



Data Structures

Ch6



Graphs

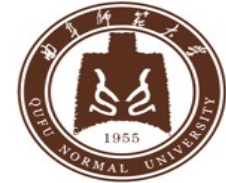
2024 年 11 月 1 日

学而不厌 诲人不倦

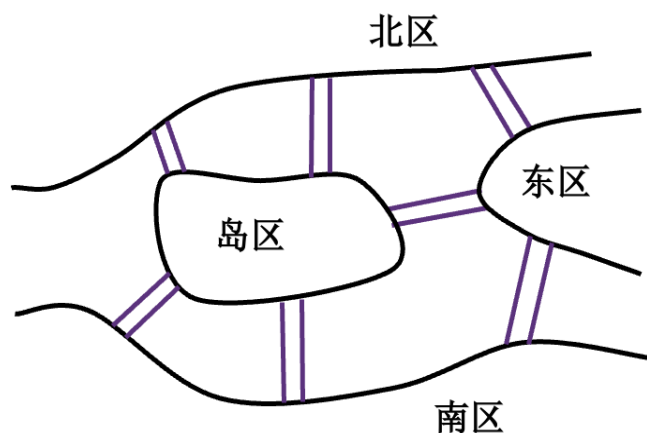
- ➡ **6.1 引言**
- ➡ **6.2 图的逻辑结构**
- ➡ **6.3 图的存储结构及实现**
- ➡ **6.4 最小生成树**
- ➡ **6.5 最短路径**
- ➡ **6.6 有向无环图及其应用**
- ➡ **6.7 扩展与提高**
- ➡ **6.8 应用实例**

6.1 引言

6.1 引言

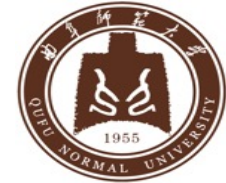


图

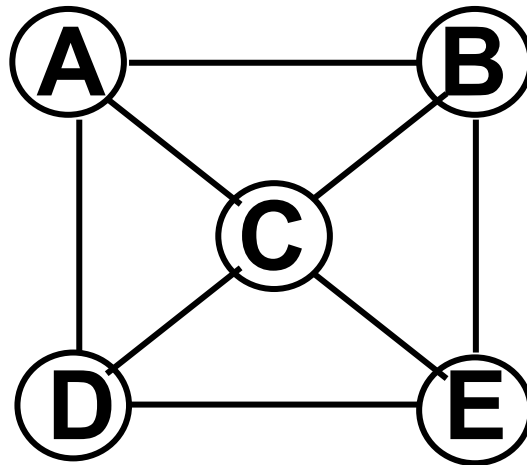


欧拉1707年出生在瑞士，19岁开始发表论文，直到76岁。几乎每一个数学领域都可以看到欧拉的名字，从初等几何的欧拉线、多面体的欧拉定理、立体解析几何的欧拉变换公式、四次方程的欧拉解法到数论中的欧拉函数、微分方程的欧拉方程、数论的欧拉常数、变分学的欧拉方程、复变函数的欧拉公式等等。欧拉对哥尼斯堡七桥问题的提出和解答开创了图论的研究。

6.1 引言



图



图是一种较线性表和树更为复杂的数据结构。

线性表： 线性结构（前驱、后继）

树： 层次结构（父子）

图： 任意两个数据元素之间都可能相关（邻接）

6.1 引言

农夫过河问题

【问题】 农夫过河问题。一个农夫带着一只狼、一只羊和一筐菜，想从河一边（左岸）乘船到另一边（右岸），由于船太小，农夫每次只能带一样东西过河，但是如果没有农夫看管，则狼会吃羊，羊会吃菜。其给出过河方案。

【想法——数据模型】 用 0 表示在河的左岸，用 1 表示在河的右岸，将每一个可能的状态抽象为一个顶点，边表示状态转移发生的条件。



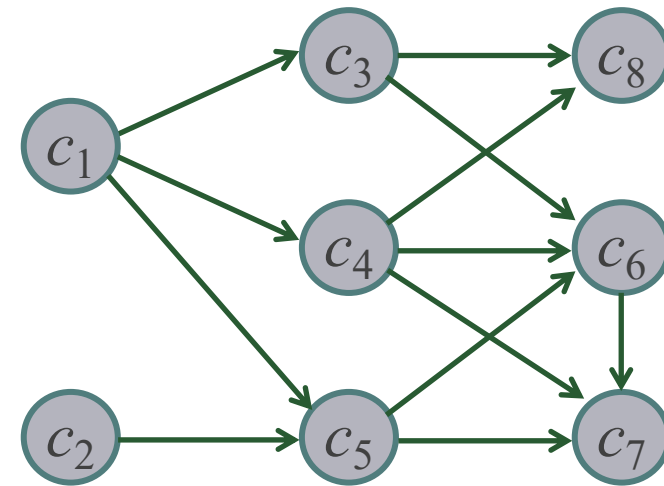
6.1 引言

教学计划编排

【问题】 已知计算机专业的核心课程如下表所示，编制合适的教学计划。

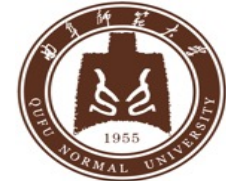
【想法——数据模型】 用顶点表示课程，如果从顶点 c_i 到 c_j 之间存在边 $\langle c_i, c_j \rangle$ ，则表示课程 c_i 是课程 c_j 的先修课。

课程编号	课程名称	先修课程
c_1	程序设计基础	无
c_2	电子技术基础	无
c_3	离散数学	c_1
c_4	数据结构	c_1
c_5	计算机原理	c_1 c_2
c_6	操作系统	c_3 c_4 c_5
c_7	计算机网络	c_4 c_5 c_6
c_8	数据库原理及应用	c_4 c_5



6.2 图的逻辑结构

6-2-1 图的定义与基本术语



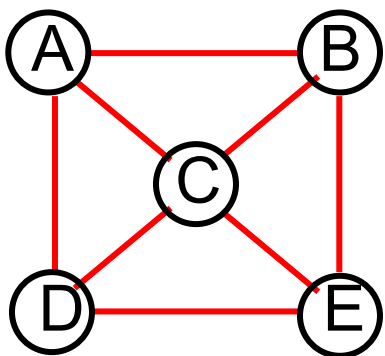
1. 图的定义

图 G 是由两个集合：顶点集 $V(G)$ 和边集 $E(G)$ 组成的，记作 $G=(V(G), E(G))$ 。
通常表示为：

$$G=(V, E)$$

V 是顶点的有穷**非空**集合

E 是两个顶点之间的关系，即边的有穷集合

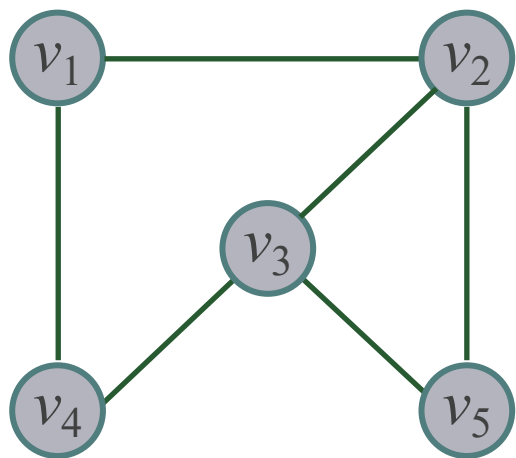




2. 图的分类

图（边是否有方向）
无向图
有向图

无向图：边是顶点的无序对，即边没有方向性。



无向边：表示为 (v_i, v_j) ，顶点 v_i 和 v_j 之间的边没有方向

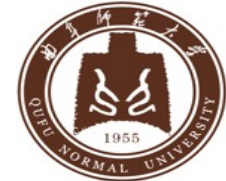
无向图：图中任意两个顶点之间的边都是无向边

$$V = \{ v_1, v_2, v_3, v_4, v_5 \}$$

$$E = \{ (v_1, v_2), (v_1, v_4), (v_2, v_3), (v_2, v_5), (v_3, v_4), (v_3, v_5) \}$$

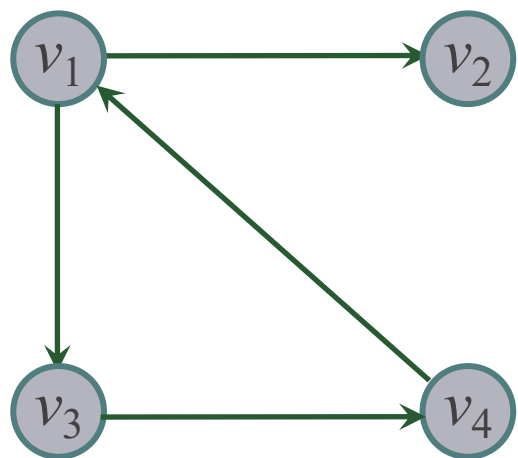
(v_1, v_2) 表示顶点 v_1 和 v_2 之间的边

$$(v_1, v_2) = (v_2, v_1)$$



2. 图的分类

有向图：其边是顶点的有序对，即边有方向性。



📌 有向边（弧）：表示为 $\langle v_i, v_j \rangle$ ，从 v_i 到 v_j 的边有方向

📌 有向图：图中任意两个顶点之间的边都是有向边

$$V = \{ v_1, v_2, v_3, v_4 \}$$

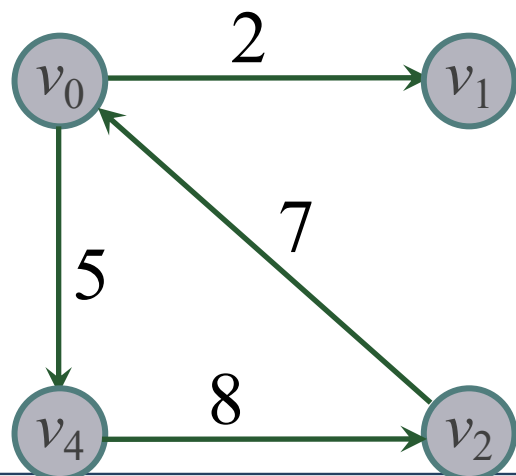
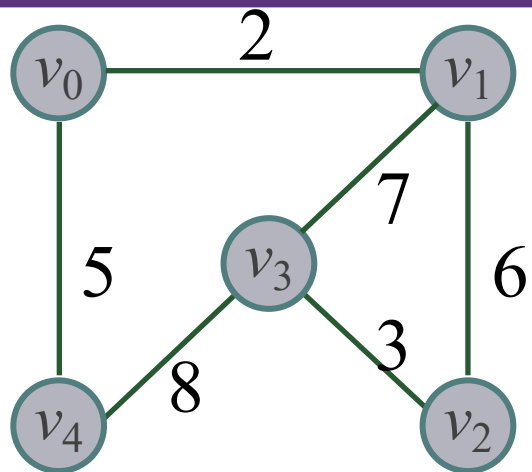
$$E = \{ \langle v_1, v_2 \rangle, \langle v_1, v_3 \rangle, \langle v_3, v_4 \rangle, \langle v_4, v_1 \rangle \}$$

通常有向图的边称为**弧**， $\langle v_1, v_2 \rangle$ 表示顶点 v_1 到 v_2 的弧。
称 v_1 为**弧尾**，称 v_2 为**弧头**。

$$\langle v_1, v_2 \rangle \neq \langle v_2, v_1 \rangle$$



2. 图的分类



图（边上是否带权）

非带权图

带权图



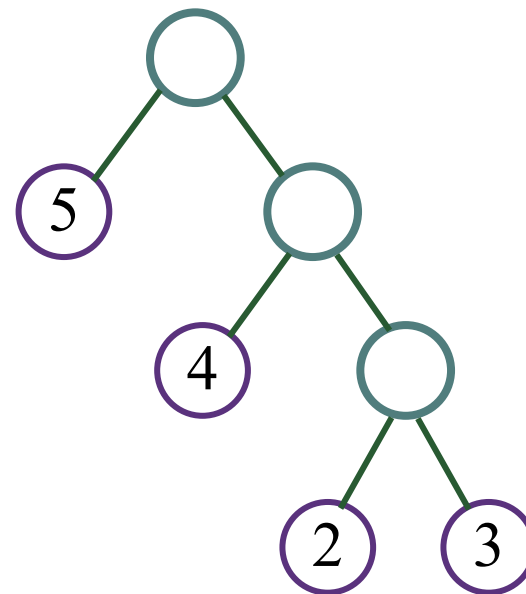
权：对边赋予的有意义的数值量



带权图（网图）：边上带权的图



树结构中，权通常赋予在结点

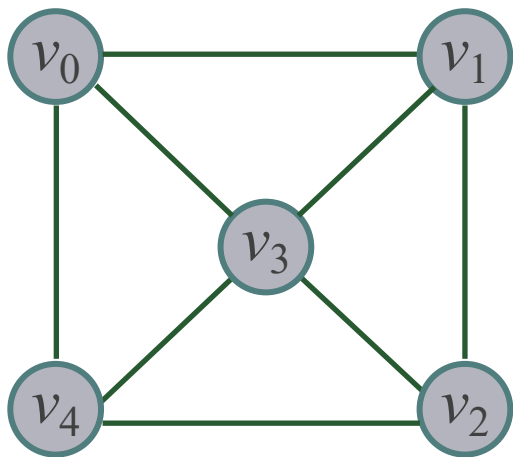


带权的无向图称为**无向网**。

带权的有向图称为**有向网**。



2. 图的分类

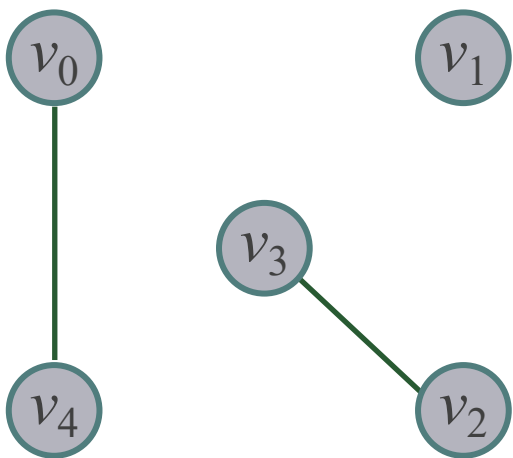


稠密图：边数很多的图

图（边数的多寡）

稠密图

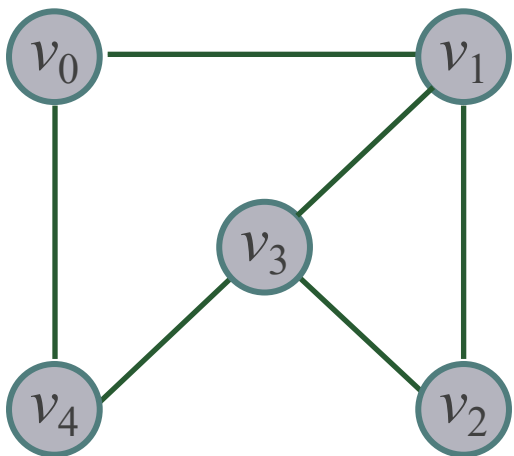
稀疏图



稀疏图：边数很少的图



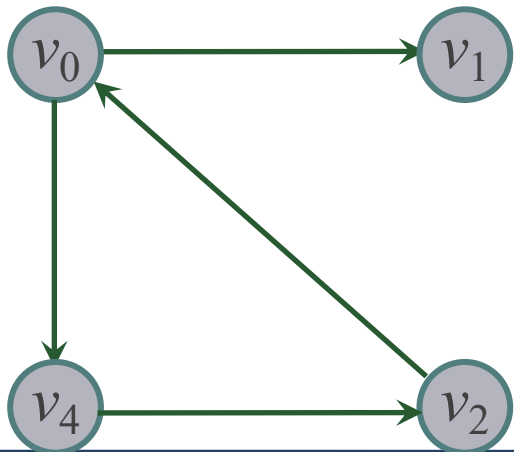
3. 图的逻辑关系



邻接、依附：无向图中，对于任意两个顶点 v_i 和顶点 v_j ，若存在边 (v_i, v_j) ，则称顶点 v_i 和顶点 v_j 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

v_0 的邻接点： v_1 、 v_4

v_2 的邻接点： v_1 、 v_3



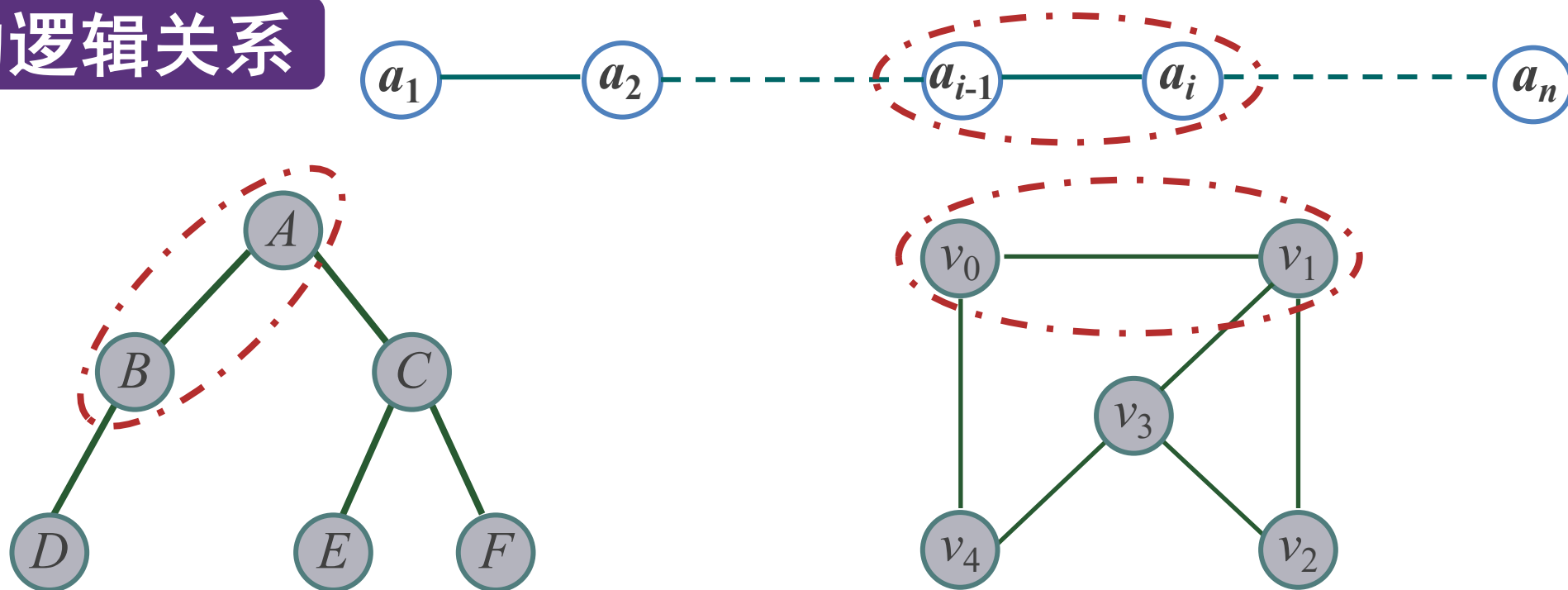
邻接、依附：有向图中，对于任意两个顶点 v_i 和顶点 v_j ，若存在弧 $\langle v_i, v_j \rangle$ ，则称顶点 v_i 邻接到 v_j ，顶点 v_j 邻接自 v_i ，同时称弧 $\langle v_i, v_j \rangle$ 依附于顶点 v_i 和顶点 v_j

v_0 的邻接点： v_1 、 v_4

v_2 的邻接点： v_0



3. 图的逻辑关系

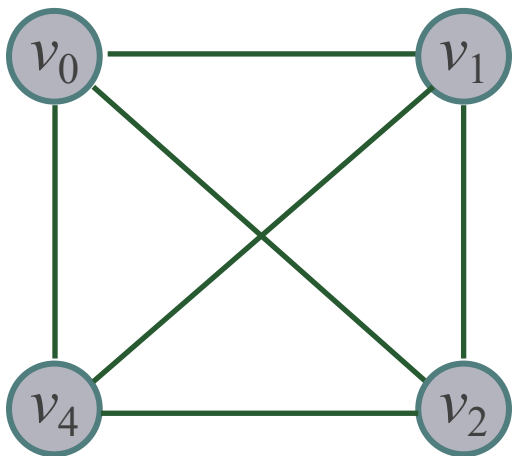


- 线性结构中，数据元素之间具有线性关系，逻辑关系表现为前驱-后继；
- 树结构中，结点之间具有层次关系，逻辑关系表现为双亲-孩子
- 图结构中，任意两个顶点之间都可能有关系，逻辑关系表现为邻接



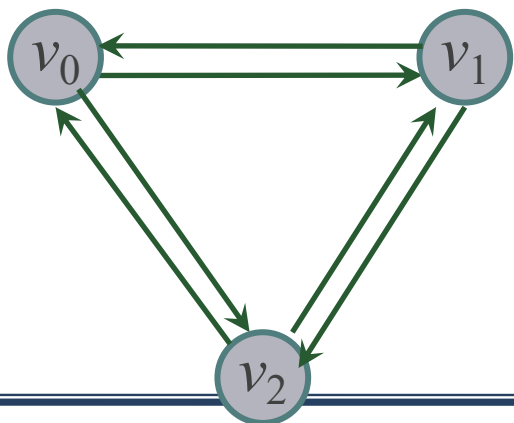
4. 图的基本术语

完全图



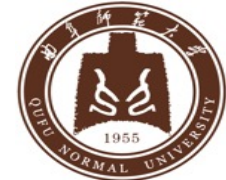
📌 无向完全图：无向图中，任意两个顶点之间都存在边

🕒 n 个顶点的无向完全图有多少条边？ $\Rightarrow n \times (n-1)/2$



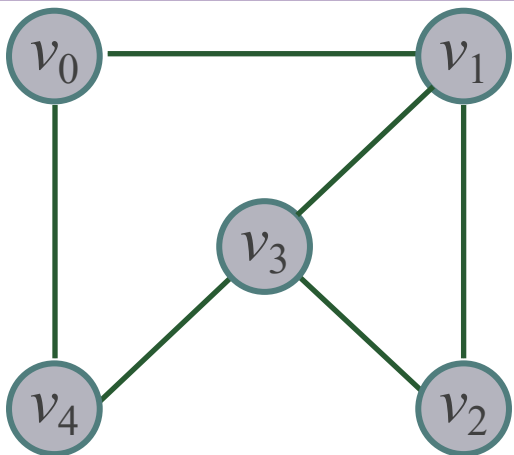
📌 有向完全图：有向图中，任意两个顶点之间都存在方向相反的两条弧

🕒 n 个顶点的有向完全图有多少条弧？ $\Rightarrow n \times (n-1)$



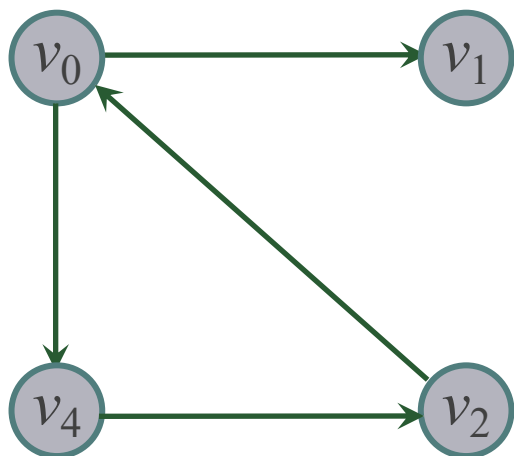
4. 图的基本术语

度、入度和出度



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 $TD(v)$

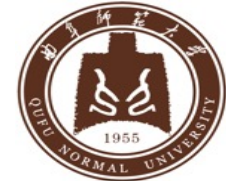
$$TD(v_0) = 2, \quad TD(v_3) = 3$$



顶点的入度：在**有向图**中，顶点 v 的入度是指以该顶点为弧头的弧的数目，记为 $ID(v)$ ；

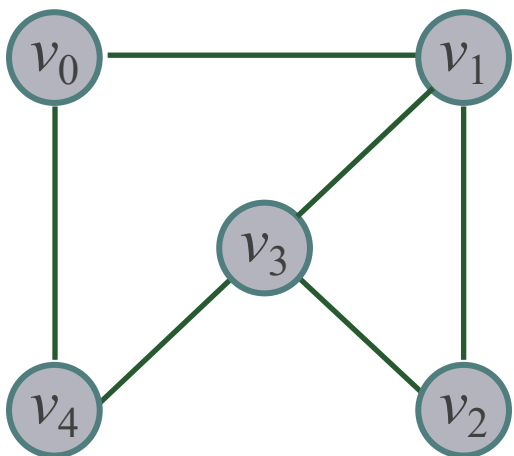
顶点的出度：在**有向图**中，顶点 v 的出度是指以该顶点为弧尾的弧的数目，记为 $OD(v)$ ；

$$ID(v_0) = 1, \quad OD(v_0) = 2,$$



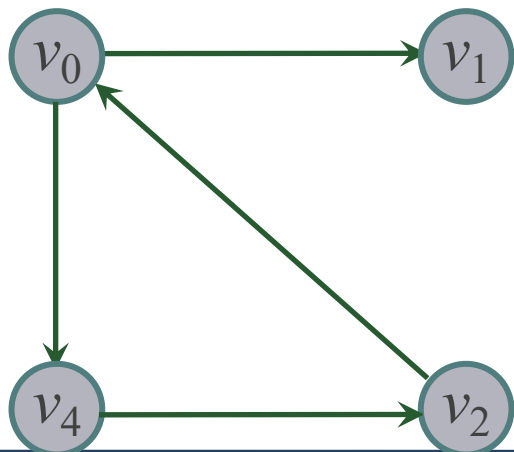
4. 图的基本术语

度、入度和出度



在具有 n 个顶点、 e 条边的**无向图**中，各顶点的度之和与边数之和有什么关系？

$$\sum_{i=0}^{n-1} TD(v_i) = 2e$$



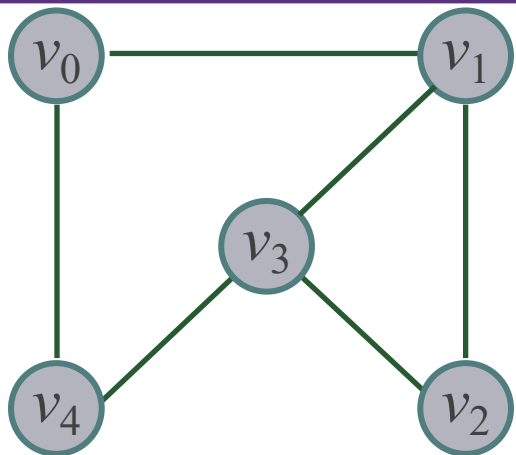
在具有 n 个顶点、 e 条边的**有向图**中，各顶点的入度之和与各顶点的出度之和有什么关系？与边数之和有什么关系？

$$\sum_{i=0}^{n-1} ID(v_i) = \sum_{i=0}^{n-1} OD(v_i) = e$$



4. 图的基本术语

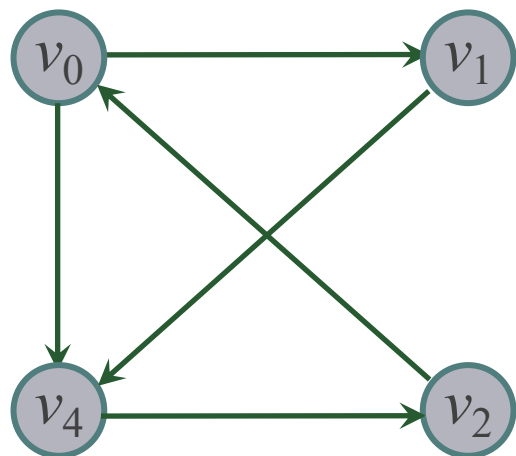
路径、回路



路径：在**无向图**中，顶点 v_p 和顶点 v_q **之间**的路径是一个**顶点序列** $(v_p=v_{i0}, v_{i1}, \dots, v_{im}=v_q)$ ，其中 $(v_{ij-1}, v_{ij}) \in E (1 \leq j \leq m)$

顶点 v_0 和顶点 v_4 之间的路径：

$v_0 v_4$ 、 $v_0 v_1 v_3 v_4$ 、 $v_0 v_1 v_2 v_3 v_4$ 、



回路（环）：第一个顶点和最后一个顶点相同的路径



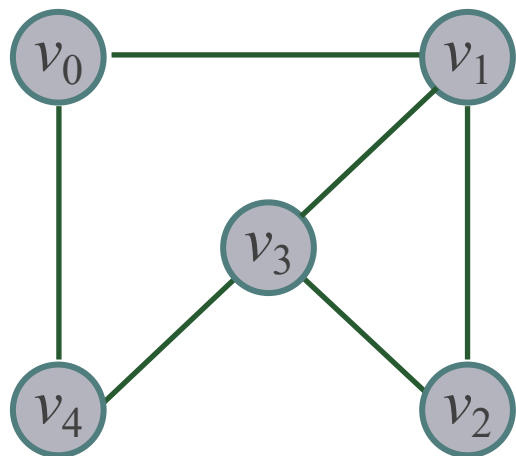
路径：在**有向图**中，**从**顶点 v_p **到**顶点 v_q 的路径是一个**顶点序列** $(v_p=v_{i0}, v_{i1}, \dots, v_{im}=v_q)$ ，其中 $\langle v_{ij-1}, v_{ij} \rangle \in E (1 \leq j \leq m)$

从顶点 v_0 到顶点 v_4 的路径： $v_0 v_4$ 、 $v_0 v_1 v_4$



4. 图的基本术语

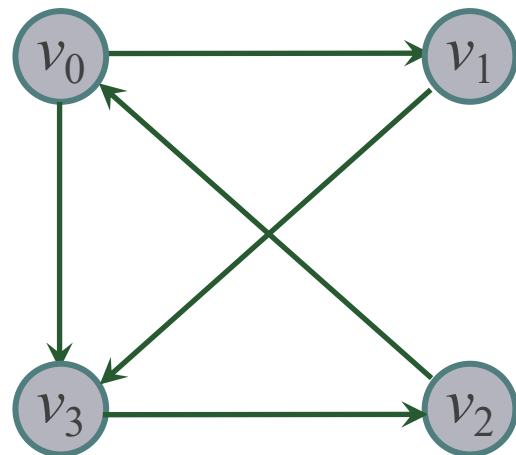
路径、回路



简单路径：序列中顶点不重复出现的路径



简单回路（简单环）：除了第一个顶点和最后一个顶点外，其余顶点不重复出现的回路。

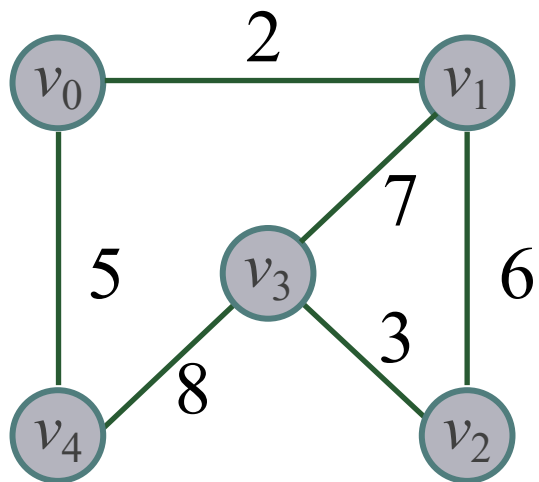
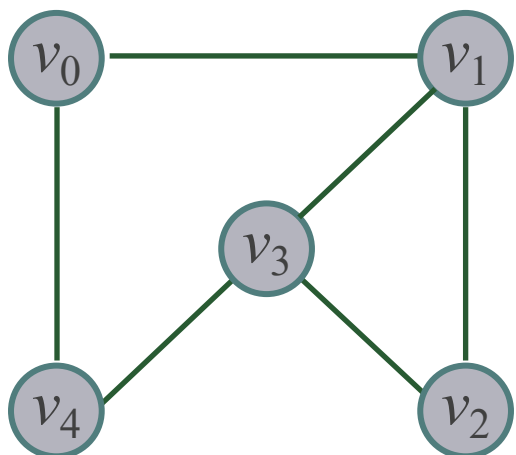


不致混淆的情况下，路径和回路都是简单的



4. 图的基本术语

路径、回路



路径长度：非带权图——路径上**边**的个数

路径长度：带权图——路径上边的**权值之和**

顶点 v_0 和顶点 v_4 之间的路径长度：

$v_0 v_4$: 1

$v_0 v_4$: 5

$v_0 v_1 v_3 v_4$: 3

$v_0 v_1 v_3 v_4$: 17

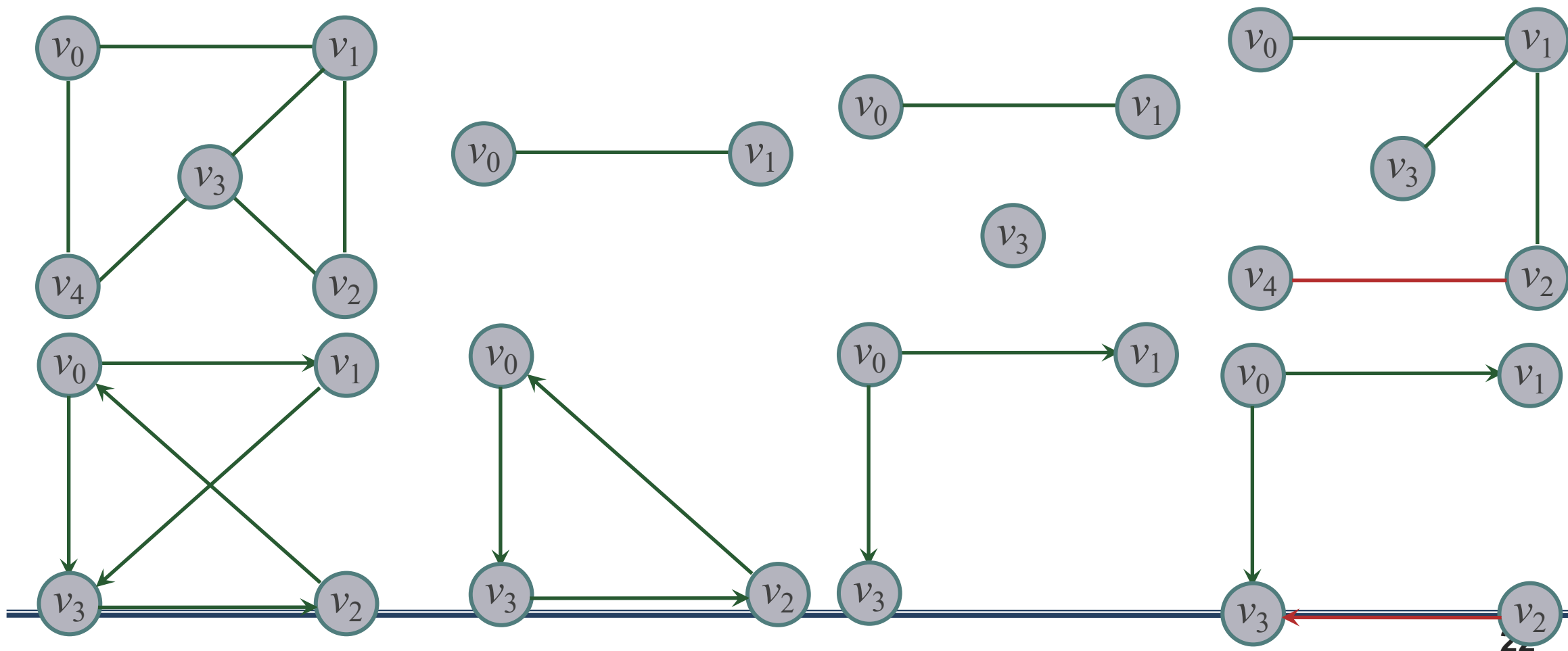
$v_0 v_1 v_2 v_3 v_4$: 4

$v_0 v_1 v_2 v_3 v_4$: 19

4. 图的基本术语

子图

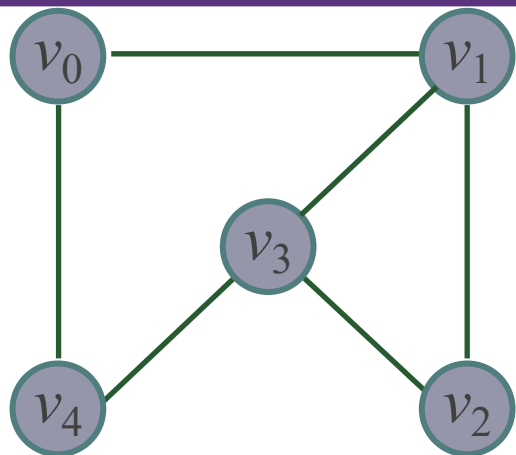
 **子图**: 若图 $G=(V, E)$, $G'=(V', E')$, 如果 $V'\subseteq V$ 且 $E'\subseteq E$, 则称图 G' 是 G 的子图





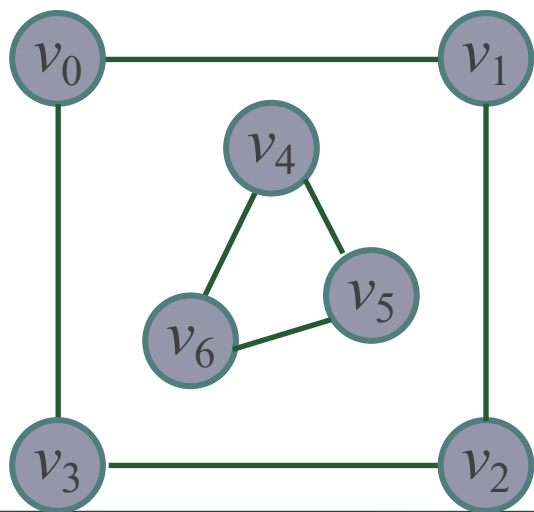
4. 图的基本术语

连通图



📌 **连通顶点**: 在**无向图**中, 如果顶点 v_i 和顶点 $v_j (i \neq j)$ 之间有**路径**, 则称顶点 v_i 和 v_j 是连通的

📌 **连通图**: 在**无向图**中, 如果任意两个顶点都是连通的, 则称该无向图是连通图



🕒 对于非连通图, 实际处理中会有什么问题?

📎 从某顶点出发进行遍历, 有一些顶点访问不到

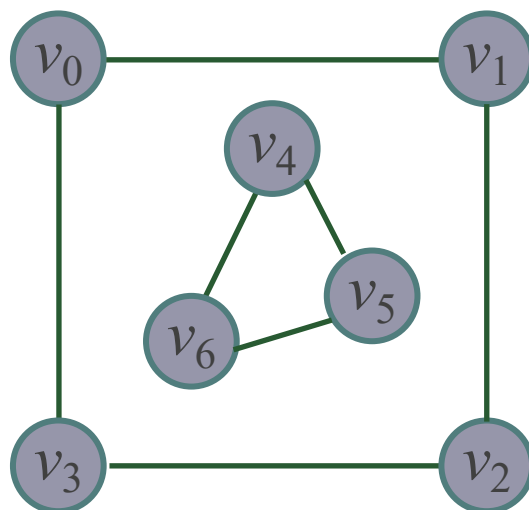


4. 图的基本术语

连通分量

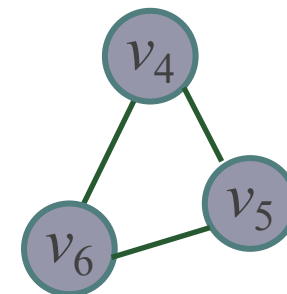
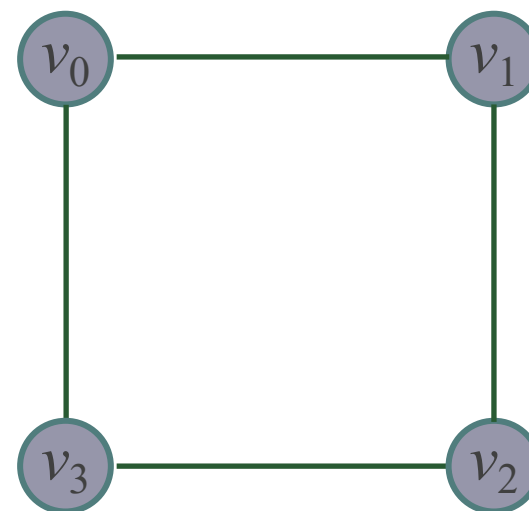
📌 连通分量：非连通图的极大连通子图

含有极大顶点数
依附于这些顶点的所有边



连通分量1

连通分量2

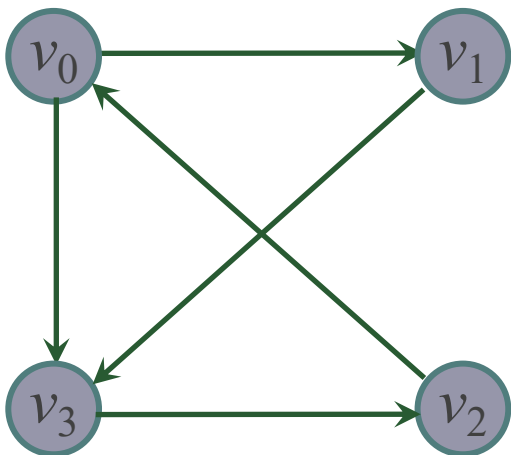
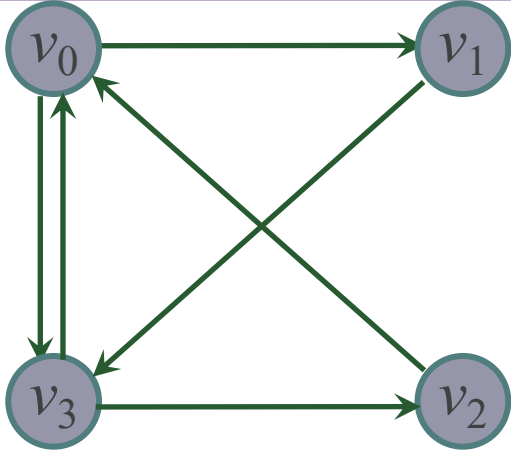


连通分量是对无向图的一种划分

6.2 图的逻辑结构

6-2-1 图的定义与基本术语

4. 图的基本术语

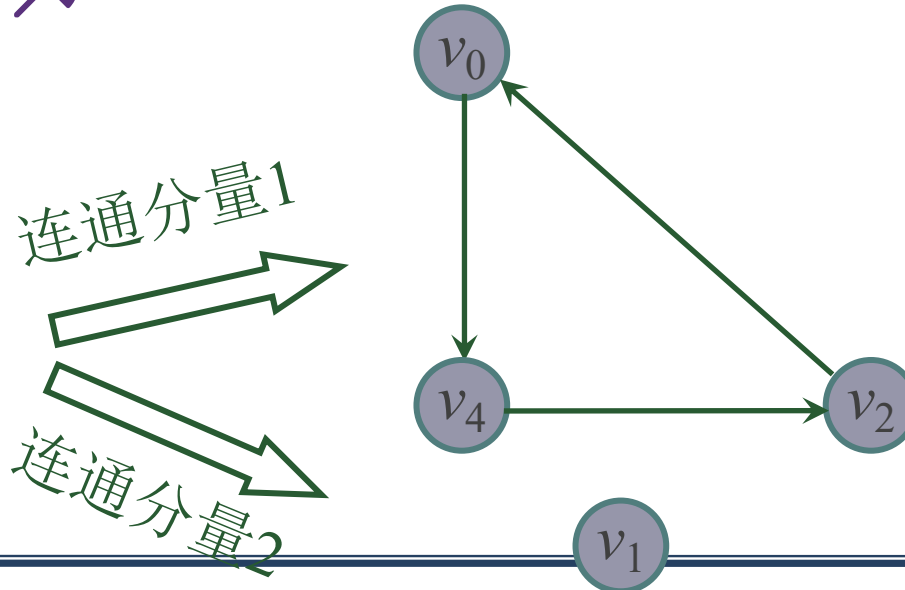


强连通图、强连通分量

强连通顶点：在**有向图**中，如果从顶点 v_i 到顶点 v_j 和从顶点 v_j 到顶点 v_i 均有路径，则称顶点 v_i 和 v_j 是强连通的

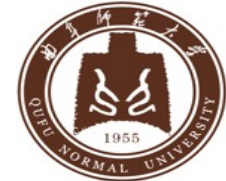
强连通图：在**有向图**中，如果任意两个顶点都是强连通的，则称该有向图是强连通图

强连通分量：非强连通图的**极大连通子图**



6.2 图的逻辑结构

6-2-2 图的抽象数据类型定义



1. 图的ADT定义

图是一种与具体应用密切相关的数据结构，基本操作有很大差别

ADT Graph

DataModel

顶点的有穷非空集合和顶点之间边的集合

Operation

CreatGraph: 图的建立

DestroyGraph: 图的销毁

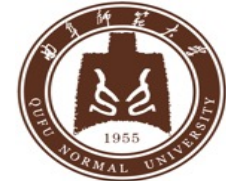
DFTraverse: 深度优先遍历图

BFTraverse: 广度优先遍历图



简单起见，只讨论图的遍历

endADT



2. 图的遍历

✚ 图的遍历：从图中某一顶点出发访问图中所有顶点，并且每个结点仅被访问一次



抽象操作，可以是对顶点的各种处理，这里简化为输出顶点的数据



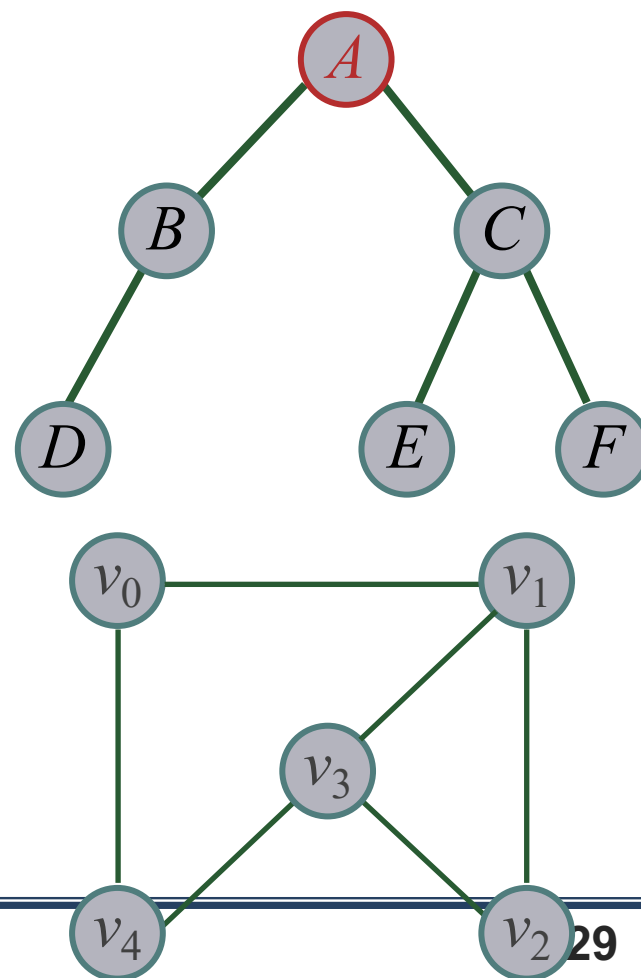
2. 图的遍历

✚ 图的遍历：从图中**某**一顶点出发**访问**图中所有顶点，并且每个结点仅被访问一次

🕒 在图中，如何选取遍历的起始顶点？



🚀 解决方案：将图中的顶点按**任意**顺序排列起来，从编号最小的顶点开始





2. 图的遍历

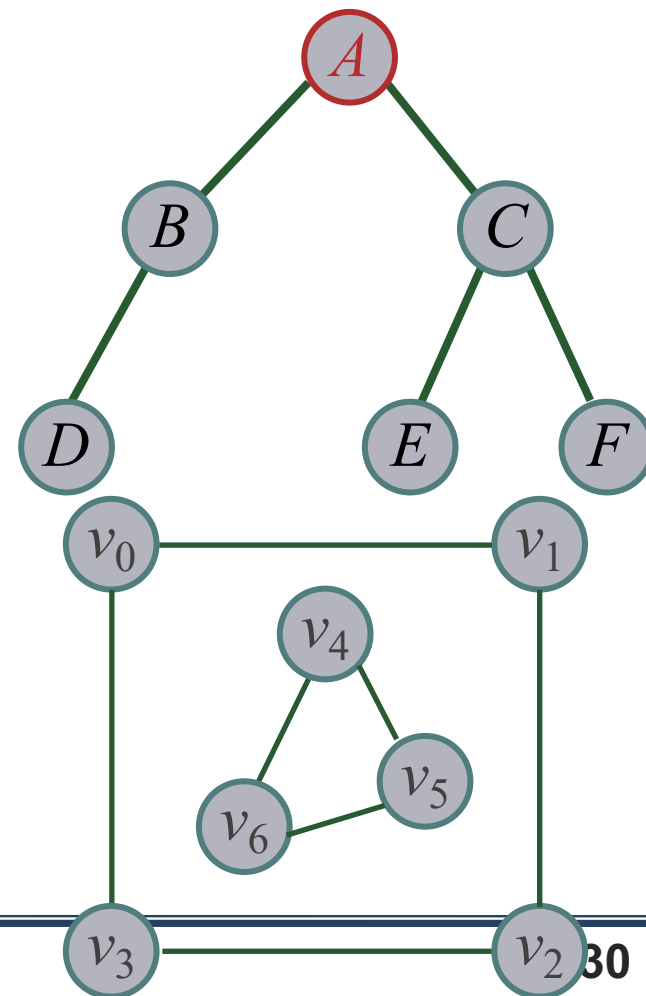
✚ 图的遍历：从图中某一顶点出发访问图中所有顶点，并且每个结点仅被访问一次

🕒 从某顶点出发能访问其他所有顶点吗？



📄 解决方案：多次调用图遍历算法

下面仅讨论从某顶点出发遍历图的算法





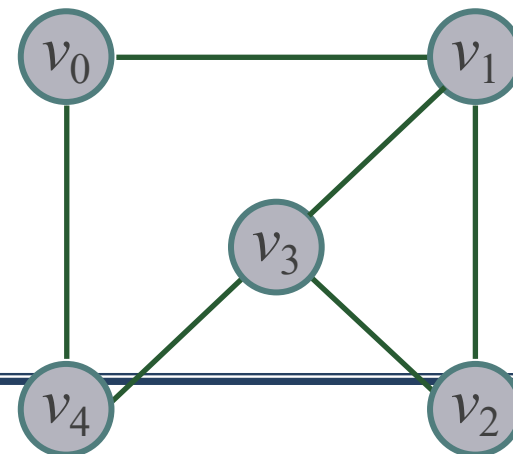
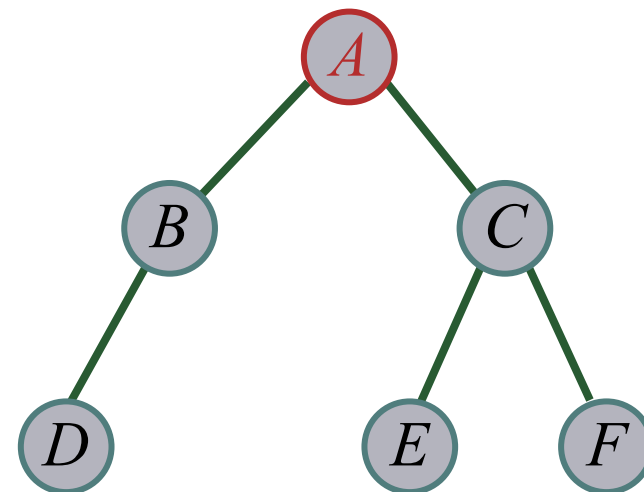
2. 图的遍历

✚ 图的遍历：从图中某一顶点出发访问图中所有顶点，并且每个结点仅被访问一次

🕒 如何避免遍历不会因回路而陷入死循环？



📌 解决方案：附设访问标志数组visited[n]



6.2 图的逻辑结构

6-2-2 图的抽象数据类型定义



2. 图的遍历

✚ 图的遍历：从图中**某**一顶点出发**访问**图中所有顶点，并且每个结点仅被访问一次

🕒 采用什么次序依次访问图中所有顶点？

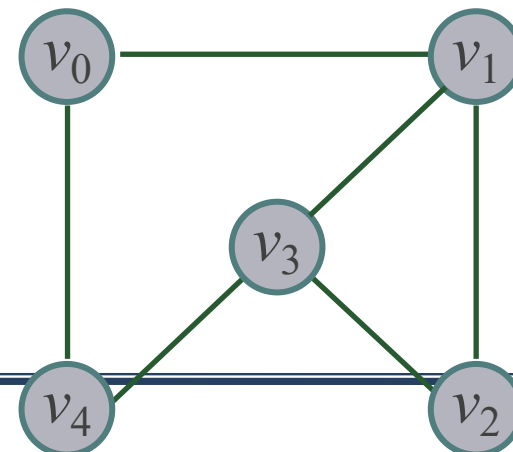
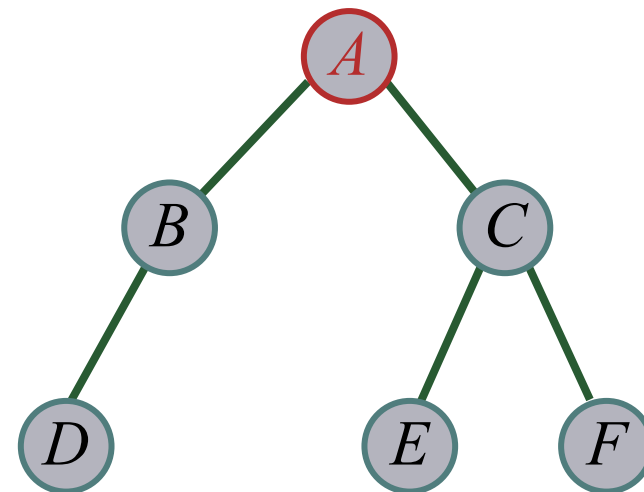


线性序： $a_1 a_2 \dots a_n$

前序： $A B D C E F$

中序： $D B A E C F$

后序： $D B E F C A$



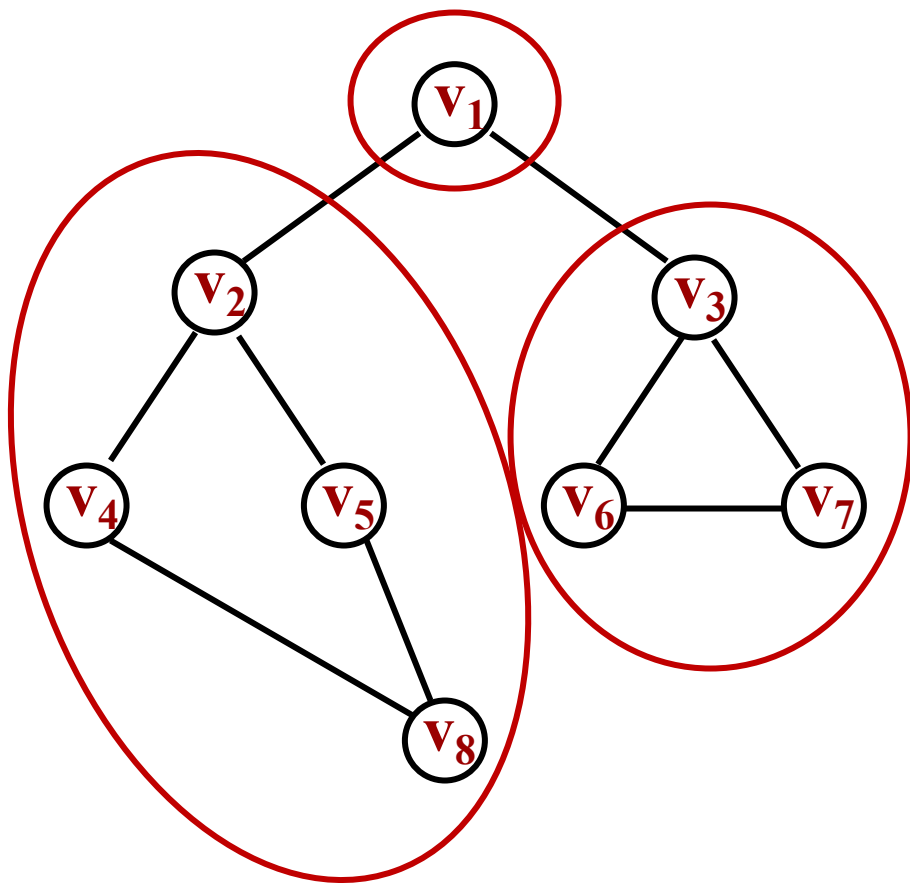
🚀 解决方案： **深度**优先遍历和**广度**优先遍历

6.2 图的逻辑结构

6-2-3 图的遍历操作



1. 图的深度优先遍历



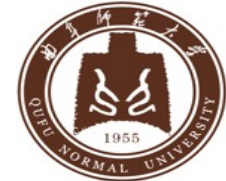
图可分为三部分:

基结点

第一个邻接结点导出的子图

其它邻接顶点导出的子图

深度优先搜索类似于树的先序遍历

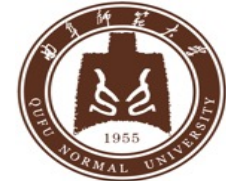


1. 图的深度优先遍历

算法描述:

- 1). 从图中某个顶点 **v** 出发，访问此顶点；
- 2). 然后依次从 **v** 的未被访问的邻接点出发进行深度优先遍历；
- 3). 直至图中所有和 **v** 有路径相通的顶点都被访问到。
- 4). 若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点做起始点，重复上述过程，直至图中所有顶点都被访问到。

深度优先遍历是一个递归的过程



1. 图的深度优先遍历

 用伪代码描述深度优先遍历的操作定义

算法: DFTraverse

输入: 顶点的编号 v

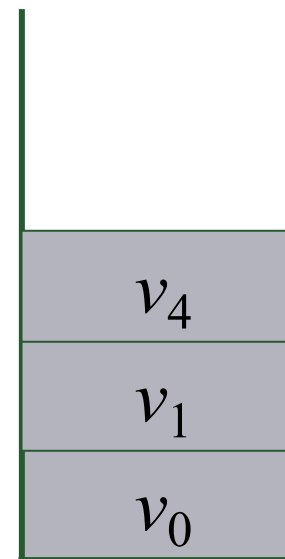
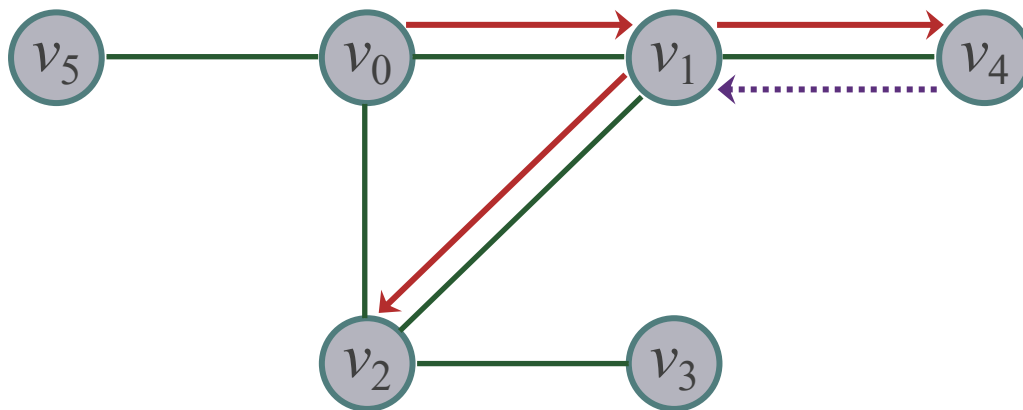
输出: 无

1. 访问顶点 v ; 修改标志 $visited[v] = 1$;
2. $w =$ 顶点 v 的第一个邻接点;
3. while (w 存在)
 - 3.1 if (w 未被访问) 从顶点 w 出发递归执行该算法;
 - 3.2 $w =$ 顶点 v 的下一个邻接点;



1. 图的深度优先遍历

 运行实例——深入理解操作过程

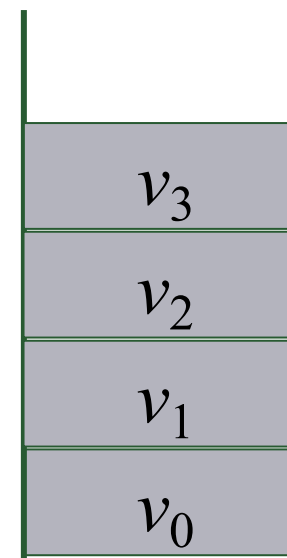
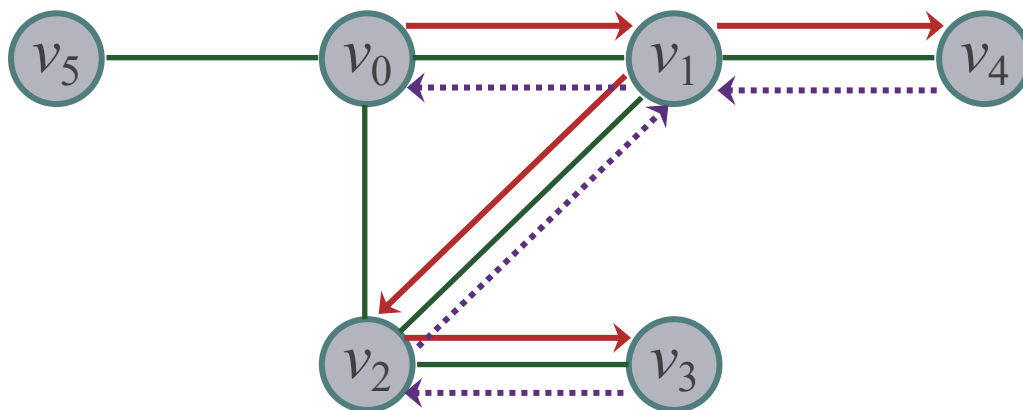


深度优先遍历序列: v_0 v_1 v_4



1. 图的深度优先遍历

 运行实例——深入理解操作过程

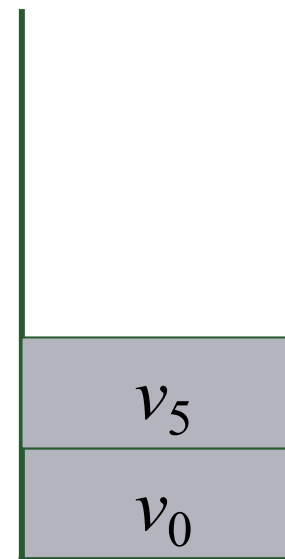
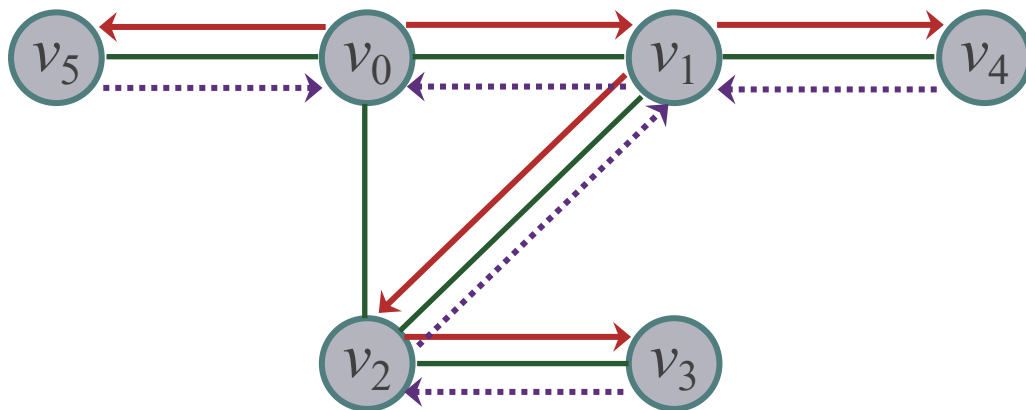


深度优先遍历序列: v_0 v_1 v_4 v_2 v_3



1. 图的深度优先遍历

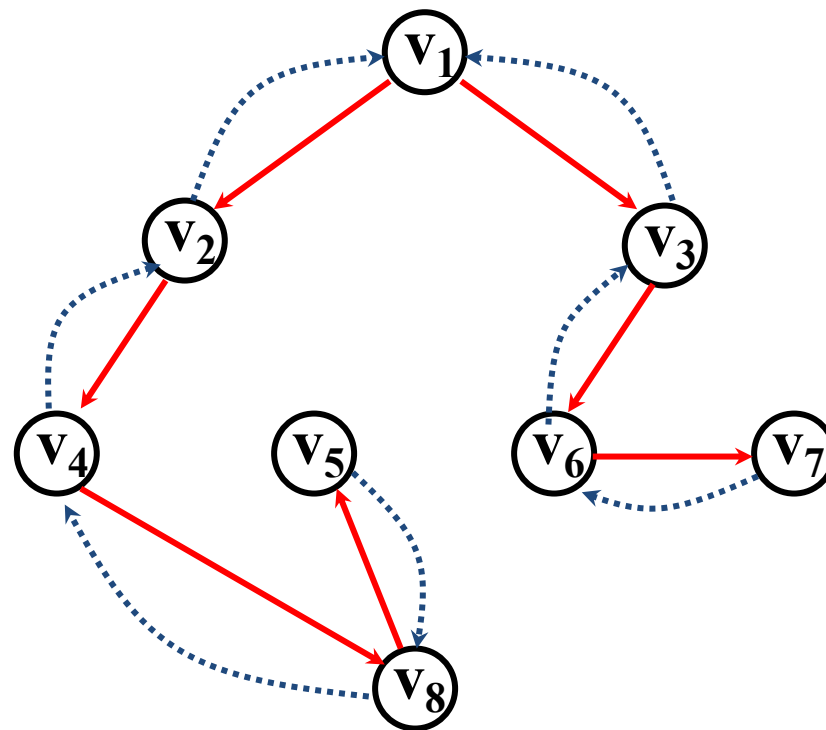
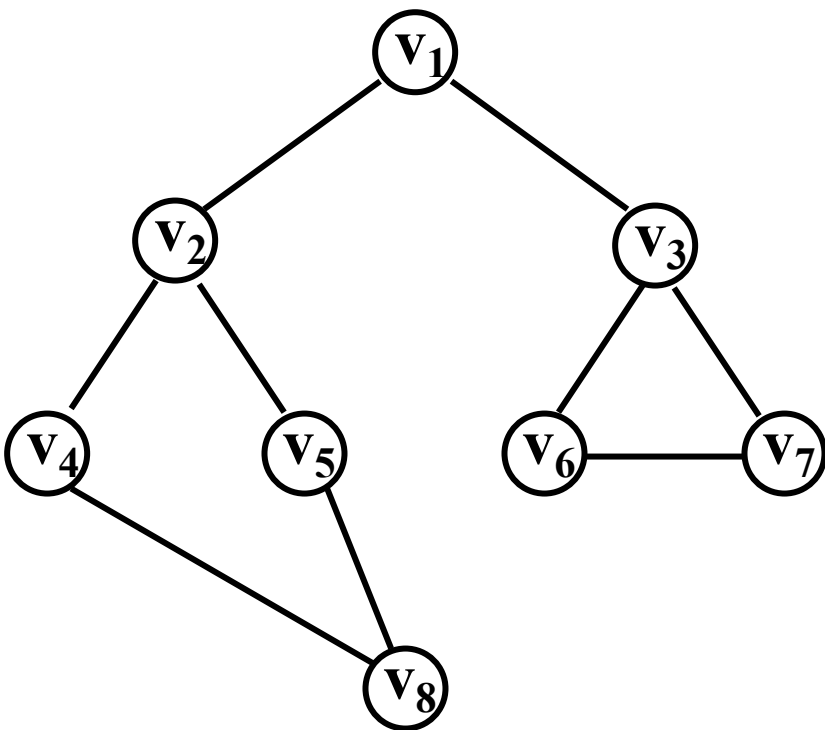
 运行实例——深入理解操作过程



深度优先遍历序列: $v_0 \ v_1 \ v_4 \ v_2 \ v_3 \ v_5$



1. 图的深度优先遍历



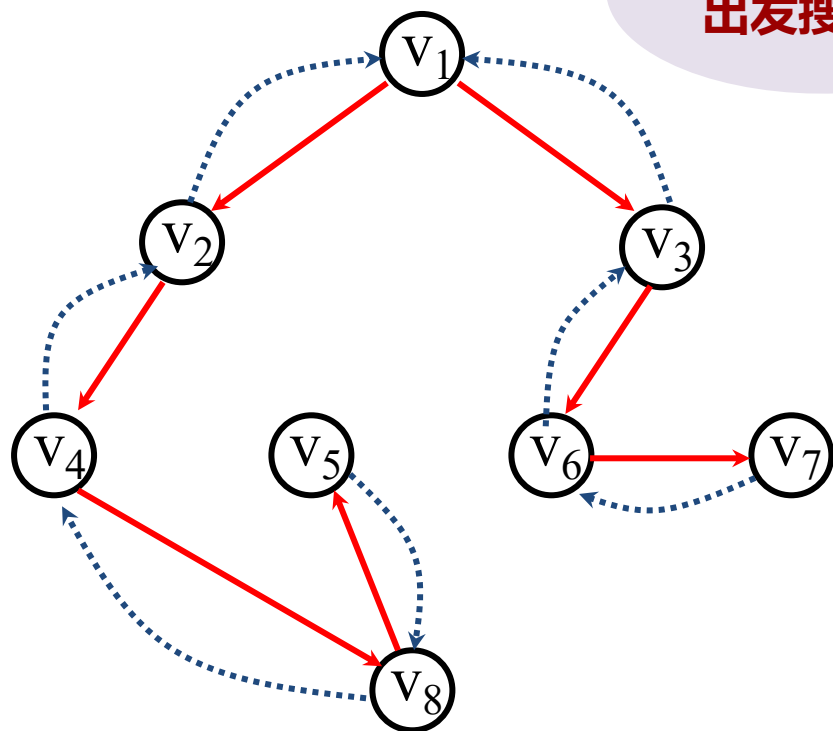
深度优先搜索顺序:

v_1 v_2 v_4 v_8 v_5 v_3 v_6 v_7



1. 图的深度优先遍历

栈实现深度优先搜索



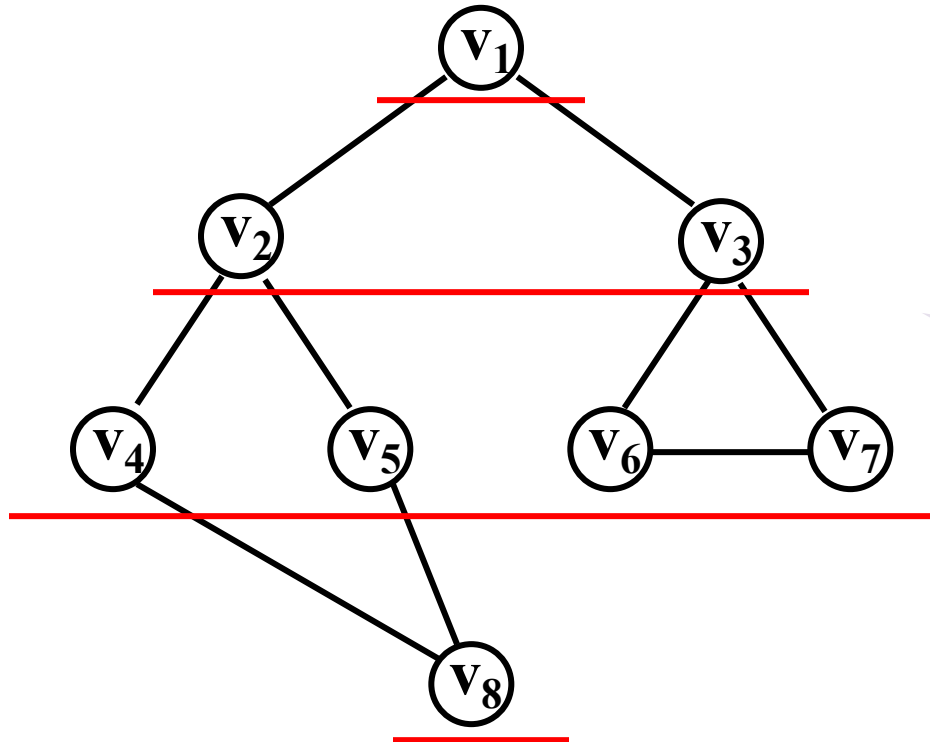
深度优先搜索顺序:

V_1 V_2 V_4 V_8 V_5 V_3 V_6 V_7

6.2 图的逻辑结构

6-2-3 图的遍历操作

2. 图的广度优先遍历



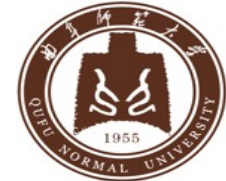
把图人为的分层，
按层遍历。

只有父辈结点被
访问后才会访问
子孙结点！


广度优先搜索顺序：

广度优先搜索类似于树的层次遍历

$V_1 \ V_2 \ V_3 \ V_4 \ V_5 \ V_6 \ V_7 \ V_8$



2. 图的广度优先遍历

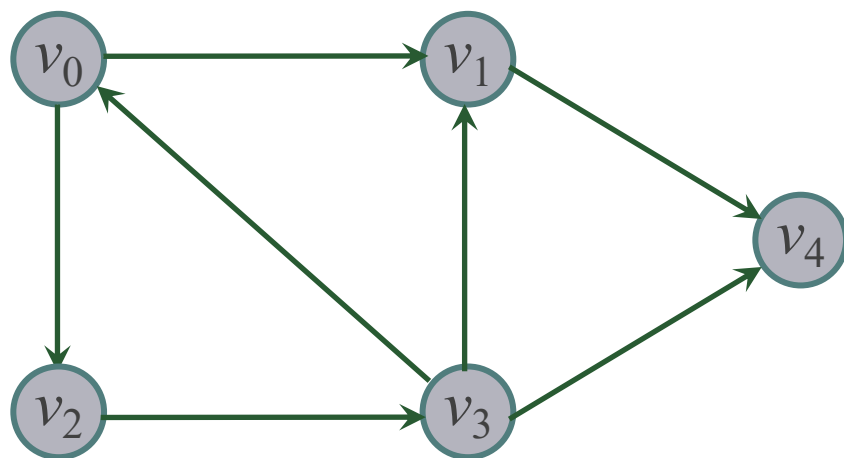
-  从顶点 v 出发进行广度优先遍历的基本思想是：
1. 从图中某个顶点 v 出发，访问此顶点；
 2. 然后依次访问 v 的各个未曾访问的邻接点；
 3. 然后依次从这些邻接点出发再依次访问它们的邻接点；
 4. 直至图中所有和 v 有路径相通的顶点都被访问到。
 5. 若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点做起始点，重复上述过程，直至图中所有顶点都被访问到。

“先被访问顶点的邻接点” 先于 “后被访问顶点的邻接点”



2. 图的广度优先遍历

 运行实例——深入理解操作过程



v_0	v_1	v_2
-------	-------	-------

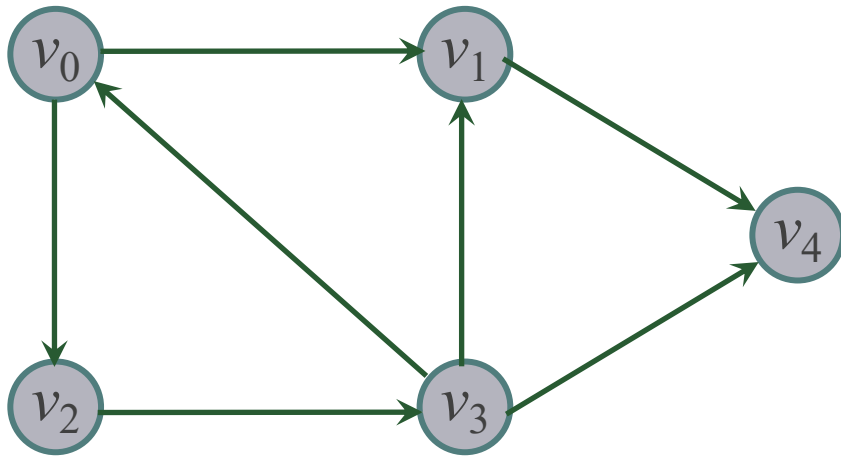
广度优先遍历序列: v_0 v_1 v_2

6.2 图的逻辑结构

6-2-3 图的遍历操作

2. 图的广度优先遍历

 运行实例——深入理解操作过程



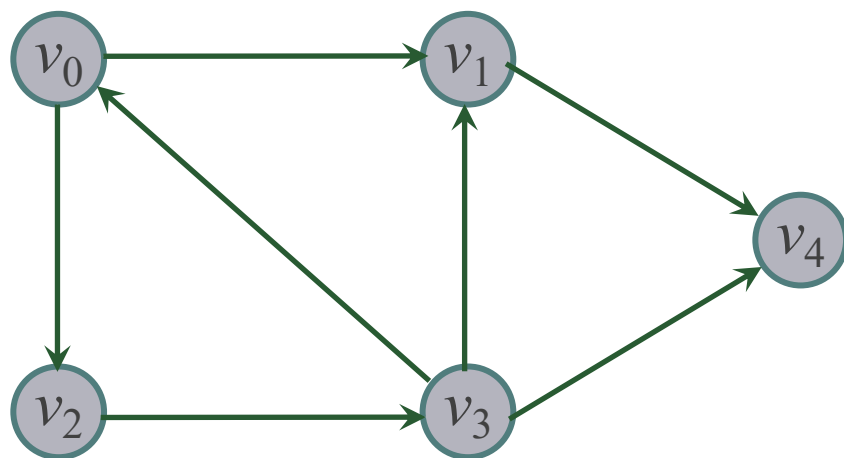
v_1 v_2 v_4 v_3

广度优先遍历序列: v_0 v_1 v_2 v_4 v_3



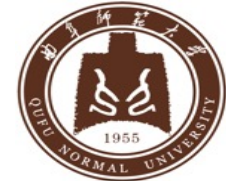
2. 图的广度优先遍历

 运行实例——深入理解操作过程



v_4 v_3

广度优先遍历序列: v_0 v_1 v_2 v_4 v_3



2. 图的广度优先遍历



用伪代码描述广度优先遍历的操作定义

算法: BFTraverse

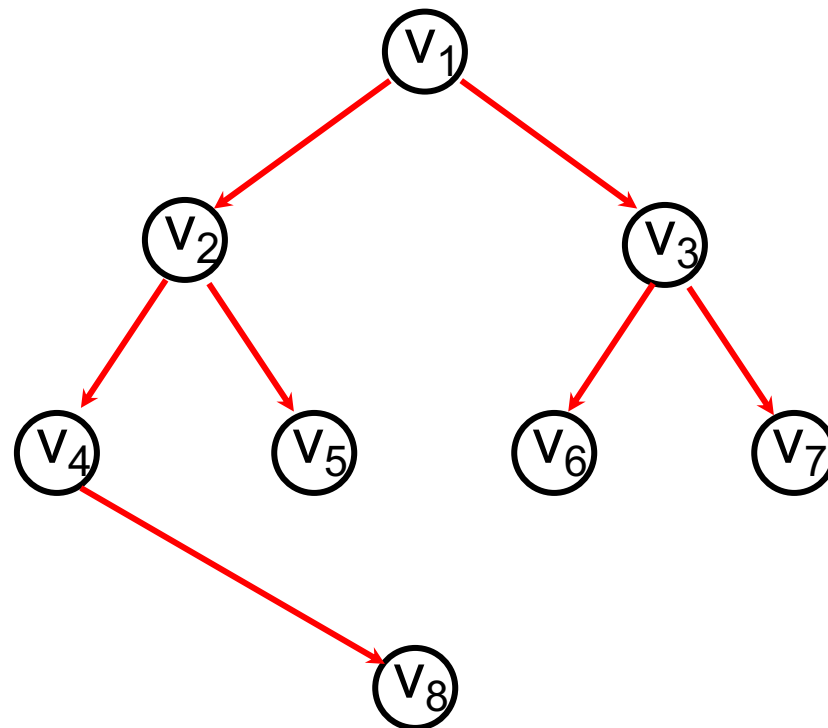
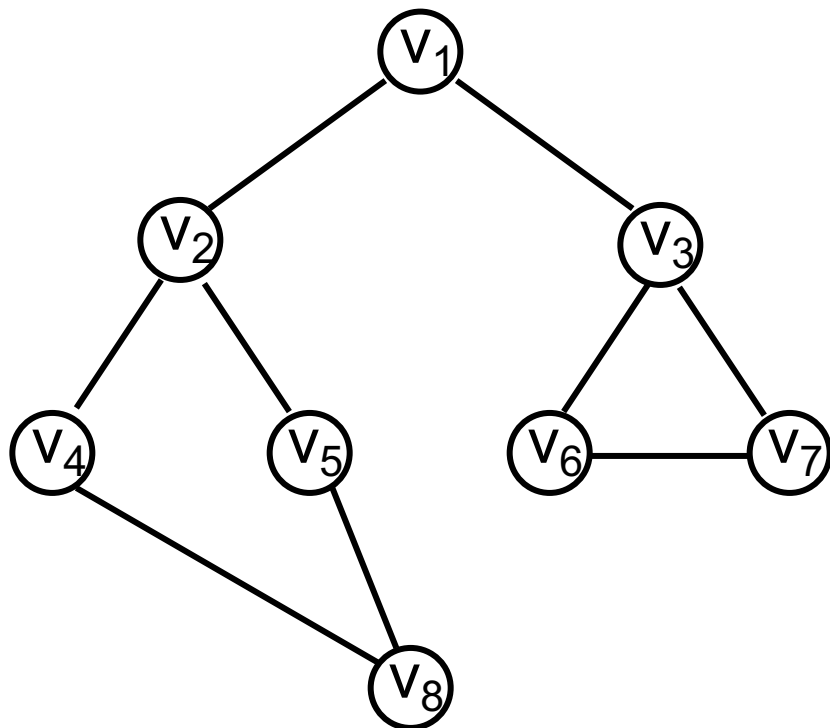
输入: 顶点的编号 v

输出: 无

1. 队列 Q 初始化;
2. 访问顶点 v ; 修改标志 $\text{visited}[v] = 1$; 顶点 v 入队列 Q ;
3. while (队列 Q 非空)
 - 3.1 v = 队列 Q 的队头元素出队;
 - 3.2 w = 顶点 v 的第一个邻接点;
 - 3.3 while (w 存在)
 - 3.3.1 如果 w 未被访问, 则
访问顶点 w ; 修改标志 $\text{visited}[w] = 1$; 顶点 w 入队列 Q ;
 - 3.3.2 w = 顶点 v 的下一个邻接点;



2. 图的广度优先遍历



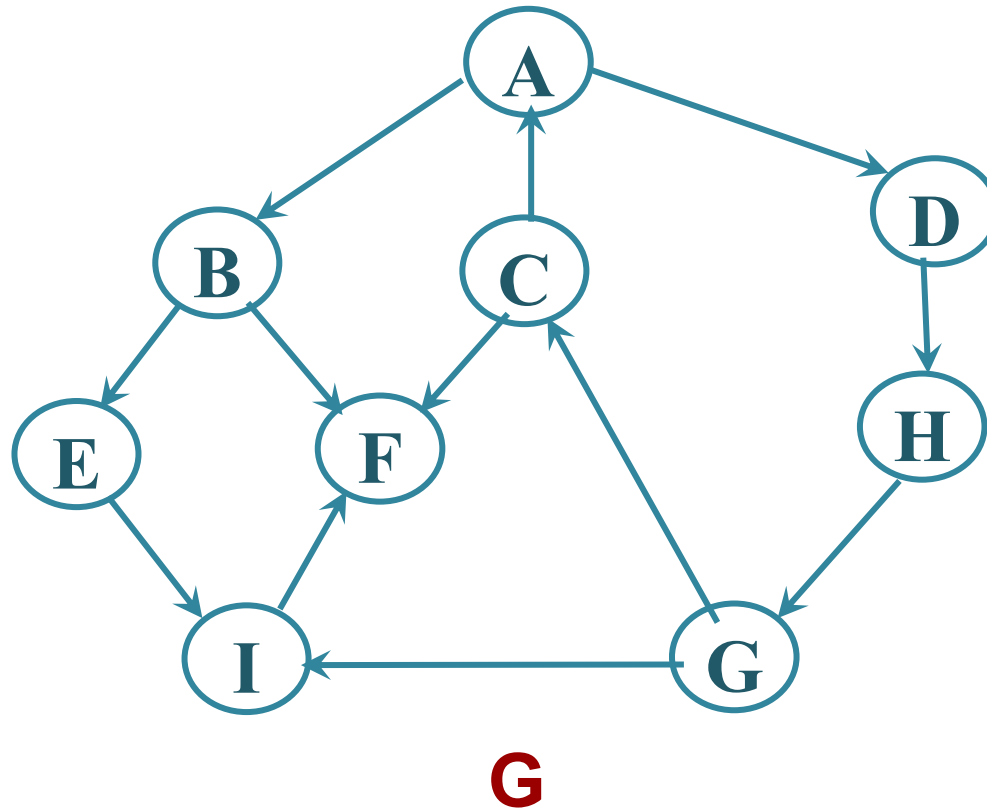
广度优先搜索顺序: $V_1 V_2 V_3 V_4 V_5 V_6 V_7 V_8$

小结

1. 掌握图的定义和基本术语
2. 理解图的抽象数据类型定义
3. 掌握图的深度/广度优先遍历方法

作业

1. 有向图G如下图所示，请从顶点A出发开始搜索，分别写出一个深度优先搜索遍历序列和一个广度优先搜索遍历序列。





Thank You !

Q & A