图

图(G)是一种非线性数据结构，它由两个集合V(G)和E(G)组成，形式上记为： G=(V, E)
V（G）是顶点Vertex的非空有限集合———————一个图最少有一个顶点
E（G）是V(G)中任意两个顶点之间的关系集合，可以为空

定义

有向图：有向图的边记为<起始顶点，终止顶点>，有向边成为弧。 有向图最多有n(n-1)个边

完全有向图就是有n（n-1）个边的有向图

无向图：无向图的边记为（顶点1，顶点2） 无向图最多有n(n-1)/2个边

完全无向图就是有n(n-1)/2个边

稀疏图：边数小于nlog2n

稠密图：边数大于nlog2n

子图：母图的一部分成为子图

图的度：就是无向图中连接某一顶点边的数量，有向图分为入度和出度，总度为入度和出度之和。

路径：从顶点出发经过一些边到达另一个顶点，经过的顶点序列称为一条路径

简单路径：路径中的顶点不重复出现

简单回路、简单环：起点和终点相同而且路径长度大于等于2的简单路径

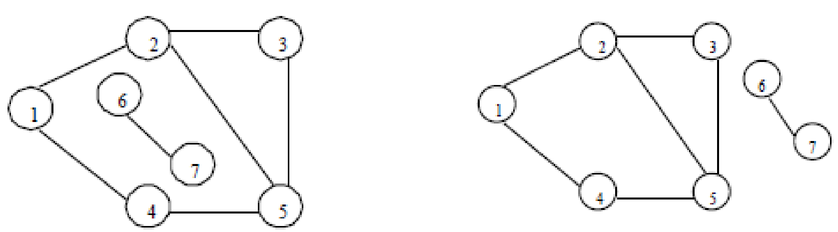
有根图：存在一个顶点可有路径到达所有其他顶点，称为有根图，这个顶点称为 **根**

顶点连通：两个顶点之间有路径相通

连通图：任意两个顶点都连通

连通分量：无向图中的最大连通子图（连通图连通分量就是自己）

无向图及其连通分量



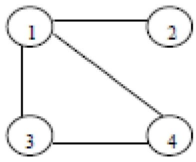
(a) 无向图G　　(b) G的连通分量

强连通：在有向图中两个顶点可以双向连通

强连通图：有向图V(G)中的任意两个顶点都是强连通的

强连通分量：有向图中的极大连通子图

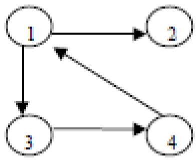
图的存储



一、顺序存储--邻接矩阵存储 用于稀疏图

$$A_2 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

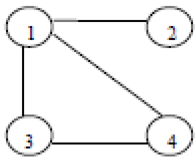
使用一个一维数组存放结点数据信息，然后使用二维数组表示图中顶点关系



(a)有向图G_2

$$A_1 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

行和列中的非零元素个数分别对应于的顶点的出度和入度

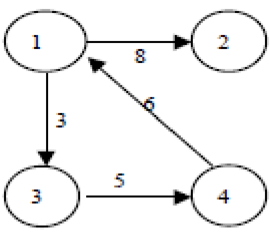


(b)无向图G_1

$$A_2 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

行或列非零元素的个数对应于该顶点的度

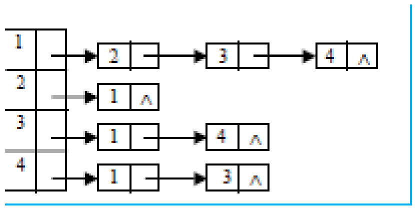
如果是带权图则存储0或者权值



$$A = \begin{bmatrix} 0 & 8 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 \\ 6 & 0 & 0 & 0 \end{bmatrix}$$

```
#define  n  4   /*图的顶点数*/
#define  e     /* 图的边数*/
typedef  struct{
    char  vexs[n];        /* 顶点数组 */
    double  arcs[n][n];       /* 邻接矩阵*/
} graph;
```

二、顺序和链式结合存储---邻接表 用于稠密图

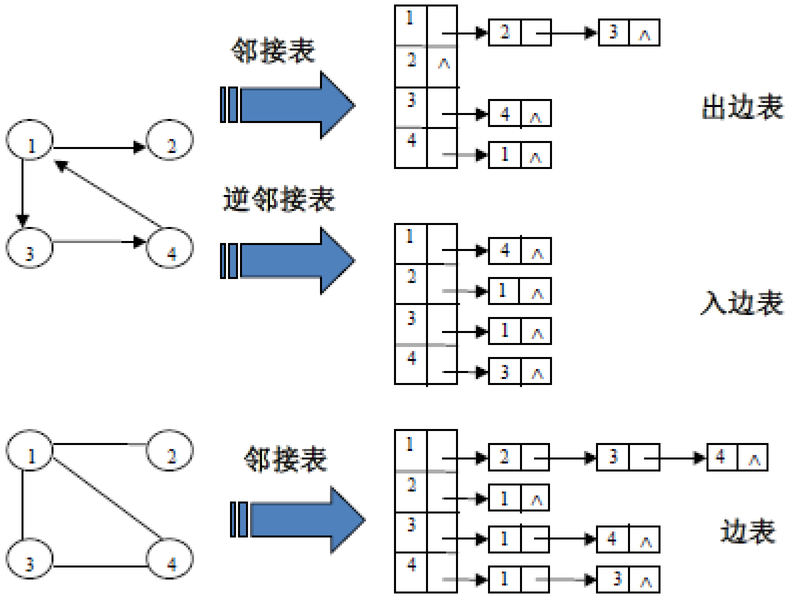


边表：无向图的邻接链表

出边表：有向图的邻接链表

逆邻接表：入边表

图的邻接表实例



2.图的存储方法

- 图的邻接表表示似曾相识~
- 树的孩子表示法

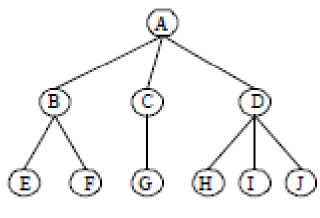
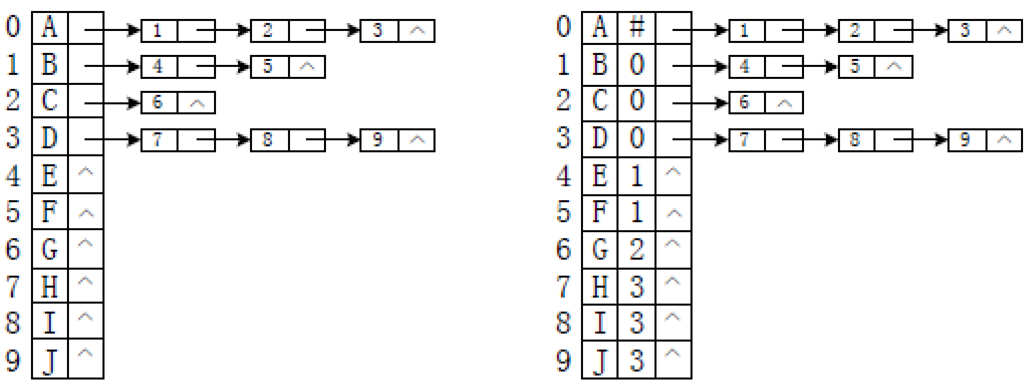


图 5-11 树的表示



(a)孩子表示法

(b)孩子双亲表示法

- 若将该树看做有向图，则孩子表示法就是其邻接表表示法。
- 若将该树看作无向图，则表示时应将孩子双亲表示法中的双亲放到每个结点后的链表内，视为与其他分枝相同的一条边。

两种表示方法的区别

邻接矩阵是唯一的，邻接表不是唯一的

邻接矩阵可以很轻松看到i和j之间是否有一条边， 但是邻接表做不到，最坏时间复杂度为o(n)

邻接矩阵不能直接求到图中边的数目，必须遍历时间复杂度为O（n*n）

但是邻接表对边表的结点个数计数即可

性质

无向图的边数，为所有顶点的度的和的二分之一

图的边可以加权，当图中每条边都有权，则带权图成为网络，简称**网**

不考虑 顶点到自己的边的情况和两点之间有多条边的情况