2.6 非线性数据结构

2.6.1　图及其基本概念

- 图是一种较之线性表和树形结构更为复杂的非线性数据结构。
- 如果数据元素集合D中的各数据元素之间存在任意的前后件关系，则此数据结构称为图。
- 图中各数据元素之间的关系可以是任意的，描述的是“多对多”的关系。
- 图是对结点的前件和后件个数不加限制的数据结构。

一、图（Graph）的定义

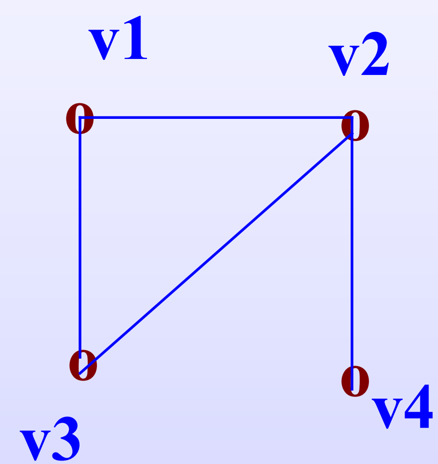
- 图G ＝（V，E ）

  其中：V={v1，v2，……，vn}是非空有穷的结点集合；E 是顶点偶对的集合。

- 例，图G1 ＝（V，E）

  V={v1，v2，v3，v4}

  E={（v1，v2），（v1，v3），
  （v2，v1），（v2，v3），
  （v2，v4），（v3，v1），
  （v3，v2），（v4，v2）}



G1

二、有向图、无向图

- 有向图（Digraph）

　　图G中顶点的偶对若是有向的，形成的图称为有向图，如图G2所示。

为示区别，其偶对用$\langle v_x, v_y \rangle$表示。

- 无向图（Undigraph）

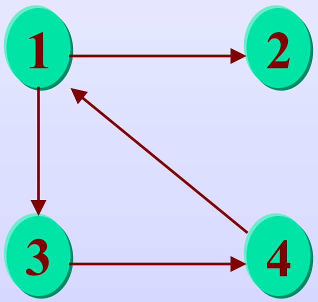
　　图G中顶点的偶对若是无向的，形成的图称为无向图，其偶对用（v_x，v_y）表示(区别在括号)，如图G1所示。

- G2=（V，E）

V={ 1，2，3，4}

E={〈1，2〉，〈1，3〉，

〈3，4〉，〈4，1〉}



G2

三、 边、弧

- *边（Edge）*

  顶点间的关系可描述为顶点的偶对，也称为顶点的边。记为： （Vx，Vy）。边是无序的，可以看成是（Vx，Vy），也可以看成是（Vy，Vx）。

- *弧（Arc）*

  若顶点间的边是有方向性（有序）的，则称该偶对为弧。记为：〈Vx，Vy〉。弧是有序的，〈Vx，Vy〉表示从Vx到Vy。

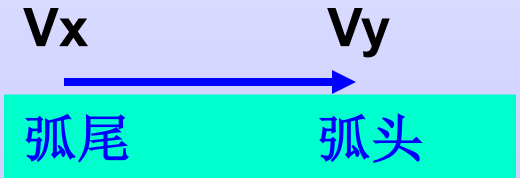
- *弧头（Head）*

  弧的终点（TerminaL Node）称为弧头（方向前方）。

- *弧尾（Tail）*

  弧的起始点（Initial Node）称为弧尾（方向后方）。

  弧 〈Vx，Vy〉表示为，



四、顶点、邻接点

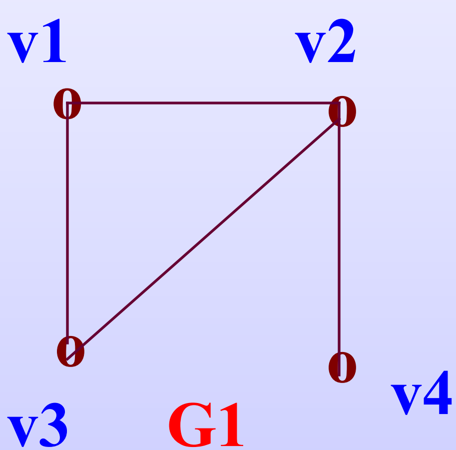
- *顶点（Vertex）：* 图中的数据元素（结点）称为顶点。

  如图G1、G2中的Ｖ1、Ｖ2，1，2。

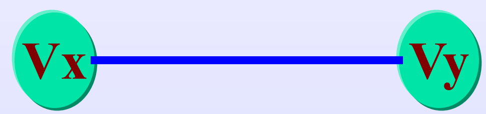
- *邻接点（Adjacent）*
  - 无向图中，若边（Ｖx，Ｖy）∊ E，

    则Ｖx、Ｖy互为邻接点。
  - 有向图中，若弧〈Ｖx，Ｖy〉 ∊ E，

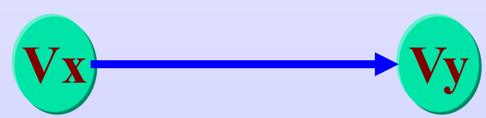
    则Ｖy是Ｖx的邻接点，反之，不是。（弧头是弧尾的邻接点）



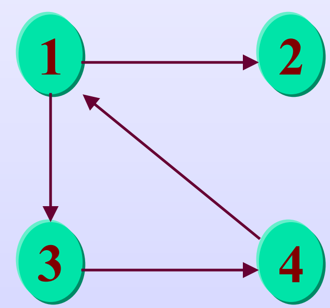
G1



Ｖx、Ｖy互为邻接点



Ｖy是Ｖx的邻接点

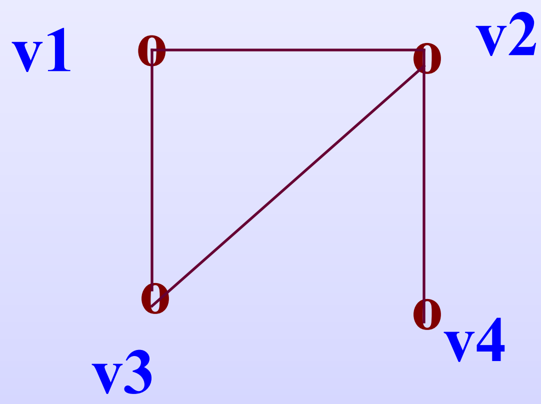


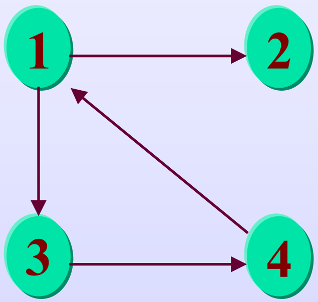
G2

五. 顶点的度（Degree）

- 在图中，一个结点的后件个数称为该结点的**出度**，其前件个数称为该结点的**入度**。一个结点的**入度与出度之和**称为该结点的**度**。对于无向图来说，其中每一个结点的入度等于该结点的出度。**图中结点的最大度称为图的度。**

G1





G2

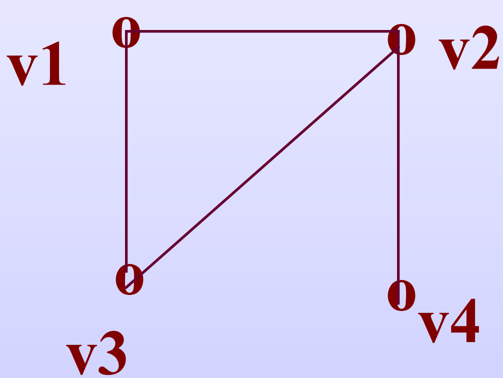
六. 路径、长度

- 路径（Path）

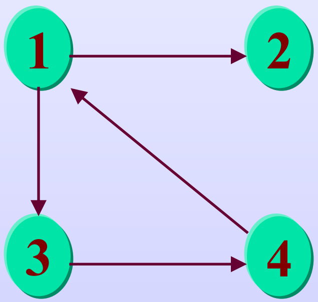
在图中，从顶点Vx到顶点Vy的顶点序列（Vx，V1，V2，...，Vn，Vy）称为从Vx到Vy的路径。路径可能是不唯一的。例如，G1中，V1到V3的路径为：（V1V2V3）或（V1V3）；而G2中，1到4的路径为<134>。

- 长度（Length）

路径的长度是该路径上边或弧的数目。例如，G1中V1到V3的长度为1或2；而G2中1到4的长度为2。



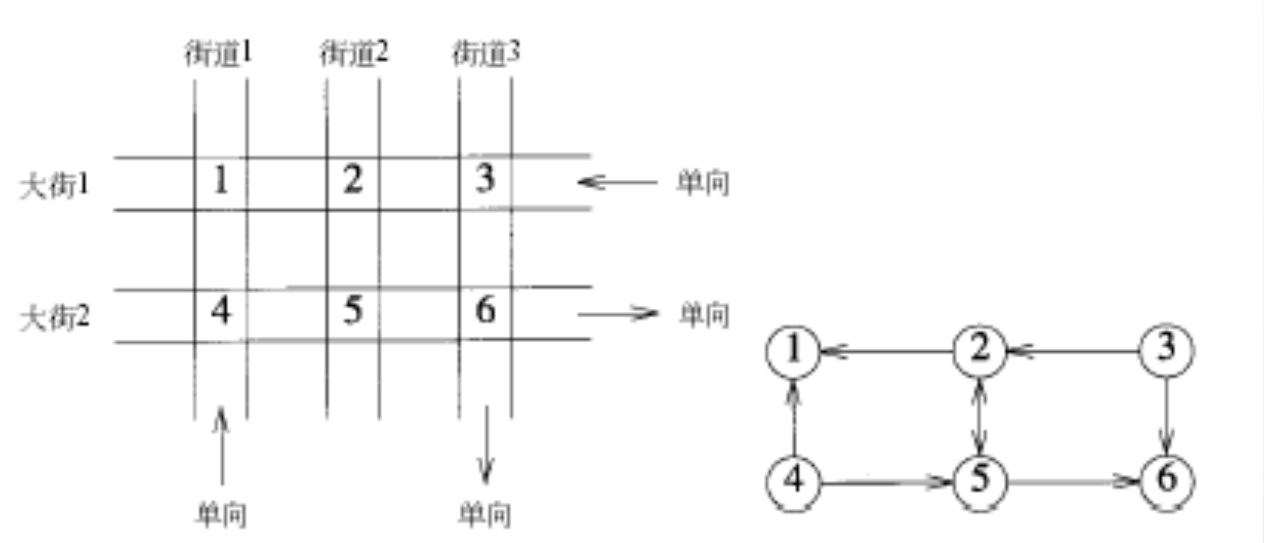
G1



G2

图的应用实例

- [路径问题] 城市中有许多街道，每一个十字路口都可以看作图中的一个顶点，邻接两个十字路口之间的每一段街道既可以看作一条，也可以看作两条有向边。如果街道是双向的，就用两条有向边。如果街道是单向的，就用一条有向边。（路线咨询）



a)   b)

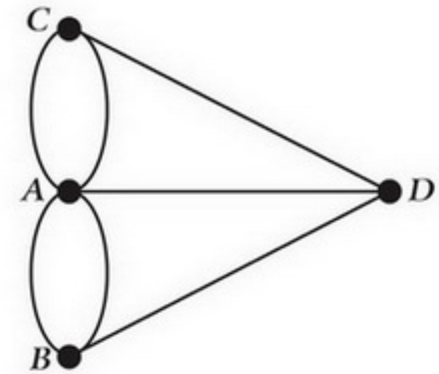
图的应用实例

- [货郎担问题]/[旅行商问题]

有n个城市，用1，2，…，n表示，城i,j之间的距离为dij，有一个货郎从城1出发到其他城市一次且仅一次，最后回到城市1，怎样选择行走路线使总路程最短？

假设有一个旅行商人要拜访n个城市，他必须选择所要走的路径，路经的限制是每个城市只能拜访一次，而且最后要回到原来出发的城市。路径的选择目标是要求得的路径路程为所有路径之中的最小值。

图的应用实例



- [哥尼斯堡七桥问题]/[一笔画问题]

18世纪东普鲁士哥尼斯堡被普列戈尔河分为四块，它们通过七座桥相互连接，如下图. 当时该城的市民热衷于这样一个游戏:“一个散步者怎样才能从某块陆地出发，经每座桥一次且仅一次回到出发点？”

图的应用实例

- [着色问题]

数学定义：

给定一个无向图G=（V, E），其中V为顶点集合，E为边集合，图着色问题即为将V分为K个颜色组，每个组形成一个独立集，即其中没有相邻的顶点。其优化版本是希望获得最小的K值。

2.6.2、图的存储结构

1、关联矩阵表示法

根据图的定义可知，图的逻辑结构分为两部分：V（顶点）和E（边或弧）的集合。因此，

- 用一个一维数组存放图中所有顶点数据；
- 用一个二维数组存放顶点间关系（边或弧）的数据，称这个二维数组为*关联矩阵*。

关联矩阵也称为邻接矩阵。

关联矩阵

- 定义

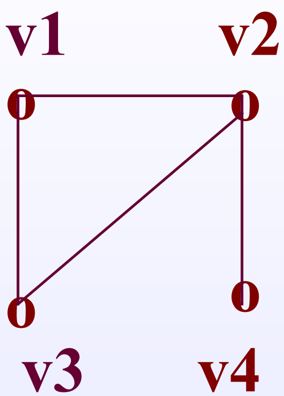
在关联矩阵R中，每一个元素R(i, j)的定义为

$$
R(i,j)=\begin{cases}1 & \text{di是dj的前件} \\ 0 & \text{di不是dj的前件}\end{cases}
$$

例如

G1的关联矩阵为：

$$R = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}_{4x4}$$

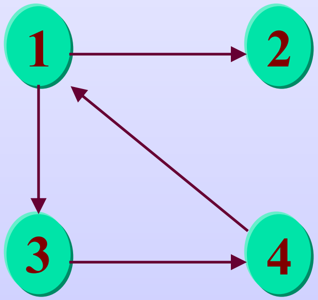


G1

★ 注意：一定对称

G2的关联矩阵为：

G2



$$R = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}_{4x4}$$

★ 注意：不一定对称

2、求值矩阵

- 关联矩阵只表示了图的结构，即图中各结点的前后件关系。但在许多实际问题中，还需要对两个关联结点之间的值进行运算。这就是说，除了要存储图中各结点值以及各结点之间的关系外，还必须存储图中每两个结点之间的**求值函数**。

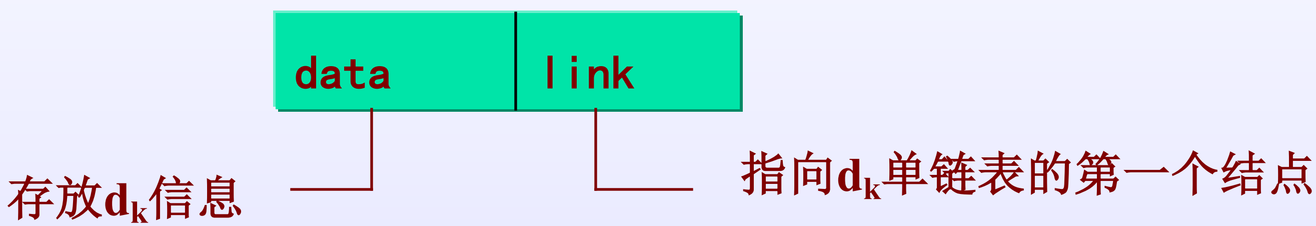
- **为了表示有值图中每两个结点之间的求值函数，可以另外用一个求值矩阵V来存储。**

例

$$R=\begin{bmatrix} 0&0&1&1&0&1&0&1\\ 0&0&1&1&0&0&1&0\\ 1&1&0&0&1&1&0&1\\ 1&1&0&0&1&0&1&0\\ 0&0&1&1&0&1&1&0\\ 1&0&1&0&1&0&0&1\\ 0&1&0&1&1&0&0&1\\ 1&0&1&0&0&1&1&0 \end{bmatrix}\quad V=\begin{bmatrix} 0&-1&30&55&-1&35&-1&20\\ -1&0&10&45&-1&-1&35&-1\\ 30&10&0&-1&30&35&-1&25\\ 55&45&-1&0&10&-1&55&-1\\ -1&-1&30&10&0&15&50&-1\\ 35&-1&35&-1&15&0&-1&20\\ -1&35&-1&55&50&-1&0&15\\ 20&-1&25&-1&-1&20&15&0 \end{bmatrix}$$

3、邻接表表示法

- 邻接表这种存储结构也称为“**顺序-索引-链接**”存储结构。
- 用一个顺序存储空间来存储图中各结点的信息。**数据域**data与**指针域**link。

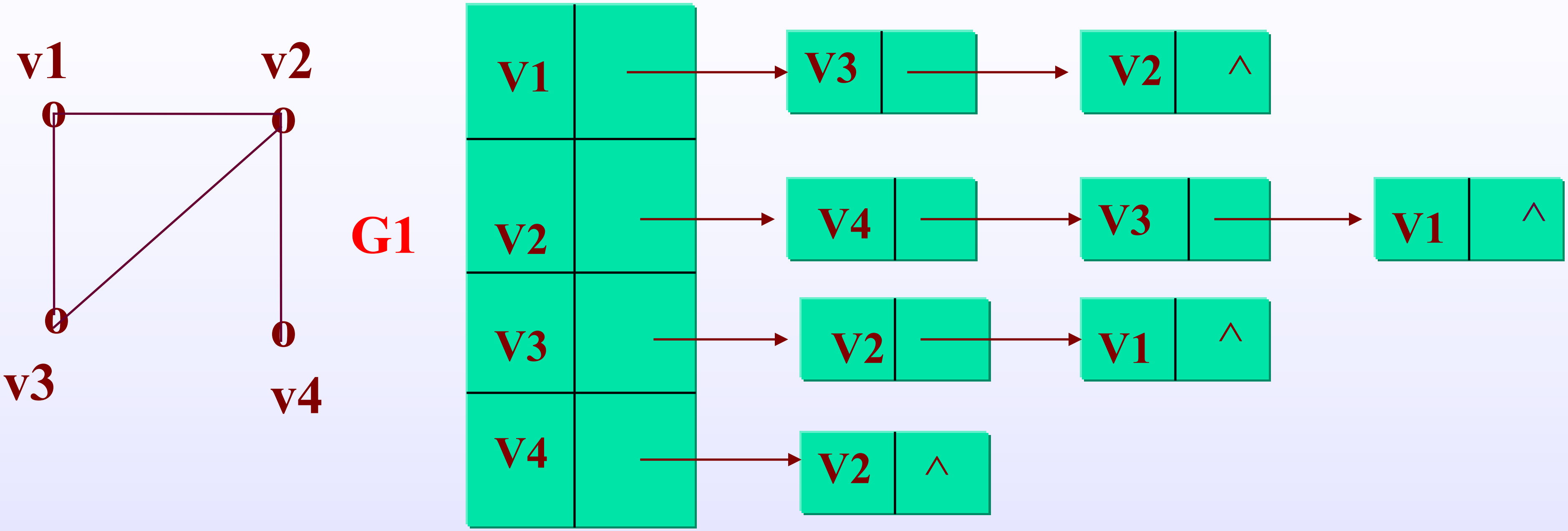


- 数据域存放图中编号为k的结点值；指针域link用于链接相应结点的后件，

- 对于图中每一个结点，构造一个单链表。该单链表的头指针即为顺序空间中的对应存储结点的指针域。
- 单链表中各存储结点的结构如图所示，其中num域用于存放图中某个**结点的编号**；val域用于存放编号为num结点的前件到num结点之间的**求值函数值**f，
- 如果不是有值图，则val域可以不要；next域用于指向与num结点是同一个前件的另一个后件信息的结点。

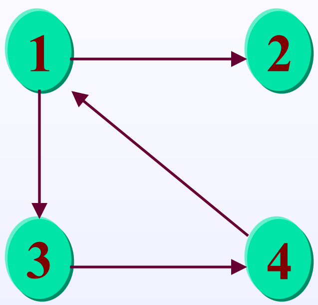
num	val	next

无向图G1的邻接表

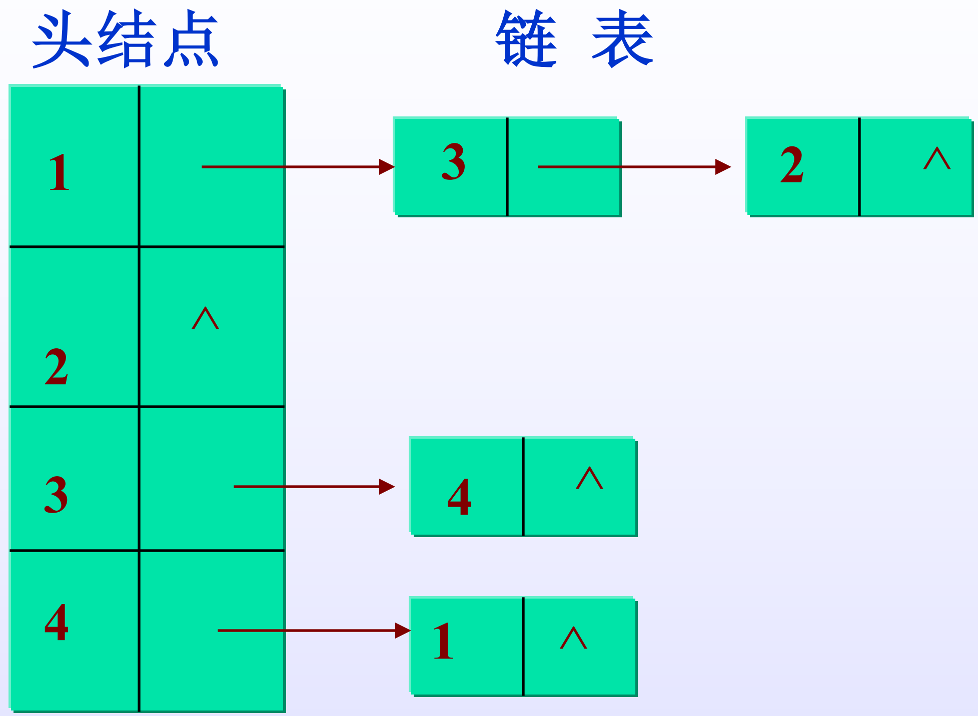


注：边无权，节点只需两个域

有向图G2的邻接表



G2



- **描述：** 在有向图中，第i个单链表中结点的个数是顶点Vi的出度；
- **特点：** 求出度方便，求入度，必须遍历整个邻接表。

建立邻接表的算法

操作步骤：

- step1 初始化邻接表的n个头结点，
- Step2 读入一条弧或边的偶对〈i，j〉或（i，j）。
- step3 申请一个结点S的空间，将S插入到第i个单链表中；
- step4 读下一条弧或边的偶对，若存在此弧或边，则继续执行step2；否则，结束。

4、邻接多重表

- 在图中，每一条边连接了两个结点。在有些应用中，需要同时找到表示一条边的两个结点，此时，邻接表的存储方式就显得不太方便了。
- 在邻接多重表中，每条边用一个存储结点表示，每个存储结点由五个域组成，其中，Mark为标志域，

mark	di	dj	di-link	dj-link

2.6.3、图的遍历

- *图的遍历（Traversing Graph）*

  从图中指定顶点出发访问图中每一个顶点，且使每个顶点只被访问一次，该过程称为图的遍历。

- 图的遍历要比树结构复杂的多。出发点不同，任一顶点的邻接点就不相同，路径也就不同。

- 因为图中元素是“多对多”的关系，为避免发生重复，设立一个辅助数组MARK[]，每访问一个顶点，便将其状态MARK[i]置为“真”。

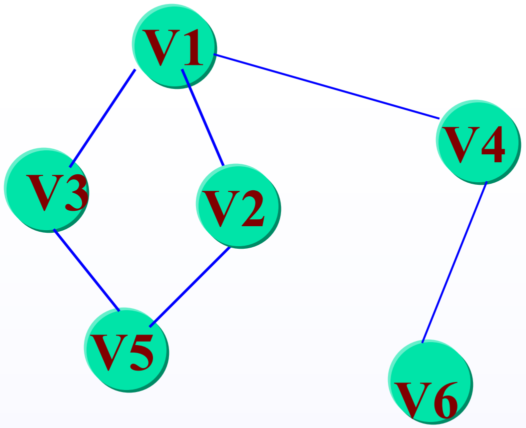
- 常用的图的遍历方法有两种：
  - 深度优先遍历法(纵向优先搜索法)
  - 广度优先遍历法(横向优先搜索法)

一、 深度优先遍历法

算法思想：

- step1 从图中某个顶点V_0出发，并访问此顶点；
- step2 从V_0出发，访问与V_0邻接的顶点V_1后，再从V_1出发，访问与V_1邻接且未被访问过的顶点V_2。重复上述过程，直到不存在未访问过的邻接点为止。
- step3 如果是连通图，从任一顶点V_0出发，就可以遍历所有相邻接的顶点；如果是非连通图，则再选择一个未被访问过的顶点作为出发点，重复step1、step2，直到全部被访问过的邻接点都被访问为止。

深度优先遍历



G6

- 深度优先遍历G6所走过的序列：

  V1→ V4 → V6→ V3 → V5 → V2

假设图有 n 个结点，采用数组a[][]存放图的邻接矩阵各元素值，图的深度优先搜索遍历算法如下：

```
dfs(int a[][],int i,int n)
{    /*以i为出发点，按深度优先搜索遍历图*/
  int j;
    printf("V=%4d",i);
    visited[i]=1;   /*标识顶点vi被访问*/
  for( j=0;j<n;j++ )
   if(a[i][j]!=0 && visited[j]==0)
    dfs( a,j,n );/*递归调用函数dfs()*/
}
```

```
/*函数dfs()被下面的dtraver()函数调用*/
dtraver( int a[][],int n )
{
int i;
  for( i=0;i<n;i++ )
visited[i]=0; /*初始化,标识顶点vi没被访问*/
for( i=0;i<n;i++ )
if( visited[i]==0 ) /*若顶点vi没有被访问，
                         则从该顶点出发遍历图*/
dfs( a,i,n );
}
```

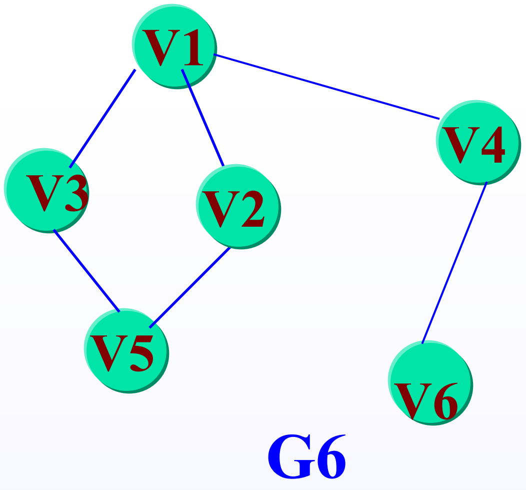
这里，辅助数组visited[]应定义为全局变量.

课堂练习

2. 广度优先遍历算法

- 先访问第1个顶点所有邻接点后，再访问下一个顶点所有未被访问的邻接点。
- 算法思想：
  - step1 从图中某个顶点V_0出发，并访问此顶点；
  - step2 从V_0出发，访问V_0的各个未曾访问的邻接点W_1，W_2，…，W_k；然后，依此从W_1，W_2，…，W_k出发访问各自未被访问的邻接点。
  - step3 重复step2，直到全部顶点都被访问为止。

广度优先遍历



- 深度优先遍历G6所走过的序列：

  V1→ V4 → V3→ V2 → V6 → V5

```
bfs(int a[][],int i,int n)
{   int j,k,b1=-1, b2=0,b[n];
    b[b2]=i;
    while(b1<b2) /*队列不空则循环*/
      { b1=b1+1;
        k=b[b1]; /*队首顶点出队*/
        visited[k]=1; /*置已被访问标识*/
        printf("V=%4d",k++);
        for(j=0;j<n;j++)
        if (a[k][j]!=0 && visited[j]==0)
          { b2=b2+1;b[b2]=j; }
                    /*没有被访问的顶点进队*/
      }
}
```

```
/*函数bfs()将被下面的函数btraver()调用*/
btraver( int a[][],int n)
{
   int  i;
   for(i=0;i<n;i++)
visited[i]=0;
/*辅助数组初始化，标识顶点vi没有被访问*/
   for(i=0;i<n;i++)
if(visited[i]==0) /*若顶点vi没有被访问，则从该顶点出发遍历图*/
bfs(a,i,n);
}
```

同样，辅助数组visited[]应定义为全局变量。

课堂练习

总结

1. 图的定义
2. 图的基本概念
3. 图的遍历