**图（graph）**



问题1：百度地图那么多地址，那么多路线，好复杂呀！采用了什么数据结构

问题2：百度地图的各个地点、线路信息是如何存储的

问题3：百度地图是如何计算出两个地点之间的合理路径的（距离短、拥堵少、红绿灯少）

问题1答案：采用的数据结构是图

问题2答案：图的存储结构，有多种

问题3答案：涉及最短路径问题

**图（graph）**是一种网状数据结构，图是由**非空的顶点集合**和**一个描述顶点之间关系**的**集合**组成

其形式化的定义如下∶

Graph=(V,E)

V={x|x∈某个数据对象}

E={<u，v>|P（u，v）^（u，v∈V）}

V是**具有相同特性的数据元素的集合**，V中的数据元素通常称为顶点（Vertex）E是**两个顶点之间关系的集合**，P（u，v）**表示u和v之间有特定的关联属性**

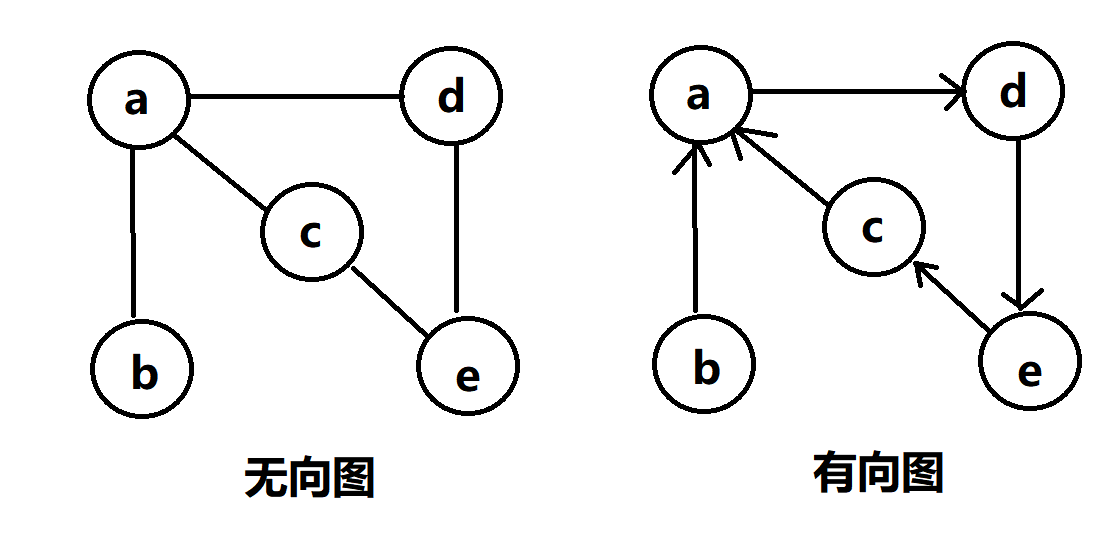
若<u，v>∈E，则<u，v>表示从顶点u到顶点v的一条弧，并称u为弧尾或起始点，称v为弧头或终止点，此时**图中的顶点之间的连线是有方向的，这样的图称为有向图（directedgraph））**

若<u，v>∈E则必有<v，u>∈E，即关系E是对称的，此时可以使用一个无序对（u，v）来代替两个有序对，

它表示顶点 u和顶点v之间的一条边，此时**图中顶点之间的连线是没有方向的，这种图称为无向图（undirected graph）**

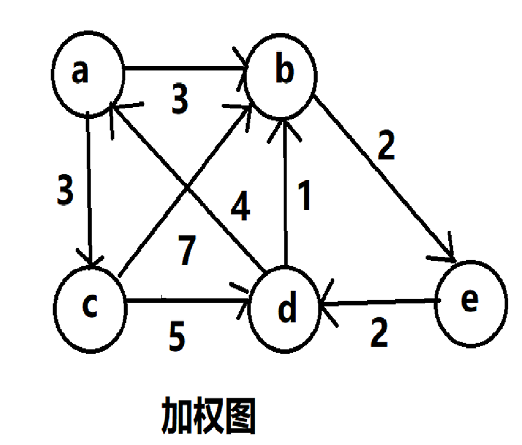
在无向图和有向图中V中的元素都称为**顶点**，而顶点之间的关系却有不同的称谓，即弧或边，为避免麻频，在不影响理解的前提下，我们统一的将它们称为**边（edge ）**

并且我们还约定顶点集与边集都是有限的，并记顶点与边的数量为|V|和|E|



**无向图实际上也是有向图，是双向图**

加权图



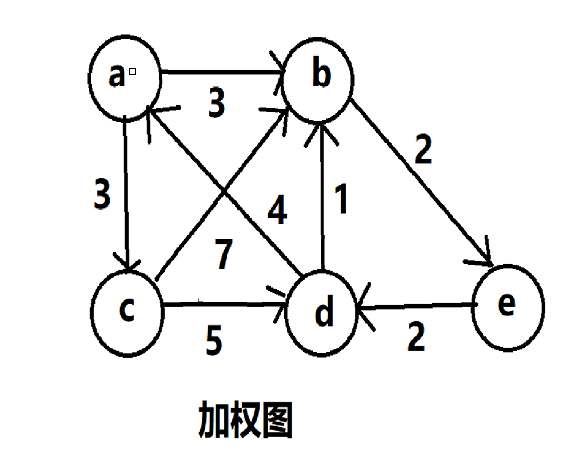
在实际应用中，**图不但需要表示元素之间是否存在某种关系**

而且**图的边往往与具有一定实际意义的数有关**，即每条边都有与它相关的实数，称为**权**

这些权值可以表示从一个顶点到另一个顶点的距离或消耗（时间）等信息

在本章中假设边的权均为正数（不考虑权值<0的情况）

这种边上具有权值的图称为 **带权图（weighted graph）**



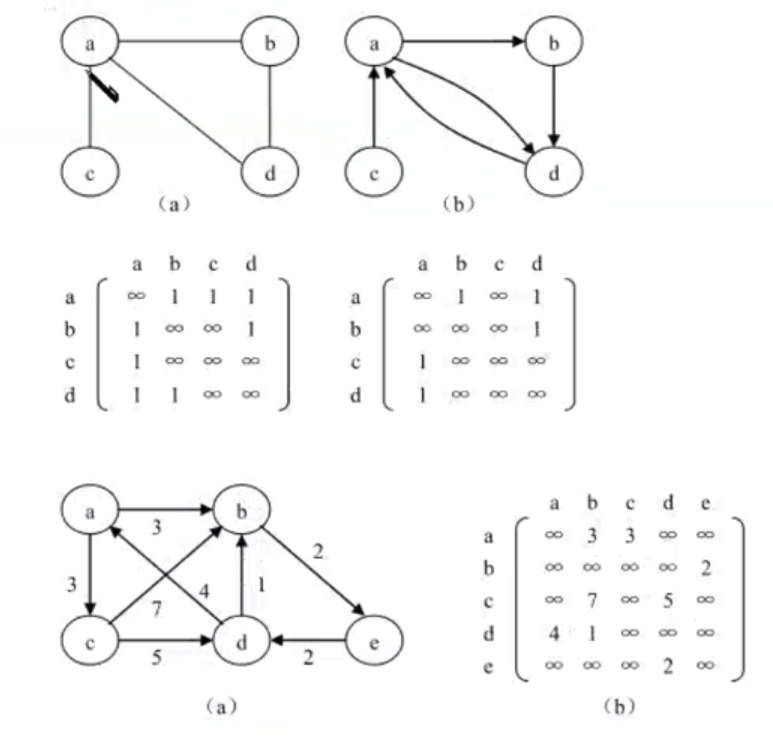
**图的存储**

方式1：邻接矩阵：二维数组  顺序存储

无向图的邻接矩阵的顺序表结构的特征：对称性

有向图的邻接矩阵的顺序表结构的特征：不对称（不是双向的）

加权有向图的邻接矩阵的顺序表结构的特征：不对称并携带具体权值



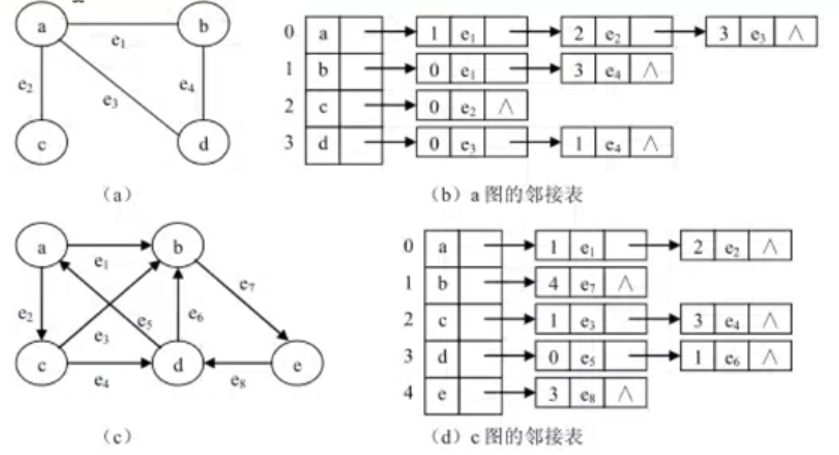
方式2：邻接表：链表 链式存储

链式存储图内容：数组+链表

结点（Node）:索引值+权值+next值

有向图：数组+链表（当前顶点指向的所有顶点）

无向图：数组+链表（当前顶点相邻的所有顶点）



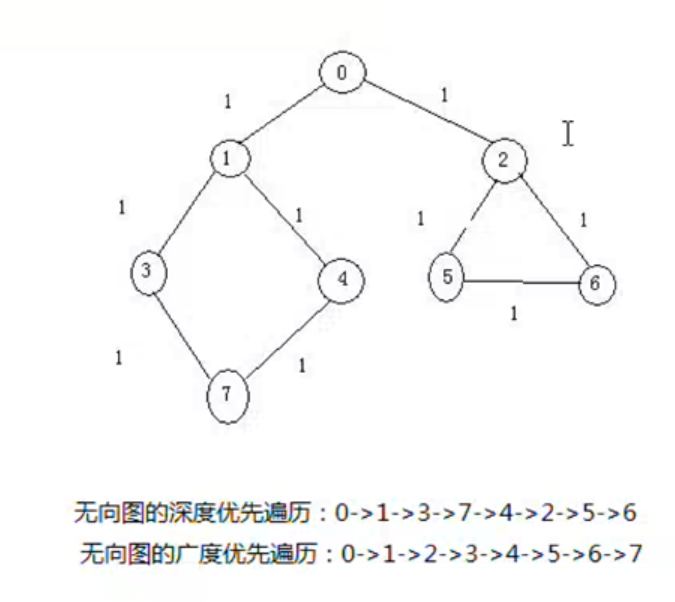
**图的遍历**

**图的遍历就是从图中某个顶点出发**，**按某种方法对图中所有顶点访问且仅访问一次**

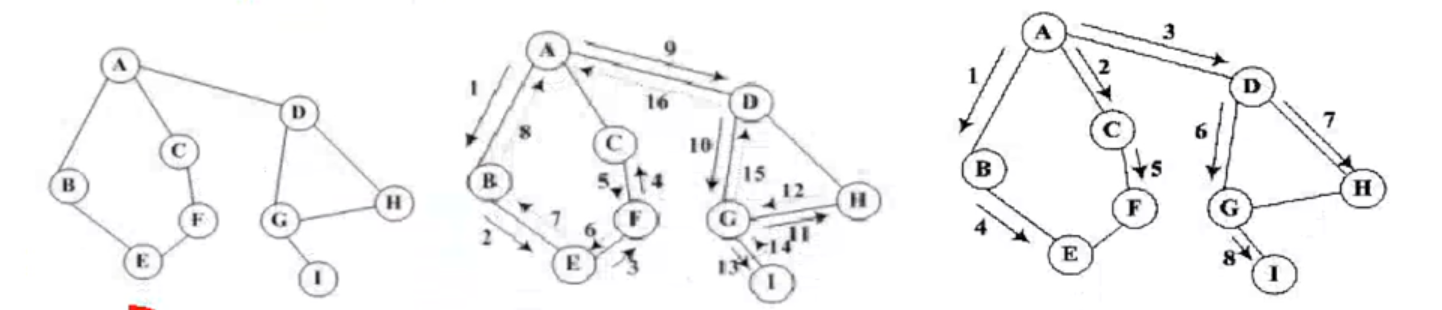
**图的遍历算法**是求解图的连通性问题、拓扑排序和求关键路径等算法的基础

**深度优先遍历（DFS depth-first search）**∶类似于树的先根遍历，是树的先根遍历的推广（可以采用递归和借助栈的非递归方式实现）

**广度优先遍历（BFS breadth-first search）**∶遍历类似于树的层次遍历，它是树的按层追历的推广（借助队列 非递归方式实现）



例：



深度优先遍历：a b e f c d g h i

广度优先遍历：a b c d e f g h i

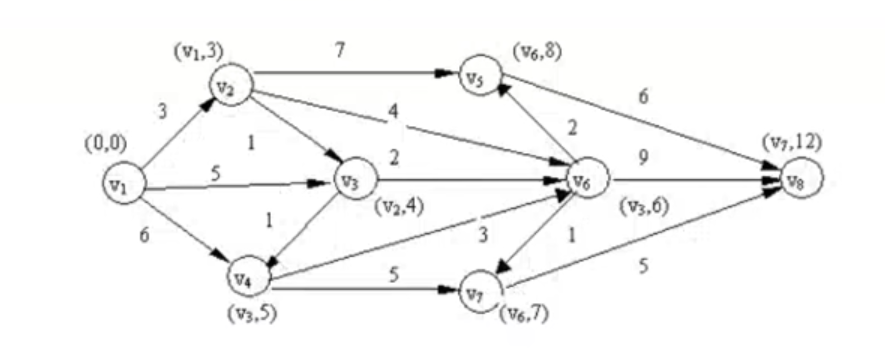
**最短路径**

在许多应用领域，带权图都被用来描述某个网络，比如通信网络、交通网络等），这种情况下，各边的权重就对应于两点之间通信的成本或交通费用

此时，一类典型的问题就是∶在任意指定的两点之间如果存在通路，那么最小的消耗是多少

这类问题实际上就是带权图中两点之间最短路径的问题

问题：计算V1到V8的最短路径



**最短路径1**∶段数最少的最短路径：

**生活案例∶换乘最少**

**解决方案∶使用广度优先搜索即可**

类似问题∶编写国际象棋AI，计算最少走多少步就可获胜

根据你的人际关系找到关系最近的医生

**类似于树的层次遍历，需要借助于队列来实现**

**最短路径2∶**权值最小的最短路径∶

**生活案例∶时间最少，距离最短**

**解决方案∶使用狄克斯特拉算法**