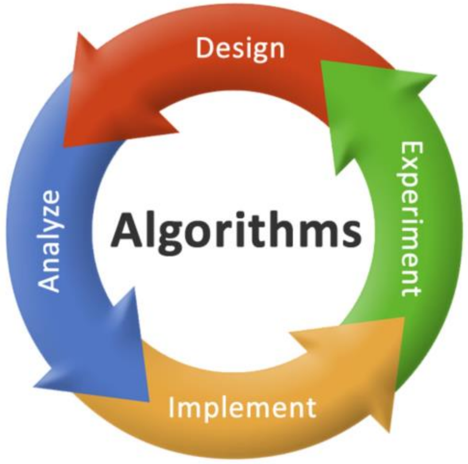


Algorithm Design and Analysis

算法设计与分析

Chapter 2: Graph Search

张乾坤

课程提要

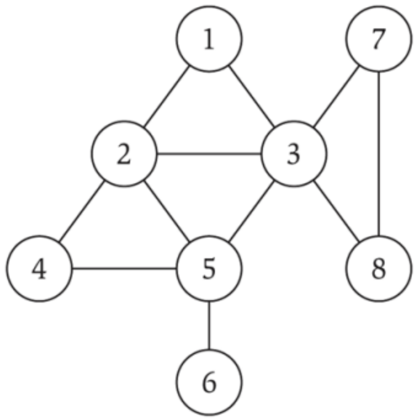
- 回顾：图的基本定义
- 图连通与图遍历
- 有向无环图和拓扑排序

课程提要

- 回顾：图的基本定义
- 图连通与图遍历
- 有向无环图和拓扑排序

无向图

- $G = (V, E)$
- $\boldsymbol{V}$: Vertices/Nodes点集； $\boldsymbol{E}$: Edges边集
- 描述图大小的参数：$n = |V|, m = |E|$



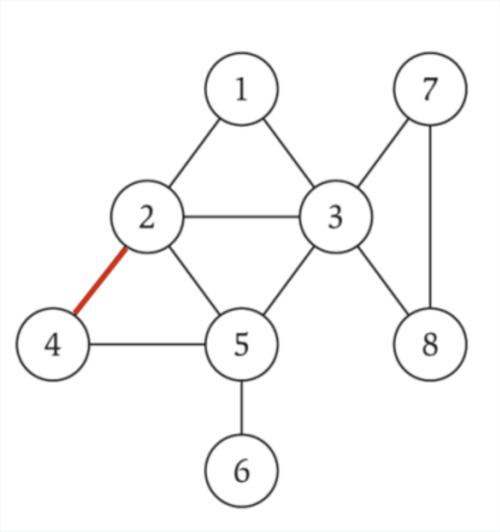
$V = \{\ 1, 2, 3, 4, 5, 6, 7, 8\ \}$

$E = \{\ 1\text{–}2, 1\text{–}3, 2\text{–}3, 2\text{–}4, 2\text{–}5, 3\text{–}5, 3\text{–}7, 3\text{–}8, 4\text{–}5, 5\text{–}6, 7\text{–}8\ \}$

$m = 11, n = 8$

图的表示

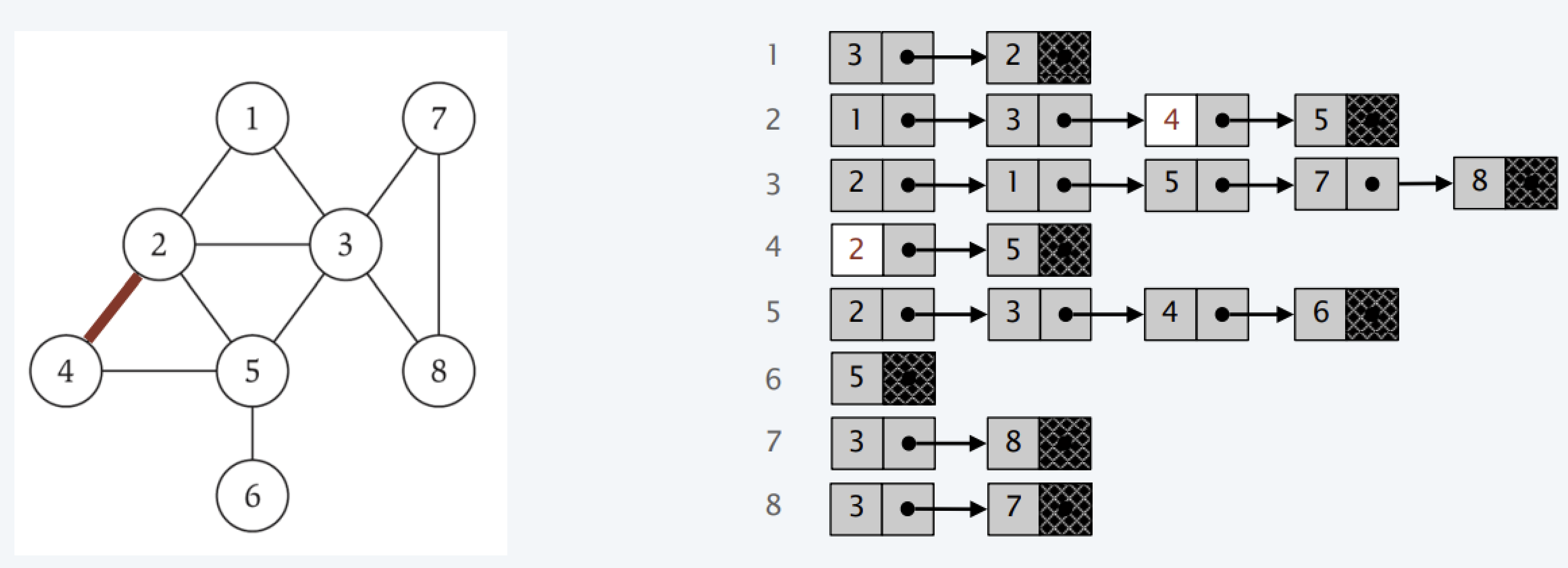
- 邻接矩阵： $n \times n$矩阵，若边(u,v)存在，则$A_{uv}=1$
- 复杂度：空间复杂度 $\Theta(n^2)$
- 判断边(u,v)是否存在？判断所有的边？ $\Theta(1)$; $\Theta(n^2)$



	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

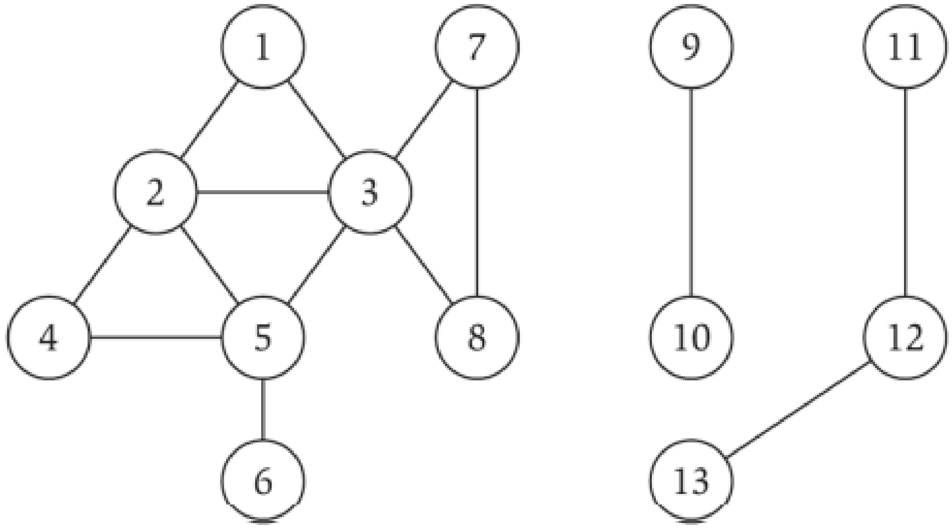
图的表示

- 邻接表： 空间复杂度 $\Theta(m+n)$
- 判断边(u,v)是否存在？查询所有的边？ $O(degree(u))$; $\Theta(m+n)$



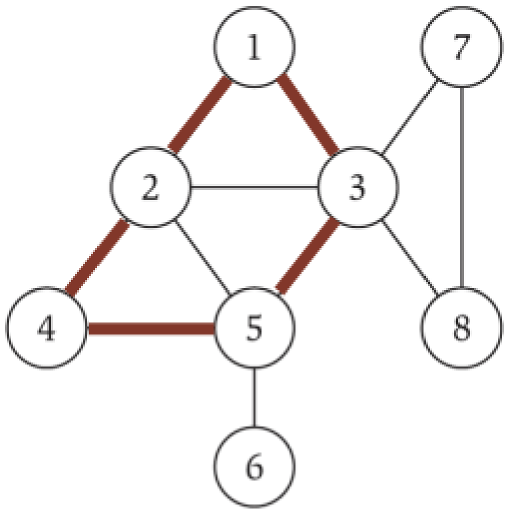
路径Path和连通性Connectivity

- 路径：一个点的序列$v_1, v_2, \ldots, v_k$，满足v_i和$v_{\mathrm{i}+1}$被不同的边连接
- 连通：对任意的两个点都至少存在一条路径



环Cycles

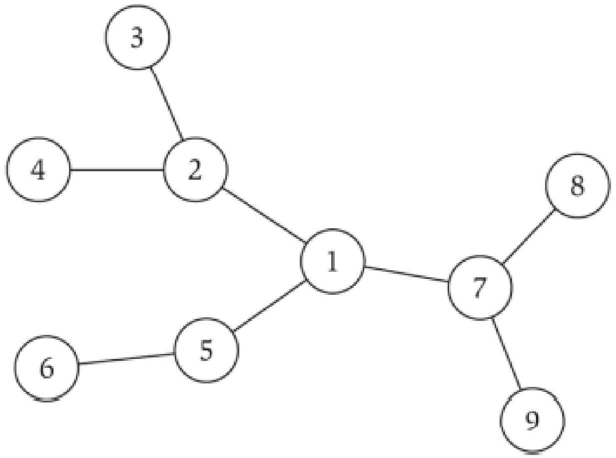
- 环： 首尾相接的路径



cycle C = 1-2-4-5-3-1

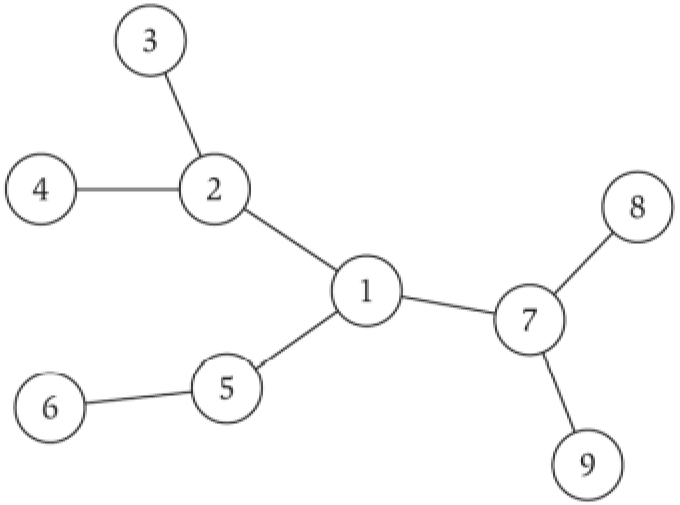
树Trees

- 树：无回路的无向连通图
- 定理：假设G是一个有n个点的来无向图，以下任意两个可推出第三：
  - ✓G连通
  - ✓G不包含环
  - ✓G包含$n-1$条边

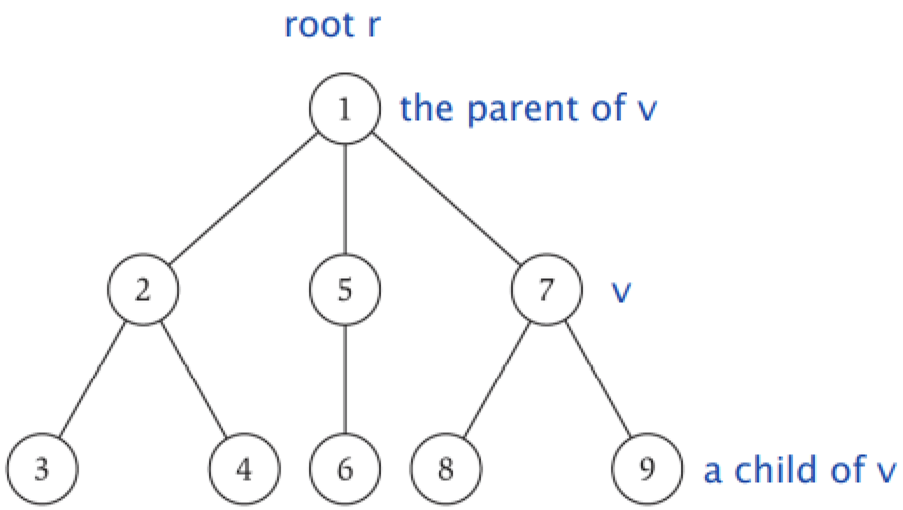


树Trees

- 根节点、父节点、子节点、祖先



a tree



the same tree, rooted at 1

课程提要

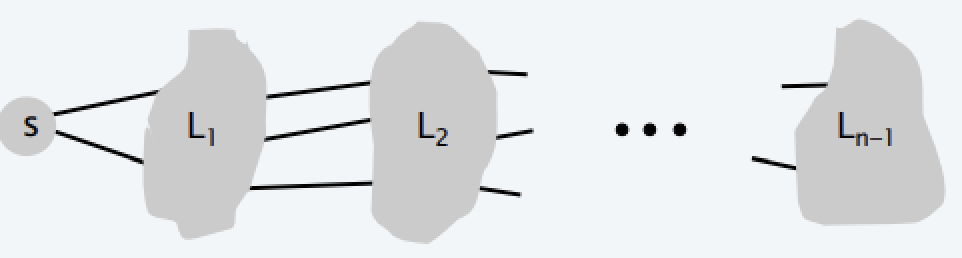
- 回顾：图的基本定义
- 图连通与图遍历
- 有向无环图和拓扑排序

图的连通性

- s-t连通性问题：给定两个节点s和t判断是否包含路径
- s-t最短路径问题：给定两个节点s和t求他们的最短路径

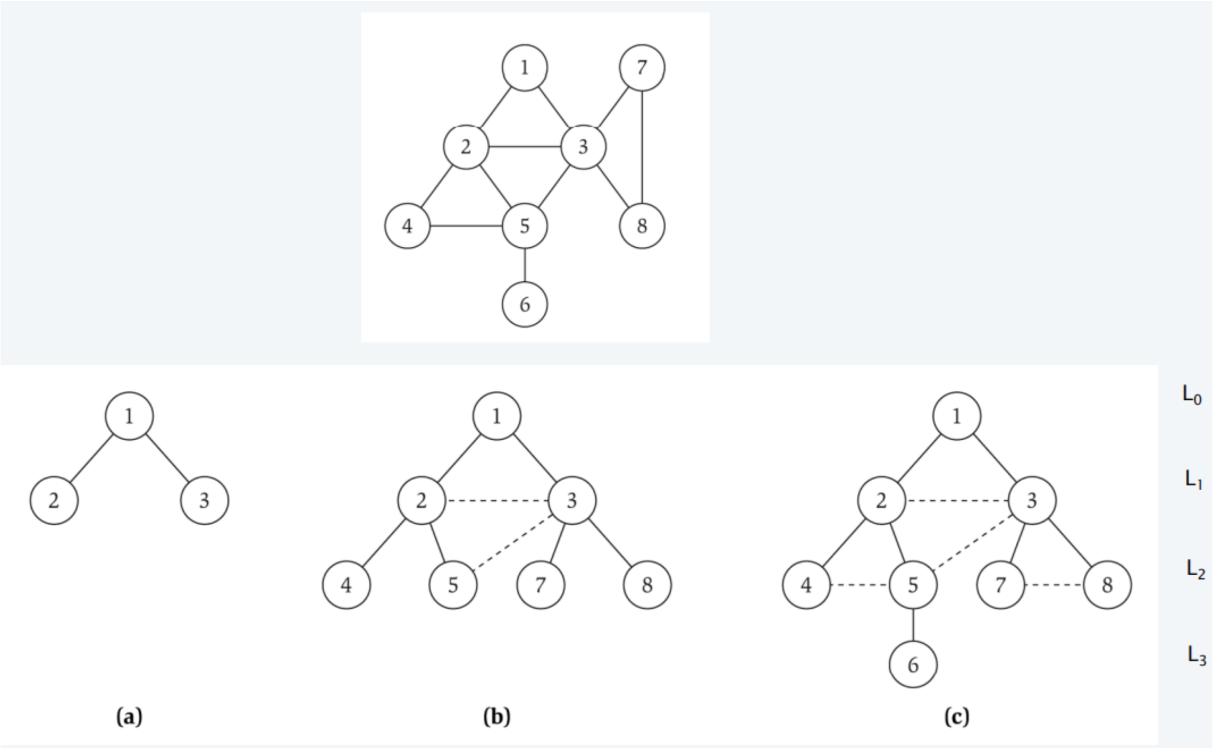
广度优先搜索Breadth-first search

- 思想：全面出击，层层突破
- BFS算法：



- 对任意的i，L_i包含所有到s距离恰为i的节点
- s和t存在路径当且仅当t出现在某个L_i中

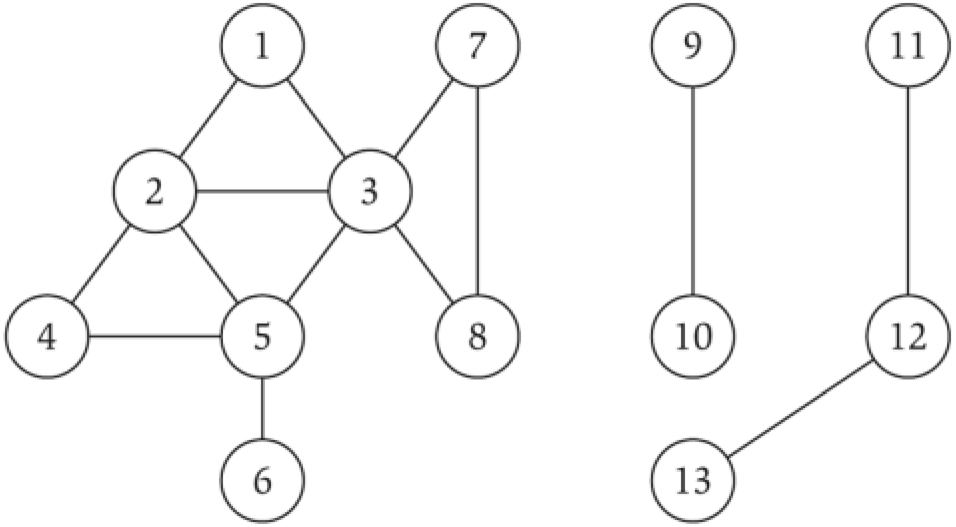
BFS树



- 在G中有边的两个点在BFS树中层数最多只差1

连通分支Connected Component

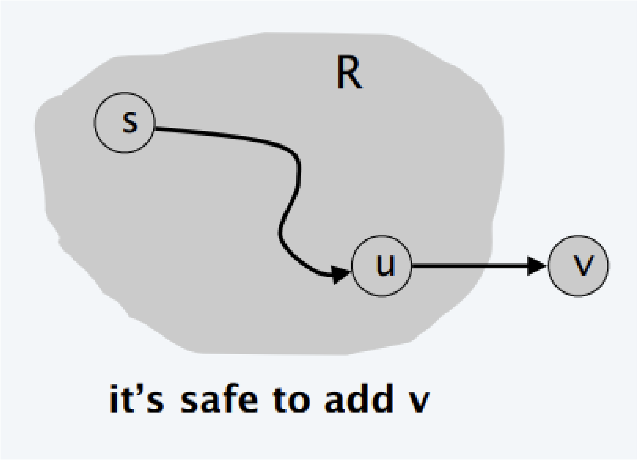
- 连通分支：包含s的连通分支指所有从s出发能到达的点集



- 包含1的连通分支{1,2,3,4,5,6,7,8}

连通分支Connected Component

R will consist of nodes to which s has a path
Initially $R = \{s\}$
While there is an edge (u, v) where $u \in R$ and $v \notin R$
  Add v to R
Endwhile



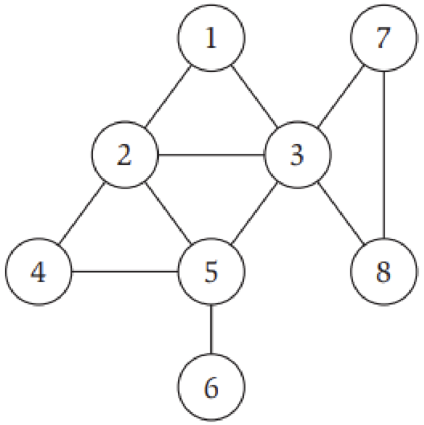
- 算法结束时， R是包含s的连通分支
- 思考：在不断扩充R时一定要使用BFS吗？

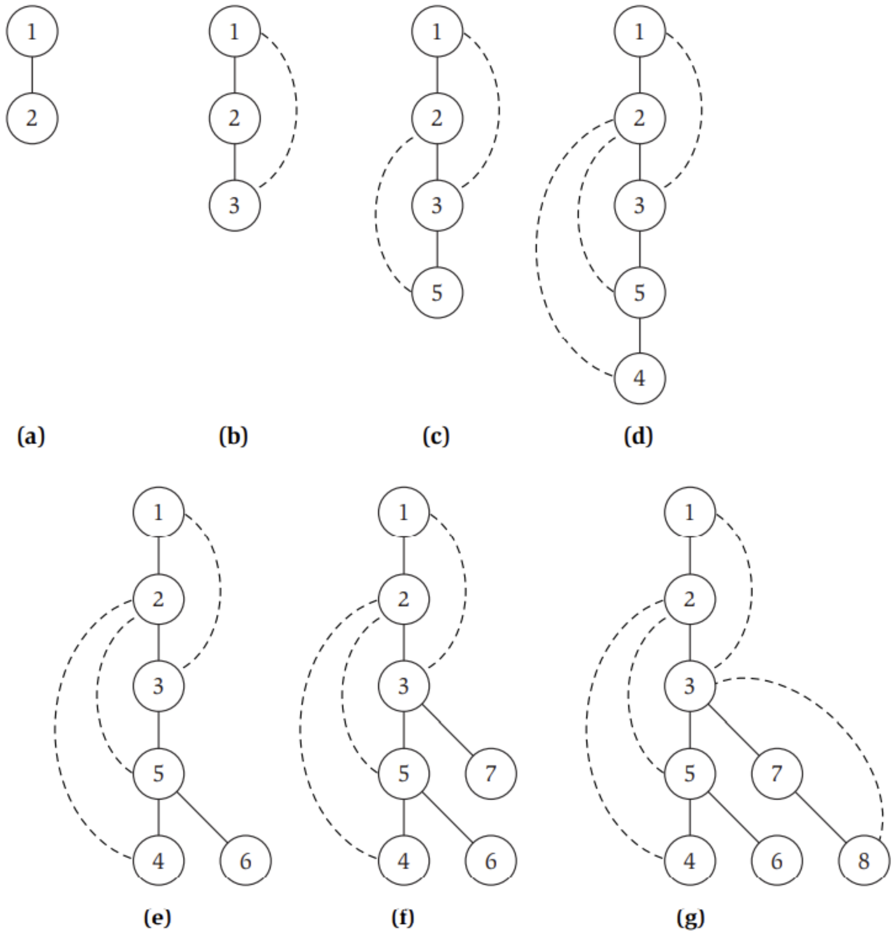
深度优先搜索Depth-first search

- 思想："走迷宫"式搜索
- 到达"死胡同"后，回溯到路径上第一个尚有未访问邻居的点

```
DFS(u):
  Mark u as "Explored" and add u to R
  For each edge (u,v) incident to u
    If v is not marked "Explored" then
      Recursively invoke DFS(v)
    Endif
  Endfor
```

DFS树





- 设 T是一棵DFS树， x和y是T中的节点，并且(x,y)为G 的边但不是T的 边，则x和y中的一个是另一个的祖先。

图遍历的实现

- 从某个点G出发，BFS和DFS访问连通分支中的点仅有顺序不同
  - ✓队列FIFO
  - ✓堆栈LIFO

- BFS：队列
- DFS：堆栈

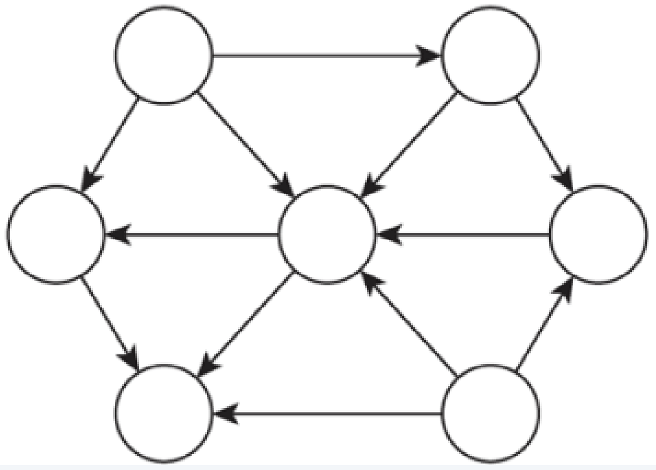
BFS的实现

```
BFS(s):
  Set Discovered[s] = true and Discovered[v] = false for all other v
  Initialize L[0] to consist of the single element s
  Set the layer counter i = 0
  Set the current BFS tree T = Ø
  While L[i] is not empty
    Initialize an empty list L[i+1]
    For each node u ∈ L[i]
      Consider each edge (u, v) incident to u
      If Discovered[v] = false then
        Set Discovered[v] = true
        Add edge (u, v) to the tree T
        Add v to the list L[i+1]
      Endif
    Endfor
    Increment the layer counter i by one
  Endwhile
```

- 若给定图用邻接表来表示，该程序运行时间为$O(m+n)$
- DFS同理

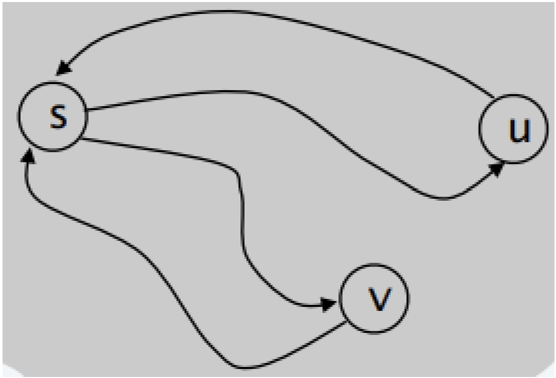
有向图

- 有向可到达性： 找到所有s能到达的点
- 有向s-t最短路径问题：给定两个节点s和t求他们的最短路径
- 图搜索：BFS可拓展至有向图



有向图：强连通性Strong connectivity

- 相互可到达：同时存在s到t和t到s的路径
- 强连通：G中任意两个点都相互可到达
- G是强连通的当且仅当存在一个点s和任意其他点能相互到达（证明？）

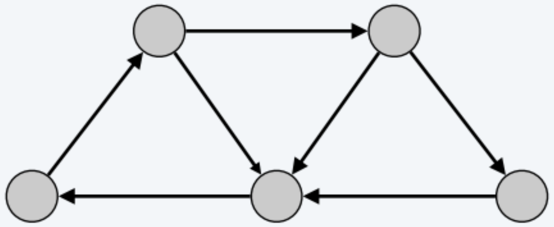


强连通性：算法

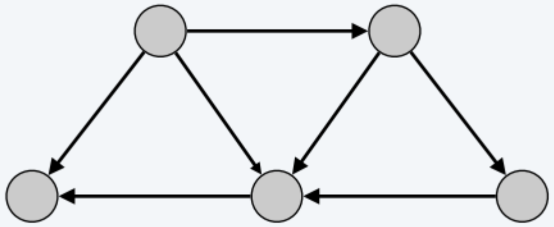
- 可在$O(m+n)$时间内判断G是否强连通

- Pick any node s.
- Run BFS from s in G.
- Run BFS from s in $G^{reverse}$.
- Return true iff all nodes reached in both BFS executions.
- Correctness follows immediately from previous lemma. ▪

reverse orientation of every edge in G



strongly connected



not strongly connected

例题：

1. 作答正确 讨论区

   现有一个 $n*m$ 大小的迷宫，其中 `1` 表示不可通过的墙壁，`0` 表示平地。每次移动只能向上下左右移动一格（不允许移动到曾经经过的位置），且只能移动到平地上。现从迷宫左上角出发，问能否在恰

   作答

2. Mr.W 要制作一个体积为 $N\pi$ 的 M 层生日蛋糕，每层都是一个圆柱体。

   设从下往上数第 i 蛋糕是半径为 R_i，高度为 H_i 的圆柱。当 $i < M$ 时，要求 $R_i > R_{i+1}$ 且 $H_i > H_{i+1}$。由于要在蛋糕上抹奶油，为尽可能节约经费，我们希望蛋糕外表面（最下一层的下底面除外）

   令 $Q = S\pi$，请编程对给出的 N 和 M，找出蛋糕的制作方案（适当的 R_i 和 H_i 的值），使 S 最小。（除 Q 外，以上所有数据皆为正整数）

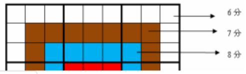
   

   作答

3. 小城和小华都是热爱数学的好学生，最近，他们不约而同地迷上了数独游戏，好胜的他们想用数独来一比高低。但普通的数独对他们来说都过于简单了，于是他们向 Z 博士请教，Z 博士拿出了他最近发明

   靶形数独的方格同普通数独一样，在 9 格宽 $\times 9$ 格高 的大九宫格中有 9 个 3 格宽 $\times 3$ 格高 的小九宫格（用粗黑色线隔开的）。在这个大九宫格中，有一些数字是已知的，根据这些数字，利用逻辑推理，

   行、每列也不能重复出现。但靶形数独有一点和普通数独不同，即 每一个方格都有一个分值，而且如同一个靶子一样，离中心越近则分值越高。（如图）

   作答

4. 作答正确 讨论区

   地上有一排格子，共 n 个位置。机器猫站在第一个格子上，需要取第 n 个格子里的东西。

   机器猫当然不愿意自己跑过去，所以机器猫从口袋里掏出了一个机器人！这个机器人的行动遵循下面的规则：

   初始时，机器人位于 1 号格子
   若机器人目前在 x 格子，那么它可以跳跃到 $x-1,x+1,2x$ 里的一个格子（不允许跳出界）
   问机器人最少需要多少次跳跃，才能到达 n 号格子。

   作答

5. 在一个 4×4 的棋盘上有 8 个黑棋和 8 个白棋，当且仅当两个格子有公共边，这两个格子上的棋是相邻的。移动棋子的规则是交换相邻两个棋子。

   给出一个初始棋盘和一个最终棋盘，请找出一个最短的移动序列使初始棋盘变为最终棋盘。

   作答

6. 为了让奶牛们娱乐和锻炼，农夫约翰建造了一个美丽的池塘。这个长方形的池子被分成了 M 行 N 列个方格 $(1 \le M, N \le 30)$。一些格子是坚固得令人惊讶的莲花，还有一些格子是岩石，其余的只

   贝西正在练习芭蕾舞，她站在一朵莲花上，想跳到另一朵莲花上去，她只能从一朵莲花跳到另一朵莲花上，既不能跳到水里，也不能跳到岩石上。

   贝西的舞步很像象棋中的马步：每次总是先横向移动一格，再纵向移动两格，或先纵向移动两格，再横向移动一格。最多时，贝西会有八个移动方向可供选择。

   约翰一直在观看贝西的芭蕾练习，发现她有时候不能跳到终点，因为中间缺了一些荷叶。于是他想要添加几朵莲花来帮助贝西完成任务。一贯节俭的约翰只想添加最少数量的莲花。当然，莲花不能放在石

   请帮助约翰确定必须要添加的莲花的最少数量。在添加莲花最少的基础上，确定贝西从起点跳到目标需要的最少步数。最后，确定满足添加的莲花数量最少时，步数最少的路径条数。

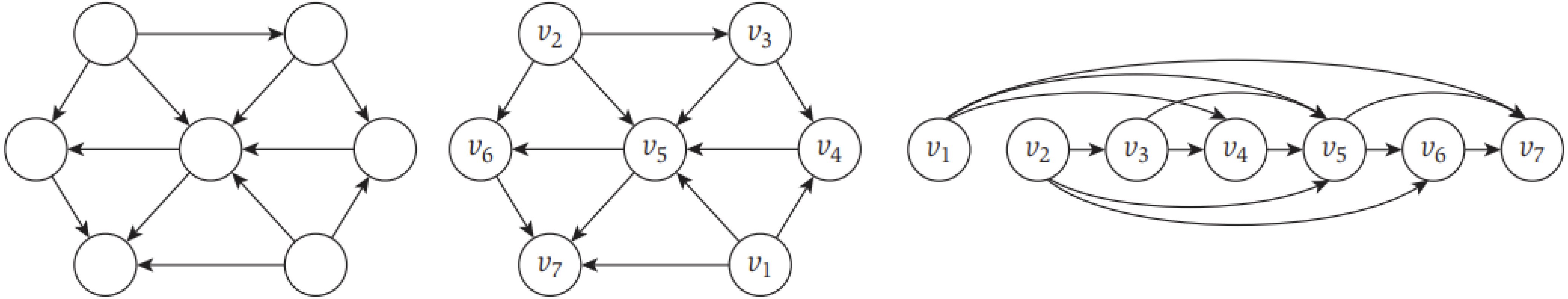
   作答

课程提要

- 回顾：图的基本定义
- 图连通与图遍历
- 有向无环图和拓扑排序

有向无环图Directed acyclic graphs (DAG)

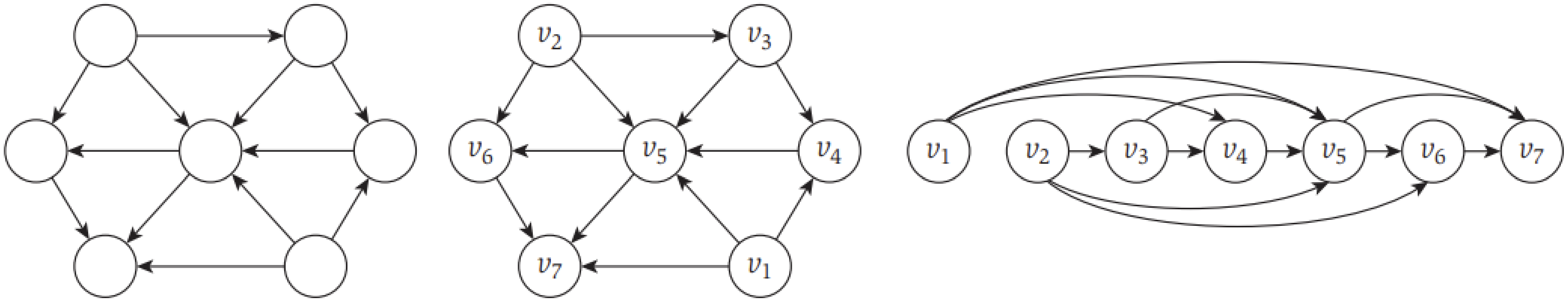
- 拓扑排序：在一个有向图G中，将所有n个点的排序成$v_1, v_2, \dots, v_n$，对任意的边(v_i, v_j)满足$i < j$



(a) (b) (c)

有向无环图Directed acyclic graphs (DAG)

- 拓扑排序：在一个有向图G中，将所有n个点的排序成$v_1, v_2, \ldots, v_n$，对任意的边(v_i, v_j)满足$i < j$

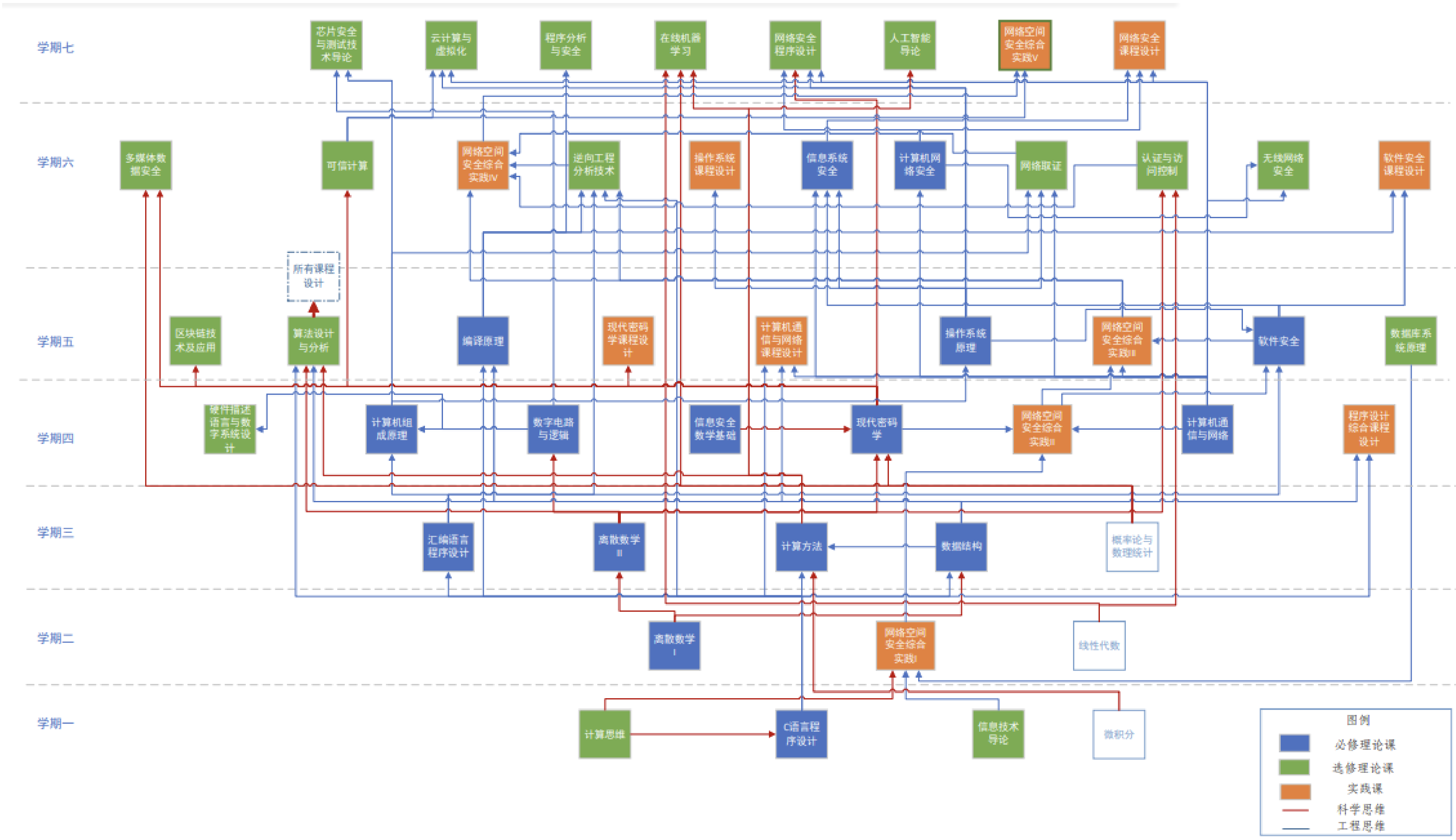


(a) (b) (c)

拓扑排序

- 优先级约束： 有向边(v_i, v_j)表示任务v_i必须在v_j之前完成
- 应用：
  - ✓课程优先级关系图
  - ✓编译代码的前后顺序
  - ✓生产用于组装大型商品的零件

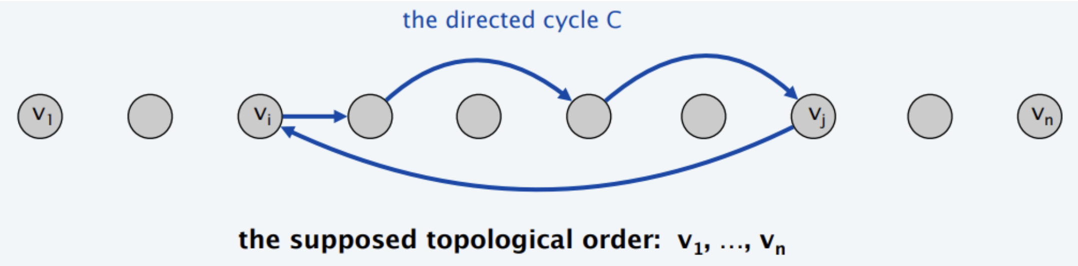
拓扑排序



学期七
学期六
学期五
学期四
学期三
学期二
学期一
芯片安全与测试技术导论
云计算与虚拟化
程序分析与安全
在线机器学习
网络安全程序设计
人工智能导论
网络空间安全综合实践V
网络安全课程设计
多媒体数据安全
可信计算
网络空间安全综合实践IV
逆向工程分析技术
操作系统课程设计
信息系统安全
计算机网络安全
网络取证
认证与访问控制
无线网络安全
软件安全课程设计
所有课程设计
区块链技术及应用
算法设计与分析
编译原理
现代密码学课程设计
计算机通信与网络课程设计
操作系统原理
网络空间安全综合实践III
软件安全
数据库系统原理
硬件描述语言与数字系统设计
计算机组成原理
数字电路与逻辑
信息安全数学基础
现代密码学
网络空间安全综合实践II
计算机通信与网络
程序设计综合课程设计
汇编语言程序设计
离散数学II
计算方法
数据结构
概率论与数理统计
离散数学I
网络空间安全综合实践I
线性代数
计算思维
C语言程序设计
信息技术导论
微积分
图例
必修理论课
选修理论课
实践课
科学思维
工程思维

DAG

- 如果G中包含拓扑排序，则G 必是一个DAG。
- 证明（反证法）：
  - ✓ 假设G中所有n个点的拓扑排序为$v_1, v_2, \ldots, v_n$，且G 包含一个环
  - ✓ 令v_i表示环中下标最小的点，令v_j是环中v_i的前一个点，所以(v_j, v_i)有边，
  - ✓ 根据v_i 的假设，$i < j$
  - ✓ 但在拓扑排序中，(v_j, v_i)是边，则$j < i$，矛盾

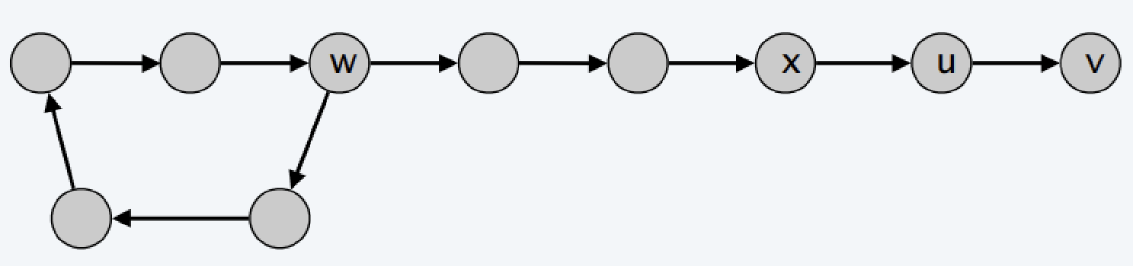


DAG

- 如果G中包含拓扑排序，则G 必是一个DAG。
- 思考题：
  - ✓任意的DAG一定存在拓扑排序吗？
  - ✓如果是，我们如何计算呢？

DAG

- 如果G是一个DAG，则G 至少有一个点没有入边。
- 证明（反证法）：
  - ✓假设G是一个DAG，且G 中所有点都有至少一条入边。
  - ✓从一个点v出发，沿着入边回退，v的上一个点u， u的上一个点x
  - ✓直到访问某个点w两次， 则形成一个环

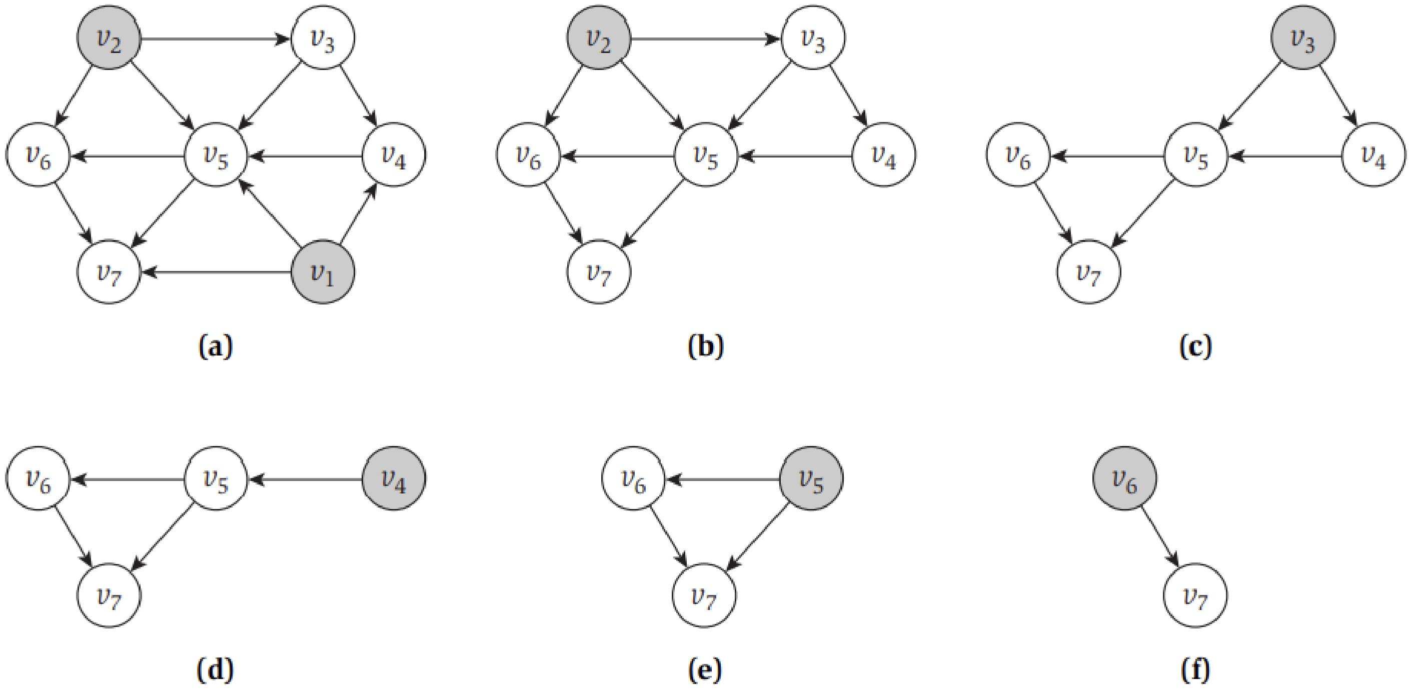


DAG

- 如果G是一个DAG，则G 包含一个拓扑排序。
- 证明（数学归纳法）：
  - ✓$n = 1$ 显然成立
  - ✓DAG包含 $n > 1$个顶点，设v是没有入边的点，
  - ✓则$G\backslash\{v\}$依然是DAG，根据归纳假设$G\backslash\{v\}$包含一个拓扑排序
  - ✓把v加入$G\backslash\{v\}$的拓扑排序最前面即得G的拓扑排序

DAG

- 计算DAG的拓扑排序：以此删除没有入边的点



时间复杂度

$O(m+n)$