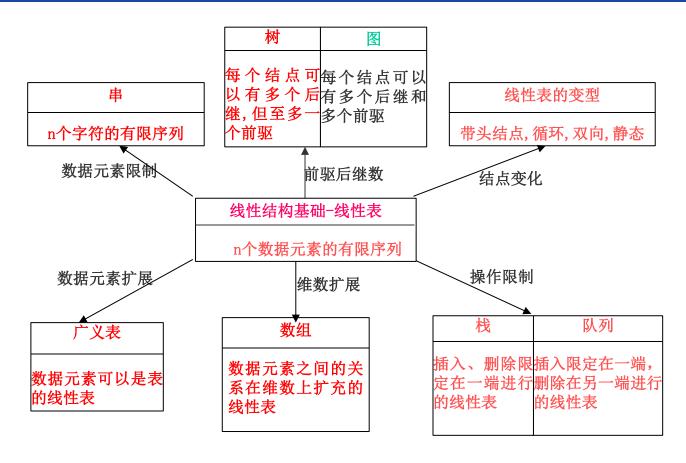
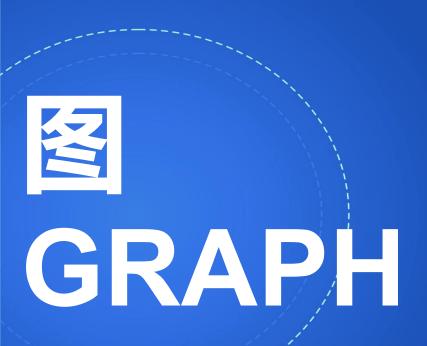
# 数据结构

2017秋季 刘鹏远









- **▲ 图的基本概念**
- 4 图的存储表示
- **4** 图的遍历与连通性
- ♣ 图的遍历应用
- ♣ 最小生成树
- # 最短路径
- ♣ 其他等(选修)



图是以多对多关系定义的网状结构。是一类非常重要 的非线性结构。是一种应用非常广泛的数据结构。

eg: 地图着色问题、交通网规划、通信网络、电路分析、寻找最短路由,项目规划,鉴别化合物、统计力学、遗传学、控制论、语言学,以及社会科学如社会计算等......

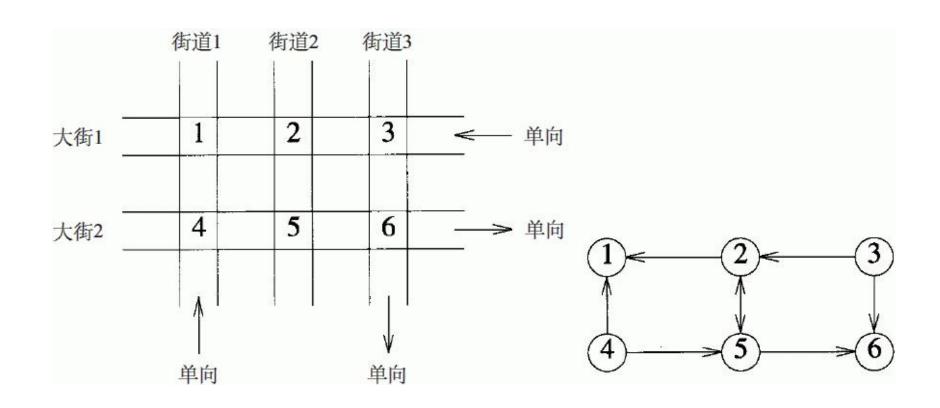
重点: 图表示及其遍历算法

难点:基于图的遍历算法的应用

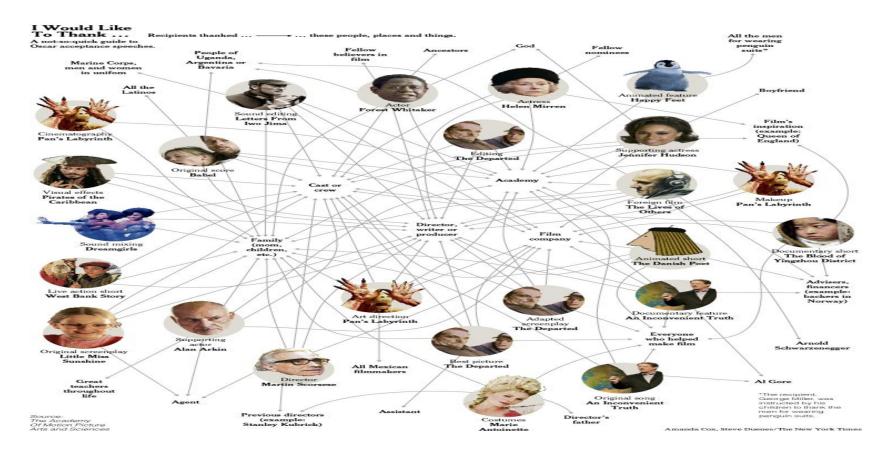














#### 图定义

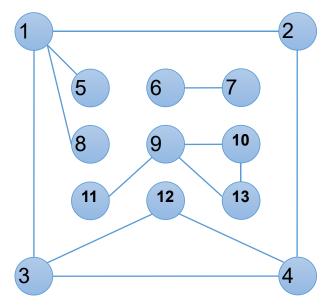
# 图是由顶点集合(vertex)及顶点间的关系(R)

集合组成的一种数据结构:

图中的数据元素称为<mark>顶点</mark> 任意两个顶点v,w间存在或

不存在关系,存在关系就用两个顶点

间的连线进行表示





设V是顶点的有穷非空集合;VR是两个顶点间关系集

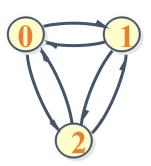
合:VR中任意有序对<v,w>,表示从v到w的一条弧

(有向边), v为弧尾(Tail)或初始点, w为弧头

(Head) 或终端点且此时的图为有向图

定义此图谓词P(v,w)表示

从v到w的一条单向通路。





设V是顶点的有穷非空集合;VR是两个顶点间关系集

合:

对VR中,对任意有序对<v,w>,若<w,v>必存在,则

VR是对称的,此时以无序对(v,w)代替上述两个有序

对,表示v与w之间的一条边(Edge),

此时的图称为无向图。



# 现在可以更明确的定义图:

$$Graph = (V, E)$$
 其中:

$$V = \{x \mid x \in X \mid x$$

$$E = \{(x, y) \mid x, y \in V\}$$
 或

$$E = \{ \langle x, y \rangle \mid x, y \in V \&\& Path(x, y) \}$$

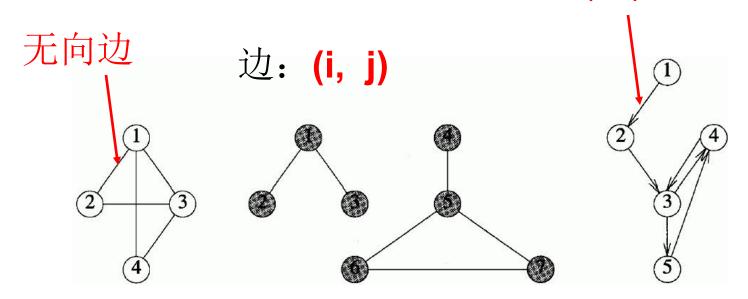
是顶点之间关系的有穷集合(边/弧集合)



- (1)顶点 (2)边 (3)无向边 (4)有向边(弧)
- (5)邻接于 (6)邻接至 (邻接顶点)
- (9)无向图 (10)有向图 (11)完全图 (12)稀疏图 (13)稠密 图
- (14)带权图 (15)子图
- (16)顶点的度 (17)入度 (18)出度
- (19)路径 (20)路径长度 (21)简单路径 (22)回路
- (23)连通图 (24)连通分量 (25)强连通图 (26)强连通分量
- (27)生成树 (28)生成森林



# 有向边(弧)<1,2>



无向 E={(1,2),(1,3),(1,4),(2,3),(3,4)}

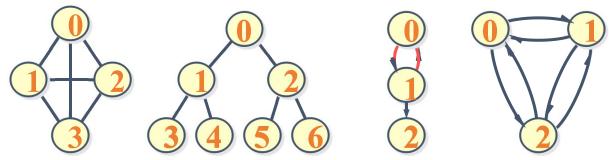
有向 E={<1,2>,<2,3>,<3,4>,<4,3>,<3,5>,<5,4>}



无向图:所有边都是无向边的图,所有顶点对(x,y)均无序

有向图:所有边都是有向边的图,所有顶点对 <x,y>均有序

完全图:边数达到最大的图(任意两点均有边),个数?



完全图的边的个数: n(n-1)/2 无向 n(n-1) 有向



#### 对无向图:

若v,w两点间存在边,则v,w互为邻接点,(v,w)依附 于顶点v,w或称(v,w)与顶点v,w相关联 顶点的度一个顶点,的度是与它相关联的边的条数。 记作TD(v)。路径:在图 G = (V, E) 中, 若从顶点  $v_i$  出发, 沿一些边经过一些顶点  $v_{p1}, v_{p2}, ..., v_{pm}$  , 到达顶点 $v_{i}$ 。 则称顶点序列  $(v_i \ v_{p1} \ v_{p2} \dots \ v_{pm} \ v_i)$  为从顶点 $v_i$ 到顶点  $v_i$ 的路径。它经过的边 $(v_i, v_{p1})$ 、 $(v_{p1}, v_{p2})$ 、…、 $(v_{pm}, v_j)$  应 是属于E的边。



#### 对有向图:

邻接到(至):对<v,w>,则称v邻接到w

邻接自(于):对<v,w>,则称w邻接自v

顶点v的度是与它相关联的弧的条数。记作TD(v)

顶点 v 的入度是以 v 为终点的弧的条数, 记作ID(v);

顶点v的出度是以v为始点的弧的条数,记作OD(v)。

TD(v)=ID(v)+OD(v)

性质:所有顶点度的和是边/弧数量的2倍



- ■路径长度 非带权图的路径长度是指此路径上边的条数。带权图的路径长度是指路径上各边的权之和(非带权相当于权均为1)。
- **■简单路径** 路径上各顶点 ν<sub>1</sub>,ν<sub>2</sub>,...,ν<sub>m</sub> 均不 互相重复
- ■回路 若路径上第一个顶点  $v_1$  与最后一个顶点 $v_m$  重合,则称这

样的路径为回路或环。

■简单路径/简单环

(只有 v起与v终一环的路径)





#### 用n 表示图中顶点数目,用e 表示图中边或弧的数目

稀疏图:有很少边的图(e < nlogn)

稠密图:有较多边的图

#### 因此:

无向图: 0 ≤ e ≤ ½ n(n-1) 完全图 e = ½ n(n-1)

有向图:0 ≤ e ≤ n(n-1) 有向完全图 e = n(n-1)

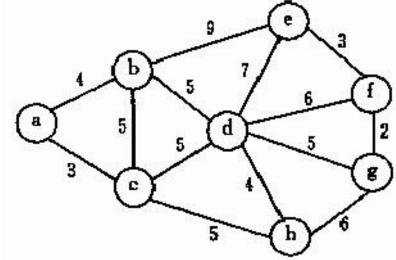
稀疏图: e < nlogn 稠密图 nlogn ≤ e ≤ n(n-1)



■加权有向图、加权无向图:

每条边/弧赋予一个权重,带权图通常称为

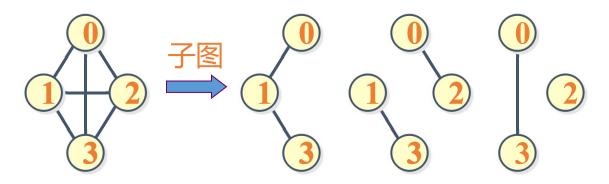
网/网络 (network)





■子图 设有两个图 G = (V, E) 和 G' = (V', E')。若  $V' \subseteq V$  且  $E' \subseteq E$ ,则称 图 G' 是 图 G 的子图。

■一个图往往有多个子图。



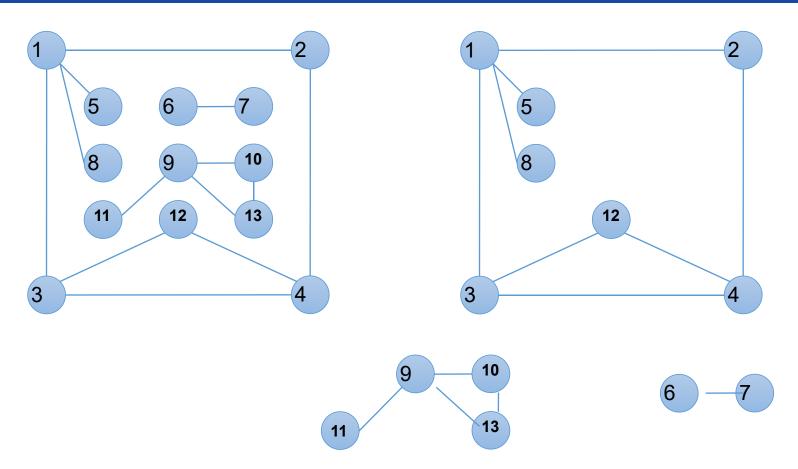


#### ■连通图与连通分量

在无向图中,若从顶点火到顶点火,有路径,则称顶点火,与火。是连通的。如果图中任意一对顶点都是连通的,则称此图是连通图。反之则为非连通图。

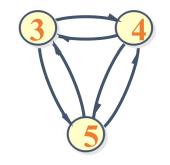
非连通图的<mark>极大</mark>连通子图叫做连通分量。<mark>极大:G中不存在</mark> 既连通又<mark>真包含</mark>连通分量的子图(包含极大顶点、极大边)。

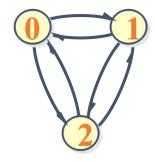






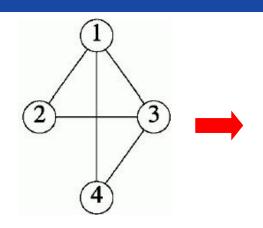
- ■强连通图 在有向图中,若对于每一对顶点水和水,都存在一条从水到水和从水到水的路径,则称此图是强连通图。 ■强连通分量 有向图中的极大强连通子图叫做强连通分量。

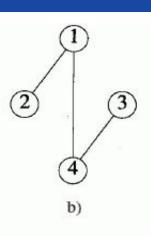


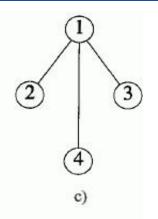


- ■生成树 一个连通图(无向图)的生成树是其极小连通子图 ■生成森林 有向树 见教材P160图7.6

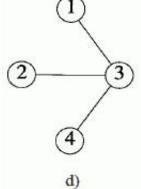


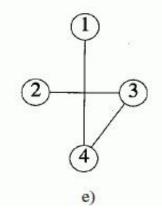


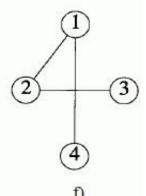




含有全部n个顶点,但只有能够构成一棵树的n-1条边,如果加上一个边,则必定构成一个环。







#### 图的定义-ADT Graph



#### ■ADT Graph//教材P156

```
■查找: LocateVex(G, u)
GetVex(G, v)
FirstAdjVex(G, v)
NextAdjVex(G, v, w)
// 顶点v的第一个邻接点
// 顶点v的在w之后的下一个邻接点
```

- ■插入: InsertVex(&G, v)
  InsertArc(&G, v, w)
- ■删除: DeleteVex(&G, v)
  DeleteArc(&G, v, w)
- ■遍历: DFSTraverse(G, v, Visit()) // 深/广度优先遍历 BFSTraverse(G, v, Visit()) 图G上所有的顶点



- ■线性表:单前驱、单后继
- ■树:单前驱、多后继
- ■图: 节点之间的关系是任意的, 图中任意两个数据元素之间都可能是相关的(也可以理解为多前驱, 多后继)。
- ■图是真实世界中最一般的情况,其他数据结构一般 也可以由图简化得到。



- ■线性表:1对1的关系,结点之间的逻辑关系容易表示
  - ■逻辑关系:线性结构
  - ■存储结构:顺序表示、链式表示
- ■二叉树:1对2的关系,两端均固定,较容易表示
  - ■逻辑关系:树型结构
  - ■存储结构:顺序表示。(结点i的左孩子为2i,右孩子为2i+1)
  - ■存储结构:二叉链表。(从根到每一个结点的一条路径都是一个线性 表)
- ■树:1对n的关系,通过"孩子-兄弟链表"转换成二叉树。



## ■图的存储表示分析

- ■特点:顶点之间的关系是多对多(m:n), m 和 n 都是不定的。m,n 均难以固定,反映顶点之间的逻辑关系困难;
- ■直接考虑多个指针域的节点表示?
- 需引入额外的存储空间来描述顶点之间的邻接关系
- ■存储:所有节点、所有边(关系)。分别考虑



- 图的存储信息
  - ■**整体信息:**顶点数、边/弧数、图的种类
  - ■顶点信息:描述n个顶点的基本情况,可用一个顺序表来描述。(顶点位置是相对概念,任何顶点均可视为第一个顶点)
  - ■边(弧)信息:描述e条边(弧)的基本信息。

对边/弧信息的不同表示方式:邻接矩阵、邻接表、十字 链表、多重邻接表

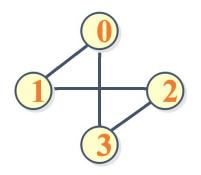


# 邻接矩阵 (Adjacency Matrix)----数组表示法

- ■在图的邻接矩阵表示中,有一个记录各个顶点信息的顶点表,还有一个表示各个顶点之间关系的邻接矩阵。
- ■设图 A = (V, E) 是一个有 n 个顶点的图,图的邻接矩阵是一个二维数组 A.edge[n][n] ,定义:

$$A.Edge[i][j] = \begin{cases} 1, & \text{若} < i, j > \in E \text{或}(i, j) \in E \\ 0, & \text{否则} \end{cases}$$





A.edge = 
$$\begin{vmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{vmatrix}$$

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

- ■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。

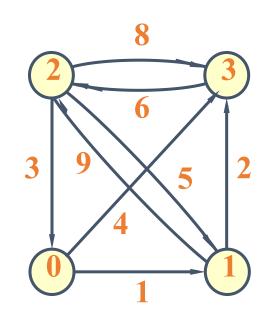


- ■在无向图中, 统计第 i 行 (列) 1 的个数可得顶点i 的度。
- ■在有向图中,统计第 *i* 行 1 的个数可得顶点 *i* 的出度,统计第 *j* 列 1 的个数可得顶点 *j* 的入度。行和为出度,列和为入度
- ■网络(有权值图)的表示:

$$\mathbf{A.edge}[i][j] = \begin{cases} \mathbf{W}(i,j), & \ddot{\mathbf{z}}i \neq j \mathbf{L} < i,j > \in \mathbf{E}\vec{\mathbf{y}}(i,j) \in \mathbf{E} \\ \infty, & \ddot{\mathbf{z}}i \neq j \mathbf{L} < i,j > \notin \mathbf{E}\vec{\mathbf{y}}(i,j) \notin \mathbf{E} \\ \mathbf{0}/\infty & \ddot{\mathbf{z}}i = j \end{cases}$$







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

#### 作业图例

#### 用邻接矩阵表示的图结构的定义



```
#define MaxValue INT MAX
                           //在imits.h>中,∞
#define NumEdges 500
#define NumVertices 50
                        //顶点数据类型
typedef char VertData;
typedef int EdgeData;
typedef struct {
 VertData VexList[NumVertices];
  EdgeData edge[NumVertices][NumVertices];
            //邻接矩阵, 可视为边之间的关系
            //图中当前的顶点个数与边数
  int n, e;
             //图的种类 如有向无向有无权值等
  //int kind;
} MGraph;
```



```
void CreatGraph(MGraph *G){//随图种类而不同
 int i,j,u,n,e,v; //可利用kind设计不同分
 scanf("%d,%d",&n, &e);
                                          G- > edge =
 for(i=0;i<n;i++){
      for(j=0;j<n;j++){
            \int_{J-\upsilon; J<n; J++){\{}} 0 0

\int_{J-\upsilon; J<n; J++){\{}} 0 0

\int_{J-\upsilon; J<n; J++){\{}} 0

\int_{J-\upsilon; J<n; J++){\{}} 0
          else G->edge[i][j] = MaxValue;//对网络
//else G->edge[i][j] = 0;//无权图
```

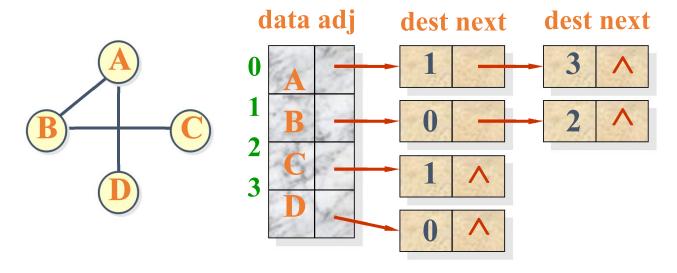


```
for(i=0;i<n;i++){
   scanf("%c", &(G->VexList[i]));
}//输入顶点表
for(i=0;i<e;i++){
   scanf("%d,%d,%d", &u,&v,&(G->edge[u][v]));
   //G->edge[v][u]=G->edge[u][v];//无向图
}//邻接矩阵输入完毕
G->n =n;G->e=e;
//以上为创建框架,随图种类不同而稍有不同
```



# ■邻接表表示

# ■无向图





节点表用数组/链表储存。

边链表存储同一个顶点发出的边,每一个结点代表一

条边, 结点中有另一顶点的下标 dest 和指针 next。

顶点 i : 设有data域,保存该顶点的其它信息,类似

单链表的头结点,指针 adj , 指向第一个边。

对无向图,度及相邻节点相互查找均方便。why?

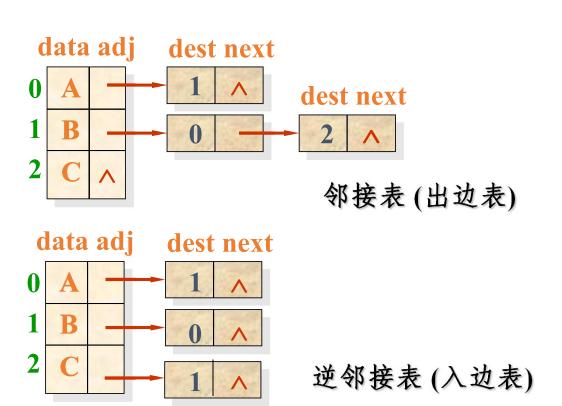


### ■邻接表表示

■有向图

出度及找相邻 节点容易,反 之则麻烦。

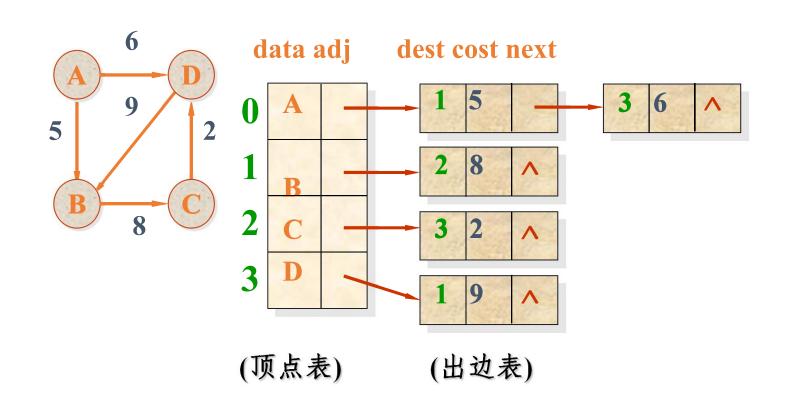
因此,可以额外存储逆邻接表。





- ■在邻接表的边链表中,各个边结点的链入顺序任意,视边结点输入次序而定。
- ■设图中有 n 个顶点, e 条边,则用邻接表表示无向图时,需要 n 个顶点结点, 2e 个边结点(why?);用邻接表表示有向图时,若不考虑逆邻接表,只需 n 个顶点结点, e 个边结点。
- ■带权图的边结点中保存该边上的权值 cost。

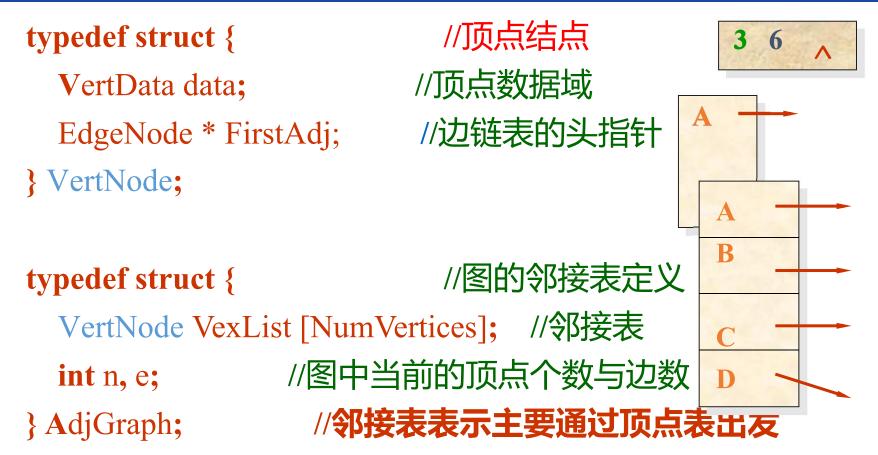






```
//顶点个数
#define NumVertices 50;
                       //顶点数据类型
typedef char VertData;
                           //边上权值类型
typedef float EdgeData;
                         //边结点
typedef struct node {
                         //目标顶点下标
  int dest;
                           //边上的权值
  EdgeData cost;
                         //下一边链接指针
  Struct node * next;
} EdgeNode;
```







```
void CreateGraph (AdjGraph* G) {//邻接表建立图(框架伪码)
  int i;
  scanf(&G->n,&G->e); //输入顶点个数和边数
  for (i = 0; i < G->n; i++)
    scanf(&G->VexList[i].data);
                              //输入顶点信息
    G->VexList[i].FirstAdj = NULL; //边链指针置空
  for (i = 0; i < e; i++) { //逐条边输入
    scanf ( &tail, &head, &weight);
    EdgeNode * p = malloc (sizeof(EdgeNode));
    //创建边结点 //以链表头插法插入
```



```
p->dest = head; p->cost = weight;
     p->next = G->VexList[tail].FirstAdj;//头插,初始NULL
     G->VexList[tail].FirstAdj = p; //指向新节点
    //下面无向图专用,两顶点间两条边//有向图不需要
     p = malloc (sizeof(EdgeNode));
     p->dest = tail; p->cost = weight;
     p->next = G->VexList[head].FirstAdj;
     G->VexList[head].FirstAdj = p;
}//O ( n+2e ) //有向图n+e
```

### 图的存储表示



- ■邻接矩阵表示的优化(参考矩阵压缩)
- ■十字链表(邻接+逆邻接, p164)
- ■邻接多重表(一个边用一个边结点, p166)

邻接矩阵占空间: 邻接表占空间:

 $n+n^2$  n+e\*2

从空间上考虑,邻接矩阵更适合稠密图

### 下周上机



# 对本讲PPT中作业图例

- 1、邻接矩阵存储。打印顶点表及邻接矩阵
- 2、邻接表存储。遍历顶点表,打印每个 顶点及其所有相邻顶点以及权值,出度

#### 下周进度



邻接矩阵\邻接表相关操作实现 图的深度优先、广度优先遍历 图的应用等等