Single-Sourced Shortest Paths

- Directed graph $G = (V, E)$
- Weight function $w : E \to \mathbb{R}$
- Weight of path $p = \langle v_0, v_1, \dots v_k \rangle$:

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

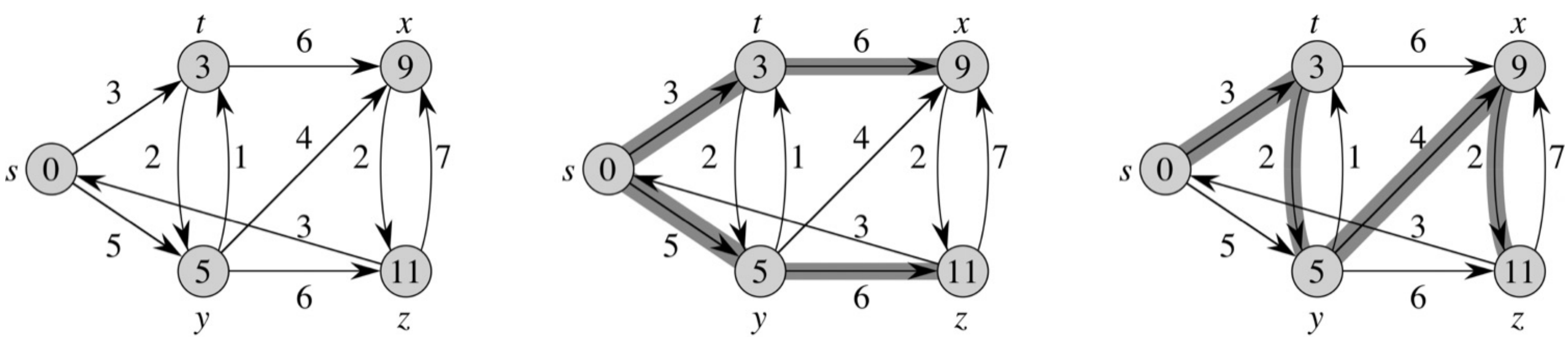
- Shortest path weight $\delta(u, v)$

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

- Shortest path from u to v is any path p such that $w(p) = \delta(u, v)$

- There may be more than one shortest path from s to some $v \in V$

- Looking at all shortest paths from s, they form a tree



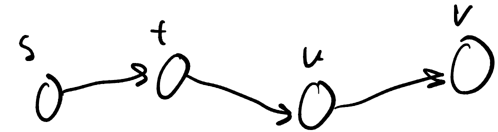
(a) (b) (c)

b + c both show shortest paths from s

- What might the weights represent?
    - Any measure that
        - Accumulates linearly along a path
        - We want to minimize
    - Travel time - edges are roads or airplane routs, etc.
    - Cost
    - Penalties

- Variants of shortest path problems
  - single-source - find shortest paths from a given source $s \in V$ to all vertices $v \in V$
  - Single-destination
  - single-pair - shortest path from u to v
  - all-pairs - find shortest path from u to v for all u, v

- Problem exhibits optimal substructure



- Lemma - any subpath of a shortest path is a shortest path
- Proof idea - If a subpath of a shortest path p was not a shortest path, we could replace it with a shorter subpath to improve p

- Output of a single-source shortest-path algorithm
  - For each vertex $v \in V$, find $v.d = \delta(s, v)$
    - Initialize $v.d = \infty$
    - Reduce $v.d$ as the algorithm progesses
      - maintaining $v.d \geq \delta(s, v)$
    - $v.d$ is a shortest path estimate during execution

- $v.\pi$ = predecessor of v on a shortest path from s

  - Using $v.\pi$ for all v, we can build a shortest path tree

- Initialization for all single-source shortest path algorithms in the book

$$
\begin{array}{l}
\text{INIT-SINGLE-SOURCE}(G, s) \\
\quad \textbf{for } \text{each } v \in G.V \\
\qquad v.d = \infty \\
\qquad v.\pi = \text{NIL} \\
\quad s.d = 0
\end{array}
$$

$\Theta(V)$

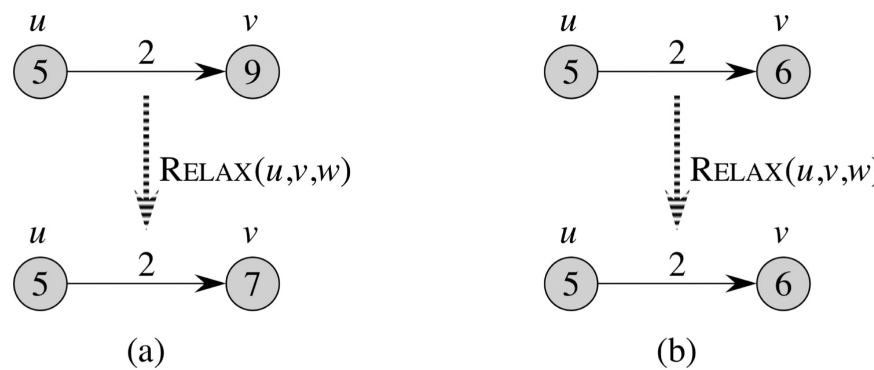
- All algorithms also involve relaxing

RELAX(u, v, w)

if $v.d > u.d + w(u, v)$

$\quad v.d = u.d + w(u, v)$

$\quad v.\pi = u$



$O(1)$

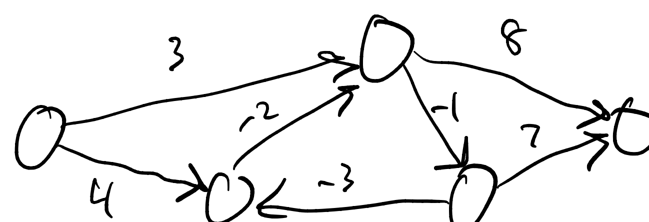
Estimate of $\delta(s, v)$ can be improved using edge (u, v)

(u, v) does not improve the estimate

- All algorithms in the book start with Init-Single-Source, then relax edges

- Some algorithms support negative weights, others do not

    - Never possible to find shortest paths if there is a negative cycle reachable from s



Negative cycles cause path lengths of $-\infty$

- Bellman - Ford Algorithm
    - Allows negative weights
    - Computes $v.d$ and $v.\pi$ for all $v \in V$
    - Returns True if there are no negative cycles, False otherwise
    - Relaxes all edges $|V| - 1$ times

BELLMAN-FORD(G, w, s)

```
INIT-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
```

Each path information propagates throughout the graph

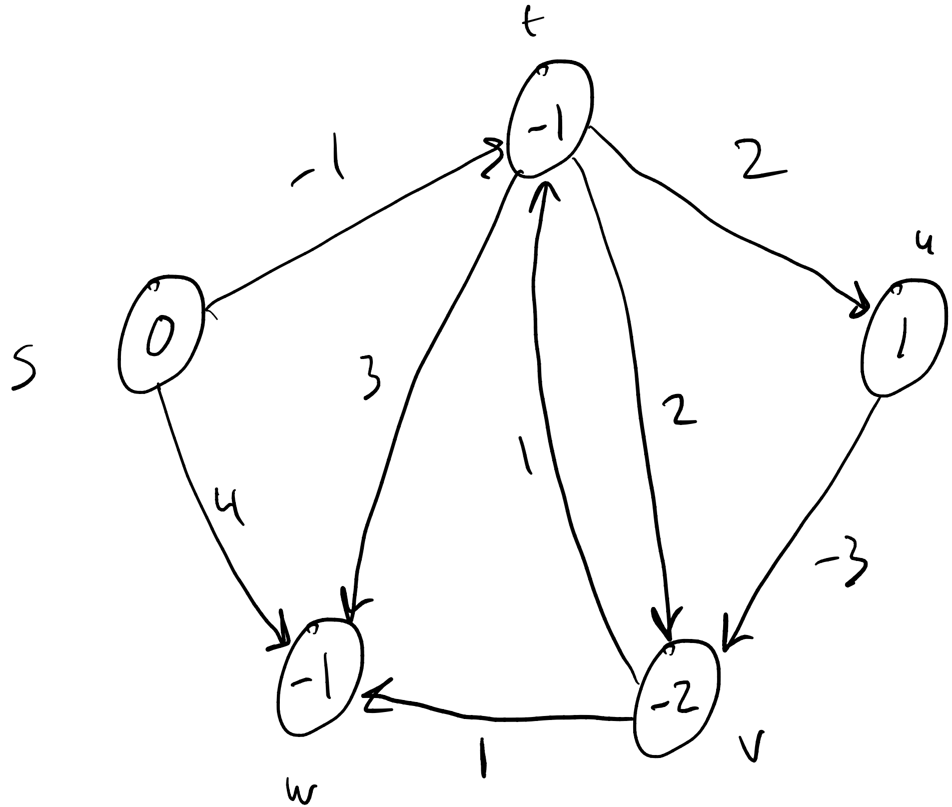
checks for negative cycles

Init-Single-Source $\Theta(V)$

Final loop $\Theta(E)$

Nested loop $\Theta(VE)$

whole thing $\Theta(VE)$



V	π
s	NIL
t	s
u	t
v	u
w	v