Dijkstra's Algorithm

Subhabrata Samajder

IIIT, Delhi
Winter Semester,
2$^{\text{nd}}$ June, 2023

Shortest Paths Problem

Motivation

Problem: Suppose a motorist wants to find the shortest possible route from Delhi to Kolkata. Given a road map of the India on which the distance between each pair of adjacent intersections is marked, how can the motorist determine the shortest route?

Motivation

Problem: Suppose a motorist wants to find the shortest possible route from Delhi to Kolkata. Given a road map of the India on which the distance between each pair of adjacent intersections is marked, how can the motorist determine the shortest route?

Possible Solutions:

- Enumerate all the routes from Delhi to Kolkata.
- Add up the distances on each route.
- Select the shortest route.
- **Issues:**
  - Even if we disallow routes that contain cycles, there are millions of possibilities.
  - Among them most of them are simply not worth considering.

    Example: A route from Delhi to Mumbai to Kolkata.

Motivation

Problem: Suppose a motorist wants to find the shortest possible route from Delhi to Kolkata. Given a road map of the India on which the distance between each pair of adjacent intersections is marked, how can the motorist determine the shortest route?

Possible Solutions:

- Modelling as a Graph Problem:
  - Vertices: Represents the intersections of road.
  - Edges: Represents the road segments between intersections.
  - Edge Weights: Represents the road distances from one intersection to another.
  - Goal: Find a *shortest path* from a given intersection in Delhi to a given intersection in Kolkata.

Shortest-paths Problem

Given: A weighted, directed graph $G = (V, E, w)$.

The weight of path $p = \langle v_0, v_1, \ldots, v_k \rangle$**:** $w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$.

Shortest-path weight from u **to** v**:**

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\leadsto} v\}; & \text{if } \exists\ u \overset{p}{\leadsto} v \\ \infty; & \text{otherwise.} \end{cases}$$

Shortest path from u **to** v**:** Any path p with $w(p) = \delta(u, v)$.

Note: BFS gives shortest-paths, but for unweighted graphs.

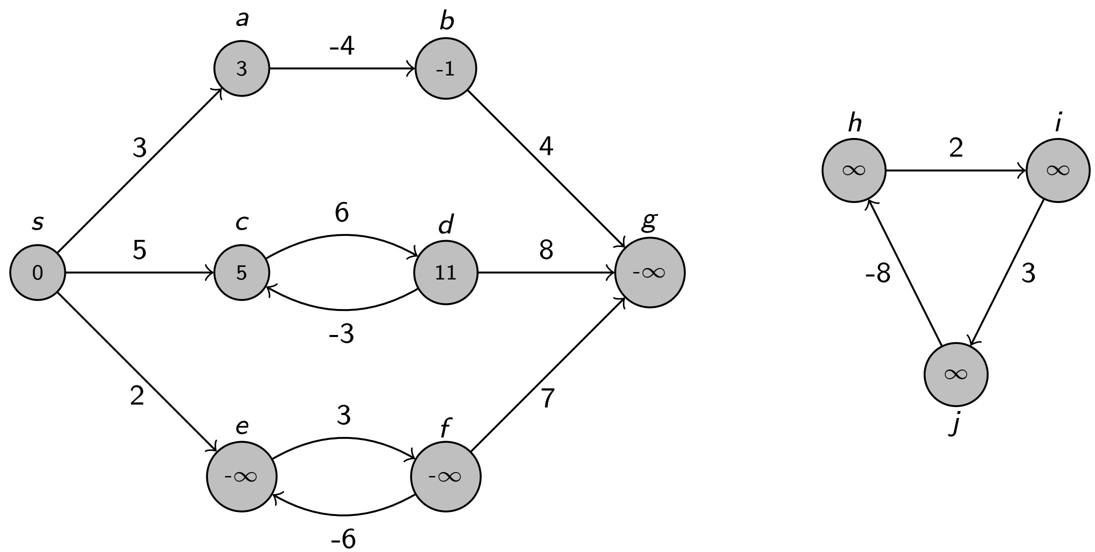
Single-source Shortest-paths Problem

Problem Statement: Given a weighted, directed graph $G = (V, E)$, with weight function $w : E \to \mathbb{R}$, find a shortest path from a given source vertex $s \in V$ to each vertex $v \in V$.

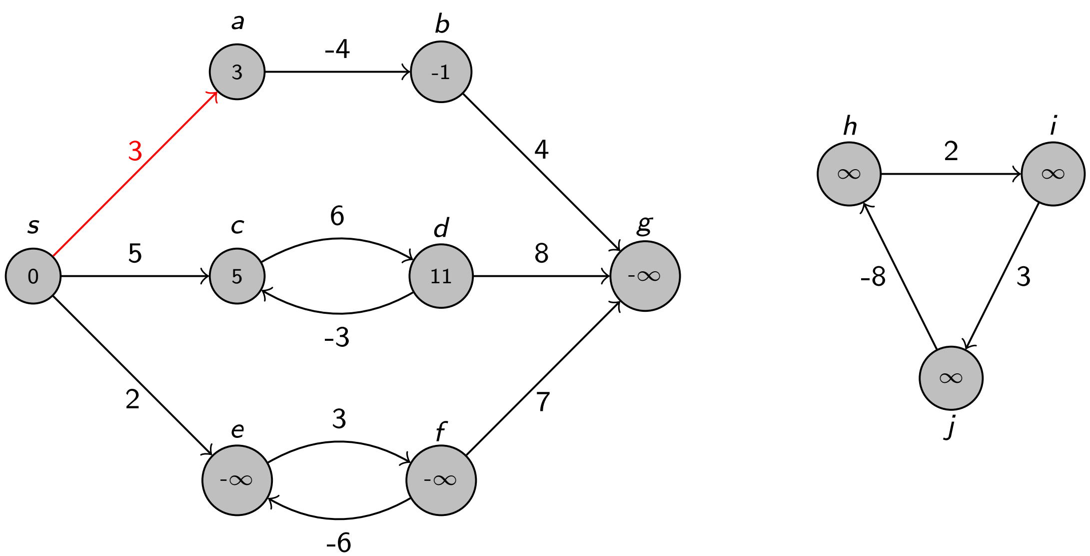
Other Related Problems

- Single-destination shortest-paths problem: Find a shortest path to a given destination vertex t from each vertex v.
  - Reverse the direction of each edge to reduce it to a single-source problem.
- Single-pair shortest-path problem: Given vertices u and v, find a shortest path from u to v.
  - Solve the single-source problem with source vertex u.
  - Stop the algorithm as soon as v is reached.
  - **Note:** No asymptotically faster algorithms are known in the worst case other than the best single-source algorithms.
- All-pairs shortest-paths problem: Find a shortest path from u to v for every pair of vertices u and v.
  - Run the single-source algorithm once from each vertex.
  - **Note:** Can usually be solved faster.

Negative-weight Edges

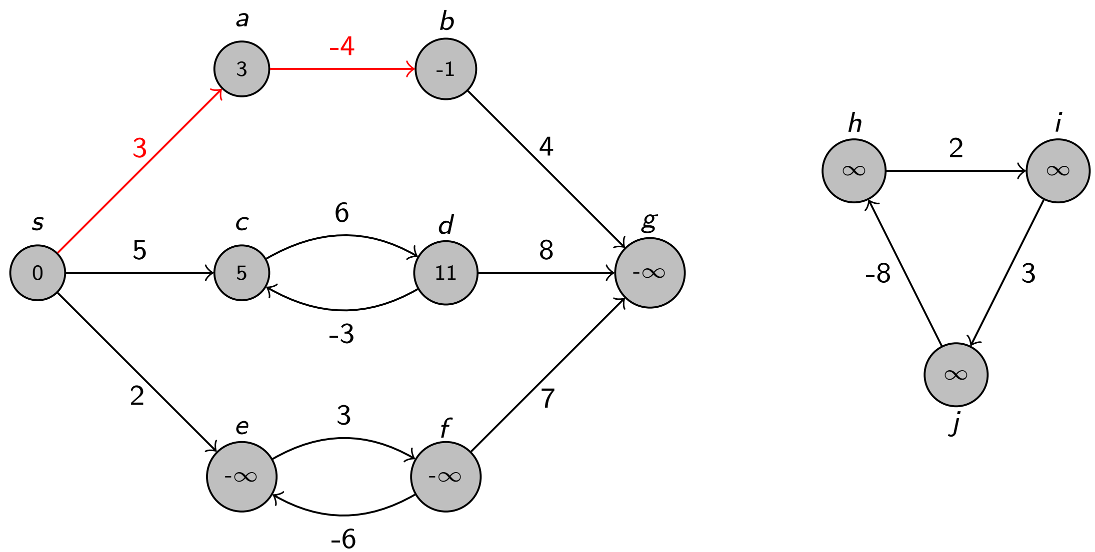


Negative-weight Edges



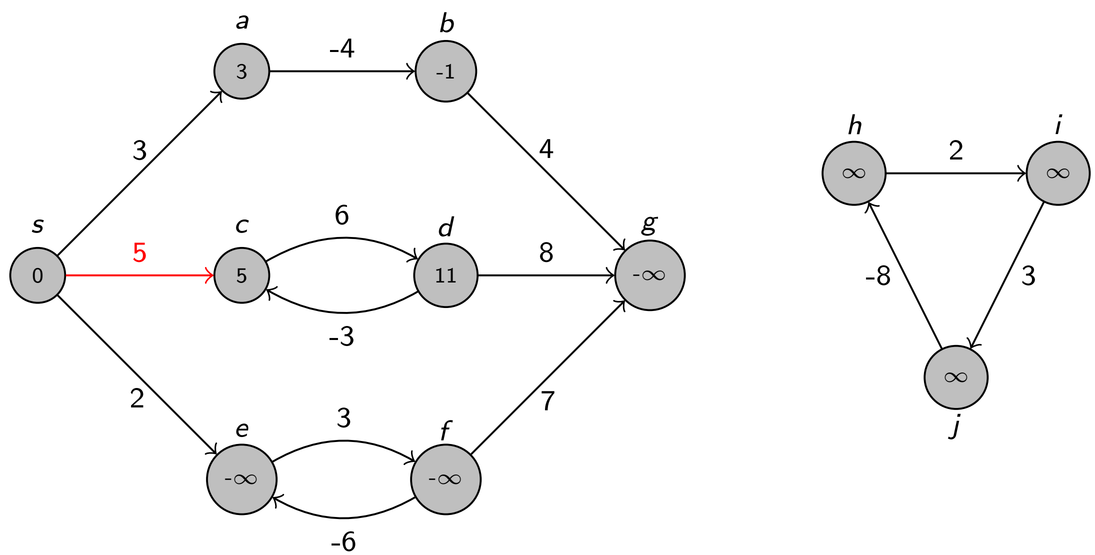
- $\delta(s, a) = w(s, a) = 3.$

Negative-weight Edges



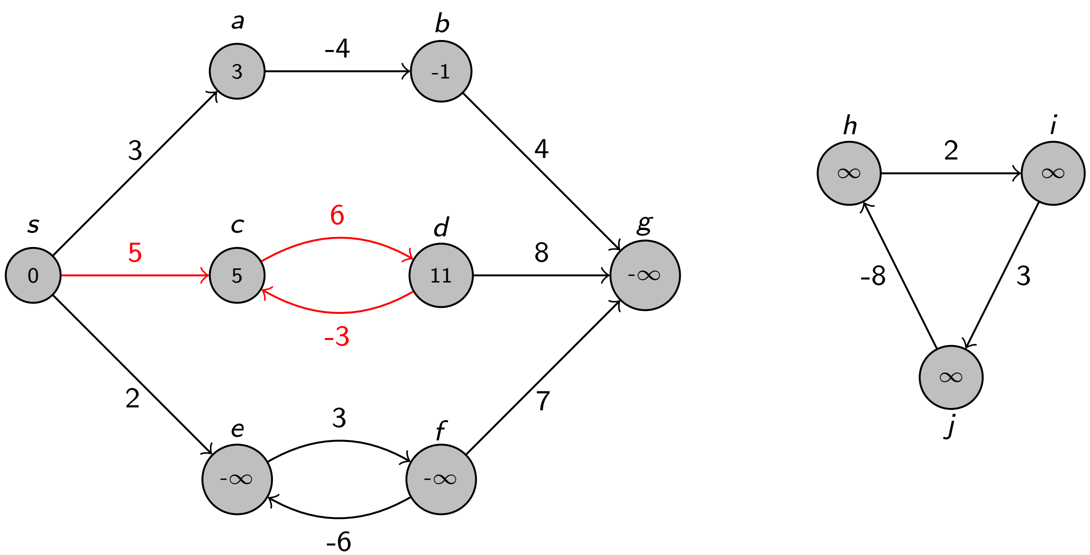
- $\delta(s,a) = w(s,a) = 3$.
- Similarly, $\delta(s,b) = w(s,a) + w(a,b) = 3 + (-4) = -1$.

Negative-weight Edges



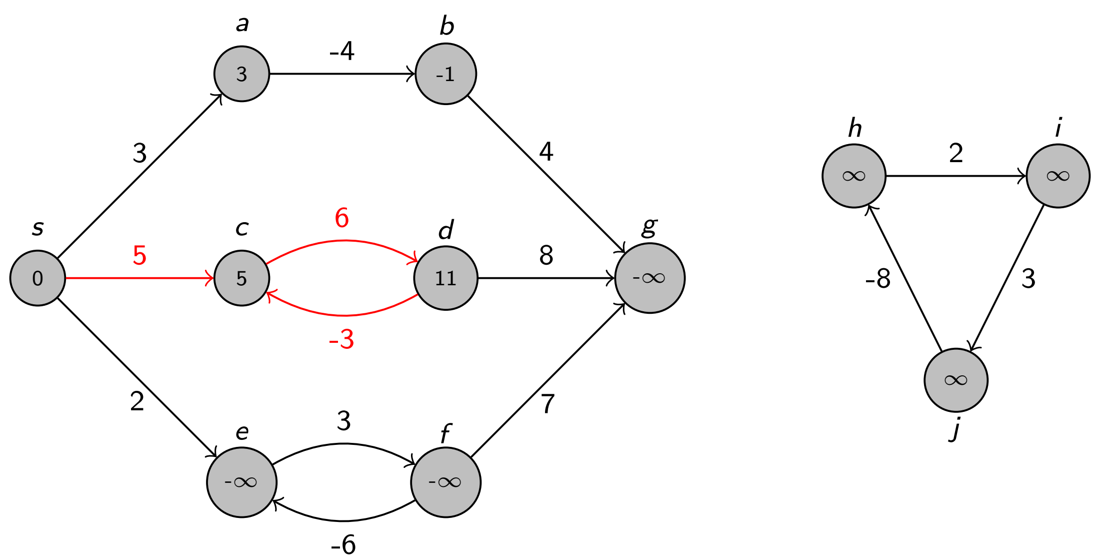
- **There are infinitely many paths from s to c: $\langle s, c \rangle$**

Negative-weight Edges



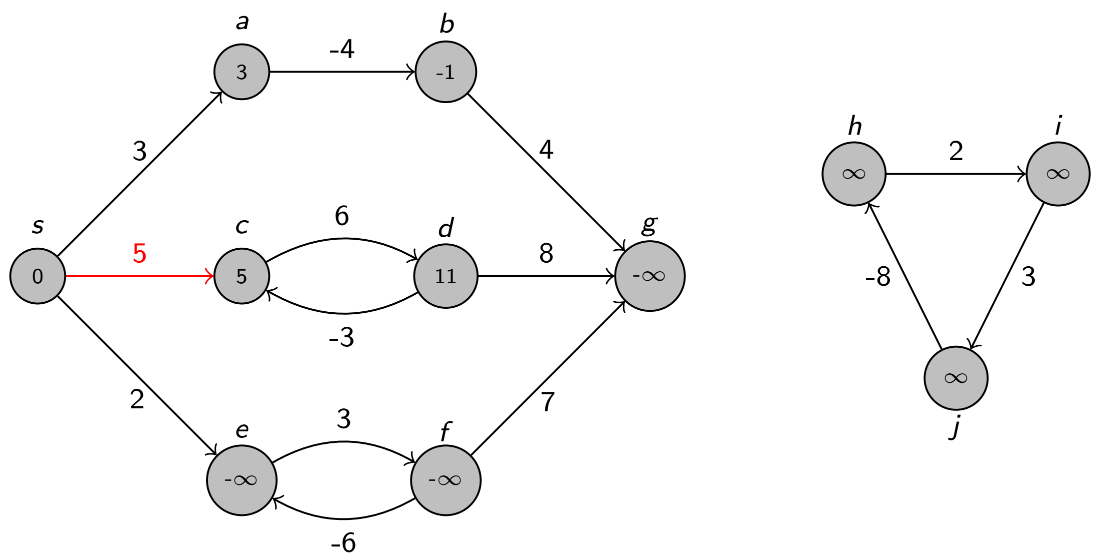
- **There are infinitely many paths from s to c:** $\langle s, c \rangle$, $\langle s, c, d, c \rangle$

Negative-weight Edges



- **There are infinitely many paths from s to c:** $\langle s, c \rangle$, $\langle s, c, d, c \rangle$, $\langle s, c, d, c, d, c \rangle$, and so on.

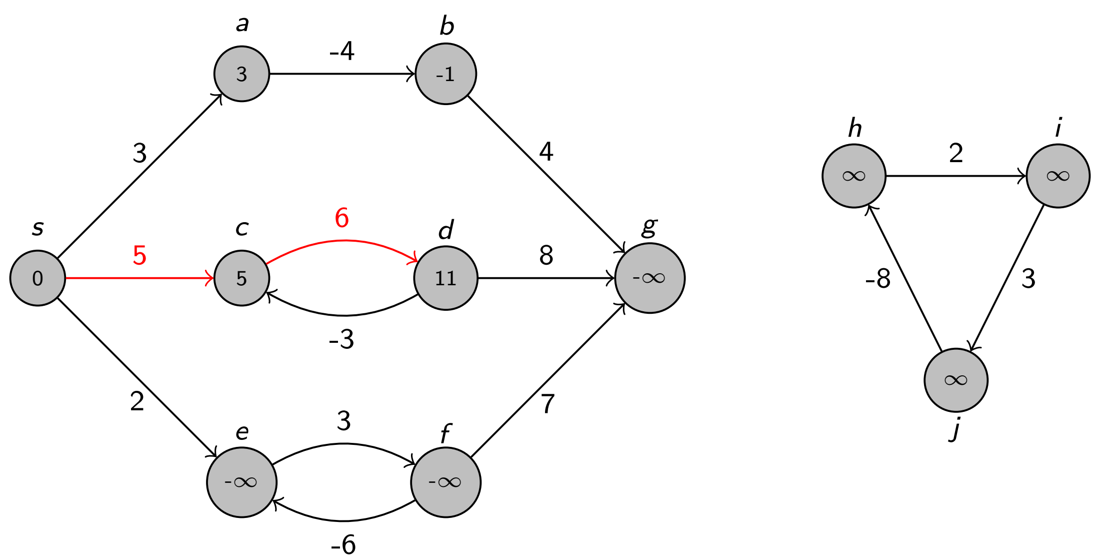
Negative-weight Edges



- **There are infinitely many paths from s to c:** $\langle s, c \rangle$, $\langle s, c, d, c \rangle$, $\langle s, c, d, c, d, c \rangle$, and so on.

  $\therefore\ \delta(s, c) = 5 \quad [\because\ w(\langle c, d, c \rangle) = 6 + (-3) = 3 > 0]$

Negative-weight Edges

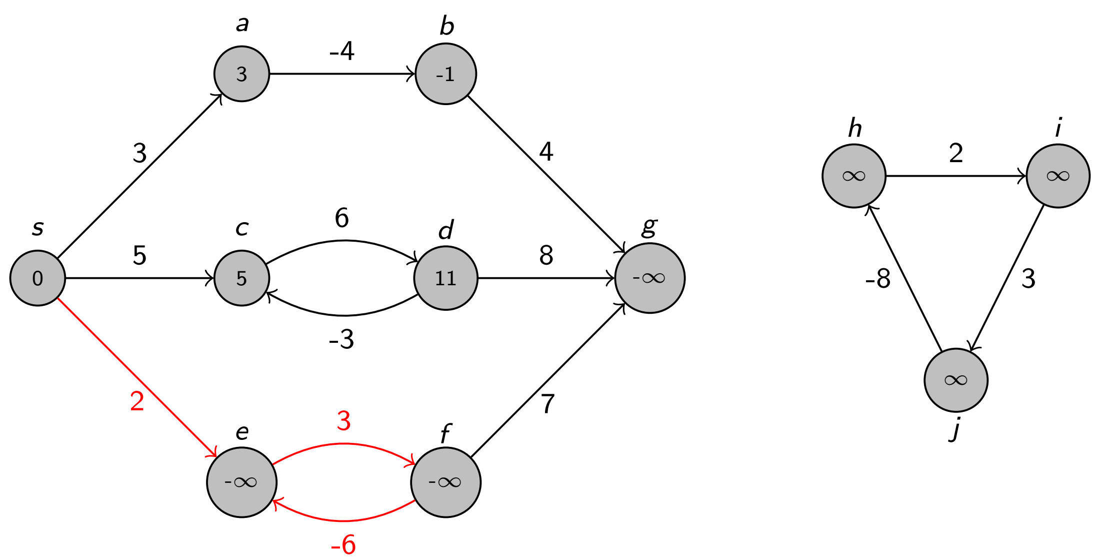


- **There are infinitely many paths from s to c:** $\langle s, c \rangle$, $\langle s, c, d, c \rangle$, $\langle s, c, d, c, d, c \rangle$, and so on.

  $\therefore \;\; \delta(s, c) = 5 \quad [\because \;\; w(\langle c, d, c \rangle) = 6 + (-3) = 3 > 0]$

  Similarly, $\delta(s, d) = w(s, c) + w(c, d) = 11$.

Negative-weight Edges

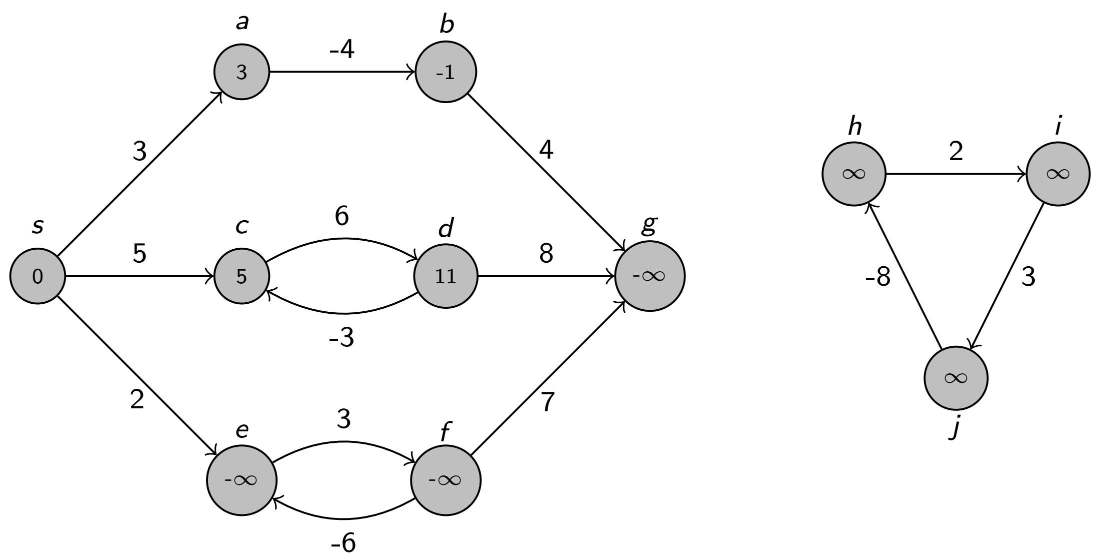


- **There are infinitely many paths from s to e:** $\langle s, e \rangle$, $\langle s, e, f, e \rangle$, $\langle s, e, f, e, f, e \rangle$, and so on.

  $\therefore\ \delta(s, e) = -\infty \quad [\because\ w(\langle e, f, f \rangle) = 3 + (-6) = -3 < 0]$

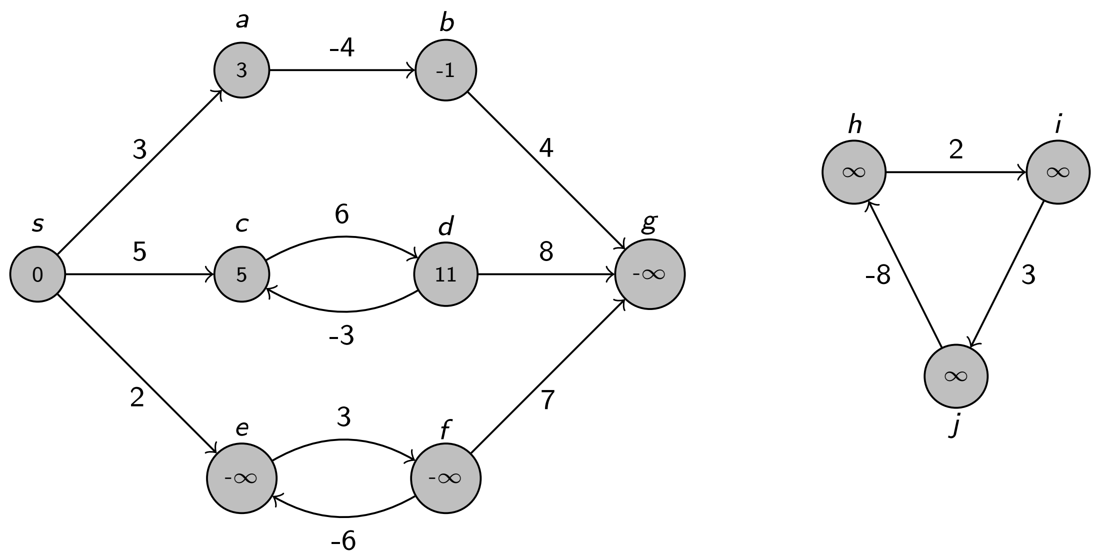
  Similarly, $\delta(s, f) = -\infty$.

Negative-weight Edges



- $\delta(s, g) = -\infty$: Because g is reachable from f

Negative-weight Edges



- **Note:** $w(\langle h, i, j \rangle) = 2 + 3 + (-8) = -3 < 0$.

  But they are not reachable from s, therefore

$$\delta(s, h) = \delta(s, i) = \delta(s, j) = \infty.$$

How Negative Cycles are Handled by Algorithms?

- Dijkstra's Algorithm: Assumes that all edge weights in the input graph are non-negative.

- Bellman-Ford Algorithm: Allows negative-weight edges.
  - Produces a correct answer as long as no negative-weight cycles are reachable from the source.
  - However, if there is such a negative-weight cycle, then the algorithm can detect and report its existence.

Dijkstra's Algorithm

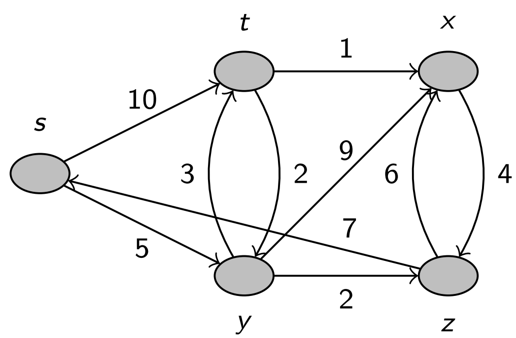
Introduction

- Was conceived by Edsger W. Dijkstra in 1956 but was published three years later.
- Many variants of the algorithm exists.
  - Original variant: Finds the shortest path between two nodes.
  - Common variant: Fixes a "source" node and finds shortest paths from the source to all other nodes (shortest-path tree).
- Original algorithm: Does not use a Min-Priority queue and runs in time $\mathcal{O}(|V|^2)$.
- Fredman and Tarjan (1984): Gave an efficient implementation based on a Min-Priority queue implemented by a Fibonacci heap and running in $\mathcal{O}(|E| + |V| \log |V|)$.
  - Asymptotically the fastest known single-source shortest-path algorithm for arbitrary directed graphs with unbounded non-negative weights.
- However, specialized cases (such as bounded/integer weights, directed acyclic graphs etc.) can be improved further.

Main Idea

- The algorithm maintains a set S of vertices whose final shortest-path weights from the source s have already been determined.
- The algorithm repeatedly selects the vertex $u \in V \setminus S$ with the minimum shortest-path estimate, adds u to S.
- Then relaxes (lines 11-13) all edges leaving u.
- It uses a greedy strategy.
- Bears some similarity with both BFS and Prim's algorithm.

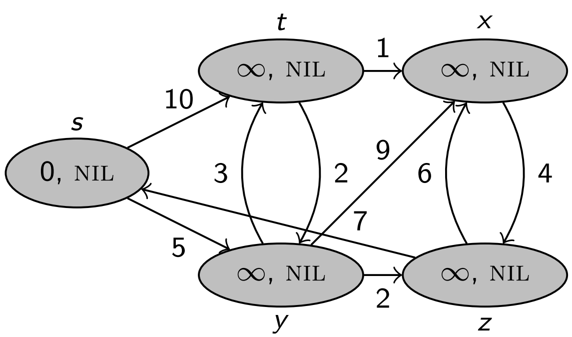
DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

DIJKSTRA(G, w, s)



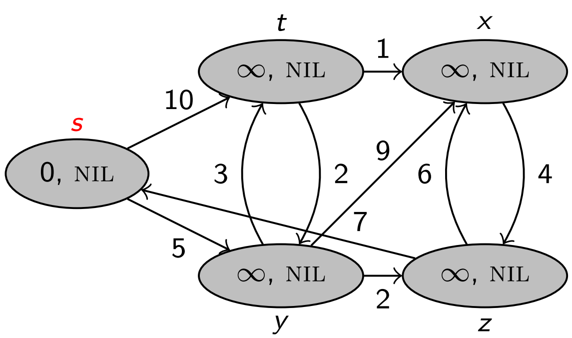
I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

Initialization Step:

1. **for** each $v \in V$
2. $d[v] \leftarrow \infty$
3. $\pi[v] \leftarrow$ NIL
4. $d[s] \leftarrow 0$
5. $S \leftarrow \emptyset$
6. $Q \leftarrow V$ // $Q(d)$: MIN-PRIORITY queue

$\textsc{Dijkstra}(G, w, s)$



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

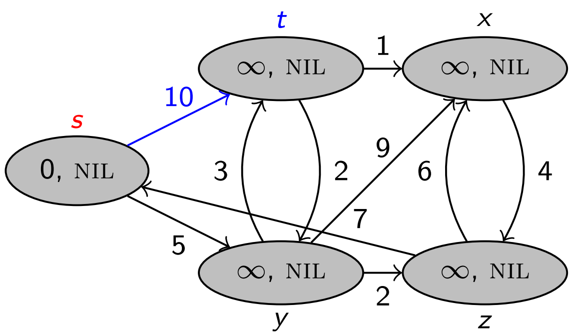
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
```

Dijkstra(G, w, s)



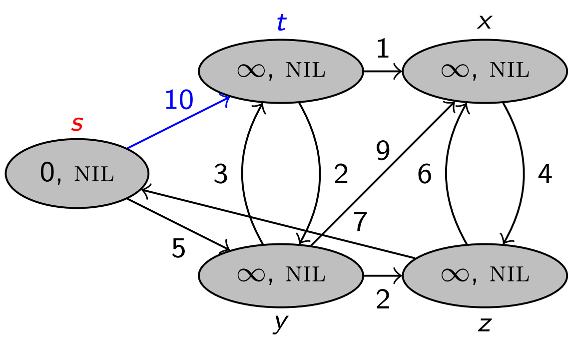
I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

```
     Initialization Step:
     for each v ∈ V
        d[v] ← ∞
        π[v] ← NIL
     d[s] ← 0
     S ← ∅
     Q ← V   // Q(d): MIN-PRIORITY queue

     Updation Step:
     while (Q ≠ ∅)
        u ← EXTRACT-MIN(Q)
        S ← S ∪ {u}
          for each vertex v ∈ Adj[u]
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

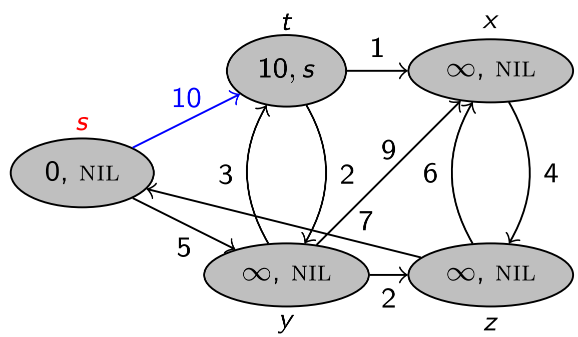
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
    for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u] + w(u, v))
```

$\textsc{Dijkstra}(G, w, s)$



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

Initialization Step:

for each $v \in V$

 $d[v] \leftarrow \infty$

 $\pi[v] \leftarrow \text{NIL}$

$d[s] \leftarrow 0$

$S \leftarrow \emptyset$

$Q \leftarrow V$ // $Q(d)$: $\textsc{Min-Priority}$ queue

Updation Step:

while $(Q \neq \emptyset)$

 $u \leftarrow \textsc{Extract-Min}(Q)$

 $S \leftarrow S \cup \{u\}$

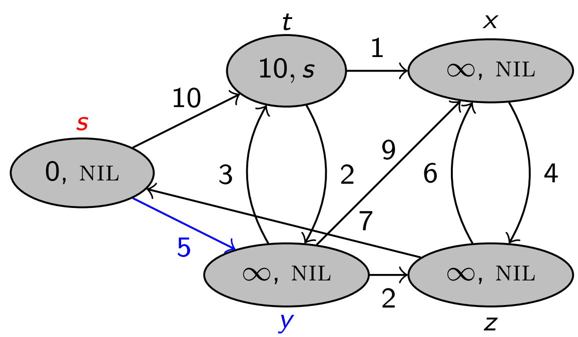
 for each vertex $v \in Adj[u]$

  if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$

   $d[v] \leftarrow d[u] + w(u, v)$

   $\pi[v] \leftarrow u$

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

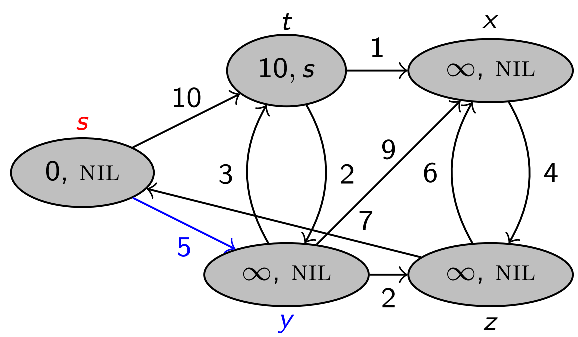
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V  // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
          d[v] ← d[u] + w(u, v)
          π[v] ← u
```

$\textsc{Dijkstra}(G, w, s)$



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

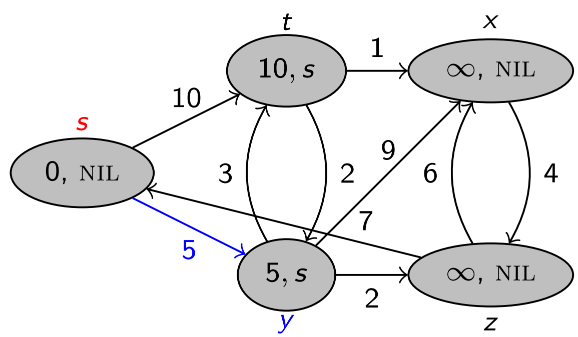
Initialization Step:

for each $v \in V$
    $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: $\textsc{Min-Priority}$ queue

Updation Step:

while $(Q \neq \emptyset)$
    $u \leftarrow \textsc{Extract-Min}(Q)$
    $S \leftarrow S \cup \{u\}$
    for each vertex $v \in Adj[u]$
        if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$
            $d[v] \leftarrow d[u] + w(u, v)$
            $\pi[v] \leftarrow u$

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

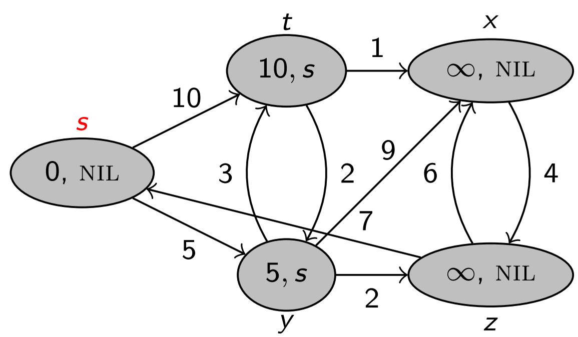
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
          if (v ∈ Q) and (d[v] > d[u]+w(u, v))
              d[v] ← d[u] + w(u, v)
              π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

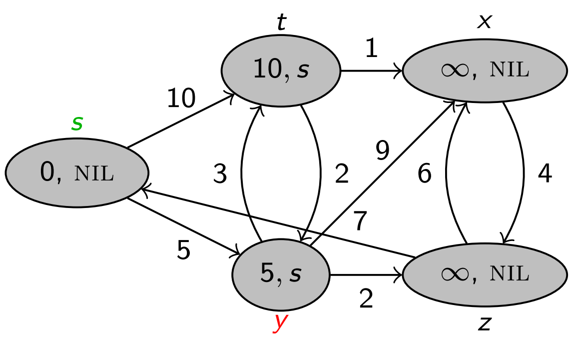
Initialization Step:

for each $v \in V$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: MIN-PRIORITY queue

Updation Step:

while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $S \leftarrow S \cup \{u\}$
   for each vertex $v \in Adj[u]$
    if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

Dijkstra(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

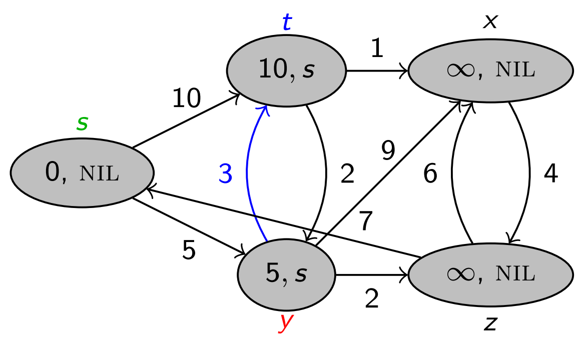
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
    for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

Dijkstra(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

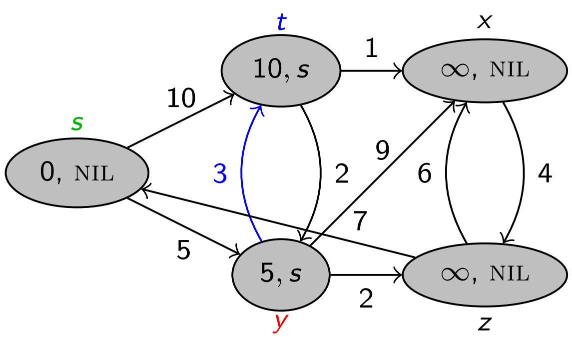
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V  // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
          d[v] ← d[u] + w(u, v)
          π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

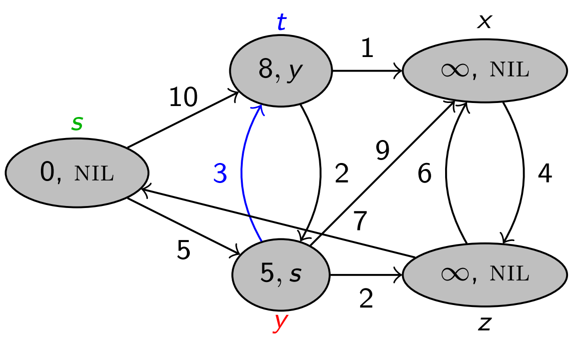
Initialization Step:

for each $v \in V$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: MIN-PRIORITY queue

Updation Step:

while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $S \leftarrow S \cup \{u\}$
  for each vertex $v \in Adj[u]$
    if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

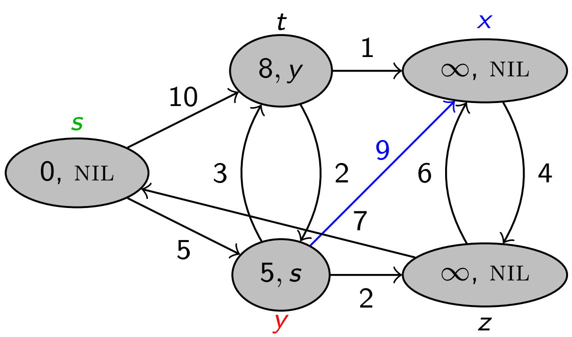
Initialization Step:

for each $v \in V$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: MIN-PRIORITY queue

Updation Step:

while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $S \leftarrow S \cup \{u\}$
  for each vertex $v \in Adj[u]$
    if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

DIJKSTRA(G, w, s)



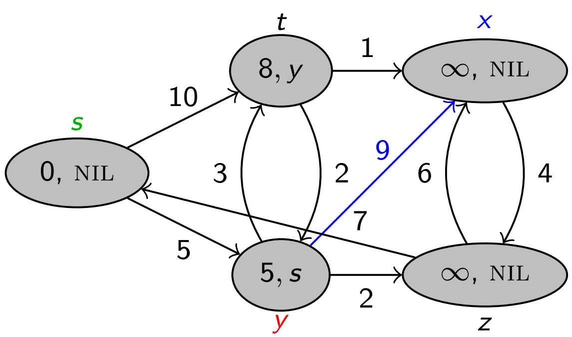
I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

```
    Initialization Step:
    for each v ∈ V
        d[v] ← ∞
        π[v] ← NIL
    d[s] ← 0
    S ← ∅
    Q ← V   // Q(d): MIN-PRIORITY queue

    Updation Step:
    while (Q ≠ ∅)
        u ← EXTRACT-MIN(Q)
        S ← S ∪ {u}
          for each vertex v ∈ Adj[u]
            if (v ∈ Q) and (d[v] > d[u] + w(u, v))
                d[v] ← d[u] + w(u, v)
                π[v] ← u
```

Dijkstra(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

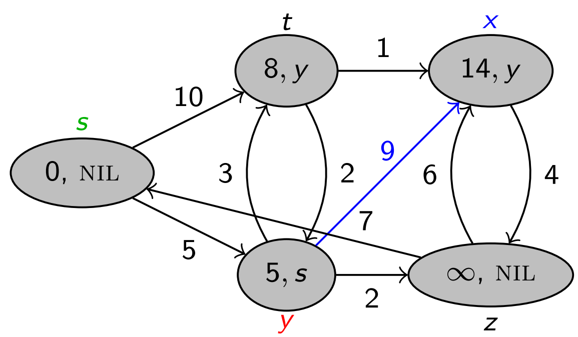
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
         if (v ∈ Q) and (d[v] > d[u]+w(u, v))
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$ and a source vertex s.

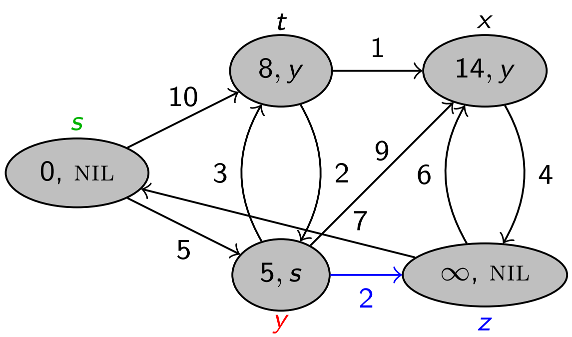
O/P: Shortest-path tree S with root vertex s.

Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
          d[v] ← d[u] + w(u, v)
          π[v] ← u
```



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

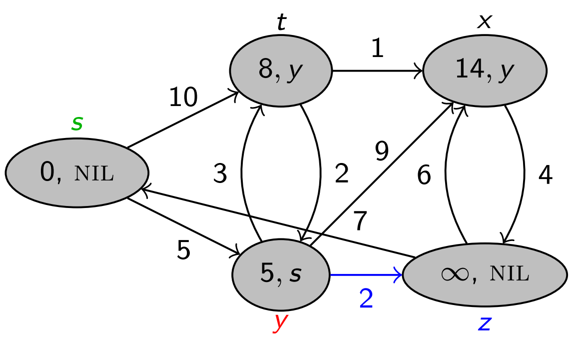
Initialization Step:

for each $v \in V$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: MIN-PRIORITY queue

Updation Step:

while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $S \leftarrow S \cup \{u\}$
   for each vertex $v \in Adj[u]$
    if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$
     $d[v] \leftarrow d[u] + w(u, v)$
     $\pi[v] \leftarrow u$

$\textsc{Dijkstra}(G, w, s)$



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

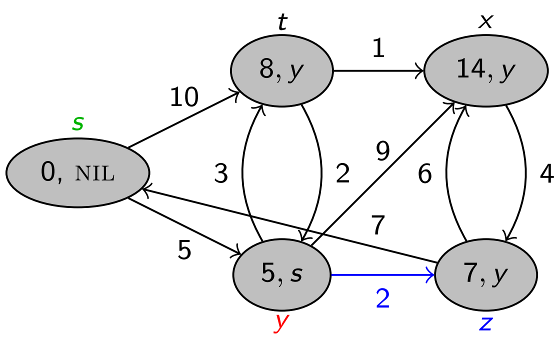
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
           d[v] ← d[u] + w(u, v)
           π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

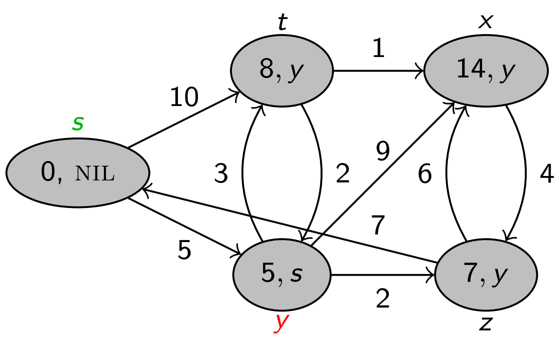
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
    for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

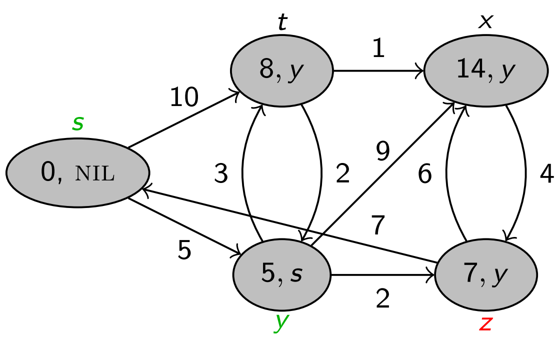
Initialization Step:

for each $v \in V$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: MIN-PRIORITY queue

Updation Step:

while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $S \leftarrow S \cup \{u\}$
    for each vertex $v \in Adj[u]$
      if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$
        $d[v] \leftarrow d[u] + w(u, v)$
        $\pi[v] \leftarrow u$

$\textsc{Dijkstra}(G, w, s)$



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

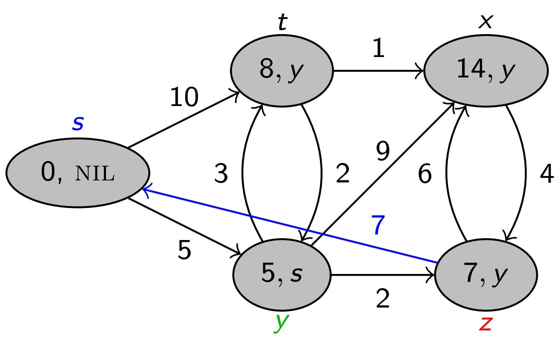
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
    for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

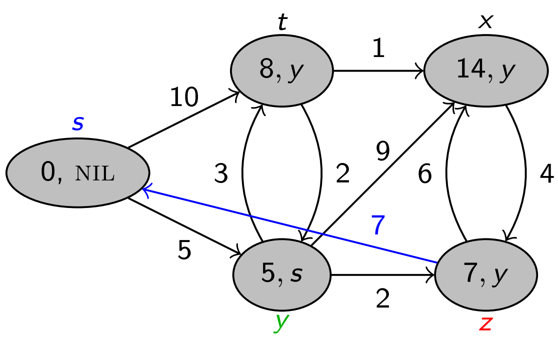
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

$\text{DIJKSTRA}(G, w, s)$



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

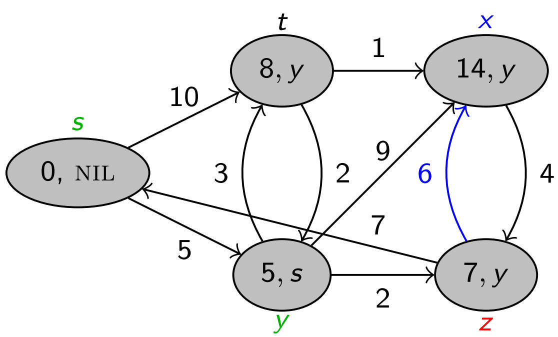
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V  // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
          if (v ∈ Q) and (d[v] > d[u]+w(u, v))
              d[v] ← d[u] + w(u, v)
              π[v] ← u
```

Dijkstra(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

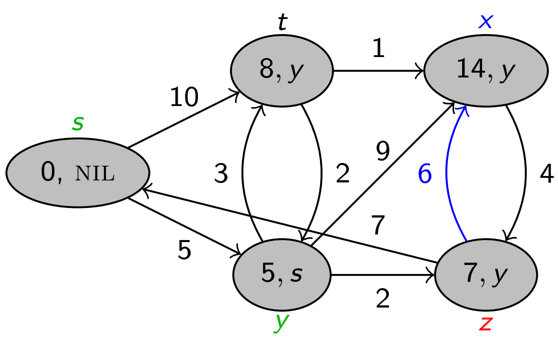
Initialization Step:

for each $v \in V$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: Min-Priority queue

Updation Step:

while $(Q \neq \emptyset)$
  $u \leftarrow$ Extract-Min(Q)
  $S \leftarrow S \cup \{u\}$
    for each vertex $v \in Adj[u]$
      if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$
        $d[v] \leftarrow d[u] + w(u, v)$
        $\pi[v] \leftarrow u$

$\textsc{Dijkstra}(G, w, s)$



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

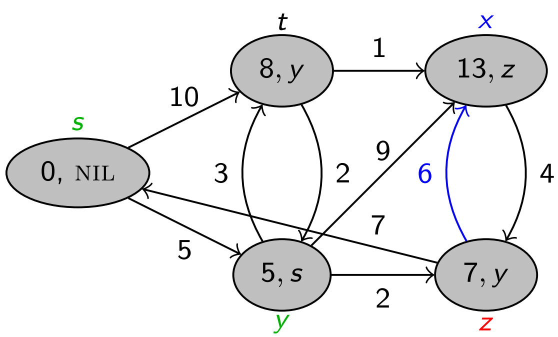
Initialization Step:

for each $v \in V$
    $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: $\textsc{Min-Priority}$ queue

Updation Step:

while $(Q \neq \emptyset)$
    $u \leftarrow \textsc{Extract-Min}(Q)$
    $S \leftarrow S \cup \{u\}$
    for each vertex $v \in Adj[u]$
        if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$
            $d[v] \leftarrow d[u] + w(u, v)$
            $\pi[v] \leftarrow u$

$\textsc{Dijkstra}(G, w, s)$



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

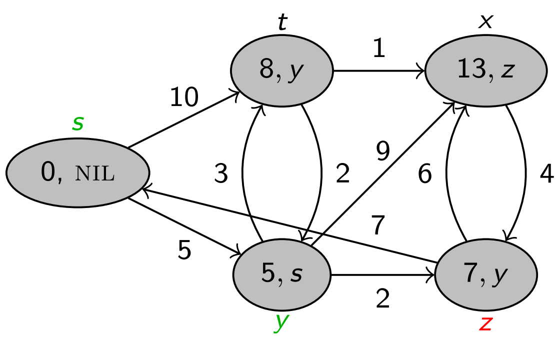
Initialization Step:

for each $v \in V$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow \textsc{nil}$
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: $\textsc{Min-Priority}$ queue

Updation Step:

while $(Q \neq \emptyset)$
  $u \leftarrow \textsc{Extract-Min}(Q)$
  $S \leftarrow S \cup \{u\}$
  for each vertex $v \in Adj[u]$
    if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

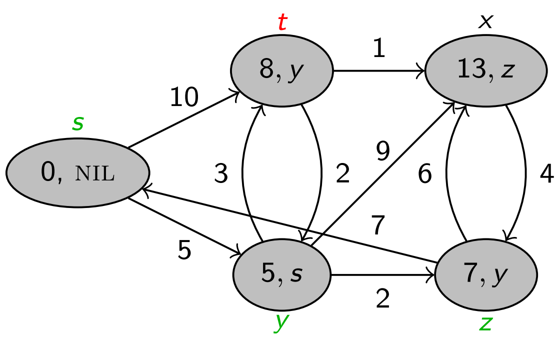
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

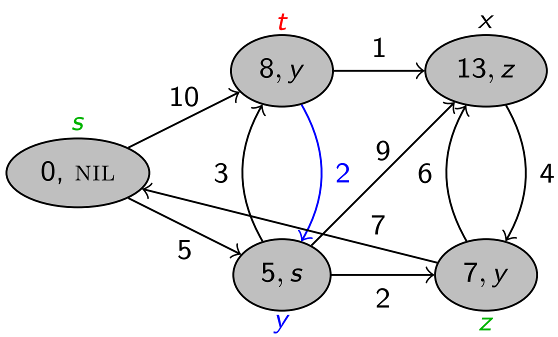
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
    for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

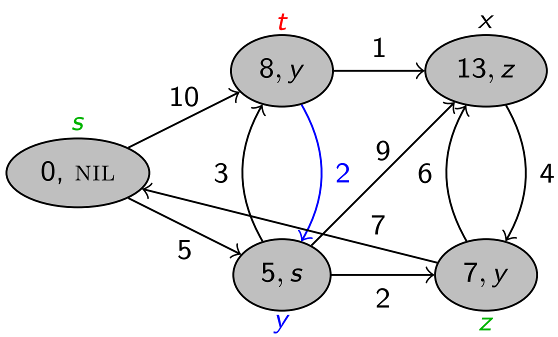
Initialization Step:

for each $v \in V$
    $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: MIN-PRIORITY queue

Updation Step:

while ($Q \neq \emptyset$)
    $u \leftarrow$ EXTRACT-MIN(Q)
    $S \leftarrow S \cup \{u\}$
    for each vertex $v \in Adj[u]$
        if ($v \in Q$) and ($d[v] > d[u] + w(u, v)$)
            $d[v] \leftarrow d[u] + w(u, v)$
            $\pi[v] \leftarrow u$

$\textsc{Dijkstra}(G, w, s)$



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

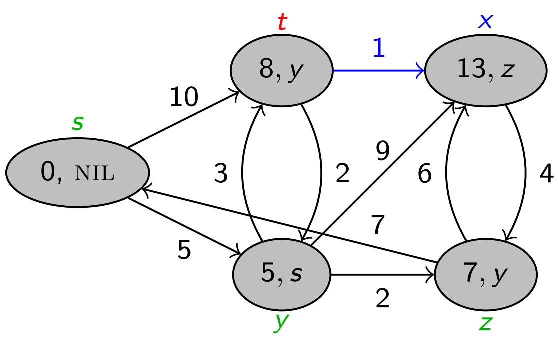
Initialization Step:

for each $v \in V$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: $\textsc{Min-Priority}$ queue

Updation Step:

while $(Q \neq \emptyset)$
  $u \leftarrow \textsc{Extract-Min}(Q)$
  $S \leftarrow S \cup \{u\}$
   for each vertex $v \in Adj[u]$
    if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$
     $d[v] \leftarrow d[u] + w(u, v)$
     $\pi[v] \leftarrow u$

$\text{DIJKSTRA}(G, w, s)$



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

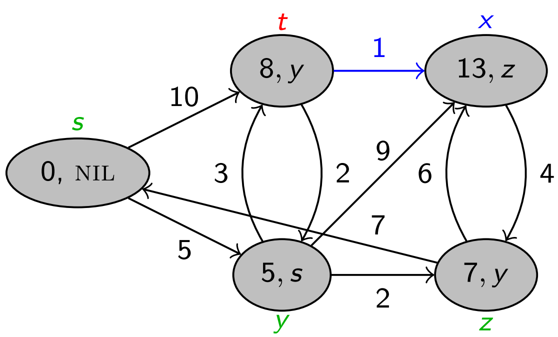
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
          if (v ∈ Q) and (d[v] > d[u]+w(u, v))
              d[v] ← d[u] + w(u, v)
              π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

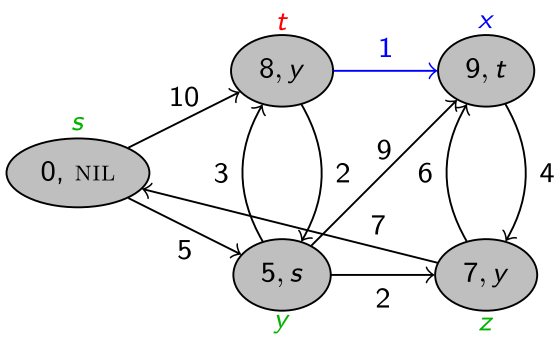
Initialization Step:

for each $v \in V$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: MIN-PRIORITY queue

Updation Step:

while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $S \leftarrow S \cup \{u\}$
  for each vertex $v \in Adj[u]$
    if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

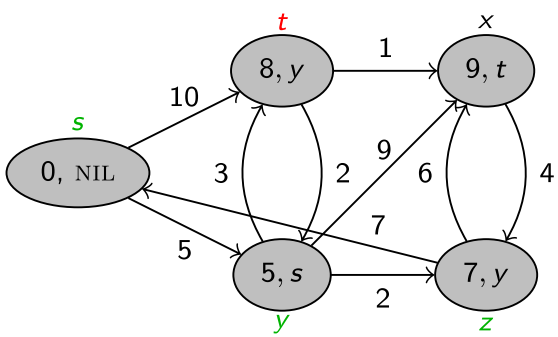
Initialization Step:

for each $v \in V$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V$ // $Q(d)$: MIN-PRIORITY queue

Updation Step:

while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $S \leftarrow S \cup \{u\}$
  for each vertex $v \in Adj[u]$
    if $(v \in Q)$ and $(d[v] > d[u] + w(u, v))$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

DIJKSTRA(G, w, s)



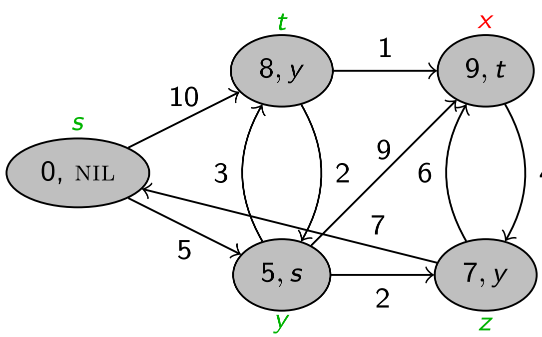
I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

```
    Initialization Step:
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue

    Updation Step:
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
          if (v ∈ Q) and (d[v] > d[u]+w(u, v))
              d[v] ← d[u] + w(u, v)
              π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

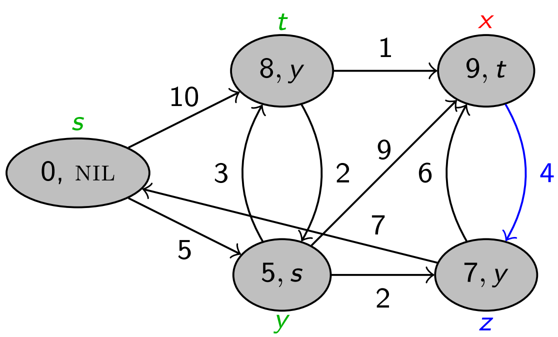
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V  // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
    for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

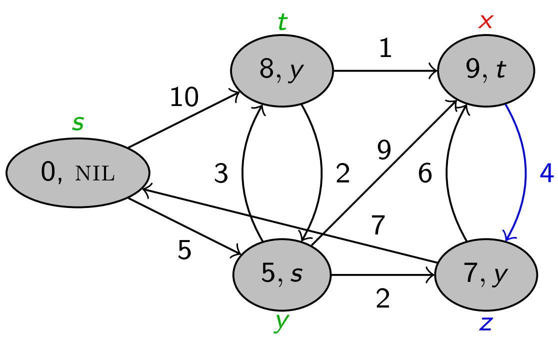
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V  // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
          d[v] ← d[u] + w(u, v)
          π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

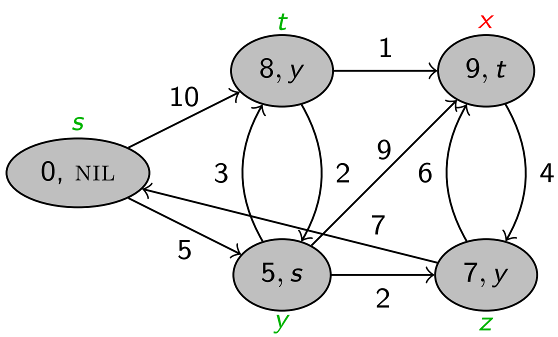
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V  // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
    for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

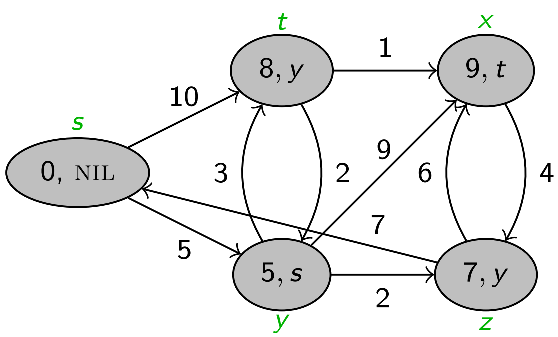
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
      for each vertex v ∈ Adj[u]
         if (v ∈ Q) and (d[v] > d[u]+w(u, v))
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

Dijkstra(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

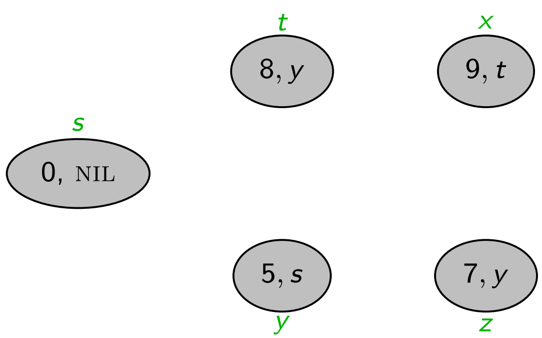
Initialization Step:

```
for each v ∈ V
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V   // Q(d): MIN-PRIORITY queue
```

Updation Step:

```
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
    for each vertex v ∈ Adj[u]
        if (v ∈ Q) and (d[v] > d[u]+w(u, v))
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

$\textsc{Dijkstra}(G, w, s)$



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

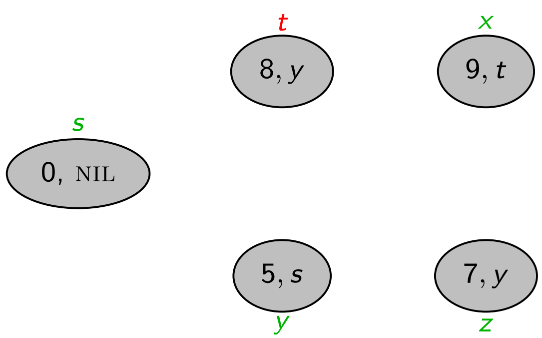
O/P: Shortest-path tree S with root vertex s.

...

Constructing the Shortest-path Tree S:

$S \leftarrow \emptyset$

DIJKSTRA(G, w, s)

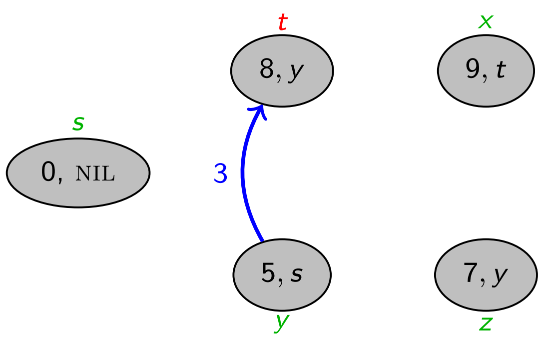


I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

```
    ...
    Constructing the Shortest-path Tree S:
    S ← ∅
    for each v ∈ V
        if (v ≠ s)
```

DIJKSTRA(G, w, s)

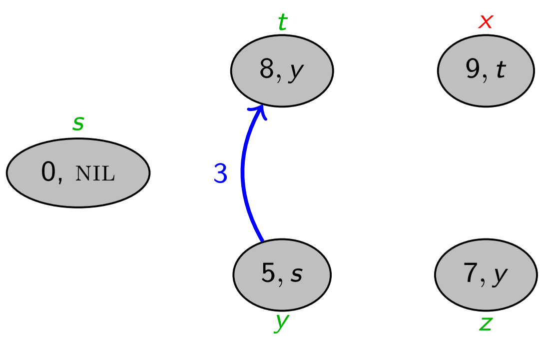


I/P: A directed graph $G = (V, E)$, with a weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

```
. . .
Constructing the Shortest-path Tree S:
S ← ∅
for each v ∈ V
   if (v ≠ s)
      S ← S ∪ {(π[v], v)}
```

DIJKSTRA(G, w, s)

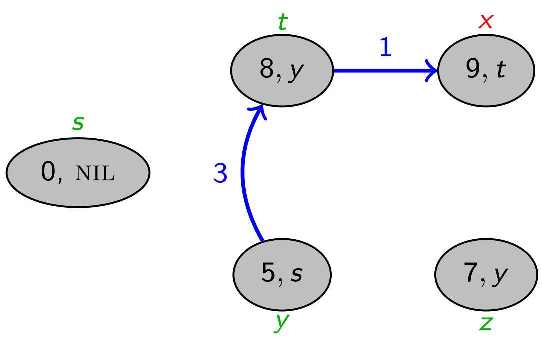


I/P: A directed graph $G = (V, E)$, with a weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

```
. . .
Constructing the Shortest-path Tree S:
S ← ∅
for each v ∈ V
   if (v ≠ s)
      S ← S ∪ {(π[v], v)}
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

...

Constructing the Shortest-path Tree S:

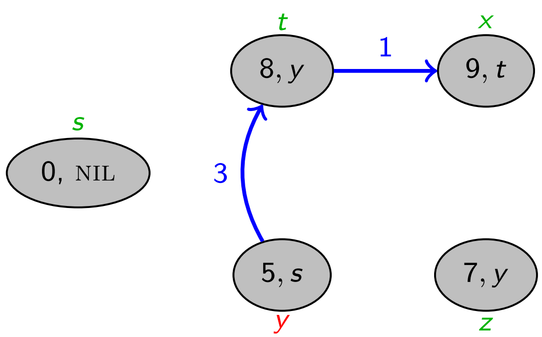
$S \leftarrow \emptyset$

for each $v \in V$

  if $(v \neq s)$

    $S \leftarrow S \cup \{(\pi[v], v)\}$

DIJKSTRA(G, w, s)

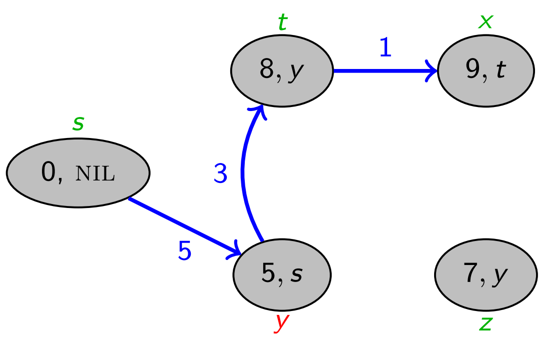


I/P: A directed graph $G = (V, E)$, with a weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

```
. . .
Constructing the Shortest-path Tree S:
S ← ∅
for each v ∈ V
   if (v ≠ s)
      S ← S ∪ {(π[v], v)}
```

DIJKSTRA(G, w, s)

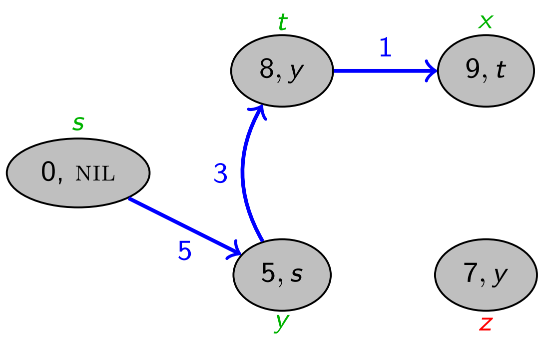


I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

```
. . .
Constructing the Shortest-path Tree S:
S ← ∅
for each v ∈ V
   if (v ≠ s)
      S ← S ∪ {(π[v], v)}
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

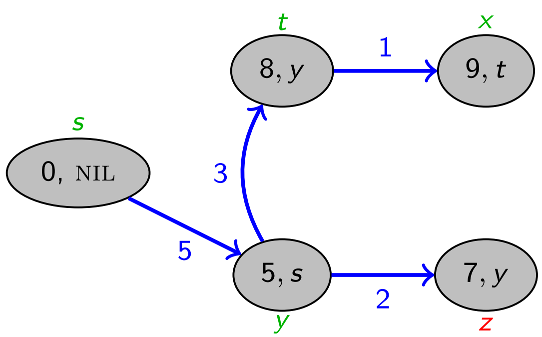
O/P: Shortest-path tree S with root vertex s.

...

Constructing the Shortest-path Tree S:

$S \leftarrow \emptyset$
for each $v \in V$
  if $(v \neq s)$
    $S \leftarrow S \cup \{(\pi[v], v)\}$

DIJKSTRA(G, w, s)

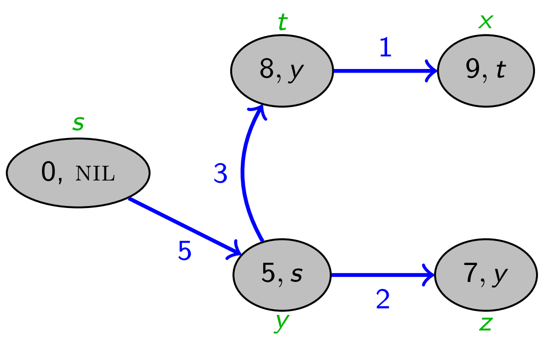


I/P: A directed graph $G = (V, E)$, with a weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

```
...
Constructing the Shortest-path Tree S:
S ← ∅
for each v ∈ V
   if (v ≠ s)
      S ← S ∪ {(π[v], v)}
```

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$ and a source vertex s.

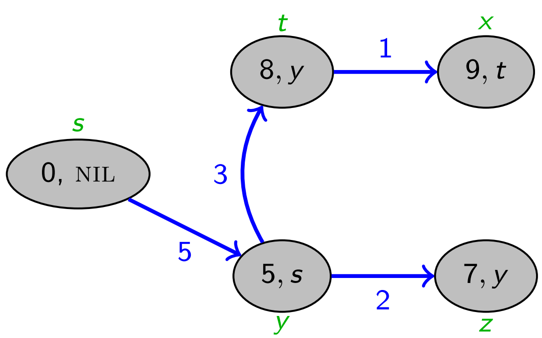
O/P: Shortest-path tree S with root vertex s.

. . .

Constructing the Shortest-path Tree S:

$S \leftarrow \emptyset$
for each $v \in V$
  if $(v \neq s)$
    $S \leftarrow S \cup \{(\pi[v], v)\}$

DIJKSTRA(G, w, s)



I/P: A directed graph $G = (V, E)$, with a weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$ and a source vertex s.

O/P: Shortest-path tree S with root vertex s.

```
. . .
Constructing the Shortest-path Tree S:
S ← ∅
for each v ∈ V
   if (v ≠ s)
      S ← S ∪ {(π[v], v)}

return S
```

Time Complexity

- MIN-PRIORITY queue operations:
  - INSERT: Implicit in line 6 - invoked once for each vertex.
    - **Total number of calls:** $|V|$.
  - EXTRACT-MIN: In line 8 - invoked once for each vertex.
    - **Total number of calls:** $|V|$.
  - DECREASE-KEY: Implicit in line 12.
    - Each vertex $v \in V$ is added to set S exactly once $\Rightarrow$ each edge in the adjacency list $Adj[v]$ is examined exactly once.
    - Recall that the total number of edges in all the adjacency lists is $|E| \Rightarrow$ there are a total of $|E|$ iterations of the for loop (lines 10-13).
    - **Total number of calls:** At most $|E|$.
- **Total Complexity:** Depends heavily on the implementation of the MIN-PRIORITY queue Q.

Array Implementation

- Take advantage of the vertices being numbered 1 to $|V|$.
- Store $d[v]$ in the v^{th} entry of an array.
- $\textsc{Insert}$ and $\textsc{Decrease-Key}$: Both takes $\mathcal{O}(1)$ time.
- $\textsc{Extract-Min}$: Takes $\mathcal{O}(|V|)$ time.
- **Total Complexity:**

$$
\begin{aligned}
& \mathcal{O}(\underbrace{|V| \cdot \mathcal{O}(1)}_{\text{lines 1-3}} + \underbrace{|V| \cdot \mathcal{O}(1)}_{\text{line 6}} + \underbrace{|V| \cdot \mathcal{O}(|V|)}_{\text{line 8}} + \underbrace{|E| \cdot \mathcal{O}(1)}_{\text{line 10-13}}) \\
= \; & \mathcal{O}(|V|^2 + |E|) \\
= \; & \mathcal{O}(|V|^2).
\end{aligned}
$$

Binary Min-Heap Implementation

- Extract-Min: $\mathcal{O}(\log |V|)$.
- Decrease-Key: $\mathcal{O}(\log |V|)$.
- Building a Min-Heap: $\mathcal{O}(|V|)$.
- **Total Complexity:**

$$\begin{aligned}
&\mathcal{O}(\underbrace{|V| \cdot \mathcal{O}(1)}_{\text{lines 1-3}} + \underbrace{\mathcal{O}(|V|)}_{\text{line 6}} + \underbrace{|V| \cdot \mathcal{O}(\log |V|)}_{\text{line 8}} + \underbrace{|E| \cdot \mathcal{O}(\log |V|)}_{\text{line 10-13}}) \\
= \; & \mathcal{O}((|V| + |E|) \log |V|),
\end{aligned}$$

  which is equal to $\mathcal{O}(|E| \log |V|)$ if all vertices are reachable from the source s.

- This is an improvement over the array implementation if $|E| = o(|V|^2 / \log |V|)$, i.e., if the graph is sufficiently sparse.
- **Note:** Can be further improved to $\mathcal{O}(|V| \log |V| + |E|)$ by using Fibonacci heap.

Correctness

Theorem

Dijkstra's algorithm, run on a weighted, directed graph $G = (V, E)$ *with non-negative weight function* w *and source* s*, terminates with* $d[u] = \delta(s, u)$ *for all vertices* $u \in V$*.*

Proof of Correctness

Loop Invariant: *At the start of each iteration of the* **while** *loop of lines 7-13,* $d[v] = \delta(s, v)$ *for each vertex* $v \in S$.

Note: It suffices to show that

- for each vertex $u \in V$, we have $d[u] = \delta(s, u)$ at the time when u is added to set S.
- We then rely on the upper-bound property ($d[u] \geq \delta(s, u)$) to show that the equality holds at all times thereafter.

Proof of Correctness

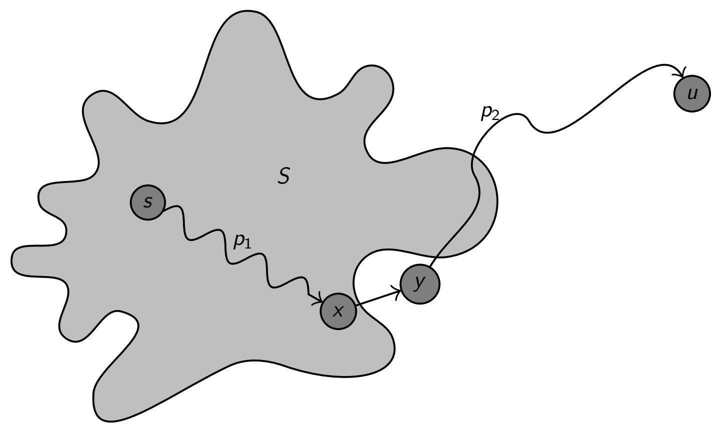
Initialization: Initially, $S = \emptyset$, and so the invariant is trivially true.

- If possible, let u be the first vertex for which $d[u] \neq \delta(s, u)$ when it is added to set S.

- **Note:** $u \neq s$ because s is the first vertex added to set S and $d[s] = \delta(s, s) = 0$ at that time $\Rightarrow$ $S \neq \emptyset$ just before u is added to S.

- Also $s \overset{p}{\rightsquigarrow} u$, otherwise $d[u] = \delta[s, u] = \infty$. $\Rightarrow\Leftarrow$!

Proof of Correctness

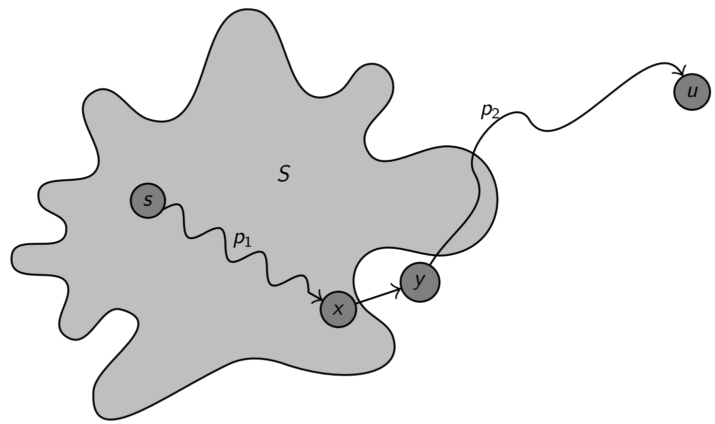


Maintenance:

- **Note:** Prior to adding u to S, path p connects a vertex in S, namely s, to a vertex in $V \setminus S$, namely u.
- Let y be the first vertex along p such that $y \in V \setminus S$, and let $x \in S$ be y's predecessor. That is p can be decomposed as

$$s \overset{p_1}{\rightsquigarrow} x \to y \overset{p_2}{\rightsquigarrow} u \quad (p_1, p_2 \text{ can be empty}).$$

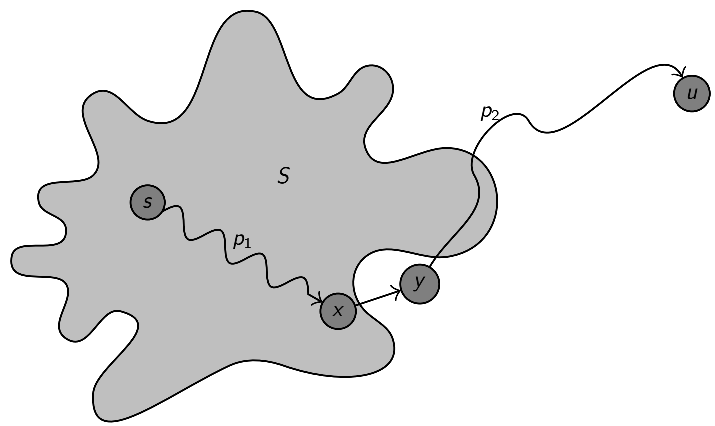
Proof of Correctness



Maintenance:

- **Claim:** $d[y] = \delta(s, y)$ when u is added to S.
- **Observe:** $x \in S$ and by our assumption $d[x] = \delta(s, x)$ when x was added to S.
- **Note:** If $d[x] = \delta(s, x)$ at some point prior to considering the edge (x, y), then $d[x] = \delta(s, x)$ thereafter ($\because\ d[x] \geq \delta(s, x)$).

Proof of Correctness



Maintenance:

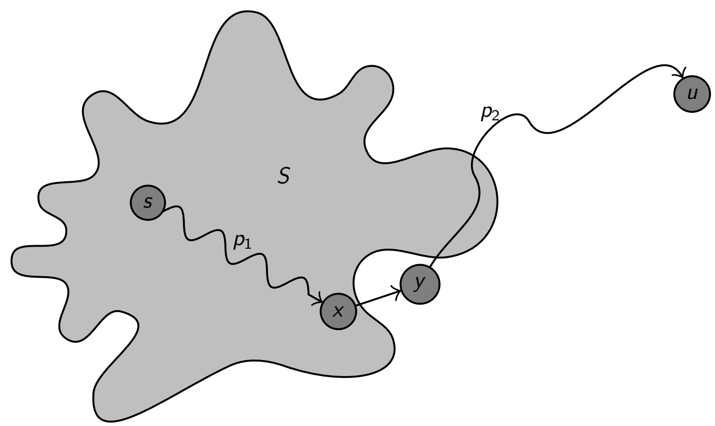
- In particular, by convergence property we have

$$\delta(s,y) \le d[y] \le d[x] + w(x,y) = \delta(s,x) + w(x,y) = \delta(s,y).$$

  which implies that $d[y] = \delta(s,y)$, thereby proving the claim.

- **Convergence Property:** If $s \rightsquigarrow u \to v$ is a shortest path in G, and if $d[u] = \delta(s,u)$ at any time prior to relaxing edge (u,v), then $d[v] = \delta(s,v)$ at all times afterward.

Proof of Correctness

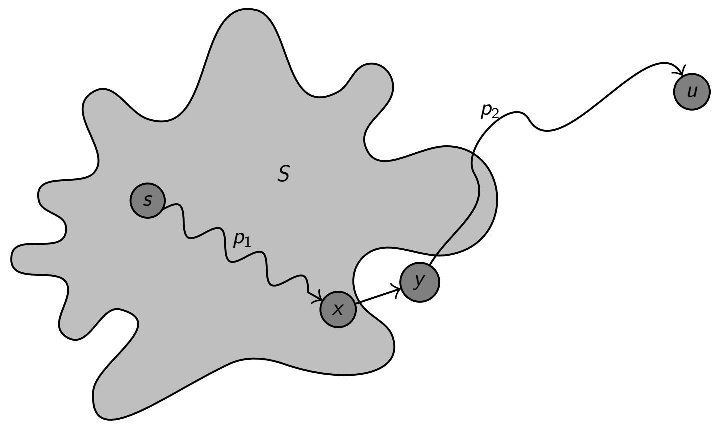


Maintenance:

- Because y occurs before u on a shortest path from s to u and all edge weights are non-negative (especially on p_2), we have $\delta(s, y) \leq \delta(s, u)$.
- Therefore

$$d[y] = \delta(s, y) \leq \delta(s, u) \leq d[u]. \tag{1}$$

Proof of Correctness



Maintenance:

- **Note:** Both vertices u and y were in $V \setminus S$ when u was chosen in line 8, which implies $d[u] \leq d[y]$.
- Then form (1), we have

$$d[y] = \delta(s, y) = \delta(s, u) = d[u],$$

which is a contradiction!!

Proof of Correctness

Termination:

- The **while** terminates, when $Q = \emptyset$ which, along with our earlier invariant that $Q = V \setminus S$, implies that $S = V$.

- Thus, $d[u] = \delta(s, u)$ for all vertices $u \in V$.

A Corollary

Corollary

If we run Dijkstra's algorithm on a weighted, directed graph $G = (V, E)$ with non-negative weight function w and source s, then at termination, the predecessor subgraph G_π is a shortest-paths tree rooted at s.

Proof: Follows immediately from the Theorem.

Books and Other Materials Consulted

1. *Introduction to Algorithms* by Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein.

Thank You for your kind attention!

Questions!!