第7章 图1

学习要点:

- 理解图的定义和与图相关的有向图、无向图、赋权图、连通图等术语。
- 理解图是一个表示复杂非线性关系的数据结构。
- 掌握图的邻接矩阵表示及其实现方法。
- 掌握图的邻接表表示及其实现方法。
- 了解图的紧缩邻接表表示方法。
- 掌握图的广度优先搜索方法。
- 掌握图的深度优先搜索方法。
- 掌握单源最短路径问题的**Dijkstra**算法。
- 掌握所有顶点对之间最短路径问题的**Floyd**算法。
- 掌握构造最小支撑树的**Prim**算法。
- 掌握构造最小支撑树的**Kruskal**算法。
- 理解图的最大匹配问题的增广路径算法。

7.1 图的基本概念

- 图(Graph)——图G是由两个集合V(G)和E(G)组成　记为G=(V,E)

  其中：V(G)是顶点的非空有限集

  E(G)是边的有限集合，边是顶点的无序对或有序对

- 有向图——有向图G是由两个集合V(G)和E(G)组成

  其中：V(G)是顶点的非空有限集

  E(G)是有向边的有限集合，弧是顶点的有序对，记为<v,w>，v,w是顶点，v为有向边的起点，w为有向边的终点

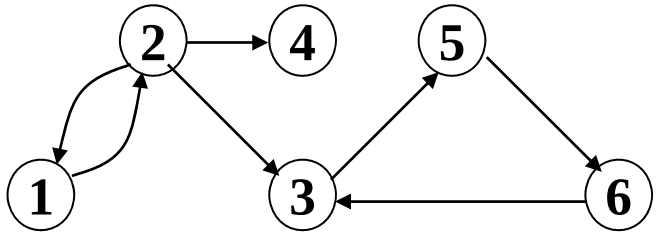
- 无向图——无向图G是由两个集合V(G)和E(G)组成

  其中：V(G)是顶点的非空有限集

  E(G)是边的有限集合，边是顶点的无序对，记为（v,w）或（w,v），并且（v,w)=(w,v)

  本书约定：不考虑顶点到其自身的边；不允许一条边在图中重复出现。即只讨论简单图。

例

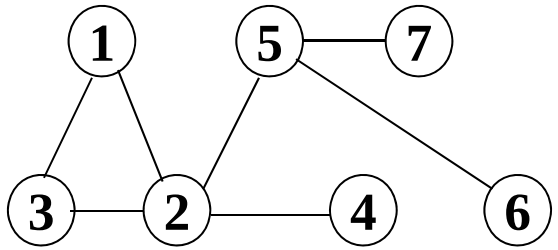


G1

图G1中：V(G1)={1,2,3,4,5,6}

E(G1)={<1,2>, <2,1>, <2,3>, <2,4>, <3,5>, <5,6>, <6,3>}

例



G2

图G2中：V(G2)={1,2,3,4,5,6,7}

E(G1)={(1,2), (1,3), (2,3), (2,4),(2,5), (5,6), (5,7)}

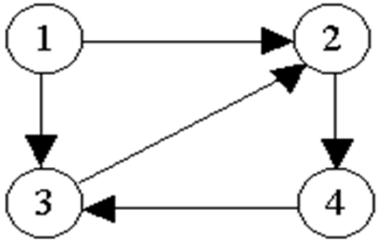
- 完全图——设|V|=n,|E|=e。对有向图G，若e=n(n-1)，则称G为完全的有向图；对无向图G，若e=n(n-1)/2，则称G为完全的无向图。
- 邻接、关联——若(u,v)是一条无向边，则称顶点u和v互为邻接点，或称u和v相邻接；并称边(u,v)关联于顶点u和v，或称边(u,v)与顶点u和v相关联。若(u,v)是一条有向边，则称v是u的邻接顶点；并称边(u,v)关联于顶点u和v，或称边(u,v)与顶点u和v相关联。
- 顶点的度
  - 无向图中，顶点v的度为关联于该顶点相连的边数，记为D(v)
  - 有向图中，顶点v的度分成入度与出度
    - 入度：以顶点v为终点的边的数目，记为ID(v)
    - 出度：以顶点v为起点的边的数目，记为OD(v)
    - D(v)=ID(v)+OD(v)

无论是有向图还是无向图，顶点数n，边数e和度数之间有如下关系：

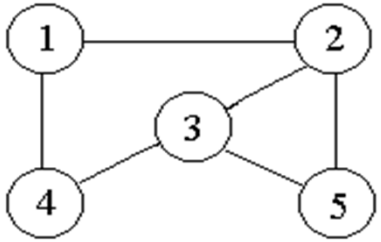
$$e = \frac{1}{2}\sum_{i=1}^{n} D(v_i)$$

- 子图——如果图G(V,E)和图G'(V',E'),满足：

  V'⊆V　E'⊆E　则称G'为G的子图

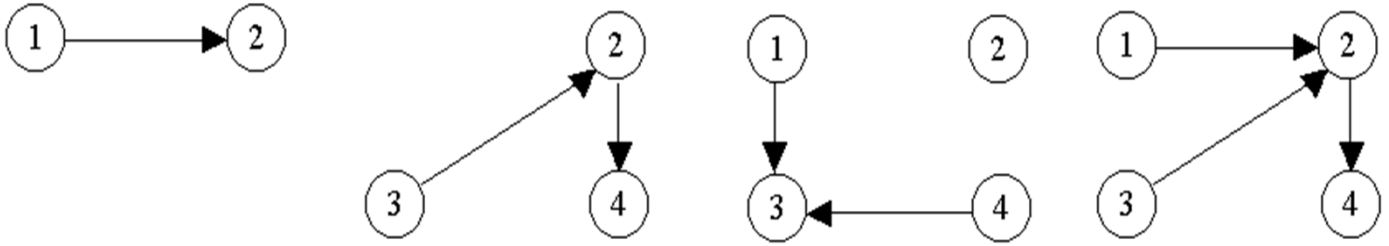


G1

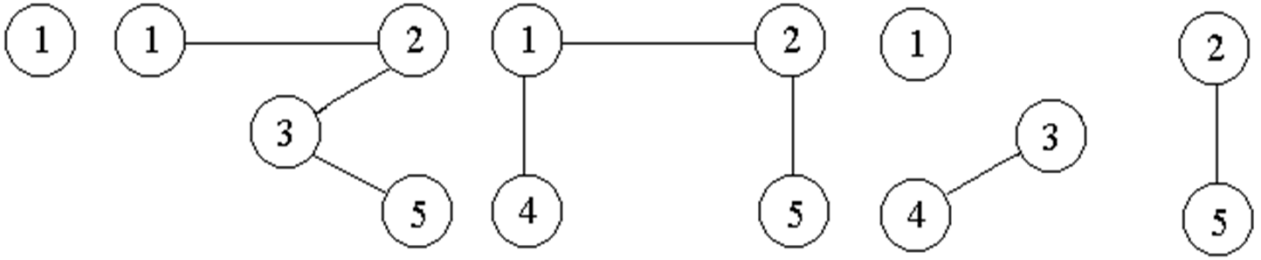


G2

有向图G1的若干子图



无向图G2的若干子图



- 路径：

  在无向图**G**中，若存在一个顶点序列**u(1),u(2),…，u(m),**使得**(u(i),u(i+1))∈E(G)，i=1,2,…，m-1**, 则称该顶点序列为顶点**u(1)**和**u(m)**之间的一条路径。其中**u(1)**称为该路径的起点，**u(m)**称为该路径的终点。

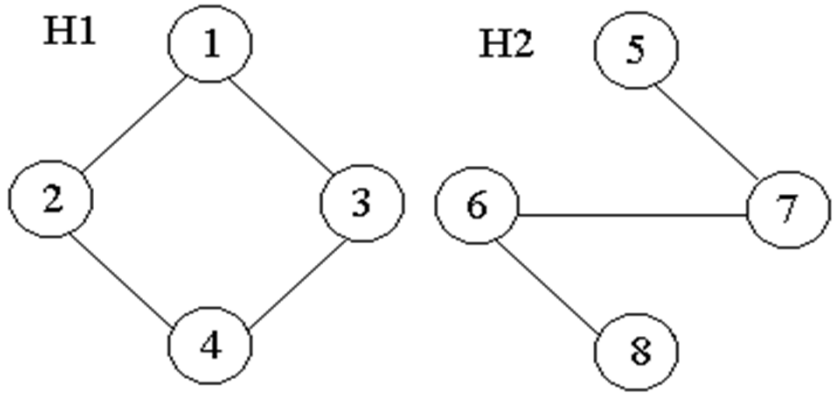
  若图**G**是有向图，则路径也是有向的，其中每条边**(u(i),u(i+1))，i=1,2,…，m-1**均为有向边。

  路径的长度：路径所包含的边数**m-1**称之。

- 简单路**—**若一条路径上除了起点和终点可能相同外，其余顶点均不相同，则称此路径为一条简单路径。
- 回路**—**起点和终点相同的简单路径称为简单回路或简单环或圈。
- 有根图**—**在一个有向图中，若有一个顶点**v**，从该顶点有路径可以到达图中其它所有顶点,则称此有向图为有根图。**v**称为该有根图的根。

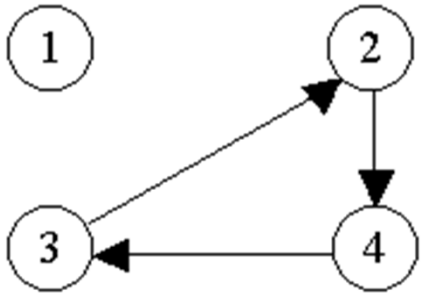
- 连通——无向图G中，若从顶点V到顶点W有一条路径，则说V和W是连通的
- 连通图——无向图中任意两个顶点都是连通的叫连通图
- 连通分支——无向图的极大连通子图叫连通分支

下图有两个连通分支：



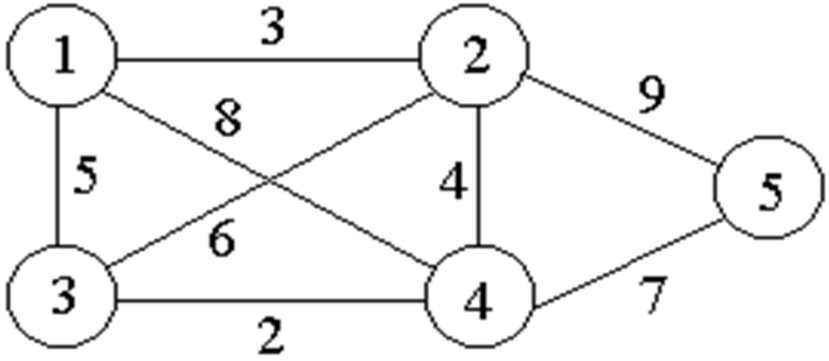
- 强连通图——有向图中，如果对每一对**Vi,Vj∈V, Vi≠Vj,** 从**Vi**到**Vj** 和从**Vj**到 **Vi**都存在路径，则称**G**是强连通图
- 强连通分支——有向图的极大强连通子图叫强连通分支

显然，强连通图只有一个强连通分支，即其自身。非强连通的有向图有多个强连通分支。如下图中的图不是强连通图，但它有2个强连通分支。



赋权图和网络

若无向图的每条边都带一个权，则称相应的图为赋权无向图。同理，若有向图的每条边都带一个权，则称相应的图为赋权有向图。通常，权是具有某种实际意义的数，比如，2个顶点之间的距离，耗费等。赋权无向图和赋权有向图统称为网络。下图就是一个网络的例子。



7.2 抽象数据类型图

ADT图支持的基本运算以有向图为基本模型。
ADT图支持的基本运算如下：

- (1) Graphinit(n,G): 创建n个孤立顶点的图。
- (2) GraphExist(i,j,G): 判断边(i,j)是否存在。
- (3) GraphEdges(G): 返回图的边数。
- (4) GraphVertices(G): 返回图的顶点数。
- (5) GraphAdd(i,j,G): 在图中加入边(i,j)。
- (6) GraphDelete(i,j,G): 删除边(i,j)。
- (7) OutDegree(i,G): 返回顶点i的出度。
- (8) InDegree(i,G): 返回顶点i的入度。

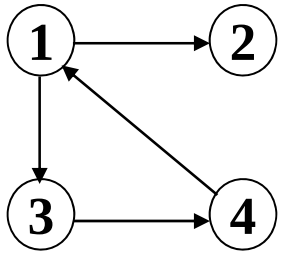
7.3 图的表示法

7.3.1 邻接矩阵——表示顶点间邻接关系的矩阵

- 定义：设**G=(V,E)**是有**n≥1**个顶点的图，**G**的邻接矩阵**A**是具有以下性质的**n**阶方阵

$$A[i,j]=\begin{cases}1, 若(v_i,v_j)或<v_i,v_j>\in E(G)\\ 0, \quad 其它\end{cases}$$

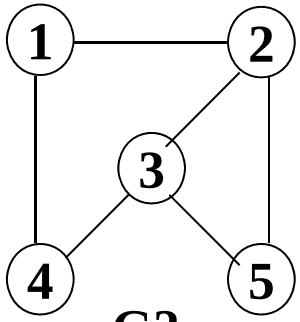
例



G1

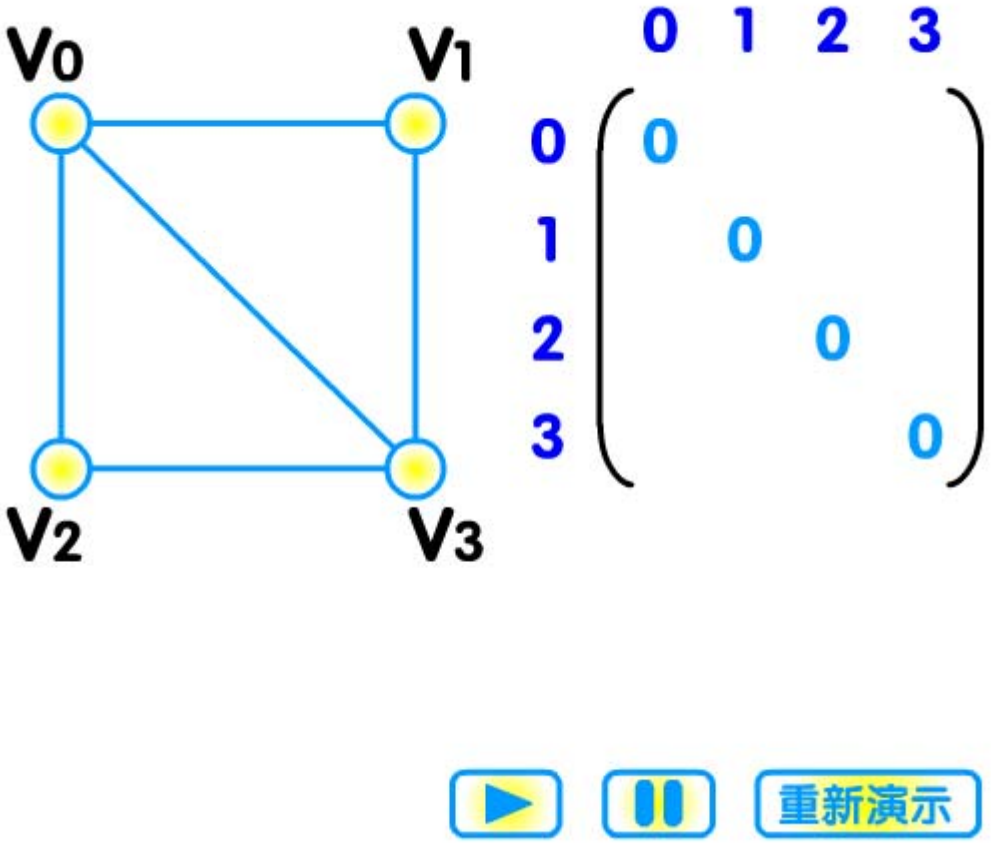
	①	②	③	④
①	0	1	1	0
②	0	0	0	0
③	0	0	0	1
④	1	0	0	0

例



G2

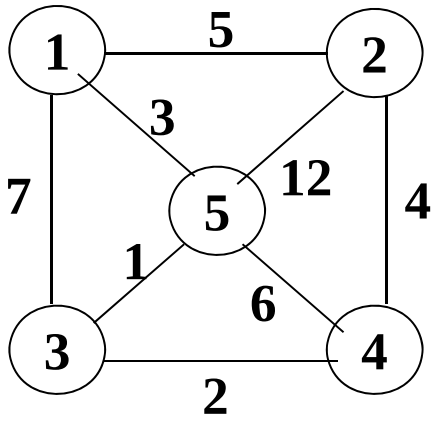
	①	②	③	④	⑤
①	0	1	0	1	0
②	1	0	1	0	1
③	0	1	0	1	1
④	1	0	1	0	0
⑤	0	1	1	0	0


V0
V1
V2
V3
0 1 2 3
0 0
1 0
2 0
3 0
重新演示

- 特点：
  - 无向图的邻接矩阵对称，可压缩存储；有**n**个顶点的无向图需存储空间为**n(n+1)/2**
  - 有向图邻接矩阵不一定对称；有**n**个顶点的有向图需存储空间为**n^2**
  - 无向图中顶点**V_i**的度**$TD(V_i)$**是邻接矩阵**A**中第**i**行元素之和
  - 有向图中：
    - 顶点**V_i**的出度是**A**中第**i**行元素之和
    - 顶点**V_i**的入度是**A**中第**i**列元素之和
  - 网络的邻接矩阵可定义为：

$$A[i,j]=\begin{cases}\omega_{ij}, 若(v_i,v_j)或<v_i,v_j>\in E(G)\\ \infty，\ 其它\end{cases}$$

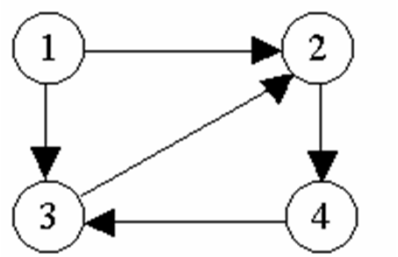
例



$$
\begin{array}{c} \\ ① \\ ② \\ ③ \\ ④ \\ ⑤ \end{array}
\begin{array}{c}
\begin{array}{ccccc} ① & ② & ③ & ④ & ⑤ \end{array} \\
\begin{bmatrix}
\infty & 5 & 7 & \infty & 3 \\
5 & \infty & \infty & 4 & 8 \\
7 & \infty & \infty & 2 & 1 \\
\infty & 4 & 2 & \infty & 6 \\
3 & 8 & 1 & 6 & \infty
\end{bmatrix}
\end{array}
$$

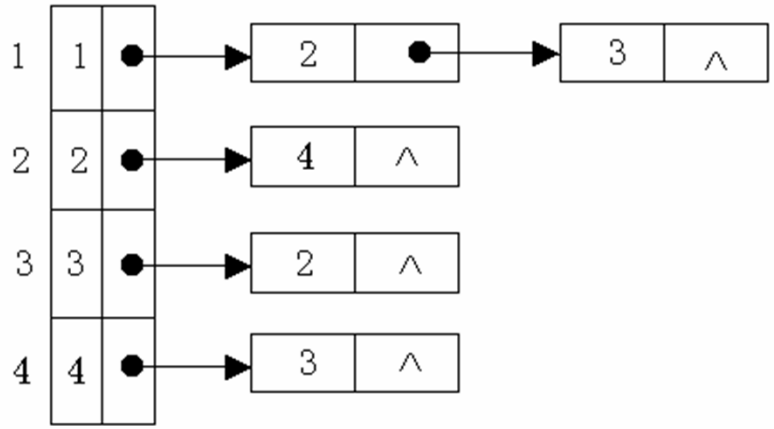
7.3.2 邻接表

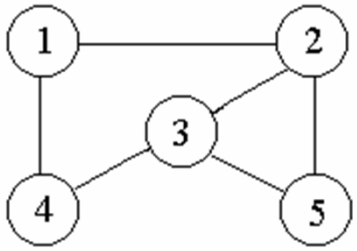
- 实现：为图中每个顶点建立一个单链表，第i个单链表存放顶点Vi的所有邻接顶点。



G1

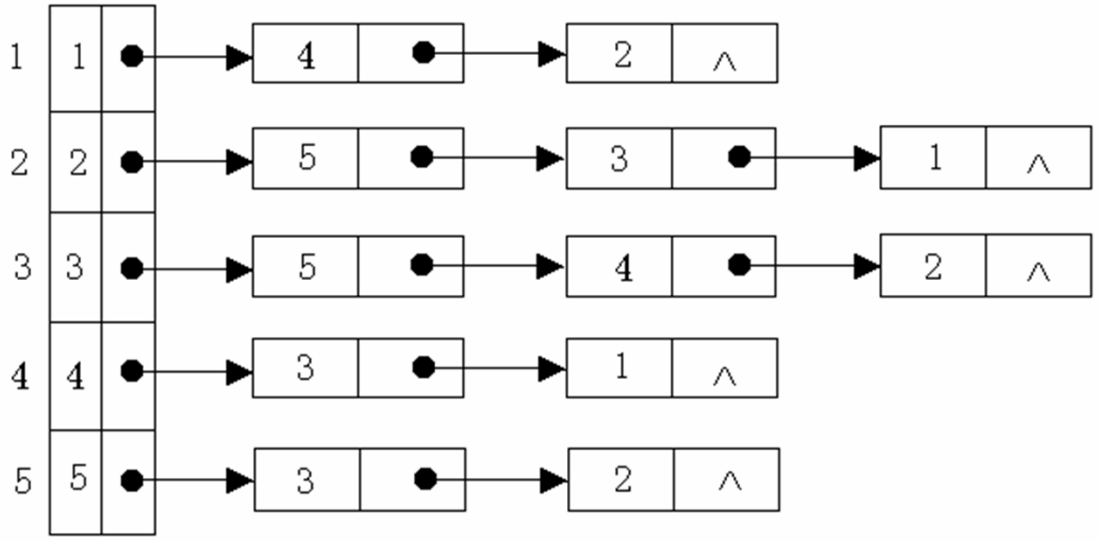
(a)

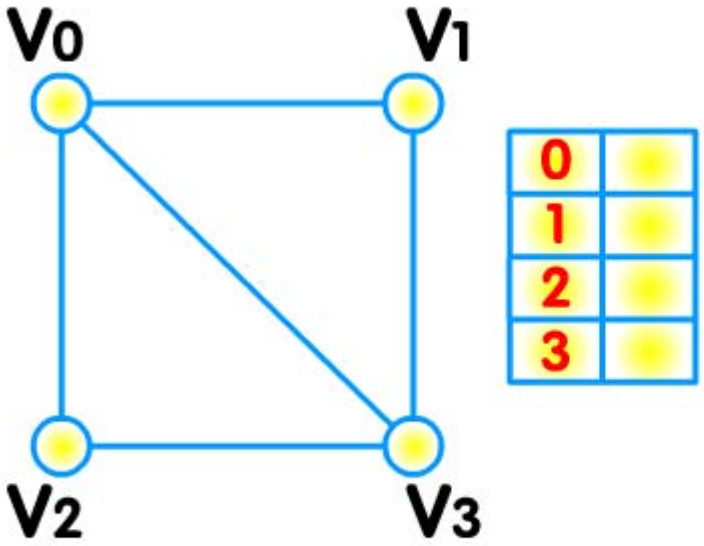




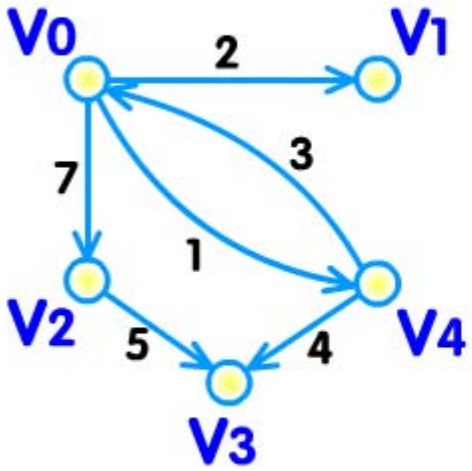
G2

(b)




V0
V1
V2
V3
0
1
2
3


重新演示

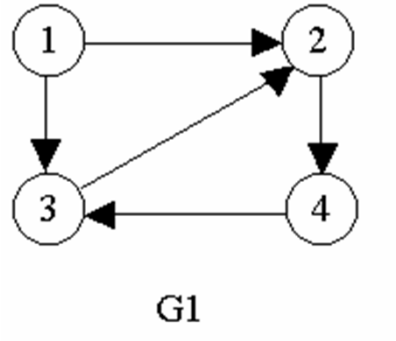


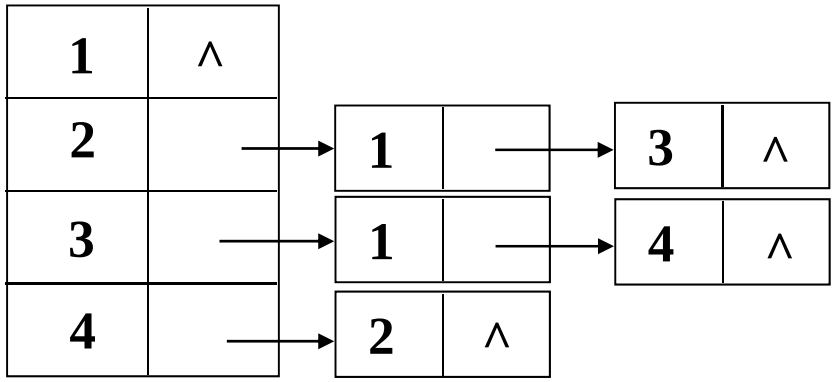
V0	
V1	
V2	
V3	
V4	

- 特点
  - 无向图中顶点V_i的度为第i个单链表中的结点数
  - 有向图中
    - 顶点V_i的出度为第i个单链表中的结点个数
    - 顶点V_i的入度为整个单链表中邻接点域值是i的结点个数
- 逆邻接表：有向图中对每个结点建立以V_i为终点的边的单链表

求解麻烦！

例

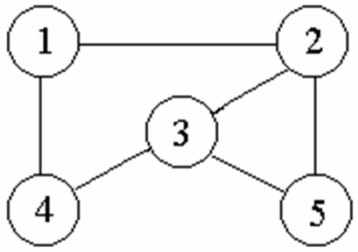




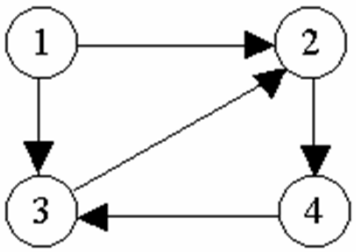
7.3.3 紧缩邻接表

紧缩邻接表将图**G**的每个顶点的邻接表紧凑地存储在**2**个一维数组**List**和**h**中。其中一维数组**List**依次存储顶点**1,2,…, n**的邻接顶点。数组单元**h[i]**存储顶点**i**的邻接表在数组**List**中的起始位置。

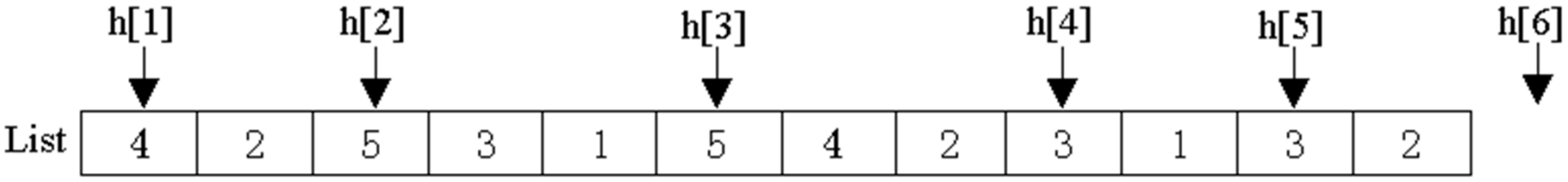
如图G2和G1的紧缩邻接表表示分别如下图(a)和(b)：



G2

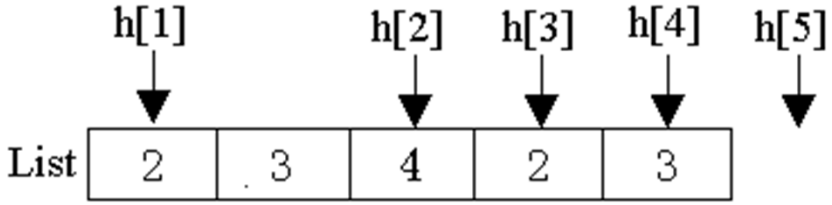


G1



h=[0,2,5,8,10,12]

(a)



h=[0,2,3,4,5]

(b)

7.4 用邻接矩阵实现图

7.4.1 用邻接矩阵实现赋权有向图

从图的结构和概念上看，可将图分为有向赋权图、无向赋权图、有向图和无向图4种不同类型。在上述4种不同类型的图中，有向赋权图具有较一般的特征。P136

```
typedef struct graph *Graph
typedef Struct graph{
    Witem NoEdge;
    int n;
    int e;
    Witem  **a;
}AWDgraph;
```

7.4.2用邻接矩阵实现赋权无向图

7.4.3用邻接矩阵实现有向图

7.4.4用邻接矩阵实现无向图

7.5 用邻接表实现图

7.5.1 用邻接表实现有向图

```
typedef  struct lnode *glink;
Struct lnode{
  int v;
  glink next;
};

typedef  struct graph *Graph;
struct graph{
  int n;
  int e;
  glink *adj;
};
```

7.5.2用邻接表实现无向图

7.5.3用邻接表实现赋权有向图

7.5.4用邻接表实现赋权无向图