算法与数据结构
8.1: 图

郝家胜

hao@uestc.edu.cn

School of Automation Engineering,
University of Electronic Sci. & Tech. of China

图的基本概念

图的基本概念

- 树中同一层结点之间没有任何横向联系
- 图中结点之间的联系是任意的，是比树更复杂的非线性数据结构
- 图的形式化定义：

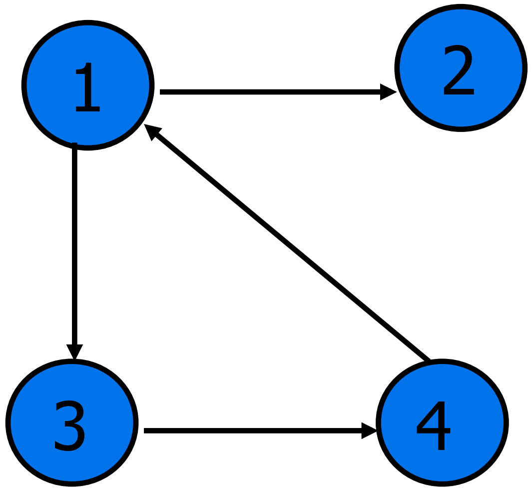
$$G = (V, E)$$

- 顶点: 图中数据元素
- V: 顶点的有穷非空集合，可记为V(G)；
- E：V中顶点偶对（称为边）的有穷集，可记为E(G).
- E(G)可为空
- 边：两顶点间的关系
- 有向图G: G中每条边是有方向的
- 无向图G: G中每条边无方向

有向图中：

- 用有序对<V_i,V_j>表示从顶点V_i到V_j的一条有向边（弧）
- V_i称为边的始点（或弧尾）
- V_j称为边的终点（或弧头）
- <V_j,V_i>是不同的有向边

例子 G_1：

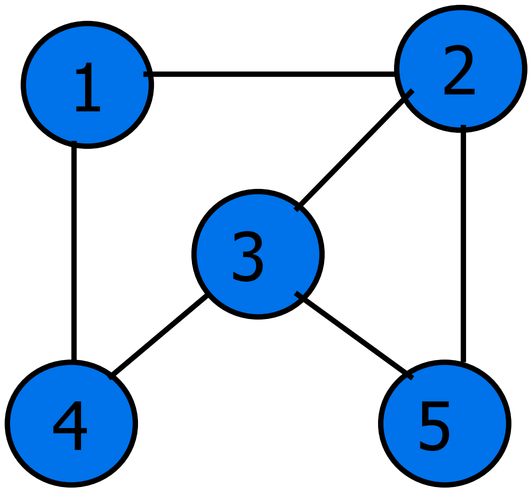


- G_1 =（V，E）
- $V(G_1)=\{1,2,3,4\}$
- $E(G_1)=\{<1,2>,<1,3>,<3,4>,<4,1>\}$

无向图中：

- 边是顶点的无序对，用(V_i,V_j)表示。

例子 G_2：



- G_2 =（V， E）
- $V(G_2)=\{1,2,3,4,5\}$
- $E(G_2)=\{(1,2),(1,4),(2,3),(3,4),(3,5),(2,5)\}$

术语：

- 1)邻接点：
- 无向图:若边(v,u)∈E，则v、u互为邻接点；
- 有向图:若弧<v,u>∈E，则u是v的邻接点

- 2)顶点的度：D(V)
- 无向图：D(V)是以该顶点为一个端点的边的条数；
- 有向图：　以某顶点为弧头的弧的数目，称为此顶点的入度（ID(V)表示）
- 以某顶点为弧尾的弧的数目，称为此顶点的出度（OD(V)表示）
- $$D(V) = ID(V) + OD(V)$$

无向图或有向图中：

n表示图中顶点数目

e表示边或弧的数目

不考虑顶点到其自身的边或弧，即若<x,y>∈VE，则x≠y。

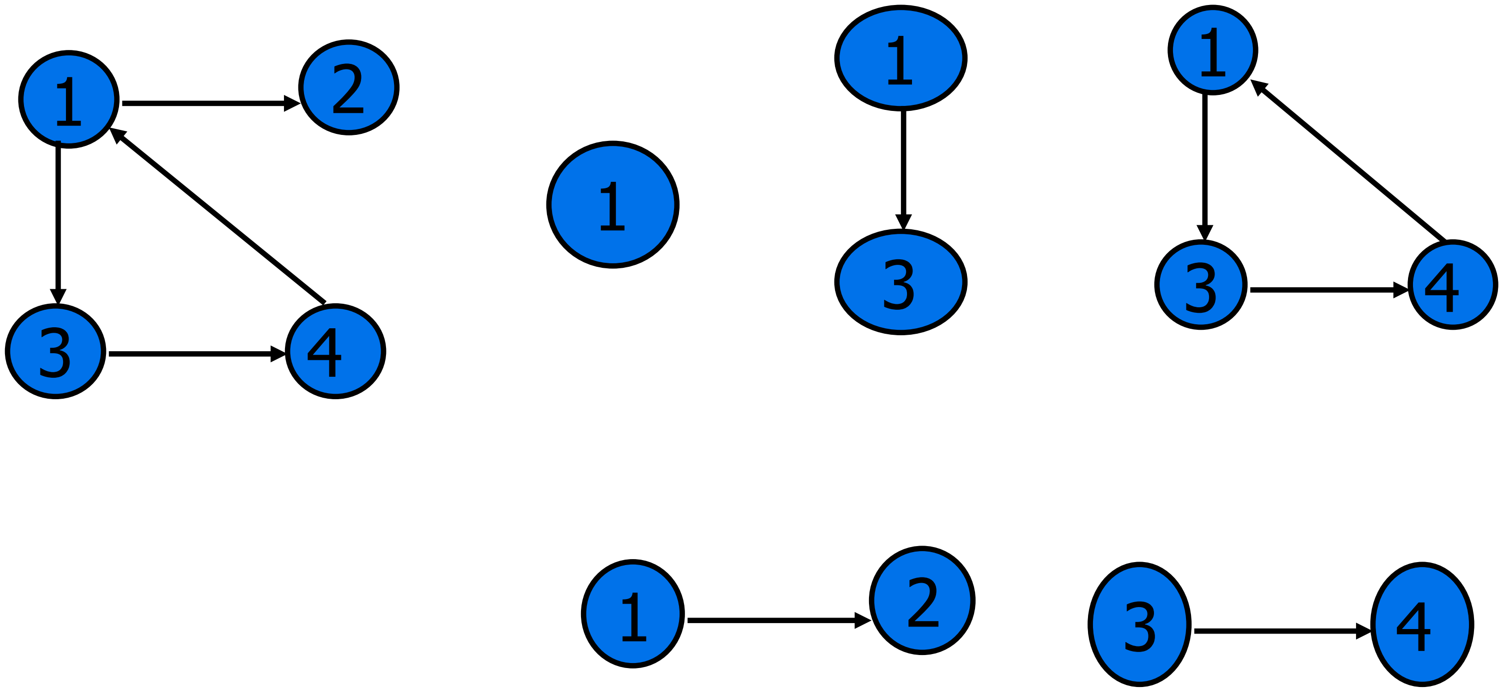
则：

- 无向图，e的取值范围是0到n(n-1)/2；
- 对于有向图，e的取值范围为0到n(n-1)。

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

- **3)子图：**
- 设有两个图$G = (V,E)$和$G' = (V',E')$
- 如果$V' \subseteq V$且$E' \subseteq E$，则称G'为G的子图。

例: G_1的子图



- **4)路径：**
- **图G中，从顶点v到顶点u的一条路径是顶点的序列（$v,v_1,v_2,\ldots,v_i,u$）**
- **且(v,v_1)，(v_1,v_2)，…，或$<v,v_1>$，$<v_1,v_2>$，…都属于集合E。**

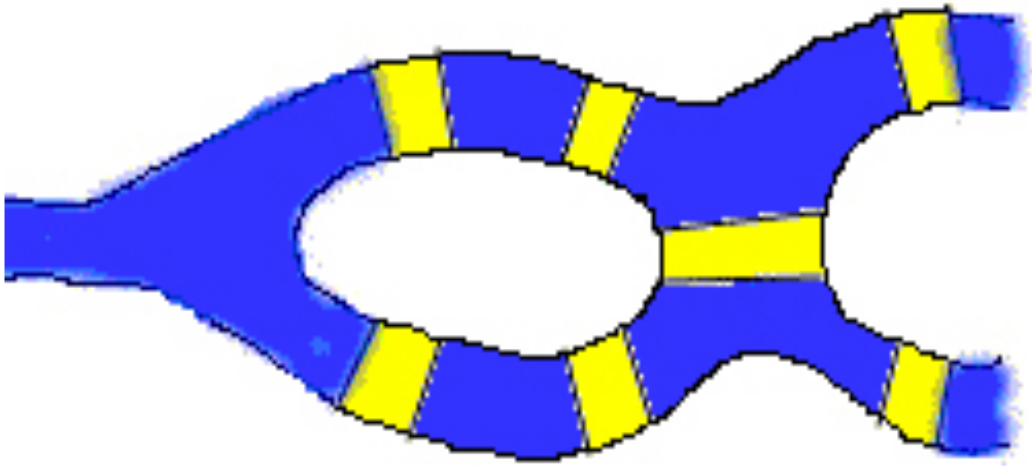
- 路径的长度：路径上弧的数目。
- 第一个顶点和最后一个顶点相同的路径称为回路或环。
- 序列中顶点不重复出现的路径称为简单路径。

- **5)连通图：**
- 无向图G中，若从顶点v到顶点u有路径，则称v和u是连通的。若对于图中每一对顶点之间都有路径，则称G是连通图。（G_2是连通图）
- 有向图中，若每一对顶点u和v之间都存在从v到u及从u到v的路径，称此图为强连通图。（G_1不是）

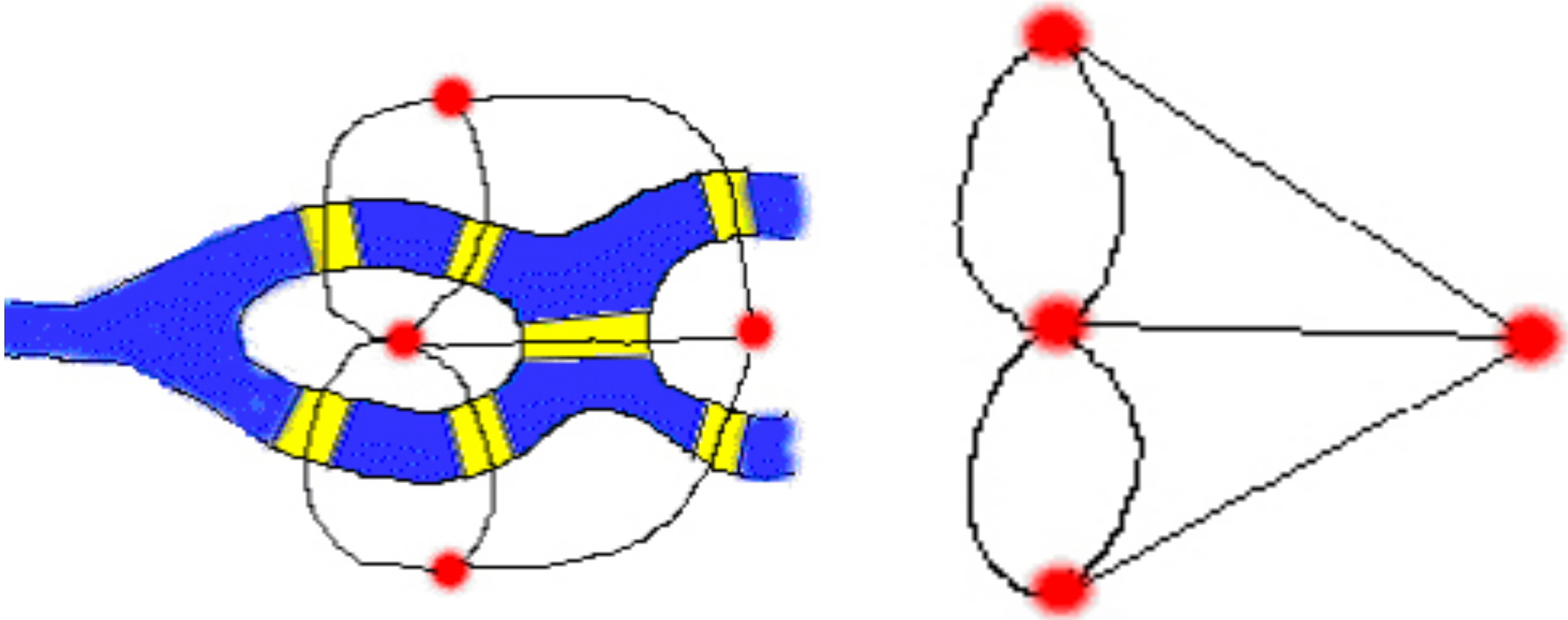
- **6)网络：**
- **若图G(V，E)中每条边都赋有反映这条边的某种特性的数据，则称此图是一个网络，其中与边相关的数据称为该边的权。**

一笔画

附：欧拉与哥尼斯堡七桥问题



欧拉与图论



一笔画

图的存储结构

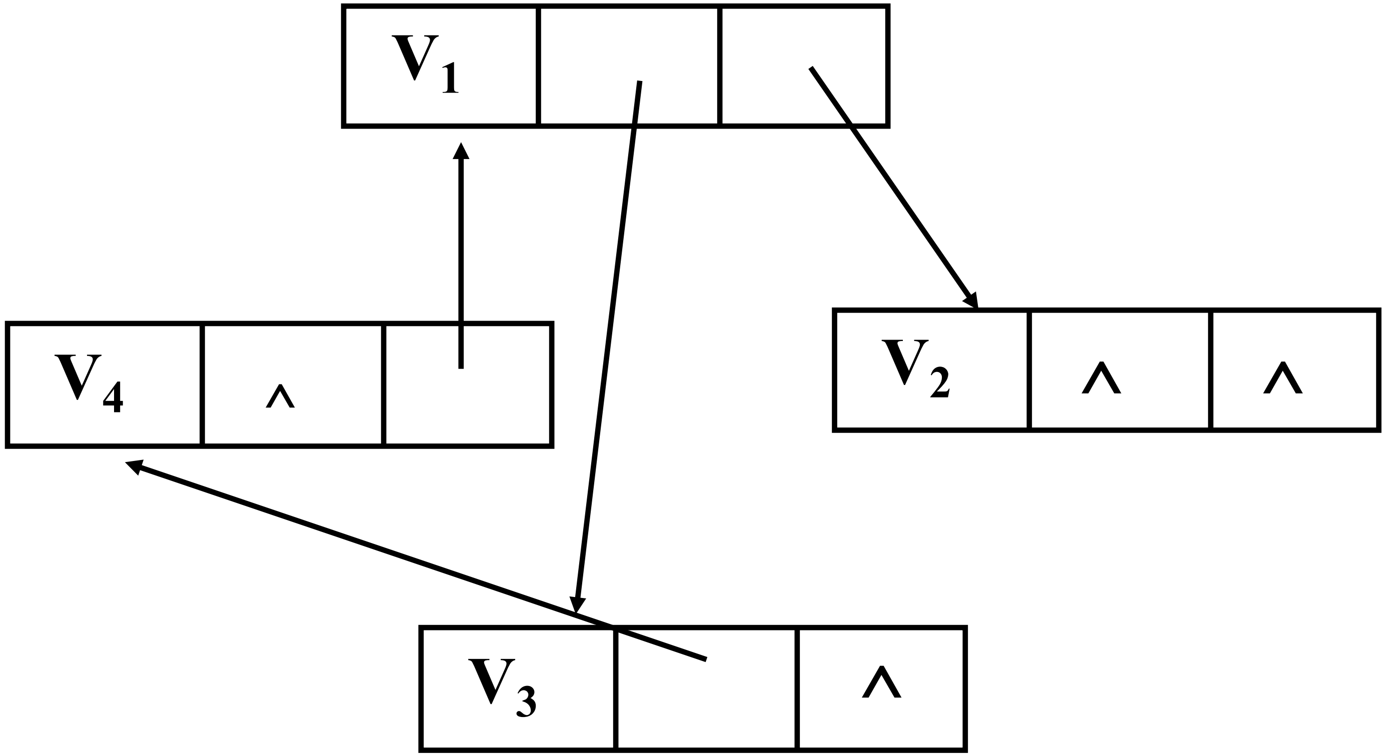
图的存储结构

- 图的结构比较复杂，任意两个顶点间都可能存在联系
- 因此无法以数据元素在存储区中的物理位置来表示元素之间的关系，
- 即图没有顺序存储结构.
- 但可以借助数组的数据类型来表示元素之间的关系

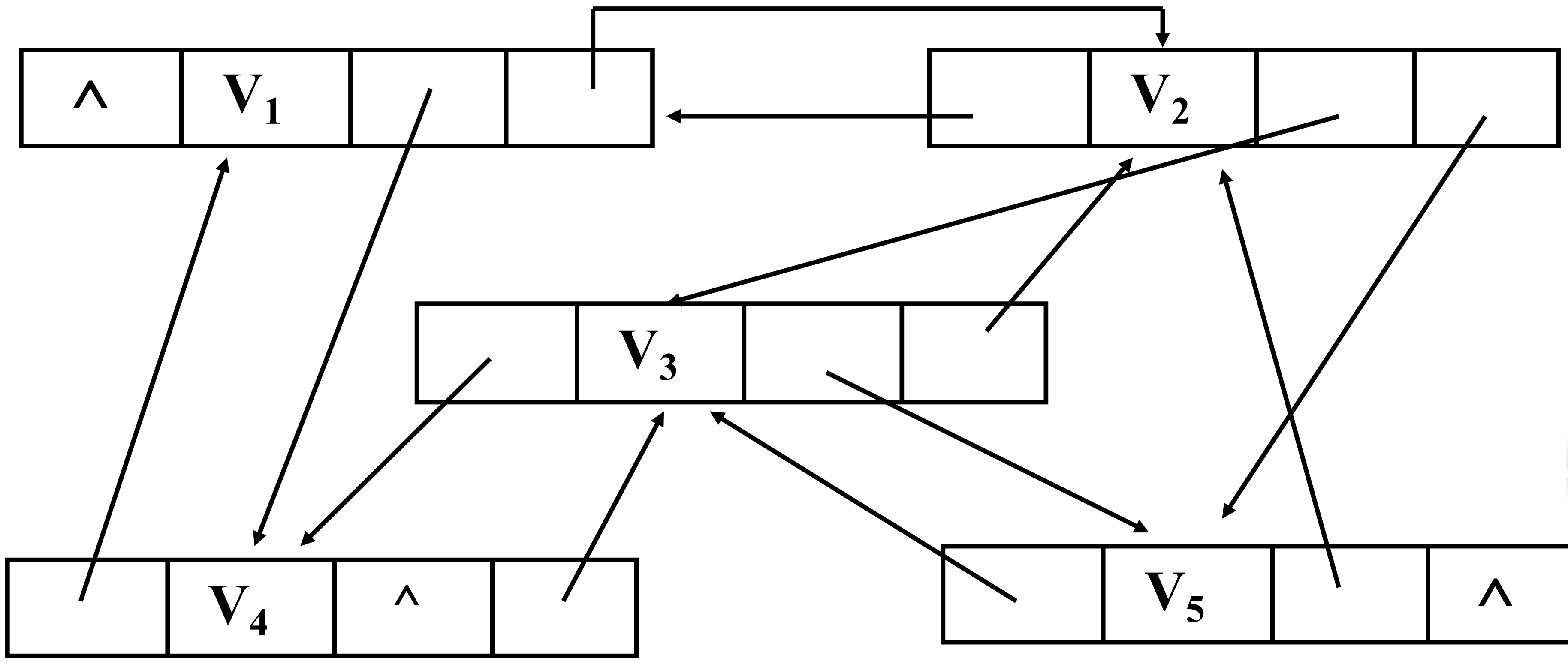
用多重链表表示图:

- 以一个由一个数据域和多个指针域组成的结点表示图中一个顶点
- 指针域存储指向其邻接点的指针
- 数据域存储该顶点的信息。

例：G1的多重链表



例：G_2的多重链表



- 由于图中各个结点的度数各不相同，最大度数和最小度数可能相差很多。
- 因此，按度数最大的顶点设计结点结构，则会浪费存储单元，
- 反之，若按每个顶点的度数设计不同的结点结构，会给运算带来不便。

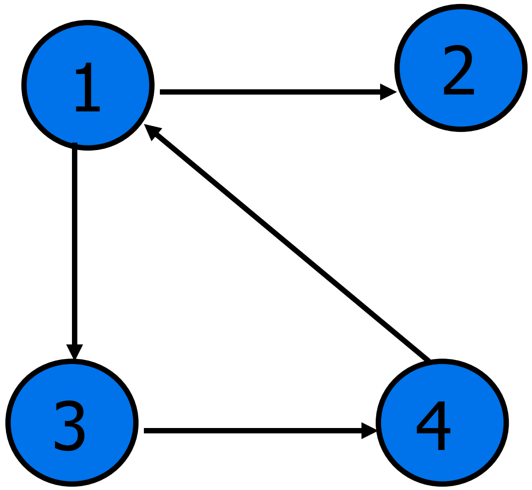
1、邻接矩阵

- 一个图的逻辑结构的说明分两部分：
- 组成图的顶点集合V；
- 顶点偶对集合E。

只需解决集合V和E的存储表示

- 集合V中的所有顶点可以利用一个一维数组表示；
- 集合E用一个二维数组来表示，此二维数组称为**邻接矩阵**，反应了图中各顶点之间的相邻关系。

例子 G_1：



邻接矩阵：

$$A_1 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

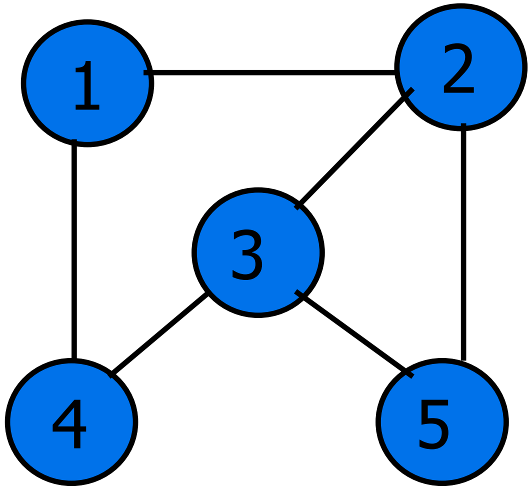
- 设G=(V，E)是有n(n≥1)个顶点的有向图
- 则G的邻接矩阵是具有如下性质的n×n矩阵：

$$A[i,j]=\begin{cases}1 & \text{当} <v_i,v_j> \in E \\ 0 & \text{当} <v_i,v_j> \notin E\end{cases}$$

- 若G = (V, E)是有n(n≥1)个顶点的无向图，
- 则G的邻接矩阵：

$$A[i,j] = A[j,i] = \begin{cases} 1 & \text{当}(v_i, v_j) \in E \\ 0 & \text{当}(v_i, v_j) \notin E \end{cases}$$

例子 G_2：



邻接矩阵：

$$A_2=\begin{bmatrix}0&1&0&1&0\\1&0&1&0&1\\0&1&0&1&1\\1&0&1&0&0\\0&1&1&0&0\end{bmatrix}$$

- 由邻接矩阵，图中顶点的度：

1>无向图:

矩阵第i行（或第i列）的元素之和是顶点v_i的度；

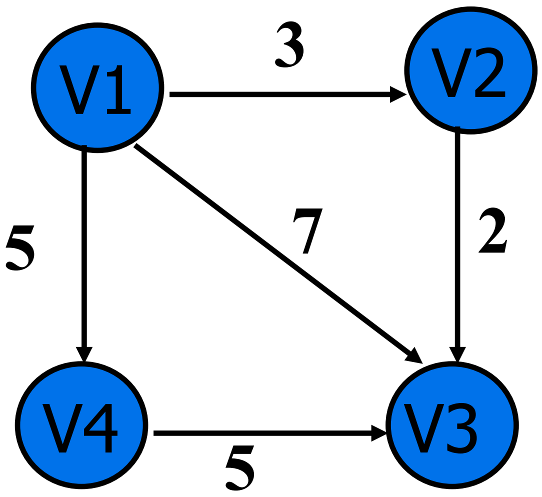
2>有向图:

矩阵第i行元素之和为顶点v_i的出度，第i列的元素之和为顶点v_i的入度。

网络的邻接矩阵可定义为：

$$A[i,j]=\begin{cases} w_{ij}(\text{边的权值}), & \text{若} < v_i, v_j > \text{或}(v_i, v_j) \in E \\ \infty & \text{反之} \end{cases}$$

例：



邻接矩阵：

$$A=\begin{bmatrix}\infty & 3 & 7 & 5\\ \infty & \infty & 2 & \infty\\ \infty & \infty & \infty & \infty\\ \infty & \infty & 5 & \infty\end{bmatrix}$$

图的邻接矩阵存储结构描述：

```
typedef  struct
  { nodetype  nodes[MAXNODE];
    int  arcs[MAXNODE][MAXNODE];
  }graph;
```

顶点数据的类型
此处没有定义

- graph为图的邻接矩阵存储结构；
- 一维数组nodes 用来表示与顶点有关的信息；
- 二维数组arcs用来表示图中顶点之间的关系。

（算法23）在图G中增加一条从顶点v到顶点w的边。

```
void  ins_arc (graph  *g, int  w, int  v)
 {    g - > arcs [v][w]=1;
        return;
   }
```

- 利用邻接矩阵，可方便地实现图的操作：
  - 找出顶点v的第一个邻接点：首先从一维数组中找到v的序号i，则二维数组arcs中第i行上第一个值为“1”的分量所在列号为v的第一个邻接点。
  - 很容易判定任意两个顶点之间是否有边相连。

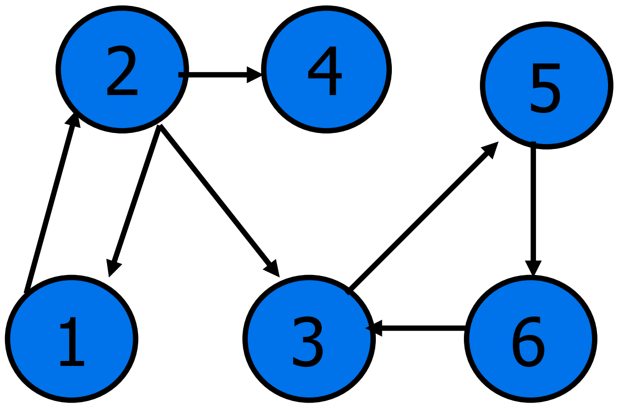
2、邻接表

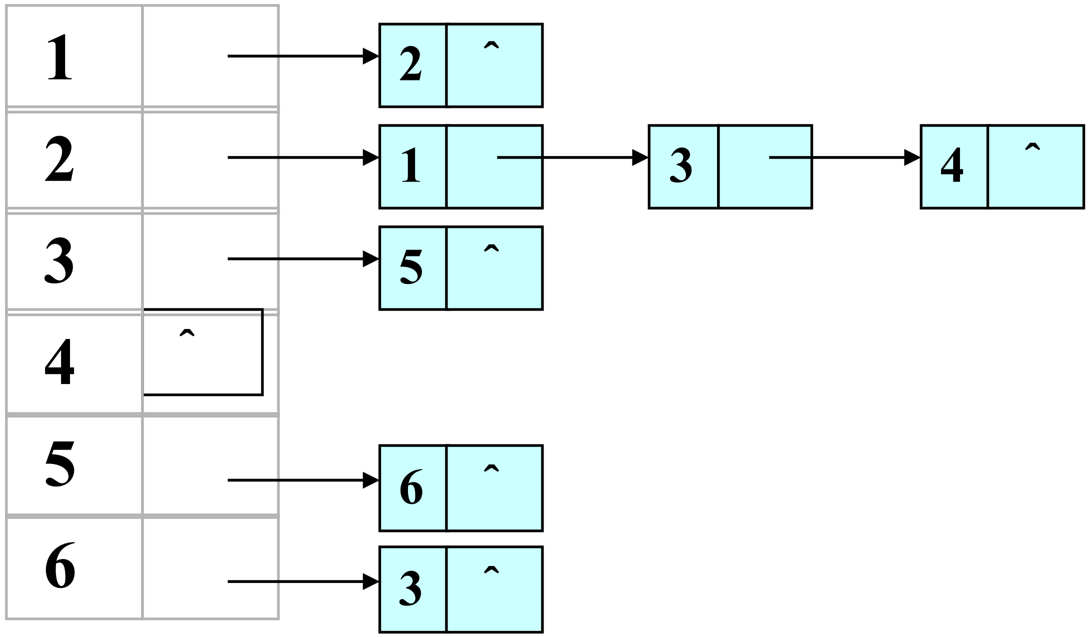
- 邻接矩阵建立时需要先知道图中顶点的个数（静态存储方法）。
- 另外，矩阵占用的存储单元数目与弧的数目无关，若矩阵为稀疏矩阵会造成存储空间的浪费。

邻接表中：

- 每个顶点（图的）建立一个单链表
- 第i个单链表中的结点包含顶点i的所有邻接点。

- 图1.37中G_2的邻接表存储结构





表中结点结构为：

adjvex	data	nextarc

adjvex为顶点Vi的邻接顶点

data为存储与边有关的权值

nextarc为指向Vi的下一个邻接顶点的指针

为了简便于邻接表操作，设一个头结点在每一个单链表上，结构为：

vexdata	firstarc

邻接表中每个单链表的头结点顺序存储。

邻接表的存储结构：

```
struct st_arc
{ int adjvex;
  weighttype date; /* 弧的权值* /
  struct st_arc * nextarc;
};
```

```
typedef struct
{ nodetype vexdata; /* 顶点数据类型 */
  struct st_arc * firstarc;
}headnode;
headnode adjlist[MAXNODE];
```

•邻接表也可容易确定图中顶点的度

•无向图：图中第i个顶点的度等于邻接表中第i条单链表中邻接结点的个数。

•有向图：图中第i个顶点的出度等于邻接表中第i条单链表中邻接结点的个数；入度则需由整个邻接表统计各条链中各个邻接结点的adivex域出现i值的次数。

小结

- 图的基本概念

- 图的存储结构
  - 邻接矩阵
  - 邻接表