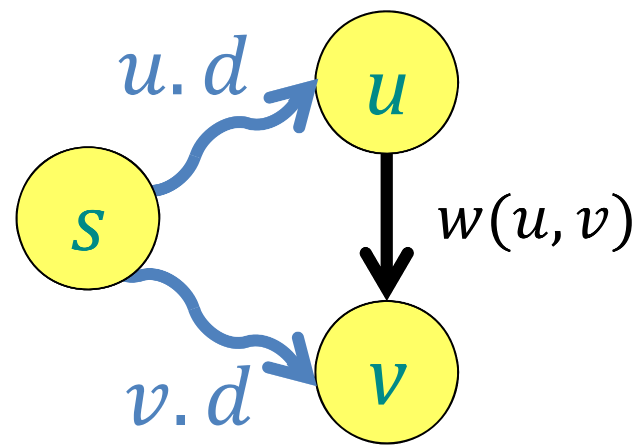
Bellman-Ford Algorithm

- Relaxation algorithm
- "Smart" order of edge relaxations
- Label edges $e_1, e_2, \dots, e_m$
- Relax in this order:

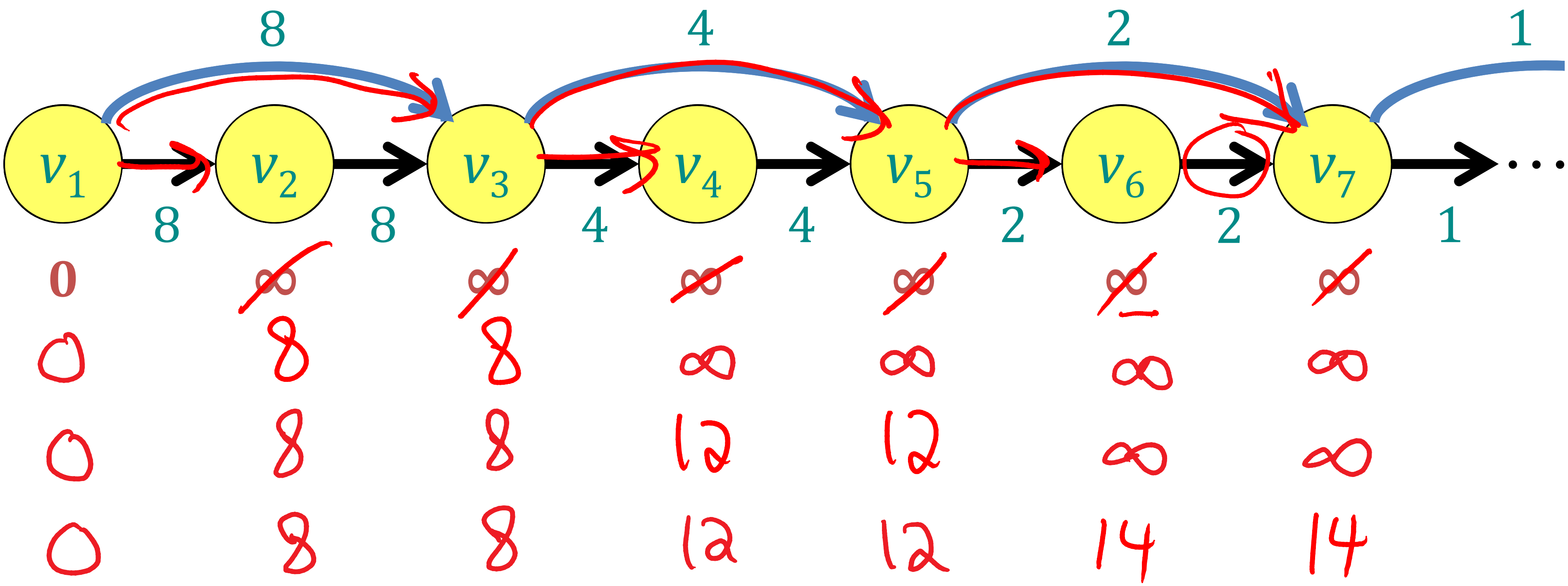
$$\underbrace{\underbrace{e_1, e_2, \dots, e_m}; \underbrace{e_1, e_2, \dots, e_m}; \dots \dots ; \underbrace{e_1, e_2, \dots, e_m}}_{|V| - 1 \text{ repetitions}}$$

Bellman-Ford Algorithm

for v in V:
    $v.d = \infty$
    $v.\pi = \text{None}$
$s.d = 0$
for i from 1 to $|V| - 1$:
    for (u, v) in E:
        relax(u, v):
            if $v.d > u.d + w(u, v)$:
                $v.d = u.d + w(u, v)$
                $v.\pi = u$

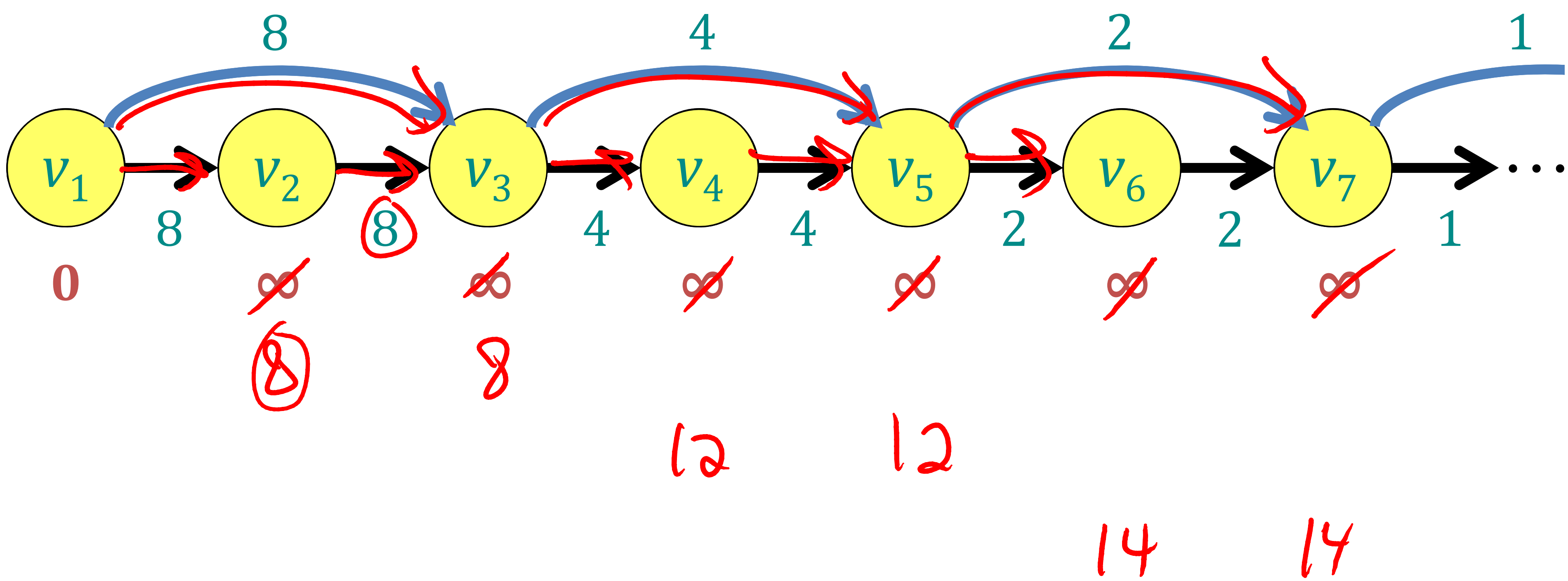


Bellman-Ford Example



edges ordered right to left

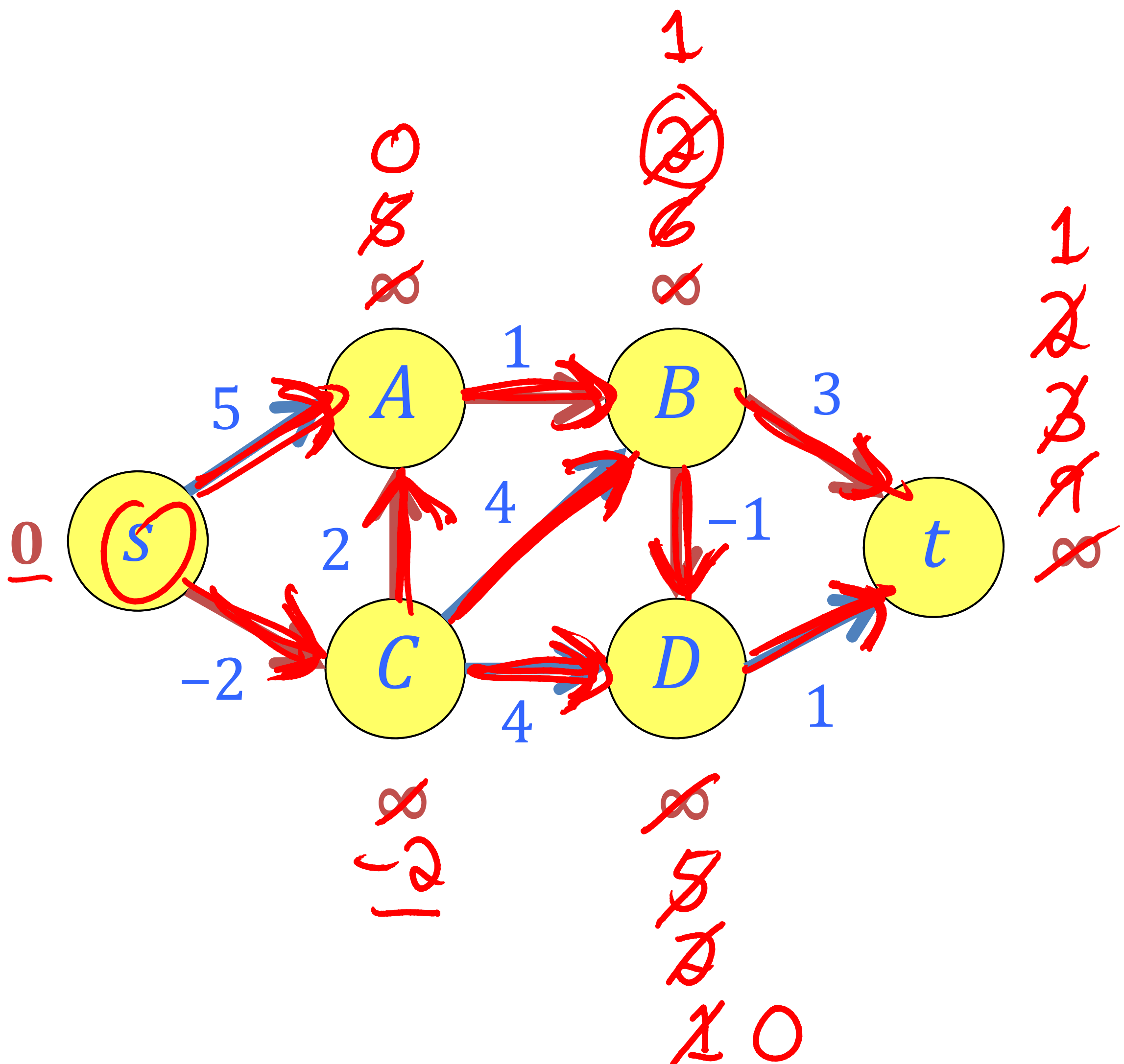
Bellman-Ford Example



one round!

edges ordered left to right

Bellman-Ford Example



edges ordered top down, left to right

Bellman-Ford in Practice

- Distance-vector routing protocol
  - Repeatedly relax edges until convergence
  - Relaxation is local!
- On the Internet:
  - Routing Information Protocol (RIP)
  - Interior Gateway Routing Protocol (IGRP)

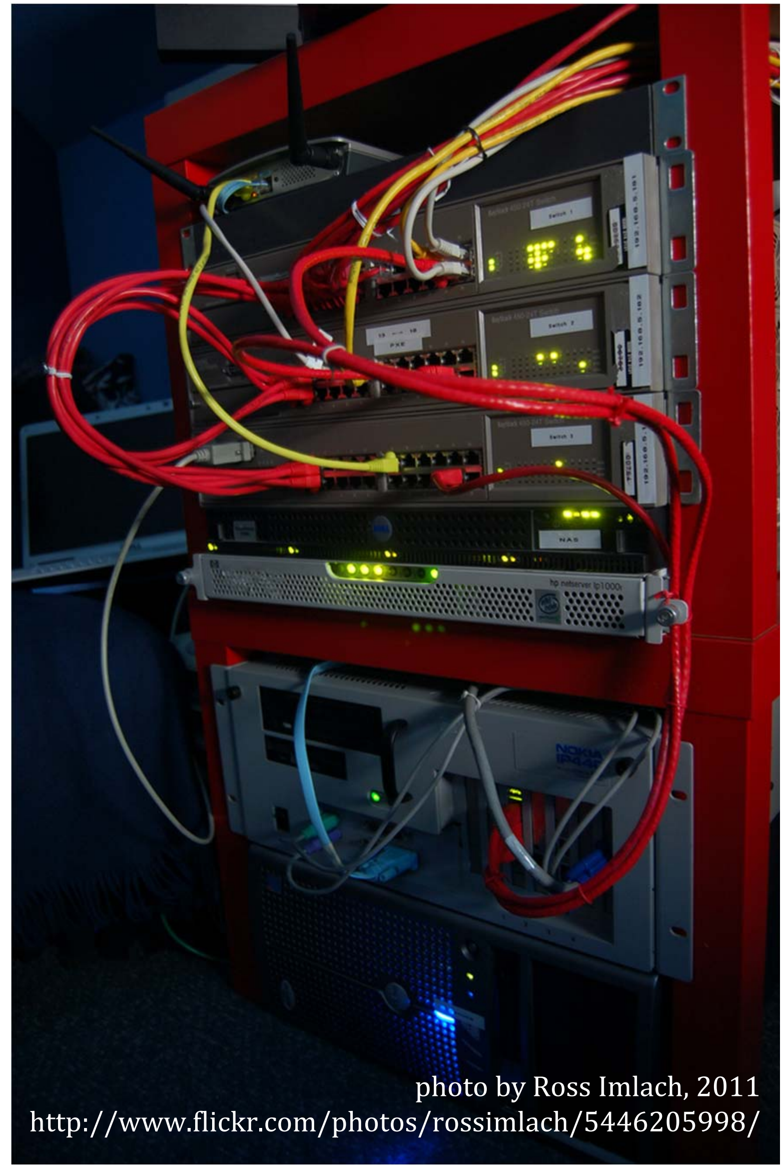
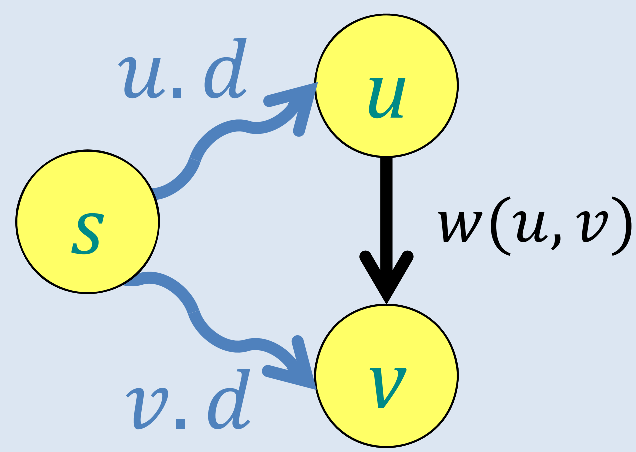


photo by Ross Imlach, 2011
http://www.flickr.com/photos/rossimlach/5446205998/

Bellman-Ford Algorithm with Negative-Weight Cycle Detection

for v in V:
    $v.d = \infty$
    $v.\pi =$ None
$s.d = 0$
for i from 1 to $|V| - 1$:
    for (u, v) in E:
        relax(u, v)
for (u, v) in E:
    if $v.d > u.d + w(u, v)$:
        report that a negative-weight cycle exists



Bellman-Ford Analysis

TOTAL: $O(VE)$

for v in V:
 $v.d = \infty$
 $v.\pi =$ None
$s.d = 0$

$O(V)$

for i from 1 to $|V| - 1$:
 for (u, v) in E:
  relax(u, v) $O(1)$

$O(E)$

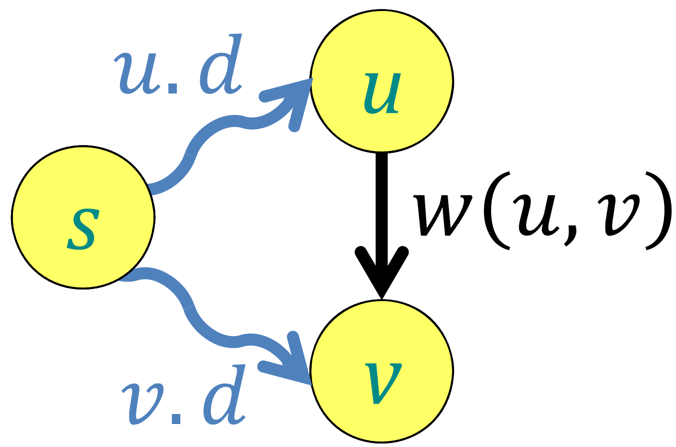
$O(VE)$

for (u, v) in E:
 if $v.d > u.d + w(u, v)$:
  report that a negative-weight cycle exists

$O(E)$

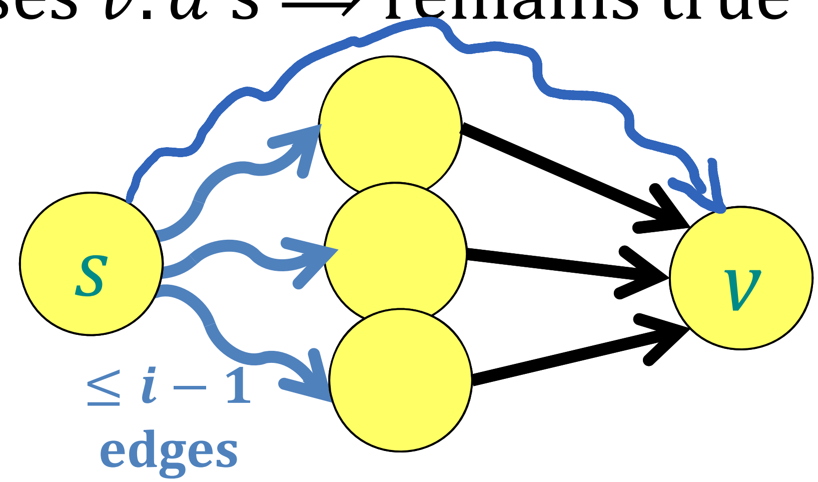
<u>Recall:</u> Relaxing Is Safe

- <u>Lemma:</u> The relaxation algorithm maintains the invariant that $v.d \geq \delta(s,v)$ for all $v \in V$.
- <u>Proof:</u> By induction on the number of steps.
  - Consider relax(u,v)
  - By induction, $u.d \geq \delta(s,u)$
  - By triangle inequality,
  $$\begin{aligned}\delta(s,v) &\leq \delta(s,u) + \delta(u,v)\\ &\leq u.d + w(u,v)\end{aligned}$$
  - So setting $v.d = u.d + w(u,v)$ is "safe" ■



Bellman-Ford Correctness

- Claim: After iteration i of Bellman-Ford, $v.d$ is at most the weight of every path from s to v using at most i edges, for all $v \in V$.
- Proof: By induction on i.
  - Before iteration i, $v.d \leq \min\{w(p) : |p| \leq i-1\}$ #edges in p
  - Relaxation only decreases $v.d$'s $\Longrightarrow$ remains true
  - Iteration i considers all paths with $\leq i$ edges when relaxing v's incoming edges ■



Bellman-Ford Correctness

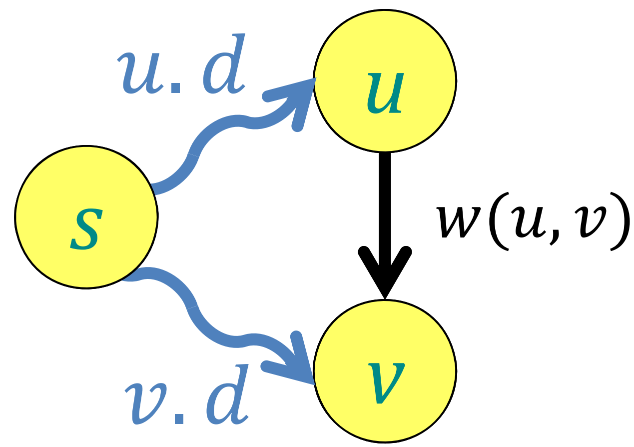
- Theorem: If $G = (V, E, w)$ has no negative-weight cycles, then at the end of Bellman-Ford, $v.d = \delta(s, v)$ for all $v \in V$.
- Proof:
  - Without negative-weight cycles, shortest paths are always simple
  - Every simple path has $\leq |V|$ vertices, so $\leq |V| - 1$ edges
  - Claim $\Longrightarrow |V| - 1$ iterations make $v.d \leq \delta(s, v)$
  - Safety $\Longrightarrow v.d \geq \delta(s, v)$ ∎

Bellman-Ford Correctness

- Theorem: Bellman-Ford correctly reports negative-weight cycles reachable from s.
- Proof:
  - If no negative-weight cycle, then previous theorem implies $v.d = \delta(s, v)$, and by triangle inequality, $\delta(s, v) \leq \delta(s, u) + w(u, v)$, so Bellman-Ford won't incorrectly report a negative-weight cycle.
  - If there's a negative-weight cycle, then one of its edges can always be relaxed (once one of its d values becomes finite), so Bellman-Ford reports. ■

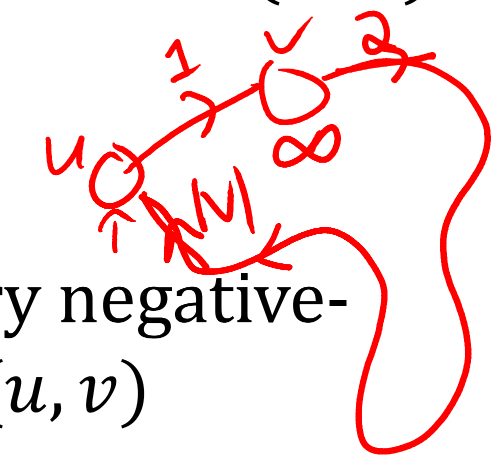
Computing $\delta(s, v)$

for v in V:
    $v.d = \infty$
    $v.\pi = \text{None}$
$s.d = 0$
for i from 1 to $|V| - 1$:
    for (u, v) in E:
        relax(u, v)
for j from 1 to $|V|$:
    for (u, v) in E:
        if $v.d > u.d + w(u, v)$:
            $v.d = -\infty$
            $v.\pi = u$

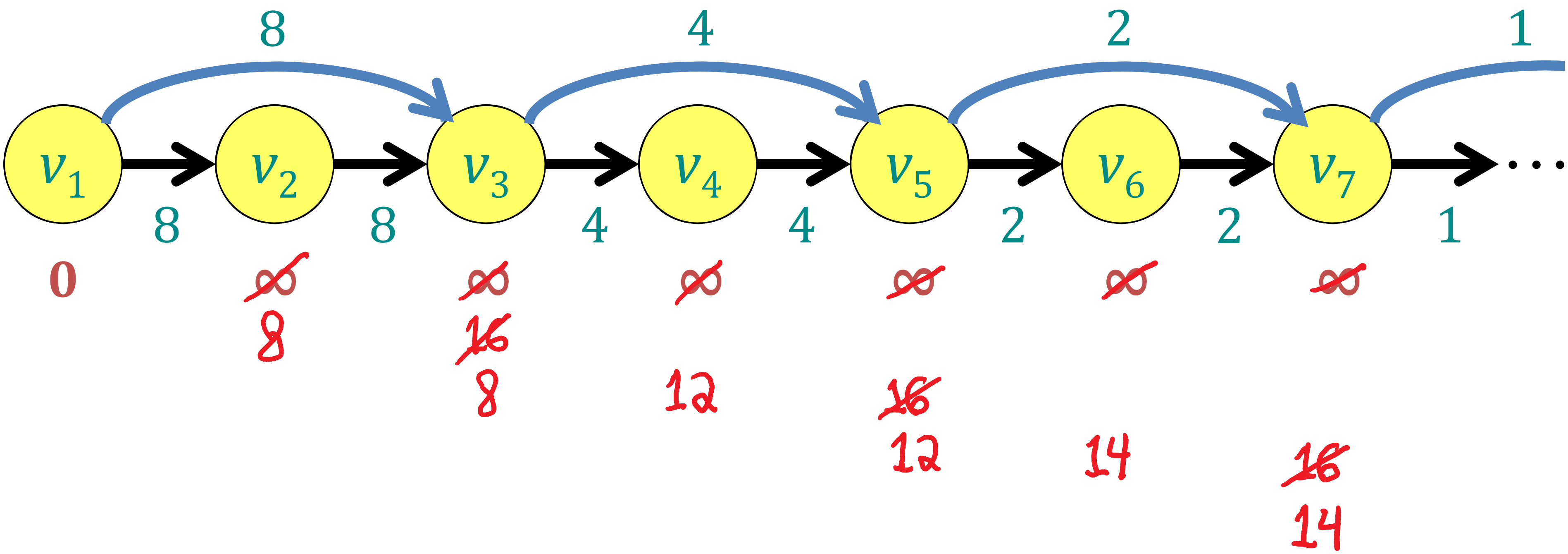


Correctness of $\delta(s, v)$

- Theorem: After the algorithm, $v.d = \delta(s, v)$ for all $v \in V$.
- Proof:
  - As argued before, after i loop, every negative-weight cycle has a relaxable edge (u, v)
  - Setting $v.d = -\infty$ takes limit of relaxation
  - All reachable nodes also have $\delta(s, x) = -\infty$
  - Path from original u to any vertex x (including u) with $\delta(s, x) = -\infty$ has at most $|V|$ edges
  - (So relaxation is impossible after j loop.) ■



Why Did This Work So Well?



- It's a **DAG** (directed acyclic graph)
- We followed a **topological sorted order**

edges ordered left to right