BFS is a single-source shortest-path algorithm that works on unweighted graphs, that is, graphs in which each edge has unit weight.

Shortest Path Algorithms

?? Minimize weights ??

Time, cost, penalties, loss, etc.

Introduction

- Given a weighted, directed graph $\boldsymbol{G = (V, E)}$, with weight function $\boldsymbol{w : E} \longrightarrow \mathbb{R}$.
- $\boldsymbol{w(p)}$, the weight of path $\boldsymbol{p}$ from $\boldsymbol{v_0}$ to $\boldsymbol{v_k}$ is given by

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

- Then shortest-path weight $\boldsymbol{\delta(u,v)}$ is defined as

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\leadsto} v\} & \text{if there is a path from } u \text{ to } v \text{ ,} \\ \infty & \text{otherwise .} \end{cases}$$

- Shortest path from vertex $\boldsymbol{u}$ to vertex $\boldsymbol{v}$ is then defined as any path $\boldsymbol{p}$ with weight $\boldsymbol{w(p) = \delta(u,v)}$.

Contd…

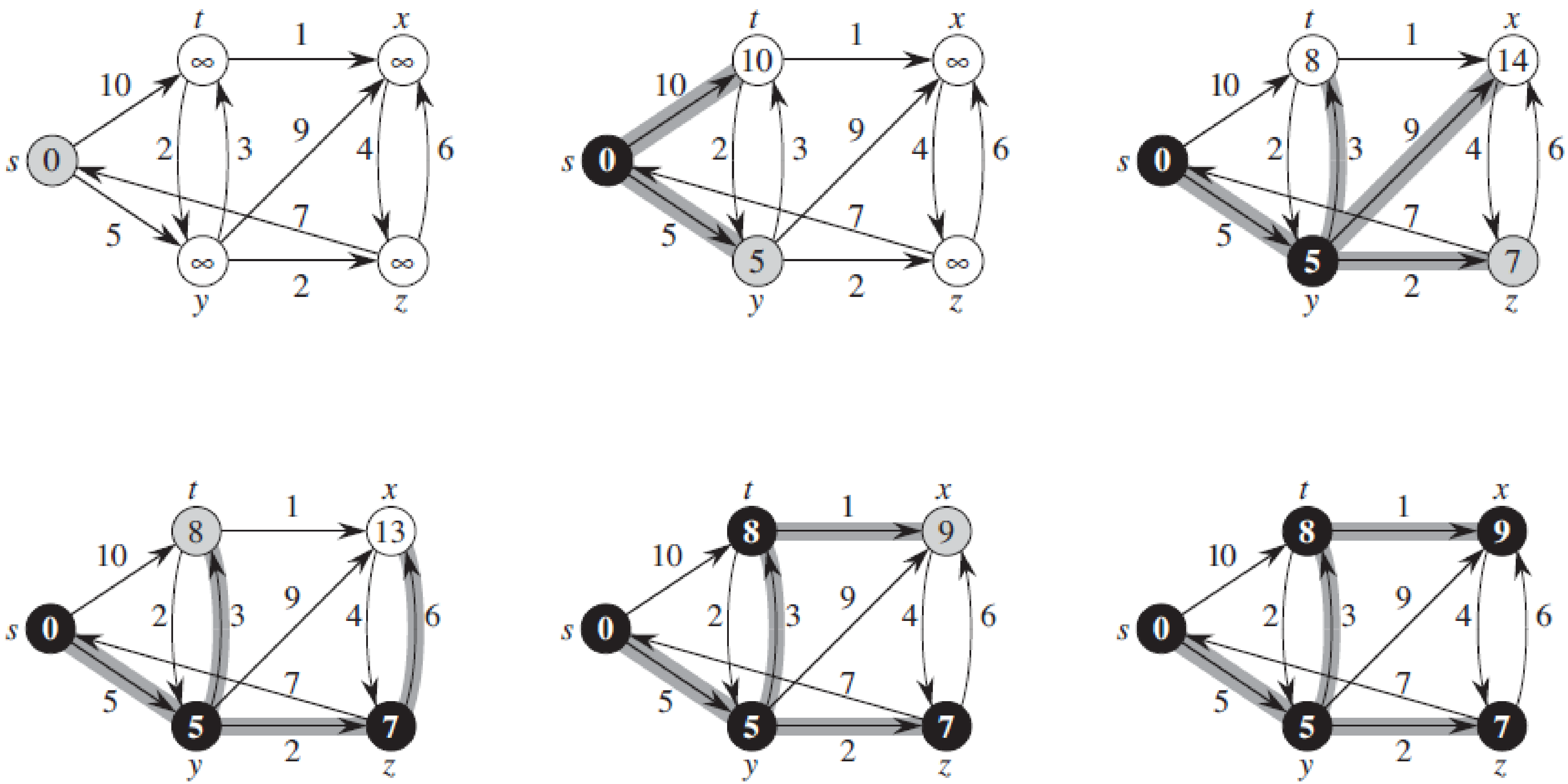
- Single-source shortest-paths problem, i.e. given a graph find a shortest path from a given source vertex to each other vertex.
  - Dijkstra’s algorithm.
- Variants:
  - Single-destination shortest-paths problem
  - Single-pair shortest-path problem
  - All-pairs shortest-paths problem, i.e. find a shortest path from u to v for every pair of vertices u and v.
    - Floyd-Warshall algorithm.

Dijkstra’s Algorithm

- Solves single-source shortest-paths problem on a weighted, directed graph in which all edge weights are nonnegative.

Example

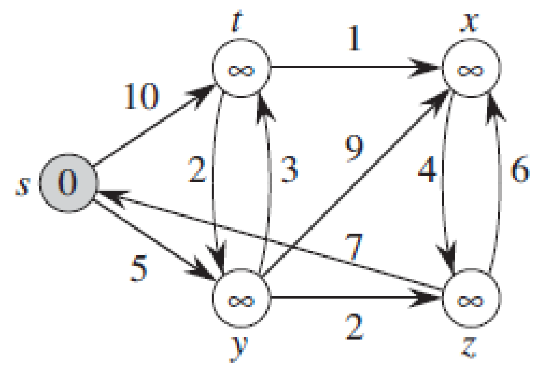
```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```



Implementation

s	→ t, 10	→ y, 5	
t	→ x, 1	→ y, 2	
x	→ z, 4		
y	→ t, 3	→ x, 9	→ z, 2
z	→ s, 7	→ x, 6	

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```




```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

Example - Execution

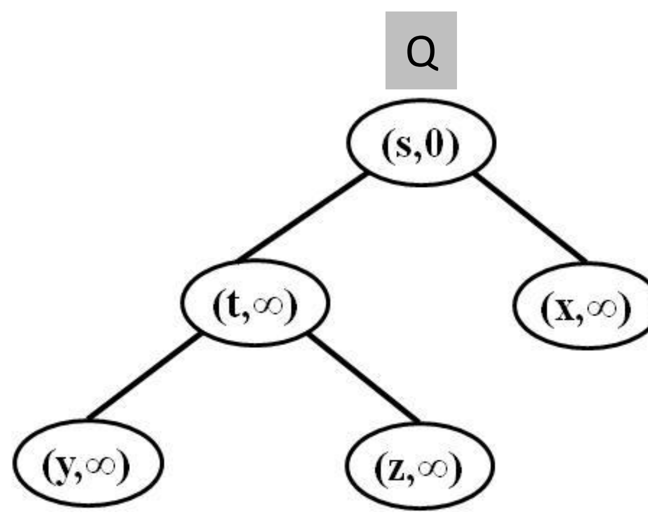
S = {}

Vertex	π	d
s	NIL	0
t	NIL	∞
x	NIL	∞
y	NIL	∞
z	NIL	∞

s → t 10 → y 5
t → x 1 → y 2
x → z 4
y → t 3 → x 9 → z 2
z → s 7 → x 6

DIJKSTRA(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```



Example - Execution

s → (t, 10) → (y, 5)
t → (x, 1) → (y, 2)
x → (z, 4)
y → (t, 3) → (x, 9) → (z, 2)
z → (s, 7) → (x, 6)

S = {s}

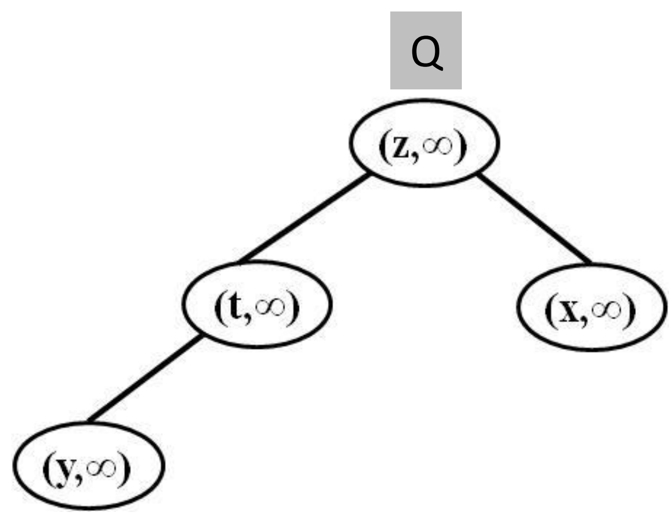
Vertex	π	d
s	NIL	0
t	NIL	∞
x	NIL	∞
y	NIL	∞
z	NIL	∞

RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

DIJKSTRA(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```



Example - Execution

S = {s}

Vertex	π	d
s	NIL	0
t	s	10
x	NIL	∞
y	NIL	∞
z	NIL	∞

s → t 10 → y 5
t → x 1 → y 2
x → z 4
y → t 3 → x 9 → z 2
z → s 7 → x 6

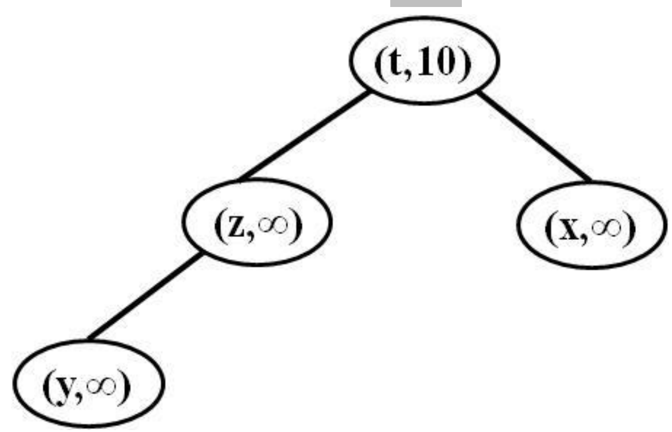
RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

DIJKSTRA(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Q



Example - Execution

S = {s}

Vertex	π	d
s	NIL	0
t	s	10
x	NIL	∞
y	s	5
z	NIL	∞

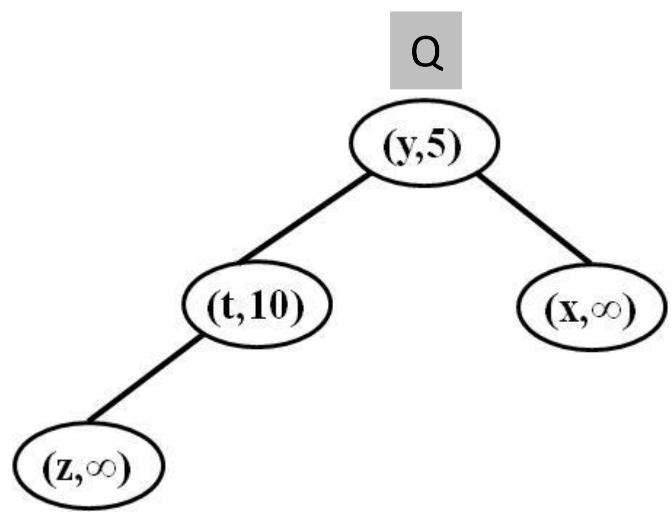
s	→ t, 10	→ y, 5	
t	→ x, 1	→ y, 2	
x	→ z, 4		
y	→ t, 3	→ x, 9	→ z, 2
z	→ s, 7	→ x, 6	

RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

DIJKSTRA(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```



Example - Execution

$S = \{s, y\}$

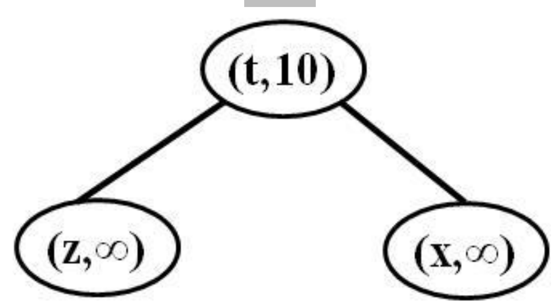
Vertex	π	d
s	NIL	0
t	s	10
x	NIL	∞
y	s	5
z	NIL	∞

s → [t | 10] → [y | 5]
t → [x | 1] → [y | 2]
x → [z | 4]
y → [t | 3] → [x | 9] → [z | 2]
z → [s | 7] → [x | 6]

RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

Q



DIJKSTRA(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

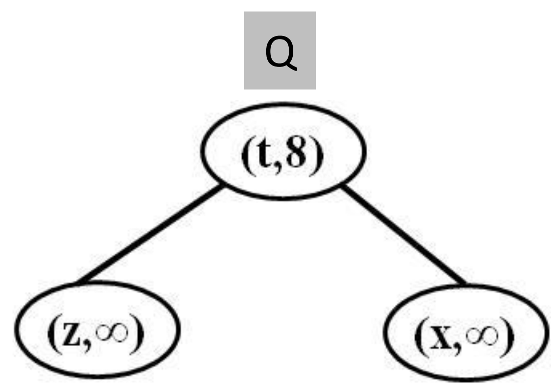
Example - Execution

S = {s, y}

Vertex	π	d
s	NIL	0
t	y	8
x	NIL	∞
y	s	5
z	NIL	∞

s → t 10 → y 5
t → x 1 → y 2
x → z 4
y → t 3 → x 9 → z 2
z → s 7 → x 6

```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```




```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Example - Execution

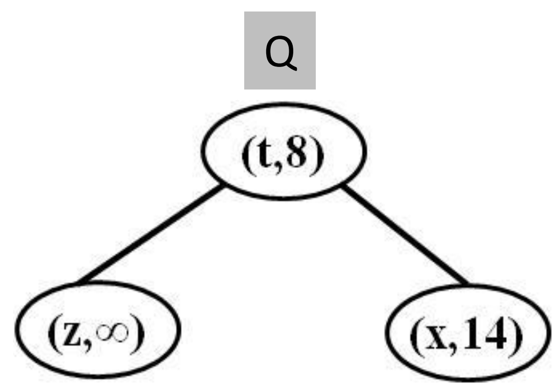
S = {s, y}

Vertex	π	d
s	NIL	0
t	y	8
x	y	14
y	s	5
z	NIL	∞

s → t 10 → y 5
t → x 1 → y 2
x → z 4
y → t 3 → x 9 → z 2
z → s 7 → x 6

RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```



DIJKSTRA(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Example - Execution

s → (t, 10) → (y, 5)
t → (x, 1) → (y, 2)
x → (z, 4)
y → (t, 3) → (x, 9) → (z, 2)
z → (s, 7) → (x, 6)

S = {s, y}

Vertex	π	d
s	NIL	0
t	y	8
x	y	14
y	s	5
z	y	7

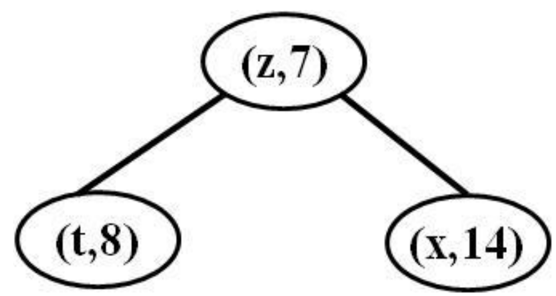
RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

DIJKSTRA(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Q



Example - Execution

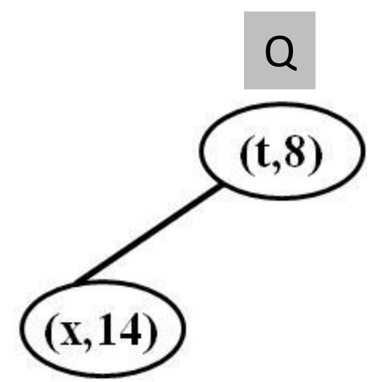
S = {s, y, z}

Vertex	π	d
s	NIL	0
t	y	8
x	y	14
y	s	5
z	y	7

s	→	t 10	→	y 5
t	→	x 1	→	y 2
x	→	z 4		
y	→	t 3	→	x 9 → z 2
z	→	s 7	→	x 6

RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```



DIJKSTRA(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Example - Execution

S = {s, y, z}

Vertex	π	d
s	NIL	0
t	y	8
x	z	13
y	s	5
z	y	7

s → | t | 10 | → | y | 5 |

t → | x | 1 | → | y | 2 |

x → | z | 4 |

y → | t | 3 | → | x | 9 | → | z | 2 |

z → | s | 7 | → | x | 6 |

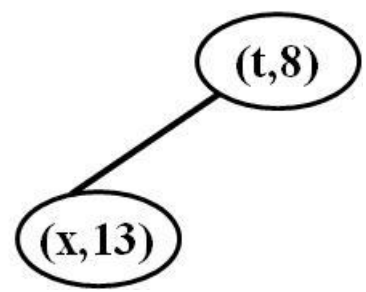
RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

DIJKSTRA(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Q



Example - Execution

$s \rightarrow \boxed{t \mid 10} \rightarrow \boxed{y \mid 5}$

$t \rightarrow \boxed{x \mid 1} \rightarrow \boxed{y \mid 2}$

$x \rightarrow \boxed{z \mid 4}$

$y \rightarrow \boxed{t \mid 3} \rightarrow \boxed{x \mid 9} \rightarrow \boxed{z \mid 2}$

$z \rightarrow \boxed{s \mid 7} \rightarrow \boxed{x \mid 6}$

S = {s, y, z, t}

Vertex	**π**	**d**
s	NIL	0
t	y	8
x	z	13
y	s	5
z	y	7

RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

Q

(x,13)

DIJKSTRA(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Example - Execution

S = {s, y, z, t}

Vertex	π	d
s	NIL	0
t	y	8
x	t	9
y	s	5
z	y	7

s → t 10 → y 5
t → x 1 → y 2
x → z 4
y → t 3 → x 9 → z 2
z → s 7 → x 6

```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Q

(x,9)

Example - Execution

s → t 10 → y 5
t → x 1 → y 2
x → z 4
y → t 3 → x 9 → z 2
z → s 7 → x 6

S = {s, y, z, t, x}

Vertex	π	d
s	NIL	0
t	y	8
x	t	9
y	s	5
z	y	7

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```