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》



算法思想

算法实例

算法分析

算法应用



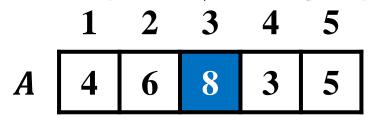
- 数组结构
 - 查询最大值:简单循环搜索所有元素,记录最大值

 1
 2
 3
 4
 5

 A
 4
 6
 8
 3
 5

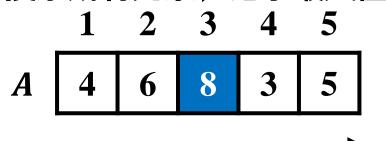


- 数组结构
 - 查询最大值:简单循环搜索所有元素,记录最大值

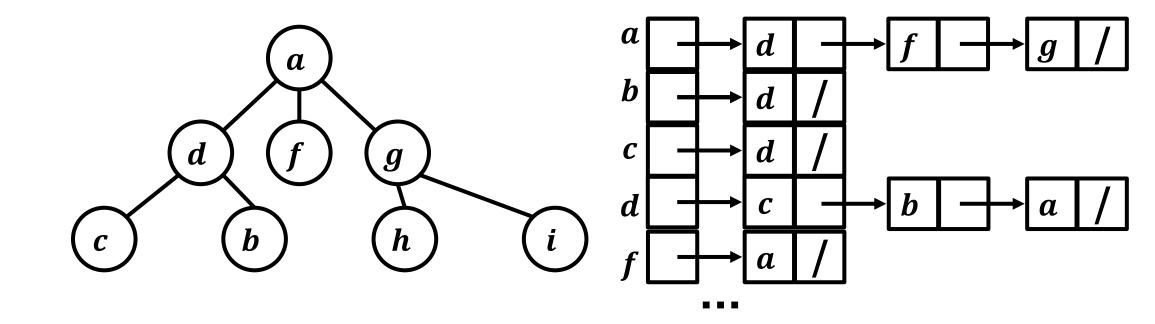




- 数组结构
 - 查询最大值:简单循环搜索所有元素,记录最大值

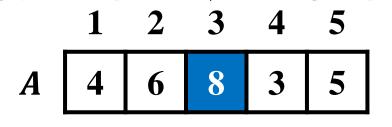


- 图结构
 - 查询相邻顶点:简单循环搜索各顶点关联的边



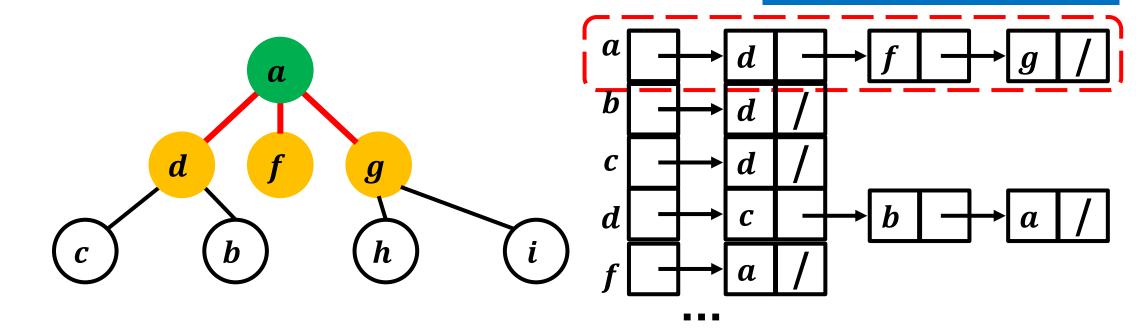


- 数组结构
 - 查询最大值:简单循环搜索所有元素,记录最大值



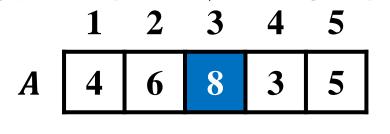
- 图结构
 - 查询相邻顶点:简单循环搜索各顶点关联的边

顶点a与 $\{d, f, g\}$ 相邻



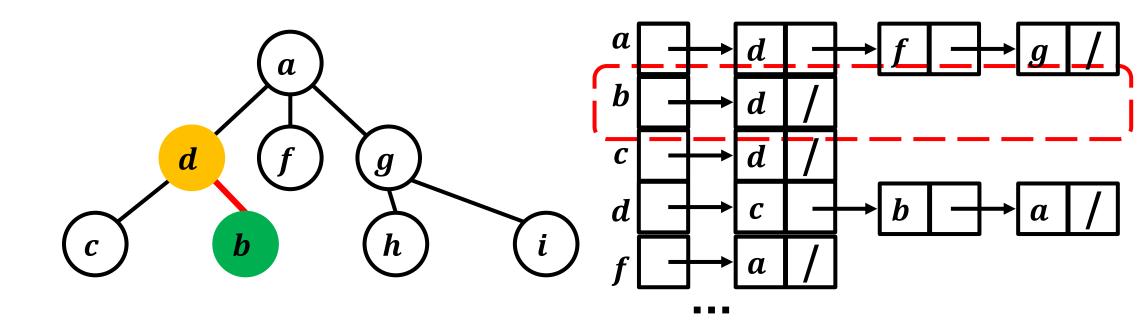


- 数组结构
 - 查询最大值:简单循环搜索所有元素,记录最大值



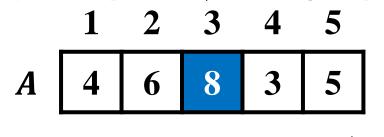
- 图结构
 - 查询相邻顶点:简单循环搜索各顶点关联的边

顶点b与 $\{d\}$ 相邻

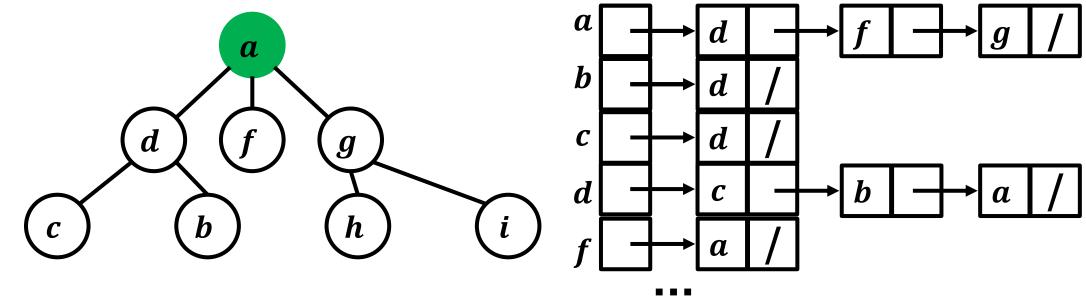




- 数组结构
 - 查询最大值:简单循环搜索所有元素,记录最大值

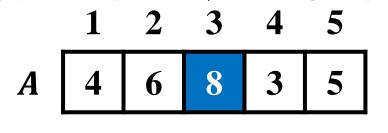


- 图结构
 - 查询相邻顶点:简单循环搜索各顶点关联的边
 - 查询可达顶点

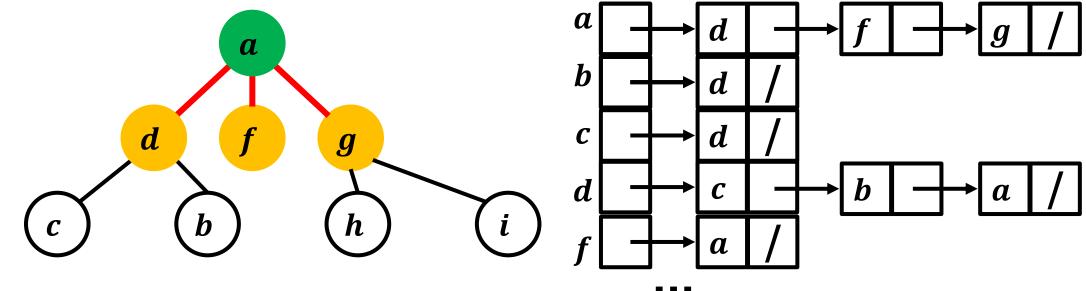




- 数组结构
 - 查询最大值:简单循环搜索所有元素,记录最大值

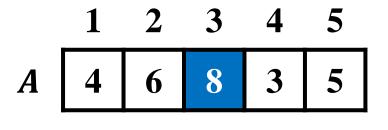


- 图结构
 - 查询相邻顶点:简单循环搜索各顶点关联的边
 - 查询可达顶点:简单循环搜索,不能找到全部可达顶点!

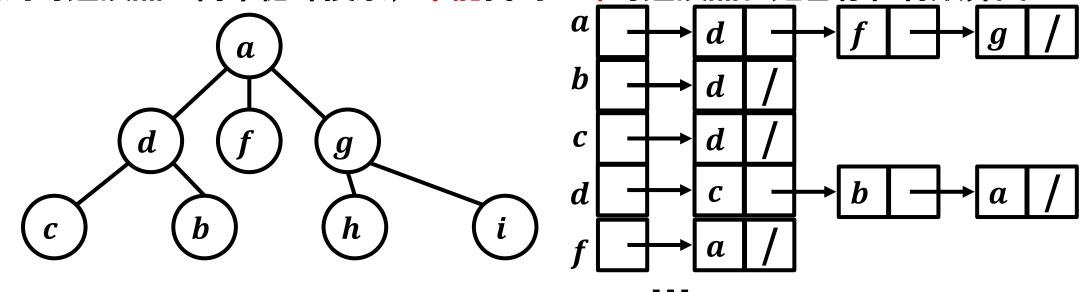




- 数组结构
 - 查询最大值:简单循环搜索所有元素,记录最大值



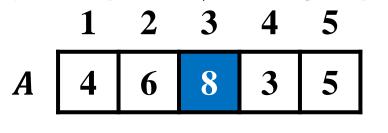
- 图结构
 - 查询相邻顶点:简单循环搜索各顶点关联的边
 - 查询可达顶点:简单循环搜索,不能找到全部可达顶点!是否存在有效算法?





• 数组结构

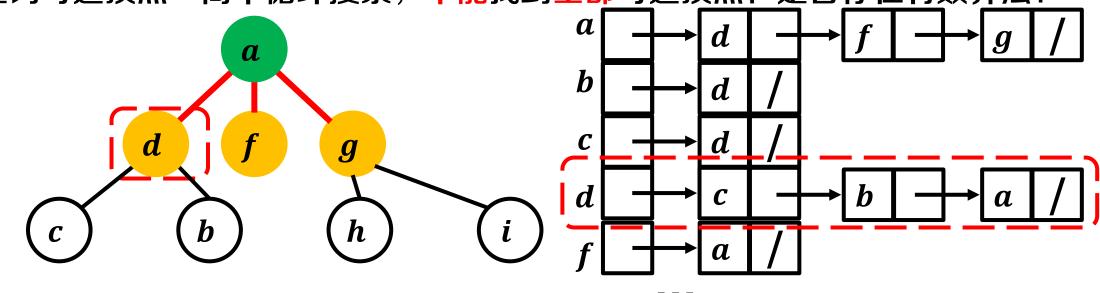
• 查询最大值:简单循环搜索所有元素,记录最大值



• 图结构

● 查询相邻顶点:简单循环搜索各顶点关联的边

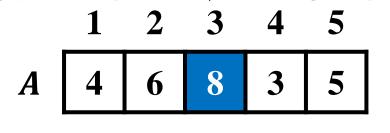
● 查询可达顶点:简单循环搜索,不能找到全部可达顶点!是否存在有效算法?





• 数组结构

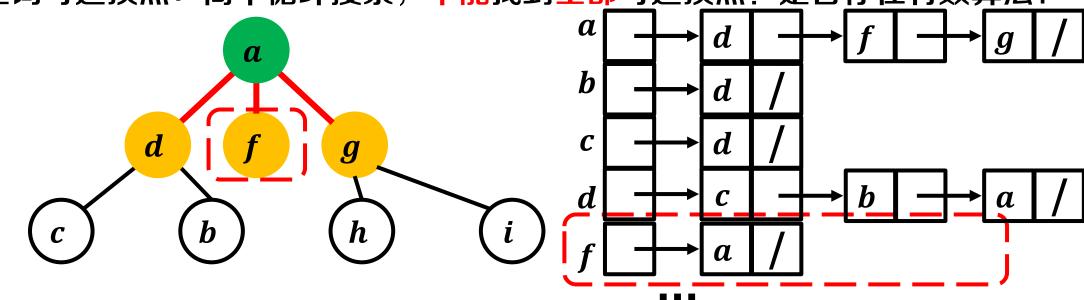
• 查询最大值:简单循环搜索所有元素,记录最大值



图结构

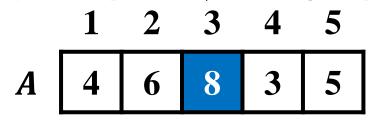
● 查询相邻顶点:简单循环搜索各顶点关联的边

● 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?

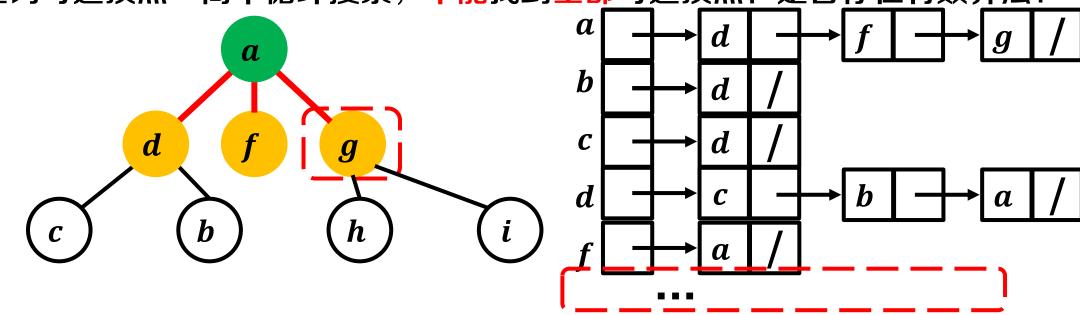




- 数组结构
 - 查询最大值:简单循环搜索所有元素,记录最大值



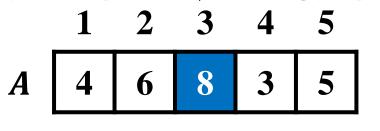
- 图结构
 - 查询相邻顶点:简单循环搜索各顶点关联的边
 - 查询可达顶点:简单循环搜索,不能找到全部可达顶点!是否存在有效算法?





• 数组结构

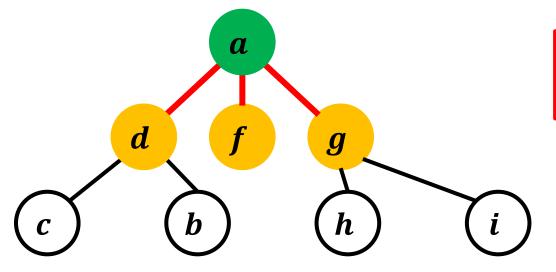
• 查询最大值:简单循环搜索所有元素,记录最大值



• 图结构

● 查询相邻顶点:简单循环搜索各顶点关联的边

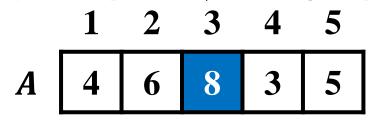
● 查询可达顶点:简单循环搜索,不能找到全部可达顶点!是否存在有效算法?



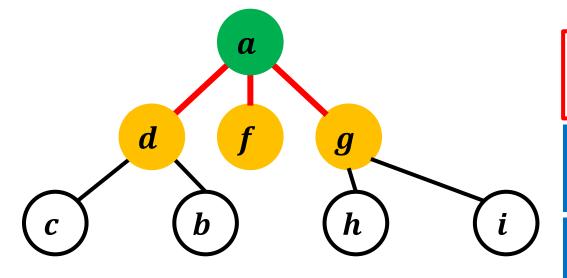
按照什么次序搜索顶点?



- 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?



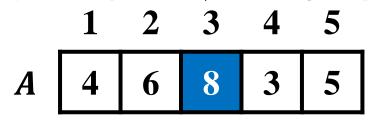
按照什么次序搜索顶点?

广度优先搜索

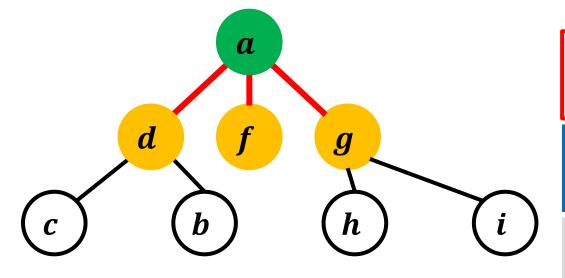
深度优先搜索



- 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



算法思想

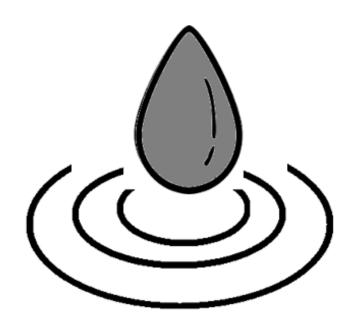
算法实例

算法分析

算法应用

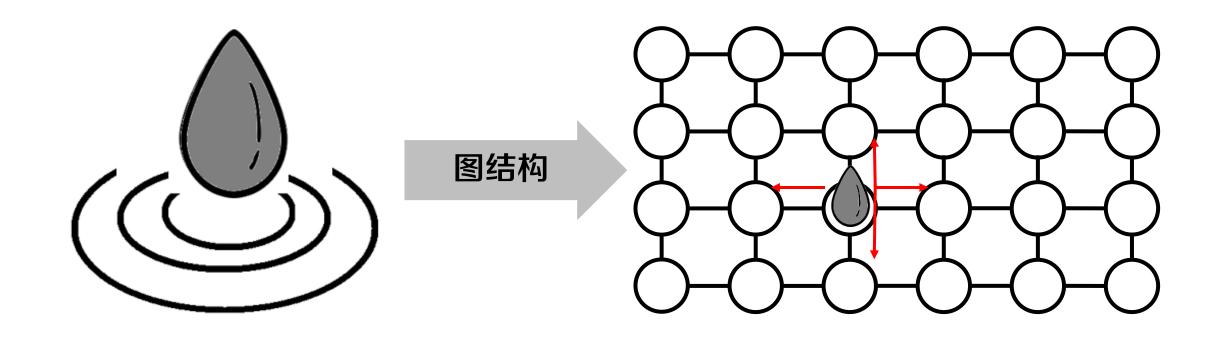


• 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



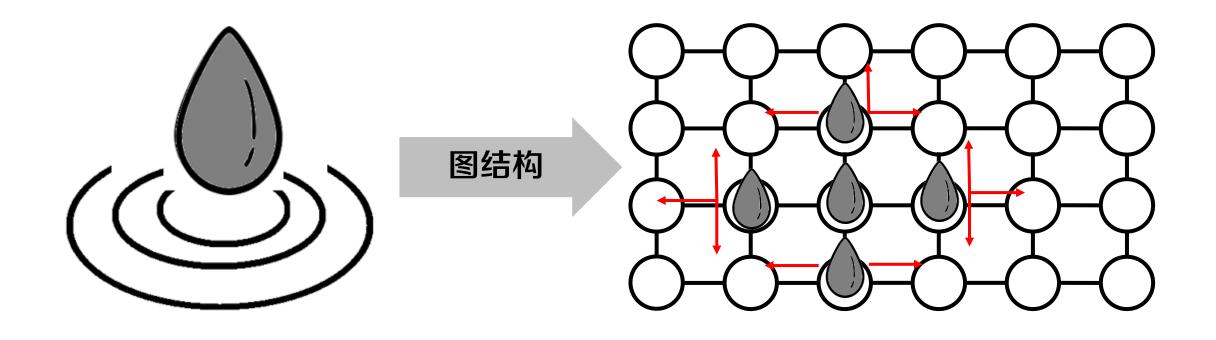


• 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



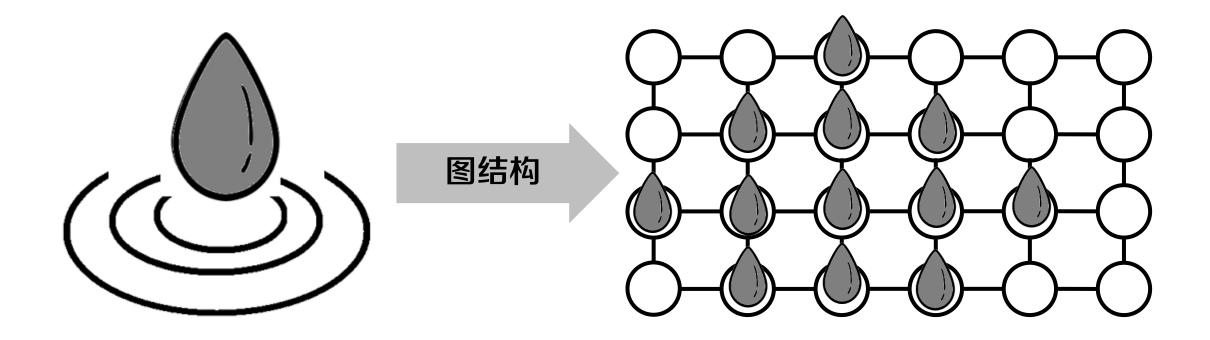


• 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散





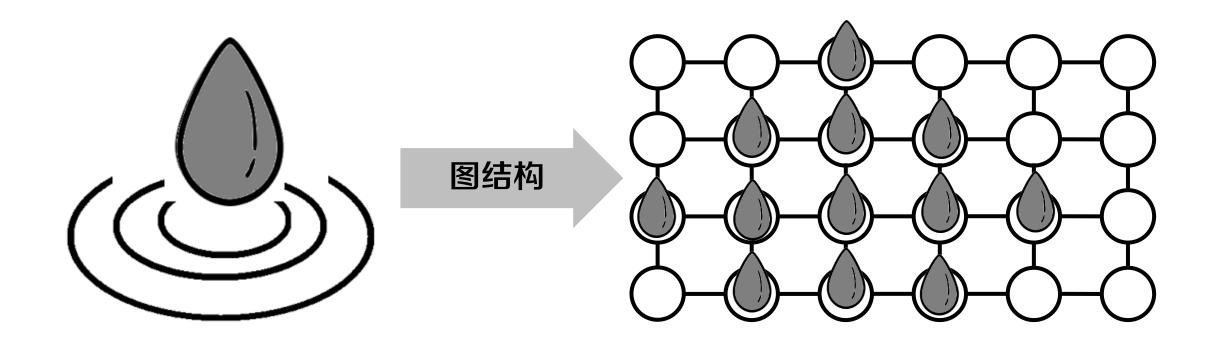
• 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



这种依次扩散的现象,蕴含了搜索图结构的一种顺序



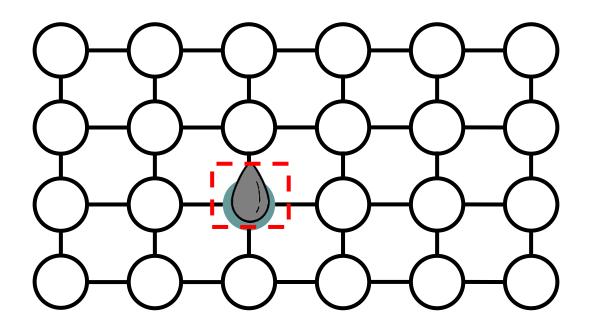
• 水滴落入水面所激起的涟漪会向相邻区域逐渐扩散



处理某顶点时,一次性发现所有其相邻顶点

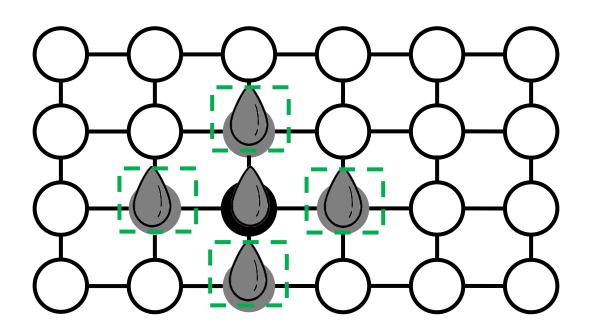


- 核心思想
 - 处理某顶点时,一次性发现其所有相邻顶点





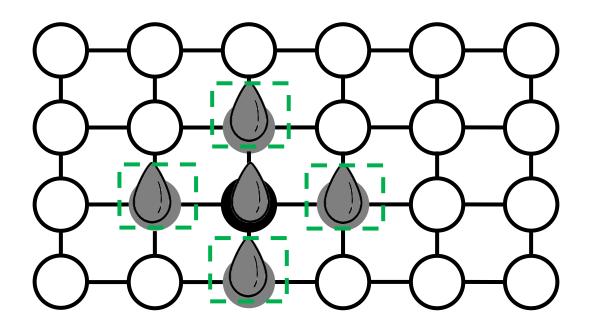
- 核心思想
 - 处理某顶点时,一次性发现其所有相邻顶点





- 核心思想
 - 处理某顶点时,一次性发现其所有相邻顶点

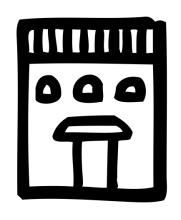
扩散中多处同时进行,一次只能处理单个顶点怎么办?

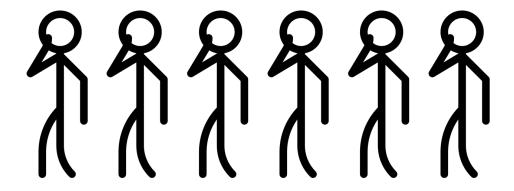


算法思想: 队列



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

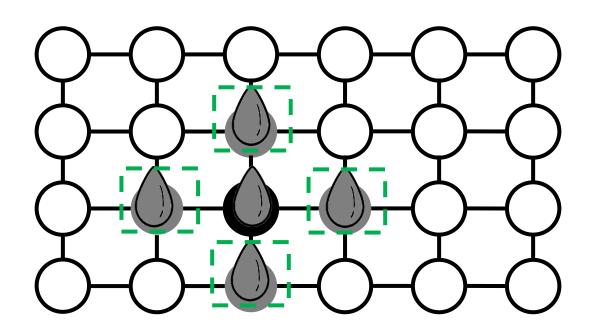




队列Q

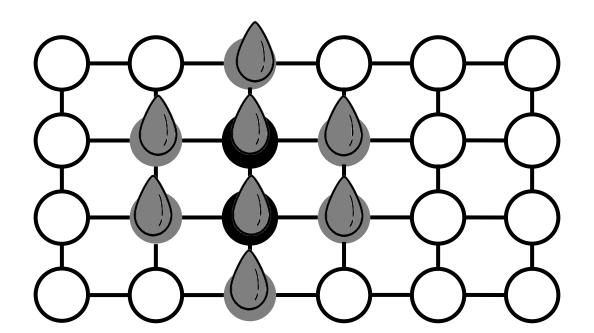


- 核心思想
 - 处理某顶点时,一次性发现其所有相邻顶点,未处理顶点加入等待队列



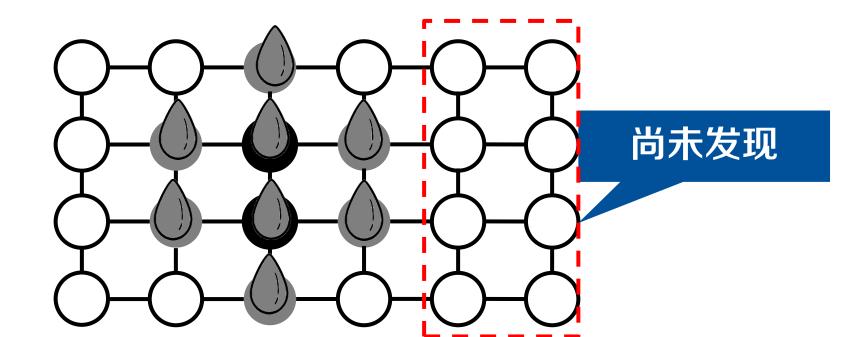


- 辅助数组
 - color表示顶点状态





- 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



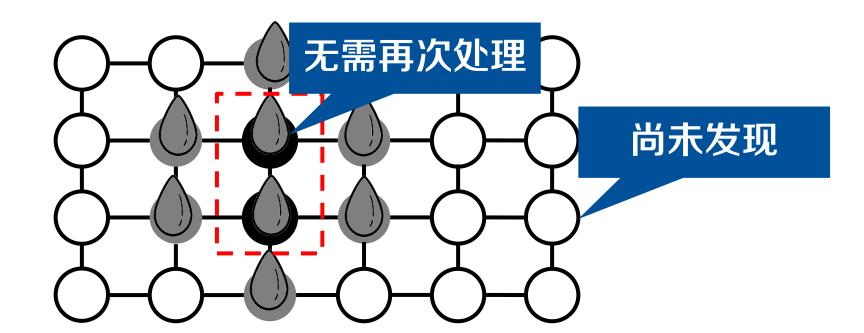


• 辅助数组

● color表示顶点状态

₀ White: 白色顶点u尚未被发现,发现后直接人队

。 Black: 黑色顶点u已被处理,无需再次入队





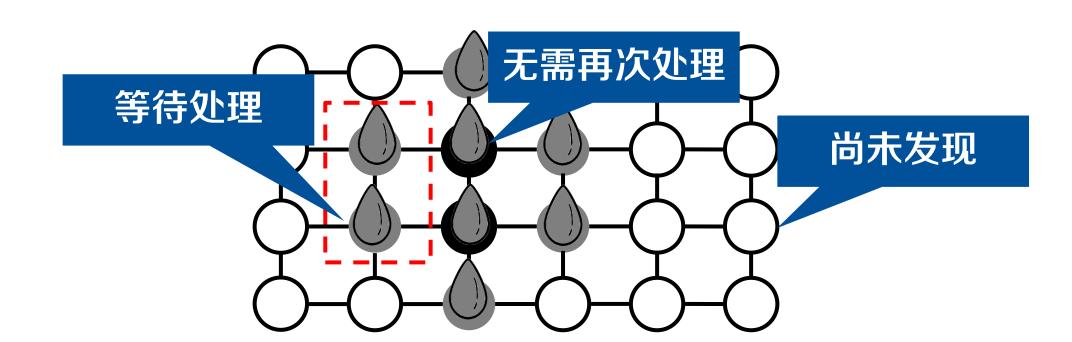
辅助数组

color表示顶点状态

₀ White: 白色顶点u尚未被发现,发现后直接人队

。 Black: 黑色顶点u已被处理,无需再次入队

₀ Gray: 灰色顶点u已加入队列,无需再次入队





• 辅助数组

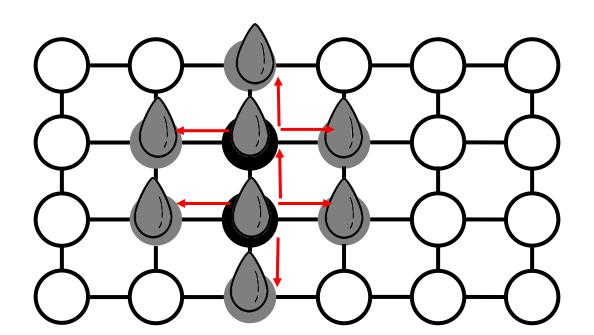
color表示顶点状态

。 White: 白色顶点u尚未被发现,发现后直接人队

。 Black: 黑色顶点u已被处理,无需再次入队

₀ 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

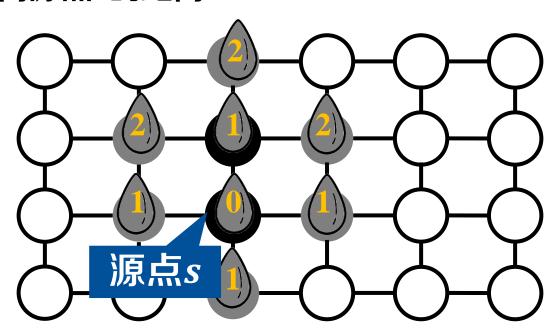
。 White: 白色顶点u尚未被发现,发现后直接人队

。 Black: 黑色顶点u已被处理,无需再次入队

₀ 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





算法思想

算法实例

算法分析

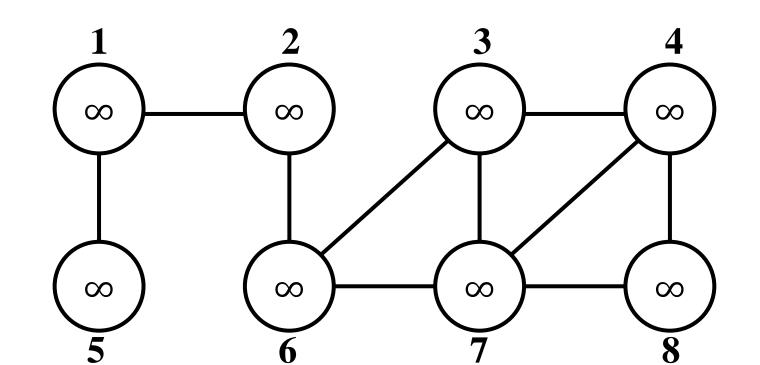
算法应用

算法实例



$oldsymbol{V}$	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



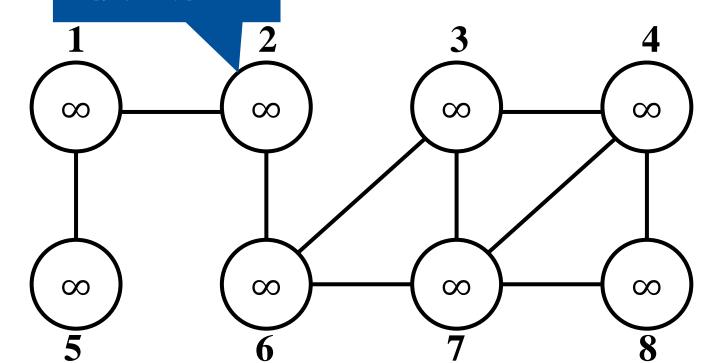
算法实例



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



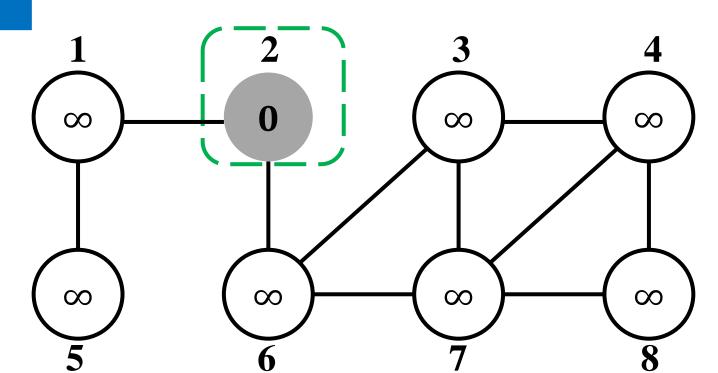


$oldsymbol{V}$	1	2	3	4	5	6	7	8
color	W	G	W	W	W	W	W	W
pred	N	N	N	N	N	N	N	N
dist	∞	0	∞	∞	∞	∞	∞	∞

待处理队列

2



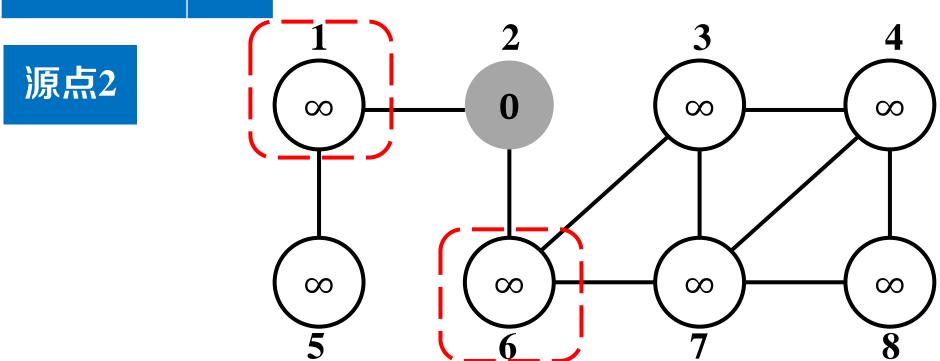




$oldsymbol{V}$	1	2	3	4	5	6	7	8
color	W	G	W	W	W	W	W	W
pred	N	N	N	N	N	N	N	N
dist	∞	0	∞	∞	∞	∞	∞	∞

待处理队列

2



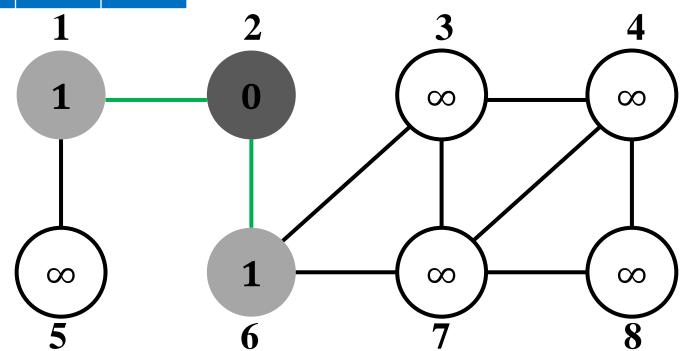


color G G W W W W W pred 2 N <t< th=""><th>V</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th></th></t<>	V	1	2	3	4	5	6	7	8	
dist 1 0 ∞ ∞ ∞ 1 ∞ ∞	color	G	G	W	W	W	G	W	W	
待处理队列 2 1 2 源点2	pred	2	N	N	N	N	2	N	N	
原点2	dist	1	0	∞	∞	∞	1	∞	∞	
源点2	待处理队列	2	1	6						
	源点2									



V	1	2	3	4	5	6	7	8	
color	G	В	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						
源占2		1		2			3		4

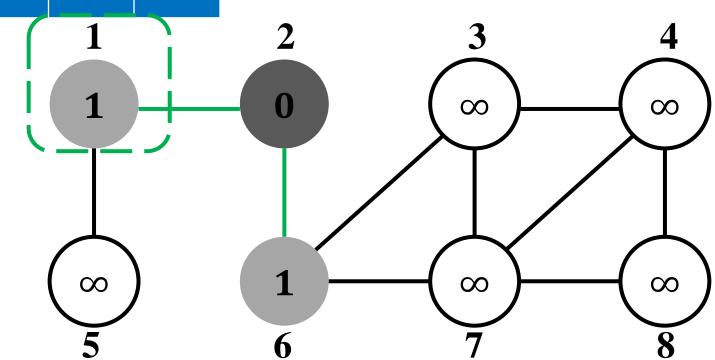






V	1	2	3	4	5	6	7	8	
color	G	В	W	W	W	G	W	W	
pred	2	N	N	N	N	2	N	N	
dist	1	0	∞	∞	∞	1	∞	∞	
待处理队列	2	1	6						
		1		2			3		4

源点2

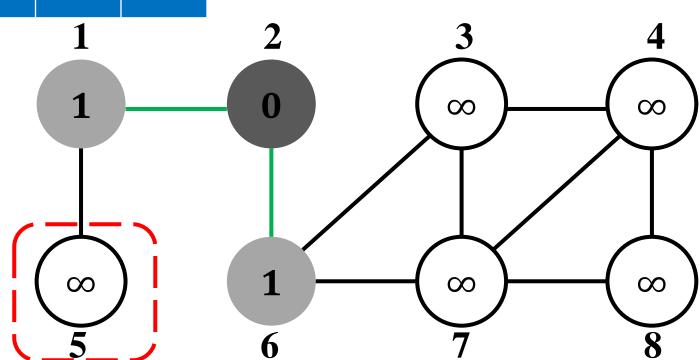




$oldsymbol{V}$	1	2	3	4	5	6	7	8
color	G	В	W	W	W	G	W	W
pred	2	N	N	N	N	2	N	N
dist	1	0	∞	∞	∞	1	∞	∞

待处理队列 2 1 6

源点2





color G B W W G G W W pred 2 N N N 1 2 N N dist 1 0 ∞ ∞ 2 1 ∞ ∞ 特处理队列 2 1 6 5	\boldsymbol{V}	1	2	3	4	5	6	7	8
dist 1 0 ∞ ∞ 2 1 ∞ ∞ 特处理队列 2 1 6 5 源点2 1 2 3	color	G	В	W	W	G	G	W	W
待处理队列 2 1 6 5 源点2 1 2 3	pred	2	N	N	N	1	2	N	N
源点2	dist	1	0	∞	∞	2	1	∞	∞
源点2	待处理队列	2	1	6	5				
	源点2								





V	1	2	3	4	5	6	7	8
color	В	В	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		1 2		1		0		



$oldsymbol{V}$	1	2	3	4	5	6	7	8	
color	В	В	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		1		0					4 ∞
		5		6					$\frac{\infty}{8}$



源点2 2 3 4	V	1	2	3	4	5	6	7	8	
dist	color	В	В	G	W	G	G	G	W	
待处理队列 2 1 6 5 3 7 源点2 1 2 3 4	pred	2	N	6	N	1	2	6	N	
源点2 2 3 4	dist	1	0	2	∞	2	1	2	∞	
源点2	待处理队列	2	1	6	5	3	7			
$egin{array}{c cccc} 2 & & & & & & & & & & & & & & & & & & $	源点2									



V	1	2	3	4	5	6	7	8	
color	В	В	G	W	G	В	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		1		0			2		4 ∞ ↑
		2		1			2	-(∞
		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 ∞
dist 1 0 2 ∞ 2 1 2 ∞ 诗处理队列 2 1 6 5 3 7 源点之 2 3 7
诗处理队列 2 1 6 5 3 7 源占2
源占2
源占2



V	1	2	3	4	5	6	7	8	
color	В	В	G	W	В	В	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		1		2 0			3		4 ∞ 1
		2		1			2	-(∞
		5		6		-	7		8



V	1	2	3	4	5	6	7	8	
color	В	В	G	W	В	В	G	\mathbf{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		1		0			2	(4 ∞ →
		2		1		(2	2	-(∞
		5		6		-	7		$\overline{\mathbf{Q}}$







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 ∞ 海流之 1 2 3 3 4 源点2 1 2 3 4 2 3 4 3 4 2 3 4 3 4 2 3 4 4 4 2 4 4 4 4 4 2 4 4 4 4 4 2 4 4 4 4 4 4	V	1	2	3	4	5	6	7	8
dist	color	В	В	В	G	В	В	G	W
待处理队列 2 1 6 5 3 7 4 源点2 1 0 2 3	pred	2	N	6	3	1	2	6	N
源点2	dist	1	0	2	3	2	1	2	∞
源点2	待处理队列	2	1	6	5	3	7	4	
	源点2		1		0				



V	1	2	3	4	5	6	7	8
color	В	В	В	G	В	В	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		1		0			3	



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 ∞ (特处理队列 2 1 6 5 3 7 4	V	1	2	3	4	5	6	7	8
dist 1 0 2 3 2 1 2 ∞ 待处理队列 2 1 6 5 3 7 4 1 2 3	color	В	В	В	G	В	В	G	W
待处理队列 2 1 6 5 3 7 4 源占2 3	pred	2	N	6	3	1	2	6	N
1 2 3 源占2	dist	1	0	2	3	2	1	2	∞
海占2	待处理队列	2	1	6	5	3	7	4	
	源点2								



V	1	2	3	4	5	6	7	8	
color	В	В	В	G	В	В	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		1 2 5		2 0 1 6			3		3 3 8



V	1	2	3	4	5	6	7	8	
color	В	В	В	G	В	В	В	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		1 2 5		2 0 1 6			3		3 3 8

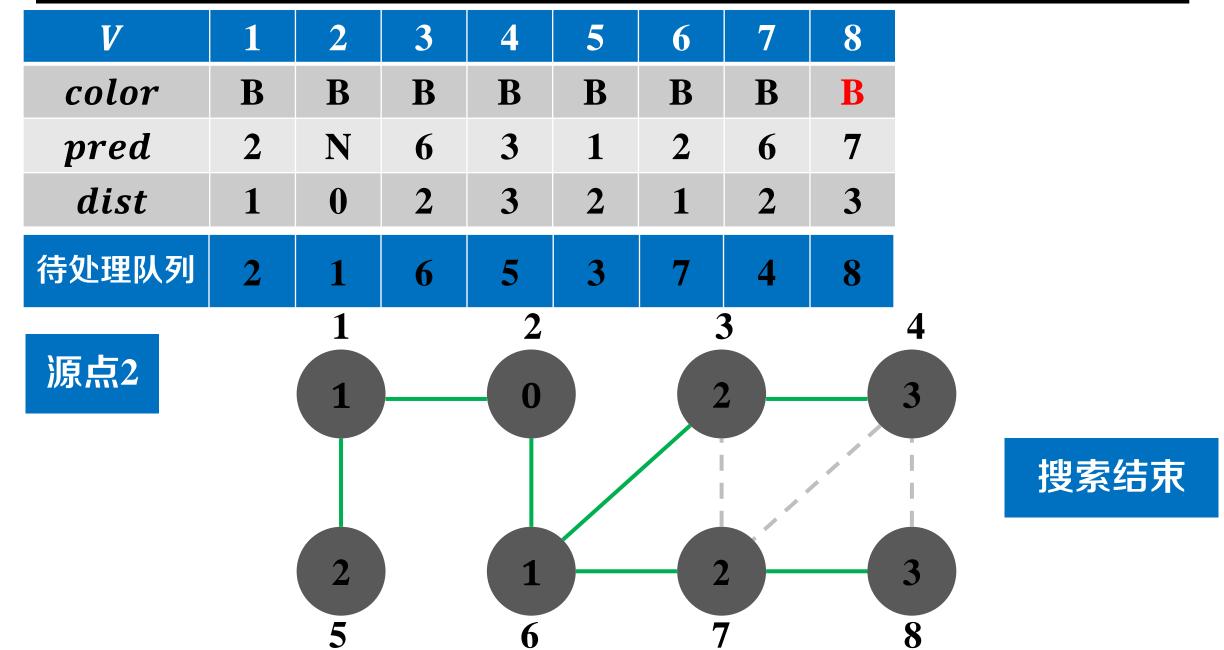


color B B B G B B B G pred 2 N 6 3 1 2 6 7 dist 1 0 2 3 2 1 2 3 添点2 1 6 5 3 7 4 8	pred 2 N 6 3 1 2 6 7 dist 1 0 2 3 2 1 2 3 特处理队列 2 1 6 5 3 7 4 8 1 2 3 4 8 4	V	1	2	3	4	5	6	7	8	
dist 1 0 2 3 2 1 2 3 待处理队列 2 1 6 5 3 7 4 8 源点2	dist 1 0 2 3 2 1 2 3 诗处理队列 2 1 6 5 3 7 4 8 源点2 1 2 3 2 3 4 3	color	В	В	В	G	В	В	В	G	
待处理队列 2 1 6 5 3 7 4 8 源点2 1 0 3 3	待处理队列 2 1 6 5 3 7 4 8 源点2 1 0 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	pred	2	N	6	3	1	2	6	7	
源点2 2 3 4 3	源点2 3 3 4 3 3 4 3 3 4 3 4 3 4 3 4 3 4 3 4	dist	1	0	2	3	2	1	2	3	
源点2	源点2	待处理队列	2	1	6	5	3	7	4	8	
		源点2		1		0					3



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



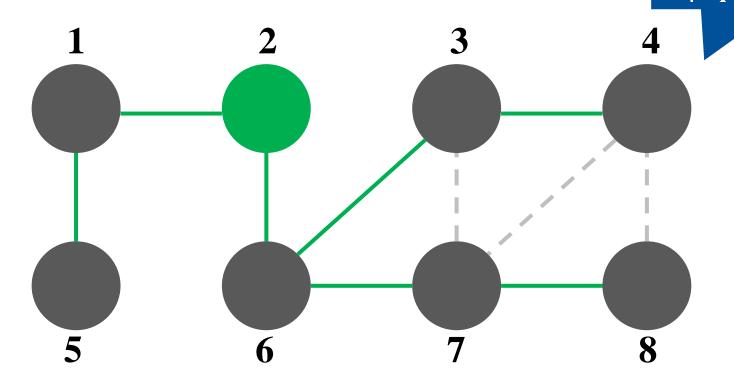




V	1	2	3	4	5	6	7	8
pred	2	N	6	3	1	2	6	7

● 辅助数组pred[]储存了一棵树

连通、无环 $|E_T| = |V_T| - 1$

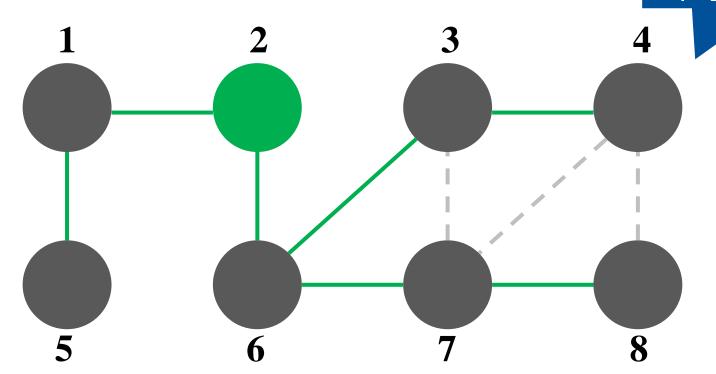




V	1	2	3	4	5	6	7	8
pred	2	N	6	3	1	2	6	7

• 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 $|E_T| = |V_T| - 1$





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 $T = \langle V_T, E_T \rangle$

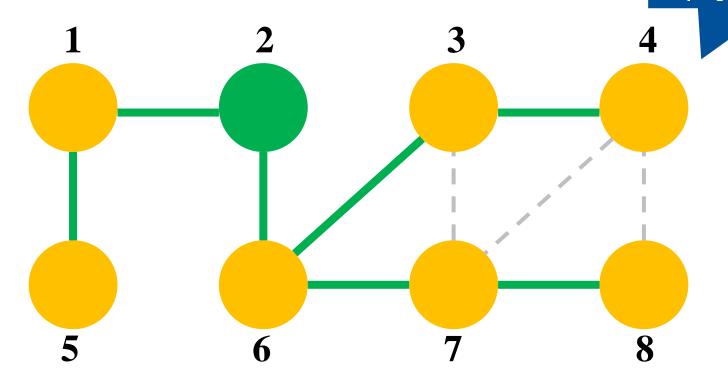
 V_T 是源点s可达的顶点集合 连通、无环 $|E_T| = |V_T| - 1$ 搜索源点



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 $|E_T| = |V_T| - 1$





图的搜索

算法思想

算法实例

算法分析

算法应用



\bullet 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
                                 初始化各记录信息数组
   color[u] \leftarrow WHITE
   pred[u] \leftarrow NULL
  dist[u] \leftarrow \infty
end
color[s] \leftarrow GRAY
dist[s] \leftarrow 0
Q.Enqueue(s)
```



 \bullet BFS(G, s)

Q.Enqueue(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
                                     初始化源点状态
dist[s] \leftarrow 0
```



 \bullet 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
```



 \bullet BFS(G, s)

```
//广度优先搜索
while 等待队列Q非空 do
  \overline{u} \leftarrow Q.Dequeue() - \overline{u}
    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



 \bullet 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[\overline{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



 \bullet 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
                                               搜索顶点u的所有相邻顶点
     | \mathbf{f} c \overline{o} l \overline{o} r | v | = WHTTE  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
```



```
//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
        \neg cotor[v] \leftarrow GRAY
           dist[v] \leftarrow dist[u] + 1
           pred[v] \leftarrow u
           Q.Enqueue(v)
        end
    end
   color[u] \leftarrow BLACK
end
```



```
//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
                                          标记已发现顶点v为待处理状态
          -dist[v] \leftarrow -dist[u] + 1
           pred[v] \leftarrow u
           Q.Enqueue(v)
       end
    end
   color[u] \leftarrow BLACK
end
```



```
//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
           \begin{array}{c} color[v] \leftarrow GRAY \\ dist[v] \leftarrow dist[u] + 1 \end{array}
                                                        记录到源点的距离
            \neg pred[v] \leftarrow u
             Q.Enqueue(v)
         end
    end
    color[u] \leftarrow BLACK
end
```



```
//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] + 1pred[v] \leftarrow u
                                                               记录前驱顶点
            \overline{Q}.\overline{E}\overline{n}q\overline{u}e\overline{u}e(\overline{v})
         end
     end
    color[u] \leftarrow BLACK
end
```



```
//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
          \bigcap_{v \in Q} pred[v] \leftarrow u 
Q.Enqueue(v)
                                                     加入待处理队列
        end
    end
    color[u] \leftarrow BLACK
end
```



```
//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
   color[u] \leftarrow BLACK
                                        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:

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

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

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:

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

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

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

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:

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

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

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



- 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:

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

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

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

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

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



- 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:

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

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

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

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

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:

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

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

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

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

在渐近记号中,约定符号V代表|V| ,符号E代表|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:

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

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

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

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

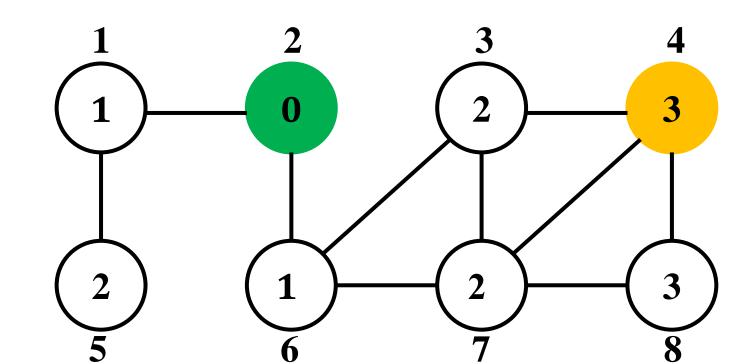
算法实例

算法分析

算法应用

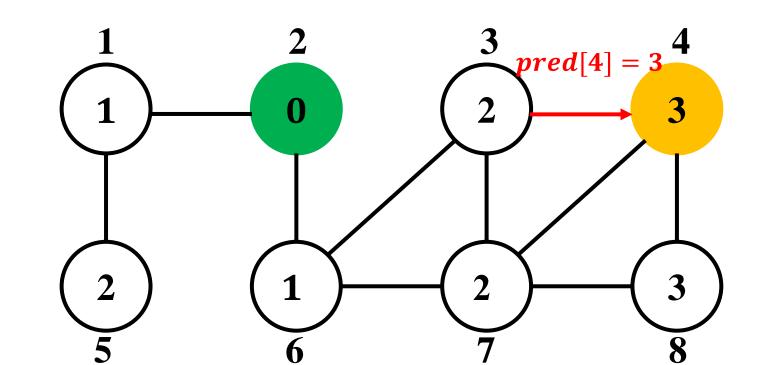


$oldsymbol{V}$	1	2	3	4	5	6	7	8
pred[]	2	N	6	3	1	2	6	7
<i>dist</i> []	1	0	2	3	2	1	2	3



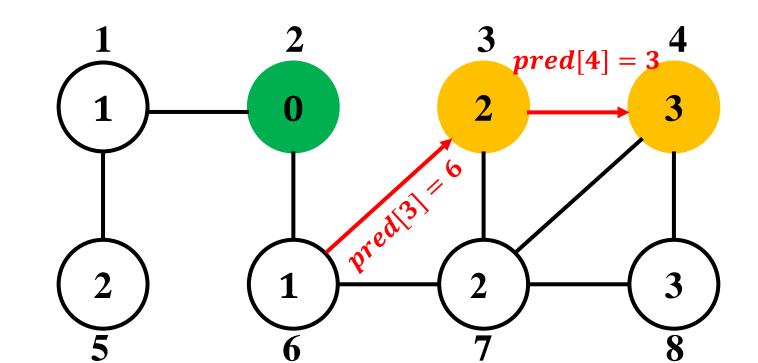


$oldsymbol{V}$	1	2	3	4	5	6	7	8
pred []	2	N	6	3	1	2	6	7
dist[]	1	0	2	3	2	1	2	3



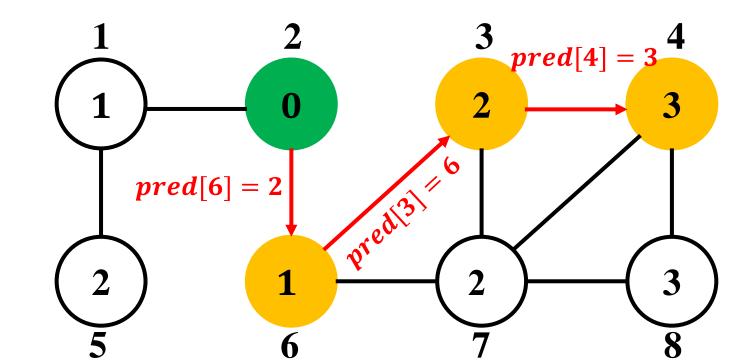


$oldsymbol{V}$	1	2	3	4	5	6	7	8
<pre>pred[]</pre>	2	N	6	3	1	2	6	7
<i>dist</i> []	1	0	2	3	2	1	2	3



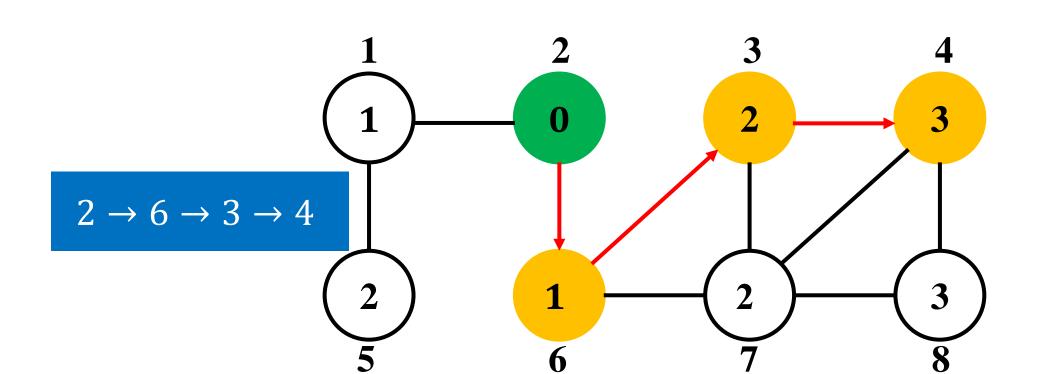


V	1	2	3	4	5	6	7	8
<pre>pred[]</pre>	2	N	6	3	1	2	6	7
dist[]	1	0	2	3	2	1	2	3





$oldsymbol{V}$	1	2	3	4	5	6	7	8
<pre>pred[]</pre>	2	N	6	3	1	2	6	7
<i>dist</i> []	1	0	2	3	2	1	2	3

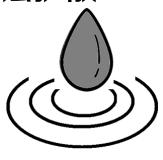


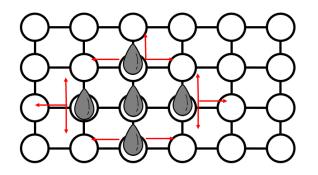
小结



• 广度优先搜索

• 算法核心思想: 逐层扩散



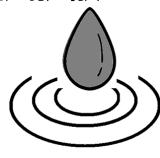


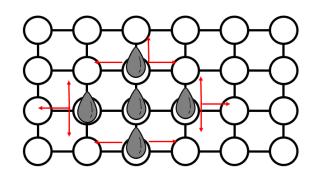
小结



• 广度优先搜索

• 算法核心思想: 逐层扩散



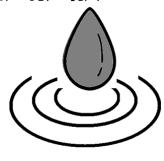


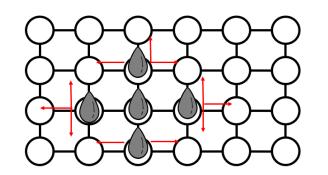
• 算法运行时间: O(|V| + |E|),简记为O(V + E)



• 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

