邻接矩阵法(适合稠密图)

结点数为 n的图 $G=(V,E)$的邻接矩阵 A是 $n\times n$的,将 G的顶点编号为 $v_1,v_2,\cdots,v_n$,则

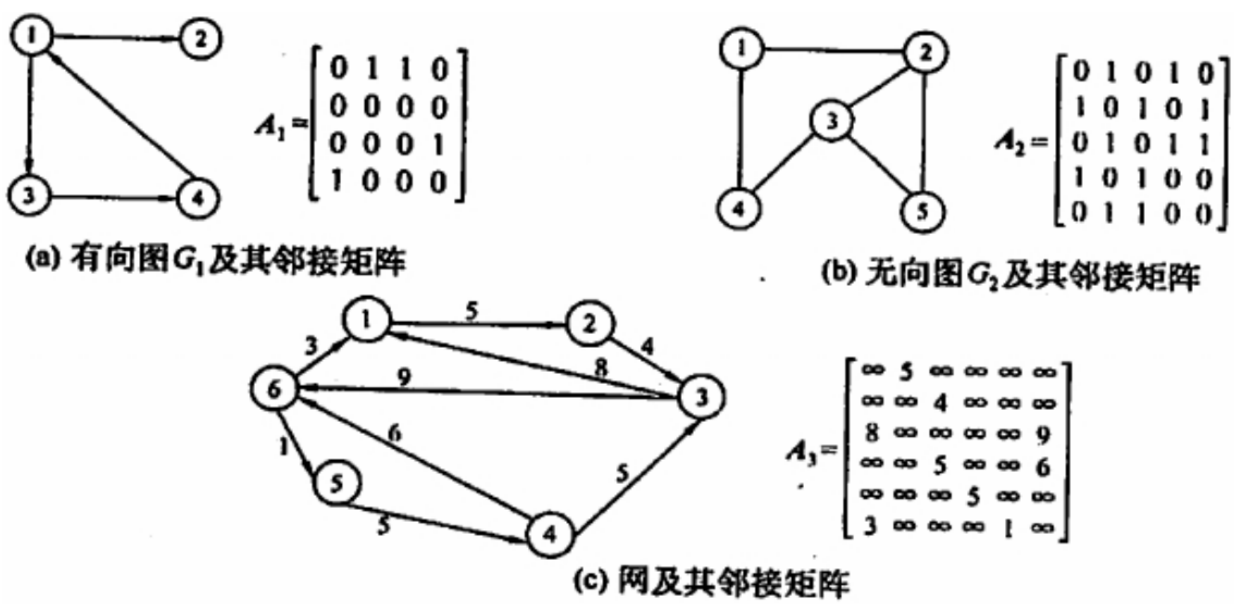
$$A[i][j]=\begin{cases}1, & 若\ (v_i,v_j)\ 或\ \langle v_i,v_j\rangle\ 是\ E(G)\ 中的边\\ 0, & 若\ (v_i,v_j)\ 或\ \langle v_i,v_j\rangle\ 不是\ E(G)\ 中的边\end{cases}$$

对于**带权图**,若顶点 v_i和 v_j之间有边相连,则邻接矩阵中对应项存放着该边**对应的权值**;若顶点 v_i和 v_j不相连,则通常用 ∞来代表着两个顶点间不存在边,则

$$A[i][j]=\begin{cases}w_{ij}, & 若\ (v_i,v_j)\ 或\ \langle v_i,v_j\rangle\ 是\ E(G)\ 中的边\\ 0\ 或\ \infty, & 若\ (v_i,v_j)\ 或\ \langle v_i,v_j\rangle\ 不是\ E(G)\ 中的边\end{cases}$$

图的邻接矩阵存储结构如下:

```c
#define MaxVertexNum 100//顶点数目的最大值
#define inf (INT_MAX)//无穷大

typedef char VertexType;//顶点的数据类型
typedef int EdgeType;//带权图中边上权值的数据类型

typedef struct{
    VertexType Vex[MaxVertexNum];//顶点表
    EdgeType Edge[MaxVertexNum][MaxVertexNum];//邻接矩阵,边表
    int vexnum,arcnum;//图的当前顶点数,图的边数/弧数
}MGraph;
```



(a) 有向图 G_1 及其邻接矩阵

(b) 无向图 G_2 及其邻接矩阵

(c) 网及其邻接矩阵

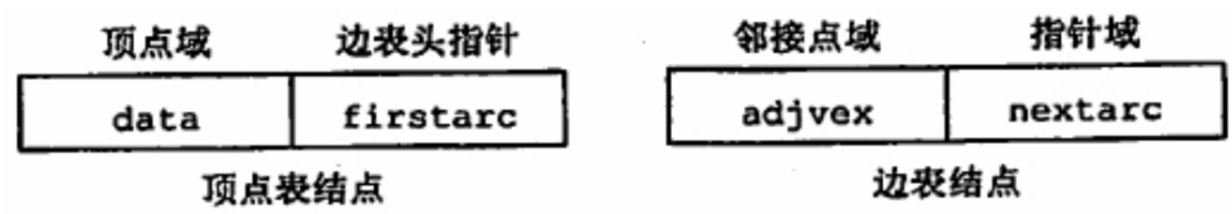
- **无向图**的邻接矩阵一定是一个**对称矩阵**(并且唯一),对规模特大的邻接矩阵可采用**压缩存储**(只存储上(或下)三角矩阵的元素)
- 对于**无向图**,邻接矩阵的第 i**行**(或第 i**列**)非零元素(或非 ∞元素)的个数是**顶点 i的度** $TD(v_i)$

- 对于**有向图**,邻接矩阵的第 i**行**非零元素(或非 ∞元素)的个数是**顶点 i的出度** $OD(v_i)$;第 i**列**非零元素(或非 ∞元素)的个数是**顶点 i的入度** $ID(v_i)$
- 用邻接矩阵存储图,可容易确定图中**任意两个顶点之间是否有边相连**.要确定图中有多少条边,必须按行、按列对每个元素进行检测,所花费的时间代价很大
- **稠密图**适合使用邻接矩阵的存储表示,空间复杂度为 $O(|V|^2)$
- 设图 G的邻接矩阵为 A,则 A^n的元素 $A^n[i][j]$等于由顶点 i到顶点 j的长度为 n的**路径的数目**

邻接表法(适合稀疏图和其他)

当一个图为**稀疏图**,使用`邻接矩阵法`会**浪费大量的存储空间**,而图的`邻接表法`结合了**顺序存储和链式存储**的方式,大大减少不必要的浪费.

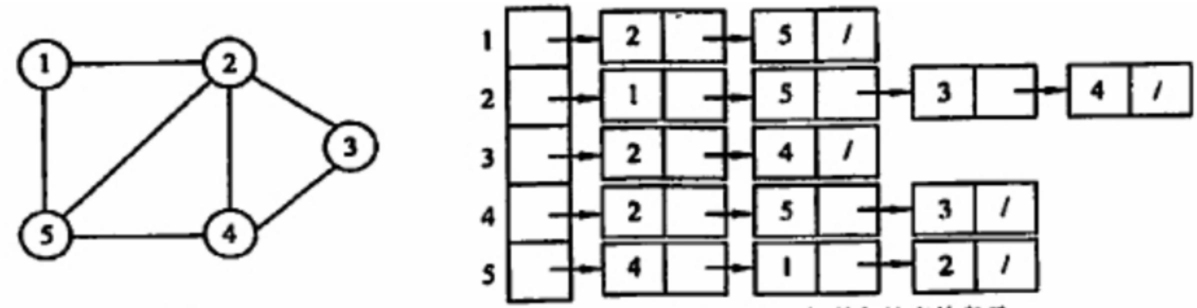
对图 G中每个顶点 v_i建立一个单链表,第 i个单链表中的结点表示依附于顶点 v_i的边(在有向图表示以顶点 v_i为尾的弧),这个单链表是顶点 v_i的**边表**(对有向图称为**出边表**).边表的头指针和顶点的数据信息采用顺序存储(称为**顶点表**),在邻接表中存在两种结点:`顶点表结点`和`边表结点`.



顶点表结点由顶点域 $data$和指向第一条邻接边的指针 $firstarc$构成,边表(邻接表)结点由邻接点域 $adjvex$和指向下一条邻接边的指针域 $nextarc$构成.

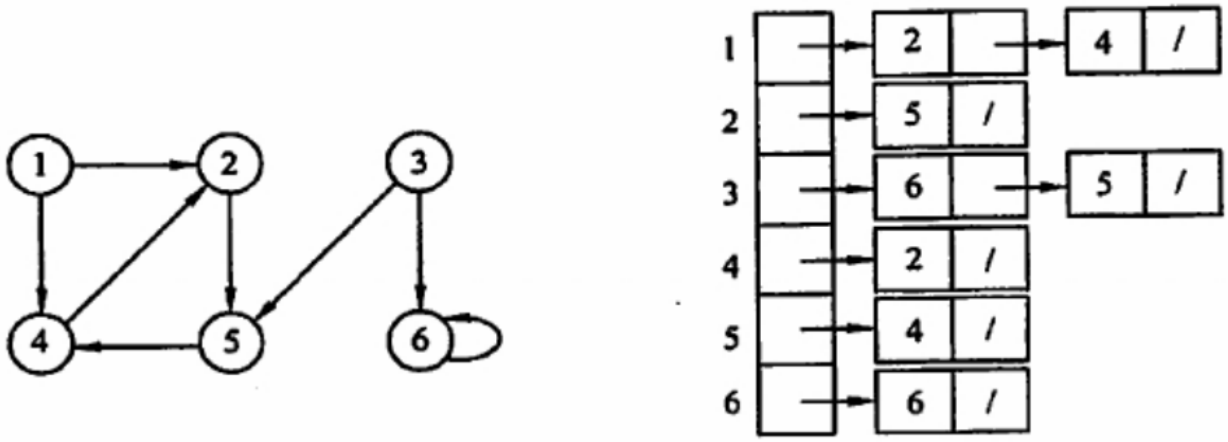
图的邻接表存储结构如下:

```
#define MaxVertexNum 100//顶点数目的最大值

typedef char VertexType;//顶点的数据类型
typedef int InfoType;//带权图中边上权值的数据类型

typedef struct ArcNode{//边表结点
    int adjvex;//该弧所指向的顶点的位置
    struct ArcNode *next;//指向下一条弧的指针
    //InfoType info;//网的边权值
}ArcNode;

typedef struct VNode{//顶点表结点
    VertexType data;//顶点信息
    ArcNode *first;//指向第一条依附该顶点的弧的指针
}VNode,AdjList[MaxVertexNum];

typedef struct{//邻接表
    AdjList vertices;//邻接表
    int vexnum,arcnum;//图的当前顶点数,图的边数/弧数
}ALGraph;
```



(a)无向图G　　(b)无向图G的邻接表的表示

图 6.7　无向图邻接表表示法实例



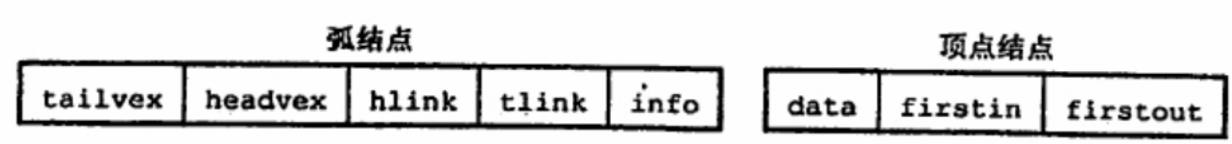
(a)有向图 G　　(b)有向图 G 的邻接表的表示

- 若 G为**无向图**,则需要的存储空间为 $O(|V|+2\times|E|)$;若 G为**有向图**,则需要的存储空间为 $O(|V|+|E|)$.前者的倍数 2由于无向图中,**每条边在邻接表中出现了两次**
- 对于**稀疏图**,采用邻接表表示将极大地节省存储空间
- 在邻接表中,给定一顶点,可容易地找出它的**所有邻边**,只需要读取它的邻接表;而在邻接矩阵中,相同的操作需要扫描一行,时间复杂度为 $O(n)$.若要确定给定两个**顶点间是否存在边**,则在邻接矩阵中可立刻查到;而在邻接表中需要再相应结点对应的边表中查找另一结点,效率极低.
- 在有向图的邻接表表示中,求一个给定顶点的出度只需要计算其邻接表中的结点个数;但求其顶点的入度则需要遍历全部的邻接表.此时可采用**逆邻接表**的存储方式来加速求解给定顶点的入度.
- 图的邻接表表示**不唯一**,在每个顶点对应的单链表中,各边结点的链接次序可以是任意的,它取决于建立**邻接表的算法及边的输入次序**

	邻接表	邻接矩阵
空间复杂度	无向图 O(|V| + 2|E|)；有向图O(|V| + |E|)	$O(\|V\|^2)$
适合用于	存储稀疏图	存储稠密图
表示方式	不唯一	唯一
计算度/出度/入度	计算有向图的度、入度不方便，其余很方便	必须遍历对应行或列
找相邻的边	找有向图的入边不方便，其余很方便	必须遍历对应行或列

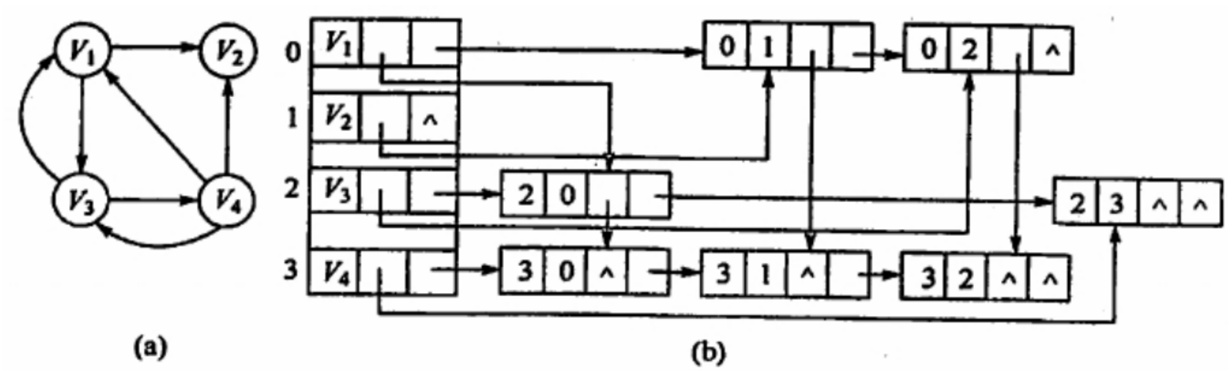
十字链表(用于有向图)

十字链表是有向图的一种链式存储方式.



弧结点中有 5个域： $tailvex$和 $headvex$两个域分别指示弧尾和弧头这两个顶点的编号; $hlink$域指向弧头相同的下一个弧结点; $tlink$域指向弧尾相同的下一个弧结点; $info$域存放该弧的相关信息.**弧头相同的弧在同一个链表上,链尾相同的弧也在同一个链表上**

顶点结点中有 3个域, $data$域存放该顶点的数据信息,如顶点名称; $firstin$域指向以该顶点为弧头的第一个弧结点; $firstout$域指向以该顶点为弧尾的第一个弧结点.

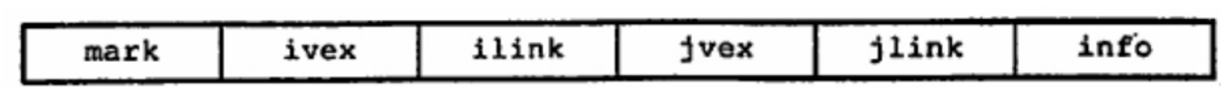


在十字链表中,既容易找到 v_i为尾的弧,又容易找到 v_i为头的弧,故容易求得**顶点的出度和入度**.图的十字链表表示依然是**不唯一**的,但一个十字链表表示**确定一个图**.

邻接多重表(用于无向图)

邻接多重表是无向图的另一种链式存储结构.在邻接表中,容易求得顶点和边的各种信息,但在邻接表中求两个顶点之间是否存在边而对边执行删除等操作时,需要分别在两个顶点的边表中遍历,效率极低.

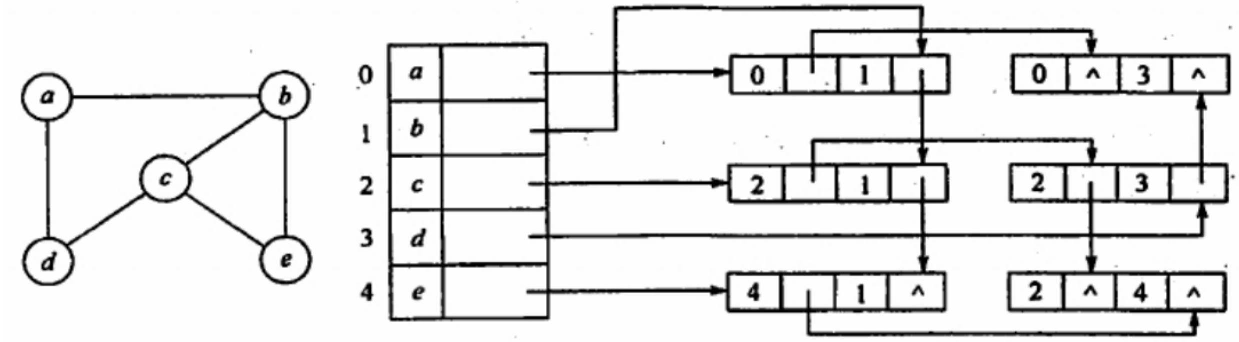
与十字链表类似,在邻接多重表中,每条边用一个结点表示,如下:



其中, $mark$为标志域(现王道书中无 $mark$域),用于标记该条边是否被搜索过; $ivex$和 $jvex$为该边依附的两个顶点在图中的位置; $ilink$指向下一条依附于顶点 $ivex$的边; $jlink$指向下一条依附于顶点 $jvex$的边, $info$指向和边相关的各种信息的指针域.

每个顶点用一个结点表示, $data$域存储该顶点的相关信息, $firstedge$域指向第一条依附于该顶点的边.

对无向图,其邻接多重表和邻接表的差别仅在于,同一条边在邻接表中用两个结点表示,而在邻接多重表中只有一个结点.



	邻接矩阵	邻接表	十字链表	邻接多重表
空间复杂度	O(|V|²)	无向图 O(|V| + 2|E|) 有向图O(|V| + |E|)	O(|V| + |E|)	O(|V| + |E|)
找相邻边	遍历对应行或列 时间复杂度为O(|V|)	找有向图的入边必须遍历整个邻接表	很方便	很方便
删除边或顶点	删除边很方便，删除顶点需要大量移动数据	无向图中删除边或顶点都不方便	很方便	很方便
适用于	稠密图	稀疏图和其他	只能存有向图	只能存无向图
表示方式	唯一	不唯一	不唯一	不唯一

> 在含有 n个顶点和 e条边的无向图的邻接矩阵中,零元素的个数为().
>
> A. e
>
> B. $2e$
>
> C. $n^2 - e$
>
> D. $n^2 - 2e$

在 n个顶点的邻接矩阵中,矩阵大小为 n^2,无向图每条边在邻接矩阵中贡献两次,即非零元素个数为 $2e$,零元素个数为 $n^2 - 2e$.

> 一个有 n个顶点的图用邻接矩阵 A表示,若图为有向图,顶点 v_i的入度是(B);若图为无向图,顶点 v_i的度是(D).
>
> A. $\sum_{i=1}^{n} A[i][j]$
>
> B. $\sum_{j=1}^{n} A[j][i]$
>
> C. $\sum_{i=1}^{n} A[j][i]$
>
> D. $\sum_{j=1}^{n} A[j][i]$ 或 $\sum_{j=1}^{n} A[i][j]$

- A表示顶点 v_j的**入度**(有向图)或 **度**(无向图);
- B表示顶点 v_i的**入度**(有向图)或 **度**(无向图);
- C表示顶点 v_j的**出度**(有向图)或 **度**(无向图);

- D第一个同 B,第二个表示顶点 v_i的**出度**(有向图)或 **度**(无向图)

扩展:

AOV网用**顶点**表示活动,边表示活动（顶点）发生的先后关系,常采用拓扑排序;而 AOE网是边表示活动的网, AOE网是**带权有向无环图**,边代表活动,顶点代表**所有指向它的边所代表的活动均已完成**这一事件,用于解决关键路径.

> 从邻接矩阵
>
> $$A=\begin{bmatrix}0 & 1 & 0\\ 1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}$$
>
> 可以看出,该图共有 (① B)个顶点;若是有向图,则该图共有(② B)条弧;若是无向图,则共有(③ D)条边.
>
> | | | | | | |
> |---|---|---|---|---|---|
> | ① | A. 9 | B. 3 | C. 6 | D. 1 | E. 以上答案均不正确 |
> | ② | A. 5 | B. 4 | C. 3 | D. 2 | E. 以上答案均不正确 |
> | ③ | A. 5 | B. 4 | C. 3 | D. 2 | E. 以上答案均不正确 |

邻接矩阵的**顶点数**等于矩阵的`行(列)数`,**有向图的边数**等于矩阵中`非零元素个数`,**无向图的边数**等于矩阵中`非零元素个数的一半`.

> n个顶点的无向图的邻接表最多有()个边表结点.
>
> A. n^2
>
> B. $n(n-1)$
>
> C. $n(n+1)$
>
> D. $n(n-1)/2$

n个顶点的无向图最多有 $\frac{n\times(n-1)}{2}$ 条边,每条边在邻接表存储两次,故边表结点最多有 $n\times(n-1)$.

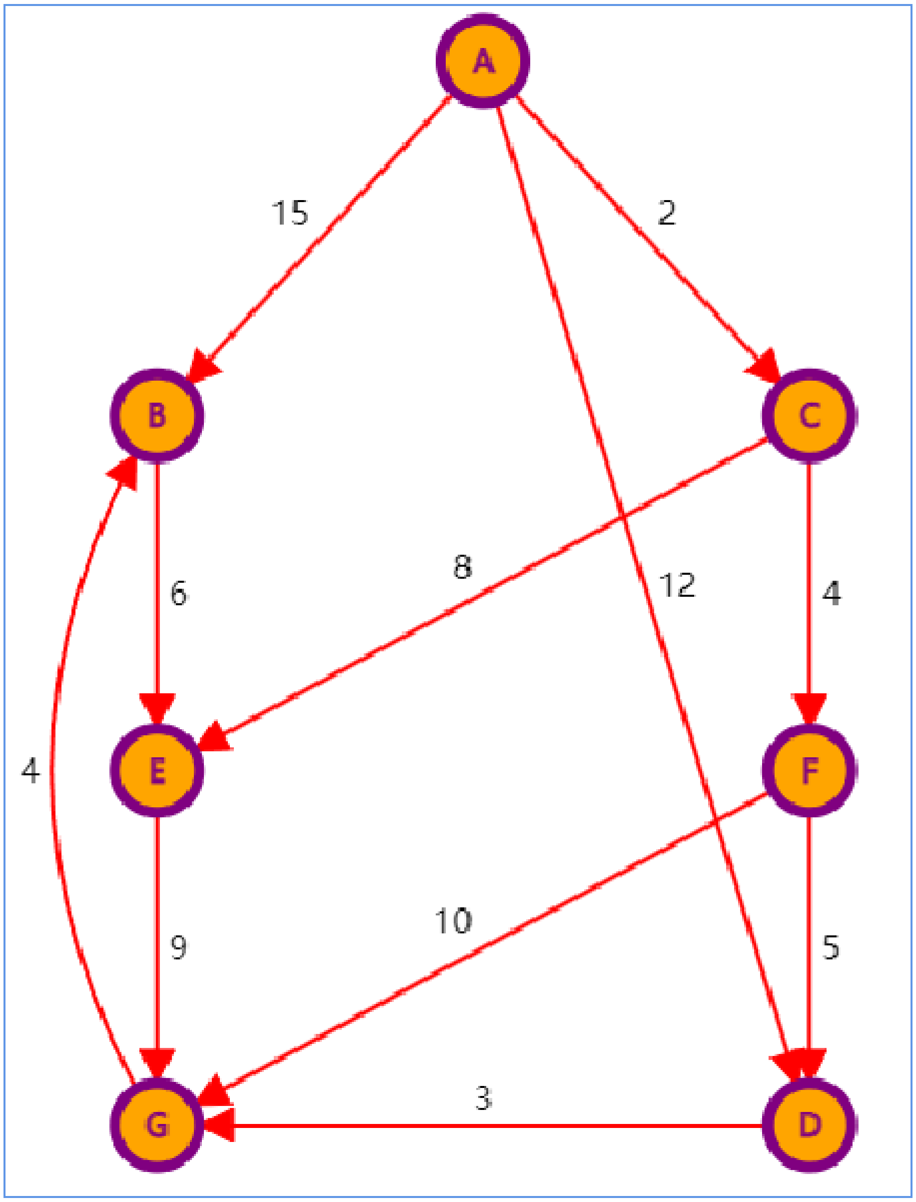
> 假设有 n个顶点, e条边的有向图用邻接表表示,则删除与某个顶点 v相关的所有边的时间复杂度为().
>
> A. $O(n)$
>
> B. $O(e)$
>
> C. $O(n+e)$
>
> D. $O(ne)$

删除与某顶点 v相关的所有边的过程如下：先删除下标为 v的**顶点表结点的单链表**,出边数最多为 $n-1$,对应时间复杂度为 $O(n)$;再扫描所有边表结点,删除所有的顶点 v的**入边**,对应时间复杂度为 $O(e)$.故总的时间复杂度为 $O(n+e)$.

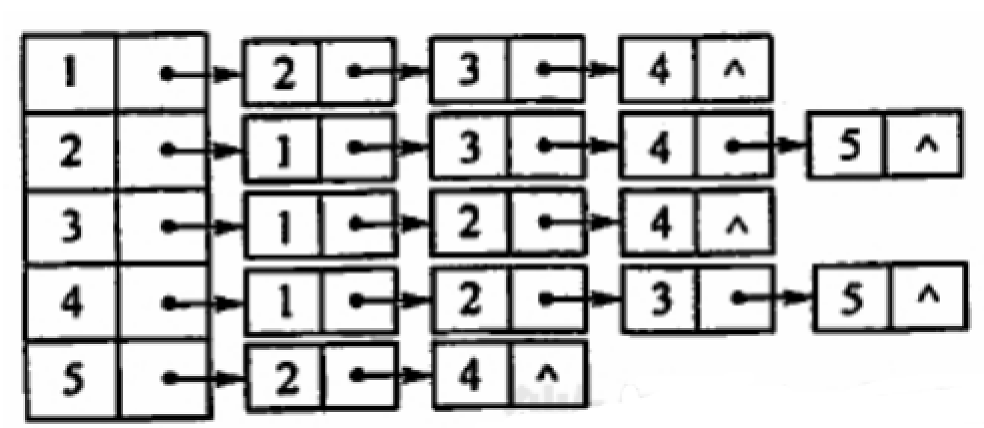
6.2.1

> 已知带权有向图 G的邻接矩阵如下图所示,请画出该带权有向图.
>
> | | | | | | | |
> |---|---|---|---|---|---|---|
> | 0 | 15 | 2 | 12 | ∞ | ∞ | ∞ |
> | ∞ | 0 | ∞ | ∞ | 6 | ∞ | ∞ |
> | ∞ | ∞ | 0 | ∞ | 8 | 4 | ∞ |
> | ∞ | ∞ | ∞ | 0 | ∞ | ∞ | 3 |
> | ∞ | ∞ | ∞ | ∞ | 0 | ∞ | 9 |
> | ∞ | ∞ | ∞ | 5 | ∞ | 0 | 10 |
> | ∞ | 4 | ∞ | ∞ | ∞ | ∞ | 0 |

设顶点集为 A, B, C, D, E, F, G,该带权有向图如下.

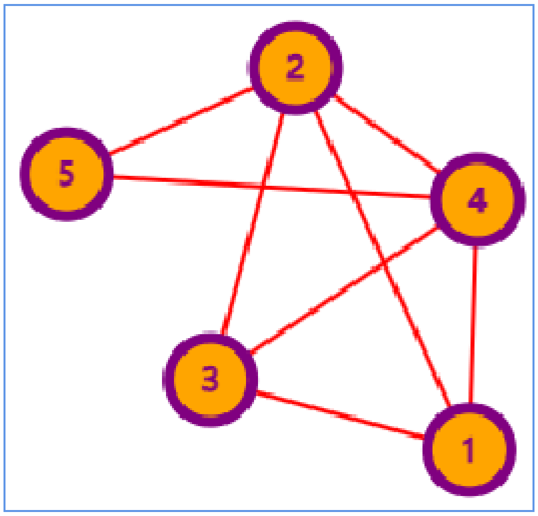


6.2.2

> 设图 $G=(V,E)$以邻接表存储,如下图所示,画出其邻接矩阵存储及图 G.
>
> 
>

邻接矩阵如下,可发现该图是**无向图**,可不用画出有向边

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



6.2.3

> 对 n个顶点的无向图和有向图,分别采用邻接矩阵和邻接表表示时,试问:
>
> 1)如何判别图中有多少条边?
>
> 2)如何判别任意两个顶点 i和 j是否有边相连?
>
> 3)任意一个顶点的度是多少?

1)

- **无向图**

在**邻接矩阵**中,统计矩阵**对角线以上部分或对角线以下部分 1的个数**(无向图在邻接矩阵中具有`对称性`)或者矩阵中**所有 1的总数除** 2;

在**邻接表**中,统计邻接表**各顶点边链表中(边)结点的个数**,无向图的每条边在邻接表存储两次,故边数应在统计的基础上**除** 2即可.

- **有向图**

在**邻接矩阵**中,统计矩阵中**所有 1的总数**;

在邻接表中,统计邻接表**各顶点边链表中(边)结点的个数**.

2)

- **邻接矩阵**

对于任意两个顶点 i和 j,邻接矩阵中 $A[i][j]$**(适用有向图和无向图)或** $A[j][i]$**(只适用无向图)为** 1 表示有边相连,否则表示无边相连.

- **邻接表**

对于任意两个顶点 i和 j,若**从顶点表结点 i出发找到编号为 j的边表结点(适用有向图和无向图)**或**从顶点表结点 j出发找到编号为 i的边表结点(只适用无向图)**,表示有边相连,否则表示无边相连.

3)

- **邻接矩阵**

在**无向图**中,顶点 i的度等于**第 i行或第 i列中 1的个数**;

在**有向图**中,顶点 i的**出度**等于第 i**行**中 1的个数,**入度**等于第 i**列**中 1的个数,该顶点的**度**等于**出度与入度的和**.

- **邻接表**

在**无向图**中,顶点 i的度等于**顶点表结点 i的单链表中边表结点的个数**;

在**有向图**中,顶点 i的出度等于**顶点表结点 i的单链表中边表结点的个数**,顶点 i的入度等于**邻接表中所有编号为 i的边表结点数**或**对应** `逆邻接表` **中顶点 i的单链表中边表结点的个数**,该顶点的**度**等于**出度与入度的和**.

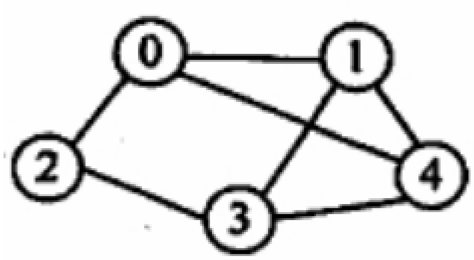
6.2.4

> 写出从图的邻接表表示转换成邻接矩阵表示的算法.

```
void ALGraph_To_MGraph(ALGraph g1,MGraph &g2)
{
    g2.vexnum=g1.vexnum;//更新顶点数
    g2.arcnum=g1.arcnum;//更新边数

    for(int i=1;i<=g1.vexnum;i++)//更新顶点表
        g2.Vex[i]=g1.vertices[i].data;

    for(int i=1;i<=g1.vexnum;i++)
    {
        ArcNode *p=g1.vertices[i].first;//取出第一条边

        while(p!=NULL)//遍历边链表
        {
            g2.Edge[i][p->adjvex]=1;
            p=p->next;
        }
    }
}
```

6.2.5

> **2015统考真题**：已知含有 5个顶点的图 G如下图所示.



请回答下列问题:

1)写出图 G的邻接矩阵 A(行、列下标从 0开始).

2)求 A^2,矩阵 A^2中位于 0行 3列元素值的含义是什么?

3)若已知具有 n($n \geq 2$)个顶点的图的邻接矩阵为 B,则 B^m($2 \leq m \leq n$)中非零元素的含义是什么?

1)

如下图所示为图 G的邻接矩阵 A

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

2)

矩阵 A^2如下图所示,其中位于第 0行第 3列的元素值 $A^2[0][3]=3$表示从顶点 0到顶点 3之间长度为 2的路径共有 3条.

$$A^2=\begin{bmatrix}0&1&1&0&1\\1&0&0&1&1\\1&0&0&1&0\\0&1&1&0&1\\1&1&0&1&0\end{bmatrix}\cdot\begin{bmatrix}0&1&1&0&1\\1&0&0&1&1\\1&0&0&1&0\\0&1&1&0&1\\1&1&0&1&0\end{bmatrix}=\begin{bmatrix}3&1&0&3&1\\1&3&2&1&2\\0&2&2&0&2\\3&1&0&3&1\\1&2&2&1&3\end{bmatrix}$$

3)

矩阵 B^m($2 \leq m \leq n$)中位于第 i行第 j列($0 \leq i,j \leq n-1$)的非零元素的含义是:**图中从顶点 i到顶点 j的长度为 m的路径条数**.

6.2.6

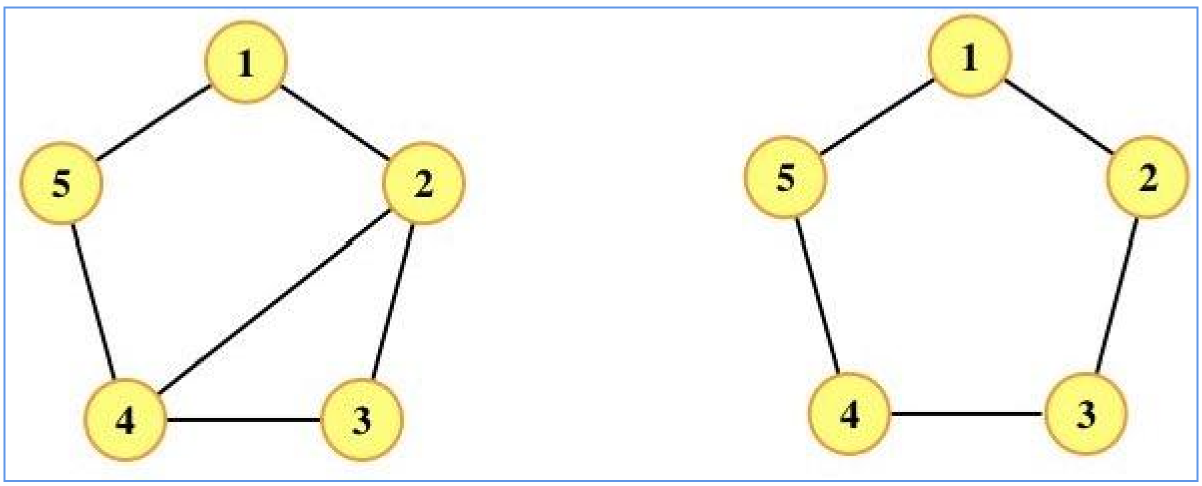
> **2021统考真题**：已知无向连通图 G由顶点集 V和边集 E组成, $|E| > 0$,当 G中度为奇数的顶点个数为不大于 2的偶数时, G存在包含所有边且长度为 $|E|$的路径(称为 EL路径).设图 G采用邻接矩阵存储,类型定义如下：

```
typedef struct {                    // 图的定义
    int numVertices, numEdges;      // 图中实际的顶点数和边数
    char VerticesList[MAXV];        // 顶点表，MAXV为已定义常量
    int Edge[MAXV][MAXV];           // 邻接矩阵
}MGraph;
```

> 请设计算法`int IsExistEL(MGraph G)`,判断 G是否存在 EL路径,若存在,则返回 1,否则返回 0.要求：
>
> 1)给出算法的基本设计思想.
>
> 2)根据设计思想，采用`C`或`C++`语言描述算法，关键之处给出注释.
>
> 3)说明你所设计算法的时间复杂度和空间复杂度.

本题提到了 EL路径,即欧拉路径. EL正是大名鼎鼎的 **欧拉(Euler)的缩写**.

欧拉路径(Euler path)：如果图 G中的一个路径包括每个边恰好一次.则该路径称为欧拉路径.



无向图存在欧拉路径的充要条件：度为奇数的点的数量为 0个或 2个,题目中也给出`奇数的顶点个数为不大于2的偶数`.

- 第一步：统计所有点的度;
- 第二步：统计所有点中度为奇数的点的个数;
- 第三步：检查度为奇数的点个数是否为 0或者 2.

```
int IsExistEL(MGraph G)
{
    int degree[MAXV];
    for(int i=1;i<=G.numVertices;i++)//初始化置0
        degree[i]=0;

    for(int i=1;i<=G.numVertices;i++)//遍历无向图统计所有点的度
```

```
            for(int j=1;j<=G.numVertices;j++)
                degree[i]+=G.Edge[i][j];

    int tot=0;//遍历degree数组统计度为奇数的点的个数
    for(int i=1;i<=G.numVertices;i++)
        if(degree[i]%2==1)
            tot++;
    //检查度为奇数的点个数是否为0或者2
    if(tot==0||tot==2)//存在EL路径
        return 1;
    return 0;//存在EL路径
}
```

时间复杂度为 $O(n^2)$,空间复杂度为 $O(n)$(开辟 $degree$数组).

由于 $degree$可以在边计算完 i的度后可直接计算,故优化后时间复杂度依然是 $O(n^2)$,空间复杂度可优化为 $O(1)$.

```
int IsExistEL(MGraph G)
{
    int tot=0;
    for(int i=1;i<=G.numVertices;i++)//遍历无向图统计所有点的度
    {
        int degree=0;
        for(int j=1;j<=G.numVertices;j++)
            degree+=G.Edge[i][j];
        if(degree%2==1)//统计度为奇数的点的个数
            tot++;
    }
    //检查度为奇数的点个数是否为0或者2
    if(tot==0||tot==2)//存在EL路径
        return 1;
    return 0;//存在EL路径
}
```