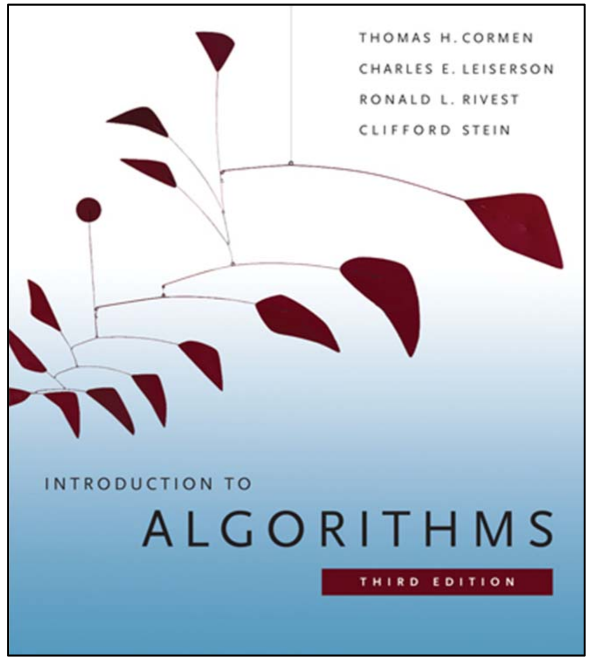
6.006

Introduction to Algorithms



Lecture 15: Shortest Paths II

Prof. Erik Demaine

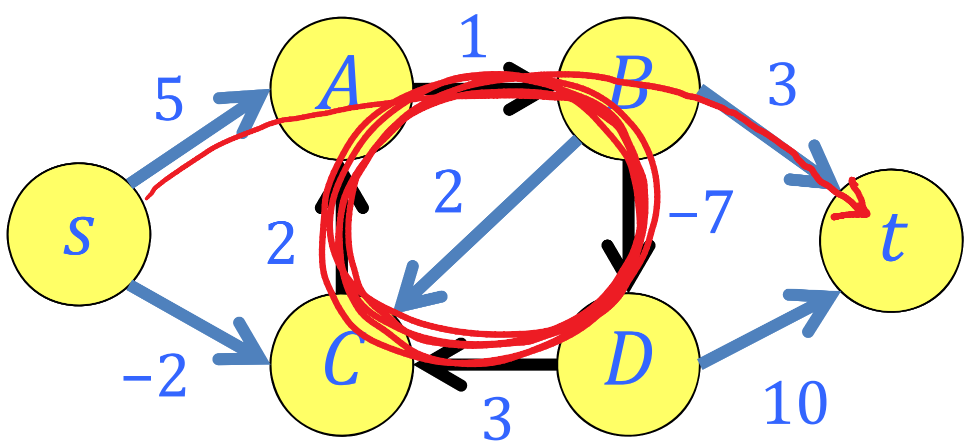
Today

- **Bellman-Ford algorithm**
  for single-source shortest paths
- Running time
- Correctness
- Handling negative-weight cycles
- Directed acyclic graphs

Recall: Shortest Paths

- $\delta(u,v) = \inf\{w(p) : p \text{ is a path from } u \text{ to } v\}$
- $\delta(u,v) = \infty$ if there's no path from u to v
- $\delta(u,v) = -\infty$ if there's a path from u to v that visits a negative-weight cycle

Example:

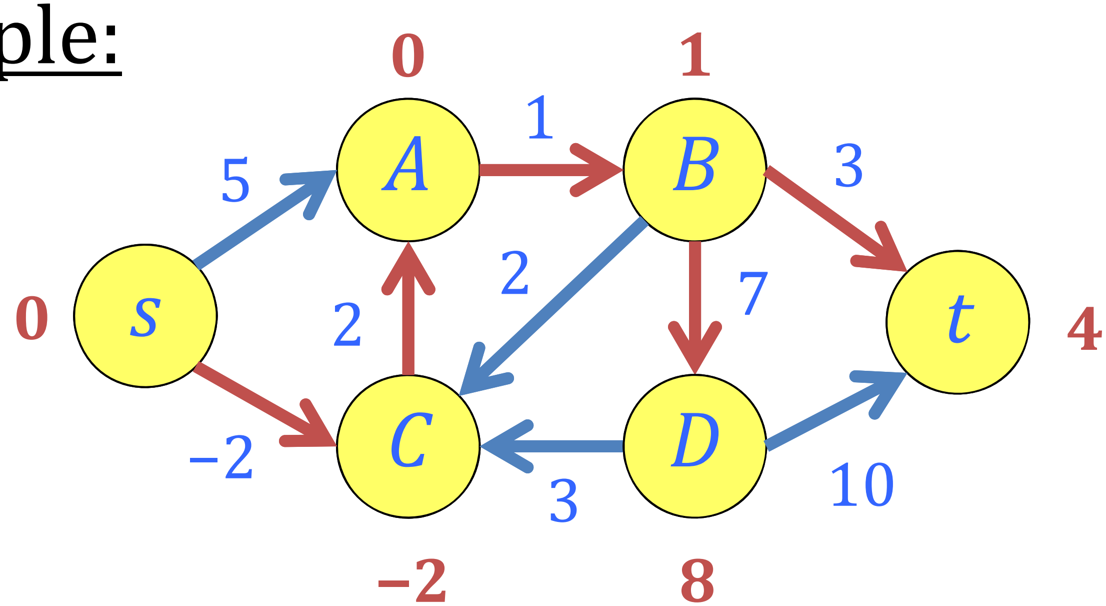


$\delta(s,t) = -\infty$

Recall: Single-Source Shortest Paths

- Problem: Given a directed graph $G = (V, E)$ with edge-weight function $w : E \to \mathbb{R}$, and a ***source*** vertex s, compute $\delta(s, v)$ for all $v \in V$
  - Also want shortest-path tree represented by $v.\pi$

Example:



$t.\pi = B$
$B.\pi = A$
$D.\pi = B$

$\delta(s, v)$

$A.\pi = C$
$C.\pi = s$
$s.\pi =$ None

Recall: Relaxation Algorithm

for v in V:

$\quad v.d = \infty$

$\quad v.\pi = \text{None}$

$s.d = 0$

while some edge (u, v) has $v.d > u.d + w(u, v)$:

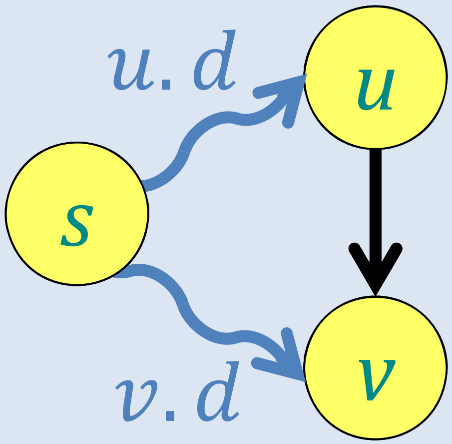
$\quad$ pick such an edge (u, v)

$\quad$ **relax**(u, v):

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

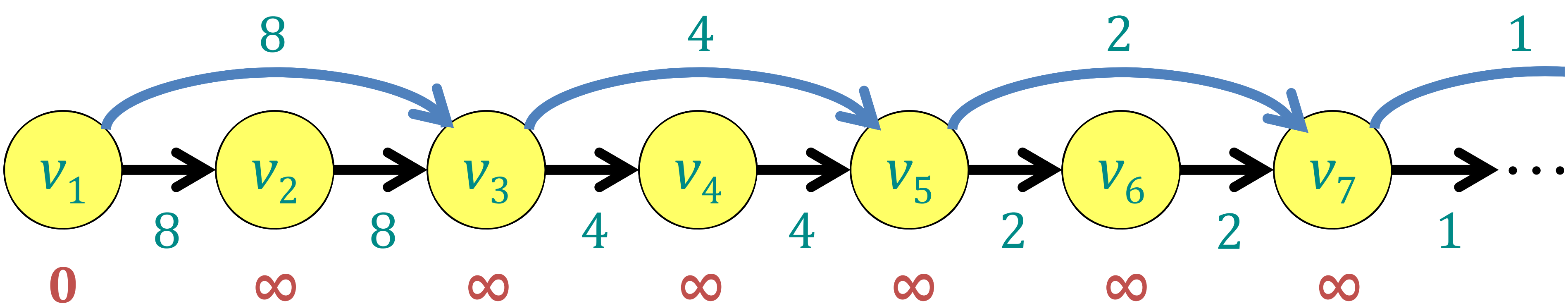
$\quad\quad\quad v.d = u.d + w(u, v)$

$\quad\quad\quad v.\pi = u$

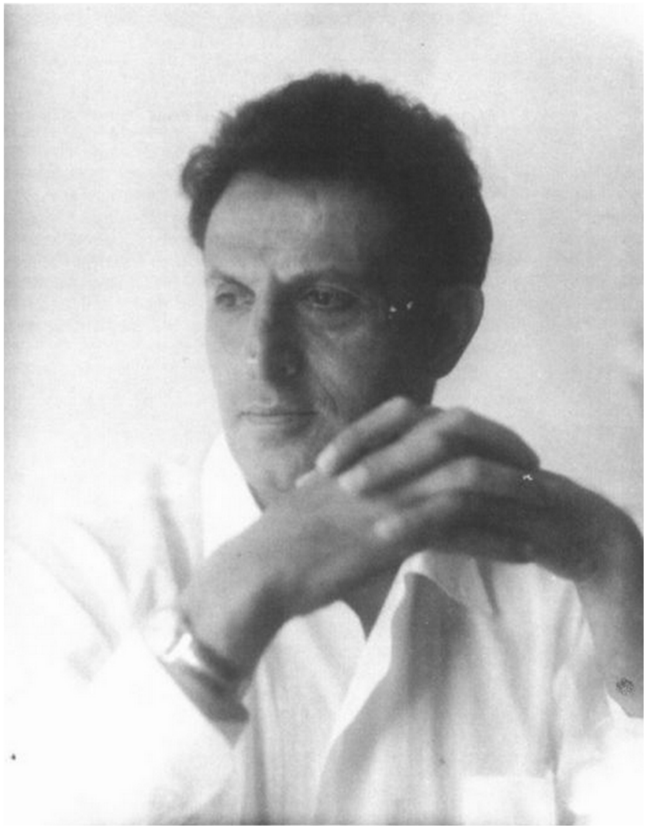


Relaxation Algorithm Issues

- Never stop relaxing in a graph with negative-weight cycles: infinite loop!
- A poor choice of relaxation order can lead to exponentially many relaxations:



Bellman & Ford



Richard E. Bellman
(1920–1984)
IEEE Medal of Honor, 1979

http://www.amazon.com/Bellman-Continuum-Collection-Works-Richard/dp/9971500906

Lester R. Ford, Jr.
(1927–)
president of MAA, 1947–48

http://www.maa.org/aboutmaa/maaapresidents.html

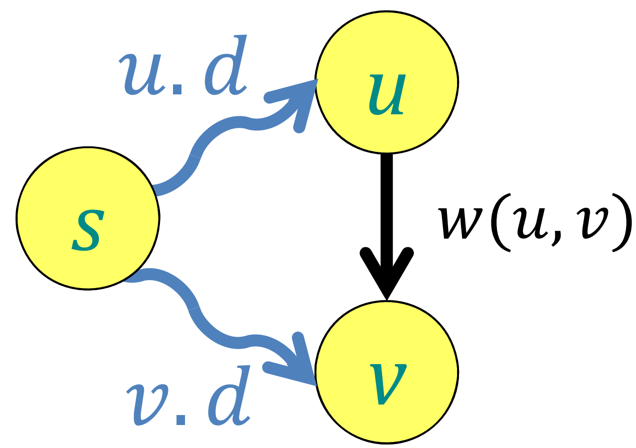
Bellman-Ford Algorithm

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

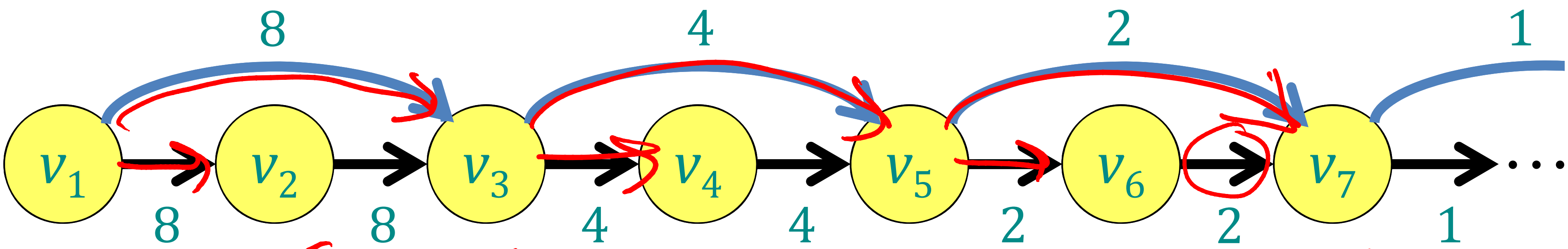
$$\underbrace{\underbrace{e_1, e_2, \dots, e_m}; \underbrace{e_1, e_2, \dots, e_m}; \dots \dots ; \underbrace{e_1, e_2, \dots, e_m}}_{|V| - 1 \text{ repetitions}}$$

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 $v.d > u.d + w(u, v)$:
                $v.d = u.d + w(u, v)$
                $v.\pi = u$



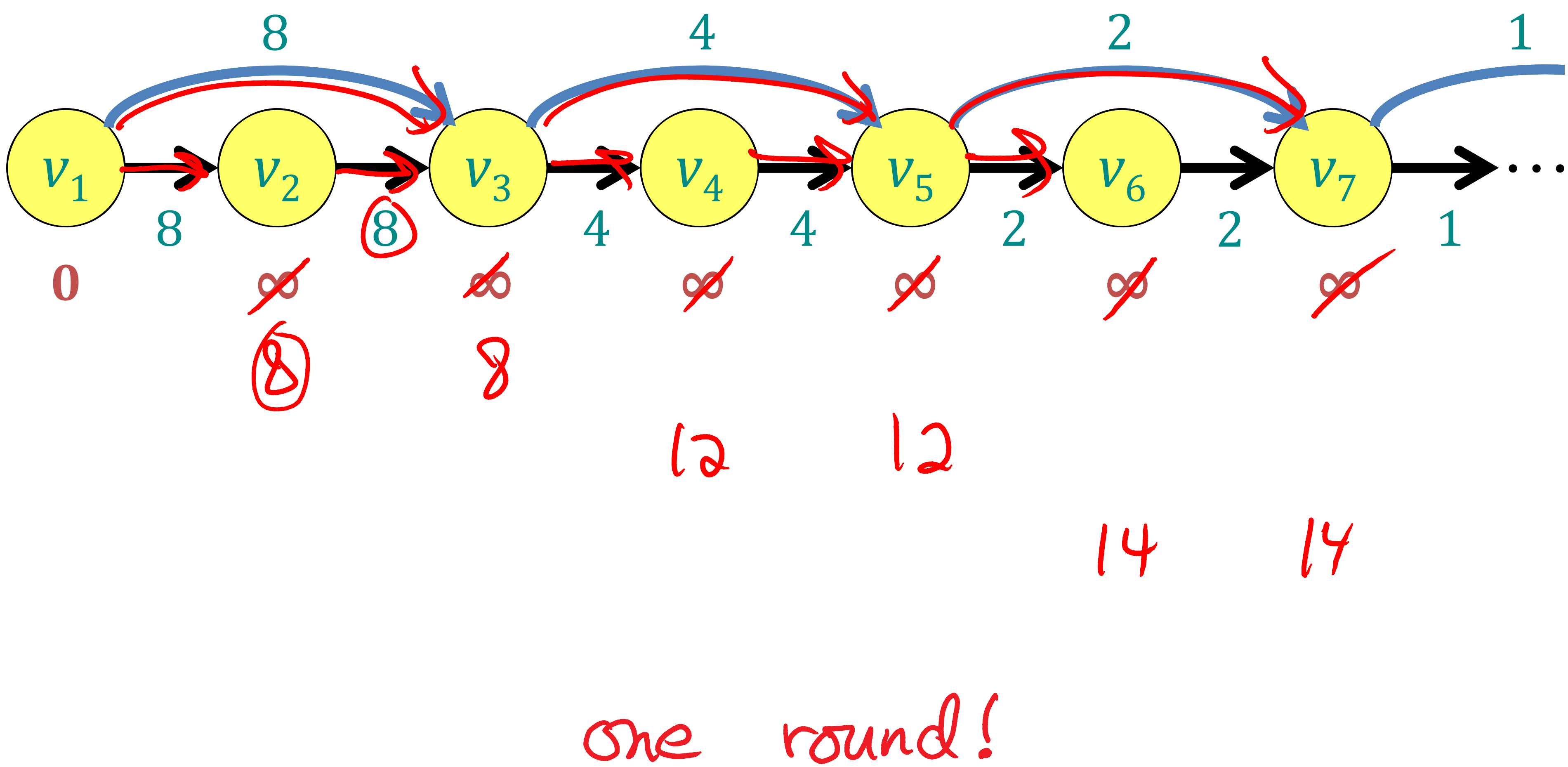
Bellman-Ford Example



0	$\cancel{\infty}$	$\cancel{\infty}$	$\cancel{\infty}$	$\cancel{\infty}$	$\cancel{\infty}$	$\cancel{\infty}$
0	8	8	∞	∞	∞	∞
0	8	8	12	12	∞	∞
0	8	8	12	12	14	14

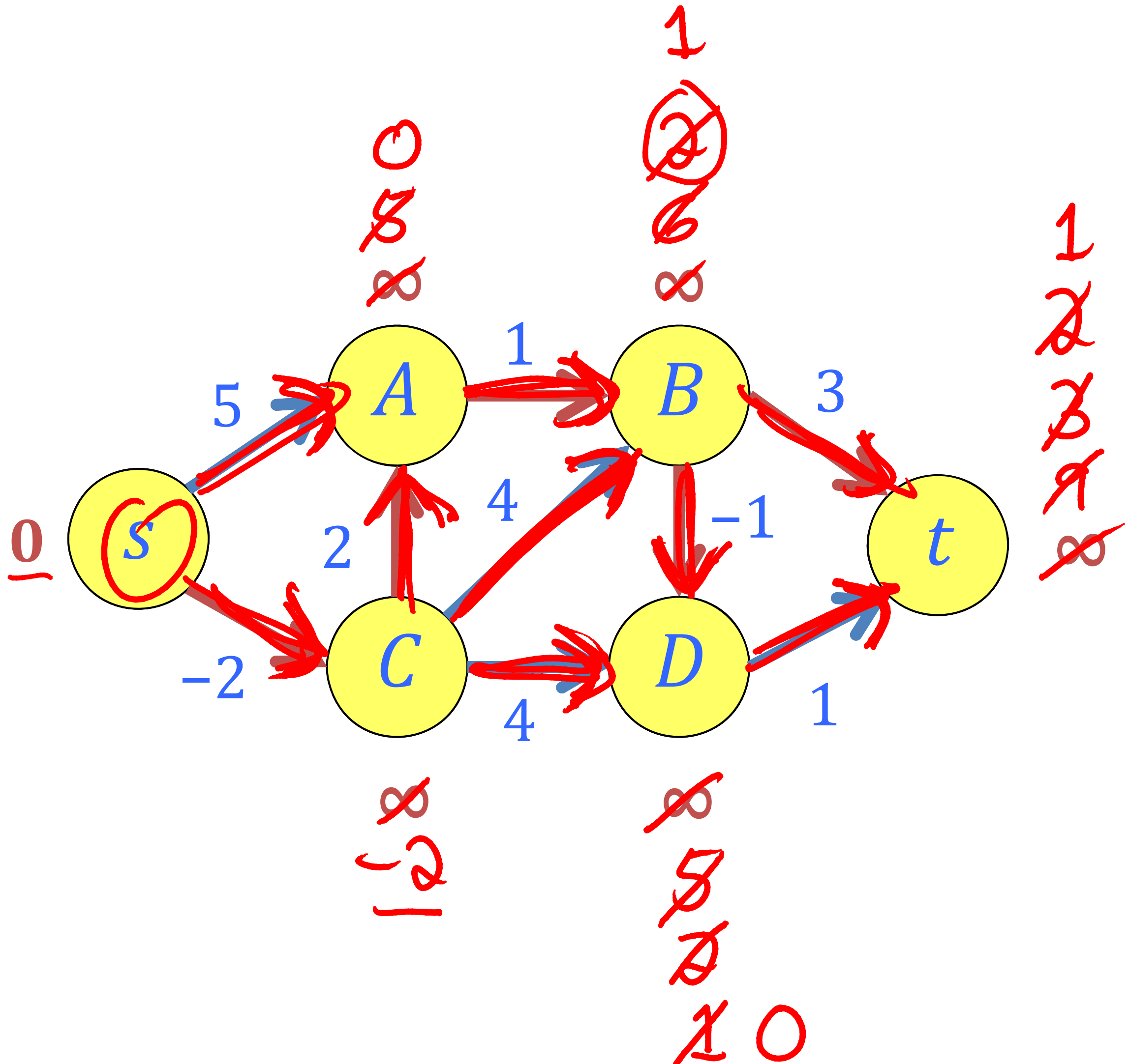
edges ordered right to left

Bellman-Ford Example



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)

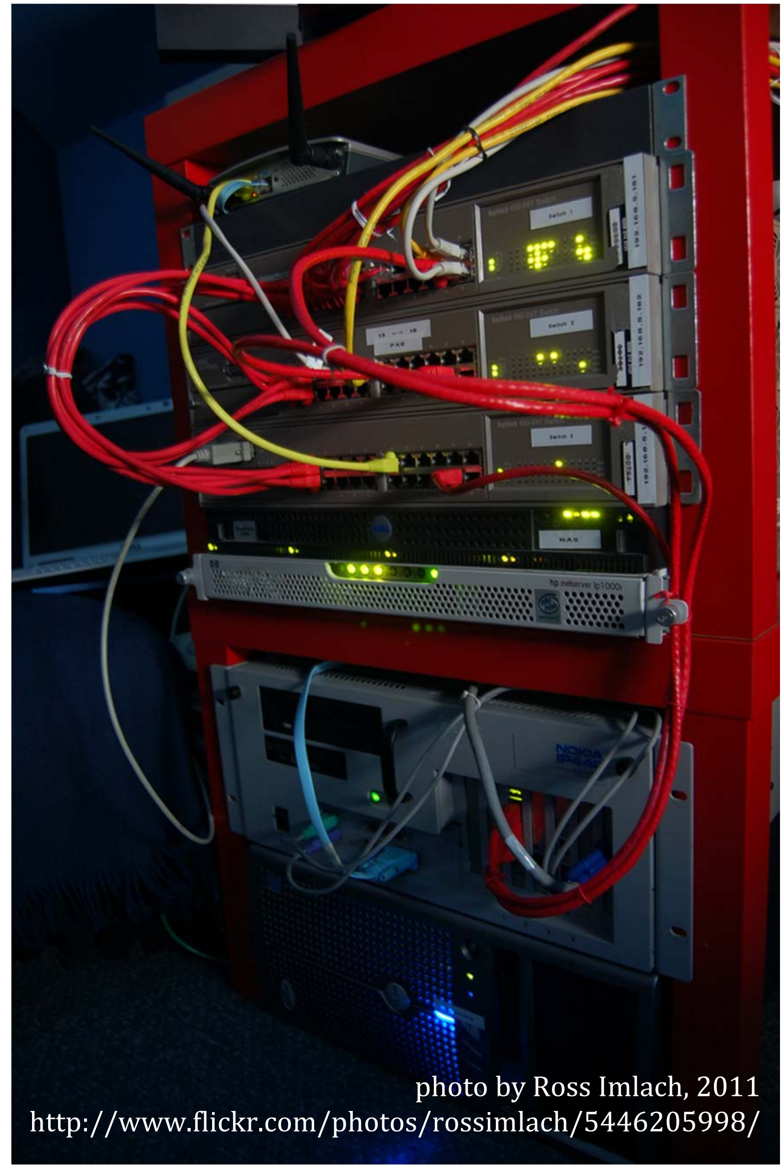
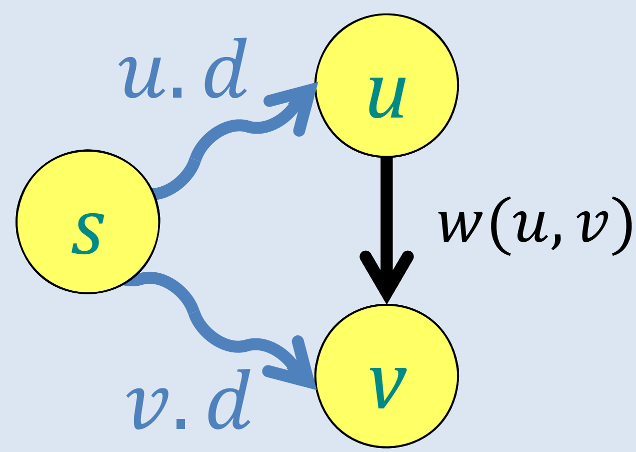


photo by Ross Imlach, 2011
http://www.flickr.com/photos/rossimlach/5446205998/

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: $O(VE)$

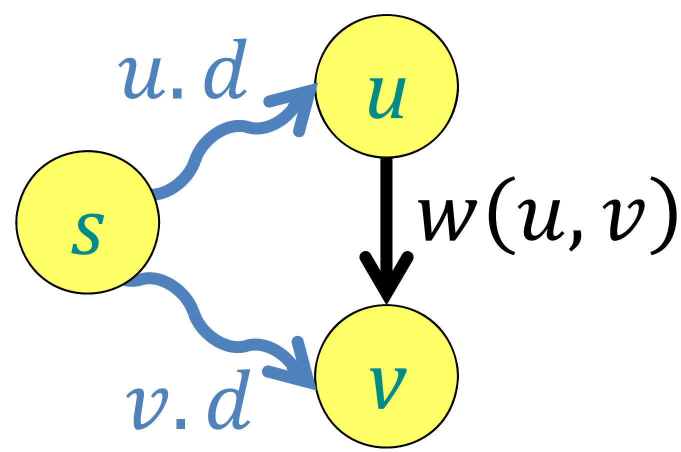
for v in V:
    $v.d = \infty$
    $v.\pi =$ None
$s.d = 0$
$O(V)$

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
$O(E)$

Recall: Relaxing Is Safe

- Lemma: The relaxation algorithm maintains the invariant that $v.d \geq \delta(s,v)$ for all $v \in V$.
- Proof: 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" ■

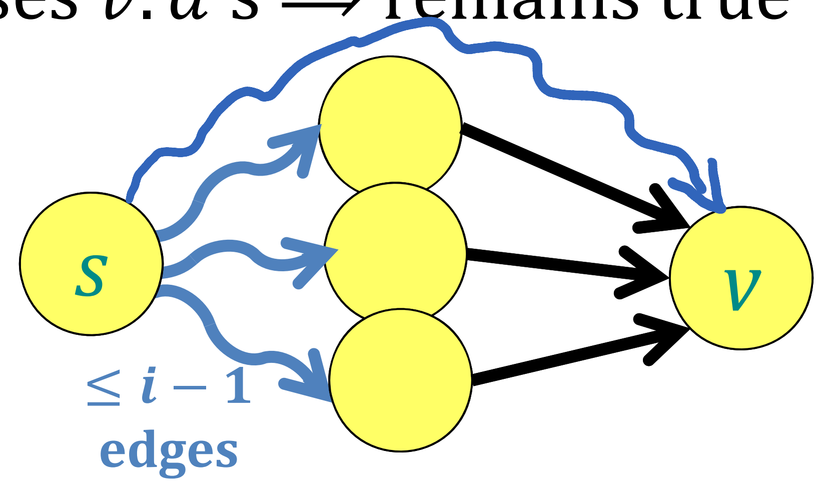


Bellman-Ford Correctness

- 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.

  #edges in p

  - Before iteration i, $v.d \leq \min\{w(p) : |p| \leq i - 1\}$
  - 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

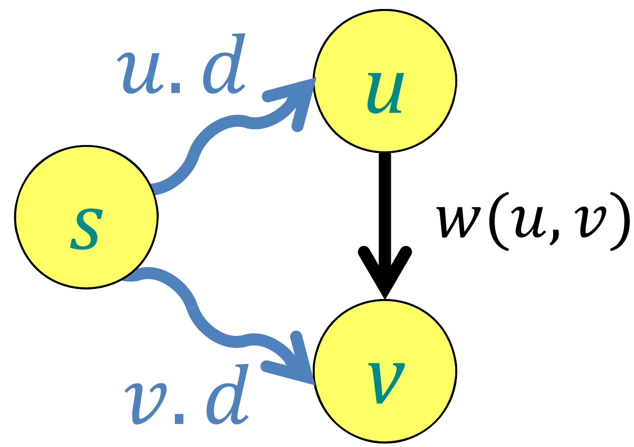
- Theorem: If $G = (V, E, w)$ has no negative-weight cycles, then at the end of Bellman-Ford, $v.d = \delta(s, v)$ for all $v \in V$.
- Proof:
  - Without negative-weight cycles, shortest paths are always simple
  - Every simple path has $\leq |V|$ vertices, so $\leq |V| - 1$ edges
  - Claim $\Longrightarrow$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

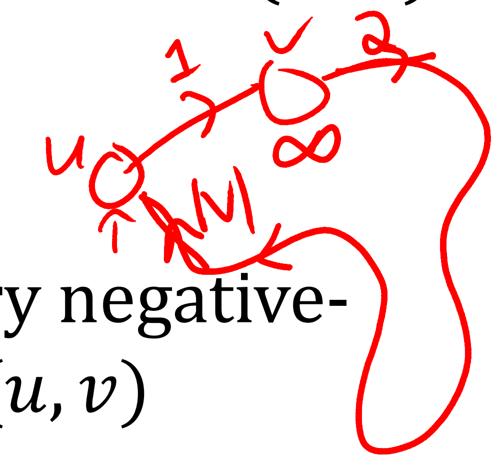
Computing $\delta(s, v)$

```
for v in V:
    v.d = ∞
    v.π = 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 = −∞
            v.π = u
```



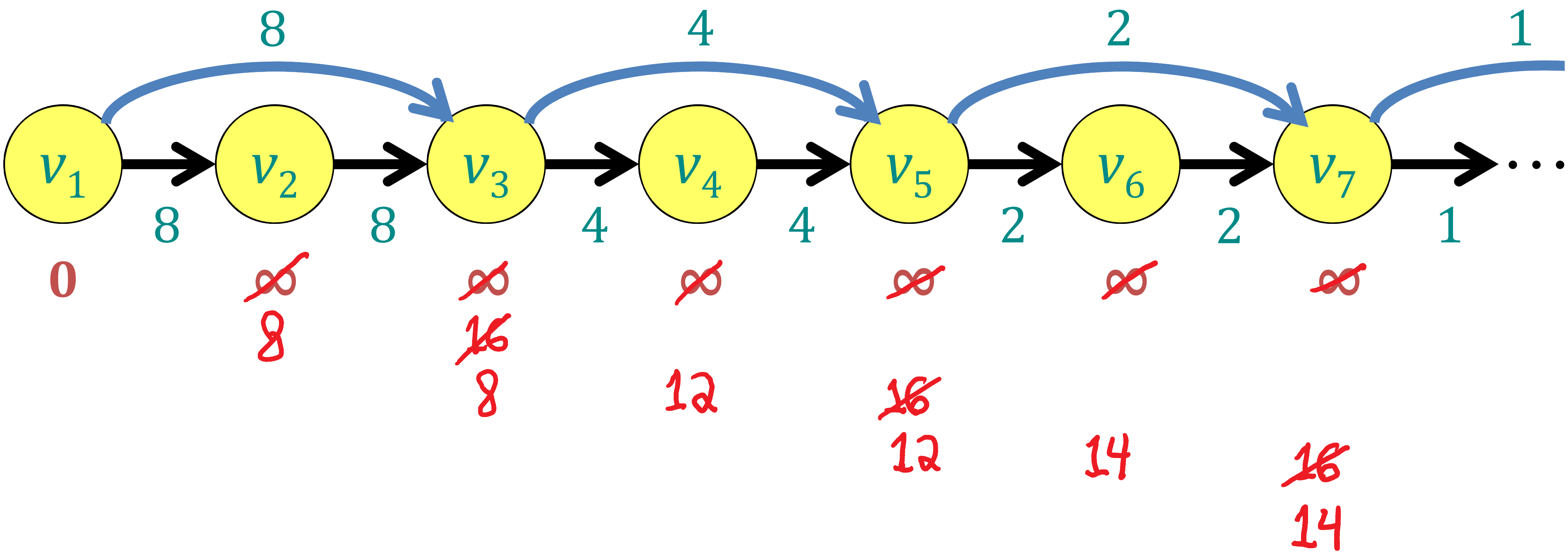
Correctness of $\boldsymbol{\delta(s, v)}$

- 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

Shortest Paths in a DAG

- Simplified Bellman-Ford: no iteration, no cycles

for v in V:
  $v.d = \infty$
  $v.\pi =$ None
$s.d = 0$
topologically sort the vertices V
$\#\ now\ (u, v) \in E \Longrightarrow \text{rank}(u) < \text{rank}(v)\ in\ V$
for u in V: (in order)
  for v in u.neighbors:
    relax(u, v)

O(V)

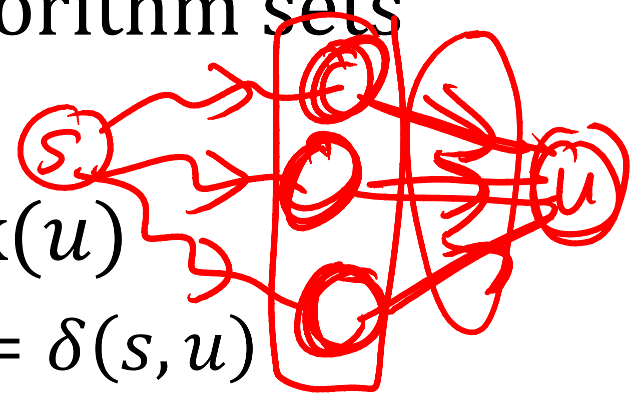
O(V+E)

O(E)

O(1)

Correctness in DAG

- <u>Theorem:</u> In a DAG, this algorithm sets $u.d = \delta(s, u)$ for all $u \in V$.

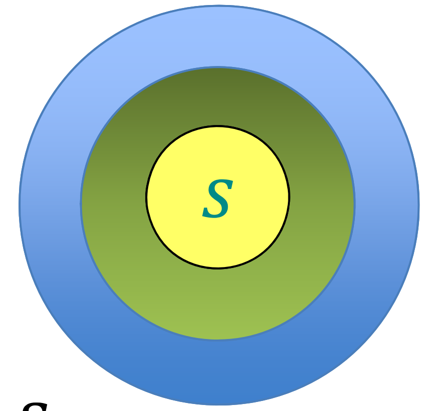


- <u>Proof:</u> By induction on rank(u)
  - Claim by induction that $u.d = \delta(s, u)$ when we hit u in outer loop
  - Base case: $s.d = 0$ correct (no cycles)

other base cases have $d = \infty$

  - When we hit u, we've already hit all previous vertices, including all vertices with edges into u
  - By induction, these vertices had correct d values when we relaxed the edges into u ■

Next Up



- **Dijkstra's algorithm**
  - Relax edges in a growing ball around s
  - Fast: nearly linear time
  - Only one pass through edges, but need logarithmic time to pick next edge to relax
  - Doesn't work with negative edge weights