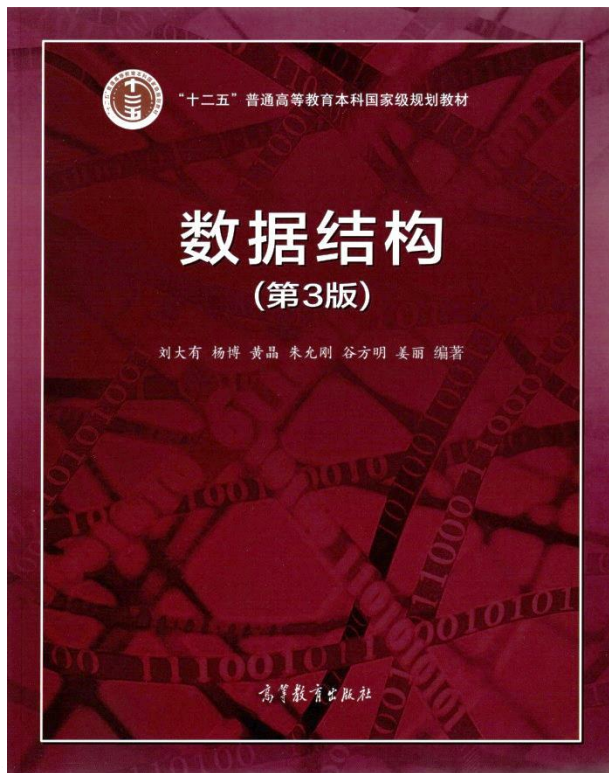




图的概念与存储结构

- 图的基本概念
- 图的存储结构



数据之法
结构之美
算法之道



*Programming is a skill best acquired
by **practice** and example rather
than from books.*

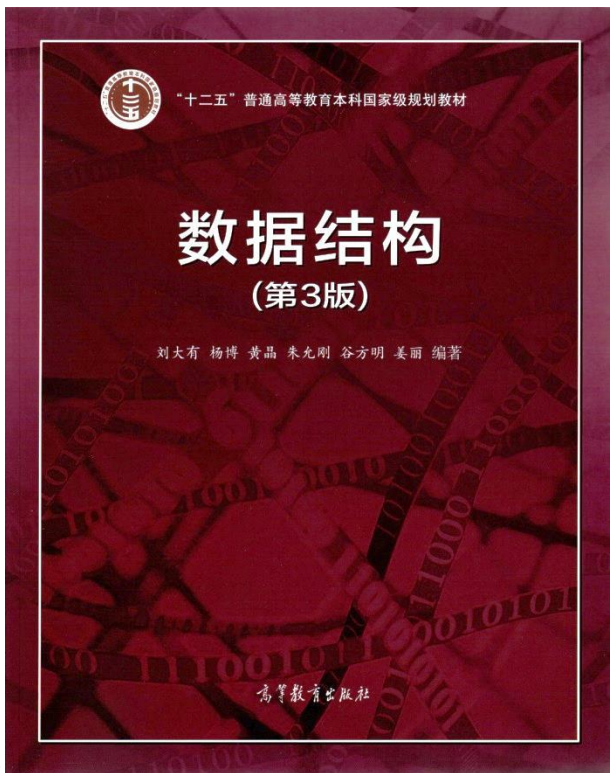
— Alan Turing

计算机科学之父
英国皇家科学院院士
普林斯顿大学博士
曼彻斯特大学教授
剑桥大学研究员



图的概念与存储结构

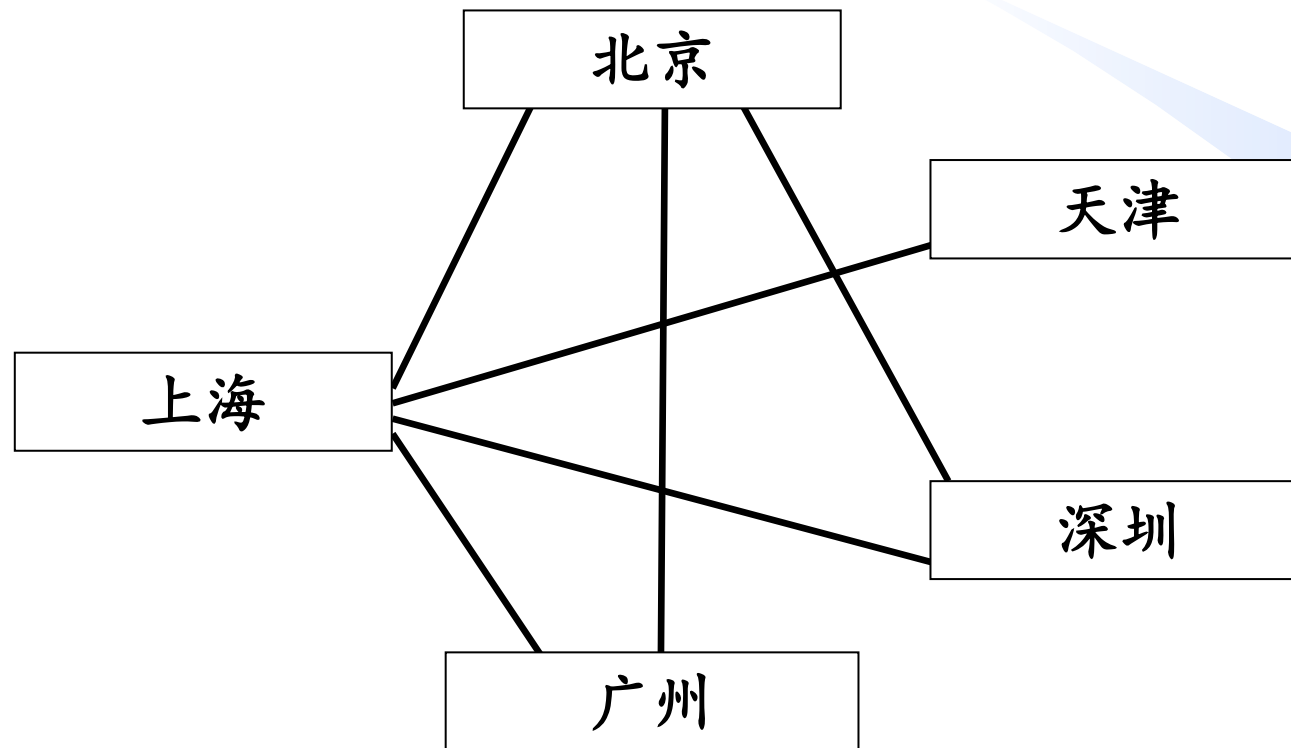
- 图的基本概念
- 图的存储结构



数据之法
结构之美
算法之道

- 图（Graph）是一种较线性表和树更为复杂的非线性结构。在图结构中，对结点（图中常称为顶点）的前驱和后继个数不加限制，即结点之间的关系是任意的。图中任意两个结点之间都可能相关。图状结构可以描述各种复杂的数据对象。
- 图的应用极为广泛，特别是近年来的迅速发展，已经渗透到诸如语言学、逻辑学、物理、化学、电讯工程、计算机科学以及数学的其它分支中。

城市航线网



计算机网络

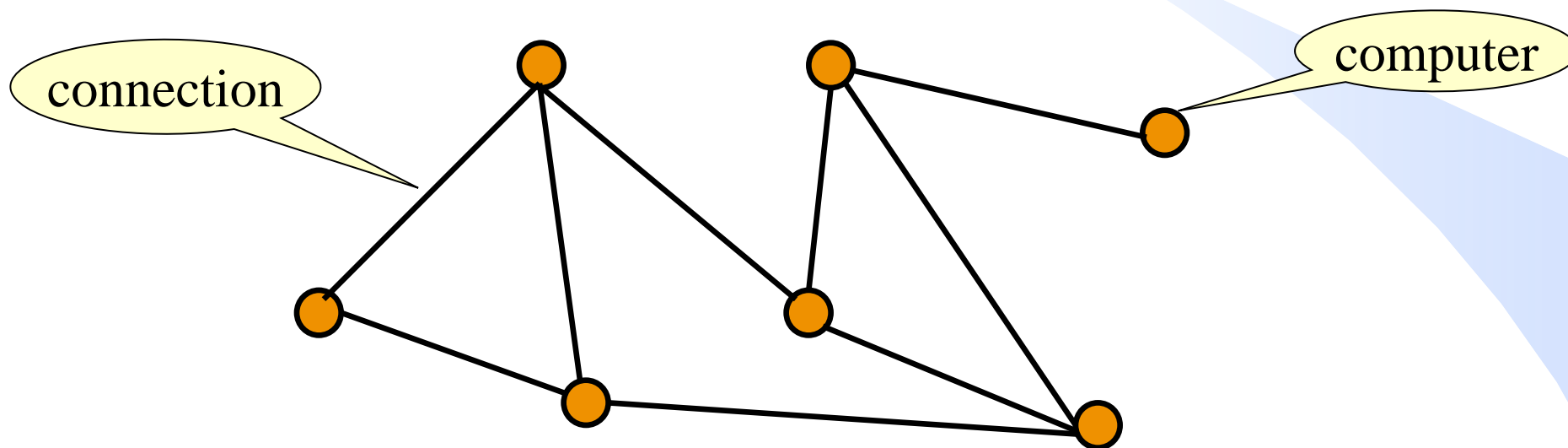


图 VS. 树

- 不一定具有一个根结点
- 没有明显的父子关系
- 从一个顶点到另一个顶点可能有多个（或0个）路径

图的基本概念

Vertex

Edge

定义 图 G 由两个集合 **V** 和 **E** 组成，记为 **$G = (V, E)$** ；其中 V 是顶点的有穷非空集合， E 是连接 V 中两个不同顶点的边的有穷集合。通常，也将图 G 的**顶点集**和**边集**分别记为 **$V(G)$** 和 **$E(G)$** 。

若图中的边限定为从一个顶点指向另一个顶点，则称此图为**有向图**。

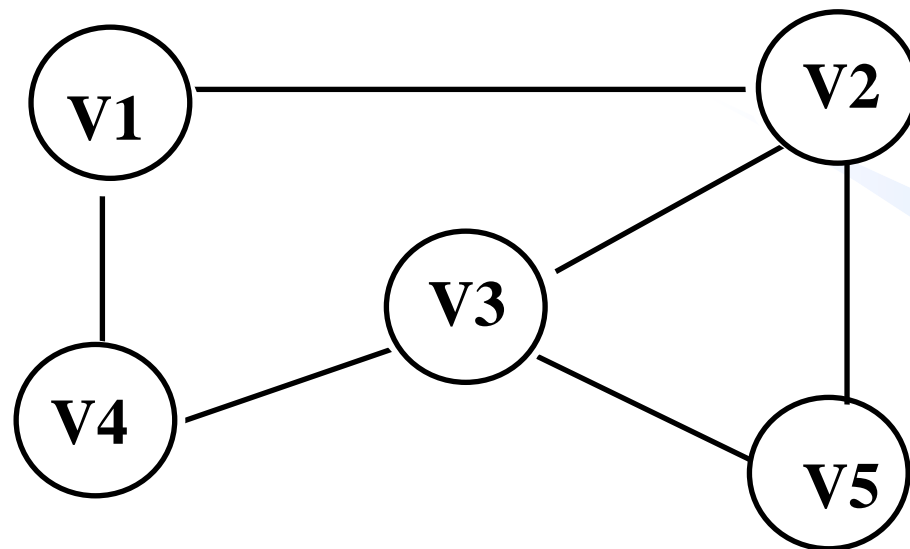
若图中的边无方向性，则称之为**无向图**。

有向边

定义 若 $G = (V, E)$ 是有向图，则它的一条有向边是由 V 中两个顶点构成的有序对，亦称为弧，记为 $\langle w, v \rangle$ ，其中 w 是边的始点，又称弧尾； v 是边的终点，又称弧头。



无向图

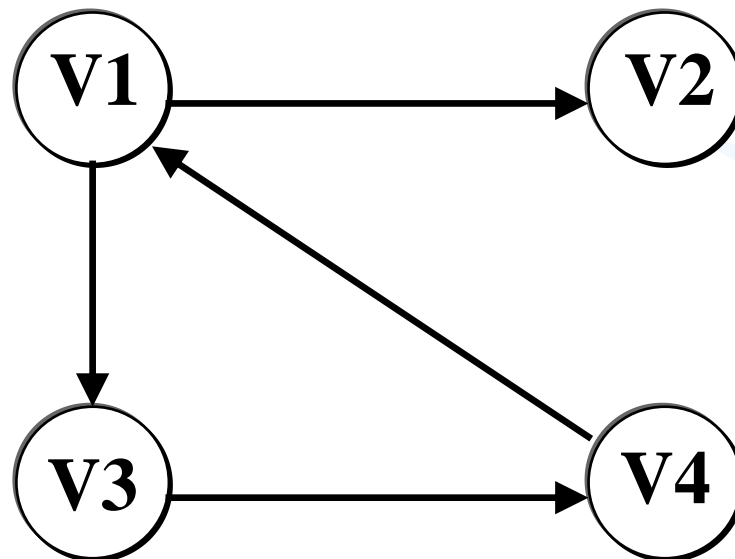


$G=(V, E)$

$V=\{V1, V2, V3, V4, V5\}$

$E=\{(V1, V4), (V1, V2), (V2, V3), (V2, V5),$
 $(V3, V4), (V3, V5)\}$

有向图



$G = (V, E)$

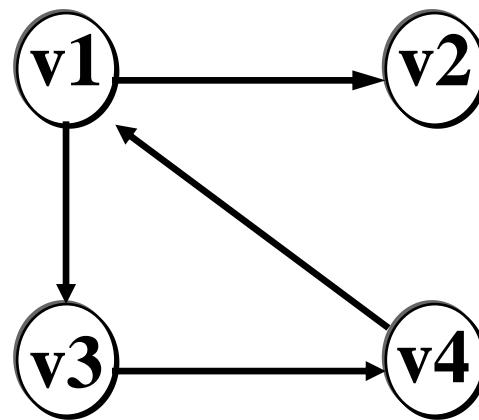
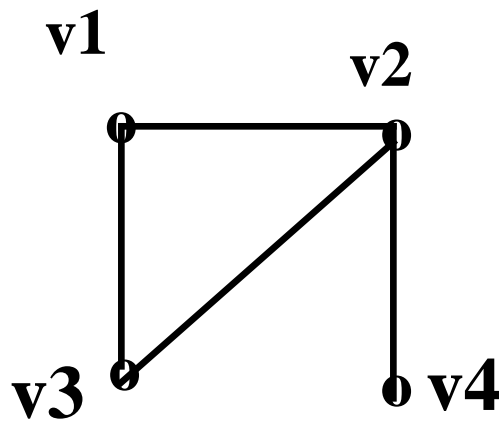
$V = \{V1, V2, V3, V4\}$

$E = \{ \langle V1, V2 \rangle, \langle V1, V3 \rangle, \langle V3, V4 \rangle, \langle V4, V1 \rangle \}$

邻接顶点

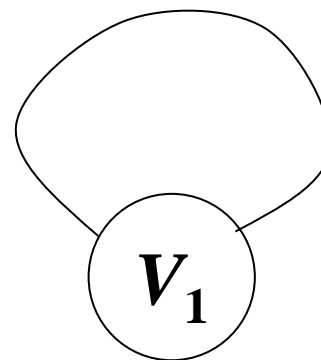
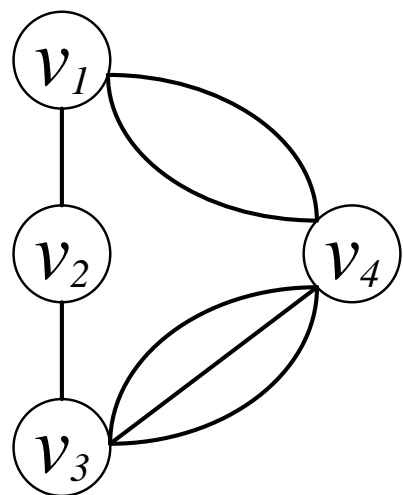
定义 在一个无向图中，若存在一条边 (w, v) ，则称 w, v 为此边的两个**端点**，它们是相邻的，并称它们互为**邻接顶点**。

在一个有向图中，若存在一条边 $\langle w, v \rangle$ ，则称顶点 w **邻接到**顶点 v ，顶点 v **邻接自**顶点 w 。



多重图和简单图

- 无向图中两个顶点之间不止有一条边，或者有向图中两个顶点之间不止有一条同方向的边，这类边称为**重边**。包含重边的图称为**多重图**。
- 一条边的两个端点是同一个顶点，则称为**自环**。
- 不含重边和自环的图称为**简单图**。
- 除非有特别说明，**本课程中的图都是简单图**。



很多问题都可以抽象成一个图结构，例如：

- 将多个城市构成顶点集 V ，如果城市 a 和城市 b 之间有一条高速公路，则在 a 和 b 之间连接一条边。允许在两个城市之间修建多条高速公路。按照这种方式建立的图是多重图。
- 社交网络中的好友关系，QQ、微信、微博、抖音等。

完全图

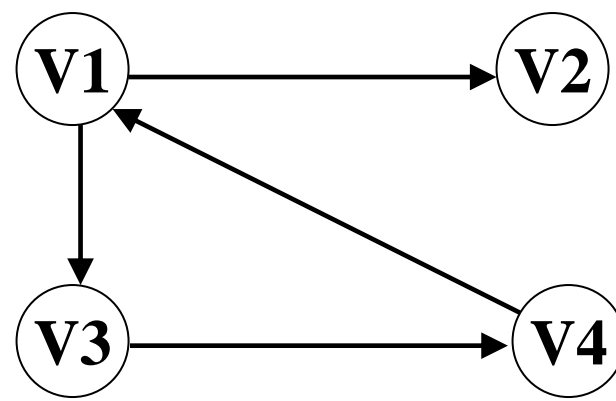
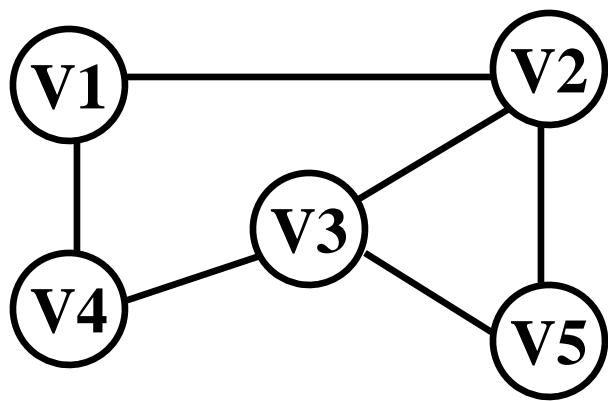
无向完全图：任意两个顶点之间都有一条边的无向图。包含 n 个顶点的无向完全图中，有 $n(n-1)/2$ 条边。

有向完全图：任意两个顶点之间都有方向相反的两条边。包含 n 个顶点的有向完全图中，有 $n(n-1)$ 条边。

对于 n 个顶点的完全图，边的条数 $e = O(n^2)$

顶点的度

- 设 G 是无向图，顶点 v 的度为以 v 为端点的边的条数。
- 若 G 是有向图，则 v 的出度为以 v 为始点的边的条数， v 的入度为以 v 为终点的边的条数，顶点的度=入度+出度。



顶点的度和边的关系

设图G（可以为有向或无向图）共有 n 个顶点和 e 条边，若顶点 v_i 的度为 $D(v_i)$ ，则

$$\sum_i D(v_i) = 2e$$

因为一条边关联两个顶点，而且使得这两个顶点的度数分别增加1。因此顶点的度数之和就是边的2倍。

已知顶点的度序列，即可求出边的条数。

例题

已知无向图G有16条边，其中度为4的顶点有3个，度为3的顶点有4个，其他顶点的度均小于3，则图G所含的顶点个数至少是_____。【考研题全国卷】

A. 10

B. 11

C. 13

D. 15

所有顶点度之和=2e，设顶点数为n

$$32 \leq 4*3 + 3*4 + 2*(n-7)$$

所有顶点度之和的最大值

$n \geq 11$, 故选B

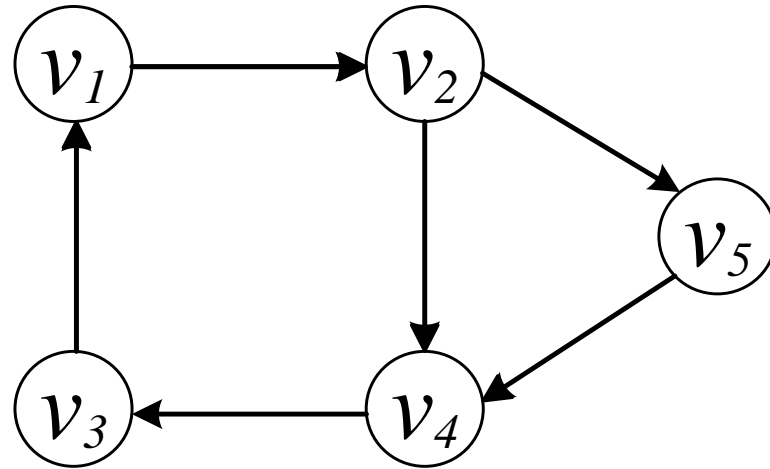
图的路径

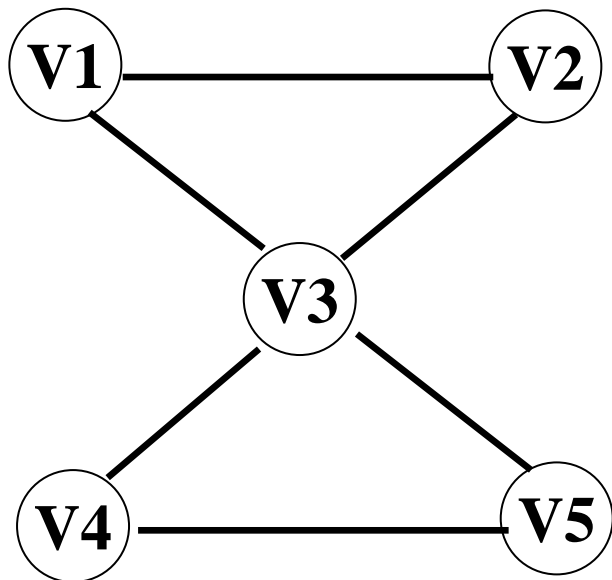
定义 设 G 是图，若存在一个顶点序列 $v_p, v_1, v_2, \dots, v_{q-1}, v_q$ 使得 $\langle v_p, v_1 \rangle, \langle v_1, v_2 \rangle, \dots, \langle v_{q-1}, v_q \rangle$ 或 $(v_p, v_1), (v_1, v_2), \dots, (v_{q-1}, v_q)$ 属于 $E(G)$ ，则称 v_p 到 v_q 存在一条路径，其中 v_p 称为起点， v_q 称为终点。

路径的长度是该路径上边的个数。

- 如果一条路径上除了起点和终点可以相同外，再不能有相同的顶点，则称此路径为简单路径。
- 如果一条简单路径的起点和终点相同，且路径长度大于等于2，则称之为简单回路。

- 下图中， v_1 到 v_3 之间存在一条路径 v_1, v_2, v_5, v_4, v_3 ，同时这也是一条简单路径； $v_1, v_2, v_5, v_4, v_3, v_1$ 是一条简单回路。

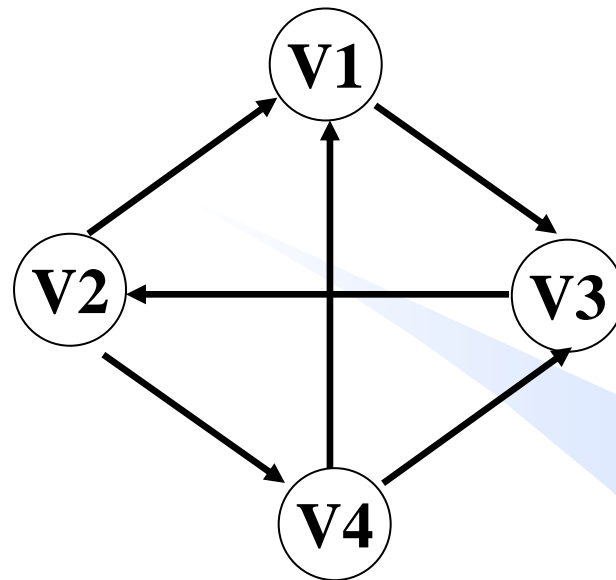




路径: $v_1 v_3 v_4 v_3 v_5$

简单路径: $v_1 v_3 v_5$

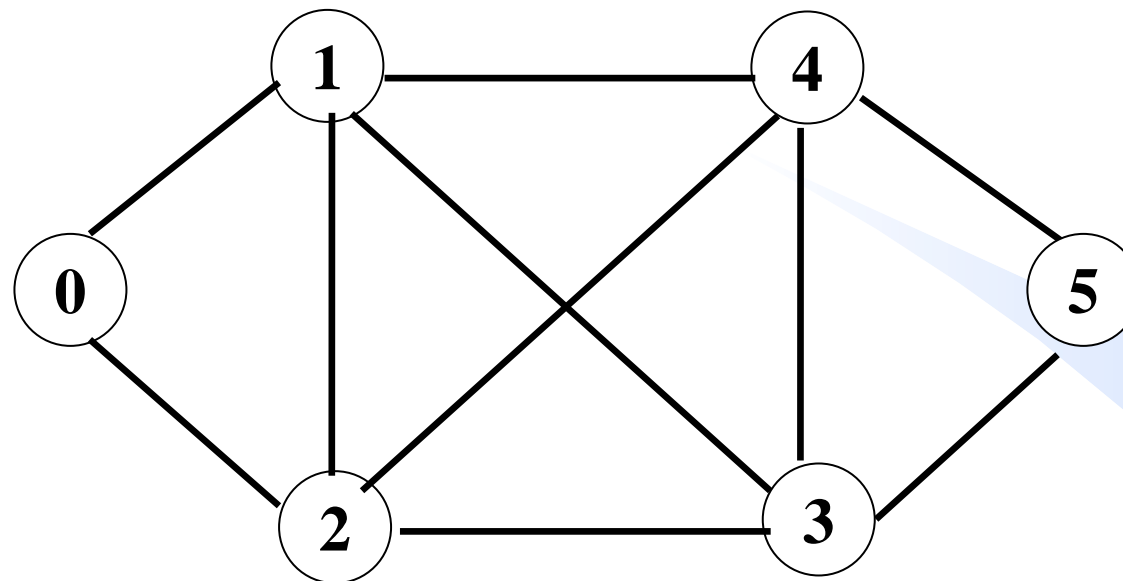
简单回路: $v_1 v_2 v_3 v_1$



路径: $v_1 v_3 v_2 v_4 v_3 v_2$

简单路径: $v_1 v_3 v_2$

简单回路: $v_1 v_3 v_2 v_1$



欧拉回路（一笔画）

每条边经过一次

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

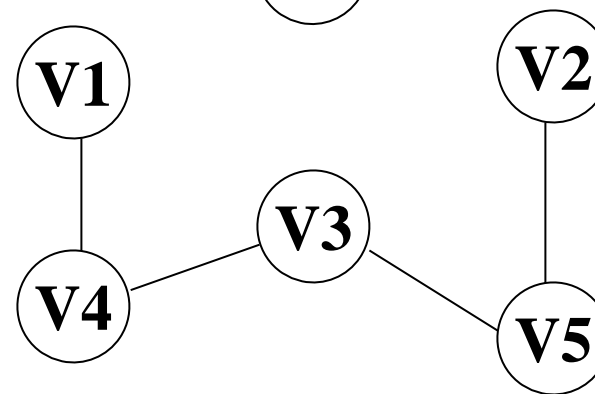
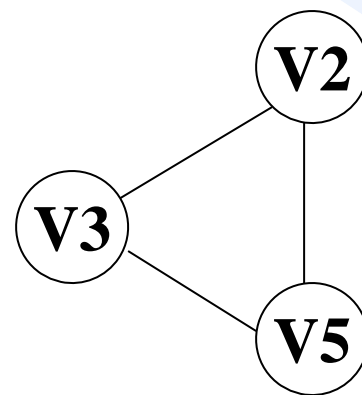
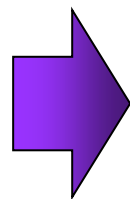
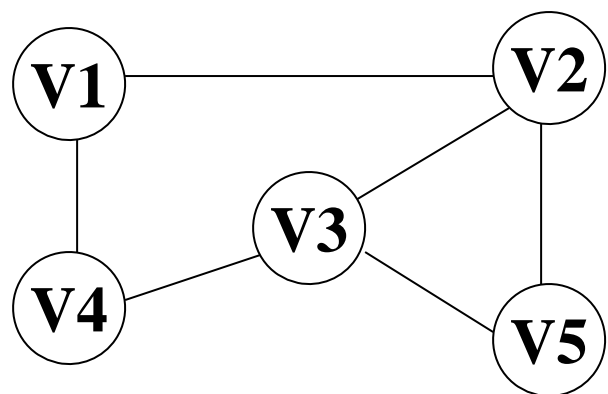
汉密尔顿回路

每个顶点经过一次

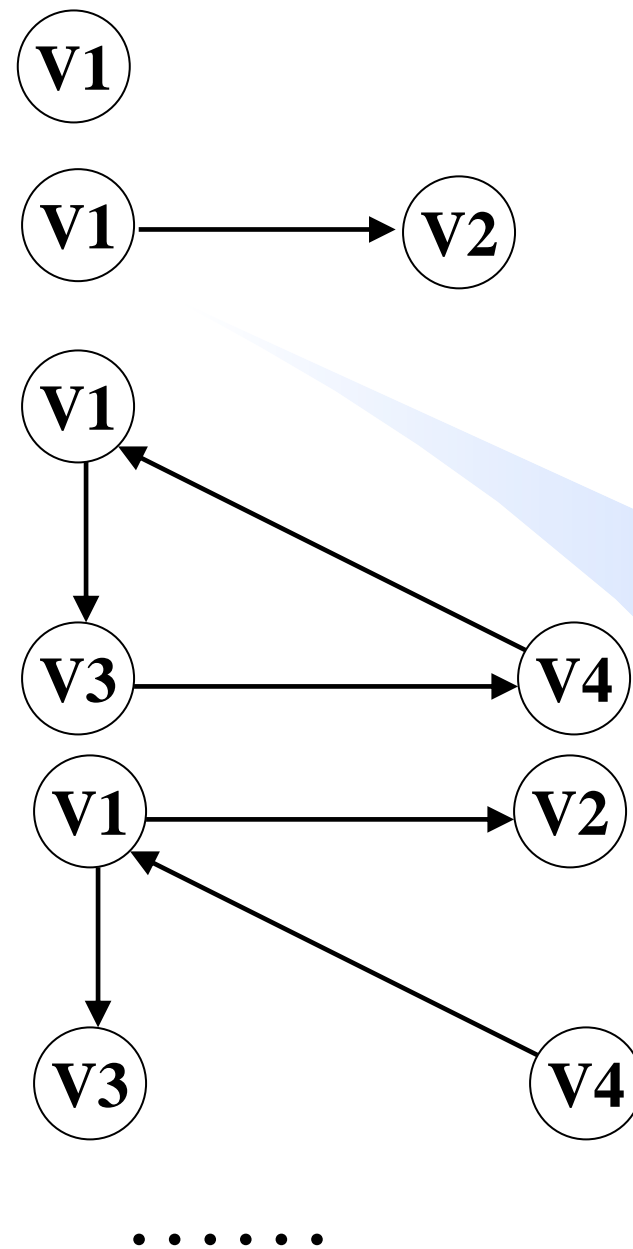
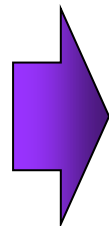
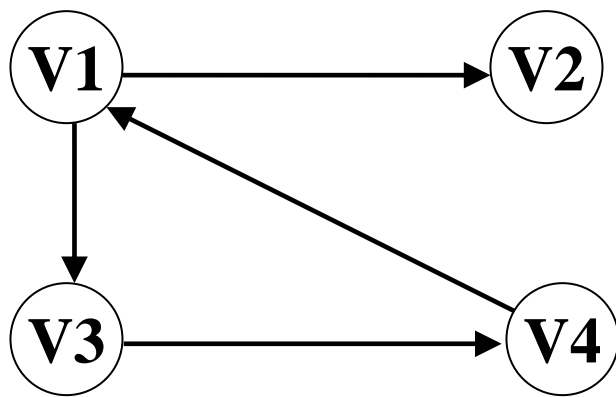
5-4-1-0-2-3-5

定义 设 G, H 是图, 如果 $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$, 则称 H 是 G 的子图, G 是 H 的母图。如果 H 是 G 的子图, 并且 $V(H) = V(G)$, 则称 H 为 G 的支撑子图。

图 G 的子图就是从 G 中抽取一部分顶点或边构成的图。

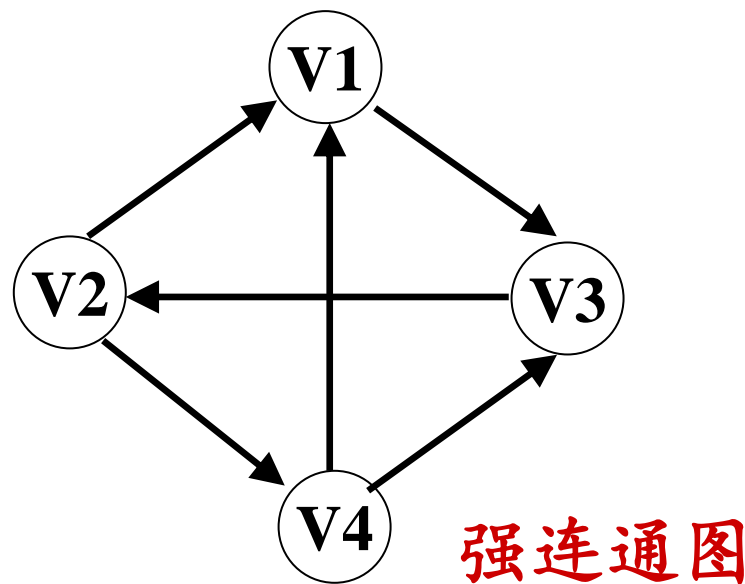
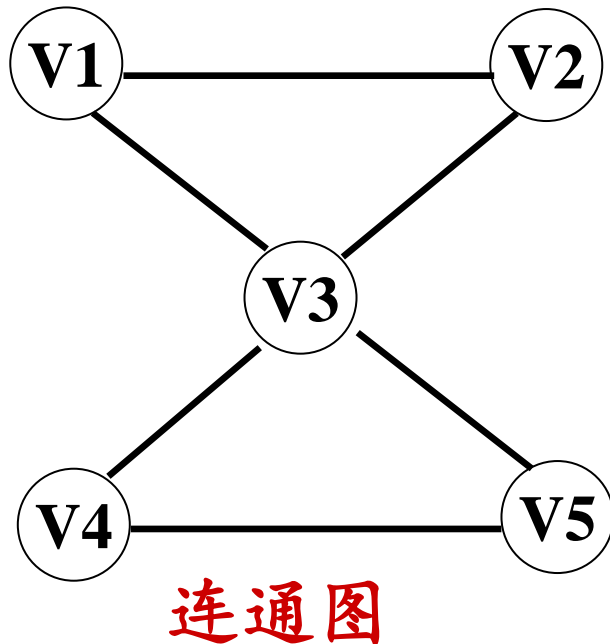


.....



连通性

- **定义** 设 G 是图，若存在一条从顶点 v_i 到顶点 v_j 的路径，则称 v_i 与 v_j **可及（连通）**。
- 若 G 为无向图，且图中任意两个顶点都可及，则称 G 为 **连通图**。
- 若 G 为有向图，且对于图中任意两个不同的顶点 v_i 和 v_j ， v_i 与 v_j 可及， v_j 与 v_i 也可及，则称 G 为 **强连通图**



例题

- 具有 n 个顶点的无向图，当至少有____条边时可**确保**它一定是一个连通图。【吉林大学考研题】

答： $n-1$ 个顶点的完全图再加1条边， C_{n-1}^2+1 ，
故 $(n-1)(n-2)/2+1$

- 无向图G包含7个顶点，要**保证G在任何情况下**都是连通的，则需要边数最少是____条。【考研题全国卷】

A. 6 B. 15 C. 16 D. 21

例题

G是一个具有36条边的无向非连通图，则G的顶点数至少为_____。【北京大学、吉林大学考研题】

A. 12 B. 11 C. 10 D. 9

无向图有 n 个顶点，则边的条数 $\leq C_n^2 = n(n-1)/2$,

即 $36 \leq n(n-1)/2$ ，即 $72 \leq n(n-1)$ ，故 $n \geq 9$ 。

对于连通图， n 至少为9，对于非连通图， n 至少为10。

故选C

课下思考

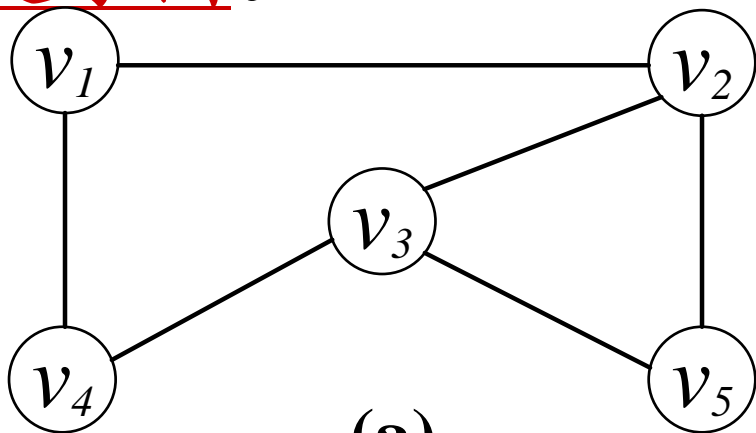
对于无向图 $G = (V, E)$ ，下列选项中正确的是_____。

【2022年考研题全国卷】

- A. 当 $|V| > |E|$ 时， G 一定是连通的
- B. 当 $|V| < |E|$ 时， G 一定是连通的
- C. 当 $|V| = |E| - 1$ 时， G 一定是不连通的
- ✓ D. 当 $|V| > |E| + 1$ 时， G 一定是不连通的

连通子图

设图 G 是**无向图**，若 G 的子图 G_K 是一个**连通图**，则称 G_K 为 G 的**连通子图**。



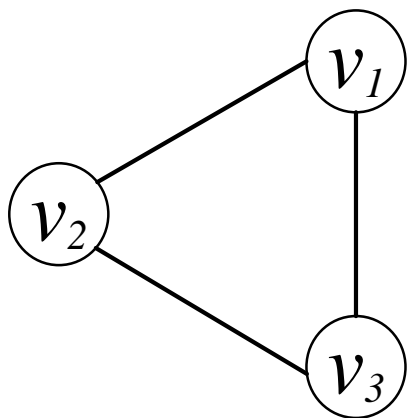
(a)



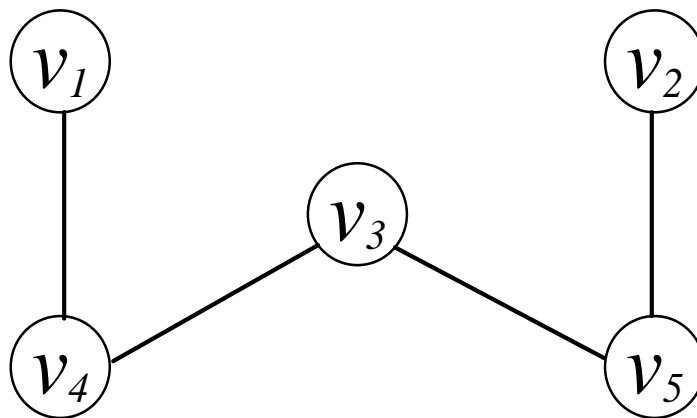
(b)



(c)



(d)



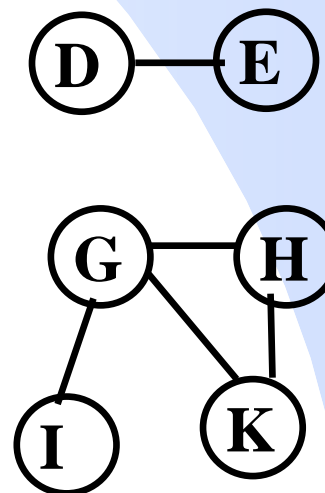
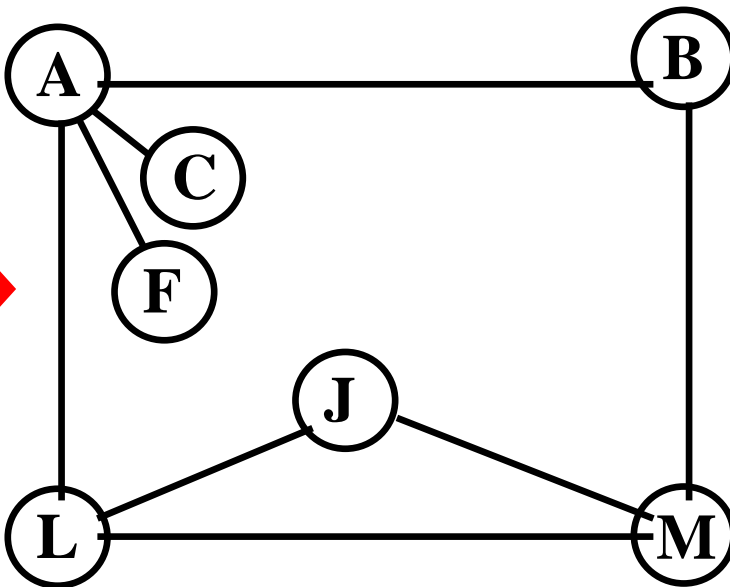
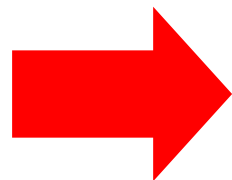
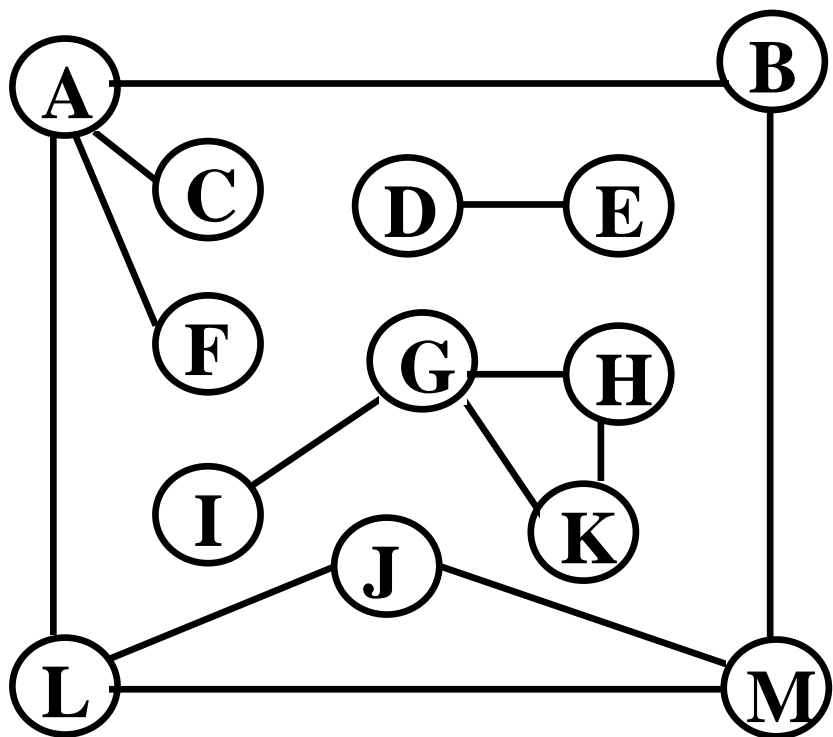
(e)

一个图的连通子图
不一定唯一

连通分量

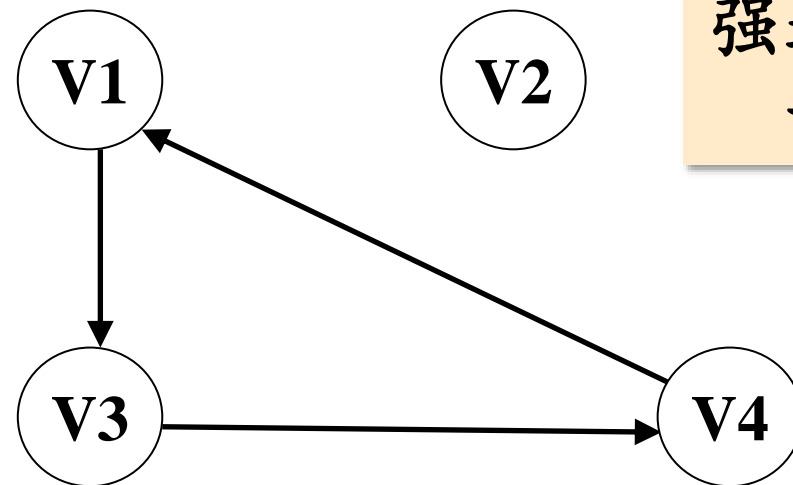
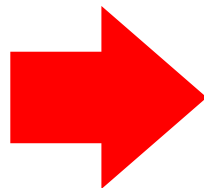
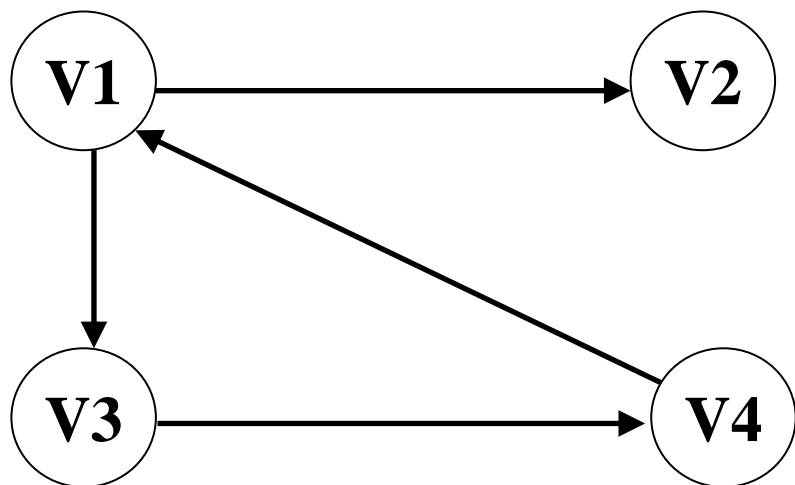
- 无向图 G 的一个连通子图 G_K ，如果不存在 G 的另一个连通子图 G' ，使得 $V(G_K) \subset V(G')$ 或 $E(G_K) \subset E(G')$ ，则称 G_K 为 G 的 连通分量。
- 无向图 G 的 连通分量 即为 G 的极大连通子图。

一个无向图的连通分量不一定唯一



强连通子图和强连通分量

- 设图 G 是有向图，若 G 的子图 G_K 是一个强连通图，则称 G_K 为 G 的强连通子图。
- 如果不存在 G 的另一个强连通子图 G' ，使得 $V(G_K) \subset V(G')$ 或 $E(G_K) \subset E(G')$ ，则称 G_K 是 G 的强连通分量。
- G 的强连通分量即为 G 的极大强连通子图。



一个有向图的
强连通分量不
一定唯一

带权图

设 $G = (V, E)$ 是图，若对图中的任意一条边 l ，都有实数 $w(l)$ 与其对应，则称 G 为 权图，记为 $G = (V, E, w)$ 。记 $w(u, v)$ 为以 u, v 为端点的边的权值，规定：

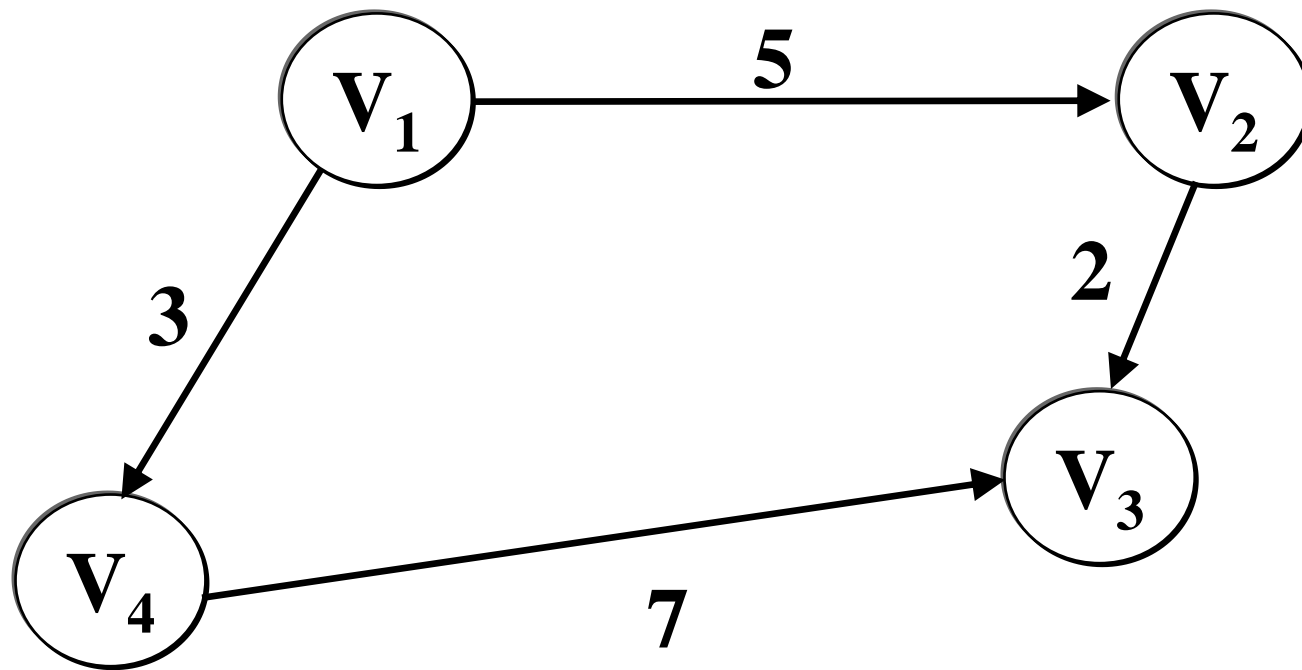
$\forall u \in V$, 有 $w(u, u) = 0$

$\forall u, v \in V$, 若顶点 u 和 v 之间不存在边，则 $w(u, v) = \infty$

权值通常用于表示一个顶点到另一个顶点的距离或费用。

带权路径

➤ 若 $\sigma = (v_0, v_1, v_2, \dots, v_k)$ 是权图 G 中的一条路径，则路径所包含的边的权值之和 $|\sigma| = \sum_{i=1}^k w(v_{i-1}, v_i)$ 称为路径 σ 的长度。

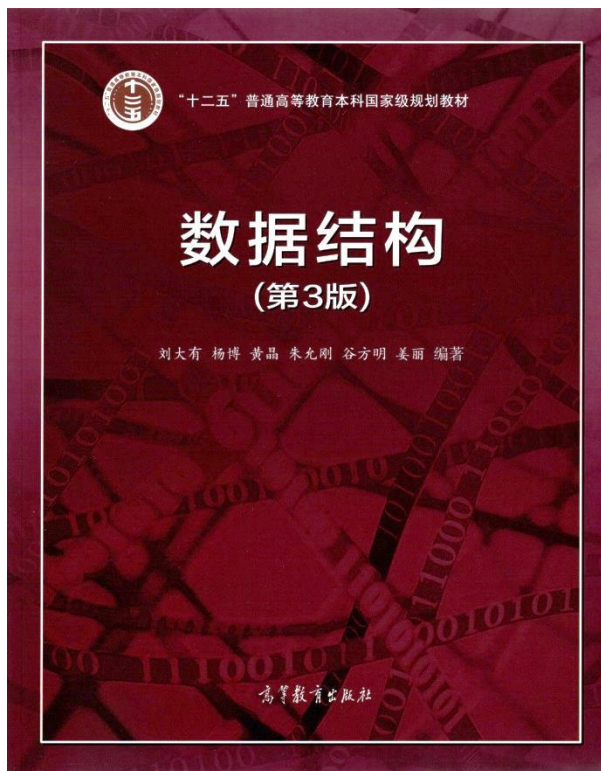


无向图	有向图
无向边	有向边（弧）
端点	弧头 弧尾
相邻的	邻接到 邻接自
度	出度 入度
连通图	强连通图
连通子图	强连通子图
连通分量	强连通分量



图的概念与存储结构

- 图的基本概念
- 图的存储结构



数据之法
结构之美
算法之道

zhuyungang@jlu.edu.cn

用**顺序方式**存储图的顶点表 v_0, v_1, \dots, v_{n-1} , 图的边用 $n \times n$ 阶矩阵 $A=(a_{ij})$ 表示, A 的定义如下:

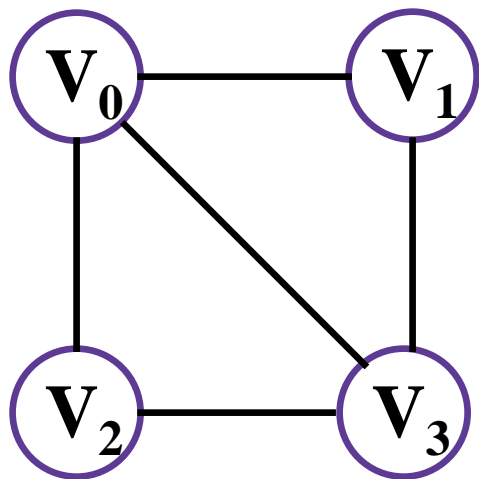
(1)若非权图, 则:

- $a_{ii} = 0$;
- $a_{ij} = 1$, 当 $i \neq j$ 且 v_i 与 v_j 之间存在边;
- $a_{ij} = 0$, 当 $i \neq j$ 且 v_i 与 v_j 之间不存在边。

(2)若权图, 则:

- $a_{ii} = 0$;
- a_{ij} = 边的权值, 当 $i \neq j$ 且 v_i 与 v_j 之间存在边;
- $a_{ij} = \infty$, 当 $i \neq j$ 且 v_i 与 v_j 之间不存在边。

[例]无向图的邻接矩阵

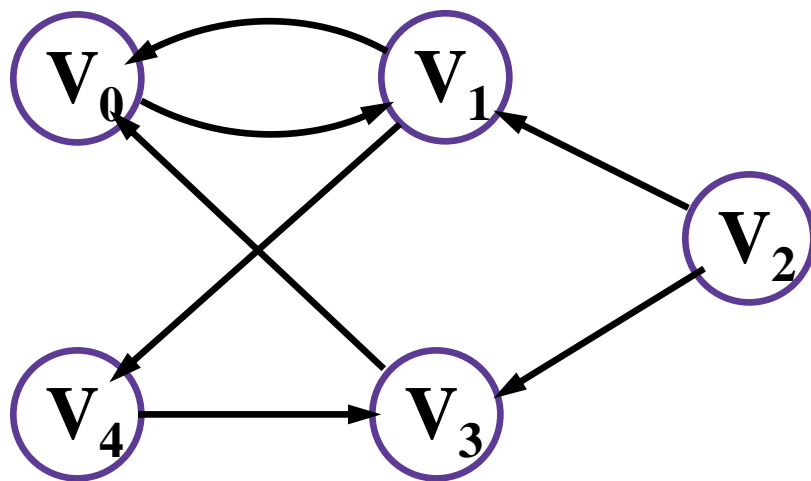


A

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

无向图的邻接矩阵是**对称矩阵**。

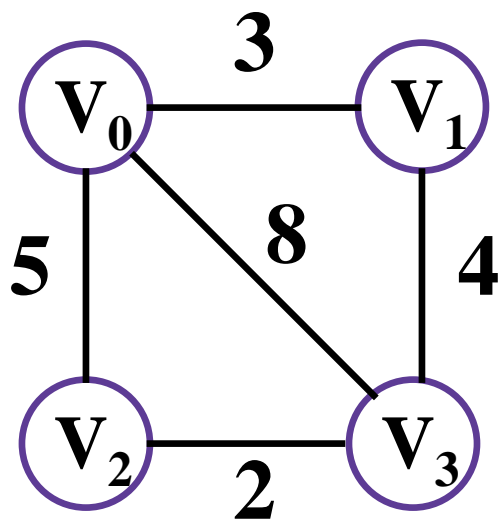
[例]有向图的邻接矩阵



A

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

[例]权图的邻接矩阵



A

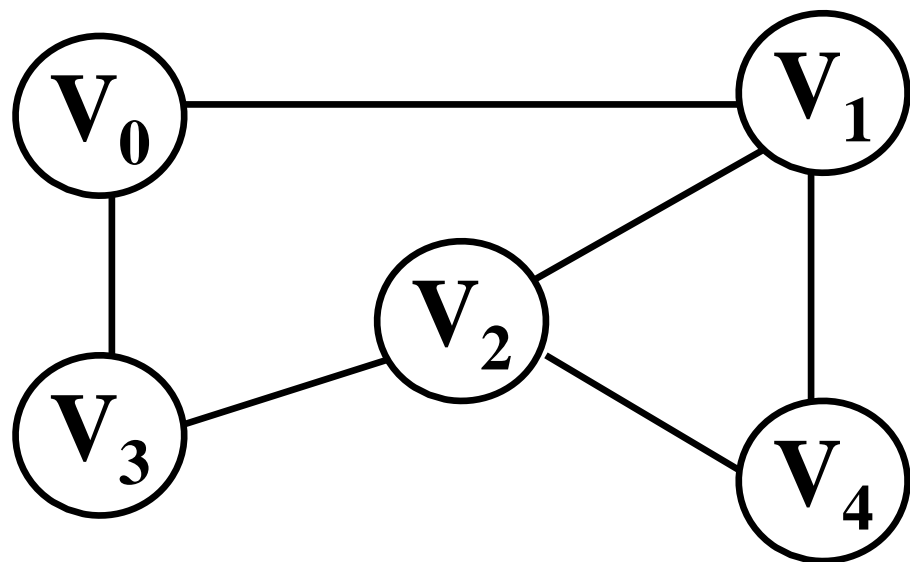
	0	1	2	3
0	0	3	5	8
1	3	0	∞	4
2	5	∞	0	2
3	8	4	2	0

特点：无向图的邻接矩阵是对称矩阵，有向图邻接矩阵不一定对称。

基于邻接矩阵求图中顶点的度

无向无权图 矩阵的第 i 行（或第 i 列）的1的个数是**顶点 V_i 的度**。

第 i 行有一个1就意味着有一条以顶点 V_i 为端点的边



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

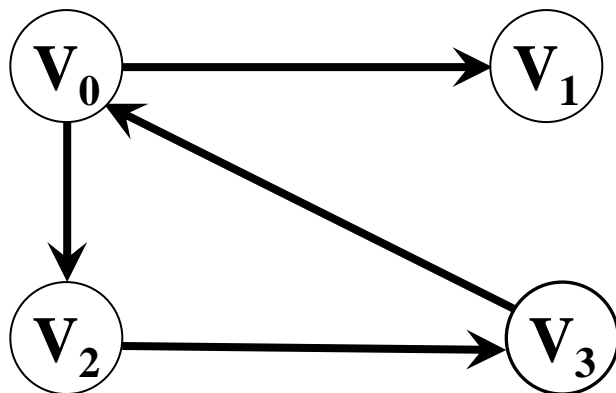
基于邻接矩阵求图中顶点的度

有向无权图邻接矩阵第 i 行的1的个数为顶点 V_i 的出度

第 i 行有一个1，就意味着有一条由 V_i 引出的边

- 第 i 列的1的个数为顶点 V_i 的入度。

第 i 列有一个1，就意味着有一条引入（指向） V_i 的边



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

课下思考

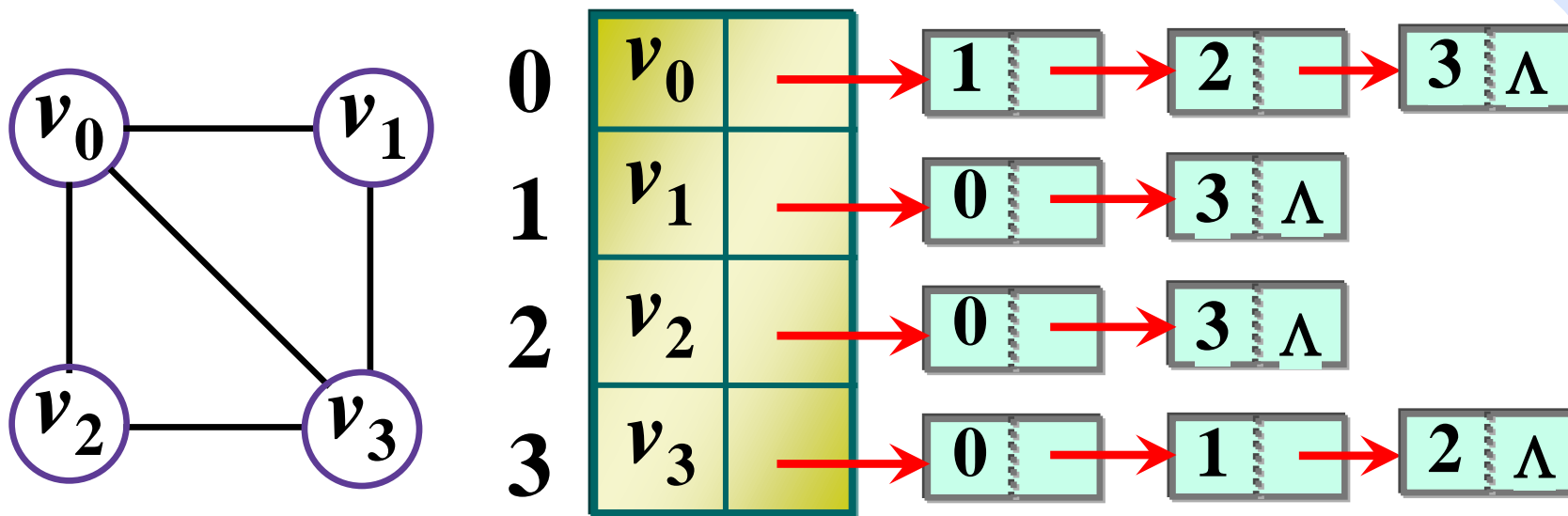
已知无向连通图 G 由顶点集 V 和边集 E 组成， $|E|>0$ ，当 G 中**度为奇数的顶点个数为0或2时**， G 存在包含所有边且长度为 $|E|$ 的路径（称为EL路径），设 G 采用**邻接矩阵**存储，请设计算法，判断 G 是否存在EL路径。【2021年考研题全国卷】

邻接矩阵

- 空间： $O(n^2)$
- 判断图中是否包含某条边(V_i, V_j): $O(1)$
- 找某个点的邻接顶点： $O(n)$
- 缺点：存储稀疏图（点多边少），邻接矩阵为稀疏矩阵，浪费空间。

2、邻接表

- 顺序存储顶点表。
- 对图的每个顶点建立一个单链表，第 i 个单链表中的结点包含顶点 v_i 的所有邻接顶点（边链表）。
- 由顺序存储的顶点表和链接存储的边链表构成的图存储结构被称为邻接表。



➤与顶点 v 邻接的所有顶点以某种次序组成的单链表称为顶点 v 的边链表。

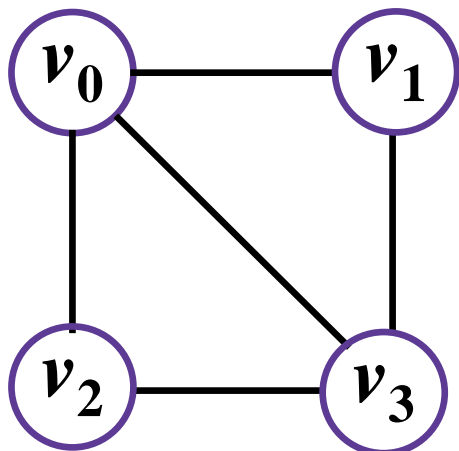
➤边链表的每一个结点叫做边结点，对于非权图和权图边结点结构分别为：

<i>VerAdj</i>	<i>link</i>
---------------	-------------

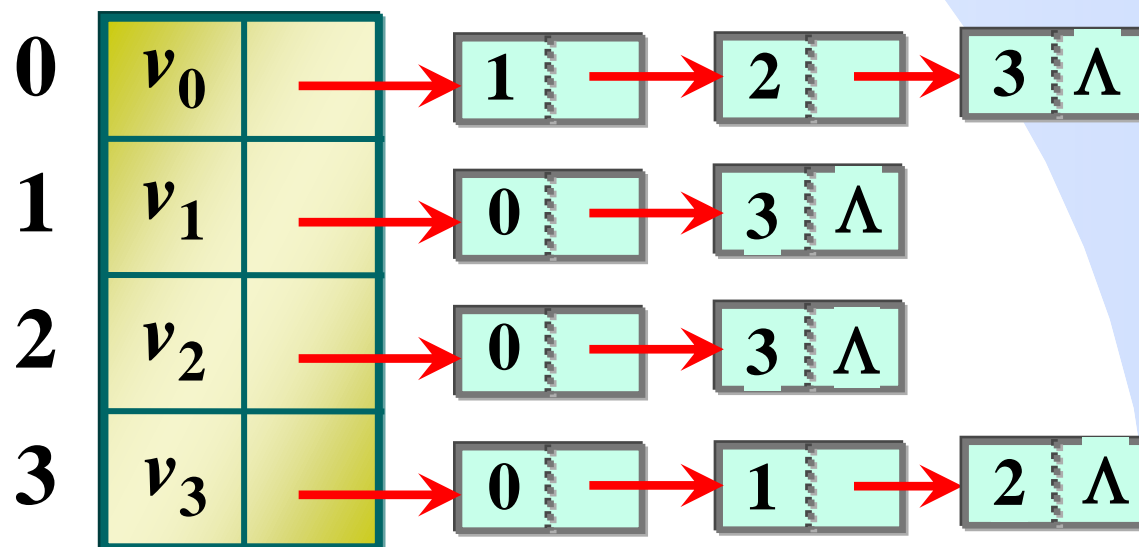
<i>VerAdj</i>	<i>cost</i>	<i>link</i>
---------------	-------------	-------------

其中，域 *VerAdj* 存放 v 的某个邻接顶点在顶点表中的下标；域 *link* 存放指向 v 的边链表中结点 *VerAdj* 的下一个结点的指针。
域 *cost* 存放边 (v, VerAdj) 或 $\langle v, \text{VerAdj} \rangle$ 的权值；

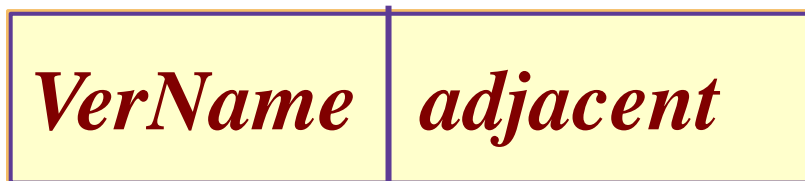
<i>VerName</i>	<i>adjacent</i>
----------------	-----------------



顶点表



顶点的结构



非权图中边结点结构为



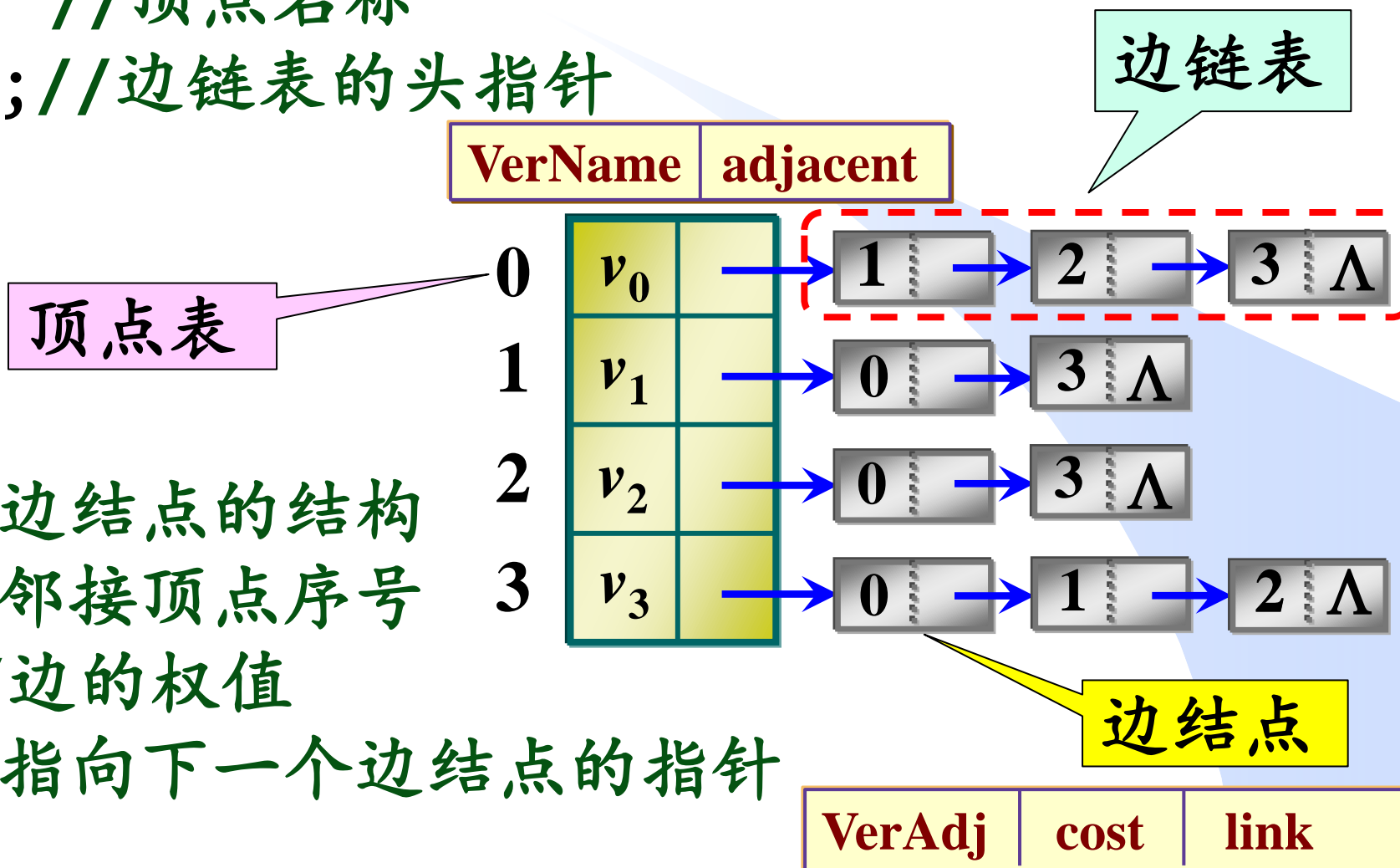
权图中边结点结构为



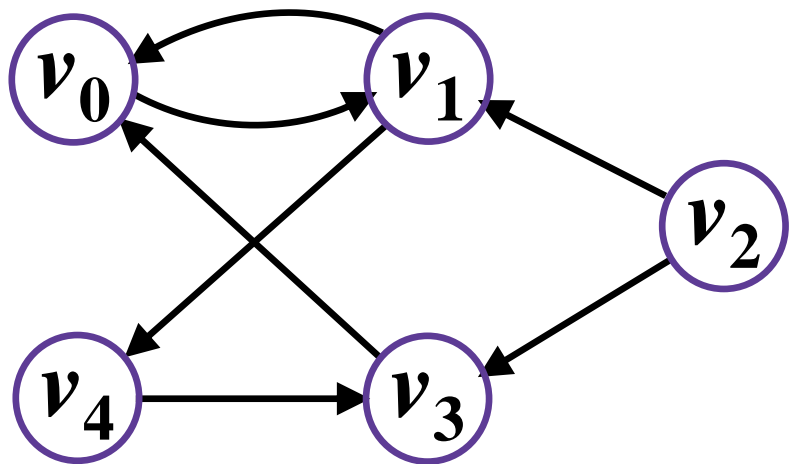
用邻接表存储图

```
struct Vertex{           //顶点表中结点的结构
    int VerName;         //顶点名称
    Edge *adjacent;      //边链表的头指针
};
```

```
struct Edge{             //边结点的结构
    int VerAdj;           //邻接顶点序号
    int cost;             //边的权值
    Edge *link;           //指向下一个边结点的指针
};
```

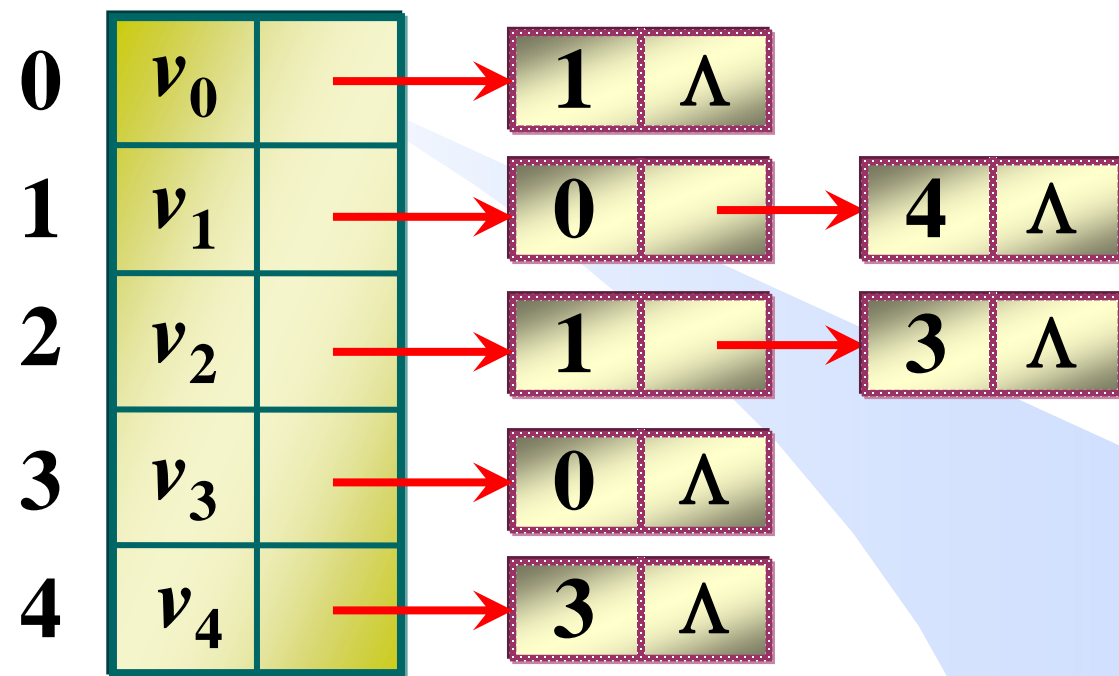


[例]有向图的邻接表



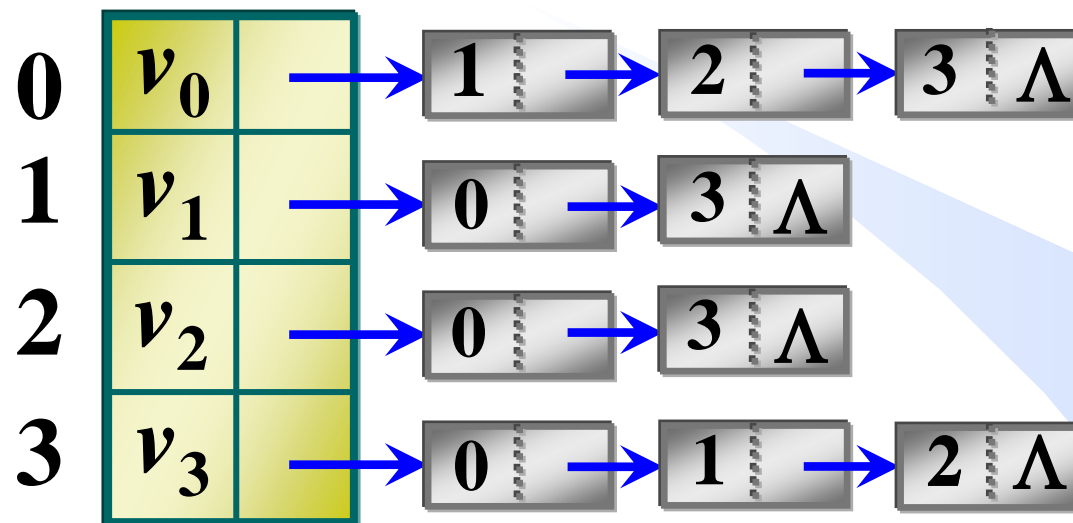
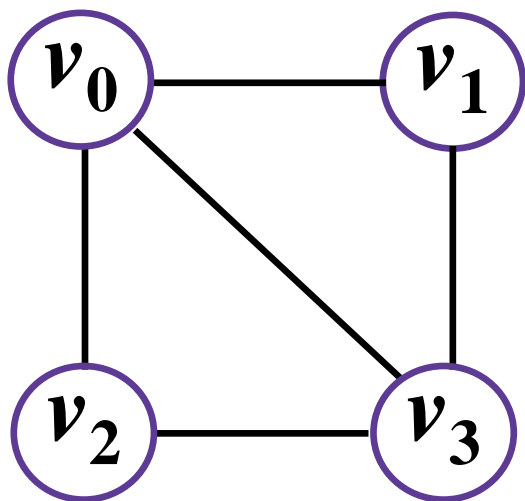
边结点的个数 = e
 e 为图中边的条数

1个边结点 \Leftrightarrow 1条边



占用空间
 $O(n+e)$

[例]无向图的邻接表

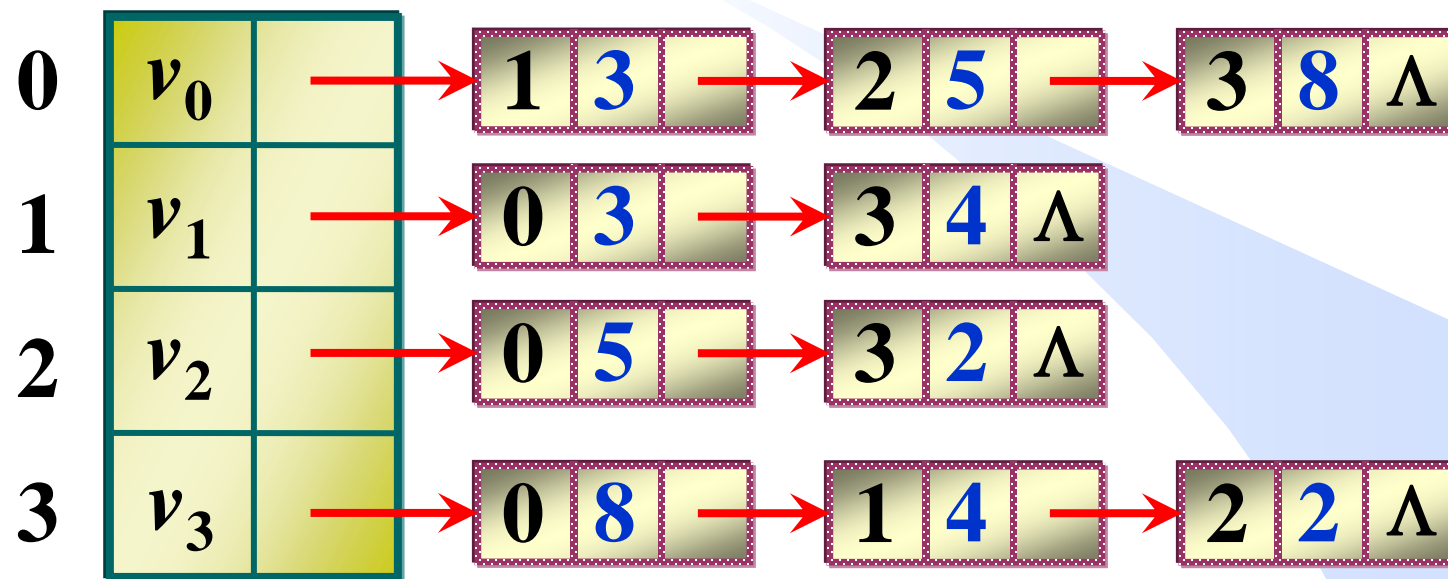
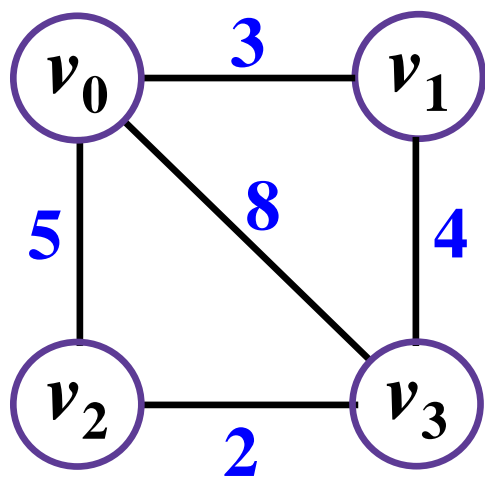


边结点的个数 = $2e$
 e 为图中边的条数

占用空间
 $O(n+e)$

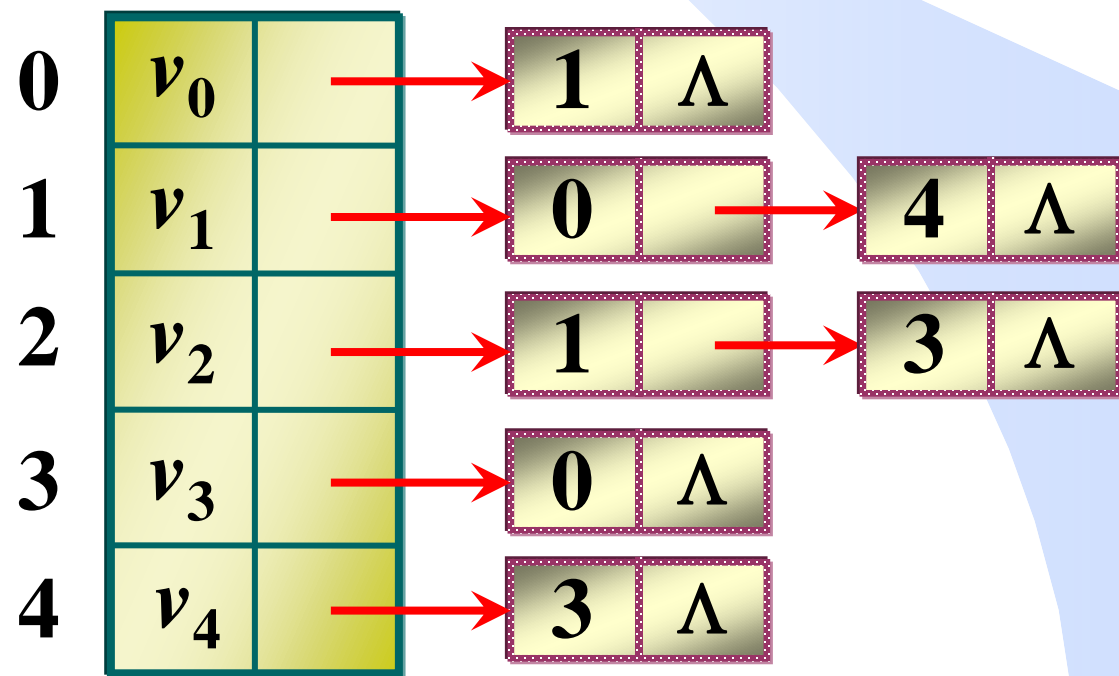
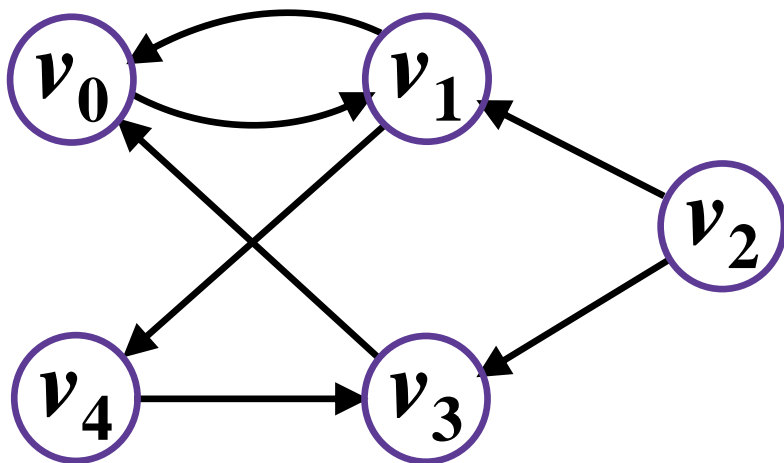
2个边结点 \Leftrightarrow 1条边

[例] 带权图的邻接表



邻接表中统计结点的度

- 根据邻接表，可统计出有向图中每个顶点的**出度**。
- 但是，如果要统计一个顶点的入度，就要遍历所有的边结点，其时间复杂度为 $O(e)$ （ e 为图中边的个数），
- 统计所有顶点入度的需要 $O(ne)$ ？（ n 为图的顶点个数）
- 只需 $O(n+e)$



VerName	adjacent
---------	----------

统计结点入度

VerAdj	link
--------	------

A

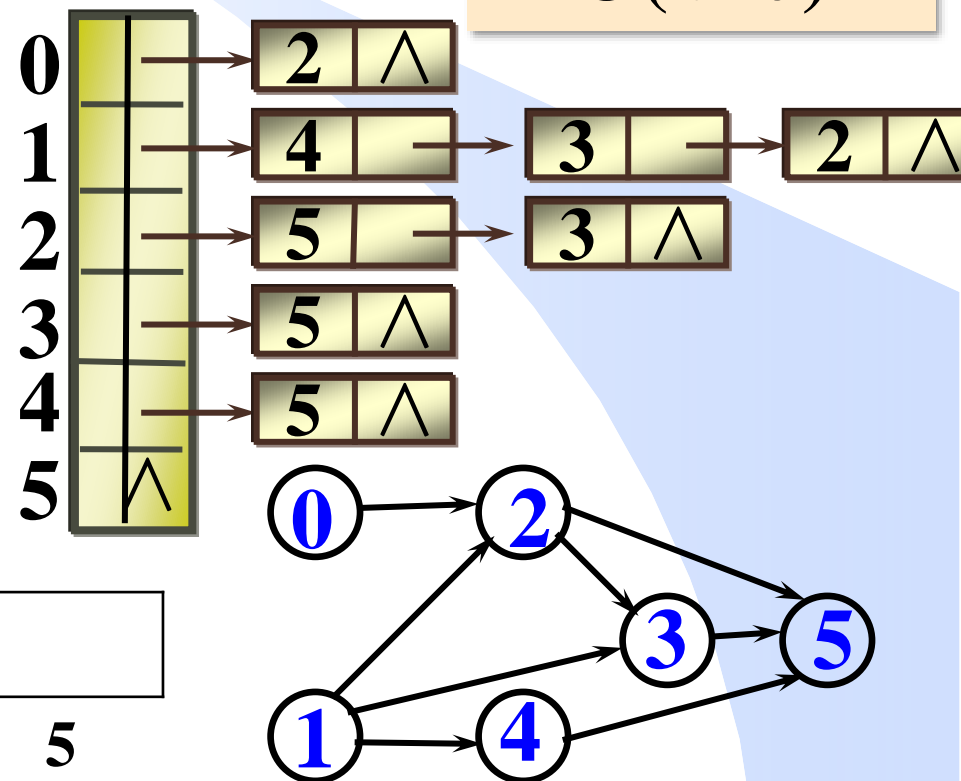
```
void getInDegree(Vertex Head[], int n, int InDegree[]) {  
    for(int i=0; i<n; i++) InDegree[i]=0;  
    for(int i=0; i<n; i++){ //用i扫描每个顶点  
        Edge* p=Head[i].adjacent;  
        while(p!=NULL){  
            int k = p->VerAdj;  
            InDegree[k]++;  
            p = p->link;  
        }  
    }  
}
```

InDegree

0	1	2	3	4	5

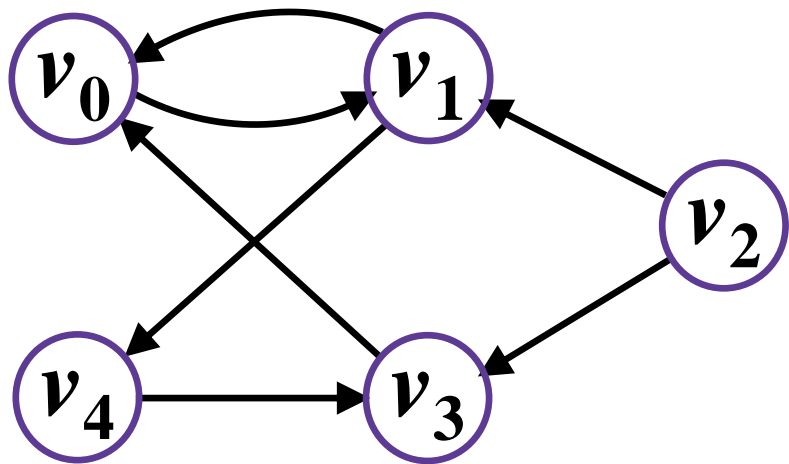
时间复杂度
 $O(n+e)$

用 p 扫描
每个顶点
对应的边
结点 (邻
接顶点)



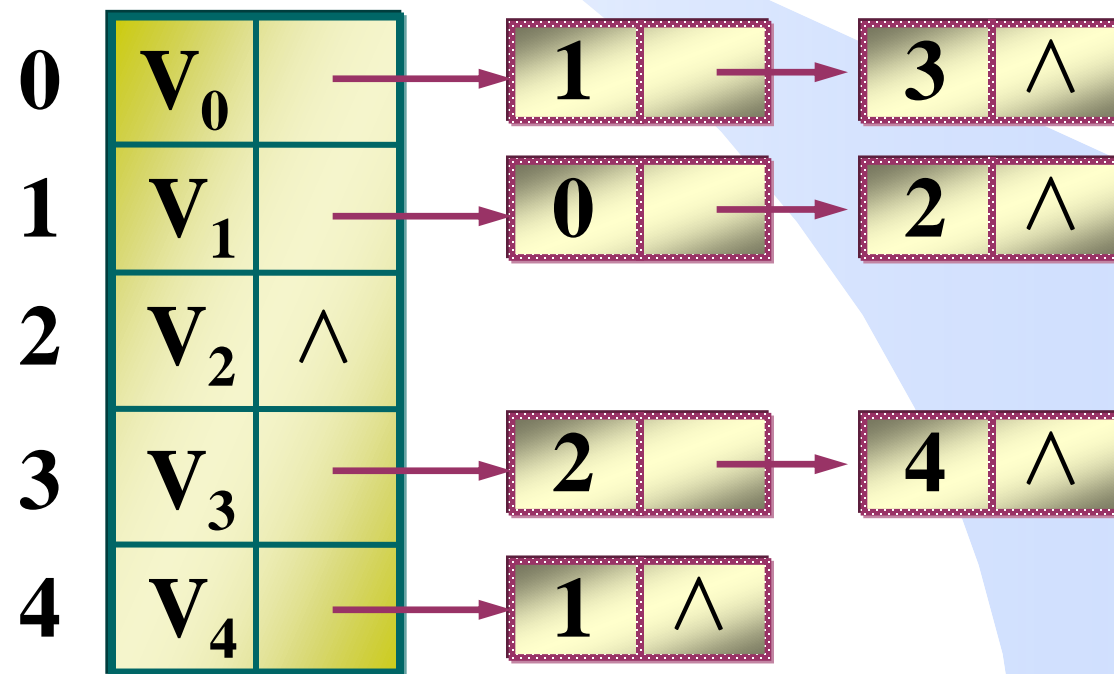
```
for(Edge* p=Head[i].adjacent; p!=NULL; p=p->link )
```

有向图的逆邻接表



对**有向图**建立逆邻接表（顶点的指向关系与邻接表恰好相反），根据逆邻接表，很容易统计出图中每个顶点的入度。

在有向图的**逆邻接表**中，第 i 个边链表链接的边都是指向**顶点 i 的边**。



邻接矩阵 vs. 邻接链表

	邻接矩阵	邻接表
检测是否存在边(V_i, V_j)	$O(1)$	$O(d_i)$
改删边(V_i, V_j)	$O(1)$	$O(d_i)$
找顶点 V_i 的邻接顶点	$O(n)$	$O(d_i)$
占用空间	$O(n^2)$	$O(n+e)$
适用于	稠密图	稀疏图
	经常查改删边	经常找邻接顶点

n 为顶点个数; e 为边的条数; d_i 为顶点 V_i 的度, $d_i \leq n$