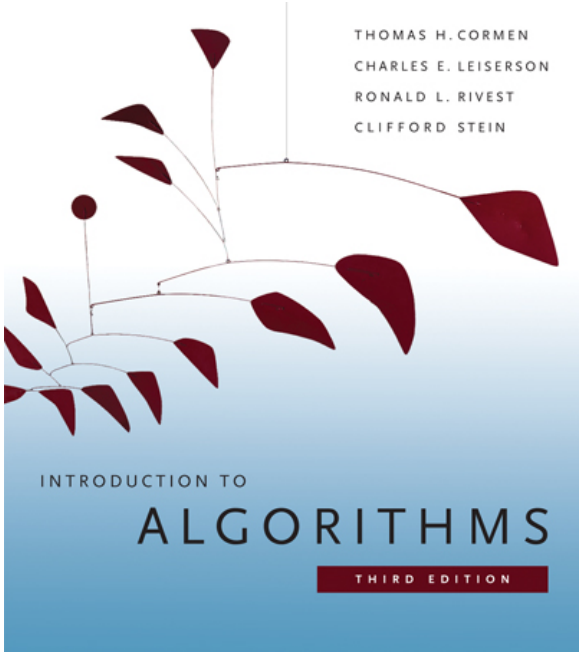
6.006- *Introduction to Algorithms*



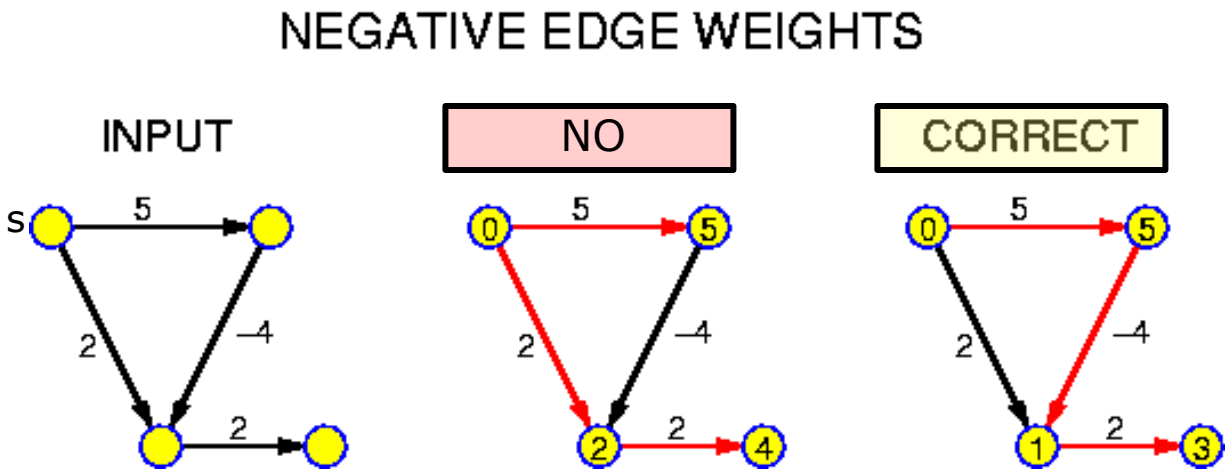
Lecture 15

Prof. Patrick Jaillet

Lecture overview

Shortest paths II

- Review, definition, properties, generic algorithm
- Bellman-Ford algorithm for the case with negative weights (CLRS 24.1)



Single source shortest path problem

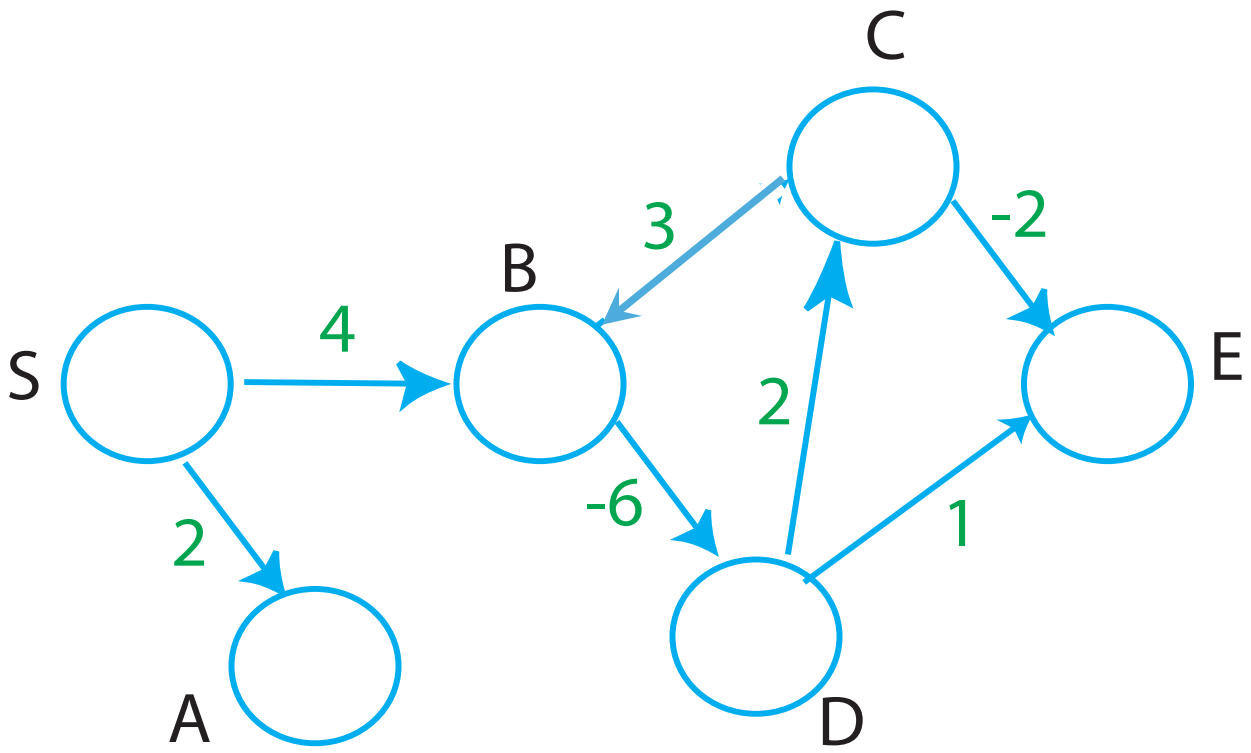
Problem: Given a directed graph $G = (V, E)$ with edge-weight function $w : E \rightarrow \mathsf{R}$, and a node s, find the shortest-path weight $\delta(s, v)$ (and a corresponding shortest path) from s to each v in V.

Special cases:

1. if no path from s to v exists: $\delta(s, v) = \infty$
2. if negative weight cycles in G => problem is not well defined.

Negative cycles ...

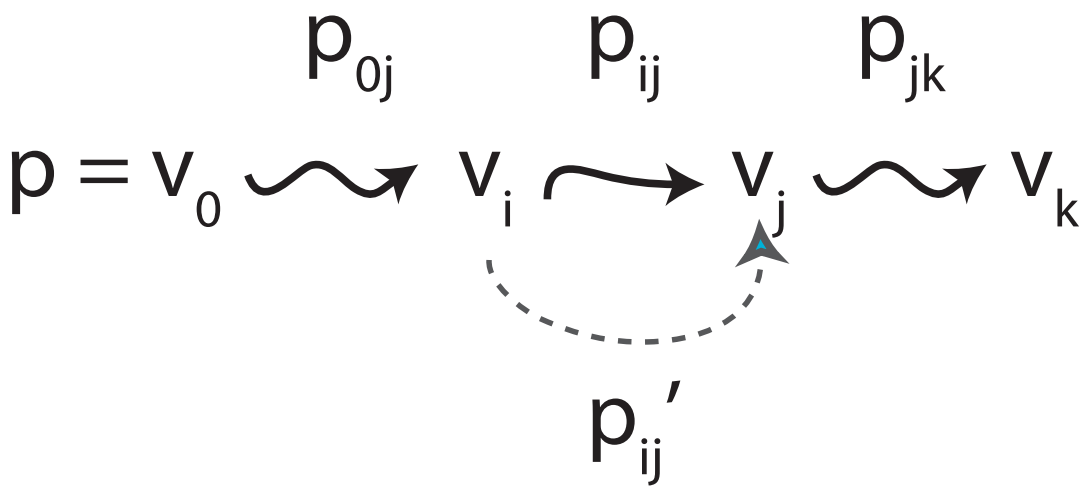
with such cycles => problem is not well defined.



Optimal substructure

Theorem. A subpath of a shortest path is a shortest path.

Proof. By contradiction ...

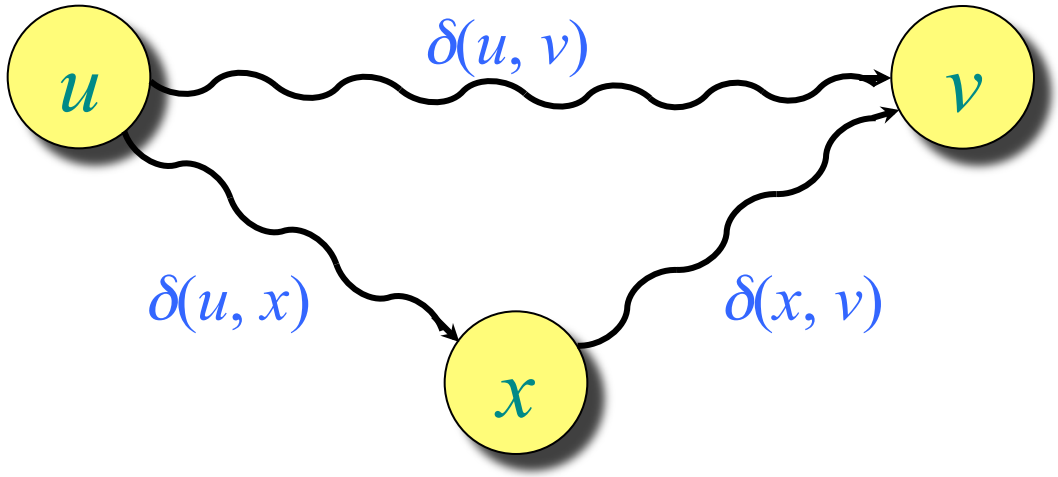


Triangle inequality

Theorem. For all $u, v, x \in V$, we have

$$\delta(u, v) \leq \delta(u, x) + \delta(x, v).$$

Proof.



Generic algorithm, data structures

- $d[v]$ = length of best path from s to v so far
- initialization $d[s] = 0$; $d[v] = \infty$ otherwise
- idea: iteratively decrease $d[v]$ (while maintaining $d[v] \geq \delta(s, v)$)

- $\pi[v]$ = predecessor of v on a best path so far
- initialization $\pi[s] = s$; $\pi[v] = \text{nil}$ otherwise

A generic algorithm

$d[s] \leftarrow 0$
$\pi[s] \leftarrow$ s
for each $v \in V - \{s\}$
  do $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow$ nil

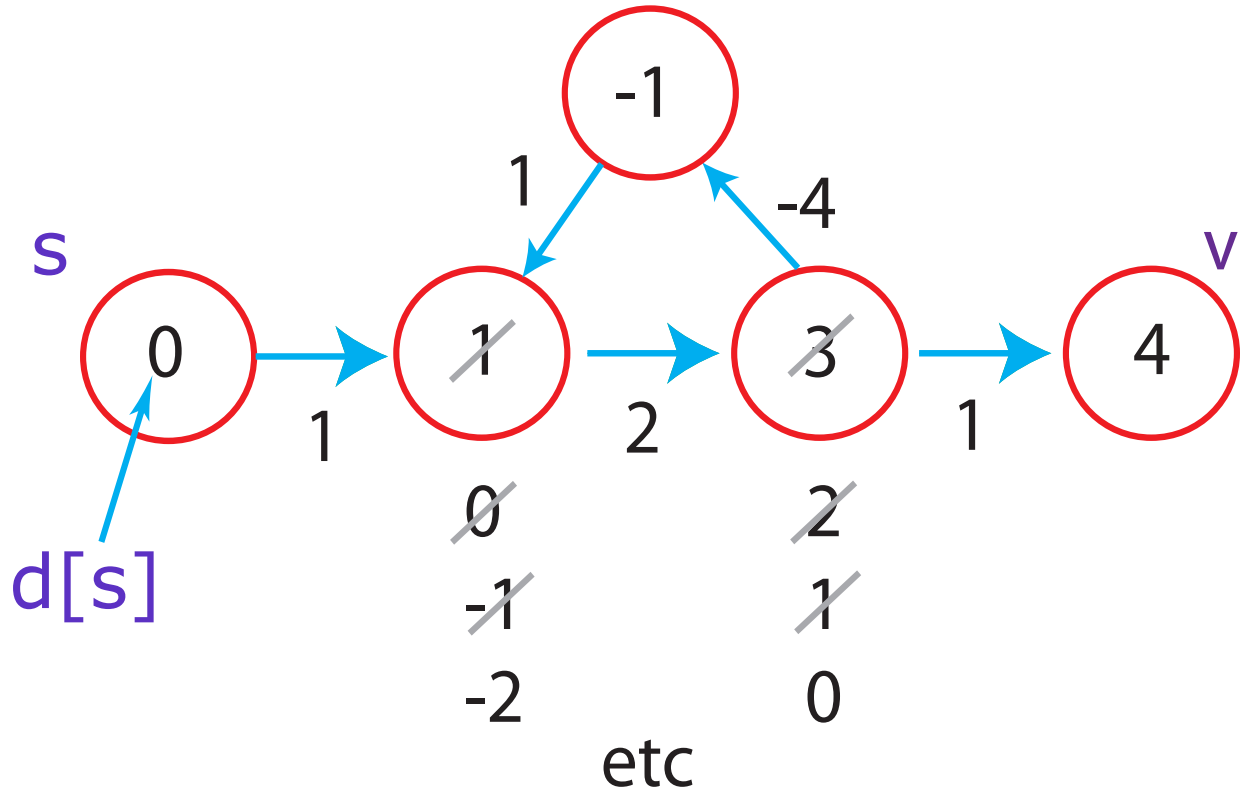
initialization

while there is an edge $(u, v) \in E$ s. t.
    $d[v] > d[u] + w(u, v)$ **do**
  select one such edge "somehow"
  set $d[v] \leftarrow d[u] + w(u, v)$
    $\pi[v] \leftarrow$ u
endwhile

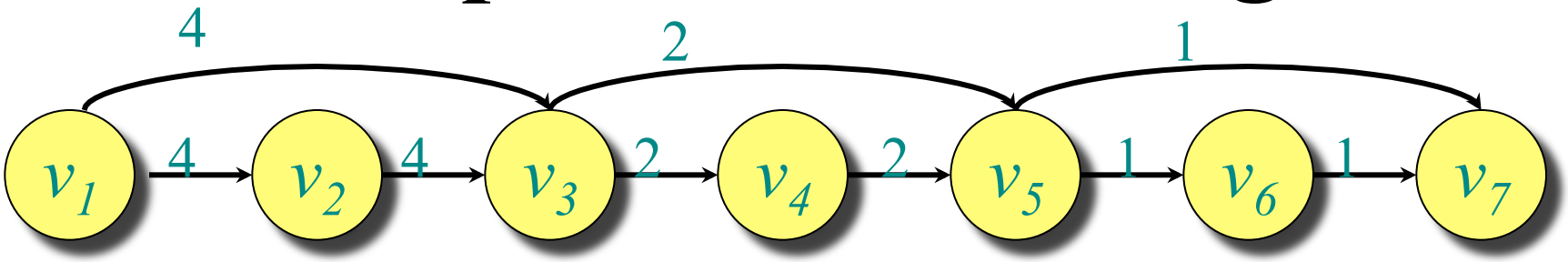
relaxation steps

(... about the "somehow" step ...)

... doesn't stop when negative cycles ...



... bad choice of "somehow" can lead to exponential running time ...



4	8	10	12	13	14
relax (v_1, v_2)					13
relax (v_2, v_3)	relax (v_1, v_3)		10	11	12
relax (v_3, v_4)	relax (v_3, v_4)				11
….					
relax (v_6, v_7)	4	6	8	9	10
relax (v_5, v_7)					9
relax (v_3, v_5)			6	7	8
					7
relax (v_5, v_6)	relax (v_6, v_7)	relax (v_5, v_7)			

Bellman-Ford algorithm

$d[s] \leftarrow 0$
$\pi[s] \leftarrow$ s
for each $v \in V - \{s\}$
  do $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow$ nil

initialization $O(n)$

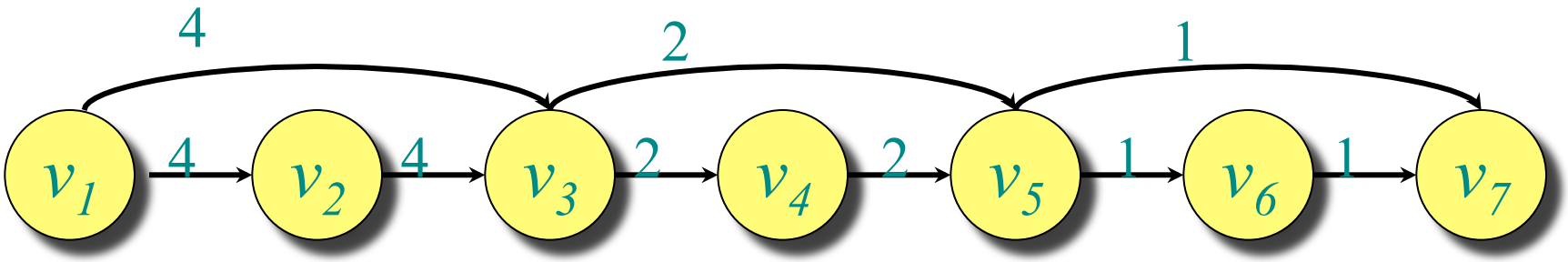
for $i \leftarrow 1$ **to** $|V|$-1
  do for each edge $(u, v) \in E$
    do if $d[v] > d[u] + w(u, v)$
      then $d[v] \leftarrow d[u] + w(u, v)$
        $\pi[v] \leftarrow$ u

relaxation steps $O(mn)$

for each edge $(u, v) \in E$
  do if $d[v] > d[u] + w(u, v)$
    then report a negative cycle

final steps $O(m)$

Bellman-Ford - example



Assume we scan edges from right to left

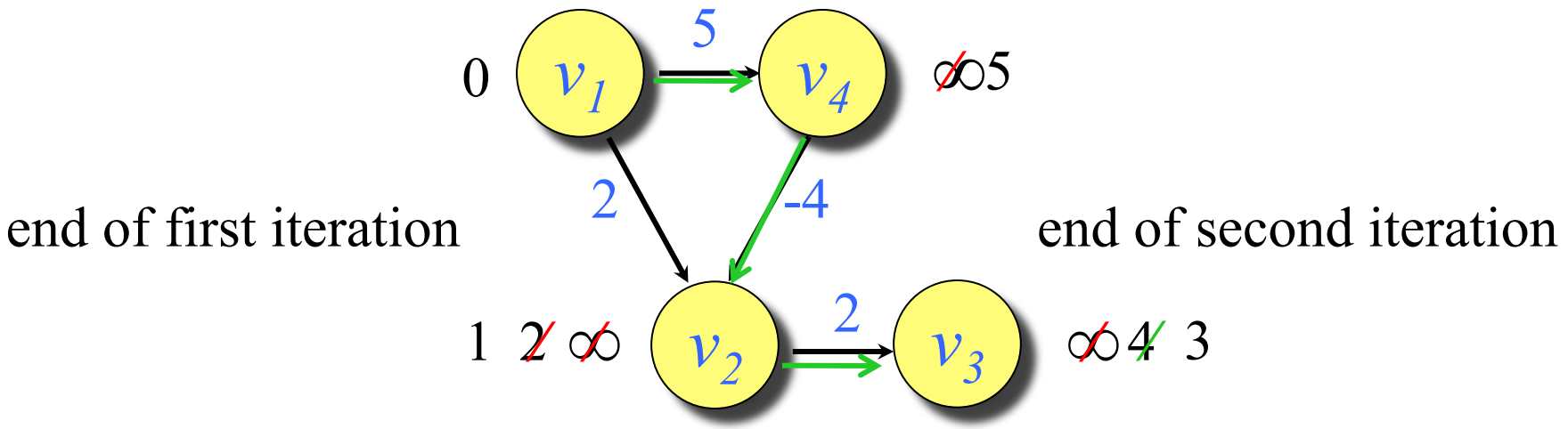
4	4				
4	4	6	6		
4	4	6	6	7	7

If we scan left to right

4	4	6	6	7	7

Bellman-Ford – another example

$E=\{(v_1,v_2); (v_1,v_4); (v_2,v_3); (v_4,v_2)\}$



the shortest paths from v_1

Bellman-Ford: Correctness, I

Theorem 1. If $G = (V, E)$ contains no negative weight cycles, then at the end of the algorithm, $d[v] = \delta(s,v)$ for all $v \in V$.

Proof. By induction on the following property $P(i)$:
After i iterations of the relaxation steps:

- if $d[v] \neq \infty$, it is the weight of some path from s to v;
- if there is a path from s to v with at most i edges, then $d[v]$ at most the weight of the shortest path from s to v with at most i edges.

Bellman-Ford: Correctness, II

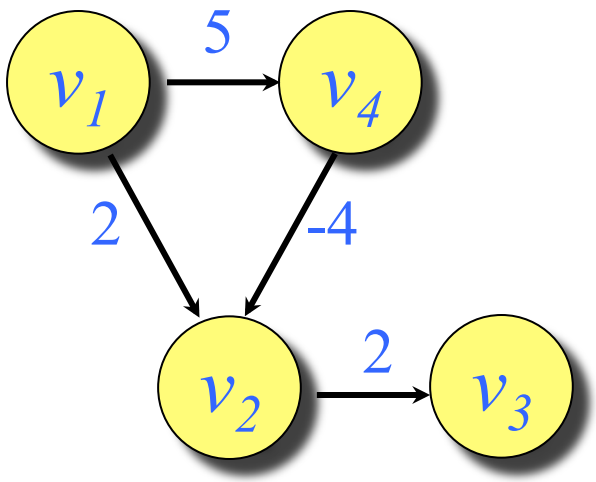
Theorem 2. The algorithm correctly reports the existence of a negative cycle in $G = (V, E)$

Proof. If $G = (V, E)$ contains a negative cycle, then there is always an edge that can be relaxed and the algorithm will then correctly report it.

Conversely if the algorithm reports a negative cycle, then there must be a cycle in a shortest path at the end of the final steps, and this must then be a negative cycle.

This graph has a special structure. Which? And how to use it with Bellman-Ford?

$E=\{(v_1,v_2); (v_1,v_4); (v_2,v_3); (v_4,v_2)\}$



... think about it for next time ...