

CSE 301

Combinatorial Optimization

Single Source Shortest Path
Bellman Ford Algorithm
DAG

Negative-Weight Edges

- $s \rightarrow a$: only one path

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

- $s \rightarrow b$: only one path

$$\delta(s, b) = w(s, a) + w(a, b) = -1$$

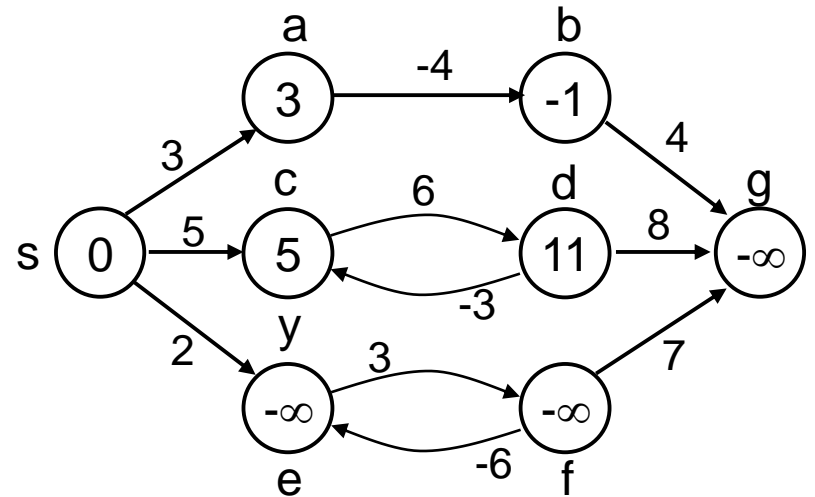
- $s \rightarrow c$: infinitely many paths

$$\langle s, c \rangle, \langle s, c, d, c \rangle, \langle s, c, d, c, d, c \rangle$$

cycle has positive weight ($6 - 3 = 3$)

$\langle s, c \rangle$ is shortest path with weight $\delta(s, c) = w(s, c) = 5$

What if we have negative-weight edges?



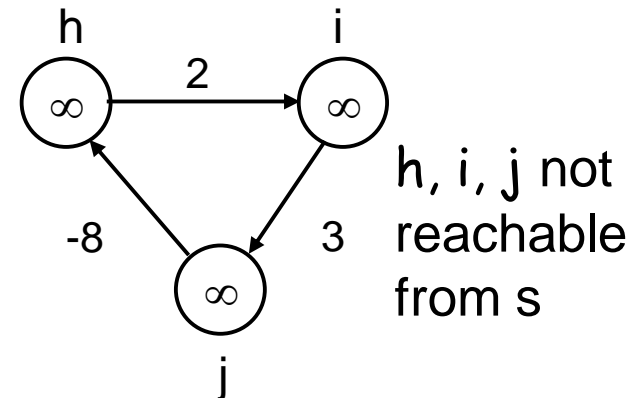
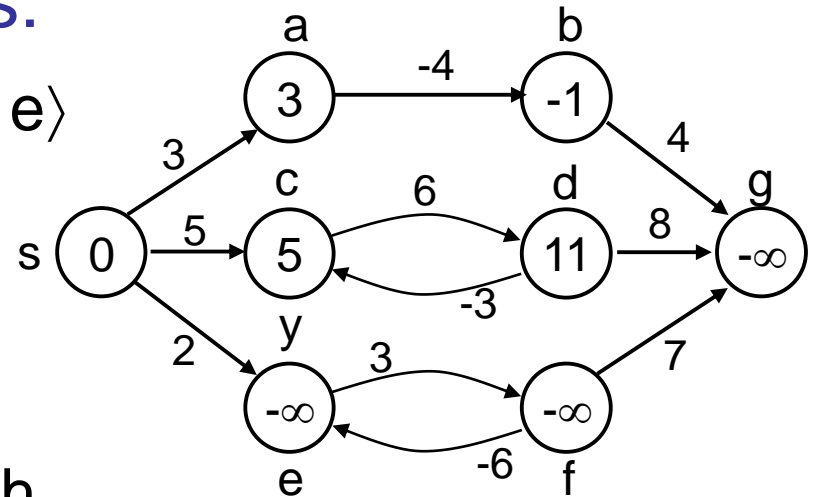
Negative-Weight Edges

- $s \rightarrow e$: infinitely many paths:

- $\langle s, e \rangle, \langle s, e, f, e \rangle, \langle s, e, f, e, f, e \rangle$
- cycle $\langle e, f, e \rangle$ has negative weight:

$$3 + (-6) = -3$$

- can find paths from s to e with arbitrarily large negative weights
- $\delta(s, e) = -\infty \Rightarrow$ no shortest path exists between s and e
- Similarly: $\delta(s, f) = -\infty$,
 $\delta(s, g) = -\infty$

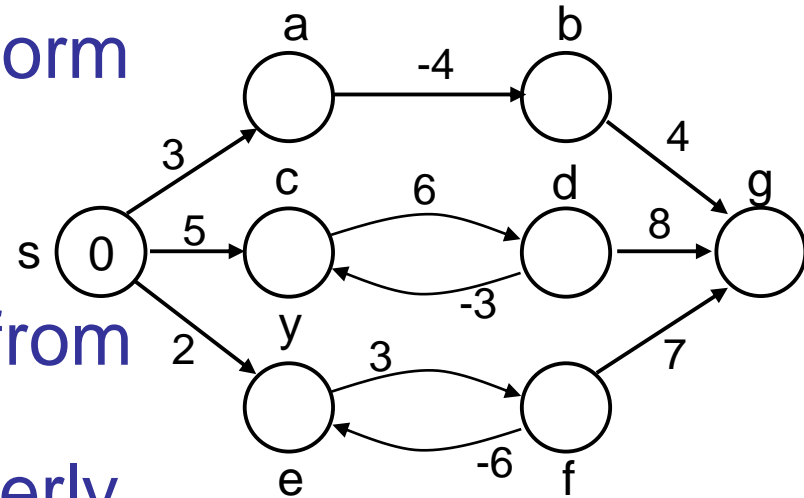


h, i, j not
reachable
from s

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

Negative-Weight Edges

- Negative-weight edges may form negative-weight cycles
- If such cycles are reachable from the source: $\delta(s, v)$ is not properly defined
 - Keep going around the cycle, and get $w(s, v) = -\infty$ for all v on the cycle



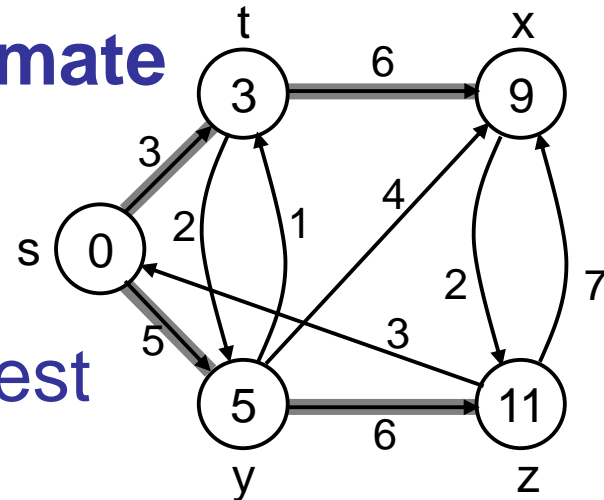
Cycles

- Can shortest paths contain cycles?
- Negative-weight cycles No!
- Positive-weight cycles: No!
 - By removing the cycle we can get a shorter path
- We will assume that when we are finding shortest paths, the paths will have no cycles

Shortest-Path Representation

For each vertex $v \in V$:

- $d[v] = \delta(s, v)$: a **shortest-path estimate**
 - Initially, $d[v] = \infty$
 - Reduces as algorithms progress
- $\pi[v] =$ **predecessor** of v on a shortest path from s
 - If no predecessor, $\pi[v] = \text{NIL}$
 - π induces a tree—**shortest-path tree**



Initialization

Alg.: INITIALIZE-SINGLE-SOURCE(V, s)

1. **for** each $v \in V$
2. **do** $d[v] \leftarrow \infty$
3. $\pi[v] \leftarrow \text{NIL}$
4. $d[s] \leftarrow 0$

- All the shortest-paths algorithms start with INITIALIZE-SINGLE-SOURCE

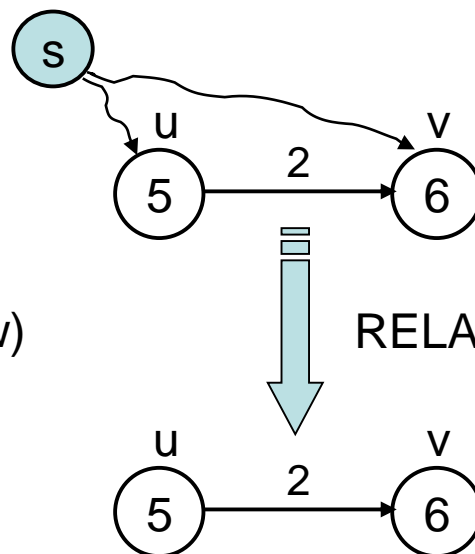
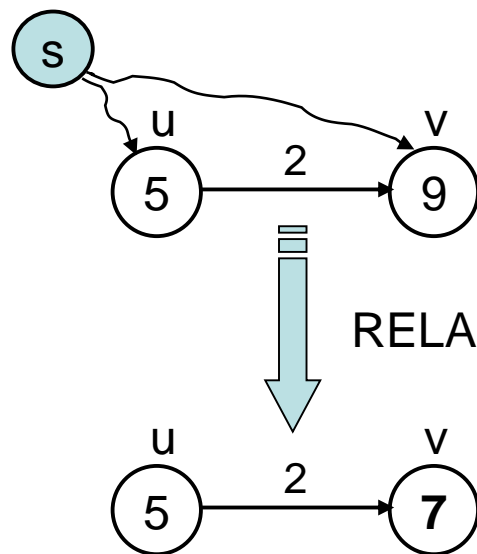
Relaxation

- **Relaxing** an edge (u, v) = testing whether we can improve the shortest path to v found so far by going through u

If $d[v] > d[u] + w(u, v)$

we can improve the shortest path to v

\Rightarrow update $d[v]$ and $\pi[v]$



After relaxation:
 $d[v] \leq d[u] + w(u, v)$

RELAX(u, v, w)

1. **if** $d[v] > d[u] + w(u, v)$
 2. **then** $d[v] \leftarrow d[u] + w(u, v)$
 3. $\pi[v] \leftarrow u$
-
- All the single-source shortest-paths algorithms
 - start by calling INIT-SINGLE-SOURCE
 - then relax edges
 - The algorithms differ in the order and how many times they relax each edge

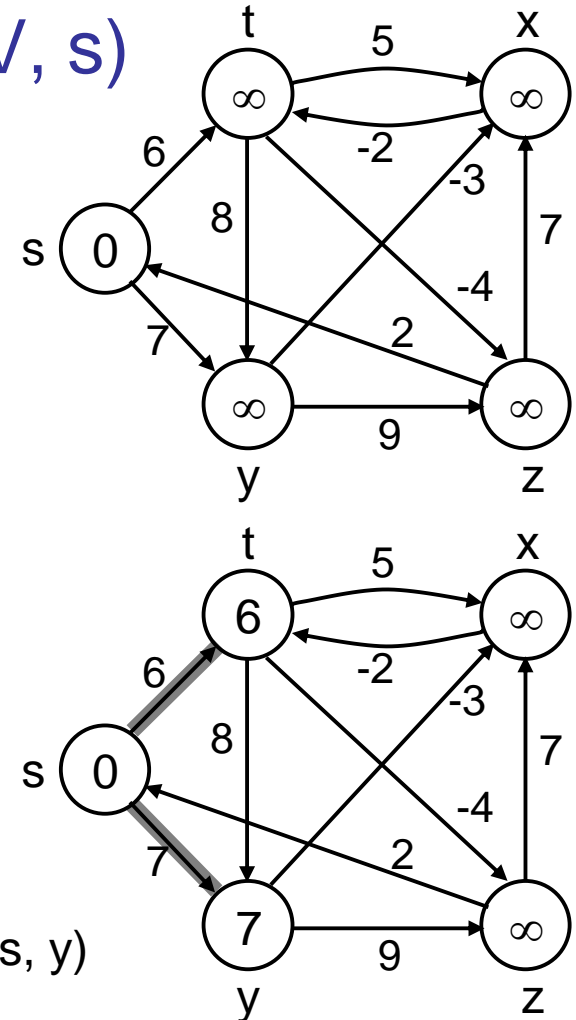
Bellman-Ford Algorithm

- Single-source shortest paths problem
 - Computes $d[v]$ and $\pi[v]$ for all $v \in V$
- Allows negative edge weights
- Returns:
 - **TRUE** if no negative-weight cycles are reachable from the source s
 - **FALSE** otherwise \Rightarrow no solution exists
- Idea:
 - Traverse all the edges $|V - 1|$ times, every time performing a relaxation step of each edge

BELLMAN-FORD(V, E, w, s)

1. INITIALIZE-SINGLE-SOURCE(V, s)
2. **for** $i \leftarrow 1$ to $|V| - 1$
3. **do for** each edge $(u, v) \in E$
4. **do** RELAX(u, v, w)
5. **for** each edge $(u, v) \in E$
6. **do if** $d[v] > d[u] + w(u, v)$
7. **then return** FALSE
8. **return** TRUE

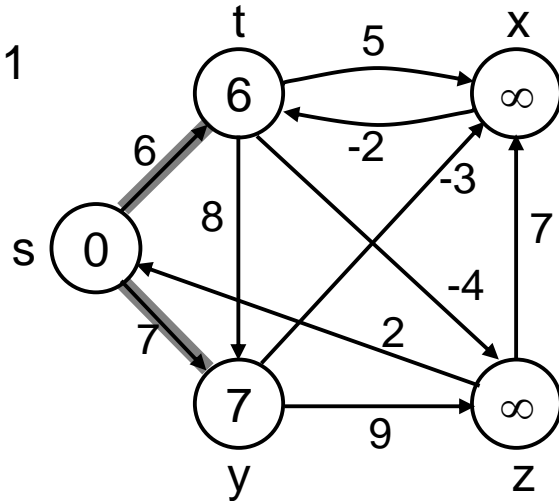
$E: (t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$



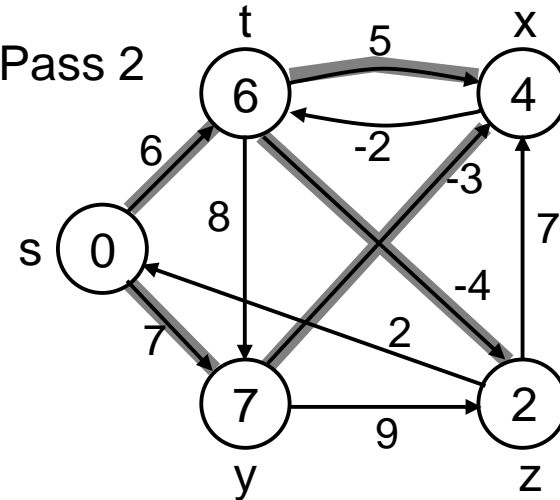
Example

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

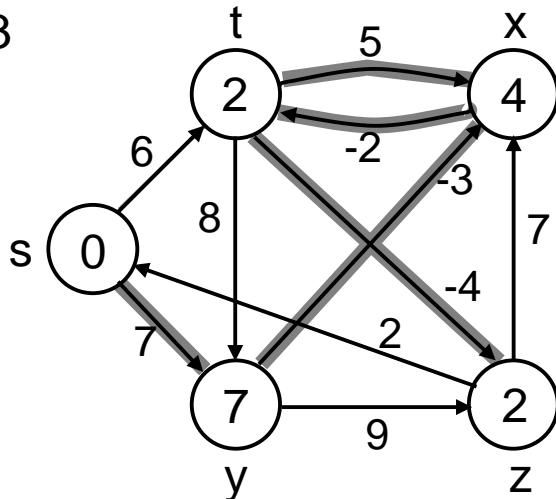
Pass 1



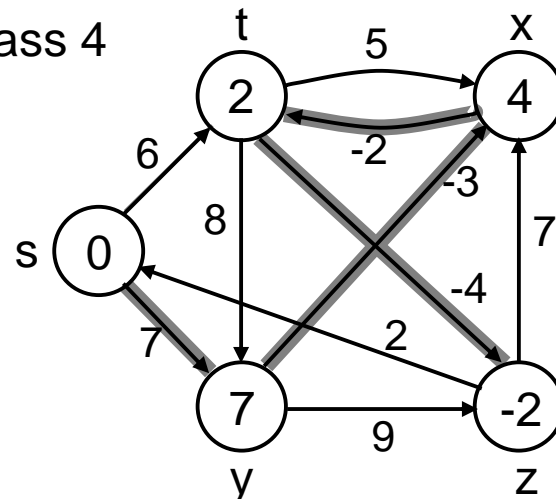
Pass 2



Pass 3

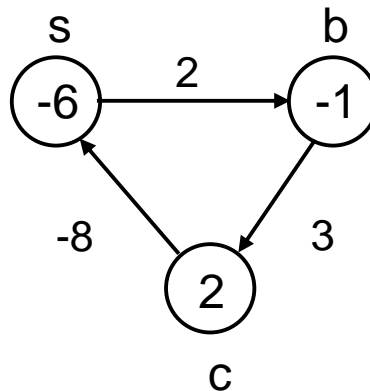
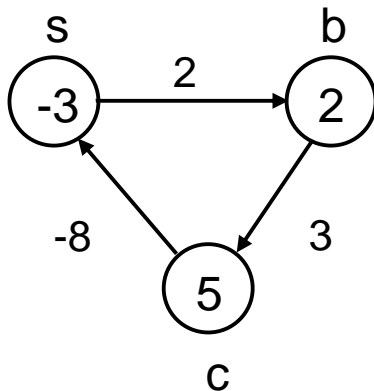
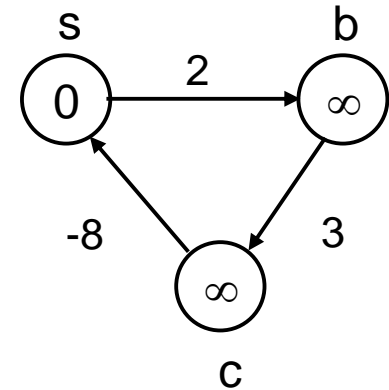


Pass 4



Detecting Negative Cycles

- **for** each edge $(u, v) \in E$
- **do if** $d[v] > d[u] + w(u, v)$
- **then return FALSE**
- **return TRUE**



Look at edge (s, b) :

$$d[b] = -1$$

$$d[s] + w(s, b) = -4$$

$$\Rightarrow d[b] > d[s] + w(s, b)$$

BELLMAN-FORD(V, E, w, s)

1. INITIALIZE-SINGLE-SOURCE(V, s) $\leftarrow \Theta(V)$
 2. **for** $i \leftarrow 1$ to $|V| - 1$ $\leftarrow O(V)$
 3. **do for** each edge $(u, v) \in E$ $\leftarrow O(E)$
 4. **do** RELAX(u, v, w)
 5. **for** each edge $(u, v) \in E$ $\leftarrow O(E)$
 6. **do if** $d[v] > d[u] + w(u, v)$
 7. **then return** FALSE
 8. **return** TRUE
- $\left. \begin{array}{l} \leftarrow O(V) \\ \leftarrow O(E) \end{array} \right\} O(VE)$

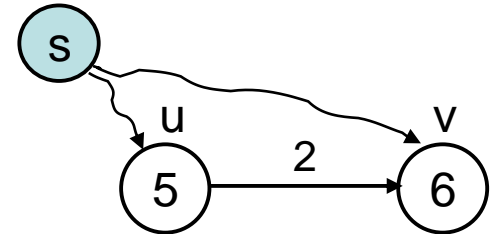
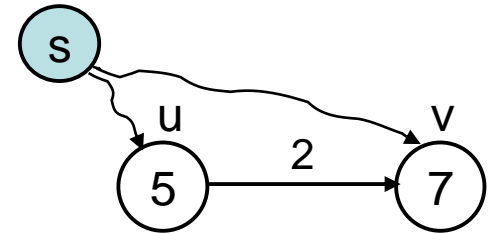
Running time: $O(VE)$

Shortest Path Properties

- **Triangle inequality**

For all $(u, v) \in E$, we have:

$$\delta(s, v) \leq \delta(s, u) + w(u, v)$$



- If u is on the shortest path to v we have the equality sign

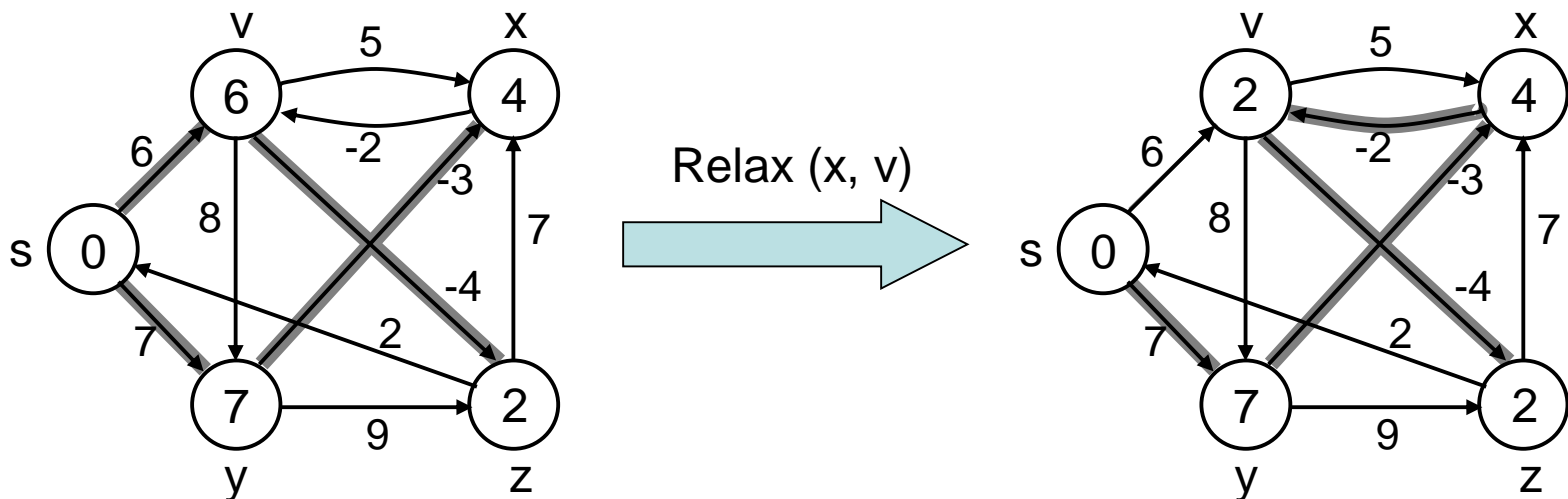
Shortest Path Properties

- Upper-bound property**

We always have $d[v] \geq \delta(s, v)$ for all v .

Once $d[v] = \delta(s, v)$, it never changes.

- The estimate never goes up – relaxation only lowers the estimate

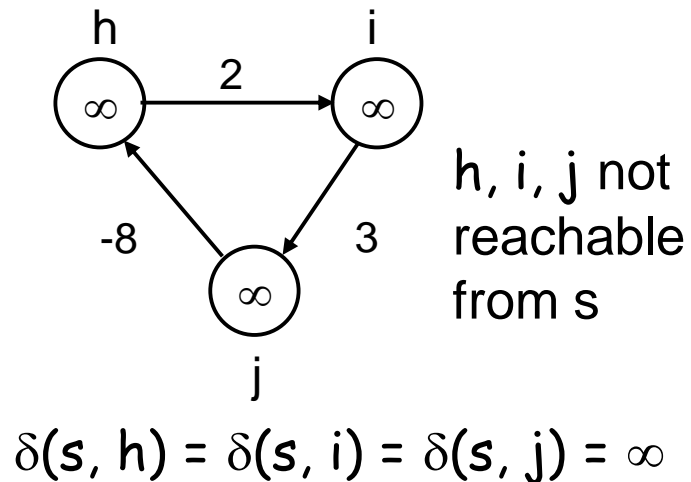
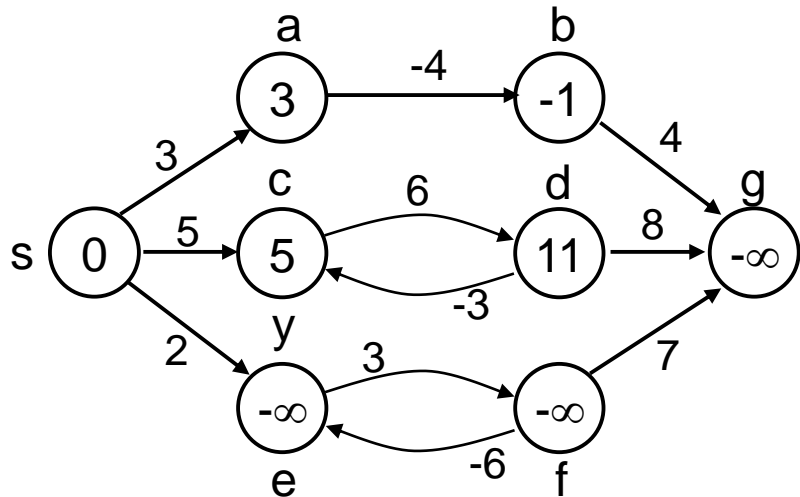


Shortest Path Properties

- No-path property**

If there is no path from s to v then $d[v] = \infty$ always.

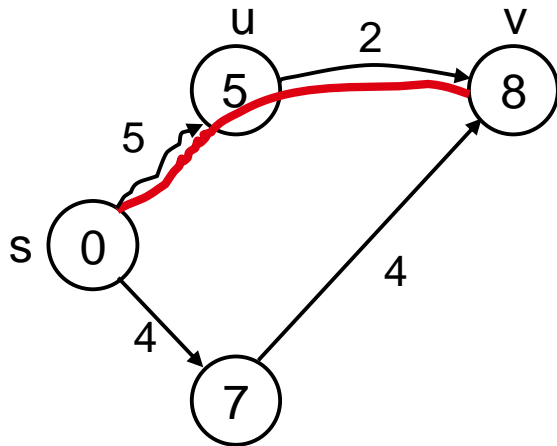
– $\delta(s, h) = \infty$ and $d[h] \geq \delta(s, h) \Rightarrow d[h] = \infty$



Shortest Path Properties

- **Convergence property**

If $s \rightsquigarrow u \rightarrow v$ is a shortest path, 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.

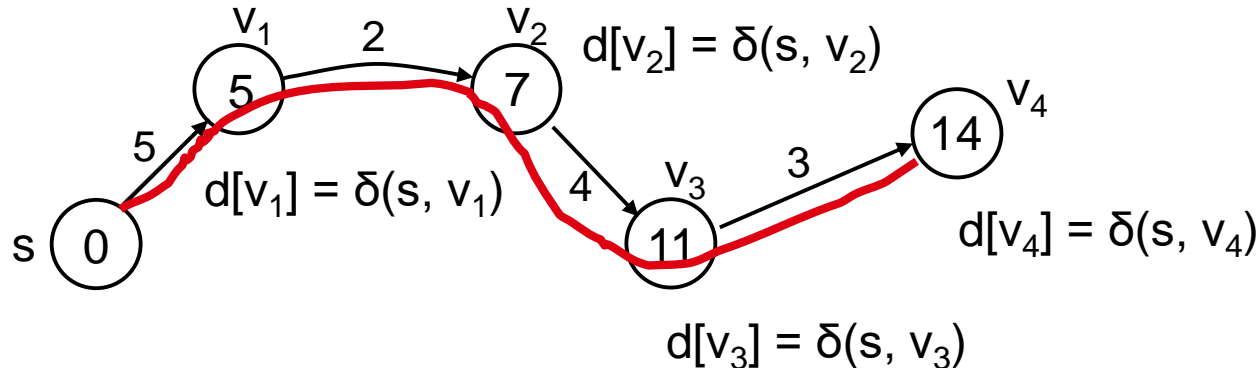


- If $d[v] > \delta(s, v) \Rightarrow$ after relaxation:
 $d[v] = d[u] + w(u, v)$
 $d[v] = 5 + 2 = 7$
- Otherwise, the value remains unchanged, because it must have been the shortest path value

Shortest Path Properties

- Path relaxation property**

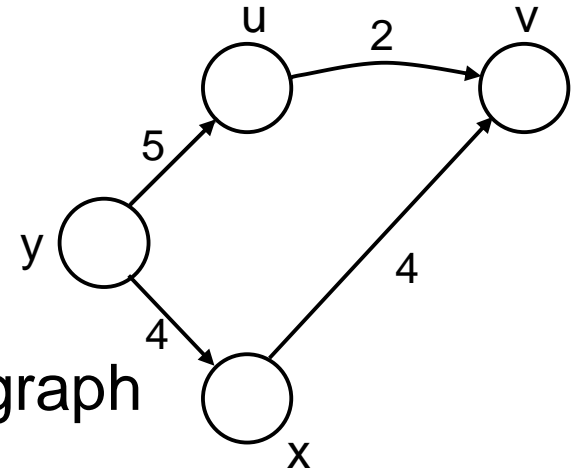
Let $p = \langle v_0, v_1, \dots, v_k \rangle$ be a shortest path from $s = v_0$ to v_k . If we relax, in order, (v_0, v_1) , (v_1, v_2) , \dots , (v_{k-1}, v_k) , even intermixed with other relaxations, then $d[v_k] = \delta(s, v_k)$.



Single-Source Shortest Paths in DAGs

Single-Source Shortest Paths in DAGs

- Given a weighted DAG: $G = (V, E)$
 - solve the shortest path problem
- Idea:
 - Topologically sort the vertices of the graph
 - Relax the edges according to the order given by the topological sort
 - for each vertex, we relax each edge that starts from that vertex
- Are shortest-paths well defined in a DAG?
 - Yes, (negative-weight) cycles cannot exist

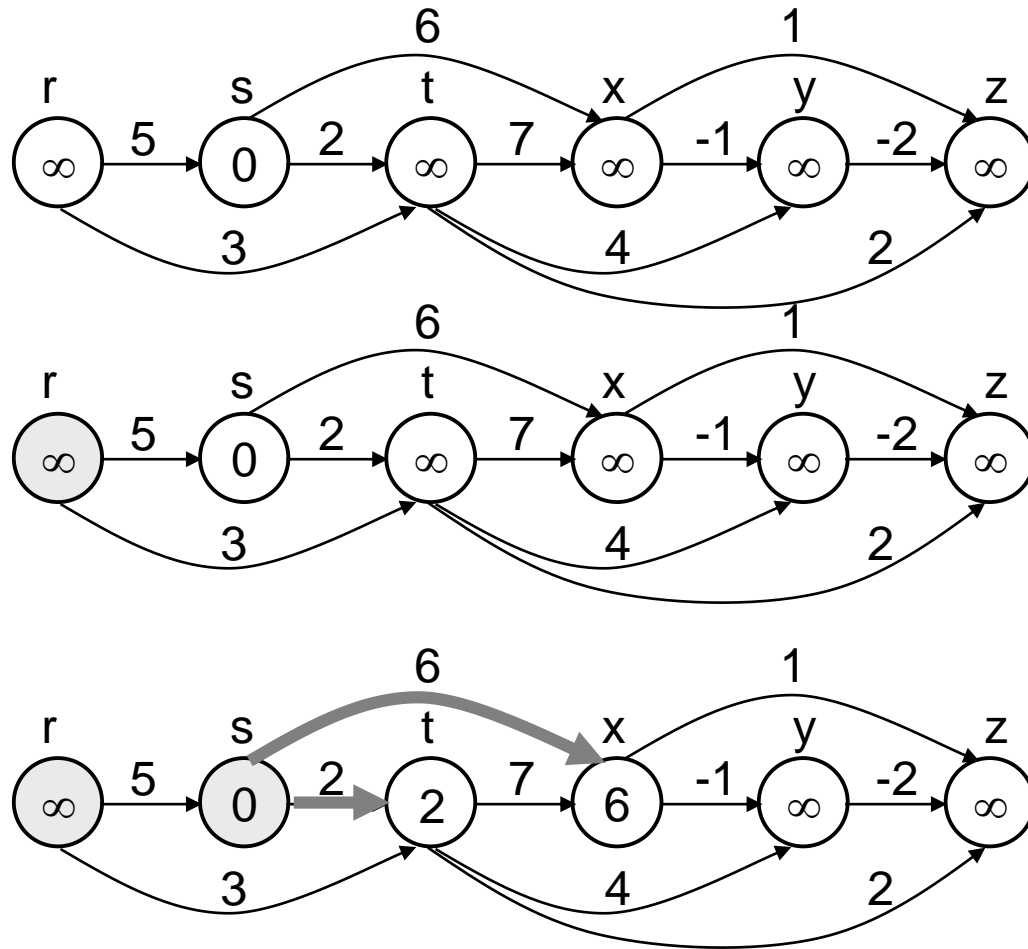


DAG-SHORTEST-PATHS(G, w, s)

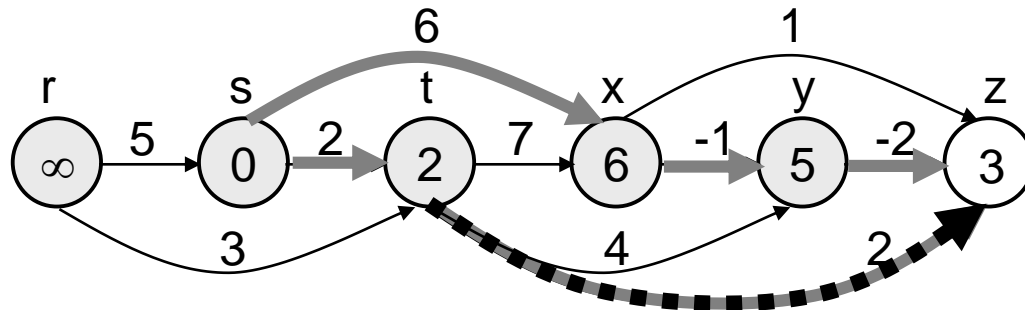
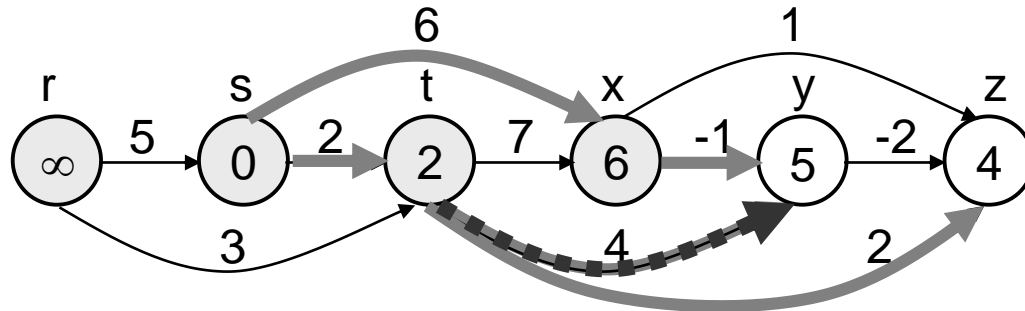
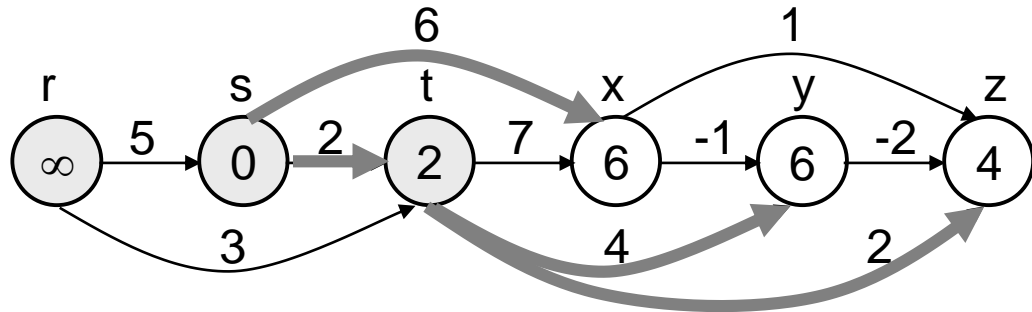
1. topologically sort the vertices of G $\leftarrow \Theta(V+E)$
 2. INITIALIZE-SINGLE-SOURCE(V, s) $\leftarrow \Theta(V)$
 3. **for** each vertex u , taken in topologically sorted order $\Theta(V)$
 4. **do for** each vertex $v \in \text{Adj}[u]$
 5. **do** RELAX(u, v, w)
- $\left. \begin{array}{l} \Theta(V) \\ \Theta(E) \end{array} \right\} \Theta(E)$

Running time: $\Theta(V+E)$

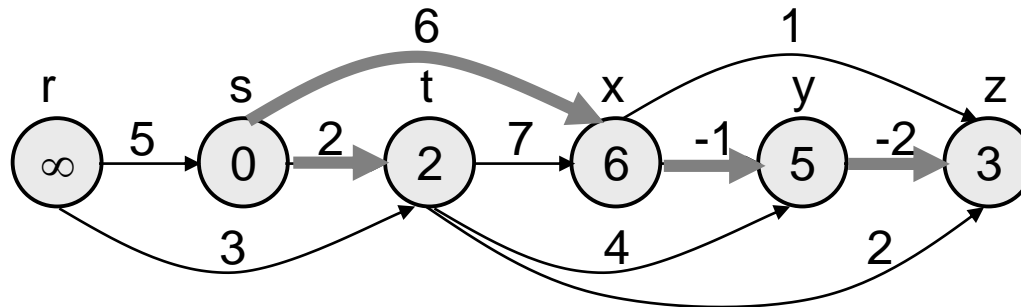
Example



Example (cont.)



Example (cont.)



Readings

- Chapter 24
- Exercise
 - 24.1-6 – Find negative cycle
 - 24.2-4 – Total Number of paths in a DAG
- Difficult Problems (Solve these if you want):
 - 24.3-6 modify dijkstra
 - 24-2 – nesting boxes
 - 24-3 - Arbitrage
 - 24.6 – Bitonic Shortest path