

图算法篇：广度优先搜索

童咏昕

北京航空航天大学
计算机学院

中国大学MOOC北航《算法设计与分析》

图的搜索

算法思想

算法实例

算法分析

算法应用

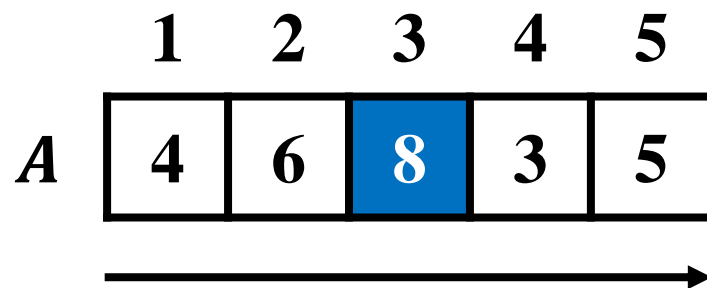
- 数组结构

- 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
<i>A</i>	4	6	8	3	5

- 数组结构

- 查询最大值：简单循环搜索所有元素，记录最大值

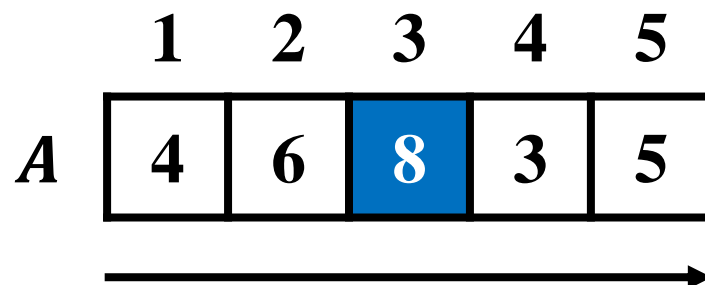


图的搜索



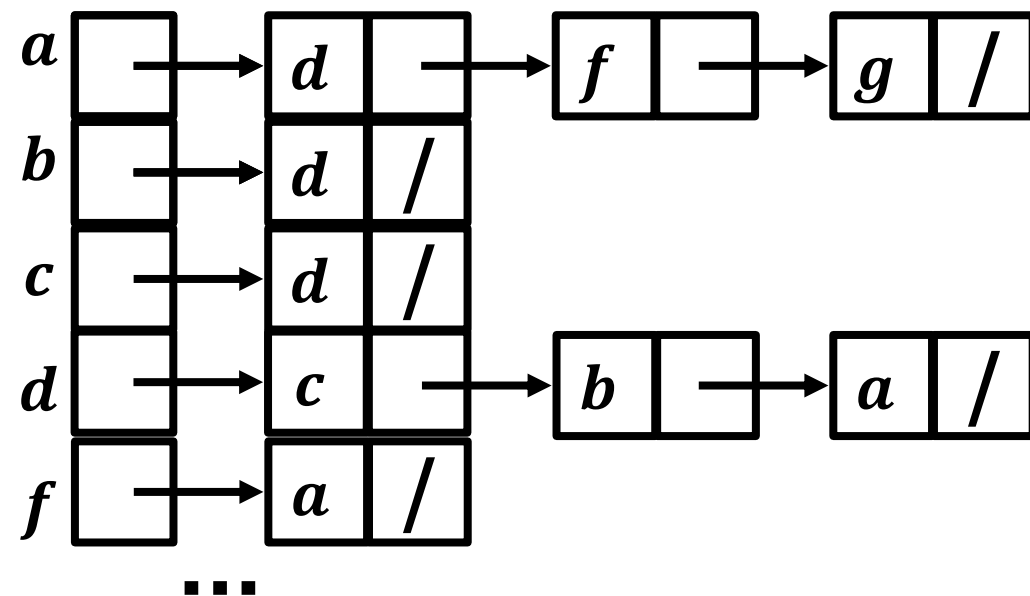
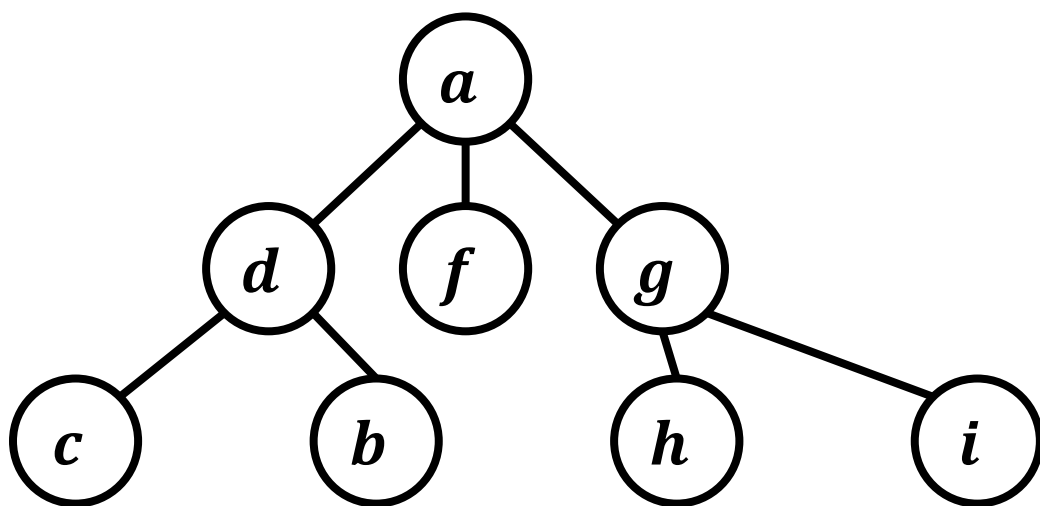
- 数组结构

- 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

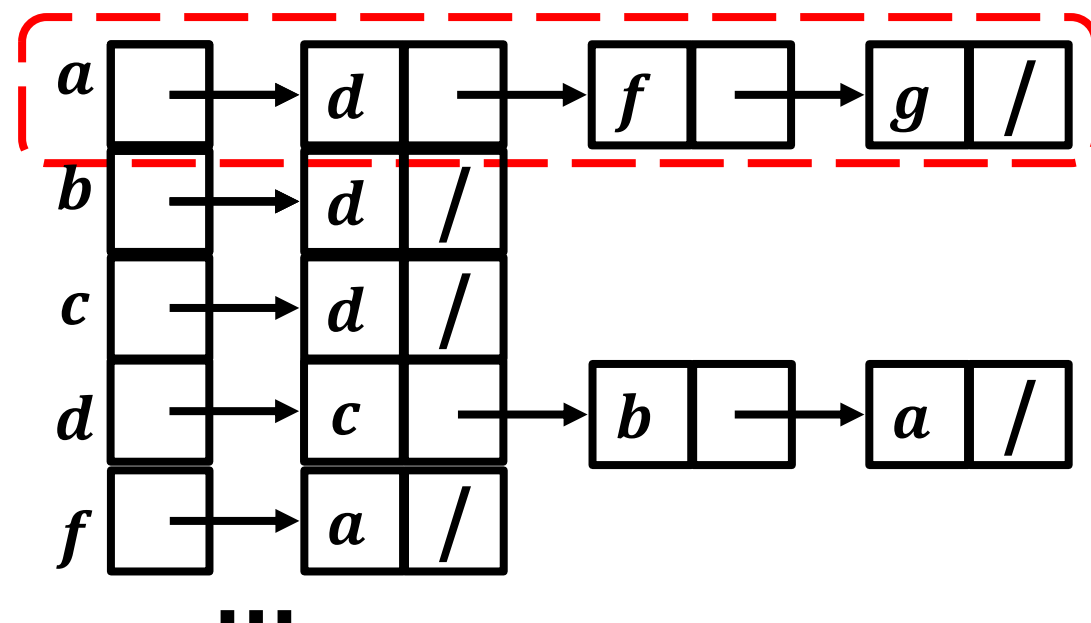
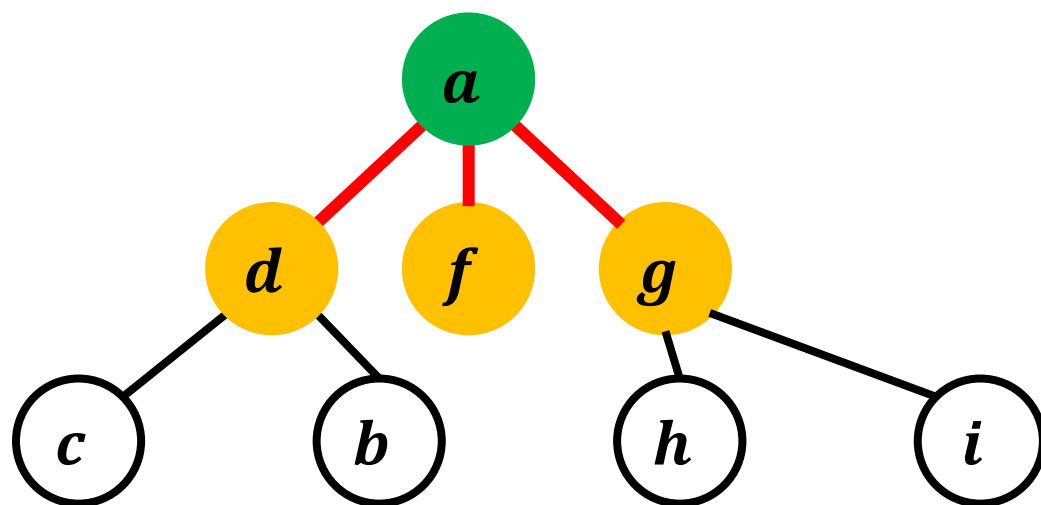
	1	2	3	4	5
<i>A</i>	4	6	8	3	5

→

● 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

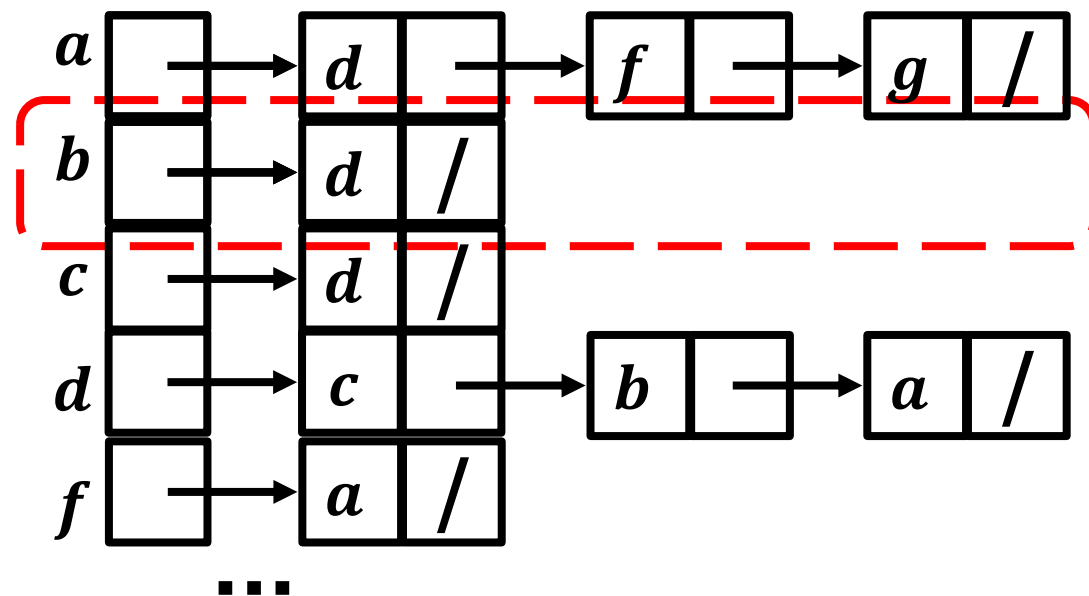
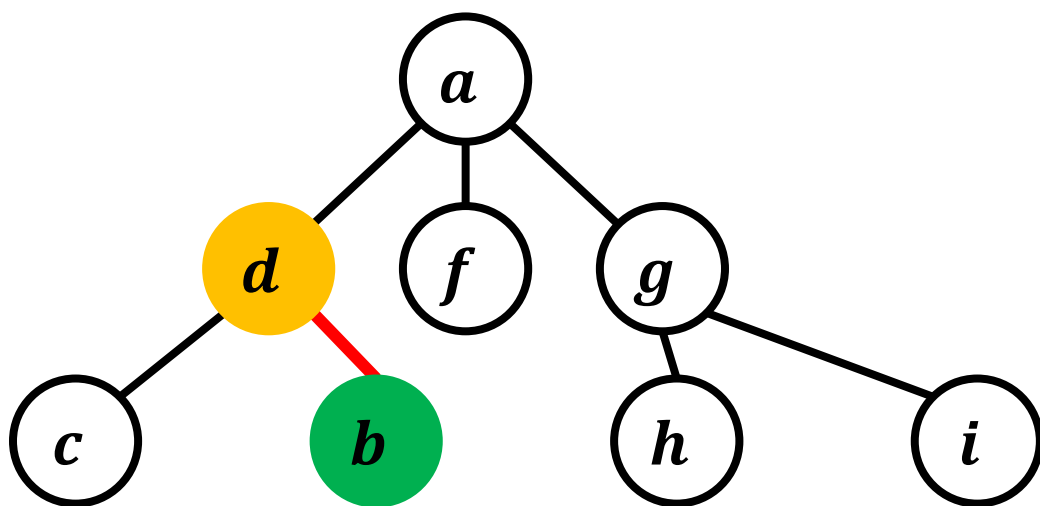
	1	2	3	4	5
<i>A</i>	4	6	8	3	5

→

● 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



- 数组结构

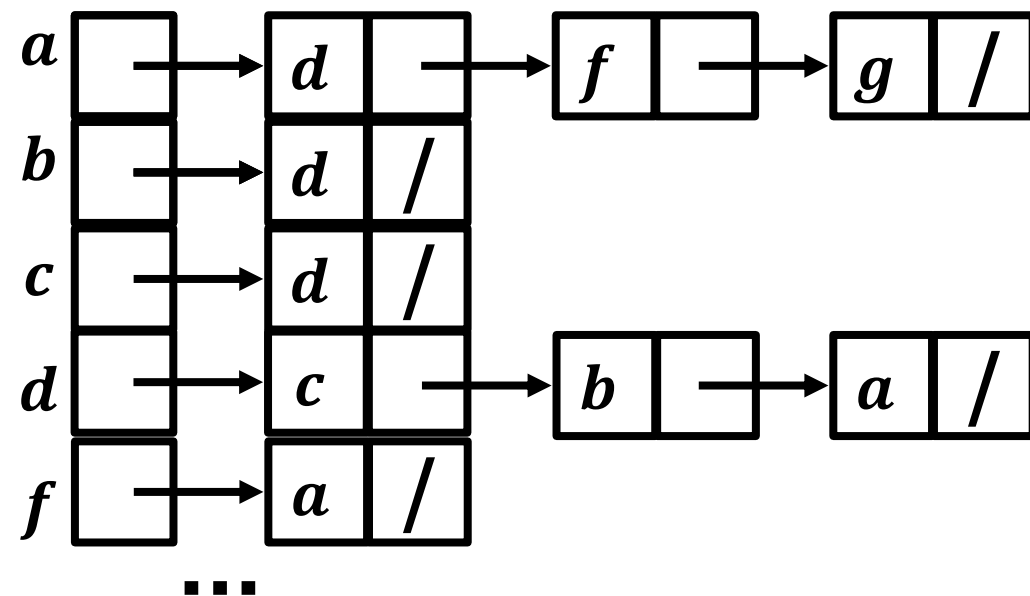
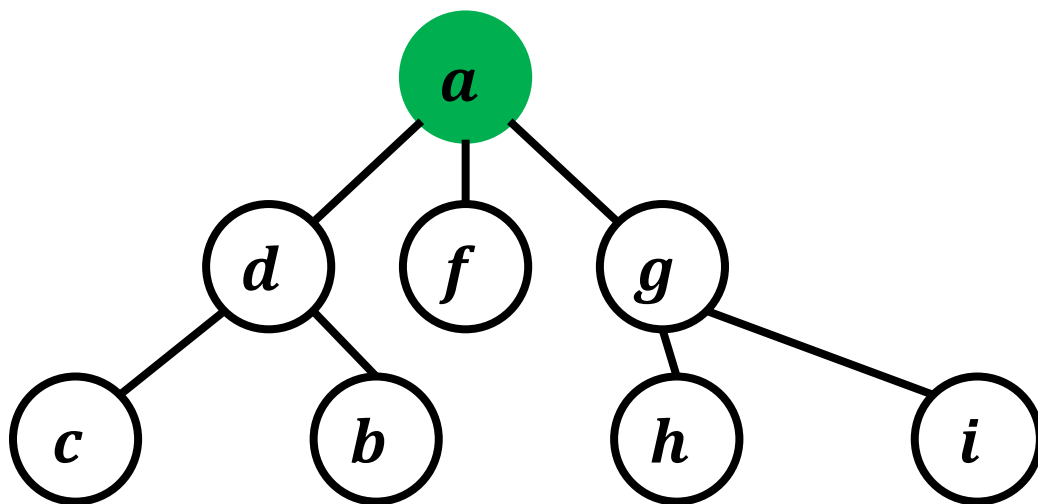
- 查询最大值：简单循环搜索所有元素，记录最大值

	1	2	3	4	5
<i>A</i>	4	6	8	3	5

→

- 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



- 数组结构

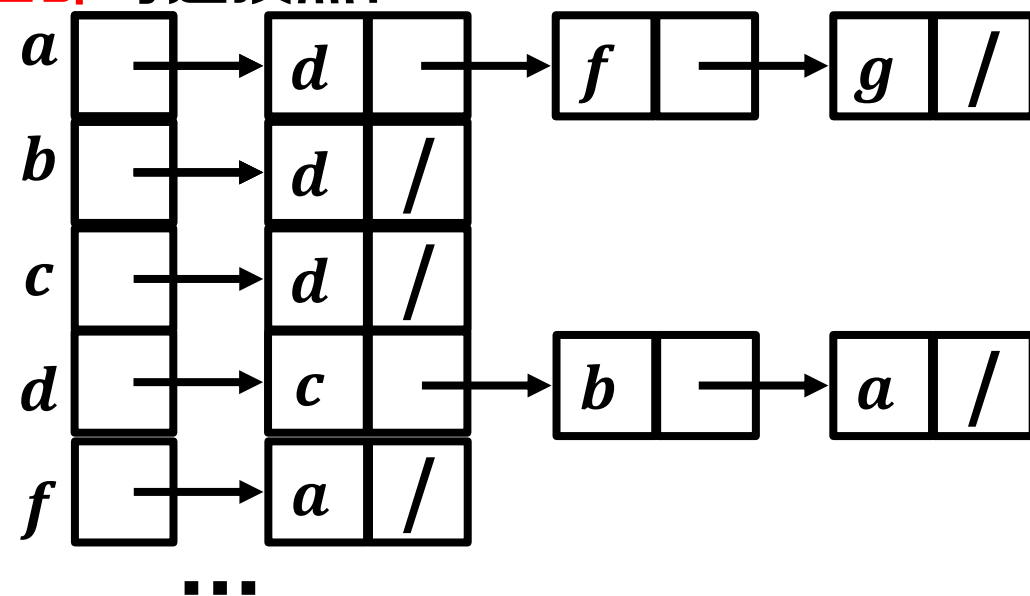
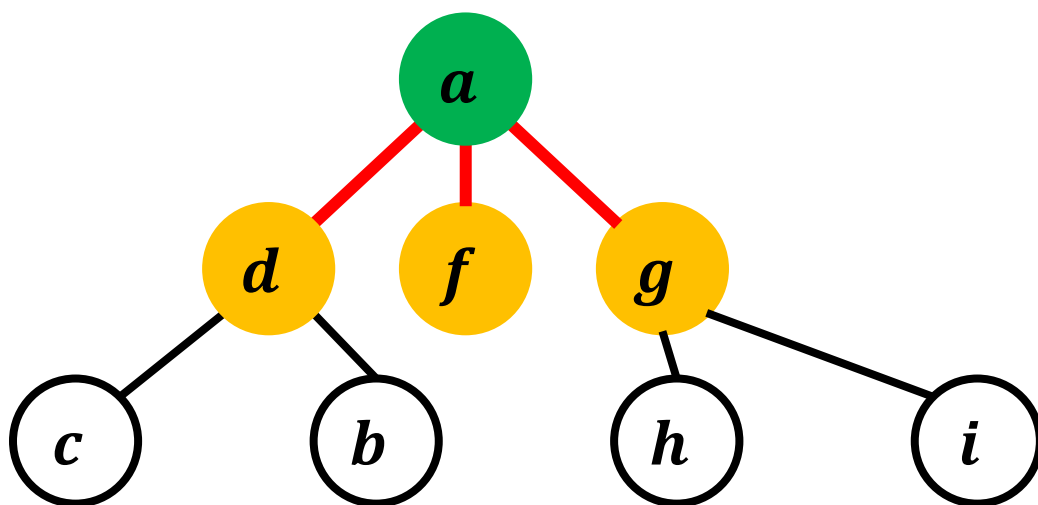
- 查询最大值：简单循环搜索所有元素，记录最大值

	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

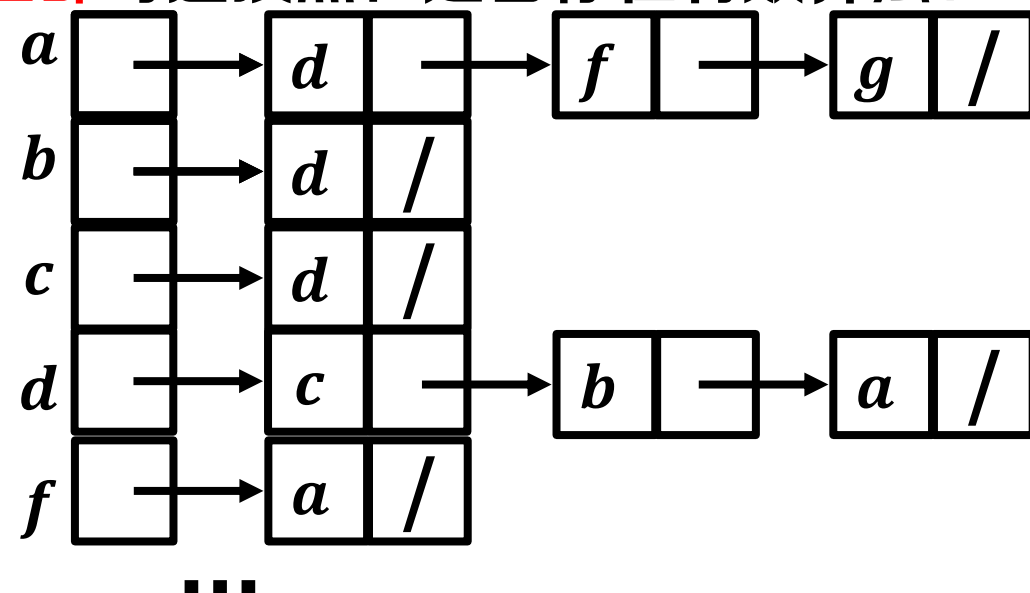
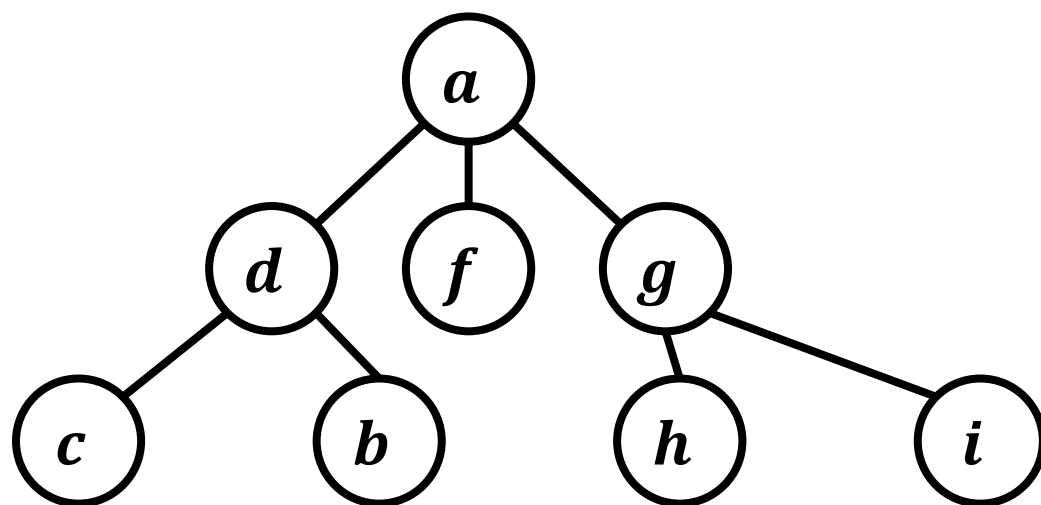
- 查询最大值：简单循环搜索所有元素，记录最大值

	1	2	3	4	5
A	4	6	8	3	5

—————→

● 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

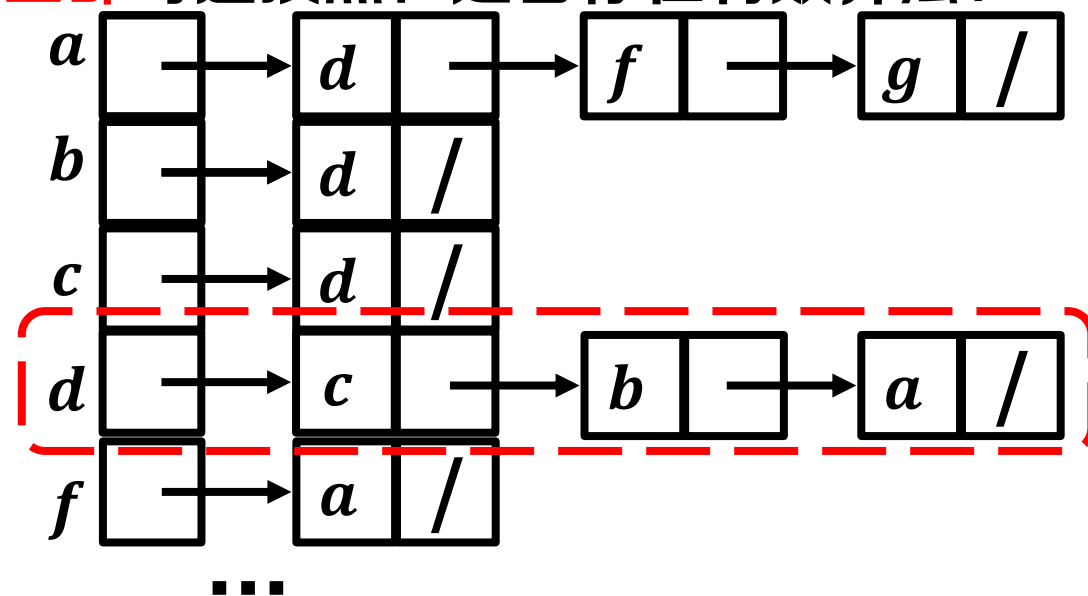
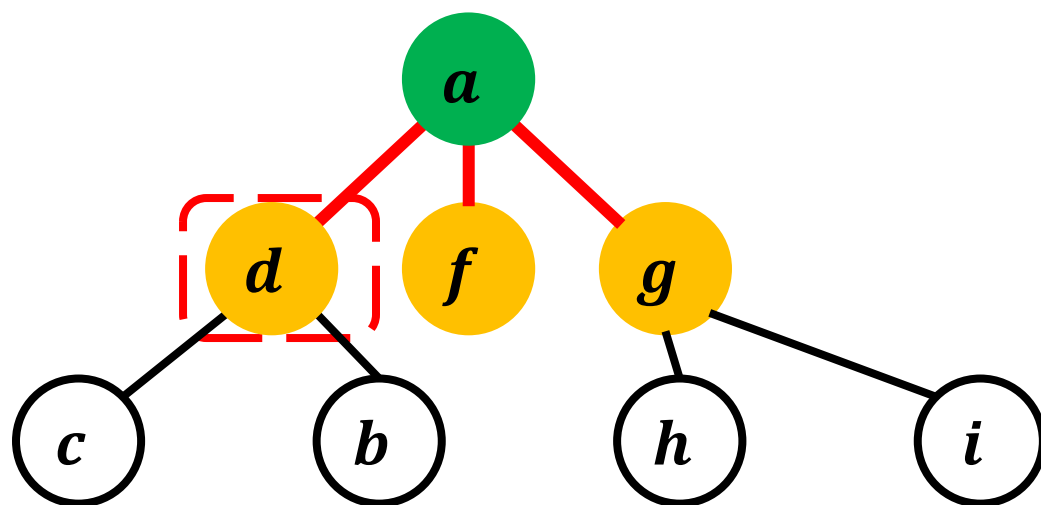
- 查询最大值：简单循环搜索所有元素，记录最大值

	1	2	3	4	5
A	4	6	8	3	5

→

• 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？



- 数组结构

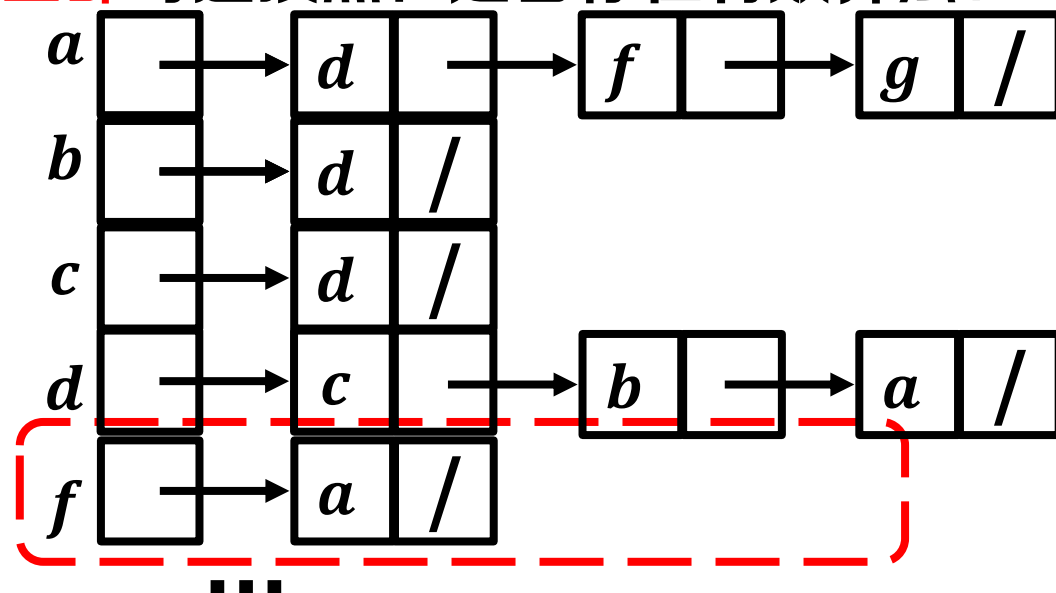
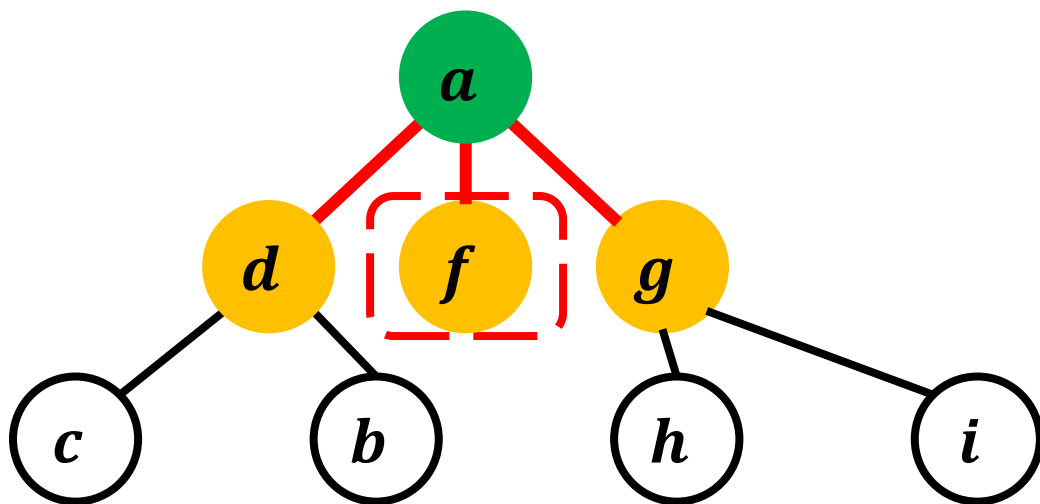
- 查询最大值：简单循环搜索所有元素，记录最大值

	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

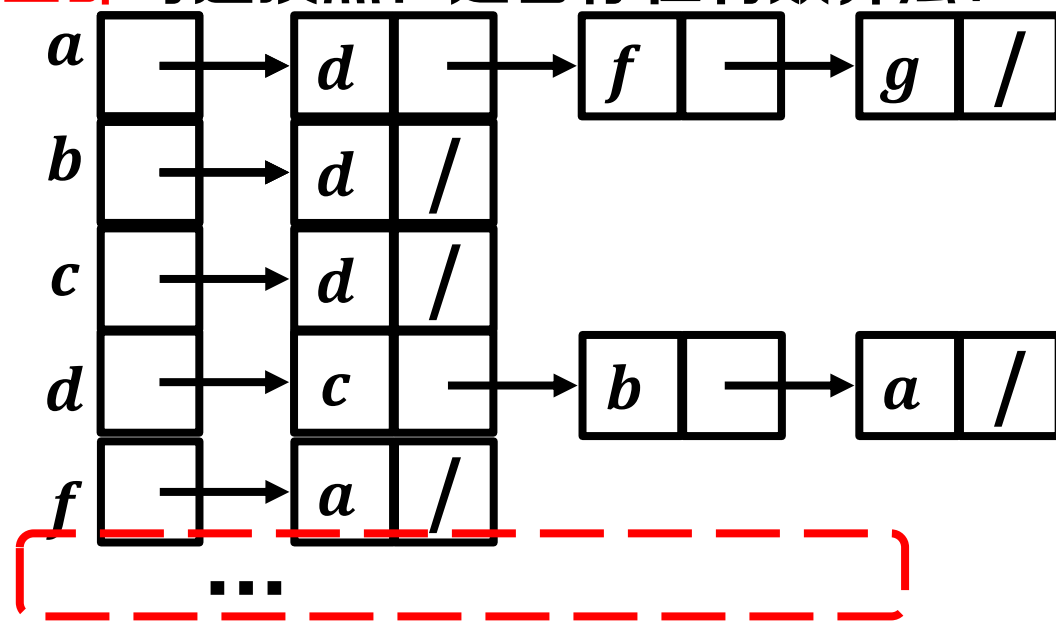
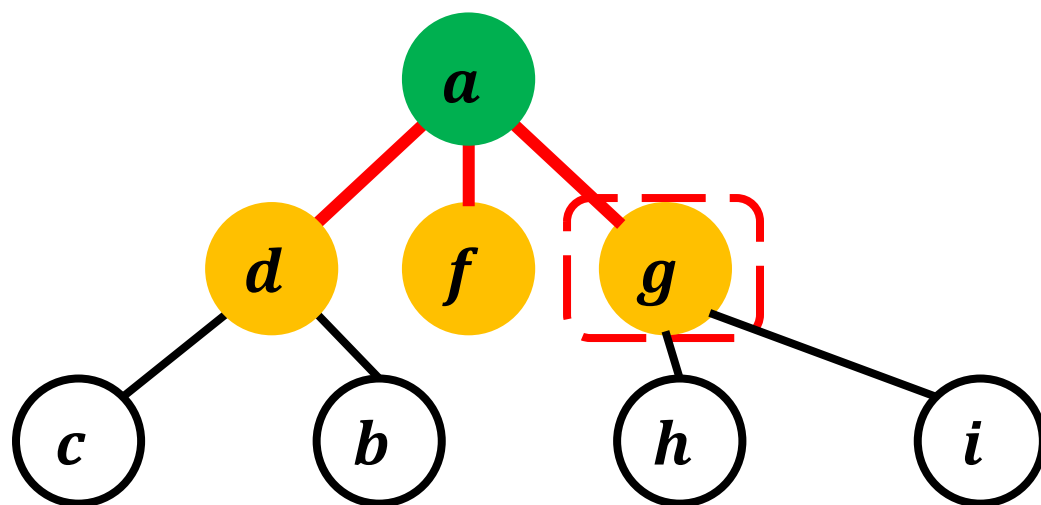
- 查询最大值：简单循环搜索所有元素，记录最大值

	1	2	3	4	5
A	4	6	8	3	5

→

● 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？



- 数组结构

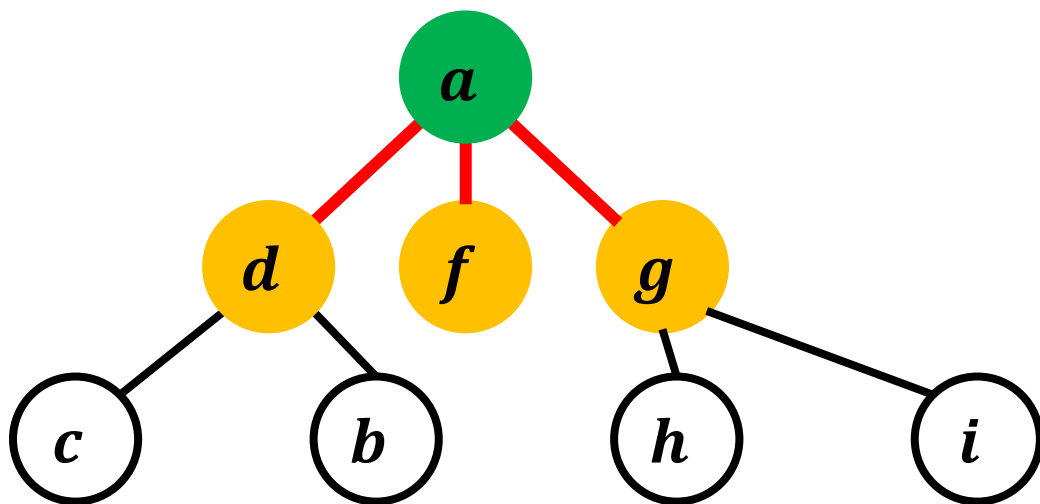
- 查询最大值：简单循环搜索所有元素，记录最大值

	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？

- 数组结构

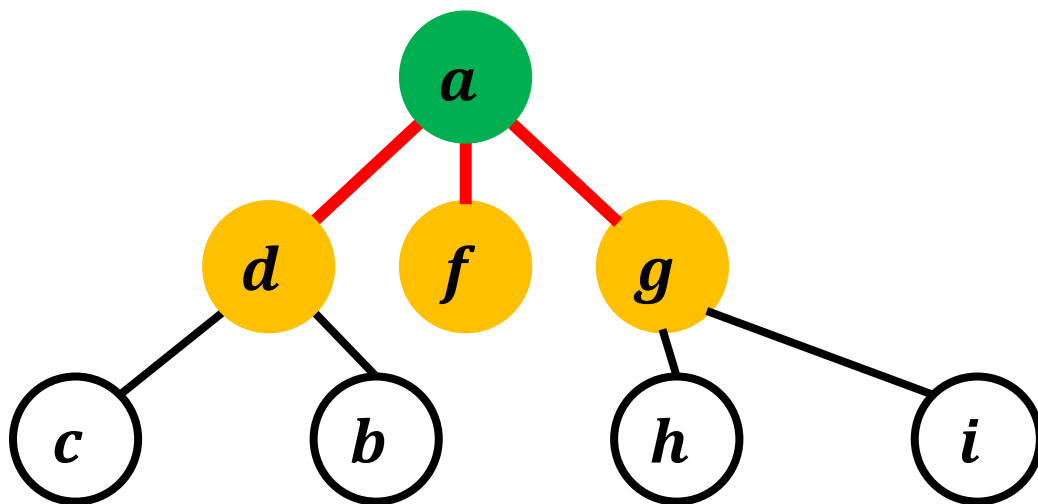
- 查询最大值：简单循环搜索所有元素，记录最大值

	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

- 数组结构

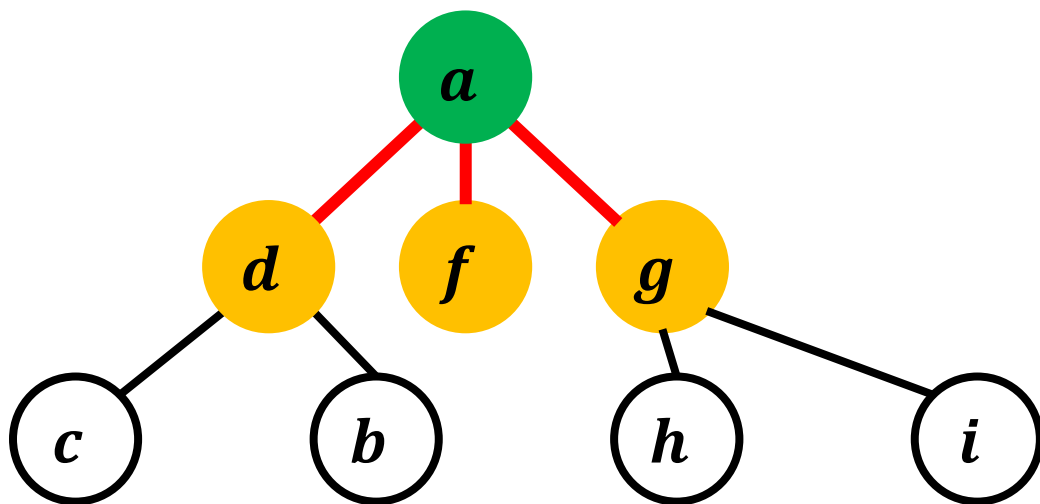
- 查询最大值：简单循环搜索所有元素，记录最大值

	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

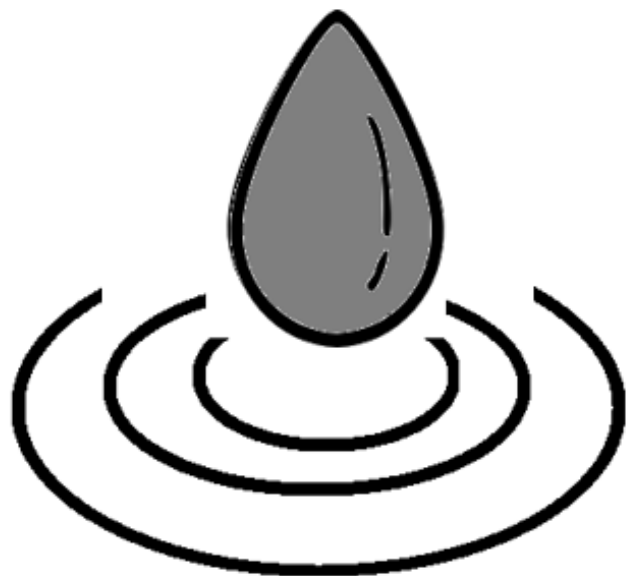
算法思想

算法实例

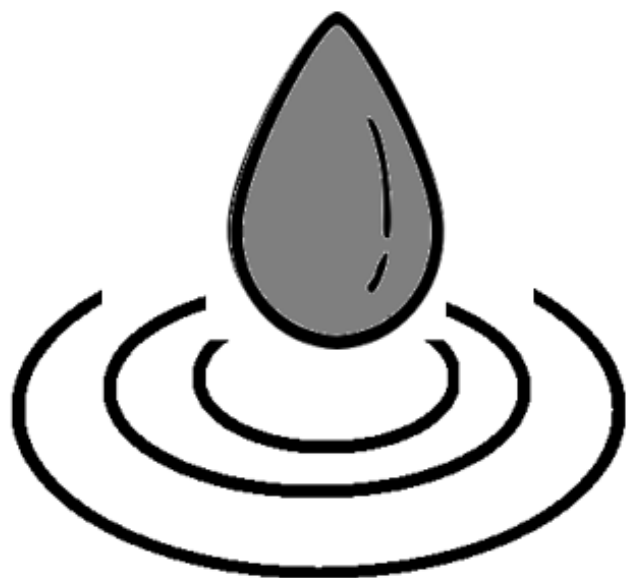
算法分析

算法应用

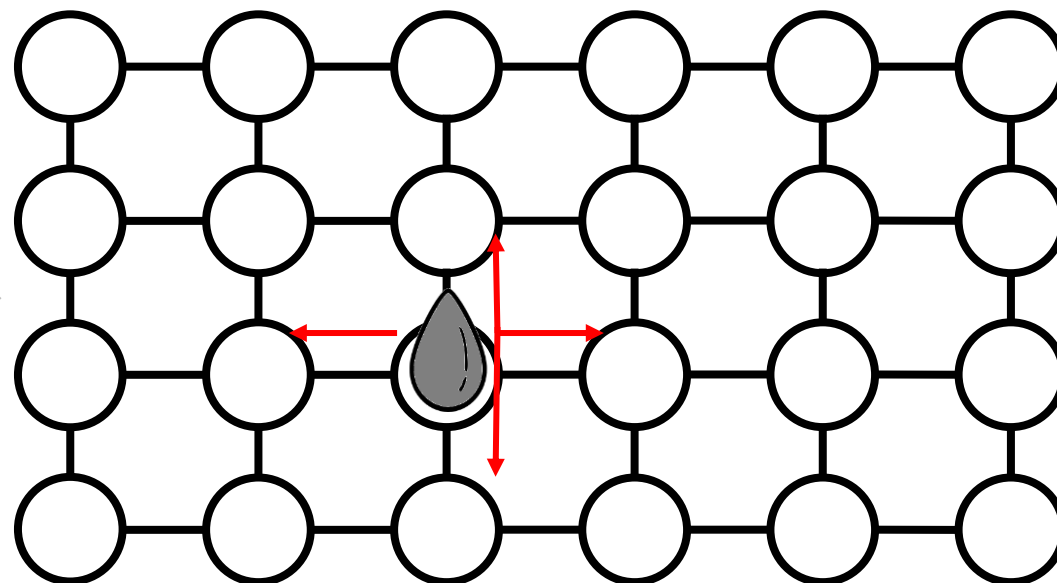
- 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



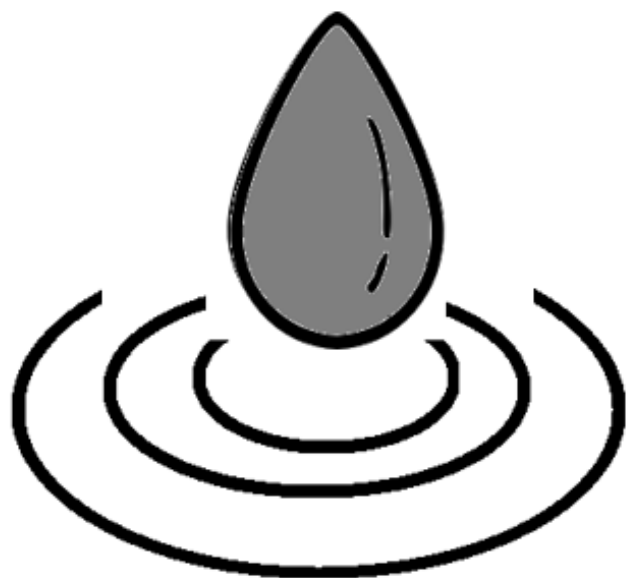
- 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



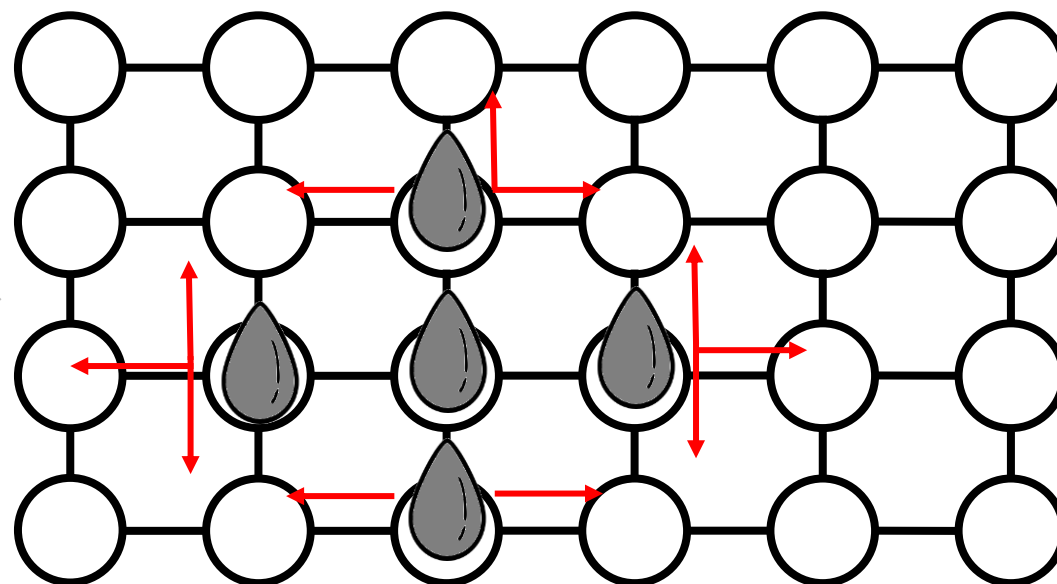
图结构



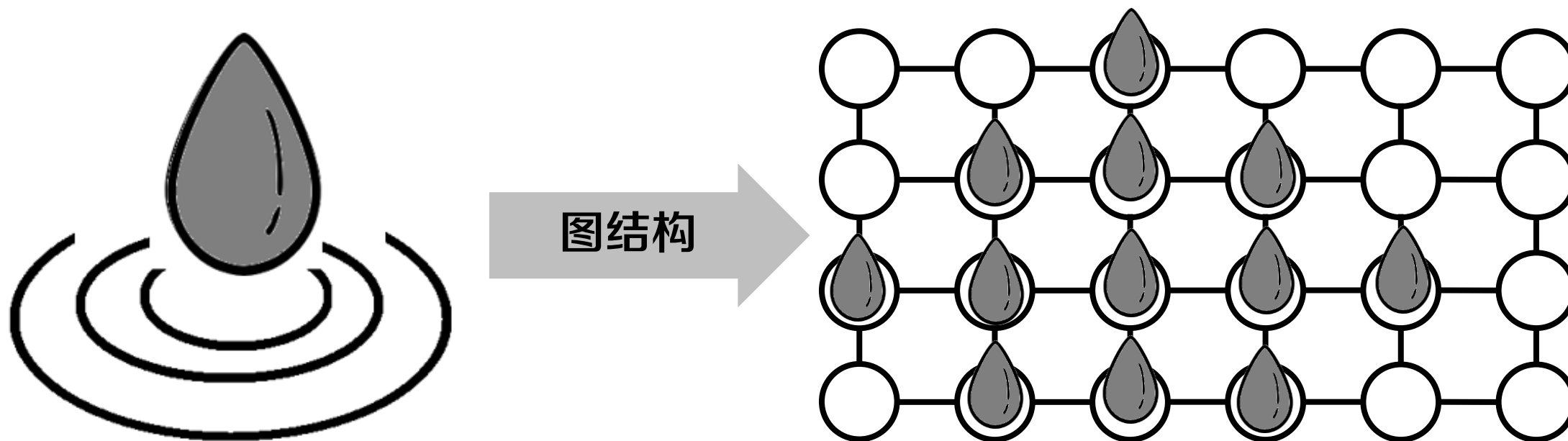
- 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



图结构

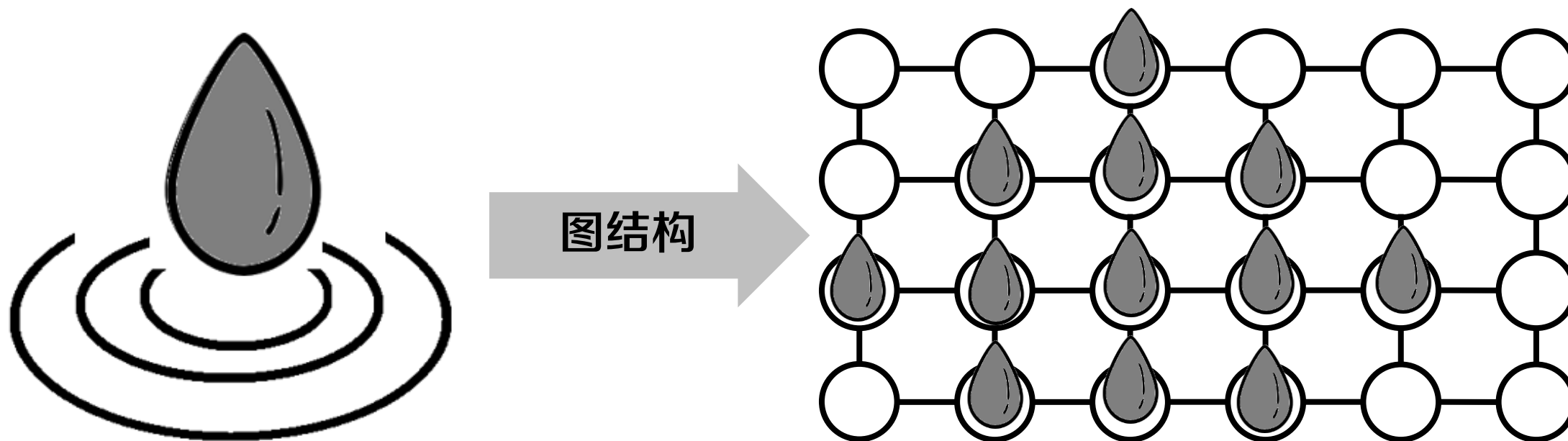


- 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



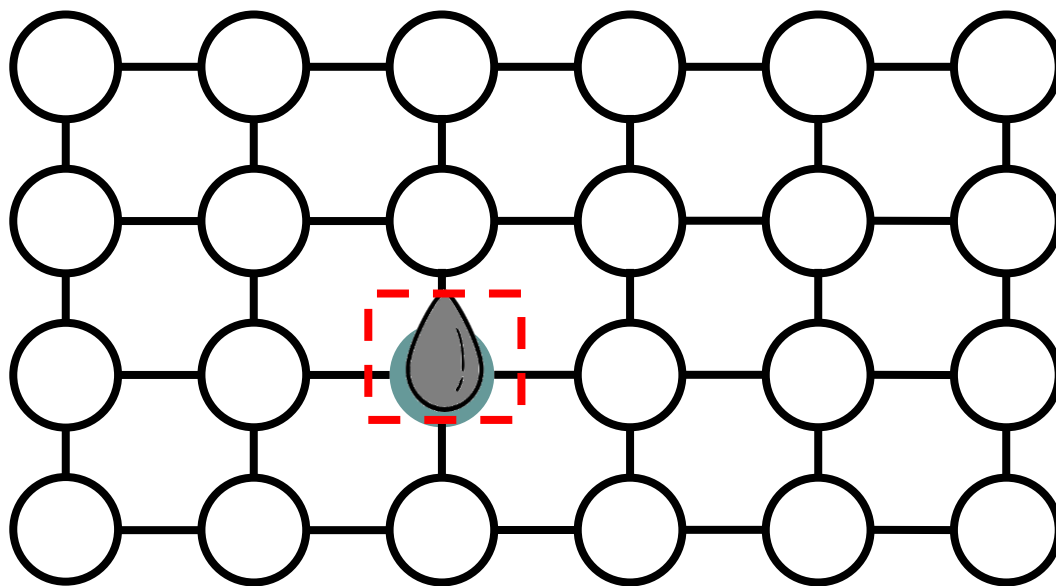
这种依次扩散的现象，蕴含了搜索图结构的一种顺序

- 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散

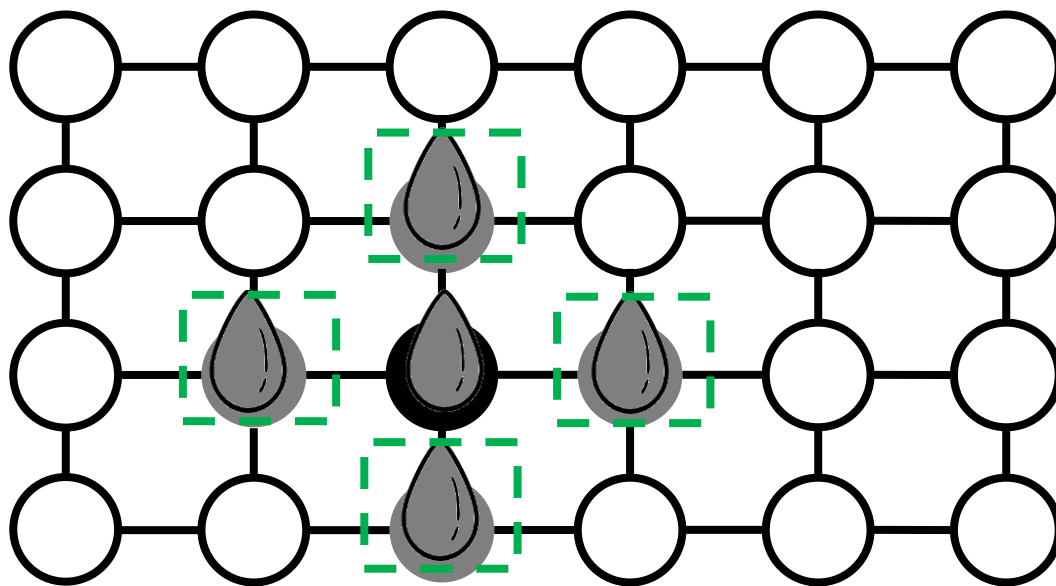


处理某顶点时，一次性发现所有其相邻顶点

- 核心思想
 - 处理某顶点时，一次性发现其所有相邻顶点

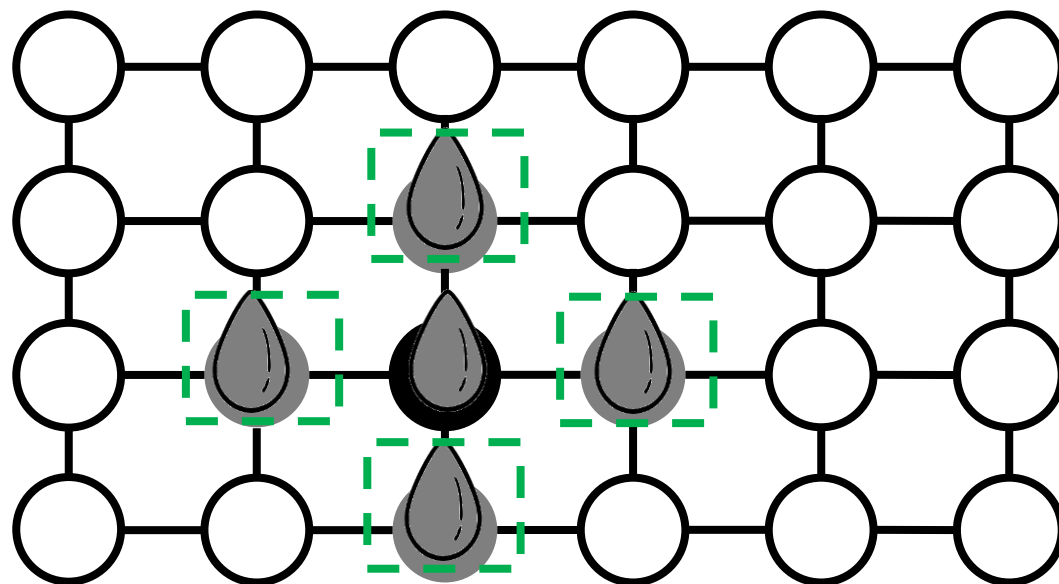


- 核心思想
 - 处理某顶点时，一次性发现其所有相邻顶点



- 核心思想
 - 处理某顶点时，一次性发现其所有相邻顶点

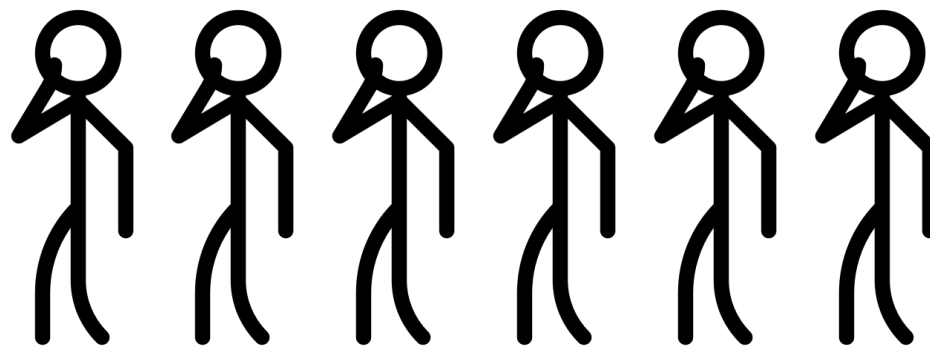
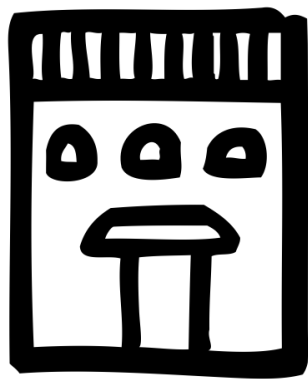
扩散中多处同时进行，一次只能处理单个顶点怎么办？



- 队列

- 先来先服务：队尾加入，队首离开

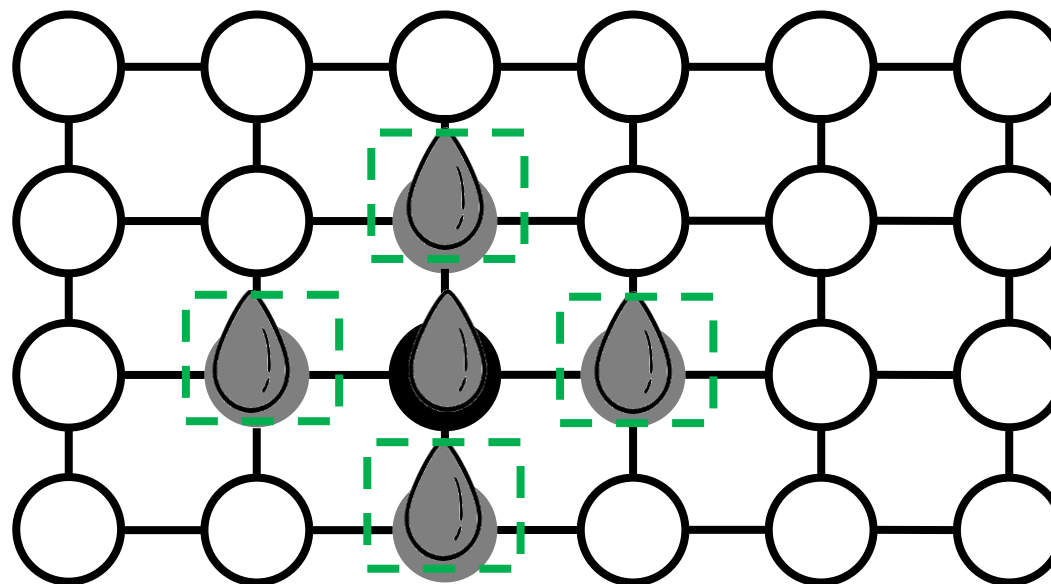
- 加入队列，*Q.Enqueue()*
- 离开队列，*Q.Dequeue()*



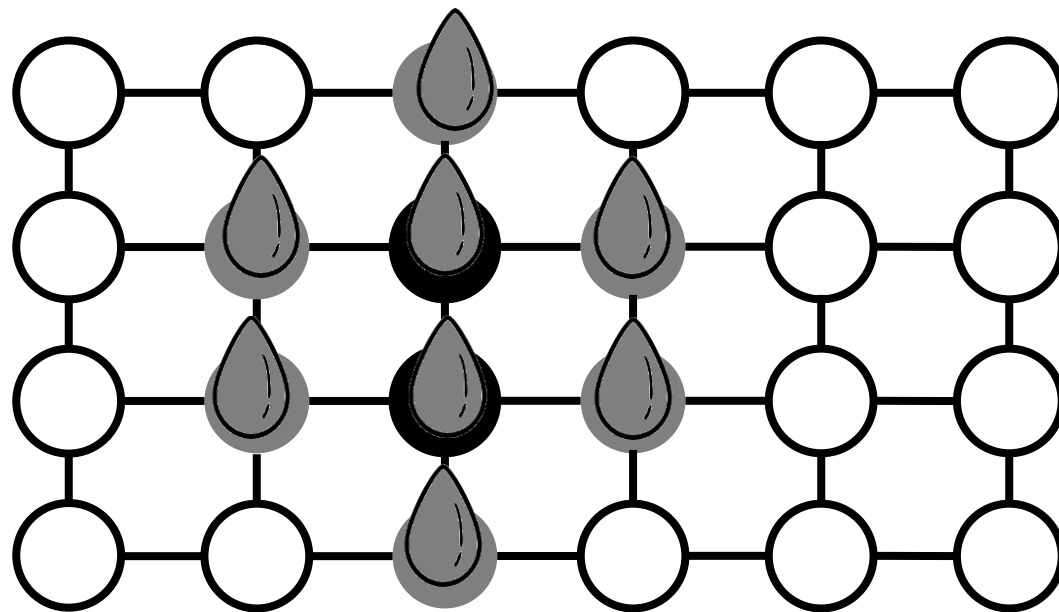
队列 Q

- 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**



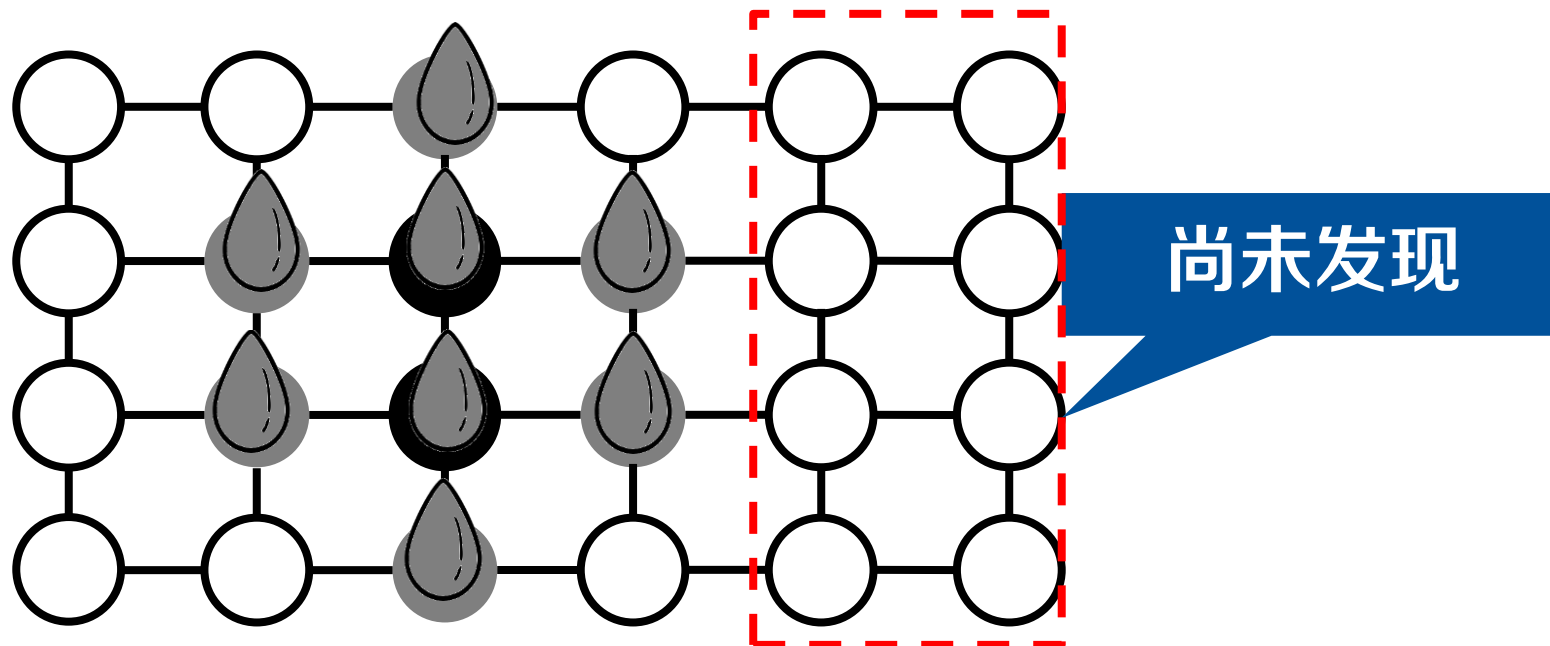
- 辅助数组
 - *color*表示顶点状态



算法思想：广度优先搜索



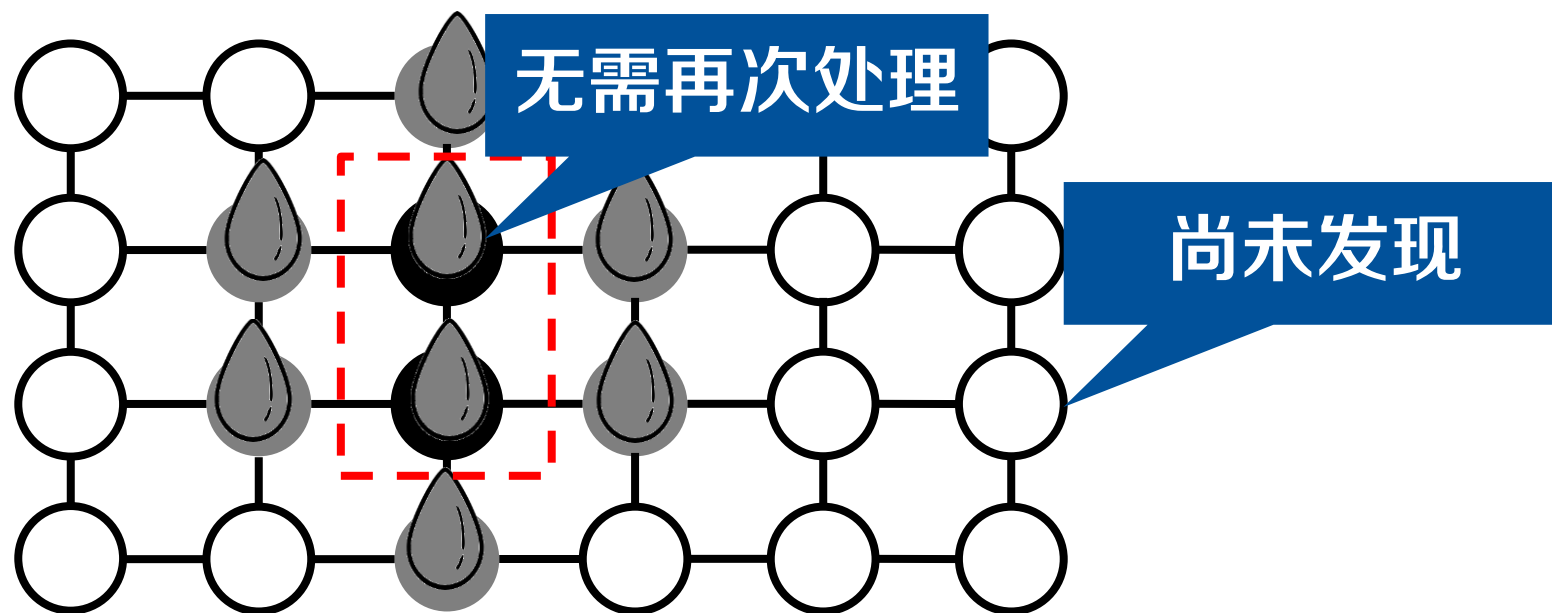
- 辅助数组
 - *color*表示顶点状态
 - *White*: 白色顶点 u 尚未被发现，发现后直接入队



- 辅助数组

- *color*表示顶点状态

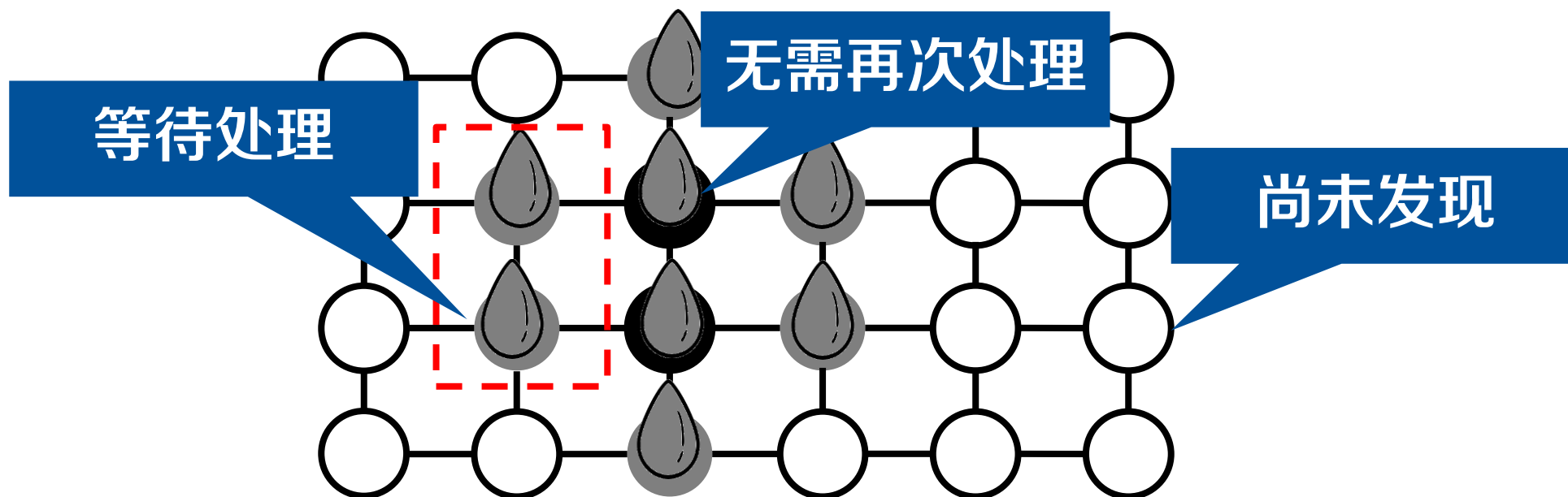
- *White*: 白色顶点 u 尚未被发现，发现后直接入队
 - *Black*: 黑色顶点 u 已被处理，无需再次入队



- 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

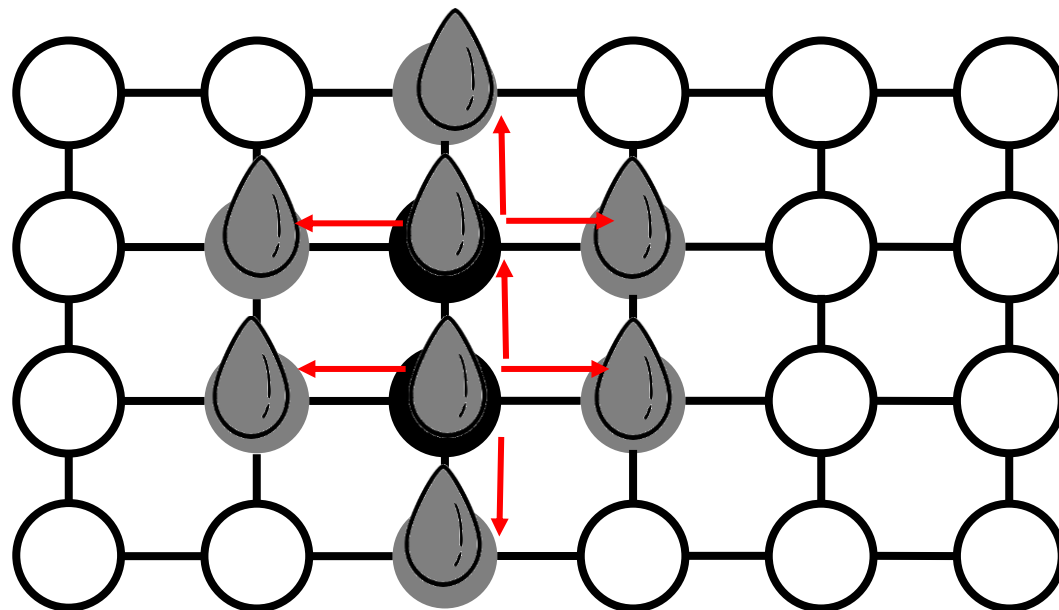


- 辅助数组

- *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



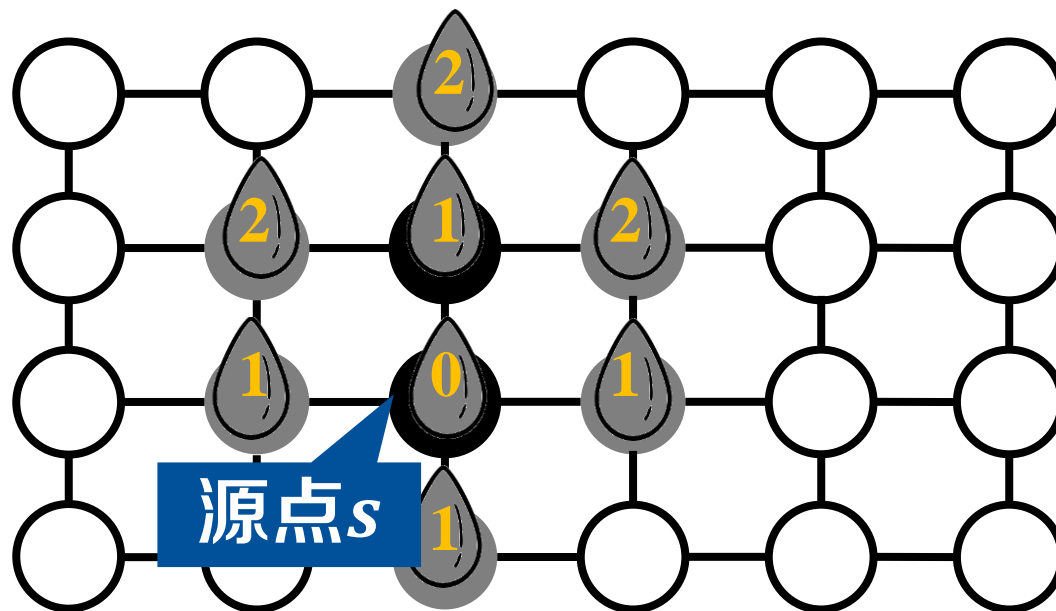
- 辅助数组

- *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



图的搜索

算法思想

算法实例

算法分析

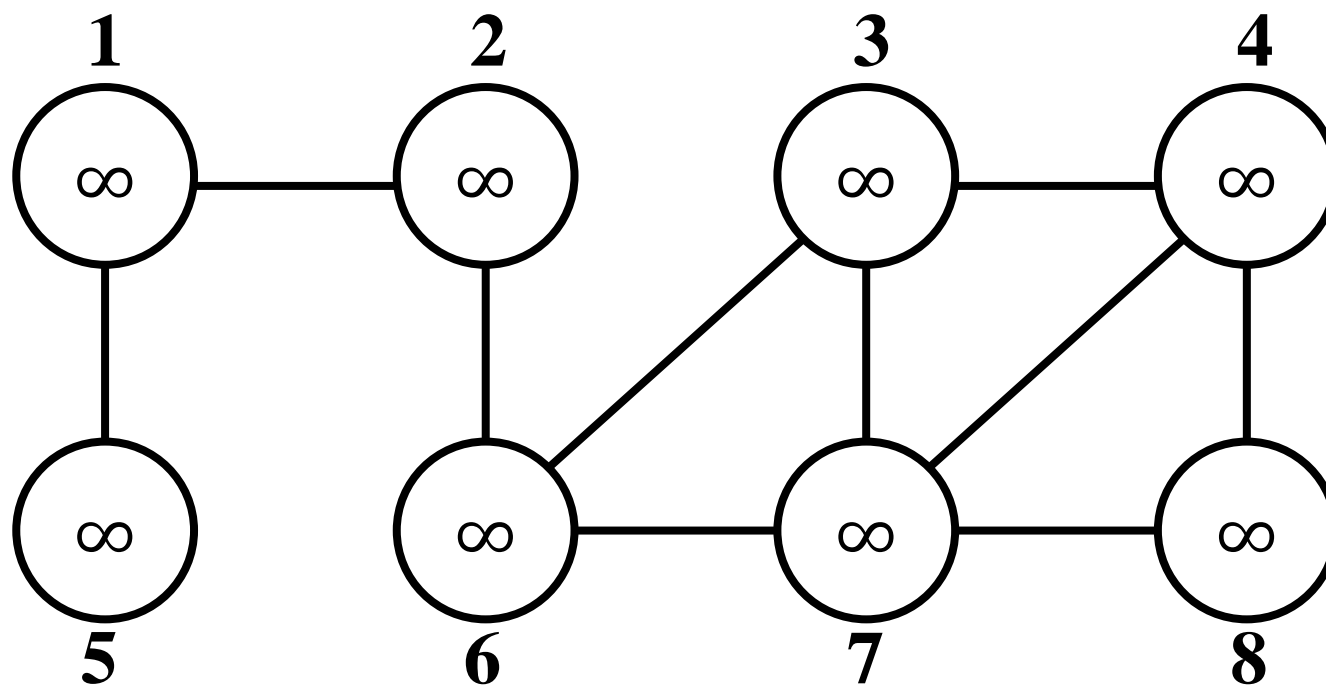
算法应用

算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	W	W	W	W	W	W	W	W
<i>pred</i>	N	N	N	N	N	N	N	N
<i>dist</i>	∞	∞	∞	∞	∞	∞	∞	∞

待处理队列



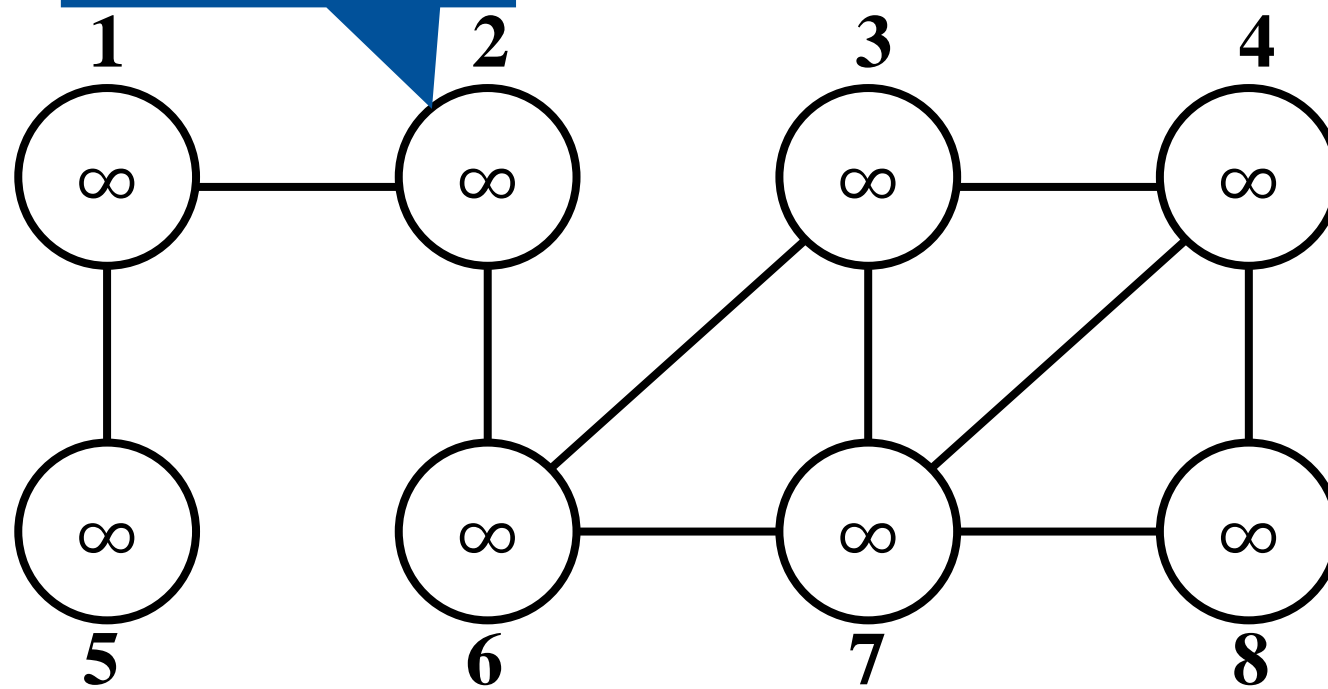
算法实例



V	1	2	3	4	5	6	7	8
<i>color</i>	W	W	W	W	W	W	W	W
<i>pred</i>	N	N	N	N	N	N	N	N
<i>dist</i>	∞	∞	∞	∞	∞	∞	∞	∞

待处理队列

搜索源点



算法实例

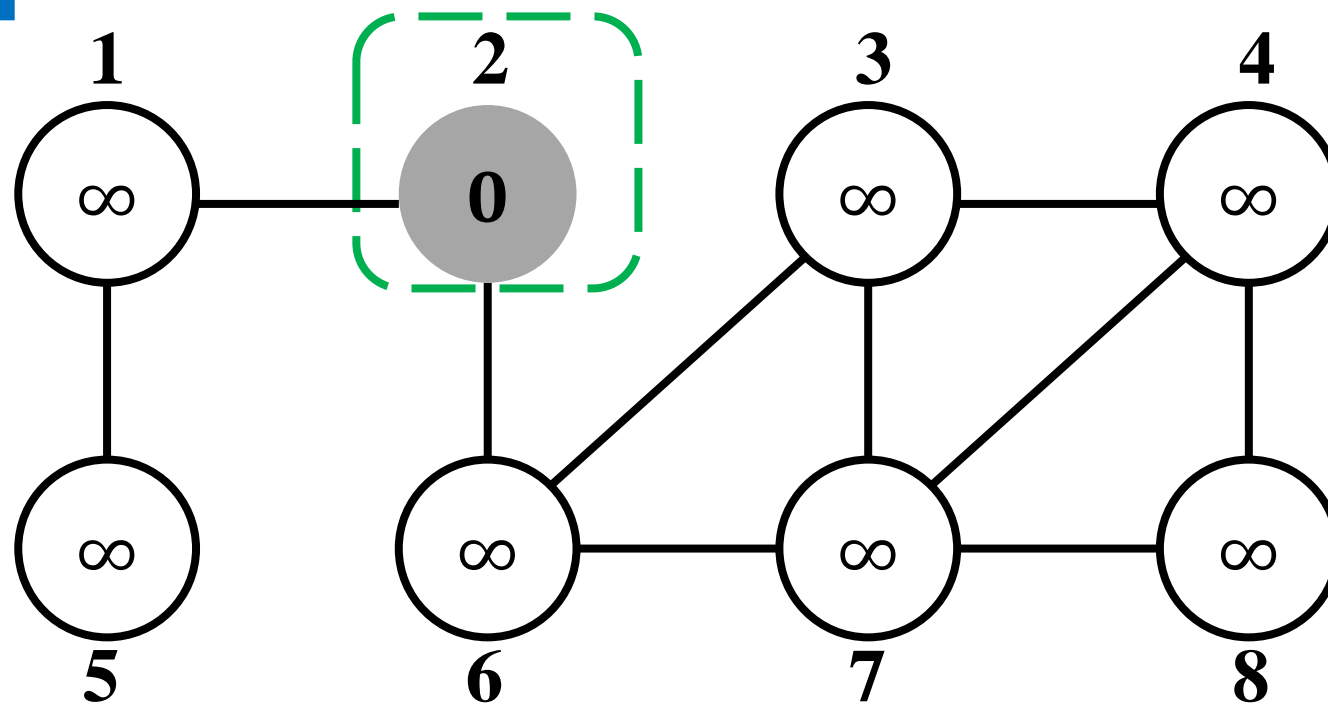


V	1	2	3	4	5	6	7	8
<i>color</i>	W	G	W	W	W	W	W	W
<i>pred</i>	N	N	N	N	N	N	N	N
<i>dist</i>	∞	0	∞	∞	∞	∞	∞	∞

待处理队列

2

源点2



算法实例

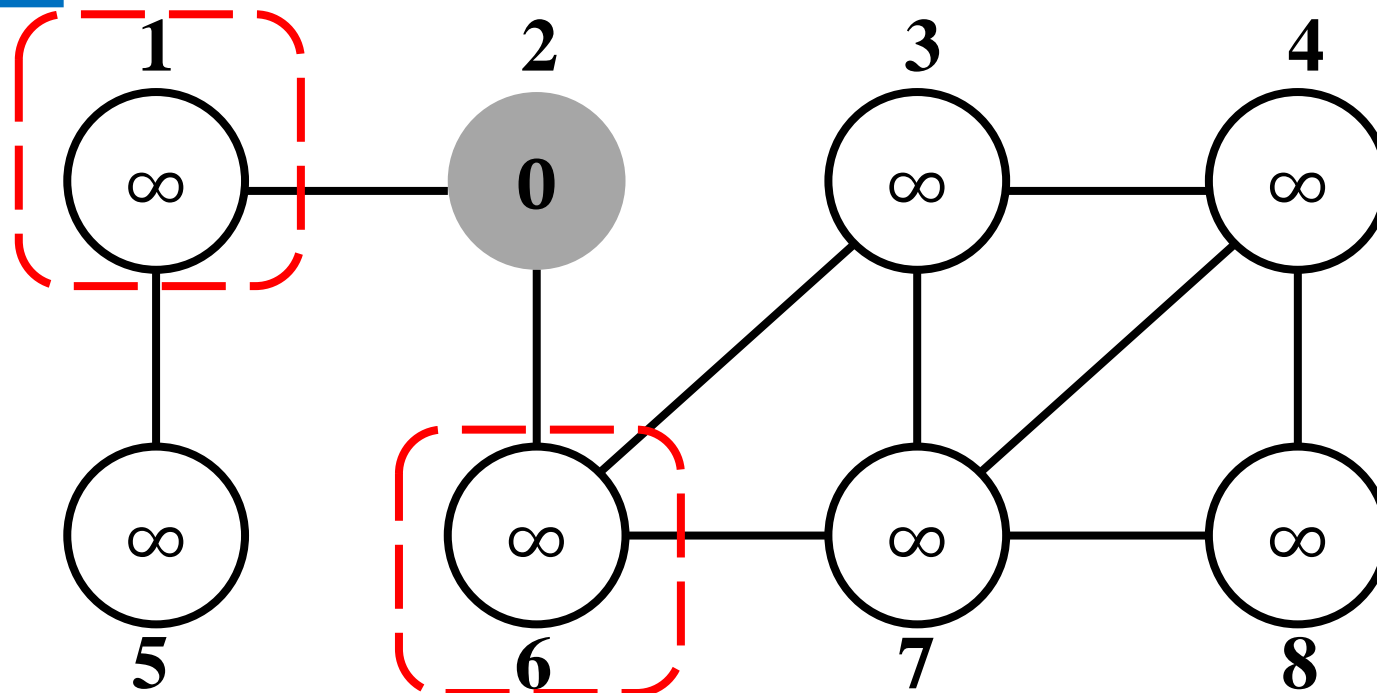


<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	W	G	W	W	W	W	W	W
<i>pred</i>	N	N	N	N	N	N	N	N
<i>dist</i>	∞	0	∞	∞	∞	∞	∞	∞

待处理队列

2

源点2



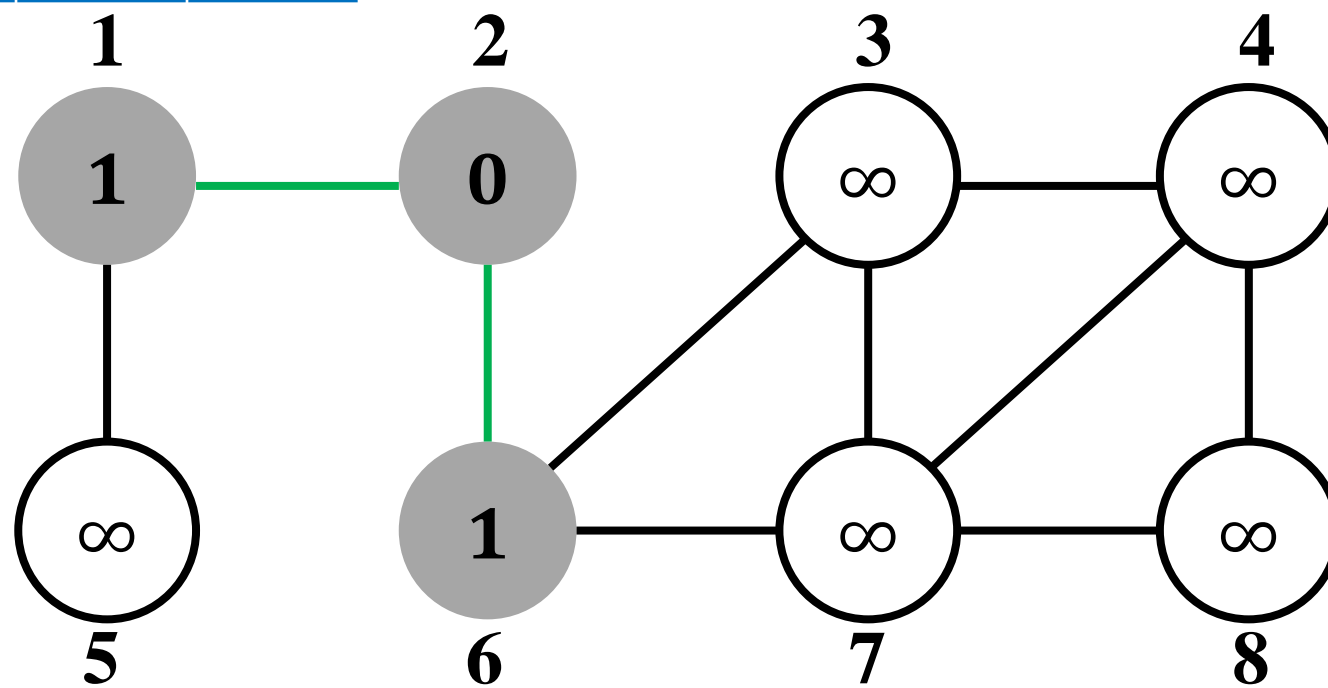
算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	G	G	W	W	W	G	W	W
<i>pred</i>	2	N	N	N	N	2	N	N
<i>dist</i>	1	0	∞	∞	∞	1	∞	∞

待处理队列	2	1	6
-------	----------	----------	----------

源点2



算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	G	B	W	W	W	G	W	W
<i>pred</i>	2	N	N	N	N	2	N	N
<i>dist</i>	1	0	∞	∞	∞	1	∞	∞

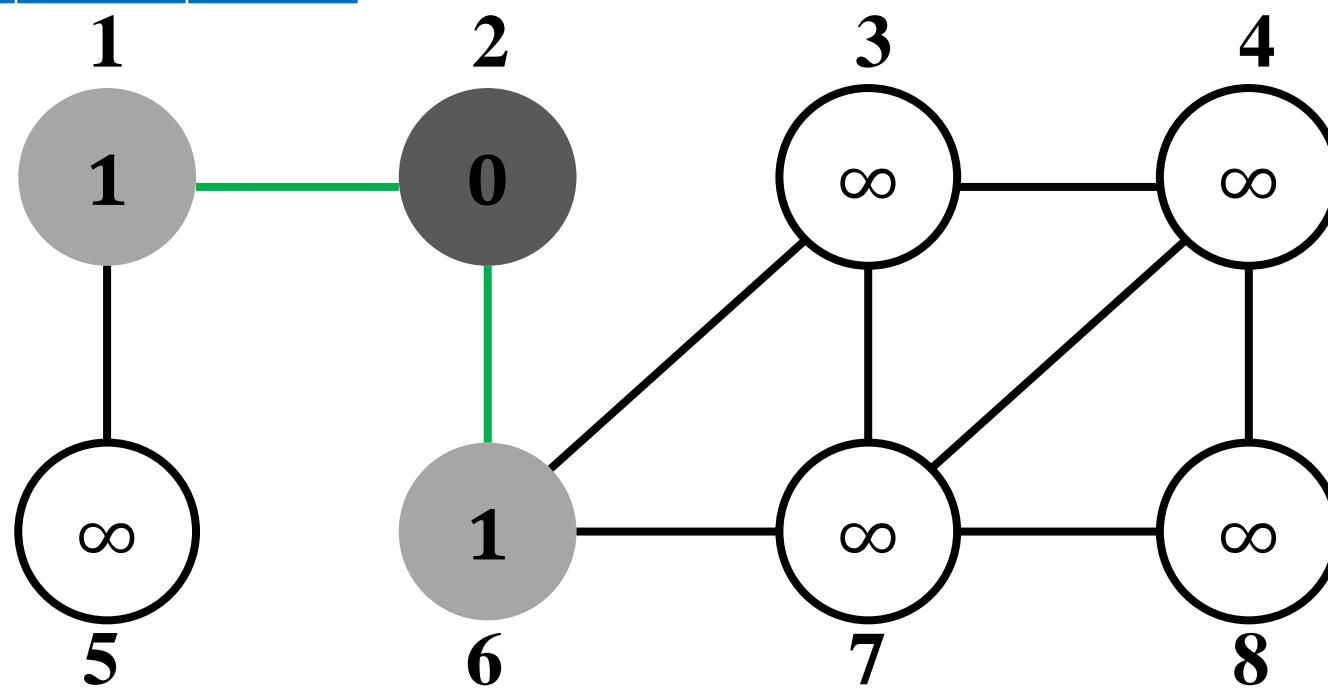
待处理队列

2

1

6

源点2



算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	G	B	W	W	W	G	W	W
<i>pred</i>	2	N	N	N	N	2	N	N
<i>dist</i>	1	0	∞	∞	∞	1	∞	∞

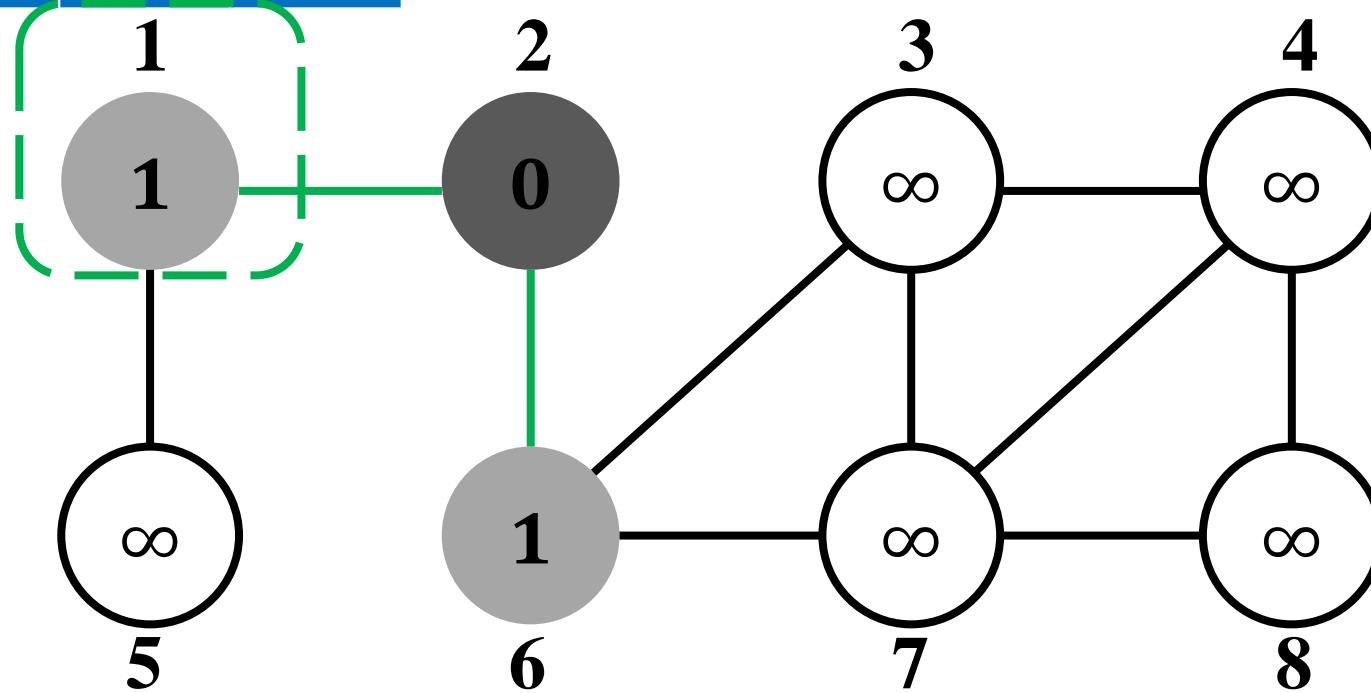
待处理队列

2

1

6

源点2



算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	G	B	W	W	W	G	W	W
<i>pred</i>	2	N	N	N	N	2	N	N
<i>dist</i>	1	0	∞	∞	∞	1	∞	∞

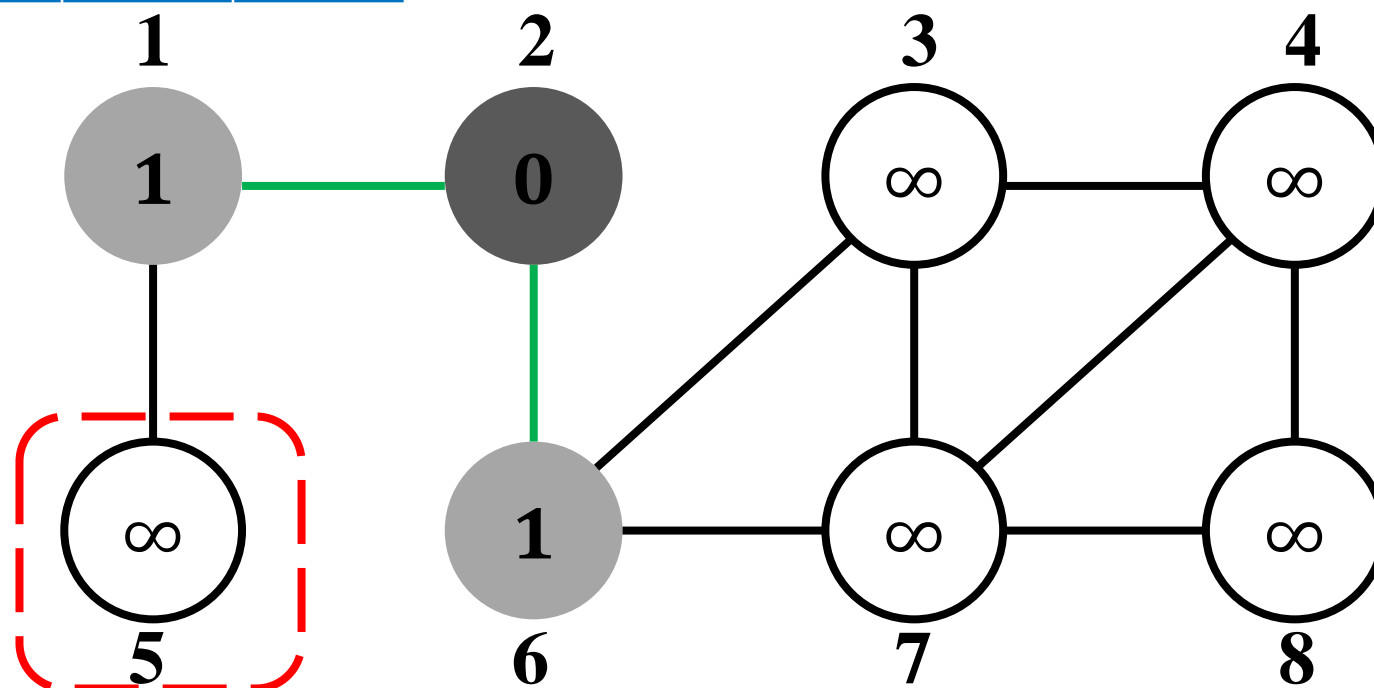
待处理队列

2

1

6

源点2



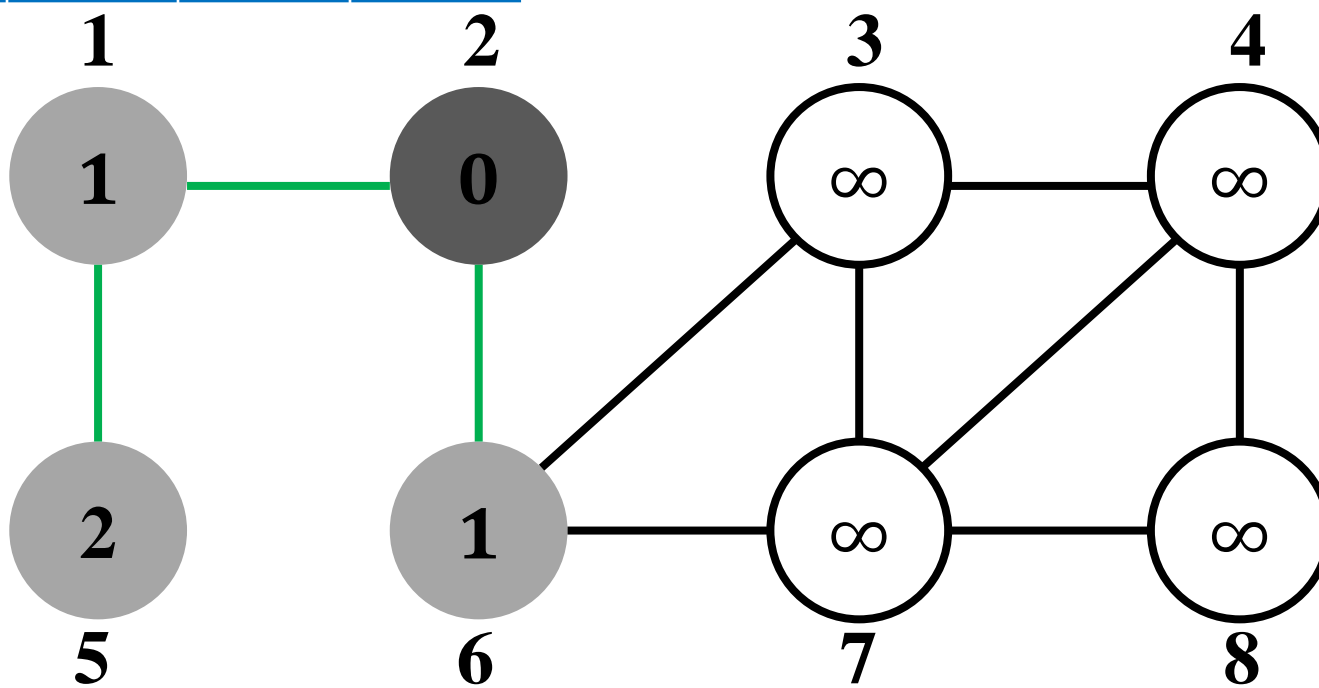
算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	G	B	W	W	G	G	W	W
<i>pred</i>	2	N	N	N	1	2	N	N
<i>dist</i>	1	0	∞	∞	2	1	∞	∞

待处理队列	2	1	6	5
-------	---	---	---	---

源点2



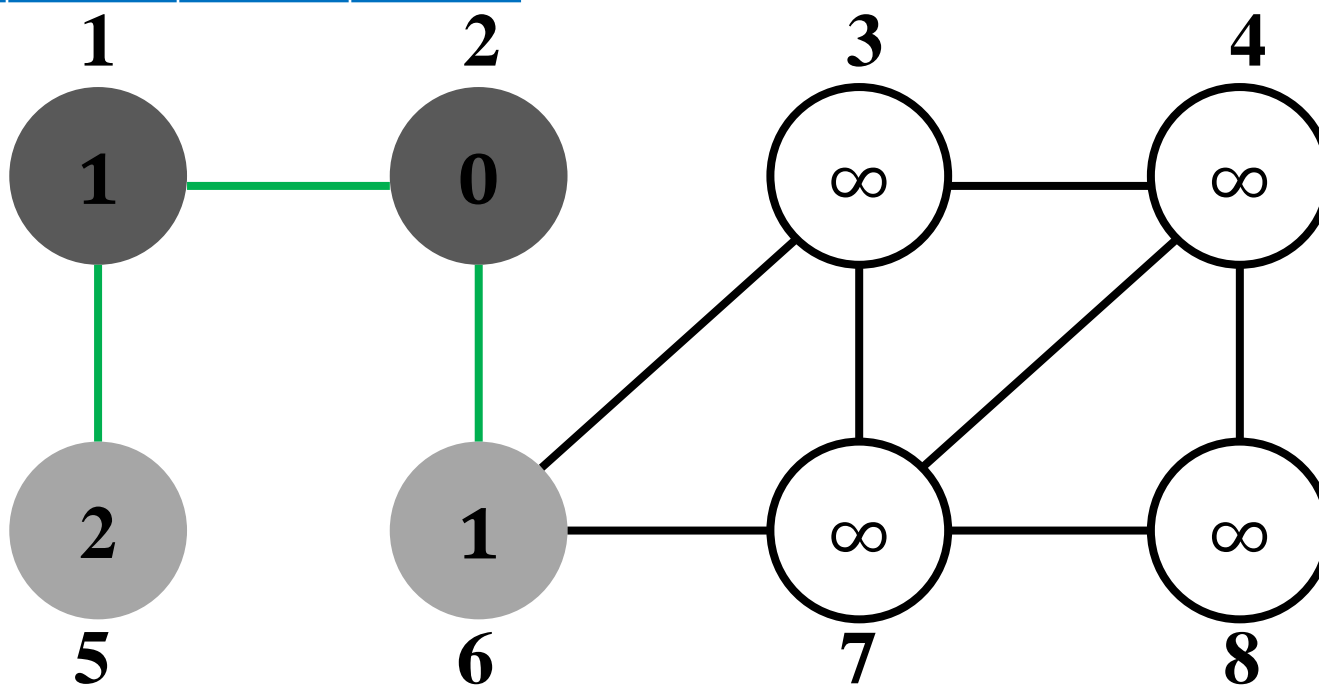
算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	B	B	W	W	G	G	W	W
<i>pred</i>	2	N	N	N	1	2	N	N
<i>dist</i>	1	0	∞	∞	2	1	∞	∞

待处理队列	2	1	6	5
-------	---	---	---	---

源点2



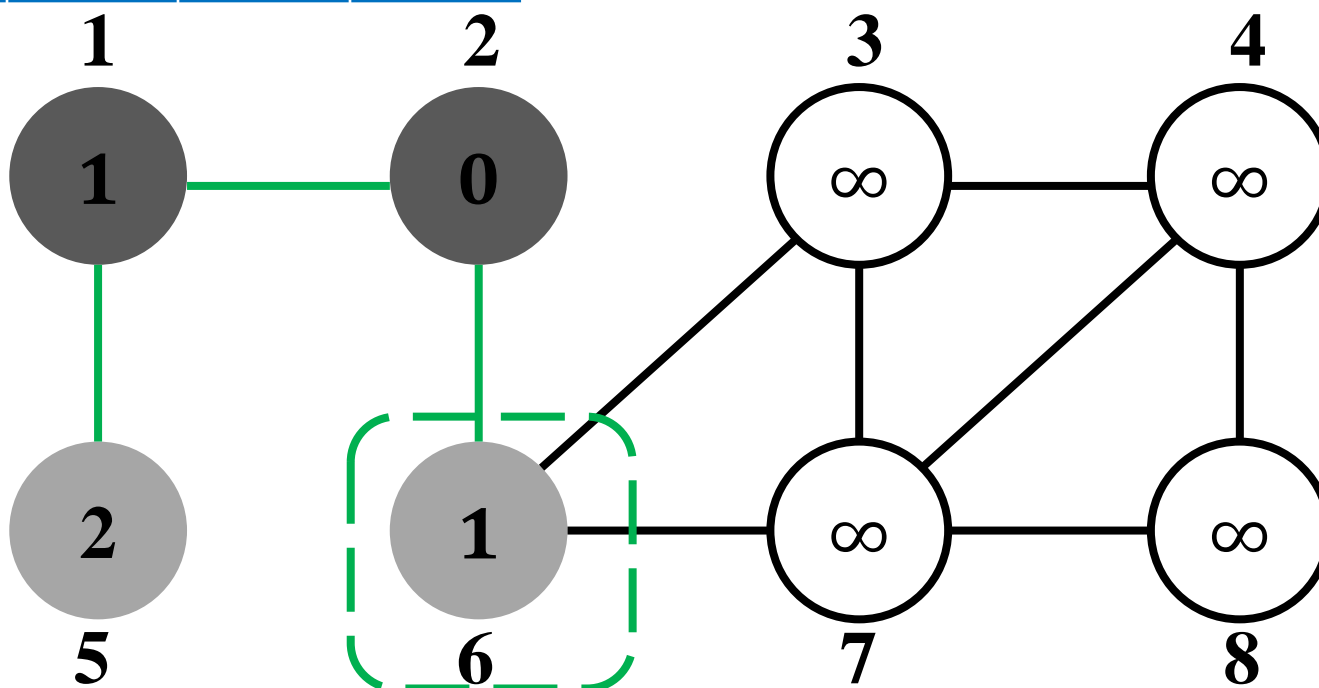
算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	B	B	W	W	G	G	W	W
<i>pred</i>	2	N	N	N	1	2	N	N
<i>dist</i>	1	0	∞	∞	2	1	∞	∞

待处理队列	2	1	6	5
-------	---	---	---	---

源点2



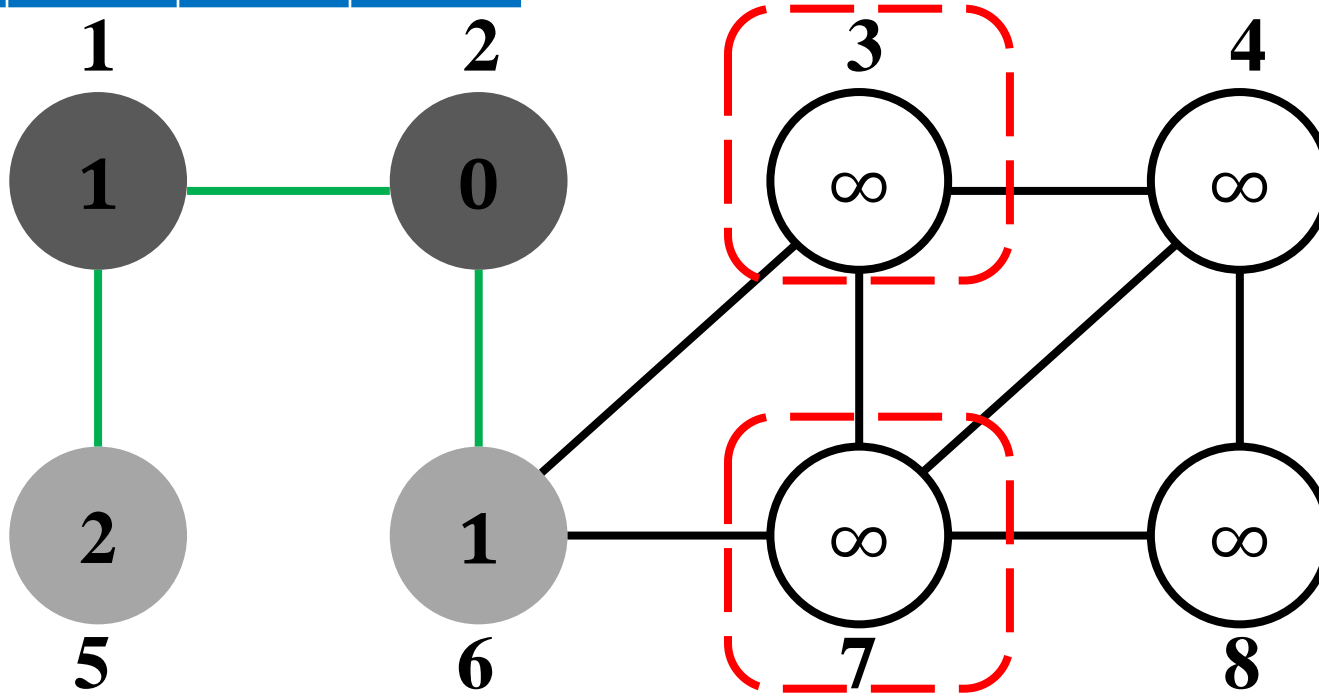
算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	B	B	W	W	G	G	W	W
<i>pred</i>	2	N	N	N	1	2	N	N
<i>dist</i>	1	0	∞	∞	2	1	∞	∞

待处理队列	2	1	6	5
-------	---	---	---	---

源点2



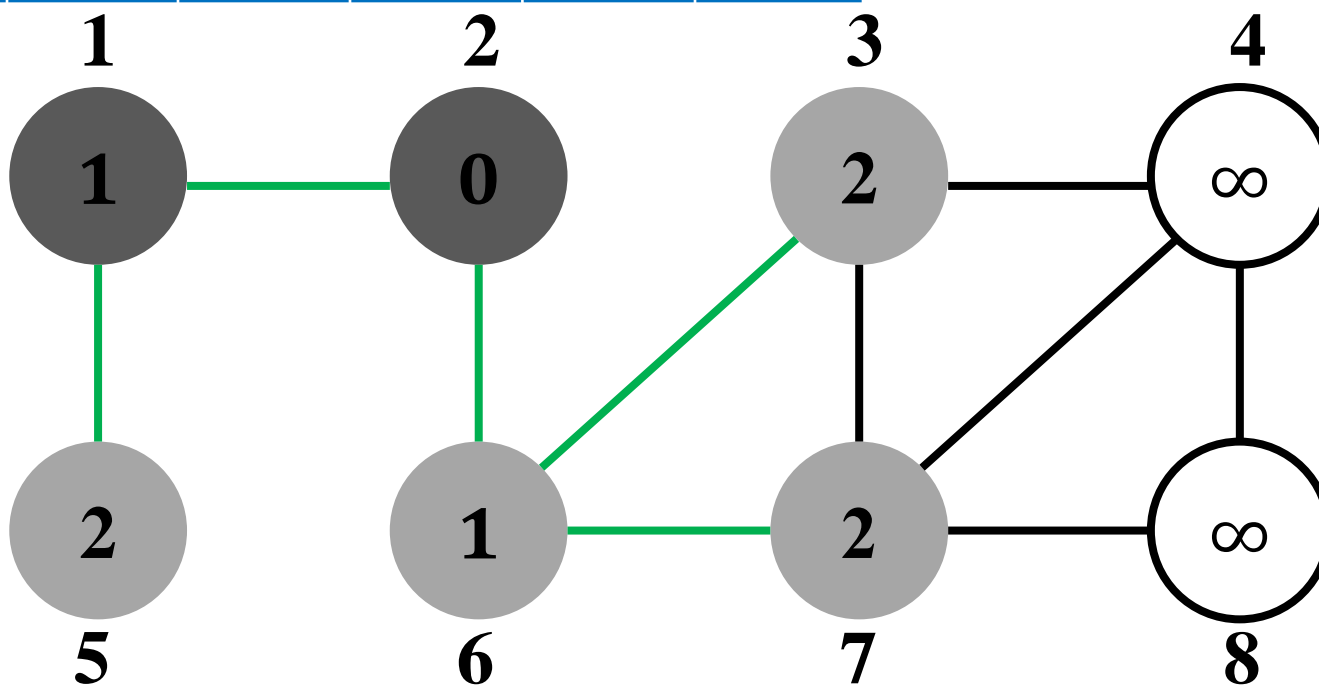
算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	B	B	G	W	G	G	G	W
<i>pred</i>	2	N	6	N	1	2	6	N
<i>dist</i>	1	0	2	∞	2	1	2	∞

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

源点2



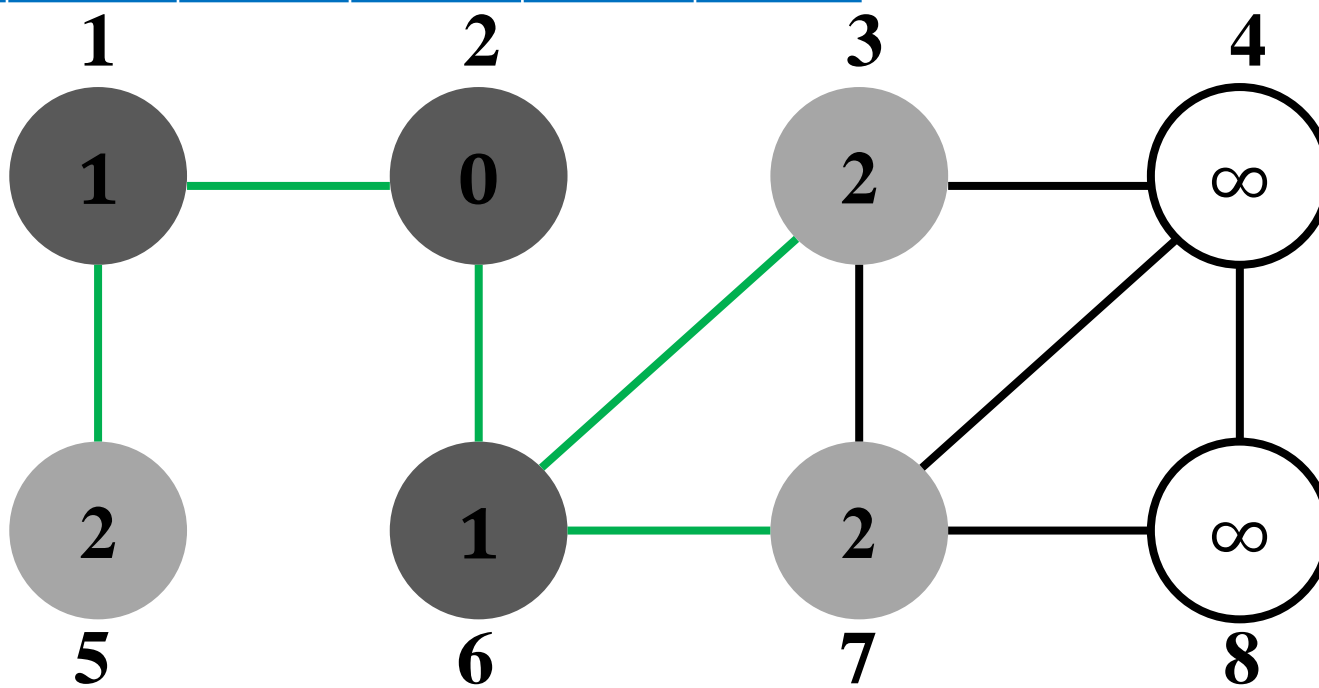
算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	B	B	G	W	G	B	G	W
<i>pred</i>	2	N	6	N	1	2	6	N
<i>dist</i>	1	0	2	∞	2	1	2	∞

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

源点2



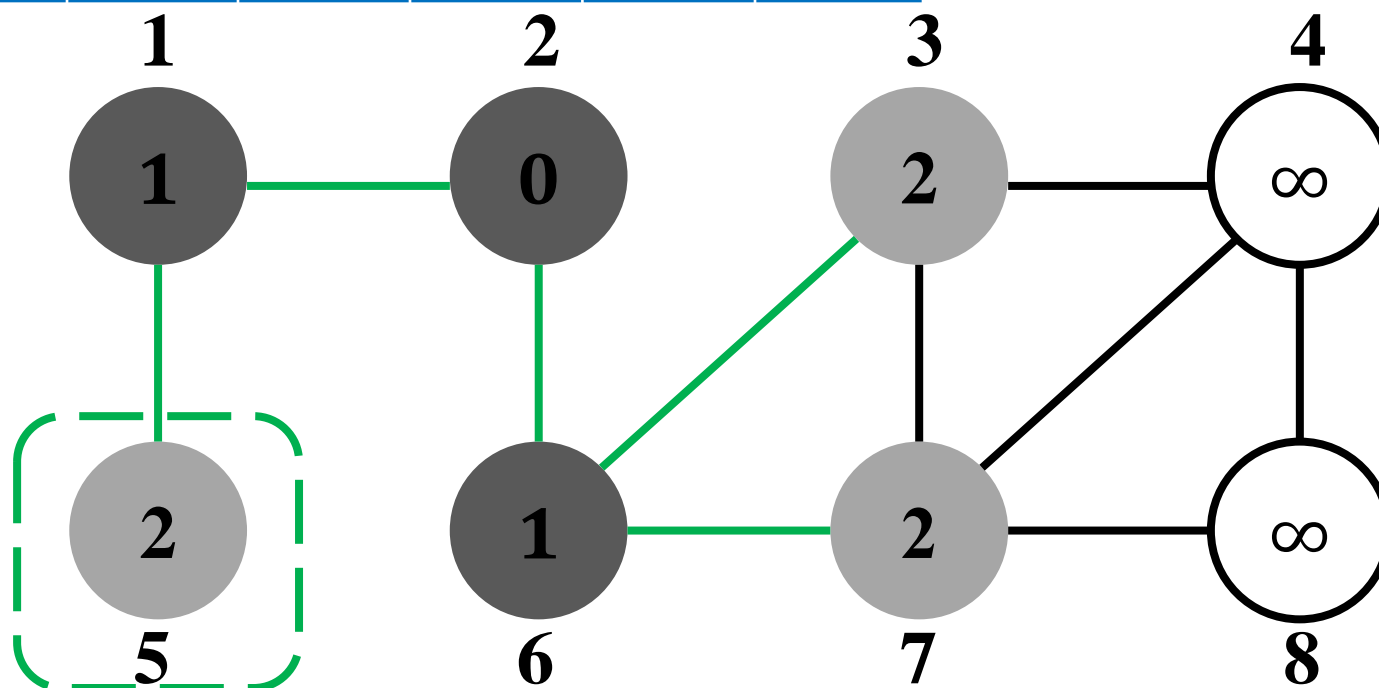
算法实例



V	1	2	3	4	5	6	7	8
<i>color</i>	B	B	G	W	G	B	G	W
<i>pred</i>	2	N	6	N	1	2	6	N
<i>dist</i>	1	0	2	∞	2	1	2	∞

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

源点2



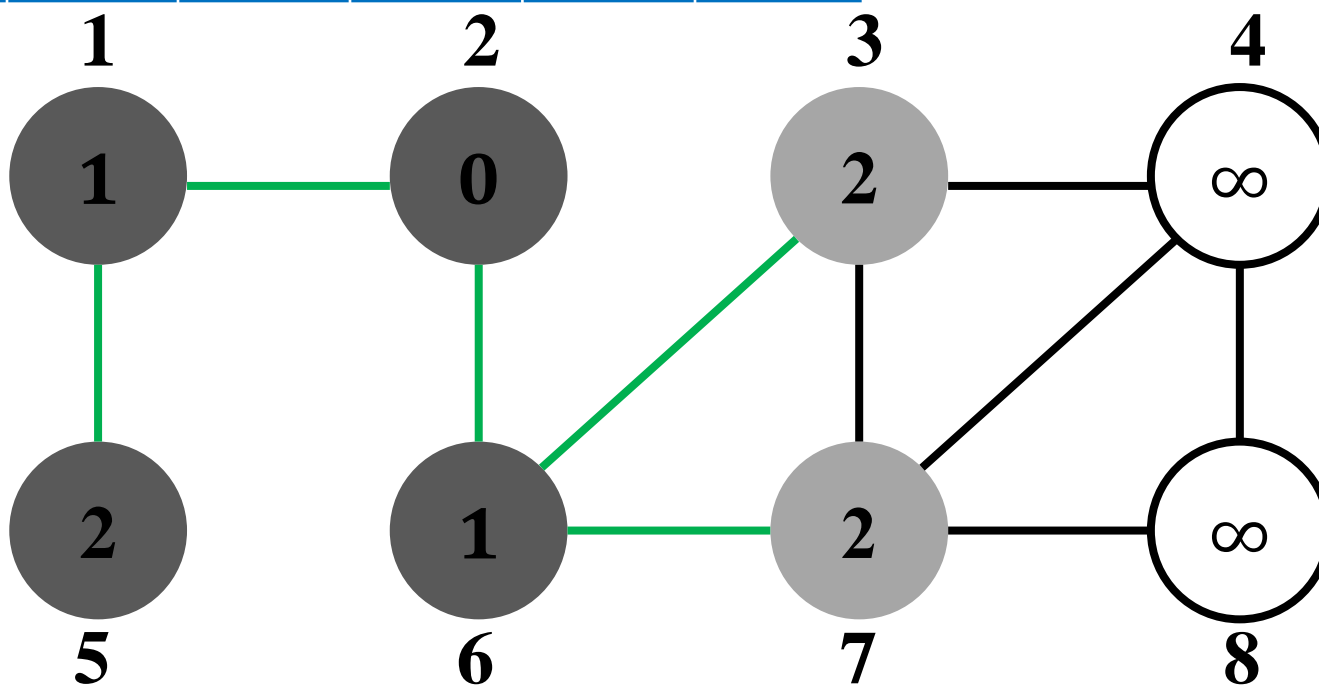
算法实例



V	1	2	3	4	5	6	7	8
<i>color</i>	B	B	G	W	B	B	G	W
<i>pred</i>	2	N	6	N	1	2	6	N
<i>dist</i>	1	0	2	∞	2	1	2	∞

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

源点2



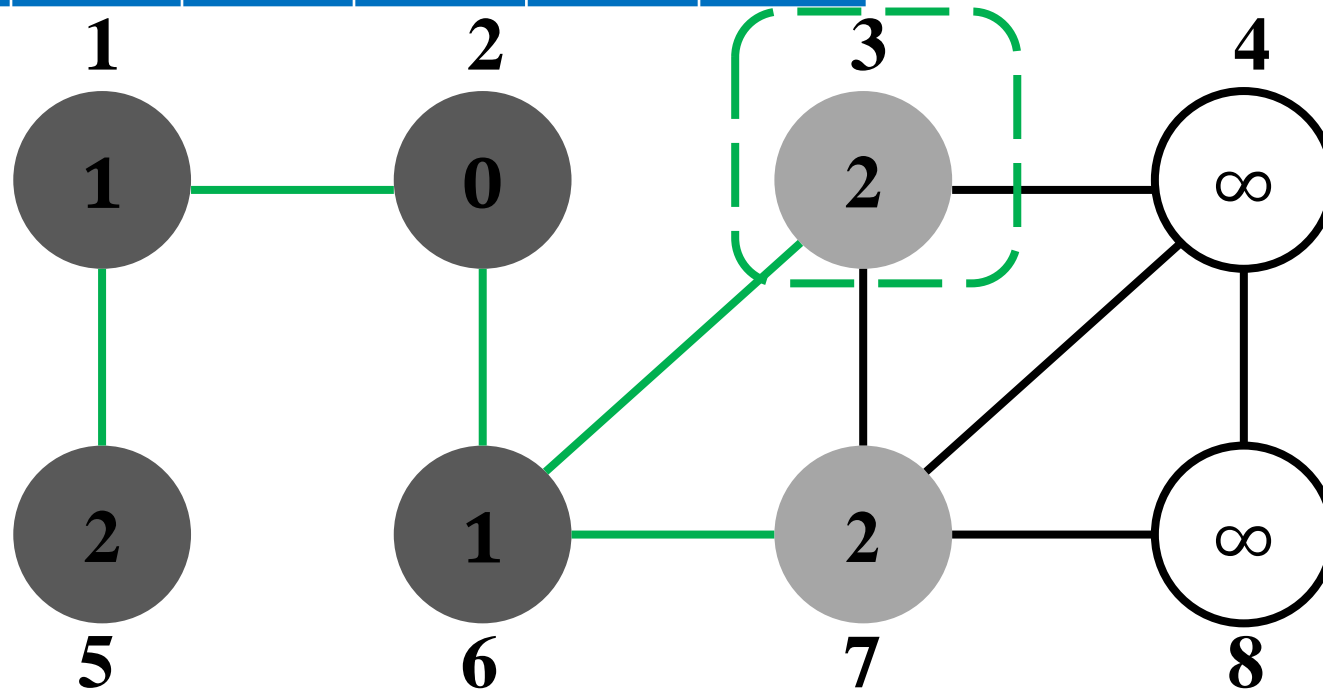
算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	B	B	G	W	B	B	G	W
<i>pred</i>	2	N	6	N	1	2	6	N
<i>dist</i>	1	0	2	∞	2	1	2	∞

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

源点2



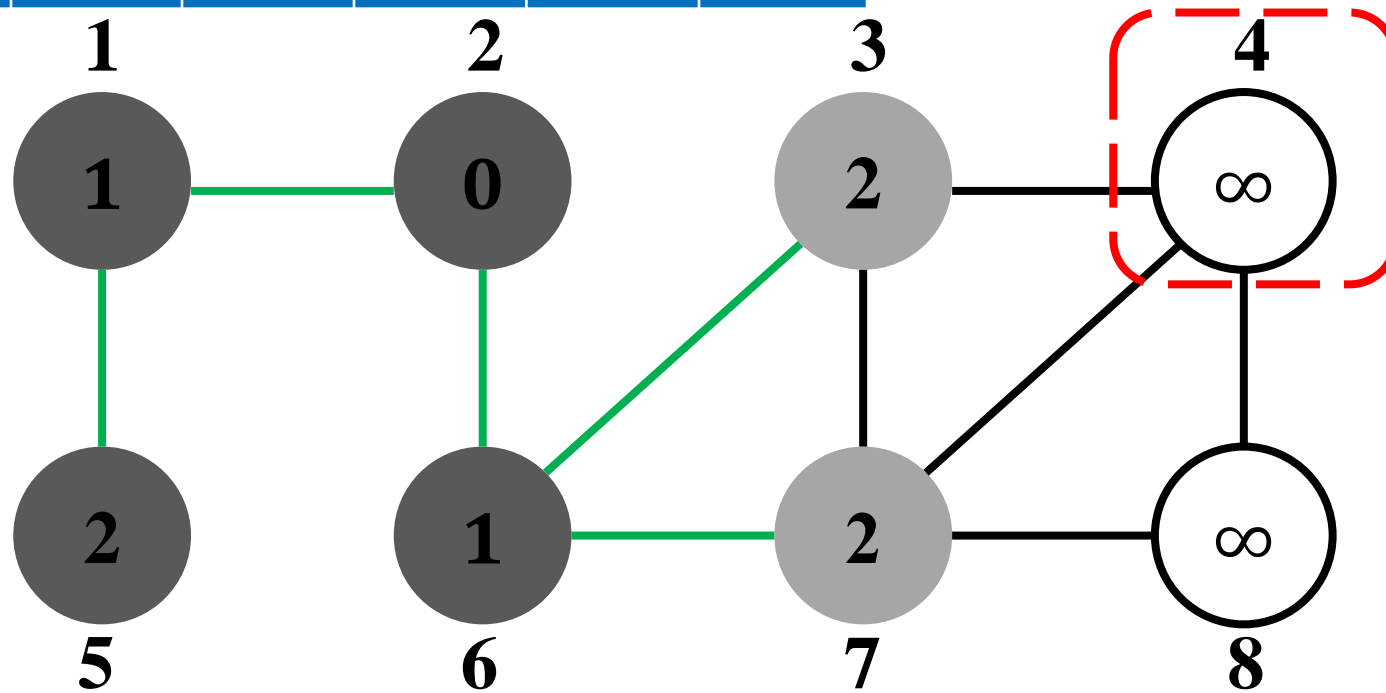
算法实例



V	1	2	3	4	5	6	7	8
<i>color</i>	B	B	G	W	B	B	G	W
<i>pred</i>	2	N	6	N	1	2	6	N
<i>dist</i>	1	0	2	∞	2	1	2	∞

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

源点2



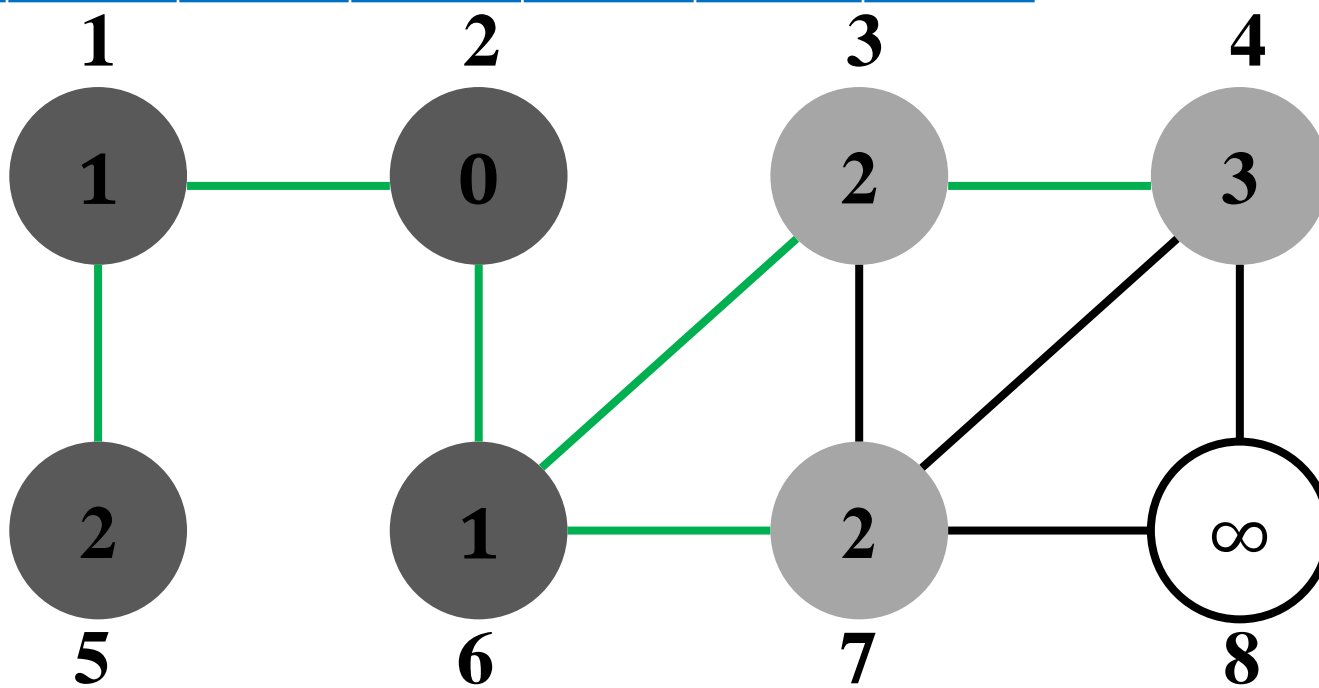
算法实例



V	1	2	3	4	5	6	7	8
<i>color</i>	B	B	G	G	B	B	G	W
<i>pred</i>	2	N	6	3	1	2	6	N
<i>dist</i>	1	0	2	3	2	1	2	∞

待处理队列	2	1	6	5	3	7	4
-------	---	---	---	---	---	---	---

源点2



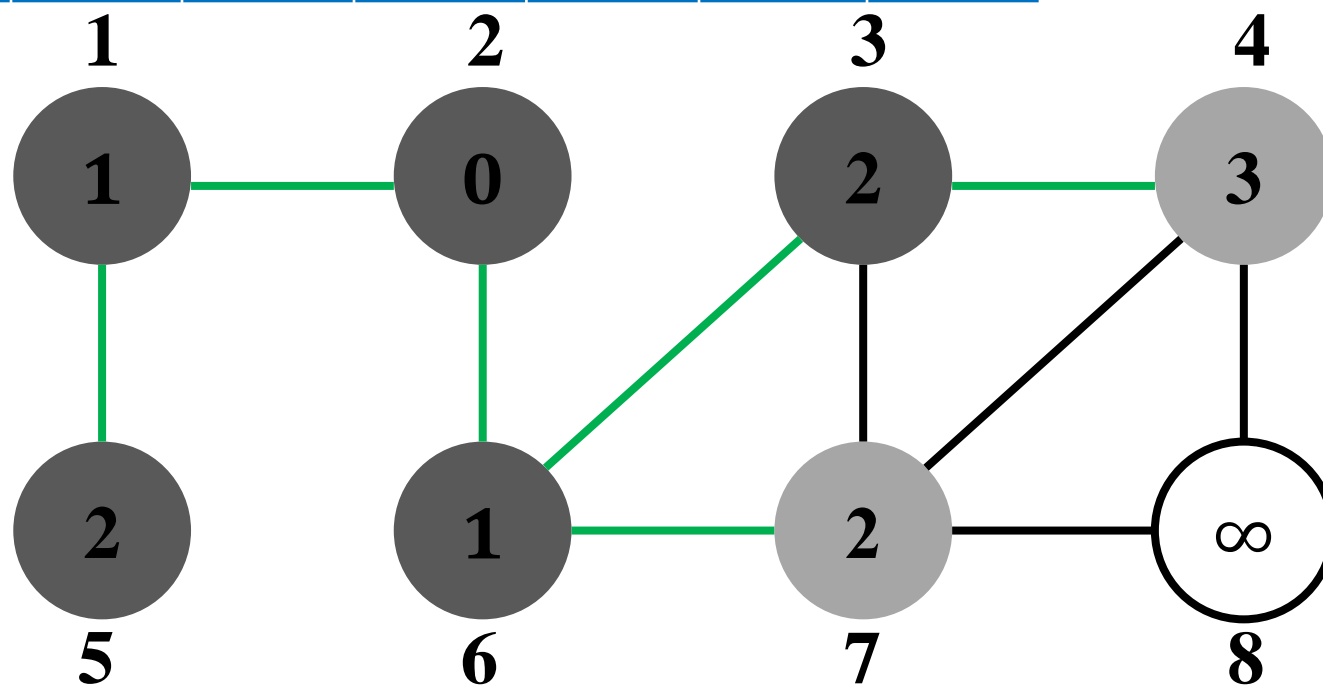
算法实例



V	1	2	3	4	5	6	7	8
<i>color</i>	B	B	B	G	B	B	G	W
<i>pred</i>	2	N	6	3	1	2	6	N
<i>dist</i>	1	0	2	3	2	1	2	∞

待处理队列	2	1	6	5	3	7	4
-------	---	---	---	---	---	---	---

源点2



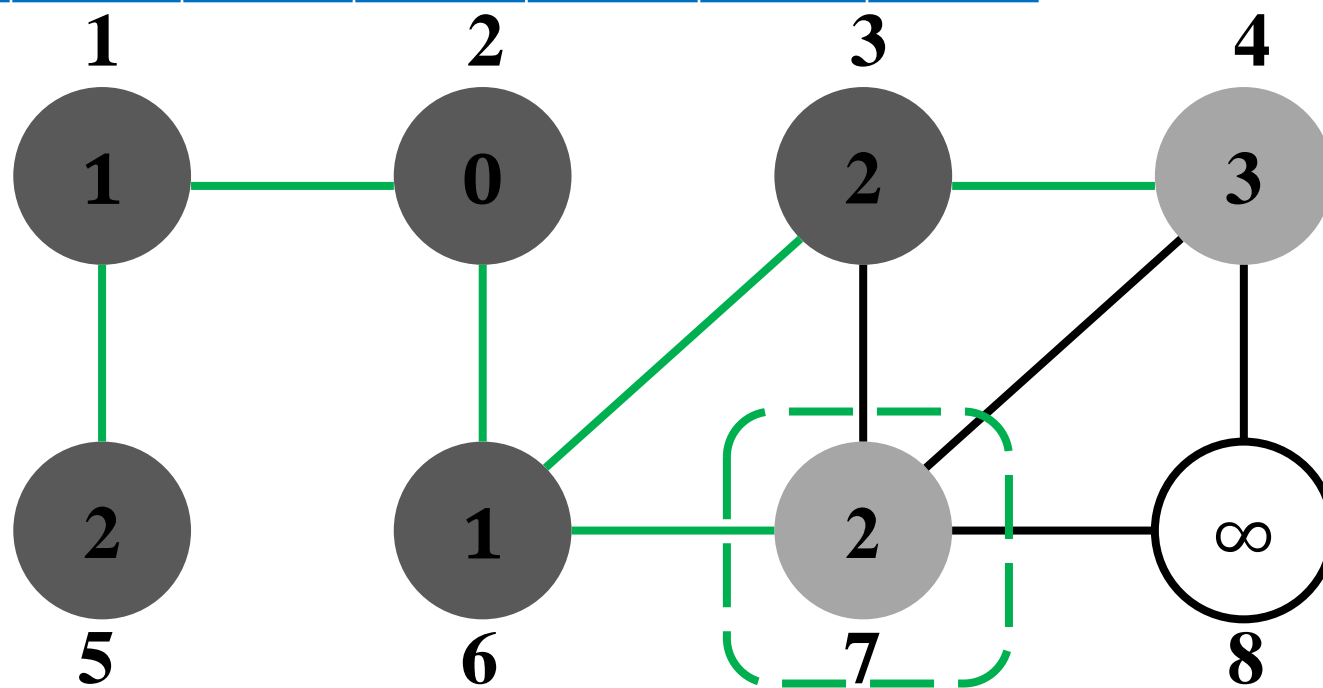
算法实例



V	1	2	3	4	5	6	7	8
<i>color</i>	B	B	B	G	B	B	G	W
<i>pred</i>	2	N	6	3	1	2	6	N
<i>dist</i>	1	0	2	3	2	1	2	∞

待处理队列	2	1	6	5	3	7	4
-------	---	---	---	---	---	---	---

源点2



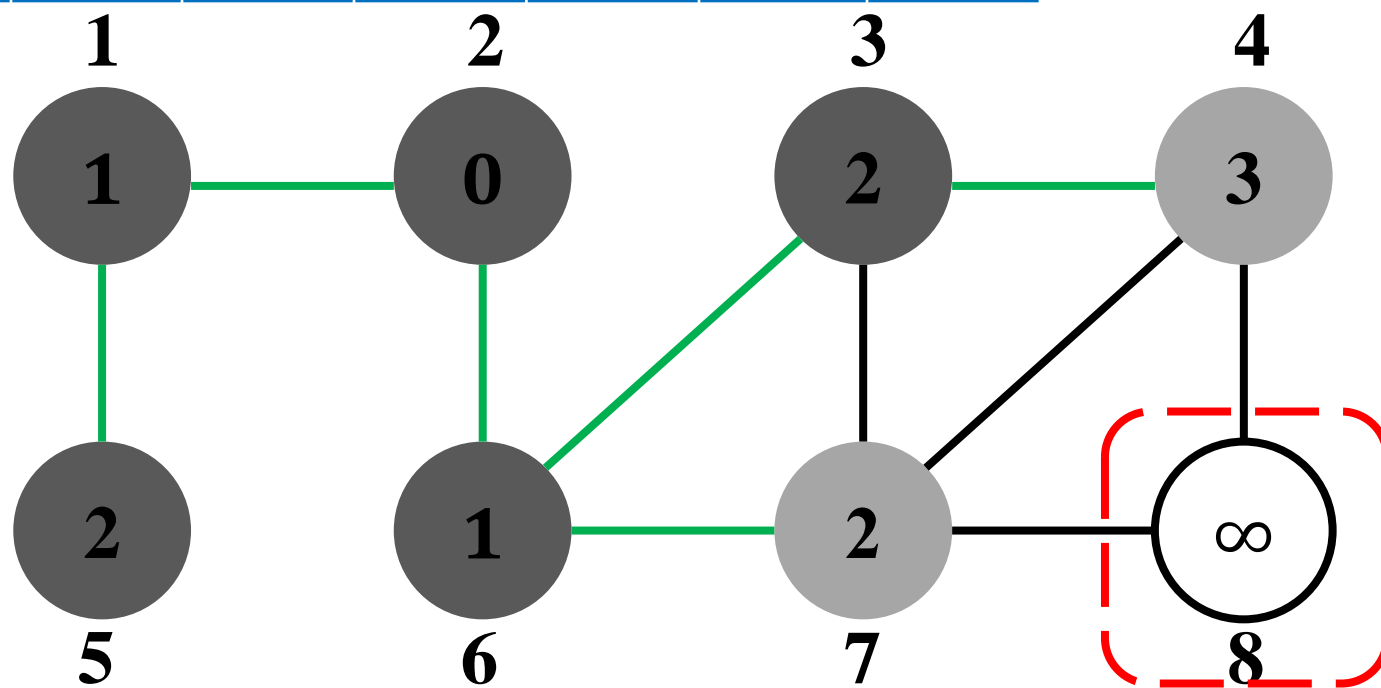
算法实例



V	1	2	3	4	5	6	7	8
<i>color</i>	B	B	B	G	B	B	G	W
<i>pred</i>	2	N	6	3	1	2	6	N
<i>dist</i>	1	0	2	3	2	1	2	∞

待处理队列	2	1	6	5	3	7	4
-------	---	---	---	---	---	---	---

源点2

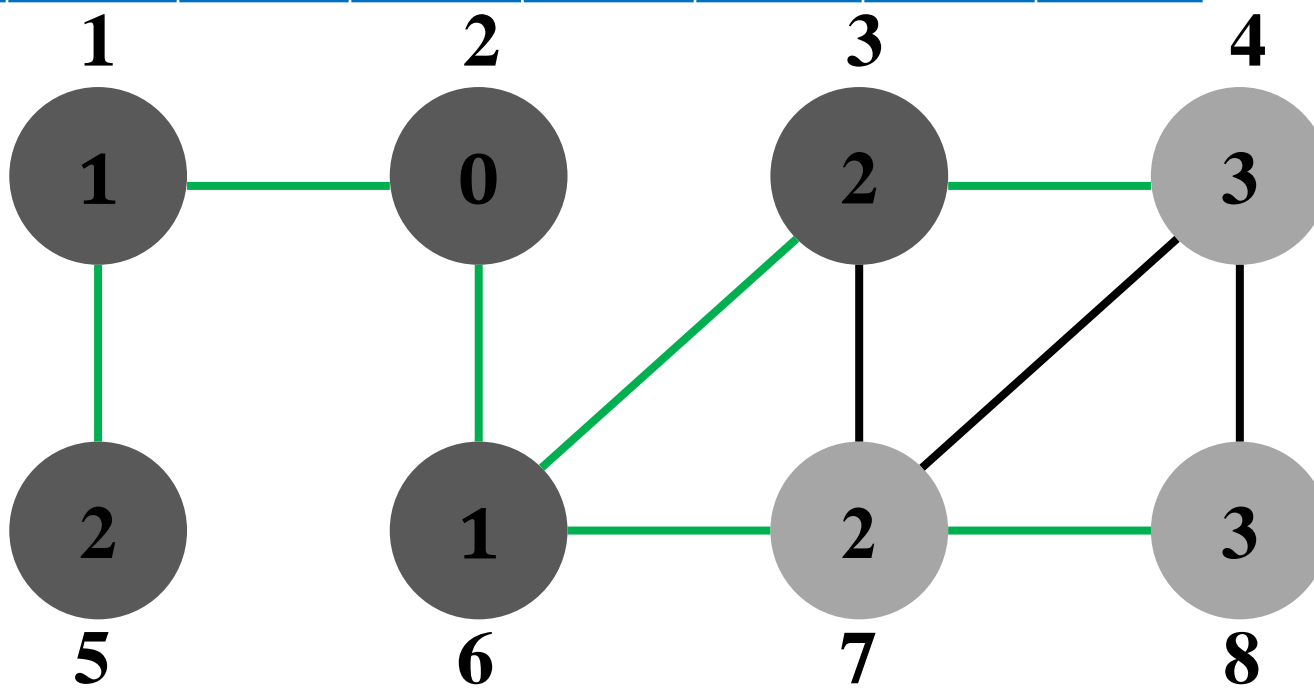


算法实例



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

源点2

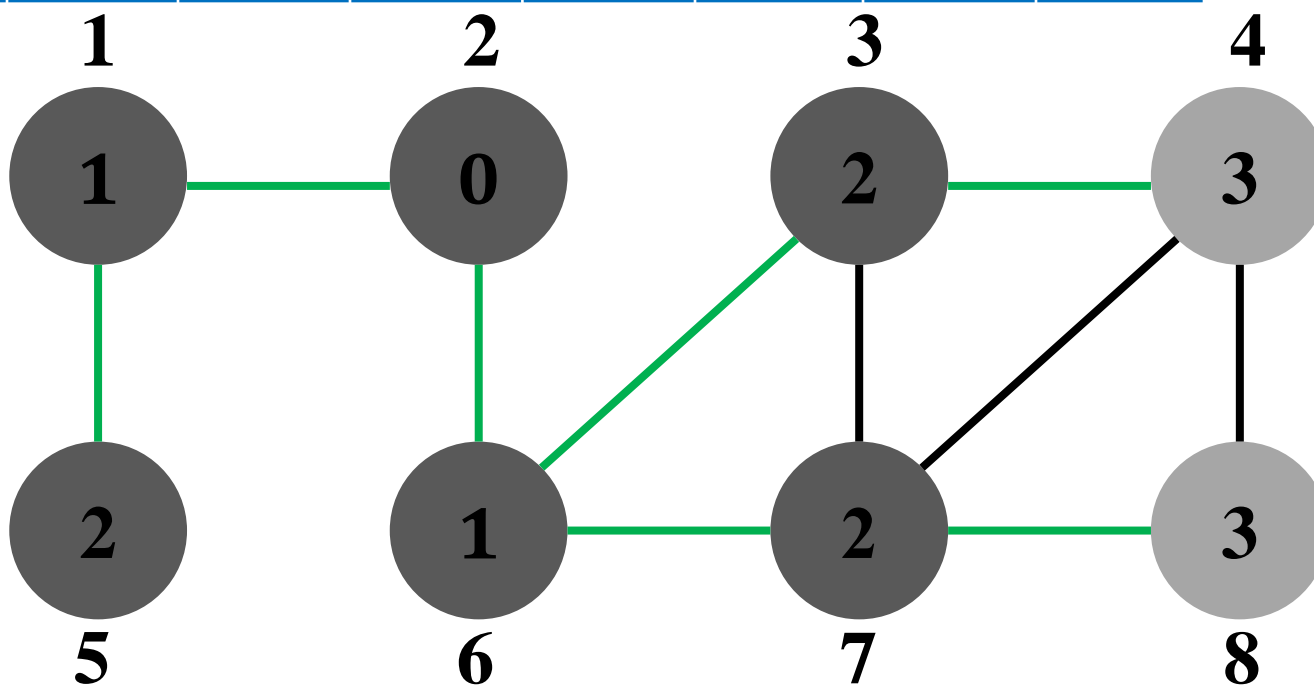


算法实例



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

源点2

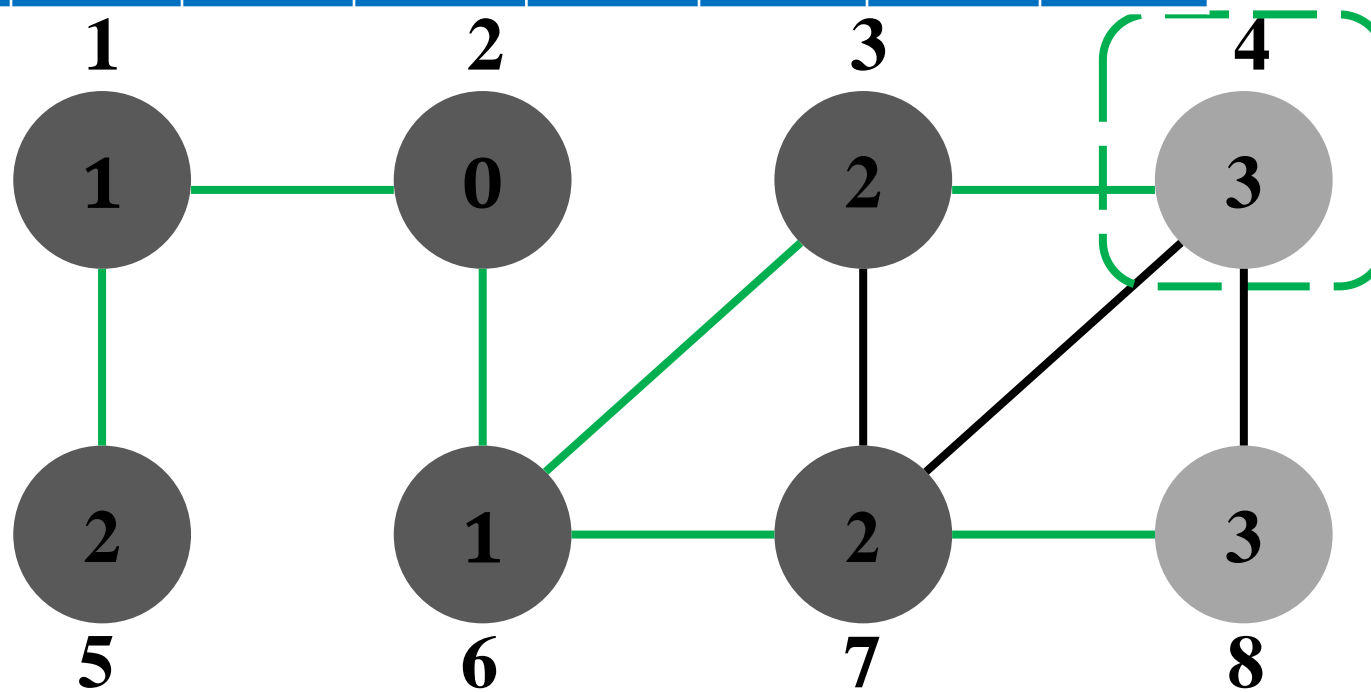


算法实例



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

源点2

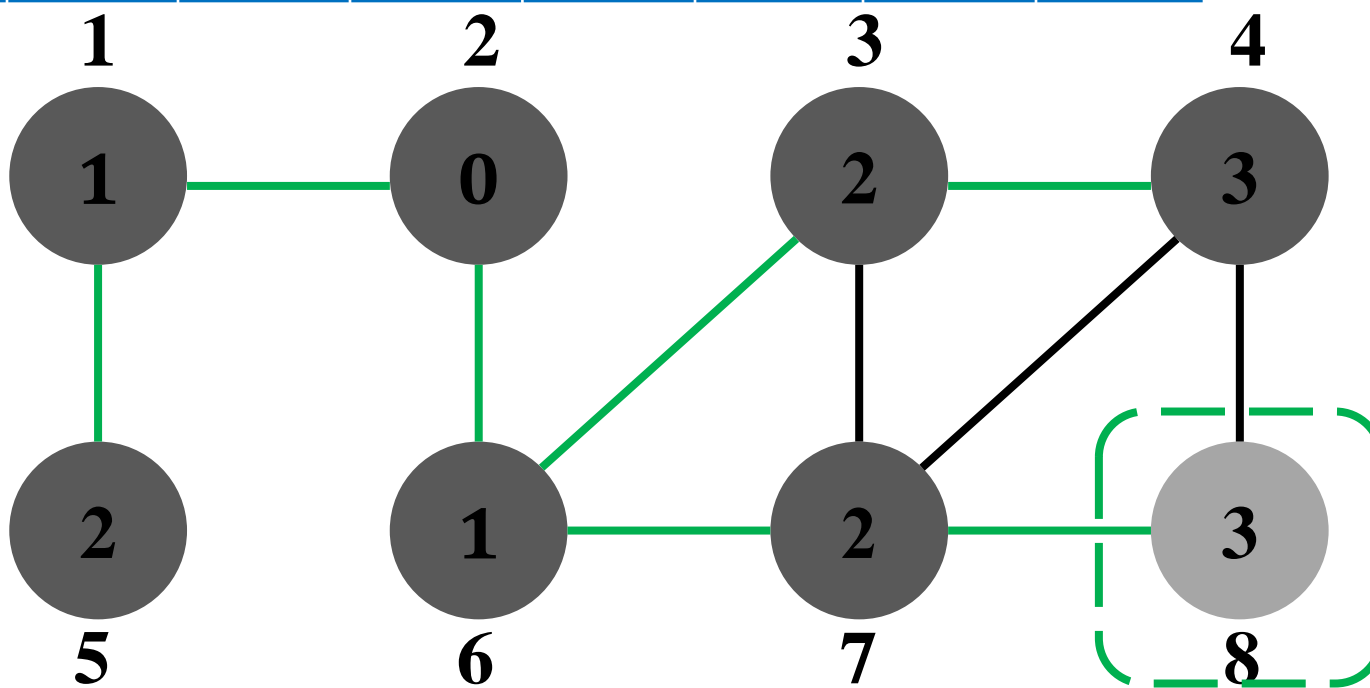


算法实例



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

源点2

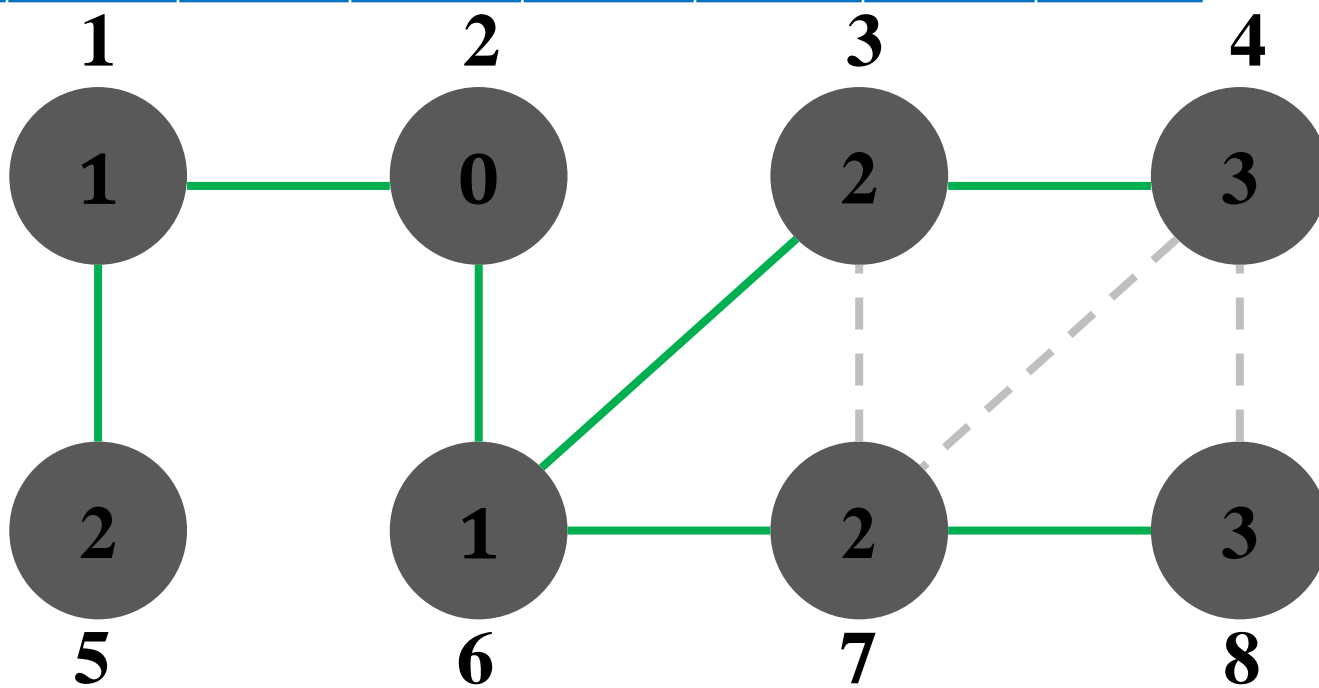


算法实例



<i>V</i>	1	2	3	4	5	6	7	8
<i>color</i>	B	B	B	B	B	B	B	B
<i>pred</i>	2	N	6	3	1	2	6	7
<i>dist</i>	1	0	2	3	2	1	2	3
待处理队列	2	1	6	5	3	7	4	8

源点2



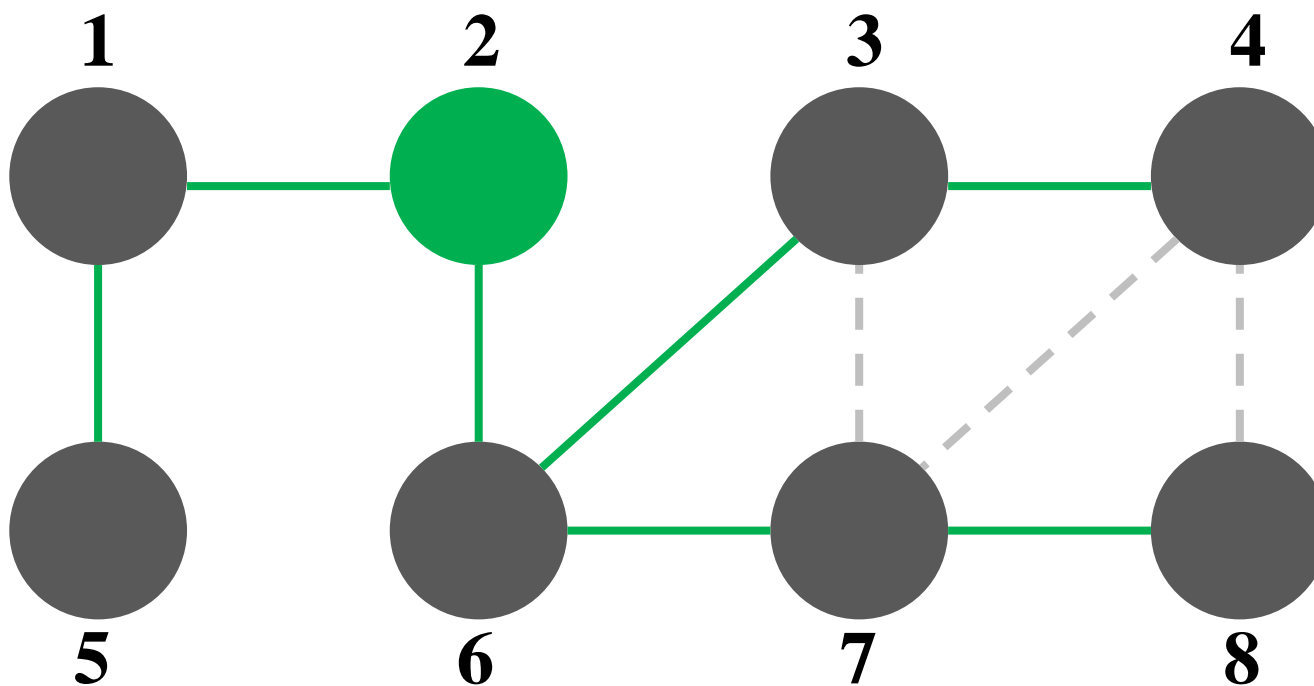
搜索结束

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



连通、无环
 $|E_T| = |V_T| - 1$

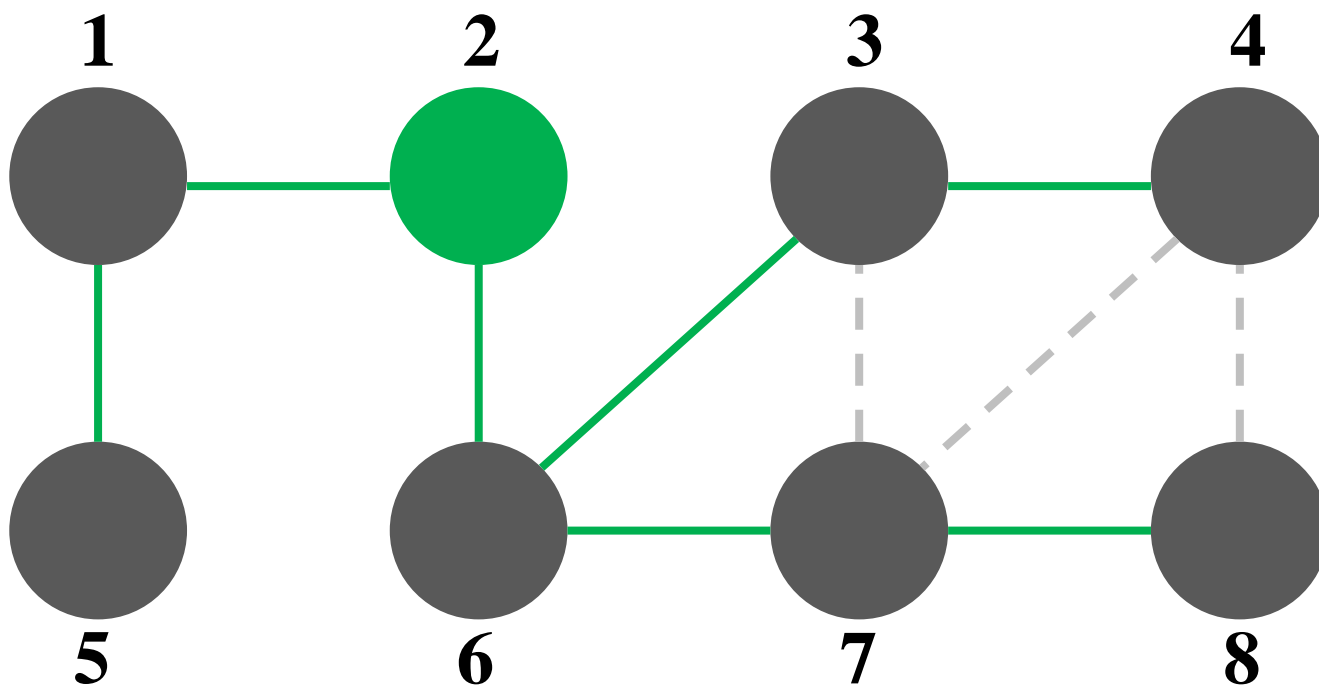
广度优先树



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** $T = \langle V_T, E_T \rangle$

连通、无环
 $|E_T| = |V_T| - 1$

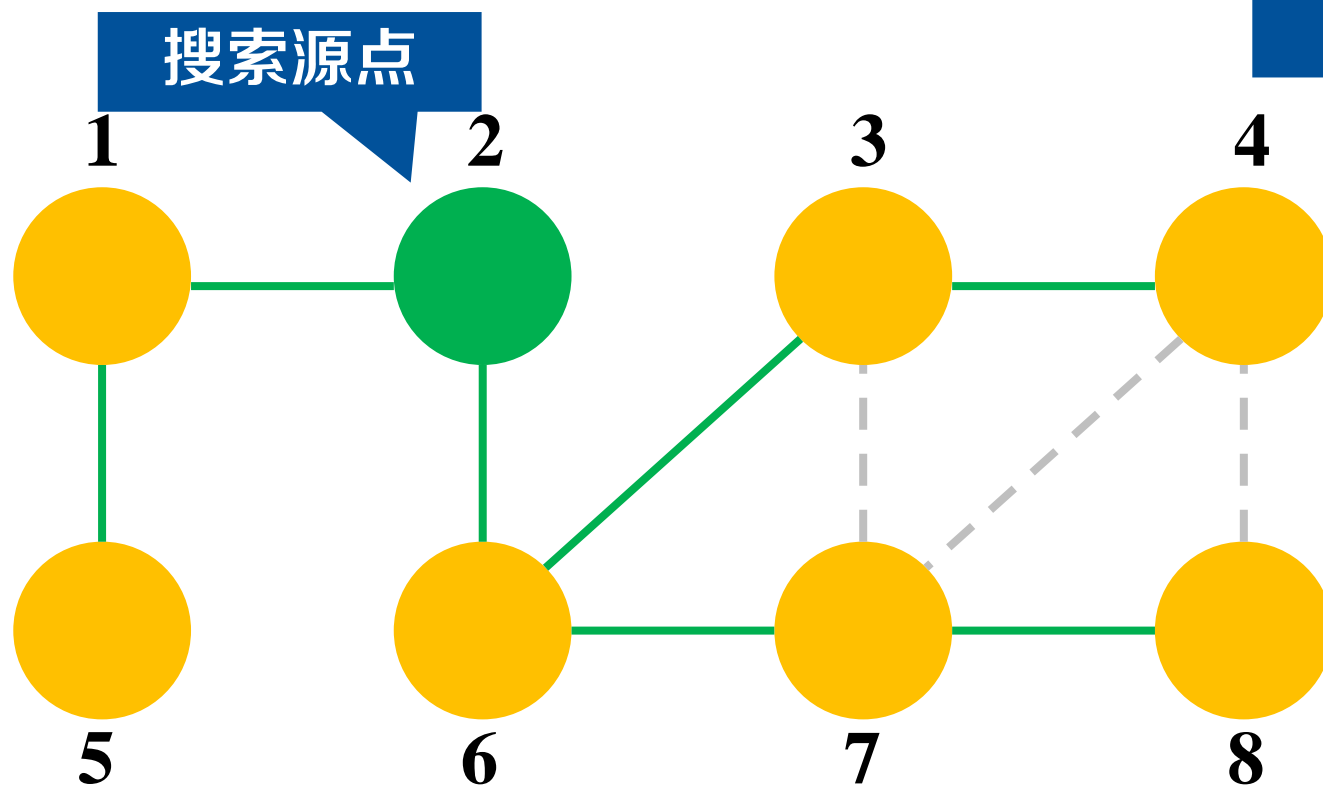


广度优先树



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 $T = \langle V_T, E_T \rangle$
- V_T 是源点 s 可达的顶点集合



连通、无环
 $|E_T| = |V_T| - 1$

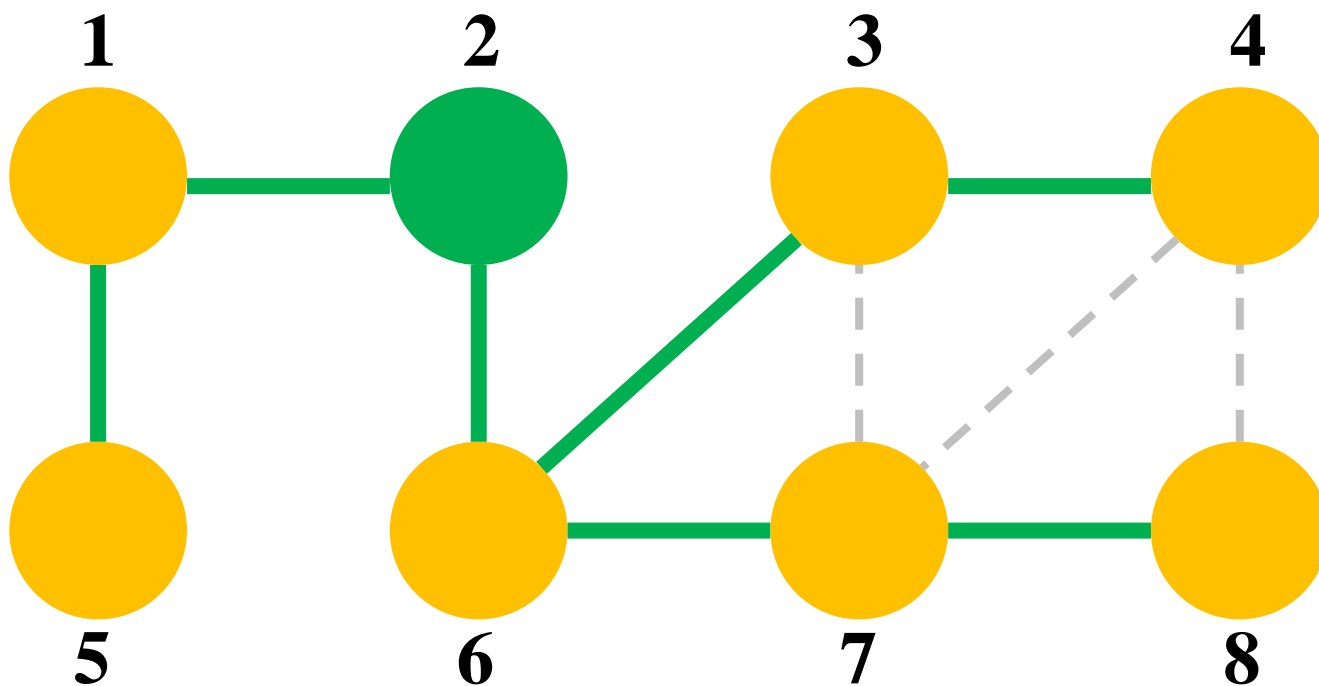
广度优先树



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 $T = \langle V_T, E_T \rangle$
- 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
 $|E_T| = |V_T| - 1$



图的搜索

算法思想

算法实例

算法分析

算法应用

- **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

- **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

- **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

- **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
```

依次处理所有顶点

- **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
```

从等待队列取出当前处理顶点 u

- **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

- **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

第一次发现顶点 v

- **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

- **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

- **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

- **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

加入待处理队列

- **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

标记当前顶点处理完成

- 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

- 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

- 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

- 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

- 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:

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

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

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

- 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：

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

简化表示

在渐近记号中，约定符号 V 代表 $|V|$ ，符号 E 代表 $|E|$

图的搜索

算法思想

算法实例

算法分析

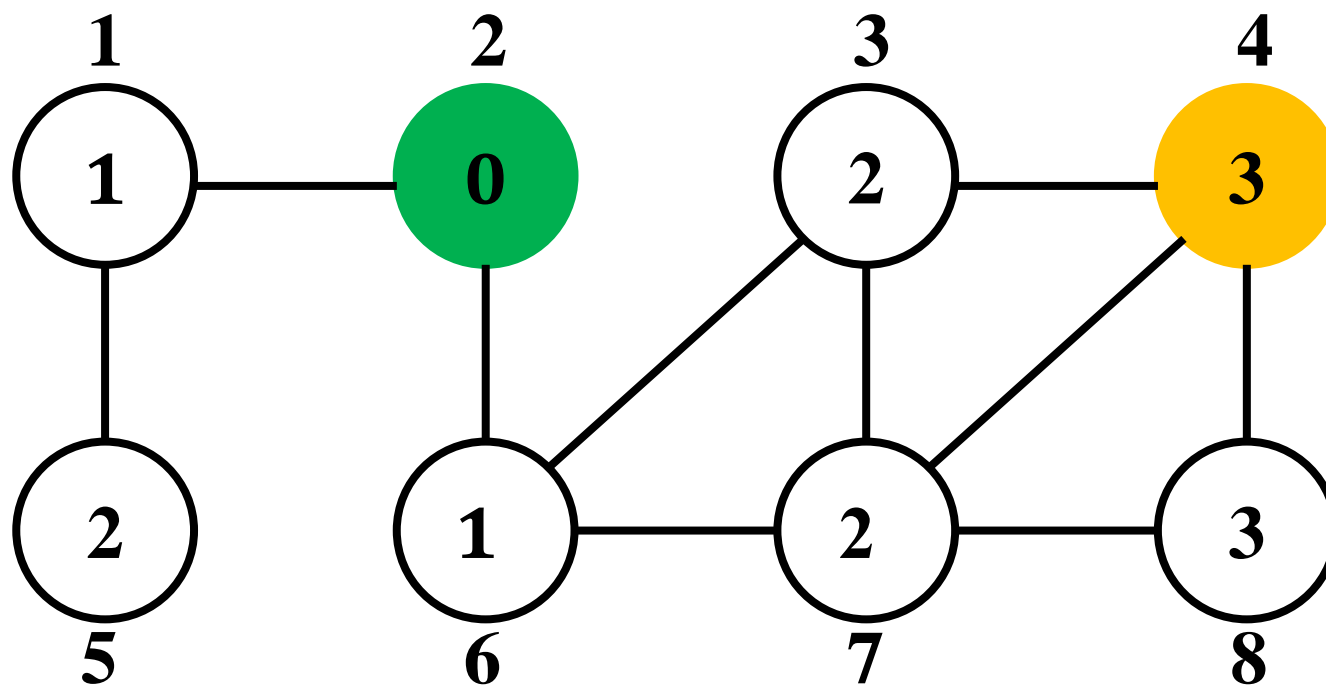
算法应用

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

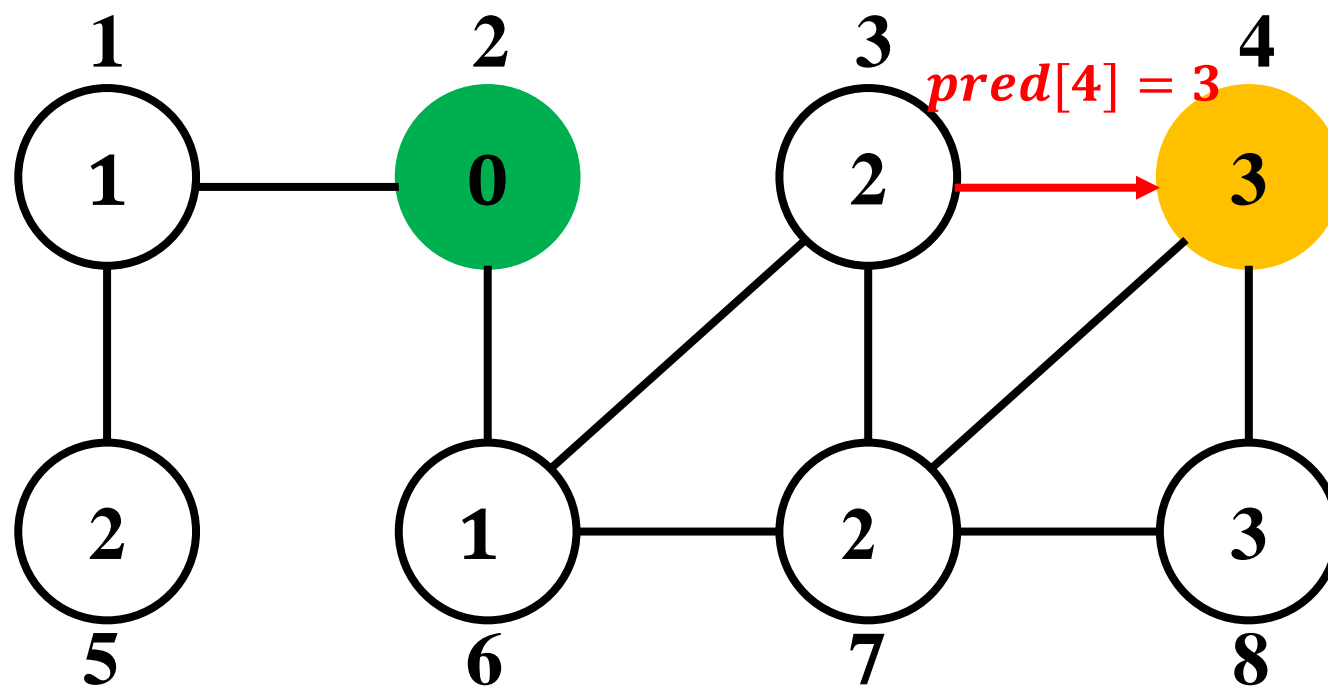


相关应用：最短路径



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

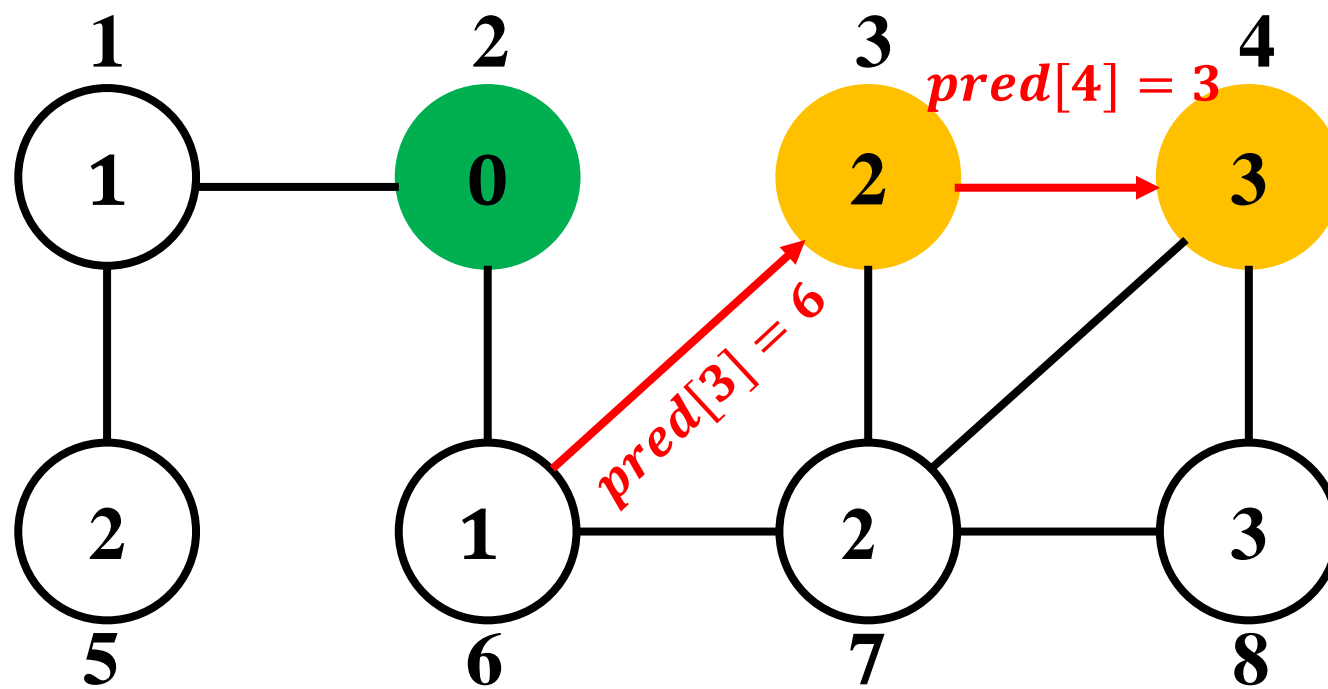


相关应用：最短路径



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

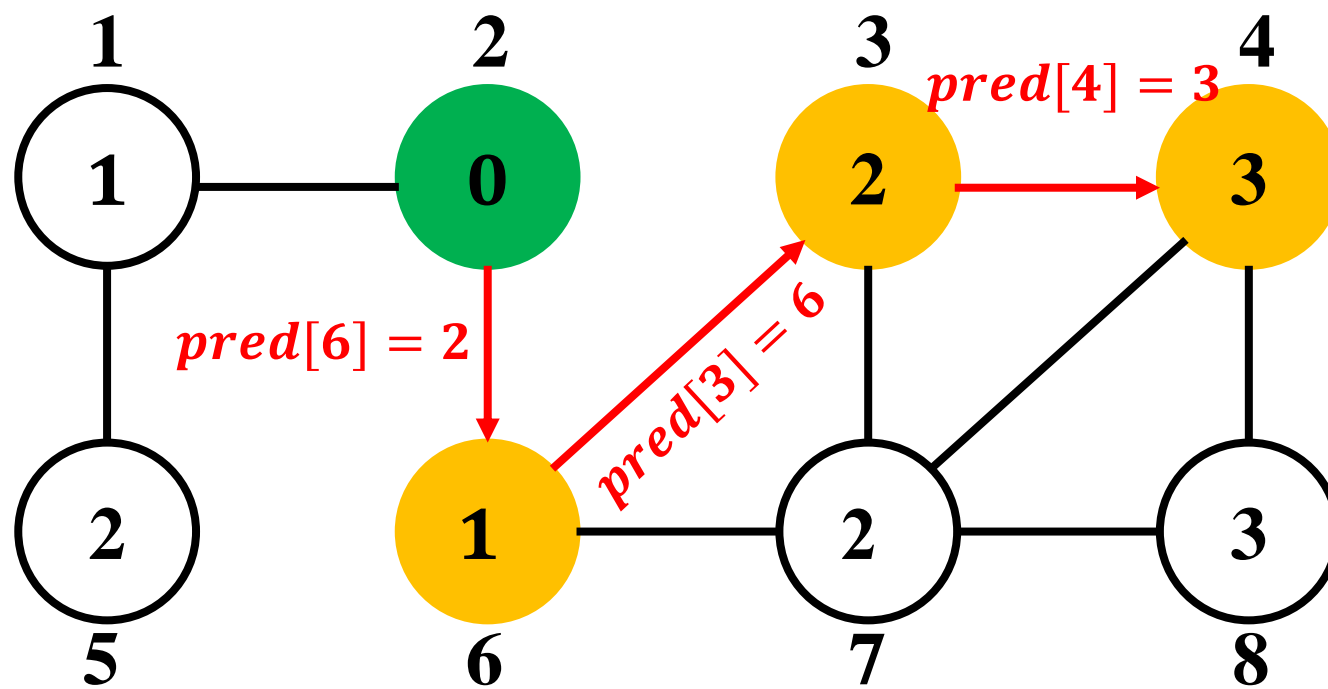


相关应用：最短路径



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

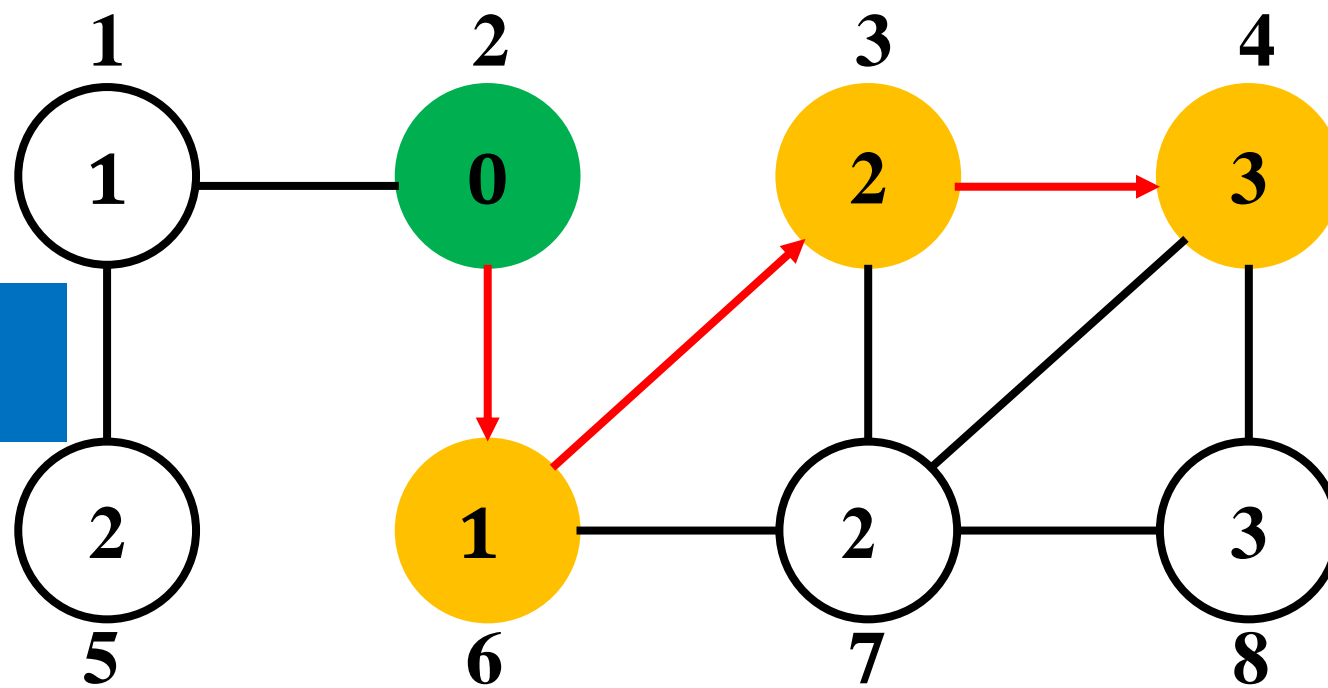


相关应用：最短路径



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

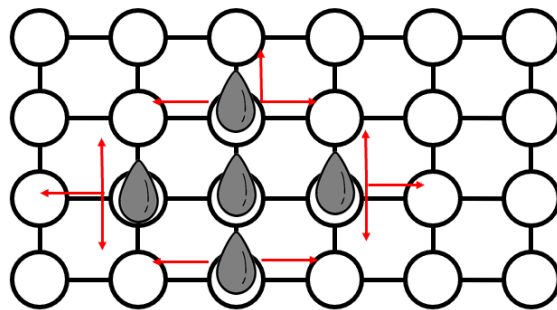
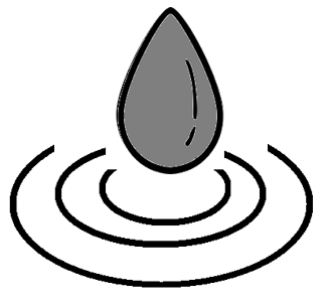
无权图 G 中，源点2到顶点4的最短路径



$2 \rightarrow 6 \rightarrow 3 \rightarrow 4$

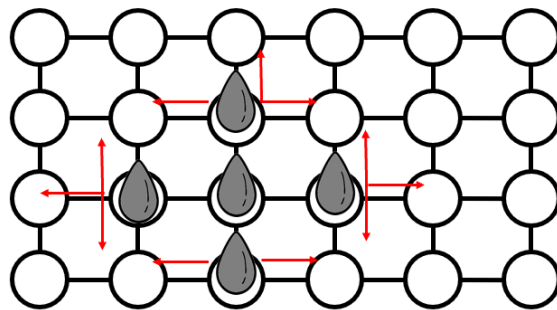
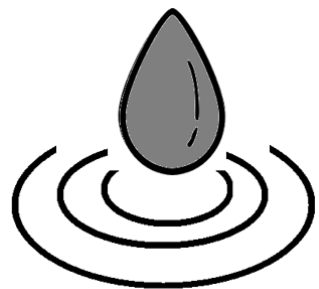
- 广度优先搜索

- 算法核心思想：逐层扩散



- 广度优先搜索

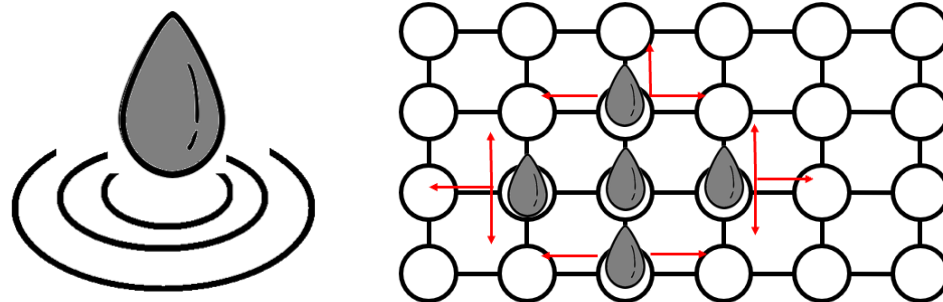
- 算法核心思想：逐层扩散



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

- 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

