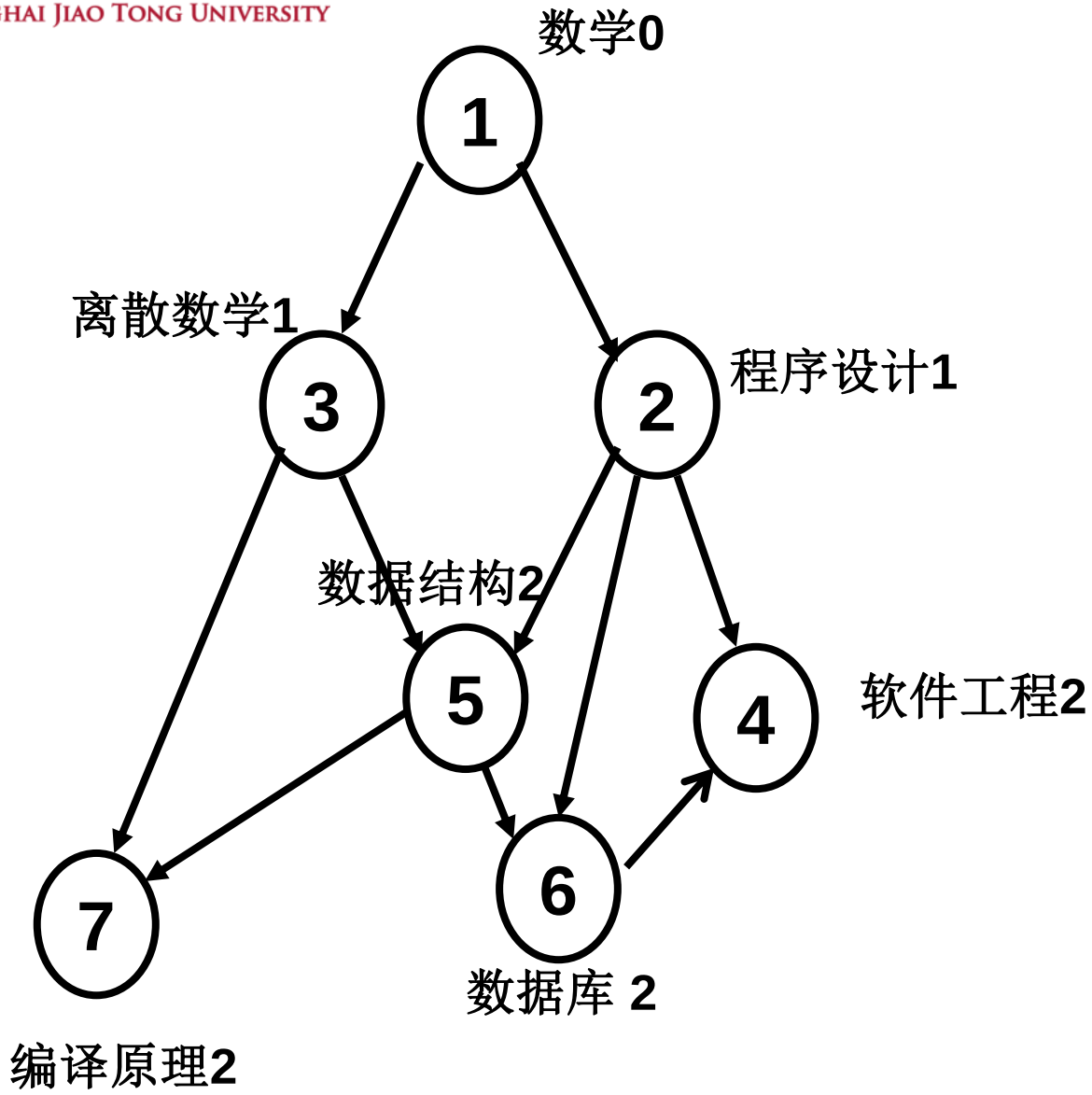
# 拓扑排序

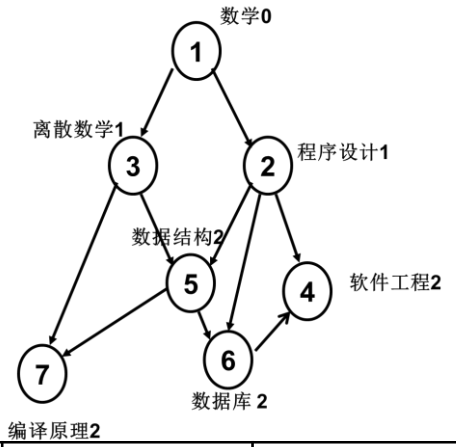
设G=(V，E)是一个具有n个顶点的**有向无环图**。V中的顶点序列$V_1$，$V_2$，…，$V_n$称为一个**拓扑序列**，**当且仅当该序列满足下列条件**：若在G中，从Vi到Vj有一条路径，则序列中**Vi必须排在Vj的前面**。

# 找出拓扑排序的过程

- 第一个输出的结点(序列中的第一个元素)：必须无前驱，即入度为0
- 后驱：必须等到它的前驱输出之后才输出。
- 无前驱及后驱的结点：任何时候都可输出。
- 逻辑删除法：当某个节点被输出后，就作为该节点被删除。所有以该节点作为前驱的所有节点的入度减1。

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY





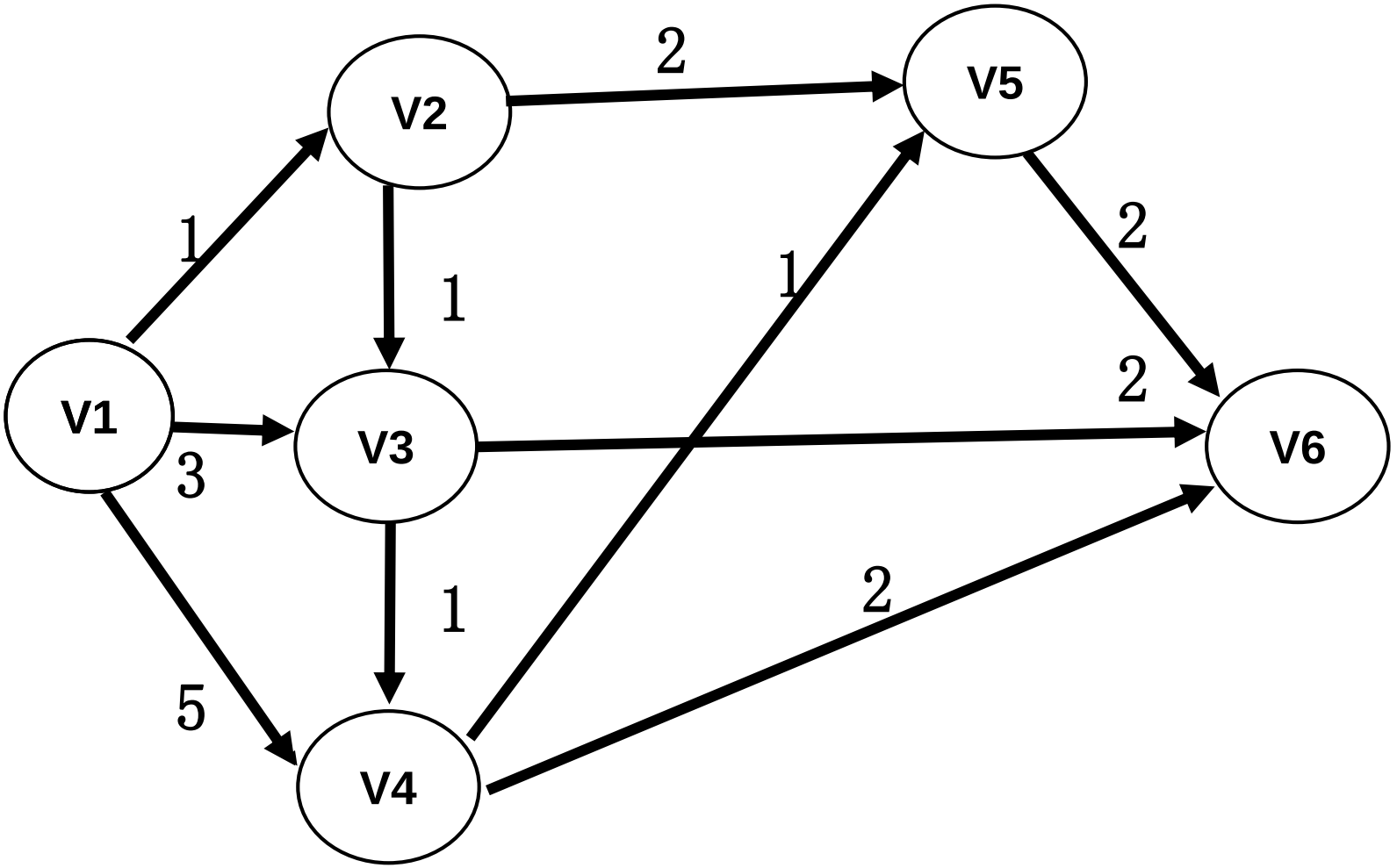

| | | | | | | |
|---|---|---|---|---|---|---|
| 数学 | 0 | | | | | |
| 离散数学 | 1 | 0 | | | | |
| 程序设计 | 1 | 0 | 0 | | | |
| 编译原理 | 2 | 2 | 1 | 1 | 0 | 0 |
| 数据结构 | 2 | 2 | 1 | 0 | | |
| 数据库 | 2 | 2 | 2 | 1 | 0 | |
| 软件工程 | 2 | 2 | 2 | 1 | 1 | 0 |

输出：数学， 离散数学， 程序设计， 数据结构， 数据库，

编译原理， 软件工程

# 关键路径

- AOE网络：**顶点表示事件，**有向边的权值表示某个活动的持续时间，有向边的方向表示事件发生的先后次序
- AOE网络可用于描述整个工程的各个活动之间的关系，活动安排的先后次序。在此基础上，可以用来估算工程的完成时间以及那些活动是关键的活动。



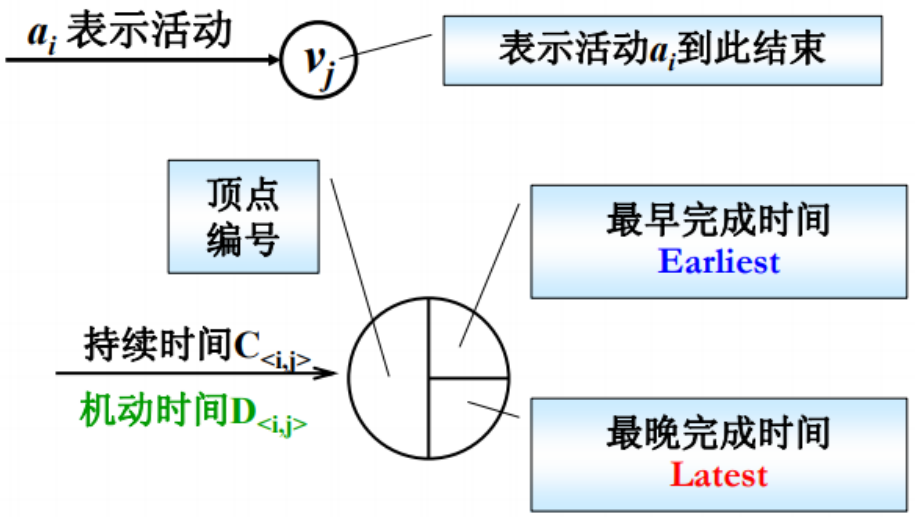

关键路径：(v1,v4,v5,v6)，路径长度是8

(v2，v3)的活动是非关键活动

# 找关键路径过程

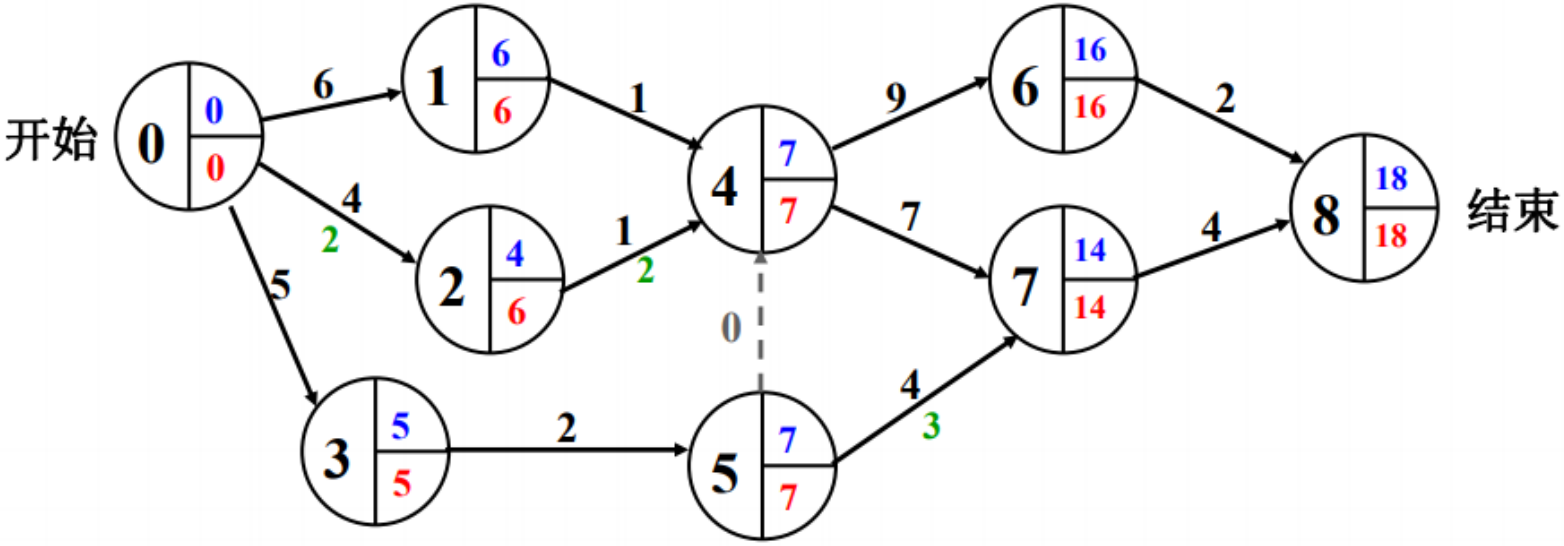
- 找出每个顶点的最早发生时间和最迟发生时间
  - 最早发生时间：每个直接前驱的最早发生时间加上从该前驱到该顶点的活动时间的最大者
  - 最迟发生时间：每个直接后继的最迟发生时间减去顶点到该直接后继的活动时间的最小者就是该顶点的最迟发生时间。
- 找出两个时间相等的顶点就是关键路径上的顶点

# 一般用于安排项目的工序



# 关键路径问题

由绝对不允许延误的活动组成的路径




问题1：整个工期有多长？ Earliest[8] = 18

Earliest[0] = 0;

Earliest[j] = $\max_{<i,j>\in E}$ { Earliest[i]+$C_{<i,j>}$ };

问题2：哪几个组有机动时间？ $D_{<i,j>}$= Latest[j]-Earliest[i]-$C_{<i,j>}$

Latest[8] = 18;

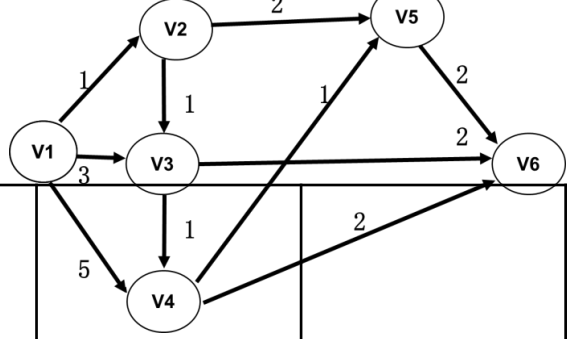
Latest[i] = $\min_{<i,j>\in E}$ { Latest[j]-$C_{<i,j>}$ };

# 算法总结

- 找出拓扑序列
- 从头到尾遍历拓扑序列找出结点最早发生时间
- 从尾到头遍历拓扑序列找到最迟发生时间
- 从头到尾遍历拓扑序列，找出最早发生时间和最迟发生时间的顶点，组成了关键路径。

# 最早发生时间

- 设结点x的最早发生时间记为ee(x)，边<u,v>的长度记为$L_{uv}$。
- 首先设所有结点的最早发生时间是0。
- 对每个被遍历的结点u检查它的后继v。如果$ee(u)+L_{uv} > ee(v)$，则更新ee(v)为$ee(u)+L_{uv}$。




| 得到最早发生时间的顶点 | ee(v1) | ee(v2) | ee(v3) | ee(v4) | ee(v5) | ee(v6) |
|---|---|---|---|---|---|---|
| 初值 | 0 | 0 | 0 | 0 | 0 | 0 |
| V1 | 0 | 1 | 3 | 5 | 0 | 0 |
| V2 | | 1 | 3 | 5 | 3 | 0 |
| V3 | | | 3 | 5 | 3 | 5 |
| V4 | | | | 5 | 6 | 7 |
| V5 | | | | | 6 | 8 |
| V6 | | | | | | 8 |

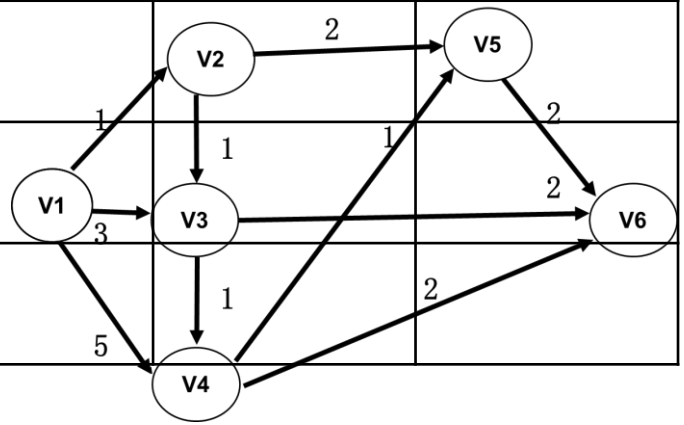
上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

# 最迟发生时间

- 设结点x的最迟发生时间记为le(x)
- 首先设所有结点的最迟发生时间是关键路径的长度
- 对每个被遍历的结点u检查它的后继v。如果 $le(v) - L_{uv} < le(u)$，则更新le(u)为$le(v)-L_{uv}$。

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

| 得到最迟发生时间的顶点 | le(v1) | le(v2) | le(v3) | le(v4) | le(v5) | le(v6) |
|---|---|---|---|---|---|---|
| 初值 | 8 | 8 | 8 | 8 | 8 | 8 |
| V6 | 8 | 8 | 8 | 8 | 8 | 8 |
| V5 | 8 | 8 | 8 | 8 | 6 | |
| V4 | 8 | 8 | 8 | 5 | | |
| V3 | 8 | 8 | 4 | | | |
| V2 | 8 | 3 | | | | |
| V1 | 0 | | | | | |



1. 下列关于图的遍历的说法不正确的是（ ）

A. 连通图的深度优先搜索是一个递归过程

B. 图的广度优先搜索中邻接点的寻找具有“先进先出”的特征

C. 非连通图不能用深度优先搜索算法

D. 图的遍历要求每一顶点仅被访问一次

答案：C

2. 已知有向图G=(V, E)， G的拓扑排序为（ ）。其中V={V1, V2, V3, V4, V5, V6, V7}, E={<V1, V2>, <V1, V3>, <V1, V4>, <V2, V5>, <V3, V5>, <V3, V6>, <V4, V6>, <V5, V7>, <V6, V7>}

答案：A

A. V1, V3, V4, V6, V2, V5, V7

B. V1, V3, V2, V6, V4, V5, V7

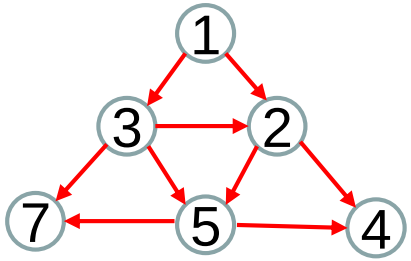
C. V1, V3, V4, V5, V2, V6, V7

D. V1, V2, V5, V3, V4, V6, V7

3. 简述如何利用邻接表和邻接矩阵判断有向图是否存在环

1. 进行拓扑排序。方法是重复寻找一个入度为0的顶点，将该顶点从图中删除并将该结点及其所有的出边从图中删除，最终若图中没有入度为0，但还有结点没有被访问，则这些点至少组成一个回路。

4. 在下面有向图的右边，写出所有的拓扑排序序列



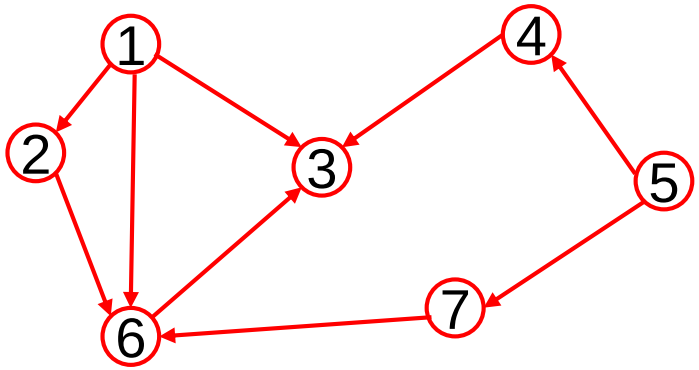

5. 四组含有1-7结点序列中，_____是下列有向图的拓扑排序序列

A.1， 2， 6， 7， 5， 4， 3

B.1， 2， 6， 3， 4， 5， 7

C.4， 1， 2， 3， 5， 6， 7

D.5， 7， 4， 1， 2， 6， 3




答案：D

6. 在有向图G的拓扑序列中，若顶点Vi在顶点Vj之前，则下列情况不可能出现的是_____

A.G中有弧<Vi,Vj>

B. G中有一条从Vi到Vj的路径

C. G中没有弧<Vi,Vj>

D. G中有一条从Vj到Vi的路径

答案：D

7. 对某个无向图的邻接矩阵来说，下列说法错误的是_____

答案：B

A.第i行上非零元素个数和第i列上非零元素个数一定相等

B.任一行上的元素不可能全都是0

C.矩阵中非零元素个数等于图中边数的2倍

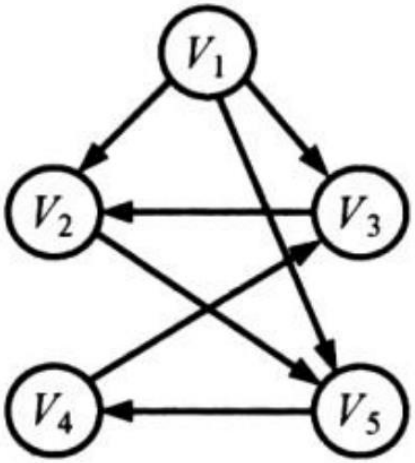
D.矩阵的列数等于图中的结点数

8.已知无向图G含有16条边，其中度为4的顶点个数为3，度为3的顶点个数为4，其他顶点 的度均小于3。图G所含的顶点个数至少是()

A．10　　B．11　　C．13　　D．15

参考答案：B

9.下列选项中，不是下图深度优先搜索序列的是()




A．V1，V5，V4，V3，V2
B．V1，V3，V2，V5，V4
C．V1，V2，V5，V4，V3
D．V1，V2，V3，V4，V5

参考答案：D

9.若将n个顶点e条弧的有向图采用邻接表存储，则拓扑排序算法的时间复杂度是()

A．O(n)　　B．O(n+e)　　C．O(n2 )　　D．O(n×e)

参考答案：B

10．设有向图G=(V，E)，顶点集V={v0，v1，v2，v3}，边集E：{<v0,v1>,<v0,v2>,<v0,v3>,<v1,v3>}。若从顶点v0。开始对图进行深度优先遍历，则可能得到的不同遍历序列个数 是 ()

A．2 B．3 C．4 D．5

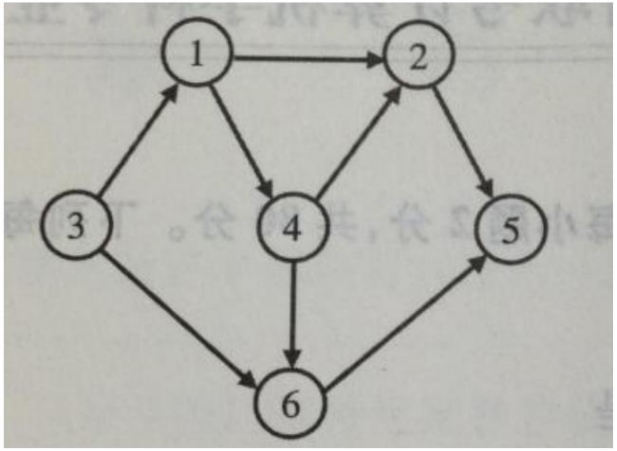
参考答案：D

11.对如下所示的有向图进行拓扑排序，得到的拓扑序列可能是 。

A．3,1,2,4,5,6　　B．3,1,2,4,6,5

C．3,1,4,2,5,6　　D．3,1,4,2,6,5

参考答案：D




12.设图的邻接矩阵 A 如下所示。各顶点的度依次是()

A. 1，2，1，2
B. 2，2，1，1
C. 3，4，2，3
D. 4，4，2，2

$$A=\begin{bmatrix}0&1&0&1\\0&0&1&1\\0&1&0&0\\1&0&0&0\end{bmatrix}$$

参考答案：C

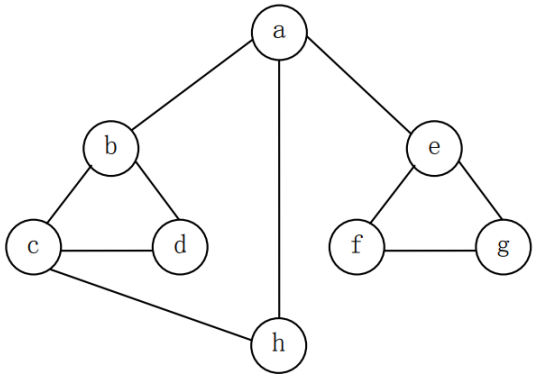
12.若对如下无向图进行遍历，则下列选项中，不是广度优先遍历序列的是（）

A. h，c，a，b，d，e，g，f

B. e，a，f，g，b，h，c，d

C. d，b，c，a，h，e，f，g

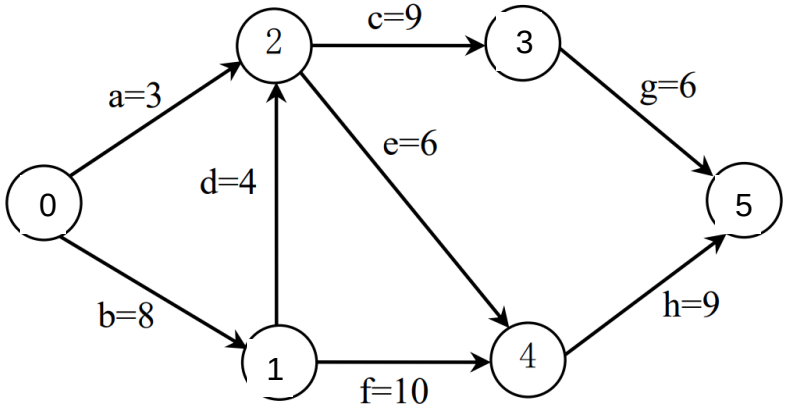
D. a，b，c，d，h，e，f，g




参考答案：D

13. 下列 AOE 网表示一项包含 8 个活动的工程。通过同时加快若干活动的进度可以缩短整个工 程的工期。下列选项中，加快其进度就可以缩短工程工期的是（）

A. c 和 e
B. d 和 e
C. f 和 d
D. f 和 h




参考答案：C