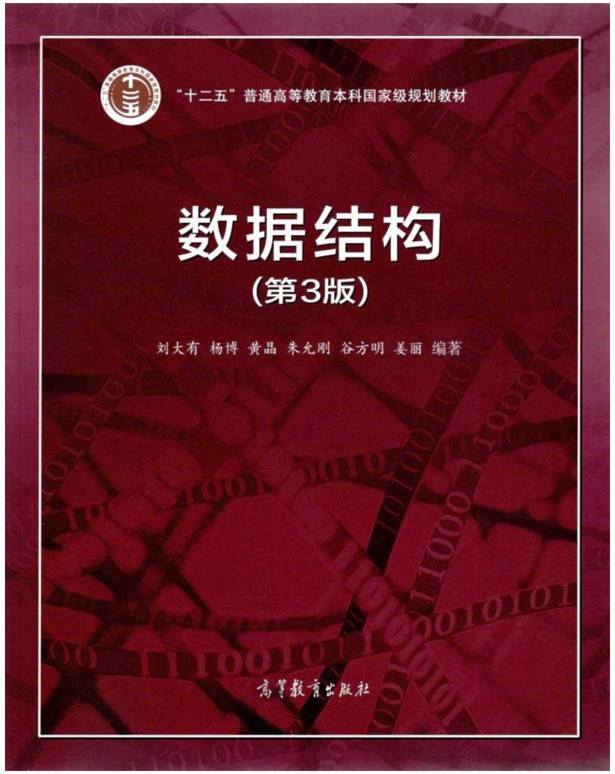
计算机学院王湘浩班

2024级



图的概念与存储结构

- 基本概念
- 图的存储结构

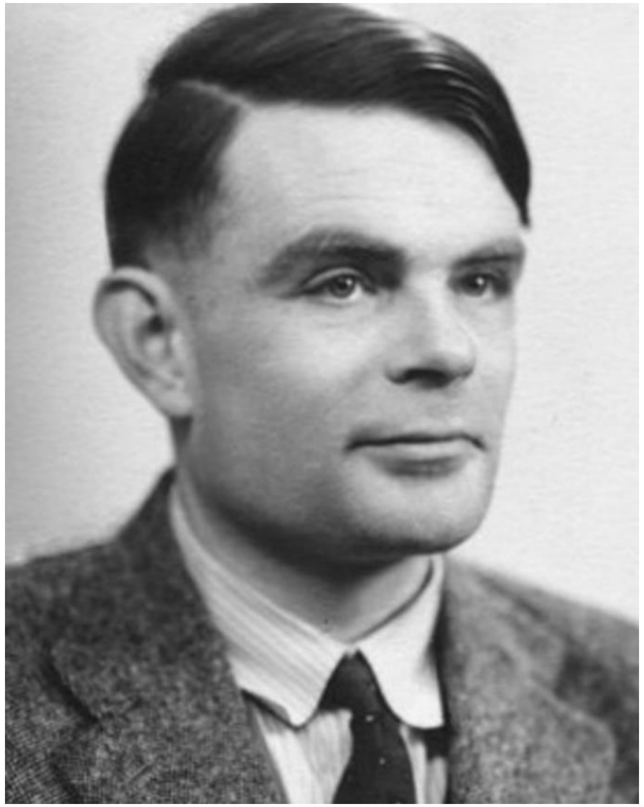
数据之法

结构之美

算法之道

Last updated on 2025.5

zhuyungang@jlu.edu.cn



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

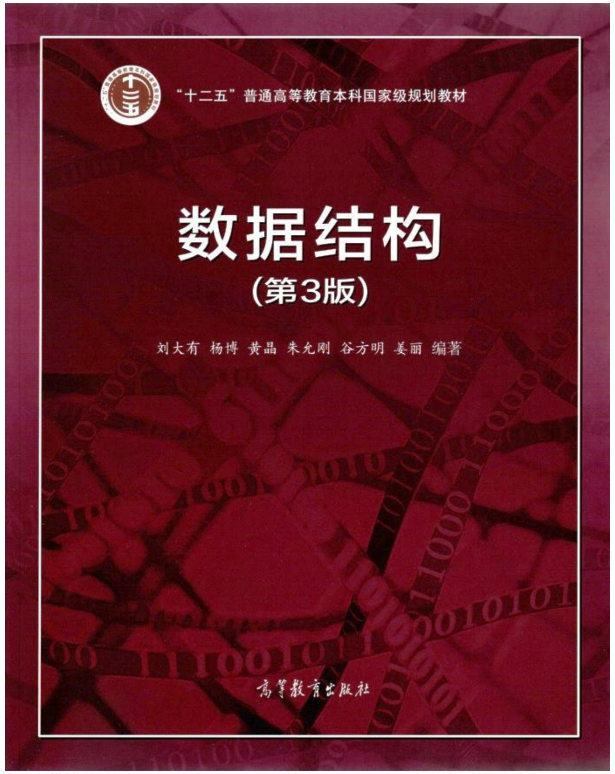
— *Alan Turing*

计算机科学之父

英国皇家科学院院士

曼彻斯特大学教授

剑桥大学研究员



图的概念与存储结构

- **基本概念**
- 图的存储结构

数据之法
结构之美
算法之道

Last updated on 2025.5

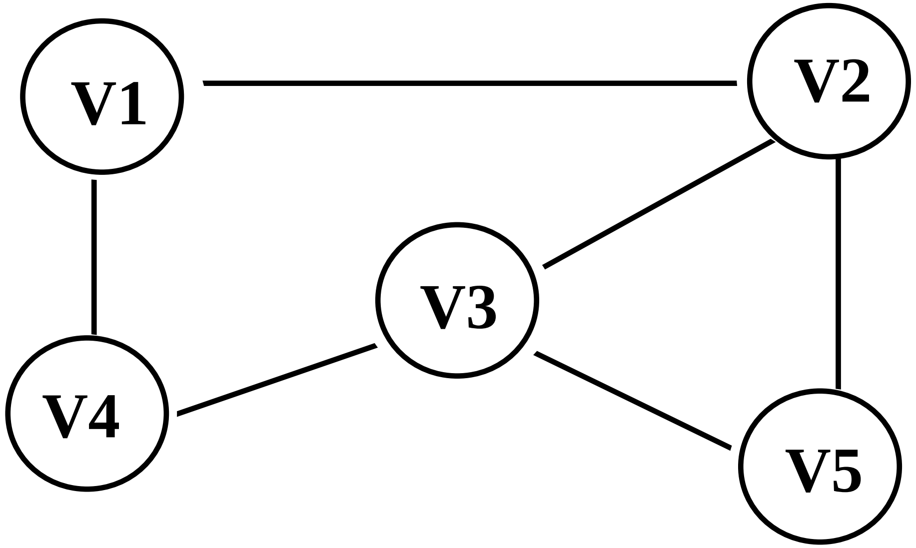
zhuyungang@jlu.edu.cn

图的基本概念

A

Vertex *Edge*

图： 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)。



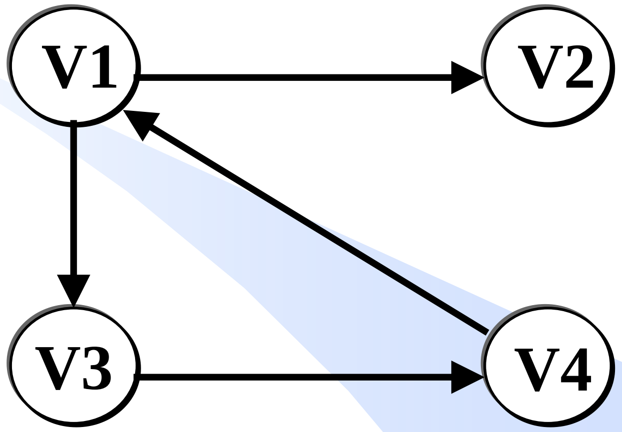
有向图与有向边

- **有向图**：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。

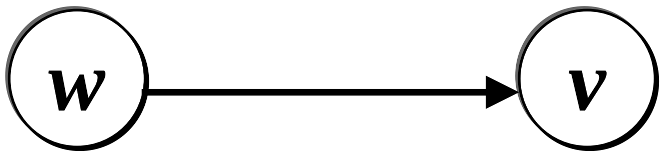
G=(V, E)

V={ V1，V2，V3，V4}

E={<V1,V2>, <V1,V3>, <V3,V4>, <V4,V1>}

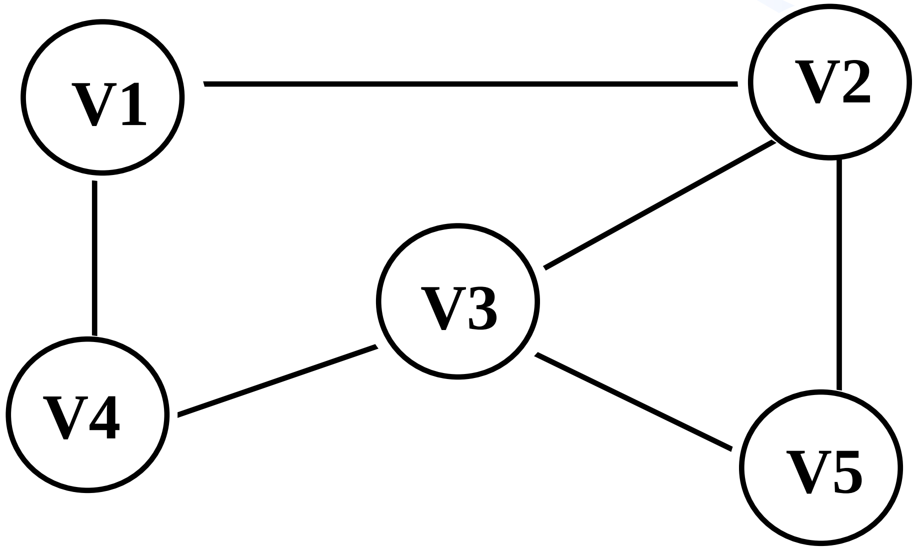


- **有向边**：若G = (V, E)是有向图，则它的一条有向边是由V中两个顶点构成的有序对，亦称为弧，记为<*w*, *v*>，其中*w*是边的始点，又称弧尾；*v*是边的终点，又称弧头。



无向图

➢ 若图中的边无方向性，则称之为无向图。



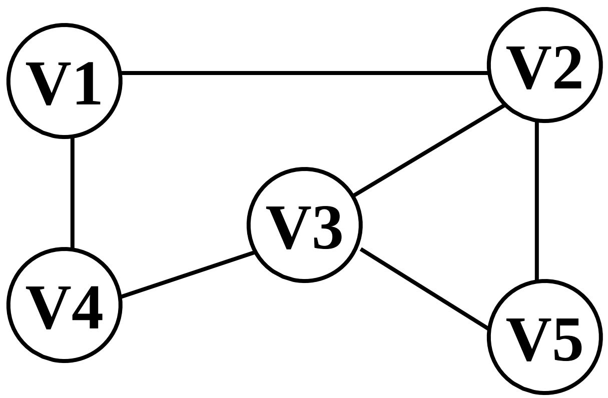
G=(V, E)

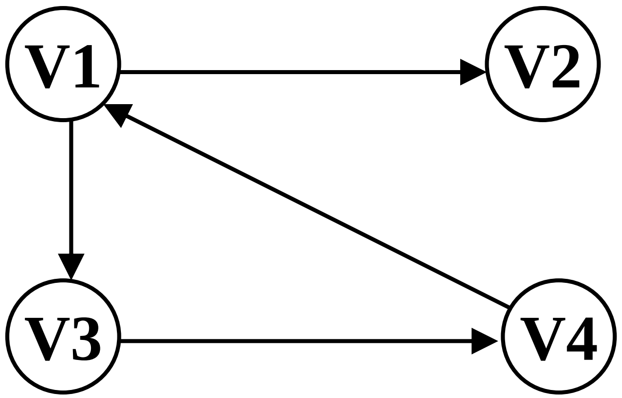
V={V1, V2, V3, V4 ,V5}

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

邻接顶点

- 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$<w, v>$，则称顶点w邻接到顶点v，顶点v邻接自顶点w。

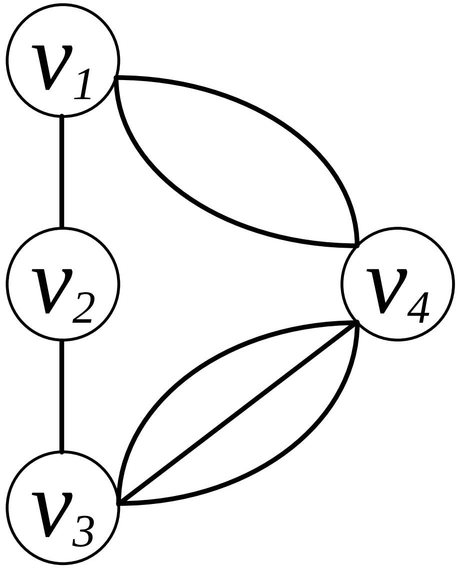


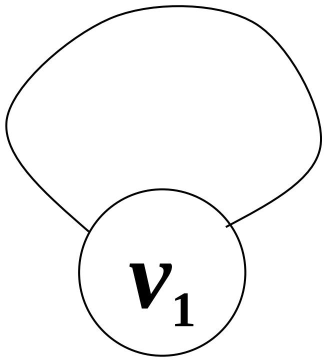


多重图和简单图

C

- 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**。





A

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

A

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。

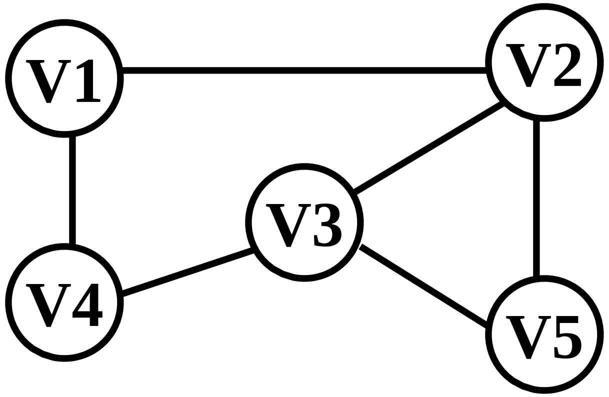
有向完全图：任意两个顶点之间都有方向相反的两条边。包含n个顶点的有向完全图中，有$n(n-1)$条边。

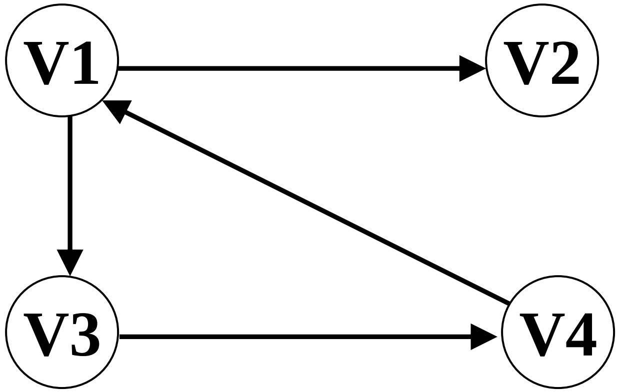
对于n个顶点的完全图，边的条数$e = O(n^2)$

顶点的度

A

- 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

A

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。

图的路径

A

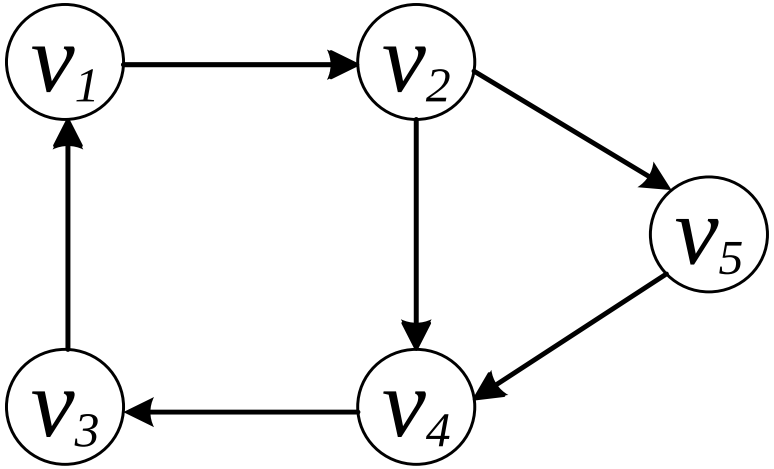
定义 设G是图，若存在一个顶点序列 $v_p, v_1, v_2, \cdots, v_{q-1}, v_q$ 使得 $<v_p, v_1>, <v_1, v_2>, \cdots, <v_{q-1}, v_q>$ 或 $(v_p, v_1), (v_1, v_2), \cdots, (v_{q-1}, v_q)$ 属于E(G)，则称v_p到v_q存在一条路径，其中v_p 称为起点，v_q称为终点。

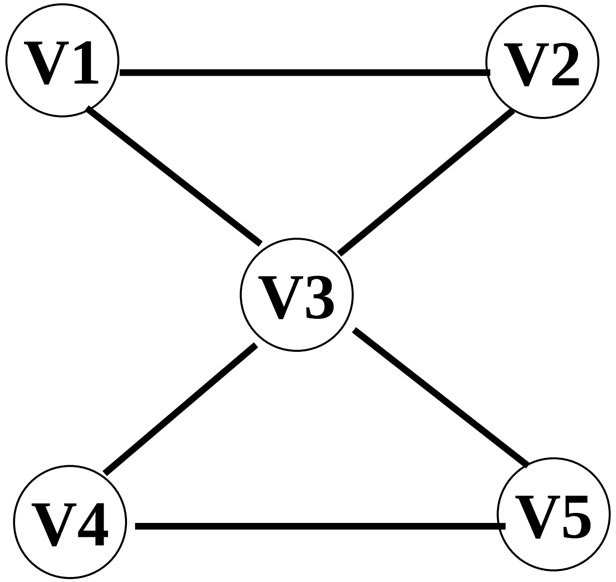
路径的长度是该路径上边的个数。

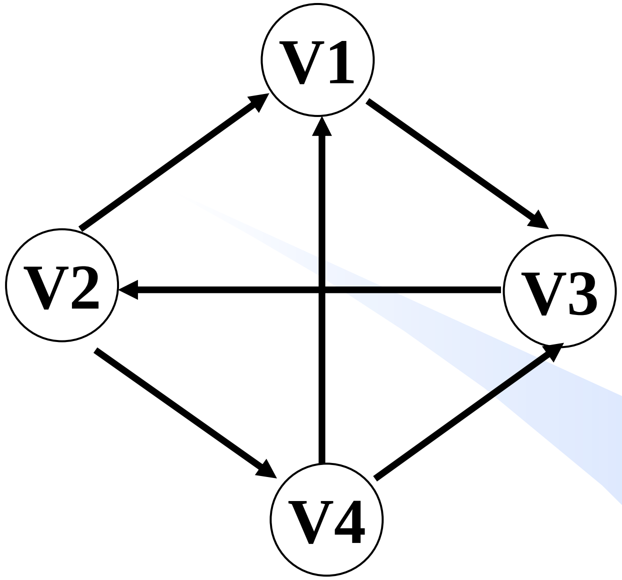
- 如果一条路径上除了起点和终点可以相同外，再不能有相同的顶点，则称此路径为简单路径。
- 如果一条简单路径的起点和终点相同，且路径长度大于等于2，则称之为简单回路。

A

- 下图中，v1到v3之间存在一条路径v1, v2, v5, v4, v3, 同时这也是一条简单路径；v1, v2, v5, v4, v3, v1是一条简单回路。







路径：v1 v3 v4 v3 v5

简单路径：v1 v3 v5

简单回路：v1 v2 v3 v1

路径：v1 v3 v2 v4 v3 v2

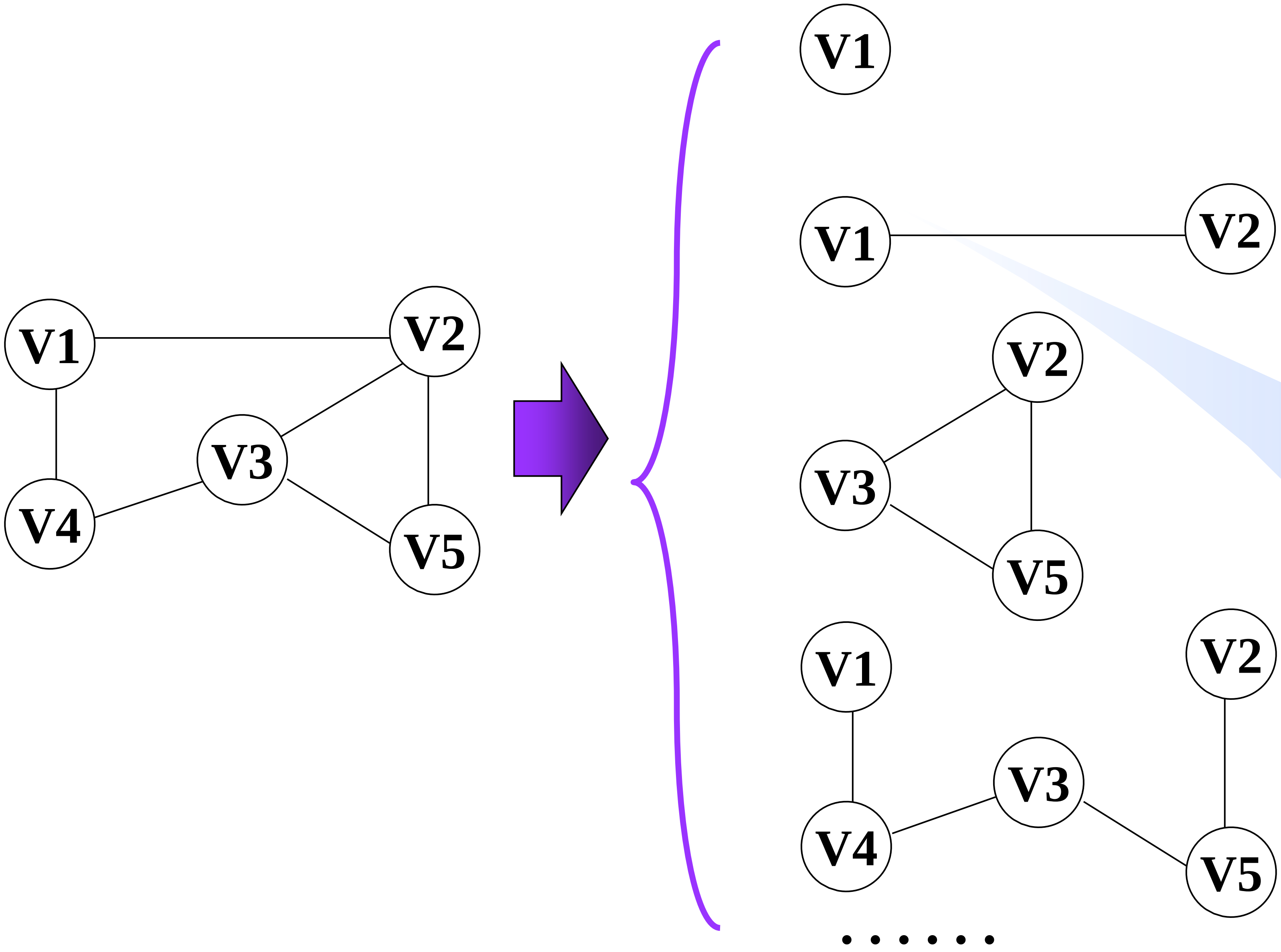
简单路径：v1 v3 v2

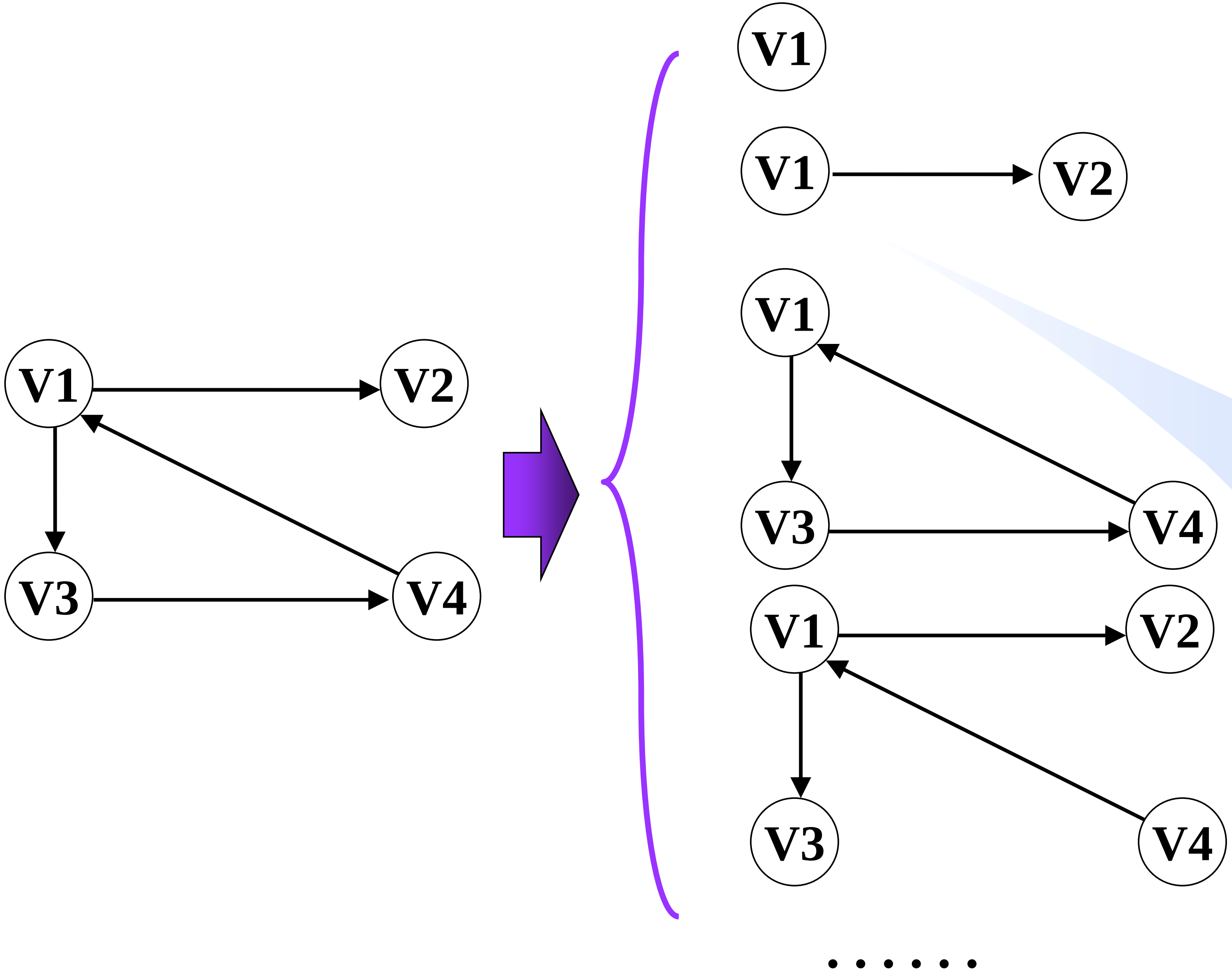
简单回路：v1 v3 v2 v1

子图

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。

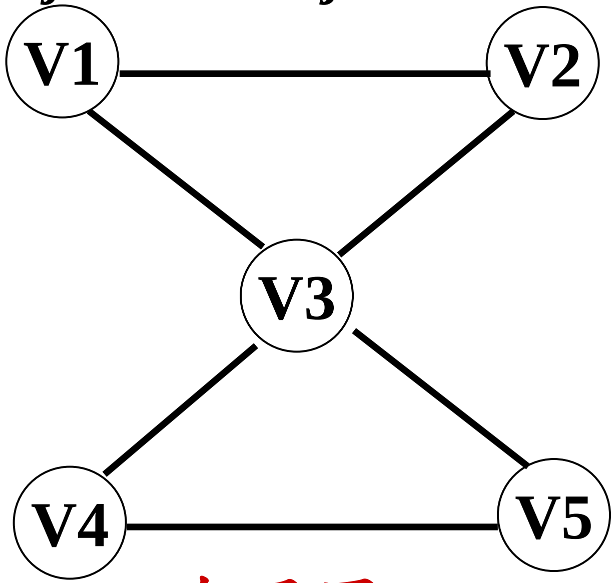
图G的子图就是从G中抽取一部分顶点或边构成的图。


V1
V2
V3
V4
V5
V1
V1
V2
V2
V3
V5
V1
V2
V3
V4
V5
……

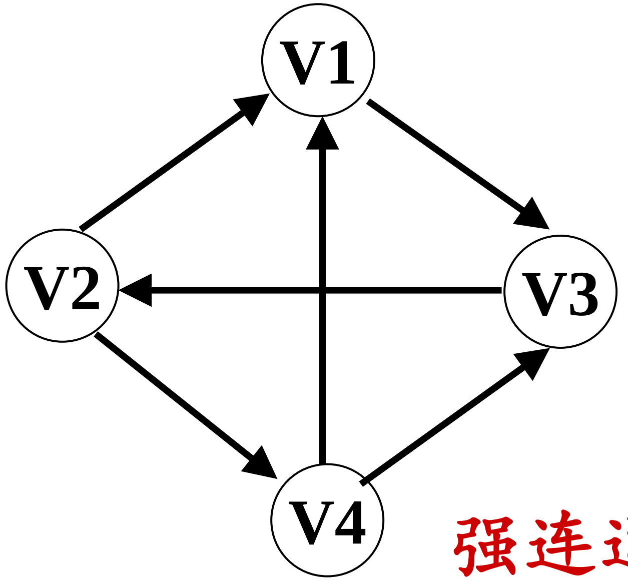

V1
V2
V3
V4
V1
V1
V2
V1
V3
V4
V1
V2
V3
V4
· · · · · ·

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，且图中任意两个顶点都可及（对于图中任意两个顶点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



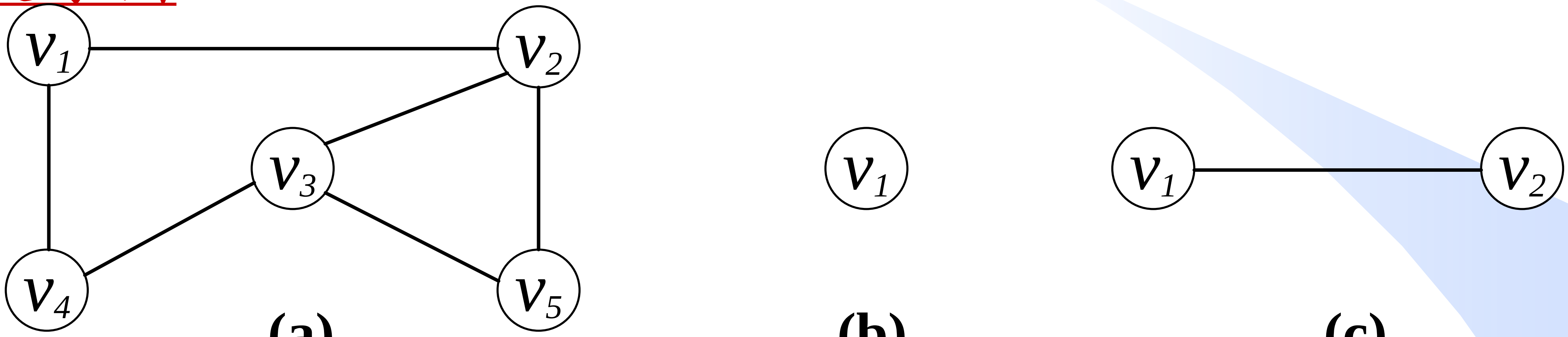
连通图



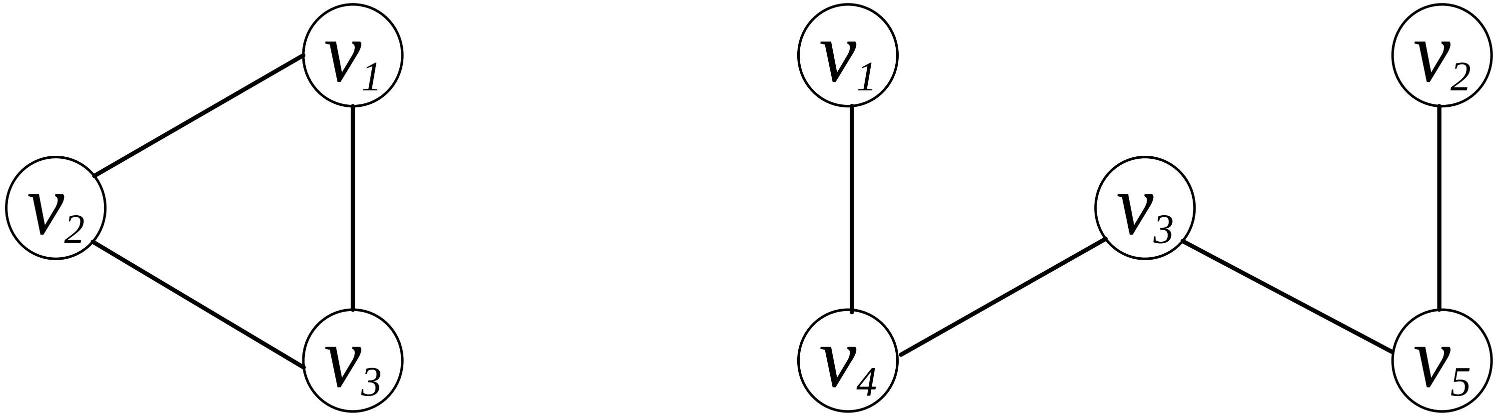
强连通图

无向图的连通子图

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<u>连通子图</u>。



(a) (b) (c)



(d) (e)

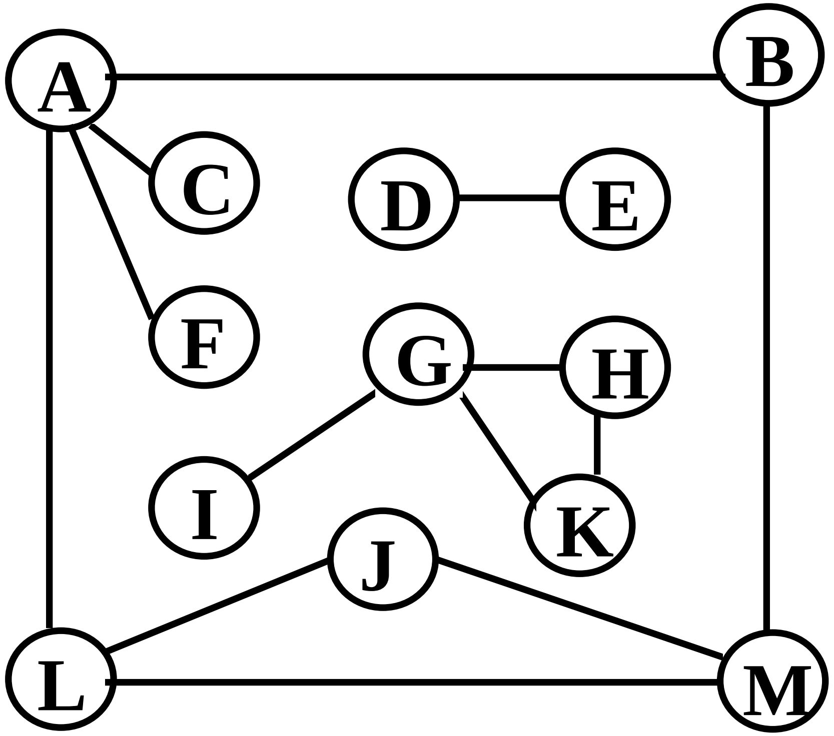
一个图的连通子图
不一定唯一

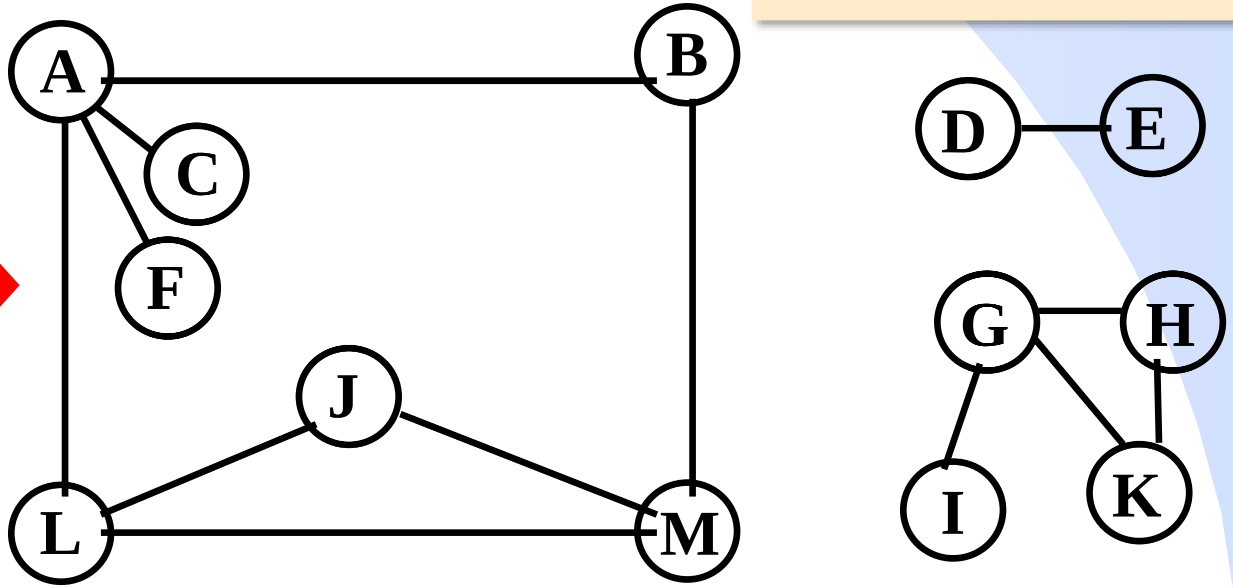
无向图的连通分量

A

- 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



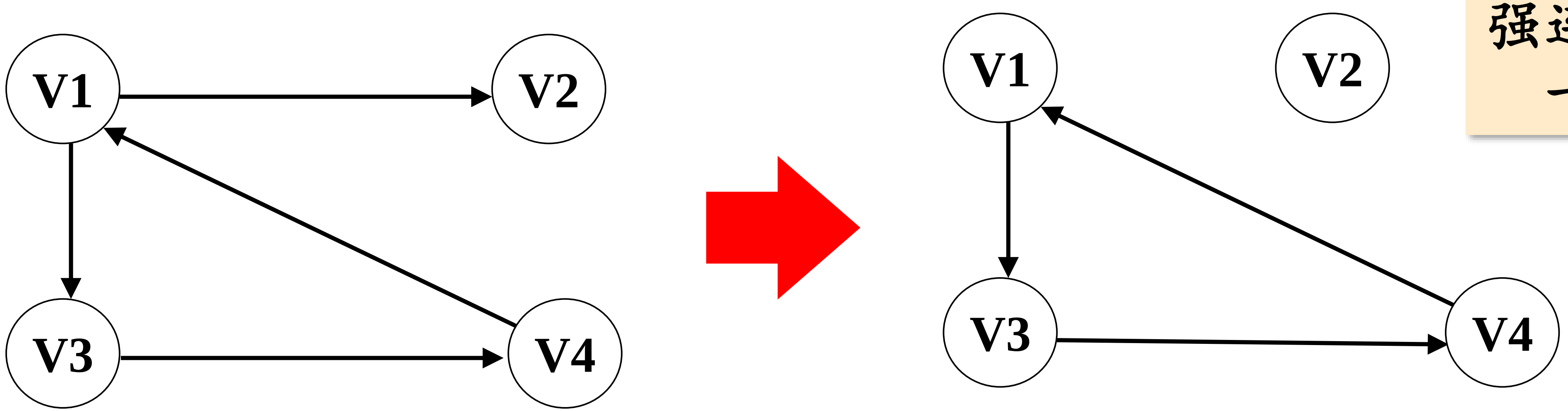


有向图的强连通子图和强连通分量

A

- 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



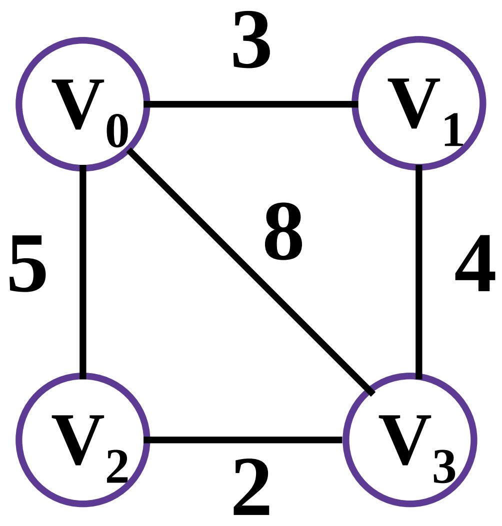
带权图

A

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<u>权图</u>，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，则则$w(u, v)=\infty$

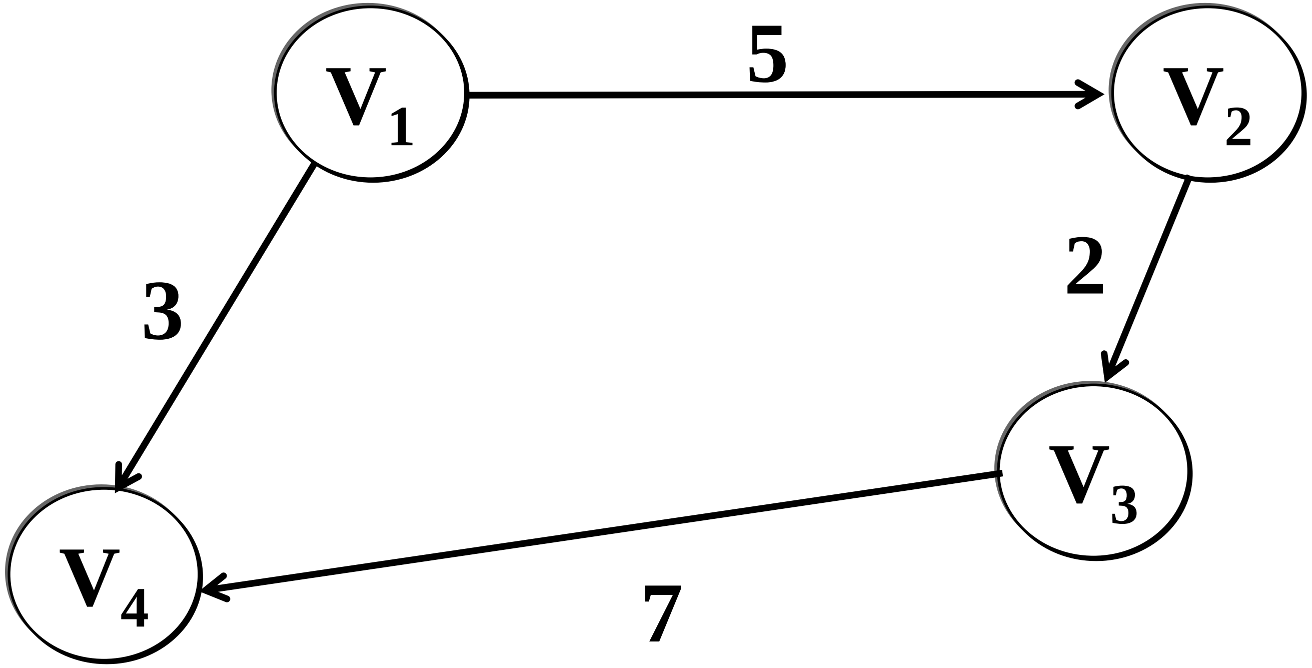


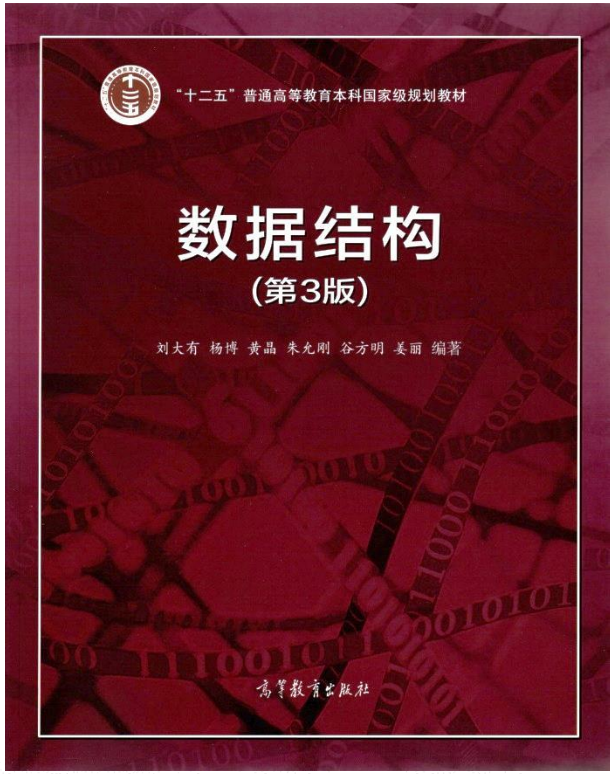
权值可用于表示一个顶点到另一个顶点的距离或费用等。

带权图中的路径长度

A

- 若 $\sigma = (v_0, v_1, v_2, \cdots, v_k)$ 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<u>路径</u> σ 的长度。





图的概念与存储结构

- 图的基本概念
- **图的存储结构**

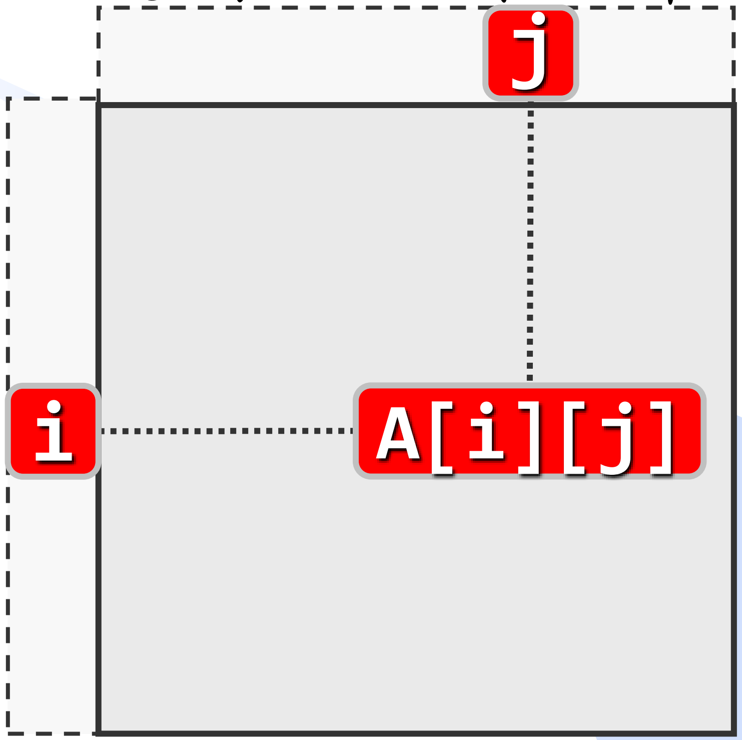
数据之法
结构之美
算法之道

Last updated on 2025.5

zhuyungang@jlu.edu.cn

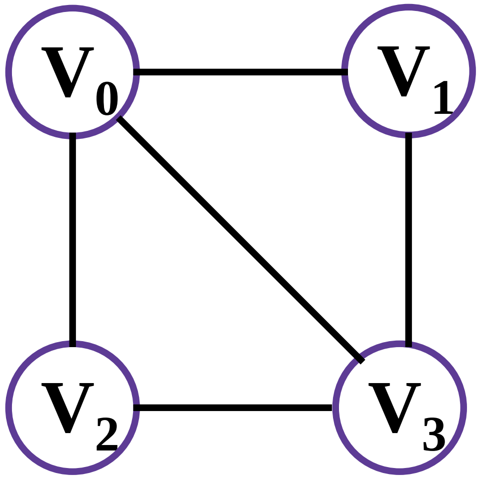
邻接矩阵

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



- 对于无权图：
  - ✓ $a_{ii}=0$；
  - ✓ $a_{ij}=1$，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。
- 对于权图：
  - ✓ $a_{ii}=0$；
  - ✓ $a_{ij}=$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

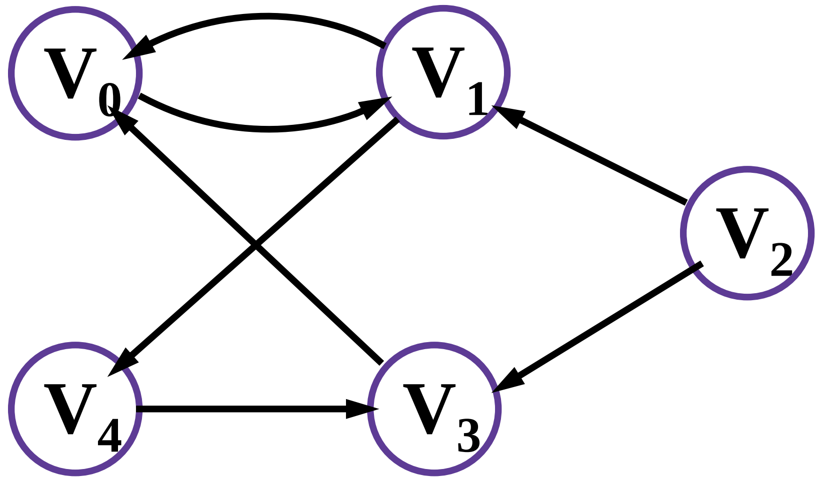


A

$$
\begin{array}{c c}
 & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}
\end{array}
$$

无向图的邻接矩阵是**对称矩阵**。

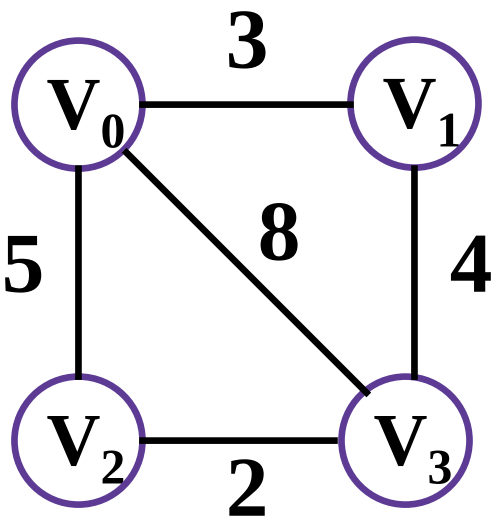
[例]有向图的邻接矩阵



A

$$
\begin{array}{c c}
 & \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} &
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0
\end{pmatrix}
\end{array}
$$

[例]权图的邻接矩阵



A

$$\begin{array}{c c} & \begin{array}{cccc} \mathbf{0} & \mathbf{1} & \mathbf{2} & \mathbf{3} \end{array} \\ \begin{array}{c} \mathbf{0} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \end{array} & \begin{pmatrix} \mathbf{0} & 3 & 5 & 8 \\ 3 & \mathbf{0} & \infty & 4 \\ 5 & \infty & \mathbf{0} & 2 \\ 8 & 4 & 2 & \mathbf{0} \end{pmatrix} \end{array}$$

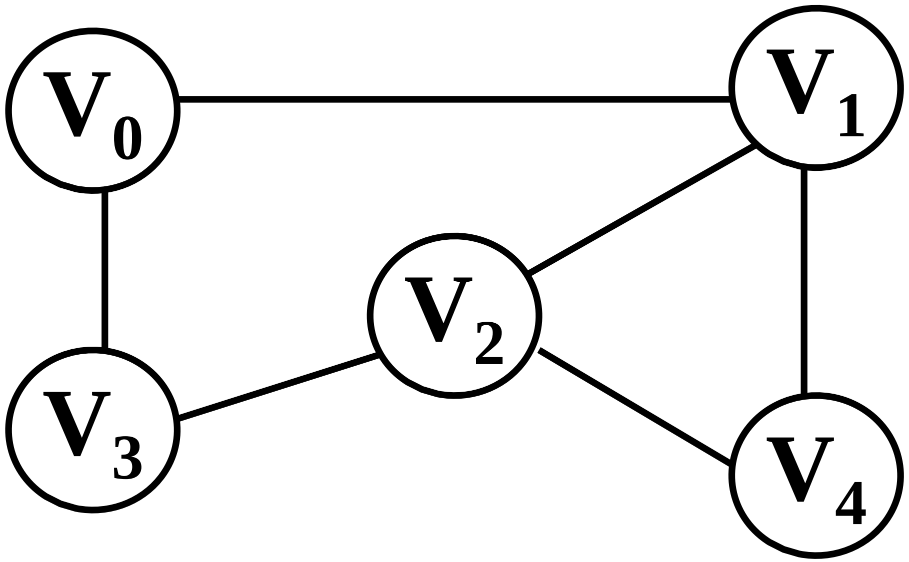
特点： 无向图的邻接矩阵是对称矩阵，有向图邻接矩阵不一定对称。

基于邻接矩阵求图中顶点的度

A

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



$$
\begin{array}{c c}
 & \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} &
\begin{pmatrix}
0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0
\end{pmatrix}
\end{array}
$$

A

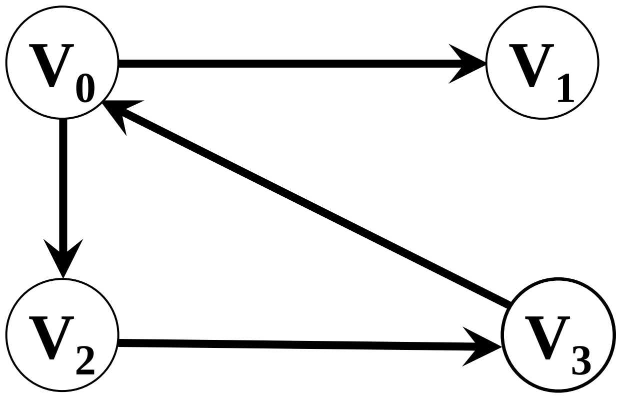
基于邻接矩阵求图中顶点的度

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

邻接矩阵第i列的1的个数为顶点V_i的入度。

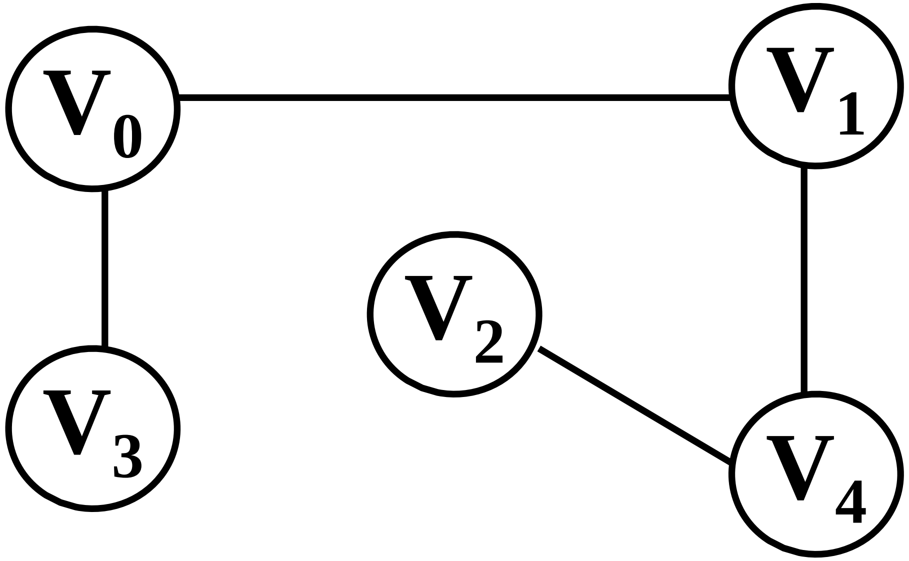
第i列有一个1，就意味着有一条引入（指向）V_i的边



$$
\begin{array}{c}
\\ 0 \\ 1 \\ 2 \\ 3
\end{array}
\begin{array}{c}
\begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\
\begin{pmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

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，浪费空间和时间。

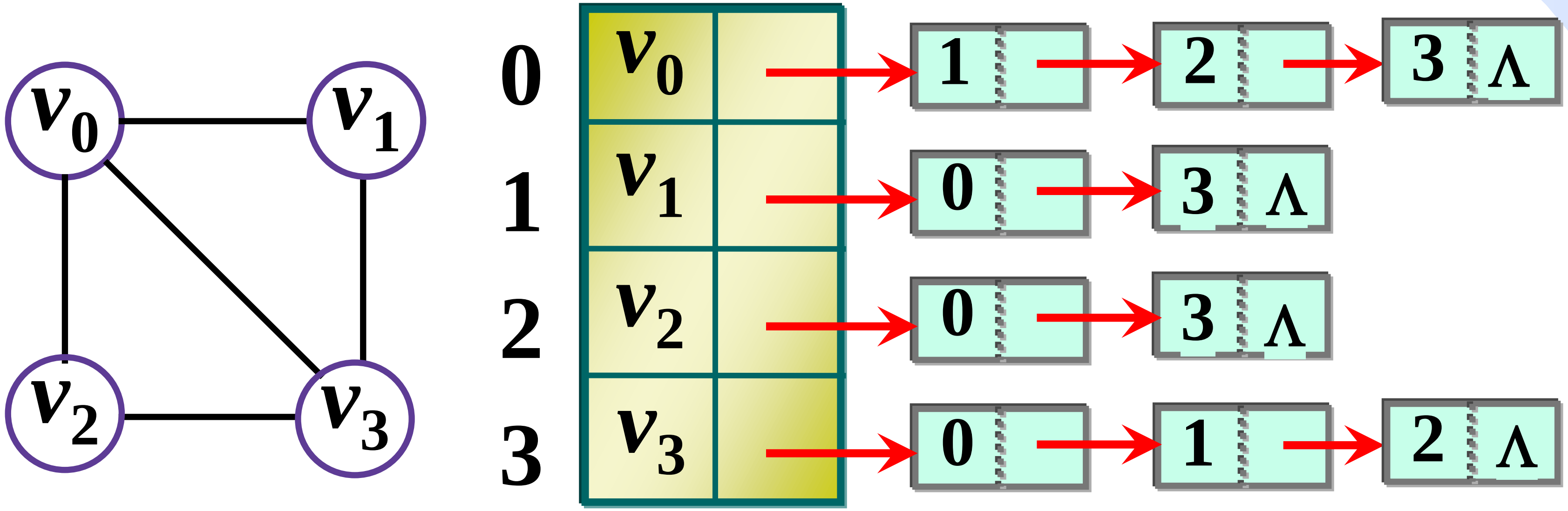


$$
\begin{array}{c c}
 & \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} &
\begin{pmatrix}
0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0
\end{pmatrix}
\end{array}
$$

邻接表

A

- 顺序存储顶点表。
- 对图的每个顶点建立一个单链表，第i个单链表中包含顶点v_i的所有邻接顶点，该链表称为边链表。
- 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<u>邻接表</u>。



邻接表

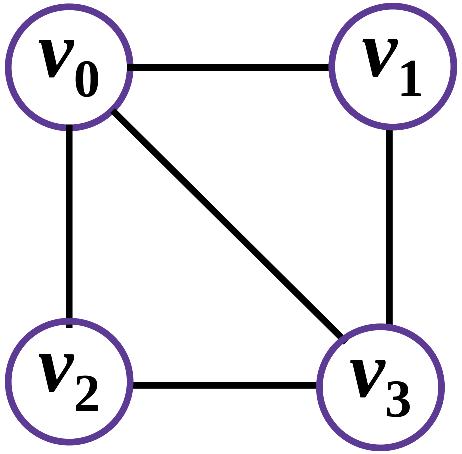
➢边链表的每一个结点叫做边结点，对于非权图和权图边结点结构分别为：

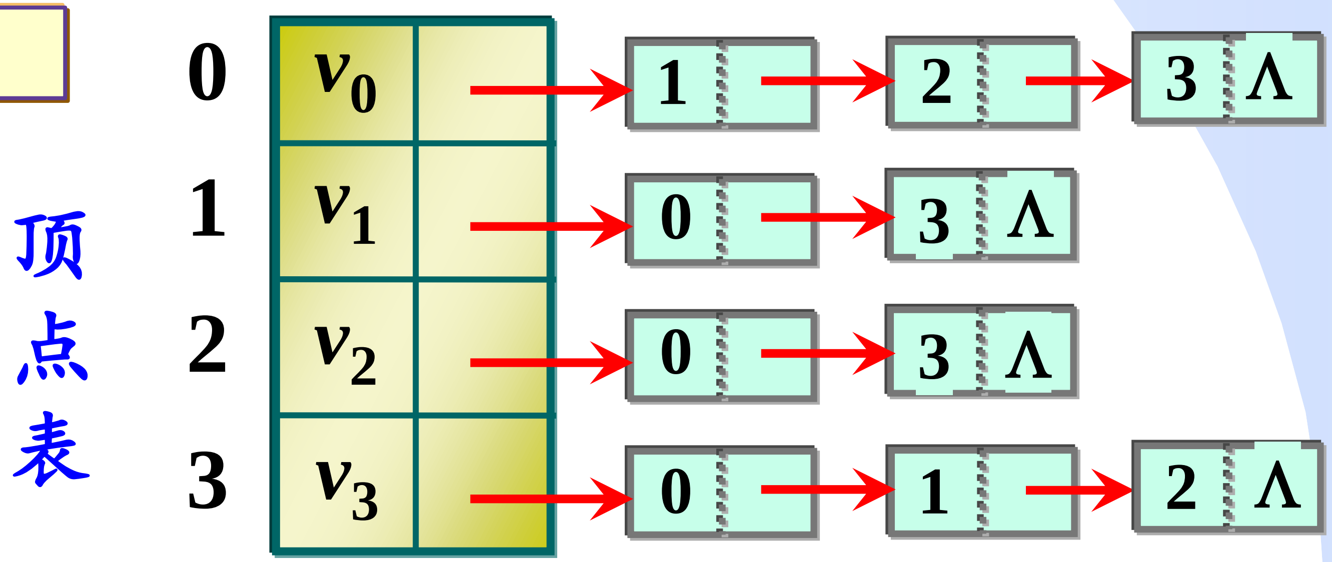
VerAdj	*link*

VerAdj	*cost*	*link*

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<*v*, VerAdj>的权值；

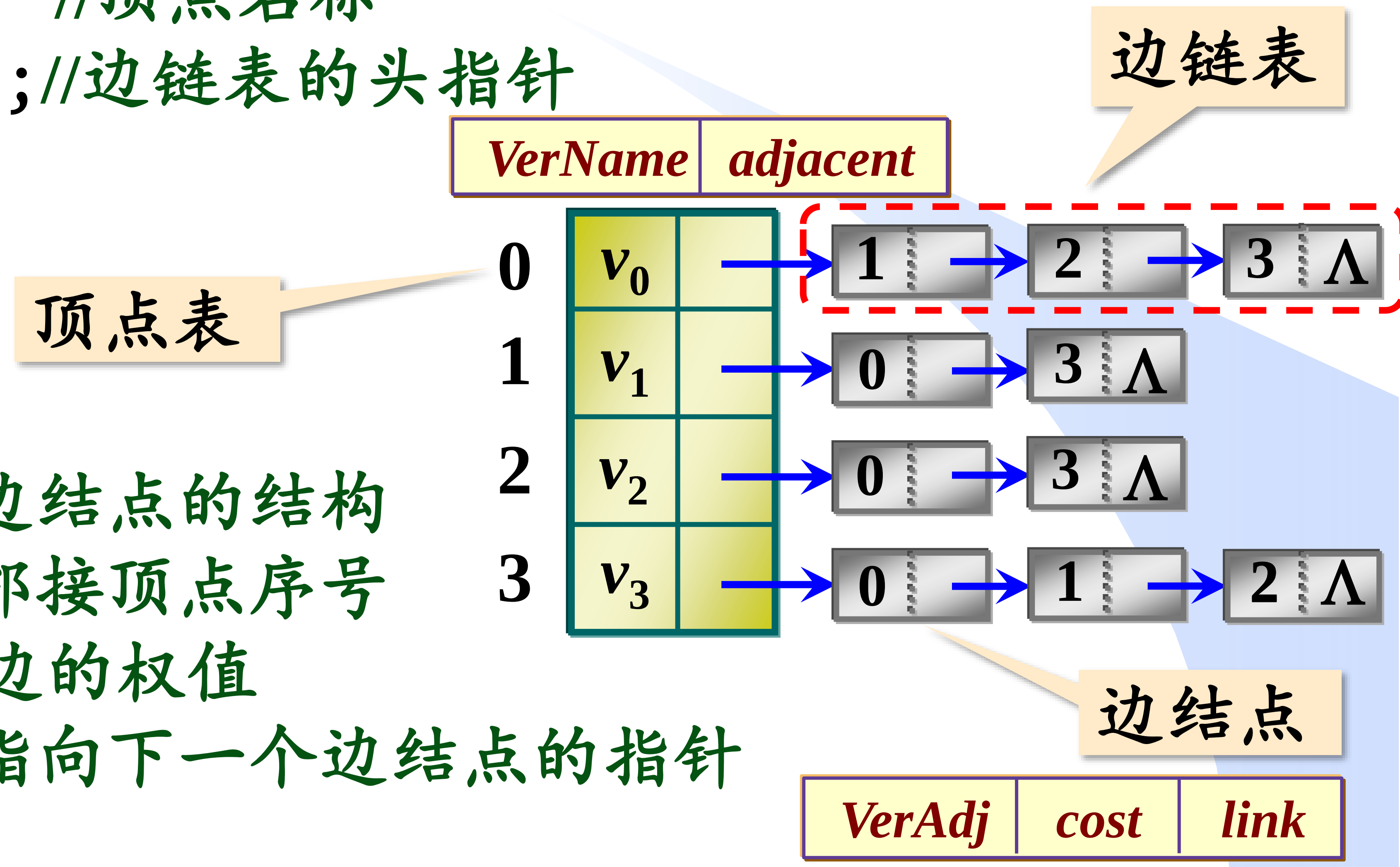
VerName	*adjacent*





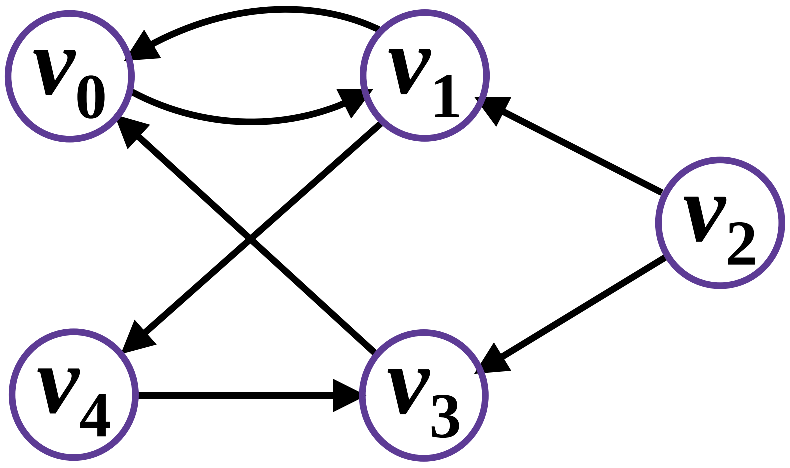
用邻接表存储图

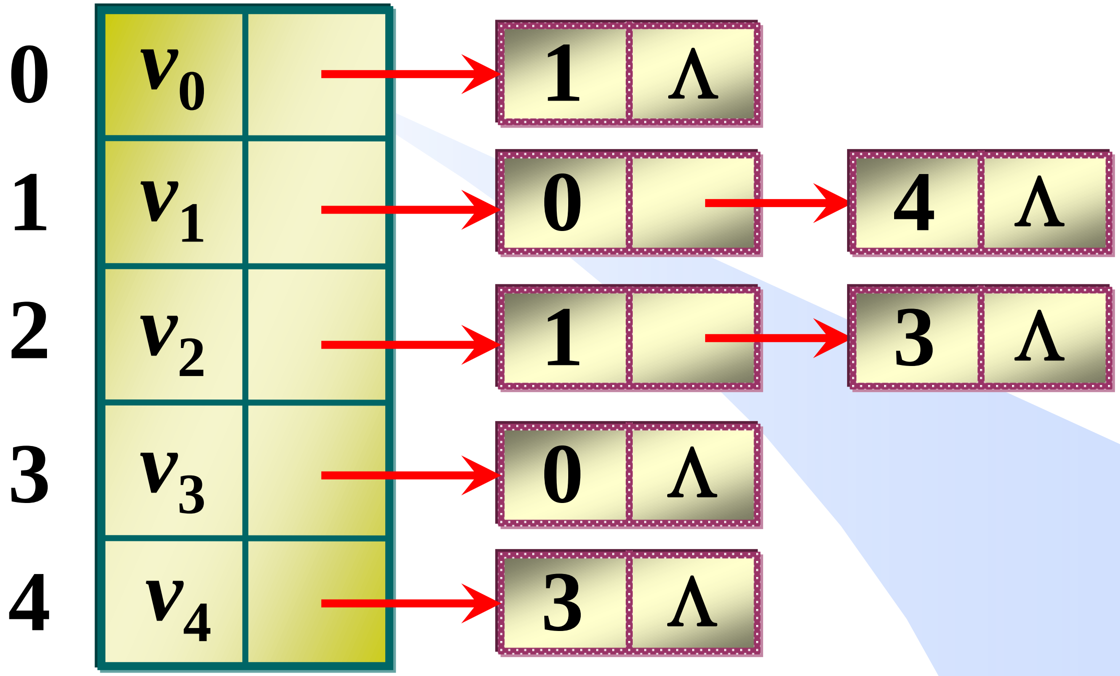
```
struct Vertex{    //顶点表中结点的结构
  int VerName;    //顶点名称
  Edge *adjacent;//边链表的头指针
};

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

A





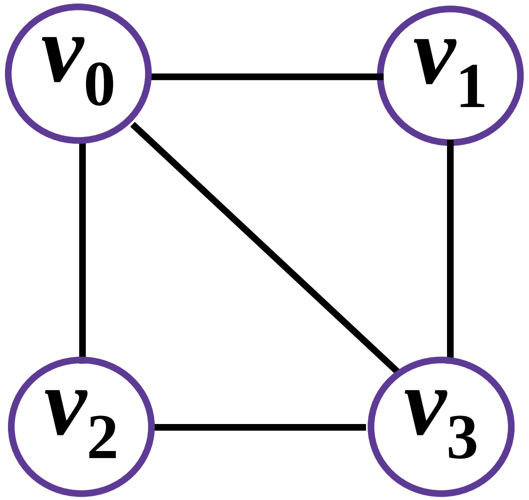
边结点的个数 = e
e为图中边的条数

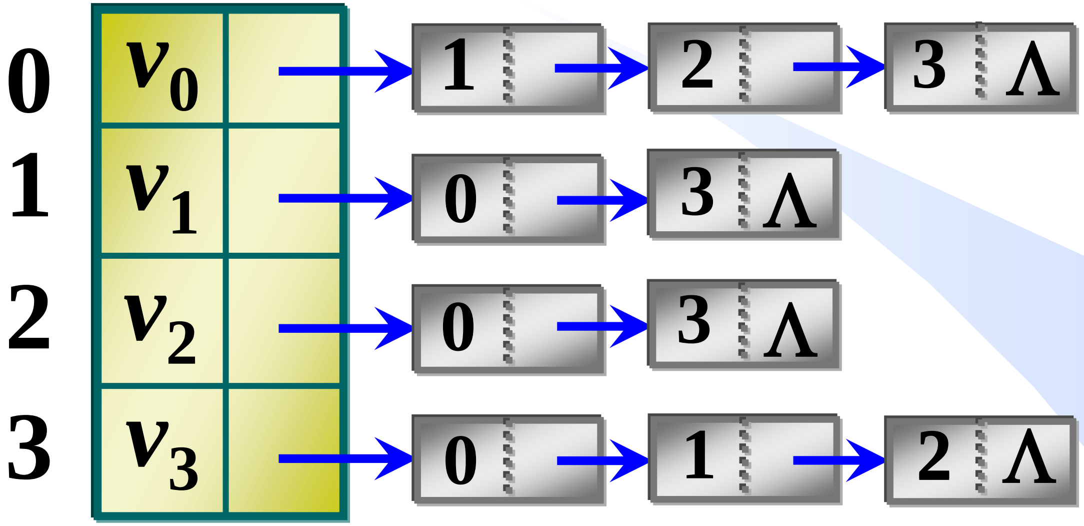
1个边结点 ⇔ 1条边

占用空间
O(n+e)

A

[例]无向图的邻接表





边结点的个数=2*e*

*e*为图中边的条数

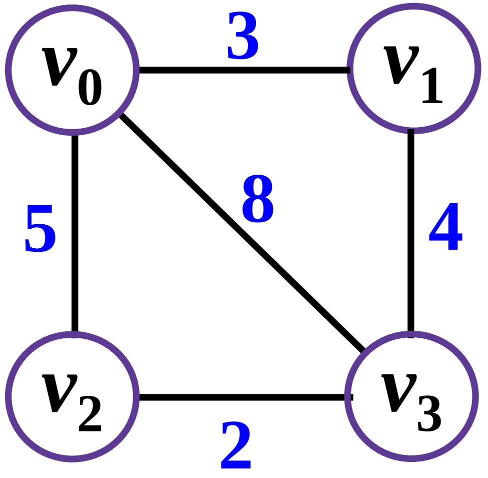
2个边结点 ⇔ 1条边

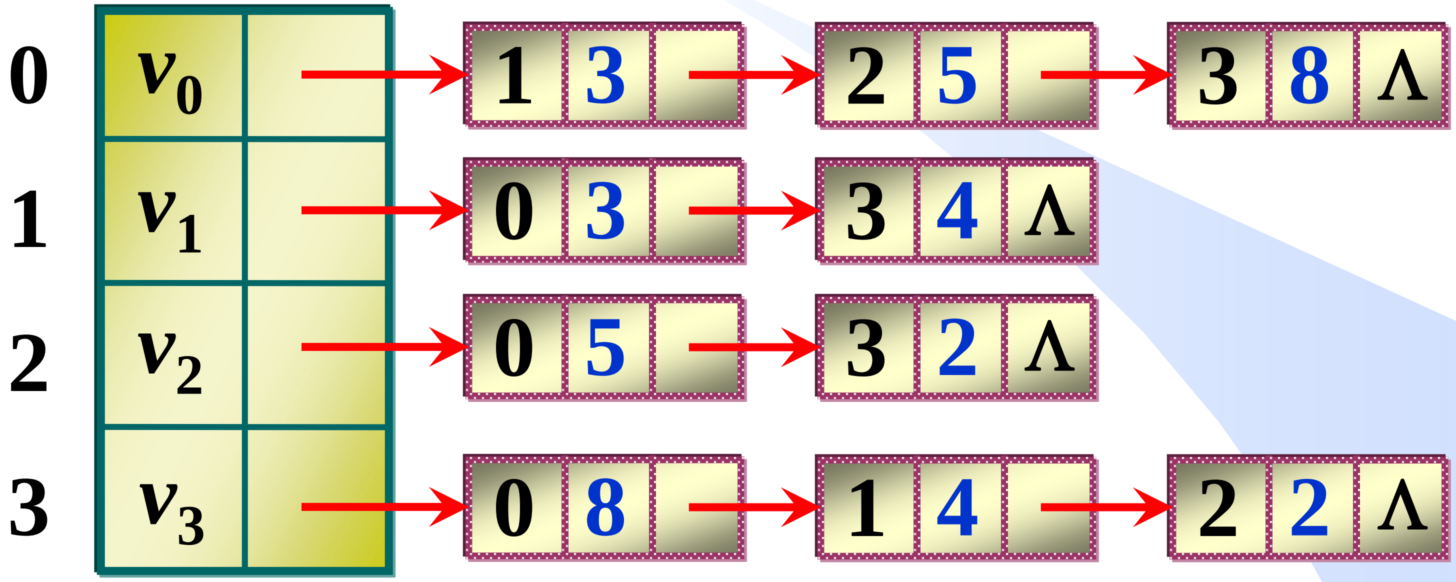
占用空间

O(*n*+*e*)

[例] 带权图的邻接表

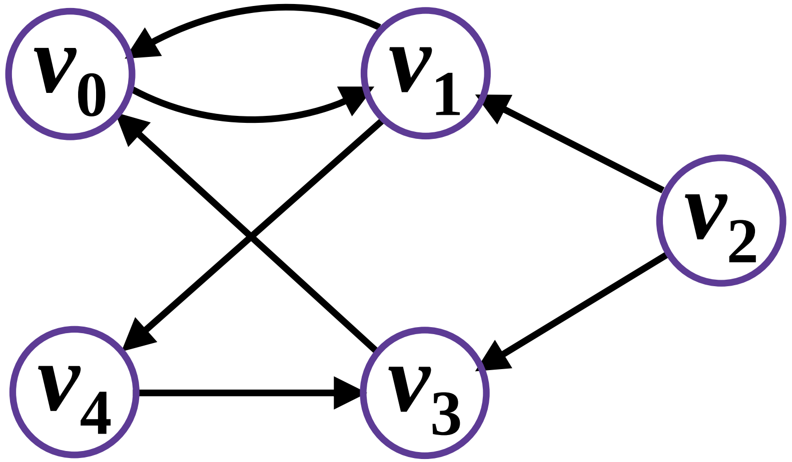
A

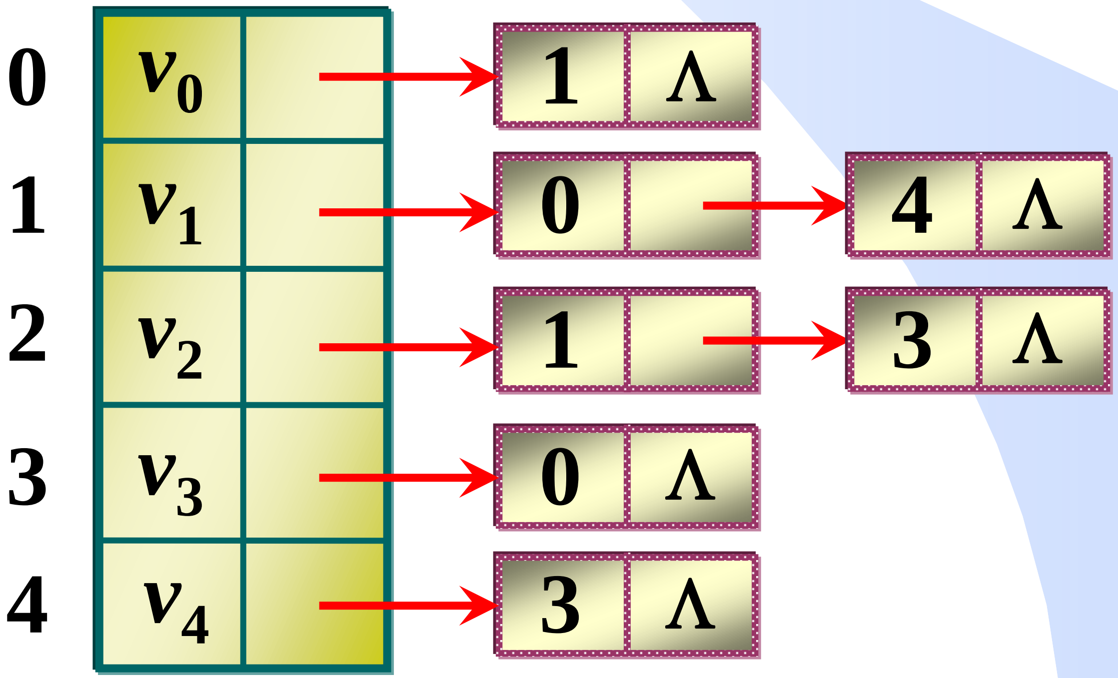




邻接表中统计结点的度

- 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

统计顶点的入度

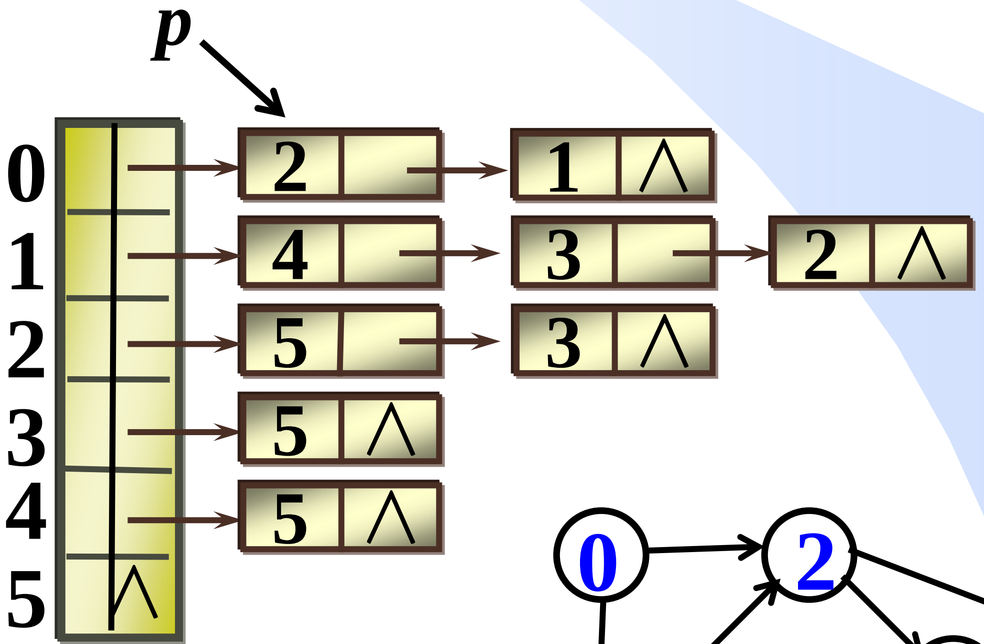
VerAdj	link

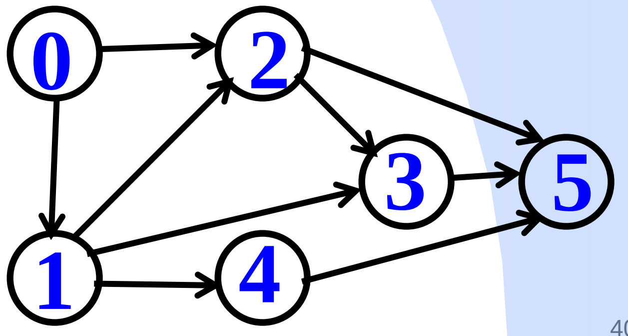
A

```
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
      for(Edge* p=Head[i].adjacent; p!=NULL; p=p->link)
         int k = p->VerAdj;
         InDegree[k]++;
      }
}
```

时间复杂度

O(n+e)



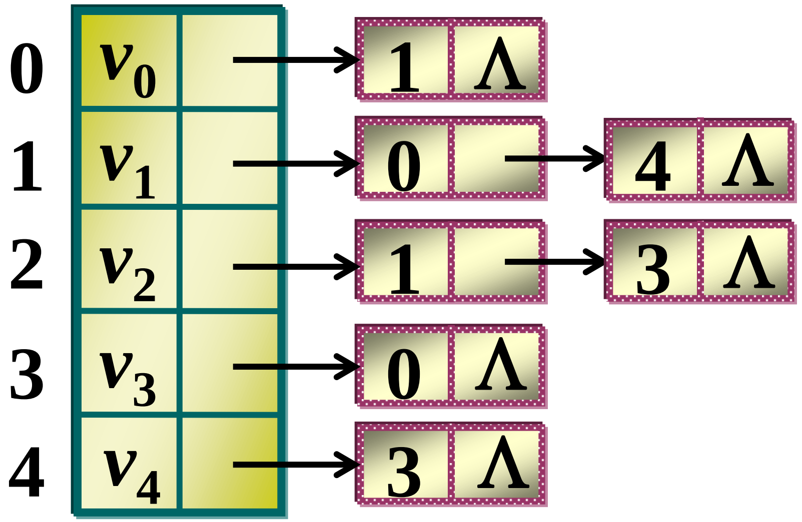


InDegree	0	1	2	3	4	5
	0	1	2	2	1	3

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)
修改/删除边(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)

$$
\begin{array}{c c}
 & \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} &
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0
\end{pmatrix}
\end{array}
$$



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

邻接矩阵 vs. 邻接链表

A

	邻接矩阵	邻接表
检测是否存在边(V_i, V_j)	O(1)	$O(d_i)$
修改/删除边(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, $d_i \le n$