7.2 图的存储结构

- 图的存储表示:

1. 图的数组(邻接矩阵)存储表示
2. 图的邻接表存储表示
3. 有向图的十字链表存储表示
4. 无向图的邻接多重表存储表示

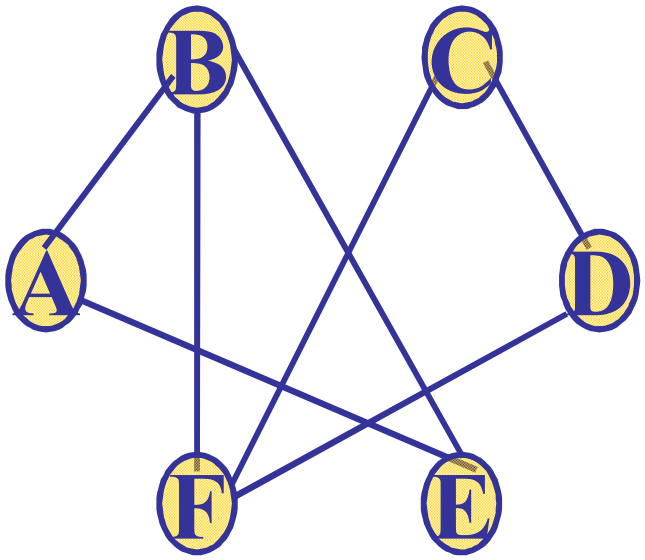
- 设计图的存储表示，应考虑方便以下操作:
  - 求入度，出度
  - 求邻接顶点
  - 判断顶点之间是否有边相连，等

无向图邻接矩阵是对称阵 可以利用对称阵的压缩存储方法存储

7.2 图的存储结构--数组(邻接矩阵)存储表示

- *n*个顶点的图用*n***n*的矩阵A存放顶点之间的逻辑关系（即：边）
- 设无向图G=(V,E), A[i][j]表示顶点i和j之间是否存在连边

$$A[i][j]=\begin{cases}1, & (i,j)\in E(G)\\ 0, & else\end{cases}$$



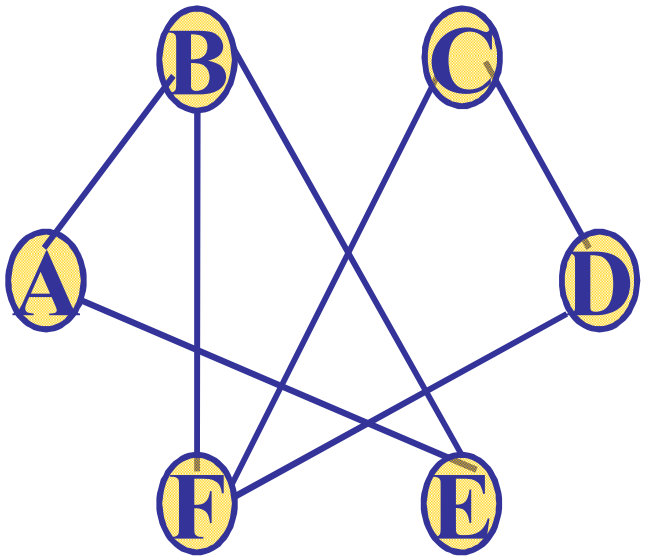
$$\begin{array}{c} \\ A\\ B\\ C\\ D\\ E\\ F\end{array}\begin{bmatrix} A & B & C & D & E & F\\ 0 & 1 & 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1 & 0 & 1\\ 0 & 0 & 1 & 0 & 0 & 1\\ 1 & 1 & 0 & 0 & 0 & 0\\ 0 & 1 & 1 & 1 & 0 & 0\end{bmatrix}$$

7.2 图的存储结构--数组(邻接矩阵)存储表示

- **无向**图顶点v_i的度

$$TD(v_i)=\sum_{j=0}^{n-1}A[i][j]$$

$$TD(v_i)=\sum_{j=0}^{n-1}A[j][i]$$

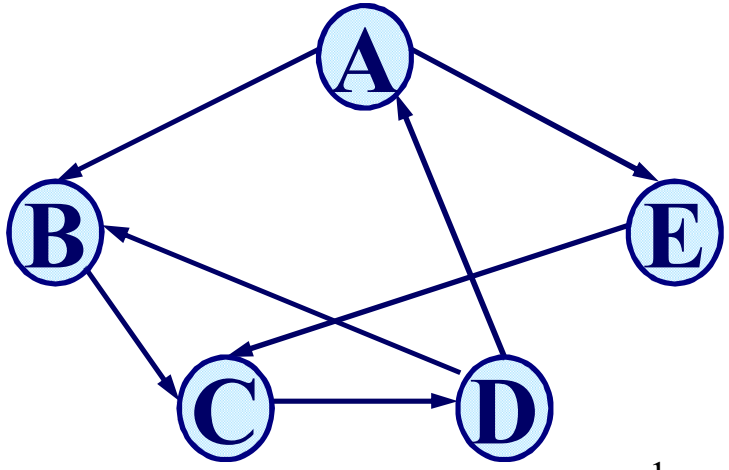


$$\begin{bmatrix}0&1&0&0&1&0\\1&0&0&0&1&1\\0&0&0&1&0&1\\0&0&1&0&0&1\\1&1&0&0&0&0\\0&1&1&1&0&0\end{bmatrix}$$

7.2 图的存储结构--数组(邻接矩阵)存储表示

- 设有向图G=(V,E), A[i][j]表示是否存在顶点i流向顶点j的弧

$$A[i][j]=\begin{cases}1, & <i,j>\in E(G)\\ 0, & else\end{cases}$$



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

顶点v_i的入度 $ID(v_i)=\sum_{j=0}^{n-1}A[j][i]$

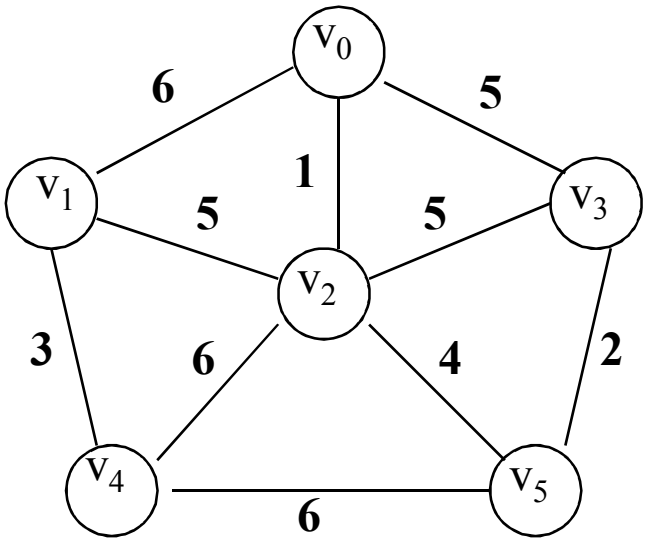
顶点v_i的出度 $OD(v_i)=\sum_{j=0}^{n-1}A[i][j]$

7.2 图的存储结构--数组(邻接矩阵)存储表示

- 无向网的邻接矩阵

$$A[i][j]=\begin{cases} w_{ij}, & (i,j)\in E(G), \\ 0, & else \end{cases}$$

w_{ij}表示在顶点i和j的连边上的权值



$$\begin{bmatrix} 0 & 6 & 1 & 5 & 0 & 0 \\ 6 & 0 & 5 & 0 & 3 & 0 \\ 1 & 5 & 0 & 5 & 6 & 4 \\ 5 & 0 & 5 & 0 & 0 & 2 \\ 0 & 3 & 6 & 0 & 0 & 6 \\ 0 & 0 & 4 & 2 & 6 & 0 \end{bmatrix}$$

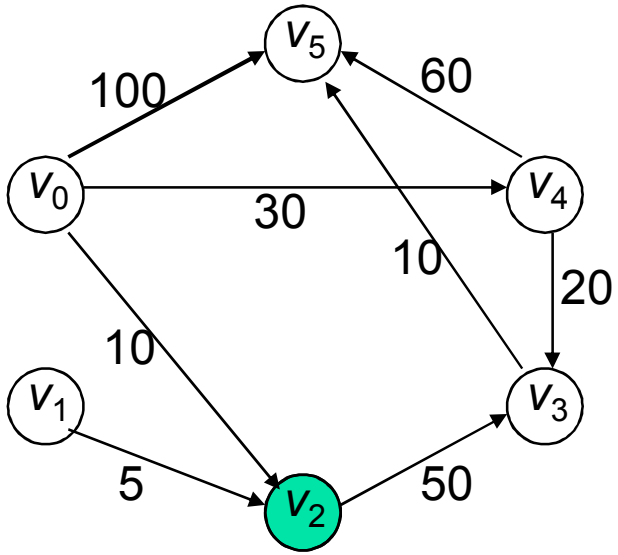
有时：也用∞代表没有边

7.2 图的存储结构--数组(邻接矩阵)存储表示

- 有向网的邻接矩阵

$$A[i][j]=\begin{cases} w_{ij}, & <i,j>\in E(G) \\ 0, & else \end{cases}$$

w_{ij}表示在顶点i流向j的弧上的权值



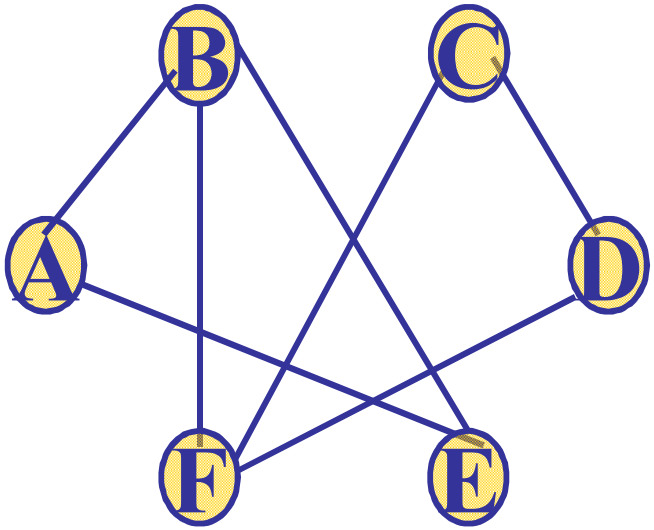
$$\begin{bmatrix} 0 & 0 & 10 & 0 & 30 & 100 \\ 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 20 & 0 & 60 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

有时：也用∞代表没有边

7.2 图的存储结构--数组(邻接矩阵)存储表示

```
#define MAXSIZE 100
typedef struct {
    VertexType      vexs[MAXSIZE]; //存放顶点信息
    int    arcs[MAXSIZE][MAXSIZE];//邻接矩阵
     int    vexnum, arcnum;  //顶点数和边数
     int   kind;  //图的类型
 } MGraph;
MGraph T;
```

7.2 图的存储结构--数组(邻接矩阵)存储表示



T.arcs[ ][ ]

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

T.vexs[ ]

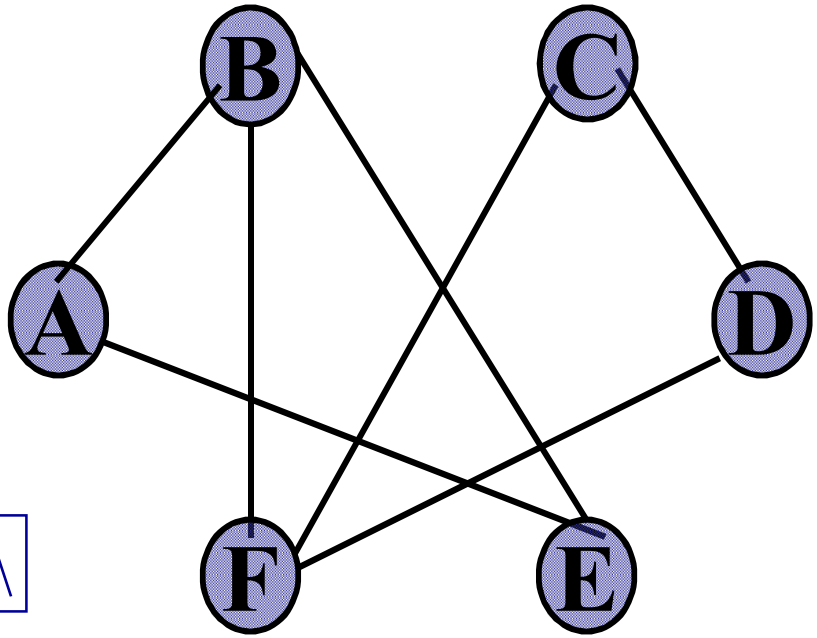
A
B
C
D
E
F

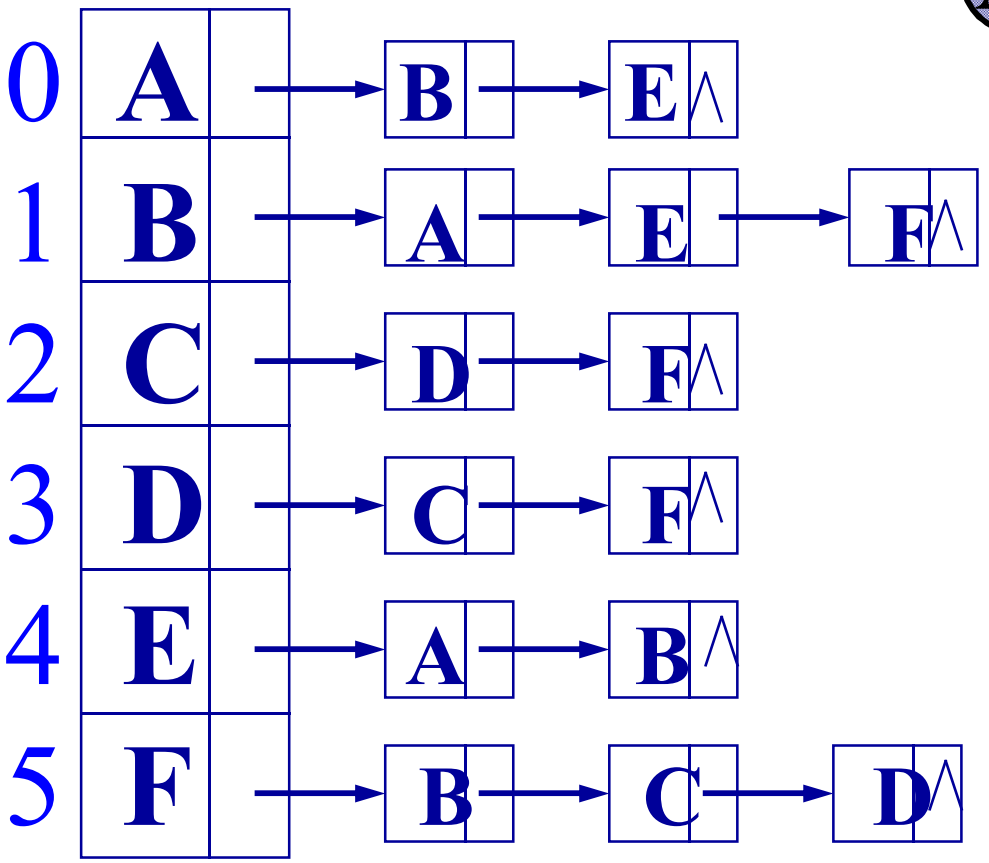
7.2 图的存储结构--邻接表存储表示

- 图的一种链式存储结构,对图中每个顶点建立一个单链表,第i个单链表中的结点是与顶点v_i相关联的边或弧。
- 每个单链表增设一个表头结点，其数据域（data）保存顶点信息，链域(link)指向表中的第一个结点。
- 所有单链表的表头结点用一个数组存放。

图的存储结构--邻接表存储表示

无向图的邻接表，第i个单链表中的结点是与顶点v_i相关联的边。

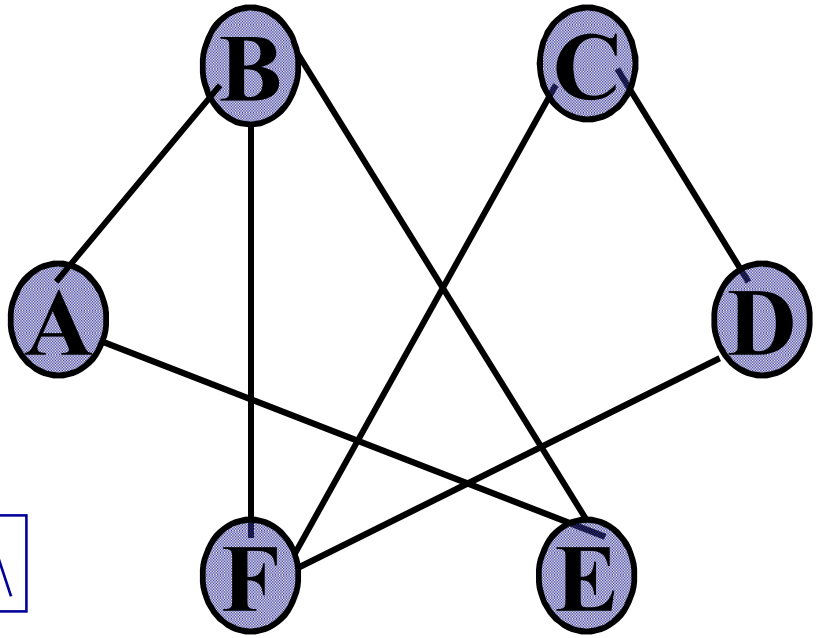


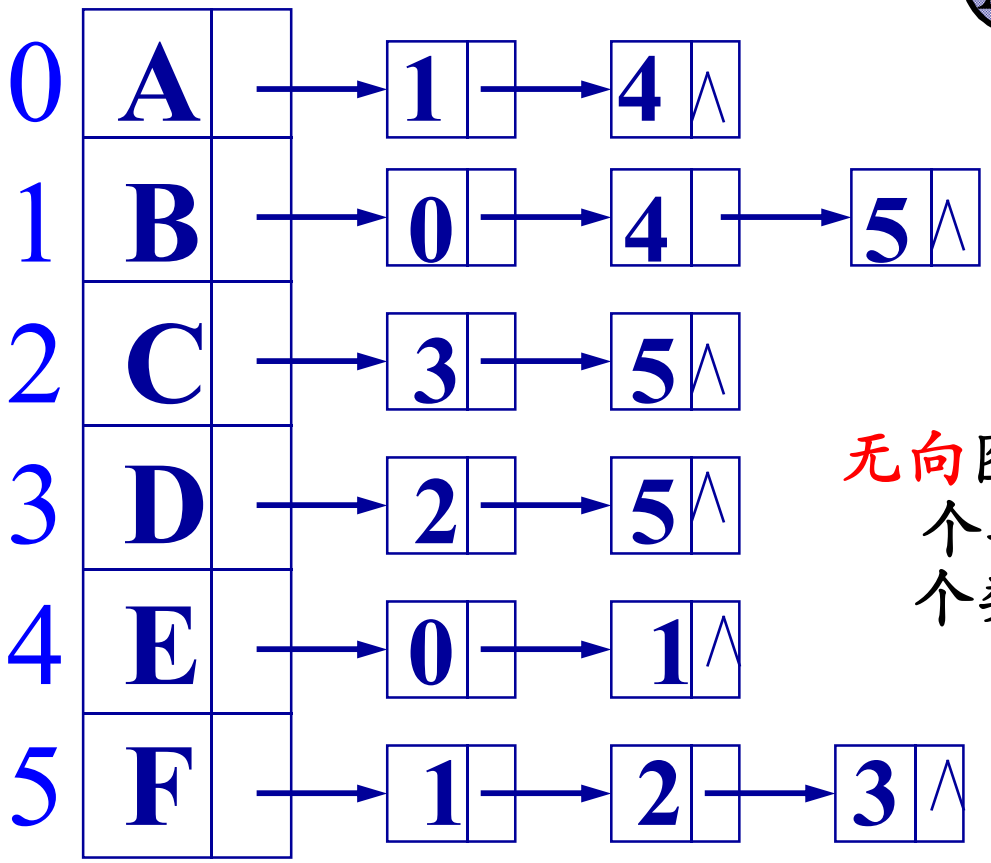


图的存储结构--邻接表存储表示

无向图的邻接表，第i个单链表中的结点是与顶点v_i相关联的边。

无向图的邻接表，一条边(i,j)存2次，在第i和第j个单链表中**同时存在**。



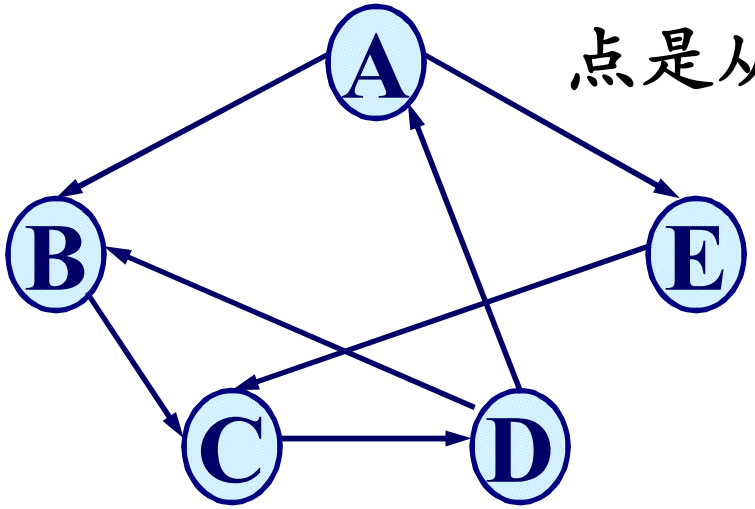


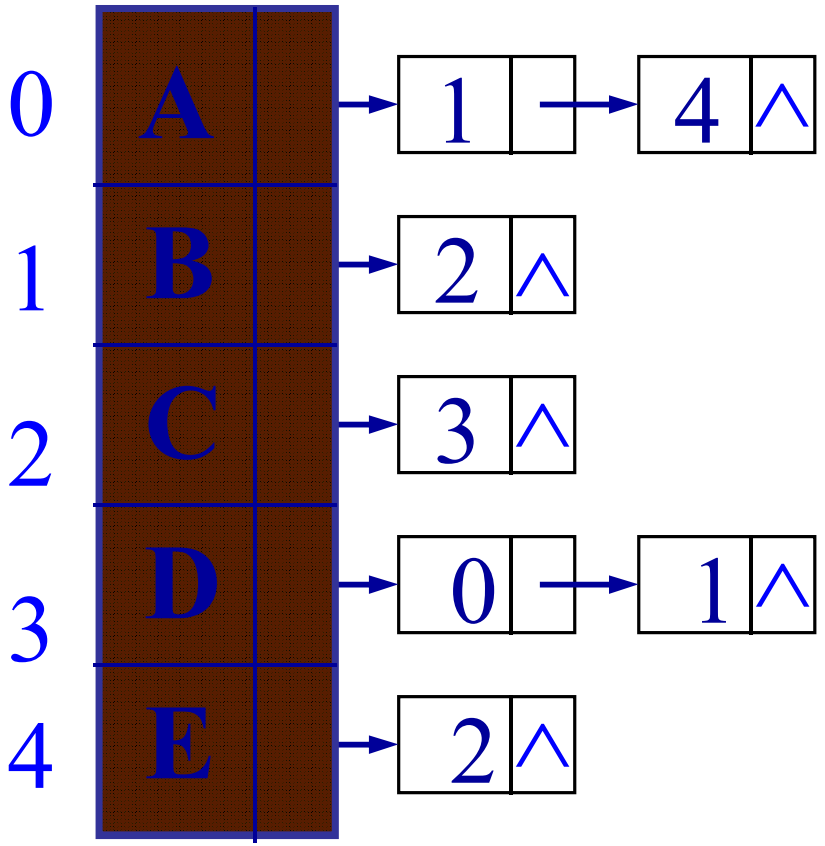
无向图的邻接表，顶点v_i的度是第i个单链表中含有的数据元素的个数，即第i个单链表的长度。

图的存储结构--邻接表存储表示

有向图的邻接表：第i个单链表中的结点是从顶点v_i流出的弧**流向**的顶点

一条弧只存一次



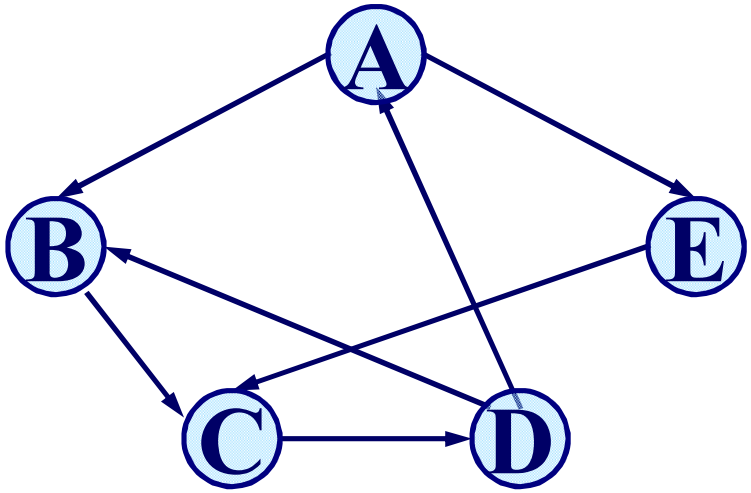


邻接表

在有向图的邻接表中不易找到指向该顶点的弧。

顶点v_i的**出度**是第i个单链表中含有的数据元素的个数，即第i个单链表的长度。 如何求入度?

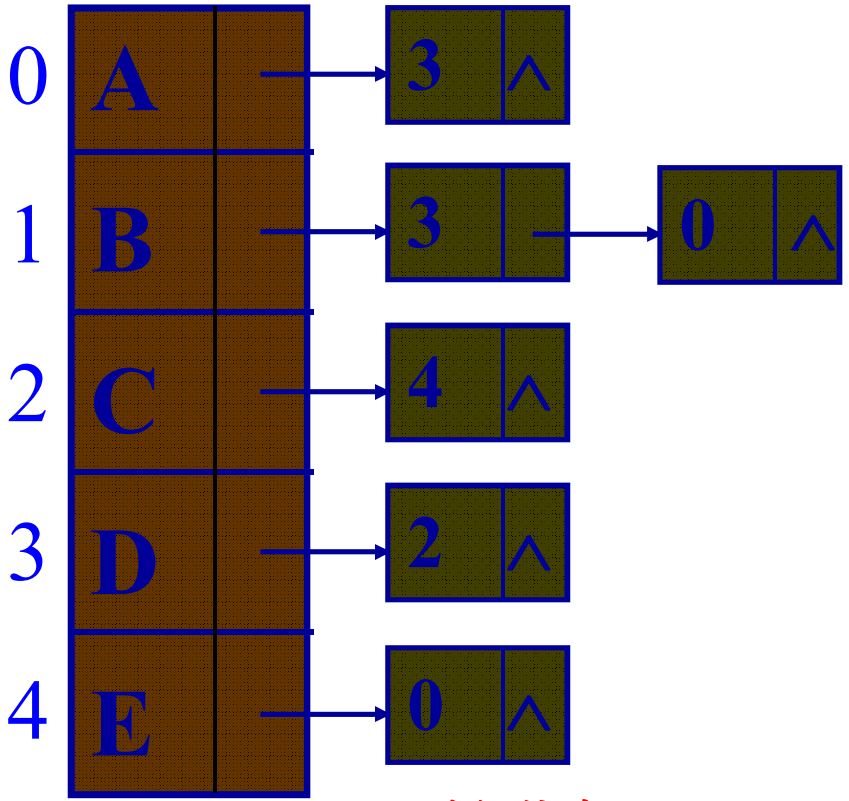
图的存储结构--邻接表存储表示



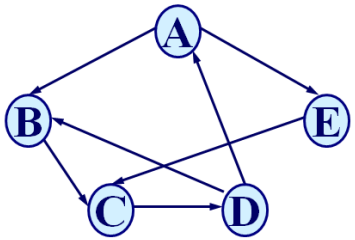
若图的边比较稀疏，则用邻接表比邻接矩阵节省存储空间。

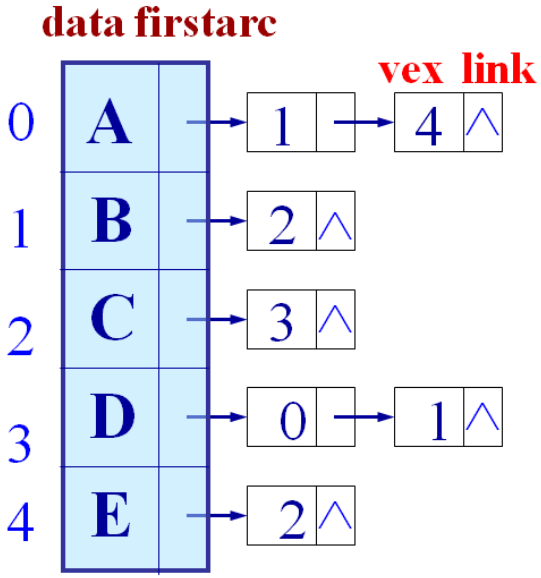
有向图的逆邻接表中，为顶点i建的单链表示流向该顶点的弧

顶点i的入度为有向图的逆邻接表中，为顶点i建的单链的长度



逆邻接表

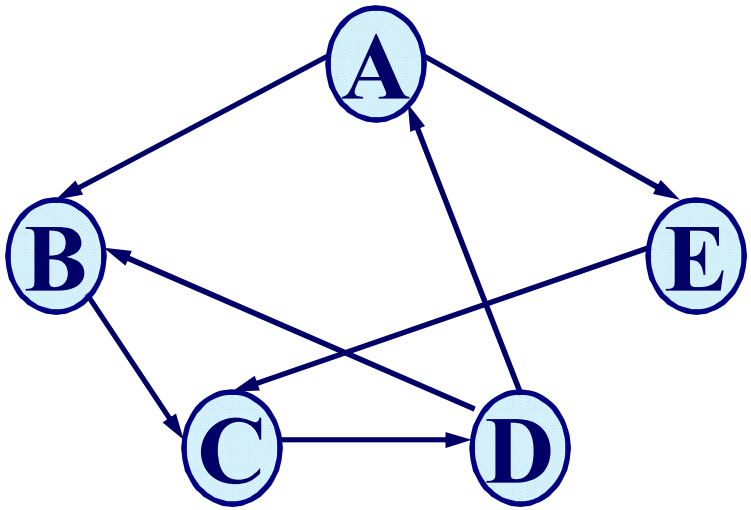


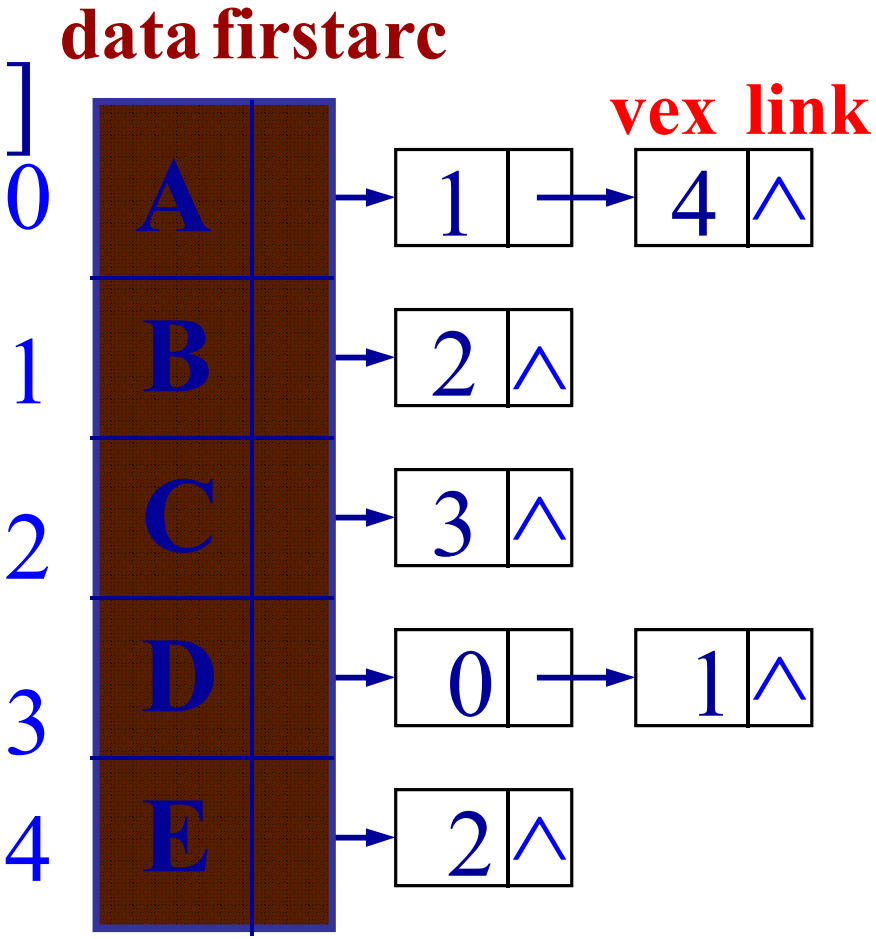



```
typedef struct ArcNode {
  int       vex;   // 该弧所指向的顶点的位置
  struct ArcNode  *link;  // 指向下一条弧的指针
  InfoType   *info;   // 该弧相关信息的指针
} ArcNode;
typedef struct VNode {
  VertexType  data;   // 顶点信息
  ArcNode  *firstarc; // 指向第一条依附该顶点的弧
  } VNode;
 typedef struct {
     VNode    arc[MAXSIZE];
     int      vexnum, arcnum;
     int      kind; //图的类型
  } Graphs;
```

图的存储结构--邻接表存储表示

Graphs T; T.arc[ ]





一条弧位于2个单链表，但只存一次

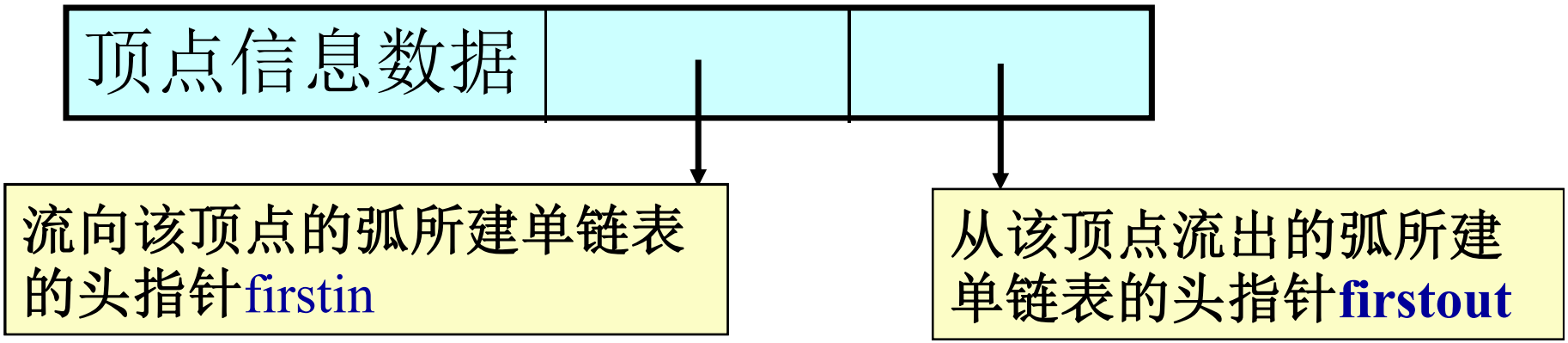
7.2 图的存储结构--有向图的十字链表存储表示

弧的结点结构

每个顶点建2个单链表：流出去的弧建一个单链表，流入的弧建一个单链表。

```
typedef struct ArcBox { // 弧的结构表示
    int tailvex, headvex;   InfoType  *info;
    struct ArcBox  *hlink, *tlink;
    } ArcBox;
```

一维数组元素的类型

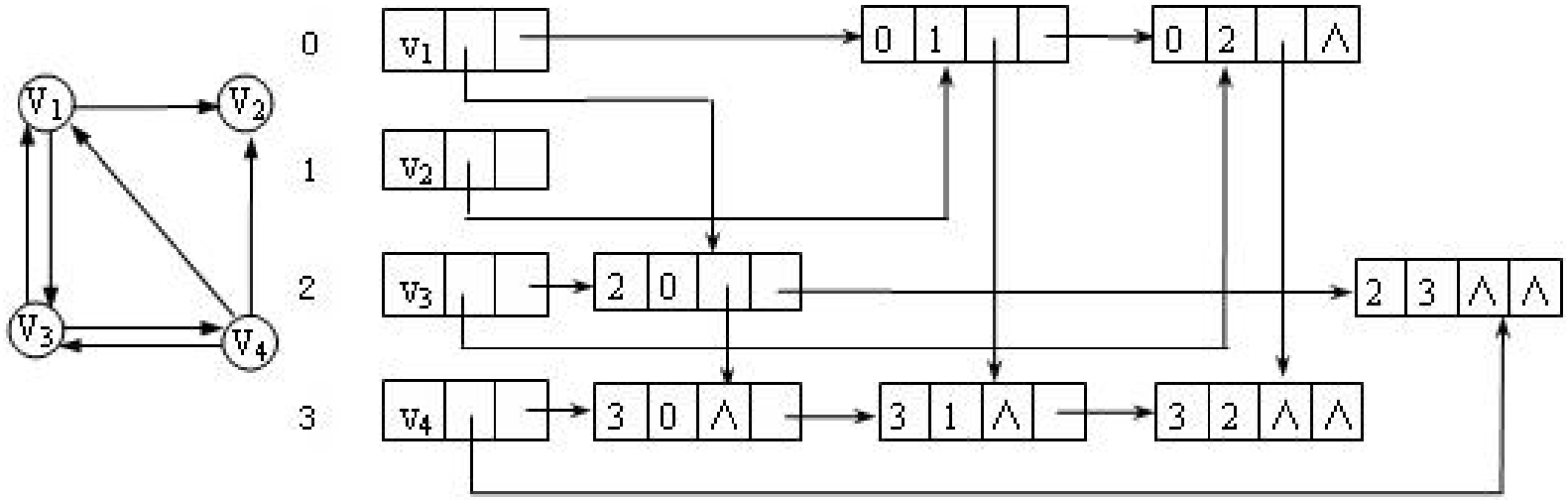



```
typedef struct VexNode { // 顶点的结构表示
    VertexType  data;
    ArcBox  *firstin, *firstout;
} VexNode;
```

一维数组保存每个顶点的信息和相应2个单链表的头指针

有向图的十字链表存储表示

```
typedef struct {
   VexNode  xlist[MAXSIZE];  // 顶点信息
   int   vexnum, arcnum; //有向图的当前顶点数和弧数
} OLGraph;
```



（a） 一个有向图G4　　（b） 有向图的十字链表

7.2 图的存储结构--无向图的邻接多重表存储表示

边的结构表示

```
typedef struct Ebox {
    int       mark;
    int       ivex, jvex;
    struct EBox  *ilink, *jlink;
} EBox;
```

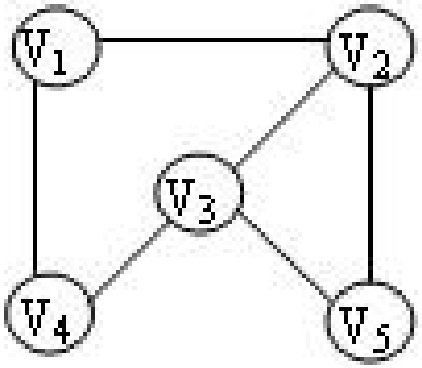
每个顶点建1个单链表：与该顶点关联的边建一个单链表

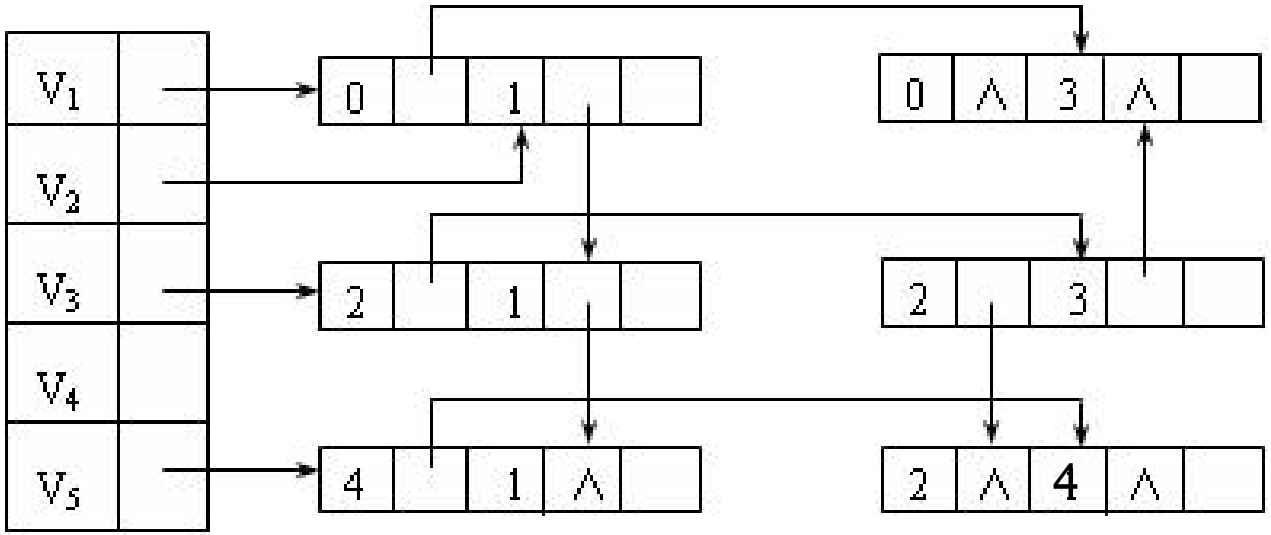
一条边位于2个单链表，但只存一次

mark	ivex	ilink	jvex	jlink

无向图的邻接多重表存储表示

一维数组保存每个顶点的信息和相应单链表的头指针





无向图的邻接多重表存储表示

数组元素类型

```
typedef struct VexBox {
   VertexType  data;
   EBox  *firstedge; // 相应顶点单链表的头指针
} VexBox;
```

无向图邻接多重表存储表示

```
typedef struct {  // 邻接多重表
    VexBox  adjmulist[MAXSIZE];
    int   vexnum, edgenum;
  } AMLGraph;
```