CS3000: Algorithms & Data
Jonathan Ullman

Lecture 15:
- Bellman-Ford

Oct 30, 2018

Schedule

- Midterm II Nov 16
- No class Thanksgiving week

Bellman-Ford

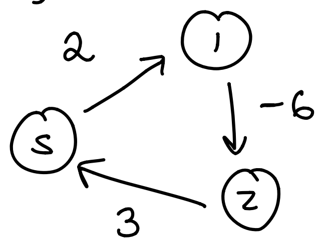
Dijkstra Recap

- **Input:** Directed, weighted graph $G = (V, E, \{\ell_e\})$, source node s
  - **Non-negative edge lengths** $\ell_e \geq 0$
- **Output:** Two arrays d, p (Single-source shortest paths)
  - $d[u]$ is the length of the shortest $s \rightsquigarrow u$ path
  - $p[u]$ is the final hop on shortest $s \rightsquigarrow u$ path
    ↳ edges in the shortest tree
- **Running time:** $O(m \log n)$
  - Implement using **binary heaps**

Ask the Audience

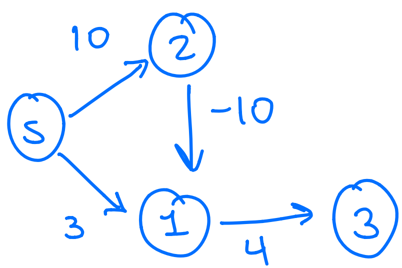
- Show that Dijkstra's Algorithm can fail in graphs with negative edge lengths, but no neg. length cycles.

① Negative-length cycles — shortest path is undefined



distances can be $-\infty$

②



Running Dijkstra gives

	s	1	2	3
d	$\boxed{0}$	∞	∞	∞
	0	$\boxed{3}$	10	∞
	0	3	10	$\boxed{7}$
	0	3	$\boxed{10}$	7
	0	0	10	7

Wrong distances →

Why Care About Negative Lengths?

- Models various phenomena
  - Transactions (credits and debits)
  - Currency exchange (log(exchange rate) can be + or -)
  - Chemical reactions (can be exo or endothermic)

- Leads to interesting algorithms
  - Variants of Bellman-Ford are used in internet routing

Bellman-Ford

- **Input:** Directed, weighted graph $G = (V, E, \{\ell_e\})$, source node s
  - **Possibly negative edge lengths $\ell_e \in \mathbb{R}$**
  - **No negative-length cycles!**

- **Output:** Two arrays d, p
  - $d[u]$ is the length of the shortest $s \rightsquigarrow u$ path
  - $p[u]$ is the final hop on shortest $s \rightsquigarrow u$ path

Ask the Audience

$C = -\min_{e \in E} \ell(e)$

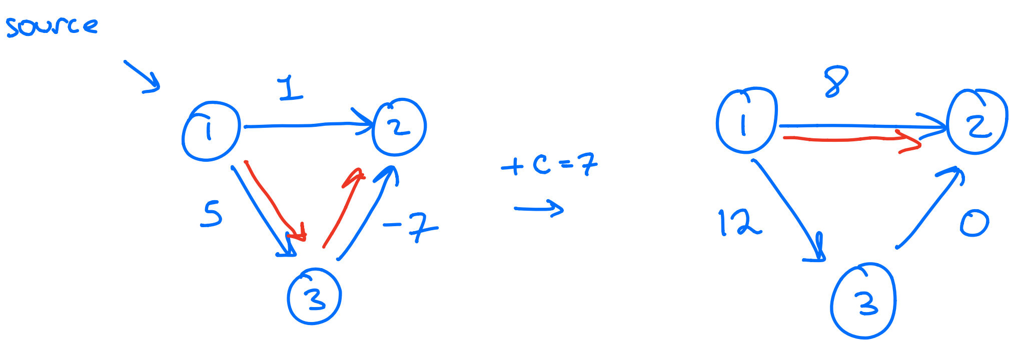
- Suppose we try the following algorithm
  - Take a graph $G = (V, E, \{\ell(e)\})$ with negative lengths
  - Add C to all lengths to make them non-negative
  - Run Dijkstra on the new graph
- Why wont this work?

$$G \quad s \xrightarrow{6} 1 \xrightarrow{-2} 2 \xrightarrow{-4} 3$$

$$G' \quad s \xrightarrow{10} 1 \xrightarrow{2} 2 \xrightarrow{0} 3 \qquad C = 4$$

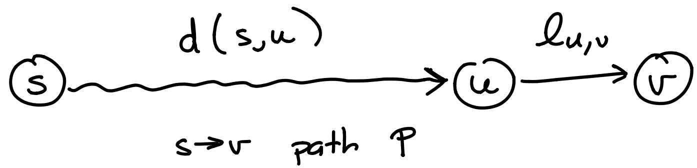
Ask the Audience

- Suppose we try the following algorithm
  - Take a graph $G = (V, E, \{\ell(e)\})$ with negative lengths
  - Add C to all lengths to make them non-negative
  - Run Dijkstra on the new graph
- Why wont this work?

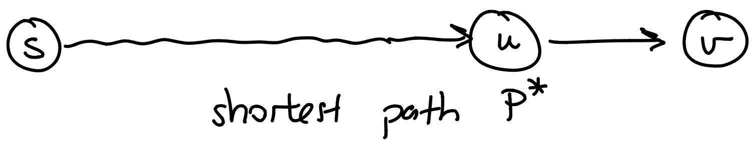


Structure of Shortest Paths

- If $(u, v) \in E$, then $d(s, v) \leq d(s, u) + \ell(u, v)$ for every node $s \in V$



- For every v, there exists an edge $(u, v) \in E$ such that $d(s, v) = d(s, u) + \ell(u, v)$



- If $(u, v) \in E$, and $d(s, v) = d(s, u) + \ell(u, v)$ then there is a shortest $s \rightsquigarrow v$-path ending with (u, v)

Dynamic Programming

- **Subproblems:** Let $\mathrm{OPT}(v)$ be the length of the shortest path from s to v

- For every v, the shortest path makes some final hop (u,v)
- Case u (the final hop is $(u,v) \in E$)

$$\hookrightarrow \mathrm{OPT}(v) = \mathrm{OPT}(u) + \ell(u,v)$$

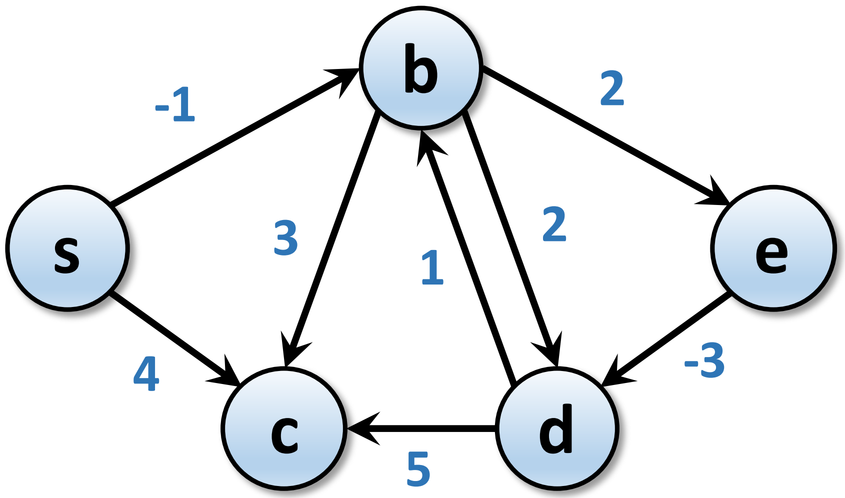
- Recurrence:

$$\mathrm{OPT}(v) = \min_{(u,v)\in E} \mathrm{OPT}(u) + \ell(u,v)$$

$$\mathrm{OPT}(0) = 0$$

Bottom-Up Implementation?

$$OPT(v) = \min_{(u,v) \in E} OPT(u) + \ell(u,v)$$



Need to fill the DP table in some order.

Cannot put the DP table "in order"

v	s	b	c	d	e
OPT(v)	0				

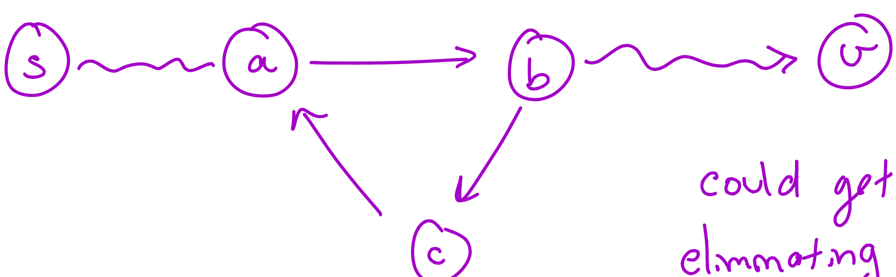
Dynamic Programming Take II

∃ a shortest path w/ $\leq n-1$ hops

- **Subproblems:** Let $\mathrm{OPT}(v, j)$ be the length of the shortest path from s to v with at most j hops

  - $0 \leq j \leq n-1$
  - Any path with $\geq n$ hops has a cycle, any cycle has ≥ 0 length

    ∴ shortest path has $\leq n-1$ hops

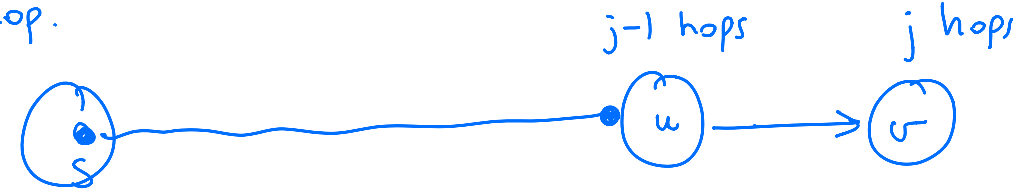


could get a shorter path by eliminating the cycle

Dynamic Programming Take II

- **Subproblems:** Let $\mathrm{OPT}(v, j)$ be the length of the shortest path from s to v with at most j hops

- Shortest path using j hops uses some (u, v) as the final hop.



$$\mathrm{OPT}(v, j) = \min \left\{ \mathrm{OPT}(v, j-1), \min_{(u,v) \in E} \mathrm{OPT}(u, j-1) + \ell(u,v) \right\}$$

$$\mathrm{OPT}(s, 0) = 0$$

$$\mathrm{OPT}(v, 0) = \infty \quad \forall v \neq s$$

Recurrence

- **Subproblems:** Let $\mathrm{OPT}(v, j)$ be the length of the shortest path from s to v with at most j hops
- **Case $\boldsymbol{u}$:** (u, v) is final edge on the shortest j-hop $s \rightsquigarrow v$ path

Recurrence:

$$\mathrm{OPT}(v, j) = \min\left\{\mathrm{OPT}(v, i-1), \min_{(u,v)\in E}\{\mathrm{OPT}(u, i-1) + \ell_{u,v}\}\right\}$$

$\mathrm{OPT}(s, j) = 0$ for every j

$\mathrm{OPT}(v, 0) = \infty$ for every v

Finding the paths

- $\mathrm{OPT}(v,j)$ is the length of the shortest $s \rightsquigarrow v$ path with at most j hops
- $P(v,j)$ is the last hop on some shortest $s \rightsquigarrow v$ path with at most j hops

(Edges in the shortest path tree only allowing $\le j$ hop paths)

Recurrence:

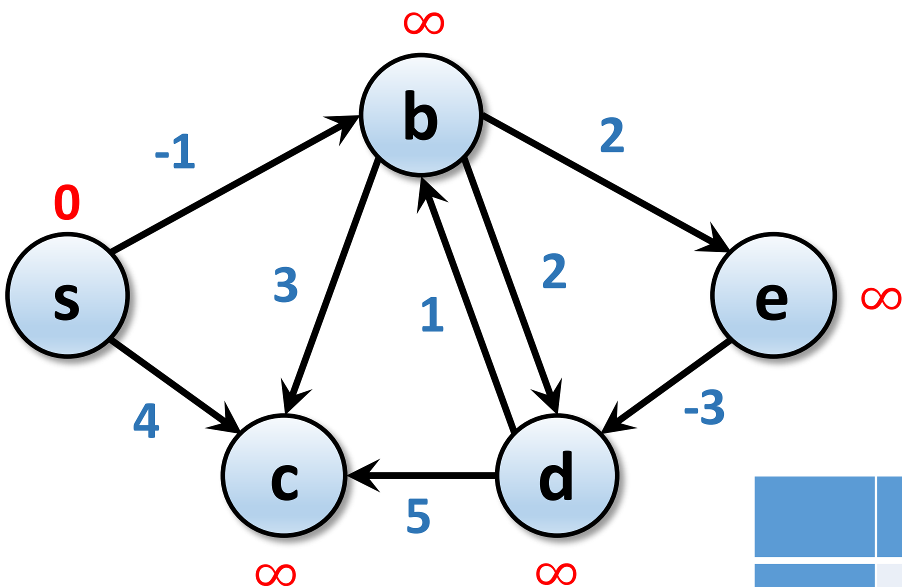
$$\mathrm{OPT}(v,j) = \min\left\{\mathrm{OPT}(v,i-1),\ \min_{(u,v)\in E}\left\{\mathrm{OPT}(u,i-1) + \ell_{u,v}\right\}\right\}$$

If $\mathrm{OPT}(v,j) = \mathrm{OPT}(u,j-1) + \ell(u,v)$ then there $\exists$ a shortest $s \rightsquigarrow v$ path w/ $\le j$ hops whose last hop is (u,v)

$\Rightarrow P(v,j) = u$

Example

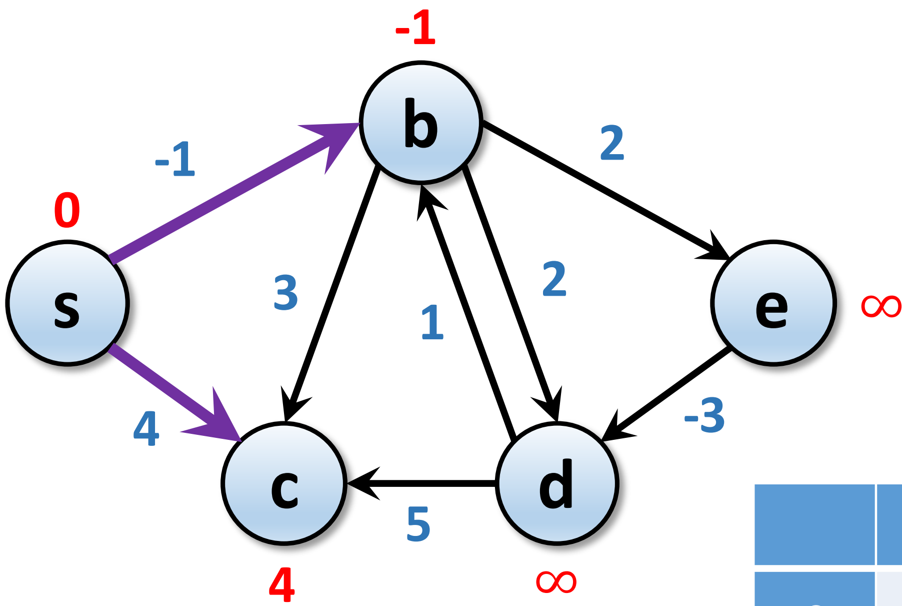
$$OPT(v,j) = \min \left\{ OPT(v,j-1),\ \min_{(u,v)\in E} OPT(u,j-1) + \ell_{u,v} \right\}$$



	0	1	2	3	4
s	0	0	0	0	0
b	∞				
c	∞				
d	∞				
e	∞				

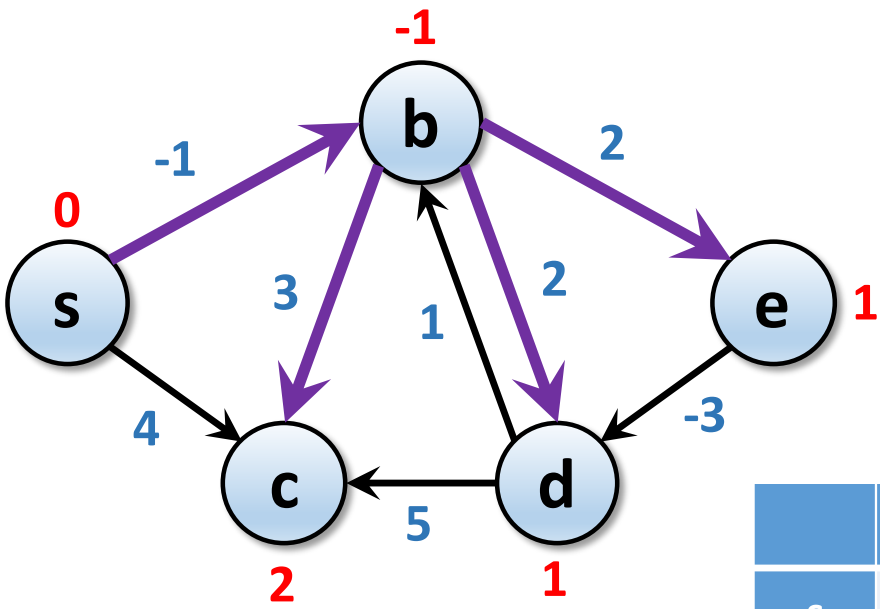
Example

$$OPT(v,j) = \min\left\{ OPT(v,j-1),\ \min_{(u,v)\in E} OPT(u,j-1) + \ell_{u,v} \right\}$$



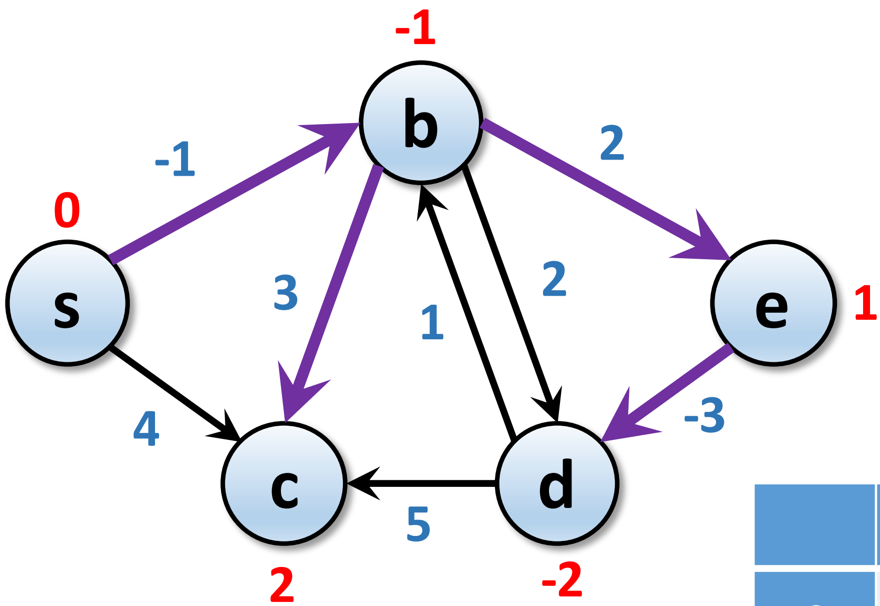
	0	1	2	3	4
s	0	0	0	0	0
b	∞	-1			
c	∞	4			
d	∞	∞			
e	∞	∞			

Example



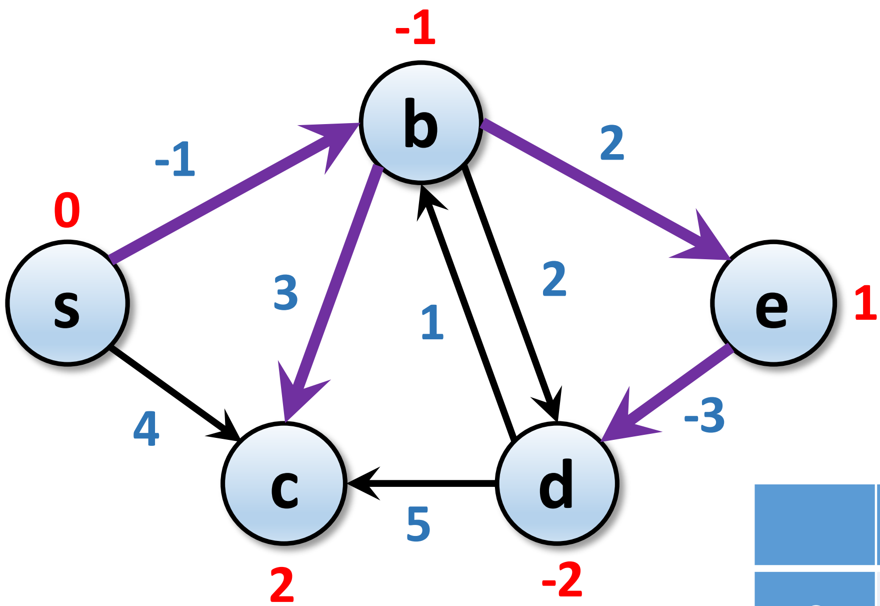
	0	1	2	3	4
s	0	0	0	0	0
b	∞	-1	-1		
c	∞	4	2		
d	∞	∞	1		
e	∞	∞	1		

Example



	0	1	2	3	4
s	0	0	0	0	0
b	∞	-1	-1	-1	
c	∞	4	2	2	
d	∞	∞	1	-2	
e	∞	∞	1	1	

Example



nothing changed

	0	1	2	3	4
s	0	0	0	0	0
b	∞	-1	-1	-1	-1
c	∞	4	2	2	2
d	∞	∞	1	-2	-2
e	∞	∞	1	1	1

Implementation (Bottom Up DP)

$D[v,j] = OPT(v,j)$

Running Time

```
Shortest-Path(G, s)
    foreach node v ∈ V
        D[v,0] ← ∞
        P[v,0] ← ⊥
    D[s,0] ← 0

    for i = 1 to n-1
        foreach node v ∈ V
            D[v,i] ← D[v,i-1]
            P[v,i] ← P[v,i-1]
            foreach edge (u,v) ∈ E
                if (D[u,i-1] + ℓuv < D[v,i])
                    D[v,i] ← D[u,i-1] + ℓuv
                    P[v,i] ← u
```

$O(n)$ time

$\leq n-1$ iterations

implements the recurrence

$O(1)$

$O(\text{indeg}(v))$

$\sum_{v \in V} O(\text{indeg}(v))$

$= O(m)$

Total Time $= O(nm)$

Total Space $= O(n^2)$

Optimizations

Once I find OPT(•, j),
don't care about
OPT(•, j−1)

- One array $d[v]$ containing shortest path found so far
- No need to check edges (u, v) unless $d[u]$ has changed
- Stop if no $d[v]$ has changed for a full pass through V

- **Theorem:**
  - Throughout the algorithm $M[v]$ is the length of some $s - v$ path
  - After i passes through the nodes, $M[v] \leq OPT(v, i)$

Implementation II

```
Efficient-Shortest-Path(G, s)
   foreach node v ∈ V
      D[v] ← ∞
      P[v] ← ⊥  (NULL)
   D[s] ← 0

   for i = 1 to n
      foreach node u ∈ V
        if (D[u] changed in the last iteration)
           foreach edge (u,v) ∈ E
            if (D[u] + ℓuv < D[v])
               D[v] ← D[u] + ℓuv
               P[v] ← u
        if (no D[u] changed): return (D,P)
```

Running Time $O(mn)$, but $O(m)$ in practice

Space $O(n)$

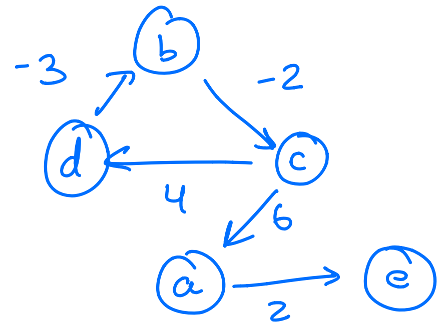
Negative Cycle Detection

- **Claim 1:** if $OPT(v, n) = OPT(v, n-1)$ then there are no negative cycles reachable from s
- **Claim 2:** if $OPT(v, n) < OPT(v, n-1)$ then any shortest $s - v$ path contains a negative cycle

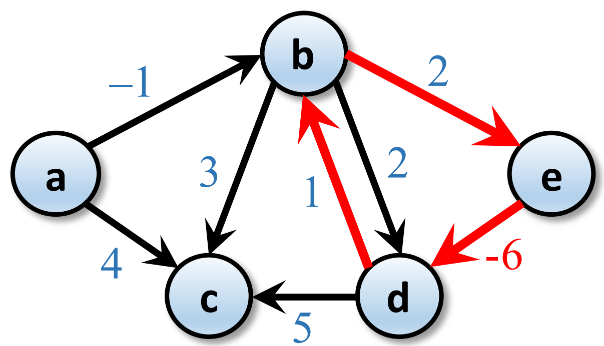
↳ shortest $s \rightsquigarrow v$ path w/ at most n hops, uses exactly n hops, and thus contains a cycle

this cycle must have negative length!

Negative Cycle Detection

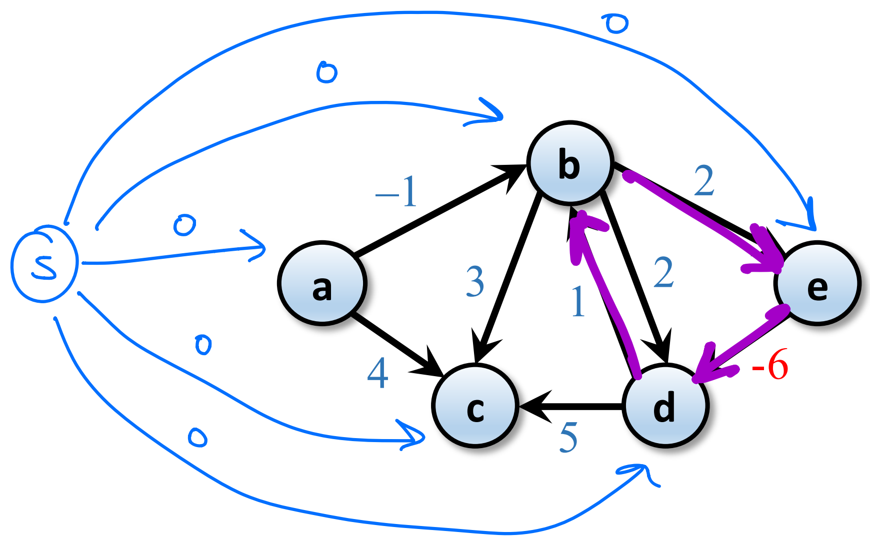


- **Algorithm:**
  - Pick a node $a \in V$
  - Run Bellman-Ford for n iterations
  - Check if $OPT(v, n) \neq OPT(v, n-1)$ for some $v \in V$
    - If no, then there are no negative cycles
    - If yes, the shortest $a - v$ path contains a negative cycle



Negative Cycle Detection

- **Algorithm:**
  - Add a new node $s \in V$, add edges (s, v) for every $v \in V$
  - Run Bellman-Ford for n iterations
  - Check if $OPT(v, n) \neq OPT(v, n-1)$ for some $v \in V$
    - If no, then there are no negative cycles
    - If yes, the shortest $s - v$ path contains a negative cycle



Shortest Paths Summary

- **Input:** Directed, weighted graph $G = (V, E, \{\ell_e\})$, source node s
- **Output:** Two arrays d, p
  - $d[u]$ is the length of the shortest $s \rightsquigarrow u$ path
  - $p[u]$ is the final hop on shortest $s \rightsquigarrow u$ path

- **Non-negative lengths**: Dijkstra's Algorithm solves in $O(m \log n)$ time
- **Negative lengths:** Bellman-Ford solves in $O(nm)$ time $O(n + m)$ space, or finds a negative cycle