Graph Algorithms (Chapter 24)
Single-Source Shortest Paths(SSSP)

University of Stavanger

Goals

The shortest weighted path problem and its variants

The optimal substructure of the shortest path

Negative weights issues

Relaxation Technique

Bellman-Ford Algorithm and its time complexity

What if we have a weighted DAG

Dijkstra Algorithm

Graph Algorithms (Chapter 24) SSSP

Shortest path problem

Given a directed graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}$ mapping edges to weights. A path $p = \langle v_0, v_1, \ldots v_k \rangle$ has weight $w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$ (sum of weights of constituent edges)

We define the ***shortest-path weight*** $\delta(u, v)$ from u to v by

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

Shortest path problem variants

Our focus

Find the a shortest path from a source $s \in V$ to each $v \in V$.

Shortest path problem variants

Variant 1

Single-destination shortest-paths problem: Find a shortest path to a given ***destination*** vertex t from each vertex v. By reversing the direction of each edge in the graph, we can reduce this problem to a single-source problem.

University of Stavanger

Shortest path problem variants

Variant 2

Single-pair shortest-path problem: Find a shortest path from u to v for given vertices u and v. If we solve the single-source problem with source vertex u, we solve this problem also. Moreover, all known algorithms for this problem have the same worst-case asymptotic running time as the best single-source algorithms.

University of Stavanger

Shortest path problem variants

Variant 3

Find a shortest path from u to v for every pair of vertices u and v. Can be solved by running single source algorithm once from each vertex.

Can be solved faster (chap 25) but not syllabus

Shortest path optimal substructure

Lemma 24.1 (Subpaths of shortest paths are shortest paths)
Given a weighted, directed graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}$, let $p = \langle v_0, v_1, \ldots, v_k \rangle$ be a shortest path from vertex v_0 to vertex v_k and, for any i and j such that $0 \le i \le j \le k$, let $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ be the subpath of p from vertex v_i to vertex v_j. Then, p_{ij} is a shortest path from v_i to v_j.

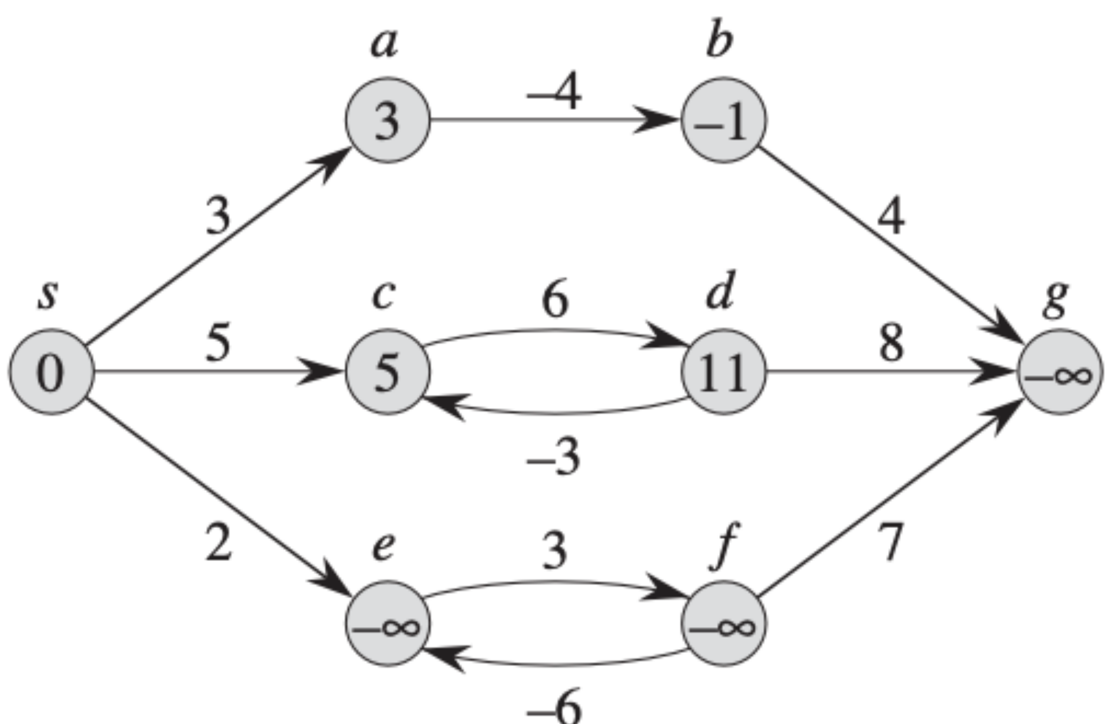
Proof If we decompose path p into $v_0 \overset{p_{0i}}{\leadsto} v_i \overset{p_{ij}}{\leadsto} v_j \overset{p_{jk}}{\leadsto} v_k$, then we have that $w(p) = w(p_{0i}) + w(p_{ij}) + w(p_{jk})$. Now, assume that there is a path p'_{ij} from v_i to v_j with weight $w(p'_{ij}) < w(p_{ij})$. Then, $v_0 \overset{p_{0i}}{\leadsto} v_i \overset{p'_{ij}}{\leadsto} v_j \overset{p_{jk}}{\leadsto} v_k$ is a path from v_0 to v_k whose weight $w(p_{0i}) + w(p'_{ij}) + w(p_{jk})$ is less than $w(p)$, which contradicts the assumption that p is a shortest path from v_0 to v_k. ■

University of Stavanger

Negative-weights and cycles

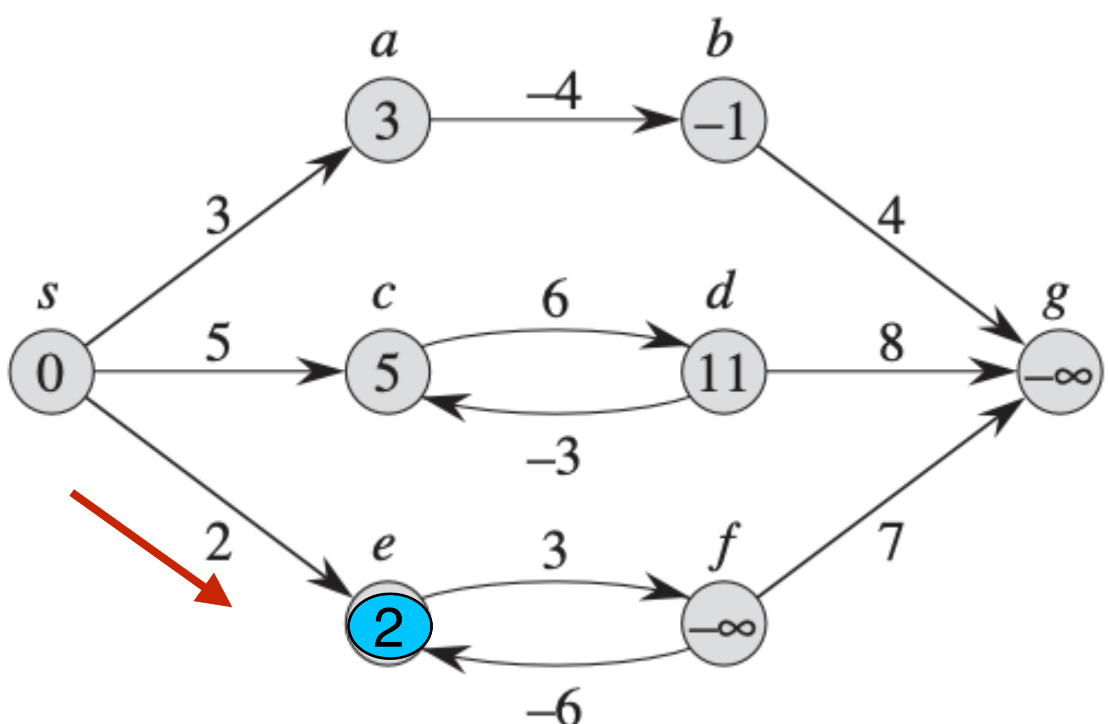
With negative weight CYCLE reachable from source s no path can be considered as the shortest path !!!

Try the shortest path $\delta(s, f)$?



Negative-weights and cycles

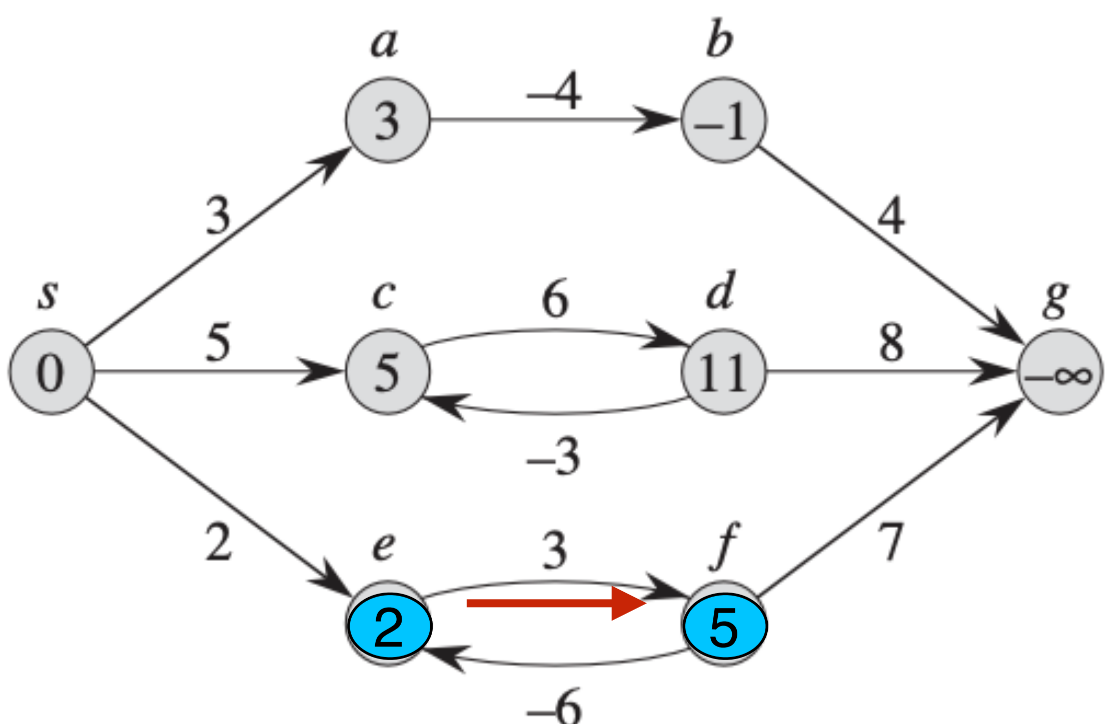
With negative weight CYCLE reachable from source s no path can be considered as the shortest path !!!



University of Stavanger

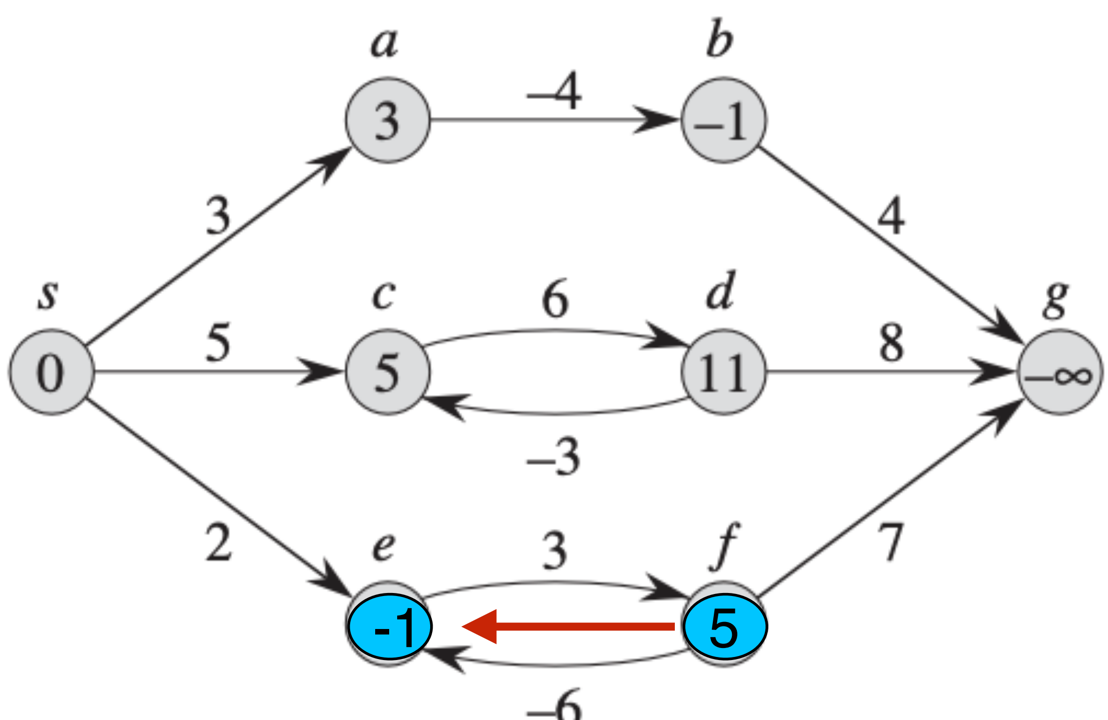
Negative-weights and cycles

With negative weight CYCLE reachable from source s no path can be considered as the shortest path !!!



Negative-weights and cycles

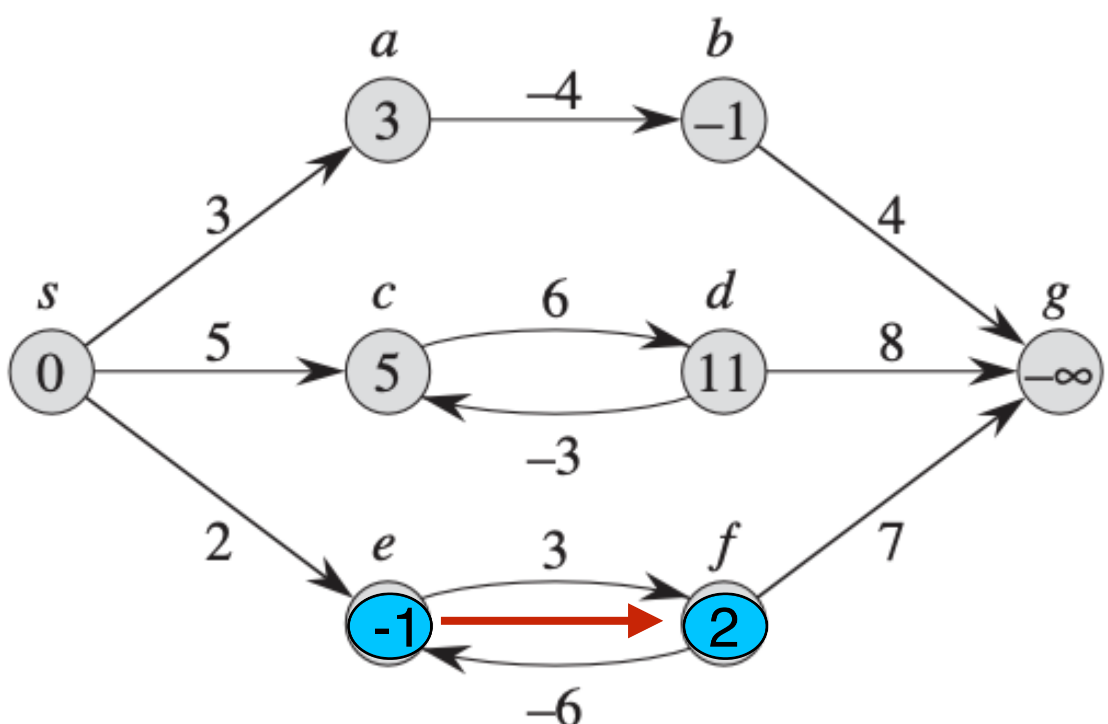
With negative weight CYCLE reachable from source s no path can be considered as the shortest path !!!



University of Stavanger

Negative-weights and cycles

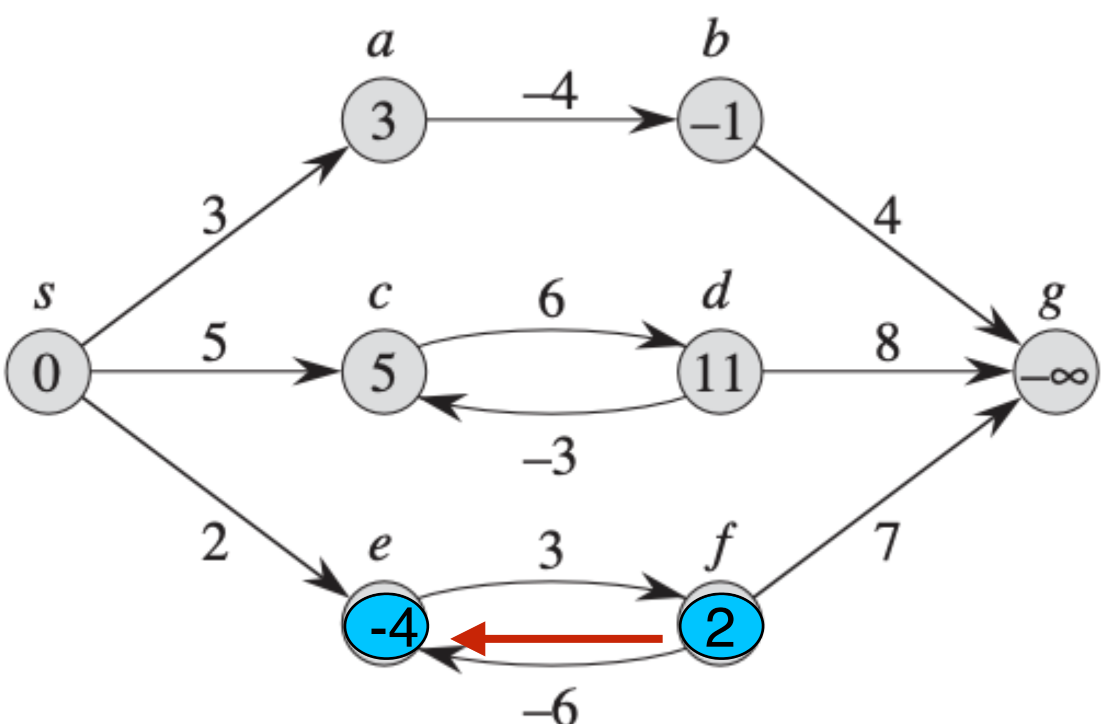
With negative weight CYCLE reachable from source s no path can be considered as the shortest path !!!



University of Stavanger

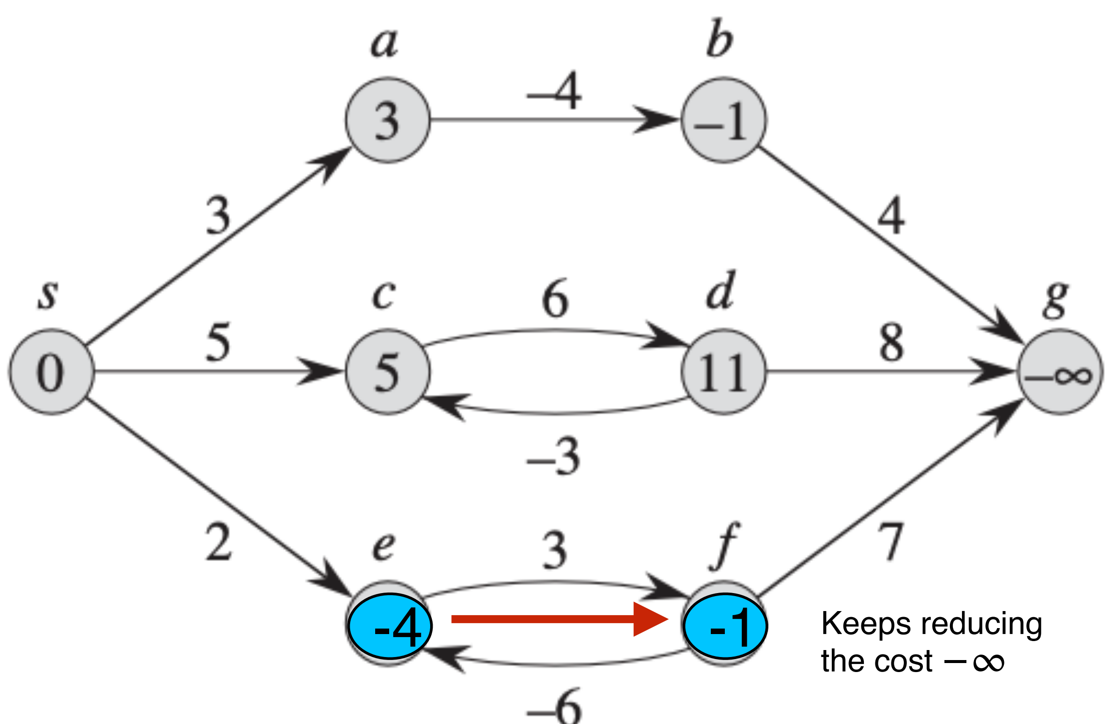
Negative-weights and cycles

With negative weight CYCLE reachable from source s no path can be considered as the shortest path !!!



Negative-weights and cycles

With negative weight CYCLE reachable from source s no path can be considered as the shortest path !!!



Shortest path representation

Just like we did with BFS

We maintain for every vertex $v \in V$ a predecessor $v.\pi$ (parent) or NIL

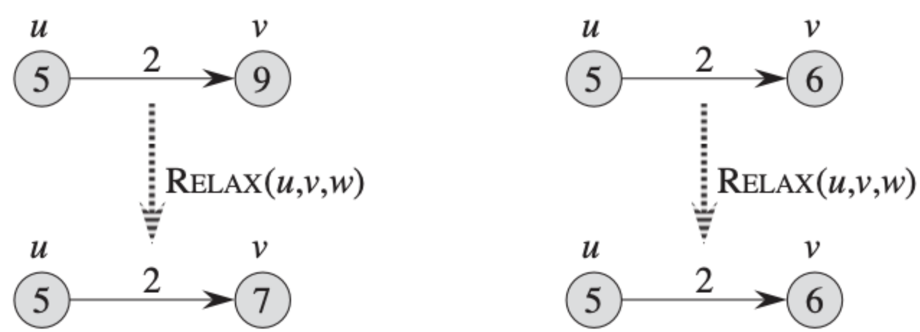
Relaxation Technique

It answers: Can I make the distance $v.d$ shorter by walking (u, v) or not.

If yes, update $v.d$ and $v.\pi$

RELAX(u, v, w)

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



(a) (b)

Relaxation Properties

Triangle inequality (Lemma 24.10)
For any edge $(u, v) \in E$, we have $\delta(s, v) \leq \delta(s, u) + w(u, v)$.

Upper-bound property (Lemma 24.11)
We always have $v.d \geq \delta(s, v)$ for all vertices $v \in V$, and once $v.d$ achieves the value $\delta(s, v)$, it never changes.

No-path property (Corollary 24.12)
If there is no path from s to v, then we always have $v.d = \delta(s, v) = \infty$.

Convergence property (Lemma 24.14)
If $s \leadsto u \to v$ is a shortest path in G for some $u, v \in V$, and if $u.d = \delta(s, u)$ at any time prior to relaxing edge (u, v), then $v.d = \delta(s, v)$ at all times afterward.

Path-relaxation property (Lemma 24.15)
If $p = \langle v_0, v_1, \ldots, v_k \rangle$ is a shortest path from $s = v_0$ to v_k, and we relax the edges of p in the order $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, then $v_k.d = \delta(s, v_k)$. This property holds regardless of any other relaxation steps that occur, even if they are intermixed with relaxations of the edges of p.

Predecessor-subgraph property (Lemma 24.17)
Once $v.d = \delta(s, v)$ for all $v \in V$, the predecessor subgraph is a shortest-paths tree rooted at s.

Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

Solves for positive and negative weights and returns no solution if negative cycle. If no negative cycle the result is the shortest path and their weights.

```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| - 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

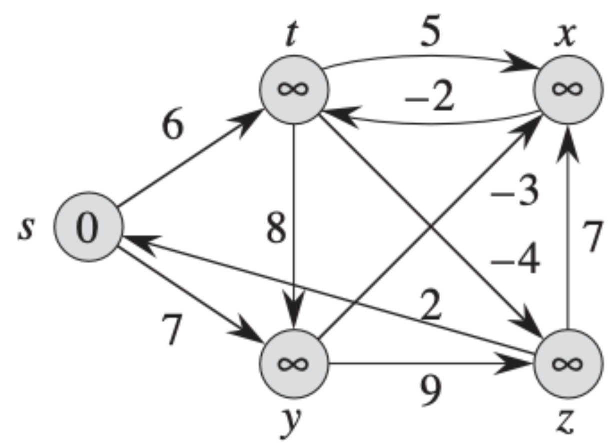
Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

Example

Assume we pass the edges in the following order:

$$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$$

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




```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

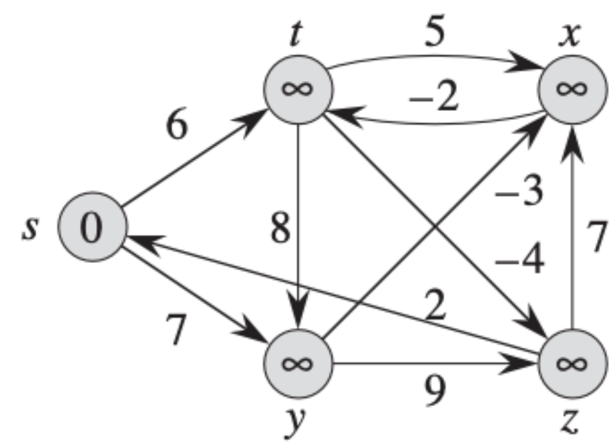
Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

Example

Assume we pass the edges in the following order:

$$(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)$$

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



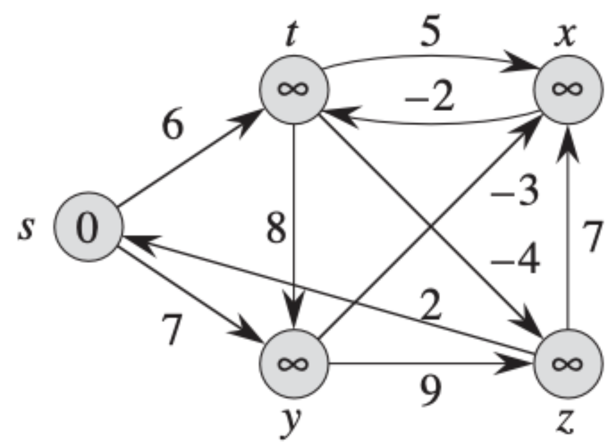

```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i ∈ {1,2,3,4}
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Example

Assume we pass the edges in the following order:

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$

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




```
BELLMAN-FORD(G, w, s)
  INITIALIZE-SINGLE-SOURCE(G, s)
  for i = 1 to |G.V| − 1        i = 1
      for each edge (u, v) ∈ G.E
          RELAX(u, v, w)
  for each edge (u, v) ∈ G.E
      if v.d > u.d + w(u, v)
          return FALSE
  return TRUE
```

Example

Assume we pass the edges in the following order:

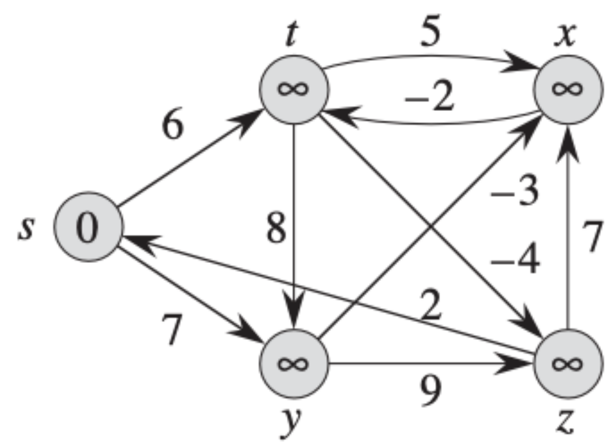
$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$

RELAX(u, v, w)

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

BELLMAN-FORD(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```



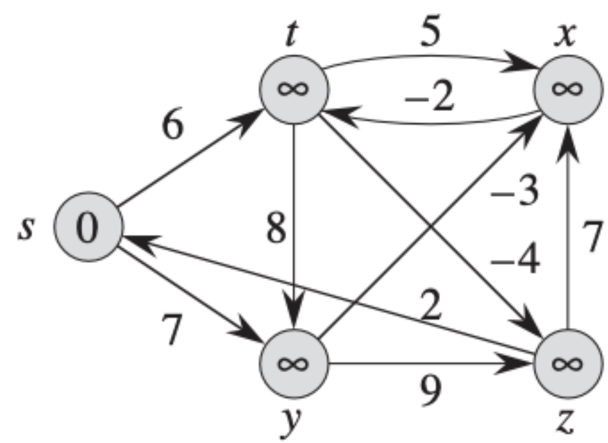
University of Stavanger

Example

Assume we pass the edges in the following order:

$(t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)$

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




```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1          i = 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

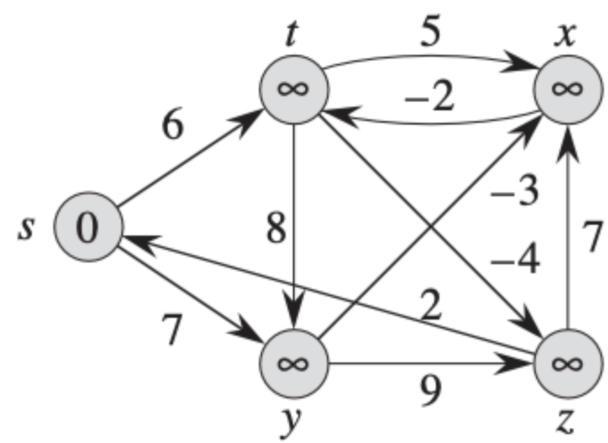
Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

Example

Assume we pass the edges in the following order:

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$

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




```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

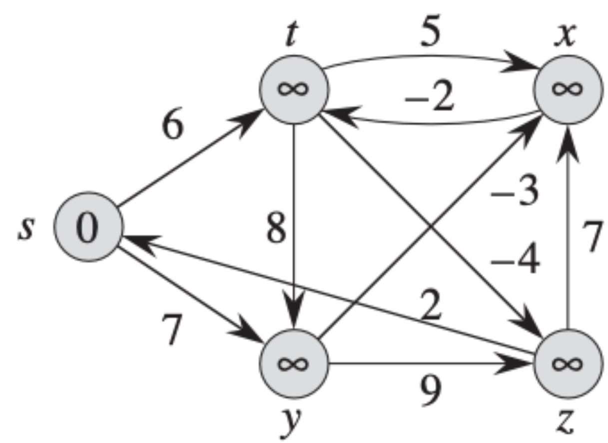
Example

Assume we pass the edges in the following order:

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$

RELAX(u, v, w)

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



BELLMAN-FORD(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1          i = 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

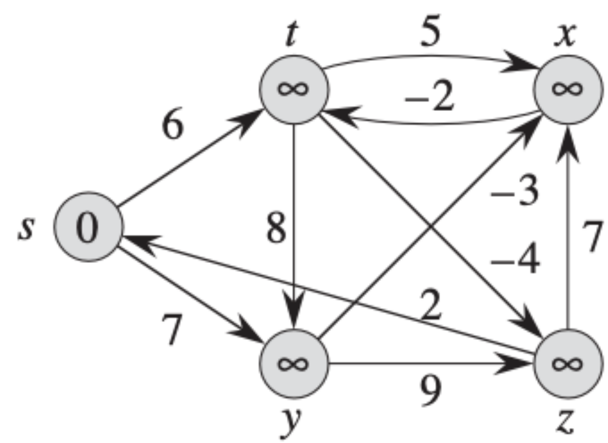
Graph Algorithms (Chapter 24.1) SSSP (Bellman-Ford algorithm)

Example

Assume we pass the edges in the following order:

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$

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




```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

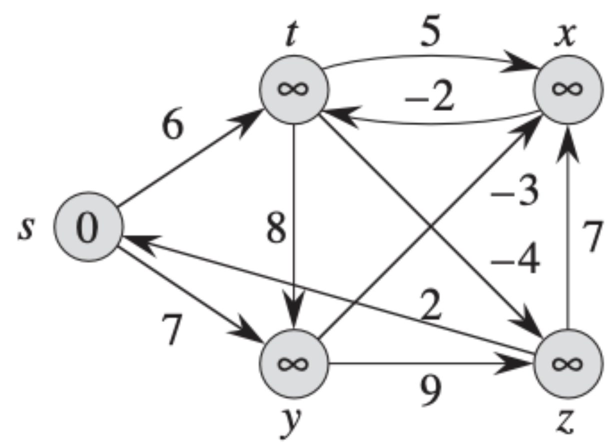
Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

Example

Assume we pass the edges in the following order:

$(t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)$

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




```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

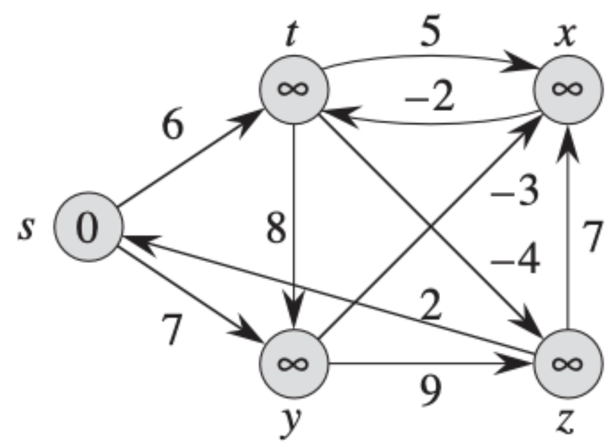
University of Stavanger

Example

Assume we pass the edges in the following order:

$(t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)$

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




```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

University of Stavanger

Example

Assume we pass the edges in the following order:

$(t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)$

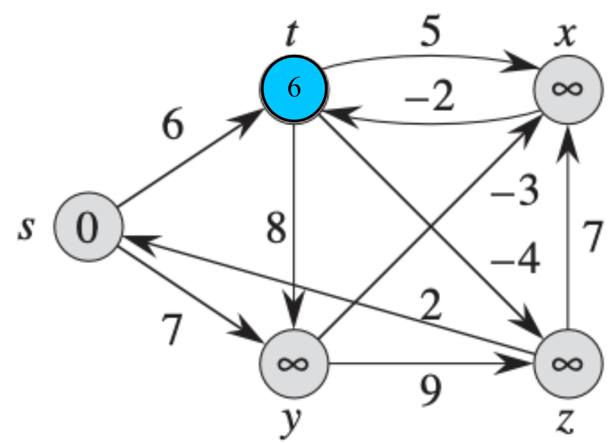
x

RELAX(u, v, w)

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

BELLMAN-FORD(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| - 1        i = 1
    for each edge (u,v) ∈ G.E
        RELAX(u, v, w)
for each edge (u,v) ∈ G.E
    if v.d > u.d + w(u,v)
        return FALSE
return TRUE
```



University of Stavanger

Example

Assume we pass the edges in the following order:

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$

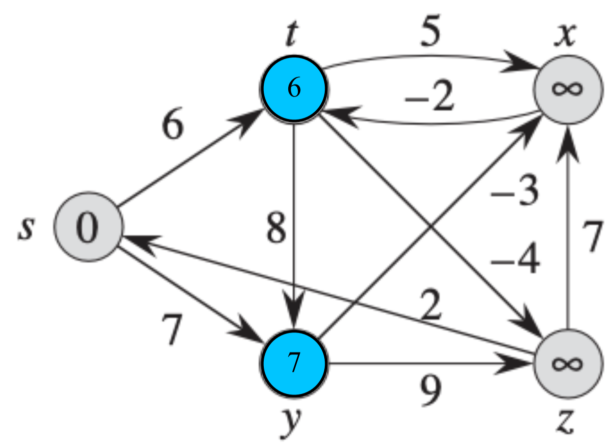
x x

RELAX(u, v, w)

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

BELLMAN-FORD(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| - 1        i = 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```



Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

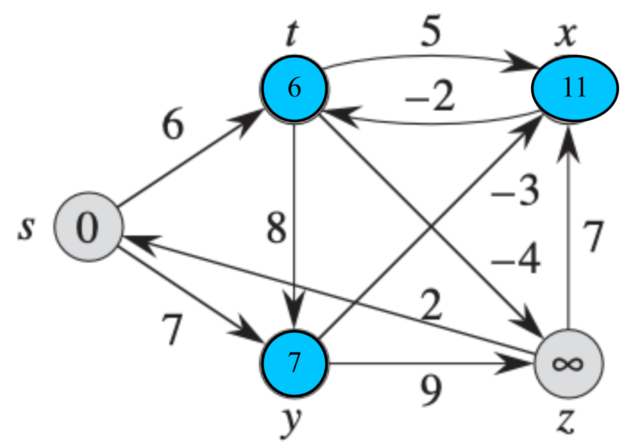
Example

Assume we pass the edges in the following order:

$(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)$

x

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




```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 2
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

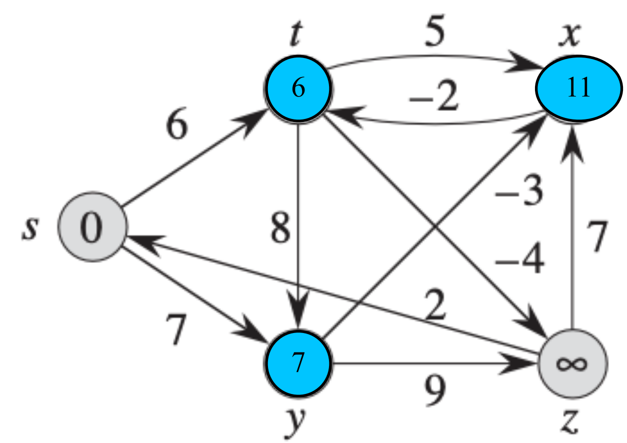
Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

Example

Assume we pass the edges in the following order:

$(t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)$

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




```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1          i = 2
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if v.d > u.d + w(u, v)
7          return FALSE
8  return TRUE
```

Graph Algorithms (Chapter 24.1) SSSP (Bellman-Ford algorithm)

University of Stavanger

Example

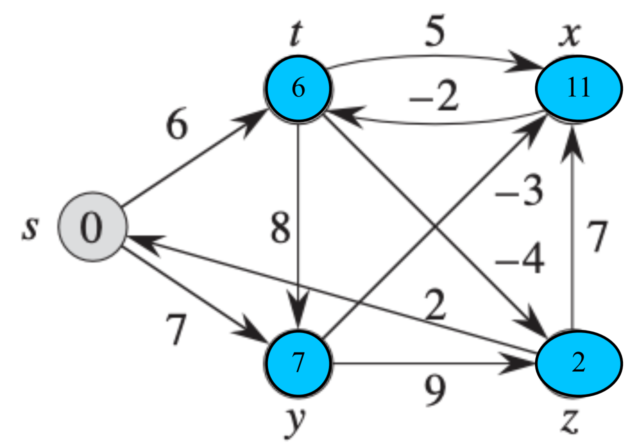
Assume we pass the edges in the following order:

$(t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)$

x x

RELAX(u, v, w)

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



BELLMAN-FORD(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 2
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

Example

Assume we pass the edges in the following order:

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$

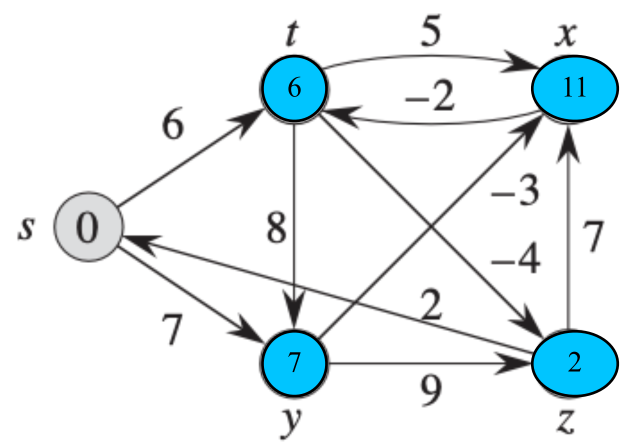
x x

RELAX(u, v, w)

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

BELLMAN-FORD(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| - 1          i = 2
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```



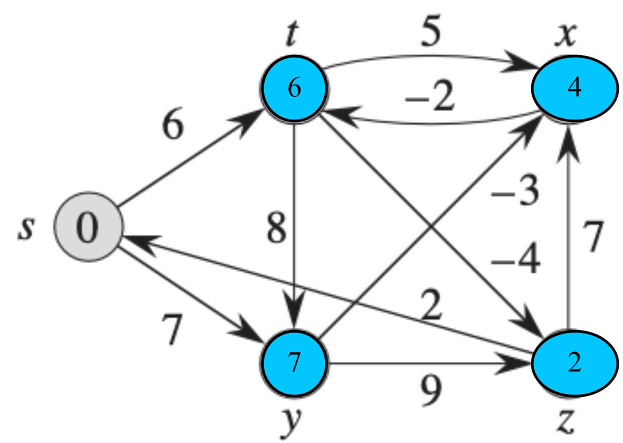
University of Stavanger

Example

Assume we pass the edges in the following order:

$(t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)$

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




```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1      i = 2
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Graph Algorithms (Chapter 24.1) SSSP (Bellman-Ford algorithm)

University of Stavanger

Example

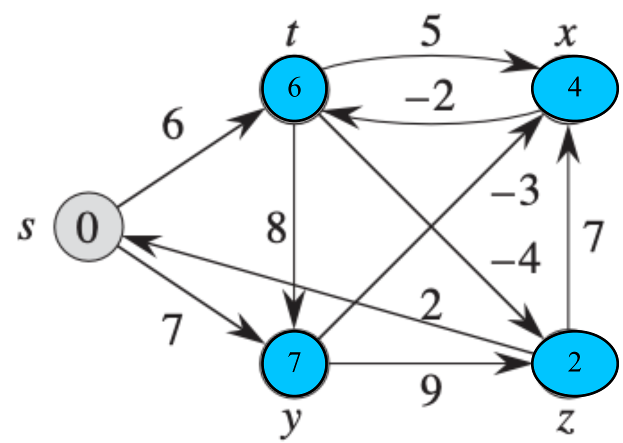
Assume we pass the edges in the following order:

$(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)$

x x x

RELAX(u, v, w)

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



BELLMAN-FORD(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 2
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

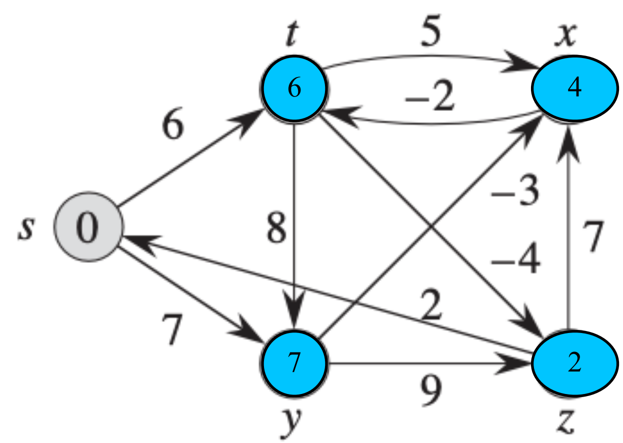
Example

Assume we pass the edges in the following order:

$(t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)$

RELAX(u, v, w)

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



BELLMAN-FORD(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 2
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Graph Algorithms (Chapter 24.1) SSSP (Bellman-Ford algorithm)

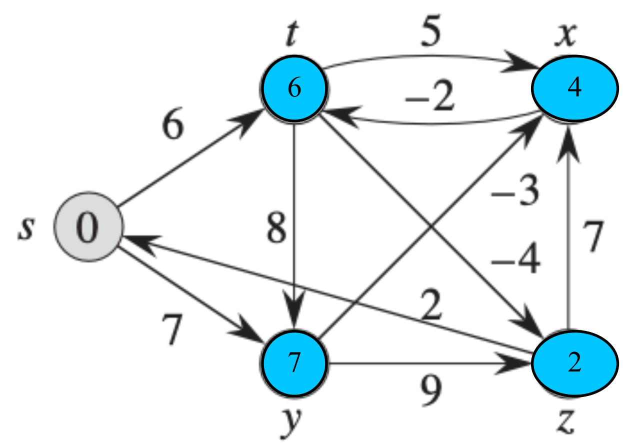
Example

Assume we pass the edges in the following order:

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$

RELAX(u, v, w)

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



BELLMAN-FORD(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 2
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

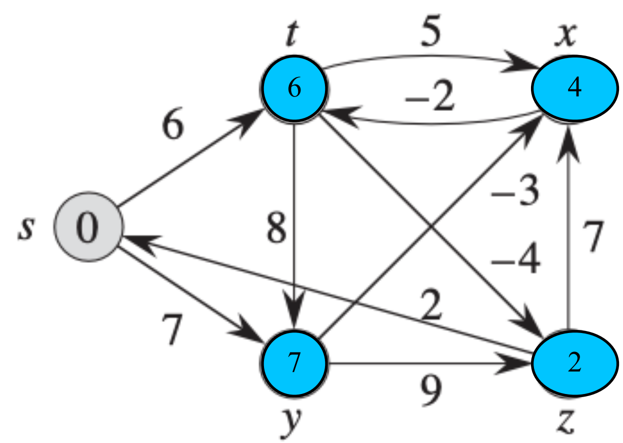
Example

Assume we pass the edges in the following order:

$(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)$

x x x

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



BELLMAN-FORD(G, w, s)
```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 2
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Graph Algorithms (Chapter 24.1) SSSP (Bellman-Ford algorithm)

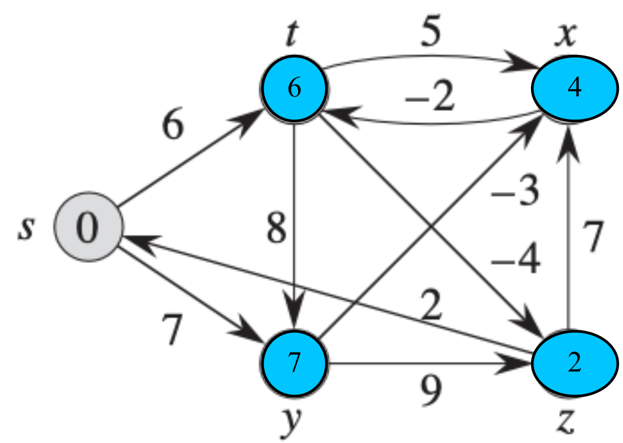
Example

Assume we pass the edges in the following order:

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$

x x x

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




```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 2
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

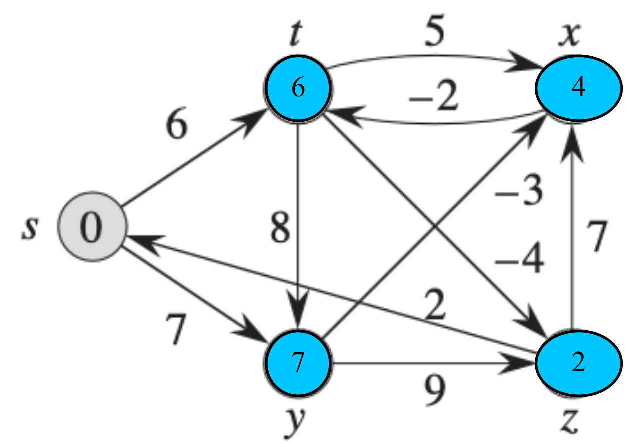
University of Stavanger

Example

Assume we pass the edges in the following order:

$(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)$

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




```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 3
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Graph Algorithms (Chapter 24.1) SSSP (Bellman-Ford algorithm)

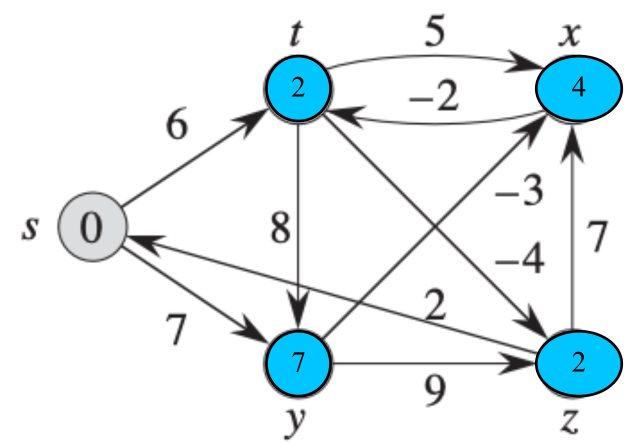
Example

Assume we pass the edges in the following order:

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$

x

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




```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 3
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

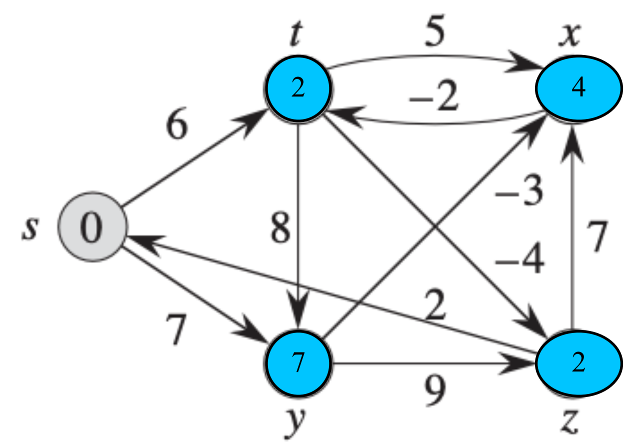
Example

Assume we pass the edges in the following order:

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$

RELAX(u, v, w)

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



BELLMAN-FORD(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| - 1        i = 4
    for each edge (u,v) ∈ G.E
        RELAX(u, v, w)
for each edge (u,v) ∈ G.E
    if v.d > u.d + w(u,v)
        return FALSE
return TRUE
```

Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

University of Stavanger

Example

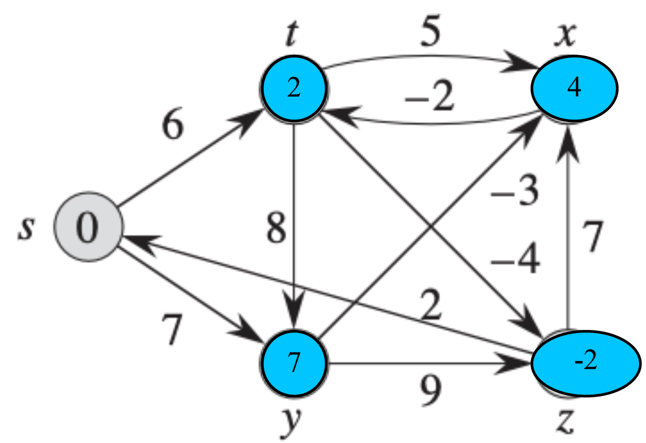
Assume we pass the edges in the following order:

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$

x

RELAX(u, v, w)

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



BELLMAN-FORD(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1        i = 4
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Graph Algorithms (Chapter 24.1) SSSP (Bellman-Ford algorithm)

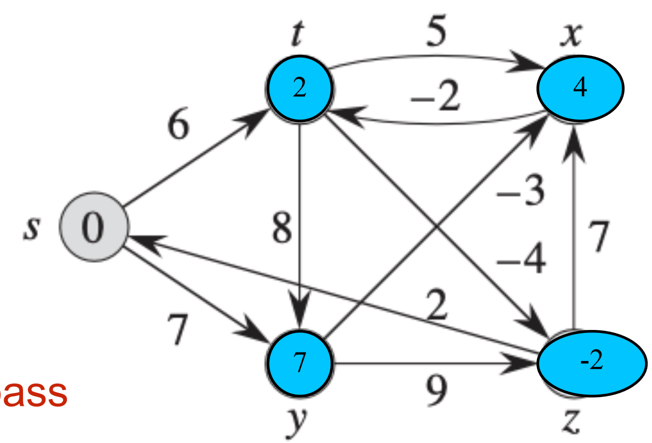
University of Stavanger

Example

Assume we pass the edges in the following order:

$$(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)$$

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




```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

What if $i = 5$ one more pass

Nothing changes any more !

$i = 4$ is enough!

Graph Algorithms (Chapter 24.1)
SSSP (Bellman-Ford algorithm)

Runtime complexity

The Bellman-Ford algorithm runs in time $O(VE)$, since the initialization in line 1 takes $\Theta(V)$ time, each of the $|V| - 1$ passes over the edges in lines 2–4 takes $\Theta(E)$ time, and the **for** loop of lines 5–7 takes $O(E)$ time.

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| - 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if v.d > u.d + w(u, v)
7          return FALSE
8  return TRUE
```

Graph Algorithms (Chapter 24.2)
SSSP (DAG)

Theorem 24.5

If a weighted, directed graph $G = (V, E)$ has source vertex s and no cycles, then at the termination of the DAG-SHORTEST-PATHS procedure, $v.d = \delta(s, v)$ for all vertices $v \in V$, and the predecessor subgraph G_π is a shortest-paths tree.

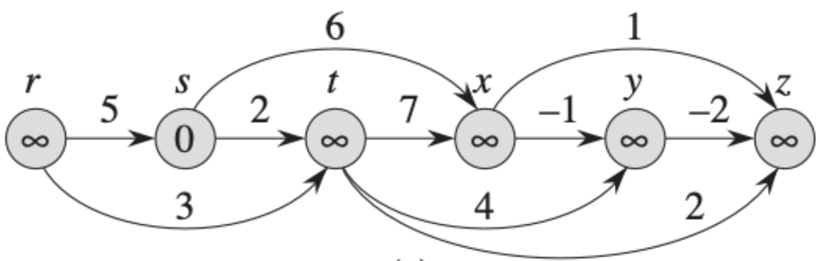
Intuition

By relaxing the edges of a weighted dag (directed acyclic graph) $G = (V, E)$ according to a topological sort of its vertices, we can compute shortest paths from a single source in $\Theta(V + E)$ time. Shortest paths are always well defined in a dag, since even if there are negative-weight edges, no negative-weight cycles can exist.

```
DAG-SHORTEST-PATHS(G, w, s)
topologically sort the vertices of G
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex u, taken in topologically sorted order
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Graph Algorithms (Chapter 24.2)
SSSP (DAG)

Example

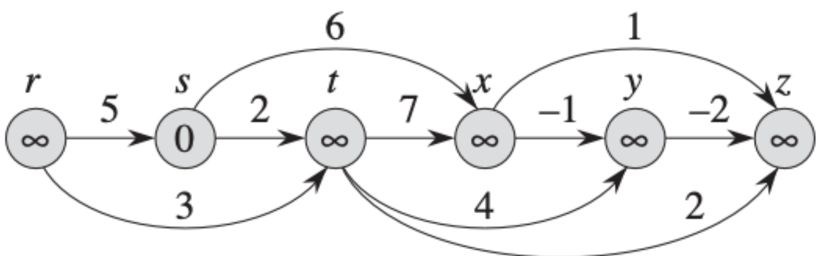


(a)

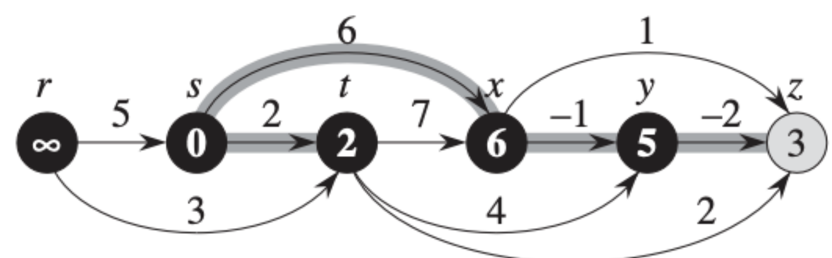
```
DAG-SHORTEST-PATHS(G, w, s)
topologically sort the vertices of G
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex u, taken in topologically sorted order
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Graph Algorithms (Chapter 24.2)
SSSP (DAG)

Example



(a)



(f)

DAG-SHORTEST-PATHS(G, w, s)

```
topologically sort the vertices of G
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex u, taken in topologically sorted order
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Graph Algorithms (Chapter 24.3)
SSSP (Dijkstra's Algorithm)

Solves SSSP on a weighted directed graph with nonnegative edge weights.

$$\forall (u, v) \in E, w(u, v) \geq 0$$

Graph Algorithms (Chapter 24.3)
SSSP (Dijkstra's Algorithm)

University of Stavanger

Intuition

Maintain a set S of vertices whose final shortest path weights from source s have already been determined. It repeatedly selects $u \in V - S$ with minimum distance estimate, adds u to S and relaxes outgoing edges from 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)
```

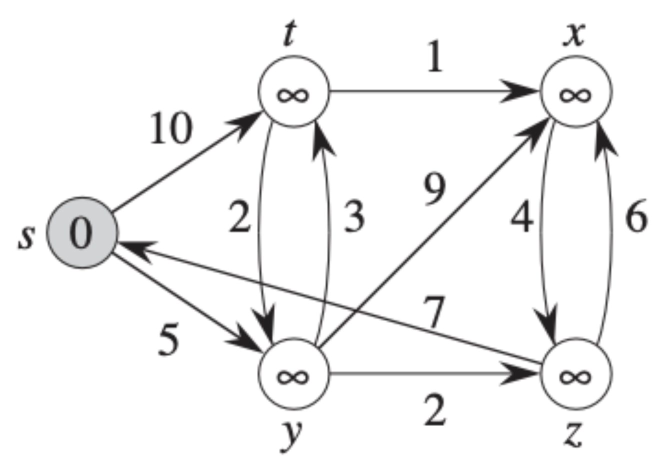
Example

Select min distance vertex = s

$S = \{s\}$

Relax (s, t), (s, y)

```
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

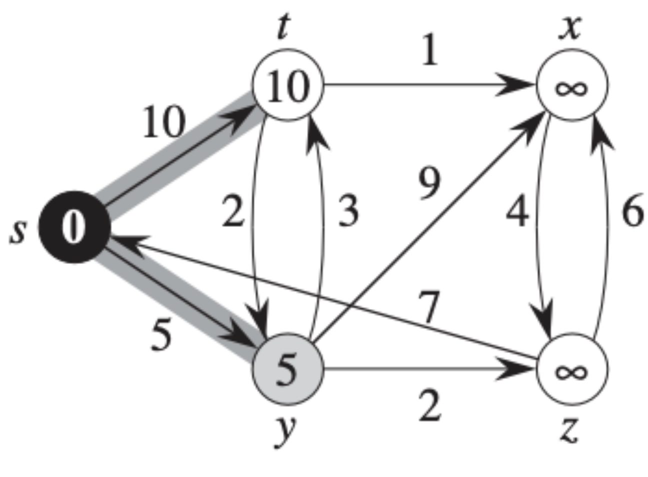
Select min distance vertex = s

$S = \{s\}$

Relax $(s, t), (s, y)$

DIJKSTRA(G, w, s)

1 INITIALIZE-SINGLE-SOURCE(G, s)
2 $S = \emptyset$
3 $Q = G.V$
4 **while** $Q \neq \emptyset$
5 $\quad u =$ EXTRACT-MIN(Q)
6 $\quad S = S \cup \{u\}$
7 $\quad$ **for** each vertex $v \in G.Adj[u]$
8 $\quad\quad$ RELAX(u, v, w)



Graph Algorithms (Chapter 24.3) SSSP (Dijkstra's Algorithm)

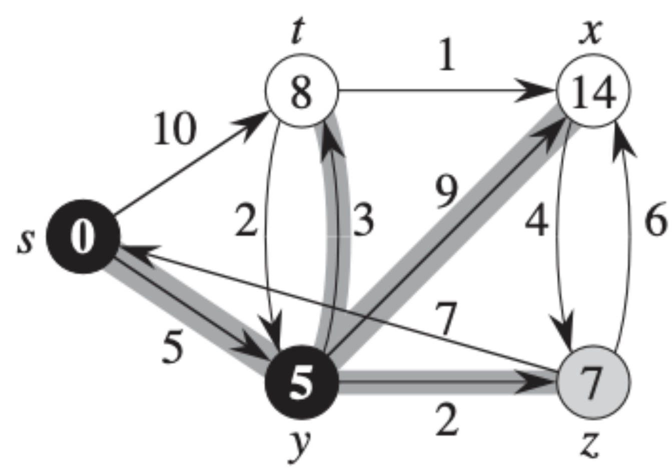
Example

Select min distance vertex = y

$S = \{s, y\}$

Relax $(y, t), (y, x), (y, z)$

```
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)
```



University of Stavanger

Example

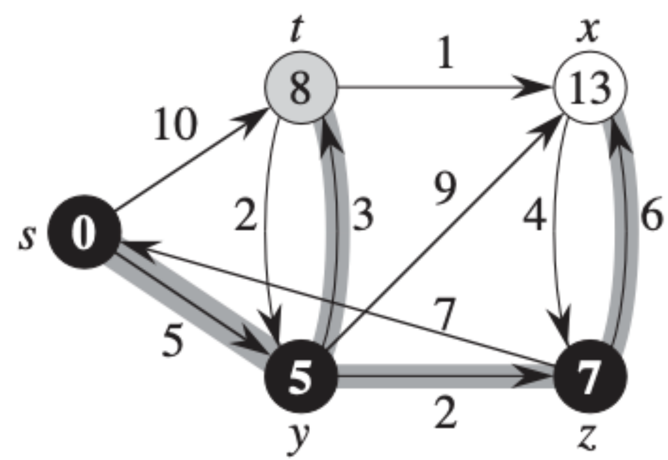
Select min distance vertex = z

$S = \{s, y, z\}$

Relax (z, s), (z, x)

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)
```



Graph Algorithms (Chapter 24.3)
SSSP (Dijkstra's Algorithm)

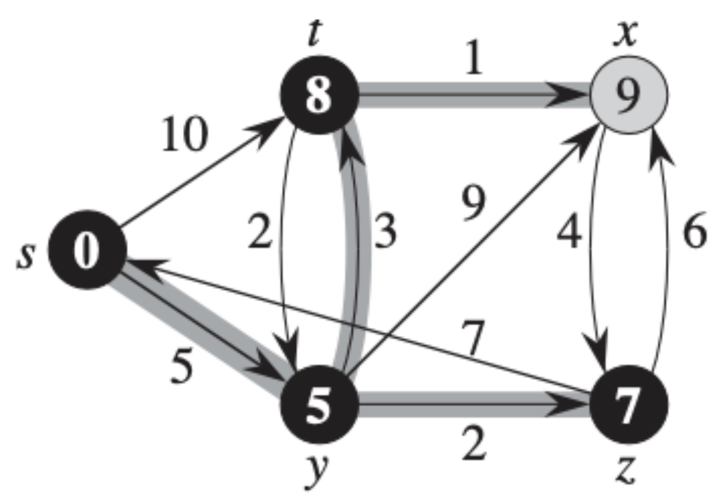
Example

Select min distance vertex = t

$S = \{s, y, z, t\}$

Relax $(t, x), (t, y)$

```
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)
```



Graph Algorithms (Chapter 24.3)
SSSP (Dijkstra's Algorithm)

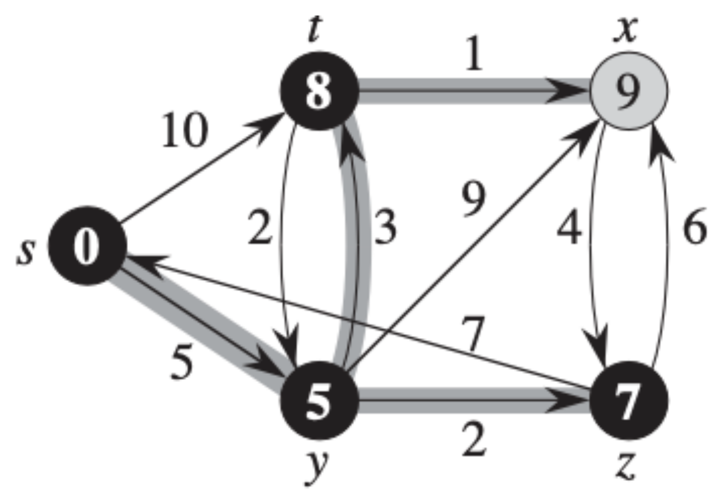
Example

Select min distance vertex = x

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

Relax (x, z)

```
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)
```



Graph Algorithms (Chapter 24.3) SSSP (Dijkstra's Algorithm)

University of Stavanger

Runtime complexity of Dijkstra's Algorithm

if we use min-heap for priority queue

$O(V \,.\, lgV)$ from lines 4-6

$O(E \,.\, lgV)$ from lines 7-8

$O((E+V) \,.\, lgV)$, if $E > V$ we have $O(E \,.\, lgV)$

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

$O(V)$ In total

$O(lgV)$ from min-heap

$O(E)$ In total

$O(lgV)$ because for decrease key in min-heap

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

Summary Chapter 24

Exercices

Will come