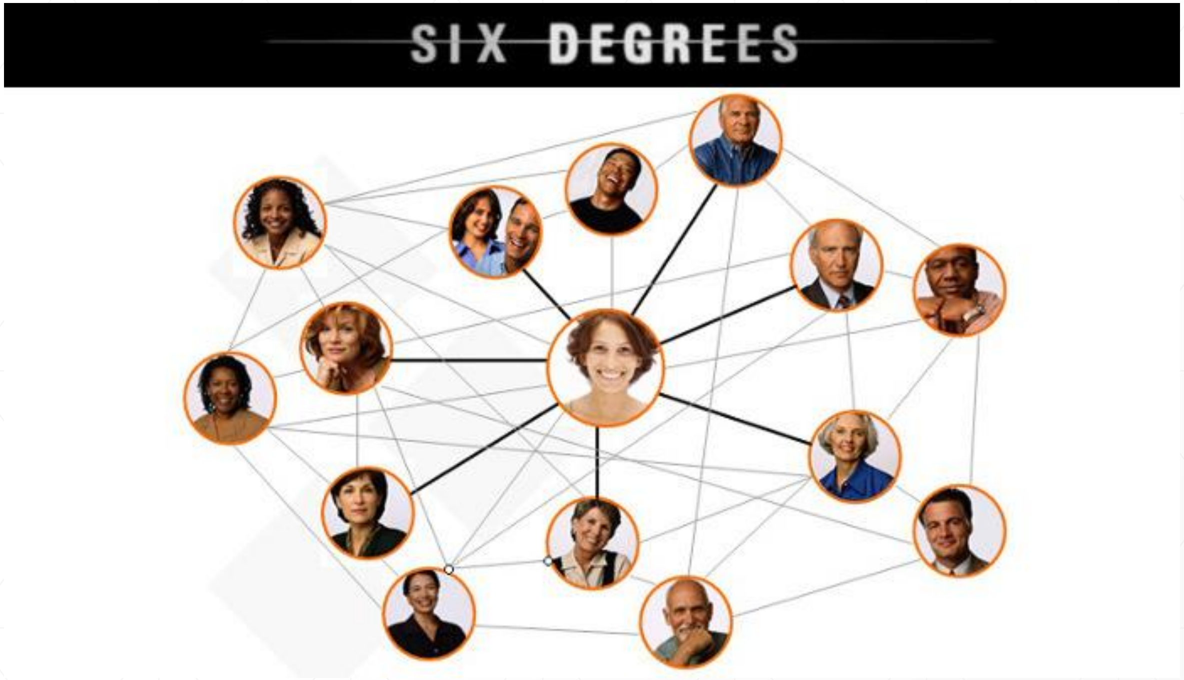
第六章 图（一）

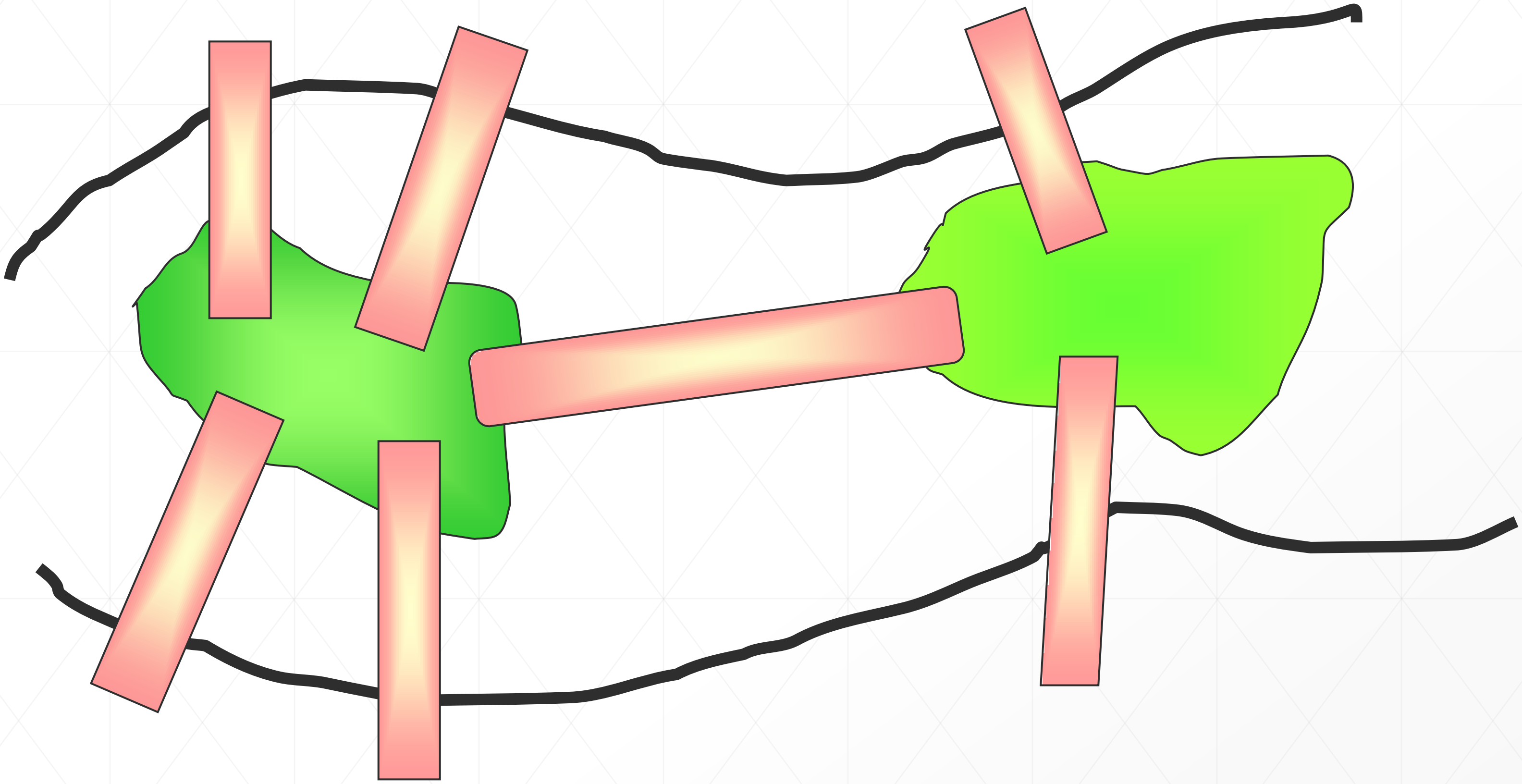
六度空间理论

六度空间理论：你和任何一个陌生人之间所间隔的人不会超过六个，也就是说最多通过六个人，你就能够认识任何一个陌生人。
--1967年哈佛大学的心理学教授斯坦利·米尔格拉姆提出



数学解释：假设一个人能认识25个人以上，那么经过七次介绍（间隔六个人），一个人可以被介绍给25^7，等于6103515625人，超过60亿。

哥尼斯堡七桥问题



- 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，如上图。当时该地的市民热衷于这样一个游戏：“怎么样从某个陆地区域出发，经过每座桥一次且仅一次，最后回到出发地？”

学习目标

- 图结构是一种非线性结构，反映了数据对象之间的任意关系，在计算机科学、数学和工程中有着非常广泛的应用；
- 了解图的定义及相关的术语，掌握图的逻辑结构及其特点；
- 了解图的存储方法，重点掌握图的邻接矩阵和邻接表存储结构；
- 掌握图的遍历方法；
- 了解图的应用，掌握最小生成树算法、最短路径算法、和拓扑排序。

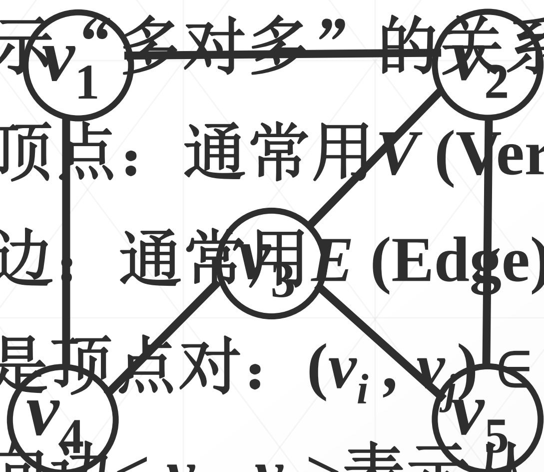
图的定义

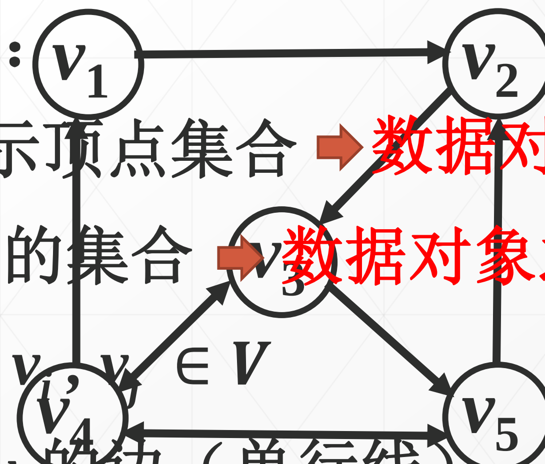
- **图(Graph)**是由**顶点(vertex)**的**有穷非空**集合和顶点之间**边(edge)**的集合组成的一种数据结构，通常表示为：

$$G = (V,\ E)$$

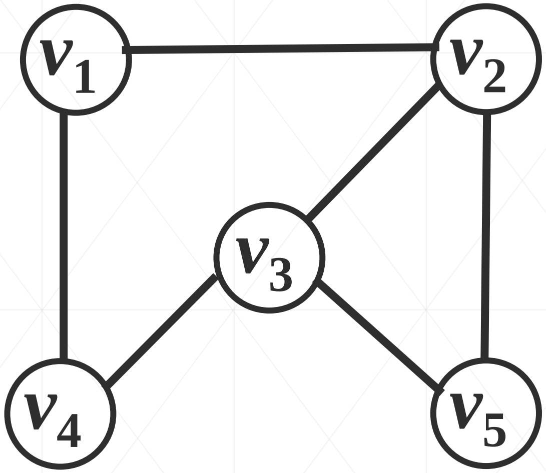
其中：G表示一个图，V是图G中顶点的集合，E是图G中顶点之间边的集合。

- 图表示“多对多”的关系，包含：
  - 一组顶点：通常用V (Vertex) 表示顶点集合 ➔ 数据对象
  - 一组边：通常用E (Edge) 表示边的集合 ➔ 数据对象之间的关系
    - 边是顶点对：$(v_i, v_j) \in E$, 其中 $v_i, v_j \in V$
    - 有向边$<v_i, v_j>$表示从 v_i 指向 v_j 的边（单行线）

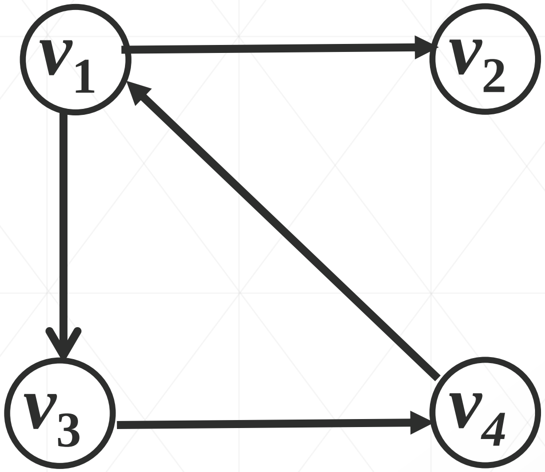




图的常见术语



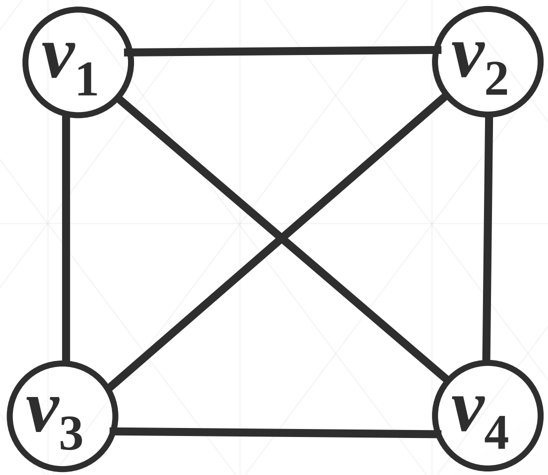
- **无向图：**
  - 若顶点v_i和v_j之间的边没有方向，则称这条边为**无向边**，表示为(v_i, v_j)
  - 如果图的任意两个顶点之间的边都是无向边，则称该图为**无向图**。

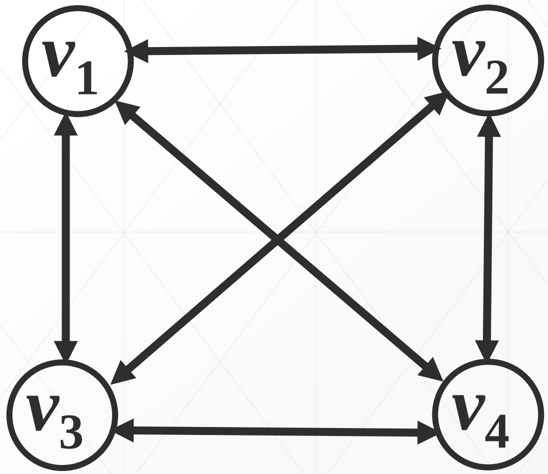


- **有向图：**
  - 若顶点v_i和v_j之间的边都有方向，则称这条边为**有向边(弧)**，表示为$< v_i, v_j >$
  - 如果图的任意两个顶点之间的边都是有向边，则称该图为**有向图**。

图的常见术语(cont.)

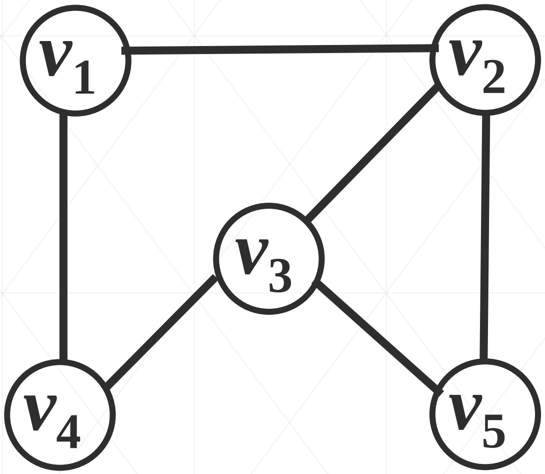
- 无向完全图：在无向图中，如果任意两个顶点之间都存在边，则称该图为无向完全图。
- 有向完全图：在有向图中，如果任意两个顶点之间都存在方向相反的两条弧，则称该图为有向完全图。



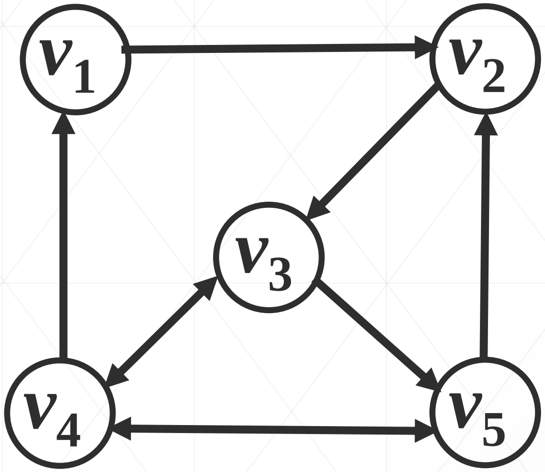


- 含有n个顶点的无向完全图有多少条边？ $n(n-1)/2$
- 含有n个顶点的有向完全图有多少条弧？ $n(n-1)$

图的常见术语(cont.)

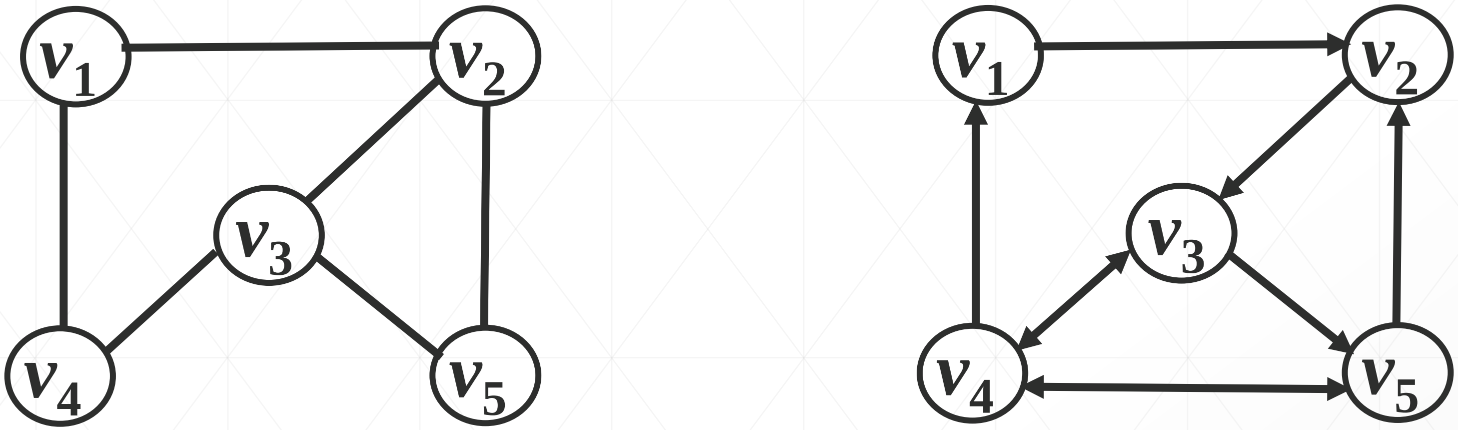


- 邻接&依附
  - 在无向图中，对于任意两个顶点v_i和v_j，若存在边(v_i, v_j)，则称顶点v_i和v_j相邻，互为邻接点，同时称边(v_i, v_j)依附于顶点v_i和顶点v_j。
    - 如：v_2的邻接点：v_1，v_3，v_5



  - 在有向图中，对于任意两个顶点v_i和v_j，若存在有向边$<v_i, v_j>$，则称顶点v_i邻接到顶点v_j，顶点v_j邻接自顶点v_i，同时称弧$<v_i, v_j>$依附于顶点v_i和v_j，其中v_i为弧尾，为v_j弧头。
    - 如：v_1邻接到v_2，v_1邻接自v_4

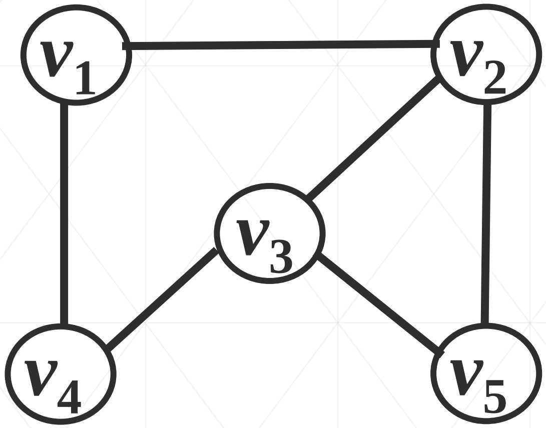
图的常见术语(cont.)

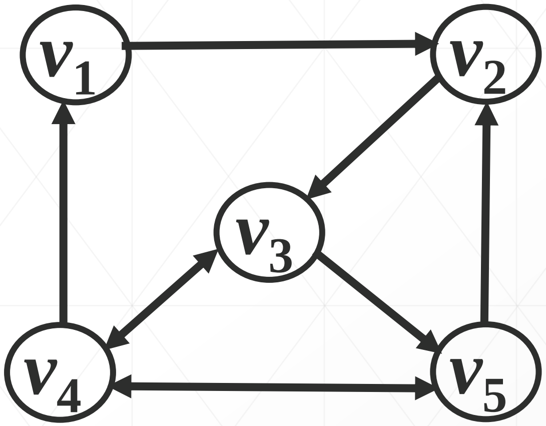


- 顶点的度：在无向图中，顶点v的度是指依附于该顶点的边数，通常记为 **TD** (v)。
- 顶点的入度：在有向图中，顶点v的入度是指以该顶点为头的弧的数目，记为**ID** (v);
- 顶点的出度：在有向图中，顶点v的出度是指以该顶点为尾的弧的数目，记为**OD** (v)。

在有向图中， **TD** (v)= **ID** (v) + **OD** (v)

图的常见术语(cont.)





- 具有n个顶点、e条边的无向图G，满足如下的关系：

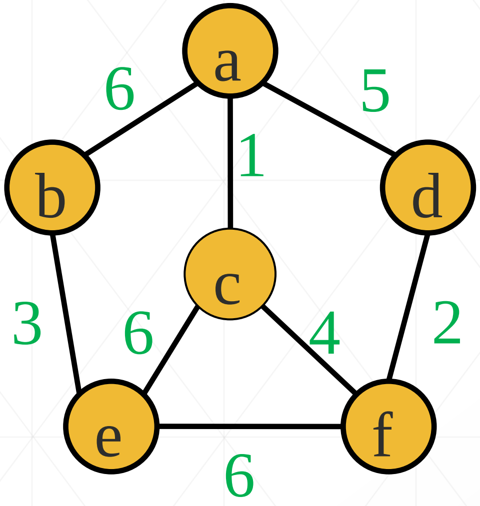
$$\sum_{i=1}^{n} TD(v_i) = 2e$$

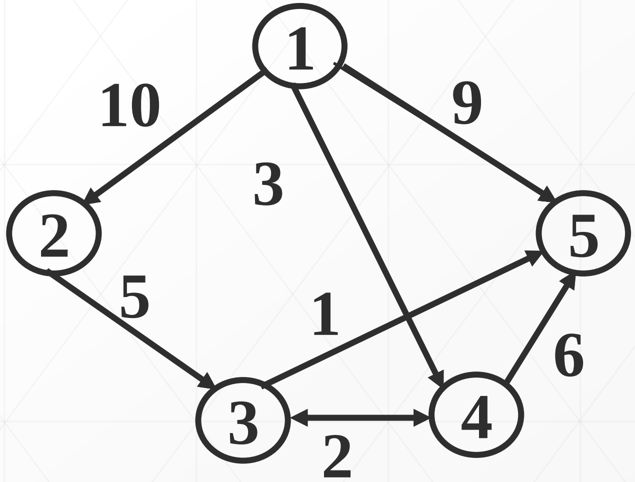
- 具有n个顶点、e条边的有向图G，满足如下的关系：

$$\sum_{i=1}^{n} ID(v_i) = \sum_{i=1}^{n} OD(v_i) = e$$

图的常见术语(cont.)

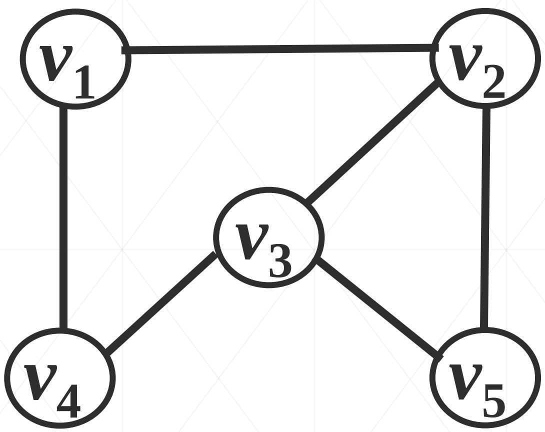
- **网(Network)**: 带权的图称为网
- **权(Weight)**: 与图的边或弧相关的数



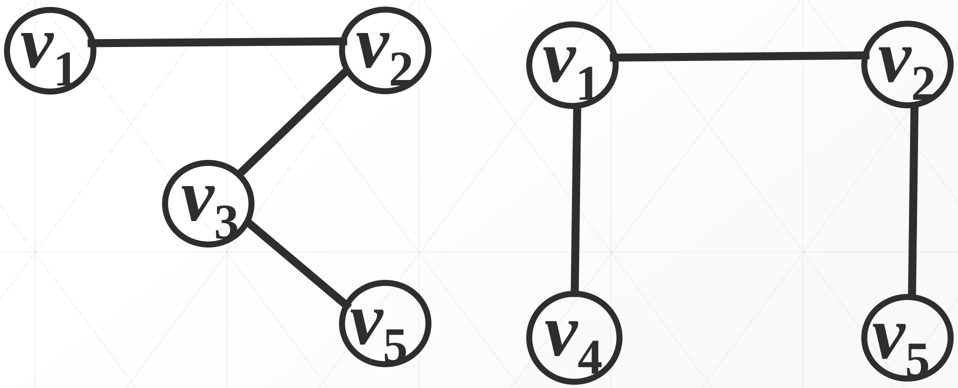


图的常见术语(cont.)

- **子图(Subgraph):** 设有有个图$G = (V，E)$和$G' = (V'，E')$, 若$V' \subseteq V$且$E' \subseteq E$, 则称图G'是图G的子图。



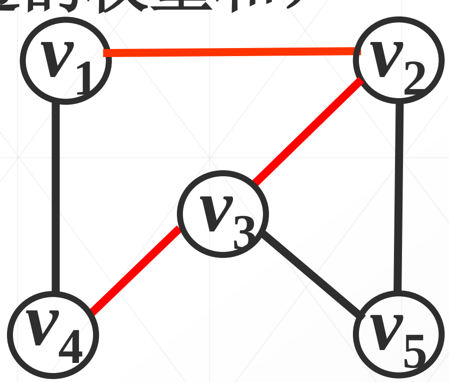
a. 图 G_1



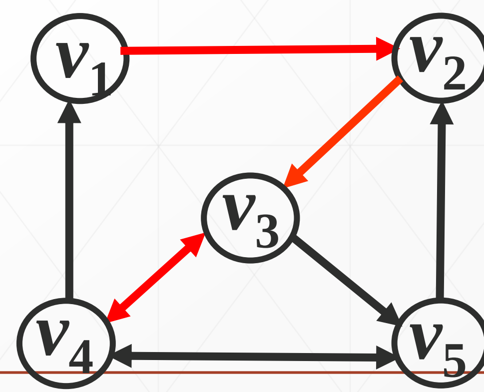
b. 图 G_1 的子图

图的常见术语(cont.)

- 在无向图G=(V, E) 中，顶点v_p到v_q的**路径**是一个**顶点序列**$(v_p , v_{i1} , v_{i2} , \ldots\ v_{im} , v_q)$，其中任一对相邻的顶点间都有图中的边，即$(v_p , v_{i1}), (v_{i1} , v_{i2}), \ldots , (v_{im} , v_q) \in E$
- 如果G是有向图，则路径也是**有向**的。在有向图G =(V, E)中，若存在一个**顶点序列**$(v_p , v_{i1} , v_{i2} , \ldots\ v_{im} , v_q)$，使得有向边$<v_p ,v_{i1}>, <v_{i1} ,v_{i2}>, \ldots , <v_{im} ,v_q> \in E$，则称顶点$v_p$到$v_q$有一条**有向路径**
- **路径长度**是指此路径上边或弧的数目（如果带权，则是所有边的权重和）



v_1到v_4有路径(v_1 , v_2, v_3, v_4)

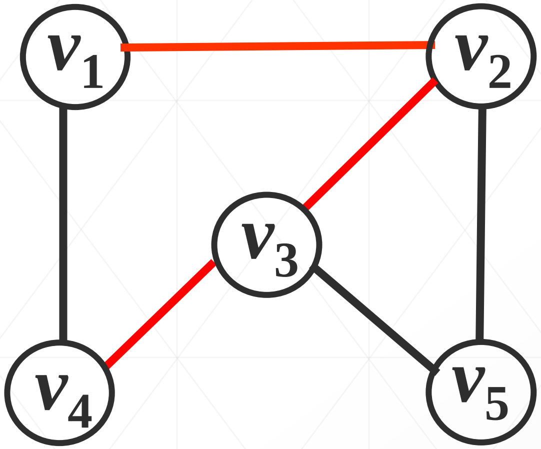


v_1到v_4有路径(v_1 , v_2, v_3, v_4)

图的常见术语(cont.)

- **回路或环**：路径的开始顶点与最后一个顶点相同，即路径中$(v_p, v_{i1}, v_{i2}, \ldots v_{im}, v_q)$，$v_p = v_q$

- **简单路径**：路径的顶点序列中顶点不重复出现，即v_p到v_q之间的所有顶点都不同



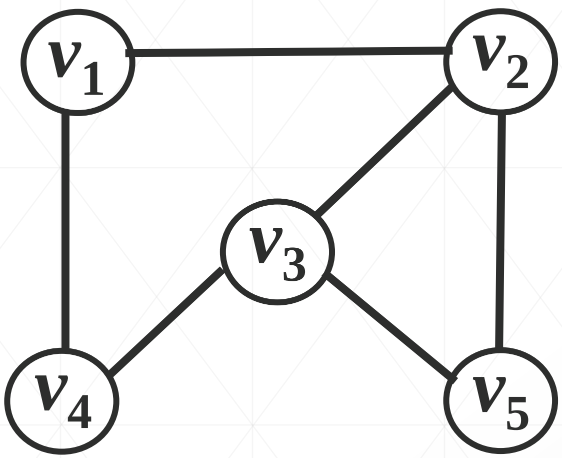
v_1到v_4有路径(v_1, v_2, v_3, v_4)

v_1到v_1构成环$(v_1, v_2, v_3, v_4, v_1)$

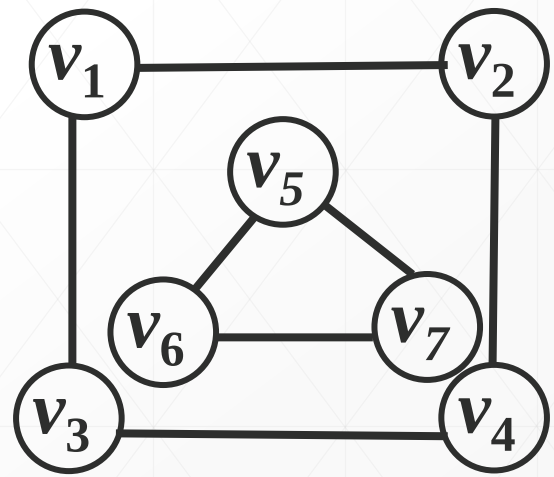
图的常见术语(cont.)

- **连通**：在无向图中，若从顶点v_i到顶点v_j $(i \neq j)$有**路径**, 则称顶点v_i与v_j是**连通**的。

- **连通图**：图中所有顶点都是连通的。



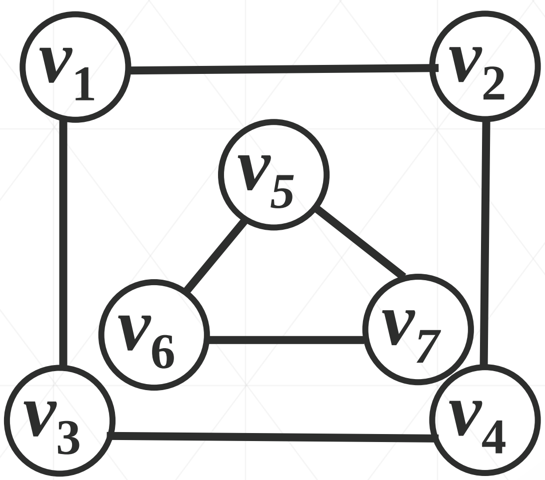
a. 连通图 G_1



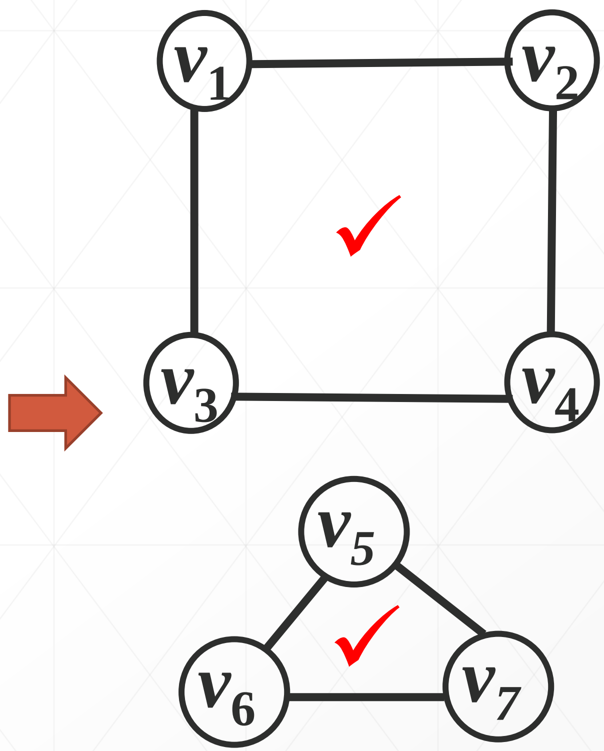
b. 非连通图 G_2

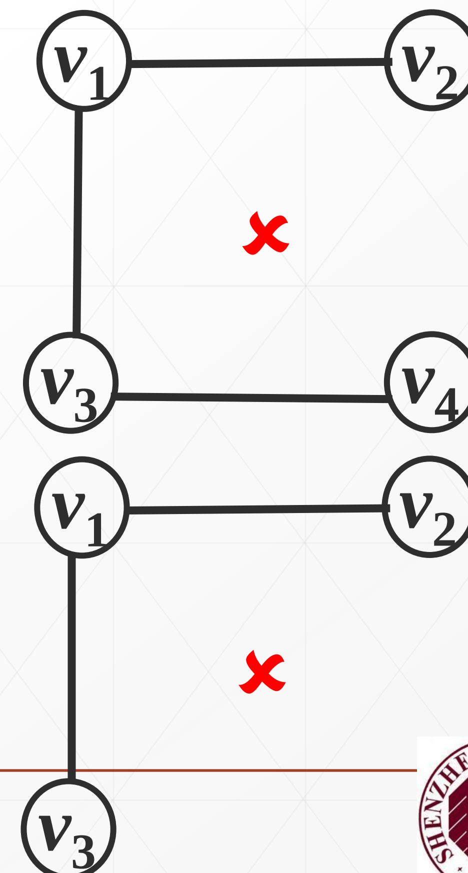
图的常见术语(cont.)

- **连通分量**：非连通的无向图的极大连通子图。
  - 极大顶点数：再加1个顶点就不连通了
  - 极大边数：包含子图中所有顶点相连的所有边



b. 非连通图 G_2

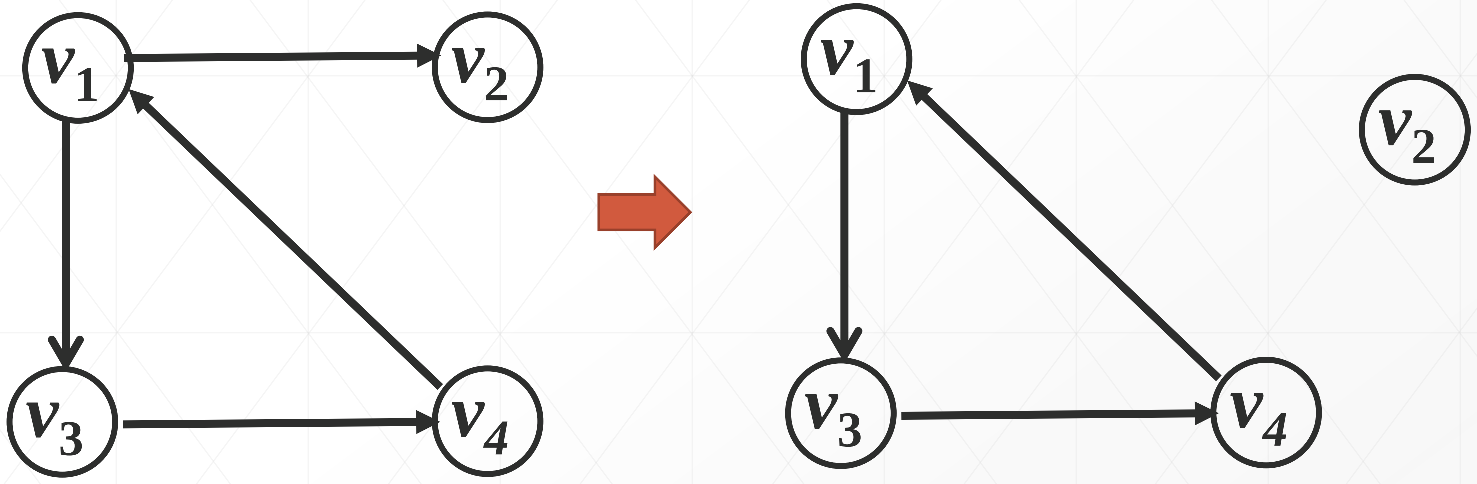




c.判断是否为非连通图 G_2 的连通分量？

图的常见术语(cont.)

- **强连通**：有向图中顶点v_i和v_j之间存在双向路径，则称v_i和v_j是强连通的
- **强连通图**：有向图中任意两顶点均强连通
- **强连通分量**：有向图的极大强连通子图



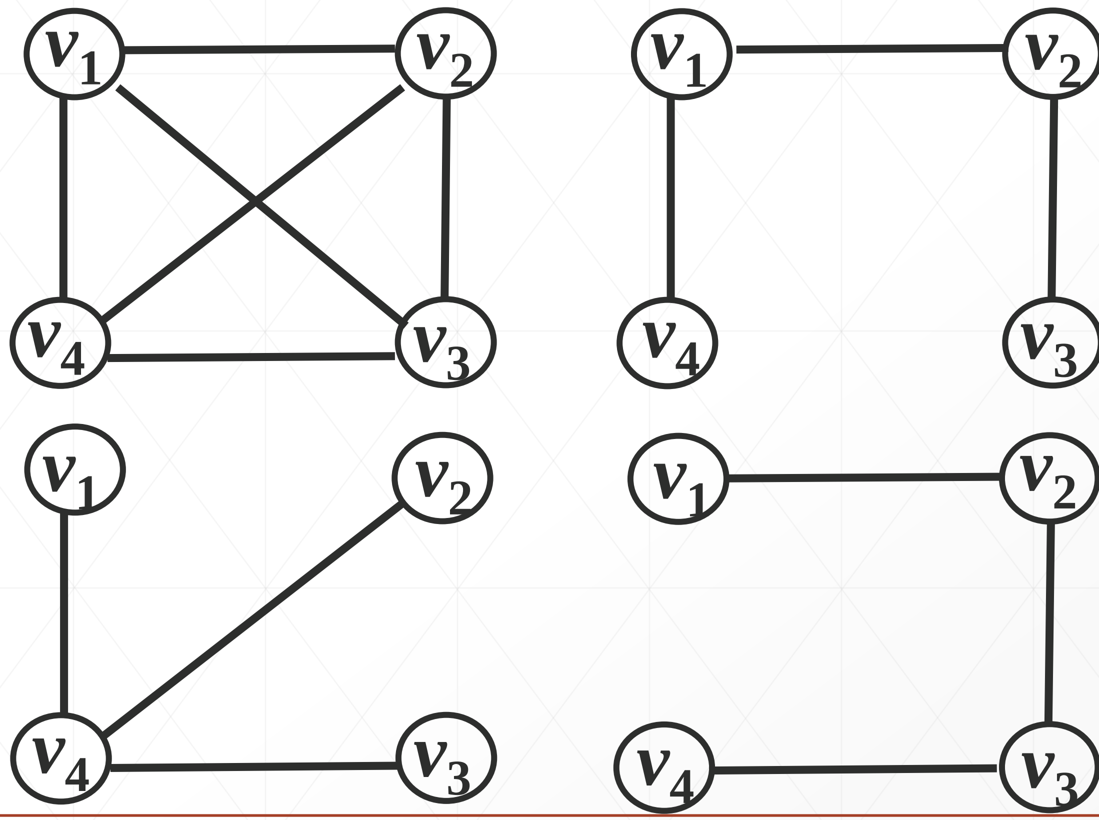
图的常见术语(cont.)

不形成任何回路

生成树存在 $\boxed{\leftrightarrow}$ 图连通

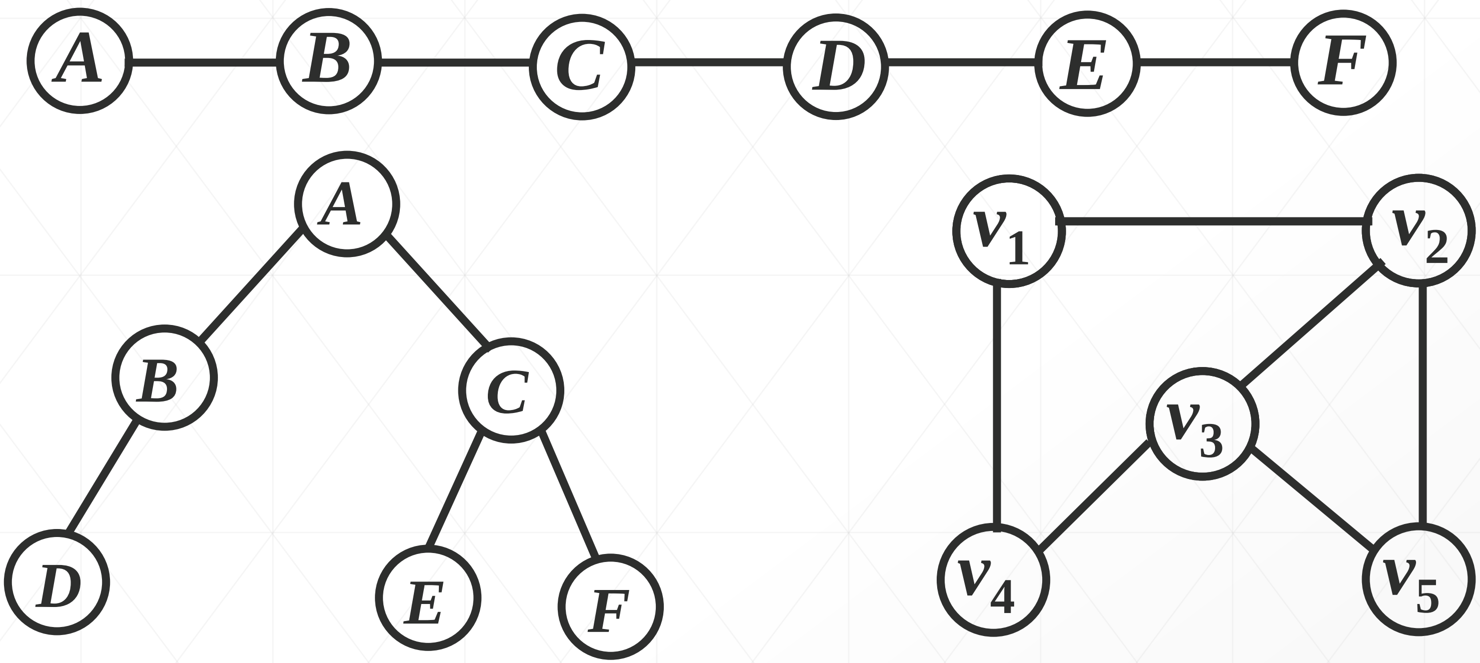
- 生成树：一个连通图的生成树是一个极小的连通子图
- 包含图的全部n个顶点，但只有足以构成一棵树的$n-1$条边

多--构成回路
少--不连通



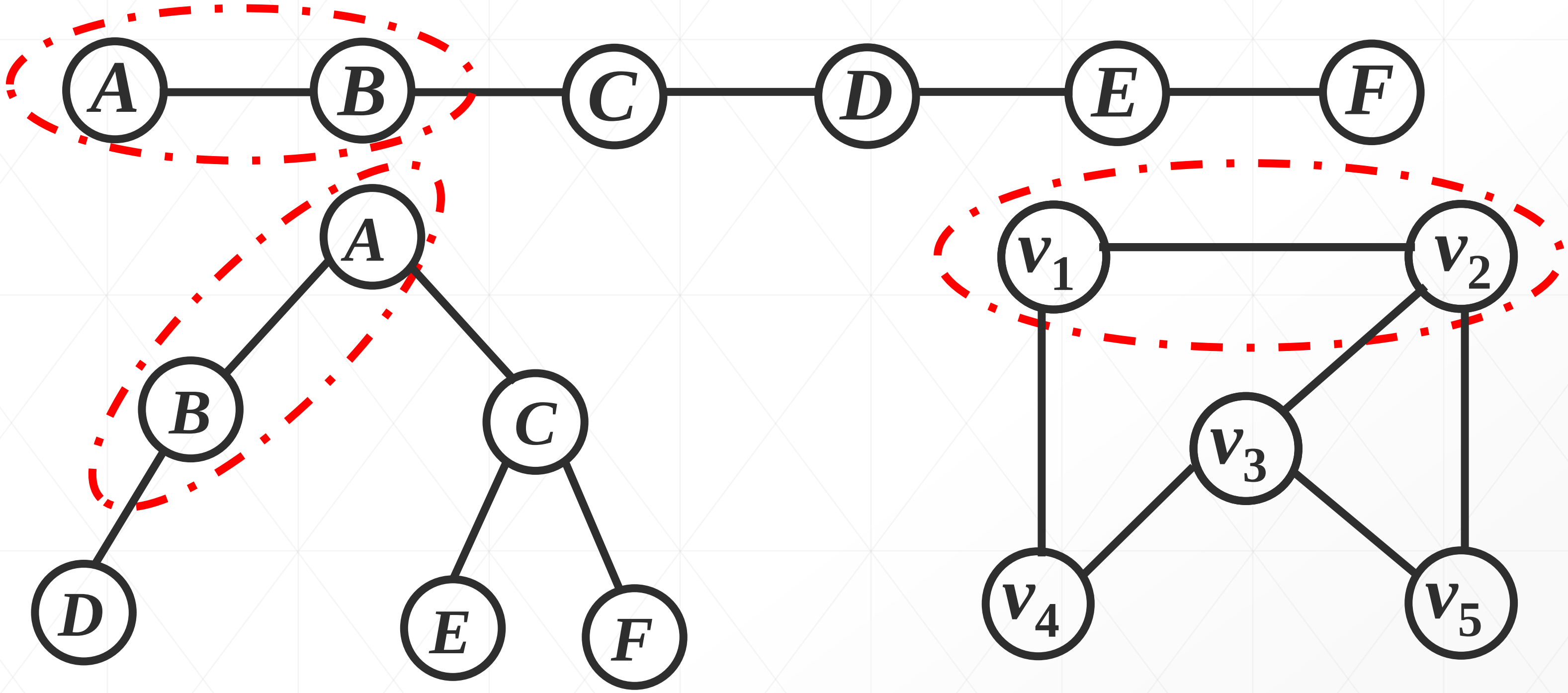
向生成树中任加一条边都一定构成回路

不同逻辑结构之间的比较



- 在线性结构中，数据元素之间仅具有**线性关系(1: 1)**；
- 在树型结构中，结点之间具有**层次关系(1: *m*)**；
- 在图型结构中，任意两个顶点之间**都可能有关系(*m*: *n*)**。

不同逻辑结构之间的比较(cont.)



- 在线性结构中，元素之间的关系为**前驱**和**后继**；
- 在树型结构中，结点之间的关系为**双亲**和**孩子**；
- 在图型结构中，顶点之间的关系为**邻接**。

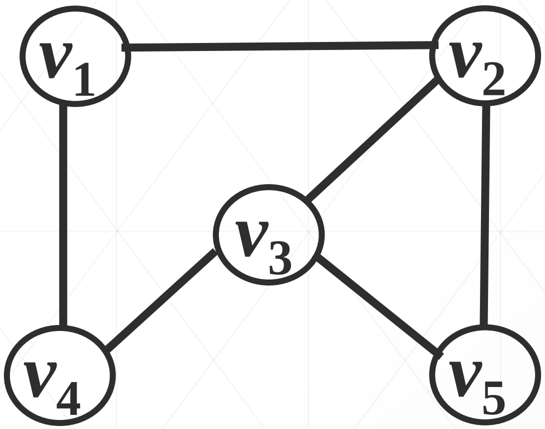
- 子图：假设有图G = (V，E)，且V’⊆V，E’⊆E，则顶点集的子集V’和边集的子集E’可构成图G的子图 ✘
  - 若E’中顶点不是V’的元素时，无法构成子图

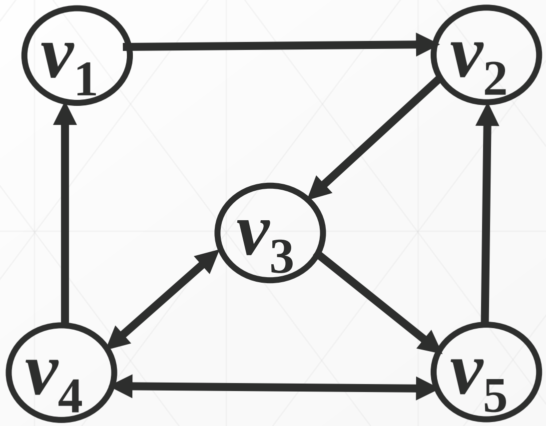
- 连通图 *vs.* 无向完全图
  - 连通图是指无向图中所有顶点都是连通的；连通是指在无向图中， 若从v_i到顶点v_j $(i \neq j)$有路径, 则称顶点v_i与v_j是连通的。
  - 无向完全图是指任意两个顶点之间都存在边；
  - n个顶点的完全图有n(n-1)/2条边；而连通图则不一定，但至少有n-1条边

- 生成树：一个连通图的生成树是一个极小的连通子图，包含图的全部n个顶点，但只有足以构成一棵树的$n-1$条边
  - 树---不形成任何回路
  - 生成树---包含全部顶点
  - 极小的连通子图：能够使全部顶点连通而又不形成任何回路，因此包含n个顶点与n-1条边

图的存储结构

- 是否可以采用顺序存储结构存储图?
  - 图的特点：顶点之间的关系是$m:n$，即任何两个顶点之间都可能存在关系（边），无法通过存储位置表示这种任意的逻辑关系，所以图无法采用顺序存储结构。
- 如何存储图?
  - 考虑图的定义，图是由顶点和边组成的；
  - 如何存储顶点、如何存储边（顶点之间的关系）？



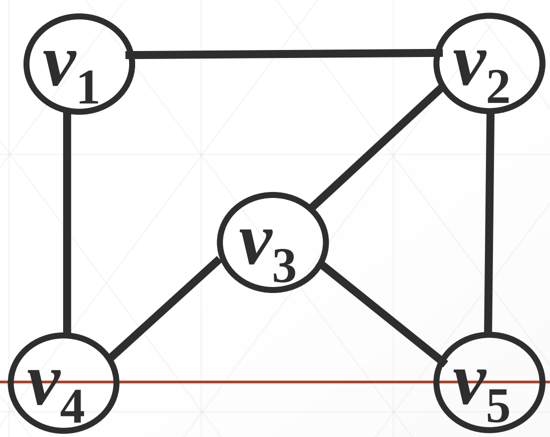


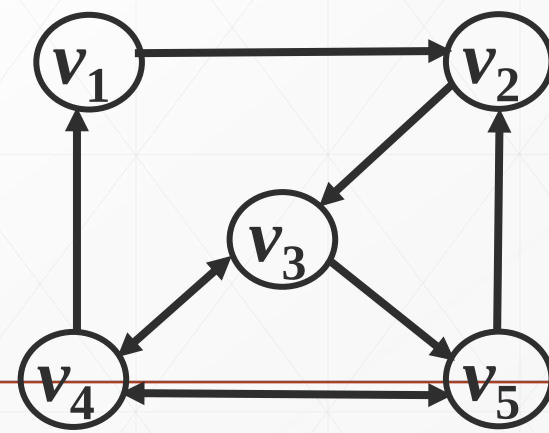
图的存储结构(cont.)

1. 邻接矩阵(***Adjacency Matrix***)表示（数组表示法）

- 基本思想：
  - 用一个一维数组存储图中顶点的信息，用一个二维数组（称为邻接矩阵）存储图中各顶点之间的邻接关系。
  - 假设图$G=(V，E)$有n个顶点，则邻接矩阵是一个$n \times n$的方阵，定义为：

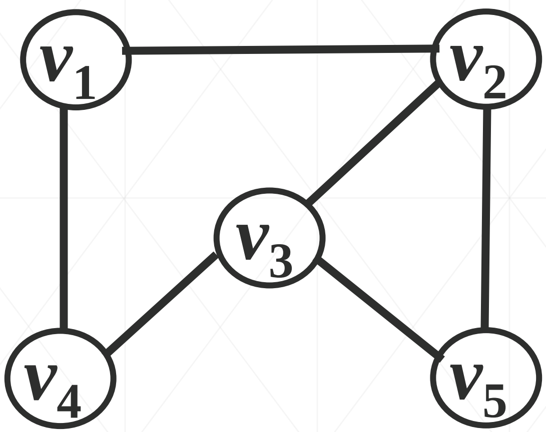
$$\text{edge}[i][j] = \begin{cases} 1 & 若 (i, j) \in E 或 \langle i, j \rangle \in E \\ 0 & 否则 \end{cases}$$





图的存储结构(cont.)

➢无向图的邻接矩阵：



vertex=

v_1	v_2	v_3	v_4	v_5

$$
\begin{array}{c}
\begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\
edge = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}
\begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array}
\end{array}
$$

- 存储结构特点：
  - 主对角线为 0 且一定是对称矩阵；

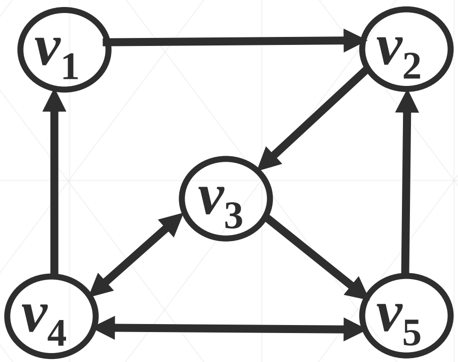
问题：1. 如何判断顶点v_i和v_j之间是否存在边？

2. 如何求顶点v_i的度？➡对应行（或列）非0元素的个数

3. 如何求顶点v_i的所有邻接点？

图的存储结构(cont.)

➢有向图的邻接矩阵：



$$\text{vertex}= \begin{array}{|c|c|c|c|c|} \hline v_1 & v_2 & v_3 & v_4 & v_5 \\ \hline \end{array}$$

$$\text{edge} = \begin{array}{cc} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} & \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} & \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \end{array}$$

- 存储结构特点：
  - 有向图的邻接矩阵一定不对称吗？

问题：1. 如何判断顶点v_i和v_j之间是否存在有向边？

2. 如何求顶点v_i的出度？ ➡ 对应行非0元素的个数

3. 如何求顶点v_i的入度？ ➡ 对应列非0元素的个数

图的存储结构(cont.)

- 网的邻接矩阵：
  - 在网络中，两个顶点邻接，只要把**edge [*i*] [*j*]**的值定义为边的权重即可。

☹ 问题：v_i和v_j之间若没有边该怎么表示？

  - 在网络中，两个顶点如果不邻接，则视为距离无穷大；如果邻接，则两个顶点间存在一个距离值（即权值）

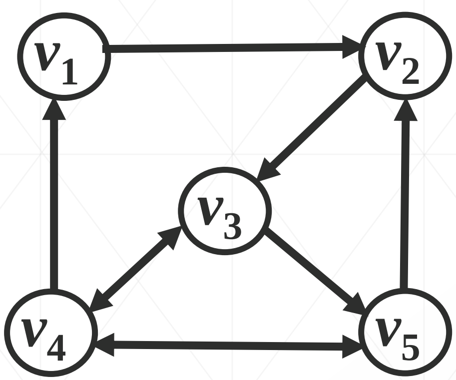
$$\textbf{edge}\ [i]\ [j] = \begin{cases} w_{i,j} & \text{若}\ (i,j) \in E\ \text{或}\ \langle i,j \rangle \in E \\ \infty & \text{否则} \end{cases}$$

图的存储结构(cont.)

假设图G有n个顶点e条边，则该图的存储需求为$O(n+n^2) = O(n^2)$，与边的条数e无关。

➢邻接矩阵表示的**存储结构定义**：

```
typedef struct {
    VertexType vertex [NumVertices];   //顶点表
    int edge[NumVertices][NumVertices];
            //邻接矩阵 - 边表
    int n, e;  //图的顶点数与边数
} MGraph;
```



vertex

v_1	v_2	v_3	v_4	v_5

$$\text{edge}=\begin{array}{c@{}c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} & \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} & \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \end{array}$$

图的存储结构(cont.)

- 邻接矩阵--有什么好处？

☑ 直观、好理解

☑ 方便检查任意一对顶点间是否存在边

☑ 方便找任一顶点的所有邻接点

☑ 方便计算任一顶点的度

  - 无向图：对应行（或列）非0元素的个数
  - 有向图：对应行非0元素的个数是出度；对应列非0元素的个数是入度

图的存储结构(cont.)

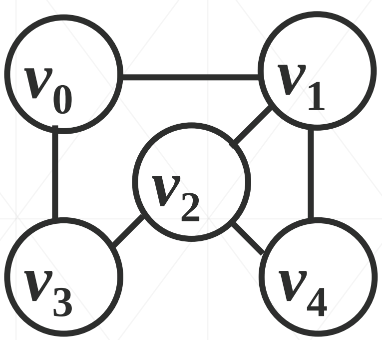
- 邻接矩阵--有什么不好?
- ☑ 浪费空间--存稀疏图（点很多而边很少）有大量无效元素
  - 对稠密图（特别是完全图）还是很合算的
- ☑ 浪费时间--统计稀疏图中一共有多少条边

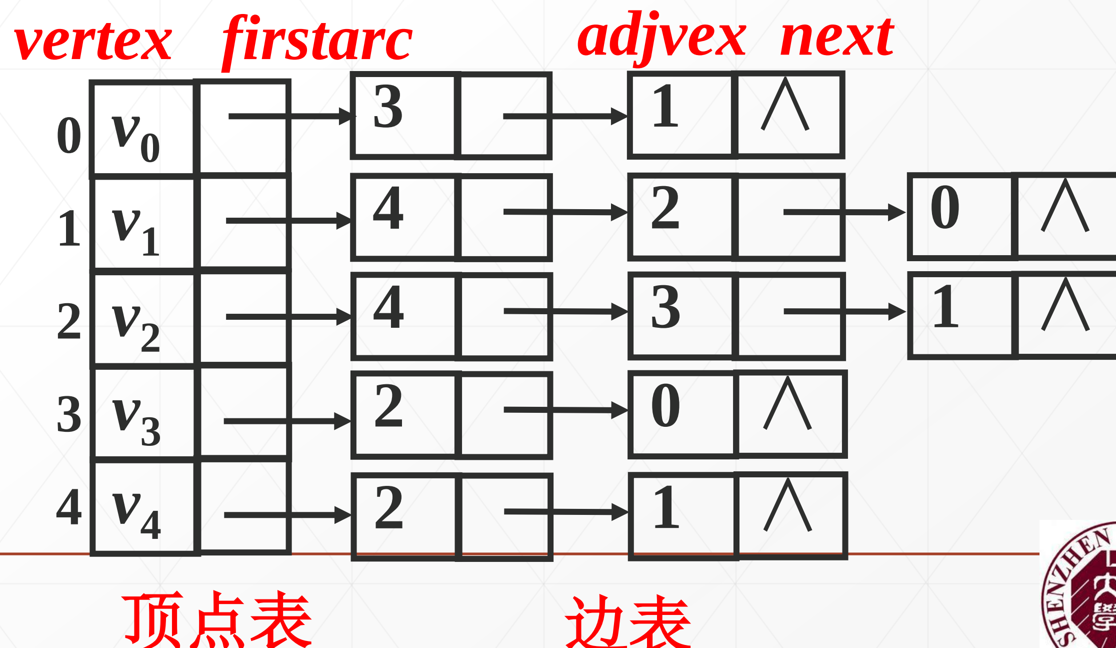
图的存储结构(cont.)

2. 邻接表(*Adjacency List*)表示

➢无向图的邻接表：

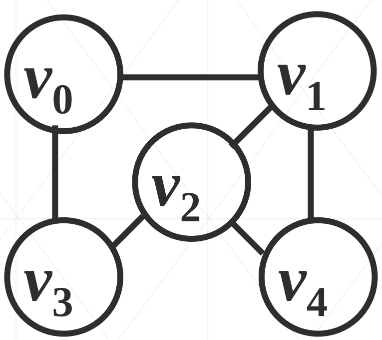
- 对于无向图的每个顶点v_i，将所有与v_i相邻的顶点链成一个单链表，称为顶点v_i的边表；
- 把所有边表的指针和存储顶点信息的一维数组构成顶点表。

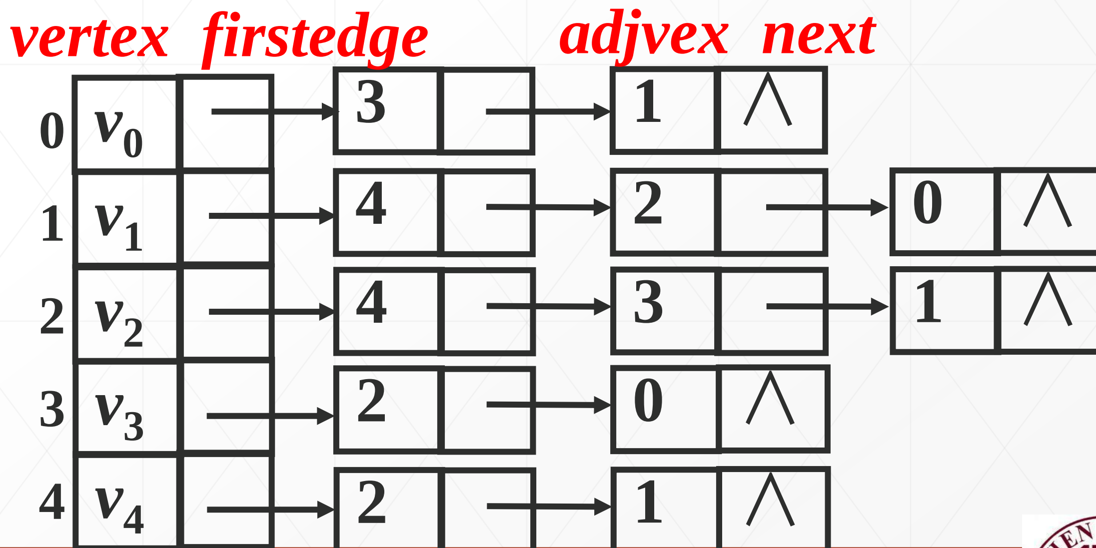




图的存储结构(cont.)

- 无向图邻接表的存储特点:
  - 如何求顶点v_i的所有邻接点? ➡ 遍历顶点v_i的边表,边表中的所有结点
  - 如何求顶点v_i的度? ➡ 顶点v_i的边表中的结点个数
  - 如何判断顶点v_i和顶点v_j之间是否存在边? ➡ 顶点v_i的边表中是否存在邻接点为j的结点
  - 存储空间需求$O(n+2e)$



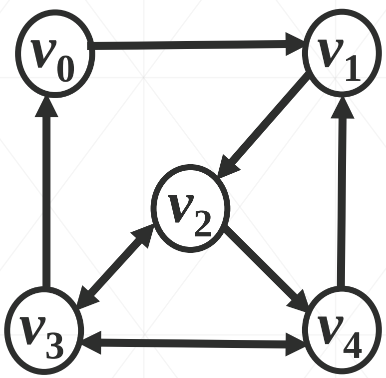


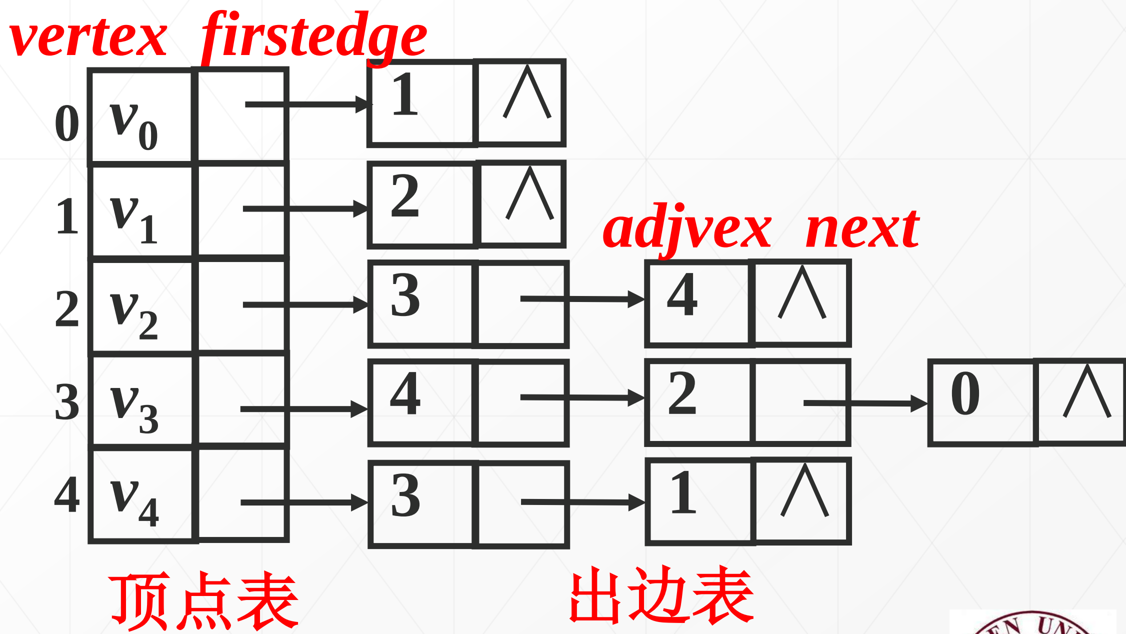
顶点表 边表

图的存储结构(cont.)

➢有向图的邻接表

- 对于**有向图**的每个顶点v_i，将**邻接于v_i**的所有顶点链成一个**单链表**，称为顶点v_i的**出边表**；
- 再把所有出边表的指针和存储顶点信息的一维数组构成**顶点表**。



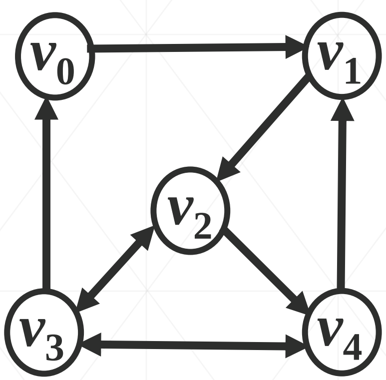


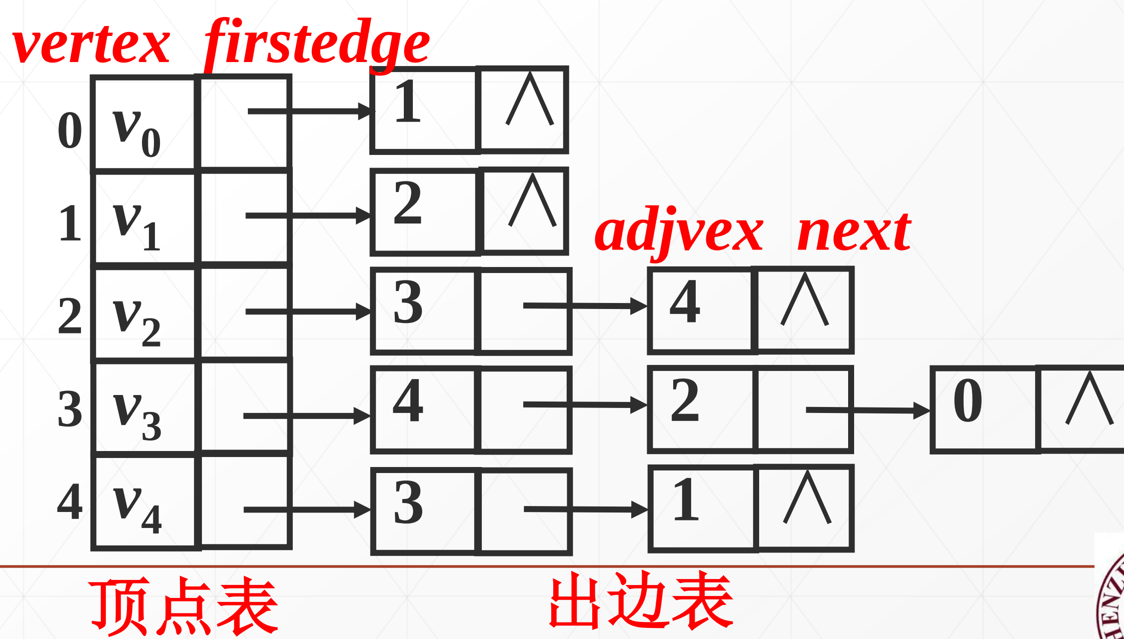
图的存储结构(cont.)

- 有向图的正邻接表的**存储特点**
  - 如何求**邻接于**顶点v_i的所有顶点？
  - 如何求顶点v_i的**出度**？如何求顶点v_i的**入度**？
  - 如何判断顶点v_i和顶点v_j之间是否存在有向边？
  - 空间需求：$O(n+e)$

出度是v_i出边表中的结点个数

需要搜索第i个或第j个链表

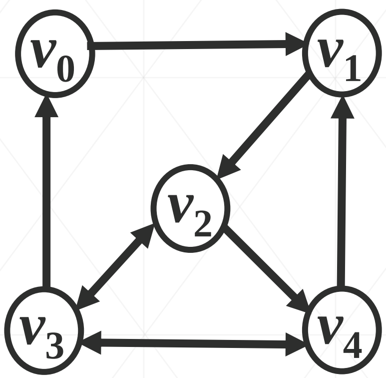


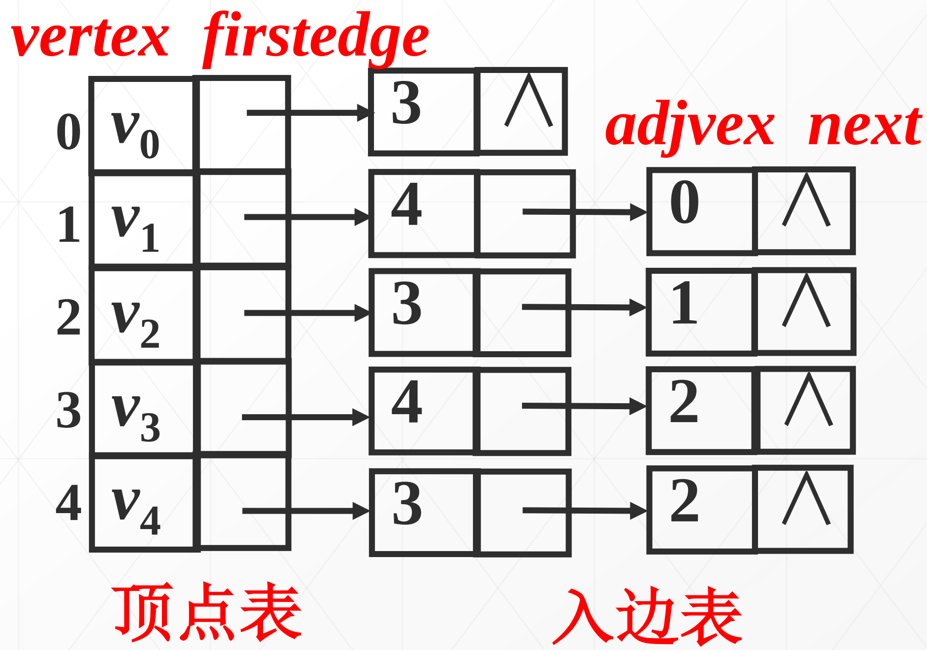


图的存储结构(cont.)

入度是v_i入边表中的结点个数

- 有向图的邻接表--逆邻接表
  - 对于有向图的每个顶点v_i，将邻接到v_i的所有顶点链成一个单链表，称为顶点v_i的入边表；
  - 再把所有入边表的指针和存储顶点信息的一维数组构成顶点表。





图的存储结构(cont.)

- 邻接表
- ☑ 方便找任一顶点的所有"邻接点"
- ☑ 节约稀疏图的空间
  - 需要n个头指针+ $2e$个结点（每个结点至少2个域）
- ☑ 方便计算任一顶点的"度"？
  - 对无向图：是的
  - 对有向图：只能计算出度； 求入度必须遍历邻接表（或是构造逆邻接表，存指向自己的边来方便计算入度）
- ☑ 方便检查任意一对顶点间是否存在边？
  - ☹ **No**

图的存储结构(cont.)

➢图的存储结构的比较---邻接矩阵和邻接表

	空间性能	时间性能	适用范围	唯一性
邻接矩阵	O（n^2）	O（n^2）	稠密图	唯一
邻接表	O（n+e）	O（n+e）	稀疏图	不唯一