

Design and Analysis of Algorithms

Part IV: Graph Algorithms

Lecture 29: Single Source Shortest Path: Dijkstra

童咏昕

**北京航空航天大学
计算机学院**



- 在算法课程第四部分“图算法”主题中，我们将主要聚焦于如下经典问题：

- Basic Concepts in Graph Algorithms (图算法的基本概念)
- Breadth-First Search (BFS, 广度优先搜索)
- Depth-First Search (DFS, 深度优先搜索)
- Cycle Detection (环路检测)
- Topological Sort (拓扑排序)
- Strongly Connected Components (强连通分量)
- Minimum Spanning Trees (最小生成树)
- Single Source Shortest Path (单源最短路径)**
- All-Pairs Shortest Paths (所有点对最短路径)
- Bipartite Graph Matching (二分图匹配)
- Maximum/Network Flows (最大流/网络流)



问题背景

算法思想

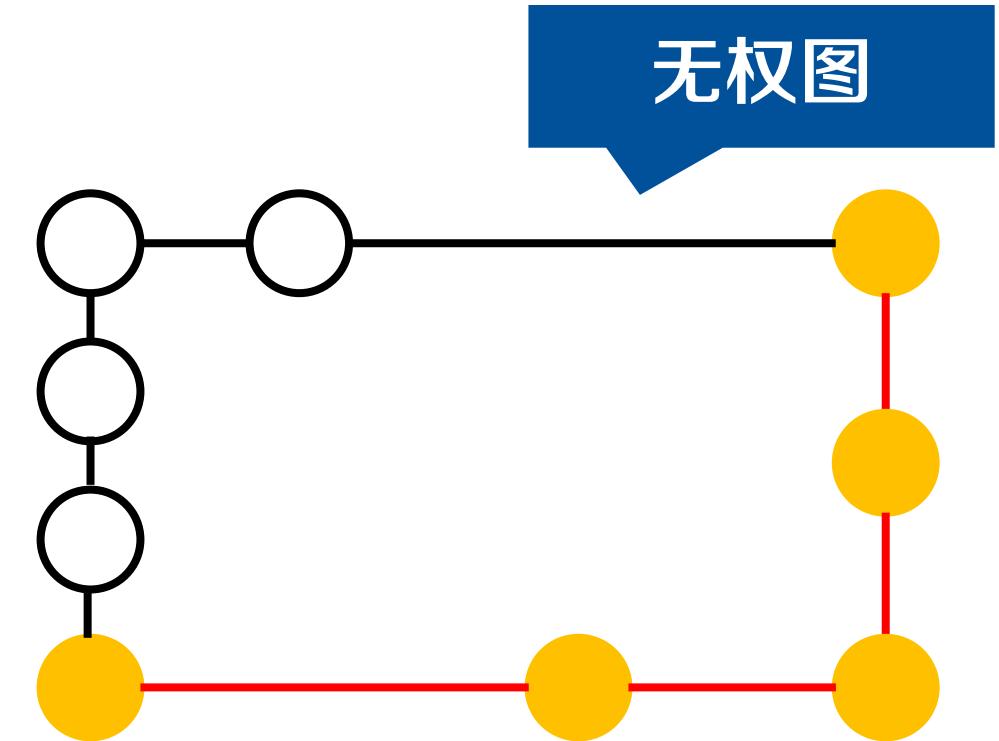
算法实例

算法分析

算法性质

问题背景

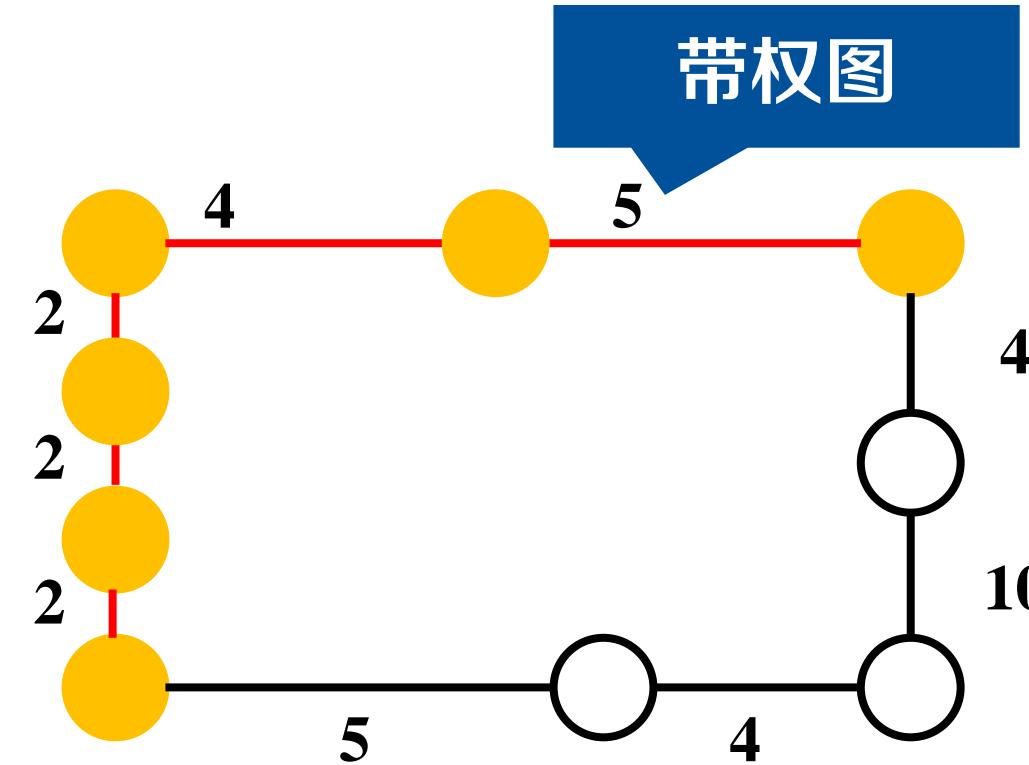
- 从知春路到其他站点，如何安排路线？



使用广度优先搜索求最短路径

问题背景

- 从知春路到其他站点，如何安排路线？



问题：如何计算**带权图**中源点到所有其他顶点的最短路径？



单源最短路径问题 (边权为正)

Single Source Shortest Paths Problem with Positive Weights

输入

- 带权图 $G = \langle V, E, W \rangle$, 其中 $w(u, v) \geq 0$ (图中所有边权为正), $(u, v) \in E$
- 源点编号 s

输出

- 源点 s 到所有其他顶点 t 的最短距离 $\delta(s, t)$ 和最短路径 $\langle s, \dots, t \rangle$



问题背景

算法思想

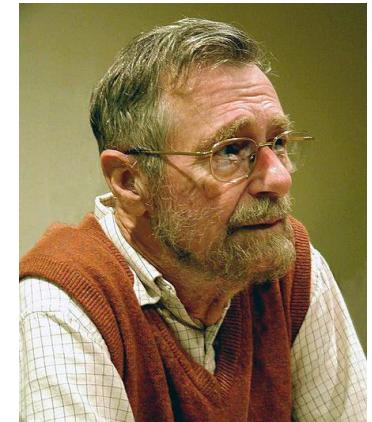
算法实例

算法分析

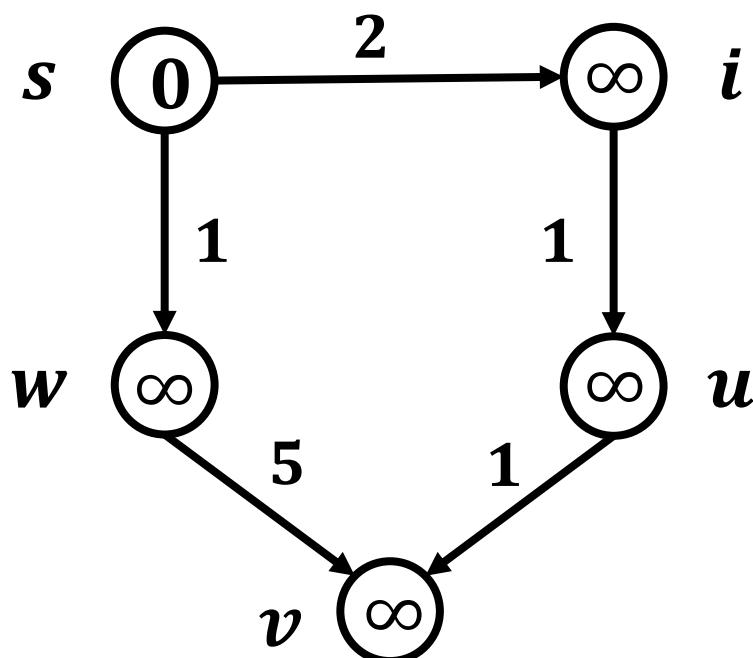
算法性质

辅助数组

- $dist$ 表示距离上界（估计距离）
 - 源点 s , $dist[s] = 0$; 其他顶点 u , $dist[u]$ 初始化为 ∞
 - $dist[u]$: 源点 s 到顶点 u 的距离上界, $\delta(s, u) \leq dist[u]$



Edsger W. Dijkstra



V	s	i	w	u	v
$dist$	0	∞	∞	∞	∞
δ	0	2	1	3	4

真实最短距离

- 辅助数组

- $dist$ 表示距离上界（估计距离）

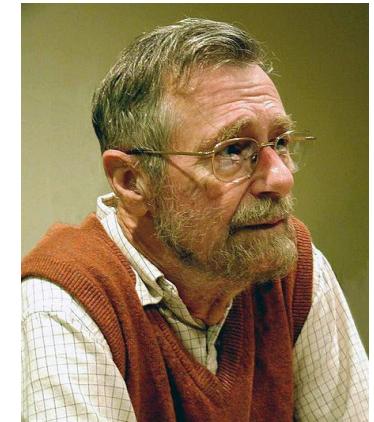
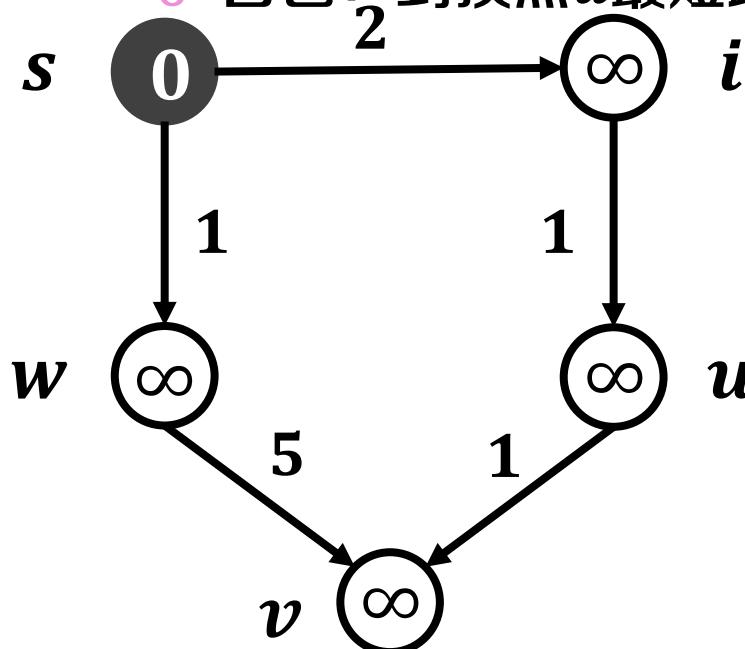
- 源点 s , $dist[s] = 0$; 其他顶点 u , $dist[u]$ 初始化为 ∞

- $dist[u]$: 源点 s 到顶点 u 的距离上界, $\delta(s, u) \leq dist[u]$

- $color$ 表示顶点状态

- 黑色: 到顶点 u 最短路已被确定

- 白色: 到顶点 u 最短路尚未确定

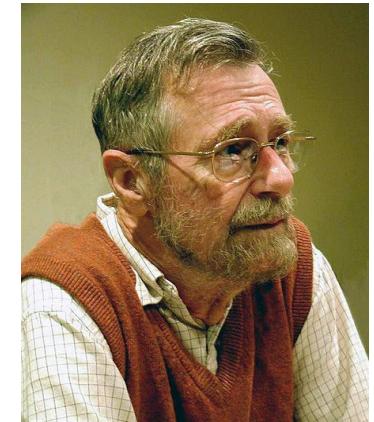


Edsger W. Dijkstra

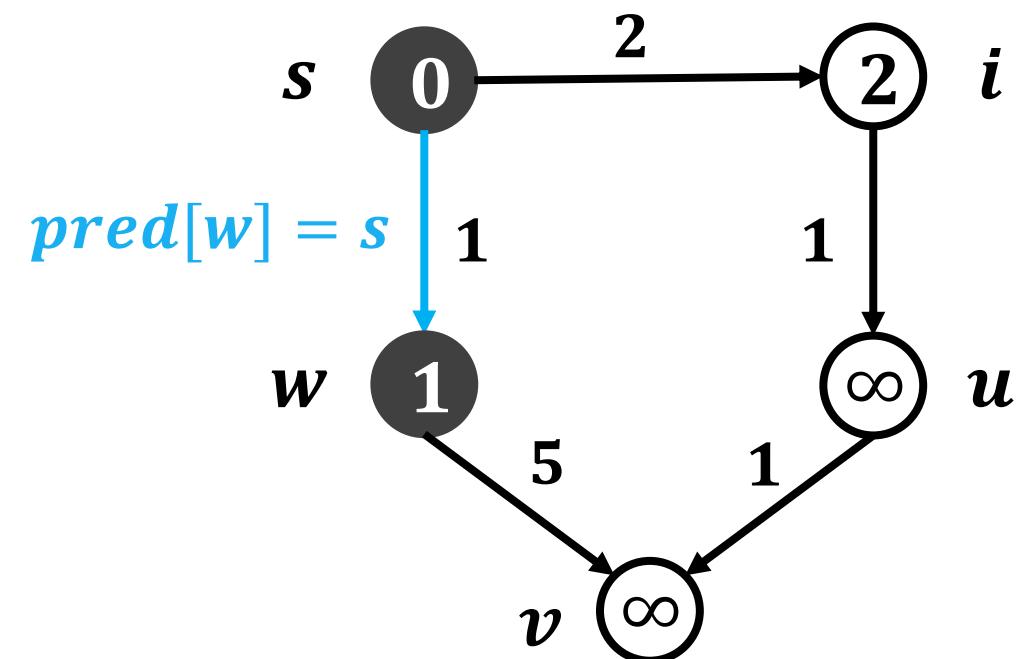
V	s	i	w	u	v
$dist$	0	∞	∞	∞	∞
δ	0	2	1	3	4

辅助数组

- $dist$ 表示距离上界（估计距离）
 - 源点 s , $dist[s] = 0$; 其他顶点 u , $dist[u]$ 初始化为 ∞
 - $dist[u]$: 源点 s 到顶点 u 的距离上界, $\delta(s, u) \leq dist[u]$
- $color$ 表示顶点状态
 - 黑色: 到顶点 u 最短路已被确定
 - 白色: 到顶点 u 最短路尚未确定
- $pred$ 表示前驱顶点
 - $(pred[u], u)$ 为最短路径上的边



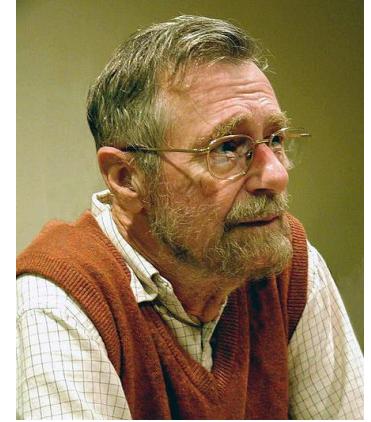
Edsger W. Dijkstra



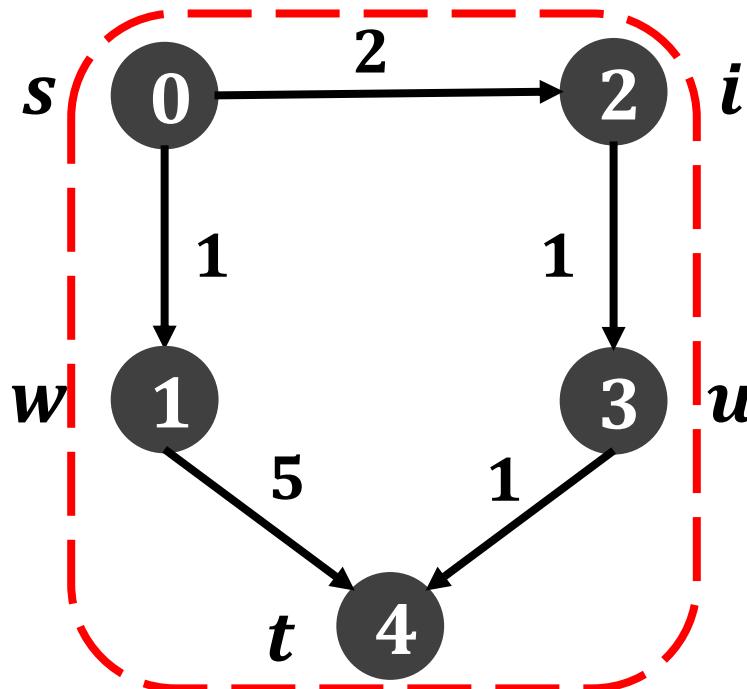
算法思想

核心思想

- 步骤1：新建空的黑色顶点集 V_A
- 步骤2：选择一个白色顶点变为黑色（到该顶点最短路被确定）
- 步骤3：重复步骤2，直至所有顶点均为黑色



Edsger W. Dijkstra



V_A 中顶点



$V - V_A$ 中顶点



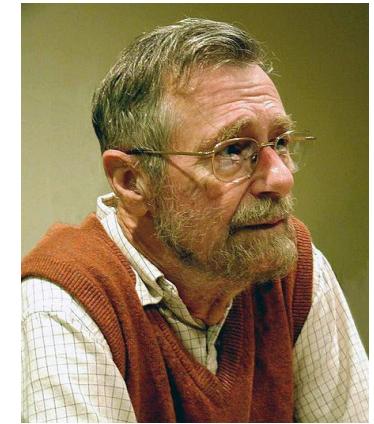
问题：选择哪个白色顶点变为黑色？

问题：如何更新每顶点的估计距离？

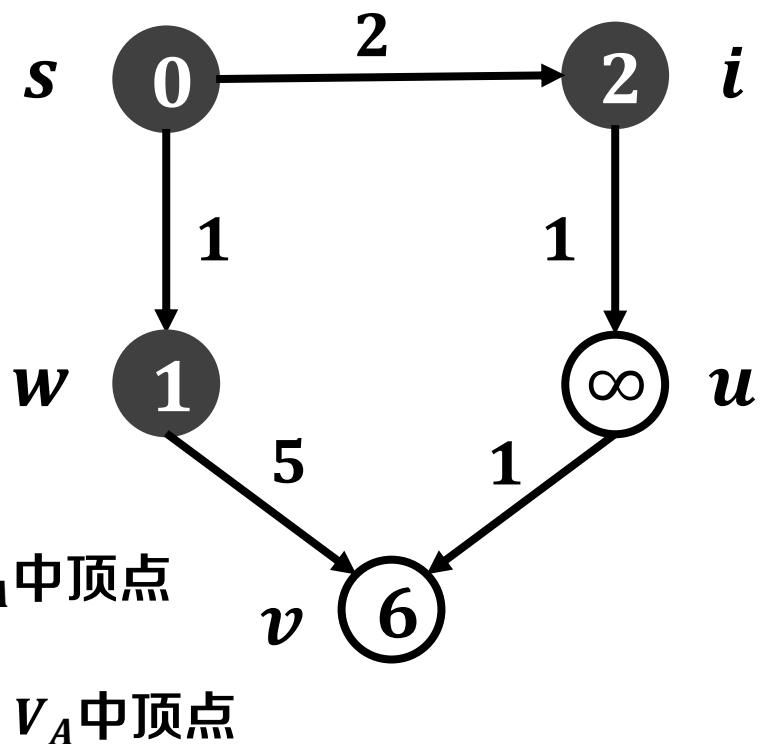
算法思想

- 问题1：选择哪个白色顶点变为黑色？采用贪心策略

- 对每个白色顶点 $y \in V - V_A$, 都有一个估计距离 $dist[y]$
- 选择估计距离最小的顶点 v , $dist[v] \leq dist[y], v, y \in V - V_A$



Edsger W. Dijkstra



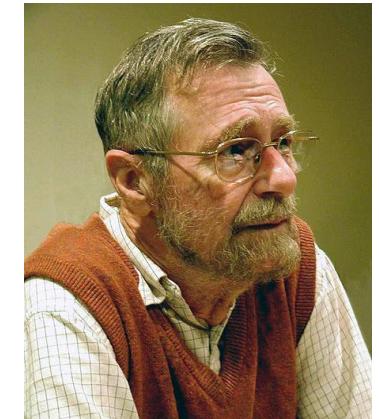
V	s	i	w	u	v
$dist$	0	2	1	∞	6
δ	0	2	1	3	4

算法思想

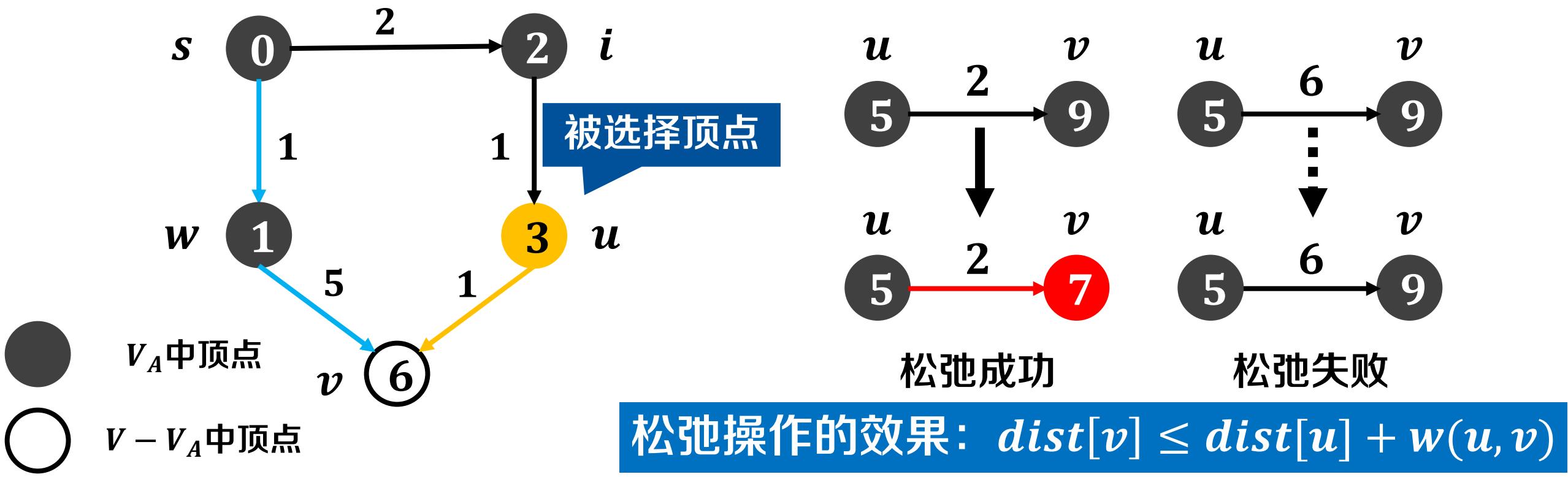
- 问题2：如何更新每顶点的估计距离？

- 当前到顶点 v 的最短路径： $< s, w, v >$ ，距离为 $dist[v]$
- 通过顶点 u 的新路径： $< s, \dots, u, v >$ ，距离为 $dist[u] + w(u, v)$
- 如果新路径更短($dist[u] + w(u, v) < dist[v]$)
 - 更新 $dist[v]$: $dist[v] = dist[u] + w(u, v)$

松弛操作



Edsger W. Dijkstra





问题背景

算法思想

算法实例

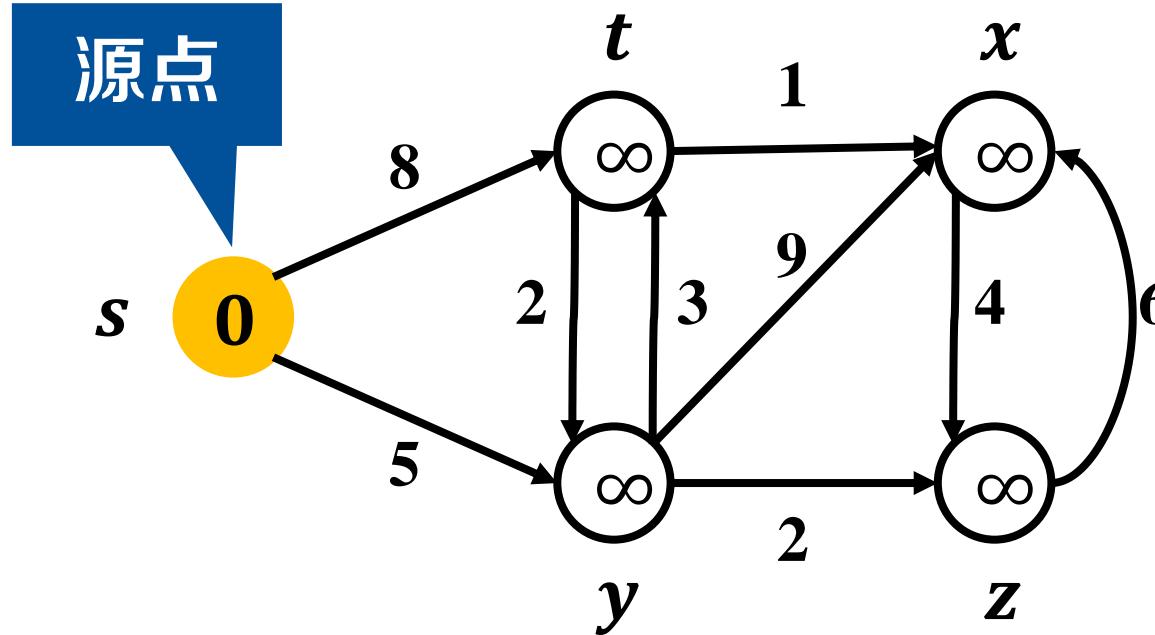
算法分析

算法性质

算法实例



V	s	t	x	y	z
$color$	W	W	W	W	W
$pred$	N	N	N	N	N
$dist$	0	∞	∞	∞	∞

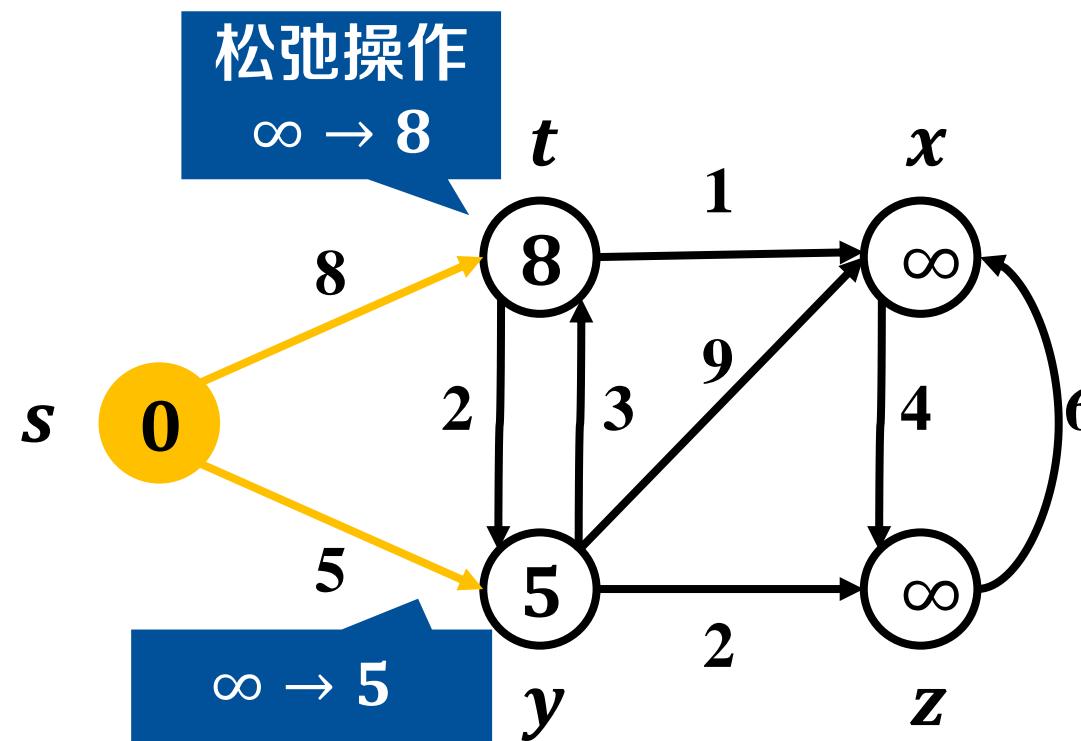


- v_A 中顶点
- $V - V_A$ 中顶点
- 被选中顶点

算法实例



V	s	t	x	y	z
$color$	W	W	W	W	W
$pred$	N	s	N	s	N
$dist$	0	8	∞	5	∞

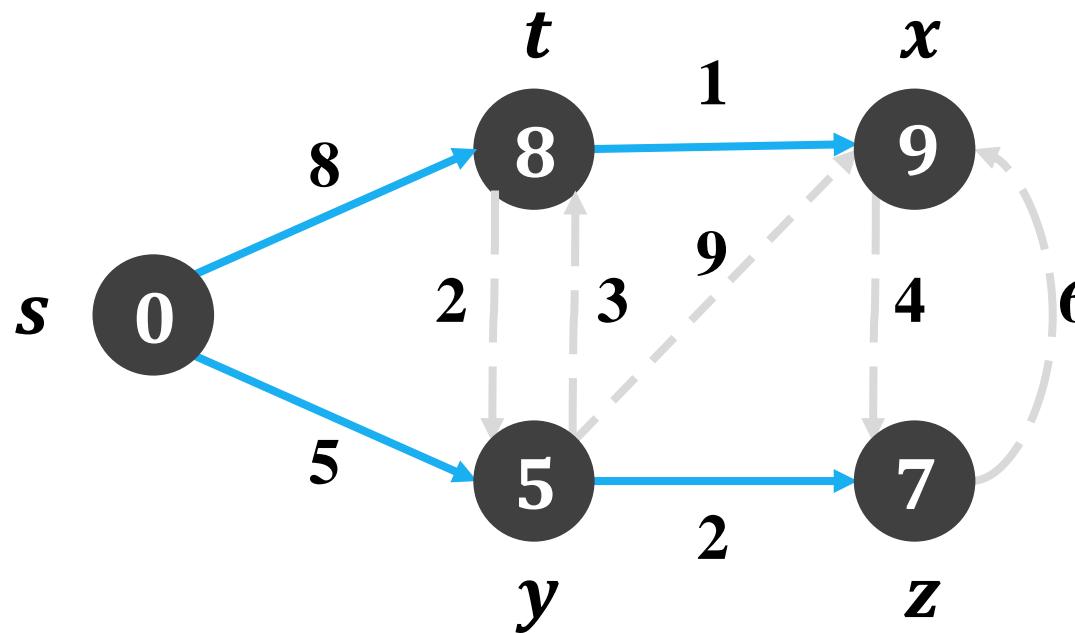


- V_A 中顶点
- $V - V_A$ 中顶点
- 被选中顶点

算法实例



V	s	t	x	y	z
$color$	B	B	B	B	B
$pred$	N	s	t	s	y
$dist$	0	8	9	5	7

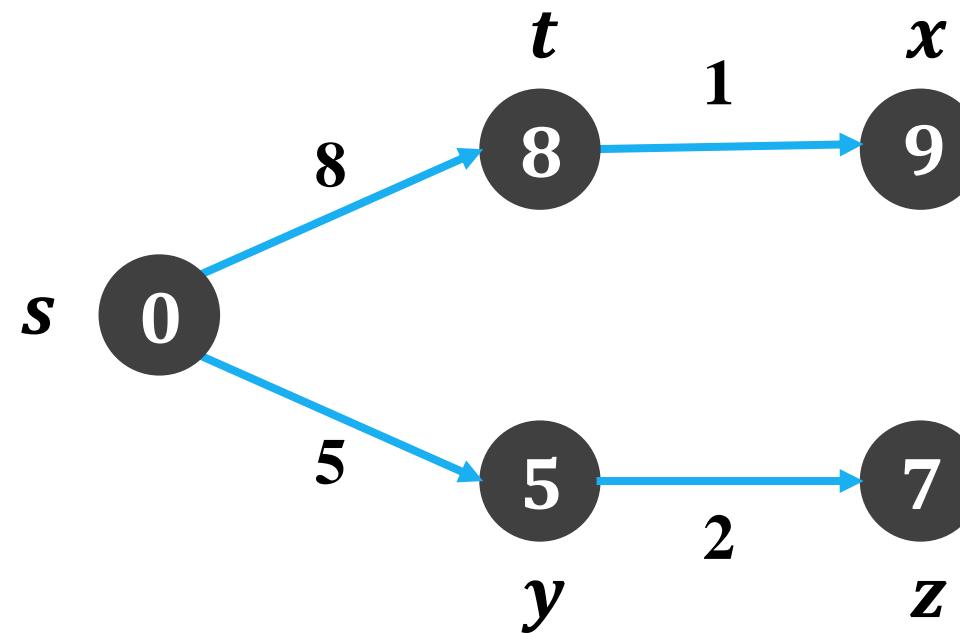


- v_A 中顶点
- $V - V_A$ 中顶点
- 被选中顶点

算法实例



V	s	t	x	y	z
$color$	B	B	B	B	B
$pred$	N	s	t	s	y
$dist$	0	8	9	5	7



- v_A 中顶点
- $V - V_A$ 中顶点
- 被选中顶点



问题背景

算法思想

算法实例

算法分析

算法性质



伪代码

- Dijkstra(G, s)

输入: 图 $G = \langle V, E, W \rangle$, 源点 s

输出: 单源最短路径 P

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

//初始化

for $u \in V$ do

$color[u] \leftarrow WHITE$

$dist[u] \leftarrow \infty$

$pred[u] \leftarrow NULL$

end

$dist[s] \leftarrow 0$



伪代码

- Dijkstra(G, s)

```
//执行单源最短路径算法
for  $i \leftarrow 1$  to  $|V|$  do
     $minDist \leftarrow \infty$ 
     $rec \leftarrow 0$ 
    for  $j \leftarrow 1$  to  $|V|$  do
        if  $color[j] \neq BLACK$  and  $dist[j] < minDist$  then
             $minDist \leftarrow dist[j]$ 
             $rec \leftarrow j$ 
        end
    end
    for  $u \in G.Adj[rec]$  do
        if  $dist[rec] + w(rec, u) < dist[u]$  then
             $dist[u] \leftarrow dist[rec] + w(rec, u)$ 
             $pred[u] \leftarrow rec$ 
        end
    end
     $color[rec] \leftarrow BLACK$ 
end
```



时间复杂度分析

- Dijkstra(G, s)

输入: 图 $G = \langle V, E, W \rangle$, 源点 s

输出: 单源最短路径 P

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

//初始化

for $u \in V$ do

$color[u] \leftarrow WHITE$

$dist[u] \leftarrow \infty$

$pred[u] \leftarrow NULL$

end

$dist[s] \leftarrow 0$

$O(|V|)$



时间复杂度分析

- Dijkstra(G, s)

//执行单源最短路径算法

```
for  $i \leftarrow 1$  to  $|V|$  do
     $minDist \leftarrow \infty$ 
     $rec \leftarrow 0$ 
    for  $j \leftarrow 1$  to  $|V|$  do
        if  $color[j] \neq BLACK$  and  $dist[j] < minDist$  then
             $minDist \leftarrow dist[j]$ 
             $rec \leftarrow j$ 
        end
    end
    for  $u \in G.Adj[rec]$  do
        if  $dist[rec] + w(rec, u) < dist[u]$  then
             $dist[u] \leftarrow dist[rec] + w(rec, u)$ 
             $pred[u] \leftarrow rec$ 
        end
    end
     $color[rec] \leftarrow BLACK$ 
end
```

$$O(|V|^2)$$



时间复杂度分析

- Dijkstra(G, s)

//执行单源最短路径算法

```
for i ← 1 to |V| do
    minDist ← ∞
    rec ← 0
    for j ← 1 to |V| do
        if color[j] ≠ BLACK and dist[j] < minDist then
            minDist ← dist[j]
            rec ← j
        end
    end
    for u ∈ G.Adj[rec] do
        if dist[rec] + w(rec, u) < dist[u] then
            dist[u] ← dist[rec] + w(rec, u)
            pred[u] ← rec
        end
    end
    color[rec] ← BLACK
end
```

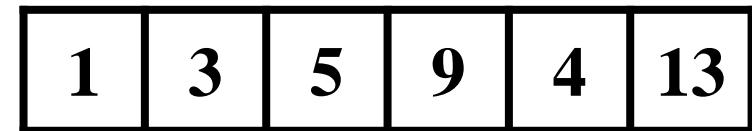
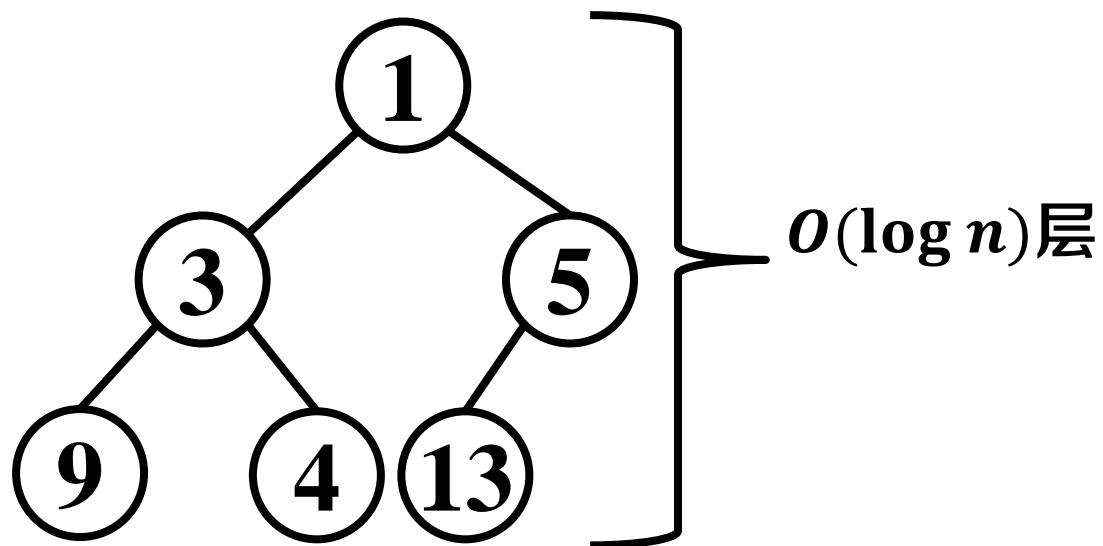


$O(\log|V|)$

使用优先队列，加速查询

优先队列

- 队列中每个元素有一个关键字，依据**关键字大小**离开队列
- 通过二叉堆来实现优先队列
 - $Q.Insert()$ 时间复杂度 $O(\log n)$
 - $Q.ExtractMin()$ 时间复杂度 $O(\log n)$
 - $Q.DecreaseKey()$ 时间复杂度 $O(\log n)$



优先队列 Q



伪代码

- Dijkstra-PriQueue(G, s)

输入: 图 $G = \langle V, E, W \rangle$, 源点 s

输出: 单源最短路径 P

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

新建空优先队列 Q

//初始化

for $u \in V$ do

$color[u] \leftarrow WHITE$

$dist[u] \leftarrow \infty$

$pred[u] \leftarrow NULL$

end

$dist[s] \leftarrow 0$

$Q.Insert(V, dist)$



伪代码

- Dijkstra-PriQueue(G, s)

```
//执行单源最短路径算法
while 优先队列Q非空 do
     $v \leftarrow Q.ExtractMin()$ 
    for  $u \in G.adj[v]$  do
        if  $dist[v] + w(v, u) < dist[u]$  then
             $dist[u] \leftarrow dist[v] + w(v, u)$ 
             $pred[u] \leftarrow v$ 
             $Q.DecreaseKey((u, dist[u]))$ 
        end
    end
     $color[v] \leftarrow BLACK$ 
end
```



时间复杂度分析

- Dijkstra-PriQueue(G, s)

输入: 图 $G = \langle V, E, W \rangle$, 源点 s

输出: 单源最短路径 P

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

新建空优先队列 Q

//初始化

```
for  $u \in V$  do
     $color[u] \leftarrow WHITE$ 
     $dist[u] \leftarrow \infty$ 
     $pred[u] \leftarrow NULL$ 
end
 $dist[s] \leftarrow 0$ 
 $Q.Insert(V, dist)$ 
```

} $O(|V|)$



时间复杂度分析

- Dijkstra-PriQueue(G, s)

```
//执行单源最短路径算法
while 优先队列Q非空 do
     $v \leftarrow Q.ExtractMin()$  -----  $O(\log|V|)$ 
    for  $u \in G.adj[v]$  do
        if  $dist[v] + w(v, u) < dist[u]$  then
             $dist[u] \leftarrow dist[v] + w(v, u)$ 
             $pred[u] \leftarrow v$ 
             $Q.DecreaseKey((u, dist[u]))$ 
        end
    end
     $color[v] \leftarrow BLACK$ 
end
```

$O(|V| \log|V|)$

$O(\deg(u) \cdot \log|V|)$

$O(|E| \log|V|)$

$$\sum_{u \in V} \deg(u) = 2|E|$$



时间复杂度分析

- Dijkstra-PriQueue(G, s)

```
//执行单源最短路径算法
while 优先队列Q非空 do
     $v \leftarrow Q.ExtractMin()$ 
    for  $u \in G.adj[v]$  do
        if  $dist[v] + w(v, u) < dist[u]$  then
             $dist[u] \leftarrow dist[v] + w(v, u)$ 
             $pred[u] \leftarrow v$ 
             $Q.DecreaseKey((u, dist[u]))$ 
        end
    end
     $color[v] \leftarrow BLACK$ 
end
```

时间复杂度 $O(|E| \cdot \log|V|)$



问题背景

算法思想

算法实例

算法分析

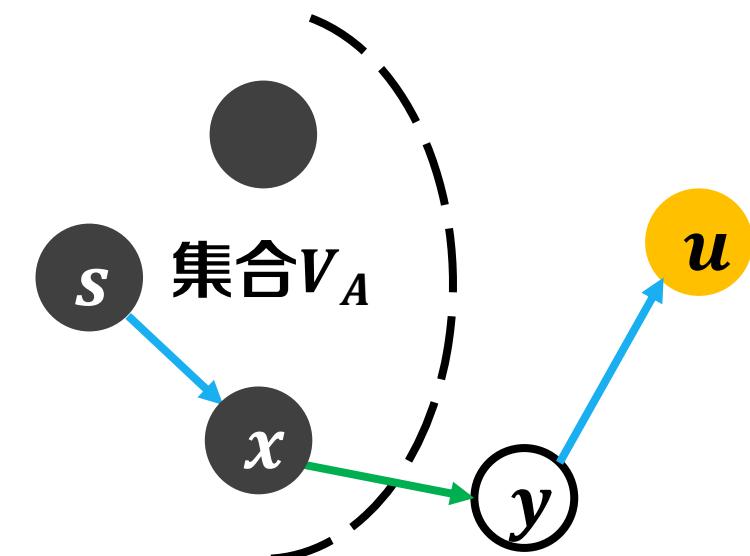
算法性质

正确性证明



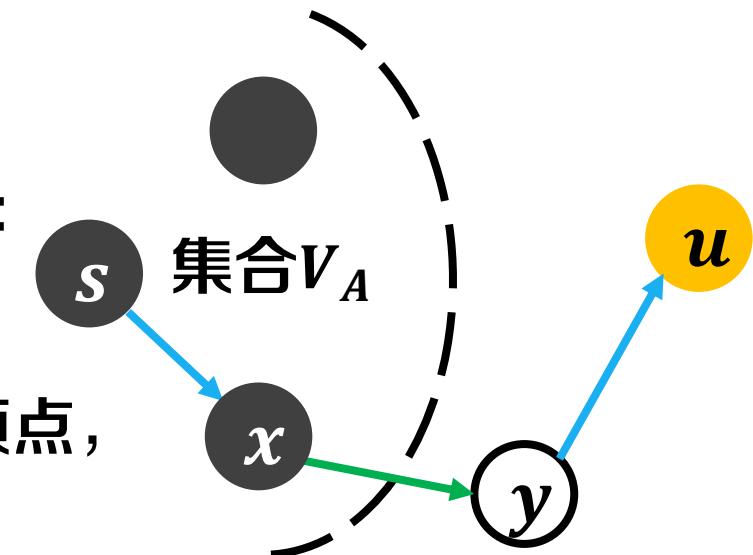
- 定理：Dijkstra算法中，顶点 u 被添加到 V_A 时， $dist[u] = \delta(s, u)$
- 证明：
 - 采用反证法，假设Dijkstra算法将顶点 u 添加到 V_A 时， $dist[u] \neq \delta(s, u)$
 - 由于 $dist[u]$ 作为 $\delta(s, u)$ 的上界，故 $dist[u] > \delta(s, u)$
 - 应存在一条长度为 $\delta(s, u)$ 的从 s 到 u 的最短路径，不妨设其为 $< s, \dots, x, y, \dots, u >$ ，其中边 (x, y) 横跨 $< V_A, V - V_A >$ ， $x \in V_A, y \in V - V_A$
 - 算法令 V_A 中的顶点满足： $dist[x] = \delta(s, x), x \in V_A$

问题： $dist[y]$ 和 $\delta(s, y)$ 具有何种关系？



正确性证明

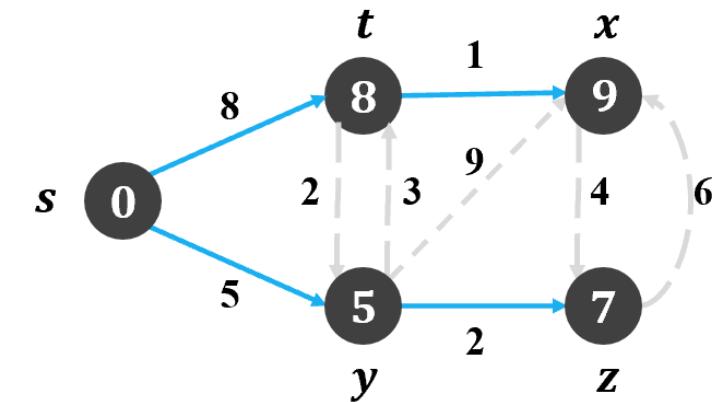
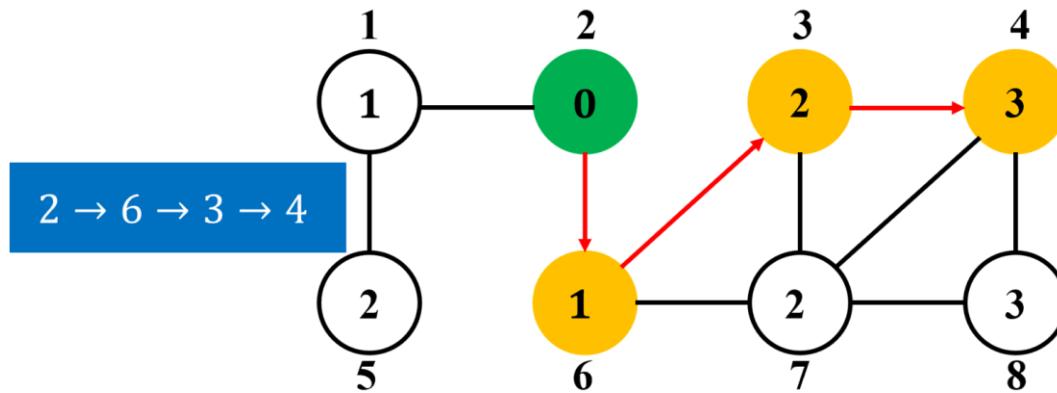
- 定理：Dijkstra算法中，顶点 u 被添加到 V_A 时， $dist[u] = \delta(s, u)$
- 证明（接上页）：
 - $< s, \dots, x, y >$ 是最短路径 $< s, \dots, x, y, \dots, u >$ 的子路径，故：
 - $\delta(s, y) = \delta(s, x) + w(x, y) = dist[x] + w(x, y)$ （公式1）
 - 算法对顶点 x 出发的所有边（包括边 (x, y) ）已进行松弛操作，故：
 - $dist[y] \leq dist[x] + w(x, y)$ （公式2）
 - 合并上述公式1和公式2，可得 $dist[y] = \delta(s, y)$
 - 最短路径 $< s, \dots, x, y, \dots, u >$ 中， y 出现在 u 之前，故：
 - $dist[u] > \delta(s, u) \geq \delta(s, y) = dist[y]$
 - $dist[u] > dist[y]$, $u \neq y$, u 不应是下一个被添加顶点，故产生矛盾



小结



	广度优先搜索	Dijkstra算法
适用范围	无权图	带权图 (所有边权为正)
数据结构	队列	优先队列
运行时间	$O(V + E)$	$O(E \cdot \log V)$





謝謝

