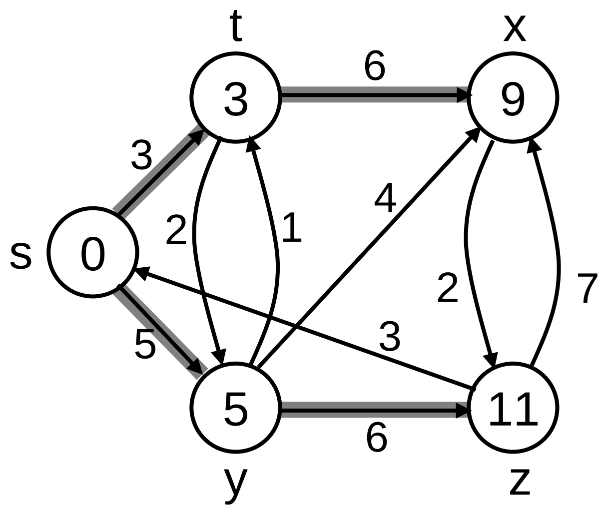
# ADVANCED ANALYSIS OF ALGORITHMS
# CPS 5440

# SHORTEST PATH ALGORITHMS

# SHORTEST PATH PROBLEM




- **Input:**
  - Directed graph $G = (V, E)$
  - Weight function $w : E \rightarrow \boldsymbol{R}$
- **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 $v$:

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

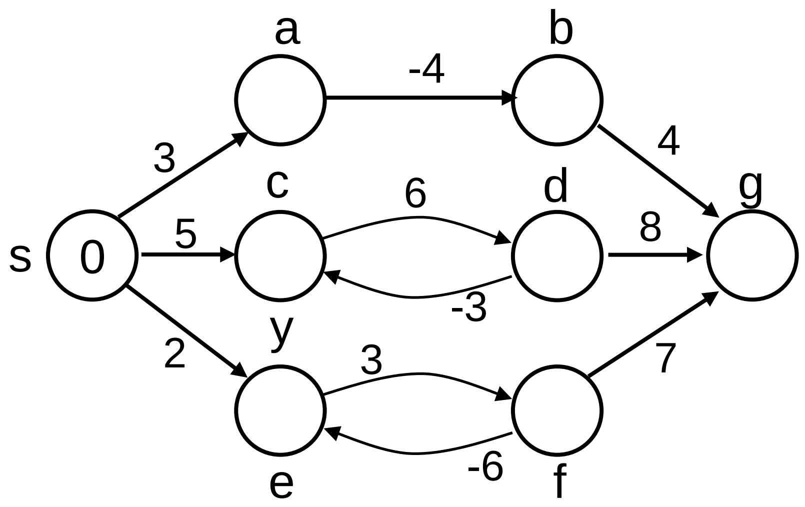
- **Note:** there might be <u>multiple shortest</u> paths from $u$ to $v$

# VARIANTS OF SHORTEST PATHS

- **Single-source shortest paths**
  - $G = (V, E)$: find a shortest path from a given source vertex $s$ to each vertex $v \in V$
- **Single-destination shortest paths**
  - Find a shortest path to a given destination vertex $\boldsymbol{t}$ from each vertex $v$
  - Reversing the direction of each edge $\Rightarrow$ single-source
- **Single-pair shortest path**
  - Find a shortest path from $u$ to $v$ for given vertices $u$ and $v$
- **All-pairs shortest-paths**
  - Find a shortest path from $u$ to $v$ for every pair of vertices $u$ and $v$

# NEGATIVE-WEIGHT EDGES

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



# NEGATIVE-WEIGHT EDGES

- $s \to a$: only one path

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

- $s \to b$: only one path

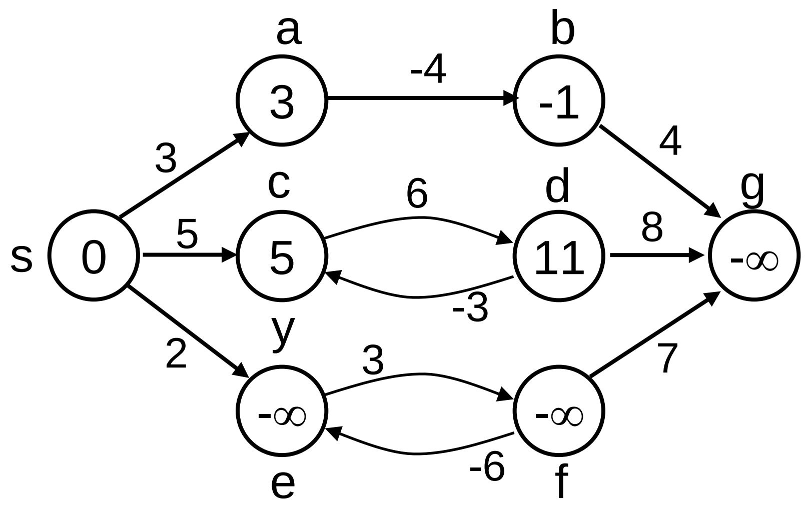
  $\delta(s,b) = w(s,a) + w(a,b) = -1$

- $s \to c$: 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 $\delta(s,c) = w(s,c) = 5$

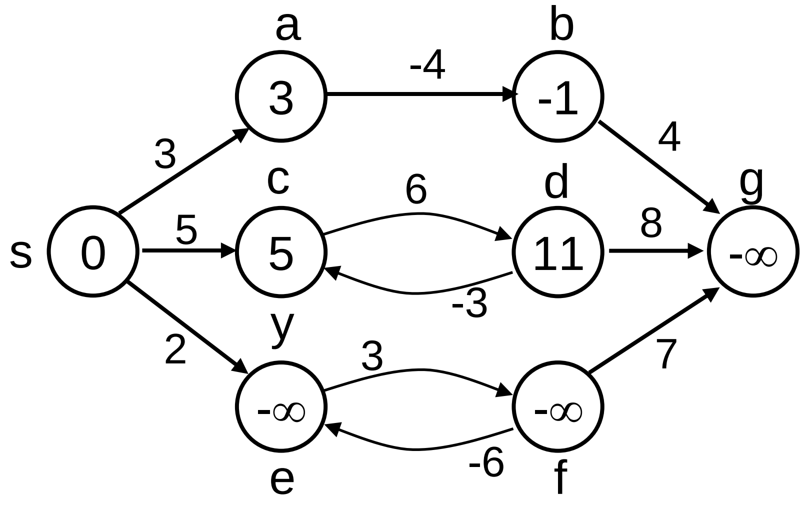


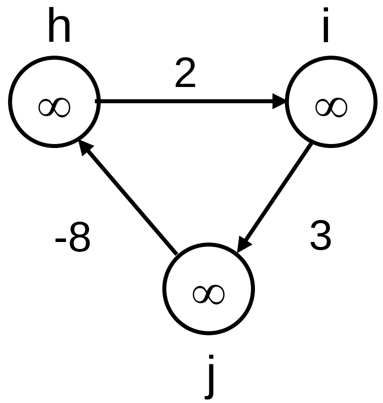
- $s \to e$: infinitely many paths:
  - $\langle s, e\rangle, \langle s, e, f, e\rangle, \langle s, e, f, e, f, e\rangle$
  - cycle $\langle e, f, e\rangle$ has negative weight:

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

  - 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

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

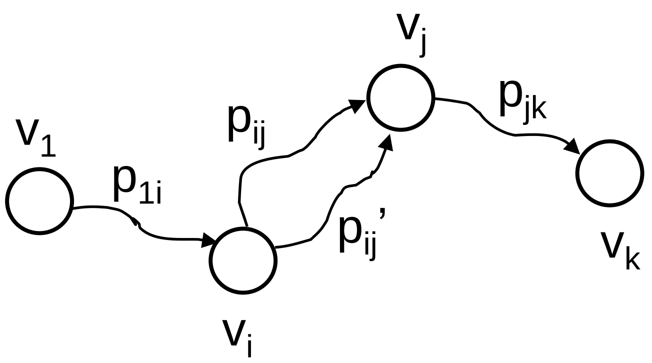
# CYCLES

- Can shortest paths contain cycles?
- Negative-weight cycles
  – Shortest path is not well defined
- Positive-weight cycles:
  – By removing the cycle, we can get a shorter path
- Zero-weight cycles
  – No reason to use them
  – Can remove them to obtain a path with same weight

# OPTIMAL SUBSTRUCTURE THEOREM

Given:

- A weighted, directed graph $G = (V, E)$
- A weight function $w\colon E \rightarrow R$,
- A shortest path $p = \langle v_1, v_2, \ldots, v_k \rangle$ from $v_1$ to $v_k$
- A subpath of $p$: $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$, with $1 \leq i \leq j \leq k$




Then: $p_{ij}$ is a shortest path from $v_i$ to $v_j$

**Proof**: $p = v_1 \overset{p_{1i}}{\leadsto} vi \overset{p_{ij}}{\leadsto} v_j \overset{p_{jk}}{\leadsto} v_k$

$$w(p) = w(p_{1i}) + w(p_{ij}) + w(p_{jk})$$

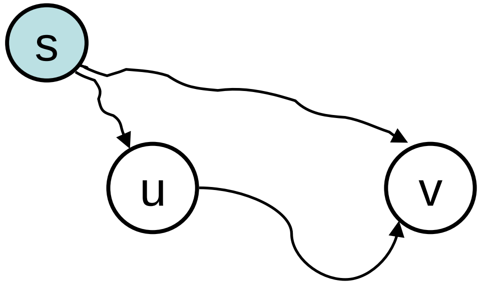
Assume $\exists$ $p_{ij}$' from $v_i$ to $v_j$ with $w(p_{ij}') < w(p_{ij})$

$\Rightarrow w(p') = w(p_{1i}) + w(p_{ij}') + w(p_{jk}) < w(p)$ contradiction!

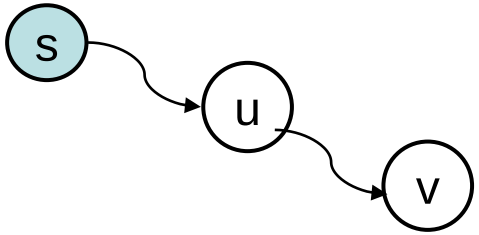
# TRIANGLE INEQUALITY

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

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




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

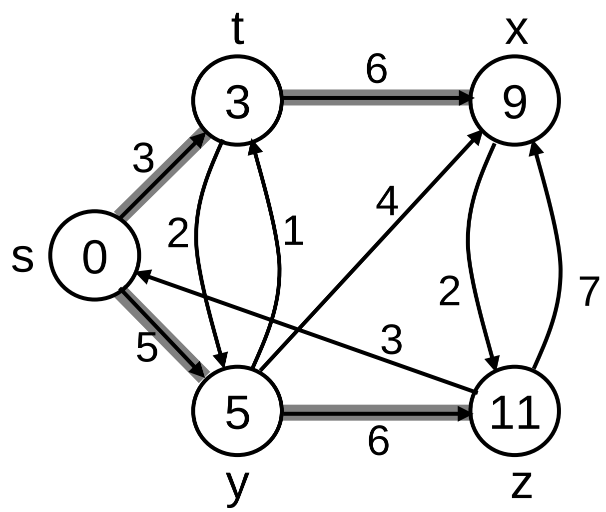


# ALGORITHMS

- Bellman-Ford algorithm
  - Negative weights are allowed
  - Negative cycles reachable from the source are not allowed.
- Dijkstra’s algorithm
  - Negative weights are not allowed
- Operations common in both algorithms:
  - Initialization
  - Relaxation

For each vertex $v \in V$:

- $\delta(s, v)$: **shortest-path weight**
- $d[v]$: shortest-path weight **estimate**
  - Initially, $d[v] = \infty$
  - $d[v] \rightarrow \delta(s, v)$ as algorithm progresses
- $\pi[v] =$ **predecessor** of $v$ on a shortest path from $s$
  - If no predecessor, $\pi[v] = NIL$
  - $\pi$ induces a tree—**shortest-path tree**



# INITIALIZATION

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

*1.* **for** $each\ v \in V$ **do**

*2.* $d[v] \leftarrow \infty$

*3.* $\pi[v] \leftarrow NIL$

*4.* $d[s] \leftarrow 0$

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

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

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

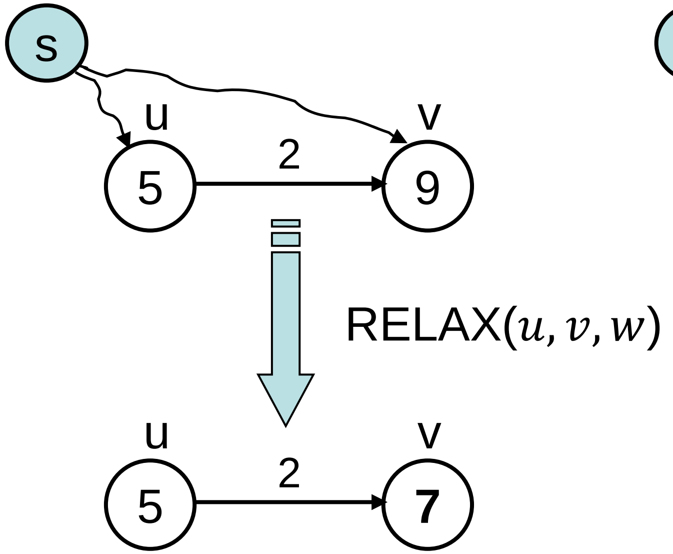
  we can improve the shortest path to $v$

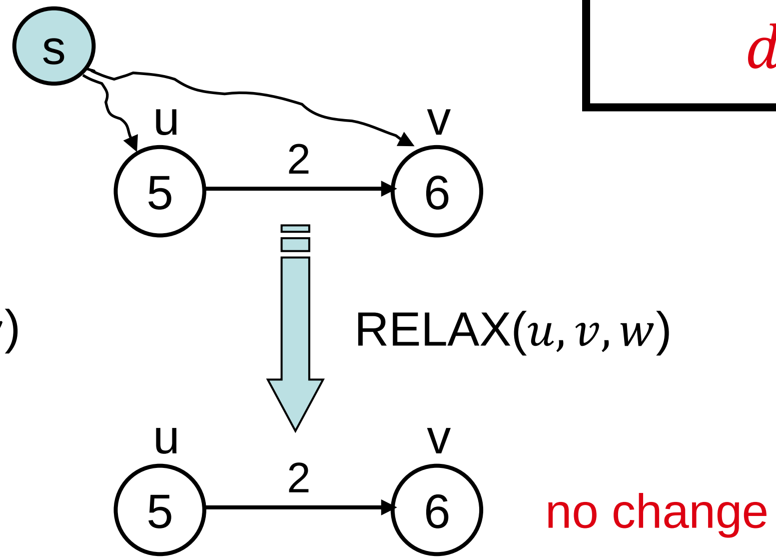
  $\Rightarrow d[v] = d[u] + w(u, v)$

  $\Rightarrow \pi[v] \leftarrow u$

After relaxation:

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



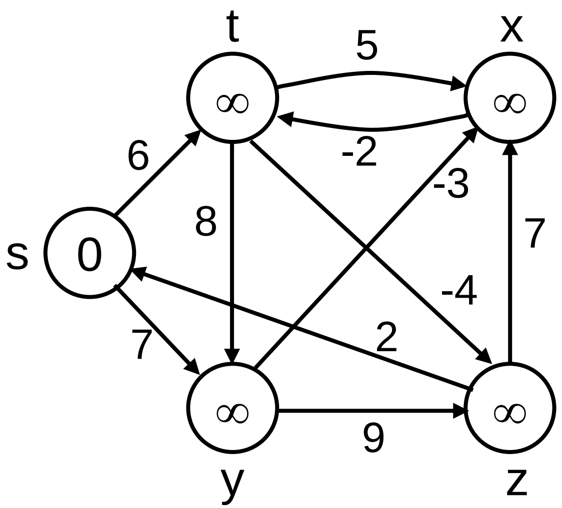



# BELLMAN-FORD ALGORITHM

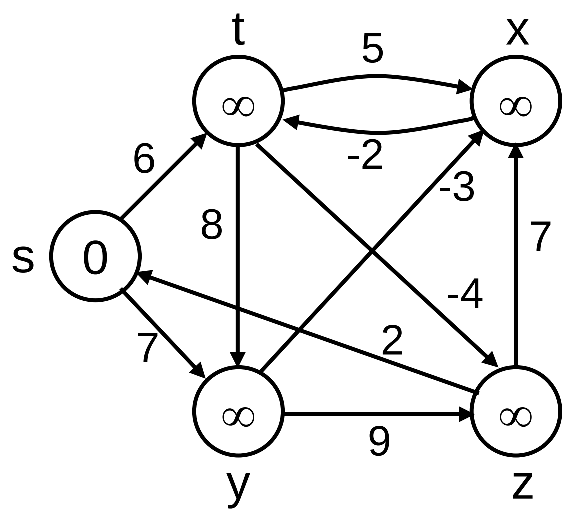
- Single-source shortest path problem
  - Computes $\delta(s, v)$ and $\pi[v]$ for all $v \in V$
- Allows negative edge weights - can detect negative cycles.
  - Returns TRUE if no negative-weight cycles are reachable from the source $s$
  - Returns FALSE otherwise $\Rightarrow$ no solution exists

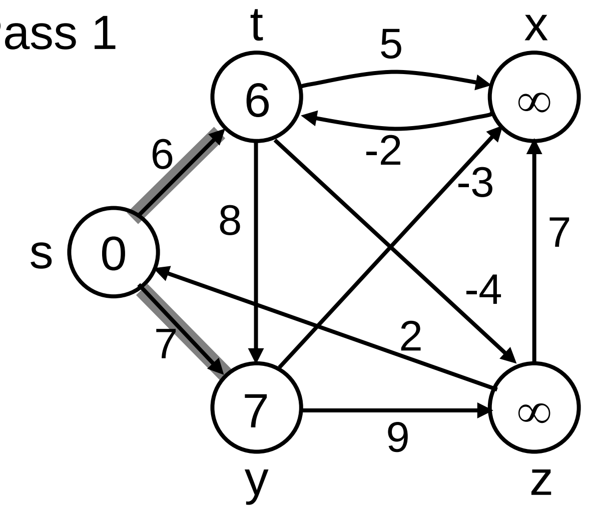
- Idea:
  - Each edge is relaxed $|V - 1|$ times by making $|V - 1|$ passes over the whole edge set.
  - To ensure that each edge is relaxed exactly $|V - 1|$ times, it puts the edges in an unordered list and goes over the list $|V - 1|$ times.

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

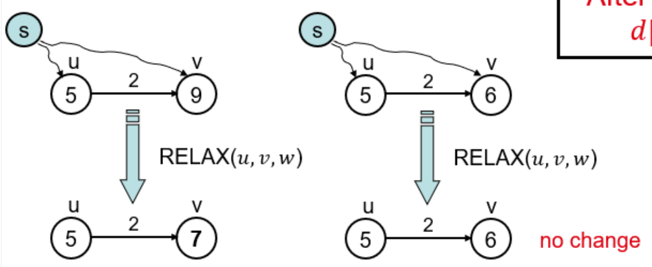


# BELLMAN-FORD(V, E, W, S)




If $d[v] > d[u] + w(u,v)$
we can improve the shortest path to $v$
$\Rightarrow d[v] = d[u] + w(u,v)$
$\Rightarrow \pi[v] \leftarrow u$



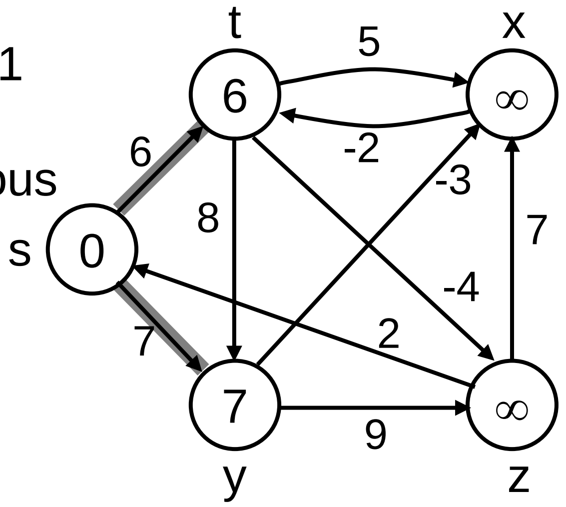

After relaxation:
$d[v] \leq d[u] + w(u,v)$

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

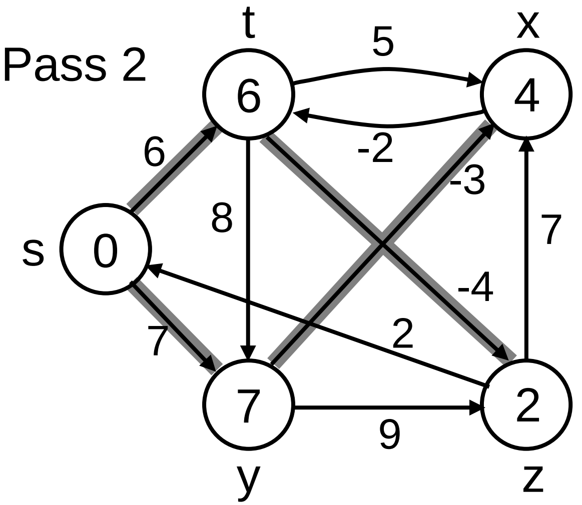
# EXAMPLE

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

Pass 1 (from previous slide)




Pass 2




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

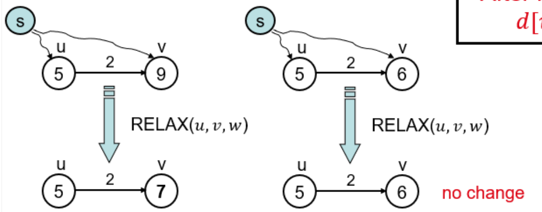
we can improve the shortest path to $v$

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

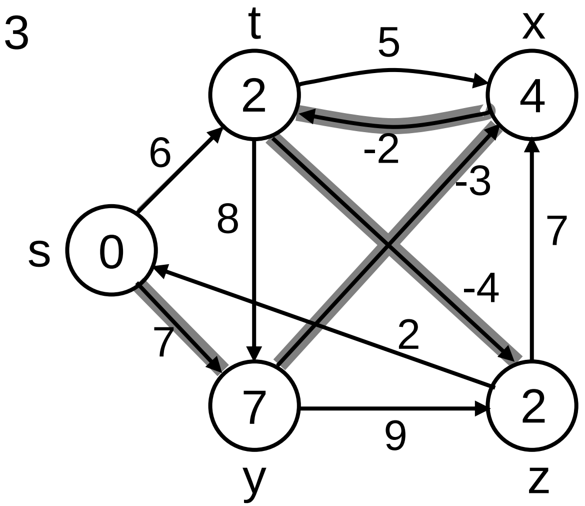
$\Rightarrow \pi[v] \leftarrow u$

After relaxation:

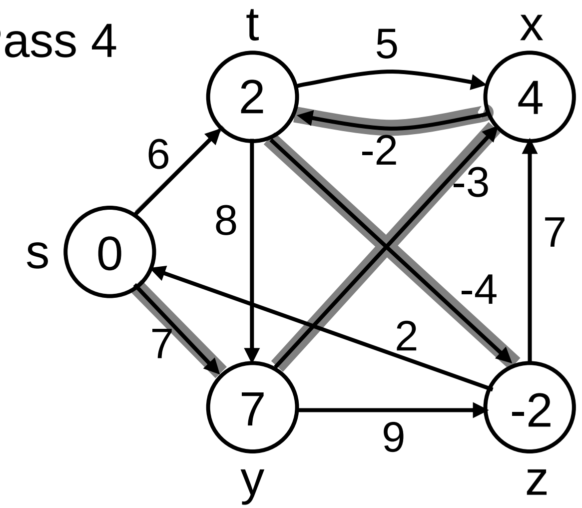
$d[v] \leq d[u] + w(u,v)$




Pass 3



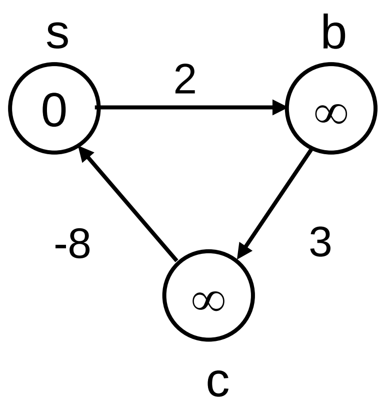

Pass 4



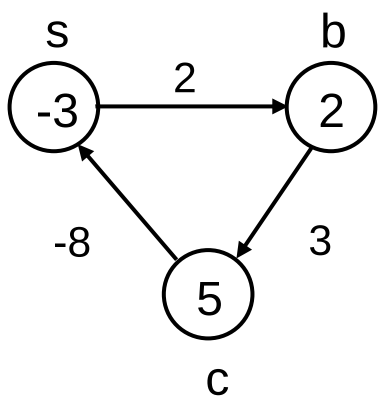
# DETECTING NEGATIVE CYCLES

❑ (perform extra test after $V - 1$ iterations)

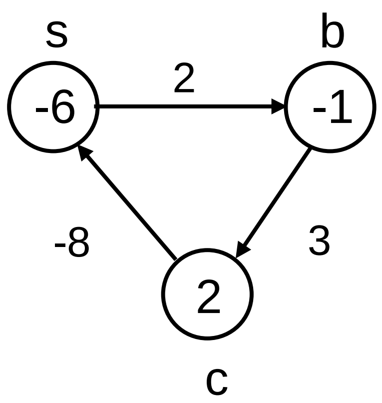
- **for** each edge $(u, v) \in E$ **do**
- $\boldsymbol{if}\ d[v] > d[u] + w(u, v)$
- **then return** FALSE
- **return** TRUE




$1^{st}$ pass




$2^{nd}$ pass




$(s, b)\ (b, c)\ (c, s)$

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)

INITIALIZE-SINGLE-SOURCE(V, s) $\longleftarrow \Theta(V)$

**for** $i \leftarrow 1\ to\ |V| - 1$ $\longleftarrow O(V)$

  **do for** each edge $(u, v) \in E$ $\longleftarrow O(E)$

$\left.\begin{matrix} O(V) \\ O(E) \end{matrix}\right\} \boldsymbol{O(VE)}$

    **do** RELAX$(u, v, w)$

**for** each edge $(u, v) \in E$ $\longleftarrow O(E)$

  **do if** $d[v] > d[u] + w(u, v)$

    **then return** FALSE

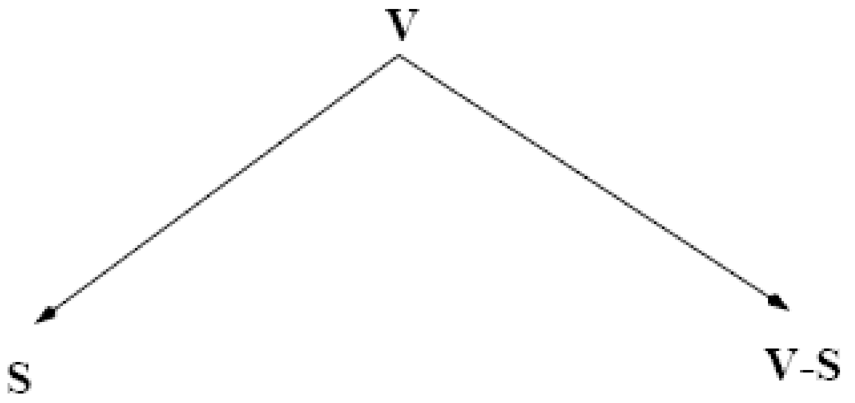
**return** TRUE

Running time: $O(V + VE + E) = O(VE)$

1. Visualization
2. Implementation

# DIJKSTRA'S ALGORITHM

- Single-source shortest path problem:
  - No negative-weight edges: $w(u, v) > 0, \ \forall (u, v) \in E$
- Each edge is relaxed **only once!**
- Maintains two sets of vertices:




$d[v] = \delta\ (s, v)$

$d[v] > \delta\ (s, v)$

# DIJKSTRA'S ALGORITHM (CONT.)

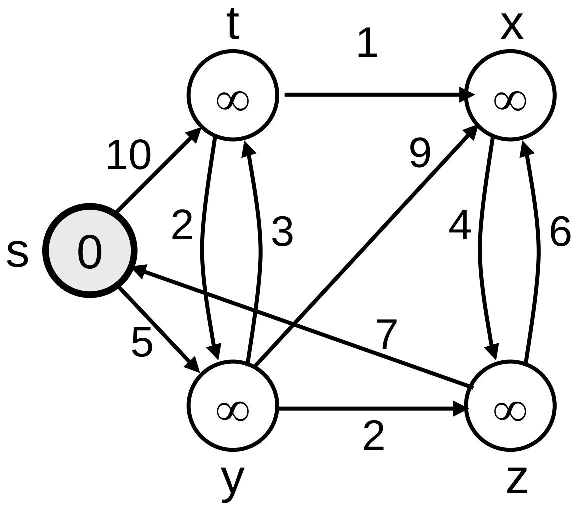
- Vertices in $V - S$ reside in a min-priority queue
  - Keys in $Q$ are estimates of shortest-path weights $d[u]$
- Repeatedly select a vertex $u \in V - S$, with the minimum shortest-path estimate $d[u]$
- Relax all edges leaving $u$

- **Steps**

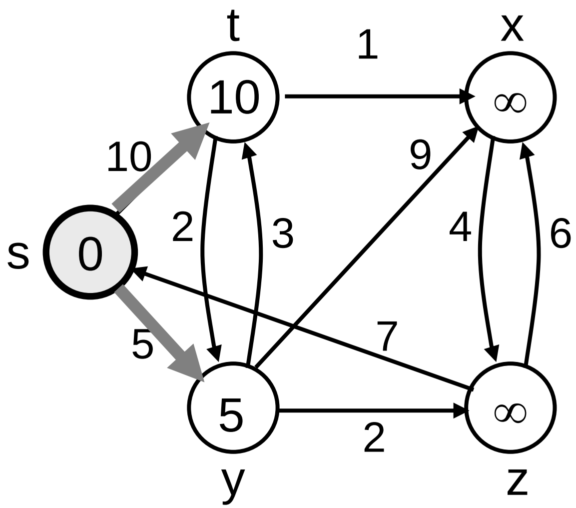
  1) Extract a vertex $u$ from $Q$ (i.e., $u$ has the highest priority)
  2) Insert $u$ to $S$
  3) Relax all edges leaving $u$
  4) Update $Q$

# DIJKSTRA (G,W,S)

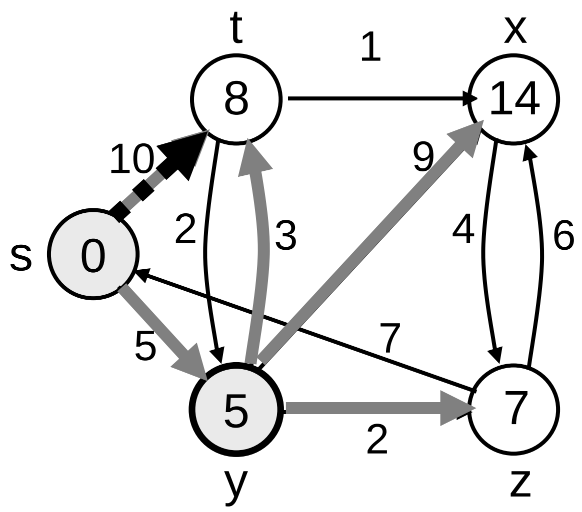
$S = <>\ \ Q = < s, t, x, z, y >$

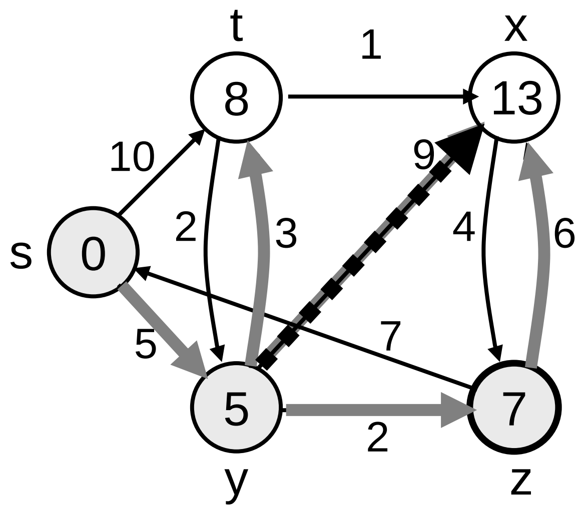



$S =< s >$



# EXAMPLE (CONT.)



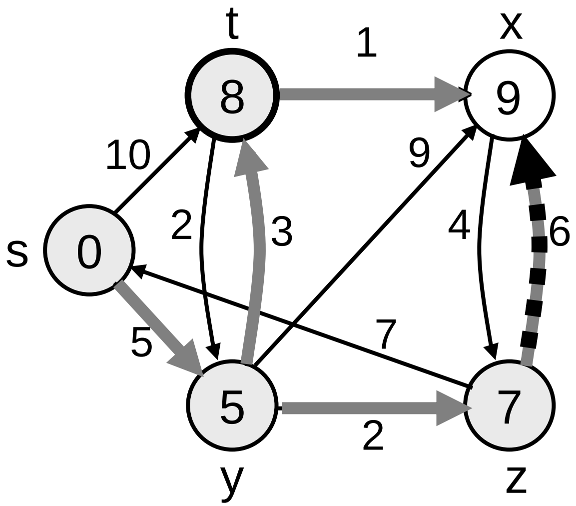




$S =< s, y > \;\; Q =< z, t, x >$

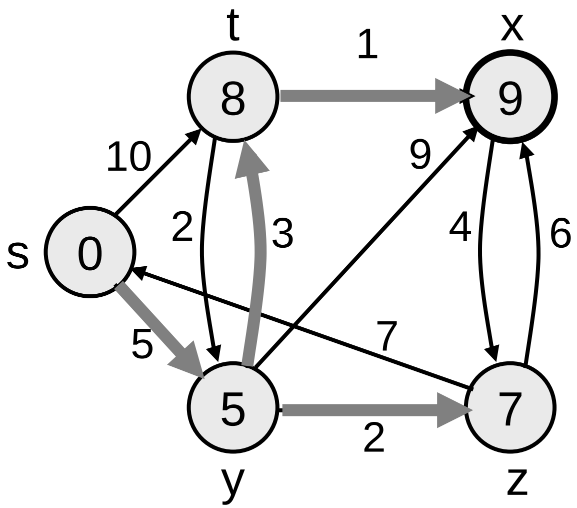
$S =< s, y, z > \;\; Q =< t, x >$

# EXAMPLE (CONT.)

$S =< s, y, z, t > \ \ Q =< x >$




$S =< s, y, z, t, x > \ \ Q =<>$



# DIJKSTRA (G, W, S)

INITIALIZE-SINGLE-SOURCE(V, s) $\longleftarrow$ $\Theta(V)$

S ← ∅

Q ← V[G] $\longleftarrow$ $O(V)$ build min-heap

**while** Q ≠ ∅ $\longleftarrow$ Executed $O(V)$ times

    **do** u ← EXTRACT-MIN(Q) $\longleftarrow$ $O(lgV)$

$O(VlgV)$

        S ← S ∪ {u}

        **for** each vertex v ∈ Adj[u] $\longleftarrow$ $O(E)$ times (total)

            **do** RELAX(u, v, w)

            Update Q (DECREASE_KEY) $\longleftarrow$ $O(lgV)$

$O(ElgV)$

Running time: $O(VlgV + ElgV) = O(ElgV)$

# DIJKSTRA (G,W,S)

- Look at different Q implementation, as did for Prim's algorithm

| Q | $T_{E\text{-}MIN}$ | $T_{D\text{-}KEY}$ | TOTAL |
|---|---|---|---|
| Linear Unsorted Array: | $O(V)$ | $O(1)$ | $O(V^2 + E)$ |
| Binary Heap: | $O(lgV)$ | $O(logV)$ | $O(VlgV + ElgV) = O(ElgV)$ |
| Fibonacci heap: | $O(lgV)$ | $O(1)$ | $O(VlgV + E)$ |

***Observe :***

- Each vertex is extracted from $Q$ and inserted into $S$ exactly once
- Each edge is relaxed exactly once
- $S$ = set of vertices whose final shortest paths have already been determined
  - $i.e., S = \{v$ Î $V: d[v] = \delta(s, v) \neq \infty\}$

# DIJKSTRA'S ALGORITHM FOR SHORTEST PATHS.

- *Similar to BFS algorithm:* $S$ corresponds to the set of black vertices in BFS, which have their correct breadth-first distances already computed

- *Greedy strategy:* Always chooses the closest(lightest) vertex in $Q = V - S$ to insert into $S$

- Relaxation may reset $d[v]$ values thus updating $Q = \boldsymbol{DECREASE - KEY}$ operation.

# DIJKSTRA'S ALGORITHM FOR SHORTEST PATHS.

- Similar to Prim's MST algorithm: Both algorithms use a priority queue to find the lightest vertex outside a given set *S*
- Insert this vertex into the set
- Adjust weights of remaining adjacent vertices outside the set accordingly

1. Visualization
2. Implementation


Q
A