第7章　图

一、图的定义和术语

1、什么是图　　>>观看视频

图(Graph)体现的是一种多对多的关系

由顶点的有穷非空集合和顶点之间边的集合组成，通常表示为G（V，E）。

教材P200

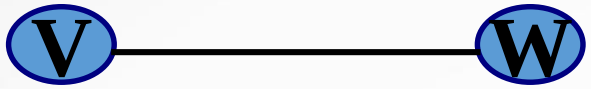
顶点(Vertex):数据元素

在图中，任意两个顶点之间都可能存在关系，顶点之间的逻辑关系用边（edge）来表示,边集可以为空。

2、图的术语（教材P200-201）

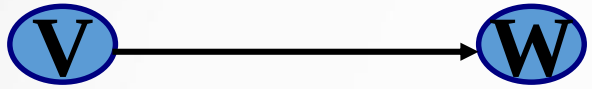
- 无向边

若顶点v到w之间的**边没有方向**，则称这条边为无向边，用无序偶对**(v, w)**来表示。



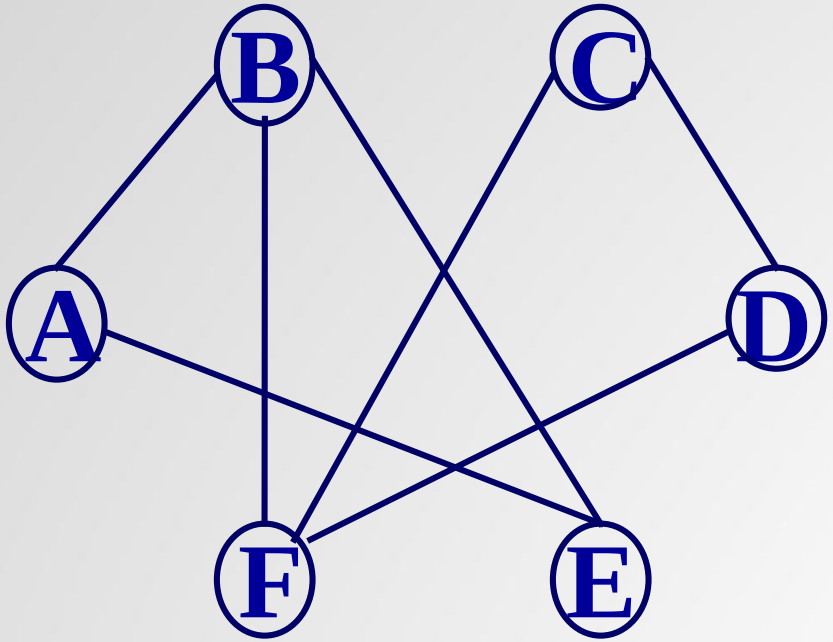
- 有向边（弧）

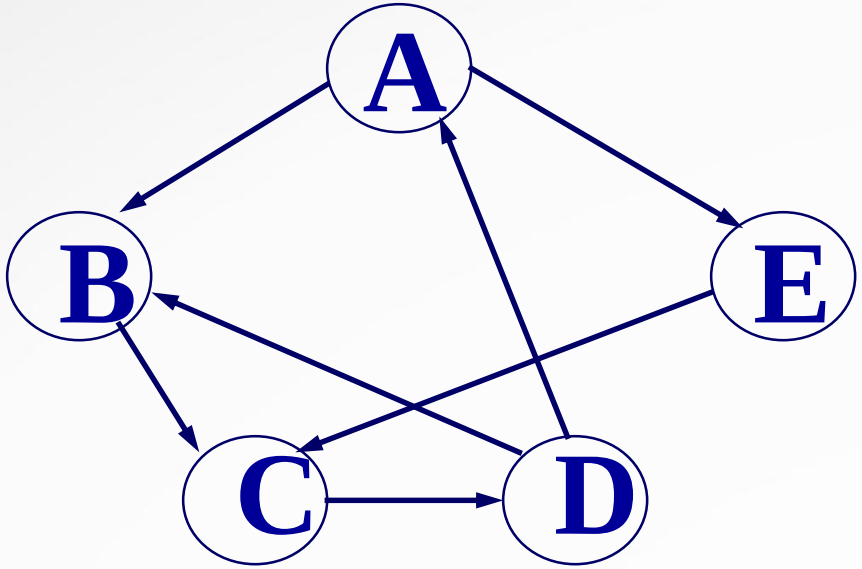
若顶点v到w之间的**边有方向**，则称这条边为有向边，也成为弧（Arc），用有序偶对**<v, w>**来表示。



v 为弧尾，w 为弧头。

- 无向图 由顶点集和边集构成的图
- 有向图 由顶点集和弧集构成的图





例如: G1=(V1,{VR1})

V1={A, B, C, D, E, F}

VR1={(A,B), (A,E),(B,E), (C,D),

(D,F), (B,F), (C,F) }

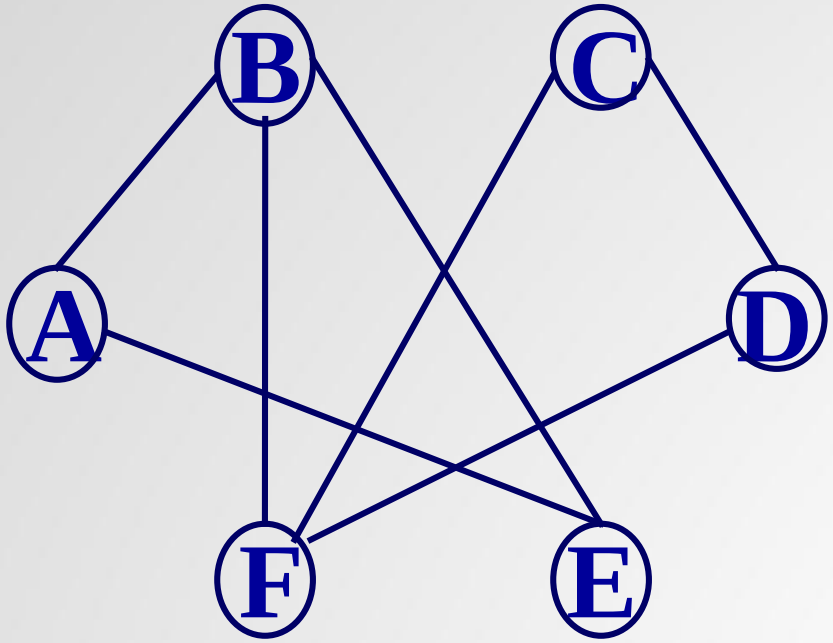
例如:G2 = (V2, {VR2})

V2={A, B, C, D, E}

VR2={<A,B>, <A,E>,<B,C>,

<C,D>, <D,B>,<D,A>, <E,C> }

完全图、稀疏图、稠密图



假设图中有 n 个顶点，e 条边

完全图:

含有 e=n(n-1)/2 条边的无向图;

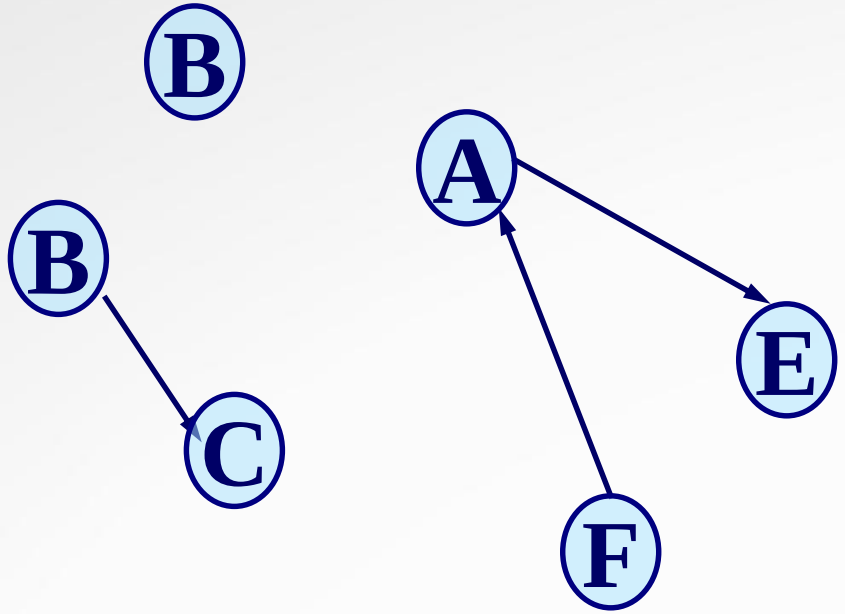
有向完全图:

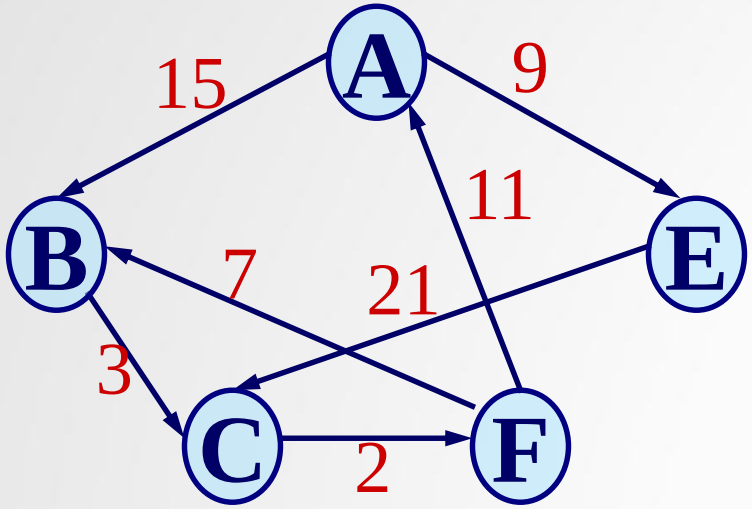
含有 e=n(n-1) 条弧的有向图;

稀疏图:边或弧的数量 e<nlogn;否则称作稠密图。

子图

设图G=(V,{VR}) 和图G′=(V′,{VR′}),且 V′⊆V,VR′⊆VR,则称 G′ 为 G 的子图。

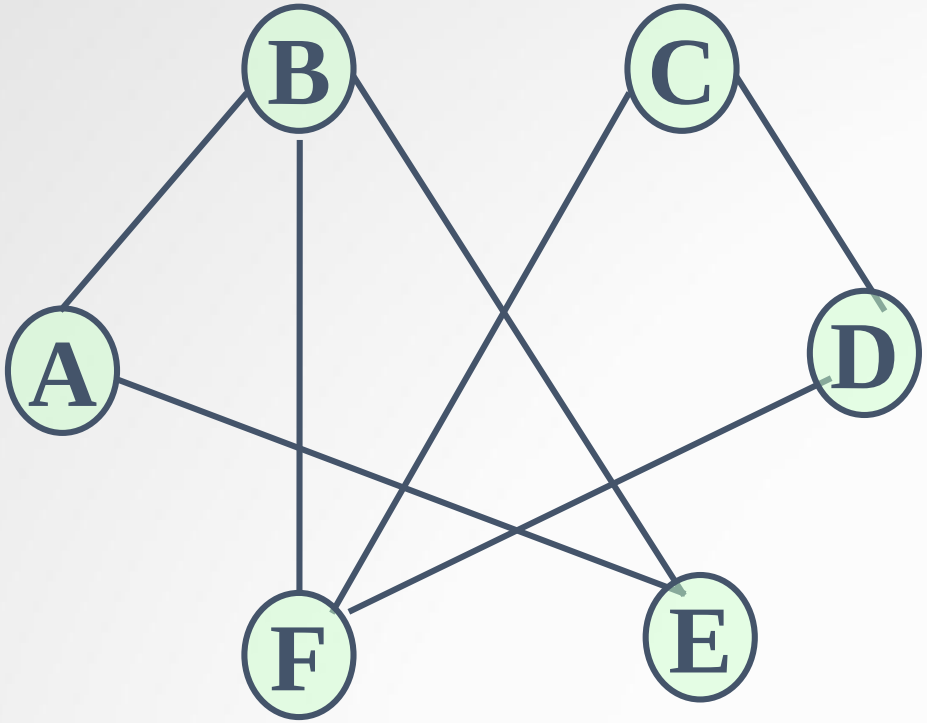


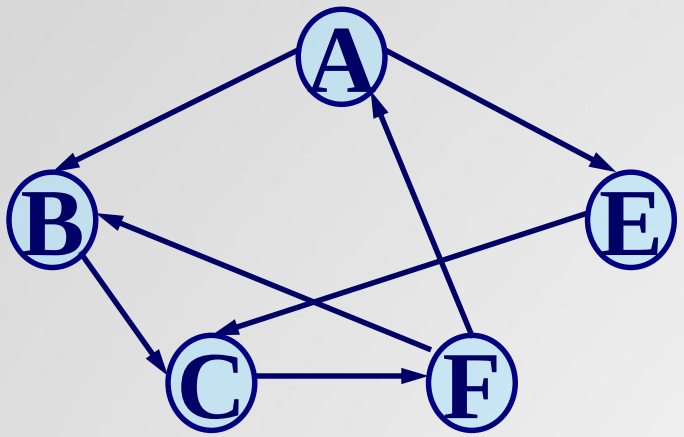


网

弧或边带权的图分别称作有向网或无向网。

- 邻接点、度、入度、出度





顶点的出度:

以顶点v为弧尾的弧的数目;

顶点的入度:

以顶点v为弧头的弧的数目。

顶点的度(TD)=出度(OD)+入度(ID)

例如:

OD(B) = 1

ID(B) = 2

TD(B) = 3

图的模拟—航空系统

顶点

边

有向图

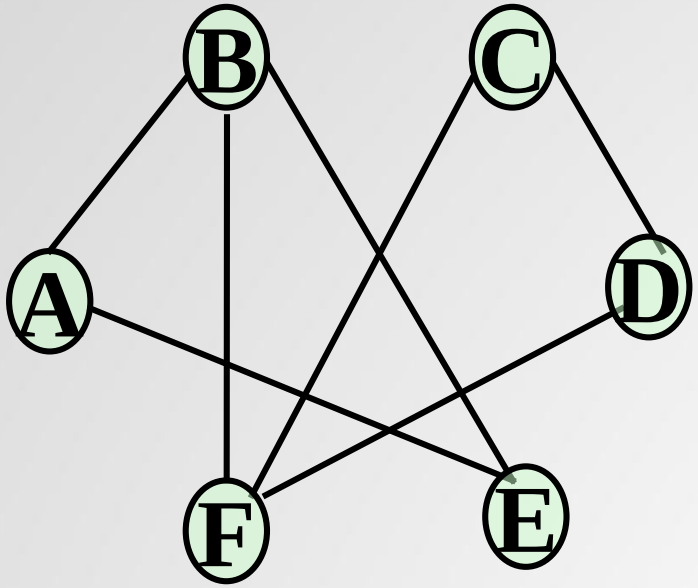
邻接点

出度、入度

【课堂作业】 P259 6.3 (1)、(2)

6.4 (1)

二、图的存储表示

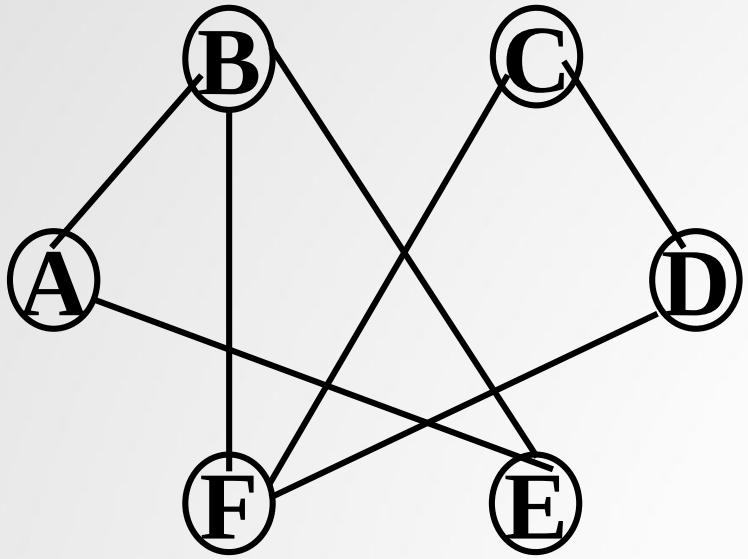


- 不可能用简单的顺序存储结构
- 多重链表存在的问题

常用的存储结构有：邻接矩阵、邻接表、十字链表、邻接多重表

1、邻接矩阵（教材P206）

用两个数组来表示图。一个一维数组存储图中顶点信息，一个二维数组（称为邻接矩阵）存储图中的边或弧的信息。

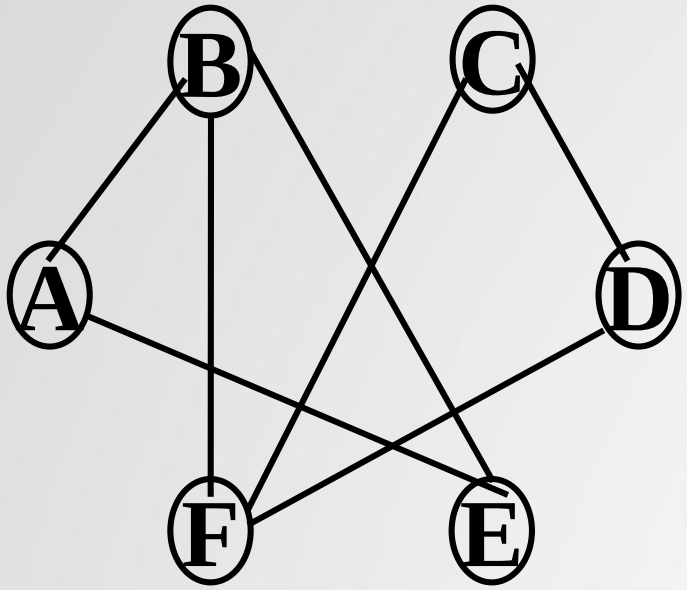


顶点数组

A	B	C	D	E	F

定义矩阵的元素为 $A_{ij}=\begin{cases}0 & (i,j)\notin E\\ 1 & (i,j)\in E\end{cases}$

边数组：



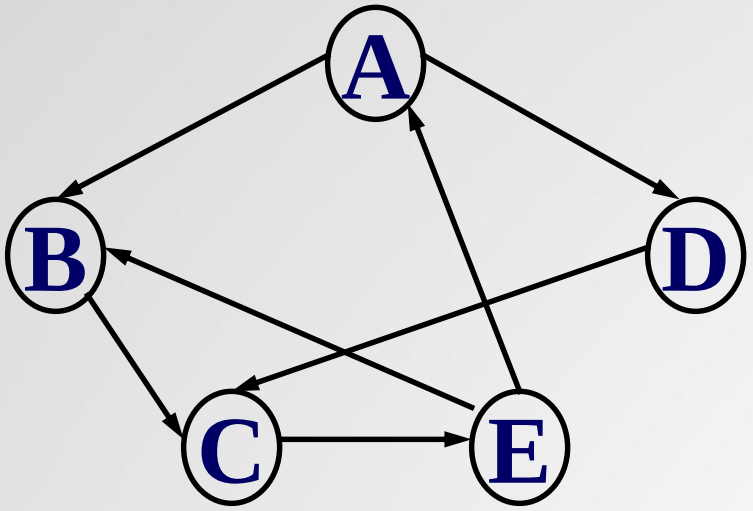
	A	B	C	D	E	F
A	0	1	0	0	1	0
B	1	0	0	0	1	1
C	0	0	0	1	0	1
D	0	0	1	0	0	1
E	1	1	0	0	0	0
F	0	1	1	1	0	0

无向图的邻接矩阵为对称矩阵

	A	B	C	D	E	F
A	0	1	0	0	1	0
B	1	0	0	0	1	1
C	0	0	0	1	0	1
D	0	0	1	0	0	1
E	1	1	0	0	0	0
F	0	1	1	1	0	0

矩阵中的信息：

- 容易判断任意两顶点是否有边无边；
- 顶点的度：二维数组对应行（或列）中1的个数；
- 求顶点vi的所有邻接点就是将矩阵中第i行元素扫描一遍，a_{ij}为1就是邻接点；
- 顶点不变，在图中增加、删除边：只需对二维数组对应分量赋值1或清0；



	A	B	C	D	E
A	0	1	0	1	0
B	0	0	1	0	0
C	0	0	0	0	1
D	0	0	1	0	0
E	1	1	0	0	0

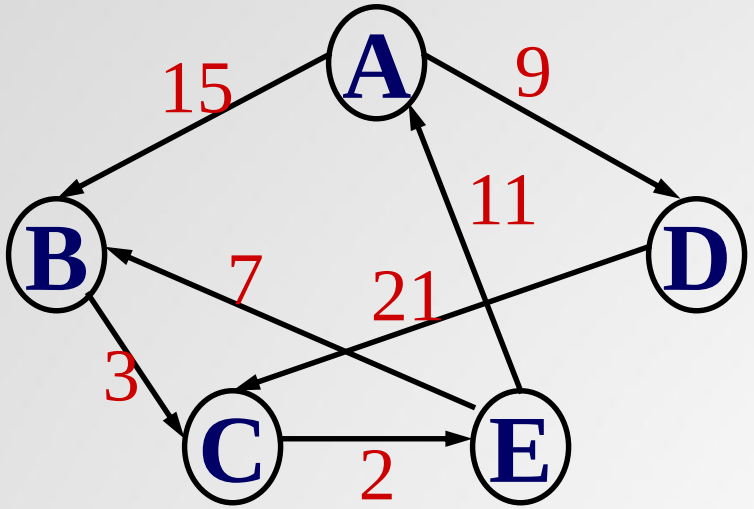
有向图的邻接矩阵为非对称矩阵

顶点的度：

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。

网图的邻接矩阵



$$A.edge = \begin{bmatrix} 0 & 15 & \infty & 9 & \infty \\ \infty & 0 & 3 & \infty & \infty \\ \infty & \infty & 0 & \infty & 2 \\ \infty & \infty & 21 & 0 & \infty \\ 11 & 7 & \infty & \infty & 0 \end{bmatrix}$$

$$\mathbf{A.edge}[i][j] = \begin{cases} \mathbf{W}(i,j), & \text{若} i \neq j \text{且} < i,j > \in \mathbf{E} \text{或} (i,j) \in \mathbf{E} \\ \infty, & \text{若} i \neq j \text{且} < i,j > \notin \mathbf{E} \text{或} (i,j) \notin \mathbf{E} \\ \mathbf{0}, & \text{若} i == j \end{cases}$$

【课堂作业】 P259 6.3 (3)

6.4 (2)

6.5 (1)

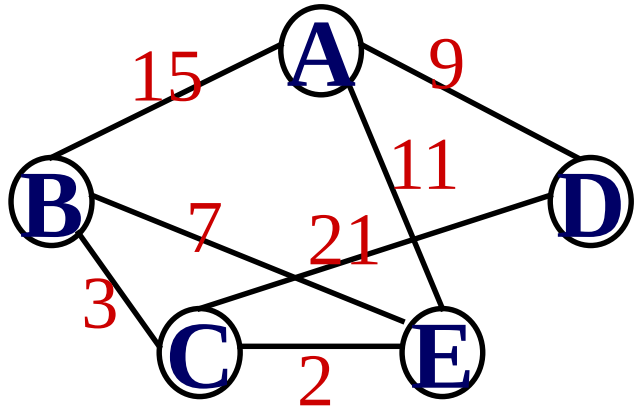
邻接矩阵的代码：（教材P208）

```
typedef struct
{
    VertexType vexs[MAXVEX];    //顶点表
    EdgeType    arc[MAXVEX][MAXVEX];  // 邻接矩阵，边表
    int     vexnum, edgenum;    // 顶点数和边数
  } MGraph;
    typedef char VertexType;
    typedef int EdgeType;
    #define MAXVEX 100          //最大顶点数
    #define INFINITY 65535      //用65535来代表 ∞
```

建立无向图：

```
void CreatMGraph(MGraph *G)
{
  int i,j,k,w;
  printf("输入顶点数和边数：\n");
  scanf("%d,%d",&G->vexnum,&G->edgenum);
  for(i=0;i<G->vexnum;i++)   //读入顶点信息,建立顶点表
      scanf("%c",&G->vex[i]);
  for(i=0;i<G->vexnum;i++)
      for(j=0;j<G->vexnum;j++)
          G->arc[i][j]=INFINITY;    //邻接矩阵初始化
  for (k=0; k<G->edgenum;k++)
  {
    printf("输入边(vi,vj)上的下标i， 下标j和权w:\n");
    scanf("%d,%d,%d",&i,&j,&w);   //输入边(vi,vj)上的权w
    G->arc[i][j]=w;
    G->arc[j][i]=G->arc[i][j];    //无向图，矩阵对称
   }
}
```

算法的时间复杂度？ $O(n+n^2+e)=O(n^2)$

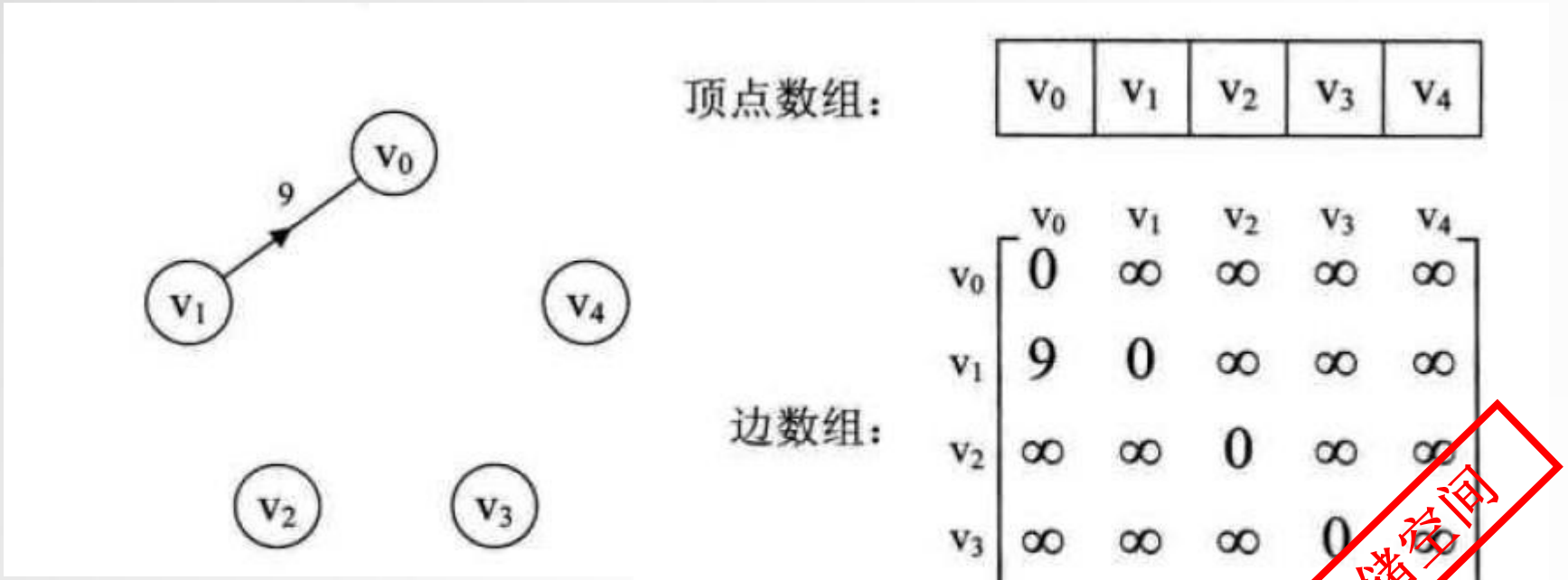


$$=\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

$$=\begin{bmatrix} 0 & 15 & \infty & 9 & 11 \\ 15 & 0 & 3 & \infty & 7 \\ \infty & 3 & 0 & 21 & 2 \\ 9 & \infty & 21 & 0 & \infty \\ 11 & 7 & 2 & \infty & 0 \end{bmatrix}$$

【邻接矩阵的问题】

对于边数相对顶点数较少的图（稀疏图）

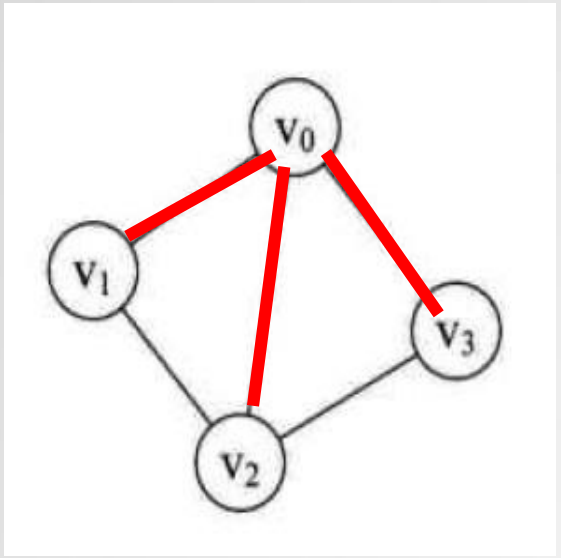


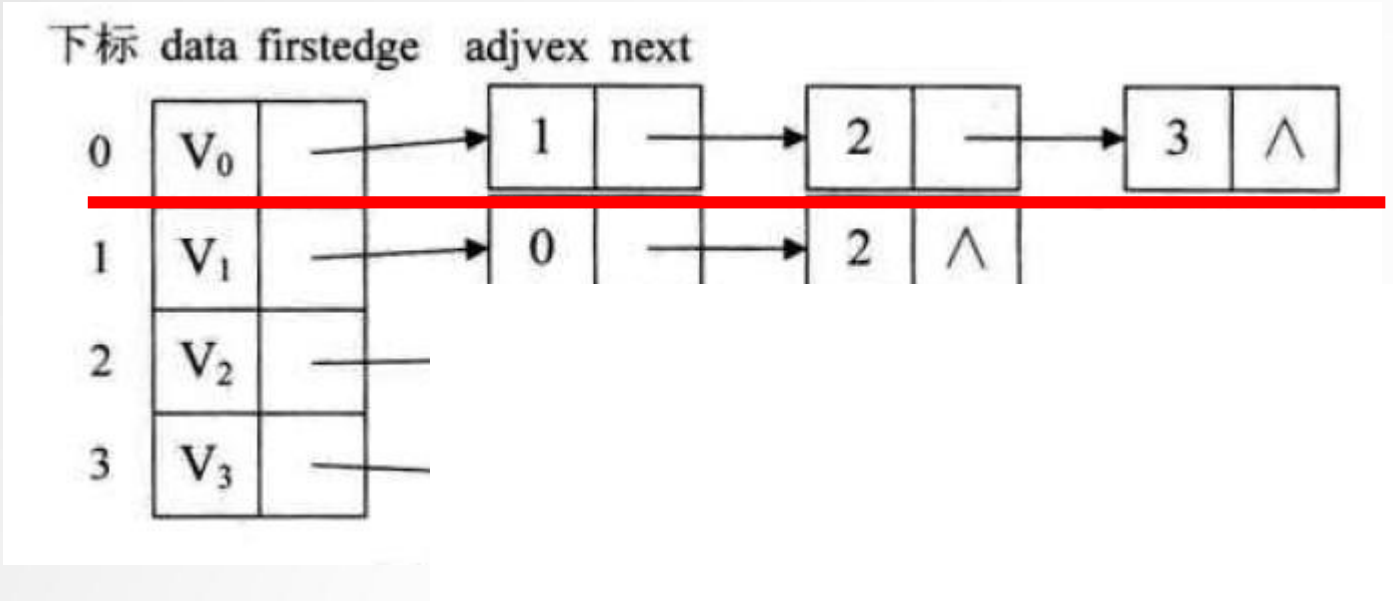
浪费存储空间

【解决办法】使用邻接表

2、邻接表（教材P210）

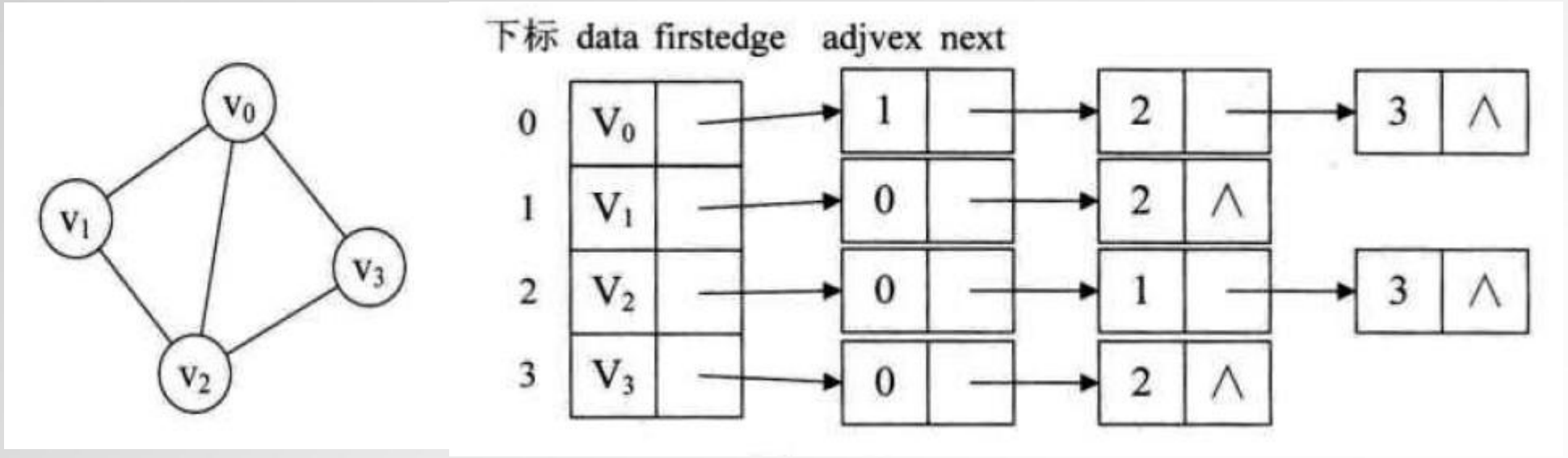
- 对图中每个顶点建立一个单链表；
- 第i个单链表中的结点表示依附于顶点vi的边





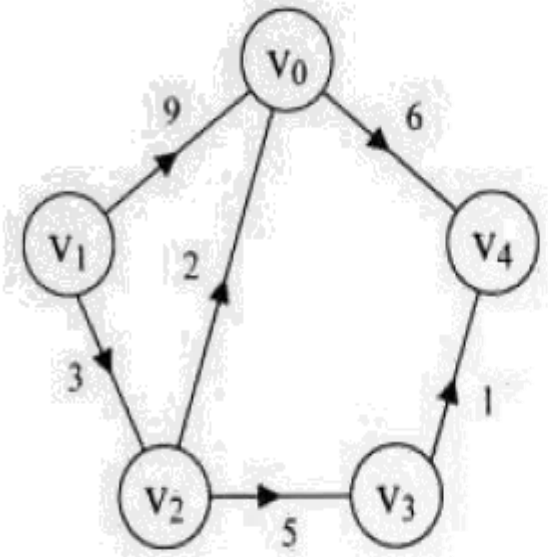
data存储顶点信息
firstedge指向边表的第一个结点。

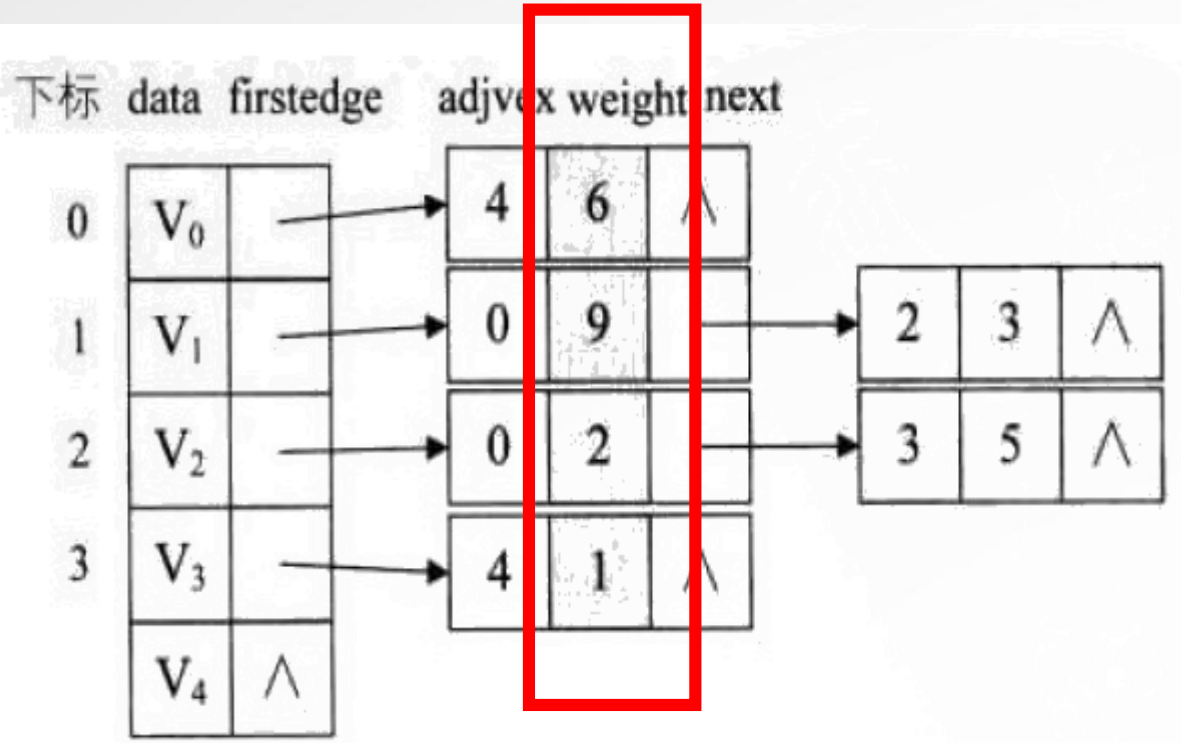
adjvex存储某顶点的邻接点在顶点表中的下标，next则存储指向边表中下一个结点的指针。



邻接表中的信息：

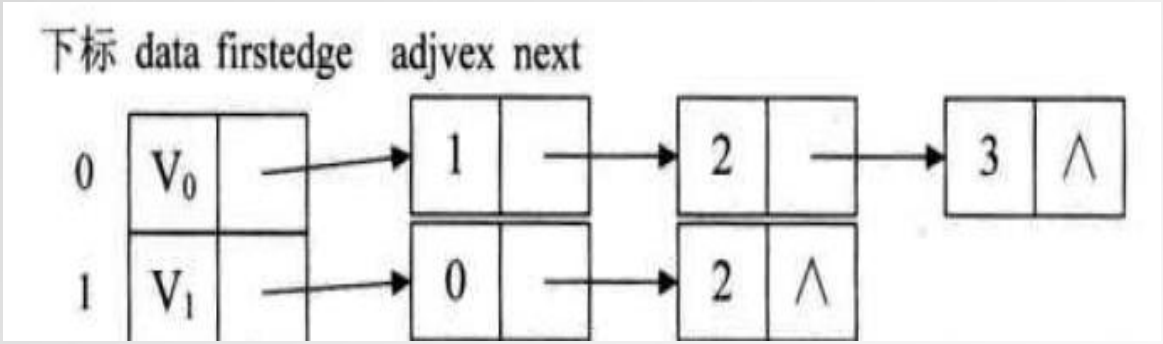
- 顶点的度：
  边表中结点的个数；
- 判断顶点vi和vj之间是否存在边
  只需要测试顶点vi的边表中adjvex是否存在vj的下标j就行了。
- 求顶点vi的所有邻接点
  就是对顶点的边表进行遍历，得到的adjvex域对应的顶点就是邻接点；


V0
V1
V2
V3
V4
9
6
2
3
1
5


下标 data firstedge adjvex weight next
0 V0
1 V1
2 V2
3 V3
V4 ∧
4 6 ∧
0 9
0 2
4 1
2 3 ∧
3 5 ∧

【课堂作业】 P259 6.4 (3)

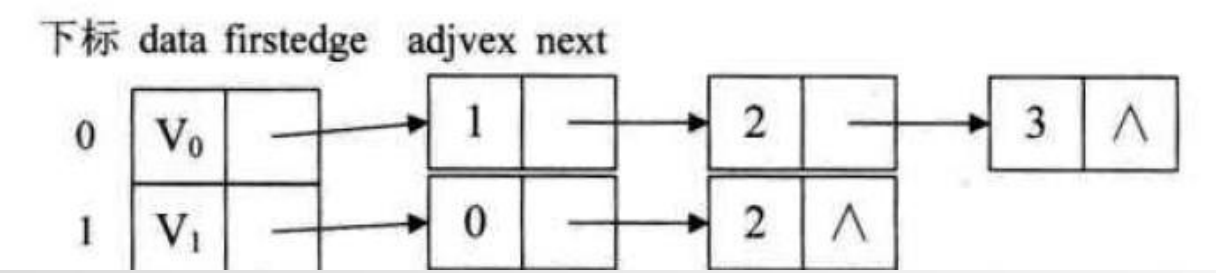
邻接表的代码：（教材P211）




```
typedef struct EdgeNode    //边表结点
{
    int adjvex;          //邻接点域，存储该顶点对应的下标
    EdgeType weight;  //存储权值
    struct EdgeNode *next;    // 指向下一个邻接点
  }EdgeNode;
```

adjvex	next

adjvex	weight	next




```
typedef struct VNode    //顶点表结点
{
   VertexType data;          //存储顶点信息
   EdgeNode *firstedge;      //边表头指针
  }VNode,AdjList[MAXVEX];
```

data	firstedge

```
            typedef struct
            {
               AdjList adjList;
               int vernum,edgenum;       // 顶点数和边数
              }ALGraph;
```

建立邻接表：

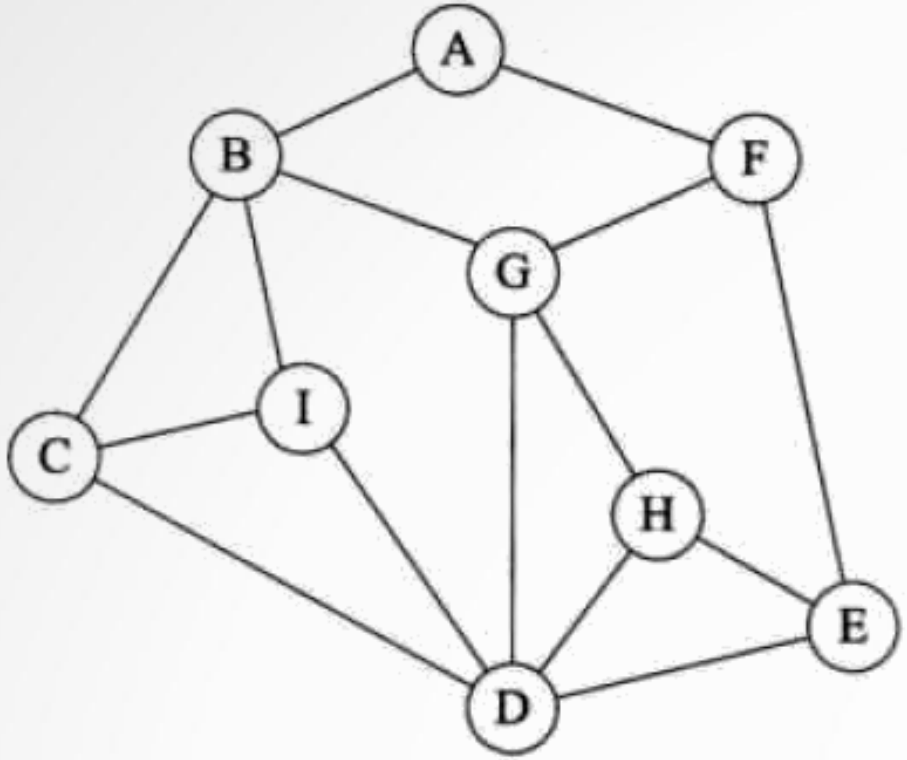
```
void CreatALGraph(ALGraph *G)
{
  int i,j,k;
  EdgeNode *e;
  printf("输入顶点数和边数：\n");
  scanf("%d,%d",&G->vernum,&G->edgenum);
  for(i=0;i<G->vernum;i++)   //读入顶点信息,建立顶点表
  {
     scanf("%c",&G->adjList[i].data);
     G->adjList[i].firstedge=NULL;   //将边表置为空表
  }
  //建立边表
```

//接上页

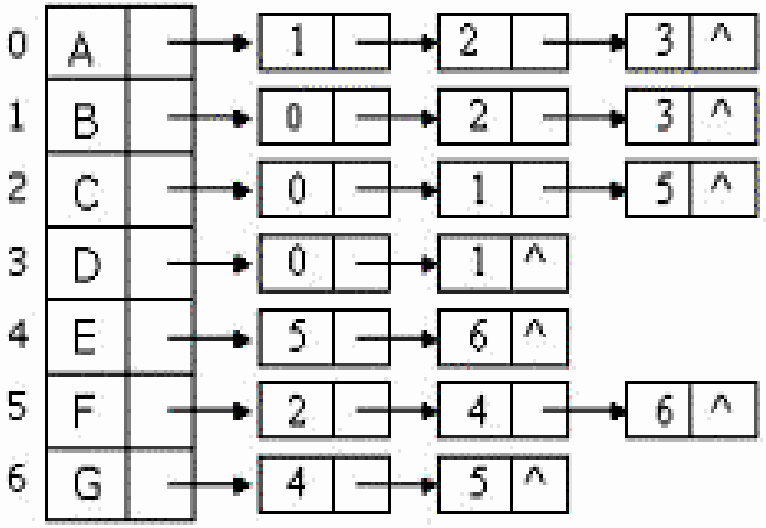
```
for (k=0; k<G->edgenum;k++)
   {
     printf("输入边(vi,vj)上顶点序号:\n");
     scanf("%d,%d",&i,&j);
     e=(EdgeNode *)malloc(sizeof(EdgeNode));
                                   //申请空间，生成边表结点
     e->adjvex=j;                  //邻接序号为j
     e->next=G->adjList[i].firstedge;
     G->adjList[i].firstedge=e;

     e=(EdgeNode *)malloc(sizeof(EdgeNode));
     e->adjvex=i;                  //邻接序号为i
     e->next=G->adjList[j].firstedge;
     G->adjList[j].firstedge=e;

   }
}
```

算法的时间复杂度? O(n+e)

【Practice】写出下图从顶点A出发，按DFS进行遍历得到的一种顶点序列。

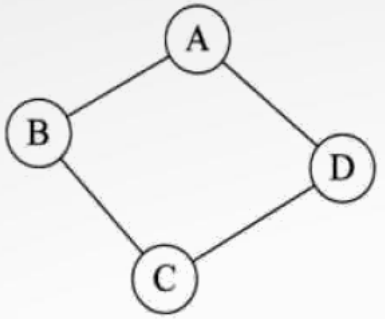


【Practice】某无向图的邻接表如下所示

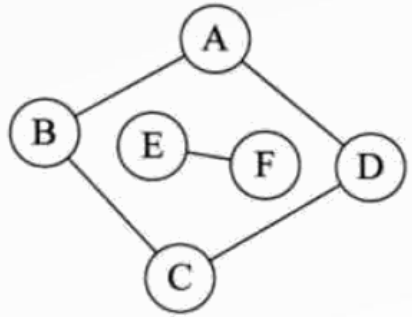


从顶点F开始对该图进行深度优先搜索，写出顶点访问序列。

//邻接表的深度优先递归算法

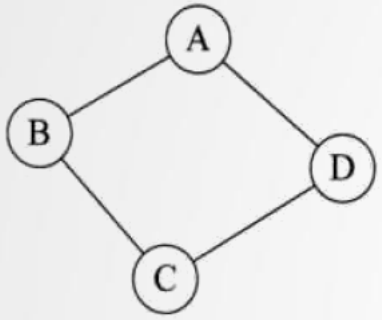


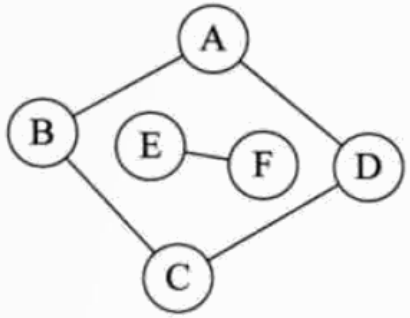

```
 int visited[N];                    //访问标志数组

void DFS(ALGraph G,int v)   //从第v个顶点出发
{
  EdgeNode *p;
  visited[v]=1;
  printf("%c",G.adjList[v].data); //打印顶点
  p=G.adjList[v].firstedge;
  while(p)
  {
     if(visited[p->adjvex]!=1)
           DFS(G,p->adjvex);   //对未访问的邻接顶点递归调用
     p=p->next;
   }
}
```



//邻接表的深度遍历操作

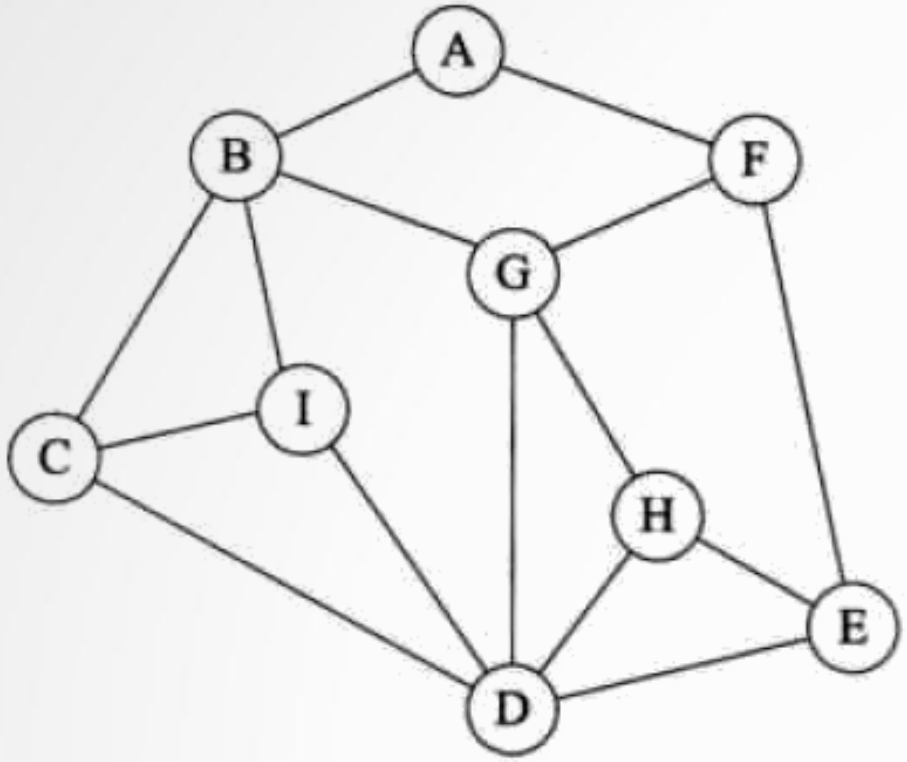
```
void DFSTraverse(ALGraph G)
{
  int i;
  for(i=0;i<G.vernum;i++)
      visited[i]=0;          //初始所有顶点状态都是未访问
  for(i=0;i<G.vernum;i++)
     if(!visited[i])         //对未访问过的顶点调用DFS
         DFS(G,i);
}
```



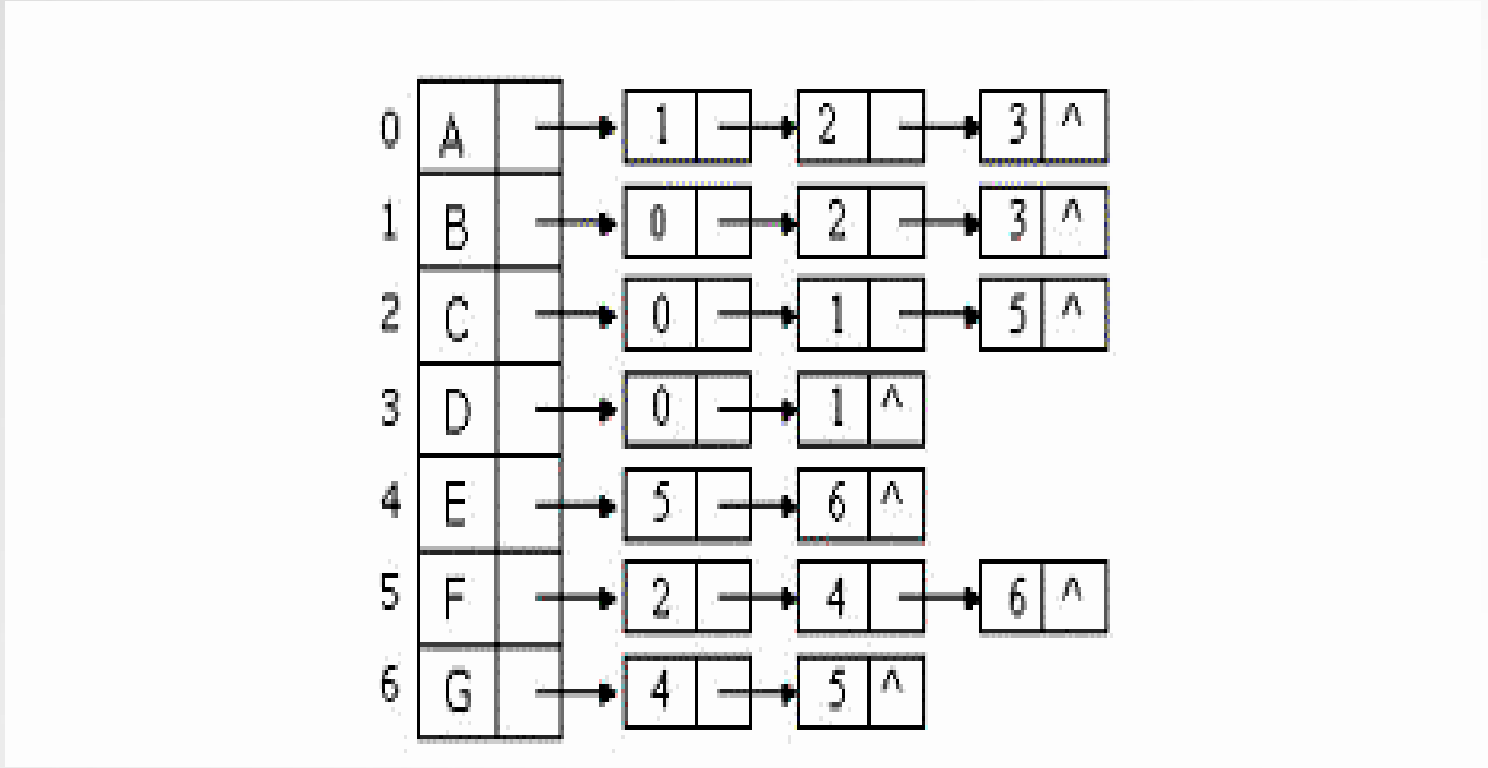


广度优先搜索（BFS，P220）　>>观看视频

【Practice】写出下图从顶点A出发，按BFS进行遍历得到的一种顶点序列。



【Practice】某无向图的邻接表如下所示



从顶点A开始对该图进行BFS，写出顶点访问序列。

【课堂作业】 P259 6.5（2）

【Question】为什么需要两种遍历？ >>观看视频