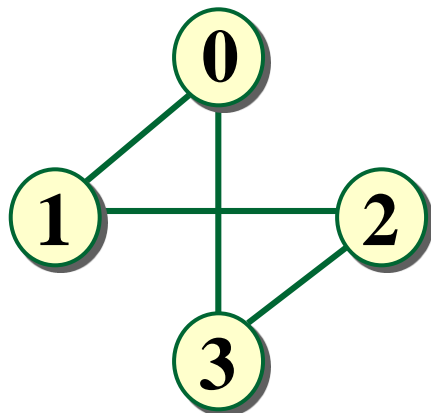


主题三 图及其应用

- 图的2种存储表示
- 图的遍历
- 最小生成树
- 最短路径
- 活动网络

邻接矩阵 (Adjacency Matrix)



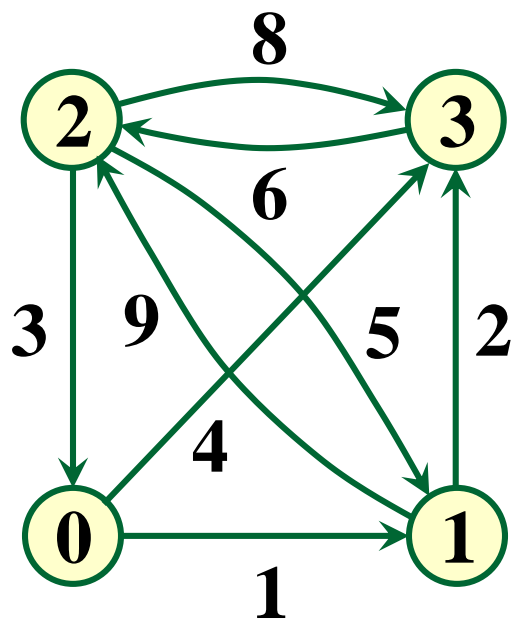
$$\mathbf{A.\text{edge}} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$



$$\mathbf{A.\text{edge}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

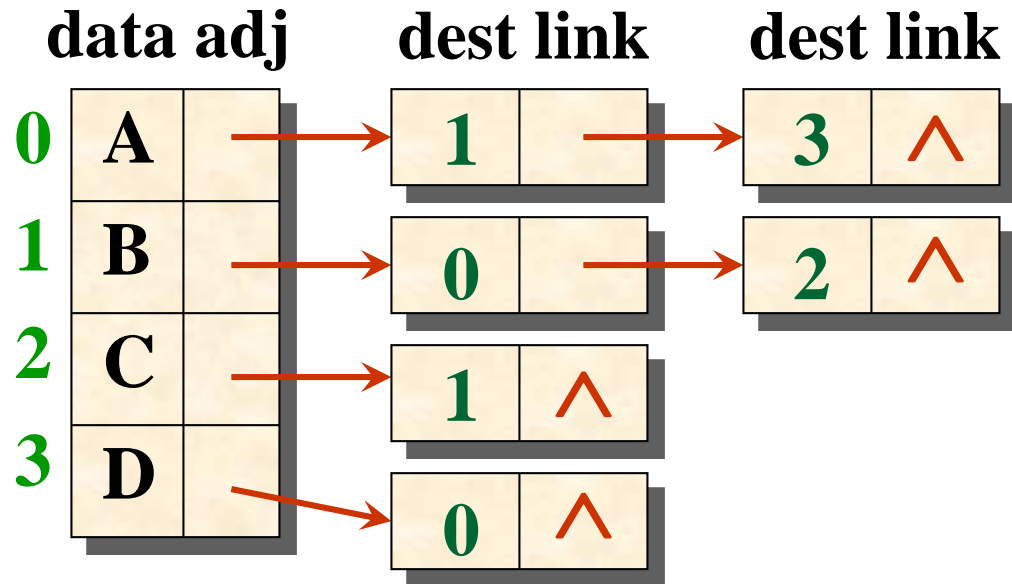
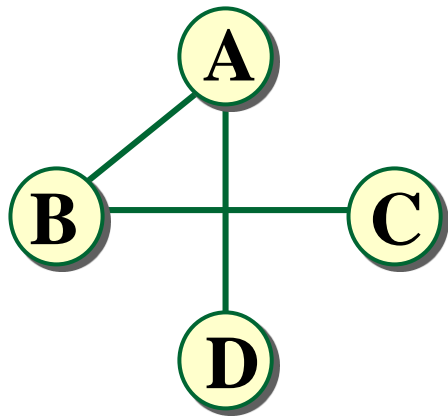
- 无向图的邻接矩阵是对称的;
- 有向图的邻接矩阵可能是不对称的。

网络(带权图)的邻接矩阵



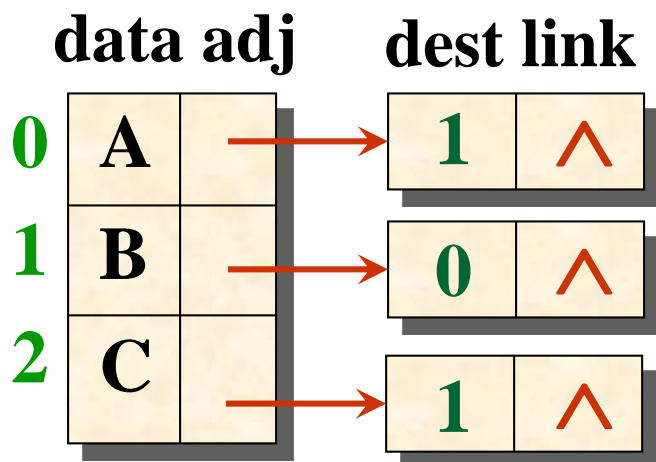
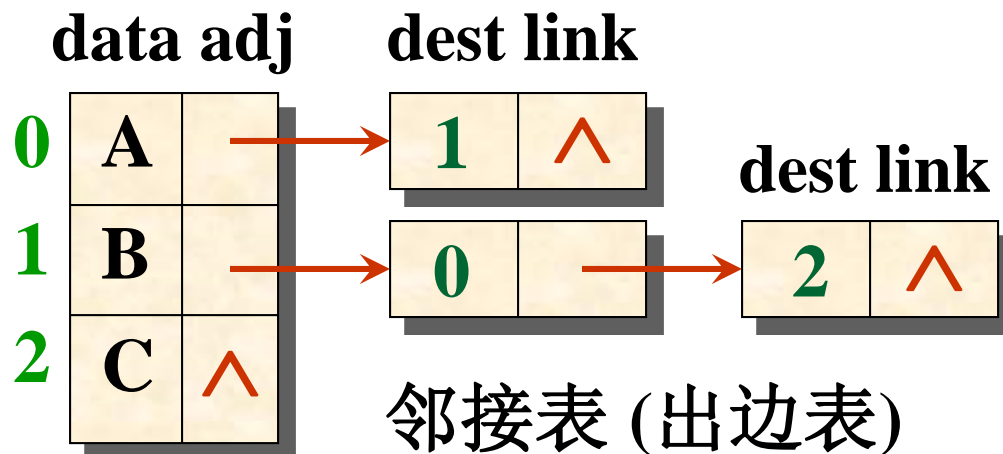
$$\mathbf{A.edge} = \begin{bmatrix} 0 & 1 & \infty & 4 \\ \infty & 0 & 9 & 2 \\ 3 & 5 & 0 & 8 \\ \infty & \infty & 6 & 0 \end{bmatrix}$$

无向图的邻接表



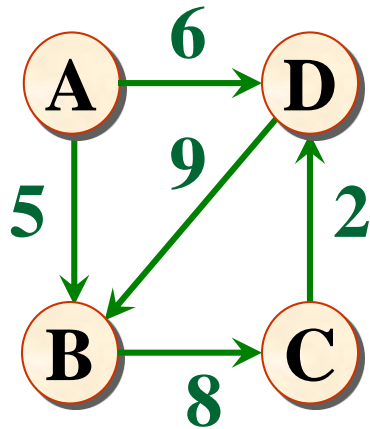
- 统计某顶点对应边链表中结点个数，可得该顶点的度。
- 某条边(v_i, v_j)在邻接表中有两个边结点，分别在第 i 个顶点和第 j 个顶点对应的边链表中。

有向图的邻接表和逆邻接表



逆邻接表 (入边表)

网络(带权图)的邻接表



	data	adj	dest	cost	link
0	A	→	1	5	→ 3 6 ^
1	B	→	2	8	^
2	C	→	3	2	^
3	D	→	1	9	^

(顶点表) (出边表)

图的遍历与连通性

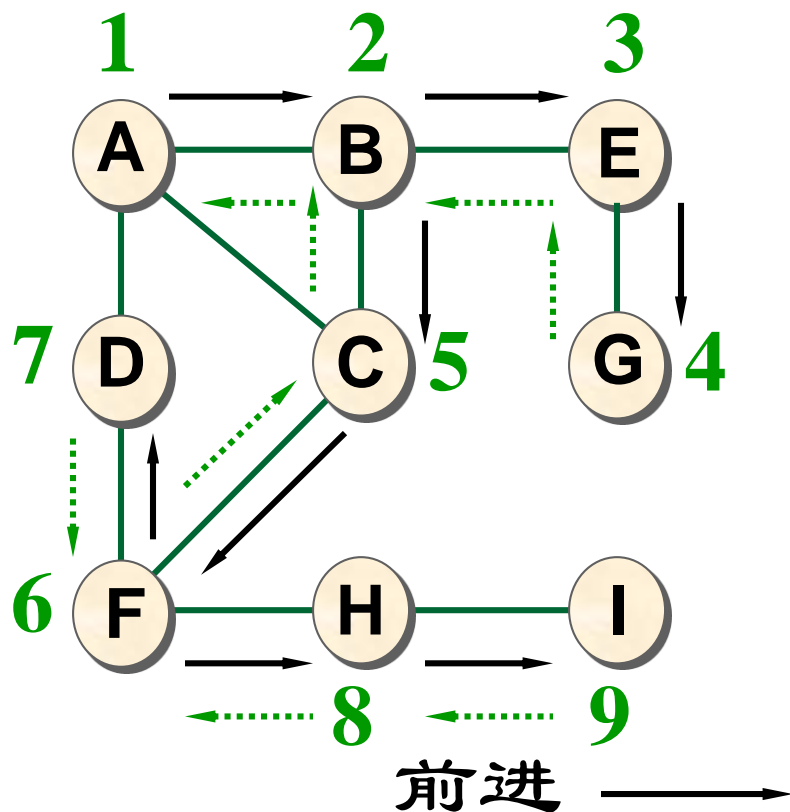
- 从已给的连通图中某一顶点出发，沿着一些边访问遍图中所有的顶点，且使每个顶点仅被访问一次，就叫做图的遍历 (Graph Traversal)。
- 图中可能存在回路，且图的任一顶点都可能与其它顶点相通，在访问完某个顶点之后可能会沿着某些边又回到了曾经访问过的顶点。
- 为了避免重复访问，可设置一个标志顶点是否被访问过的辅助数组 **visited []**。

图的遍历

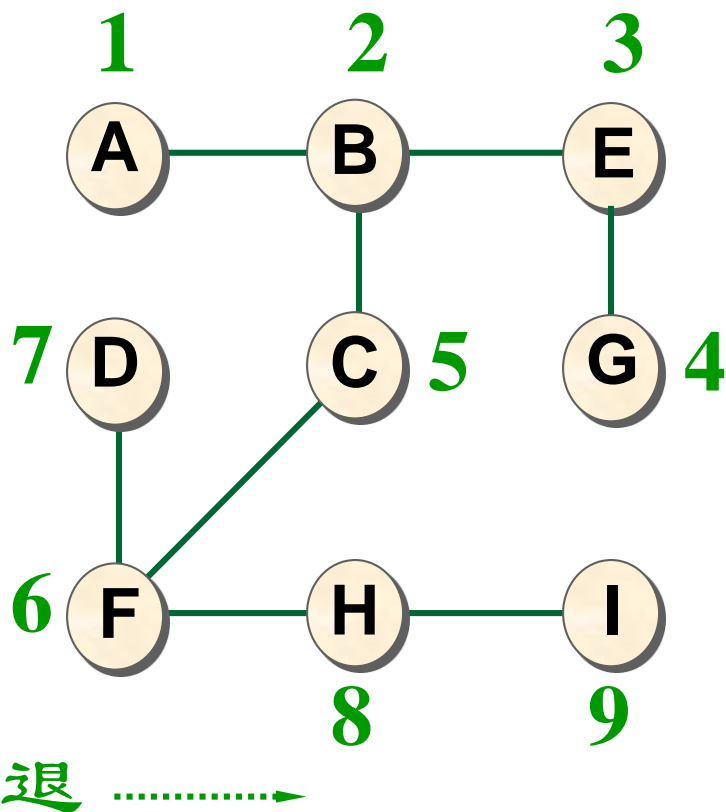
- 从已给的连通图中某一顶点出发，沿着一些边访遍图中所有的顶点，且使每个顶点仅被访问一次，就叫做图的遍历 (**Graph Traversal**)。
- 图的遍历的分类：
 - ◆ 深度优先搜索
DFS (Depth First Search)
 - ◆ 广度优先搜索
BFS (Breadth First Search)

深度优先搜索DFS (Depth First Search)

■ 深度优先搜索的示例



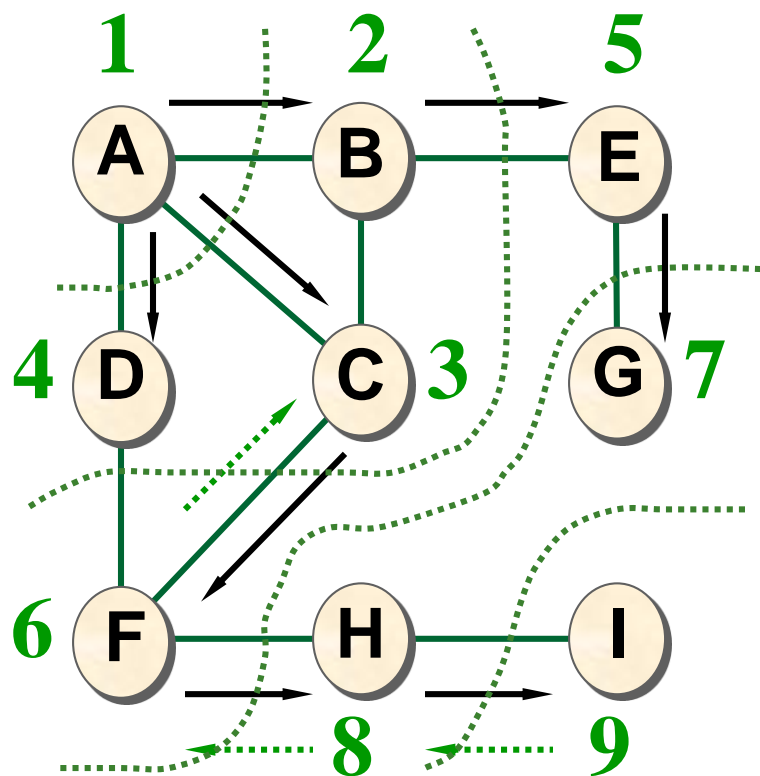
深度优先搜索过程



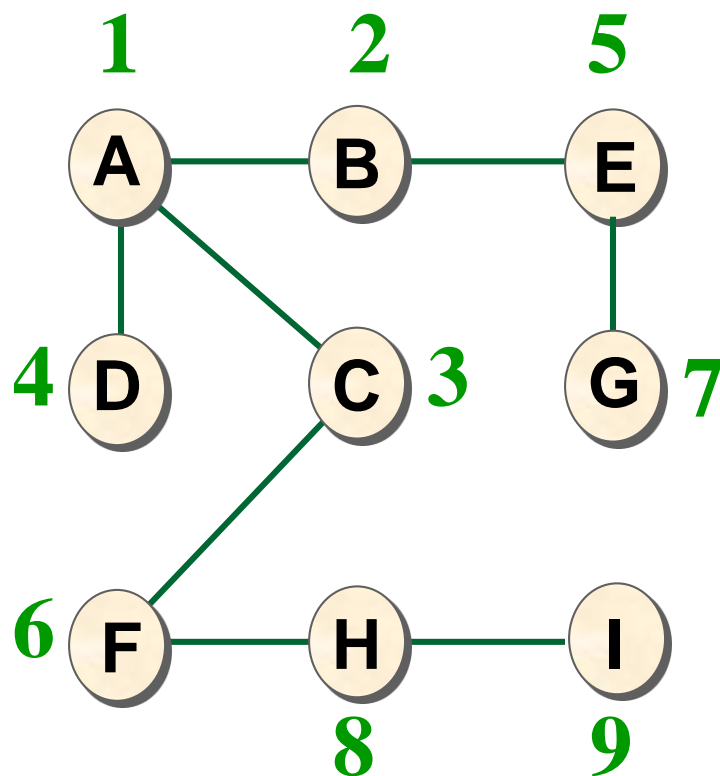
深度优先生成树

广度优先搜索BFS (Breadth First Search)

■ 广度优先搜索的示例



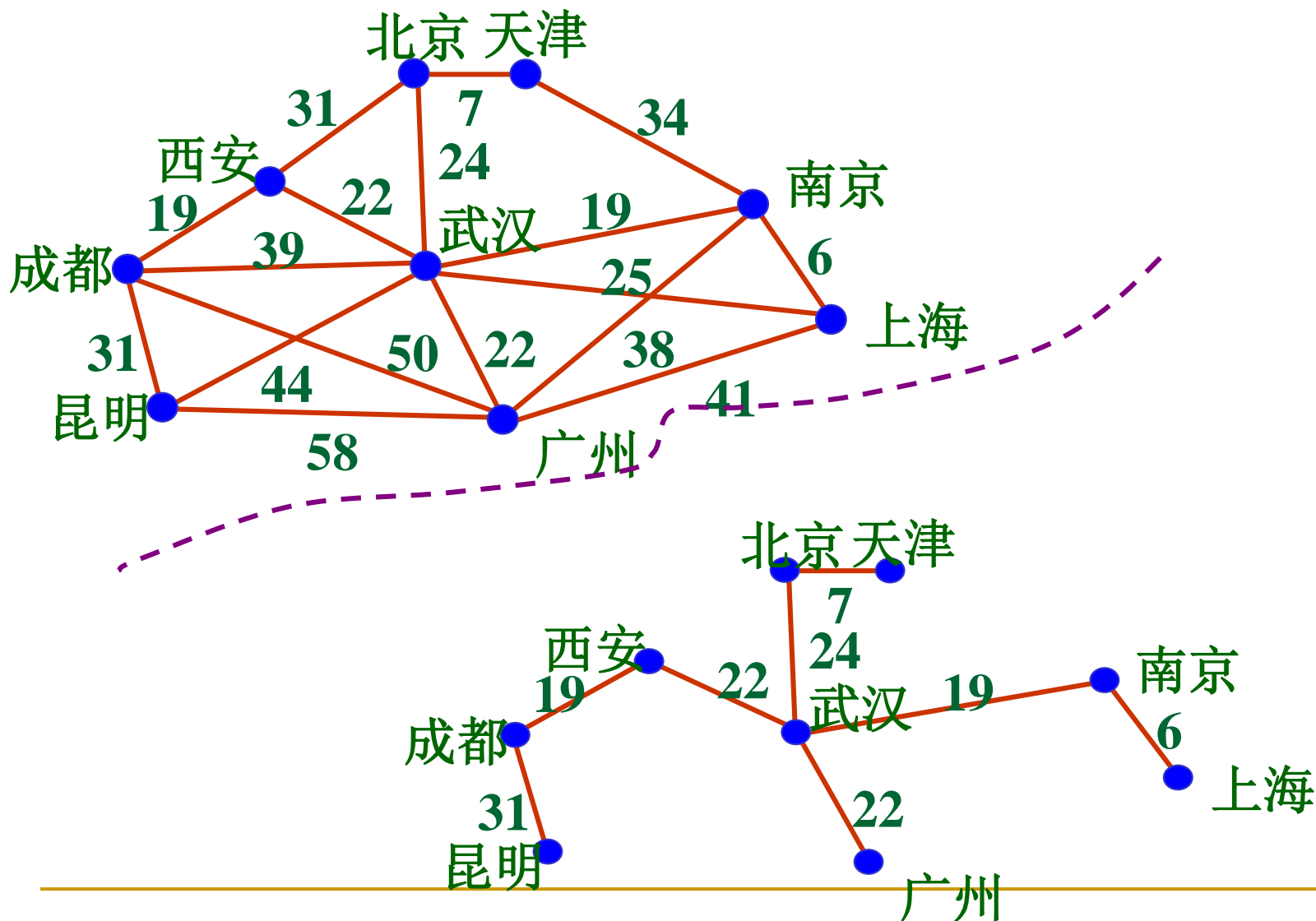
广度优先搜索过程



广度优先生成树

- **DFS**算法 在实现的时候, 使用了递归函数调用的形式。
- **BFS** 算法中使用队列以记忆正在访问的这一层和上一层的顶点, 以便于向下一层访问。
- 为避免重复访问, 需要一个辅助数组 **visited** [], 给被访问过的顶点加标记。

最小生成树

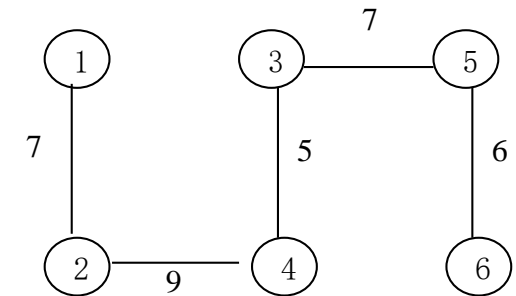
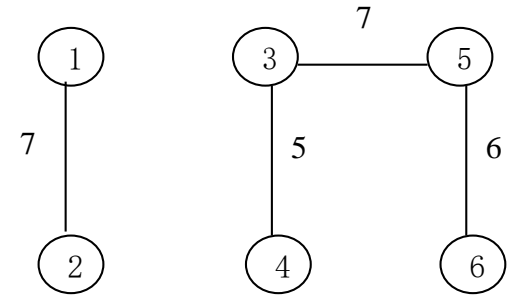
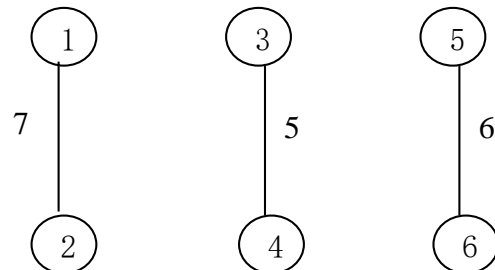
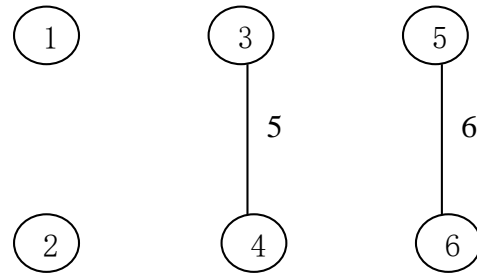
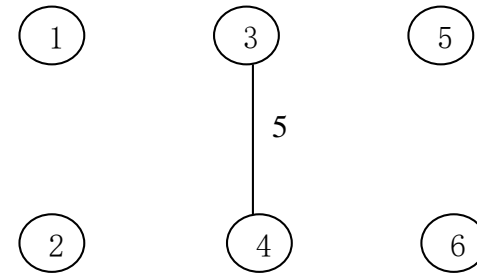
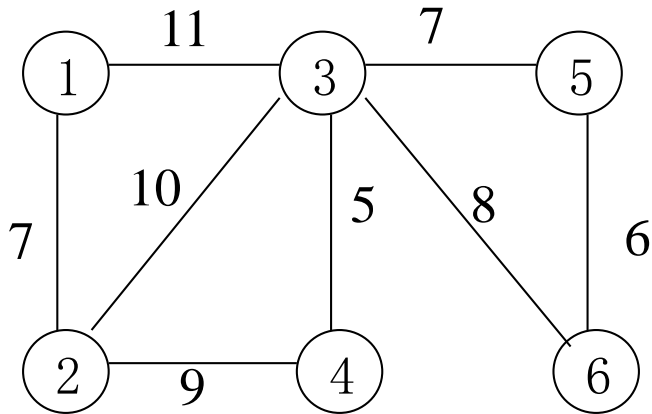


- 构造最小生成树的准则
 - 必须使用且仅使用该网络中的 $n-1$ 条边来联结网络中的 n 个顶点;
 - 不能使用产生回路的边;
 - 各边上的权值的总和达到最小。

- 两种最小生成树的算法
 - 克鲁斯卡尔 (Kruskal) 算法 (基于边, 适用于边稀疏的网络)
 - 普里姆(Prim)算法 (基于顶点, 适合于边稠密情形)

练习：请用Kruskal算法或Prim算法构造下图的最小生成树。要求画出构造的每一步：

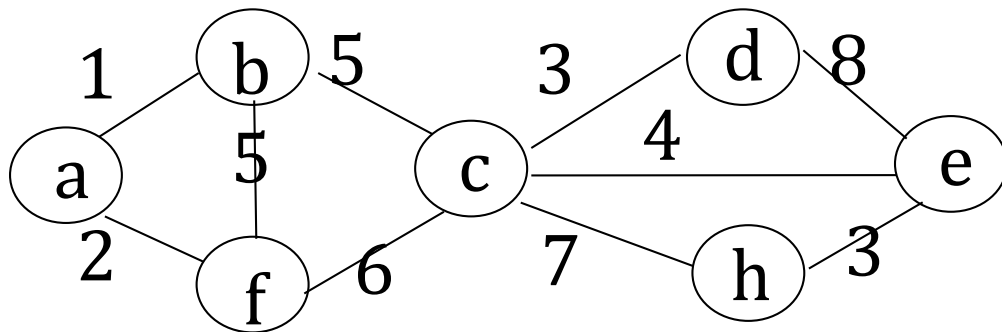
解答：Kruskal算法



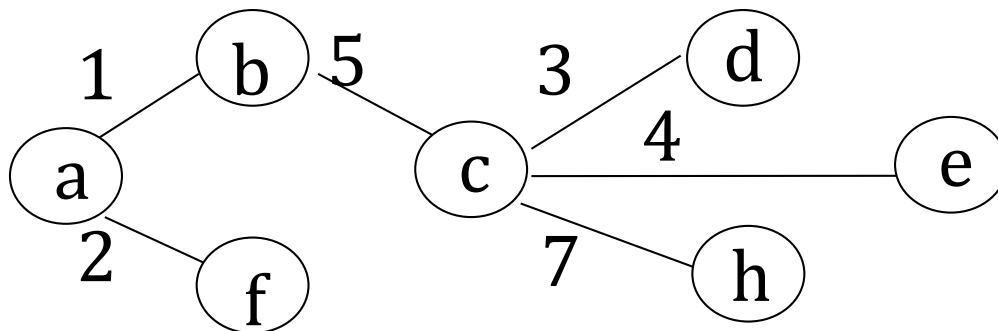
最短路径

- 最短路径问题：如果从图中某一顶点（称为源点）另一顶点（称为终点）的路径可能不止一条，如何找到一条路径使得沿此路径上各边上的权值总和达到最小。
 - ◆ 边上权值非负情形的单源最短路径问题
 - **Dijkstra**算法
 - ◆ 边上权值为任意值的单源最短路径问题
 - **Bellman**和**Ford**算法
 - ◆ 所有顶点之间的最短路径
 - **Floyd**算法

练习： 源点a到所有点的最短路径



解答：



源点a到各个点的最短路径及
路径长度：

a -> b : 1

a -> f : 2

a -> b -> c : 6

a -> b -> c -> d : 9

a -> b -> c -> e : 10

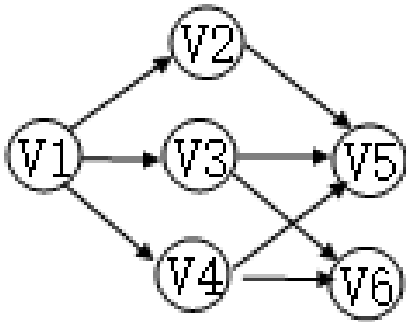
a -> b -> c -> h : 13

活动网络 (Activity Network)

- 用顶点表示活动的网络 (AOV网络)
- 用边表示活动的网络 (AOE网络)

AOV网络的应用——拓扑排序

练习：求出下图中所有的拓扑序列

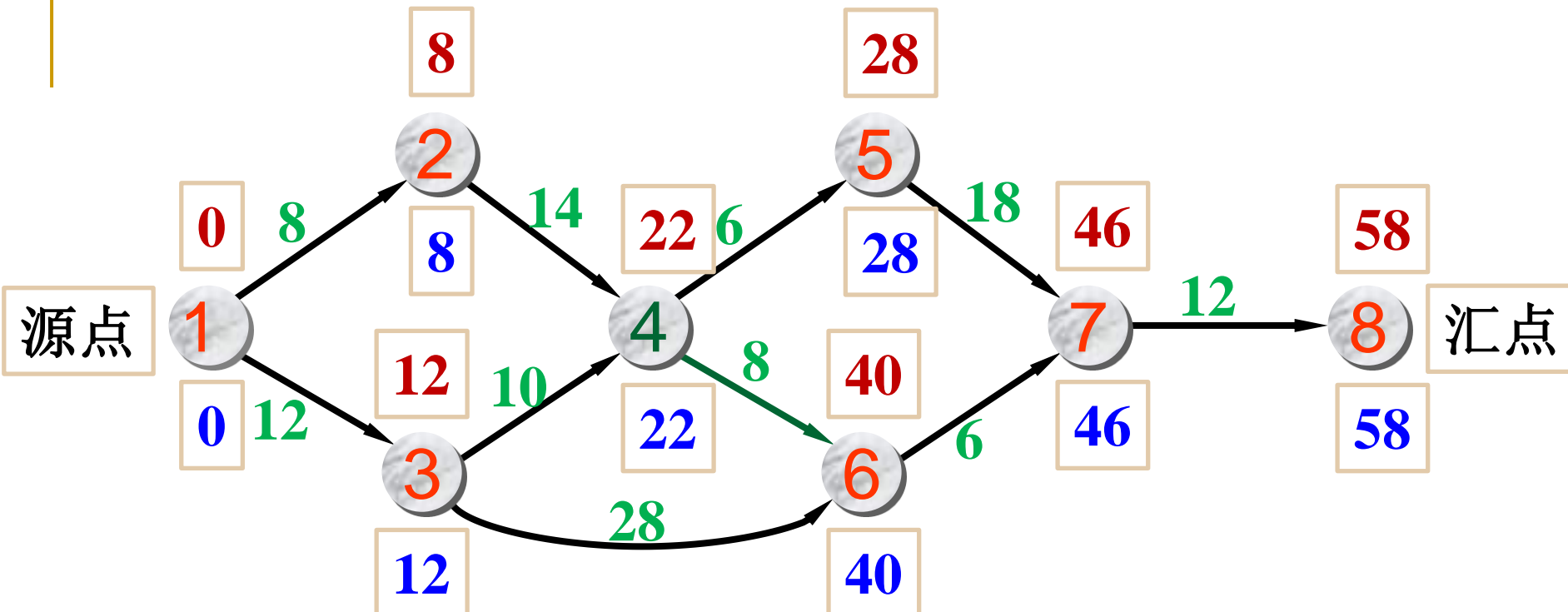



解答：


V1- V2- V3- V4- V5- V6
V1- V2- V3- V4- V6- V5
V1- V2- V4- V3- V5- V6
V1- V2- V4- V3- V6- V5
V1- V3 - V2- V4- V5- V6
V1- V3 - V2- V4- V6- V5
V1- V3 - V4- V6- V2- V5
V1- V3 - V4- V2- V5- V6
V1- V3 - V4- V2- V6- V5
V1- V4 - V3- V6- V2- V5
V1- V4 - V3- V2- V5- V6
V1- V4 - V3- V2- V6- V5
V1- V4 - V2- V3- V5- V6
V1- V4 - V2- V3- V6- V5


AOE网络的应用——工程估算


- ◆ 完成整个工程至少需要多少时间(假设网络中没有环)?
 - 从源点到汇点的最长路径长度
- ◆ 为缩短完成工程所需的时间,应当加快哪些活动?
 - 关键路径上的所有活动,即关键活动
- ◆ 从源点到汇点的有向路径可能不止一条。这些路径的长度也可能不同。完成不同路径的活动所需的时间虽然不同,但只有各条路径上所有活动都完成了,整个工程才算完成。



 **事件**

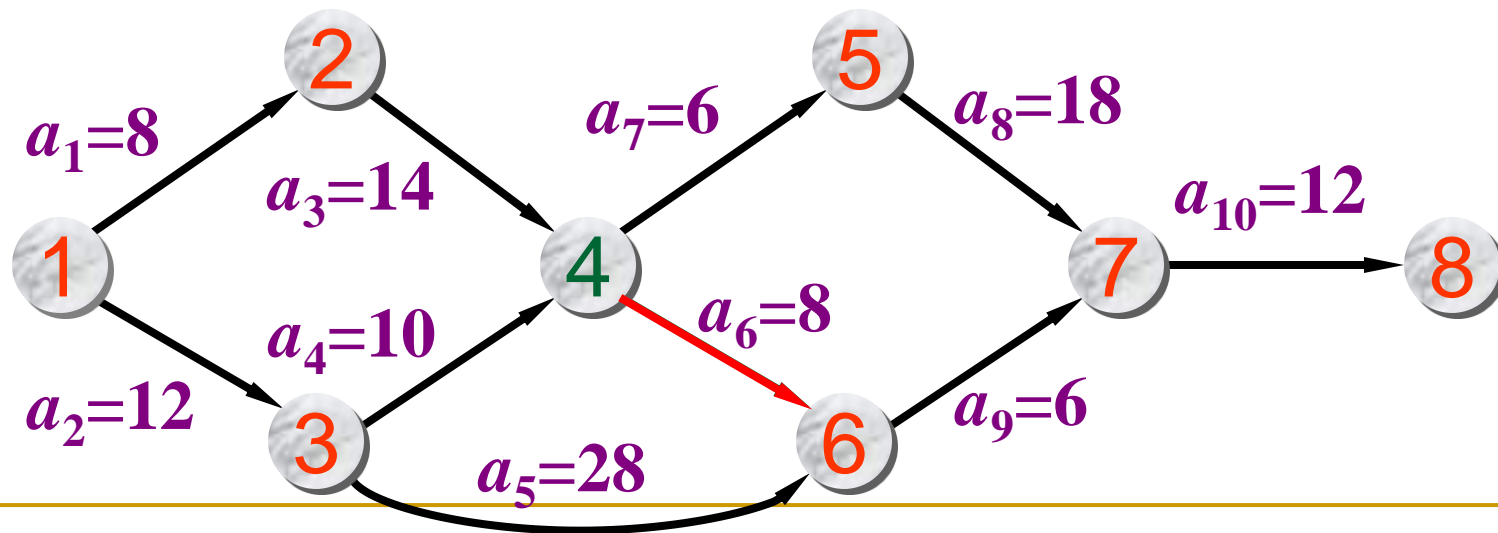
 **活动**

 **事件最早发生时间**

 **事件最晚发生时间**

		1	2	3	4	5	6	7	8
事件最早	Ve	0	8	12	22	28	40	46	58
事件最晚	Vl	0	8	12	22	28	40	46	58

		1	2	3	4	5	6	7	8	9	10
活动最早	e	0	0	8	12	12	22	22	28	40	46
活动最晚	l	0	0	8	12	12	32	22	28	40	46



下列关于AOE网的叙述中，不正确的是_____。

(1)关键活动不按期完成就会影响整个工程的完成时间

(2)任何一个关键活动提前完成，那么整个工程将会提前完成

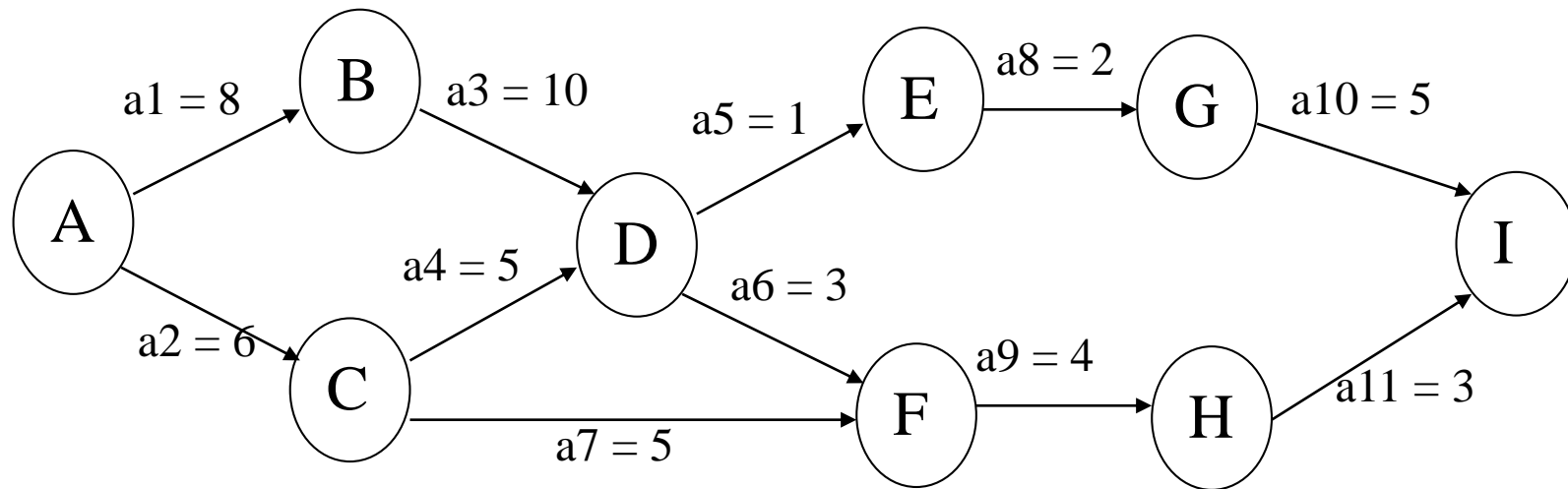
(3)所有的关键活动提前完成，那么整个工程将会提前完成

(4)在任何一个AOE网中，关键路径只有一条

A. (1)(2) B. (2) (3) C. (3)(4) D.(2)(4)

答案： D

•练习：给定如下的AOE网络，求出该网络中的关键路径和关键活动。

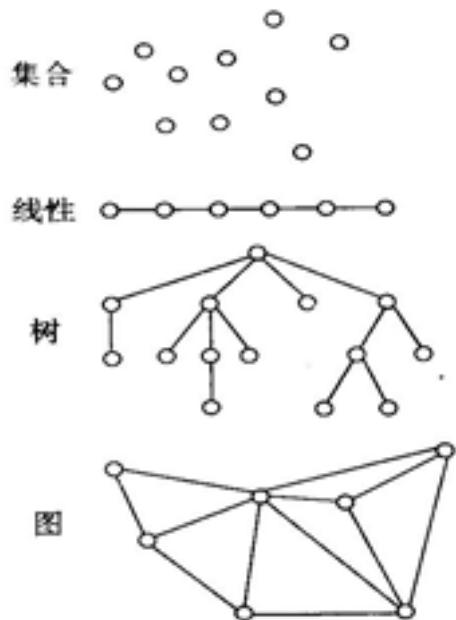
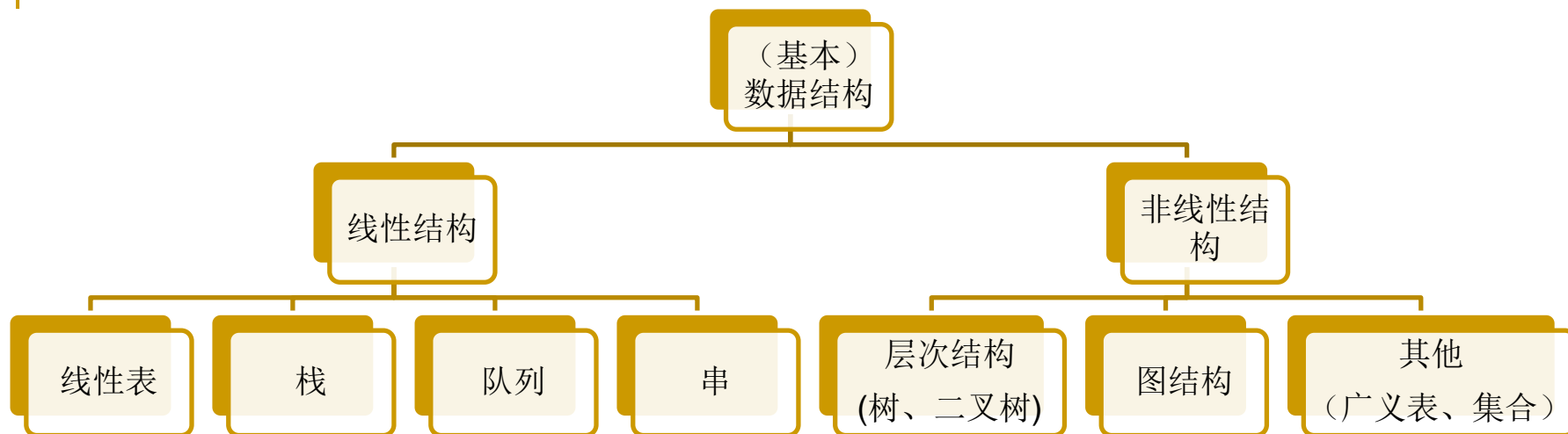


解答：关键路径为 A->B->D->F->H->I
关键活动为：a1, a3, a6, a9, a11

总结

- 数据结构主要关注那些方面问题？
- 算法的是如何进行性能分析与度量？
- 拿到问题后怎么分析？设计数据结构和算法？
- 如何提高自己的Programming的能力？
- 如何为成为算法工程师做准备？

数据的组织



集合结构：仅同属一个集合

线性结构：一对一 (1:1) → 线性

树结构：一对多 (1:n)

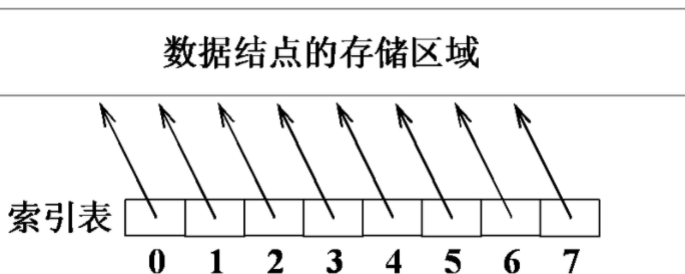
图结构：多对多 (m:n)

} 非线性

数据的存储和运算



- ✓ 索引方法：构造一个由整数域 Z 映射到存储地址域 D 的函数 $Y: Z \rightarrow D$

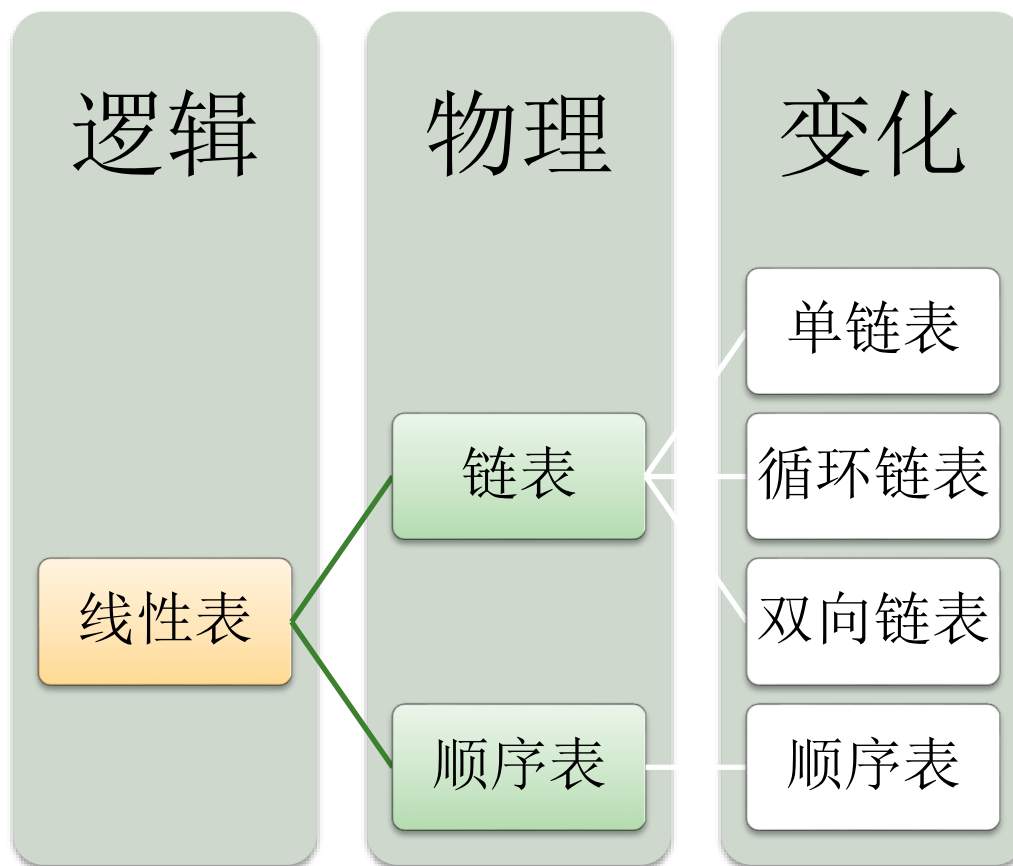


- ✓ 散列方法：通过构造散列函数 $H(k)=D$ 进行关键码 \rightarrow 地址的计算。

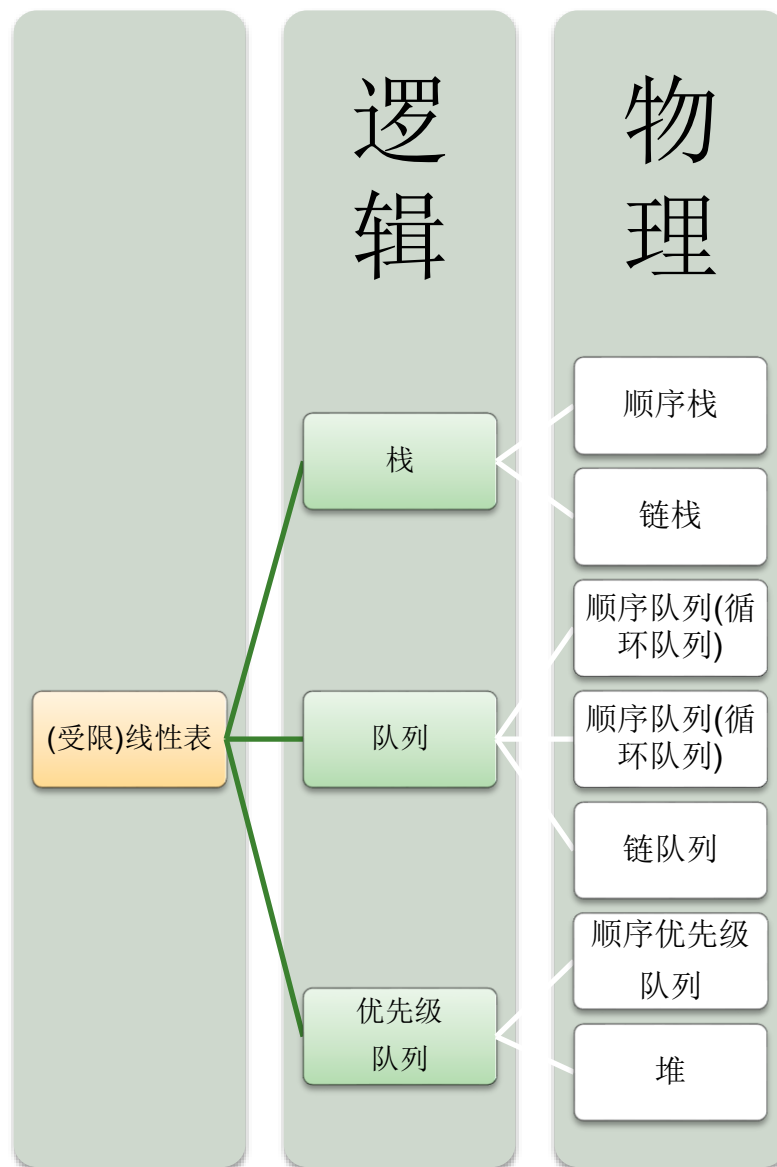
散列函数 $H(k) = k \% 5$

0	1	2	3	4
15		22	3	24

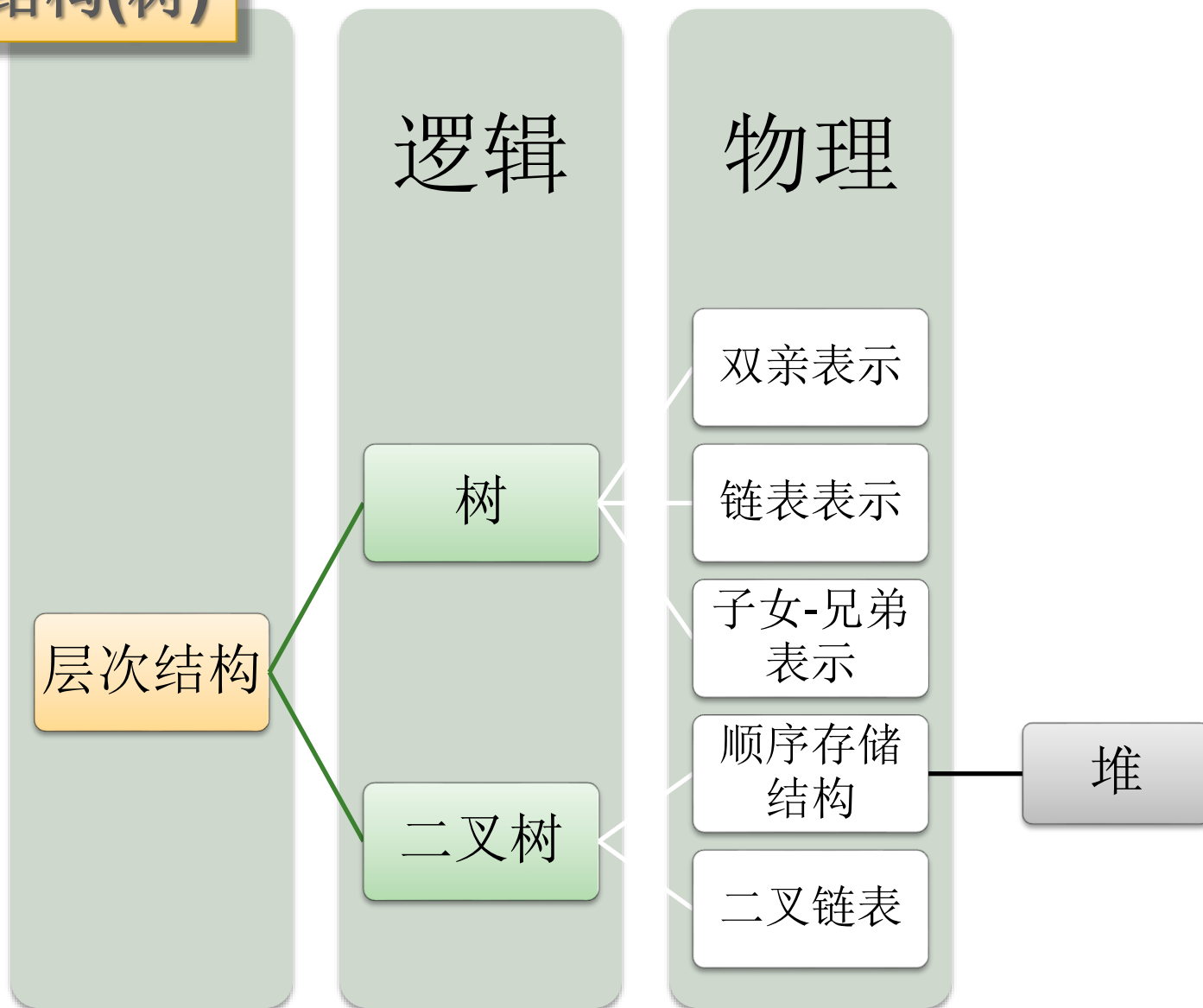
线性结构



线性结构



层次结构(树)



其他非线性结构

