

主要内容



- 图的基本概念
- 宽度优先搜索
- 深度优先搜索
- 有向图上的深度优先搜索
- 有向图中的环路判断
- 拓扑排序

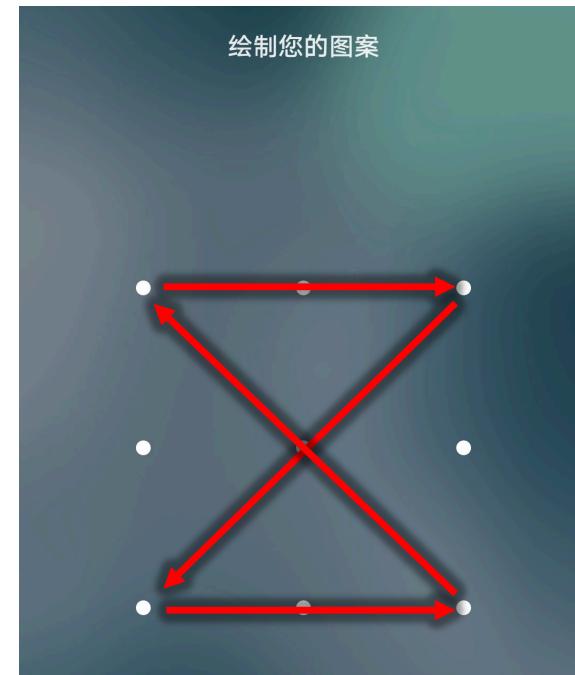
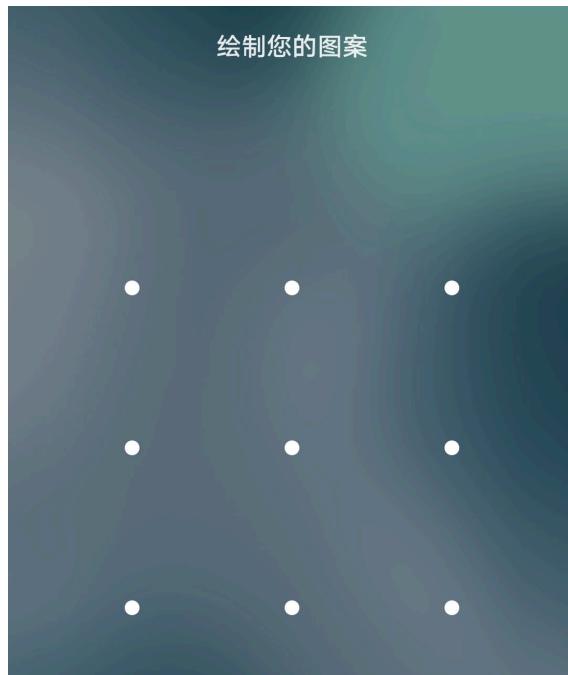
图算法篇：图的基本概念

北京航空航天大学
计算机学院

图的背景



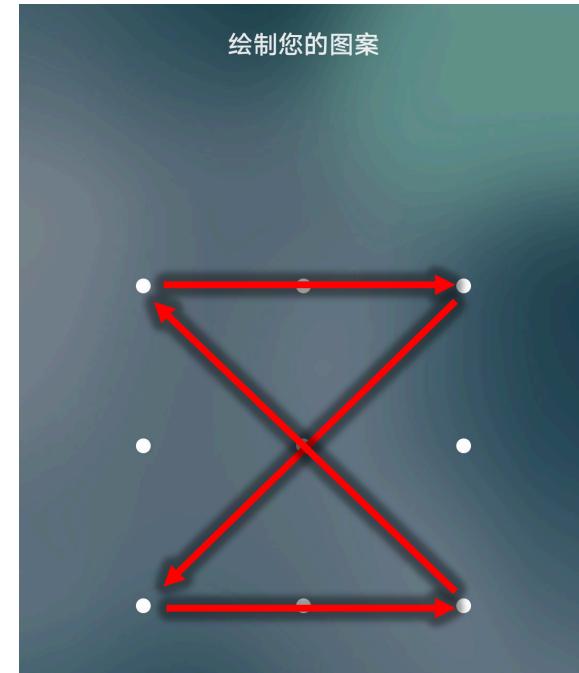
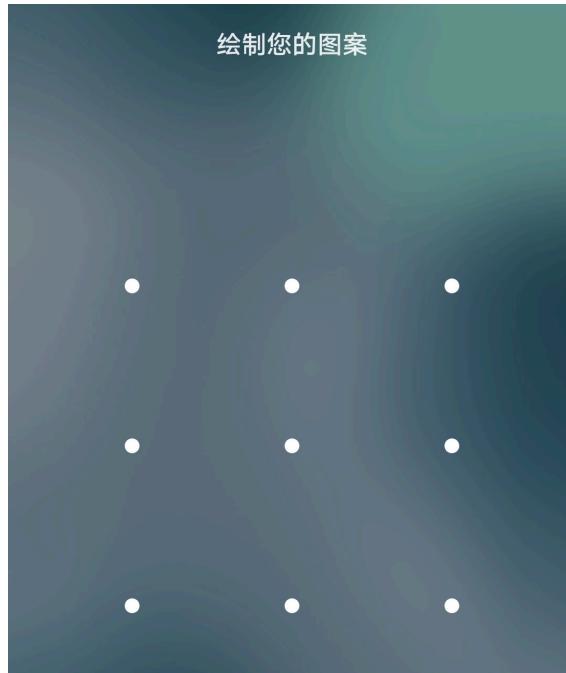
- 一笔画问题：手机解锁图案需一笔画出





图的背景

- 一笔画问题：手机解锁图案需一笔画出

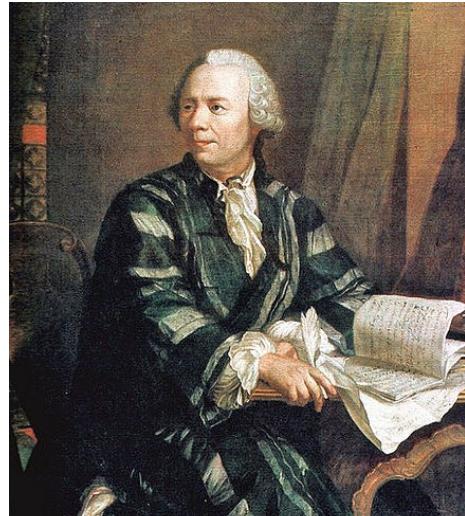


哪些图案可以仅用一笔就画出来？

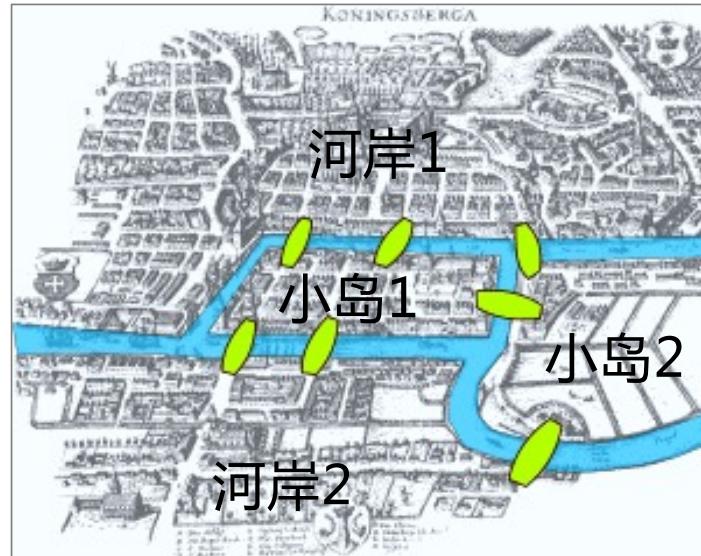


图的背景

- 柯尼斯堡七桥问题：七座桥连接河岸和两个小岛，步行者怎样才能不重复、不遗漏地一次走完七座桥？



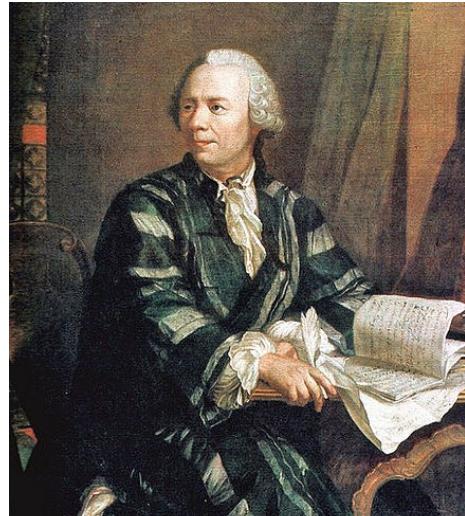
瑞士数学家
欧拉



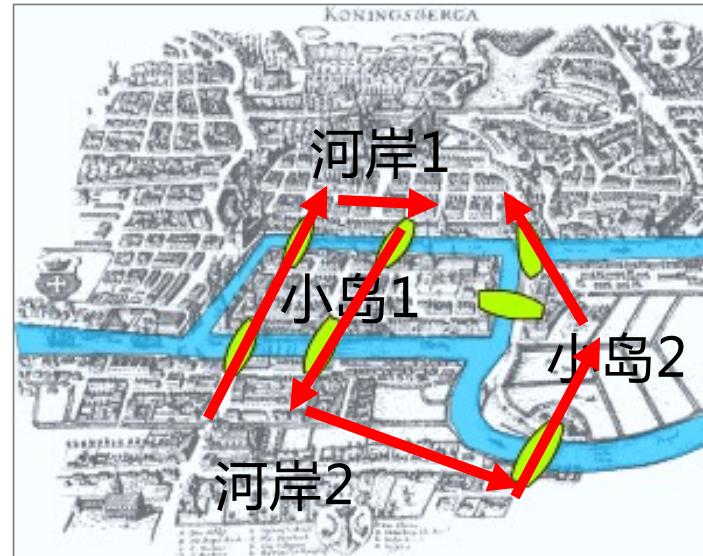


图的背景

- 柯尼斯堡七桥问题：七座桥连接河岸和两个小岛，步行者怎样才能不重复、不遗漏地一次走完七座桥？



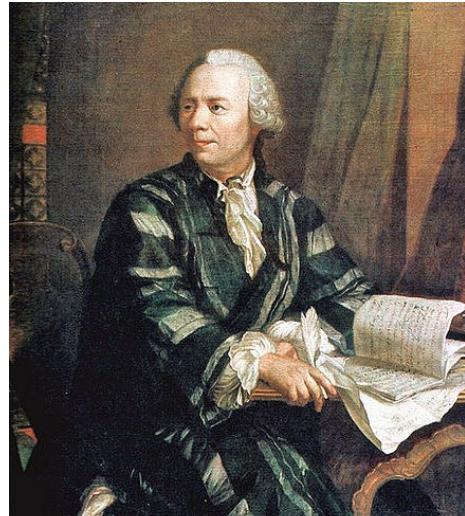
瑞士数学家
欧拉



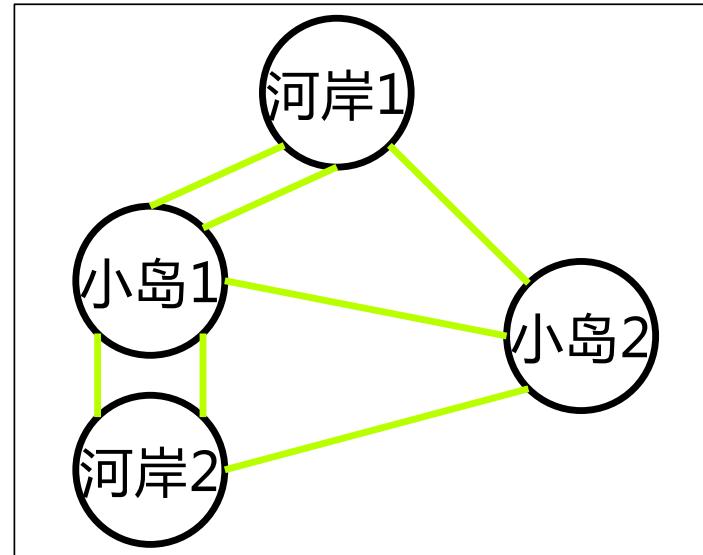


图的背景

- 柯尼斯堡七桥问题：七座桥连接河岸和两个小岛，步行者怎样才能不重复、不遗漏地一次走完七座桥？



瑞士数学家
欧拉

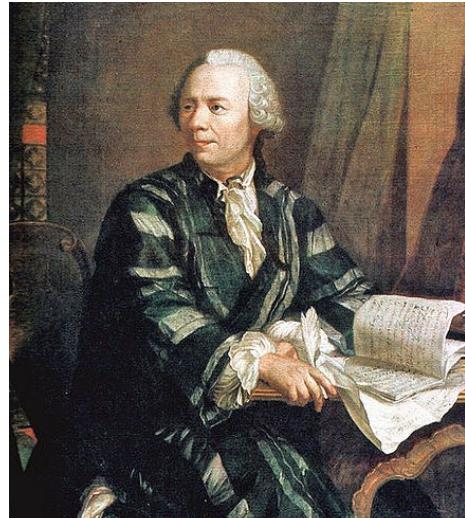


经过抽象之后，仅保留点和边的结构称为图

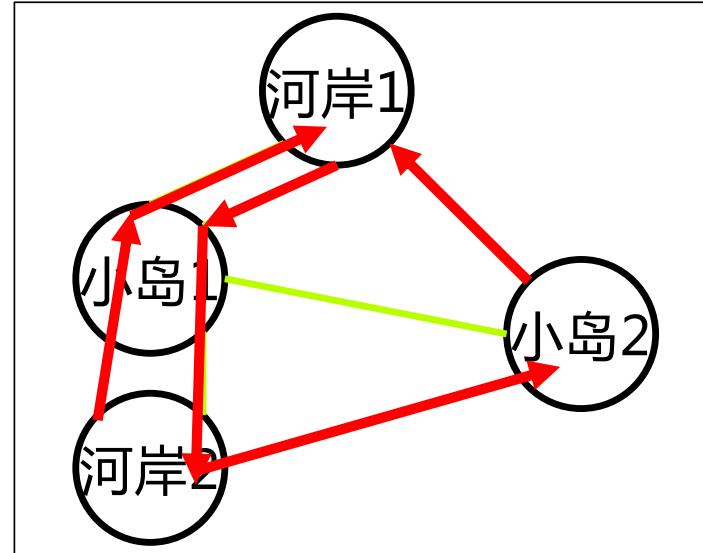


图的背景

- 柯尼斯堡七桥问题：七座桥连接河岸和两个小岛，步行者怎样才能不重复、不遗漏地一次走完七座桥？



瑞士数学家
欧拉

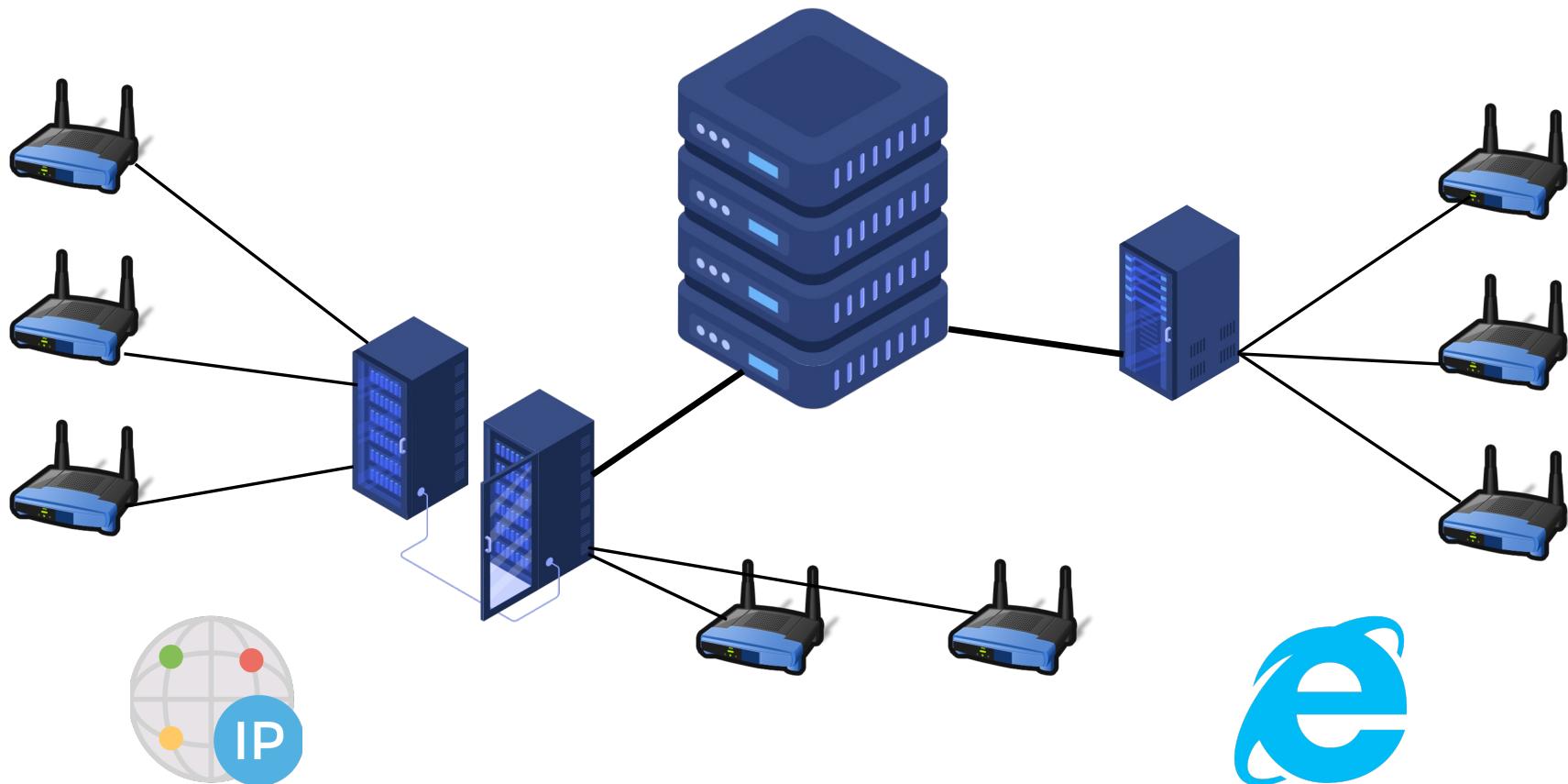


忽略河岸和小岛的形状大小，重新建模为一笔画问题

图的背景



- 图：现实中常见结构
 - 计算机网络：因特网，万维网



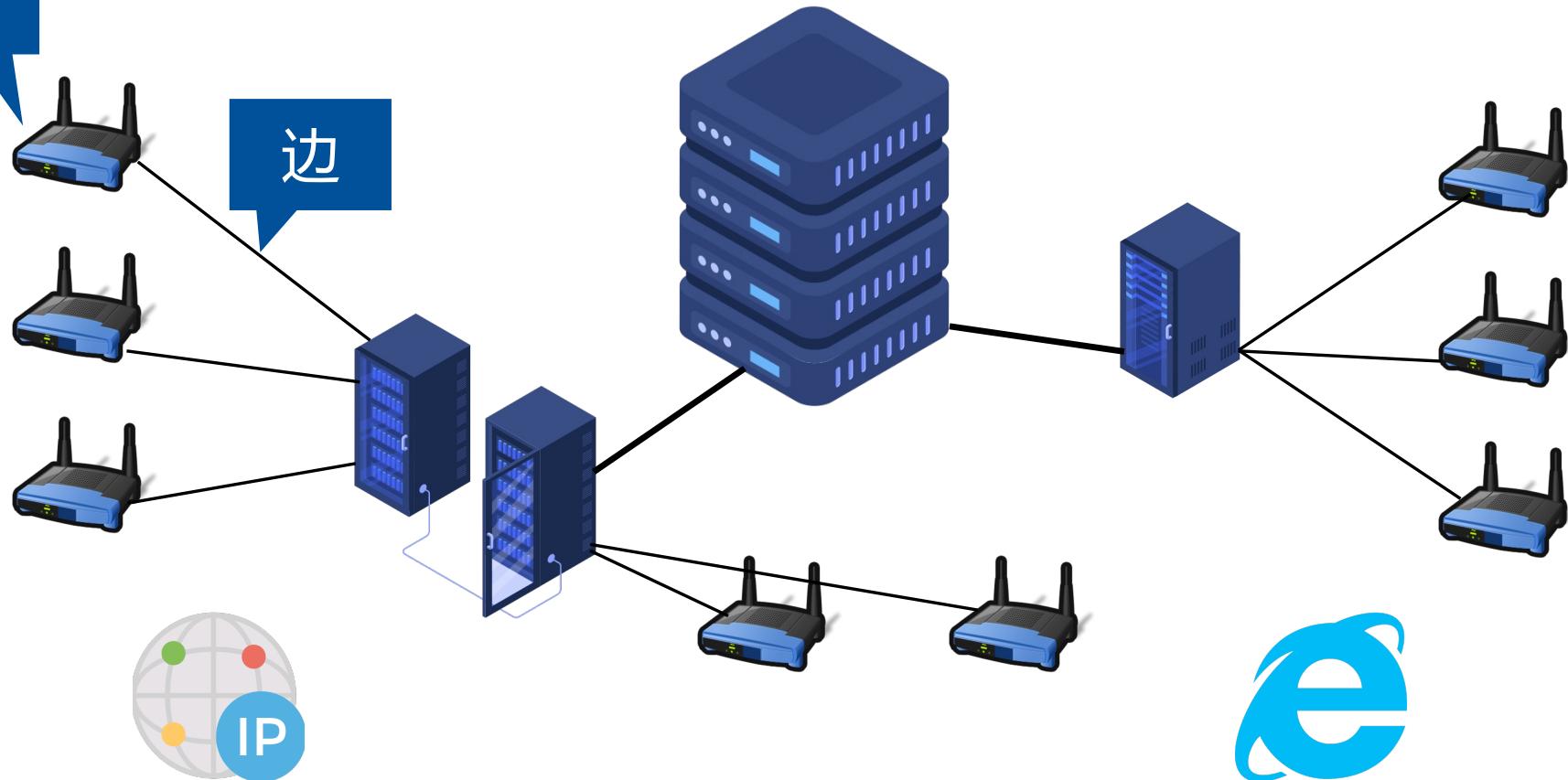
图的背景



- 图：现实中常见结构
 - 计算机网络：因特网，万维网

点

边





图的背景

- 图：现实中常见结构
 - 交通出行：北京地铁图





图的背景

- 图：现实中常见结构
 - 交通出行：北京地铁图



图的背景



- 图：现实中常见结构
 - 社交网络：微博





图的背景

- 图：现实中常见结构

- 社交网络：微博





图的概念：图的定义

- 图可以表示为一个二元组 $G = \langle V, E \rangle$ ，其中
 - V 表示非空顶点集，其元素称为顶点(Vertex)
 - E 表示边集，其元素称为边(Edge)



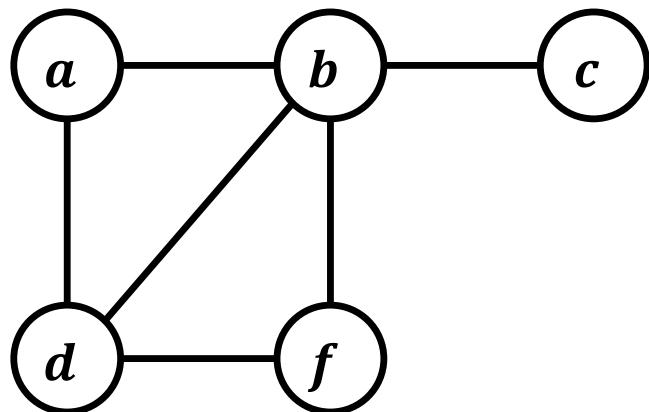
图的概念：图的定义

- 图可以表示为一个二元组 $G = \langle V, E \rangle$ ，其中
 - V 表示非空顶点集，其元素称为顶点(Vertex)
 - E 表示边集，其元素称为边(Edge)
- $e = (u, v)$ 表示一条边，其中 $u \in V, v \in V, e \in E$



图的概念：图的定义

- 图可以表示为一个二元组 $G = \langle V, E \rangle$ ，其中
 - V 表示非空顶点集，其元素称为顶点(Vertex)
 - E 表示边集，其元素称为边(Edge)
- $e = (u, v)$ 表示一条边，其中 $u \in V, v \in V, e \in E$
- 无向图 $G_1 = \langle V_1, E_1 \rangle$



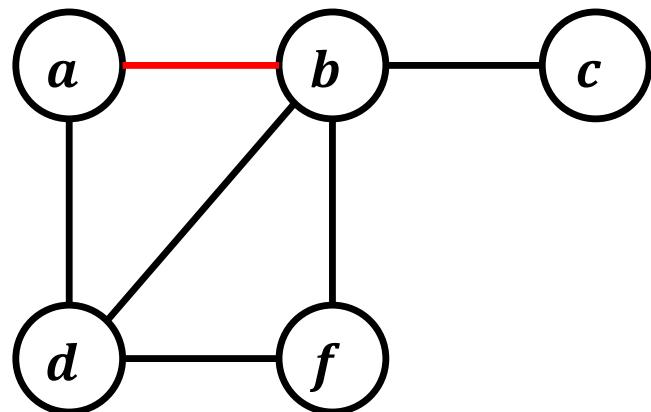
$$V_1 = \{a, b, c, d, f\}$$

$$E_1 = \{(a, b), (a, d), (b, c), (b, d), (b, f), (d, f)\}$$



图的概念：图的定义

- 图可以表示为一个二元组 $G = \langle V, E \rangle$ ，其中
 - V 表示非空顶点集，其元素称为顶点(Vertex)
 - E 表示边集，其元素称为边(Edge)
- $e = (u, v)$ 表示一条边，其中 $u \in V, v \in V, e \in E$
- 无向图 $G_1 = \langle V_1, E_1 \rangle$



$$V_1 = \{a, b, c, d, f\}$$

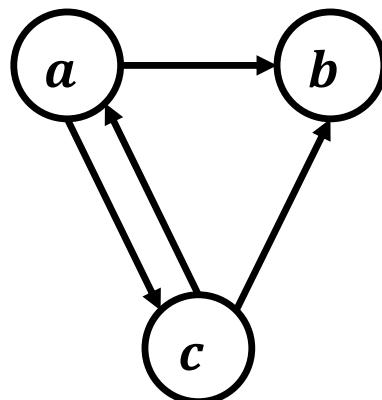
$$E_1 = \{(a, b), (a, d), (b, c), (b, d), (b, f), (d, f)\}$$

(a, b) 和 (b, a) 被视作同一条边



图的概念：图的定义

- 图可以表示为一个二元组 $G = \langle V, E \rangle$ ，其中
 - V 表示非空顶点集，其元素称为顶点(Vertex)
 - E 表示边集，其元素称为边(Edge)
- $e = (u, v)$ 表示一条边，其中 $u \in V, v \in V, e \in E$
- 有向图 $G_2 = \langle V_2, E_2 \rangle$



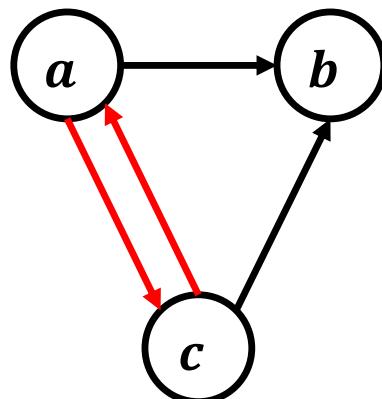
$$V_2 = \{a, b, c\}$$

$$E_2 = \{(a, b), (a, c), (c, a), (c, b)\}$$



图的概念：图的定义

- 图可以表示为一个二元组 $G = \langle V, E \rangle$ ，其中
 - V 表示非空顶点集，其元素称为顶点(Vertex)
 - E 表示边集，其元素称为边(Edge)
- $e = (u, v)$ 表示一条边，其中 $u \in V, v \in V, e \in E$
- 有向图 $G_2 = \langle V_2, E_2 \rangle$



$$V_2 = \{a, b, c\}$$

$$E_2 = \{(a, b), (a, c), (c, a), (c, b)\}$$

(a, c) 和 (c, a) 是两条不同的边

图的概念：相邻与关联



- 相邻(**Adjacent**)
 - 边 (u, v) 连接的顶点 u 和 v **相邻**
- 关联(**Incident**)
 - 边 (u, v) 和其连接的顶点 u (或 v) **相互关联**



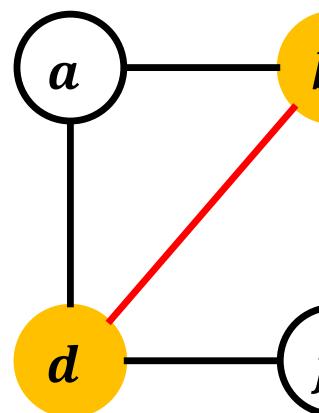
图的概念：相邻与关联

- **相邻(Adjacent)**

- 边 (u, v) 连接的顶点 u 和 v 相邻

- **关联(Incident)**

- 边 (u, v) 和其连接的顶点 u (或 v) 相互关联



顶点 b 和 d 相邻

(b, d) 与顶点 b 和 d 关联



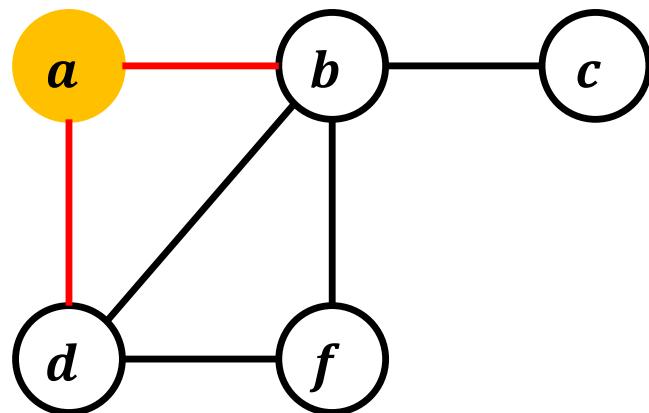
图的概念：度

- 顶点的度(Degree of a Vertex)
 - 顶点 v 的度 $\deg(v)$ 是 v 关联的边数
- 图的度(Degree of a Graph)
 - 图 $G = \langle V, E \rangle$ 的度，是图各顶点的度之和， $\deg(G) = \sum_{v \in V} \deg(v)$



图的概念：度

- 顶点的度(Degree of a Vertex)
 - 顶点 v 的度 $\deg(v)$ 是 v 关联的边数
- 图的度(Degree of a Graph)
 - 图 $G = \langle V, E \rangle$ 的度，是图各顶点的度之和， $\deg(G) = \sum_{v \in V} \deg(v)$
- 图 $G_1 = \langle V_1, E_1 \rangle$



$$V_1 = \{a, b, c, d, f\}$$

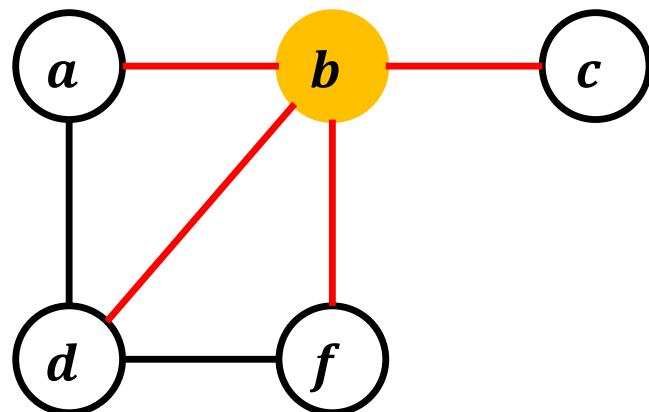
$$E_1 = \{(a, b), (a, d), (b, c), (b, d), (b, e), (d, f)\}$$

$$\deg(a) = 2$$



图的概念：度

- 顶点的度(Degree of a Vertex)
 - 顶点 v 的度 $\deg(v)$ 是 v 关联的边数
- 图的度(Degree of a Graph)
 - 图 $G = \langle V, E \rangle$ 的度，是图各顶点的度之和， $\deg(G) = \sum_{v \in V} \deg(v)$
- 图 $G_1 = \langle V_1, E_1 \rangle$



$$V_1 = \{a, b, c, d, f\}$$

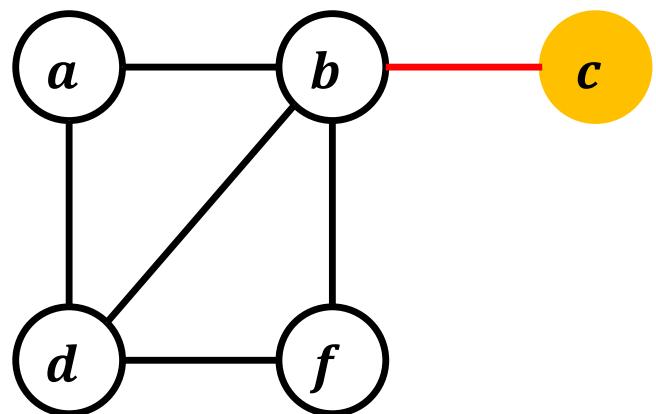
$$E_1 = \{(a, b), (a, d), (b, c), (b, d), (b, e), (D, E)\}$$

$$\deg(a) = 2 \quad \deg(b) = 4$$



图的概念：度

- 顶点的度(Degree of a Vertex)
 - 顶点 v 的度 $\deg(v)$ 是 v 关联的边数
- 图的度(Degree of a Graph)
 - 图 $G = \langle V, E \rangle$ 的度，是图各顶点的度之和， $\deg(G) = \sum_{v \in V} \deg(v)$
- 图 $G_1 = \langle V_1, E_1 \rangle$



$$V_1 = \{a, b, c, d, f\}$$

$$E_1 = \{(a, b), (a, d), (b, c), (b, d), (b, e), (d, e)\}$$

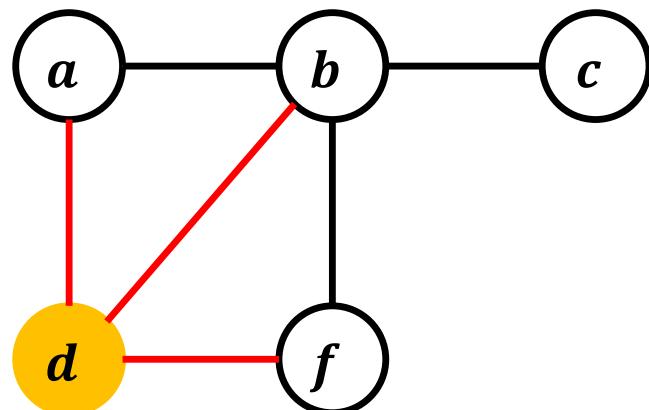
$$\deg(a) = 2 \quad \deg(b) = 4$$

$$\deg(c) = 1$$



图的概念：度

- 顶点的度(Degree of a Vertex)
 - 顶点 v 的度 $\deg(v)$ 是 v 关联的边数
- 图的度(Degree of a Graph)
 - 图 $G = \langle V, E \rangle$ 的度，是图各顶点的度之和， $\deg(G) = \sum_{v \in V} \deg(v)$
- 图 $G_1 = \langle V_1, E_1 \rangle$



$$V_1 = \{a, b, c, d, f\}$$

$$E_1 = \{(a, b), (a, d), (b, c), (b, d), (b, e), (D, E)\}$$

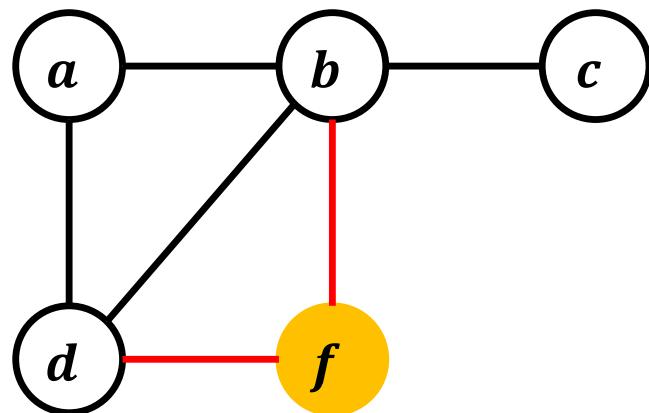
$$\deg(a) = 2 \quad \deg(b) = 4$$

$$\deg(c) = 1 \quad \deg(d) = 3$$



图的概念：度

- 顶点的度(Degree of a Vertex)
 - 顶点 v 的度 $\deg(v)$ 是 v 关联的边数
- 图的度(Degree of a Graph)
 - 图 $G = \langle V, E \rangle$ 的度，是图各顶点的度之和， $\deg(G) = \sum_{v \in V} \deg(v)$
- 图 $G_1 = \langle V_1, E_1 \rangle$



$$V_1 = \{a, b, c, d, f\}$$

$$E_1 = \{(a, b), (a, d), (b, c), (b, d), (b, e), (D, E)\}$$

$$\deg(a) = 2 \quad \deg(b) = 4$$

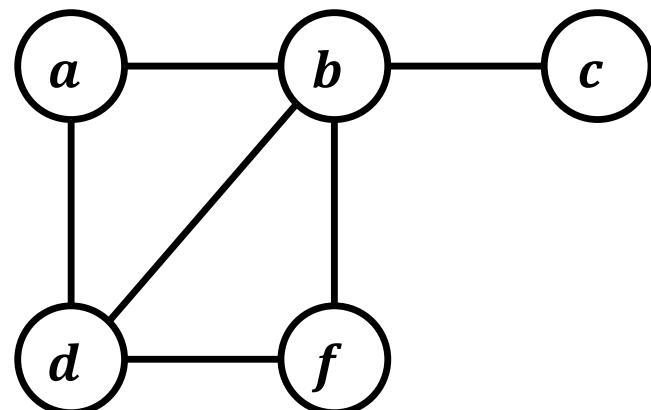
$$\deg(c) = 1 \quad \deg(d) = 3$$

$$\deg(f) = 2$$



图的概念：度

- 顶点的度(Degree of a Vertex)
 - 顶点 v 的度 $\deg(v)$ 是 v 关联的边数
- 图的度(Degree of a Graph)
 - 图 $G = \langle V, E \rangle$ 的度，是图各顶点的度之和， $\deg(G) = \sum_{v \in V} \deg(v)$
- 图 $G_1 = \langle V_1, E_1 \rangle$



$$V_1 = \{a, b, c, d, f\}$$

$$E_1 = \{(a, b), (a, d), (b, c), (b, d), (b, e), (d, e)\}$$

$$\deg(a) = 2 \quad \deg(b) = 4$$

$$\deg(c) = 1 \quad \deg(d) = 3$$

$$\deg(f) = 2$$

$$\deg(G_1) = 2 + 4 + 1 + 3 + 2 = 12$$

握手定理

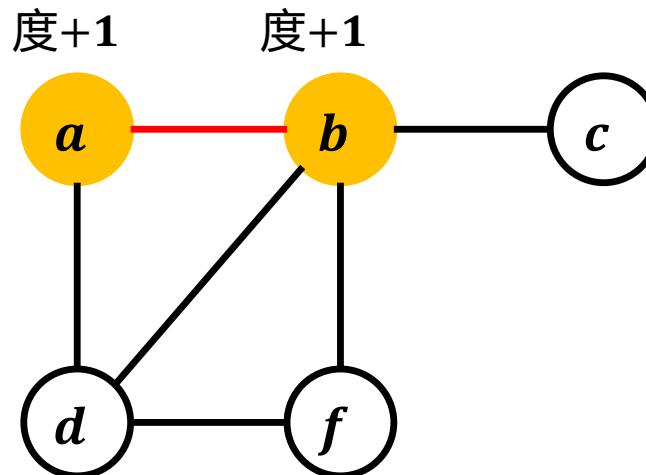


- 握手定理(Handshaking Lemma)
 - 无向图的度是边数的两倍 , $\deg(G) = 2|E|$



握手定理

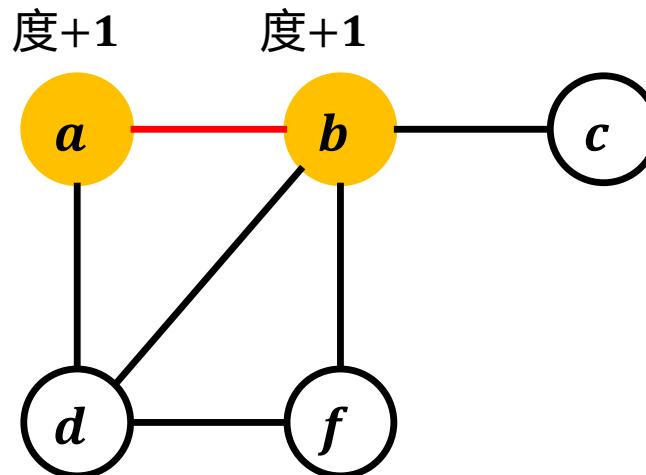
- 握手定理(Handshaking Lemma)
 - 无向图的度是边数的两倍， $\deg(G) = 2|E|$
- 证明
 - 边 $e = (u, v)$ 在 $\deg(u)$ 和 $\deg(v)$ 中各被计算一次





握手定理

- 握手定理(Handshaking Lemma)
 - 无向图的度是边数的两倍 , $\deg(G) = 2|E|$
- 证明
 - 边 $e = (u, v)$ 在 $\deg(u)$ 和 $\deg(v)$ 中各被计算一次
 - 每条边为图的度贡献为 2 , $\deg(G) = \sum_{e \in E} 2 = 2|E|$





图的概念：路径

- **路径(Path)**

- 图中一个的顶点序列 $\langle v_0, v_1, \dots, v_k \rangle$ 称为 v_0 到 v_k 的路径

图的概念：路径



• 路径(Path)

- 图中一个的顶点序列 $< v_0, v_1, \dots, v_k >$ 称为 v_0 到 v_k 的路径
- 路径包含顶点 v_0, v_1, \dots, v_k 和边 $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$

图的概念：路径



• 路径(Path)

- 图中一个的顶点序列 $< v_0, v_1, \dots, v_k >$ 称为 v_0 到 v_k 的路径
- 路径包含顶点 v_0, v_1, \dots, v_k 和边 $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$
- 存在路径 $< v_0, v_1, \dots, v_k >$ ，则 v_0 可达 v_k



图的概念：路径

• 路径(Path)

- 图中一个的顶点序列 $< v_0, v_1, \dots, v_k >$ 称为 v_0 到 v_k 的路径
- 路径包含顶点 v_0, v_1, \dots, v_k 和边 $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$
- 存在路径 $< v_0, v_1, \dots, v_k >$ ，则 v_0 可达 v_k
- 如果 v_0, v_1, \dots, v_k 互不相同，则该路径是简单的

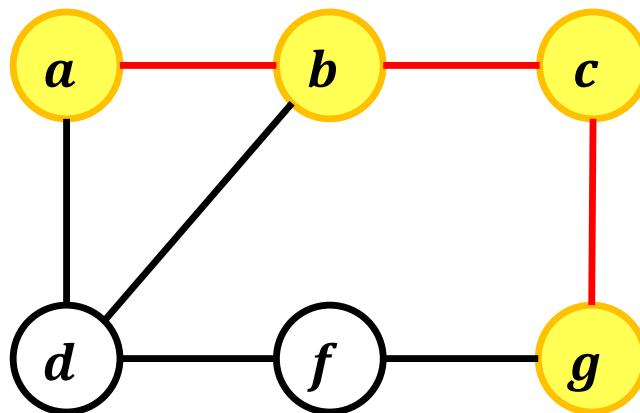


图的概念：路径

- **路径(Path)**

- 图中一个的顶点序列 $< v_0, v_1, \dots, v_k >$ 称为 v_0 到 v_k 的**路径**
- 路径包含**顶点** v_0, v_1, \dots, v_k 和**边** $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$
- 存在路径 $< v_0, v_1, \dots, v_k >$, 则 v_0 可达 v_k
- 如果 v_0, v_1, \dots, v_k 互不相同 , 则该路径是**简单的**

- 路径 $P = < a, b, c, g >$, 顶点 a 可达顶点 g



图的概念：环路



- 环路(Cycle)

- 如果路径 $\langle v_0, v_1, \dots, v_k \rangle$ 中 $v_0 = v_k$ 且至少包含一条边，则该路径构成环路

图的概念：环路



• 环路(Cycle)

- 如果路径 $< v_0, v_1, \dots, v_k >$ 中 $v_0 = v_k$ 且至少包含一条边，则该路径构成环路
- 如果 v_1, v_2, \dots, v_k 互不相同，则该环路是简单的

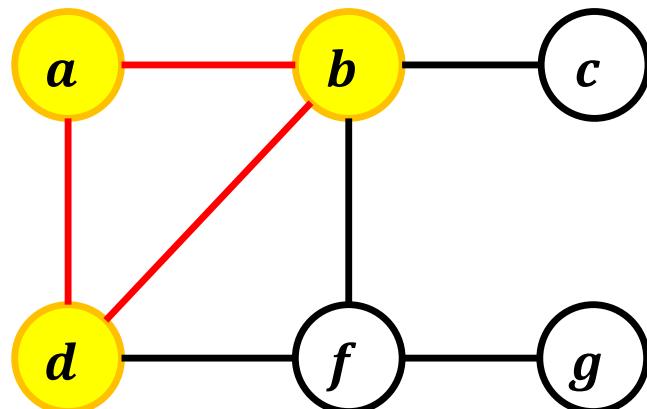
图的概念：环路



- **环路(Cycle)**

- 如果路径 $< v_0, v_1, \dots, v_k >$ 中 $v_0 = v_k$ 且至少包含一条边，则该路径构成**环路**
- 如果 v_1, v_2, \dots, v_k 互不相同，则该环路是**简单的**

- 环路 $C = < a, b, d, a >$



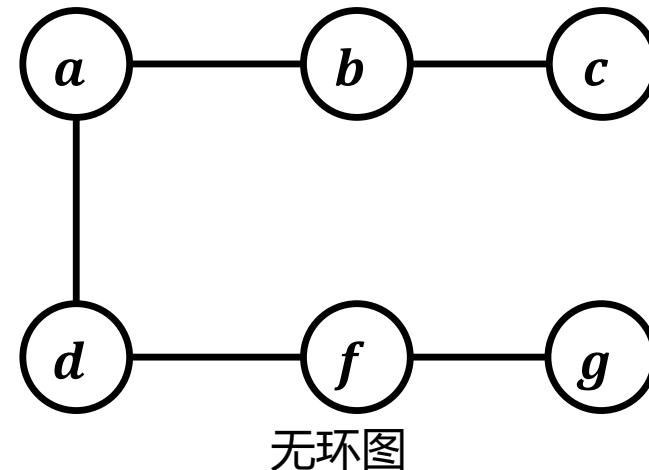
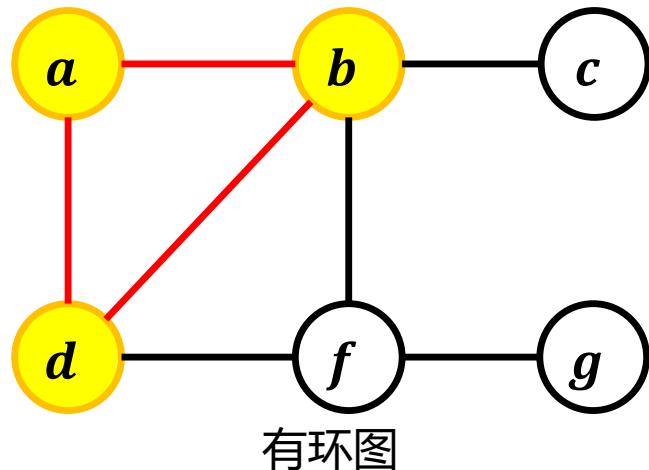
图的概念：环路



- **环路(Cycle)**

- 如果路径 $< v_0, v_1, \dots, v_k >$ 中 $v_0 = v_k$ 且至少包含一条边，则该路径构成**环路**
- 如果 v_1, v_2, \dots, v_k 互不相同，则该环路是**简单的**

- **无环图(Acyclic Graph) : 图中不存在环路**



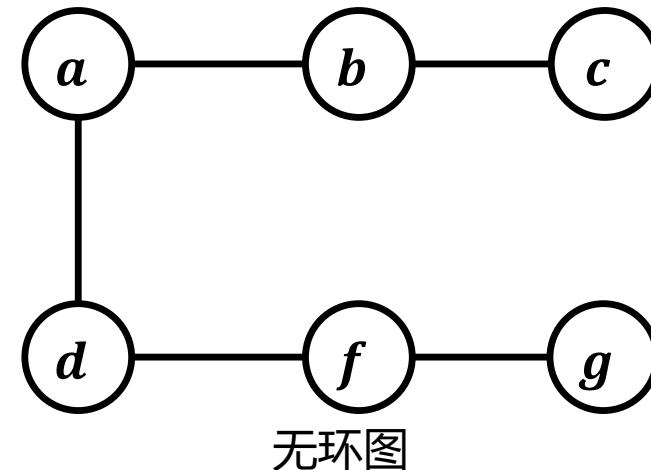
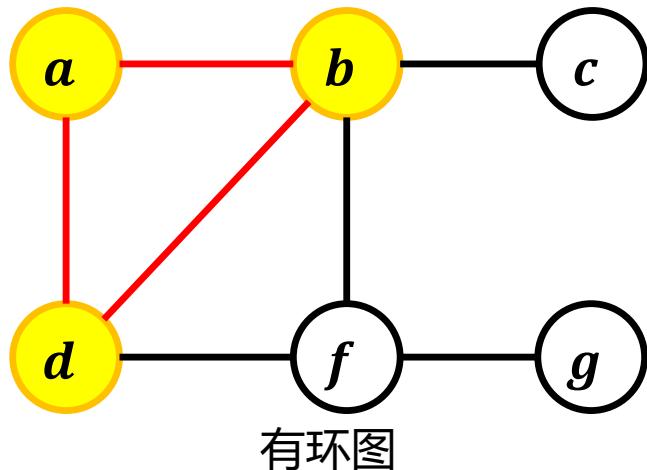
图的概念：环路



- 环路(Cycle)

- 如果路径 $\langle v_0, v_1, \dots, v_k \rangle$ 中 $v_0 = v_k$ 且至少包含一条边，则该路径构成环路
- 如果 v_1, v_2, \dots, v_k 互不相同，则该环路是简单的

- 无环图(Acyclic Graph)：图中不存在环路

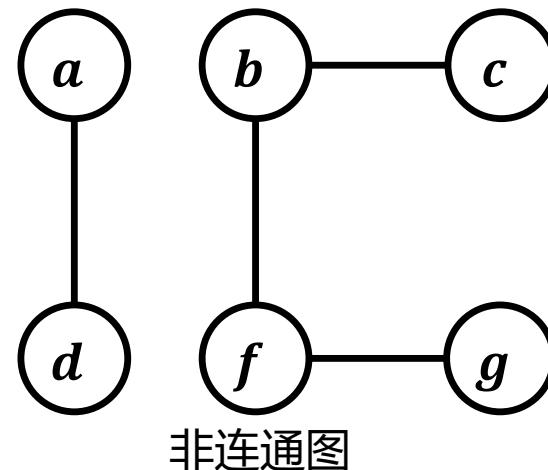
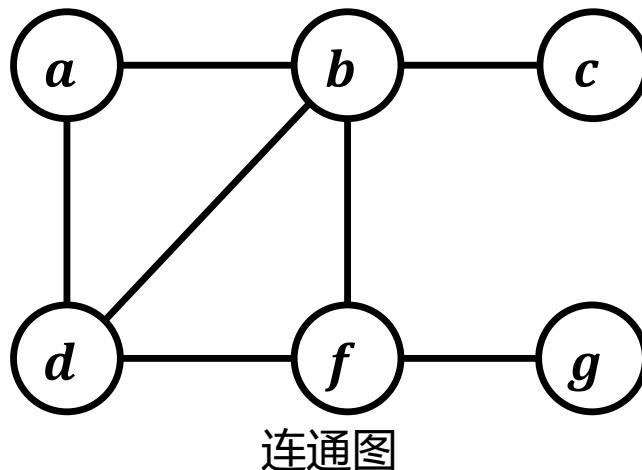


图的概念：连通



- **连通(Connectivity)**

- 如果图的任意一对顶点均互相可达，则称该图是连通的，反之称为非连通



图的概念：连通

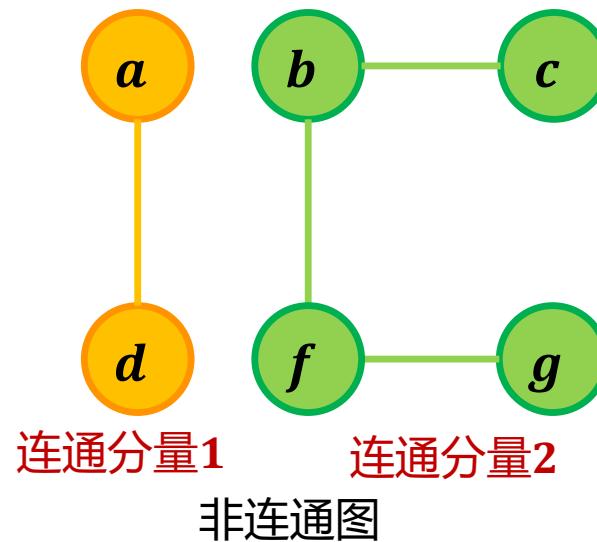


- **连通(Connectivity)**

- 如果图的任意一对顶点均互相可达，则称该图是连通的，反之称为非连通

- **连通分量(Connected Components)**

- 根据是否连通将顶点进行分组，相互可达的顶点集称为连通分量

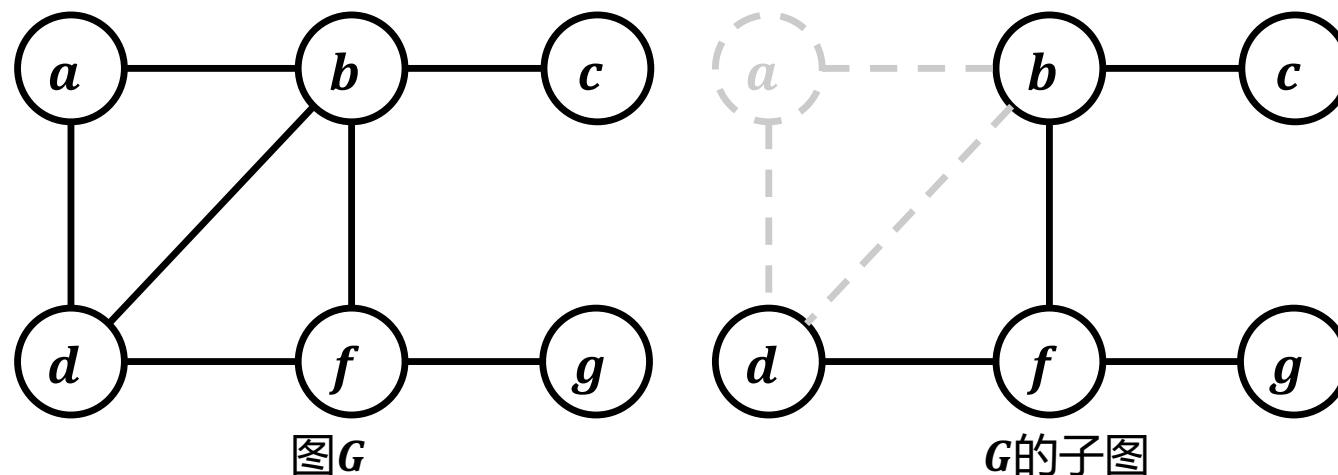




图的概念：子图

- 子图(Subgraph)

- 如果 $V' \subseteq V, E' \subseteq E$ ，则称图 $G' = < V', E' >$ 是图 G 的一个子图



图的概念：子图



- 子图(Subgraph)

- 如果 $V' \subseteq V, E' \subseteq E$ ，则称图 $G' = < V', E' >$ 是图 G 的一个子图

- 生成子图(Spanning Subgraph)

- 如果 $V' = V, E' \subseteq E$ ，则称图 $G' = < V', E' >$ 是图 G 的一个生成子图

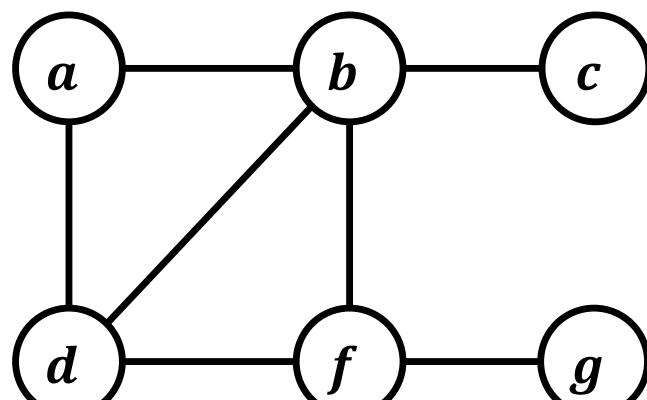
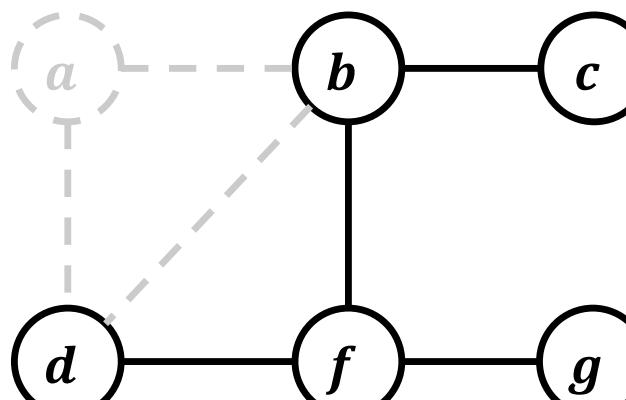
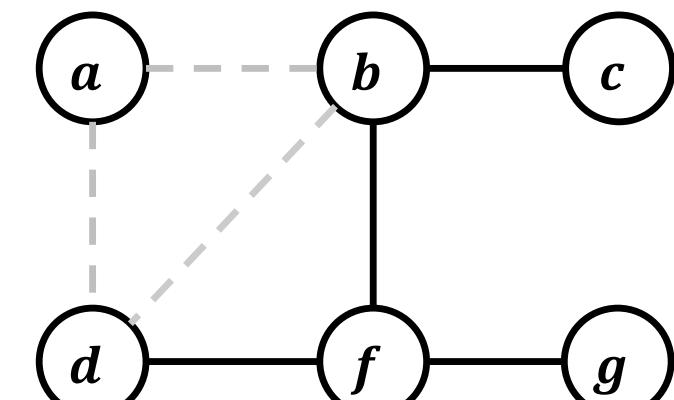


图 G



G 的子图

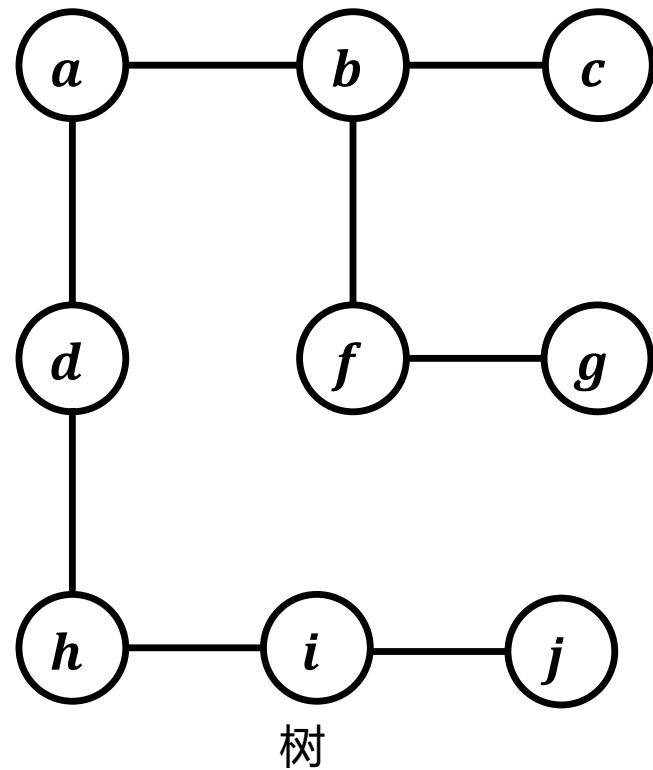


G 的生成子图

图的概念：树



- 树(Tree)
 - 连通、无环图 $T = \langle V_T, E_T \rangle$

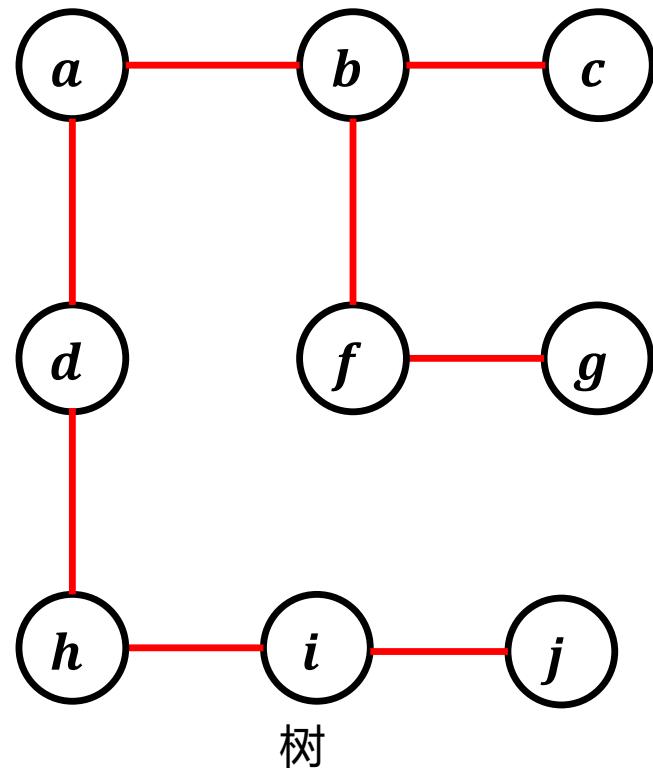




图的概念：树

- 树(Tree)

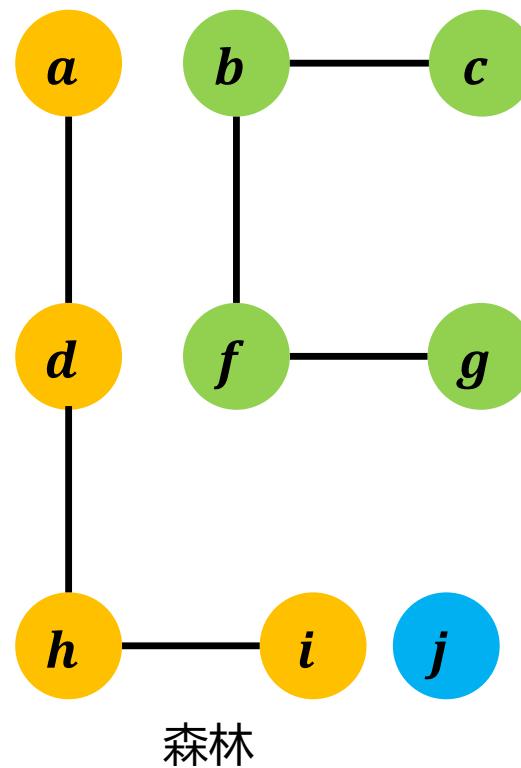
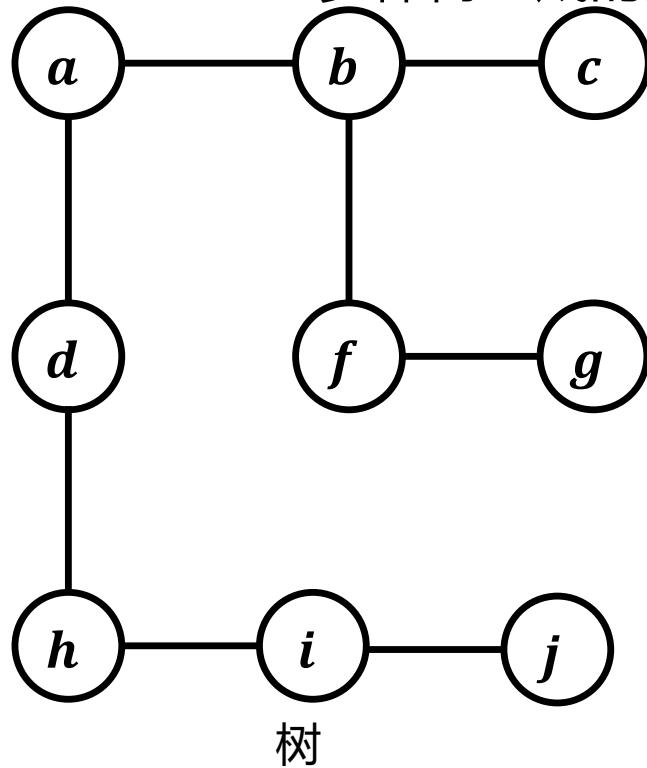
- 连通、无环图 $T = \langle V_T, E_T \rangle$, 树有 $|V_T| - 1$ 条边



图的概念：树



- 树(Tree)
 - 连通、无环图 $T = \langle V_T, E_T \rangle$, 树有 $|V_T| - 1$ 条边
- 森林(Forest)
 - 一至多棵树组成的无环图



图的概念：树

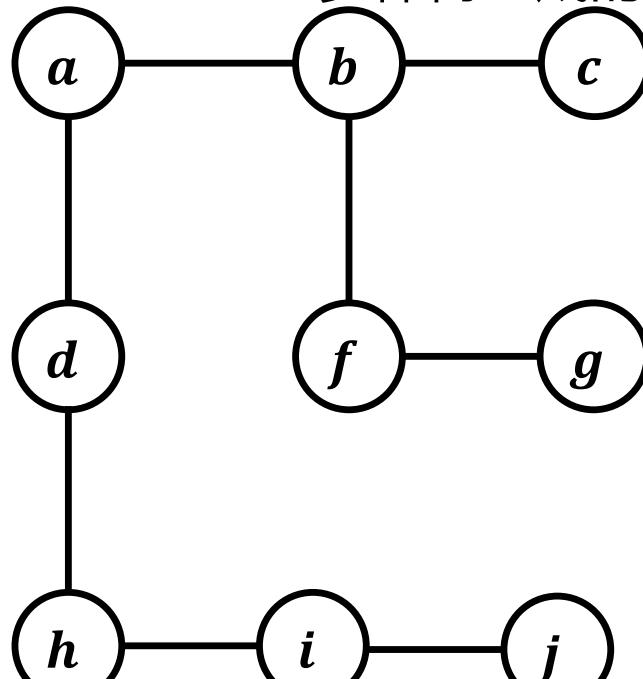


- **树(Tree)**

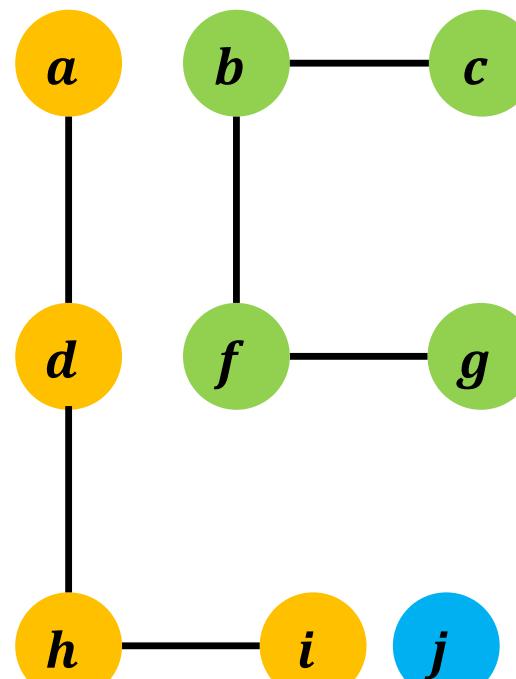
- 连通、无环图 $T = \langle V_T, E_T \rangle$ ，树有 $|V_T| - 1$ 条边

- **森林(Forest)**

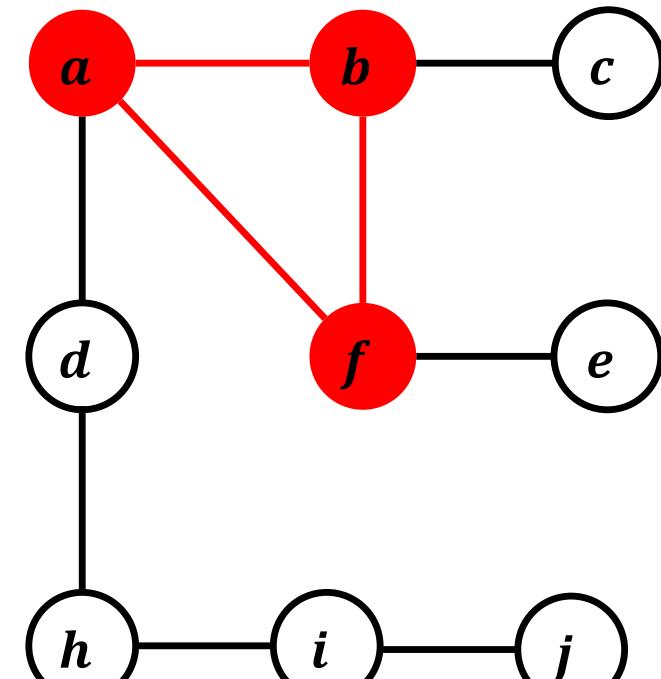
- 一至多棵树组成的无环图



树



森林

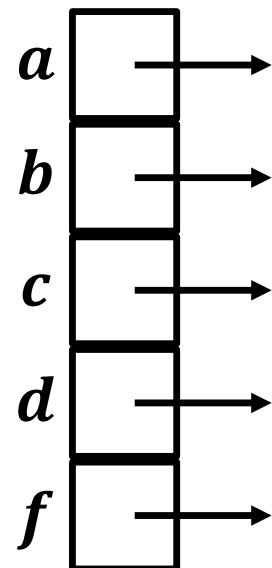
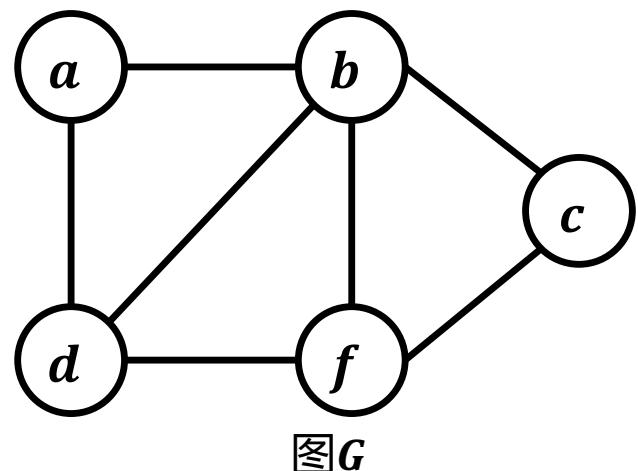


非树、非森林



图的表示：邻接链表

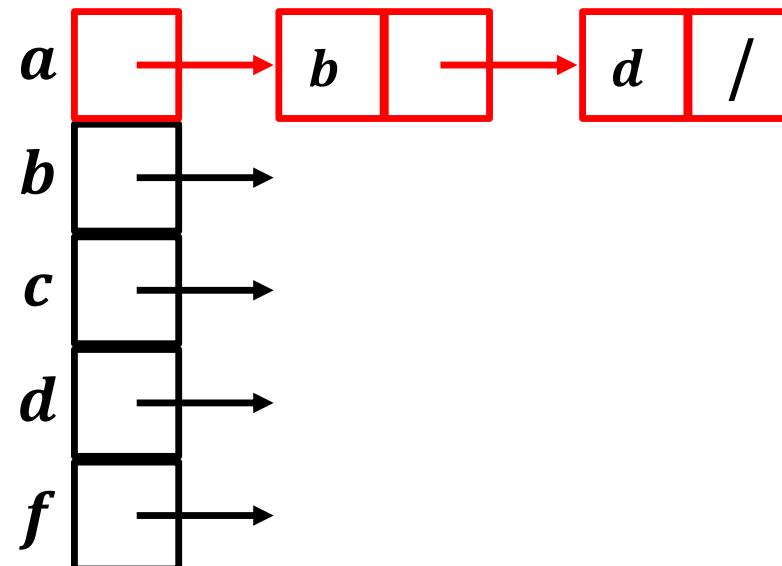
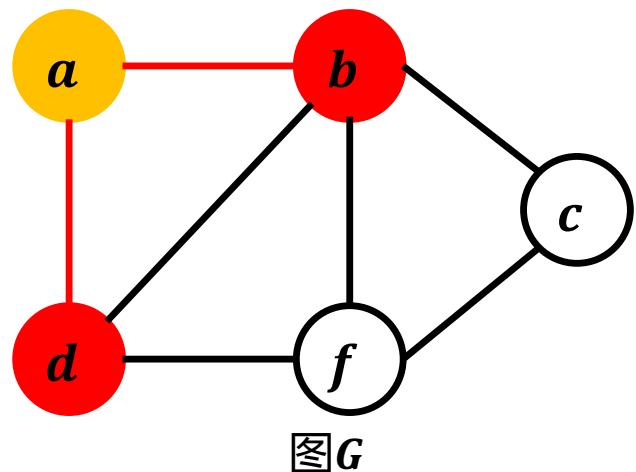
- 图 $G = \langle V, E \rangle$ ，其邻接链表由 $|V|$ 条链表的数组构成





图的表示：邻接链表

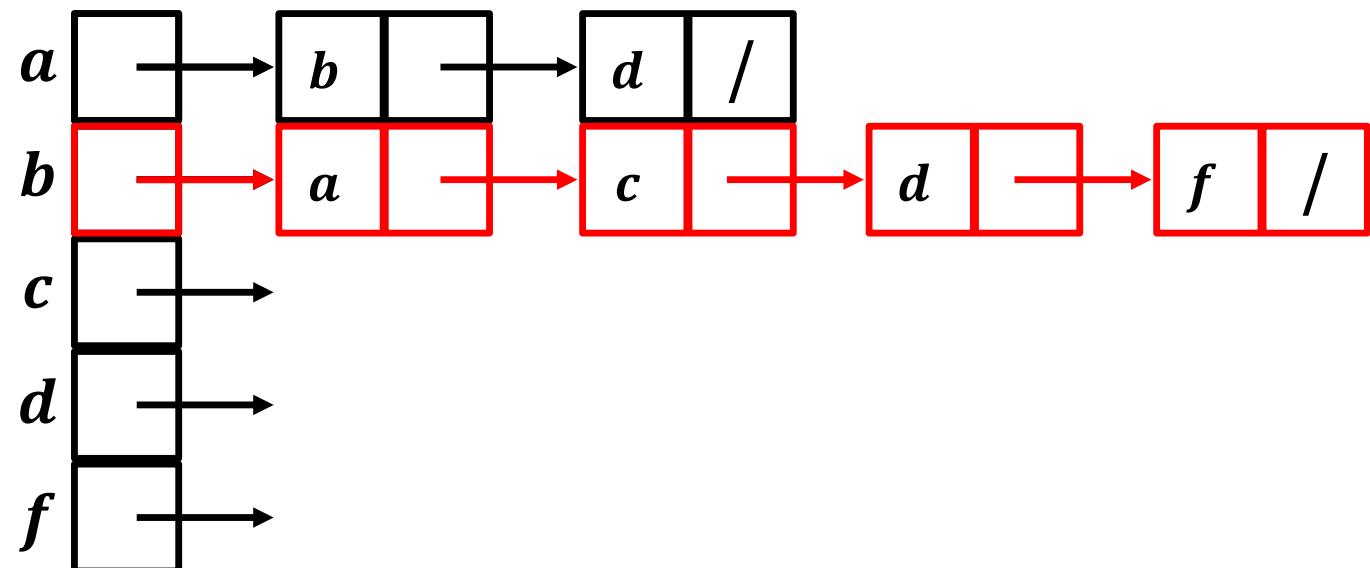
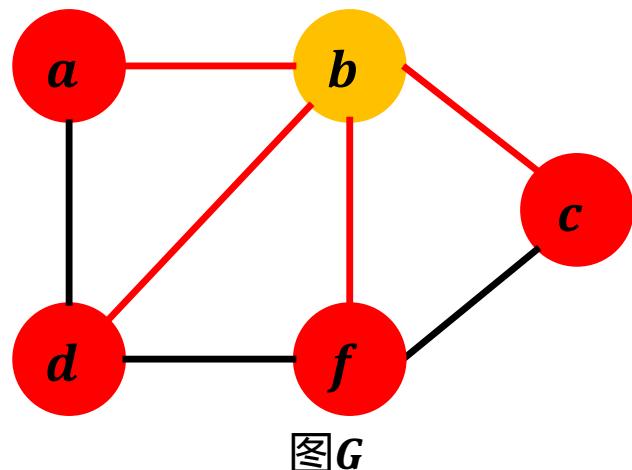
- 图 $G = \langle V, E \rangle$ ，其邻接链表由 $|V|$ 条链表的数组构成
- 每个顶点有一条链表，包含所有与其相邻的顶点
 - $Adj[a] = \{b, d\}$;





图的表示：邻接链表

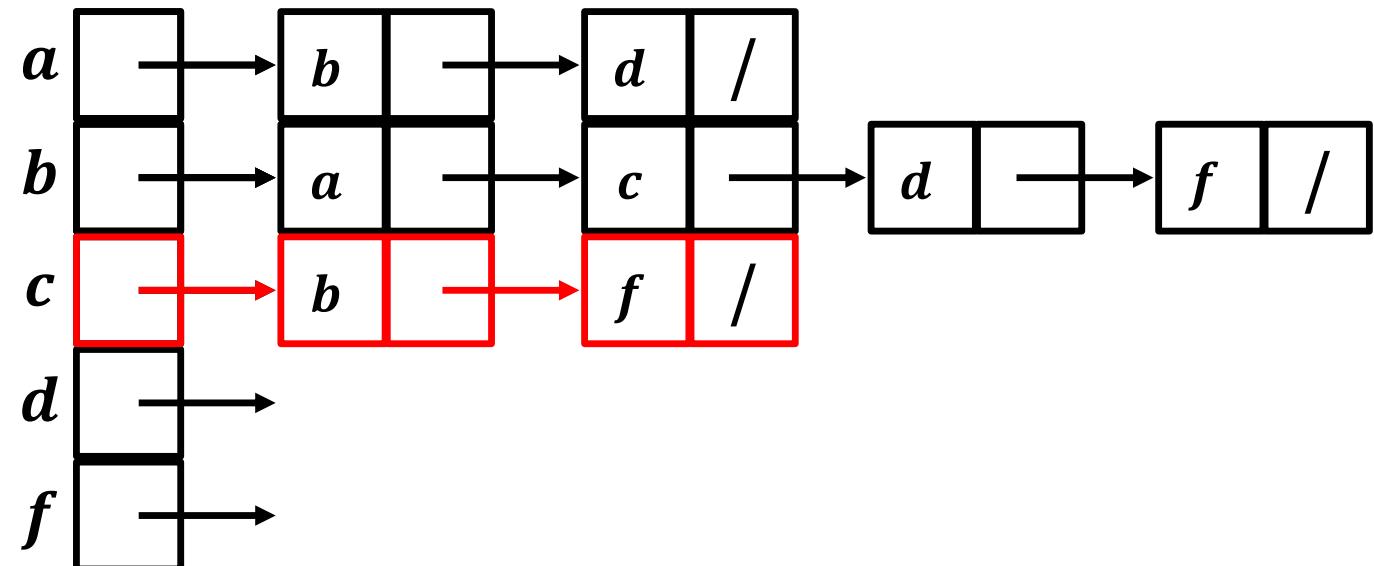
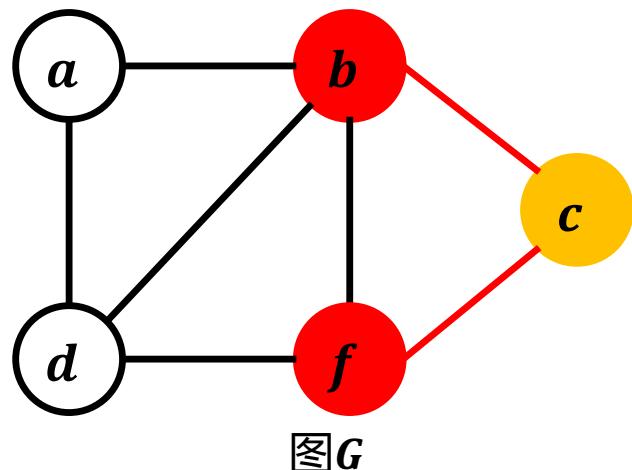
- 图 $G = \langle V, E \rangle$ ，其邻接链表由 $|V|$ 条链表的数组构成
- 每个顶点有一条链表，包含所有与其相邻的顶点
 - $Adj[a] = \{b, d\}; Adj[b] = \{a, c, d, f\};$





图的表示：邻接链表

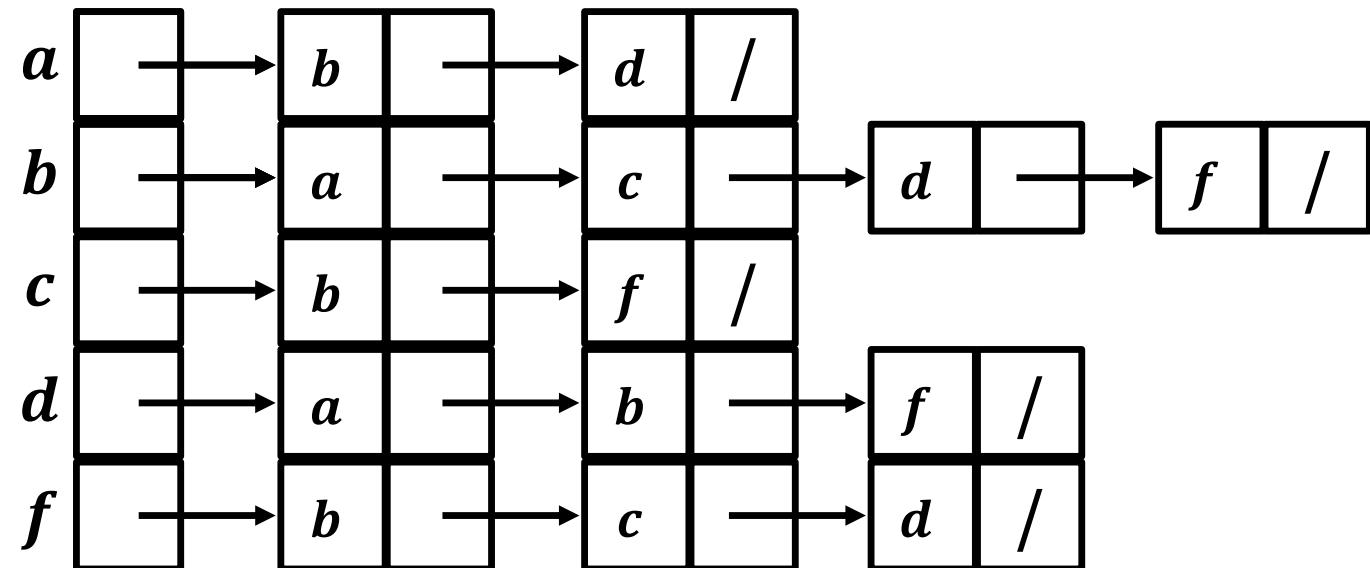
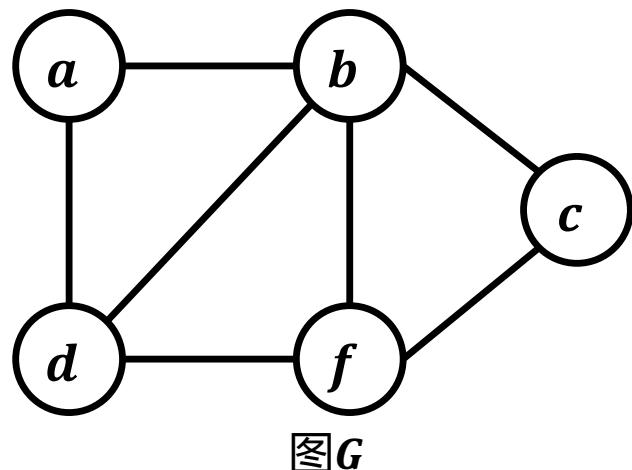
- 图 $G = \langle V, E \rangle$ ，其邻接链表由 $|V|$ 条链表的数组构成
- 每个顶点有一条链表，包含所有与其相邻的顶点
 - $Adj[a] = \{b, d\}; Adj[b] = \{a, c, d, f\}; Adj[c] = \{b, f\};$





图的表示：邻接链表

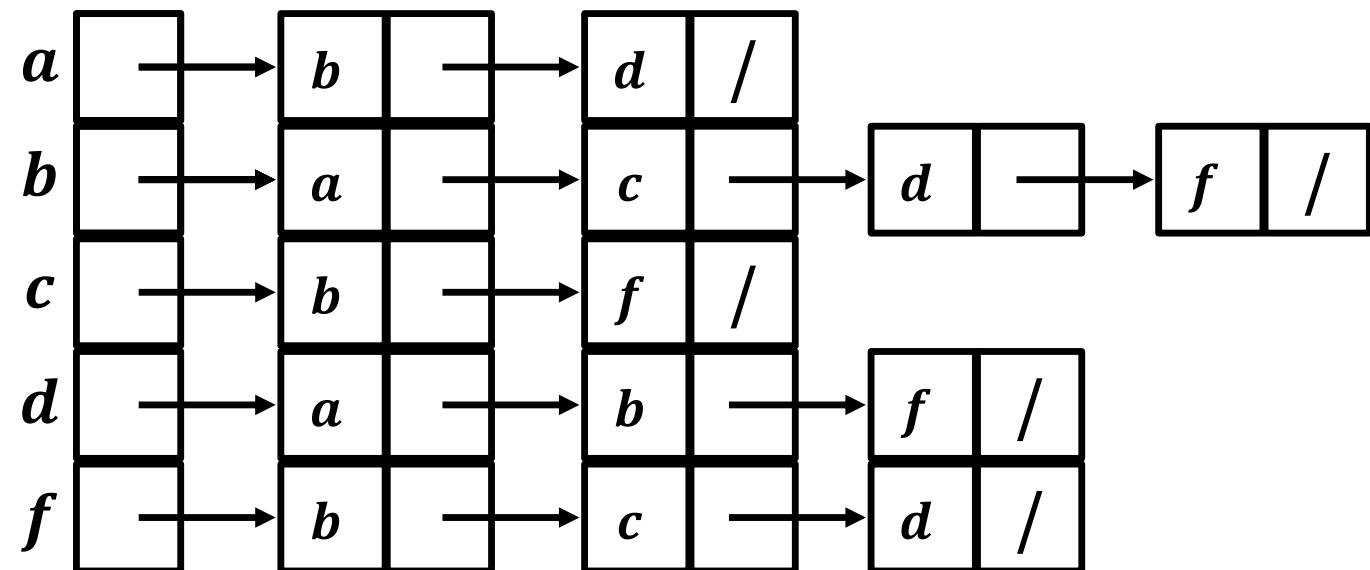
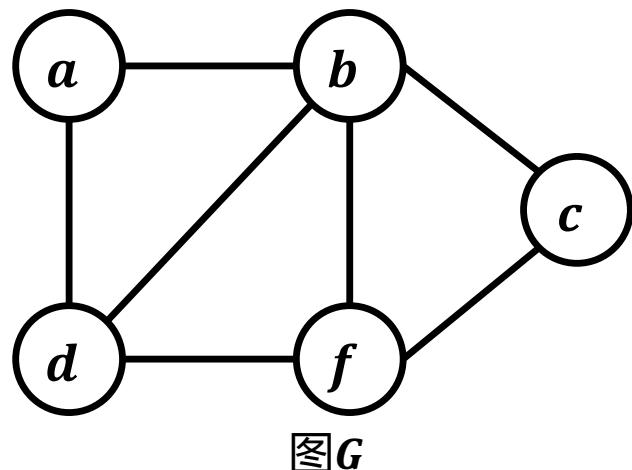
- 图 $G = \langle V, E \rangle$ ，其邻接链表由 $|V|$ 条链表的数组构成
- 每个顶点有一条链表，包含所有与其相邻的顶点
 - $Adj[a] = \{b, d\}; Adj[b] = \{a, c, d, f\}; Adj[c] = \{b, f\}; \dots$





图的表示：邻接链表

- 图 $G = \langle V, E \rangle$ ，其邻接链表由 $|V|$ 条链表的数组构成
- 每个顶点有一条链表，包含所有与其相邻的顶点
 - $Adj[a] = \{b, d\}; Adj[b] = \{a, c, d, f\}; Adj[c] = \{b, f\}; \dots$
- 空间大小 $O(|V| + |E|)$



图的表示：邻接矩阵



- 图 $G = \langle V, E \rangle$ 的邻接矩阵由 $|V| \times |V|$ 的二维数组 A 构成，满足：

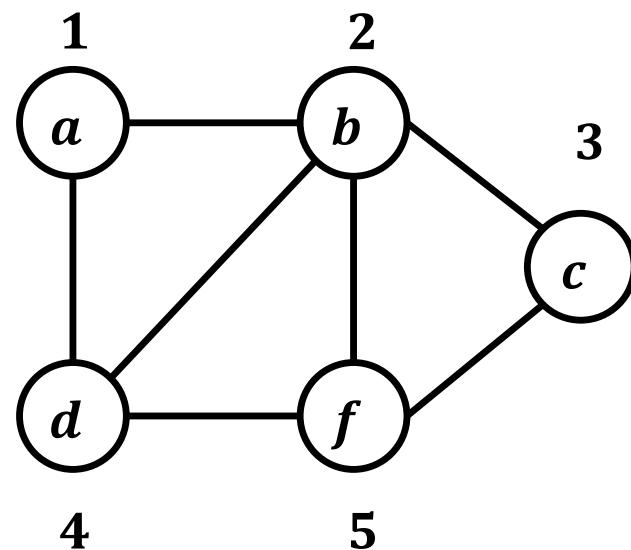
$$A_{ij} = \begin{cases} 1, & (i, j) \in E \\ 0, & (i, j) \notin E \end{cases}$$



图的表示：邻接矩阵

- 图 $G = \langle V, E \rangle$ 的邻接矩阵由 $|V| \times |V|$ 的二维数组 A 构成，满足：

$$A_{ij} = \begin{cases} 1, & (i, j) \in E \\ 0, & (i, j) \notin E \end{cases}$$



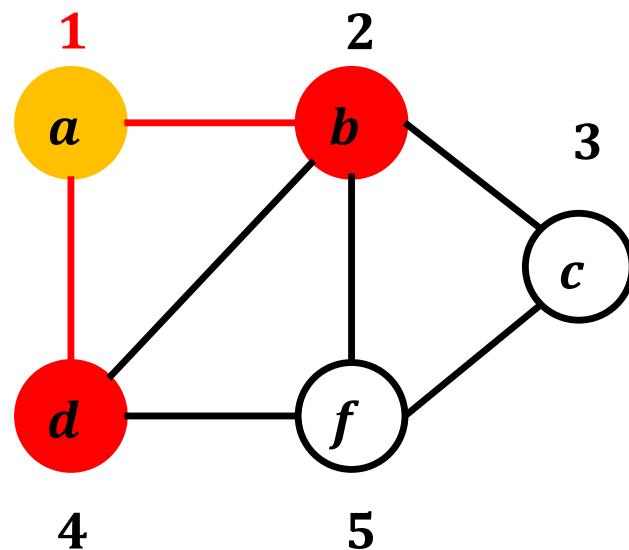
	a	b	c	d	f
a					
b					
c					
d					
f					



图的表示：邻接矩阵

- 图 $G = \langle V, E \rangle$ 的邻接矩阵由 $|V| \times |V|$ 的二维数组 A 构成，满足：

$$A_{ij} = \begin{cases} 1, & (i, j) \in E \\ 0, & (i, j) \notin E \end{cases}$$



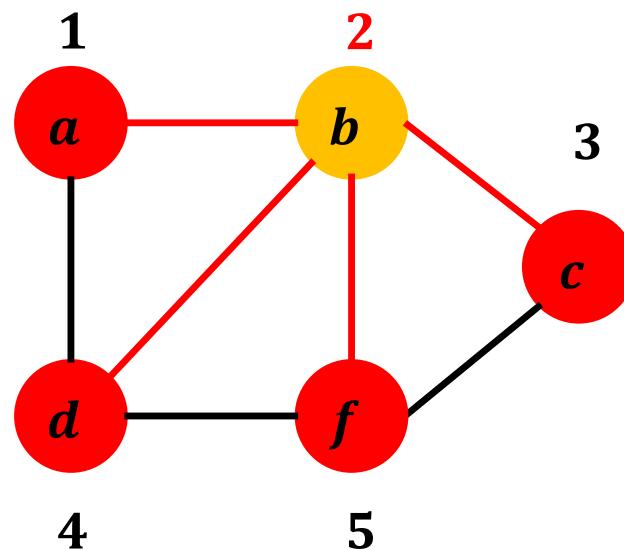
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>f</i>
<i>a</i>	1	0	1	0	1
<i>b</i>	2				
<i>c</i>					
<i>d</i>					
<i>f</i>					



图的表示：邻接矩阵

- 图 $G = \langle V, E \rangle$ 的邻接矩阵由 $|V| \times |V|$ 的二维数组 A 构成，满足：

$$A_{ij} = \begin{cases} 1, & (i, j) \in E \\ 0, & (i, j) \notin E \end{cases}$$



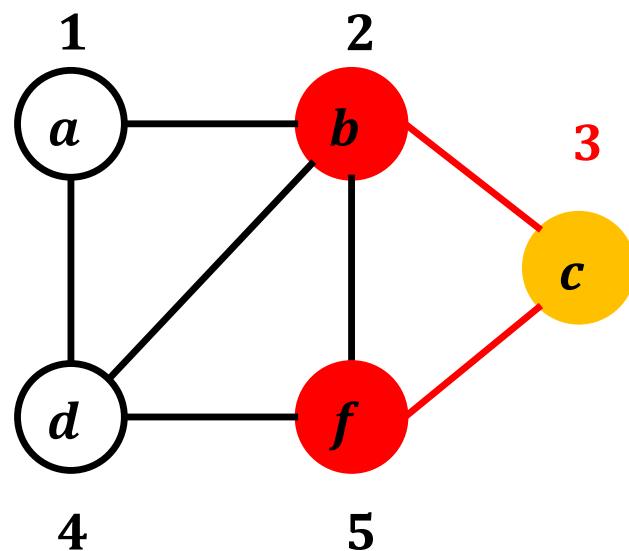
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>f</i>
<i>a</i>	1	0	1	0	1
<i>b</i>	2	1	0	1	1
<i>c</i>	3				
<i>d</i>	4				
<i>f</i>	5				



图的表示：邻接矩阵

- 图 $G = \langle V, E \rangle$ 的邻接矩阵由 $|V| \times |V|$ 的二维数组 A 构成，满足：

$$A_{ij} = \begin{cases} 1, & (i, j) \in E \\ 0, & (i, j) \notin E \end{cases}$$



	a	b	c	d	f
a	1	2	3	4	5
b	2	1	0	1	1
c	3	0	1	0	0
d	4				
f	5				

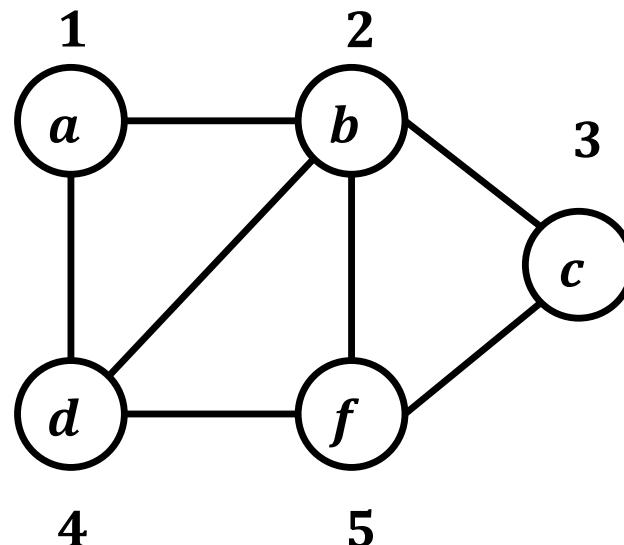


图的表示：邻接矩阵

- 图 $G = \langle V, E \rangle$ 的邻接矩阵由 $|V| \times |V|$ 的二维数组 A 构成，满足：

$$A_{ij} = \begin{cases} 1, & (i, j) \in E \\ 0, & (i, j) \notin E \end{cases}$$

- 空间大小 $O(|V|^2)$, $O(1)$ 判断是否有边



	a	b	c	d	f	
a	1	0	1	0	1	0
b	2	1	0	1	1	1
c	3	0	1	0	0	1
d	4	1	1	0	0	1
f	5	0	1	1	1	0

小结

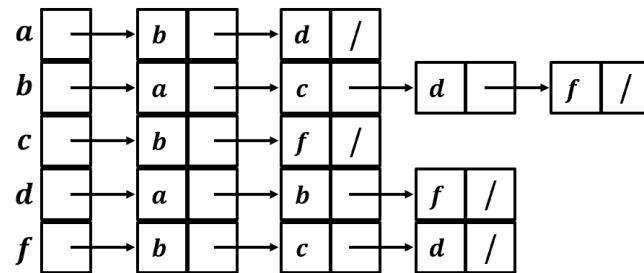
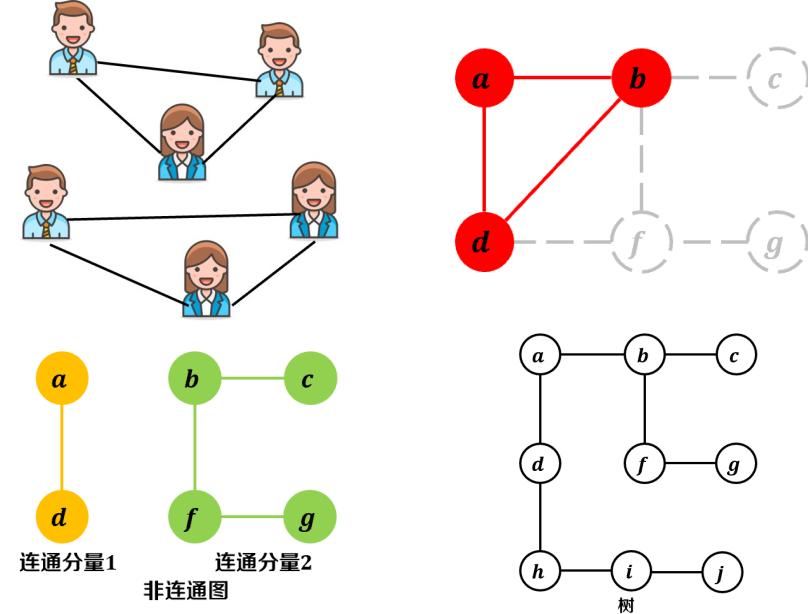


图的概念

- 图的定义、相邻与关联
- 顶点的度与图的度、握手定理
- 路径与环路
- 连通、连通分量
- 子图、生成子图、树

图的表示

- 邻接链表与邻接矩阵



	a	b	c	d	f	
a	1	0	1	0	1	0
b	2	1	0	1	1	1
c	3	0	1	0	0	1
d	4	1	1	0	0	1
f	5	0	1	1	1	0

图算法篇：广度优先搜索

北京航空航天大学
计算机学院

提纲



图的搜索

算法思想

算法伪代码

算法实例

算法分析

算法应用



图的搜索

- 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值

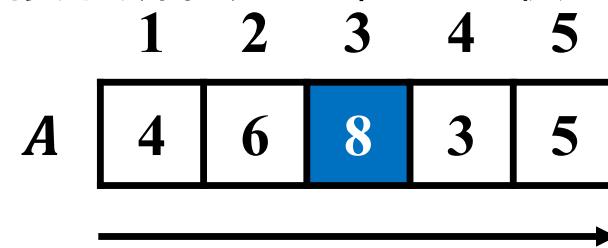
	1	2	3	4	5
A	4	6	8	3	5

图的搜索



● 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值

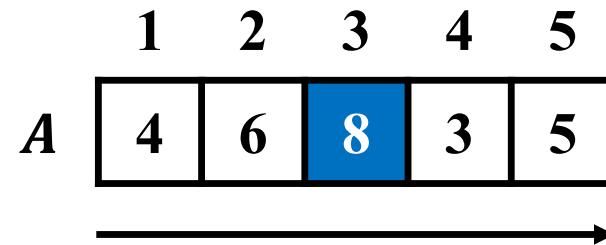




图的搜索

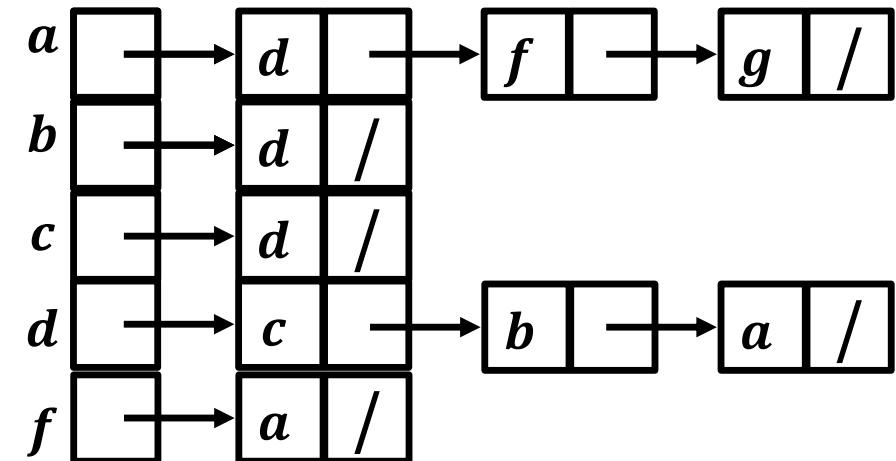
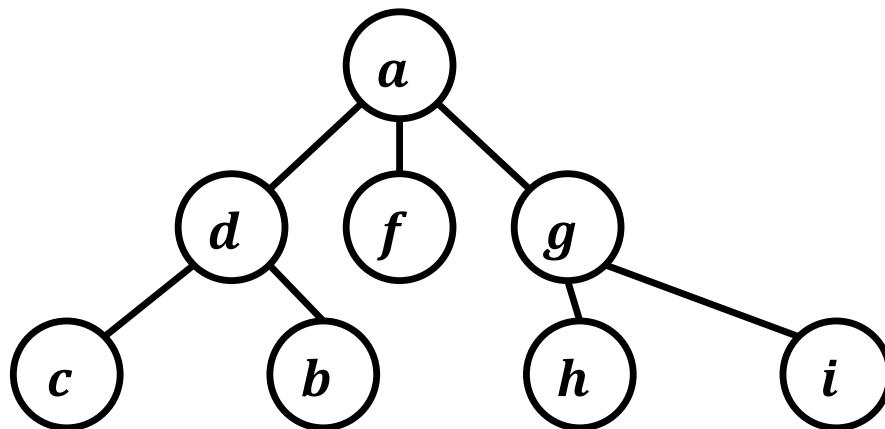
- 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值



- 图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边



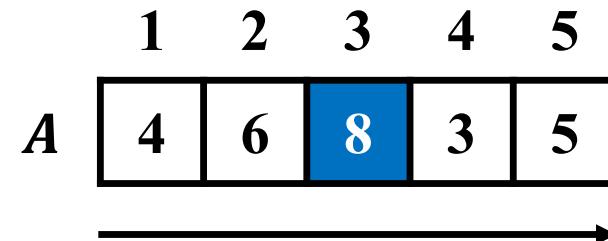
...

图的搜索



• 数组结构

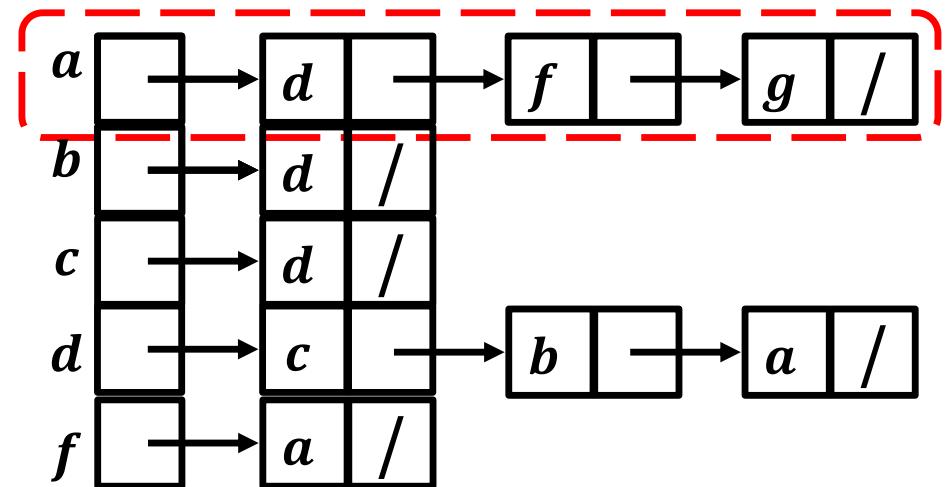
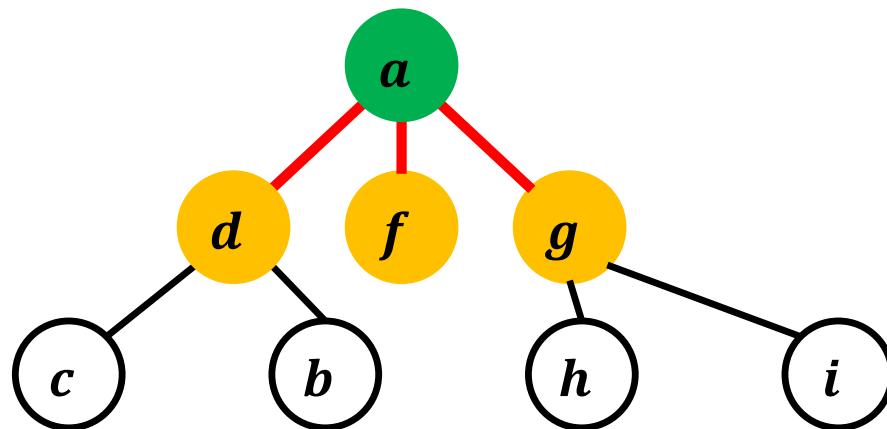
- 查询最大值：简单循环搜索所有元素，记录最大值



• 图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边

顶点 a 与 $\{d, f, g\}$ 相邻

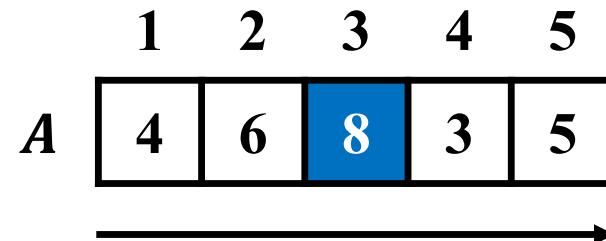




图的搜索

- 数组结构

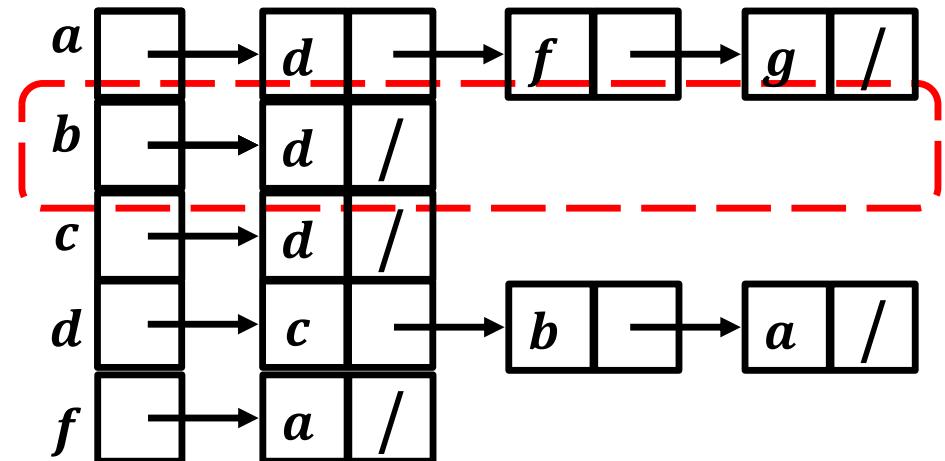
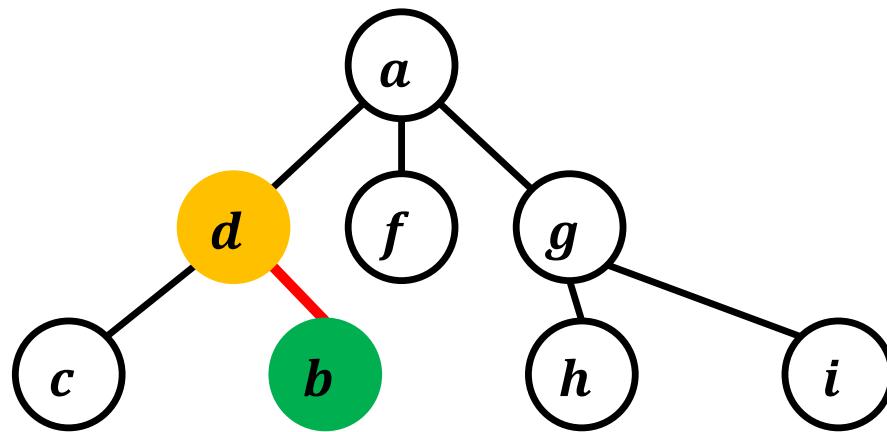
- 查询最大值：简单循环搜索所有元素，记录最大值



- 图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边

顶点 b 与 $\{d\}$ 相邻



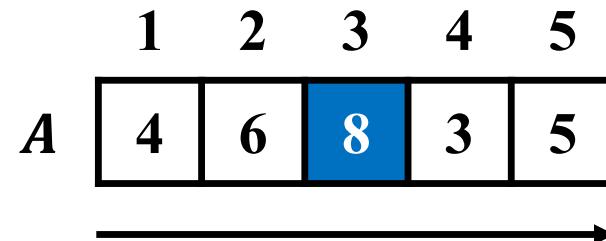
...



图的搜索

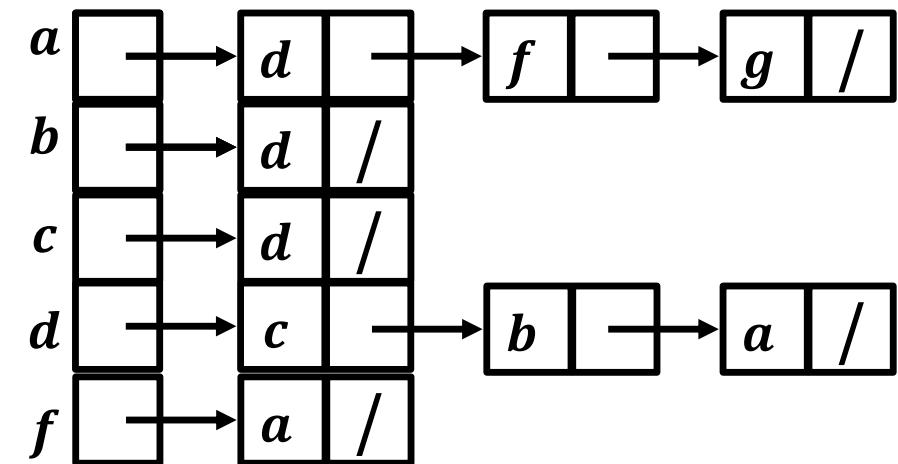
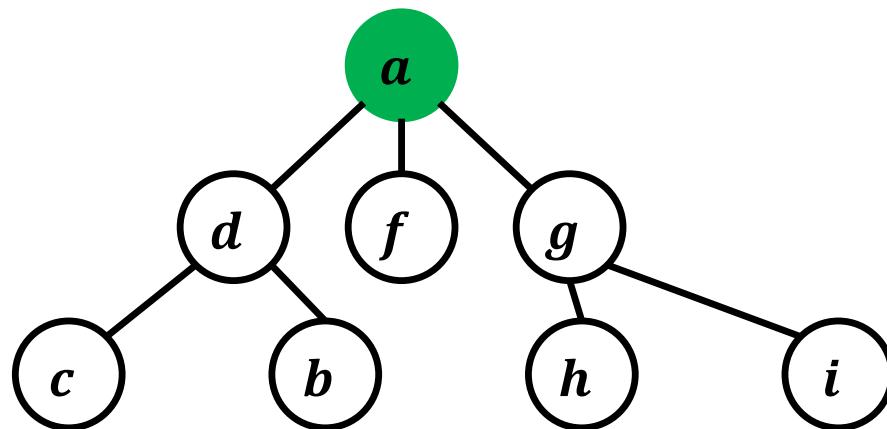
- 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值



- 图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边
- 查询可达顶点

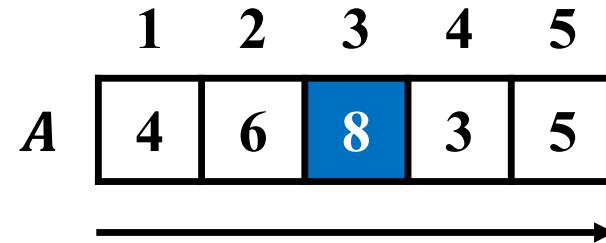




图的搜索

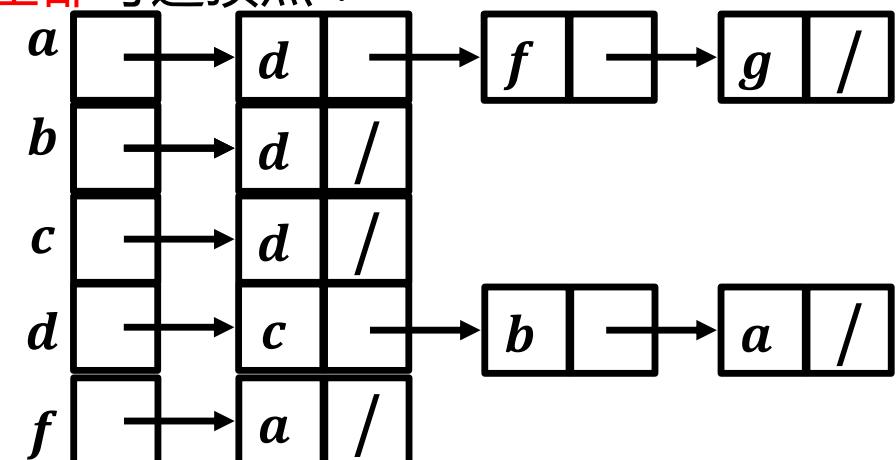
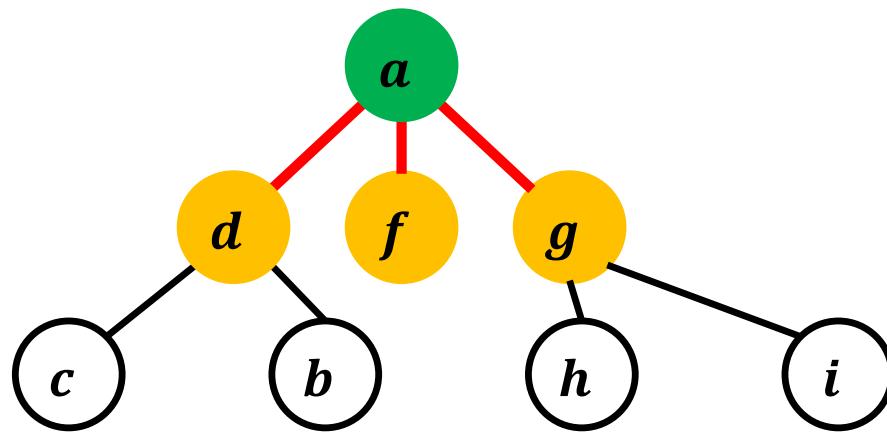
• 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值



• 图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边
- 查询**可达顶点**：简单循环搜索，**不能找到全部可达顶点！**

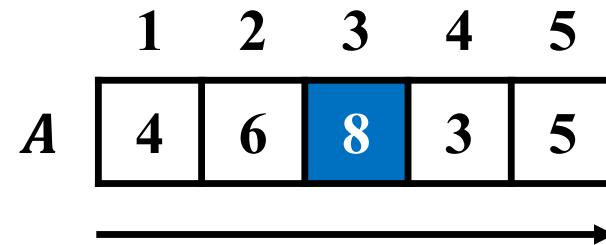


图的搜索



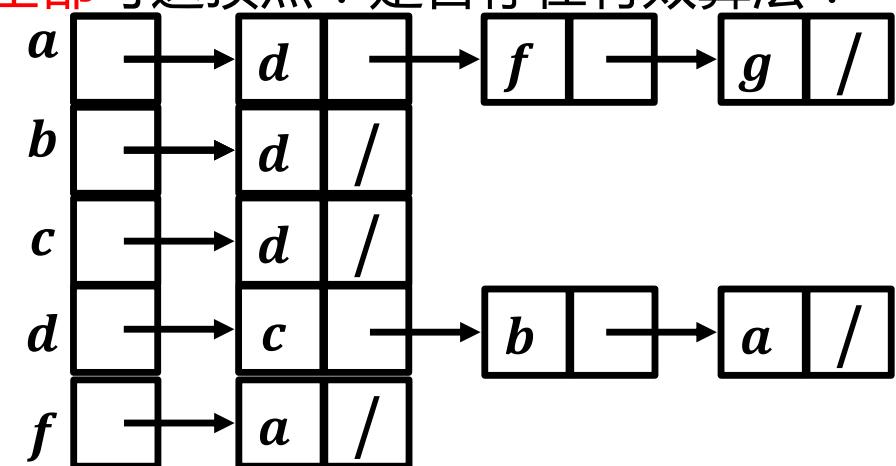
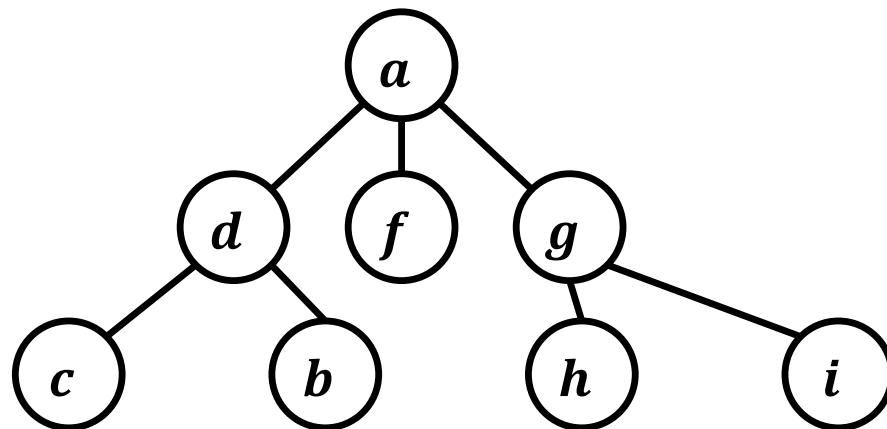
• 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值



• 图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边
- 查询可达顶点：简单循环搜索，**不能找到全部可达顶点！是否存在有效算法？**



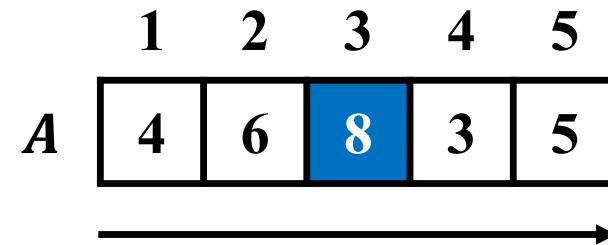
...



图的搜索

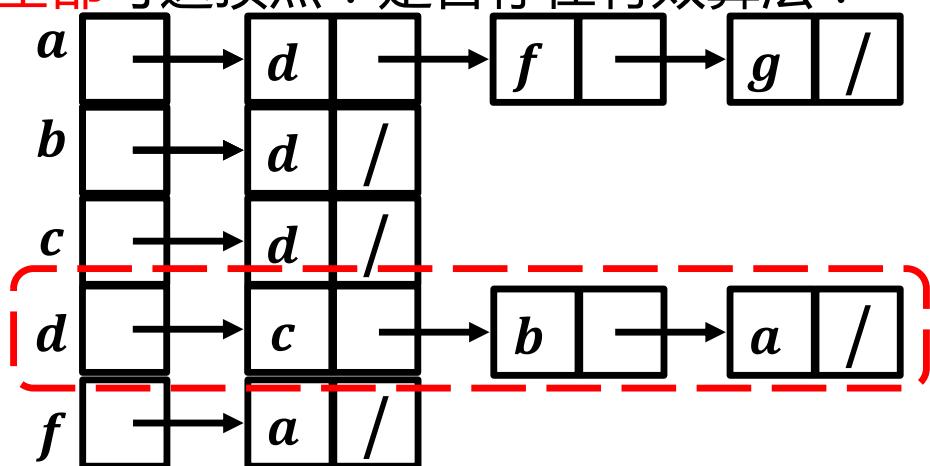
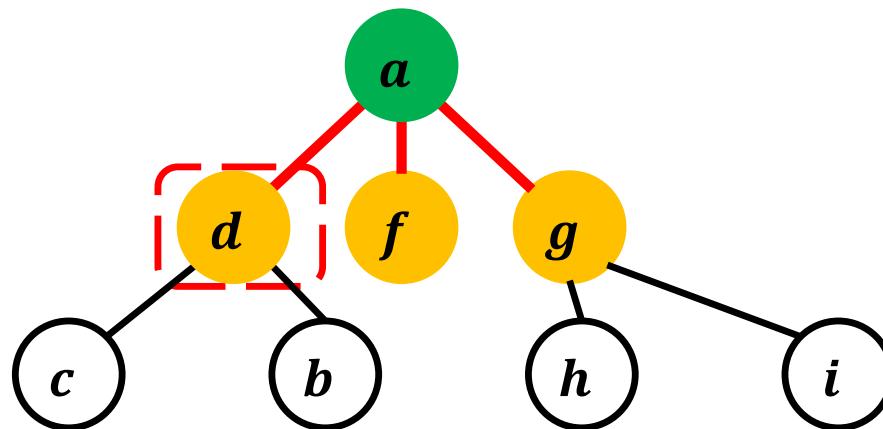
- 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值



- 图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边
- 查询可达顶点：简单循环搜索，**不能找到全部可达顶点！是否存在有效算法？**

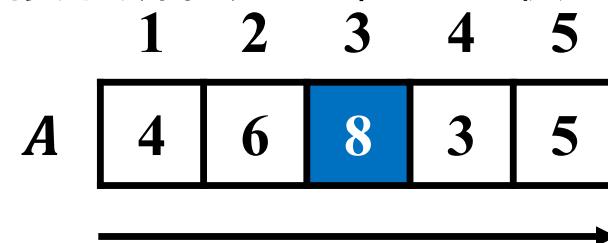


图的搜索



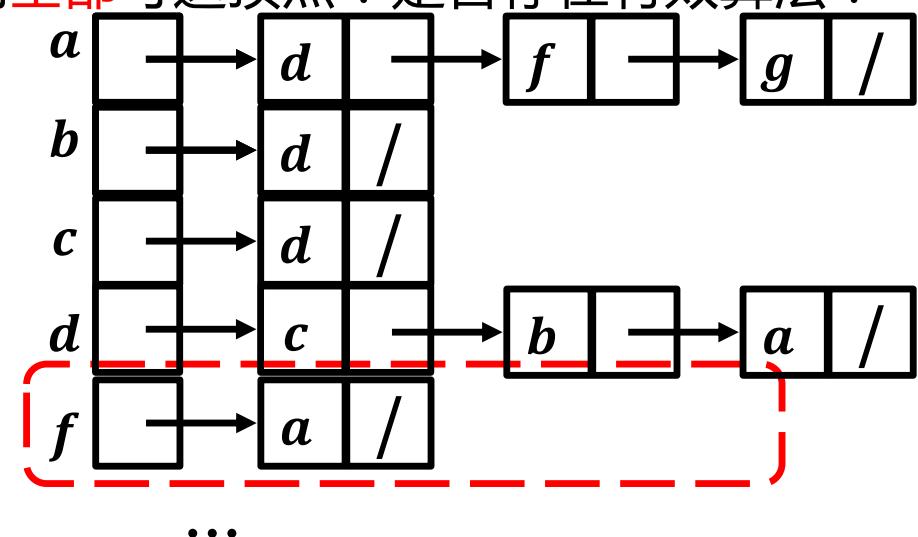
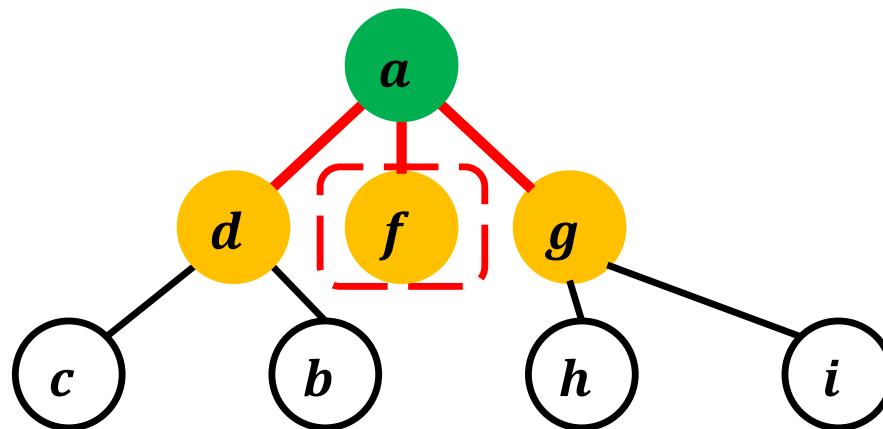
数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值



图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边
- 查询可达顶点：简单循环搜索，**不能找到全部可达顶点！是否存在有效算法？**

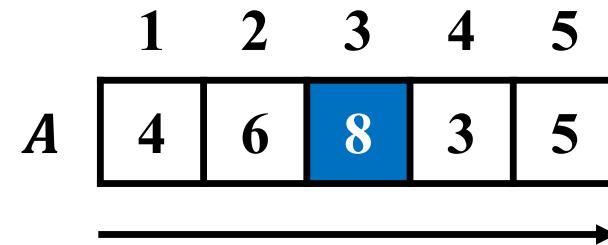


图的搜索



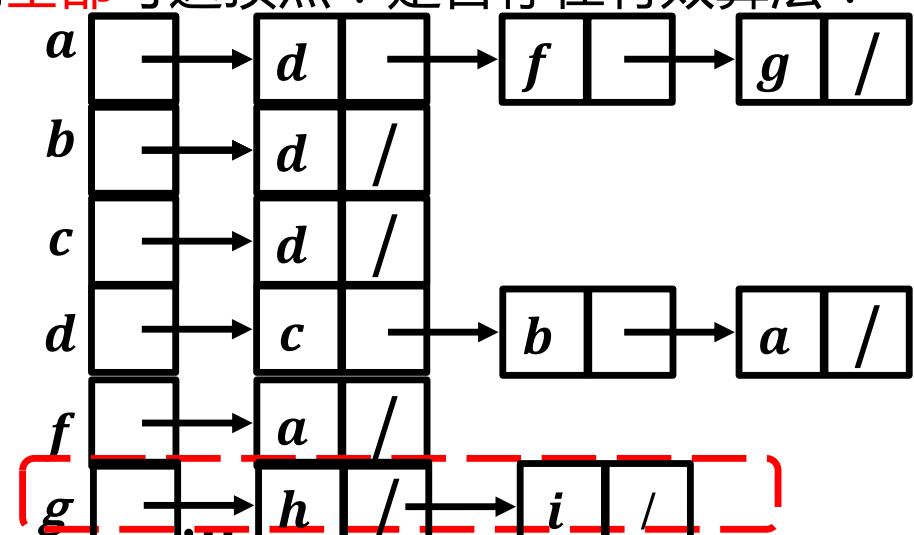
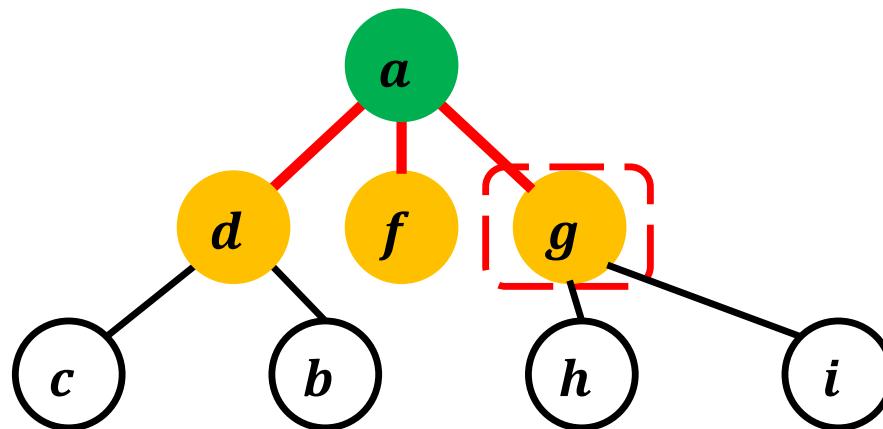
• 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值



• 图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边
- 查询可达顶点：简单循环搜索，**不能找到全部可达顶点！是否存在有效算法？**

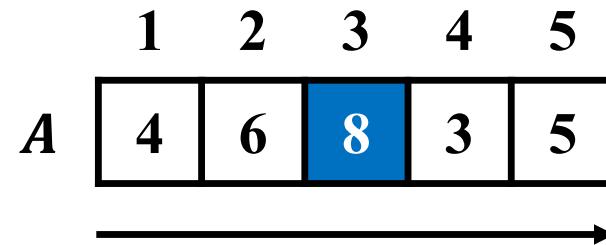




图的搜索

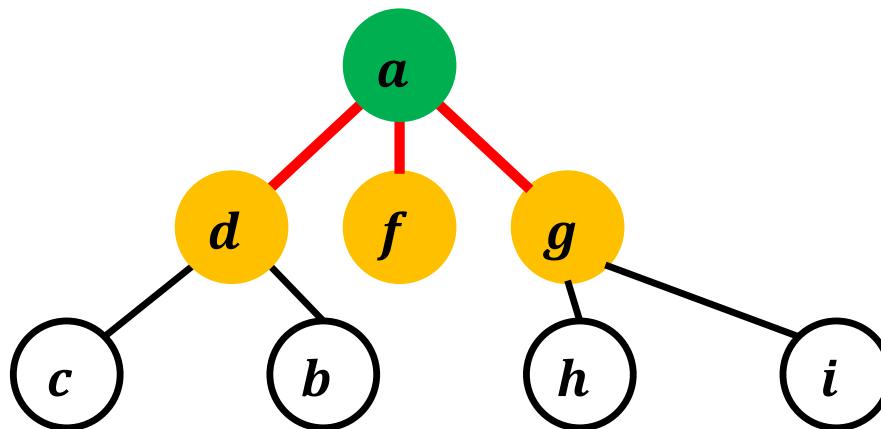
- 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值



- 图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边
- 查询可达顶点：简单循环搜索，不能找到全部可达顶点！是否存在有效算法？



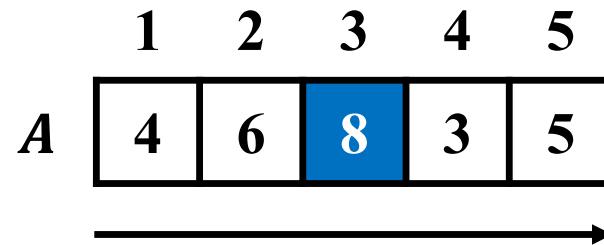
按照什么次序搜索顶点？



图的搜索

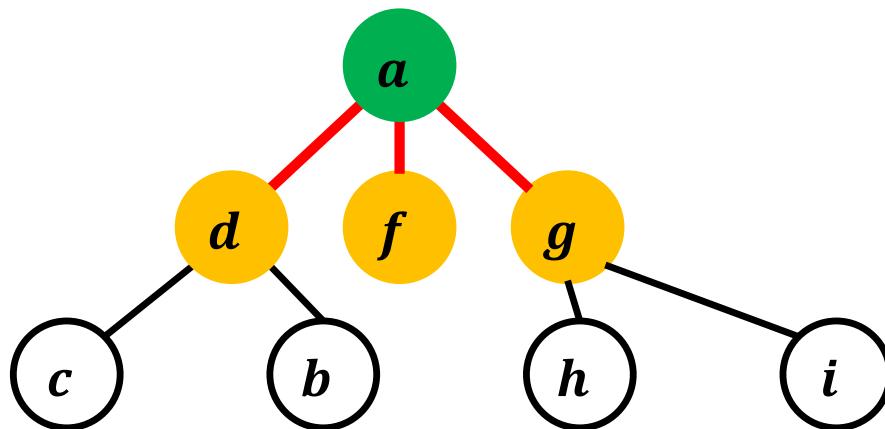
- 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值



- 图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边
- 查询可达顶点：简单循环搜索，不能找到全部可达顶点！是否存在有效算法？



按照什么次序搜索顶点？

广度优先搜索

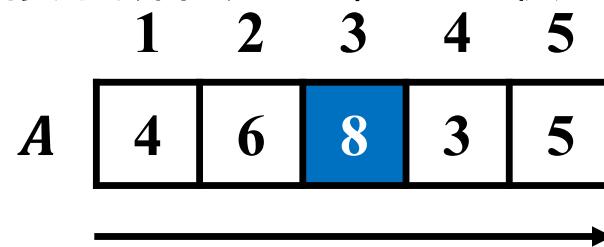
深度优先搜索



图的搜索

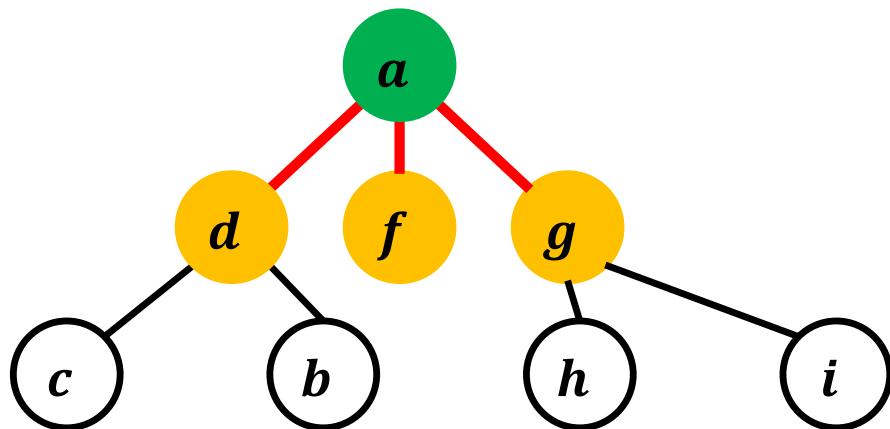
- 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值



- 图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边
- 查询可达顶点：简单循环搜索，不能找到全部可达顶点！是否存在有效算法？



按照什么次序搜索顶点？

广度优先搜索

深度优先搜索

提纲



图的搜索

算法思想

算法伪代码

算法实例

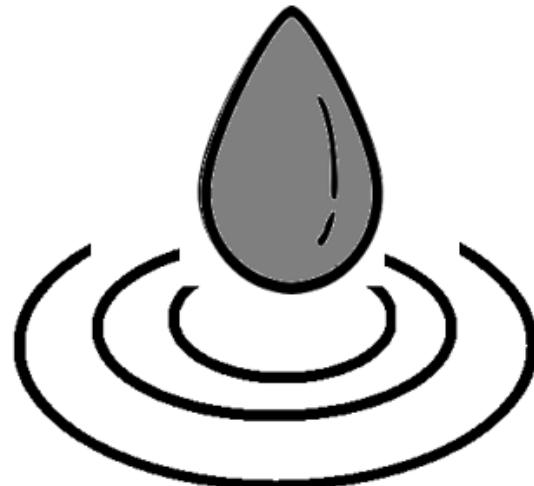
算法分析

算法应用

算法思想：现象启发



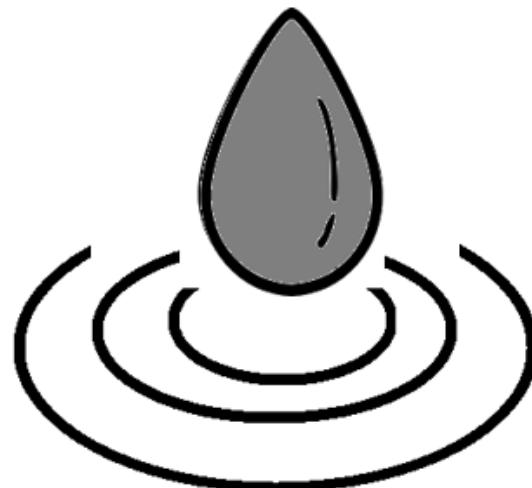
- 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



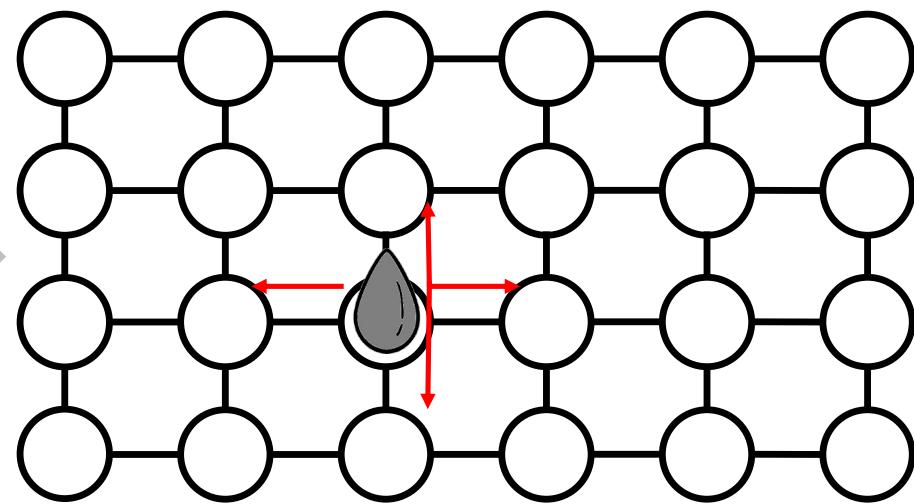
算法思想：现象启发



- 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



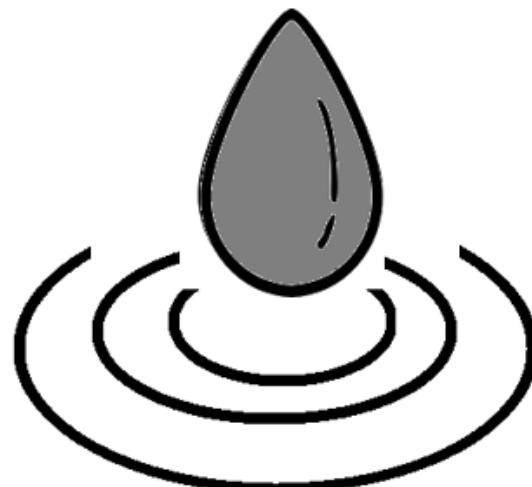
图结构



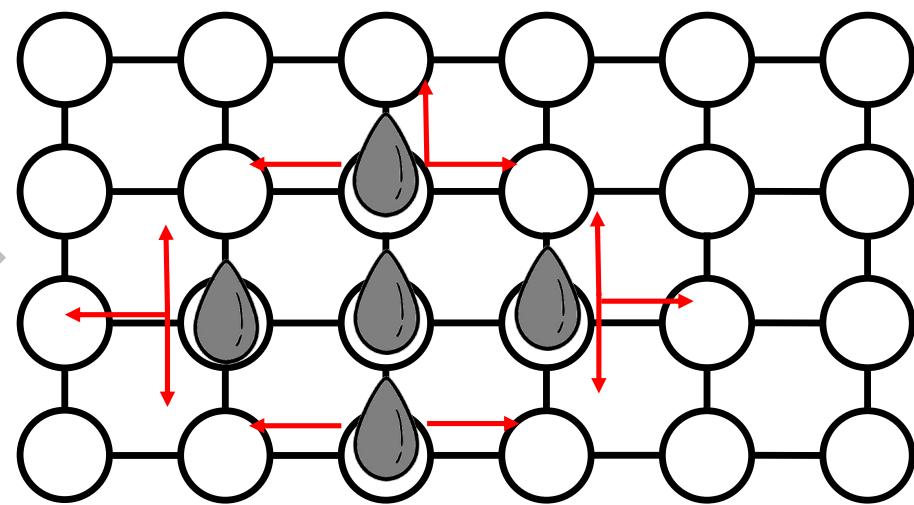
算法思想：现象启发



- 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



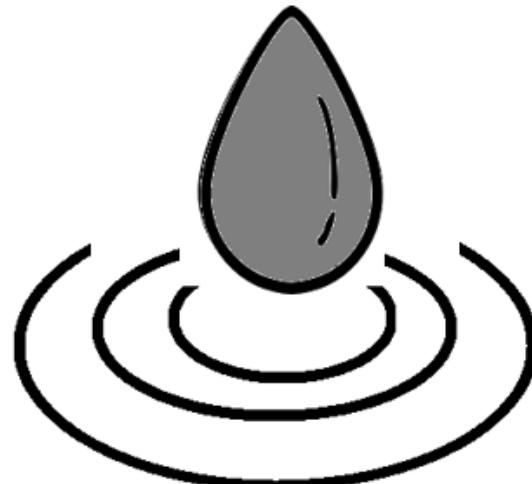
图结构



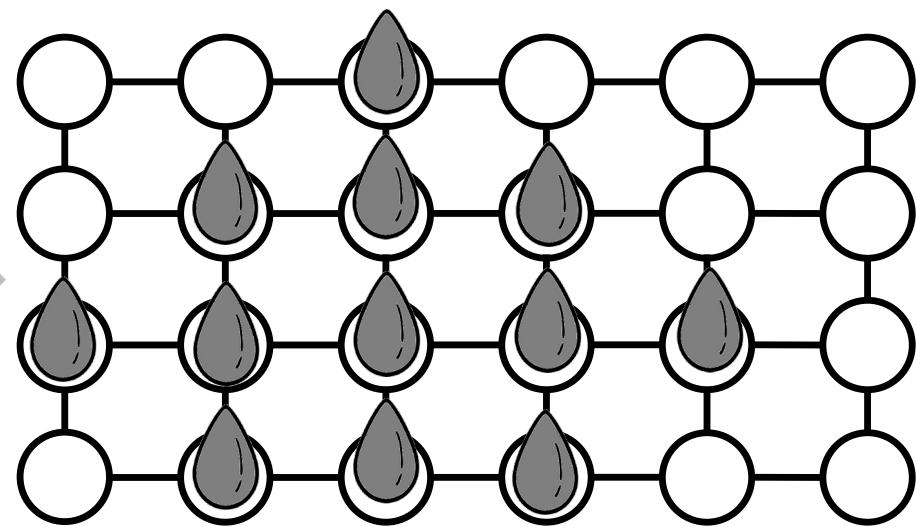
算法思想：现象启发



- 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



图结构

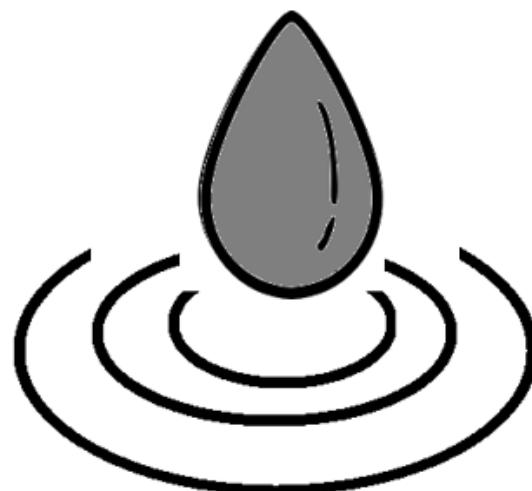


这种依次扩散的现象，蕴含了搜索图结构的一种顺序

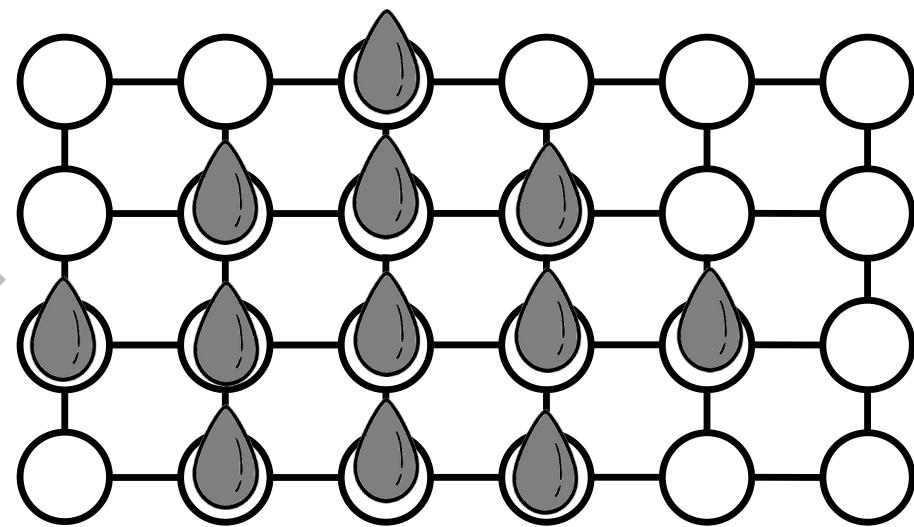
算法思想：现象启发



- 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



图结构

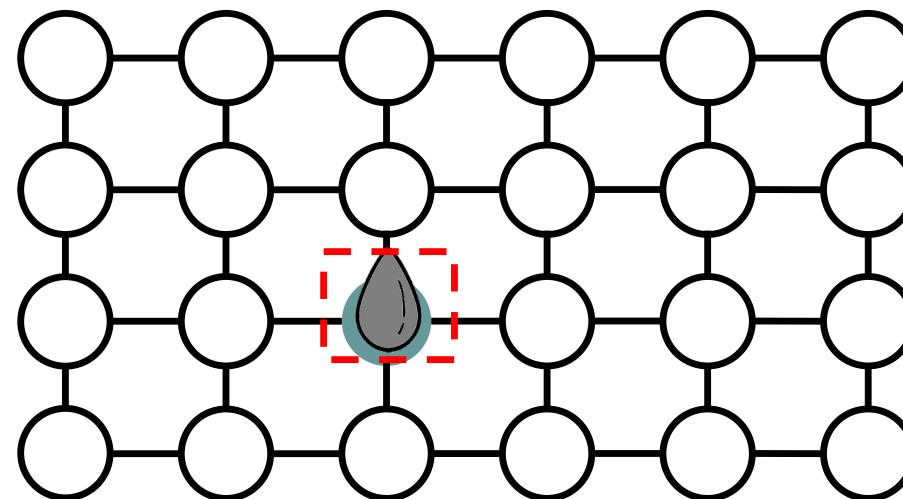


处理某顶点时，一次性发现所有其相邻顶点

算法思想：广度优先搜索



- 核心思想
 - 处理某顶点时，一次性发现其所有相邻顶点

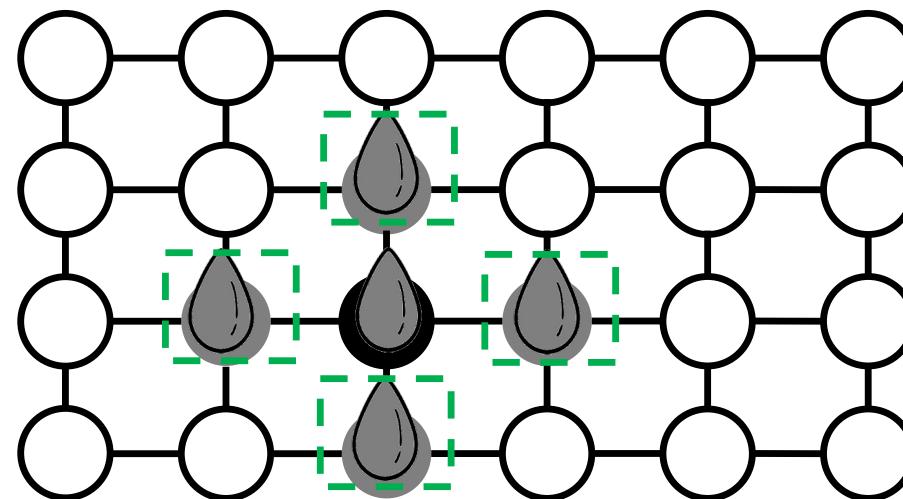


算法思想：广度优先搜索



- 核心思想

- 处理某顶点时，一次性发现其所有相邻顶点



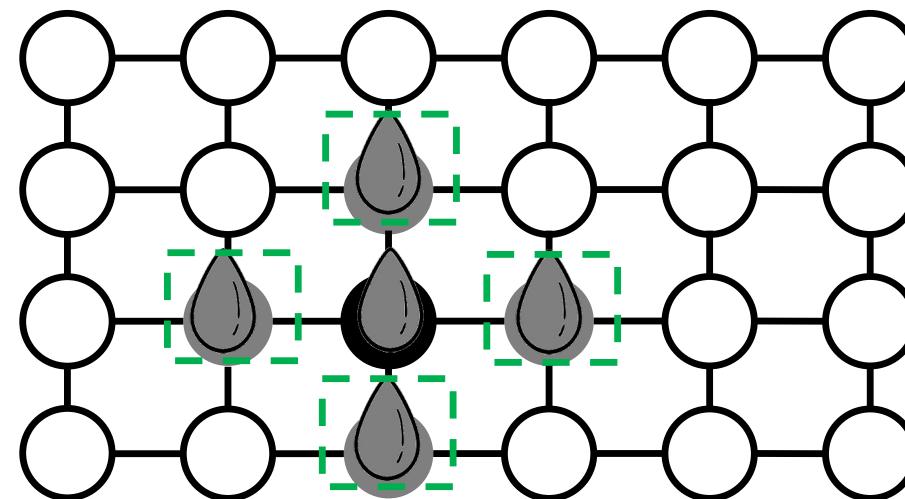
算法思想：广度优先搜索



- 核心思想

- 处理某顶点时，一次性发现其所有相邻顶点

扩散中多处同时进行，一次只能处理单个顶点怎么办？

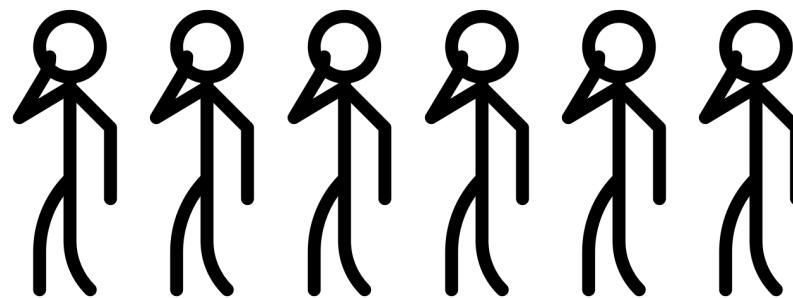
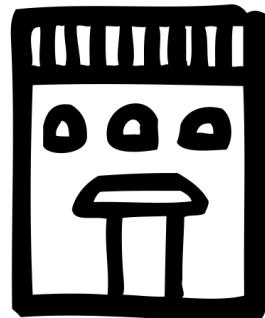


算法思想：队列



- 队列

- 先来先服务：队尾加入，队首离开
 - 加入队列， $Q.Enqueue()$
 - 离开队列， $Q.Dequeue()$



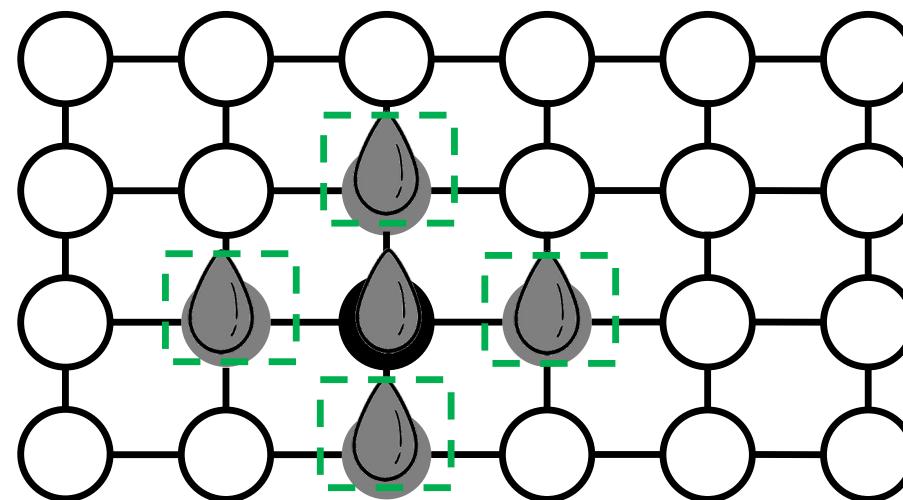
队列 Q

算法思想：广度优先搜索



- 核心思想

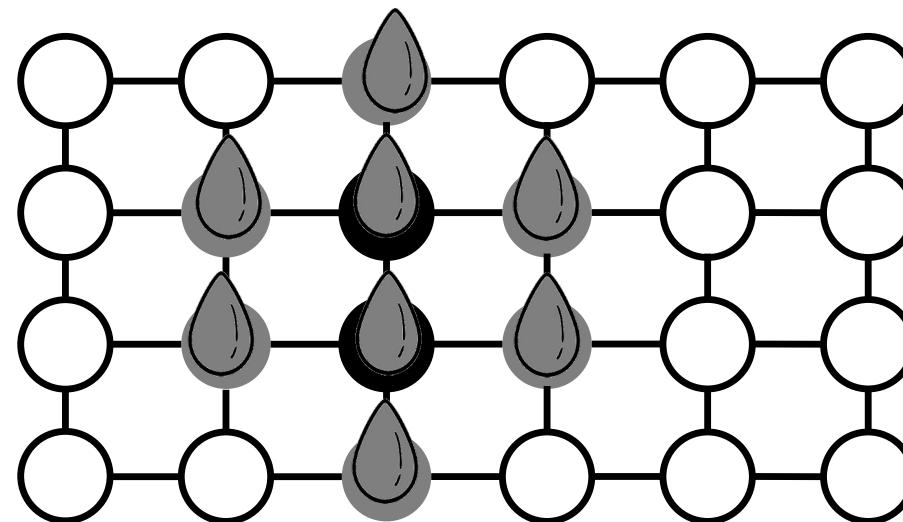
- 处理某顶点时，一次性发现其所有相邻顶点，未处理顶点加入等待队列



算法思想：广度优先搜索



- 辅助数组
 - $color$ 表示顶点状态

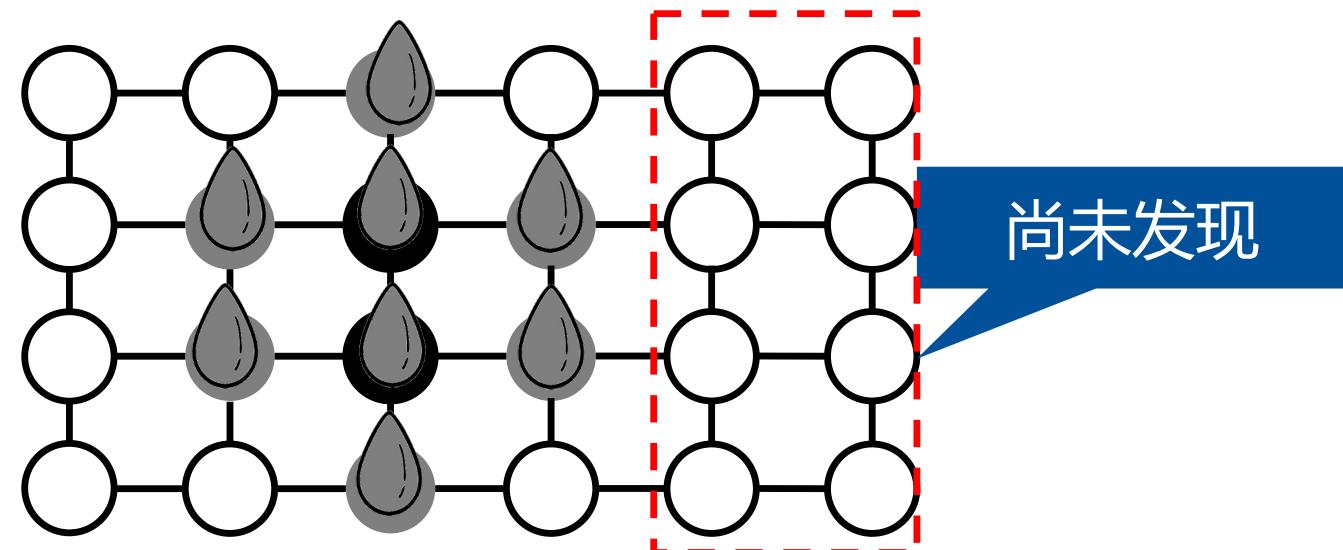


算法思想：广度优先搜索



- 辅助数组

- $color$ 表示顶点状态
 - $White$ ：白色顶点 u 尚未被发现，发现后直接入队

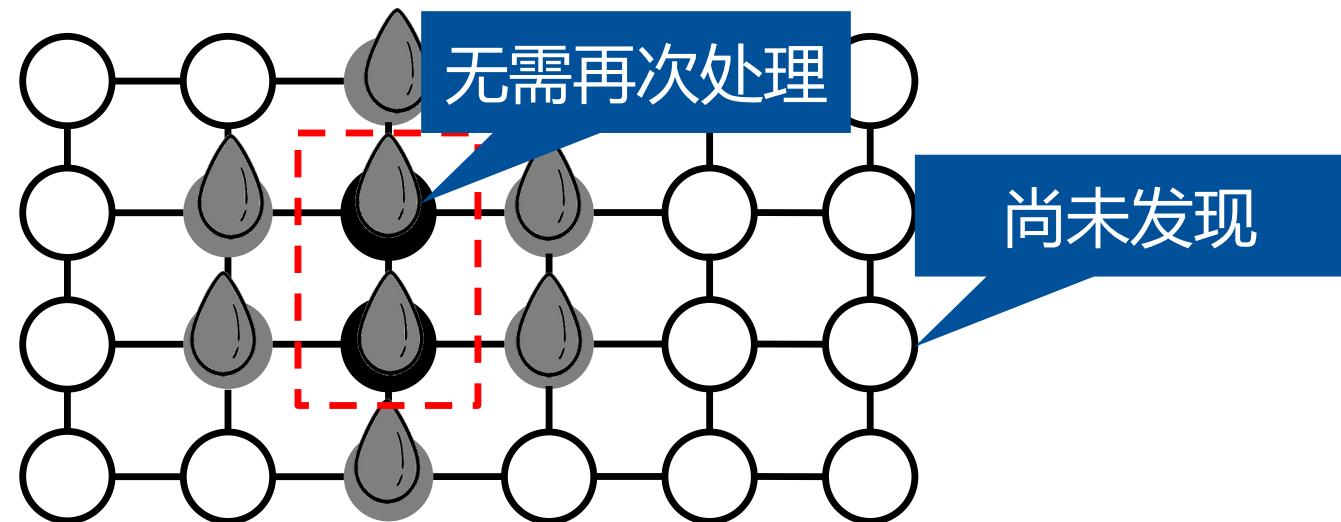


算法思想：广度优先搜索



- 辅助数组

- $color$ 表示顶点状态
 - $White$ ：白色顶点 u 尚未被发现，发现后直接入队
 - $Black$ ：黑色顶点 u 已被处理，无需再次入队

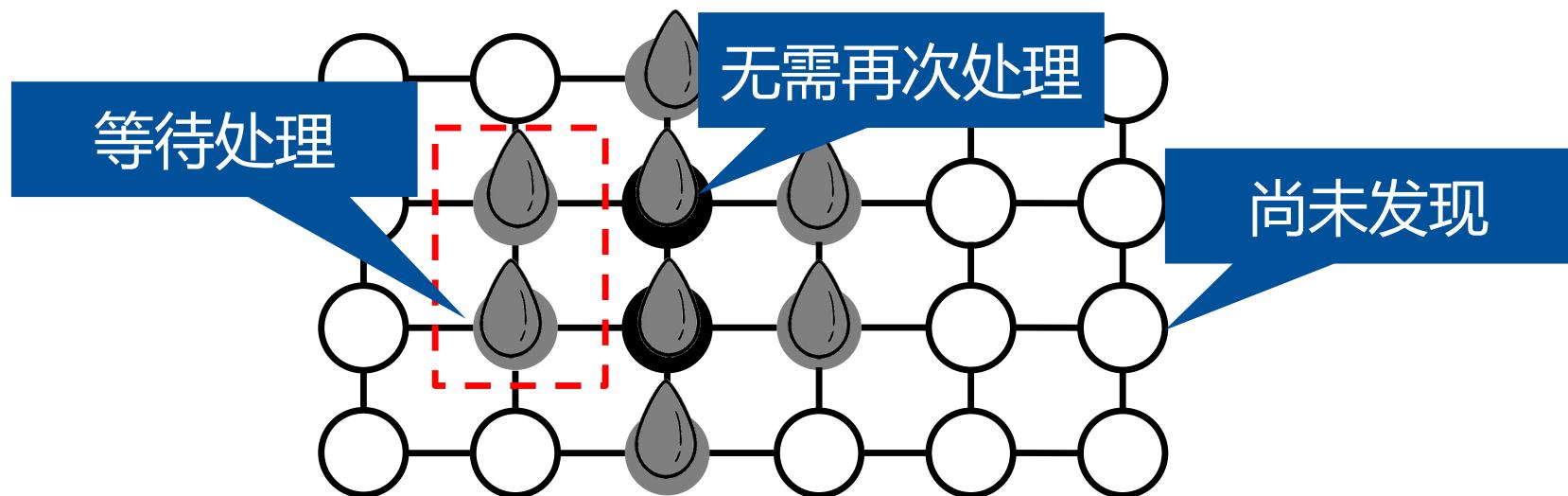


算法思想：广度优先搜索



- 辅助数组

- $color$ 表示顶点状态
 - $White$ ：白色顶点 u 尚未被发现，发现后直接入队
 - $Black$ ：黑色顶点 u 已被处理，无需再次入队
 - $Gray$ ：灰色顶点 u 已加入队列，无需再次入队

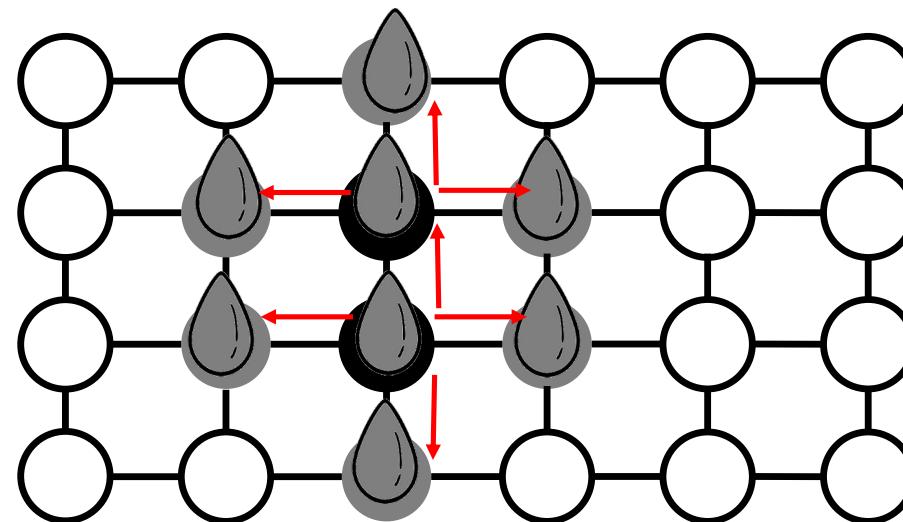


算法思想：广度优先搜索



- 辅助数组

- $color$ 表示顶点状态
 - $White$ ：白色顶点 u 尚未被发现，发现后直接入队
 - $Black$ ：黑色顶点 u 已被处理，无需再次入队
 - $Gray$ ：灰色顶点 u 已加入队列，无需再次入队
- $pred$ ：顶点 u 由 $pred[u]$ 发现

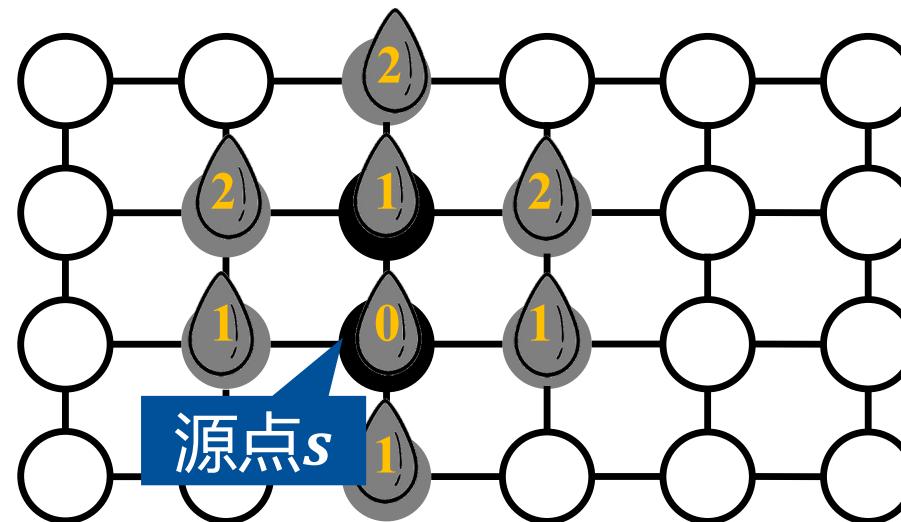


算法思想：广度优先搜索



- 辅助数组

- $color$ 表示顶点状态
 - $White$ ：白色顶点 u 尚未被发现，发现后直接入队
 - $Black$ ：黑色顶点 u 已被处理，无需再次入队
 - $Gray$ ：灰色顶点 u 已加入队列，无需再次入队
- $pred$ ：顶点 u 由 $pred[u]$ 发现
- $dist$ ：顶点 u 距离源点 s 的距离



提纲



图的搜索

算法思想

算法伪代码

算法实例

算法分析

算法应用



伪代码

- **BFS(G, s)**

输入: 图 G , 源点 s

输出: 前驱数组 $pred[]$, 距离数组 $dist[]$

新建一维数组 $color[1..|V|]$, $pred[1..|V|]$, $dist[1..|V|]$

新建空队列 Q

//初始化

```
| for  $u \in V$  do  
|   |  $color[u] \leftarrow WHITE$   
|   |  $pred[u] \leftarrow NULL$   
|   |  $dist[u] \leftarrow \infty$ 
```

| end

$color[s] \leftarrow GRAY$

$dist[s] \leftarrow 0$

$Q.Enqueue(s)$

初始化各记录信息数组



伪代码

- **BFS(G, s)**

输入: 图 G , 源点 s

输出: 前驱数组 $pred[]$, 距离数组 $dist[]$

新建一维数组 $color[1..|V|]$, $pred[1..|V|]$, $dist[1..|V|]$

新建空队列 Q

//初始化

for $u \in V$ do

$color[u] \leftarrow WHITE$

$pred[u] \leftarrow NULL$

$dist[u] \leftarrow \infty$

end

$color[s] \leftarrow GRAY$

$dist[s] \leftarrow 0$

$Q.Enqueue(s)$

初始化源点状态



伪代码

- **BFS(G, s)**

输入: 图 G , 源点 s

输出: 前驱数组 $pred[]$, 距离数组 $dist[]$

新建一维数组 $color[1..|V|]$, $pred[1..|V|]$, $dist[1..|V|]$

新建空队列 Q

//初始化

for $u \in V$ do

$color[u] \leftarrow WHITE$

$pred[u] \leftarrow NULL$

$dist[u] \leftarrow \infty$

end

$color[s] \leftarrow GRAY$

$dist[s] \leftarrow 0$

$[Q.Enqueue(s)]$

把源点加入队列 Q



伪代码

- **BFS(G, s)**

//广度优先搜索

```
while 等待队列Q非空 do
    u ← Q.Dequeue()
    for v ∈ G.Adj[u] do
        if color[v] = WHITE then
            color[v] ← GRAY
            dist[v] ← dist[u] + 1
            pred[v] ← u
            Q.Enqueue(v)
        end
    end
    color[u] ← BLACK
end
```

依次处理所有顶点



伪代码

- **BFS(G, s)**

//广度优先搜索

```
while 等待队列Q非空 do
    u ← Q.Dequeue()
    for v ∈ G.Adj[u] do
        if color[v] = WHITE then
            color[v] ← GRAY
            dist[v] ← dist[u] + 1
            pred[v] ← u
            Q.Enqueue(v)
        end
    end
    color[u] ← BLACK
end
```

从等待队列取出当前处理顶点 u



伪代码

- **BFS(G, s)**

```
//广度优先搜索
while 等待队列Q非空 do
     $u \leftarrow Q.Dequeue()$ 
    for  $v \in G.Adj[u]$  do
        if  $color[v] = WHITE$  then
             $color[v] \leftarrow GRAY$ 
             $dist[v] \leftarrow dist[u] + 1$ 
             $pred[v] \leftarrow u$ 
             $Q.Enqueue(v)$ 
        end
    end
     $color[u] \leftarrow BLACK$ 
end
```

搜索顶点 u 的所有相邻顶点



伪代码

- **BFS(G, s)**

```
//广度优先搜索
```

```
while 等待队列Q非空 do
```

```
     $u \leftarrow Q.Dequeue()$ 
```

```
    for  $v \in G.Adj[u]$  do
```

```
        if  $color[v] = WHITE$  then
```

第一次发现顶点 v

```
             $color[v] \leftarrow GRAY$ 
```

```
             $dist[v] \leftarrow dist[u] + 1$ 
```

```
             $pred[v] \leftarrow u$ 
```

```
             $Q.Enqueue(v)$ 
```

```
        end
```

```
    end
```

```
     $color[u] \leftarrow BLACK$ 
```

```
end
```



伪代码

- **BFS(G, s)**

```
//广度优先搜索
while 等待队列Q非空 do
     $u \leftarrow Q.Dequeue()$ 
    for  $v \in G.Adj[u]$  do
        if  $color[v] = WHITE$  then
             $color[v] \leftarrow GRAY$ 
             $dist[v] \leftarrow dist[u] + 1$ 
             $pred[v] \leftarrow u$ 
             $Q.Enqueue(v)$ 
        end
    end
     $color[u] \leftarrow BLACK$ 
end
```

标记已发现顶点 v 为待处理状态



伪代码

- **BFS(G, s)**

```
//广度优先搜索
```

```
while 等待队列Q非空 do
```

```
     $u \leftarrow Q.Dequeue()$ 
```

```
    for  $v \in G.Adj[u]$  do
```

```
        if  $color[v] = WHITE$  then
```

```
             $color[v] \leftarrow GRAY$ 
```

```
             $dist[v] \leftarrow dist[u] + 1$ 
```

```
             $pred[v] \leftarrow u$ 
```

```
             $Q.Enqueue(v)$ 
```

```
        end
```

```
    end
```

```
     $color[u] \leftarrow BLACK$ 
```

```
end
```

记录到源点的距离



伪代码

- **BFS(G, s)**

```
//广度优先搜索
```

```
while 等待队列 $Q$ 非空 do
```

```
     $u \leftarrow Q.Dequeue()$ 
```

```
    for  $v \in G.Adj[u]$  do
```

```
        if  $color[v] = WHITE$  then
```

```
             $color[v] \leftarrow GRAY$ 
```

```
             $dist[v] \leftarrow dist[u] + 1$ 
```

```
             $pred[v] \leftarrow u$ 
```

```
             $Q.Enqueue(v)$ 
```

```
        end
```

```
    end
```

```
     $color[u] \leftarrow BLACK$ 
```

```
end
```

记录前驱顶点



伪代码

- **BFS(G, s)**

```
//广度优先搜索
```

```
while 等待队列 $Q$ 非空 do
```

```
     $u \leftarrow Q.Dequeue()$ 
```

```
    for  $v \in G.Adj[u]$  do
```

```
        if  $color[v] = WHITE$  then
```

```
             $color[v] \leftarrow GRAY$ 
```

```
             $dist[v] \leftarrow dist[u] + 1$ 
```

```
             $pred[v] \leftarrow u$ 
```

```
             $Q.Enqueue(v)$ 
```

```
        end
```

```
    end
```

```
     $color[u] \leftarrow BLACK$ 
```

```
end
```

把 v 加入待处理队列



伪代码

- **BFS(G, s)**

```
//广度优先搜索
```

```
while 等待队列Q非空 do
```

```
     $u \leftarrow Q.Dequeue()$ 
```

```
    for  $v \in G.Adj[u]$  do
```

```
        if  $color[v] = WHITE$  then
```

```
             $color[v] \leftarrow GRAY$ 
```

```
             $dist[v] \leftarrow dist[u] + 1$ 
```

```
             $pred[v] \leftarrow u$ 
```

```
             $Q.Enqueue(v)$ 
```

```
        end
```

```
    end
```

```
     $color[u] \leftarrow BLACK$ 
```

```
end
```

标记当前顶点 u 处理完成

提纲



图的搜索

算法思想

算法伪代码

算法实例

算法分析

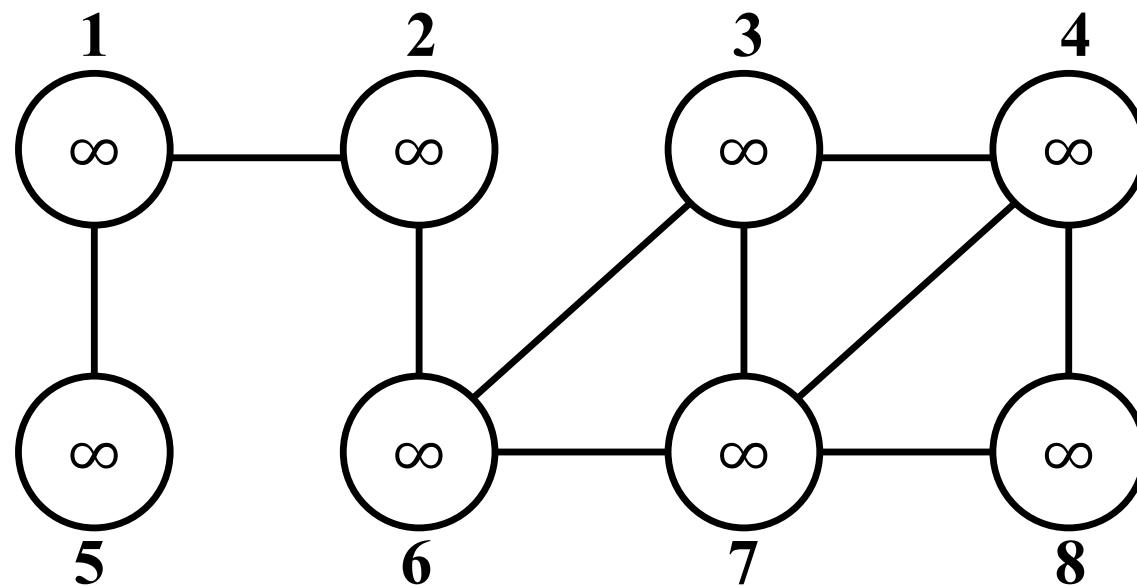
算法应用

算法实例



V	1	2	3	4	5	6	7	8
$color$	W	W	W	W	W	W	W	W
$pred$	N	N	N	N	N	N	N	N
$dist$	∞							

待处理队列



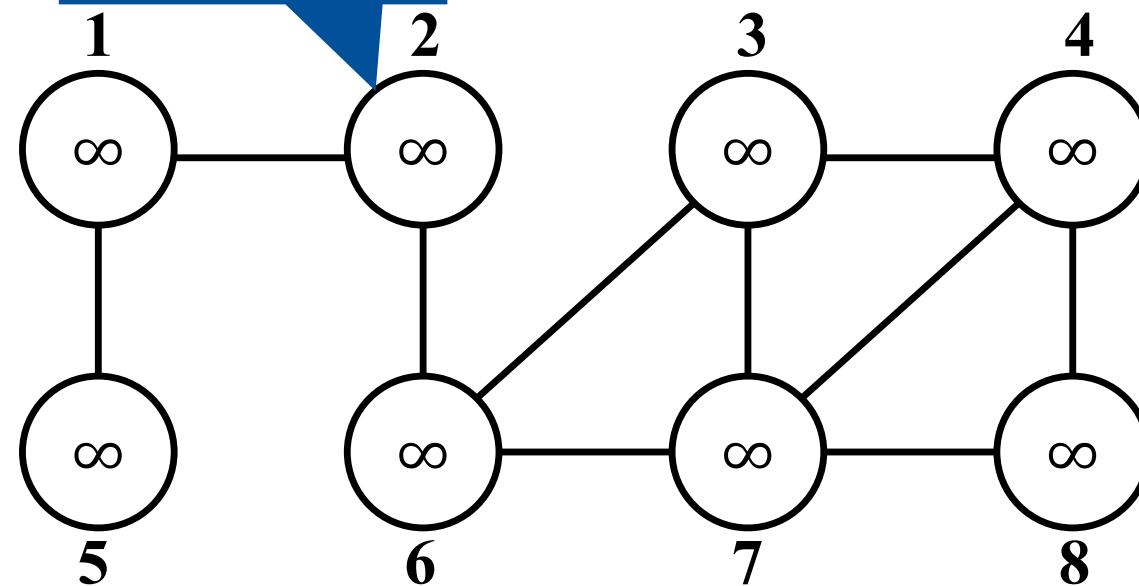
算法实例



V	1	2	3	4	5	6	7	8
$color$	W	W	W	W	W	W	W	W
$pred$	N	N	N	N	N	N	N	N
$dist$	∞							

待处理队列

搜索源点



算法实例

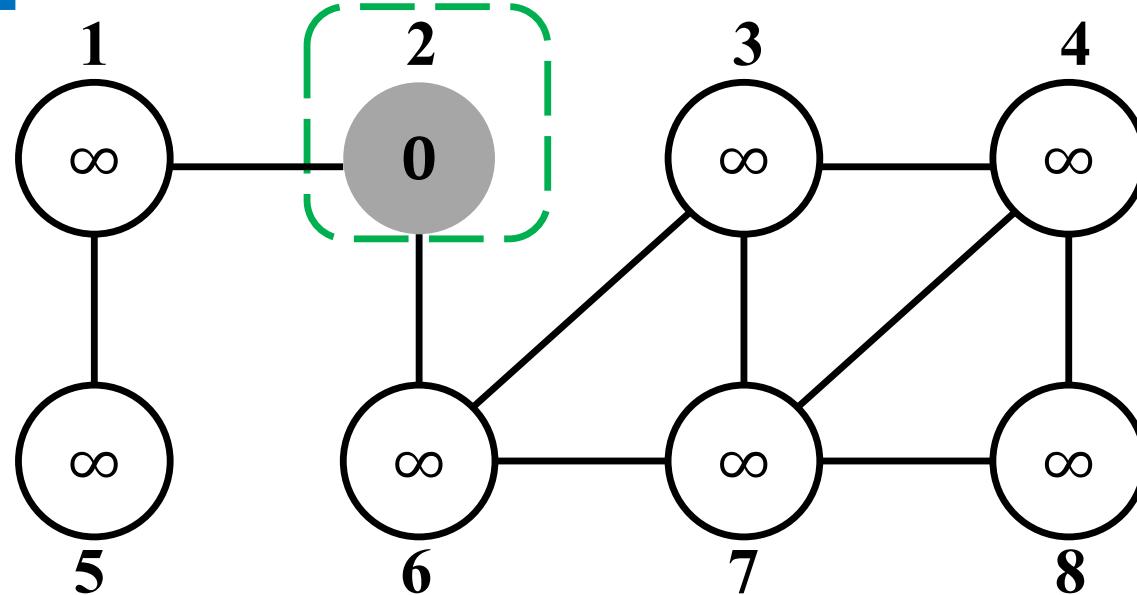


V	1	2	3	4	5	6	7	8
$color$	W	G	W	W	W	W	W	W
$pred$	N	N	N	N	N	N	N	N
$dist$	∞	0	∞	∞	∞	∞	∞	∞

待处理队列

2

源点2



算法实例

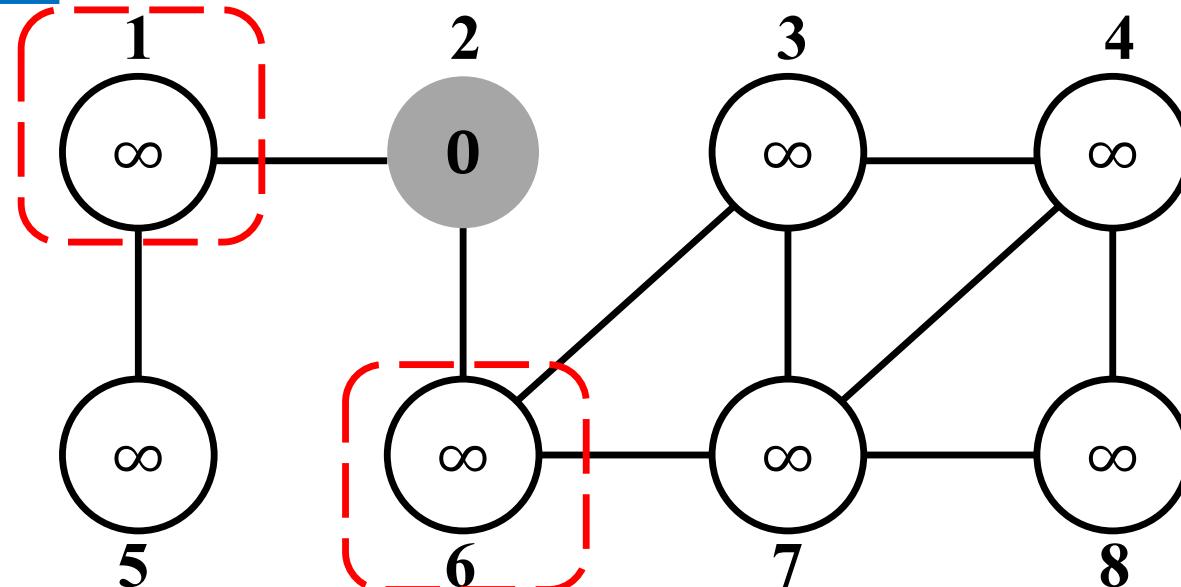


V	1	2	3	4	5	6	7	8
$color$	W	G	W	W	W	W	W	W
$pred$	N	N	N	N	N	N	N	N
$dist$	∞	0	∞	∞	∞	∞	∞	∞

待处理队列

2

源点2



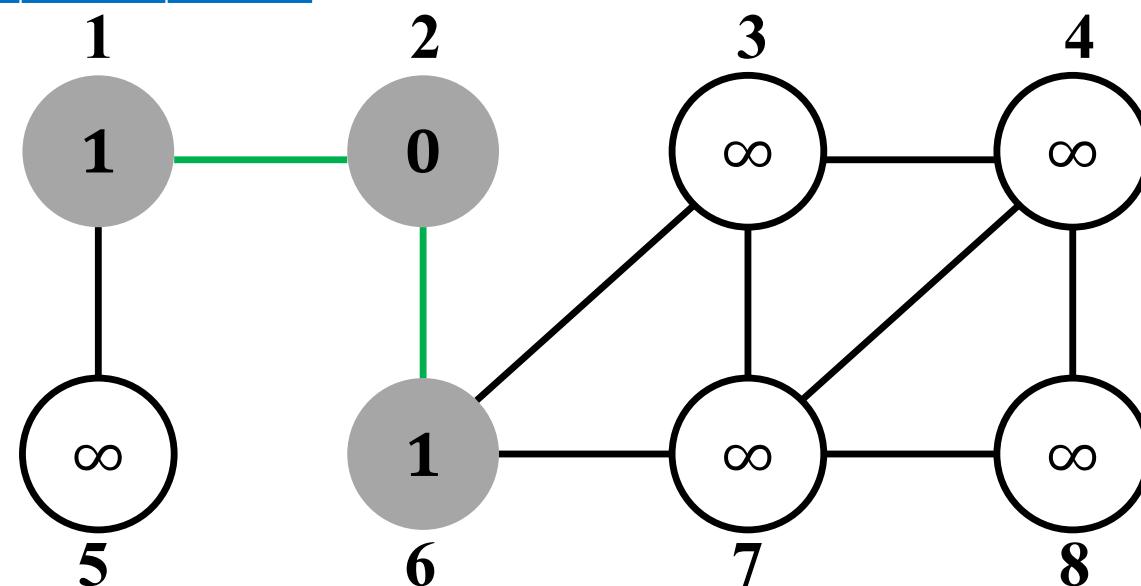
算法实例



V	1	2	3	4	5	6	7	8
$color$	G	G	W	W	W	G	W	W
$pred$	2	N	N	N	N	2	N	N
$dist$	1	0	∞	∞	∞	1	∞	∞

待处理队列	2	1	6
-------	---	---	---

源点2



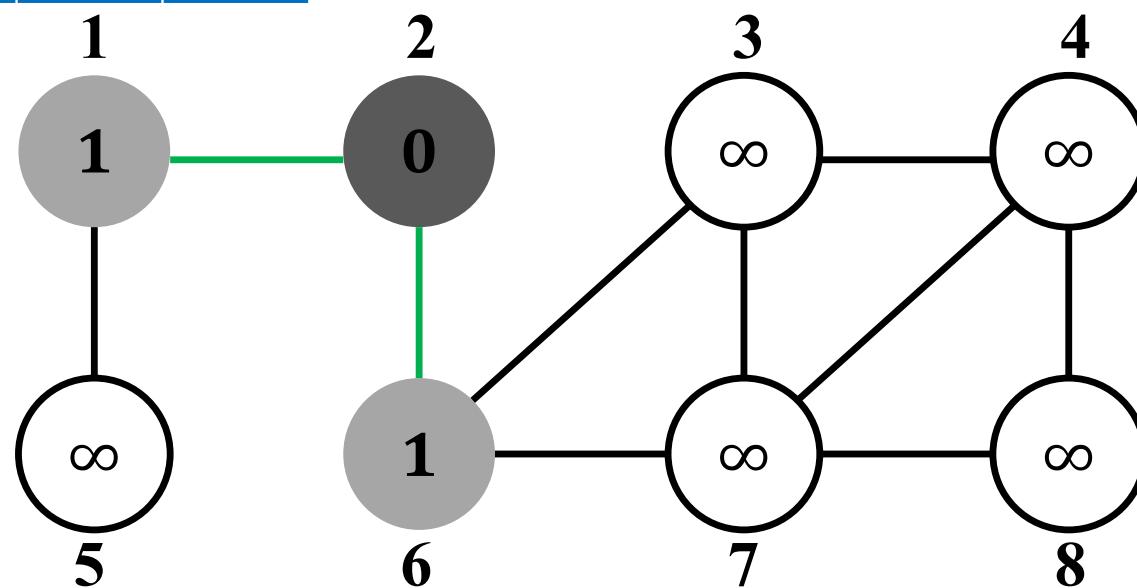
算法实例



V	1	2	3	4	5	6	7	8
$color$	G	B	W	W	W	G	W	W
$pred$	2	N	N	N	N	2	N	N
$dist$	1	0	∞	∞	∞	1	∞	∞

待处理队列	2	1	6
-------	---	---	---

源点2



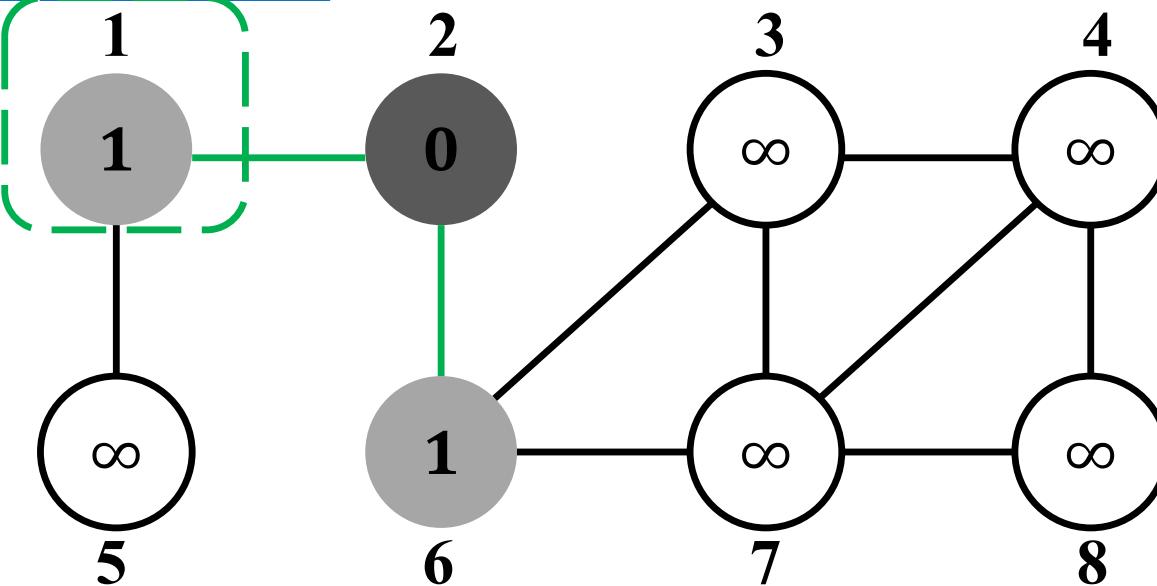
算法实例



V	1	2	3	4	5	6	7	8
$color$	G	B	W	W	W	G	W	W
$pred$	2	N	N	N	N	2	N	N
$dist$	1	0	∞	∞	∞	1	∞	∞

待处理队列	2	1	6
-------	---	---	---

源点2



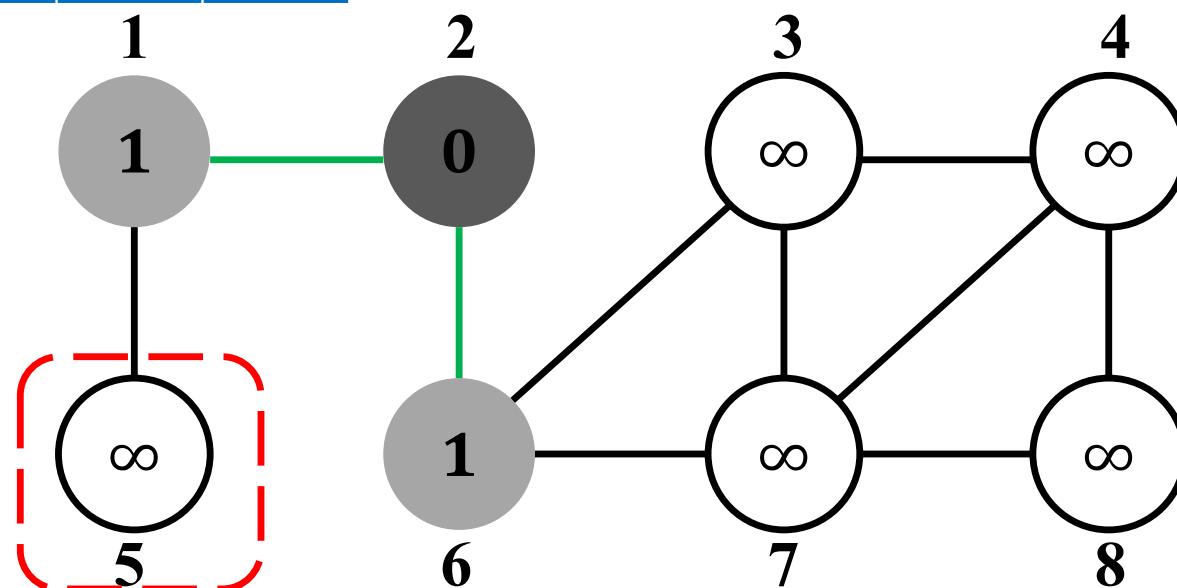
算法实例



V	1	2	3	4	5	6	7	8
$color$	G	B	W	W	W	G	W	W
$pred$	2	N	N	N	N	2	N	N
$dist$	1	0	∞	∞	∞	1	∞	∞

待处理队列	2	1	6
-------	---	---	---

源点2



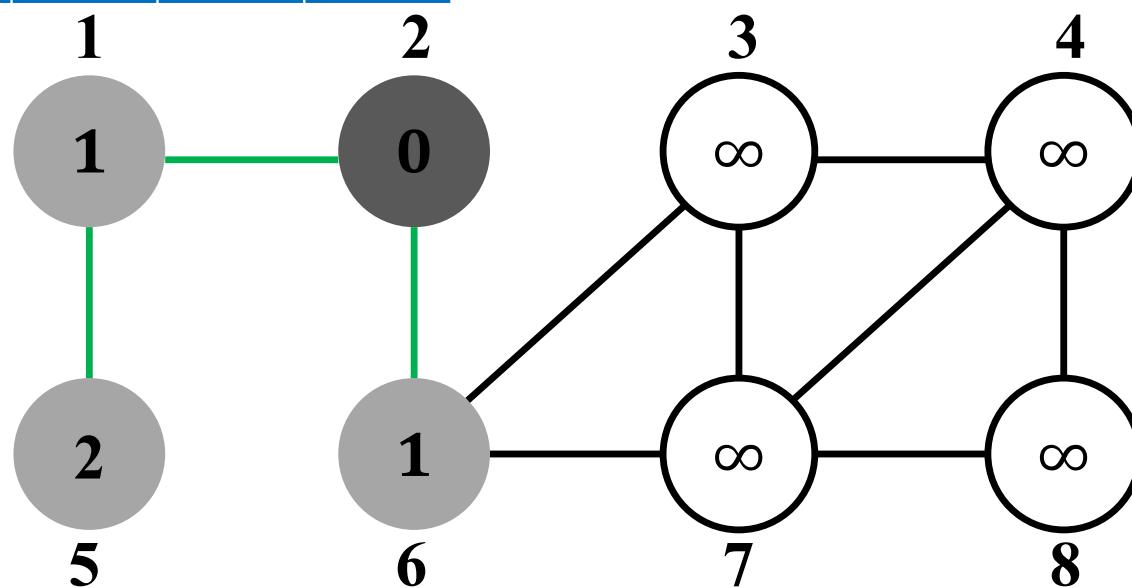
算法实例



V	1	2	3	4	5	6	7	8
$color$	G	B	W	W	G	G	W	W
$pred$	2	N	N	N	1	2	N	N
$dist$	1	0	∞	∞	2	1	∞	∞

待处理队列	2	1	6	5
-------	---	---	---	---

源点2



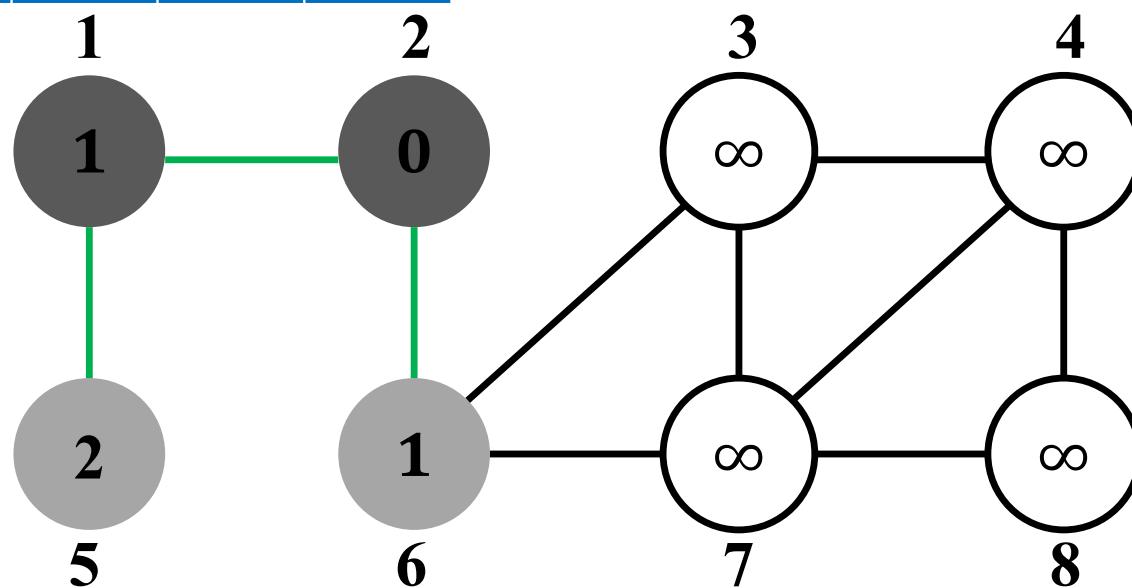
算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	W	W	G	G	W	W
$pred$	2	N	N	N	1	2	N	N
$dist$	1	0	∞	∞	2	1	∞	∞

待处理队列	2	1	6	5
-------	---	---	---	---

源点2



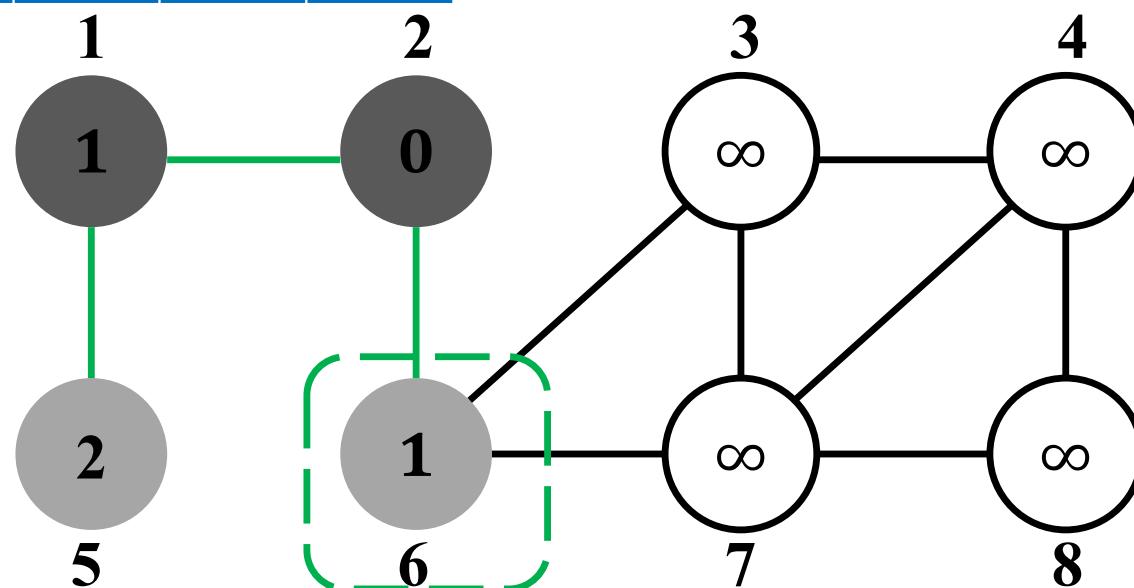
算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	W	W	G	G	W	W
$pred$	2	N	N	N	1	2	N	N
$dist$	1	0	∞	∞	2	1	∞	∞

待处理队列	2	1	6	5
-------	---	---	---	---

源点2



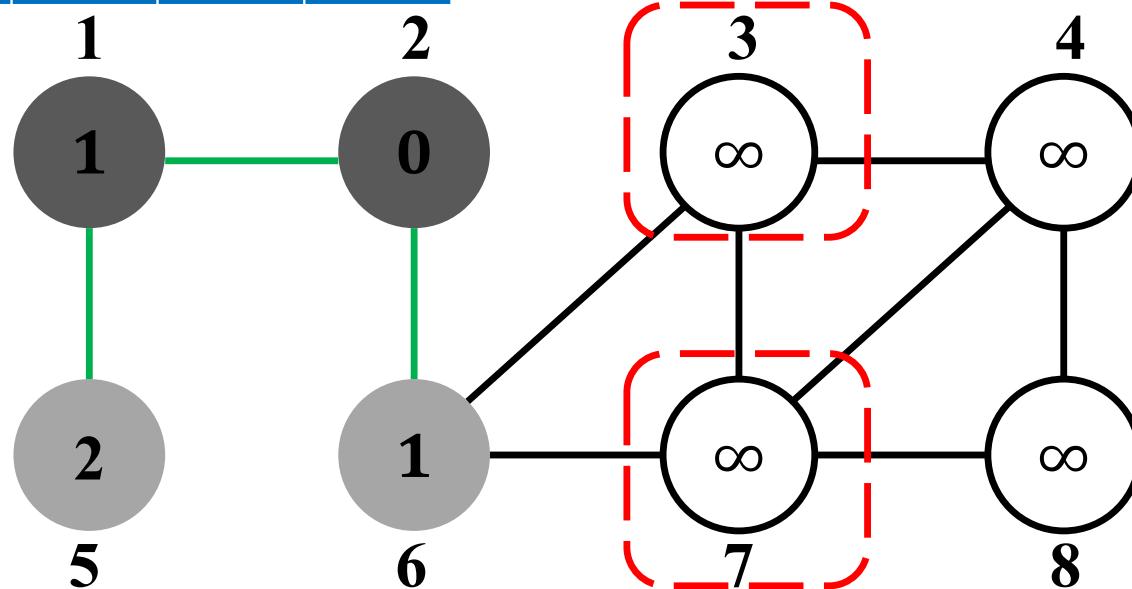
算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	W	W	G	G	W	W
$pred$	2	N	N	N	1	2	N	N
$dist$	1	0	∞	∞	2	1	∞	∞

待处理队列	2	1	6	5
-------	---	---	---	---

源点2



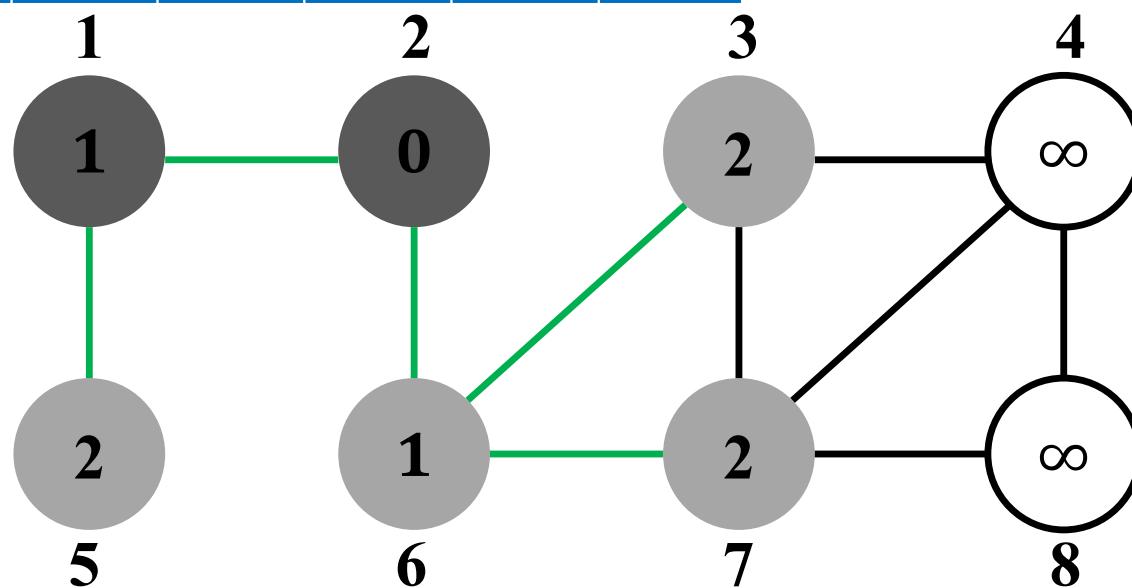
算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	G	W	G	G	G	W
$pred$	2	N	6	N	1	2	6	N
$dist$	1	0	2	∞	2	1	2	∞

待处理队列	2	1	6	5	3	7
-------	---	---	---	---	---	---

源点2

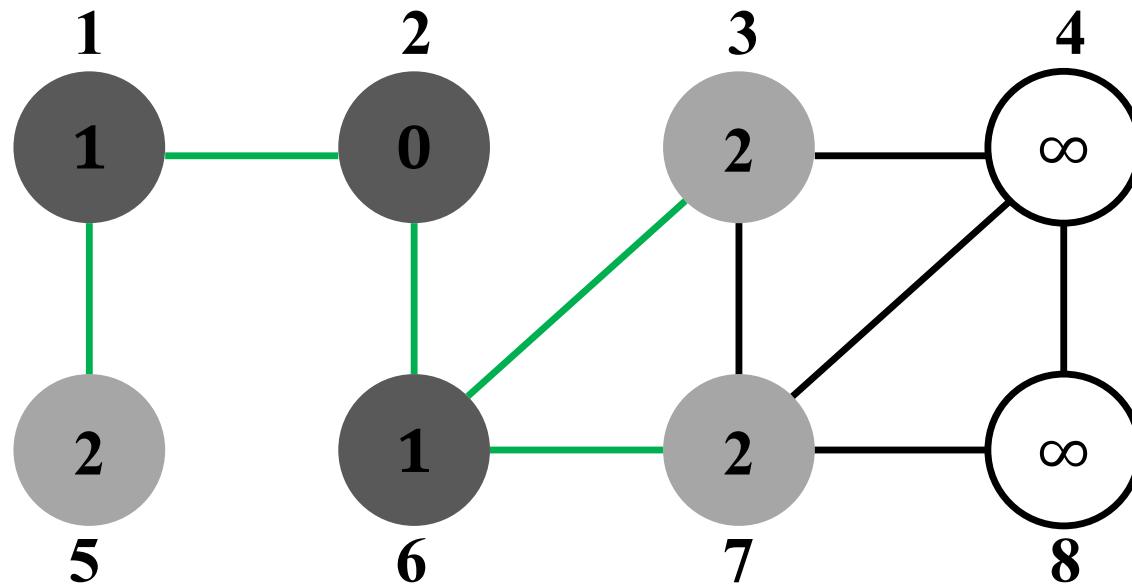


算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	G	W	G	B	G	W
$pred$	2	N	6	N	1	2	6	N
$dist$	1	0	2	∞	2	1	2	∞
待处理队列	2 1 6 5 3 7							

源点2



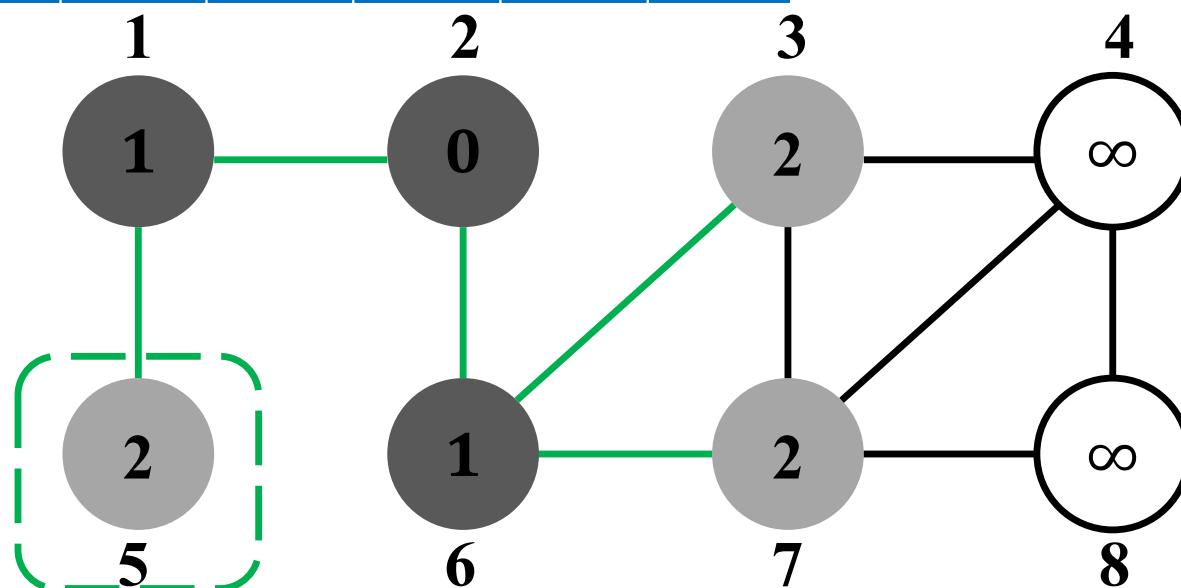
算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	G	W	G	B	G	W
$pred$	2	N	6	N	1	2	6	N
$dist$	1	0	2	∞	2	1	2	∞

待处理队列	2	1	6	5	3	7
-------	---	---	---	---	---	---

源点2

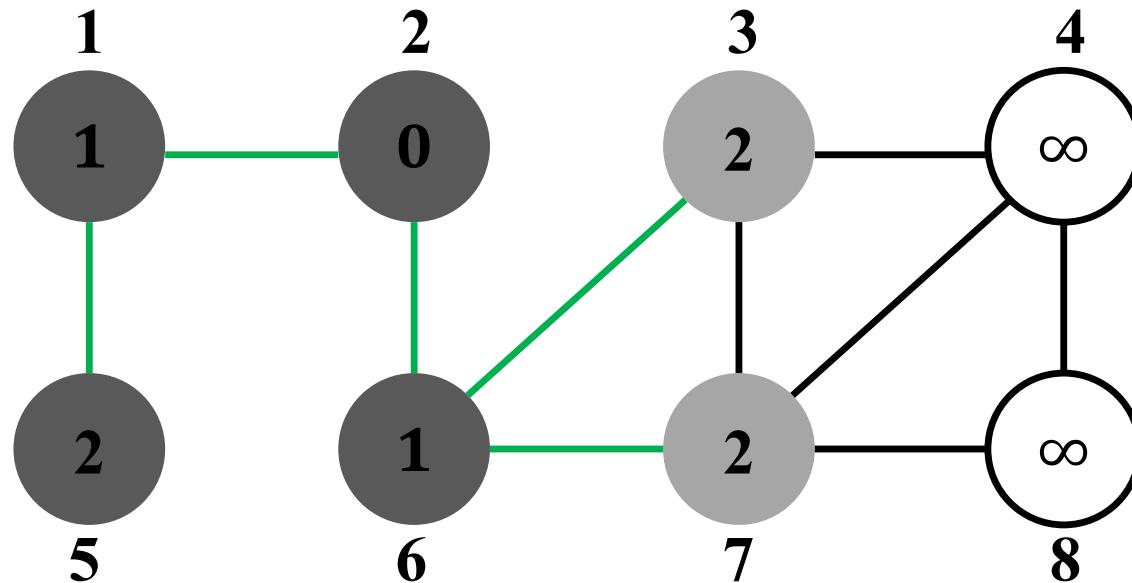


算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	G	W	B	B	G	W
$pred$	2	N	6	N	1	2	6	N
$dist$	1	0	2	∞	2	1	2	∞
待处理队列	2 1 6 5 3 7							

源点2



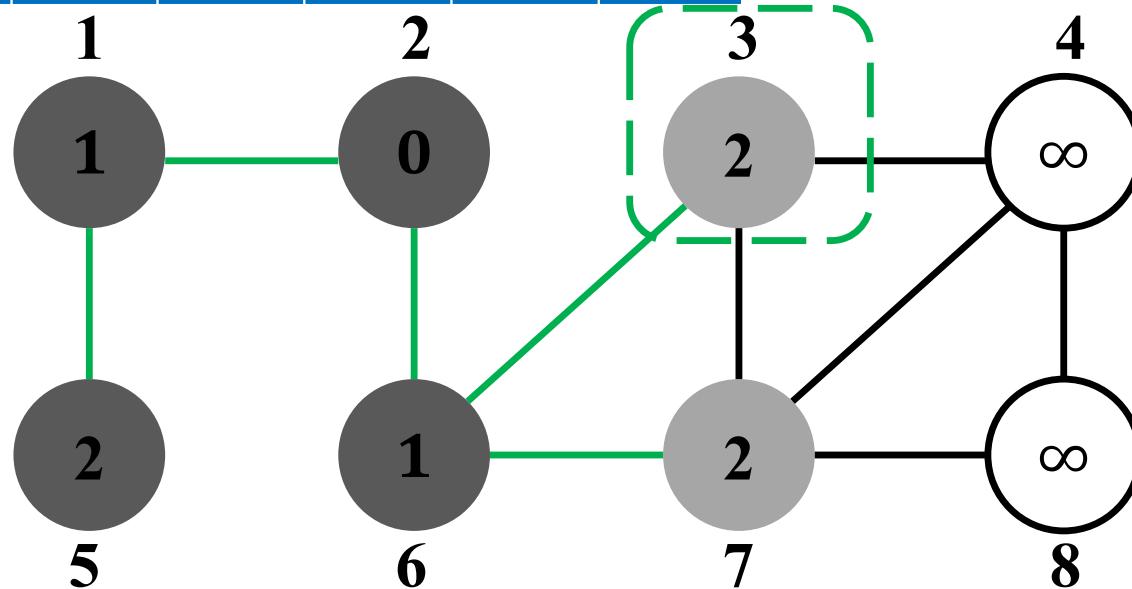
算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	G	W	B	B	G	W
$pred$	2	N	6	N	1	2	6	N
$dist$	1	0	2	∞	2	1	2	∞

待处理队列	2	1	6	5	3	7
-------	---	---	---	---	---	---

源点2



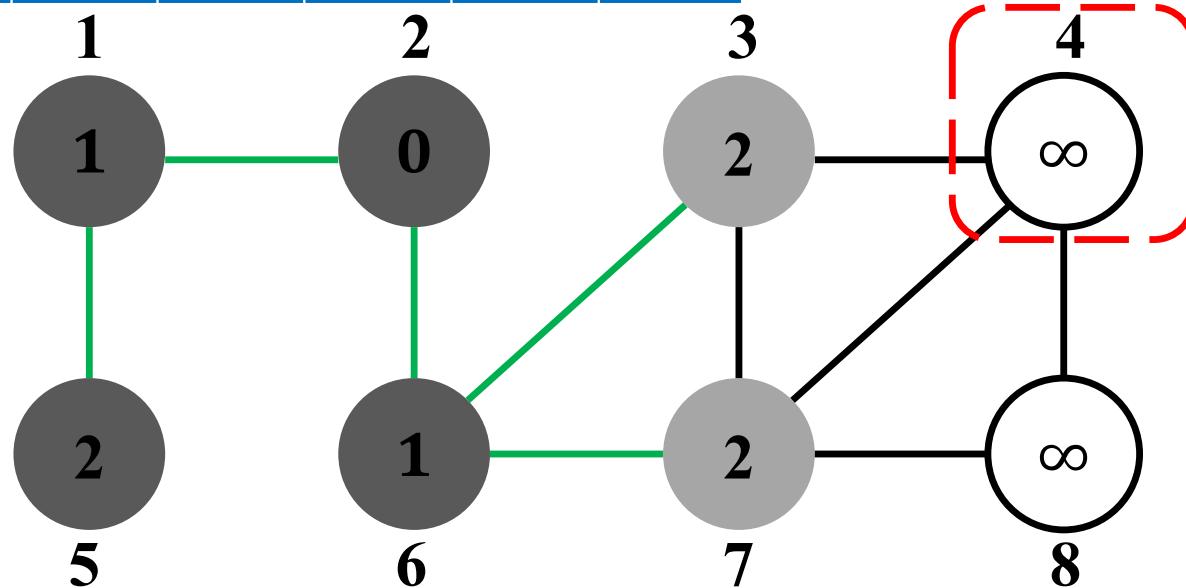
算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	G	W	B	B	G	W
$pred$	2	N	6	N	1	2	6	N
$dist$	1	0	2	∞	2	1	2	∞

待处理队列	2	1	6	5	3	7
-------	---	---	---	---	---	---

源点2

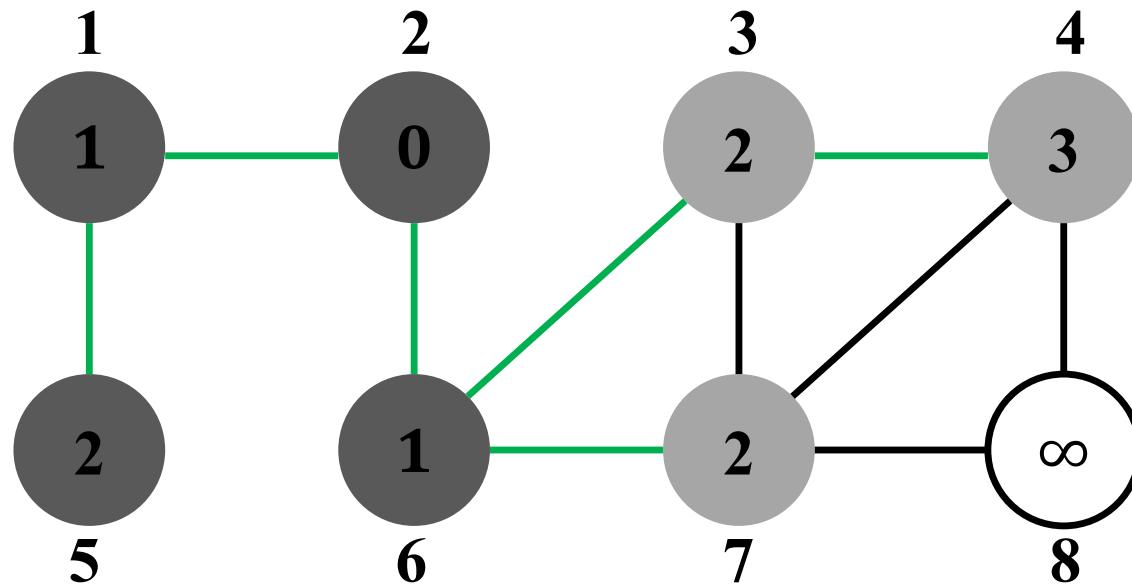


算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	G	G	B	B	G	W
$pred$	2	N	6	3	1	2	6	N
$dist$	1	0	2	3	2	1	2	∞
待处理队列	2 1 6 5 3 7 4							

源点2

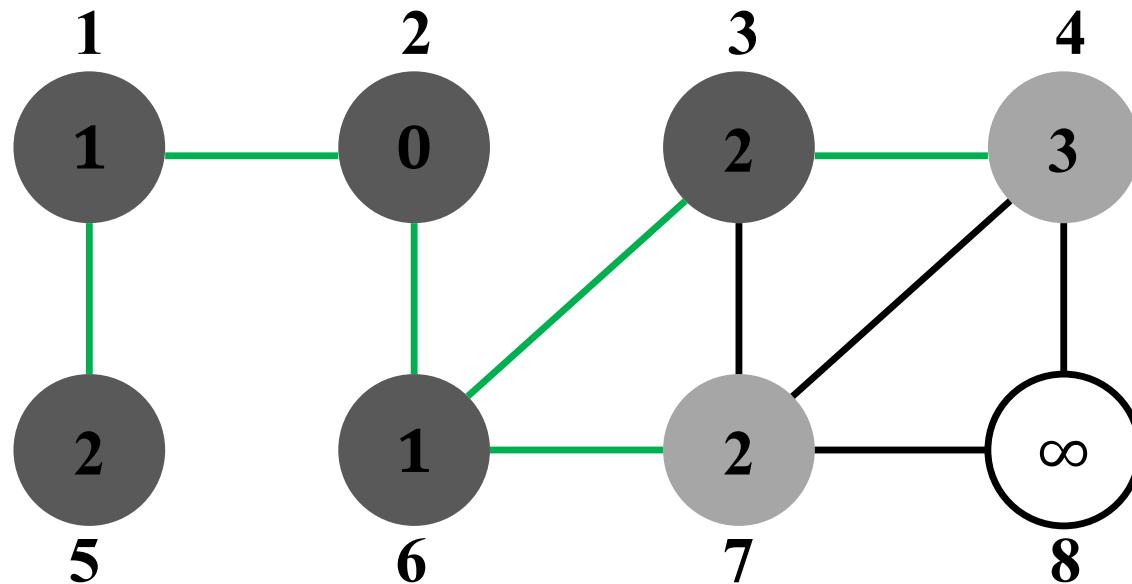


算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	B	G	B	B	G	W
$pred$	2	N	6	3	1	2	6	N
$dist$	1	0	2	3	2	1	2	∞
待处理队列	2 1 6 5 3 7 4							

源点2

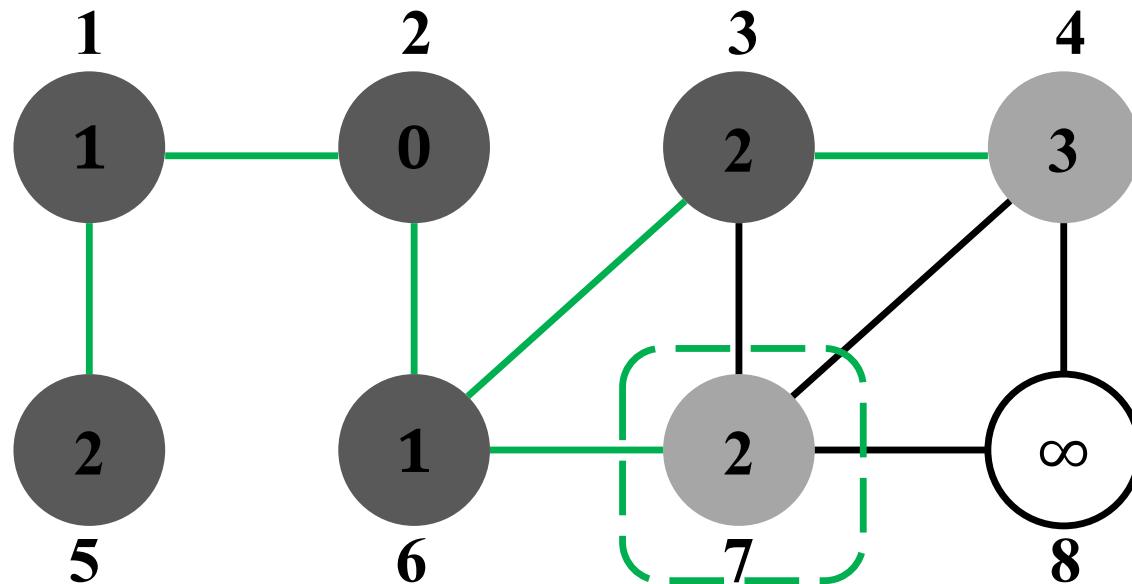


算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	B	G	B	B	G	W
$pred$	2	N	6	3	1	2	6	N
$dist$	1	0	2	3	2	1	2	∞
待处理队列	2 1 6 5 3 7 4							

源点2

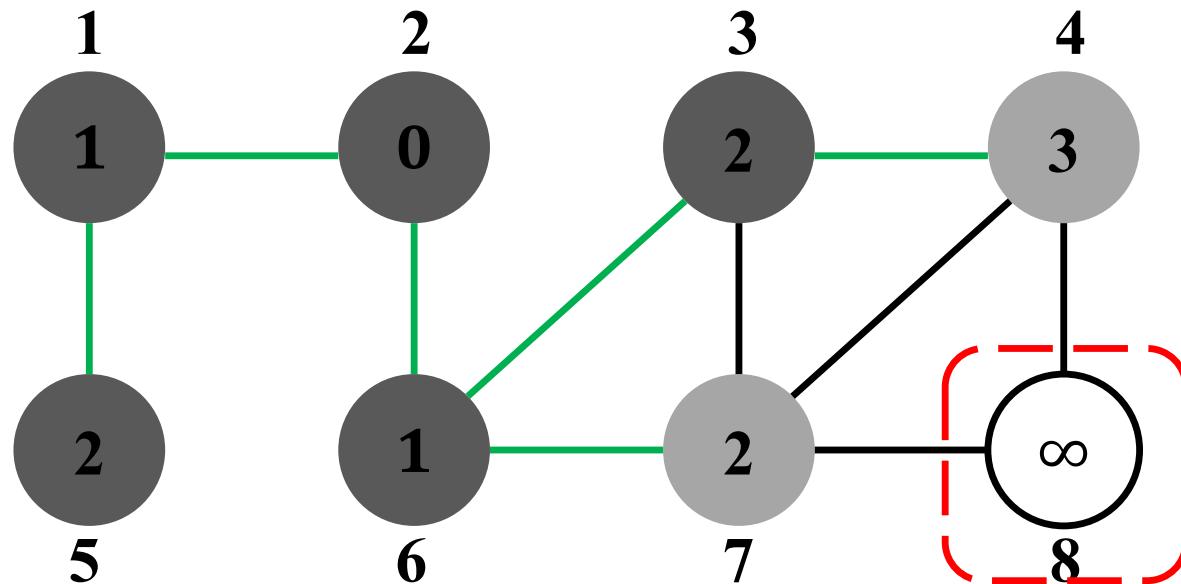


算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	B	G	B	B	G	W
$pred$	2	N	6	3	1	2	6	N
$dist$	1	0	2	3	2	1	2	∞
待处理队列	2 1 6 5 3 7 4							

源点2

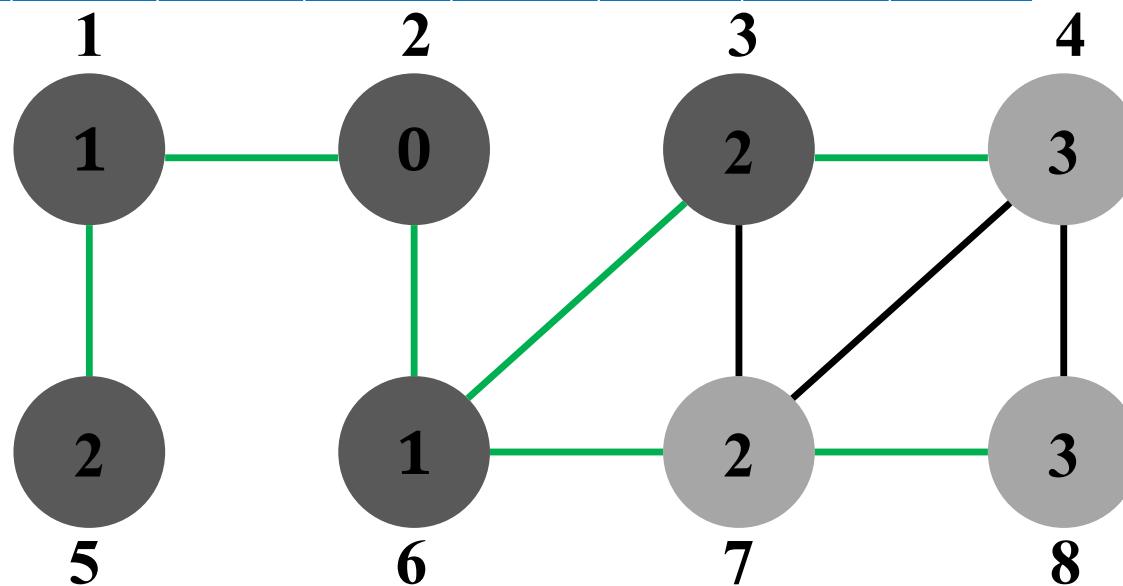


算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	B	G	B	B	G	G
$pred$	2	N	6	3	1	2	6	7
$dist$	1	0	2	3	2	1	2	3
待处理队列	2	1	6	5	3	7	4	8

源点2

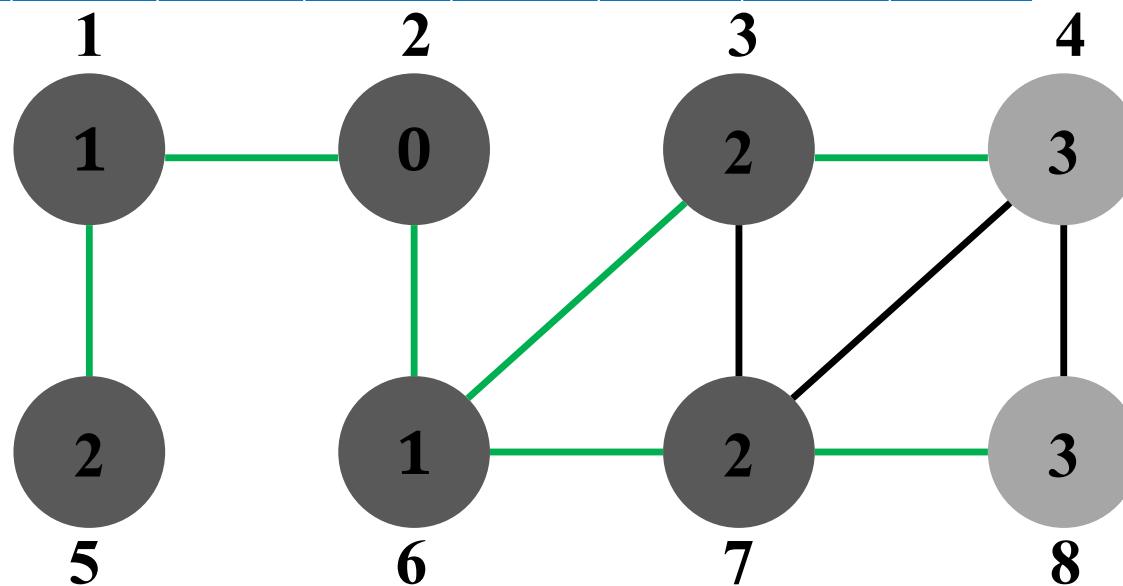


算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	B	G	B	B	B	G
$pred$	2	N	6	3	1	2	6	7
$dist$	1	0	2	3	2	1	2	3
待处理队列	2	1	6	5	3	7	4	8

源点2

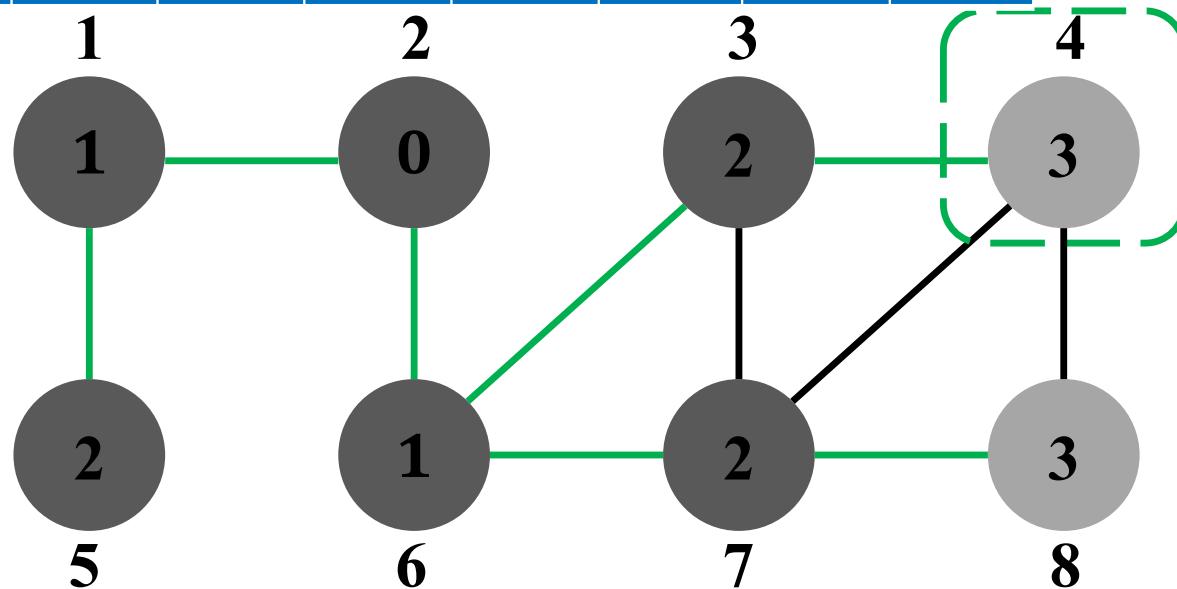


算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	B	G	B	B	B	G
$pred$	2	N	6	3	1	2	6	7
$dist$	1	0	2	3	2	1	2	3
待处理队列	2	1	6	5	3	7	4	8

源点2

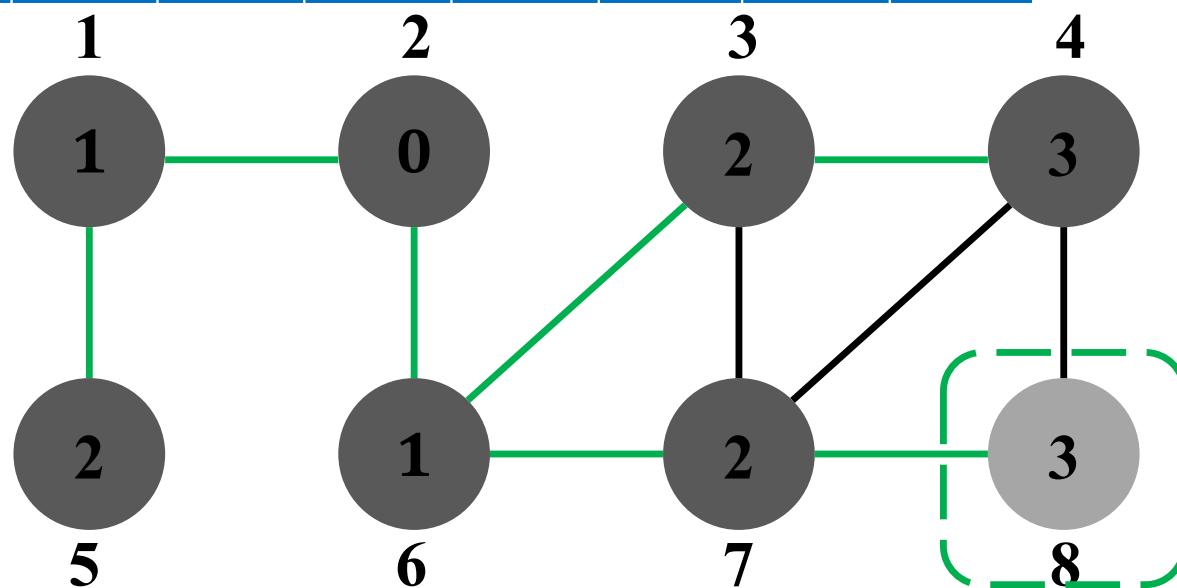


算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	B	B	B	B	B	G
$pred$	2	N	6	3	1	2	6	7
$dist$	1	0	2	3	2	1	2	3
待处理队列	2	1	6	5	3	7	4	8

源点2

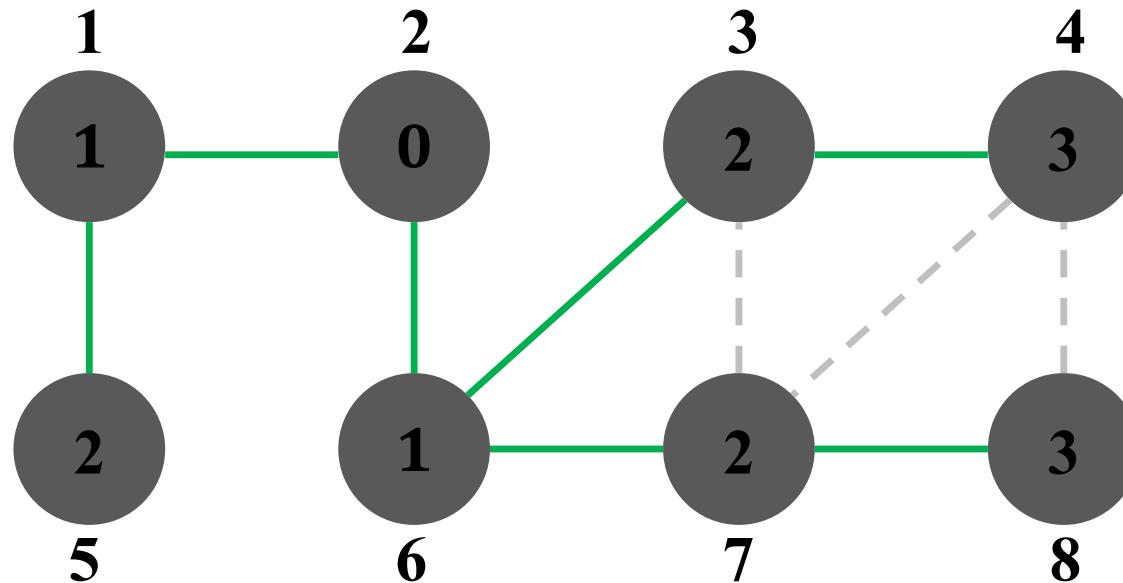


算法实例



V	1	2	3	4	5	6	7	8
$color$	B	B	B	B	B	B	B	B
$pred$	2	N	6	3	1	2	6	7
$dist$	1	0	2	3	2	1	2	3
待处理队列	2	1	6	5	3	7	4	8

源点2



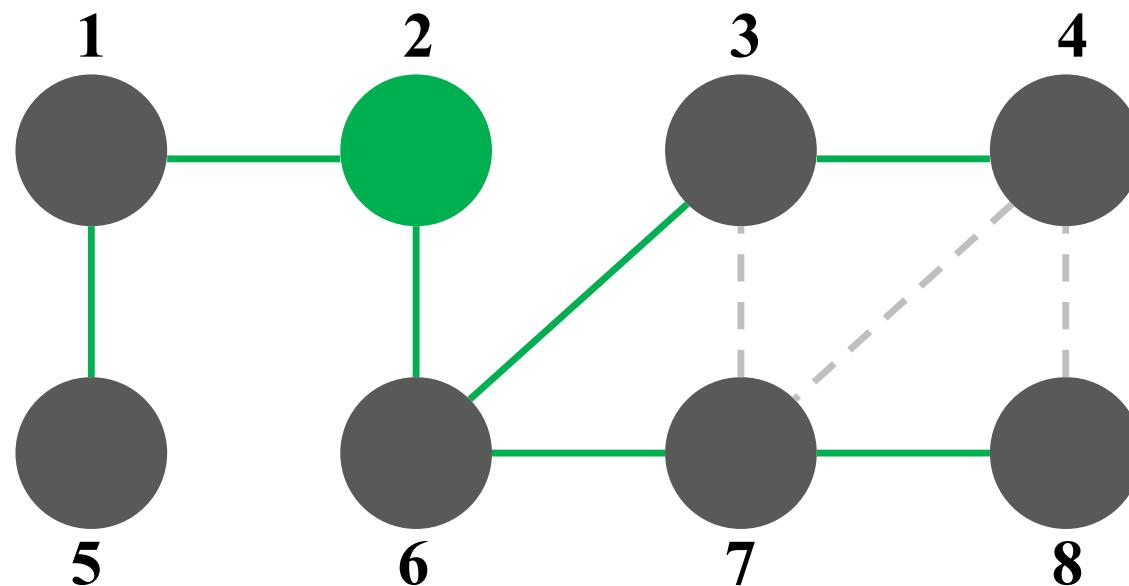
搜索结束

广度优先树



V	1	2	3	4	5	6	7	8
$pred$	2	N	6	3	1	2	6	7

- 辅助数组 $pred[]$ 储存了一棵树



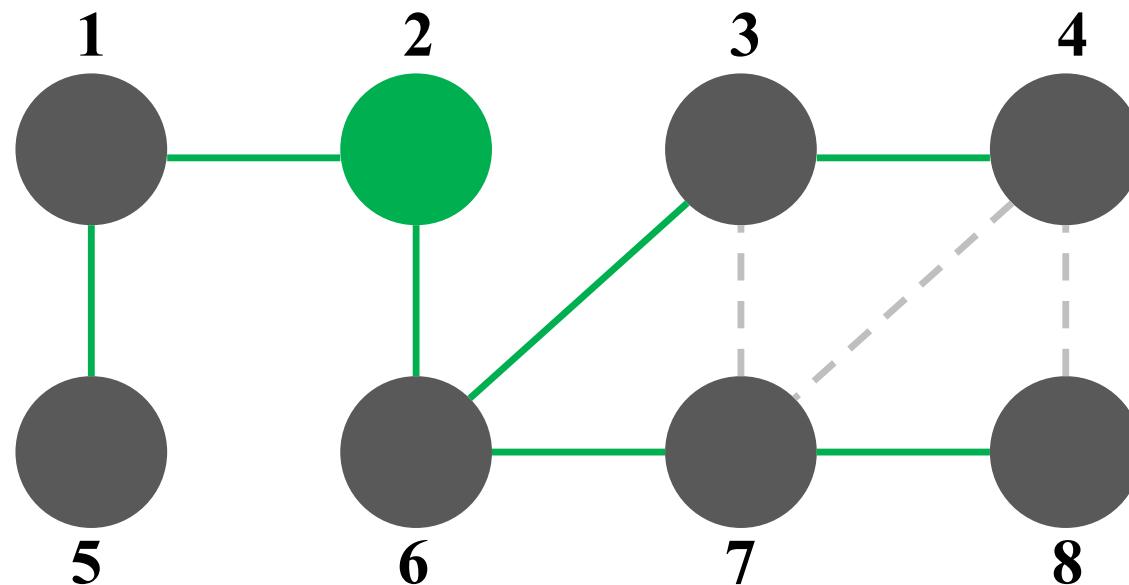


广度优先树

V	1	2	3	4	5	6	7	8
$pred$	2	N	6	3	1	2	6	7

- 辅助数组 $pred[]$ 储存了一棵树，广度优先树 $T = < V_T, E_T >$

连通、无环
 $|E_T| = |V_T| - 1$

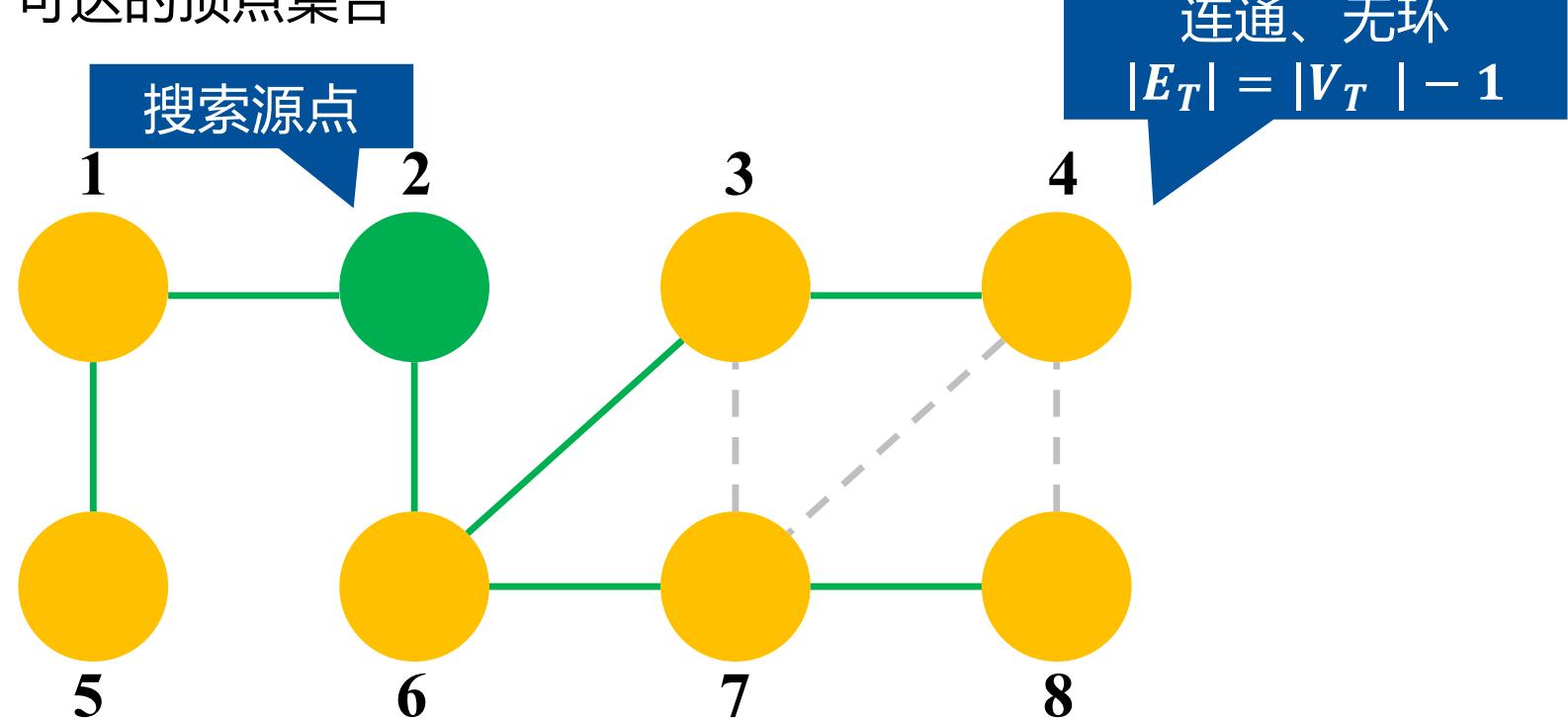




广度优先树

V	1	2	3	4	5	6	7	8
$pred$	2	N	6	3	1	2	6	7

- 辅助数组 $pred[]$ 储存了一棵树，广度优先树 $T = < V_T, E_T >$
- V_T 是源点 s 可达的顶点集合



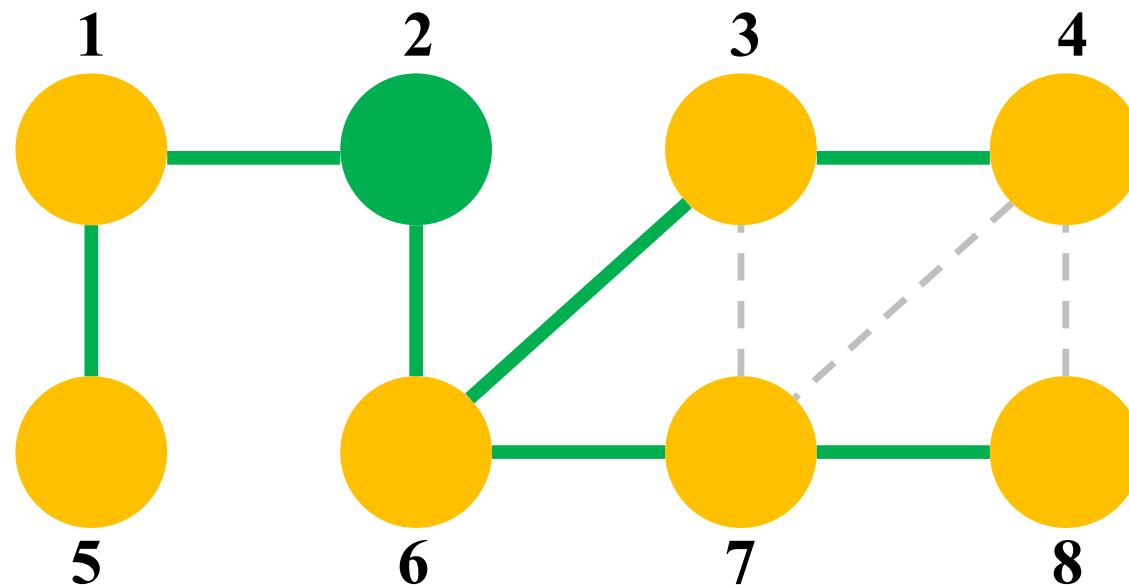


广度优先树

V	1	2	3	4	5	6	7	8
$pred$	2	N	6	3	1	2	6	7

- 辅助数组 $pred[]$ 储存了一棵树，广度优先树 $T = < V_T, E_T >$
- V_T 是源点 s 可达的顶点集合，
 $E_T = \{ (pred[u], u) : u \in V_T \}$

连通、无环
 $|E_T| = |V_T| - 1$



提纲



图的搜索

算法思想

算法伪代码

算法实例

算法分析

算法应用



输入: 图 G , 源点 s

输出: 前驱数组 $pred[]$, 距离数组 $dist[]$

新建一维数组 $color[1..|V|]$, $pred[1..|V|]$, $dist[1..|V|]$

新建空队列 Q

//初始化

for $u \in V$ do

$color[u] \leftarrow WHITE$

$pred[u] \leftarrow NULL$

$dist[u] \leftarrow \infty$

end

$color[s] \leftarrow GRAY$

$dist[s] \leftarrow 0$

$Q.Enqueue(s)$

//广度优先搜索

while 等待队列 Q 非空 do

$u \leftarrow Q.Dequeue()$

 for $v \in G.Adj[u]$ do

 if $color[v] = WHITE$ then

$color[v] \leftarrow GRAY$

$dist[v] \leftarrow dist[u] + 1$

$pred[v] \leftarrow u$

$Q.Enqueue(v)$

 end

 end

$color[u] \leftarrow BLACK$

end



复杂度分析

- 对于每个顶点 u ，搜索相邻顶点消耗时间 $T_u = O(1 + \deg(u))$
- 总运行时间：

$$T \leq \sum_{u \in V} T_u$$

依次搜索所有顶点



复杂度分析

- 对于每个顶点 u ，搜索相邻顶点消耗时间 $T_u = O(1 + \deg(u))$
- 总运行时间：

$$\begin{aligned} T &\leq \sum_{u \in V} T_u \\ &= \sum_{u \in V} O(1 + \deg(u)) \end{aligned}$$

搜索所有相邻顶点



复杂度分析

- 对于每个顶点 u ，搜索相邻顶点消耗时间 $T_u = O(1 + \deg(u))$
- 总运行时间：

$$\begin{aligned} T &\leq \sum_{u \in V} T_u \\ &= \sum_{u \in V} O(1 + \deg(u)) \\ &= \sum_{u \in V} O(1) + \sum_{u \in V} O(\deg(u)) \end{aligned}$$

公式化简



复杂度分析

- 对于每个顶点 u ，搜索相邻顶点消耗时间 $T_u = O(1 + \deg(u))$
- 总运行时间：

$$\begin{aligned} T &\leq \sum_{u \in V} T_u \\ &= \sum_{u \in V} O(1 + \deg(u)) \\ &= \sum_{u \in V} O(1) + \sum_{u \in V} O(\deg(u)) \end{aligned}$$

图的度



复杂度分析

- 对于每个顶点 u ，搜索相邻顶点消耗时间 $T_u = O(1 + \deg(u))$
- 总运行时间：

$$T \leq \sum_{u \in V} T_u$$

$$= \sum_{u \in V} O(1 + \deg(u))$$

$$= \sum_{u \in V} O(1) + \sum_{u \in V} O(\deg(u))$$

$$= O(|V| + |E|)$$

$$\deg(G) = 2 \cdot |E|$$



复杂度分析

- 对于每个顶点 u ，搜索相邻顶点消耗时间 $T_u = O(1 + \deg(u))$
- 总运行时间：

$$\begin{aligned} T &\leq \sum_{u \in V} T_u \\ &= \sum_{u \in V} O(1 + \deg(u)) \\ &= \sum_{u \in V} O(1) + \sum_{u \in V} O(\deg(u)) \\ &= O(|V| + |E|) \end{aligned}$$

广度优先搜索的时间复杂度为 $O(|V| + |E|)$



复杂度分析

- 对于每个顶点 u ，搜索相邻顶点消耗时间 $T_u = O(1 + \deg(u))$
- 总运行时间：

$$\begin{aligned} T &\leq \sum_{u \in V} T_u \\ &= \sum_{u \in V} O(1 + \deg(u)) \\ &= \sum_{u \in V} O(1) + \sum_{u \in V} O(\deg(u)) \\ &= O(|V| + |E|) \end{aligned}$$

在渐近记号中，约定符号 V 代表 $|V|$ ，符号 E 代表 $|E|$



复杂度分析

- 对于每个顶点 u ，搜索相邻顶点消耗时间 $T_u = O(1 + \deg(u))$
- 总运行时间：

$$\begin{aligned} T &\leq \sum_{u \in V} T_u \\ &= \sum_{u \in V} O(1 + \deg(u)) \\ &= \sum_{u \in V} O(1) + \sum_{u \in V} O(\deg(u)) \\ &= O(V + E) \end{aligned}$$

简化表示

在渐近记号中，约定符号 V 代表 $|V|$ ，符号 E 代表 $|E|$

提纲



图的搜索

算法思想

算法实例

算法分析

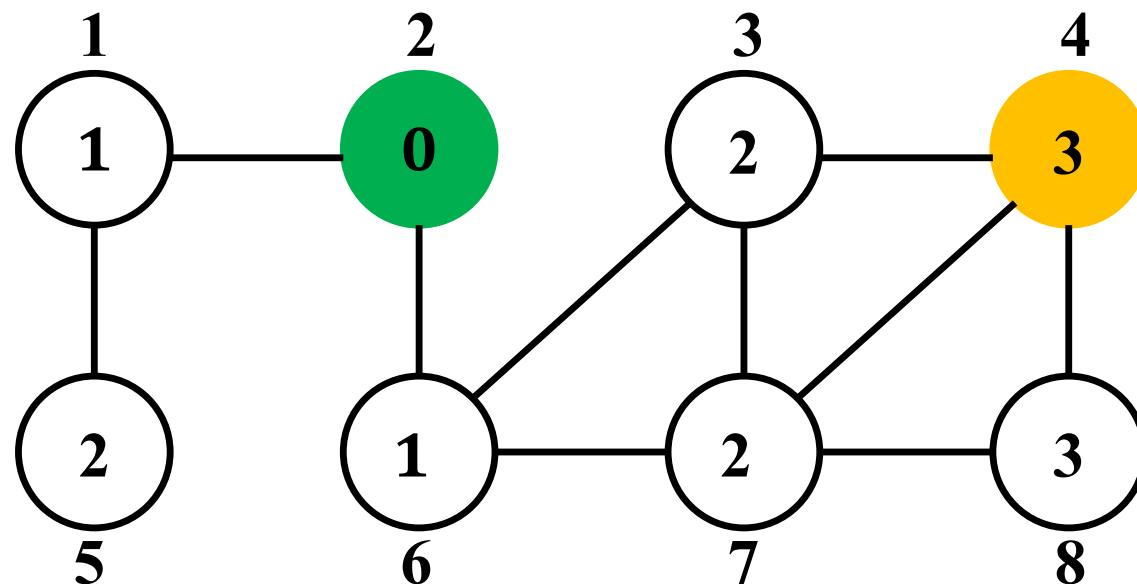
算法应用



相关应用：最短路径

V	1	2	3	4	5	6	7	8
$pred[]$	2	N	6	3	1	2	6	7
$dist[]$	1	0	2	3	2	1	2	3

无权图 G 中，源点 2 到顶点 4 的最短路径

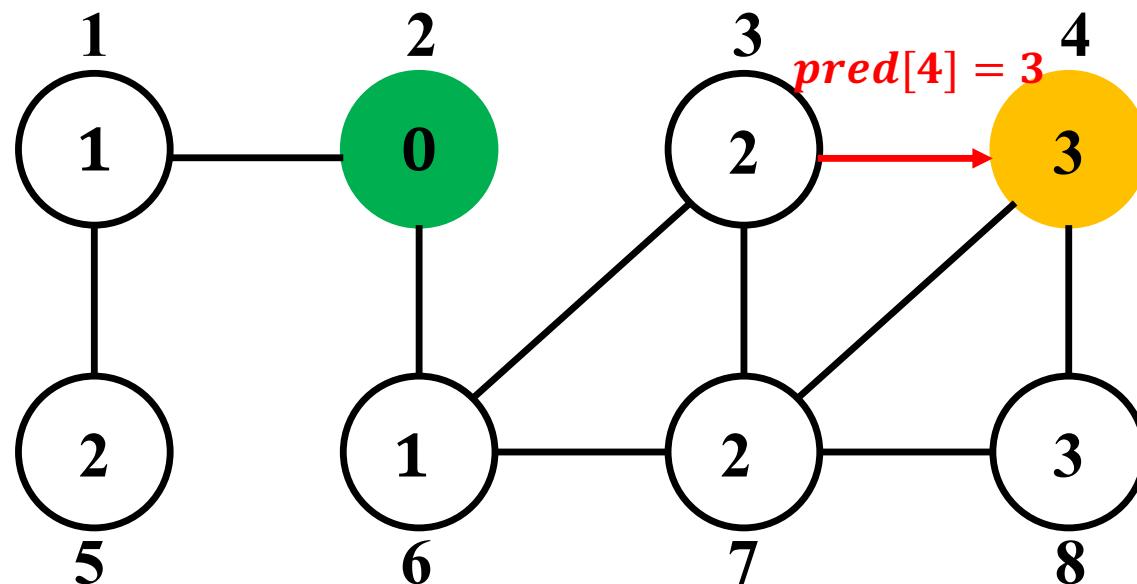




相关应用：最短路径

V	1	2	3	4	5	6	7	8
$pred[]$	2	N	6	3	1	2	6	7
$dist[]$	1	0	2	3	2	1	2	3

无权图 G 中，源点 2 到顶点 4 的最短路径

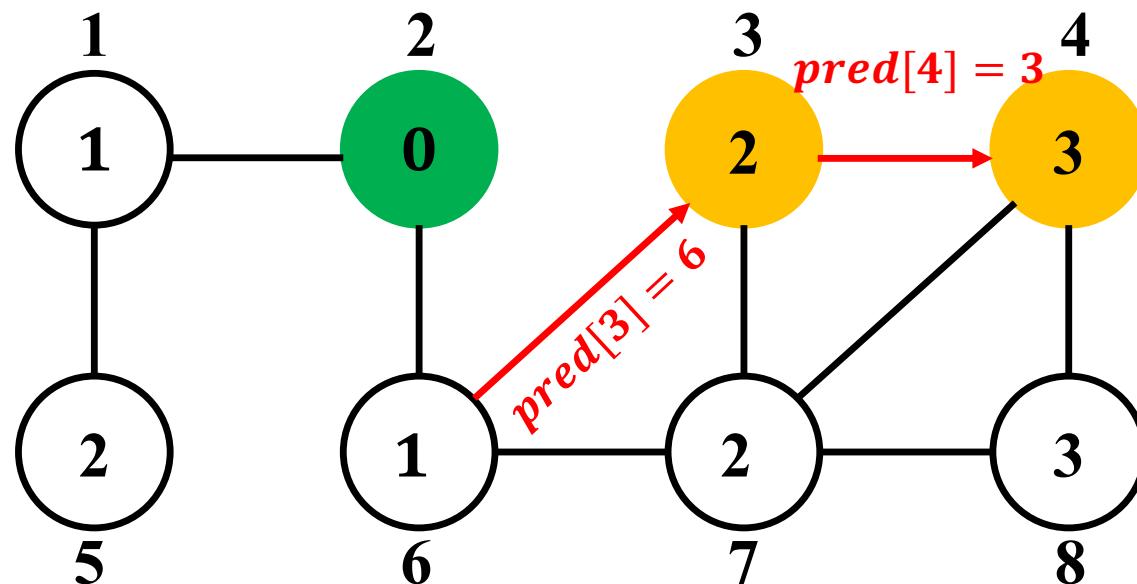




相关应用：最短路径

V	1	2	3	4	5	6	7	8
$pred[]$	2	N	6	3	1	2	6	7
$dist[]$	1	0	2	3	2	1	2	3

无权图 G 中，源点 2 到顶点 4 的最短路径

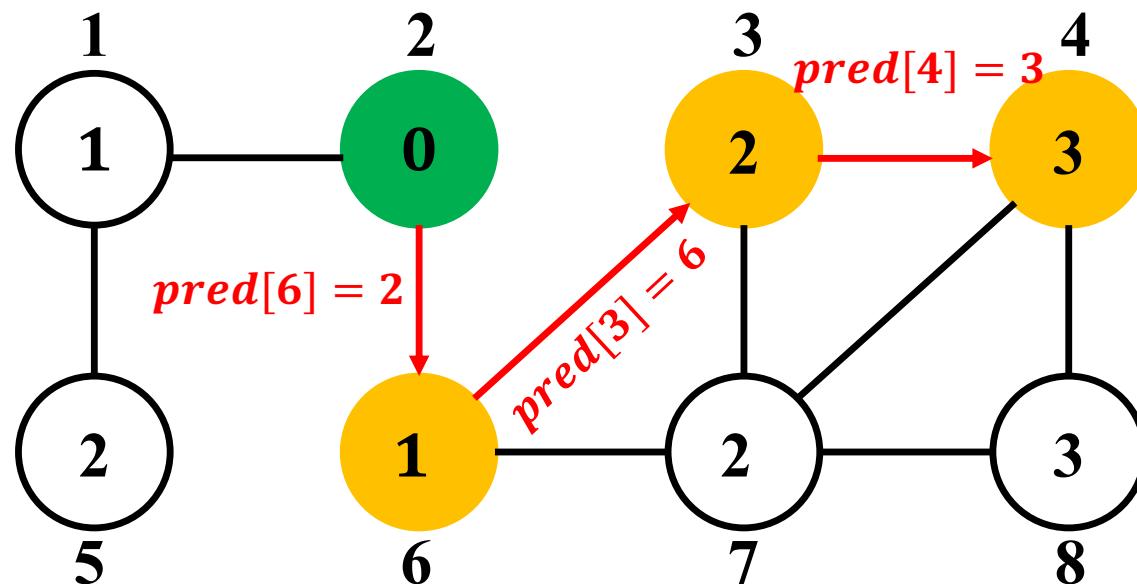




相关应用：最短路径

V	1	2	3	4	5	6	7	8
$pred[]$	2	N	6	3	1	2	6	7
$dist[]$	1	0	2	3	2	1	2	3

无权图 G 中，源点 2 到顶点 4 的最短路径

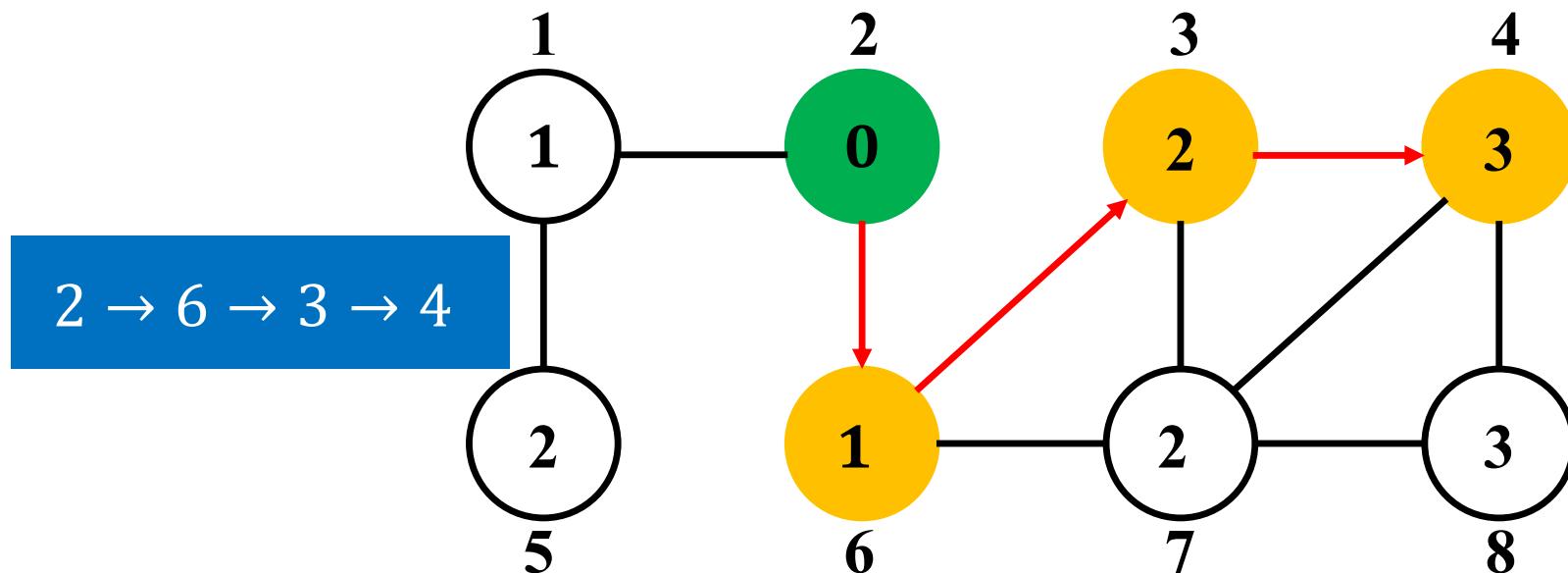




相关应用：最短路径

V	1	2	3	4	5	6	7	8
$pred[]$	2	N	6	3	1	2	6	7
$dist[]$	1	0	2	3	2	1	2	3

无权图 G 中，源点 2 到顶点 4 的最短路径

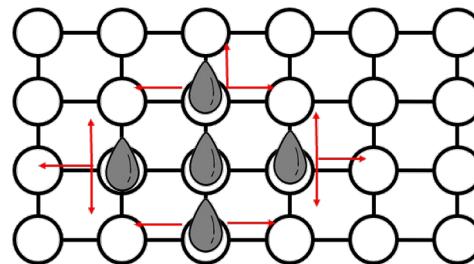
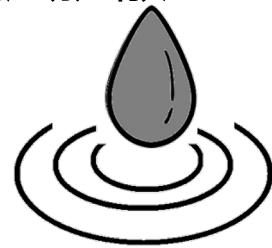


小结



- 广度优先搜索

- 算法核心思想：逐层扩散

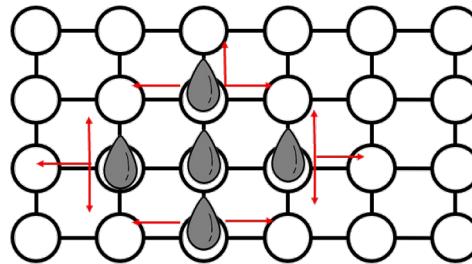
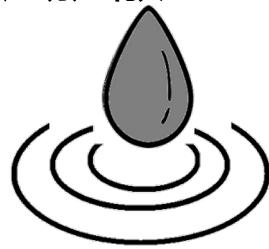


小结



- 广度优先搜索

- 算法核心思想：逐层扩散



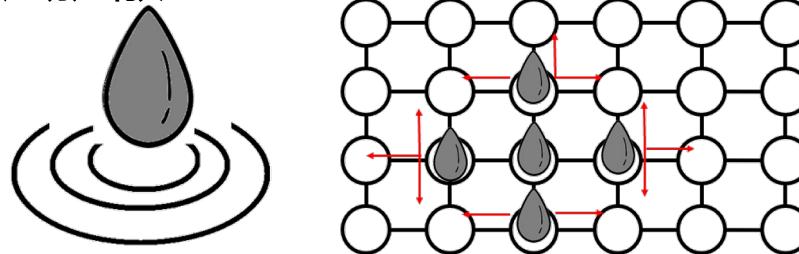
- 算法运行时间： $O(|V| + |E|)$ ，简记为 $O(V + E)$

小结

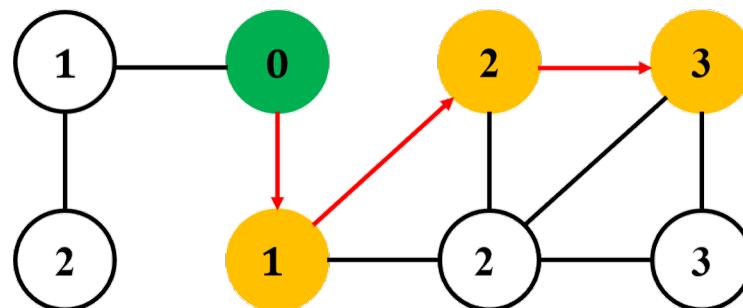


- 广度优先搜索

- 算法核心思想：逐层扩散



- 算法运行时间： $O(|V| + |E|)$ ，简记为 $O(V + E)$
 - 算法相关应用：计算最短路径



图算法篇：深度优先搜索

北京航空航天大学
计算机学院

提纲



问题回顾

算法思想

算法伪代码

算法实例

算法分析

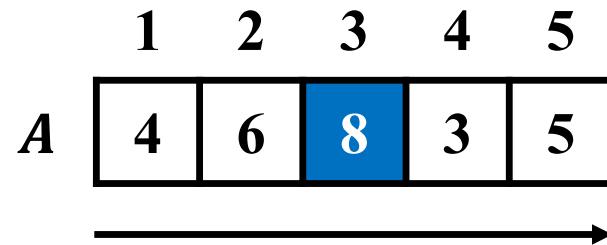
算法性质



图的搜索

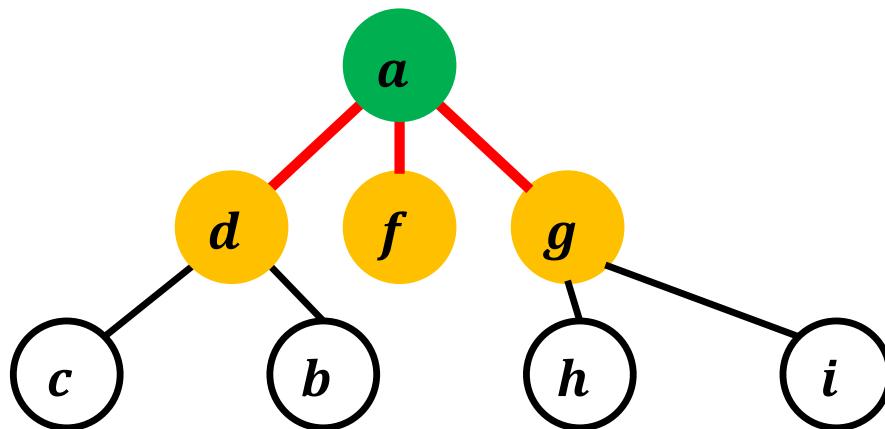
- 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值



- 图结构

- 查询相邻顶点：简单循环搜索各顶点关联的边
- 查询可达顶点：简单循环搜索，不能找到全部可达顶点！是否存在有效算法？



按照什么次序搜索顶点？

广度优先搜索

深度优先搜索

提纲



问题回顾

算法思想

算法伪代码

算法实例

算法分析

算法性质

算法思想

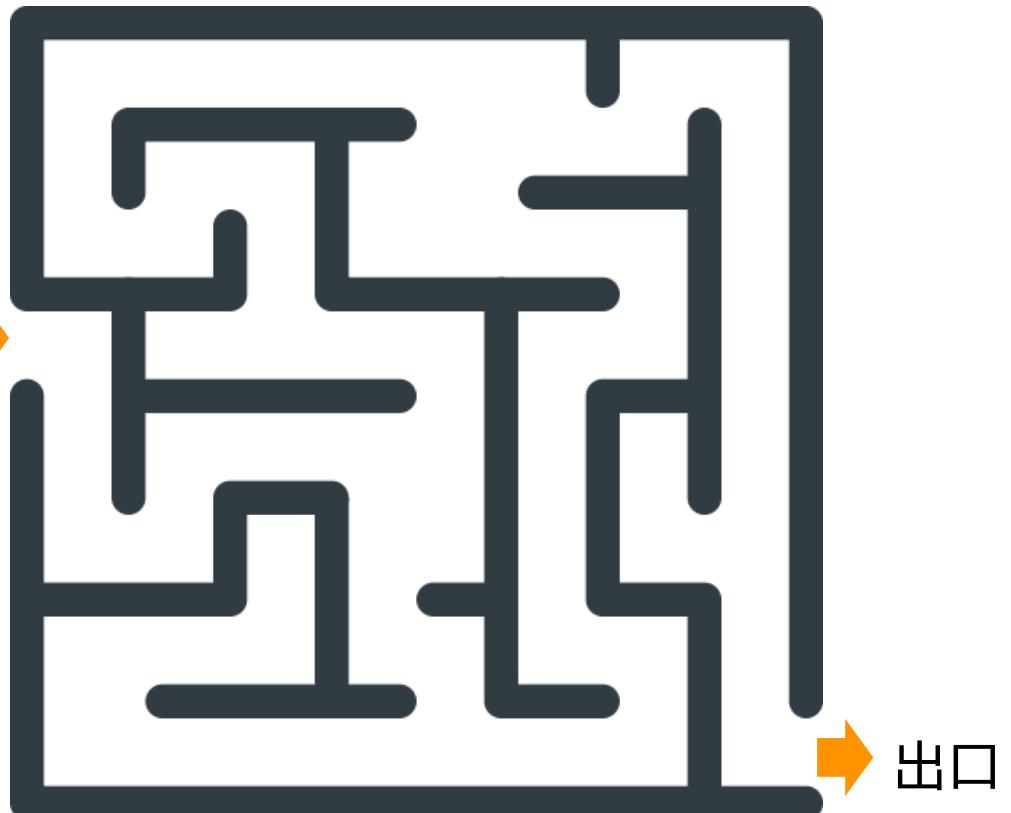


- 走迷宫



问题：如何走出迷宫？

入口

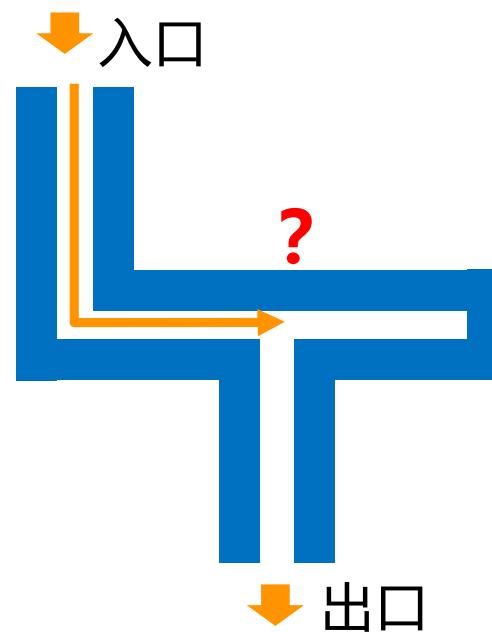


出口

算法思想



- 算法步骤
 - 分叉时，任选一条边深入

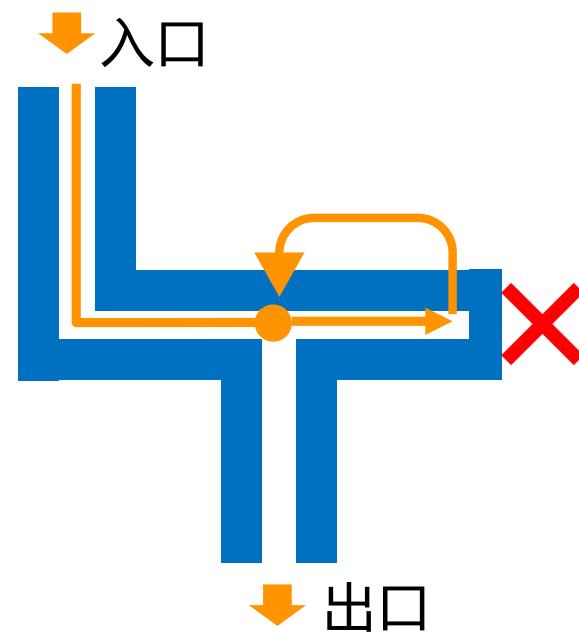


算法思想



● 算法步骤

- 分叉时，任选一条边深入
 - 无边时，后退一步找新边

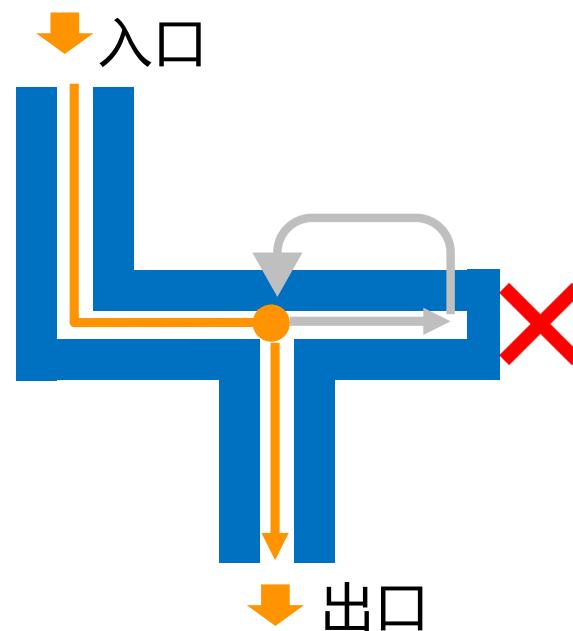


算法思想



• 算法步骤

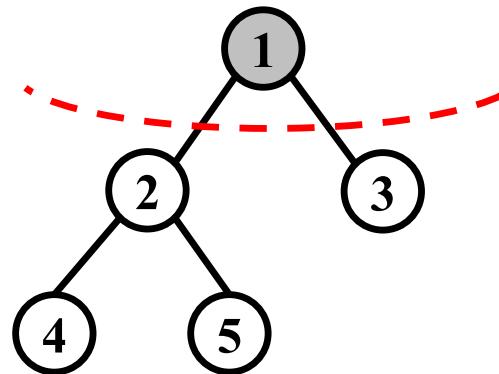
- 分叉时，任选一条边深入
- 无边时，后退一步找新边
- 找到边，从新边继续深入



算法思想



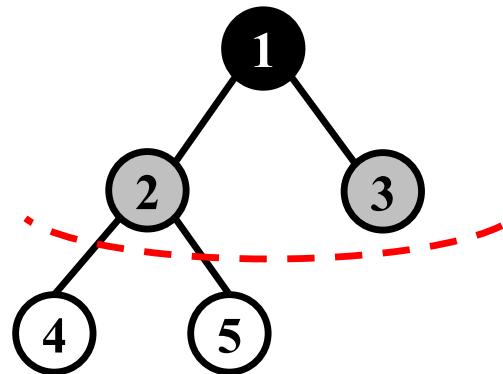
- 广度优先搜索
- 深度优先搜索



算法思想



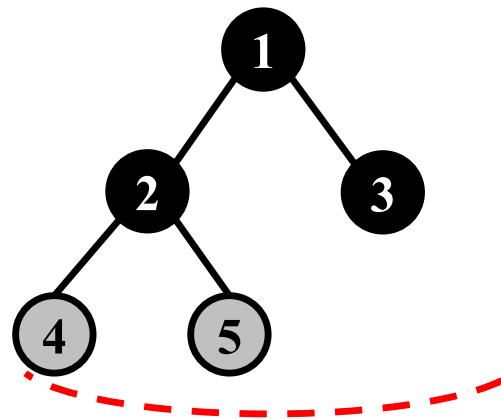
- 广度优先搜索
- 深度优先搜索



算法思想



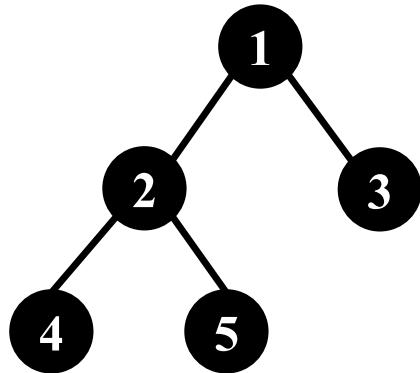
- 广度优先搜索
- 深度优先搜索



算法思想



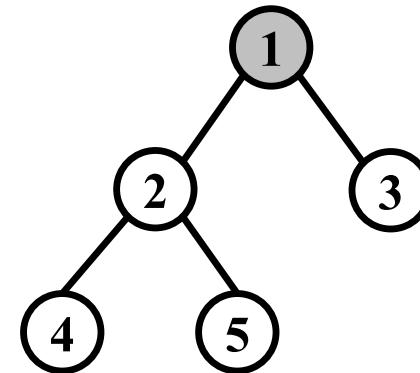
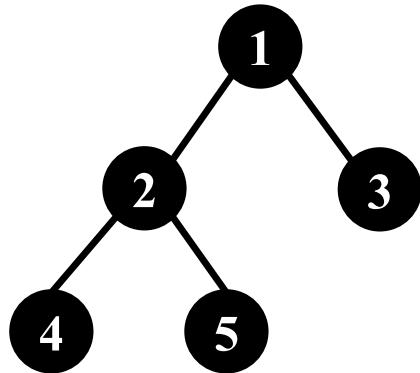
- 广度优先搜索
- 深度优先搜索



算法思想



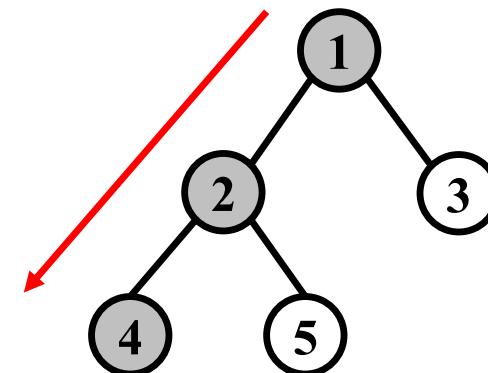
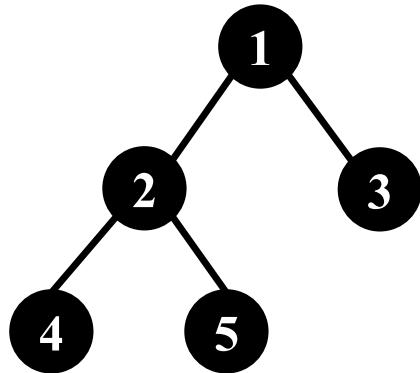
- 广度优先搜索
- 深度优先搜索



算法思想



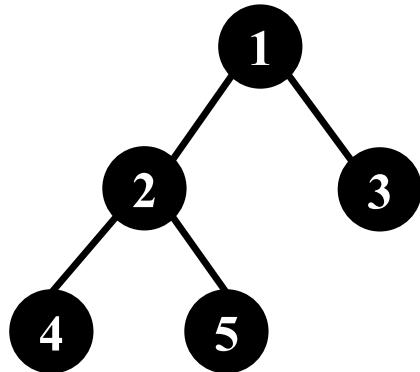
- 广度优先搜索
- 深度优先搜索



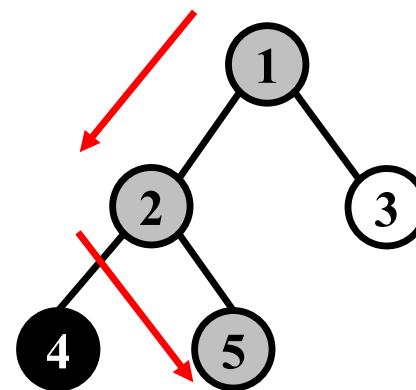
算法思想



- 广度优先搜索



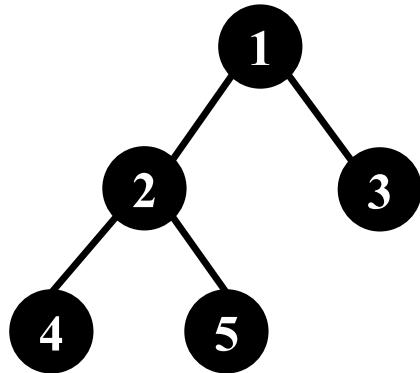
- 深度优先搜索



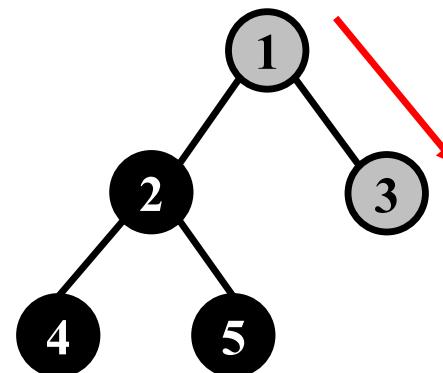
算法思想



- 广度优先搜索



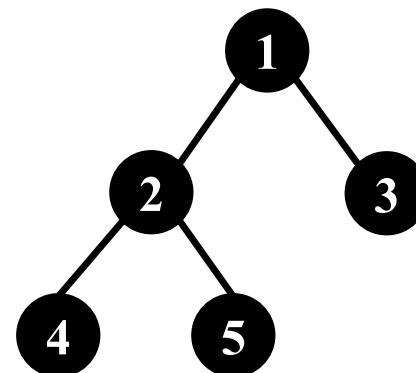
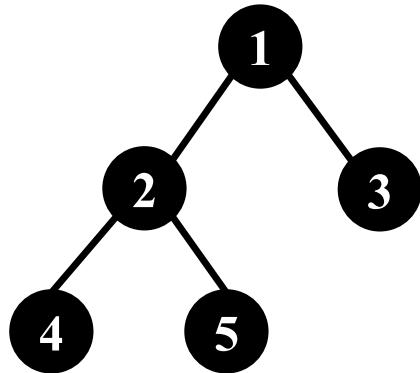
- 深度优先搜索



算法思想



- 广度优先搜索
- 深度优先搜索

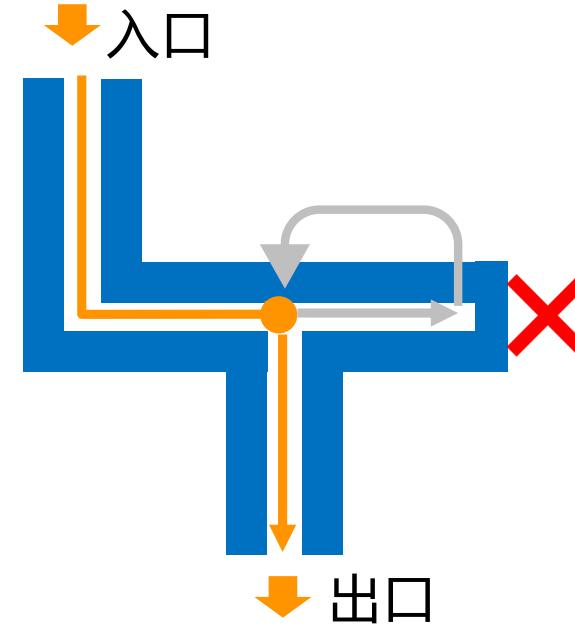




算法思想

- 算法步骤

- 分叉时，任选一条边深入
- 无边时，**后退一步**找新边
- 找到边，从新边继续深入



算法思想

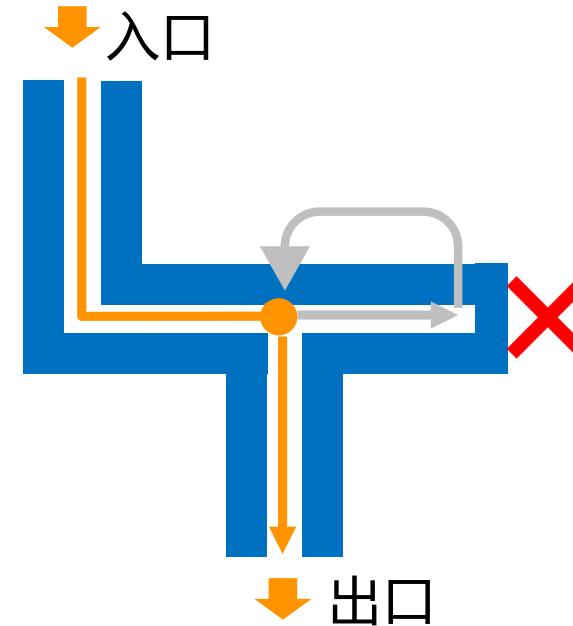


- 算法步骤

- 分叉时，任选一条边深入
- 无边时，**后退一步**找新边
- 找到边，从新边继续深入

- 辅助数组

- *color*: 用颜色表示顶点状态
 - *White*: 白色顶点**尚未被发现**
 - *Black*: 黑色顶点**已被处理**
 - *Gray*: 灰色顶点**正在处理，尚未完成**
- *pred*: 顶点 u 由 $\text{pred}[u]$ 发现
- *d*: 顶点**发现时刻**（变成**灰色**的时刻）
- *f*: 顶点**完成时刻**（变成**黑色**的时刻）



提纲



问题回顾

算法思想

算法伪代码

算法实例

算法分析

算法性质



伪代码

- **DFS(G)**

输入: 图 G

输出: 祖先数组 $pred$, 发现时刻 d , 结束时刻 f

新建数组 $color[1..V], pred[1..V], d[1..V], f[1..V]$

新建数组

//初始化

for $v \in V$ do

| $pred[v] \leftarrow NULL$

| $color[v] \leftarrow WHITE$

end

$time \leftarrow 0$

for $v \in V$ do

| if $color[v] = WHITE$ then

| | DFS-Visit(G, v)

| end

end

return $pred, d, f$

$d[i], f[i]$ 分别记录顶点 i 的发现时刻与结束时刻



伪代码

- **DFS(G)**

输入: 图 G

输出: 祖先数组 $pred$, 发现时刻 d , 结束时刻 f

新建数组 $color[1..V], pred[1..V], d[1..V], f[1..V]$

//初始化

for $v \in V$ do

| $pred[v] \leftarrow NULL$

| $color[v] \leftarrow WHITE$

end

$time \leftarrow 0$

for $v \in V$ do

| if $color[v] = WHITE$ then

| | DFS-Visit(G, v)

| end

end

return $pred, d, f$

初始化



伪代码

- **DFS(G)**

输入: 图 G

输出: 祖先数组 $pred$, 发现时刻 d , 结束时刻 f

新建数组 $color[1..V], pred[1..V], d[1..V], f[1..V]$

//初始化

for $v \in V$ do

| $pred[v] \leftarrow NULL$

| $color[v] \leftarrow WHITE$

end

$time \leftarrow 0$

for $v \in V$ do

| if $color[v] = WHITE$ then

| | DFS-Visit(G, v)

| end

end

return $pred, d, f$

保证搜索完全



伪代码

- **DFS-Visit(G, v)**

输入: 图 G , 顶点 v

```
color[v] ← GRAY  
time ← time + 1  
d[v] ← time  
for w ∈ G.Adj[v] do  
    if color[w] = WHITE then  
        pred[w] ← v  
        DFS-Visit(G, w)  
    end  
end  
color[v] ← BLACK  
time ← time + 1  
f[v] ← time
```

修改当前顶点颜色、发现时刻



伪代码

- **DFS-Visit(G, v)**

输入: 图 G , 顶点 v

$color[v] \leftarrow GRAY$

$time \leftarrow time + 1$

$d[v] \leftarrow time$

for $w \in G.Adj[v]$ do

 if $color[w] = WHITE$ then

$pred[w] \leftarrow v$

 DFS-Visit(G, w)

 end

end

$color[v] \leftarrow BLACK$

$time \leftarrow time + 1$

$f[v] \leftarrow time$

搜索相邻顶点



伪代码

- **DFS-Visit(G, v)**

输入: 图 G , 顶点 v

$color[v] \leftarrow GRAY$

$time \leftarrow time + 1$

$d[v] \leftarrow time$

for $w \in G.Adj[v]$ **do**

if $color[w] = WHITE$ **then**

$pred[w] \leftarrow v$

 DFS-Visit(G, w)

end

end

$color[v] \leftarrow BLACK$

$time \leftarrow time + 1$

$f[v] \leftarrow time$

结束搜索



伪代码

- **DFS(G)**

输入: 图 G

输出: 祖先数组 $pred$, 发现时刻 d , 结束时刻 f

新建数组 $color[1..V], pred[1..V], d[1..V], f[1..V]$

//初始化

```
for  $v \in V$  do
    |  $pred[v] \leftarrow NULL$ 
    |  $color[v] \leftarrow WHITE$ 
```

end

$time \leftarrow 0$

```
for  $v \in V$  do
    | if  $color[v] = WHITE$  then
        | | DFS-Visit( $G, v$ )
    end
```

end

return $pred, d, f$

DFS-Visit(G, v)

输入: 图 G , 顶点 v

$color[v] \leftarrow GRAY$

$time \leftarrow time + 1$

$d[v] \leftarrow time$

for $w \in G.Adj[v]$ do

```
    | if  $color[w] = WHITE$  then
        | |  $pred[w] \leftarrow v$ 
        | | DFS-Visit( $G, w$ )
    end
```

end

$color[v] \leftarrow BLACK$

$time \leftarrow time + 1$

$f[v] \leftarrow time$

提纲



问题回顾

算法思想

算法实例

算法分析

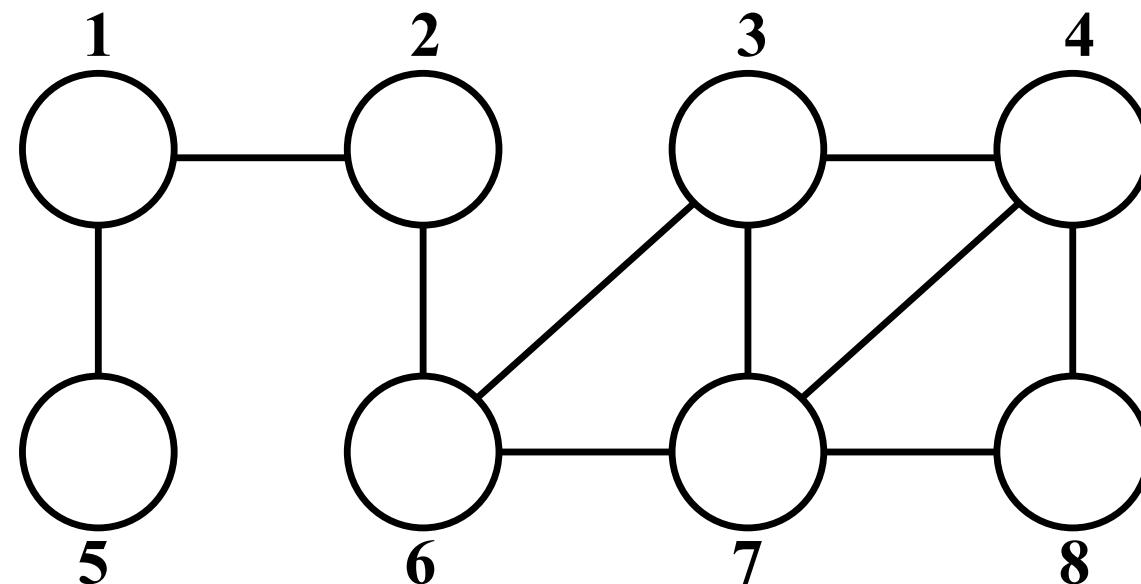
算法性质

算法实例



V	1	2	3	4	5	6	7	8	$time = 0$
$color$	W	W	W	W	W	W	W	W	
$pred$	N	N	N	N	N	N	N	N	
d									
f									

搜索源点

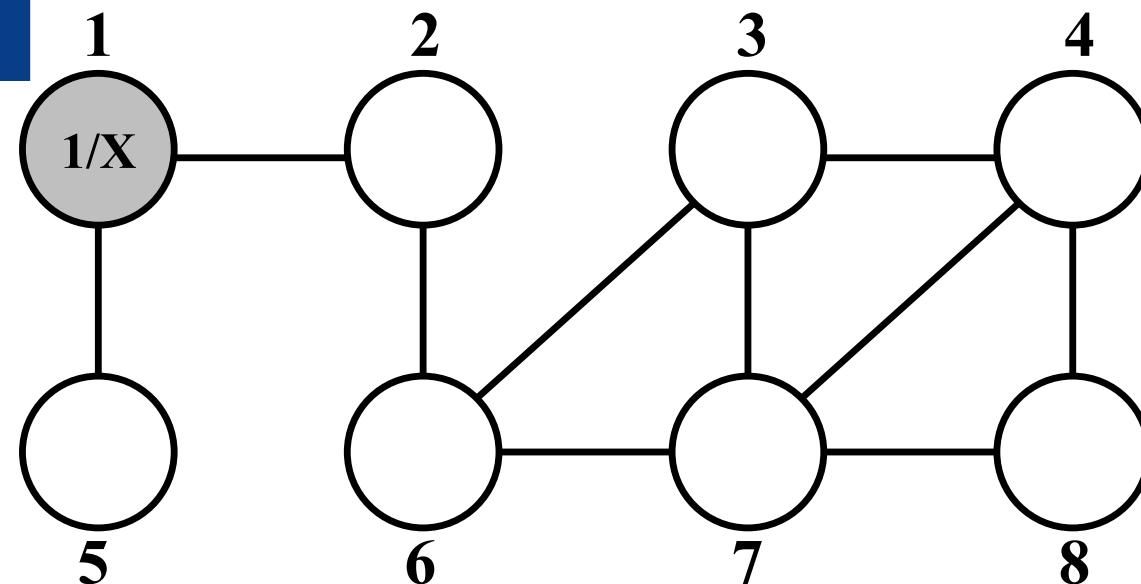


算法实例



V	1	2	3	4	5	6	7	8	$time = 1$
$color$	G	W	W	W	W	W	W	W	
$pred$	N	N	N	N	N	N	N	N	
d	1								
f									

发现时刻为1

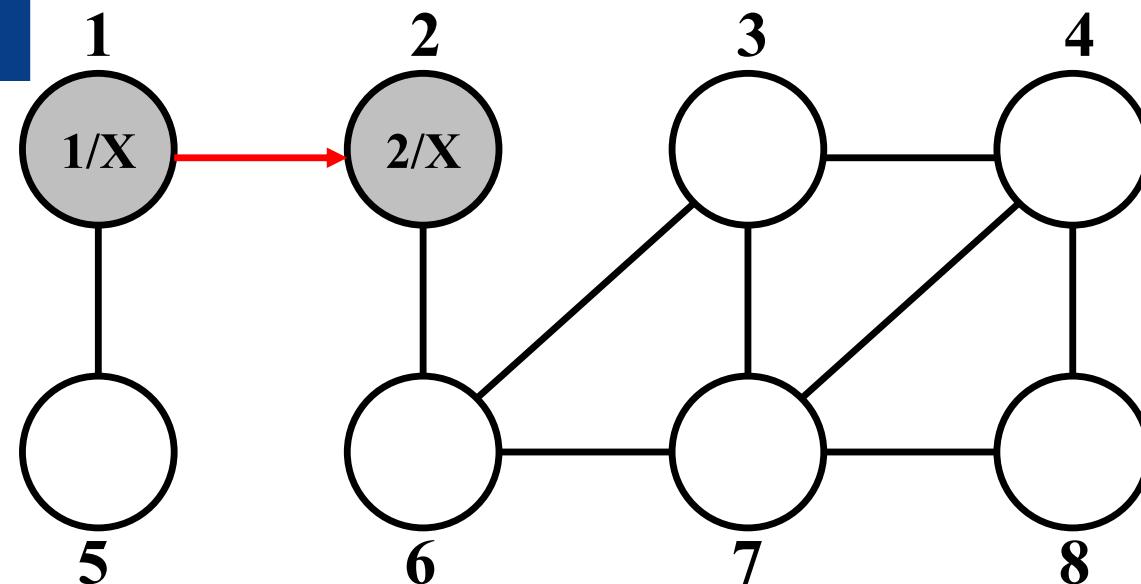


算法实例



V	1	2	3	4	5	6	7	8	$time = 2$
$color$	G	G	W	W	W	W	W	W	
$pred$	N	1	N	N	N	N	N	N	
d	1	2							
f									

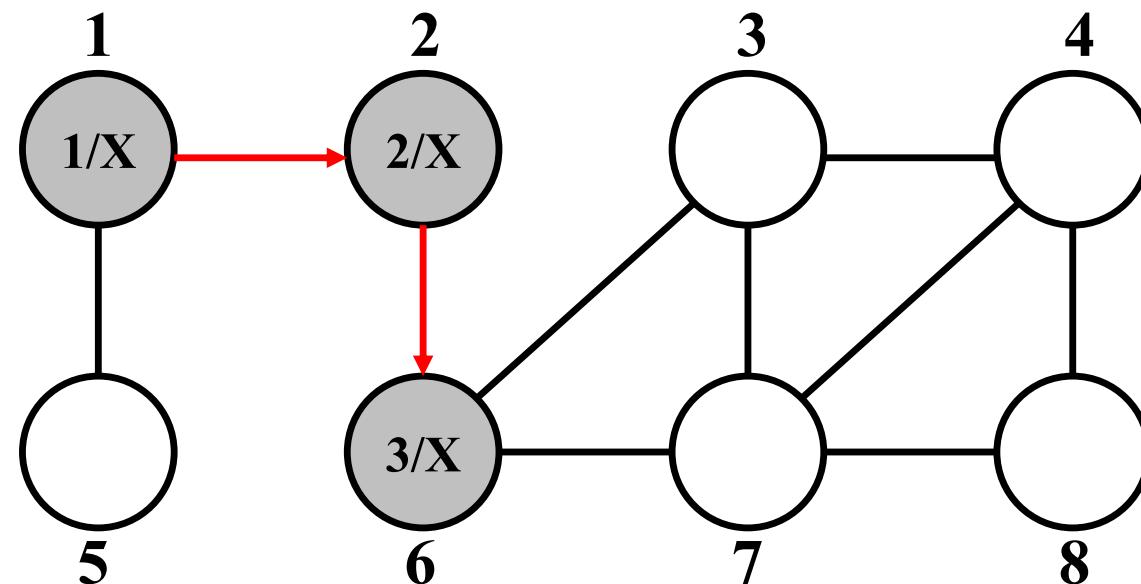
发现时刻为2



算法实例



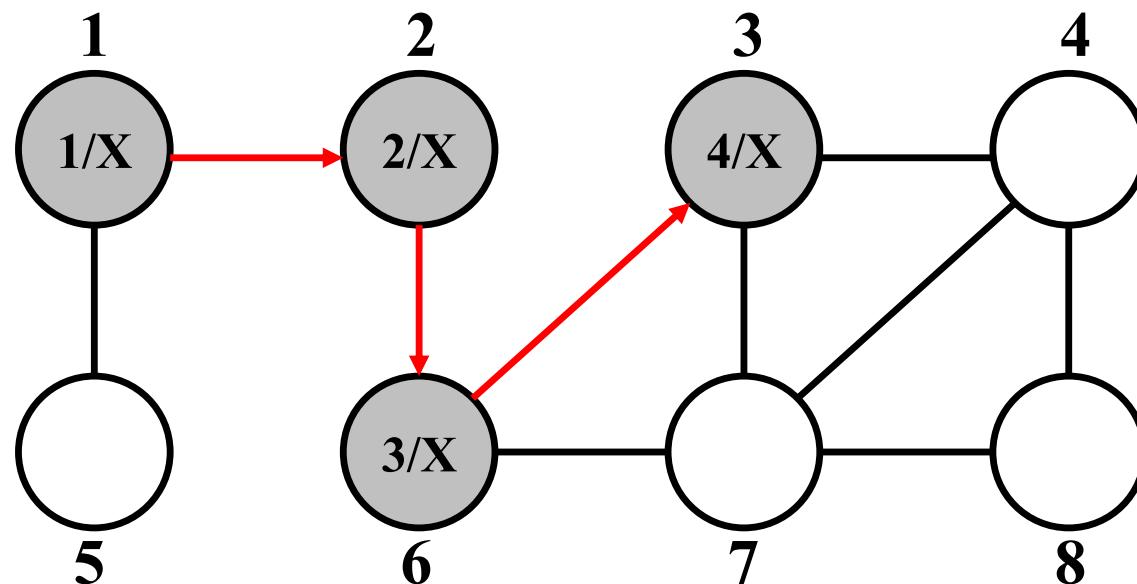
V	1	2	3	4	5	6	7	8	$time = 3$
$color$	G	G	W	W	W	G	W	W	
$pred$	N	1	N	N	N	2	N	N	
d	1	2				3			
f									



算法实例



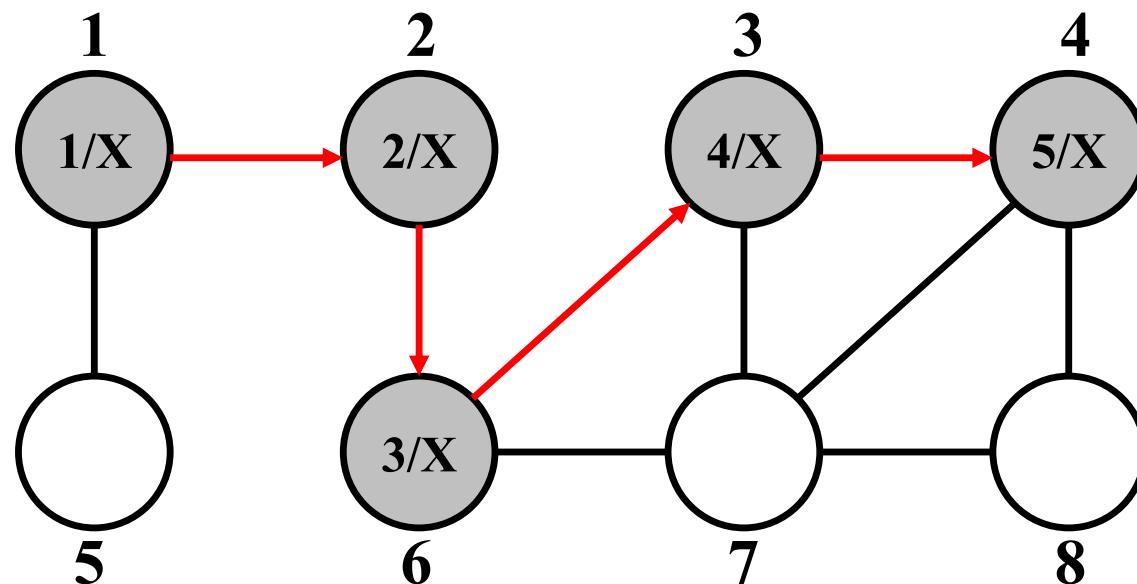
V	1	2	3	4	5	6	7	8	$time = 4$
$color$	G	G	G	W	W	G	W	W	
$pred$	N	1	6	N	N	2	N	N	
d	1	2	4			3			
f									



算法实例



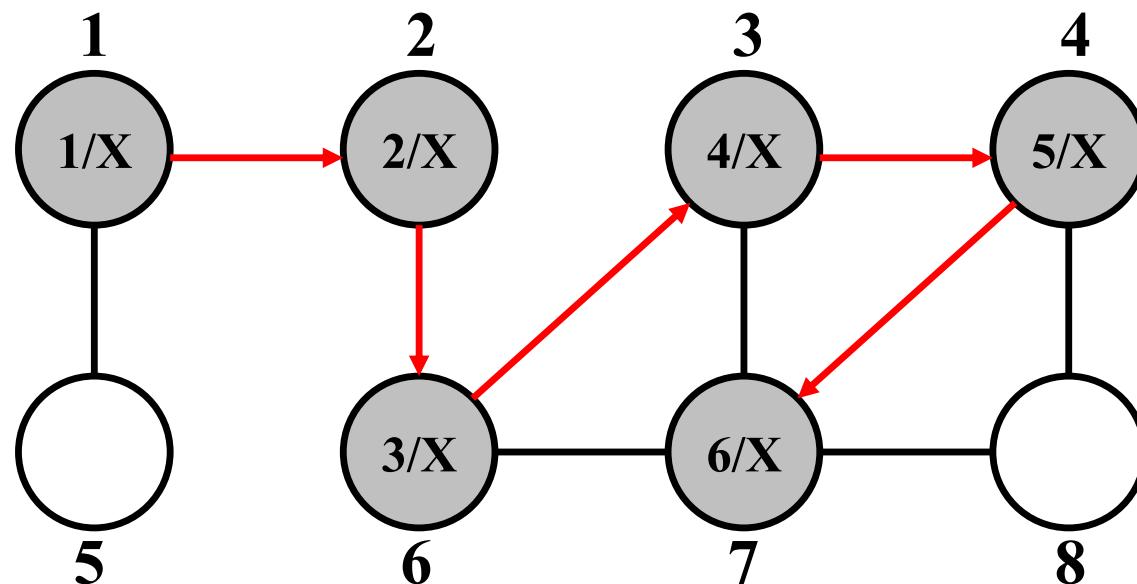
V	1	2	3	4	5	6	7	8	$time = 5$
$color$	G	G	G	G	W	G	W	W	
$pred$	N	1	6	3	N	2	N	N	
d	1	2	4	5		3			
f									



算法实例



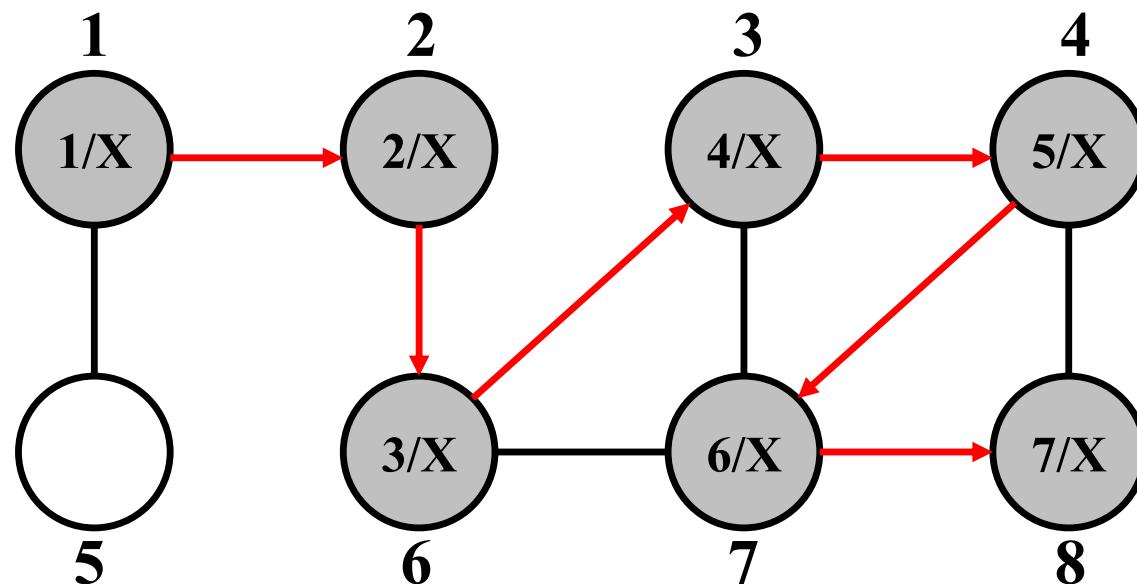
V	1	2	3	4	5	6	7	8	$time = 6$
$color$	G	G	G	G	W	G	G	W	
$pred$	N	1	6	3	N	2	4	N	
d	1	2	4	5		3	6		
f									



算法实例



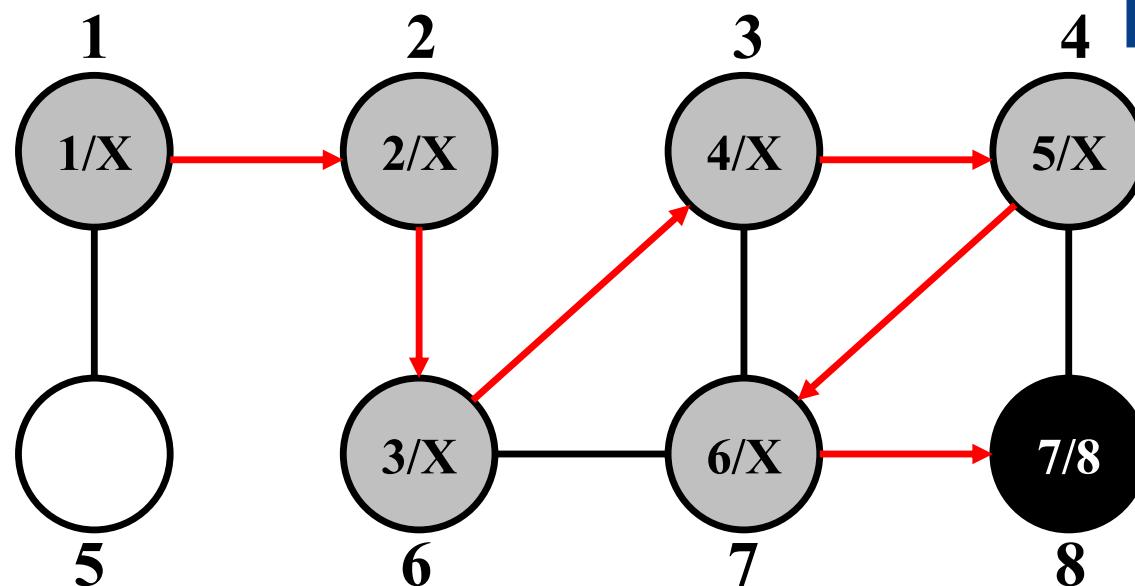
V	1	2	3	4	5	6	7	8	$time = 7$
$color$	G	G	G	G	W	G	G	G	
$pred$	N	1	6	3	N	2	4	7	
d	1	2	4	5		3	6	7	
f									



算法实例



V	1	2	3	4	5	6	7	8	$time = 8$
$color$	G	G	G	G	W	G	G	B	
$pred$	N	1	6	3	N	2	4	7	
d	1	2	4	5		3	6	7	
f								8	

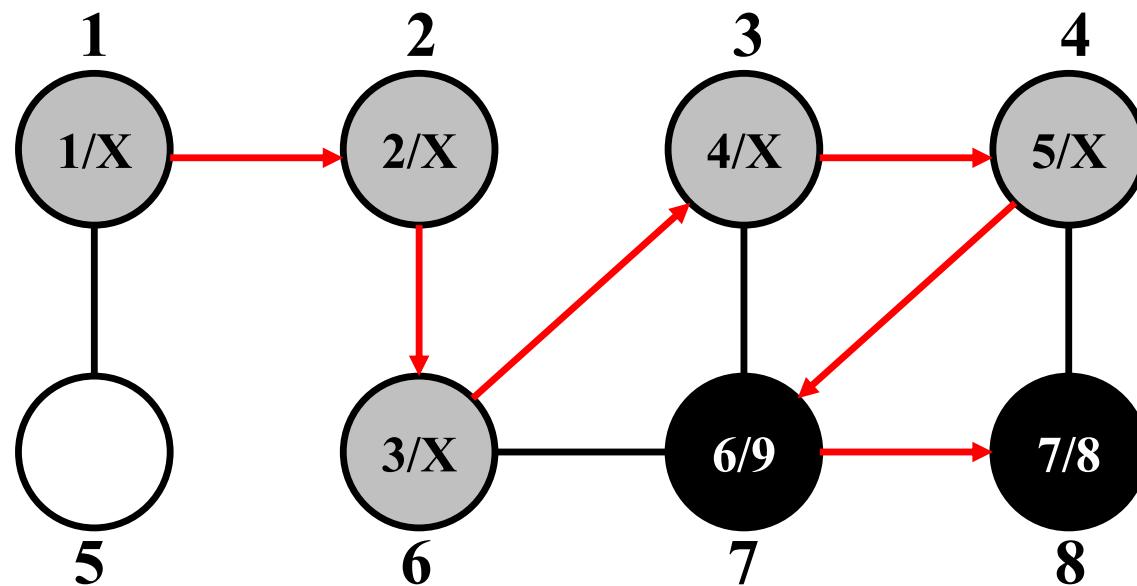


结束时刻为8

算法实例



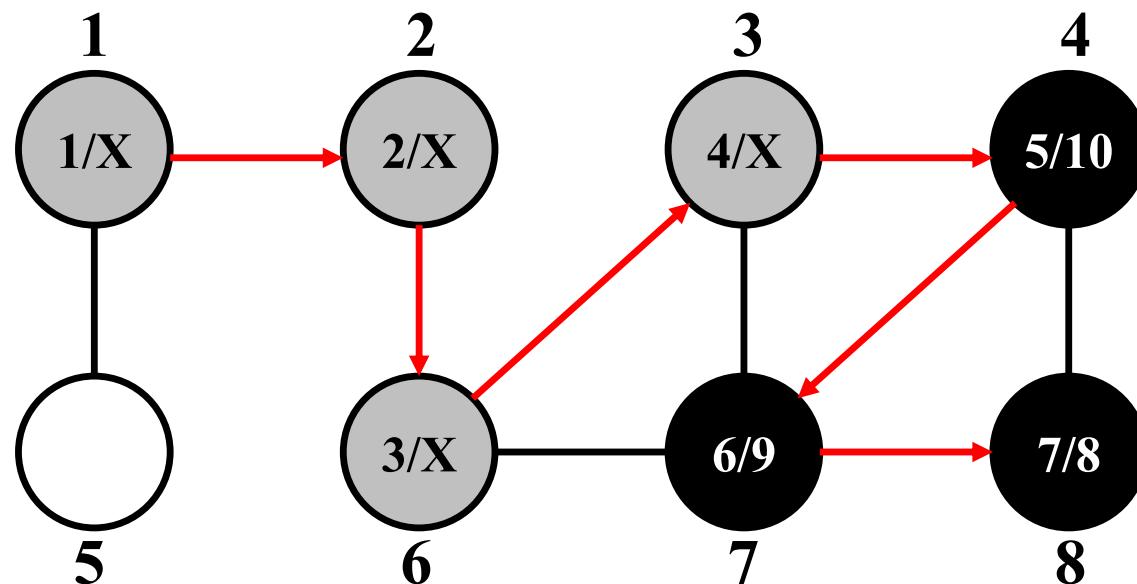
V	1	2	3	4	5	6	7	8	$time = 9$
$color$	G	G	G	G	W	G	B	B	
$pred$	N	1	6	3	N	2	4	7	
d	1	2	4	5		3	6	7	
f							9	8	



算法实例



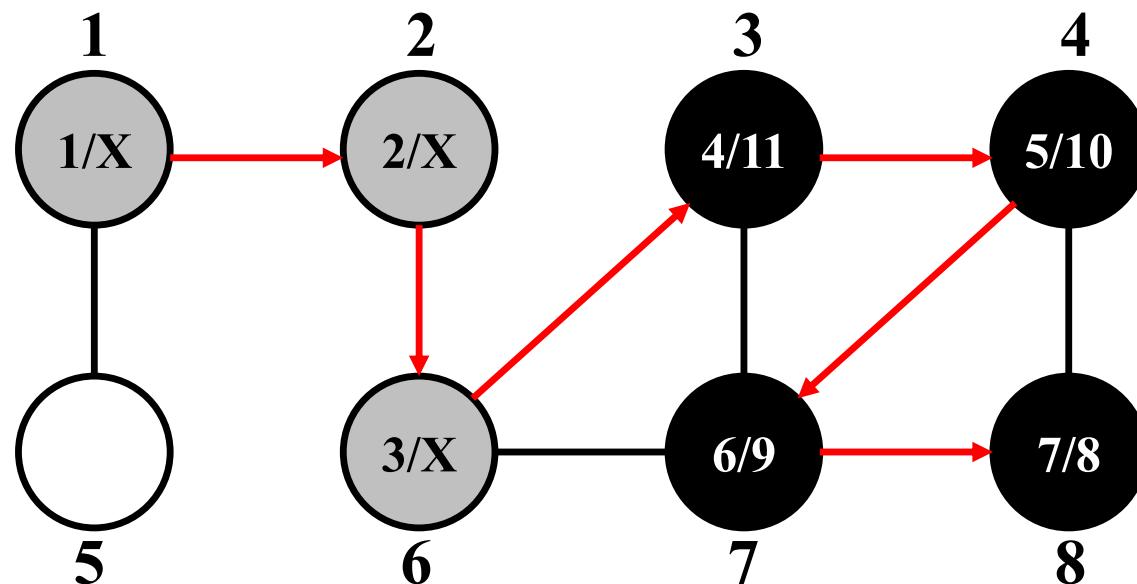
V	1	2	3	4	5	6	7	8	$time = 10$
$color$	G	G	G	B	W	G	B	B	
$pred$	N	1	6	3	N	2	4	7	
d	1	2	4	5		3	6	7	
f				10			9	8	



算法实例



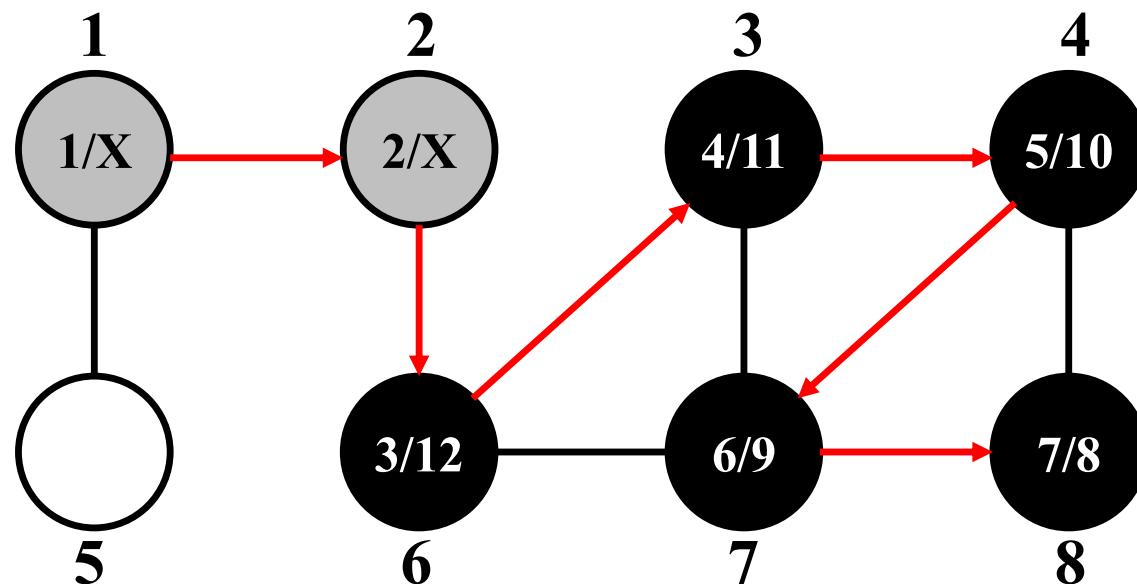
V	1	2	3	4	5	6	7	8	$time = 11$
$color$	G	G	B	B	W	G	B	B	
$pred$	N	1	6	3	N	2	4	7	
d	1	2	4	5		3	6	7	
f			11	10			9	8	



算法实例



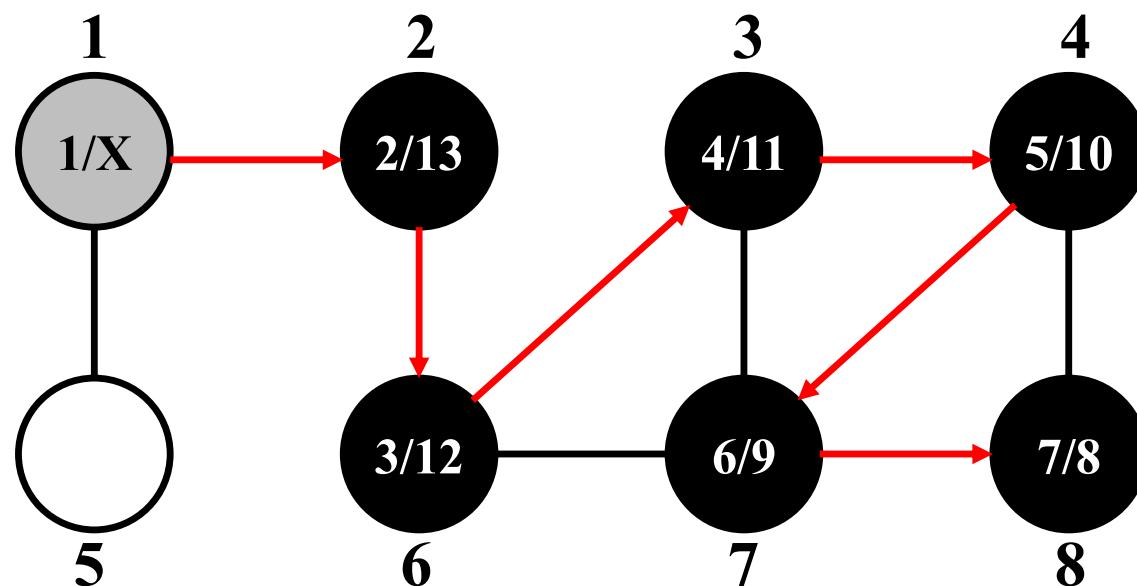
V	1	2	3	4	5	6	7	8	$time = 12$
$color$	G	G	B	B	W	B	B	B	
$pred$	N	1	6	3	N	2	4	7	
d	1	2	4	5		3	6	7	
f			11	10		12	9	8	



算法实例



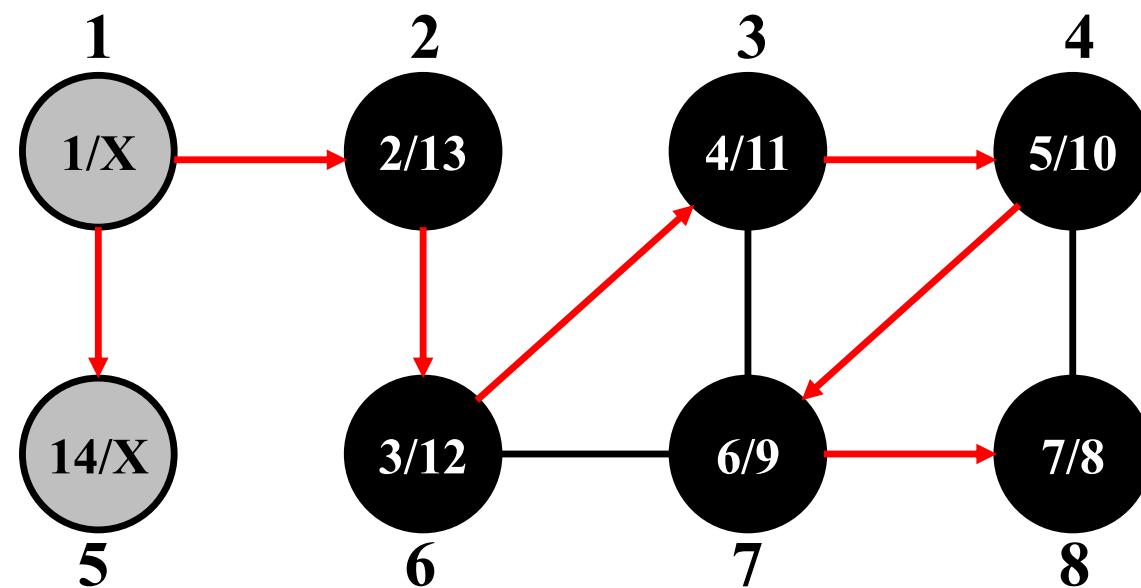
V	1	2	3	4	5	6	7	8	$time = 13$
$color$	G	B	B	B	W	B	B	B	
$pred$	N	1	6	3	N	2	4	7	
d	1	2	4	5		3	6	7	
f		13	11	10		12	9	8	



算法实例



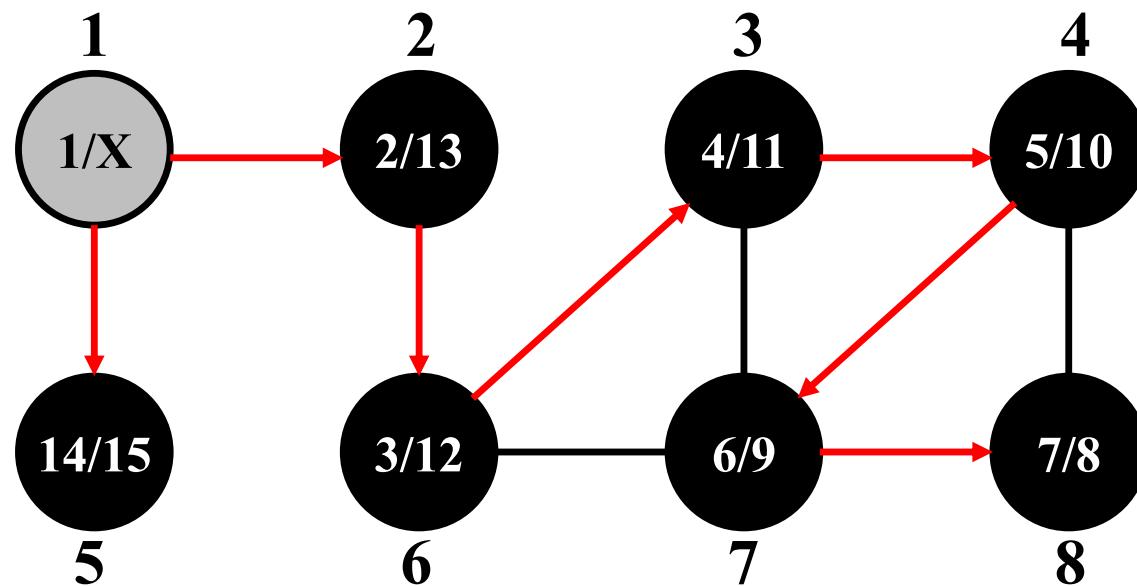
V	1	2	3	4	5	6	7	8	$time = 14$
$color$	G	B	B	B	G	B	B	B	
$pred$	N	1	6	3	1	2	4	7	
d	1	2	4	5	14	3	6	7	
f		13	11	10		12	9	8	



算法实例



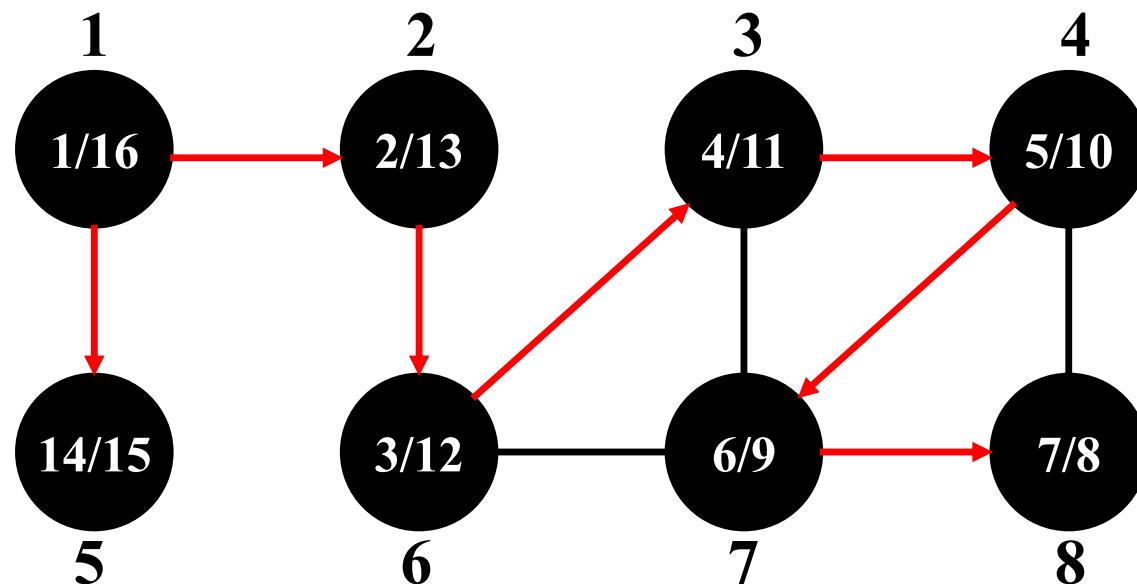
V	1	2	3	4	5	6	7	8	$time = 15$
$color$	G	B	B	B	B	B	B	B	
$pred$	N	1	6	3	1	2	4	7	
d	1	2	4	5	14	3	6	7	
f		13	11	10	15	12	9	8	



算法实例



V	1	2	3	4	5	6	7	8	$time = 16$
$color$	B	B	B	B	B	B	B	B	
$pred$	N	1	6	3	1	2	4	7	
d	1	2	4	5	14	3	6	7	
f	16	13	11	10	15	12	9	8	



提纲



问题回顾

算法思想

算法伪代码

算法实例

算法分析

算法性质



复杂度分析

- 尝试递归算法常用的主定理和递归树分析 子问题个数：不确定

输入: 图 G , 顶点 v

$color[v] \leftarrow GRAY$

$time \leftarrow time + 1$

$d[v] \leftarrow time$

for $w \in G.Adj[v]$ do

子问题个数取决于顶点的度

 if $color[w] = WHITE$ then

$pred[w] \leftarrow v$

 DFS-Visit(G, w)

 end

end

$color[v] \leftarrow BLACK$

$time \leftarrow time + 1$

$f[v] \leftarrow time$

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$



复杂度分析

- 尝试递归算法常用的主定理和递归树分析

子问题个数：不确定

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

子问题规模：不确定

```
输入: 图 $G$ , 顶点 $v$ 
color[ $v$ ]  $\leftarrow$  GRAY
time  $\leftarrow$  time + 1
d[ $v$ ]  $\leftarrow$  time
for  $w \in G.Adj[v]$  do
    if color[ $w$ ] = WHITE then
        pred[ $w$ ]  $\leftarrow v$ 
        DFS-Visit( $G, w$ )
    end
end
color[ $v$ ]  $\leftarrow$  BLACK
time  $\leftarrow$  time + 1
f[ $v$ ]  $\leftarrow$  time
```

子问题规模取决于顶点的连通性



复杂度分析

- 尝试递归算法常用的主定理和递归树分析

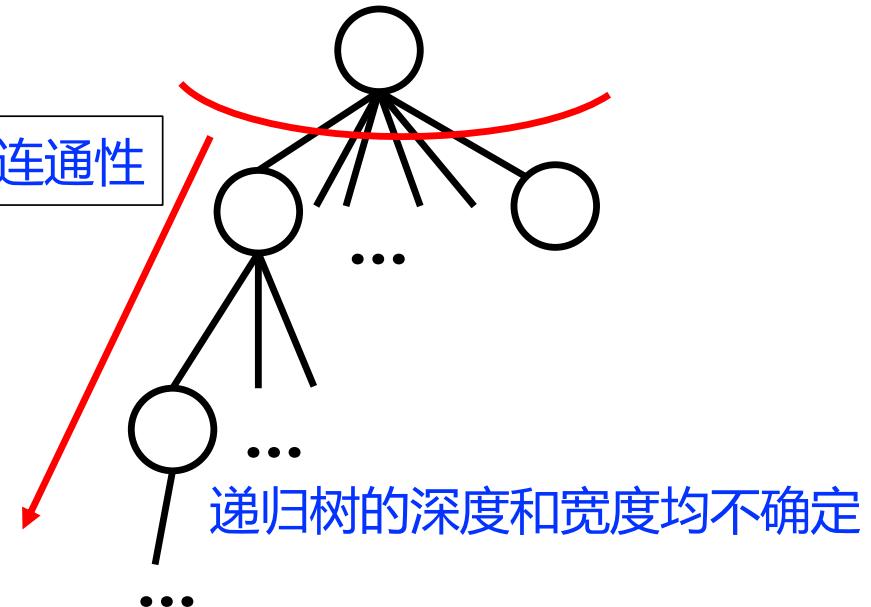
子问题个数：不确定

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

子问题规模：不确定

```
输入: 图 $G$ , 顶点 $v$ 
color[ $v$ ]  $\leftarrow$  GRAY
time  $\leftarrow$  time + 1
d[ $v$ ]  $\leftarrow$  time
for  $w \in G.Adj[v]$  do
    if color[ $w$ ] = WHITE then
        pred[ $w$ ]  $\leftarrow v$ 
        DFS-Visit( $G, w$ )
    end
end
color[ $v$ ]  $\leftarrow$  BLACK
time  $\leftarrow$  time + 1
f[ $v$ ]  $\leftarrow$  time
```

子问题规模取决于顶点的连通性





复杂度分析

- 尝试递归算法常用的主定理和递归树分析

子问题个数：不确定

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

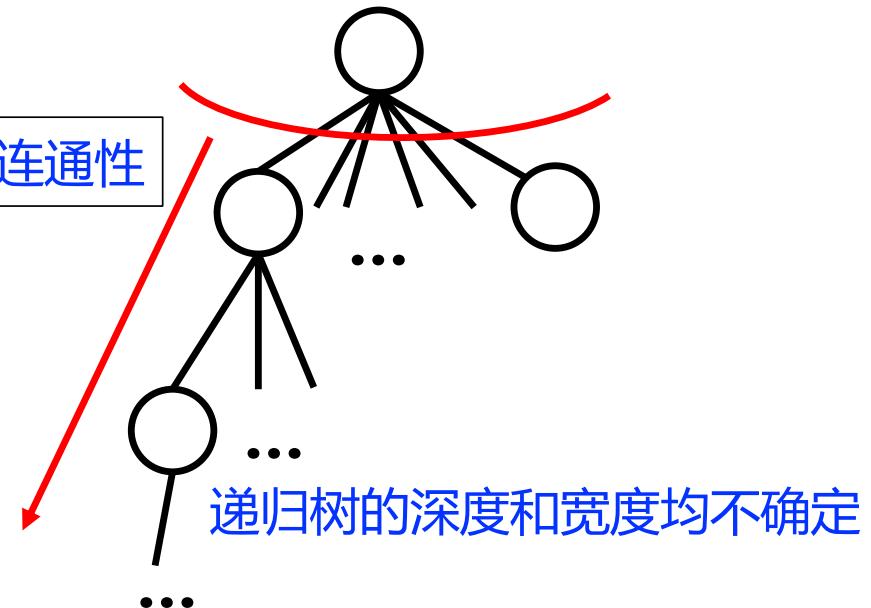
子问题规模：不确定

```
输入: 图G, 顶点v  
color[v] ← GRAY  
time ← time + 1  
d[v] ← time  
for w ∈ G.Adj[v] do  
    if color[w] = WHITE then  
        pred[w] ← v  
        DFS-Visit(G, w)  
    end  
end  
color[v] ← BLACK  
time ← time + 1  
f[v] ← time
```

子问题规模取决于顶点的连通性

能否借助广度优先搜索复杂度分析思想？

递归树的深度和宽度均不确定





回顾：广度优先搜索算法复杂度分析

- 对于每个顶点 u ，搜索相邻顶点消耗时间 $T_u = O(1 + \deg(u))$
- 总运行时间：

$$\begin{aligned} T &= \sum_{u \in V} T_u && \text{分析每个顶点的时间开销} \\ &\leq \sum_{u \in V} O(1 + \deg(u)) \\ &= \sum_{u \in V} O(1) + \sum_{u \in V} O(\deg(u)) \\ &= O(|V| + |E|) \end{aligned}$$

广度优先搜索的时间复杂度为 $O(|V| + |E|)$



复杂度分析

输入: 图 G , 顶点 v

$color[v] \leftarrow GRAY$

$time \leftarrow time + 1$

$d[v] \leftarrow time$

for $w \in G.Adj[v]$ do

 if $color[w] = WHITE$ then

$pred[w] \leftarrow v$

 DFS-Visit(G, w)

 end

end

$color[v] \leftarrow BLACK$

$time \leftarrow time + 1$

$f[v] \leftarrow time$

搜索顶点 w 的开销

$O(1)$

$O(|Adj[v]|)$

搜索顶点 v 的开销
 $O(1 + |Adj[v]|)$

- 搜索顶点 v 的开销: $O(1 + |Adj[v]|)$



复杂度分析

输入: 图 G , 顶点 v

$color[v] \leftarrow GRAY$

$time \leftarrow time + 1$

$d[v] \leftarrow time$

for $w \in G.Adj[v]$ do

if $color[w] = WHITE$ then

$pred[w] \leftarrow v$

DFS-Visit(G, w)

end

end

$color[v] \leftarrow BLACK$

$time \leftarrow time + 1$

$f[v] \leftarrow time$

搜索后不是白色

只有白色才搜索

搜索后不是白色

- 搜索顶点 v 的开销: $O(1 + |Adj[v]|)$
- 每个顶点只搜索一次, 共 $|V|$ 次



复杂度分析

```
输入: 图 $G$ , 顶点 $v$ 
color[ $v$ ]  $\leftarrow$  GRAY
time  $\leftarrow$  time + 1
d[ $v$ ]  $\leftarrow$  time
for  $w \in G.Adj[v]$  do
    if color[ $w$ ] = WHITE then
        pred[ $w$ ]  $\leftarrow v$ 
        DFS-Visit( $G, w$ )
    end
end
color[ $v$ ]  $\leftarrow$  BLACK
time  $\leftarrow$  time + 1
f[ $v$ ]  $\leftarrow$  time
```

- 搜索顶点 v 的开销: $O(1 + |Adj[v]|)$
- 每个顶点只搜索一次, 共 $|V|$ 次
- 总时间复杂度
 - $O(\sum_{v \in V}(1 + |Adj[v]|)) = O(|V| + \sum_{v \in V} |Adj[v]|) = O(|V| + |E|)$

$$\sum_{v \in V} \deg(v) = O(|E|)$$

提纲



问题回顾

算法思想

算法伪代码

算法实例

算法分析

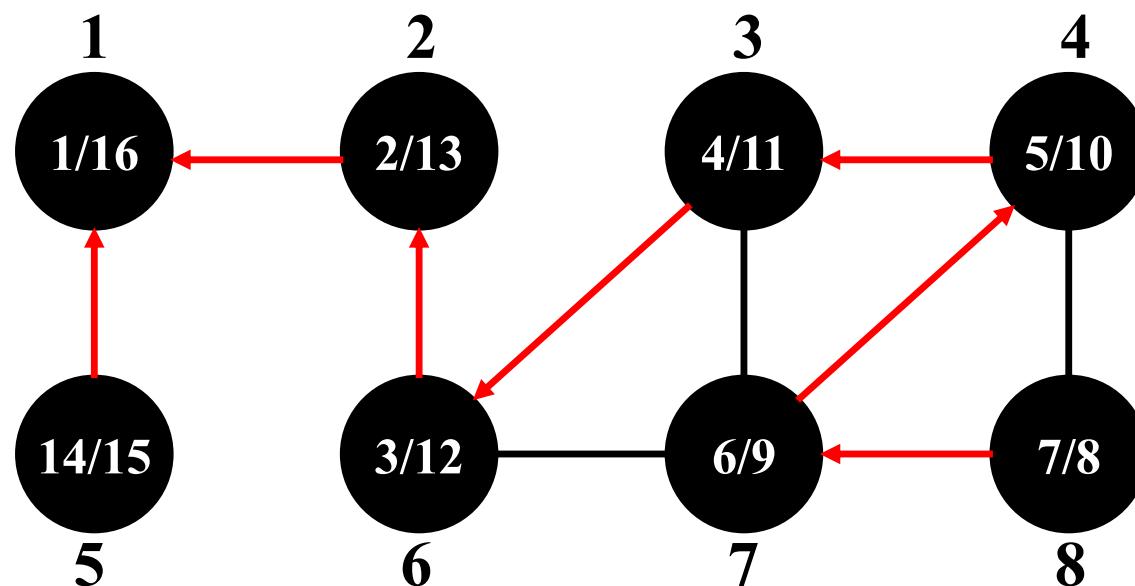
算法性质



深度优先树

V	1	2	3	4	5	6	7	8
$color$	B	B	B	B	B	B	B	B
$pred$	N	1	6	3	1	2	4	7
d	1	2	4	5	14	3	6	7
f	16	13	11	10	15	12	9	8

深度优先树：顶点以前驱为祖先形成的树

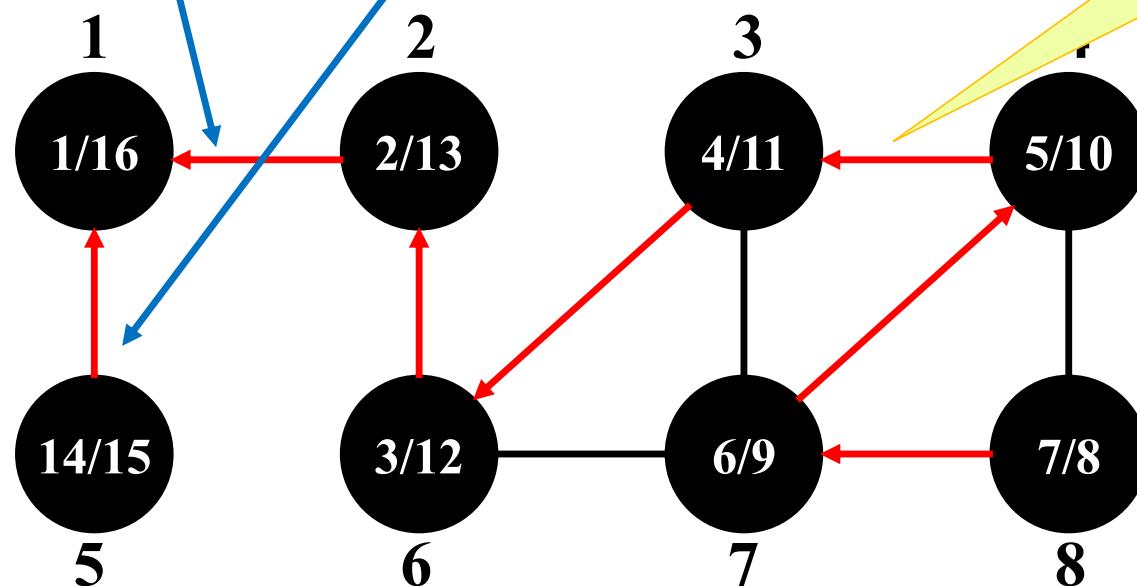


深度优先树



V	1	2	3	4	5	6	7	8
$color$	B	B	B	B	B	B	B	B
$pred$	N	1	6	3	1	2	4	7
d	1	2	4	5	14	3	6	7
f	16	13	11	10	15	12	9	8

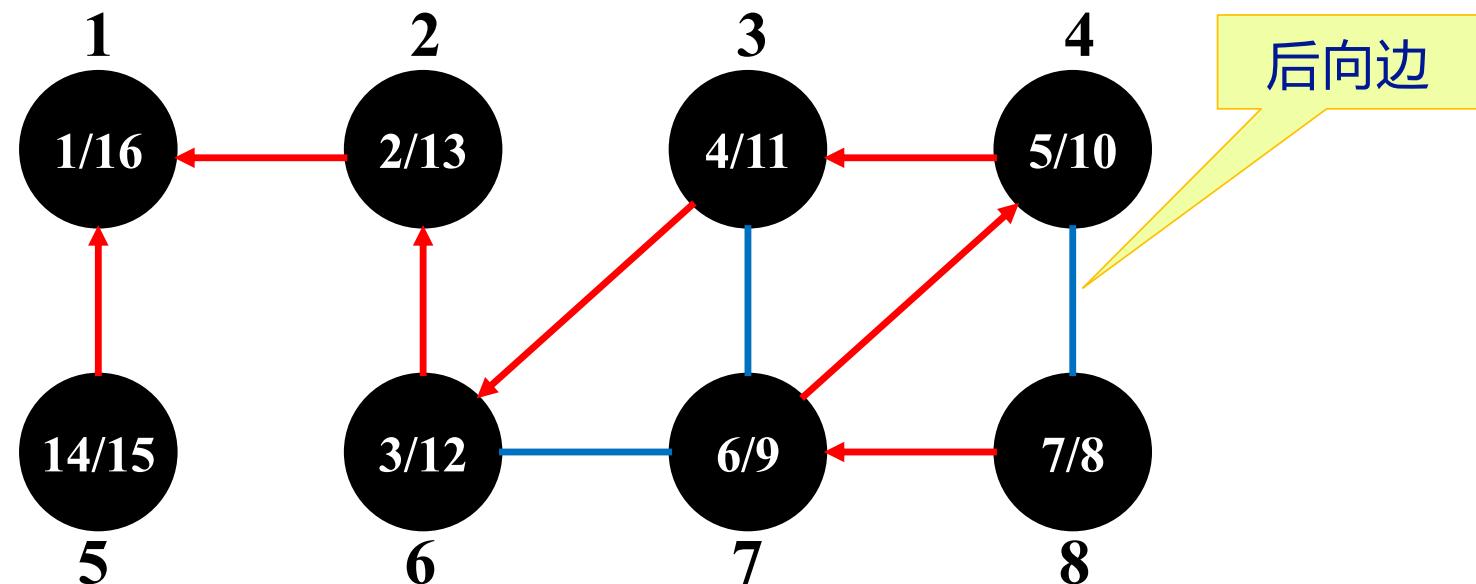
树边：
深度优先树中的边



边的性质



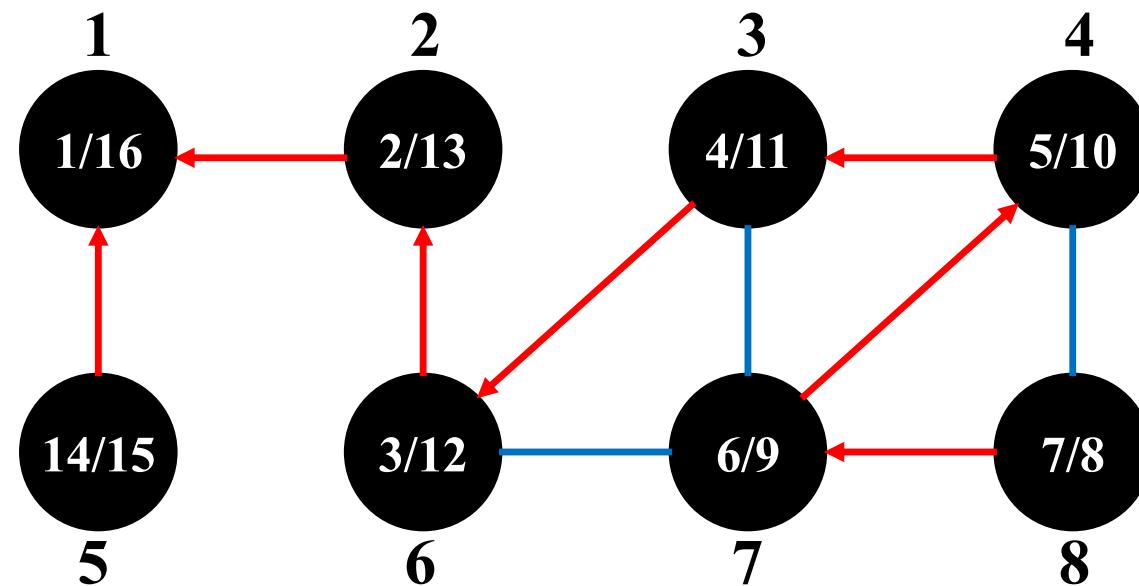
- 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系



边的性质



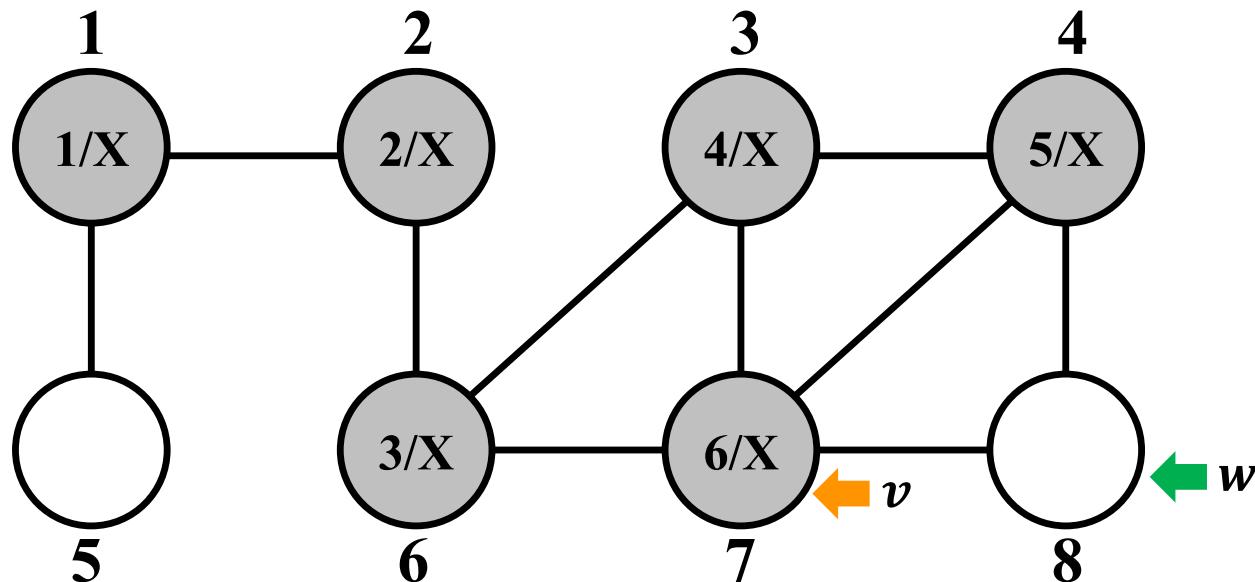
- 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系
- 对于无向图，非树边一定是后向边





边的性质

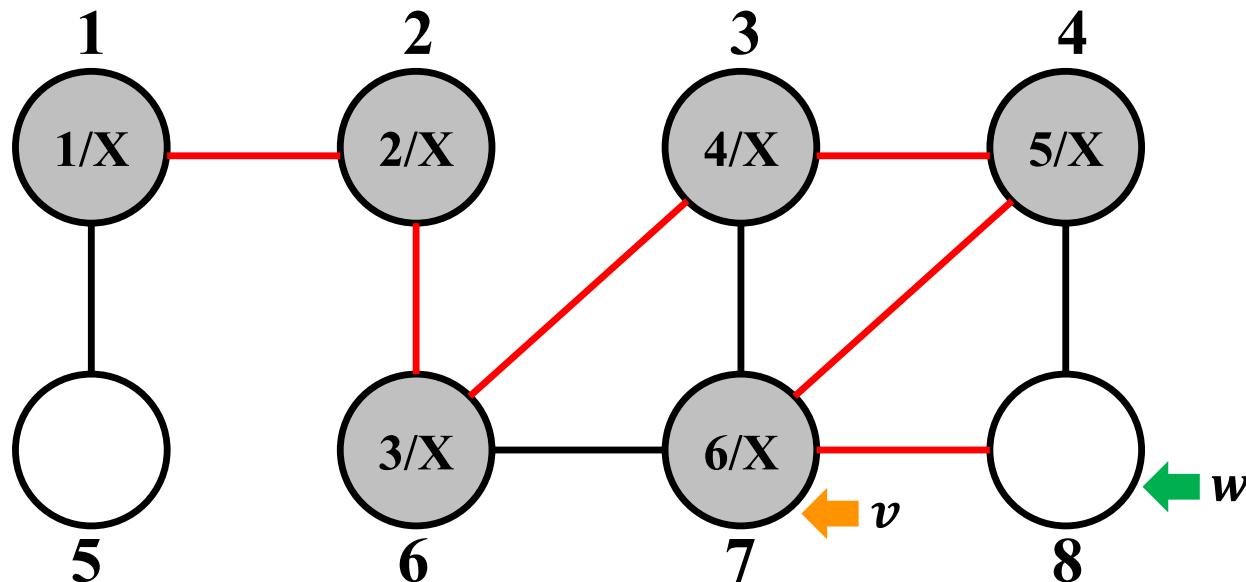
- 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系
- 对于无向图，非树边一定是后向边
 - 证明：从顶点 v ，搜索顶点 w





边的性质

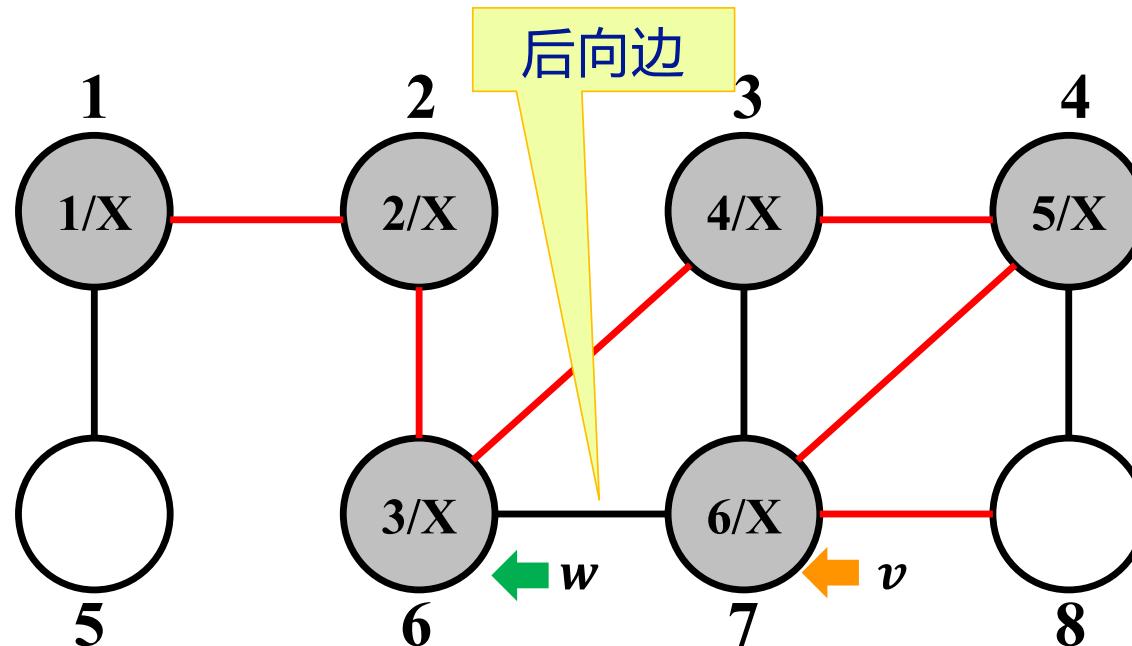
- 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系
- 对于无向图，非树边一定是后向边
 - 证明：从顶点 v ，搜索顶点 w
 - 若 w 为白色， (v, w) 为树边





边的性质

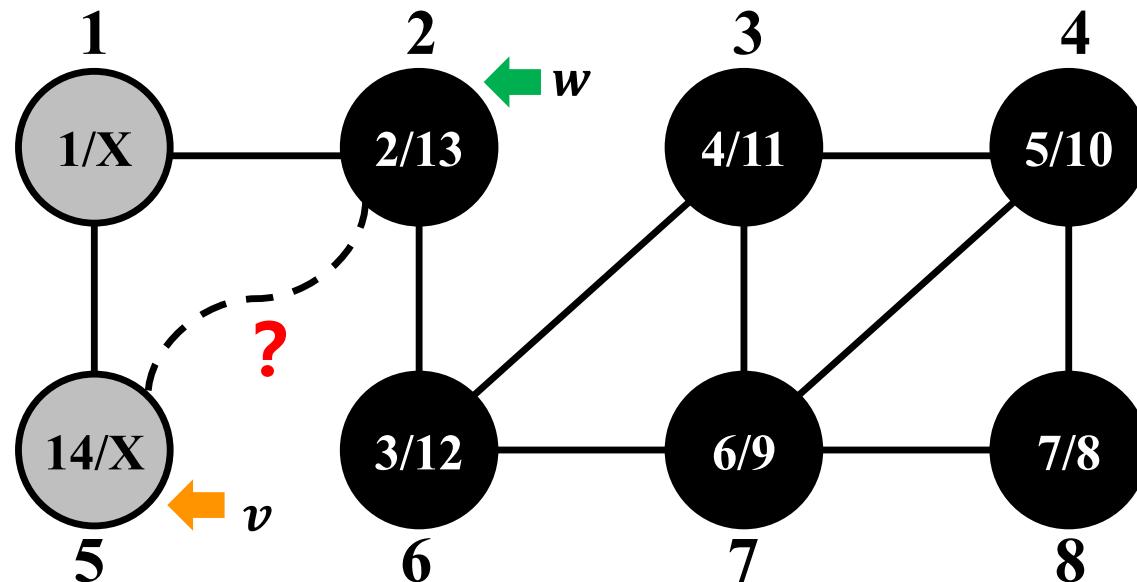
- 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系
- 对于无向图，非树边一定是后向边
 - 证明：从顶点 v ，搜索顶点 w
 - 若 w 为白色， (v, w) 为树边
 - 若 w 为灰色， (v, w) 为后向边





边的性质

- 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系
- 对于无向图，非树边一定是后向边
 - 证明：从顶点 v ，搜索顶点 w
 - 若 w 为白色， (v, w) 为树边
 - 若 w 为灰色， (v, w) 为后向边
 - 若 w 为黑色？不可能！若存在图中虚线边， w 变黑前一定已经搜索过 v



算法性质



- 边的性质
 - 对于无向图，非树边一定是后向边

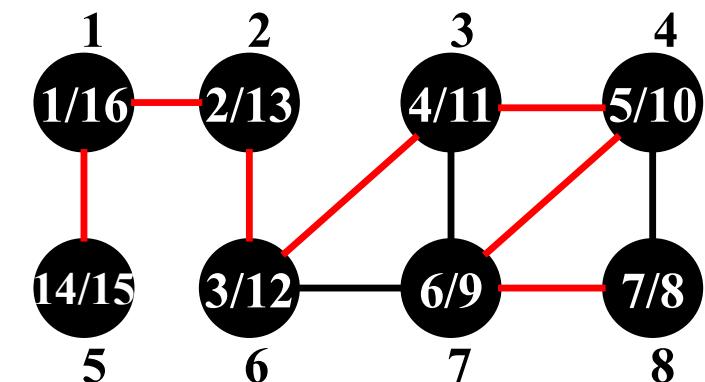
点的性质



- 括号化定理

- 点 v 发现时刻和结束时刻构成区间 $[d[v], f[v]]$

1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	5	
6	3	4	5	6	7	8	9	10	11	12						
3	4	5	6	7	8	9	10	11								
4	5	6	7	8	9	10										
7	6	7	8	9												
8	7	8														

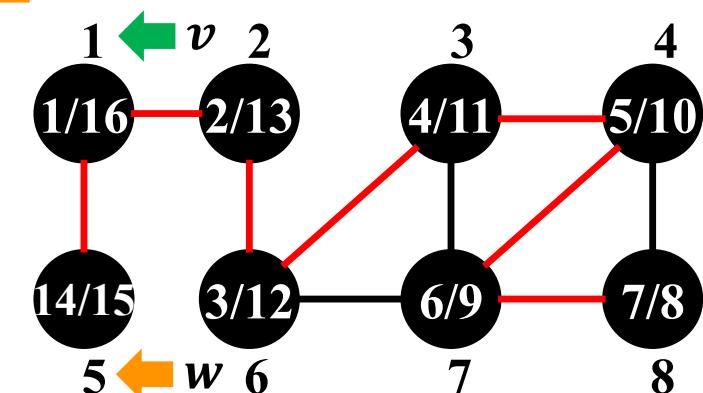
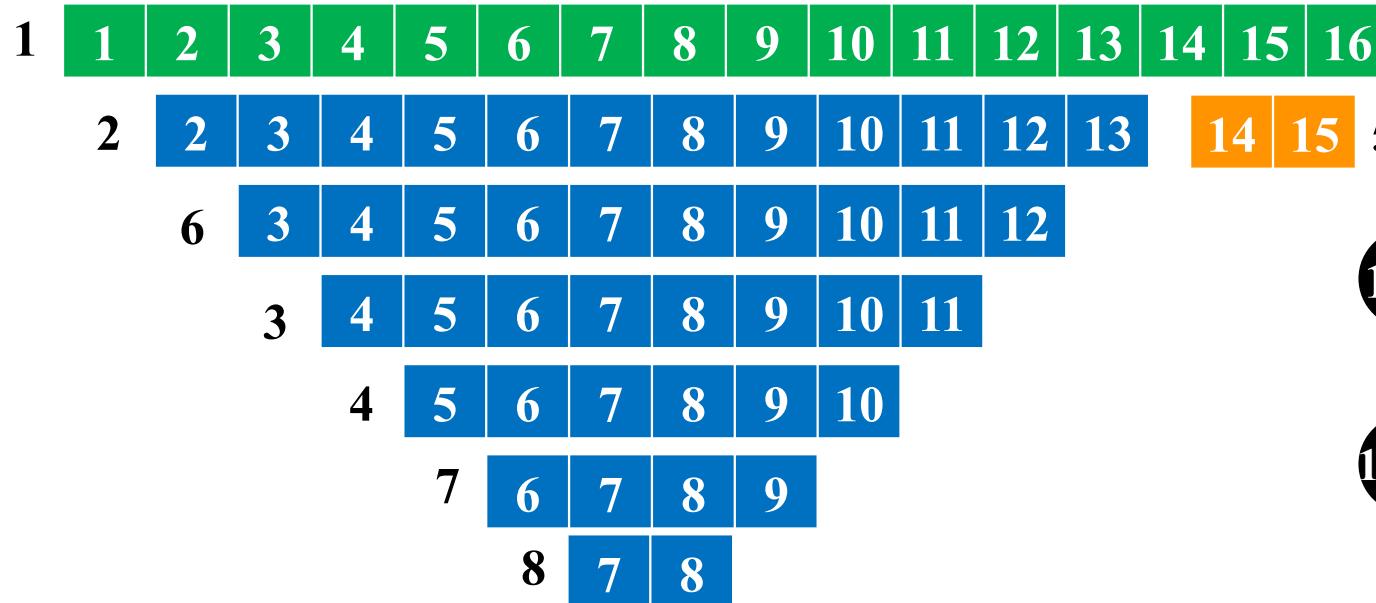


点的性质



• 括号化定理

- 点 v 发现时刻和结束时刻构成区间 $[d[v], f[v]]$
- 任意两点 v, w 必满足以下情况之一：
 - $[d[v], f[v]]$ 包含 $[d[w], f[w]]$, w 是 v 的后代

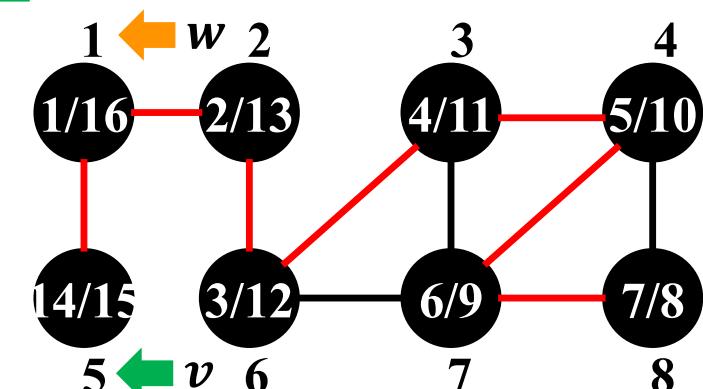
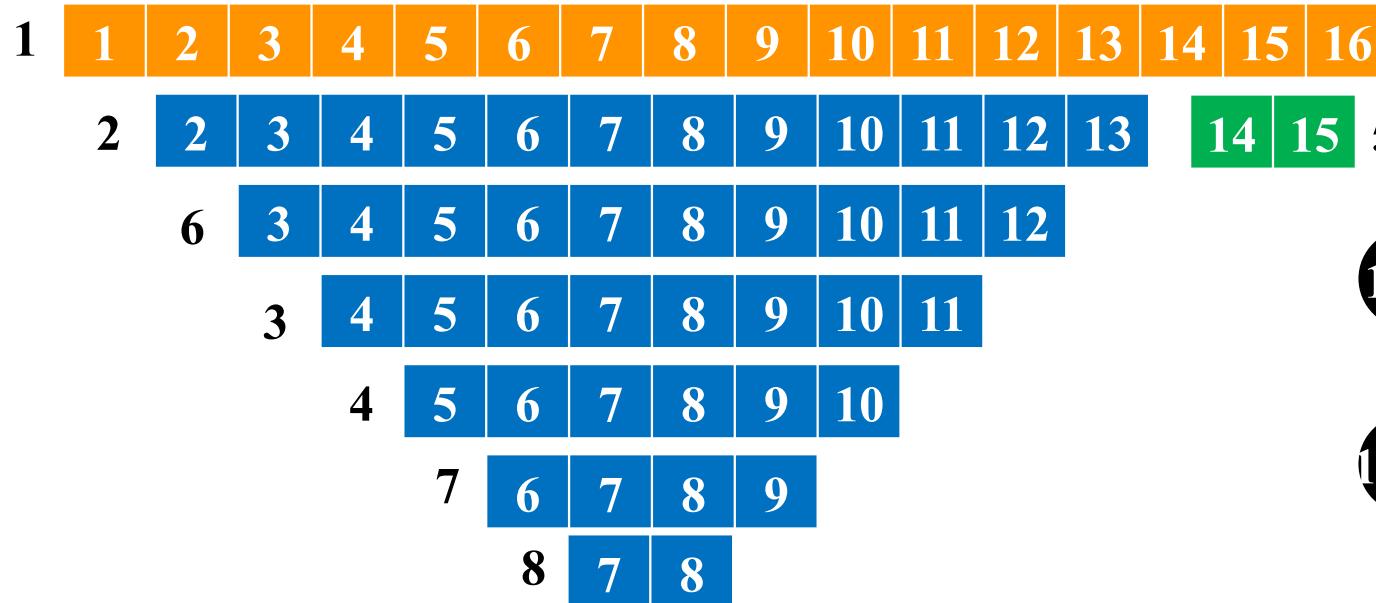


点的性质



• 括号化定理

- 点 v 发现时刻和结束时刻构成区间 $[d[v], f[v]]$
- 任意两点 v, w 必满足以下情况之一：
 - $[d[v], f[v]]$ 包含 $[d[w], f[w]]$, w 是 v 的后代
 - $[d[w], f[w]]$ 包含 $[d[v], f[v]]$, v 是 w 的后代

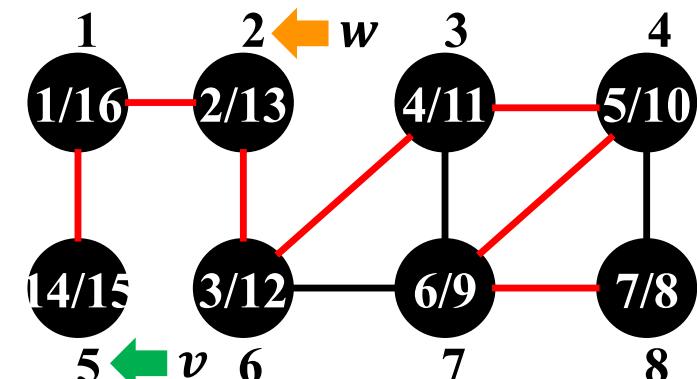
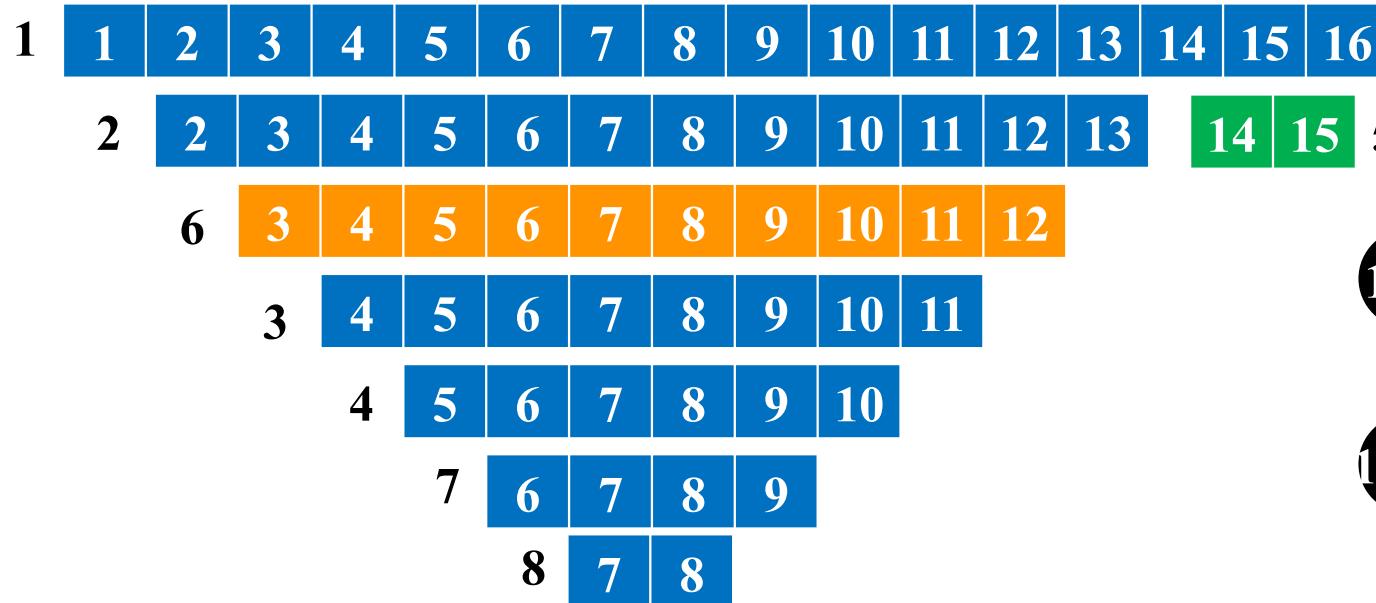


点的性质



• 括号化定理

- 点 v 发现时刻和结束时刻构成区间 $[d[v], f[v]]$
- 任意两点 v, w 必满足以下情况之一：
 - $[d[v], f[v]]$ 包含 $[d[w], f[w]]$, w 是 v 的后代
 - $[d[w], f[w]]$ 包含 $[d[v], f[v]]$, v 是 w 的后代
 - $[d[v], f[v]]$ 和 $[d[w], f[w]]$ 完全不重合, v 和 w 均不是对方后代





点的性质

- 括号化定理

- 证明

观察1：仅在区间内为灰色

```
输入: 图 $G$ , 顶点 $v$ 
color[ $v$ ]  $\leftarrow$  GRAY
time  $\leftarrow$  time + 1
d[ $v$ ]  $\leftarrow$  time
for  $w \in G.Adj[v]$  do
    if color[ $w$ ] = WHITE then
        pred[ $w$ ]  $\leftarrow v$ 
        DFS-Visit( $G, w$ )
    end
end
color[ $v$ ]  $\leftarrow$  BLACK
time  $\leftarrow$  time + 1
f[ $v$ ]  $\leftarrow$  time
```

$d[v]$: 变灰时刻

$f[v]$: 变黑时刻



$d[v]$

$f[v]$



点的性质

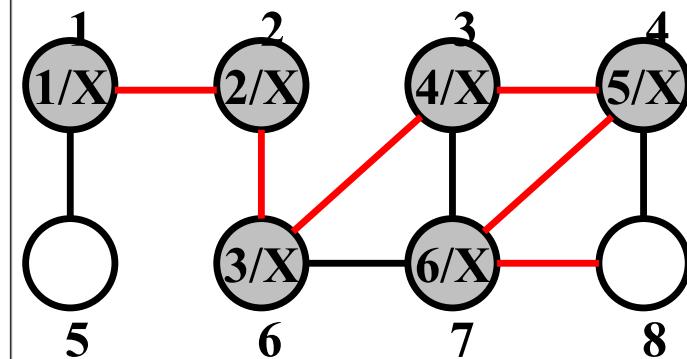
- 括号化定理

- 证明

观察1：仅在区间内为灰色

观察2：白点必是灰点后代

```
输入: 图G, 顶点v
color[v] ← GRAY
time ← time + 1
d[v] ← time
for w ∈ G.Adj[v] do
    if color[w] = WHITE then
        pred[w] ← v
        DFS-Visit(G, w)
    end
end
color[v] ← BLACK
time ← time + 1
f[v] ← time
```



$d[v]$

$f[v]$

点的性质



- 括号化定理

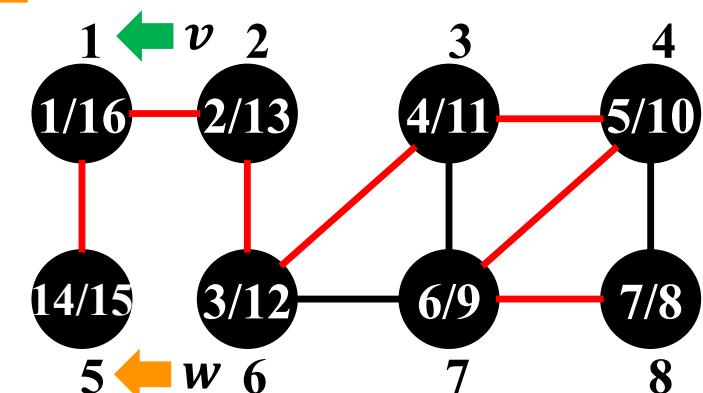
- 证明(不妨设 $d[v] < d[w]$)

观察1：仅在区间内为灰色

观察2：白点必是灰点后代

- 若 $f[v] > d[w]$:

1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	5	
6	3	4	5	6	7	8	9	10	11	12						
3	4	5	6	7	8	9	10	11								
4	5	6	7	8	9	10										
7	6	7	8	9												
8	7	8														



点的性质



- 括号化定理

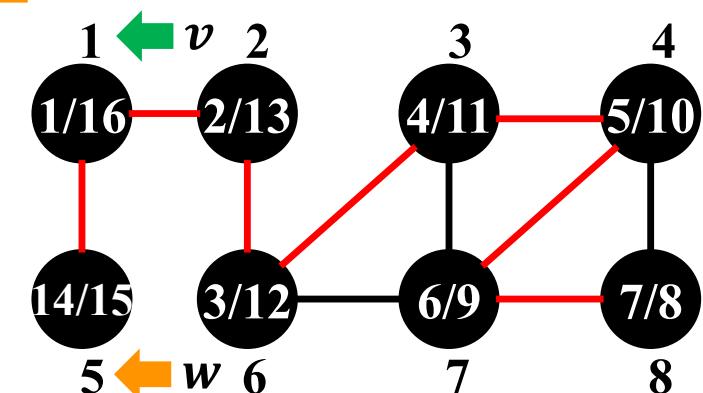
- 证明(不妨设 $d[v] < d[w]$)

观察1：仅在区间内为灰色

观察2：白点必是灰点后代

- 若 $f[v] > d[w]$: w 被发现时, v 为灰色,

1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	5	
6	3	4	5	6	7	8	9	10	11	12						
3	4	5	6	7	8	9	10	11								
4	5	6	7	8	9	10										
7	6	7	8	9												
8	7	8														



点的性质



• 括号化定理

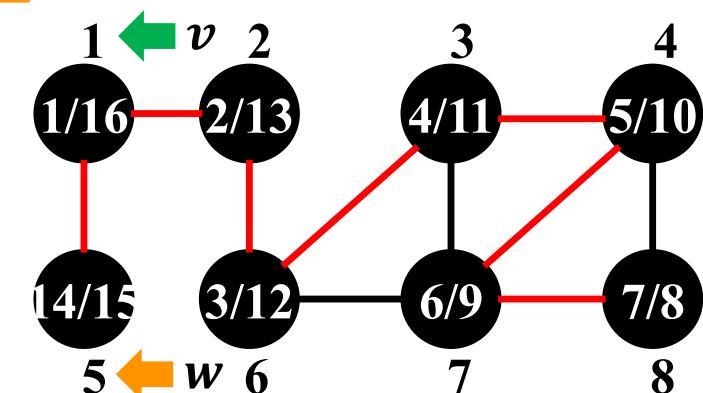
- 证明(不妨设 $d[v] < d[w]$)

观察1：仅在区间内为灰色

观察2：白点必是灰点后代

- 若 $f[v] > d[w]$: w 被发现时, v 为灰色, 所以 w 是 v 的后代

1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	5	
6	3	4	5	6	7	8	9	10	11	12						
3	4	5	6	7	8	9	10	11								
4	5	6	7	8	9	10										
7	6	7	8	9												
8	7	8														





点的性质

- 括号化定理

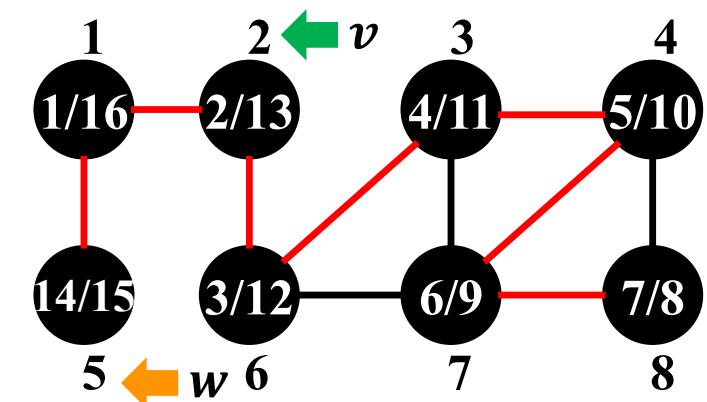
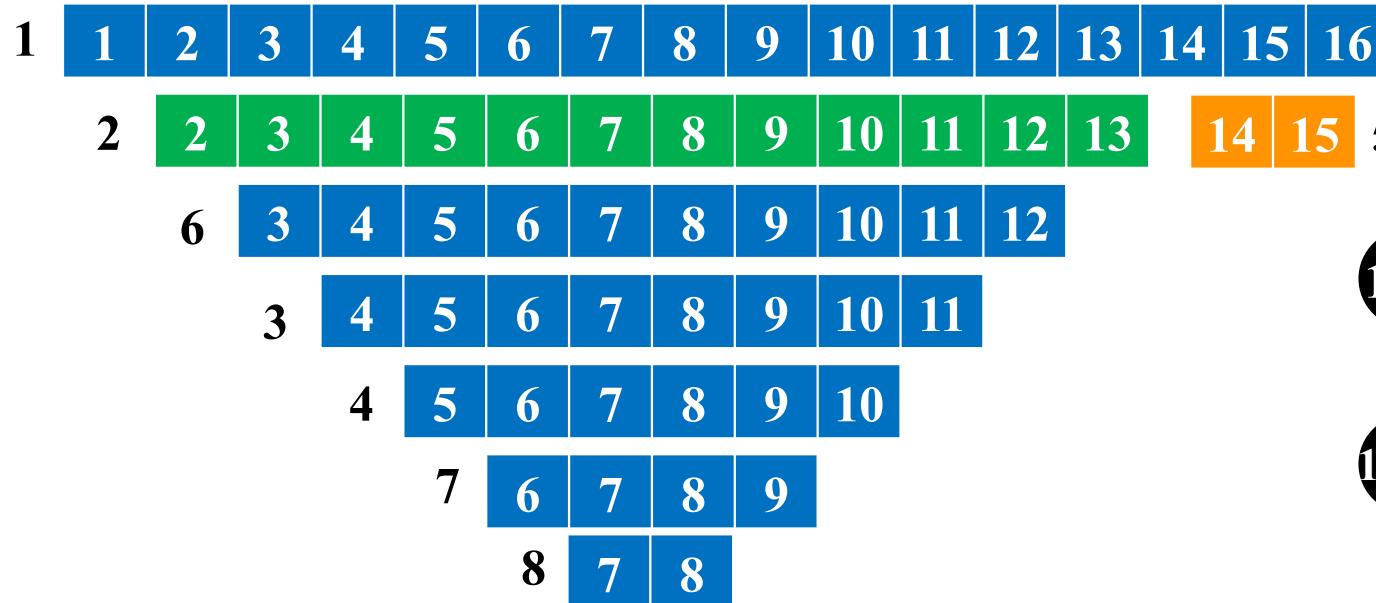
- 证明(不妨设 $d[v] < d[w]$)

观察1：仅在区间内为灰色

观察2：白点必是灰点后代

◦ 若 $f[v] > d[w]$: w 被发现时， v 为灰色，所以 w 是 v 的后代

◦ 若 $f[v] < d[w]$:



点的性质



• 括号化定理

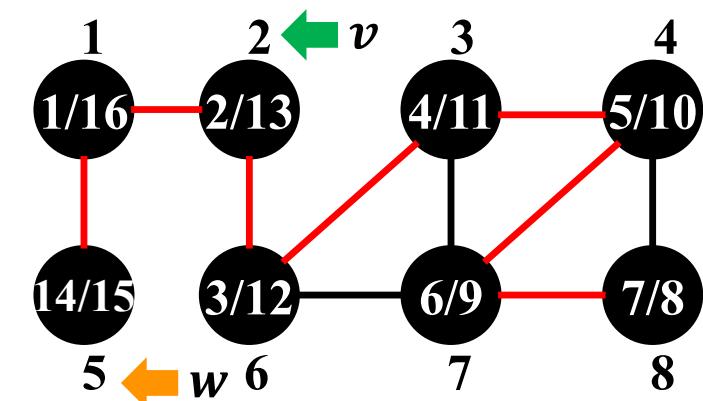
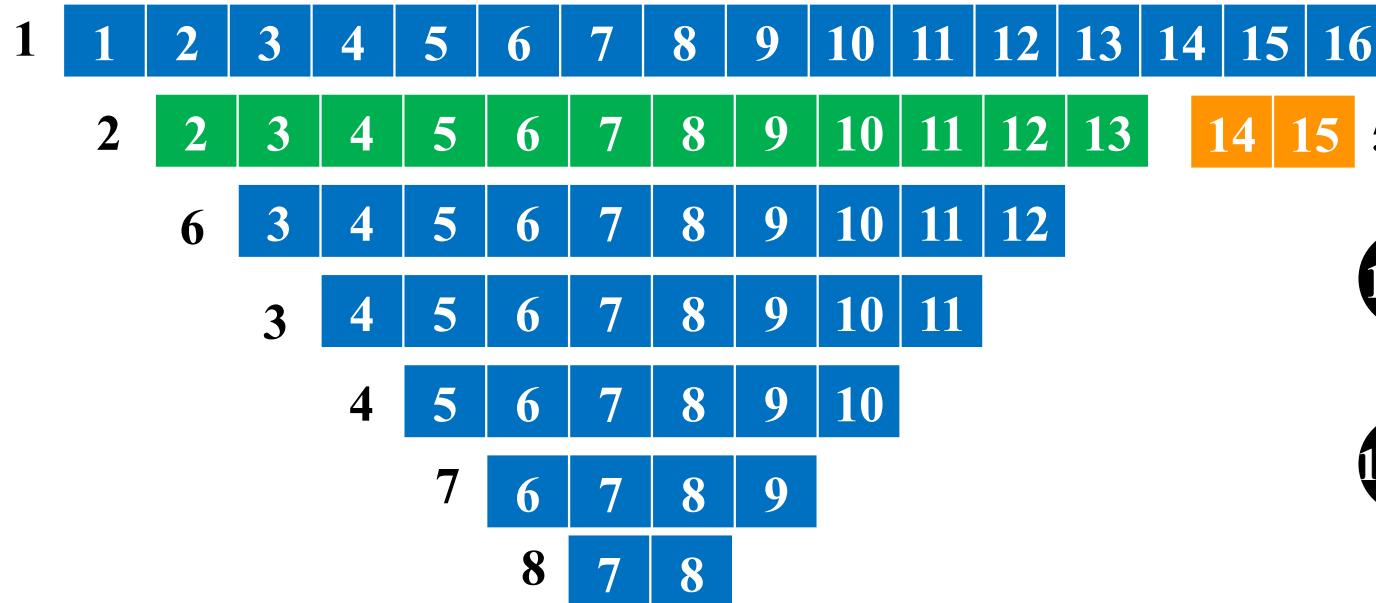
- 证明(不妨设 $d[v] < d[w]$)

观察1：仅在区间内为灰色

观察2：白点必是灰点后代

◦ 若 $f[v] > d[w]$: w 被发现时, v 为灰色, 所以 w 是 v 的后代

◦ 若 $f[v] < d[w]$: w 被发现时, v 为黑色,





点的性质

• 括号化定理

- 证明(不妨设 $d[v] < d[w]$)

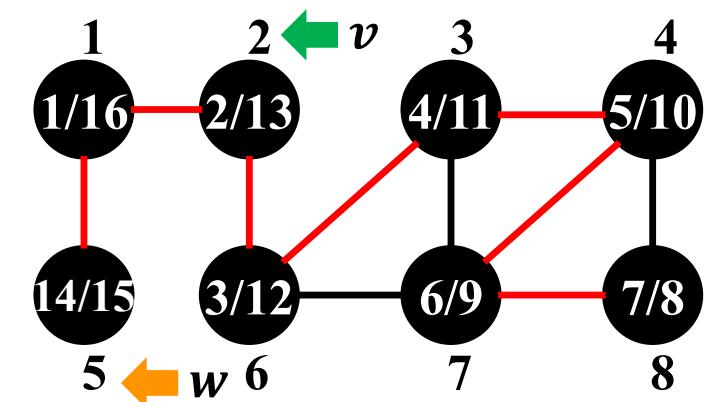
观察1：仅在区间内为灰色

观察2：白点必是灰点后代

◦ 若 $f[v] > d[w]$: w 被发现时, v 为灰色, 所以 w 是 v 的后代

◦ 若 $f[v] < d[w]$: w 被发现时, v 为黑色, 所以 v 和 w 均不是对方后代

1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	5	
6	3	4	5	6	7	8	9	10	11	12						
3	4	5	6	7	8	9	10	11								
4	5	6	7	8	9	10										
7	6	7	8	9												
8	7	8														





算法性质

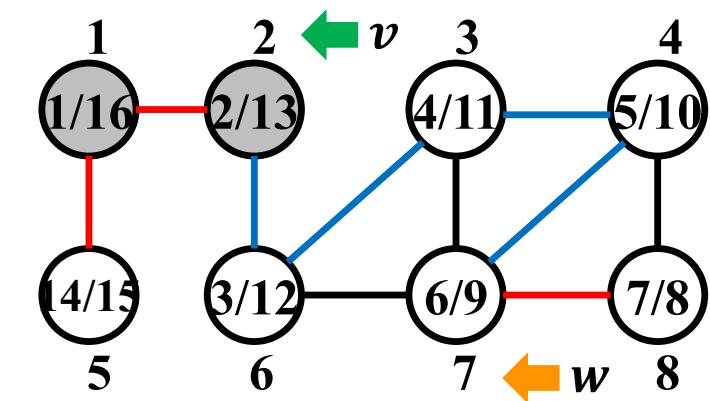
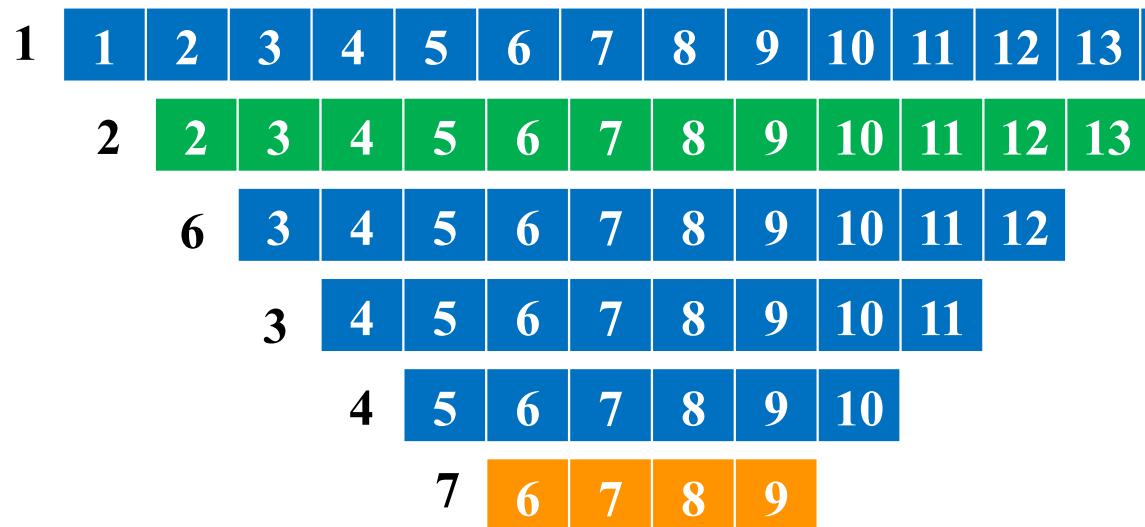
- 边的性质
 - 对于无向图，非树边一定是后向边
- 点的性质（括号定理）
 - 任意两点 v, w 必满足以下情况之一：
 - $[d[v], f[v]]$ 包含 $[d[w], f[w]]$, w 是 v 的后代
 - $[d[w], f[w]]$ 包含 $[d[v], f[v]]$, v 是 w 的后代
 - $[d[v], f[v]]$ 和 $[d[w], f[w]]$ 完全不重合, v 和 w 均不是对方后代

路径性质



- 白色路径定理

- 在深度优先树中，顶点 v 是 w 的祖先 \Leftrightarrow 在 v 被发现前，从 v 到 w 存在全为白色顶点构成的路径

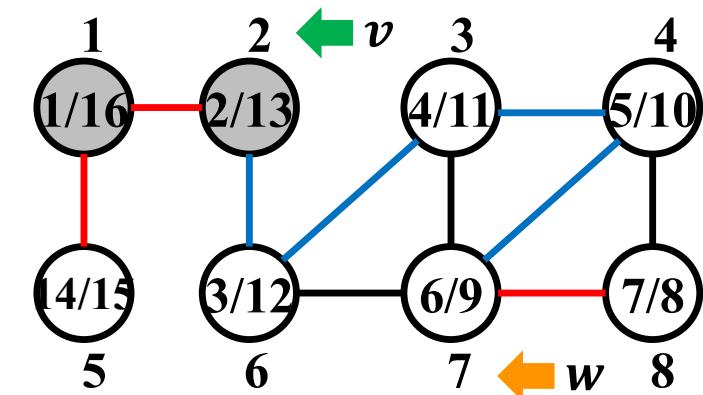
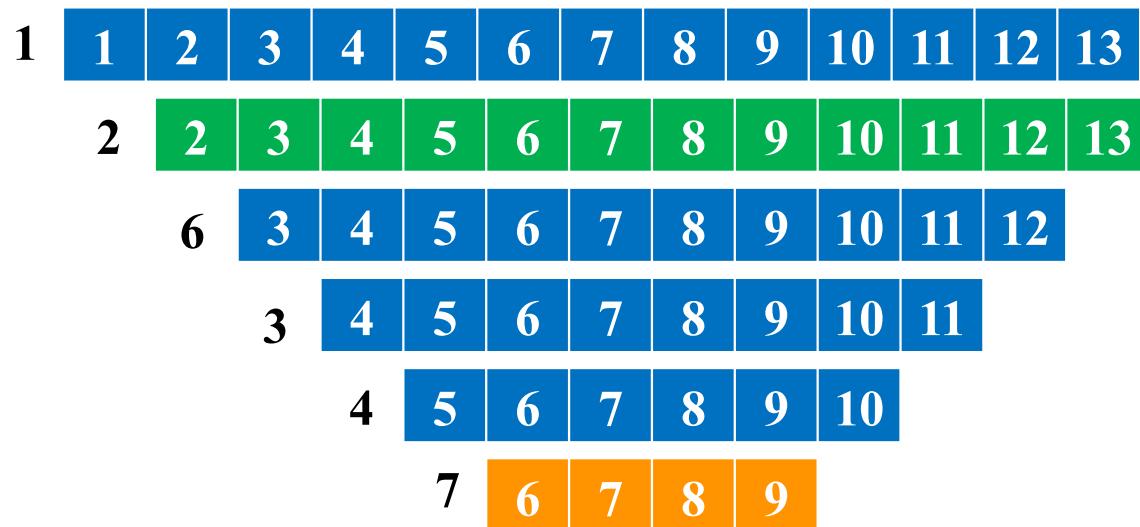


路径性质



白色路径定理

- 在深度优先树中，顶点 v 是 w 的祖先 \Leftrightarrow 在 v 被发现前，从 v 到 w 存在全为白色顶点构成的路径
- 证明（基本思想）
 \Rightarrow : 由括号化定理， v 在发现之前，其后代全为白色

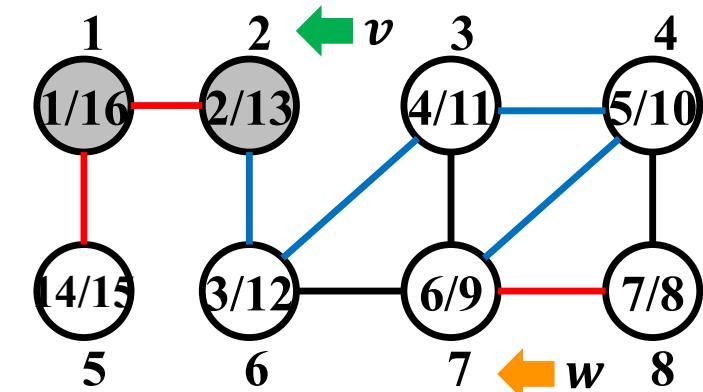
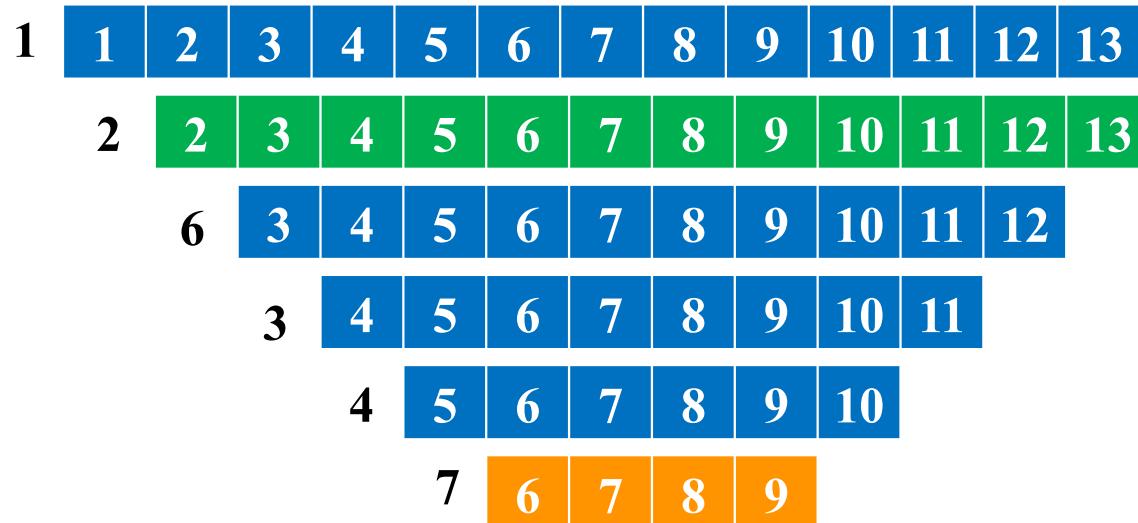


路径性质



白色路径定理

- 在深度优先树中，顶点 v 是 w 的祖先 \Leftrightarrow 在 v 被发现前，从 v 到 w 存在全为白色顶点构成的路径
- 证明（基本思想）
 - \Rightarrow : 由括号化定理， v 在发现之前，其后代全为白色
 - \Leftarrow : v 刚发现时，路径上除 v 以外的点仍都为白色，按深度优先搜索特点，一定会搜索路径上的所有点

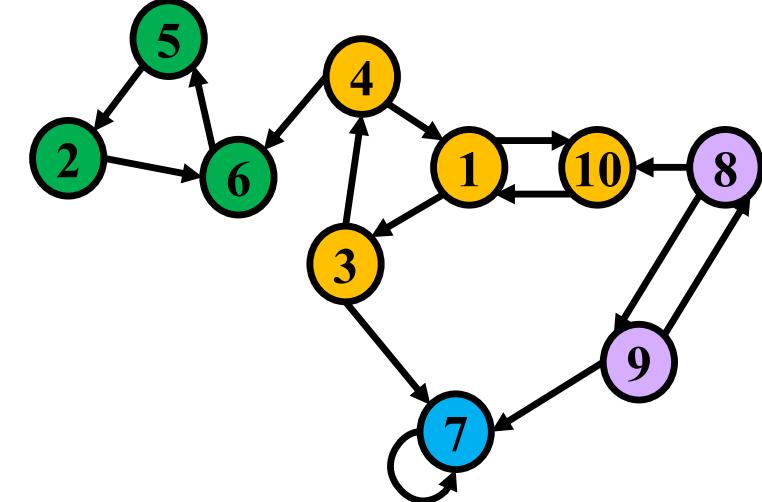
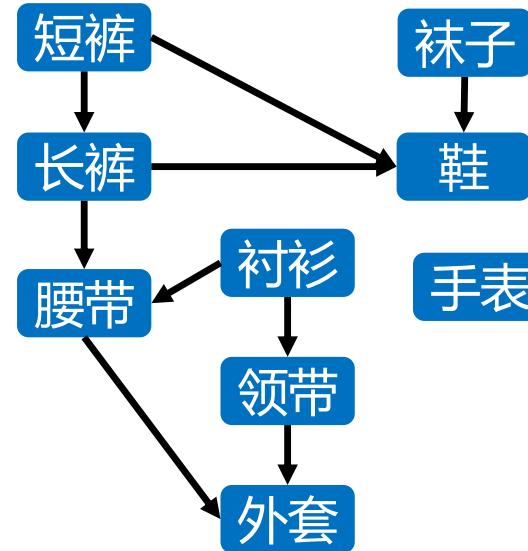
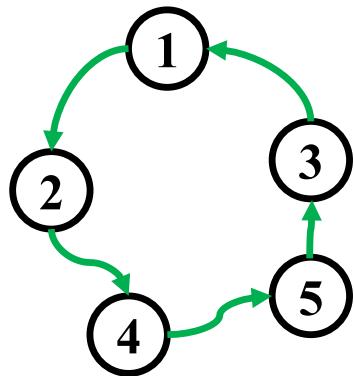




算法性质

- 边的性质
 - 对于无向图，非树边一定是后向边
- 点的性质（括号定理）
 - 任意两点 v, w 必满足以下情况之一：
 - $[d[v], f[v]]$ 包含 $[d[w], f[w]]$, w 是 v 的后代
 - $[d[w], f[w]]$ 包含 $[d[v], f[v]]$, v 是 w 的后代
 - $[d[v], f[v]]$ 和 $[d[w], f[w]]$ 完全不重合, v 和 w 均不是对方后代
- 白色路径定理
 - 在深度优先树中，顶点 v 是 w 的祖先 \Leftrightarrow 在 v 被发现前，从 v 到 w 存在全为白色顶点构成的路径

算法应用



环路的存在性判断

拓扑排序

强连通分量

图算法篇：有向图的深度优先搜索

北京航空航天大学
计算机学院



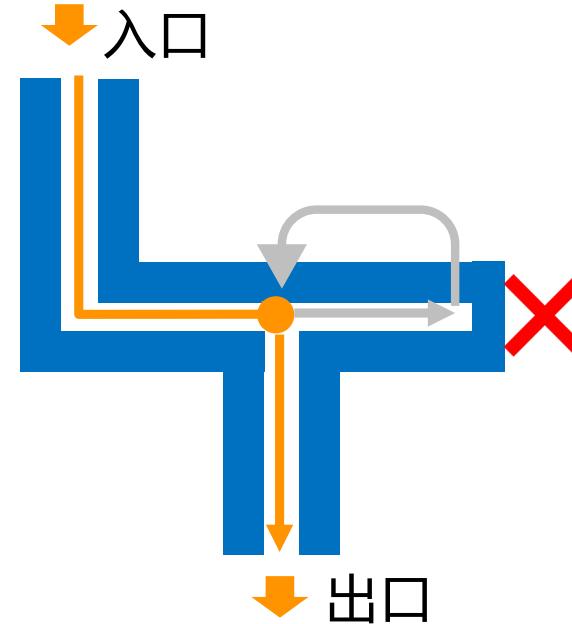
深度优先搜索回顾：算法思想

- 算法步骤

- 分叉时，任选一条边深入
- 无边时，后退一步找新边
- 找到边，从新边继续深入

- 辅助数组

- *color*: 表示顶点状态
 - *White*: 白色顶点尚未被发现
 - *Black*: 黑色顶点已被处理
 - *Gray*: 正在处理，尚未完成
- *pred*: 顶点 u 由*pred*[u]发现
- *d*: 顶点发现时刻（变成灰色的时刻）
- *f*: 顶点完成时刻（变成黑色的时刻）





伪代码

- **DFS(G)**

输入: 图 G

输出: 祖先数组 $pred$, 发现时刻 d , 结束时刻 f

新建数组 $color[1..V], pred[1..V], d[1..V], f[1..V]$

新建数组

//初始化

for $v \in V$ do

$pred[v] \leftarrow NULL$

$color[v] \leftarrow WHITE$

end

$time \leftarrow 0$

for $v \in V$ do

 if $color[v] = WHITE$ then

 DFS-Visit(G, v)

 end

end

return $pred, d, f$

$d[i], f[i]$ 分别记录顶点 i 的发现时刻与结束时刻



伪代码

- **DFS(G)**

输入: 图 G

输出: 祖先数组 $pred$, 发现时刻 d , 结束时刻 f

新建数组 $color[1..V], pred[1..V], d[1..V], f[1..V]$

//初始化

for $v \in V$ do

| $pred[v] \leftarrow NULL$

| $color[v] \leftarrow WHITE$

end

$time \leftarrow 0$

for $v \in V$ do

| if $color[v] = WHITE$ then

| | DFS-Visit(G, v)

| end

end

return $pred, d, f$

初始化



伪代码

- **DFS(G)**

输入: 图 G

输出: 祖先数组 $pred$, 发现时刻 d , 结束时刻 f

新建数组 $color[1..V], pred[1..V], d[1..V], f[1..V]$

//初始化

for $v \in V$ do

| $pred[v] \leftarrow NULL$

| $color[v] \leftarrow WHITE$

end

$time \leftarrow 0$

for $v \in V$ do

| if $color[v] = WHITE$ then

| | DFS-Visit(G, v)

| end

end

return $pred, d, f$

保证搜索完全



伪代码

- **DFS-Visit(G, v)**

输入: 图 G , 顶点 v

```
color[v] ← GRAY  
time ← time + 1  
d[v] ← time  
for w ∈ G.Adj[v] do  
    if color[w] = WHITE then  
        pred[w] ← v  
        DFS-Visit(G, w)  
    end  
end  
color[v] ← BLACK  
time ← time + 1  
f[v] ← time
```

修改当前顶点颜色、发现时刻



伪代码

- **DFS-Visit(G, v)**

输入: 图 G , 顶点 v

$color[v] \leftarrow GRAY$

$time \leftarrow time + 1$

$d[v] \leftarrow time$

for $w \in G.Adj[v]$ do

 if $color[w] = WHITE$ then

$pred[w] \leftarrow v$

 DFS-Visit(G, w)

 end

end

$color[v] \leftarrow BLACK$

$time \leftarrow time + 1$

$f[v] \leftarrow time$

搜索相邻顶点



伪代码

- **DFS-Visit(G, v)**

输入: 图 G , 顶点 v

$color[v] \leftarrow GRAY$

$time \leftarrow time + 1$

$d[v] \leftarrow time$

for $w \in G.Adj[v]$ do

 if $color[w] = WHITE$ then

$pred[w] \leftarrow v$

 DFS-Visit(G, w)

 end

end

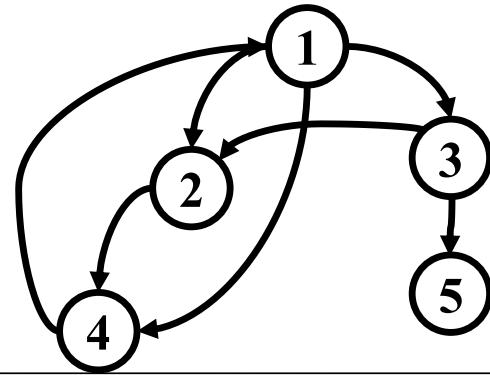
color[v] $\leftarrow BLACK$

$time \leftarrow time + 1$

$f[v] \leftarrow time$

结束搜索

深度优先搜索回顾：有向图算法实例



```
color[v] ← GRAY
time ← time + 1
d[v] ← time
for  $w \in Adj[v]$  do
    if color[w] = WHITE then
        pred[w] ← v
        DFS-Visit(w)
    end
end
color[v] ← BLACK
time ← time + 1
f[v] ← time
```

$time = 0$

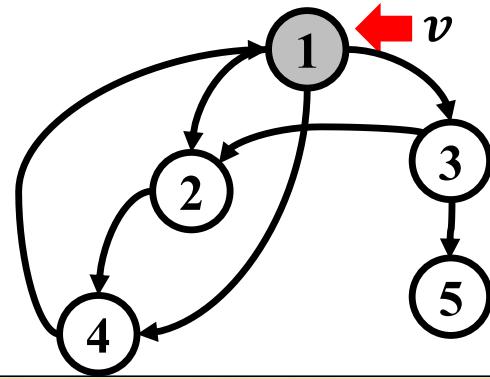
v	1	2	3	4	5
$pred$	N	N	N	N	N
v	1	2	3	4	5

v	1	2	3	4	5
color	W	W	W	W	W
v	1	2	3	4	5

v	1	2	3	4	5
d					
v	1	2	3	4	5

v	1	2	3	4	5
f					
v	1	2	3	4	5

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 0$

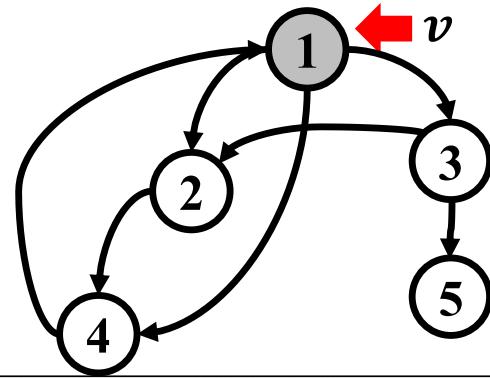
v	1	2	3	4	5
$pred$	N	N	N	N	N

v	1	2	3	4	5
$color$	G	W	W	W	W

v	1	2	3	4	5
d					

v	1	2	3	4	5
f					

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 1$

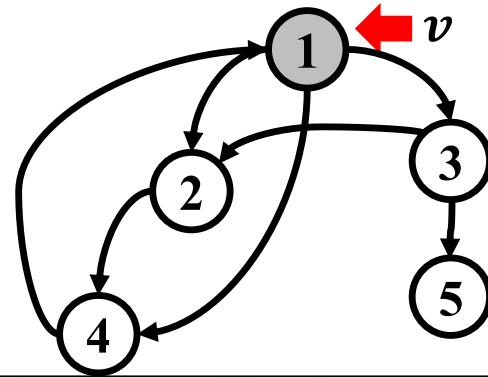
v	1	2	3	4	5
$pred$	N	N	N	N	N
v	1	2	3	4	5

v	1	2	3	4	5
$color$	G	W	W	W	W
v	1	2	3	4	5

v	1	2	3	4	5
d					
v	1	2	3	4	5

v	1	2	3	4	5
f					
v	1	2	3	4	5

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 1$

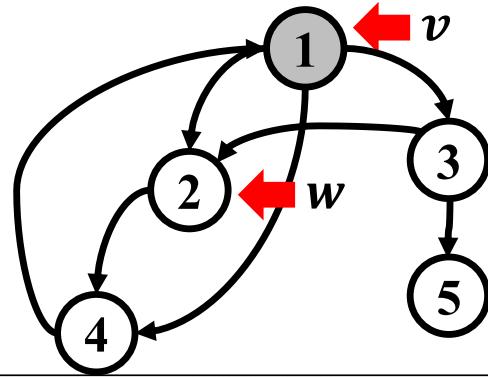
v	1	2	3	4	5
$pred$	N	N	N	N	N
v	1	2	3	4	5

v	1	2	3	4	5
$color$	G	W	W	W	W
v	1	2	3	4	5

v	1	2	3	4	5
d	1				
v	1	2	3	4	5

v	1	2	3	4	5
f					
v	1	2	3	4	5

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 1$

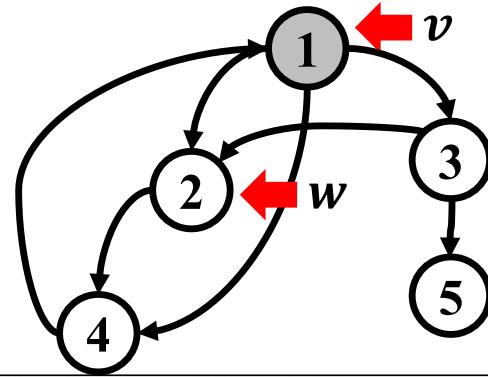
v	1	2	3	4	5
$pred$	N	N	N	N	N
v	1	2	3	4	5

v	1	2	3	4	5
$color$	G	W	W	W	W
v	1	2	3	4	5

v	1	2	3	4	5
d	1				
v	1	2	3	4	5

v	1	2	3	4	5
f					
v	1	2	3	4	5

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 1$

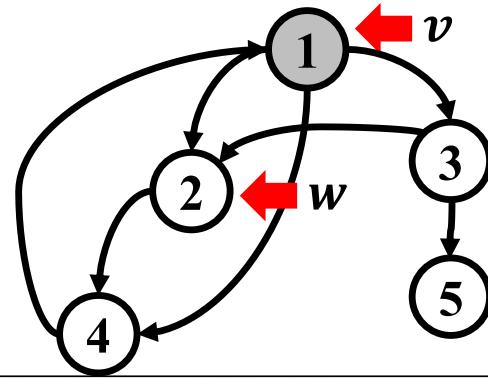
v	1	2	3	4	5
$pred$	N	N	N	N	N

v	1	2	3	4	5
$color$	G	W	W	W	W

v	1	2	3	4	5
d	1				

v	1	2	3	4	5
f					

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 1$

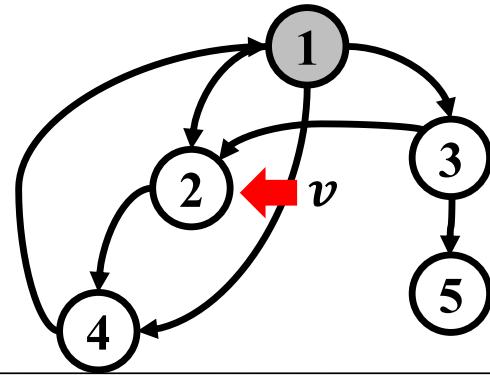
v	1	2	3	4	5
$pred$	N	1	N	N	N

v	1	2	3	4	5
$color$	G	W	W	W	W

v	1	2	3	4	5
d	1				

v	1	2	3	4	5
f					

深度优先搜索回顾：有向图算法实例



```
color[v] ← GRAY
time ← time + 1
d[v] ← time
for w ∈ Adj[v] do
    if color[w] = WHITE then
        pred[w] ← v
        DFS-Visit(w)
    end
end
color[v] ← BLACK
time ← time + 1
f[v] ← time
```

$time = 1$

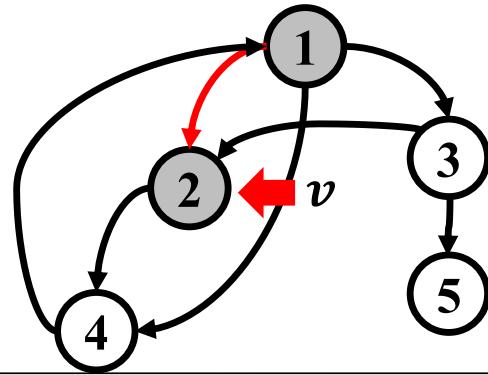
v	1	2	3	4	5
$pred$	N	1	N	N	N
v	1	2	3	4	5

v	1	2	3	4	5
color	G	W	W	W	W
v	1	2	3	4	5

v	1	2	3	4	5
d	1				
v	1	2	3	4	5

v	1	2	3	4	5
f					
v	1	2	3	4	5

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 1$

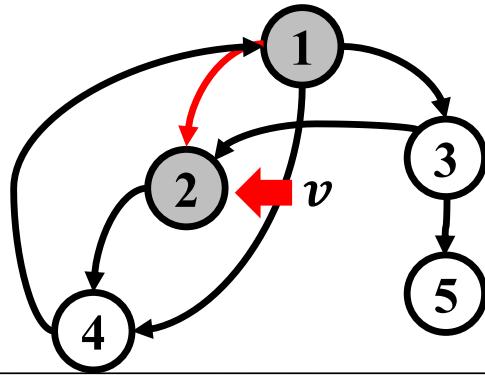
v	1	2	3	4	5
$pred$	N	1	N	N	N

v	1	2	3	4	5
$color$	G	G	W	W	W

v	1	2	3	4	5
d	1				

v	1	2	3	4	5
f					

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 

 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 2$

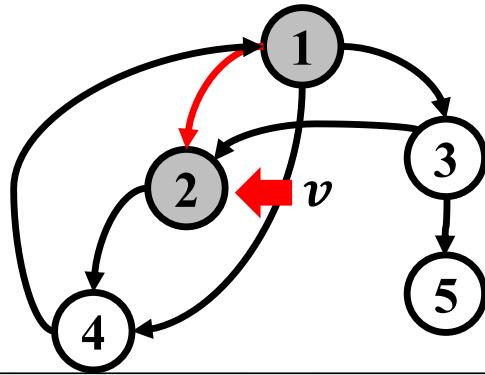
v	1	2	3	4	5
$pred$	N	1	N	N	N
v	1	2	3	4	5

v	1	2	3	4	5
$color$	G	G	W	W	W
v	1	2	3	4	5

v	1	2	3	4	5
d	1				
v	1	2	3	4	5

v	1	2	3	4	5
f					
v	1	2	3	4	5

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 2$

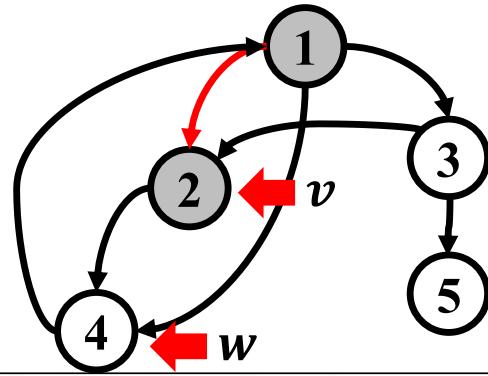
v	1	2	3	4	5
$pred$	N	1	N	N	N
v	1	2	3	4	5

v	1	2	3	4	5
$color$	G	G	W	W	W
v	1	2	3	4	5

v	1	2	3	4	5
d	1	2			
v	1	2	3	4	5

v	1	2	3	4	5
f					
v	1	2	3	4	5

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 2$

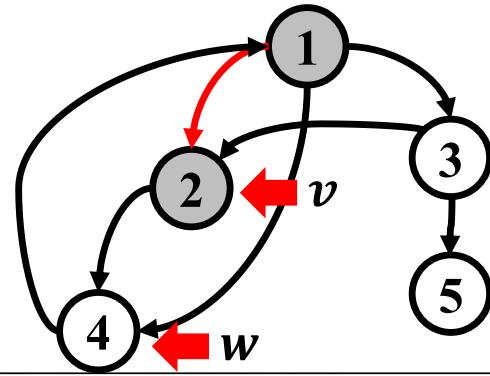
v	1	2	3	4	5
$pred$	N	1	N	N	N

v	1	2	3	4	5
$color$	G	G	W	W	W

v	1	2	3	4	5
d	1	2			

v	1	2	3	4	5
f					

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 2$

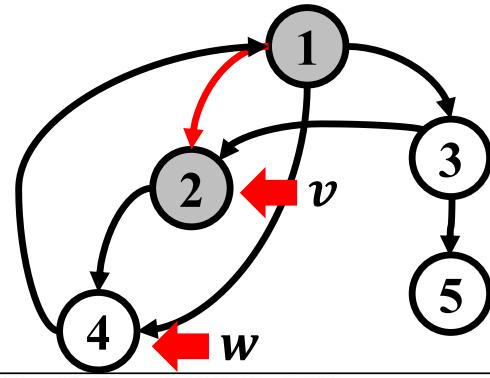
v	1	2	3	4	5
$pred$	N	1	N	N	N

v	1	2	3	4	5
$color$	G	G	[W]	W	W

v	1	2	3	4	5
d	1	2			

v	1	2	3	4	5
f					

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 2$

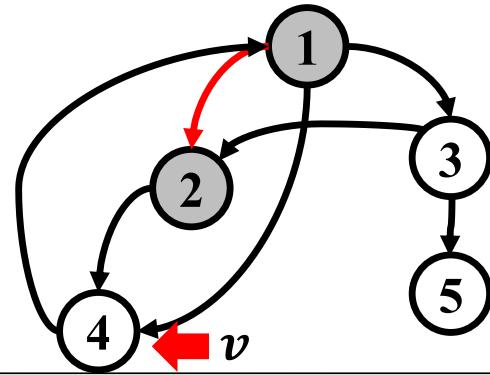
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	G	W	W	W

v	1	2	3	4	5
d	1	2			

v	1	2	3	4	5
f					

深度优先搜索回顾：有向图算法实例



```
color[v] ← GRAY
time ← time + 1
d[v] ← time
for w ∈ Adj[v] do
    if color[w] = WHITE then
        pred[w] ← v
        DFS-Visit(w)
    end
end
color[v] ← BLACK
time ← time + 1
f[v] ← time
```

$time = 2$

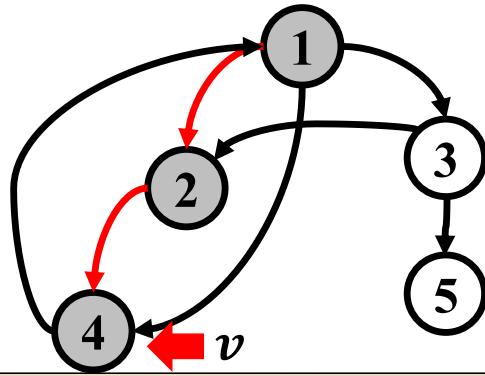
v	1	2	3	4	5
$pred$	N	1	N	2	N
v	1	2	3	4	5

v	1	2	3	4	5
color	G	G	W	W	W
v	1	2	3	4	5

v	1	2	3	4	5
d	1	2			
v	1	2	3	4	5

v	1	2	3	4	5
f					
v	1	2	3	4	5

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 2$

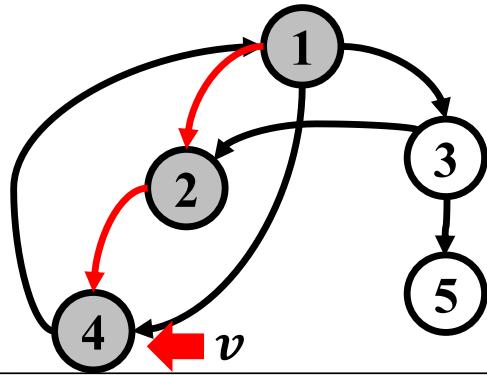
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	G	W	G	W

v	1	2	3	4	5
d	1	2			

v	1	2	3	4	5
f					

深度优先搜索回顾：有向图算法实例



```
color[v] ← GRAY
time ← time + 1
d[v] ← time
for w ∈ Adj[v] do
    if color[w] = WHITE then
        pred[w] ← v
        DFS-Visit(w)
    end
end
color[v] ← BLACK
time ← time + 1
f[v] ← time
```

time = 3

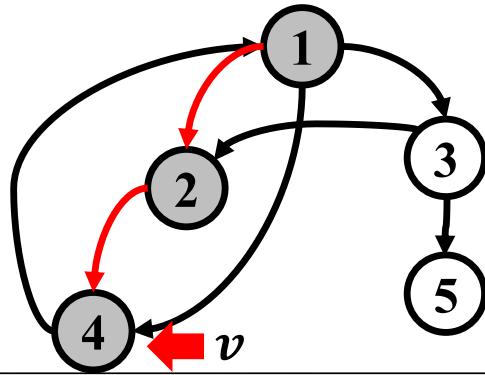
<i>v</i>	1	2	3	4	5
<i>pred</i>	N	1	N	2	N
<i>v</i>	1	2	3	4	5

<i>v</i>	1	2	3	4	5
color	G	G	W	G	W
<i>v</i>	1	2	3	4	5

<i>v</i>	1	2	3	4	5
<i>d</i>	1	2			
<i>v</i>	1	2	3	4	5

<i>v</i>	1	2	3	4	5
<i>f</i>					
<i>v</i>	1	2	3	4	5

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 3$

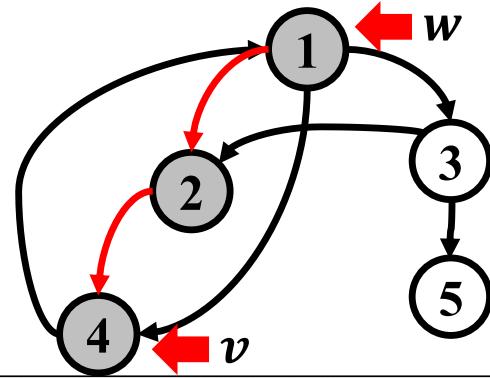
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	G	W	G	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f					

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 3$

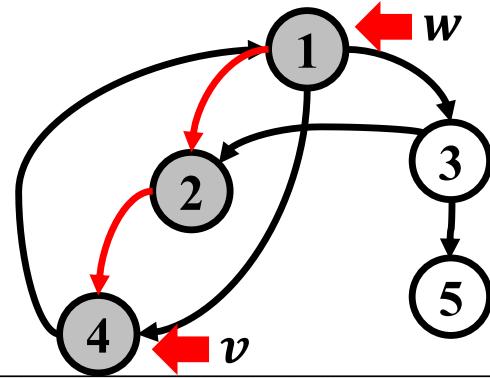
v	1	2	3	4	5
$pred$	N	1	N	2	N
v	1	2	3	4	5

v	1	2	3	4	5
$color$	G	G	W	G	W
v	1	2	3	4	5

v	1	2	3	4	5
d	1	2		3	
v	1	2	3	4	5

v	1	2	3	4	5
f					
v	1	2	3	4	5

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 3$

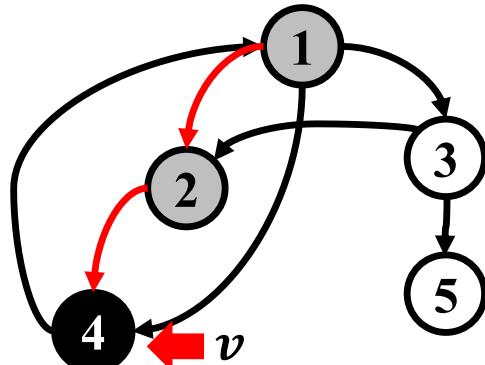
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	G	W	G	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f					

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 3$

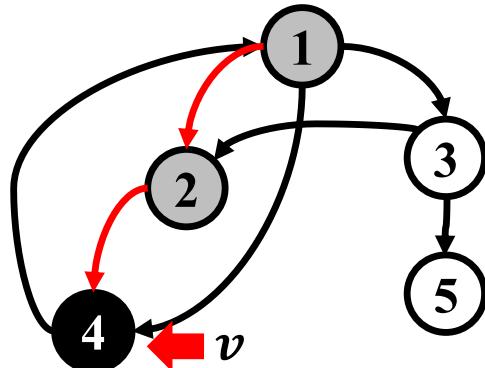
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	G	W	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f					

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 4$

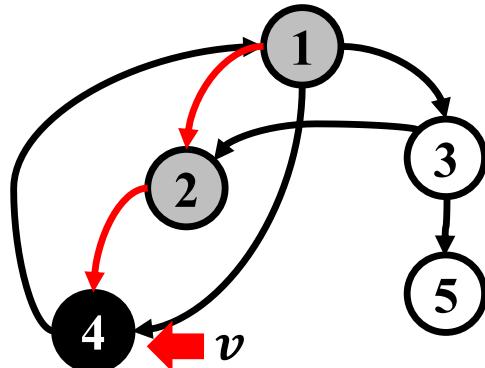
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	G	W	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f					

深度优先搜索回顾：有向图算法实例



```
color[v] ← GRAY  
time ← time + 1  
d[v] ← time  
for w ∈ Adj[v] do  
    if color[w] = WHITE then  
        pred[w] ← v  
        DFS-Visit(w)  
    end  
end  
color[v] ← BLACK  
time ← time + 1  
f[v] ← time
```

$time = 4$

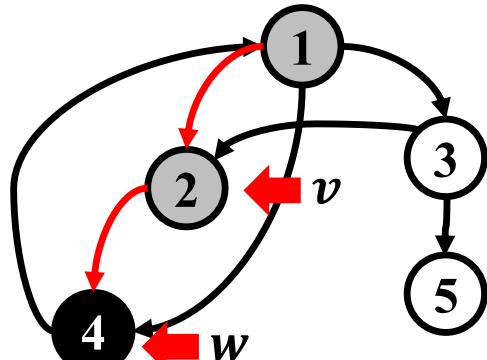
v	1	2	3	4	5
$pred$	N	1	N	2	N
v	1	2	3	4	5

v	1	2	3	4	5
color	G	G	W	B	W
v	1	2	3	4	5

v	1	2	3	4	5
d	1	2		3	
v	1	2	3	4	5

v	1	2	3	4	5
f				4	
v	1	2	3	4	5

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 4$

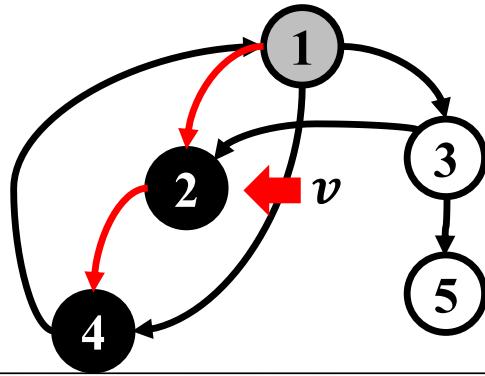
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	G	W	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f				4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 4$

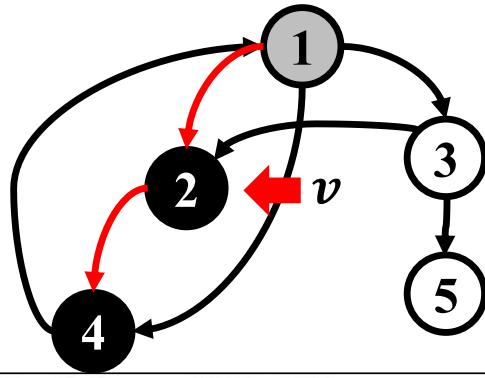
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	B	W	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f				4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 5$

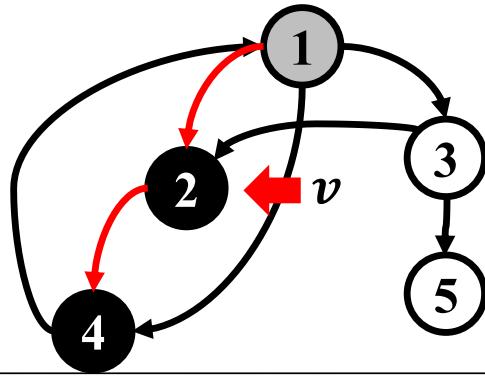
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	B	W	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f				4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 5$

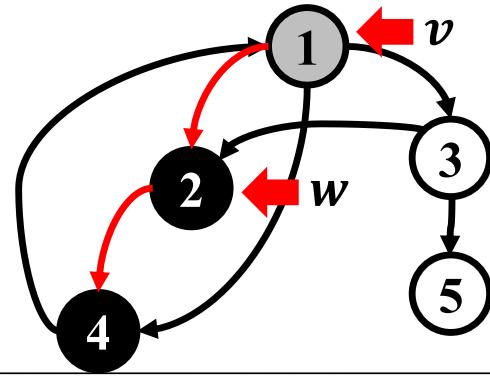
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	B	W	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 5$

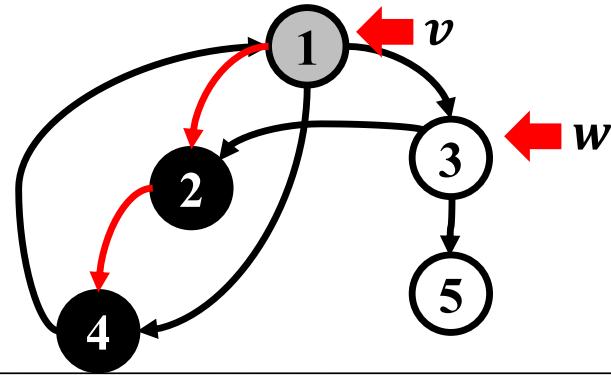
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	B	W	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 5$

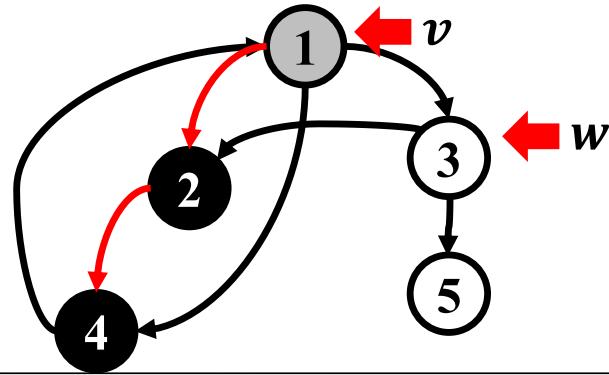
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	B	W	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 5$

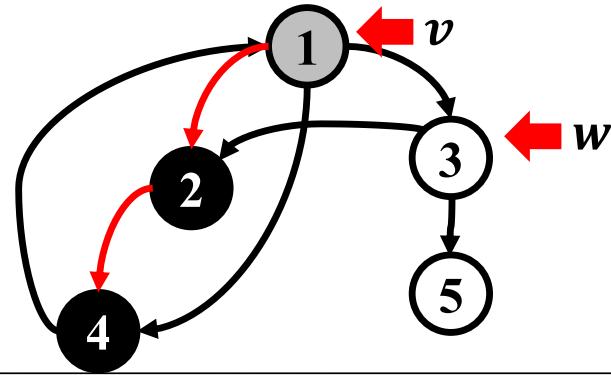
v	1	2	3	4	5
$pred$	N	1	N	2	N

v	1	2	3	4	5
$color$	G	B	W	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 5$

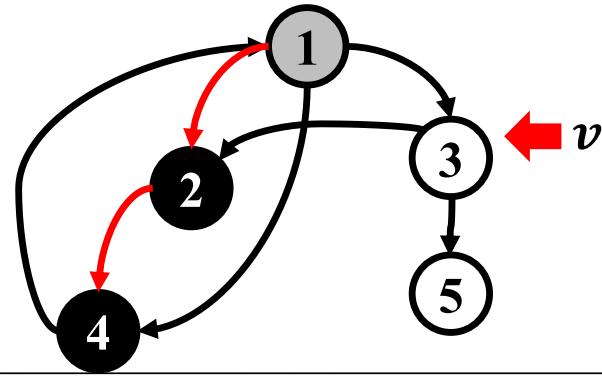
v	1	2	3	4	5
$pred$	N	1	1	2	N

v	1	2	3	4	5
$color$	G	B	W	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 5$

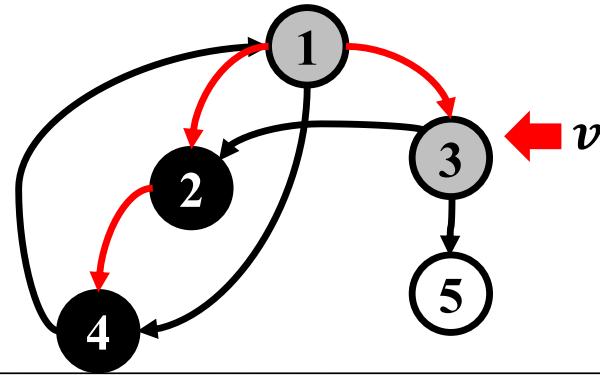
v	1	2	3	4	5
$pred$	N	1	1	2	N

v	1	2	3	4	5
$color$	G	B	W	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 5$

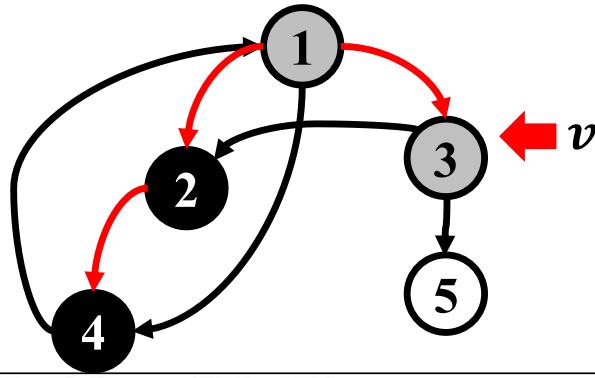
v	1	2	3	4	5
$pred$	N	1	1	2	N

v	1	2	3	4	5
$color$	G	B	G	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 6$

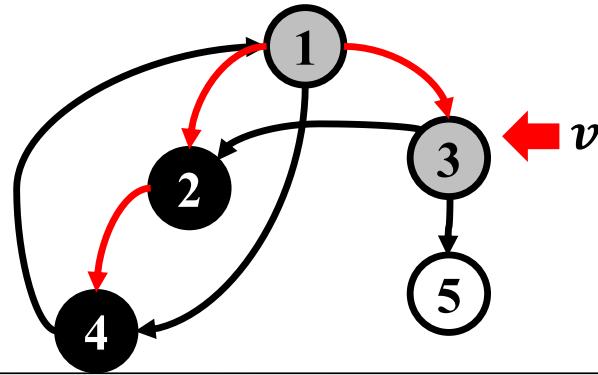
v	1	2	3	4	5
$pred$	N	1	1	2	N

v	1	2	3	4	5
$color$	G	B	G	B	W

v	1	2	3	4	5
d	1	2		3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 6$

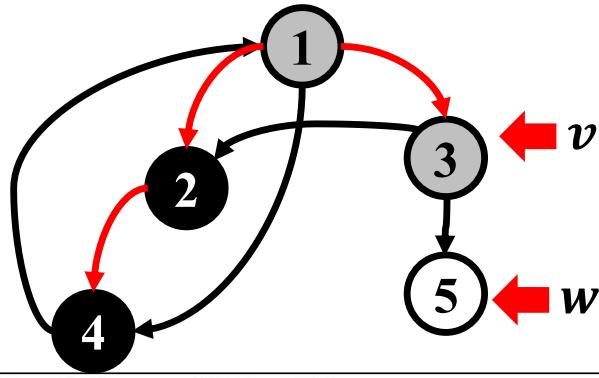
v	1	2	3	4	5
$pred$	N	1	1	2	N

v	1	2	3	4	5
$color$	G	B	G	B	W

v	1	2	3	4	5
d	1	2	6	3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 6$

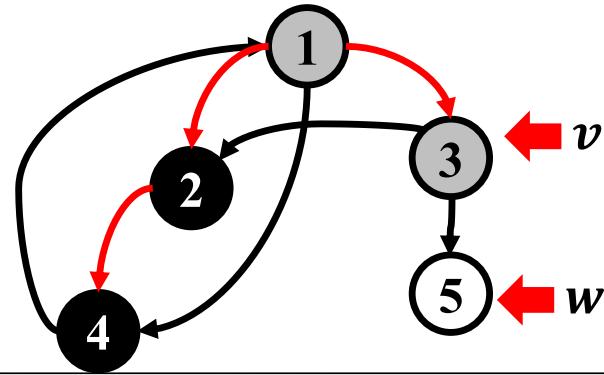
v	1	2	3	4	5
$pred$	N	1	1	2	N

v	1	2	3	4	5
$color$	G	B	G	B	W

v	1	2	3	4	5
d	1	2	6	3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 6$

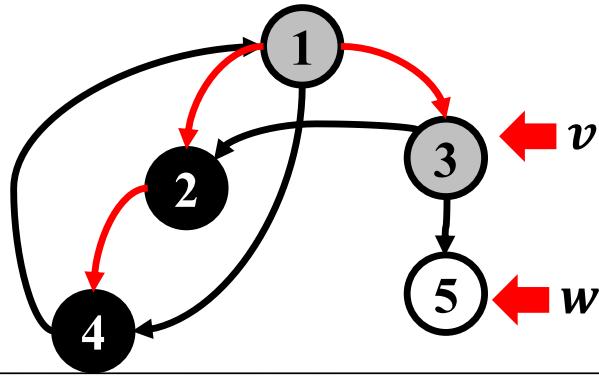
v	1	2	3	4	5
$pred$	N	1	1	2	N

v	1	2	3	4	5
$color$	G	B	G	B	W

v	1	2	3	4	5
d	1	2	6	3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 6$

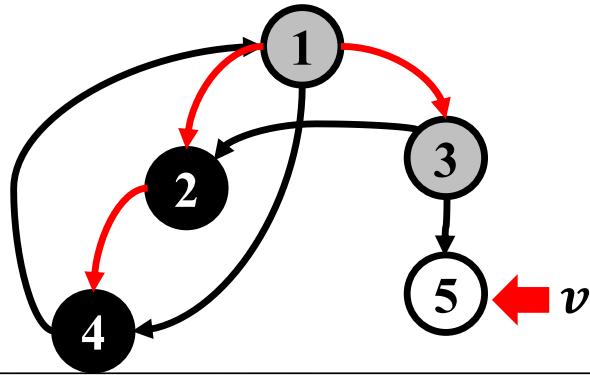
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	G	B	W

v	1	2	3	4	5
d	1	2	6	3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 6$

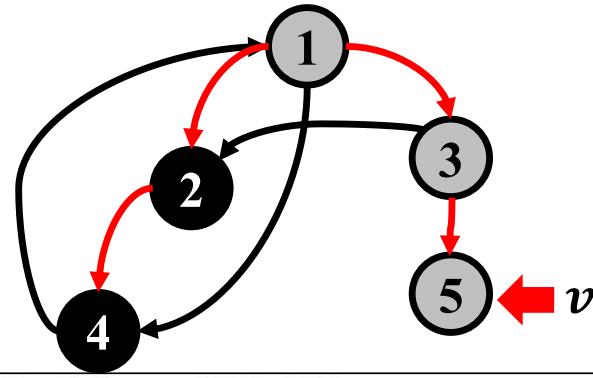
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	G	B	W

v	1	2	3	4	5
d	1	2	6	3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 6$

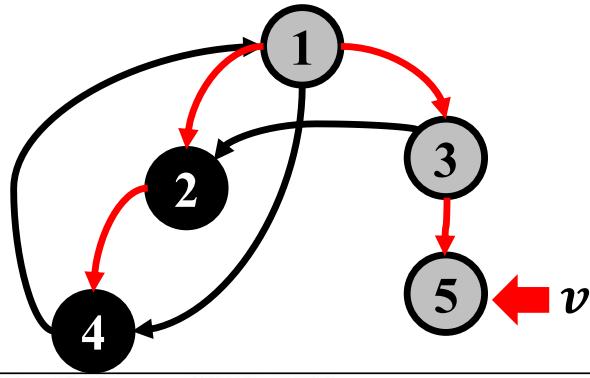
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	G	B	G

v	1	2	3	4	5
d	1	2	6	3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 7$

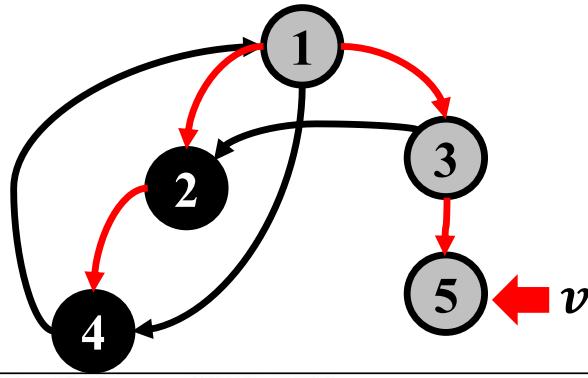
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	G	B	G

v	1	2	3	4	5
d	1	2	6	3	

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 7$

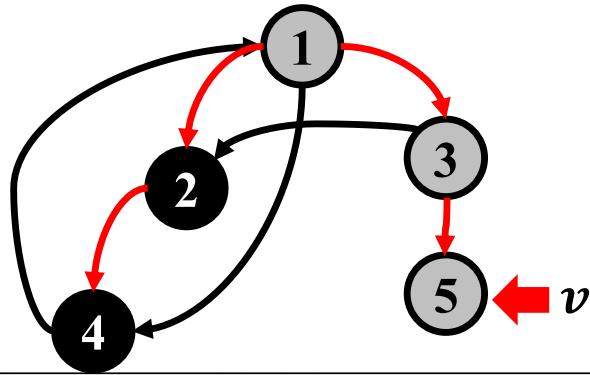
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	G	B	G

v	1	2	3	4	5
d	1	2	6	3	7

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 7$

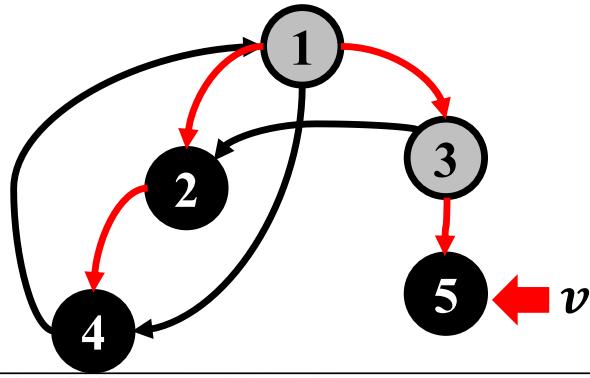
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	G	B	G

v	1	2	3	4	5
d	1	2	6	3	7

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 7$

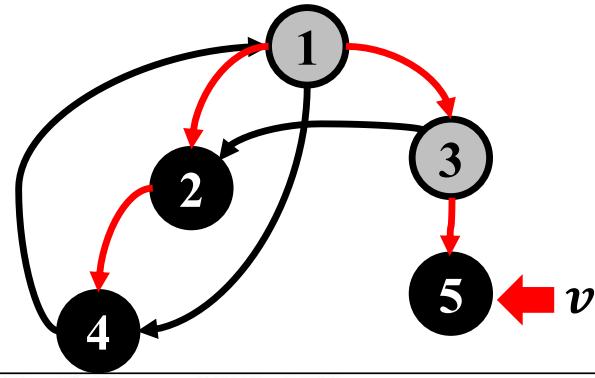
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	G	B	B

v	1	2	3	4	5
d	1	2	6	3	7

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 8$

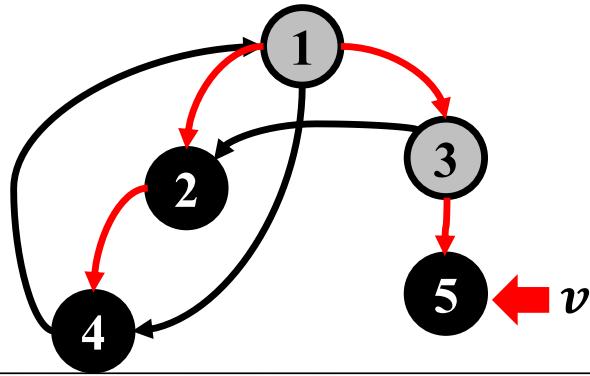
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	G	B	B

v	1	2	3	4	5
d	1	2	6	3	7

v	1	2	3	4	5
f		5		4	

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 8$

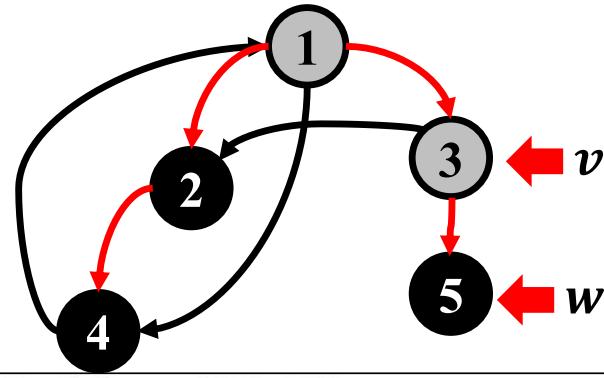
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	G	B	B

v	1	2	3	4	5
d	1	2	6	3	7

v	1	2	3	4	5
f		5		4	8

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 8$

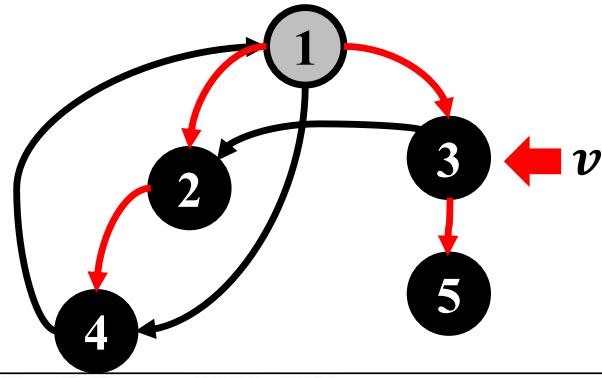
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	G	B	B

v	1	2	3	4	5
d	1	2	6	3	7

v	1	2	3	4	5
f		5		4	8

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 8$

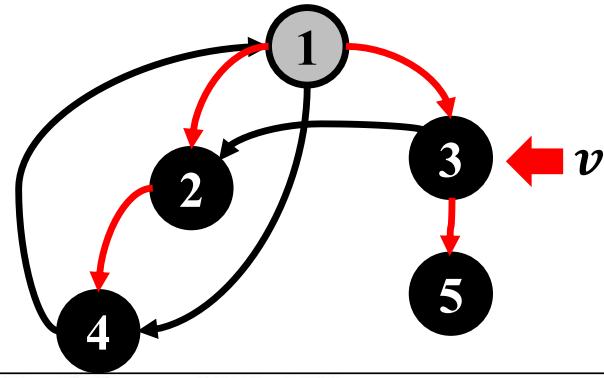
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	B	B	B

v	1	2	3	4	5
d	1	2	6	3	7

v	1	2	3	4	5
f		5		4	8

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$  //
 $f[v] \leftarrow time$ 

```

$time = 9$

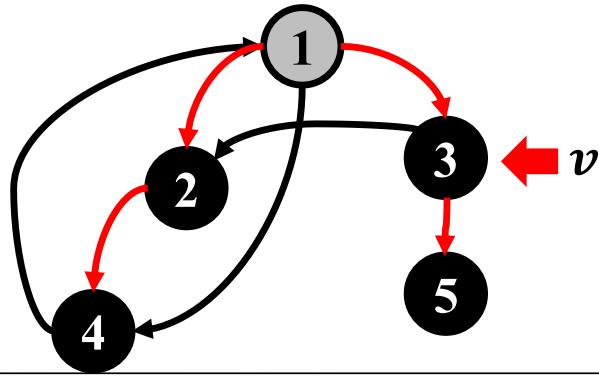
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	B	B	B

v	1	2	3	4	5
d	1	2	6	3	7

v	1	2	3	4	5
f		5		4	8

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 9$

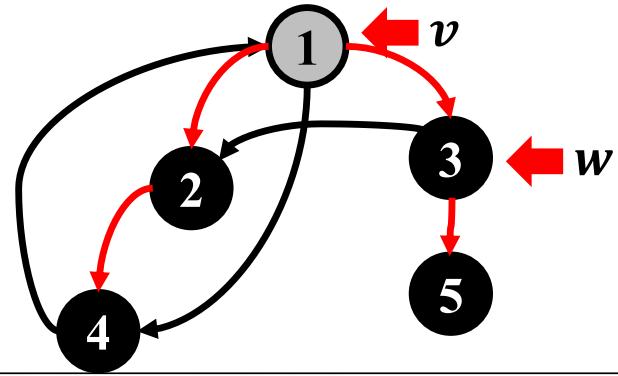
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	B	B	B

v	1	2	3	4	5
d	1	2	6	3	7

v	1	2	3	4	5
f		5	9	4	8

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 9$

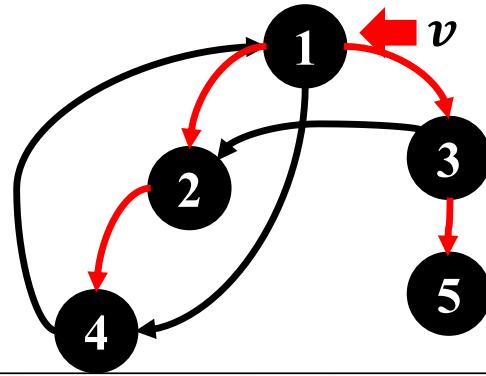
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	G	B	B	B	B

v	1	2	3	4	5
d	1	2	6	3	7

v	1	2	3	4	5
f		5	9	4	8

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 9$

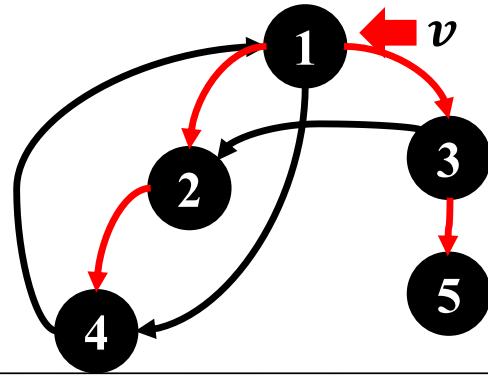
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	B	B	B	B	B

v	1	2	3	4	5
d	1	2	6	3	7

v	1	2	3	4	5
f		5	9	4	8

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 10$

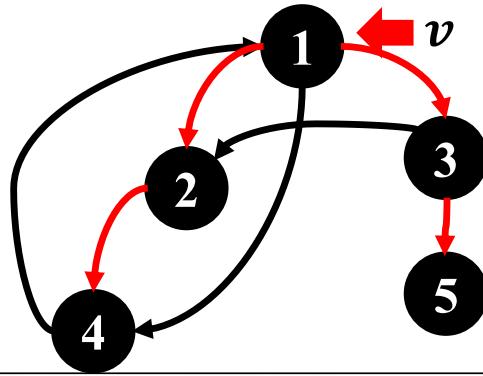
v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	B	B	B	B	B

v	1	2	3	4	5
d	1	2	6	3	7

v	1	2	3	4	5
f		5	9	4	8

深度优先搜索回顾：有向图算法实例



```

 $color[v] \leftarrow GRAY$ 
 $time \leftarrow time + 1$ 
 $d[v] \leftarrow time$ 
for  $w \in Adj[v]$  do
    if  $color[w] = WHITE$  then
         $pred[w] \leftarrow v$ 
        DFS-Visit( $w$ )
    end
end
 $color[v] \leftarrow BLACK$ 
 $time \leftarrow time + 1$ 
 $f[v] \leftarrow time$ 

```

$time = 10$

v	1	2	3	4	5
$pred$	N	1	1	2	3

v	1	2	3	4	5
$color$	B	B	B	B	B

v	1	2	3	4	5
d	1	2	6	3	7

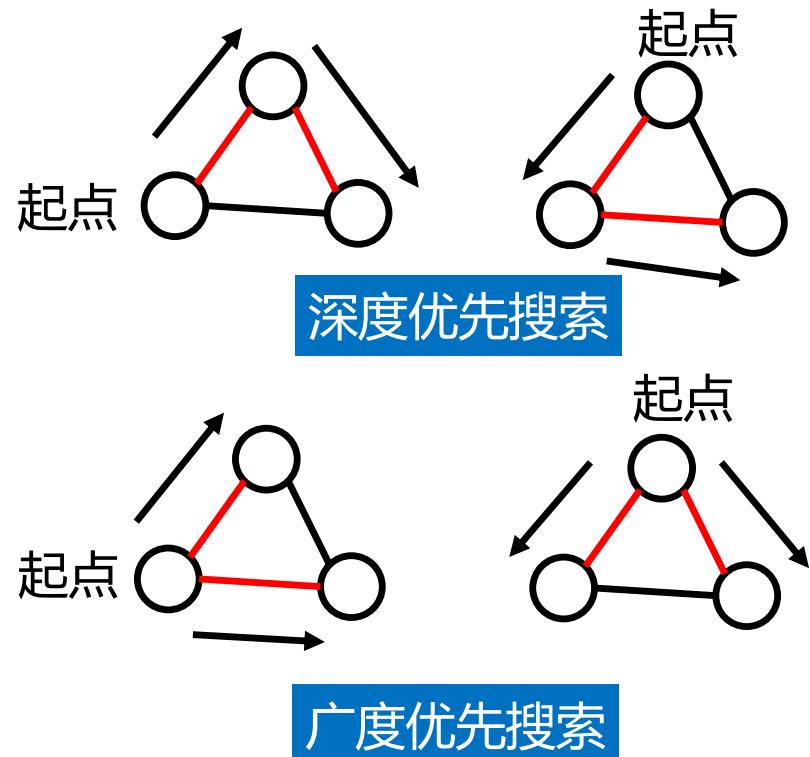
v	1	2	3	4	5
f	10	5	9	4	8

连通无向图的优先树 & 连通有向图的优先森林



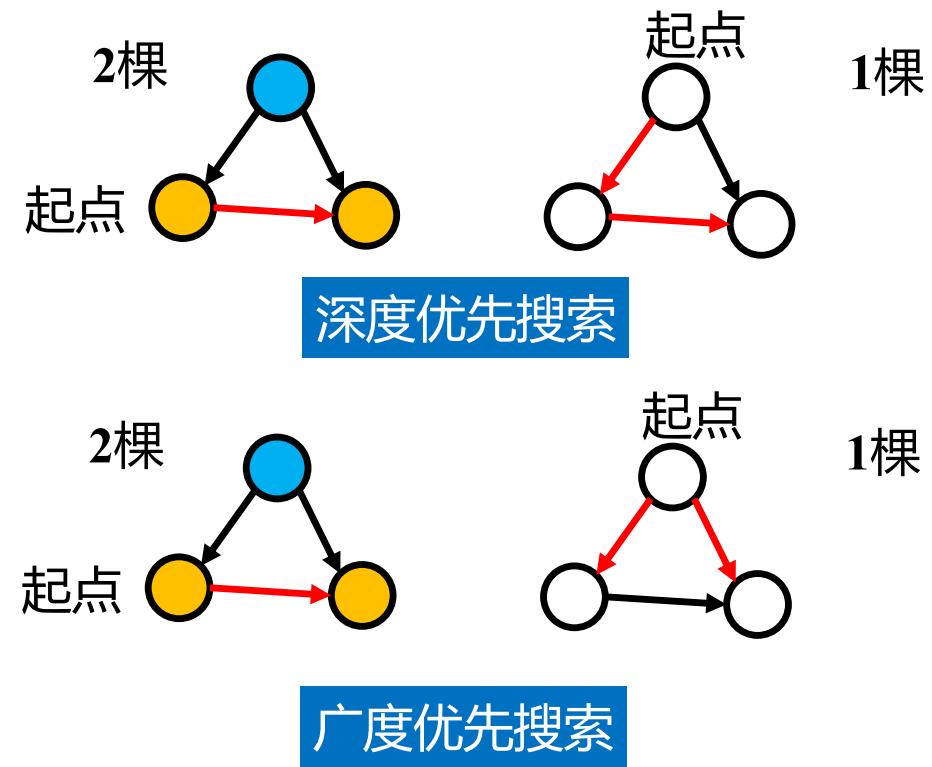
• 无向图

- 树的形状：取决于搜索顺序
- 树的数量：确定1棵优先树

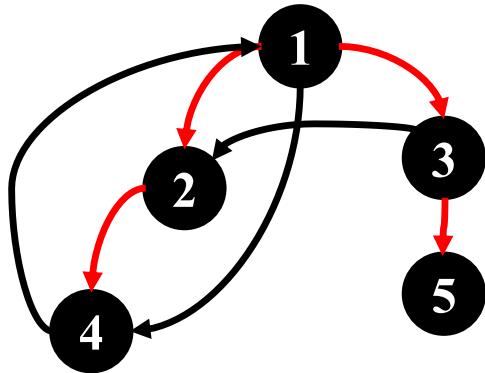


• 有向图

- 树的形状：取决于搜索顺序
- 树的数量：取决于搜索顺序



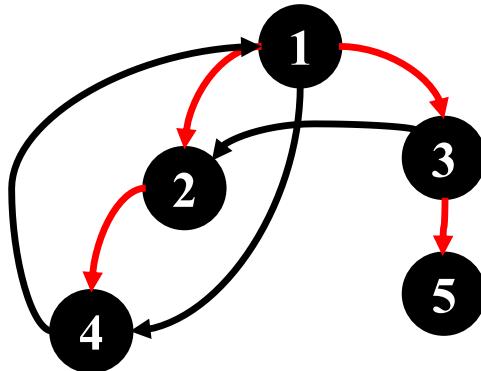
有向图深度优先森林



$time = 10$

v	1	2	3	4	5
$pred$	N	1	1	2	3

有向图深度优先森林

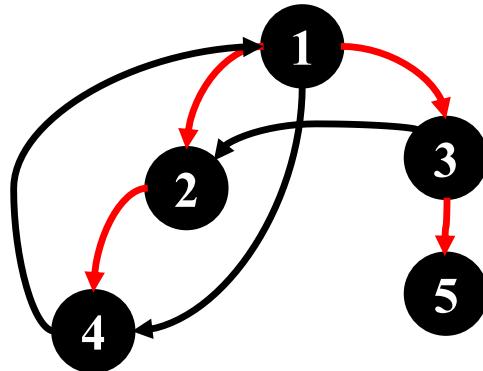


$time = 10$

v	1	2	3	4	5
$pred$	N	1	1	2	3

- 回顾深度优先搜索边的性质
 - 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系
 - 对于无向图，非树边一定是后向边

有向图深度优先森林



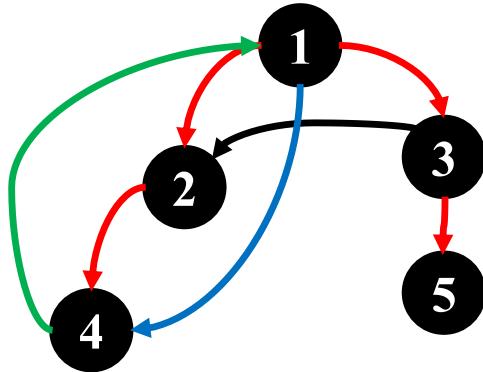
$time = 10$

v	1	2	3	4	5
$pred$	N	1	1	2	3

区别1：祖先指向后代？还是相反？

- 回顾深度优先搜索边的性质
 - 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系
 - 对于无向图，非树边一定是后向边

有向图深度优先森林



$time = 10$

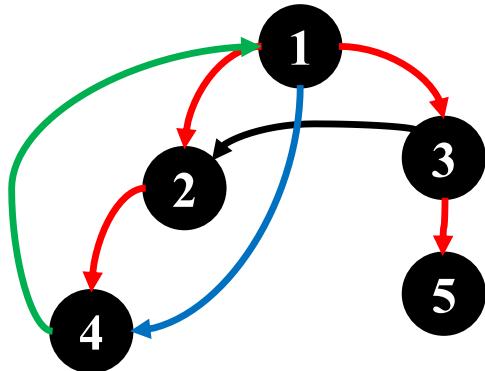
v	1	2	3	4	5
$pred$	N	1	1	1	3

区别1：祖先指向后代？还是相反？

- 回顾深度优先搜索边的性质
 - 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系
 - 对于无向图，非树边一定是后向边



有向图深度优先森林



$time = 10$

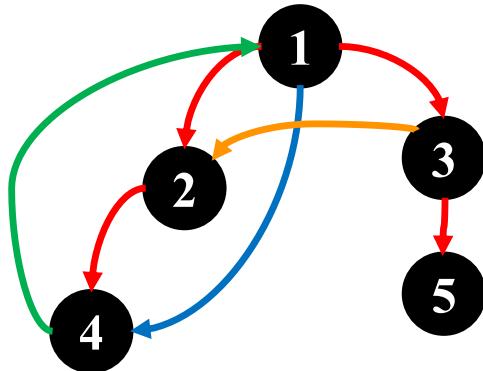
v	1	2	3	4	5
$pred$	N	1	1	1	3

区别1：祖先指向后代？还是相反？

- 回顾深度优先搜索边的性质
 - 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系
 - 对于无向图，非树边一定是后向边

区别2：非树边出现在兄弟顶点之间

有向图深度优先森林



$time = 10$

v	1	2	3	4	5
$pred$	N	1	1	2	3

区别1：祖先指向后代？还是相反？

- 回顾深度优先搜索边的性质

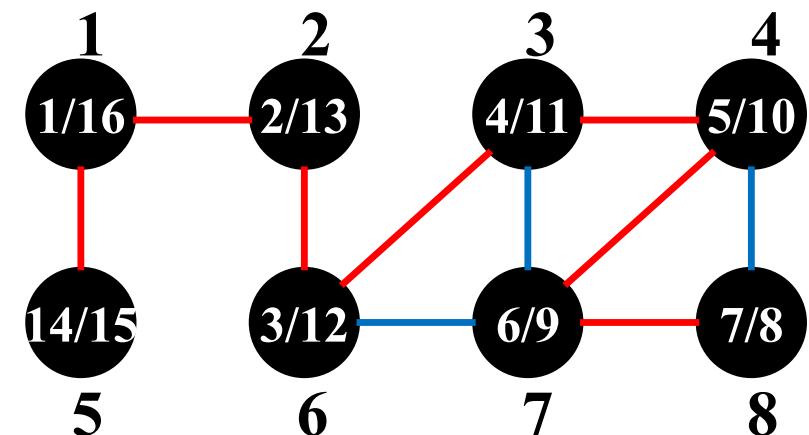
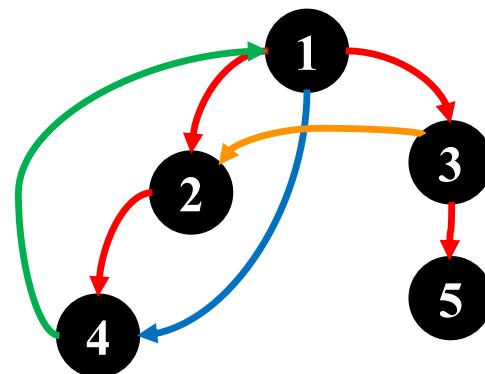
- 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系
- 对于无向图，非树边一定是后向边

区别2：非树边出现在兄弟顶点之间

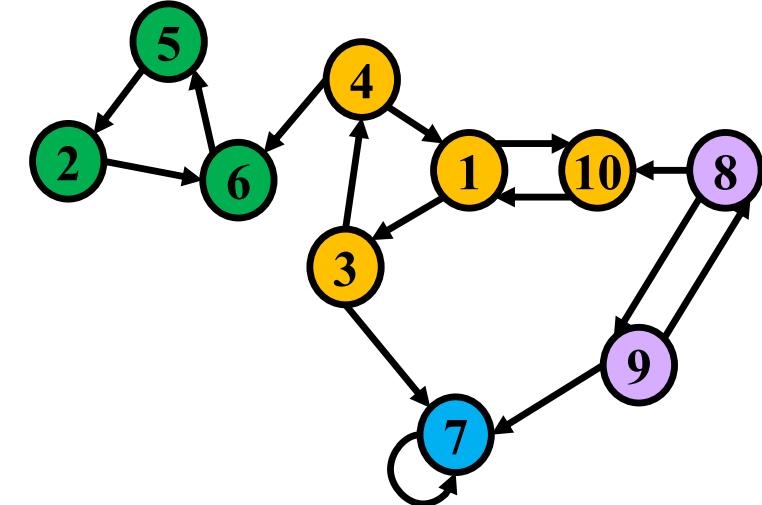
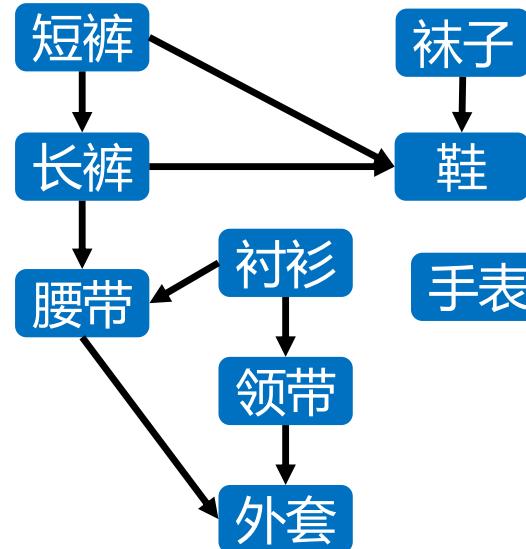
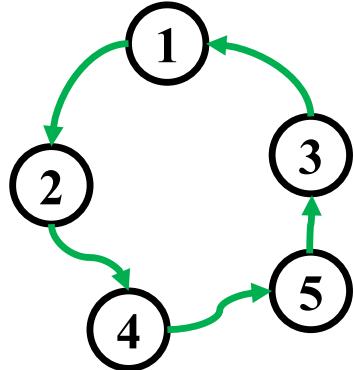
深度优先搜索边的分类



- 有向图，深度优先搜索有4类边
 - 树边：在深度优先树中的边
 - 前向边：不在深度优先树中，从祖先指向后代的边
 - 后向边：从后代指向祖先的边
 - 横向边：顶点不具有祖先后代关系的边
- 无向图，深度优先搜索有2类边
 - 树边：在深度优先树中的边
 - 后向边：两顶点有祖先后代关系的非树边



深度优先搜索应用



环路的存在性判断

拓扑排序

强连通分量

图算法篇：有向图中环路的存在性判断

问题定义



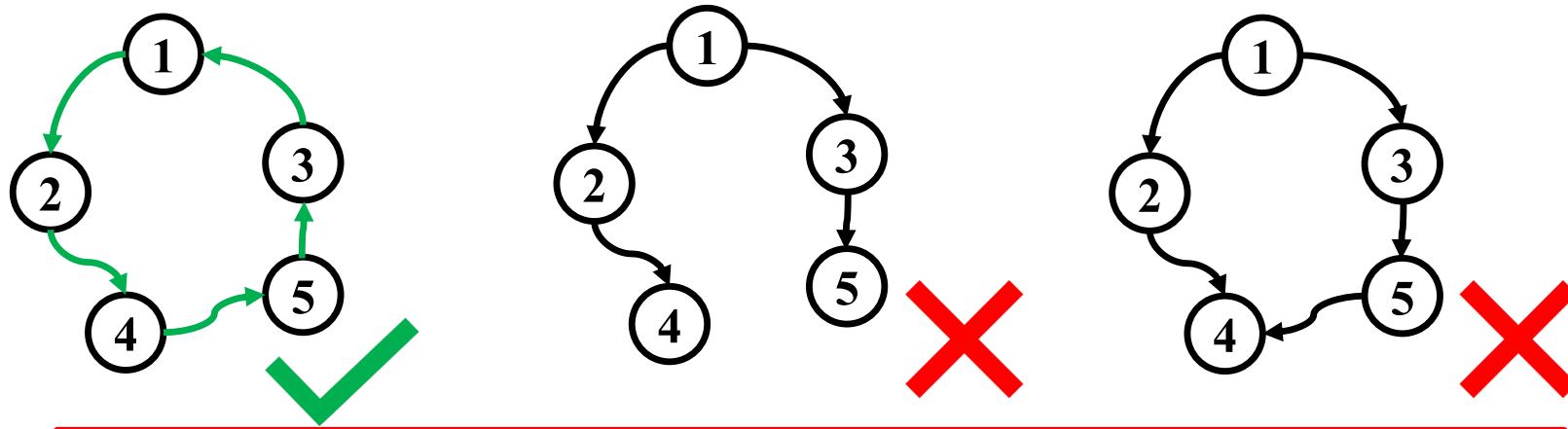
有向图中环路的存在性判断

输入

- 有向图 $G = \langle V, E \rangle$, V 是顶点集合, E 是边的集合

输出

- 图 G 是否存在环

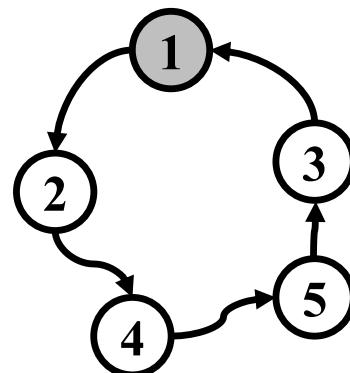


问题：深度优先搜索边的性质能否帮助解决问题？



问题观察

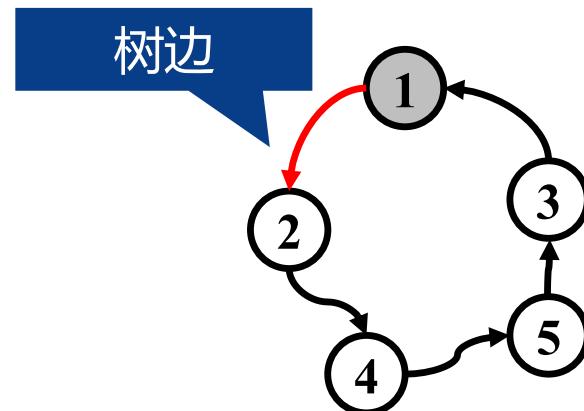
- 观察环路上深度优先搜索时边的种类
 - 树边：在深度优先树中的边
 - 前向边：不在深度优先树中，从祖先指向后代的边
 - 后向边：从后代指向祖先的边
 - 横向边：顶点不具有祖先后代关系的边





问题观察

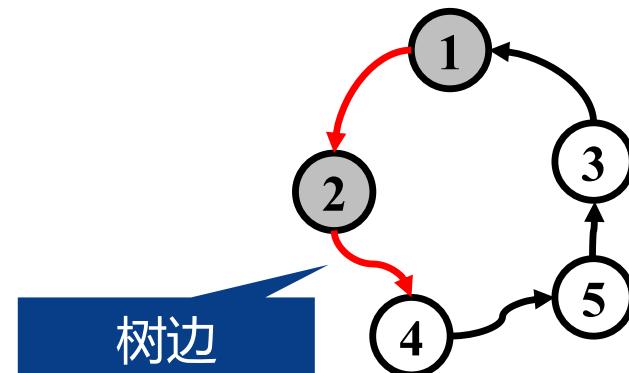
- 观察环路上深度优先搜索时边的种类
 - 树边：在深度优先树中的边
 - 前向边：不在深度优先树中，从祖先指向后代的边
 - 后向边：从后代指向祖先的边
 - 横向边：顶点不具有祖先后代关系的边





问题观察

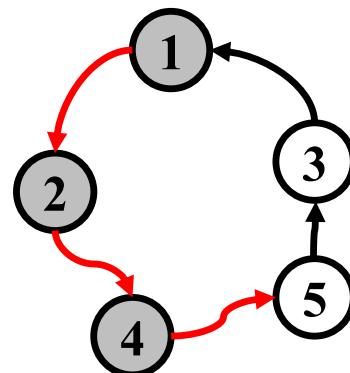
- 观察环路上深度优先搜索时边的种类
 - 树边：在深度优先树中的边
 - 前向边：不在深度优先树中，从祖先指向后代的边
 - 后向边：从后代指向祖先的边
 - 横向边：顶点不具有祖先后代关系的边





问题观察

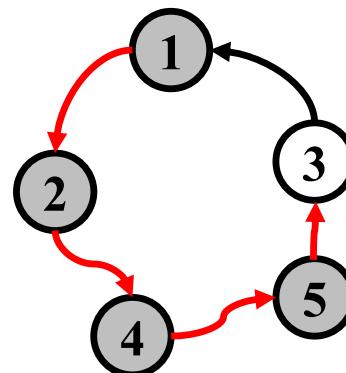
- 观察环路上深度优先搜索时边的种类
 - 树边：在深度优先树中的边
 - 前向边：不在深度优先树中，从祖先指向后代的边
 - 后向边：从后代指向祖先的边
 - 横向边：顶点不具有祖先后代关系的边





问题观察

- 观察环路上深度优先搜索时边的种类
 - 树边：在深度优先树中的边
 - 前向边：不在深度优先树中，从祖先指向后代的边
 - 后向边：从后代指向祖先的边
 - 横向边：顶点不具有祖先后代关系的边

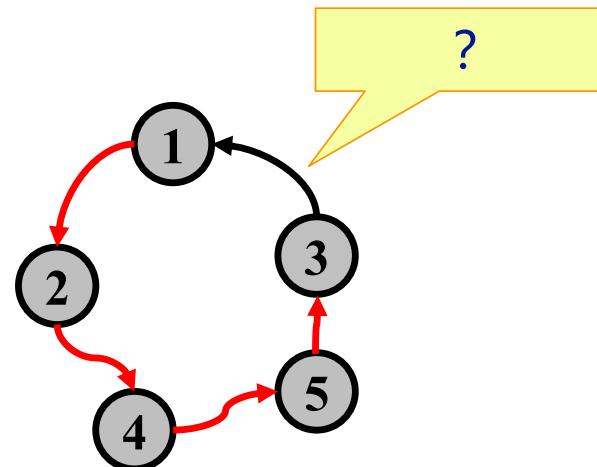




问题观察

- 观察环路上深度优先搜索时边的种类

- 树边：在深度优先树中的边
- 前向边：不在深度优先树中，从祖先指向后代的边
- 后向边：从后代指向祖先的边
- 横向边：顶点不具有祖先后代关系的边

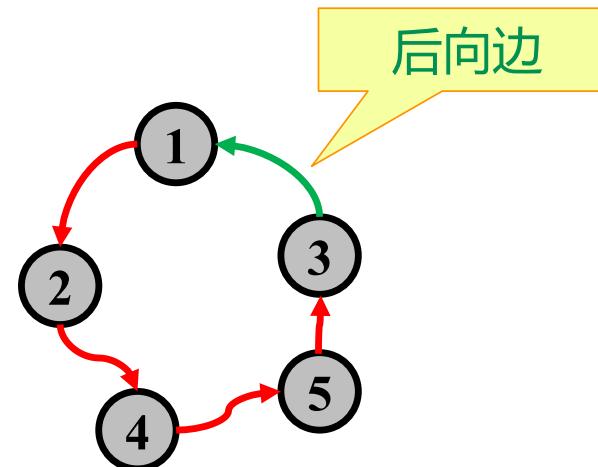




问题观察

- 观察环路上深度优先搜索时边的种类

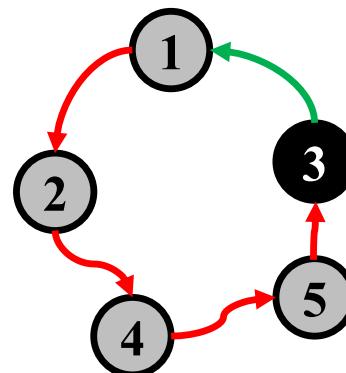
- 树边：在深度优先树中的边
- 前向边：不在深度优先树中，从祖先指向后代的边
- 后向边：从后代指向祖先的边
- 横向边：顶点不具有祖先后代关系的边





问题观察

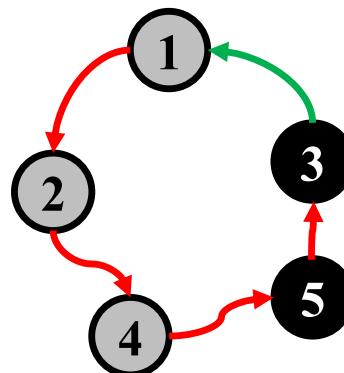
- 观察环路上深度优先搜索时边的种类
 - 树边：在深度优先树中的边
 - 前向边：不在深度优先树中，从祖先指向后代的边
 - 后向边：从后代指向祖先的边
 - 横向边：顶点不具有祖先后代关系的边





问题观察

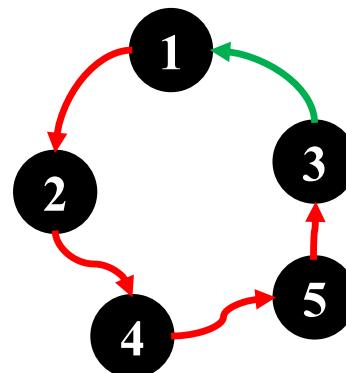
- 观察环路上深度优先搜索时边的种类
 - 树边：在深度优先树中的边
 - 前向边：不在深度优先树中，从祖先指向后代的边
 - 后向边：从后代指向祖先的边
 - 横向边：顶点不具有祖先后代关系的边





问题观察

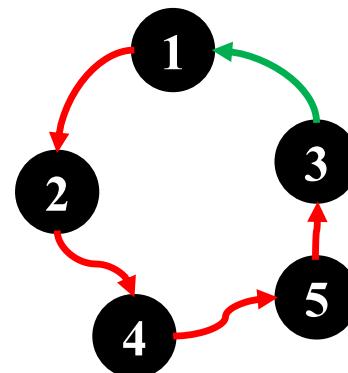
- 观察环路上深度优先搜索时边的种类
 - 树边：在深度优先树中的边
 - 前向边：不在深度优先树中，从祖先指向后代的边
 - 后向边：从后代指向祖先的边
 - 横向边：顶点不具有祖先后代关系的边





问题观察

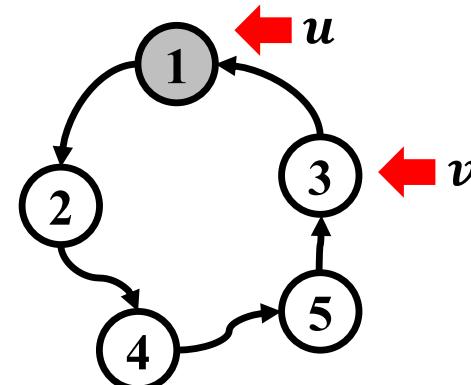
- 观察环路上深度优先搜索时边的种类
 - 树边：在深度优先树中的边
 - 前向边：不在深度优先树中，从祖先指向后代的边
 - 后向边：从后代指向祖先的边
 - 横向边：顶点不具有祖先后代关系的边
- 猜想：有向图存在环路 \Leftrightarrow 搜索时出现后向边



猜想证明



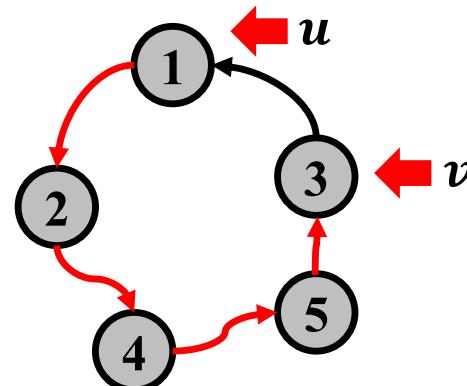
- 猜想：有向图存在环路 \Leftrightarrow 搜索时出现后向边
- 证明：有向图存在环路 \Rightarrow 搜索时出现后向边
 - 不妨设环路上被搜索的第一个点为 u , v 是在环路上指向 u 的点





猜想证明

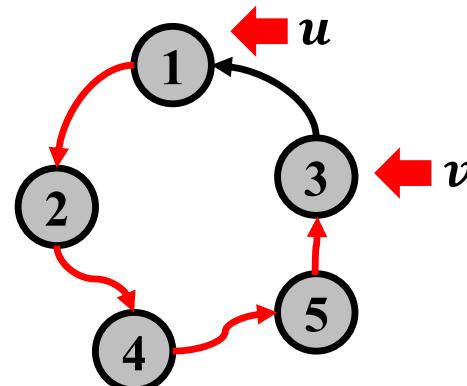
- 猜想：有向图存在环路 \Leftrightarrow 搜索时出现后向边
- 证明：有向图存在环路 \Rightarrow 搜索时出现后向边
 - 不妨设环路上被搜索的第一个点为 u , v 是在环路上指向 u 的点
 - u 可达 v , 深度优先搜索可以搜索到 v





猜想证明

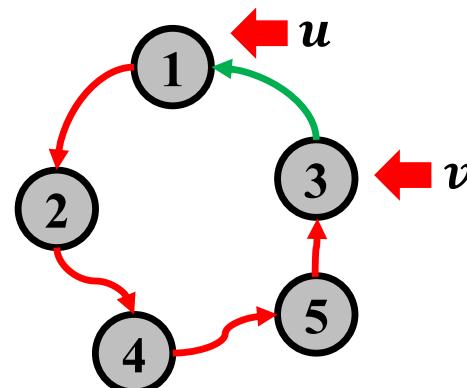
- 猜想：有向图存在环路 \Leftrightarrow 搜索时出现后向边
- 证明：有向图存在环路 \Rightarrow 搜索时出现后向边
 - 不妨设环路上被搜索的第一个点为 u , v 是在环路上指向 u 的点
 - u 可达 v , 深度优先搜索可以搜索到 v
 - 搜索 v 时, 由于 v 指向 u , 必能再次发现顶点 u





猜想证明

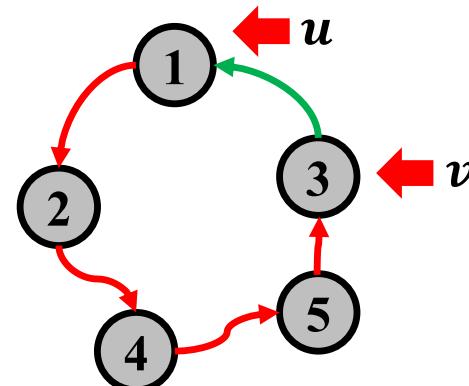
- 猜想：有向图存在环路 \Leftrightarrow 搜索时出现后向边
- 证明：有向图存在环路 \Rightarrow 搜索时出现后向边
 - 不妨设环路上被搜索的第一个点为 u , v 是在环路上指向 u 的点
 - u 可达 v , 深度优先搜索可以搜索到 v
 - 搜索 v 时, 由于 v 指向 u , 必能再次发现顶点 u
 - 从后代搜索祖先, 出现后向边



猜想证明



- 猜想：有向图存在环路 \Leftrightarrow 搜索时出现后向边
- 证明：有向图存在环路 \Rightarrow 搜索时出现后向边
 - 深度优先树中祖先可达后代
 - 后向边从后代指向祖先
 - 后代和祖先之间存在环路





伪代码

- DFS-Judge-Cycle(G)

```
输入: 图 $G$ 
输出: 是否存在环路
新建数组  $color[1..V], pred[1..V]$ 
//初始化
for  $v \in V$  do
    |  $pred[v] \leftarrow NULL$ 
    |  $color[v] \leftarrow WHITE$ 
end
for  $v \in V$  do
    | if  $color[v] = WHITE$  then
    |   | if  $DFS\text{-Visit-Judge-Cycle}(G, v) = TRUE$  then
    |   |   | return  $TRUE$ 
    |   | end
    | end
end
return  $FALSE$ 
```

发现环则返回True



伪代码

- **DFS-Visit-Judge-Cycle(G, v)**

输入: 图 G , 顶点 v
输出: 顶点 v 是否在某环路中

```
color[v]  $\leftarrow$  GRAY
for  $w \in G.Adj[v]$  do
    if color[w] = GRAY then
        return TRUE
    end
    if color[w] = WHITE then
        pred[w]  $\leftarrow v$ 
        if DFS-Visit-Judge-Cycle( $G, w$ ) = TRUE then
            return TRUE
        end
    end
end
color[v]  $\leftarrow$  BLACK
return FALSE
```

搜索到灰色点
(发现后向边)



伪代码

- **DFS-Visit-Judge-Cycle(G, v)**

输入: 图 G , 顶点 v

输出: 顶点 v 是否在某环路中

$color[v] \leftarrow GRAY$

for $w \in G.Adj[v]$ **do**

if $color[w] = GRAY$ **then**

return $TRUE$

end

if $color[w] = WHITE$ **then**

$pred[w] \leftarrow v$

if $DFS\text{-Visit-Judge-Cycle}(G, w) = TRUE$ **then**

return $TRUE$

end

end

end

$color[v] \leftarrow BLACK$

return $FALSE$

递归发现环



复杂度分析

- DFS-Visit-Judge-Cycle(G, v)

输入: 图 G , 顶点 v

输出: 顶点 v 是否在某环路中

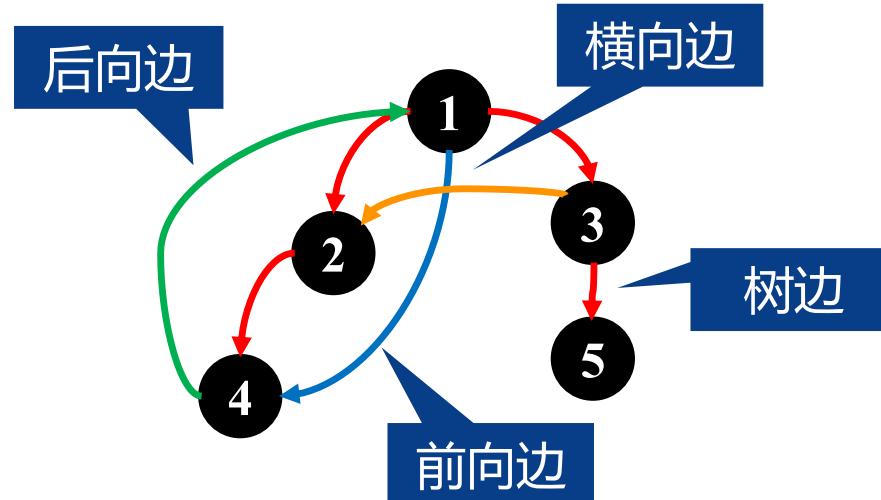
```
color[v] ← GRAY
for w ∈ G.Adj[v] do
    if color[w] = GRAY then
        | return TRUE
    end
    if color[w] = WHITE then
        | pred[w] ← v
        | if DFS-Visit-Judge-Cycle(G, w) = TRUE then
            |     | return TRUE
        | end
    end
end
color[v] ← BLACK
return FALSE
```

时间复杂度 : $O(|V| + |E|)$

小结



- 有向图深度优先树中边的分类



- 树边和后向边的综合利用，使深度优先搜索可判断环的存在性

