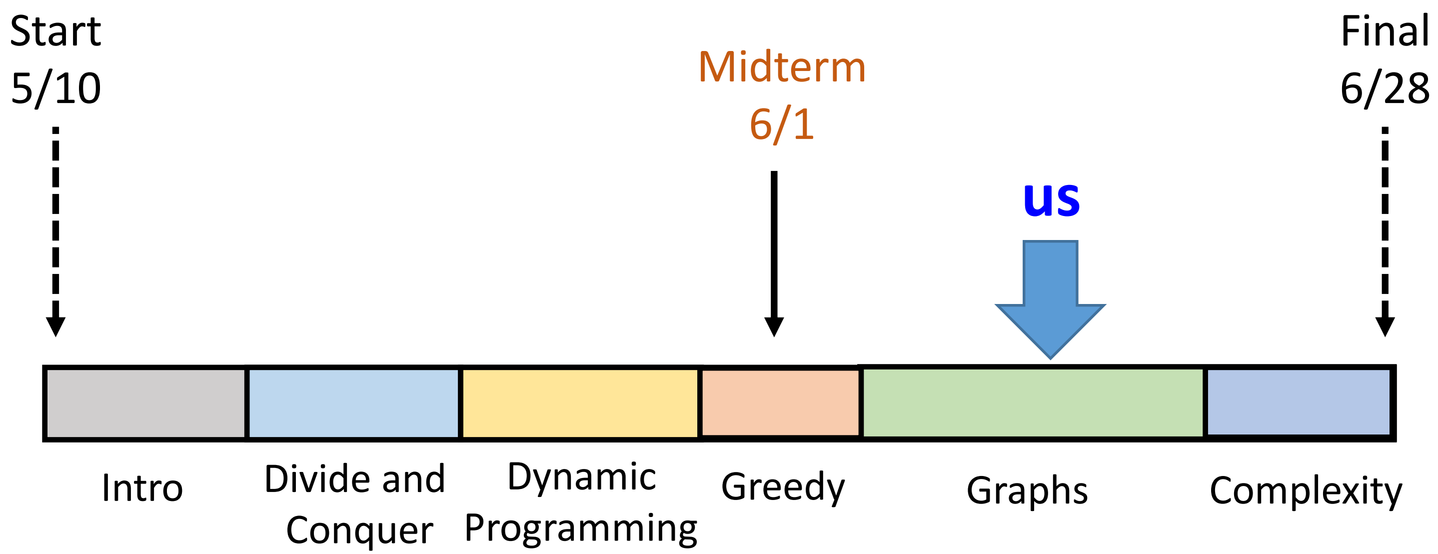
CS3000: Algorithms & Data
Drew van der Poel

Lecture 19
- Bellman-Ford

June 14, 2021

Outline

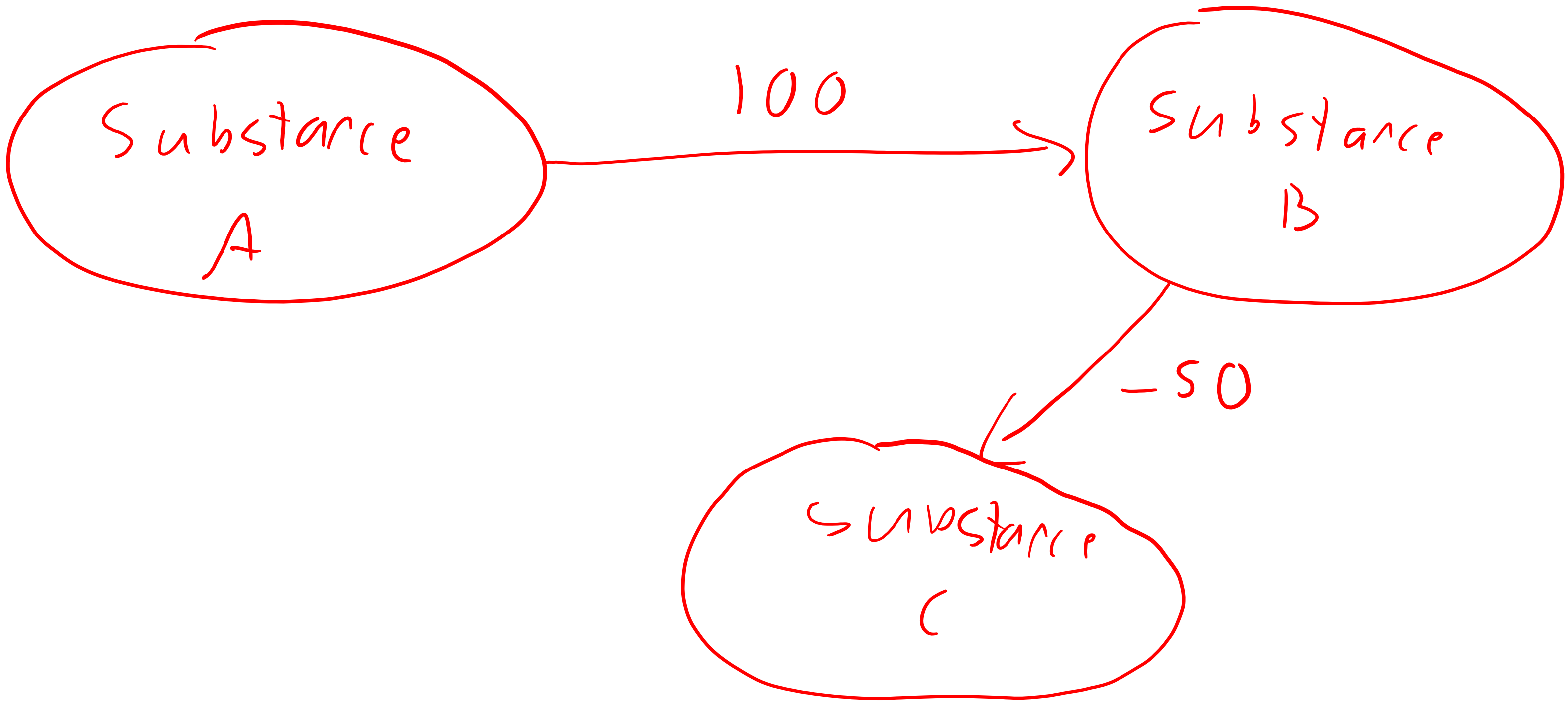


Last class: Graphs: Dijkstra's + Heaps

Next class: Graphs: Minimum Spanning Tree

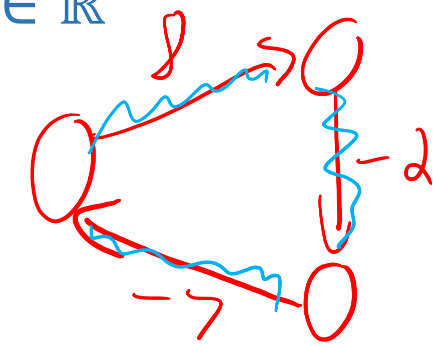
Why Care About Negative Edge Weights?

- Models various phenomena
  - Chemical reactions (can be exo- or endothermic)
  - Changes in level state (e.g. happiness)
  - …



Bellman-Ford

- **Input:** Directed, weighted graph $G = (V, E, \{w_e\})$, source node s
    - **Possibly negative edge lengths** $w_e \in \mathbb{R}$
    - **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

Parent of u

- Problems: counting students, stable matching, sorting, n-digit multiplication, array searching, selection, weighted interval scheduling, segmented least squares, knapsack, prefix-free encoding, graph exploration, bipartiteness, topological sorting, (strongly) connected components, **shortest paths**

- Alg. techniques: divide & conquer, dynamic programming, greedy, Dijkstra's

- Analysis: asymptotic analysis, recursion trees, Master Thm., Graph Terminology/representations

- Proof techniques: (strong) induction, contradiction, greedy stays ahead, exchange argument

Structure of Shortest Paths

last time

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

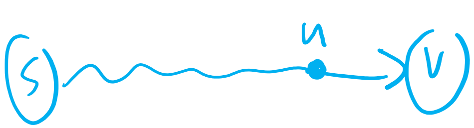
"shortest s-v path is not longer than the shortest s-u path + (u,v)"

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

if bound from above is tight, then then there is a shortest s-v path s.t. $P[v] = u$

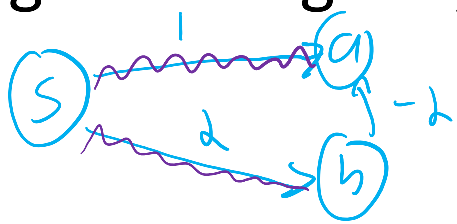
- For every *v*, there exists an edge $(u, v) \in E$ such that *d(s,v) = d(s,u) + w(u,v)*

"there is some final edge"



Ask the Audience

- Show that Dijkstra's Algorithm can fail in graphs with negative edge lengths (even without negative length cycles)



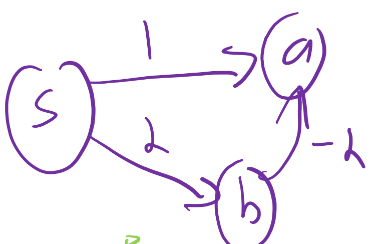
$d(s,a) = 0 < 1$

given by Dijkstra's

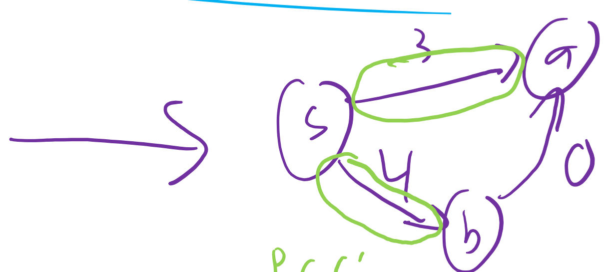
s	a	b
0	∞	∞
	1	2
		2

- Why won't the following work?
  - Take a graph $G = (V, E, \{w(e)\})$ with negative lengths
  - Add $|\min w(e)|$ to all lengths to make them non-negative
  - Run Dijkstra's on the new graph

distance increases based on # of edges



$P \in G$



$P \in G'$

len in $G = 0$

len in $G' = 4$

$= 0 + 2 \cdot (\#\text{ of edges})$

Dynamic Programming

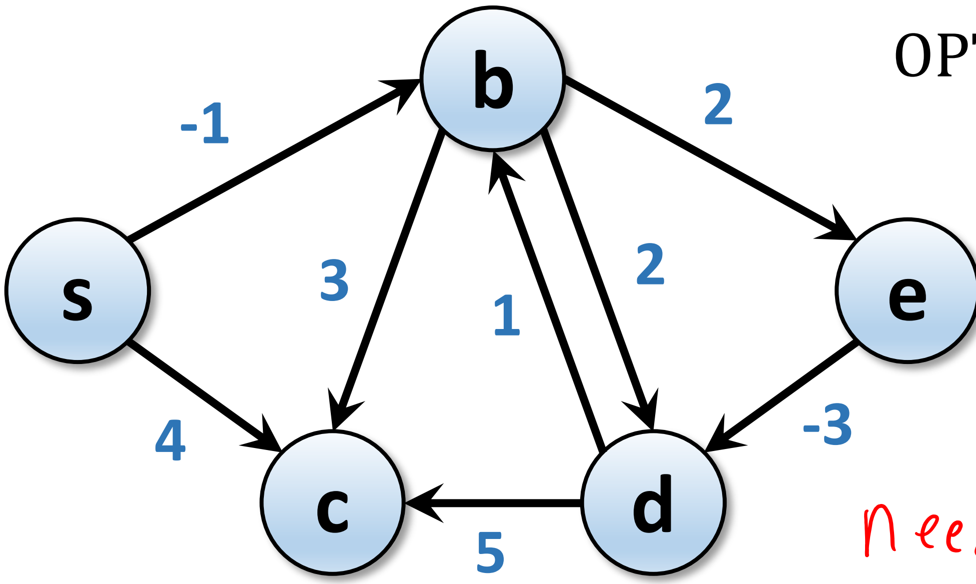
- **Subproblems:** Let OPT(*v*) be the length of the shortest path from *s* to *v*
- For every *v*, the shortest path makes some final hop *(u,v)* "last edge"

  $\to u \in IN[v]$
- Case *u:* the final hop is *(u,v)*
  - *OPT(v)* = $OPT(u) + w(u,v)$
- Recurrence:

$$OPT(v) = \min_{u \in IN[v]} \left(OPT(u) + w(u,v) \right)$$

$$OPT(s) = 0$$

Bottom-Up Implementation?



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

problem!

Need an order in which to fill the DP table!

v	s	b	c	d	e
OPT(v)	0				

Dynamic Programming Take II

- **Subproblems:** Let $\text{OPT}(v, j)$ be the length of the shortest path from s to v with at most j hops $\quad (0 \le j \le n - 1)$

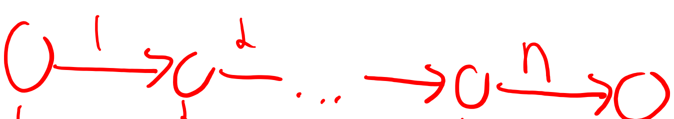
"edges"

Why $j \le n - 1$?

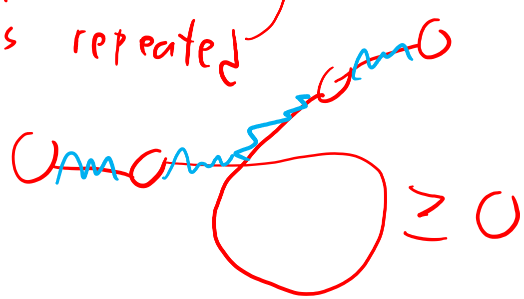
→ Any path w/ $\ge n$ edges has a cycle

→ if (−) no solution

→ if (≥ 0), cycle is useless



same node is repeated



∴ shortest s-v path uses $\le n-1$ hops

Recurrence

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

$$\mathrm{OPT}(v,j) = \min\left(\mathrm{OPT}(v, j-1),\ \min_{u \in IN[v]} \left(\mathrm{OPT}(u, j-1) + w(u,v) \right) \right)$$

Recurrence

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

Recurrence:

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

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

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

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

Recurrence:

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

Finding $P(v, j)$: If $OPT(v,j) == OPT(v,j-1)$

$\rightarrow P(v,j) = P(v,j-1)$

If $OPT(v,j) == OPT(u,j-1) + w(u,v)$

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

- Problems: counting students, stable matching, sorting, n-digit multiplication, array searching, selection, weighted interval scheduling, segmented least squares, knapsack, prefix-free encoding, graph exploration, bipartiteness, topological sorting, (strongly) connected components, shortest paths

- Alg. techniques: divide & conquer, dynamic programming, greedy, Dijkstra's, **Bellman-Ford**

- Analysis: asymptotic analysis, recursion trees, Master Thm., Graph Terminology/representations

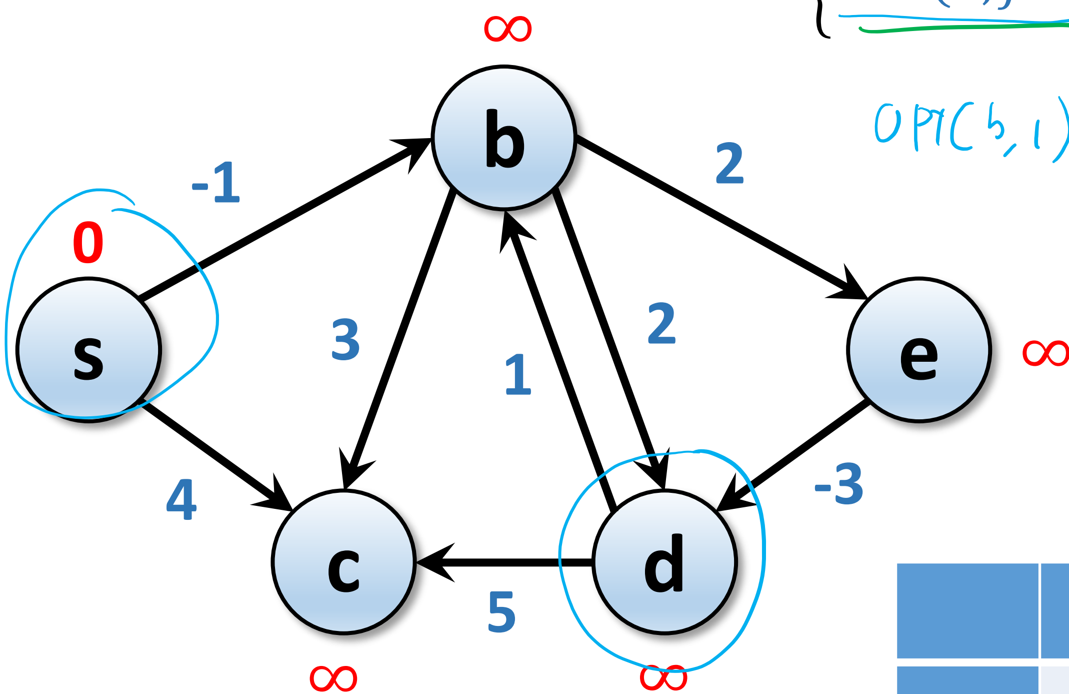
- Proof techniques: (strong) induction, contradiction, greedy stays ahead, exchange argument

Example

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

$u \in IN[v]$

$\mathrm{OPT}(b,1) = \min(\mathrm{OPT}(b,0),$
$\min(\mathrm{OPT}(s,0) + w(s,b),$
$\mathrm{OPT}(d,0) + w(d,b)))$
$= \min(\infty, \min(-1, \infty)) = -1$



$\mathrm{OPT}(c,1)$
$= \min(\mathrm{OPT}(c,0),$
$\min(\mathrm{OPT}(s,0) + w(s,c),$
$\mathrm{OPT}(b,0) + w(b,c),$
$\mathrm{OPT}(d,0) + w(d,c)))$
$= \min(\infty, \min(4, \infty, \infty)) = 4$

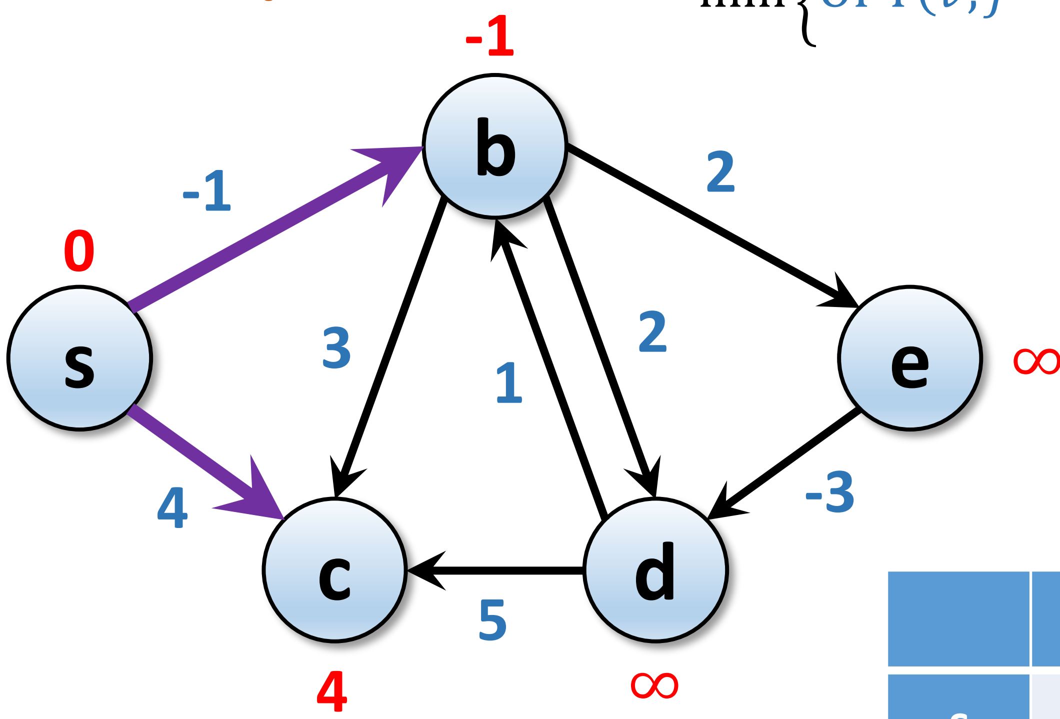
j

v

	0	1	2	3	4
s	0	0			
b	∞	-1			
c	∞	$\boxed{4}$			
d	∞	∞			
e	∞	∞			

Example

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



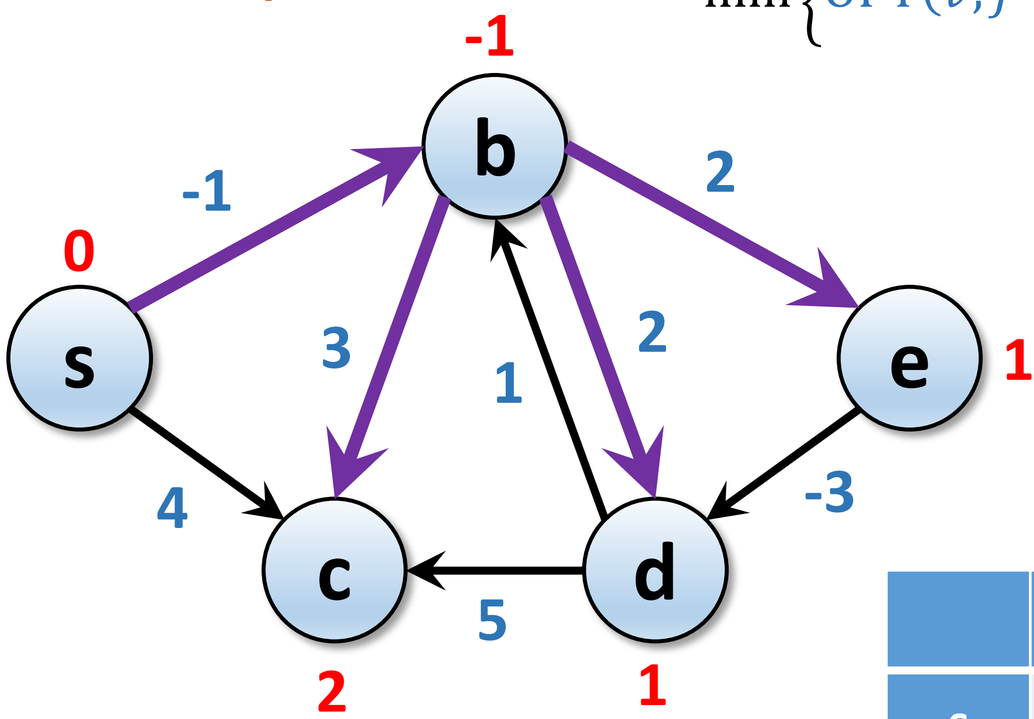
j

v

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

Example

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



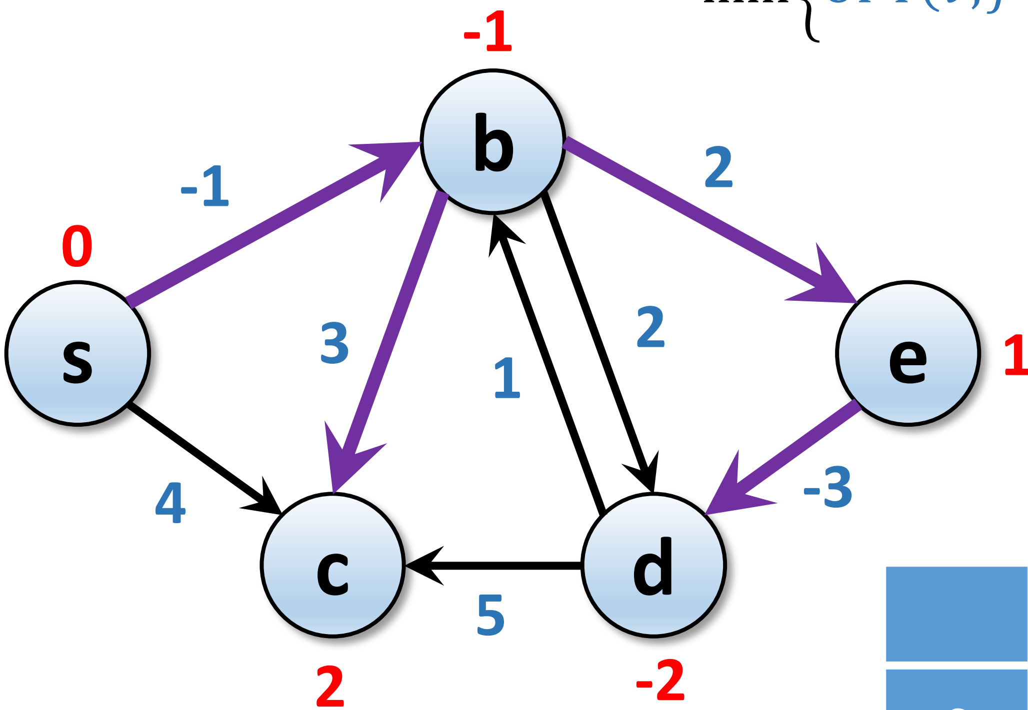
j

v

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

Example

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



nothing changed

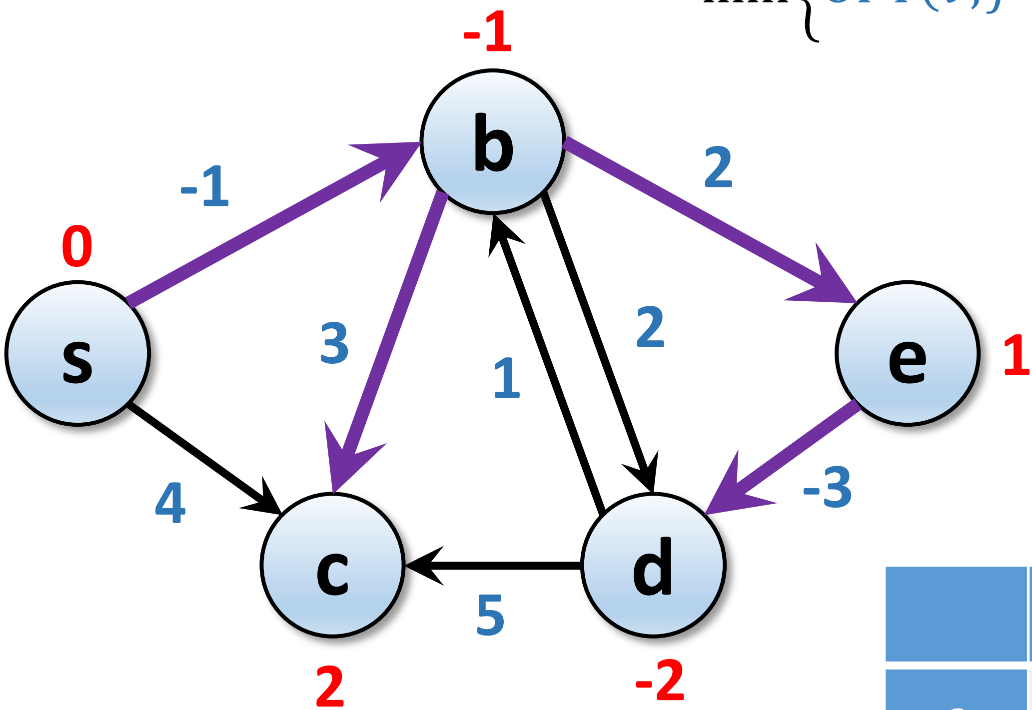
j

v

	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

Example

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



j

v

	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)

G = (V, E)

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

   for j = 1 to n-1
      foreach node v ∈ V
         D[v,j] ← D[v,j-1]
         P[v,j] ← P[v,j-1]
         foreach ~~edge~~ node u ∈ IN[v]
            if (D[u,j-1] + w_uv < D[v,j])
               D[v,j] ← D[u,j-1] + w_uv
               P[v,j] ← u
```

$O(n)$

one column @ a time

$O(n)$ →

per value of j: $O(n)$

total: $O(n^2)$

per value of v: $O(\text{in-deg}(v))$

per value of j: $\boxed{O(m)}$

total: $O(nm)$

Time: $O(n^2 + nm)$

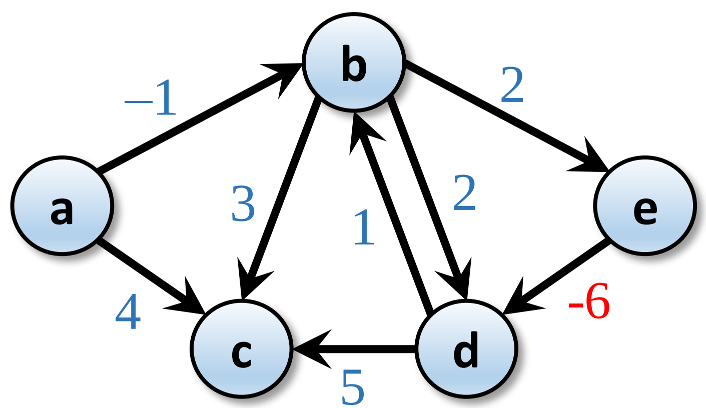
Space: $O(n^2) \leftarrow n \times (n-1)$

Optimizations

- One array $d[v]$ containing shortest path found so far
  - Once you have *OPT(v, j)*, don't care about *OPT(v, j-1)*
- No need to check edge (u, v) unless $d[u]$ has changed
- Stop if no $d[v]$ has changed for a full pass through V
- **Time:** $O(nm + n^2)$ (better in practice)
- **Space:** $O(n)$ (+ storing the graph)

Negative Cycle Detection

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



Optimized Implementation w/ Negative Cycle Detection

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

   for j = 1 to n
      foreach node v ∈ V
           foreach u ∈ IN[v] where D[u] changed
              during last iteration
           if (D[u] + w_uv < D[v])
              D[v] ← D[u] + w_uv
              P[v] ← u
      if (no D[u] changed): return (D,P)
```

Shortest Paths Summary

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

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