Design And Analysis of Algorithms

Single Source Shortest Path

Bellman Ford Algorithm

Negative-Weight Edges

- $s \rightarrow a$: only one path

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

- $s \rightarrow b$: only one path

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

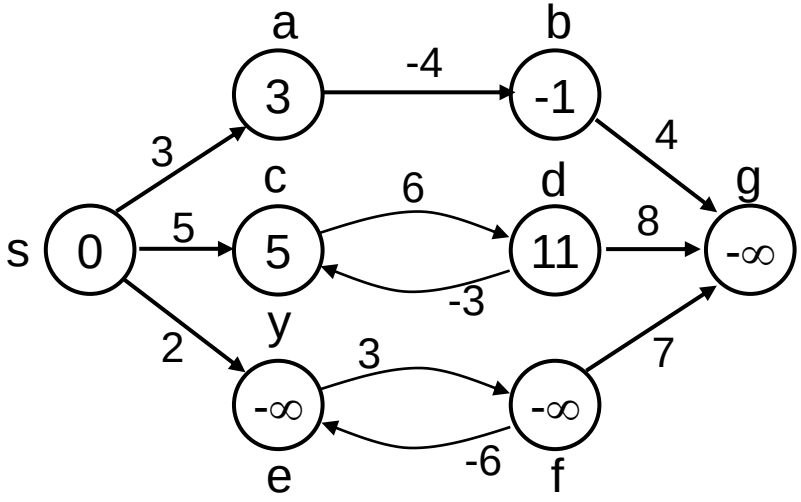
- $s \rightarrow c$: infinitely many paths

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

  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?

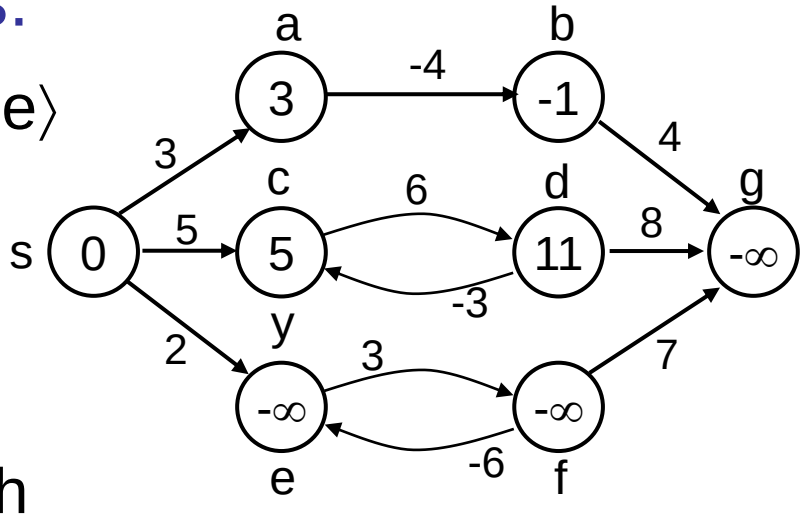


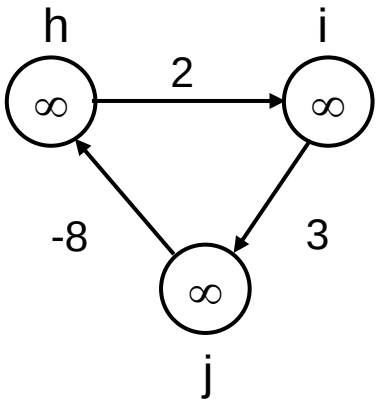
Negative-Weight Edges

- $s \rightarrow e$: 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: $\delta(s, f) = -\infty$,

    $\delta(s, g) = -\infty$





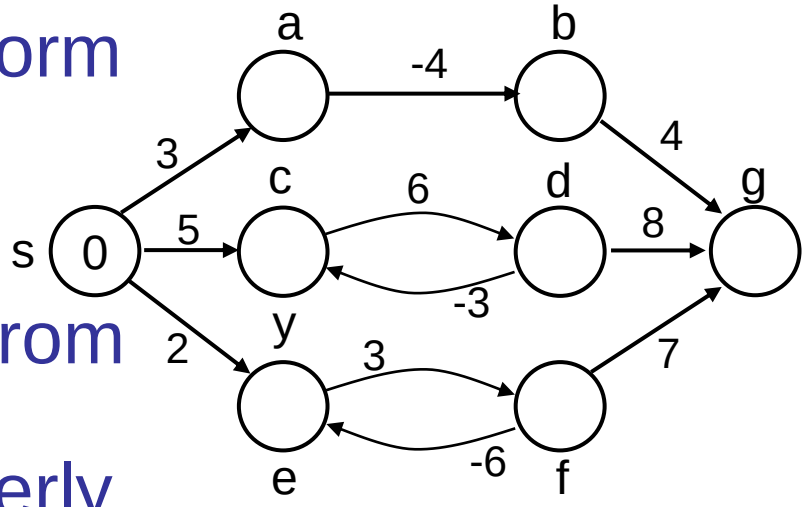
h, i, j not reachable from s

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

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



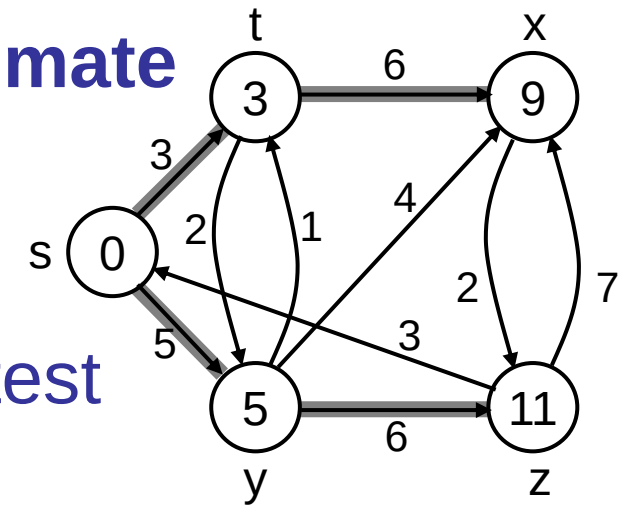
Cycles

- 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, $\pi[v]$ = 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. $\pi[v] \leftarrow$ NIL
4. $d[s] \leftarrow 0$

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

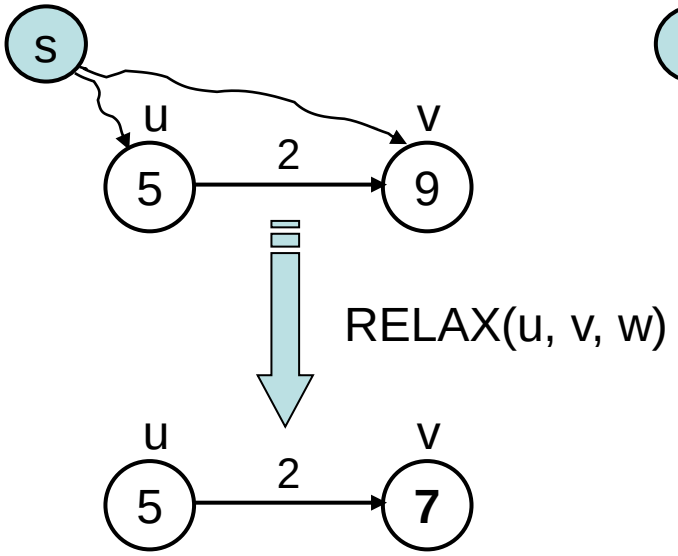
Relaxation

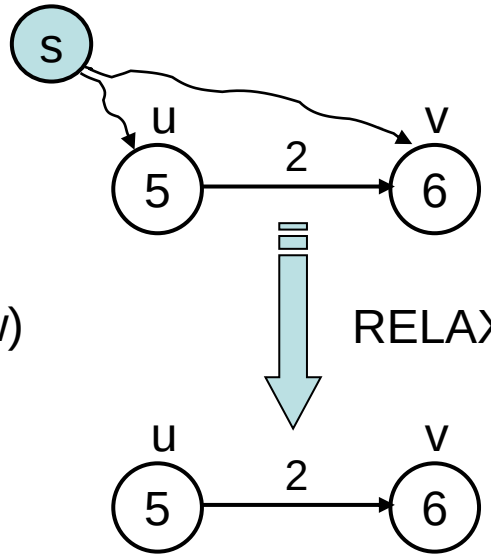
- **Relaxing** an edge (u, v) = testing whether we can improve the shortest path to v found so far by going through u

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

  we can improve the shortest path to v

  $\Rightarrow$ update d[v] and π[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. $\pi[v] \leftarrow u$

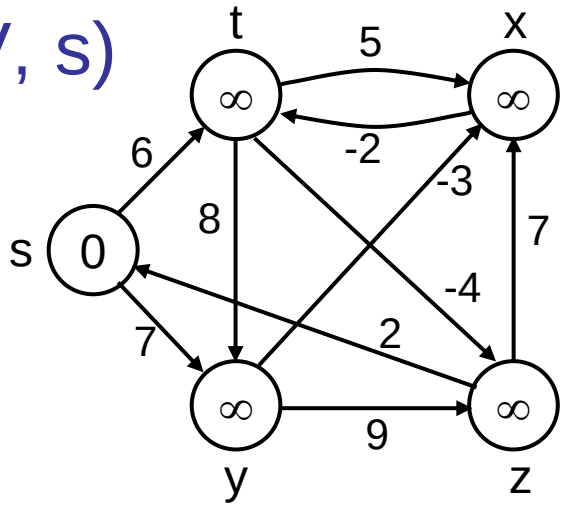
- All the single-source shortest-paths algorithms
  - start by calling INIT-SINGLE-SOURCE
  - then relax edges
- The algorithms differ in the order and how many times they relax each edge

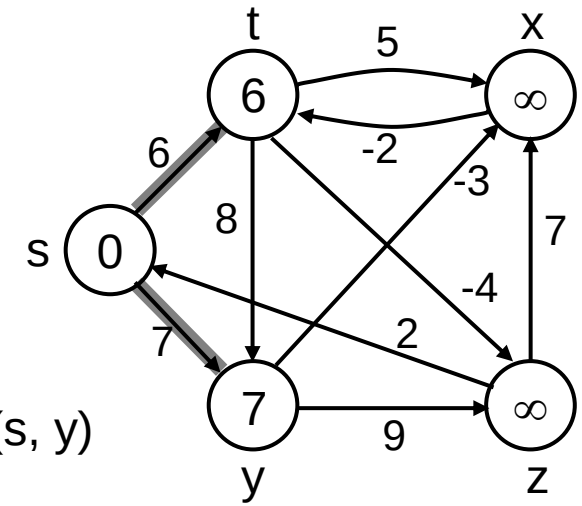
Bellman-Ford Algorithm

- Single-source shortest paths problem
  - Computes d[v] and π[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 ← 1 to |V| - 1
3. **do for** each edge (u, v) ∈ E
4. **do** RELAX(u, v, w)
5. **for** each edge (u, v) ∈ E
6. **do if** d[v] > d[u] + w(u, v)
7. **then return** FALSE
8. **return** TRUE

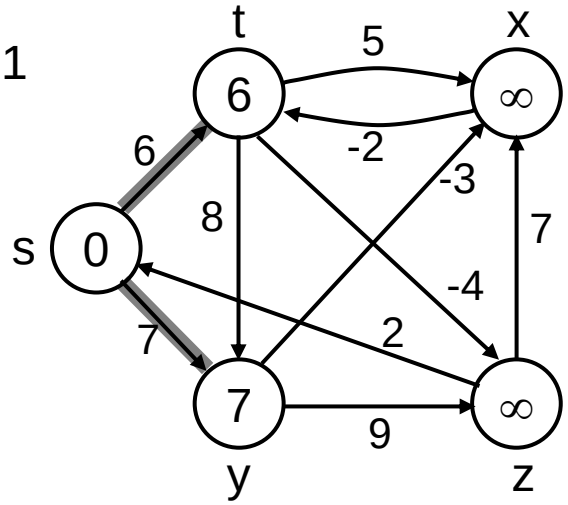




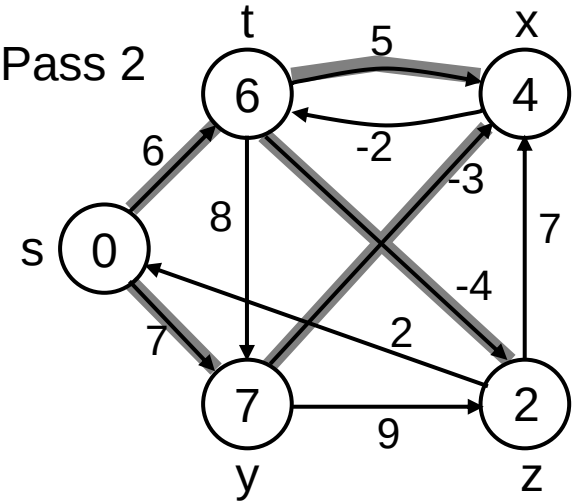
E: (t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)

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

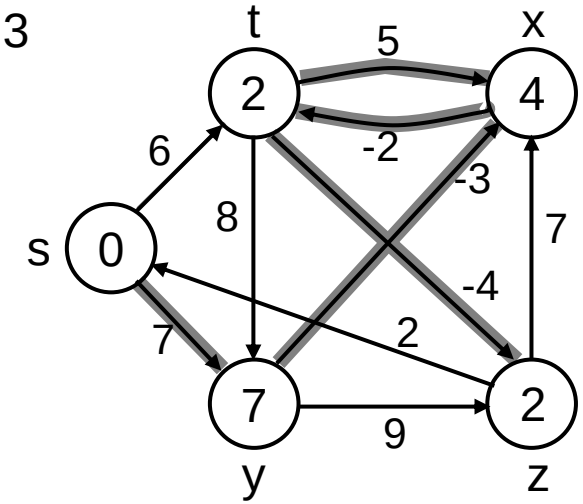
Pass 1



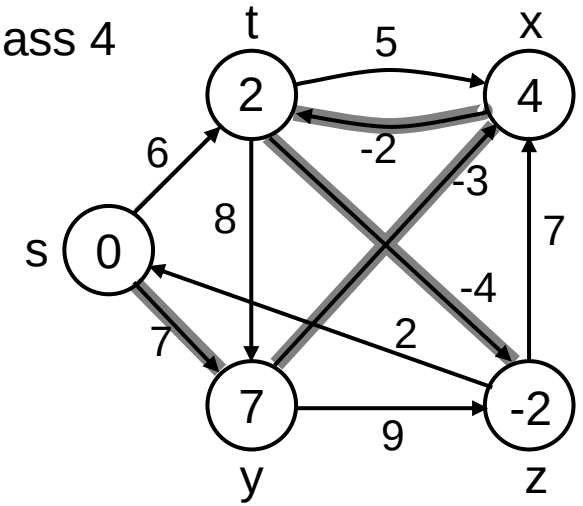
Pass 2



Pass 3

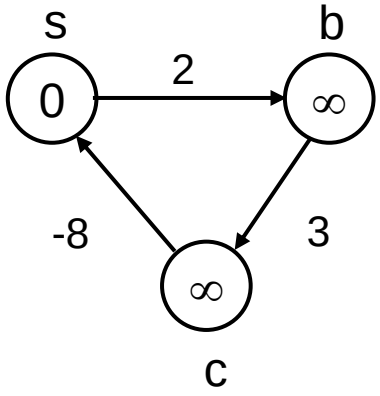


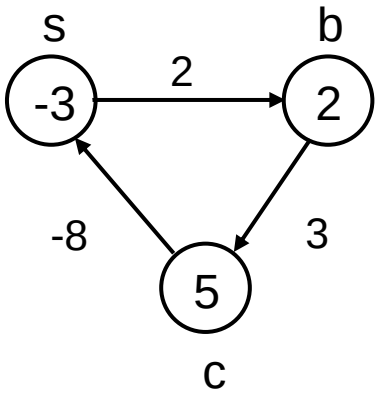
Pass 4

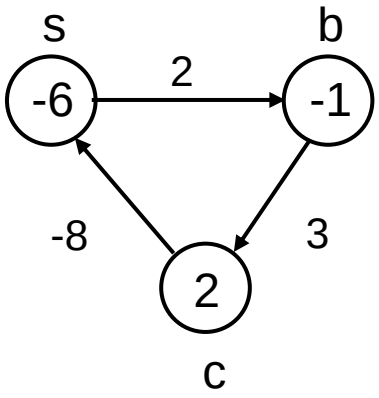


Detecting Negative Cycles

- **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)          ← Θ(V)
2.  for i ← 1 to |V| - 1                    ← O(V)
3.      do for each edge (u, v) ∈ E         ← O(E)      } O(VE)
4.          do RELAX(u, v, w)
5.  for each edge (u, v) ∈ E                ← O(E)
6.      do if d[v] > d[u] + w(u, v)
7.          then return FALSE
8.  return TRUE
```

Running time: O(VE)