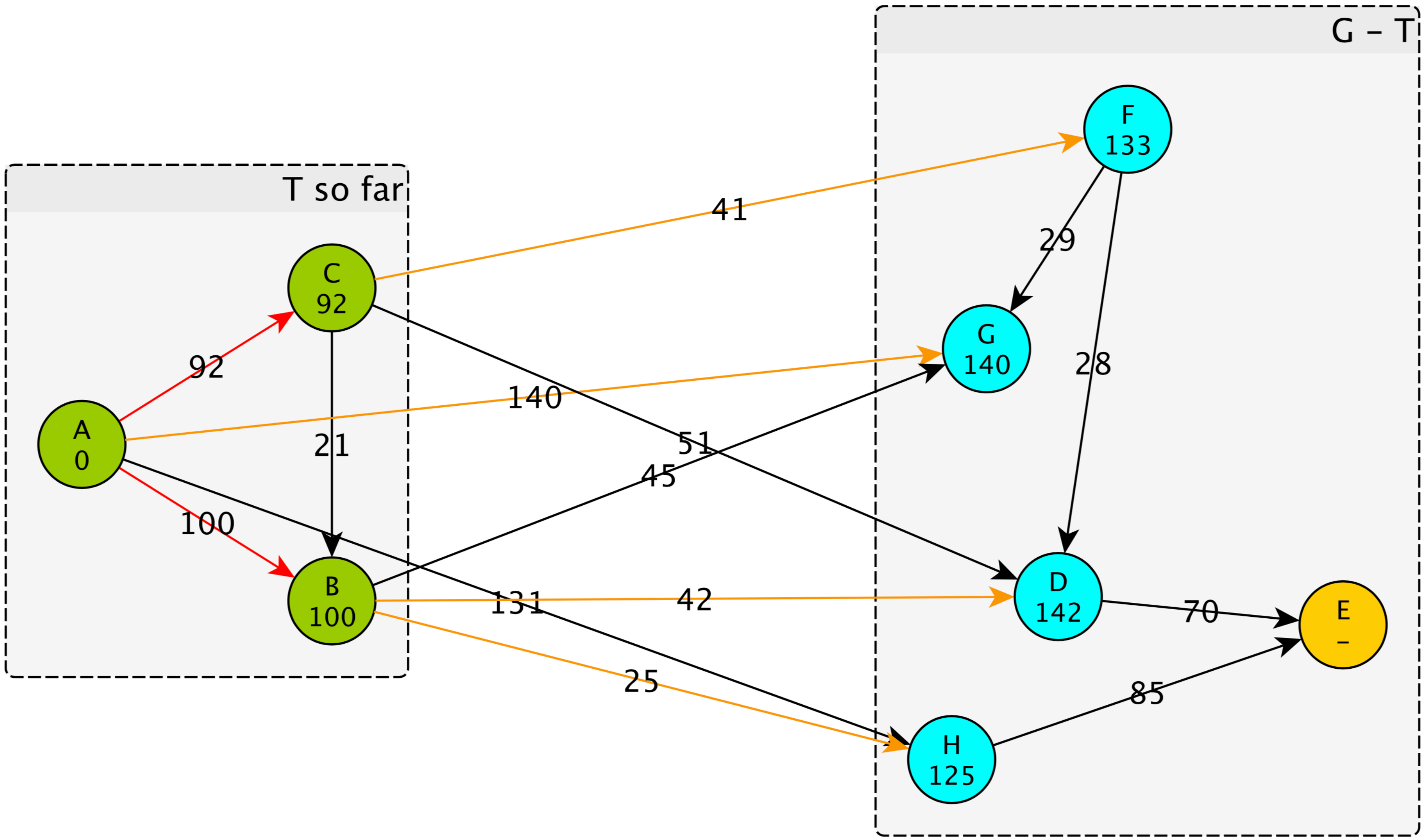
Last Topic in Dynamic Programming:

Shortest Paths Revisited

Shortest Path Problem

- **Single-Source Shortest Path Problem**.
  Given a directed graph $G = (V, E)$ with edge weights w_e on each $e \in E$ and a a source node s, find the shortest path from s to to all nodes in G.
- **Negative weights**. The edge-weights w_e in G can be negative. (When we studied Dijkstra's, we assumed non-negative weights.)
- Let P be a path from s to t, denoted $s \rightsquigarrow t$.
  - The **length** of P is the number of edges in P
  - The cost or weight of P is $w(P) = \sum_{e \in P} w_e$
- Goal: **cost** of the shortest path from s to all nodes

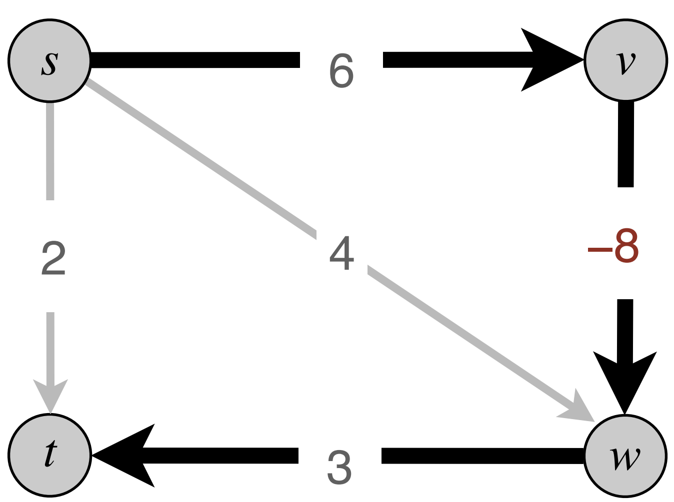
Remember Dijkstra's Algorithm?



Estimate at vertex v is the weight of shortest path in T followed by a single edge from T to $G - T$

Negative Weights & Dijkstra's

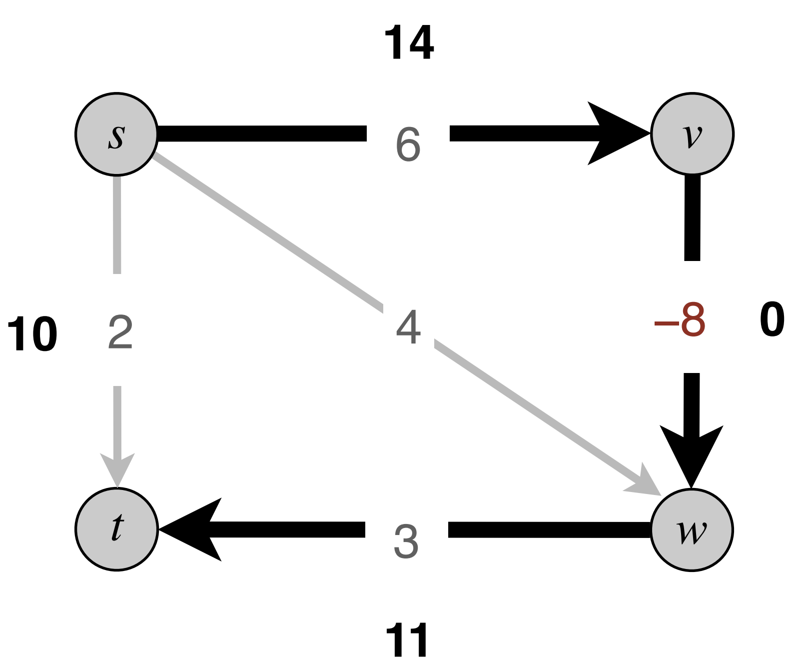
- **Dijkstra's Algorithm**. Does the greedy approach work for graphs with negative edge weights?
  - Dijkstra's will explore s's neighbor and add t, with $d[t] = w_{sv} = 2$ to the shortest path tree
  - Dijkstra assumes that there cannot be a "longer path" that has lower cost (relies on edge weights being non-negative)



Dijkstra's will find $s \to t$ as shortest path with cost 2
But the shortest path is $s \to v \to w \to t$ with cost 1

Negative Weights: Failed Attempt

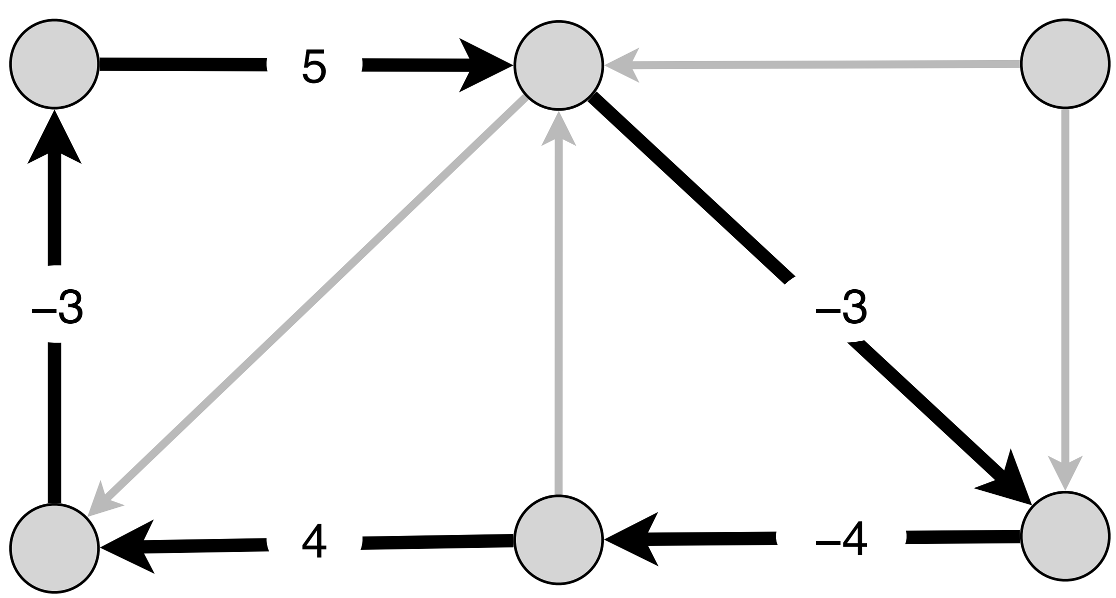
- What if we add a large enough constant C such that all weights become positive
    - $w'_{ij} = w_{ij} + C > 0$
    - Run Dijkstra's algorithm based with w'
- Does this give us the shortest path in the original graph?



Adding C = 8 to all weights does not work

Negative Cycles

- **Definition**. A negative cycle is a directed cycle C such that the sum of all the edge weights in C is less than zero
- **Question**. How do negative cycles affect shortest path?



a negative cycle W : $\ell(W) = \sum_{e \in W} \ell_e < 0$

Negative Cycles & Shortest Paths

- **Claim.** If a path from s to some node v contains a negative cycle, then there does not exist a shortest path from s to v.

- **Proof**.
  - Suppose there exists a shortest $s \rightsquigarrow v$ path with cost d that traverses the negative cycle t times for $t \geq 0$.
  - Can construct a shorter path by traversing the cycle $t + 1$ times

  $\Rightarrow\!\Leftarrow$ ∎

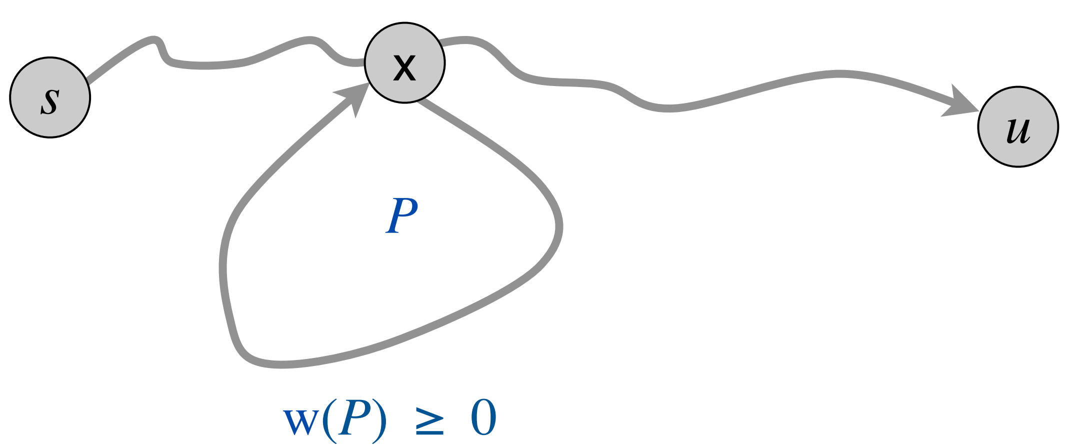
- **Assumption.** G has no negative cycle.
- Later in the lecture: how can we detect whether the input graph G contains a negative cycle?

Dynamic Programming Approach

- First step to a dynamic program? Recursive formulation
  - Subproblem with an “optimal substructure”
- **Structure of the problem**. Interested in optimal cost path (can have any length)
  - Easier to build on subproblems if we keep track of length of paths considered so far
- How long can the shortest path from s to any node u be, assuming no negative cycle?
- **Claim**. If G has no negative cycles, then exists a shortest path from s to any node u that uses at most $n-1$ edges.

No. of Edges in Shortest Path

- **Claim.** If G has no negative cycles, then exists a shortest path from s to any node u that uses at most $n - 1$ edges.
- **Proof**. Suppose there exists a shortest path from s to u made up of n or more edges
- A path of length at least n must visit at least $n + 1$ nodes
- There exists a node x that is visited more than once (**pigeonhole principle**). Let P denote the portion of the path between the successive visits.
- Can remove P without increasing cost of path. ∎



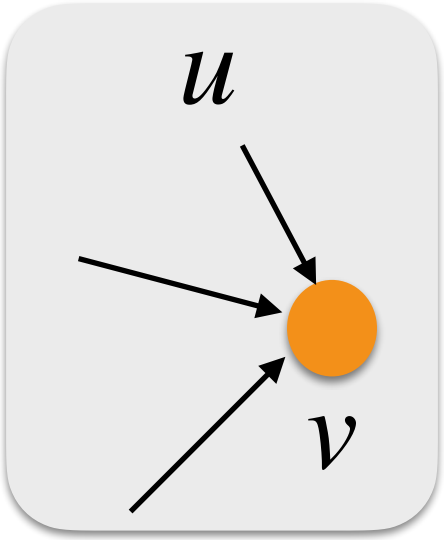
Shortest Paths: Dynamic Program

- **Subproblem**. $D[v, i]$: (optimal) cost of shortest path from s to v using $\leq i$ edges
- **Base cases**.
  - $D[s, i] = 0$ for any i
  - $D[v, 0] = \infty$ for any $v \neq s$
- **Final answer** for shortest path cost to node v
  - $D[v, n - 1]$
- How do we formulate the **recurrence**?
  - **Case 1**. Shortest path to v uses exactly i edges
  - **Case 2**. Shortest path to v uses less than i edges (that is, uses $\leq i - 1$ edges)

Shortest Paths: Recurrence

- **Subproblem**. $D[v, i]$: (optimal) cost of shortest path from s to v using $\leq i$ edges
- **Base cases**.
  - $D[s, i] = 0$ for any i
  - $D[v,0] = \infty$ for any $v \neq s$
- **Final answer** for shortest path cost to node v
  - $D[v, n - 1]$
- **Recurrence.**

$$D[v, i] = \min\{D[v, i-1], \min_{(u,v)\in E} \{D[u, i-1] + w_{uv}\}\}$$



- Called the **Bellman-Ford-Moore** algorithm

Bellman-Ford-Moore Algorithm

- **Subproblem**. $D[v, i]$: (optimal) cost of shortest path from s to v using $\leq i$ edges
- **Base cases.** $D[s, i] = 0$ for any i and $D[v, 0] = \infty$ for any $v \neq s$
- **Final answer** for shortest path cost to node v: $D[v, n-1]$
- **Recurrence.**
  $$D[v, i] = \min\{D[v, i-1], \min_{(u,v)\in E} \{D[u, i-1] + w_{uv}\}\}$$
- **Memoization structure**. Two-dimensional array
- **Evaluation order**.
  - $i : 1 \rightarrow n-1$ (column major order)
  - Starting from s, the row of vertices can be in any order

Bellman-Ford: Running Time

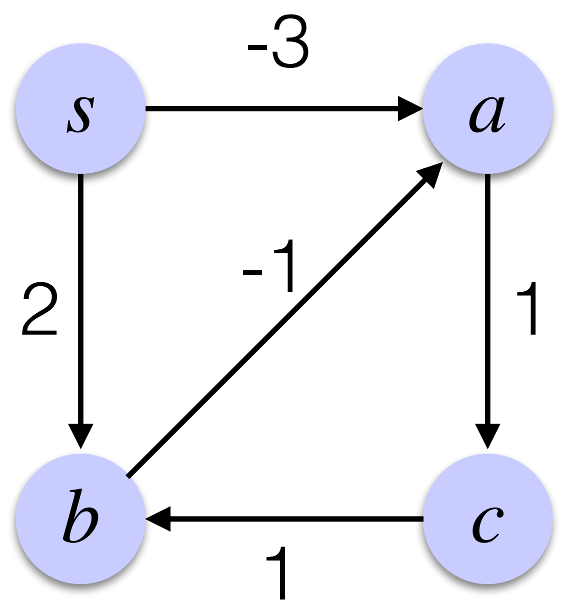
- **Recurrence**.
  $D[v,i] = \min\{D[v,i-1], \min_{(u,v)\in E} \{D[u,i-1] + w_{uv}\}\}$
- **Naive analysis**. $O(n^3)$ time
  - Each entry takes $O(n)$ to compute, there are $O(n^2)$ entries
- **Improved analysis**. For a given i, v, $d[v,i]$ looks at each incoming edge of v
  - Takes indegree(v) accesses to the table
  - For a given i, filling $d[-,i]$ takes $\sum_{v\in V}$ indegree(v) accesses
  - At most $O(n+m) = O(m)$ accesses for connected graphs where $m \geq n-1$
- Overall running time is $O(nm)$

- **Shortest-Path Summary.** Assuming there are no negative cycles in G, we can compute the shortest path from s to all nodes in G in $O(nm)$ time using the Bellman-Ford-Moore algorithm

Dynamic Programming
Shortest Path:
Bellman-Ford-Moore Example

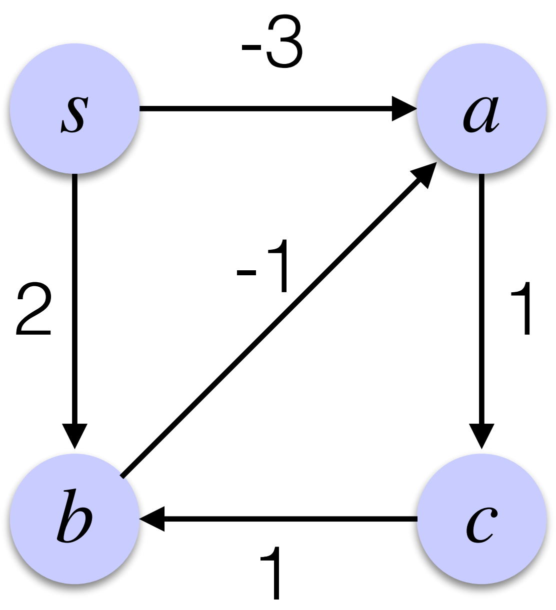
- $D[s, i] = 0$ for any i
- $D[v, 0] = \infty$ for any $v \neq s$

	0	1	2	3
s	0	0	0	0
a	inf			
b	inf			
c	inf			



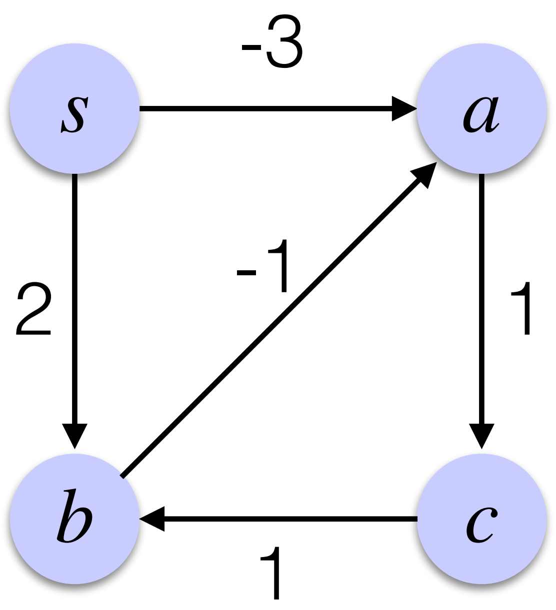
- $D[v,1] = \min\{D[v,0], \min_{u,v \in E} \{D[u,0] + w_{uv}\}$

	0	1	2	3
s	0	0	0	0
a	inf			
b	inf			
c	inf			



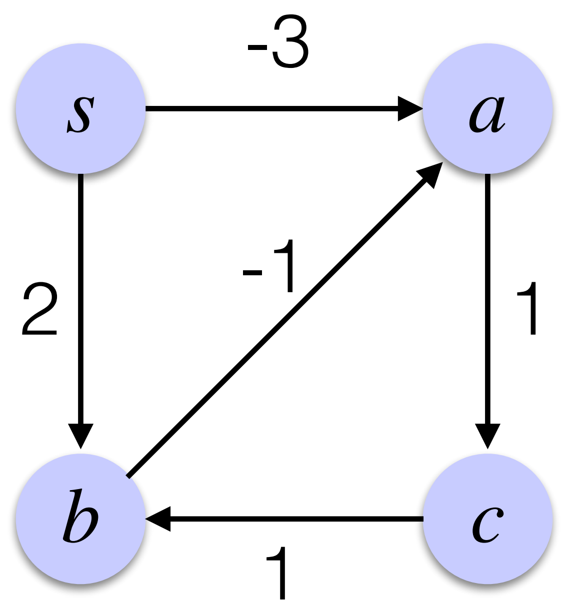
- $D[v,1] = \min\{D[v,0], \min_{u,v \in E} \{D[u,0] + w_{uv}\}$

	0	1	2	3
s	0	0	0	0
a	inf	-3		
b	inf			
c	inf			



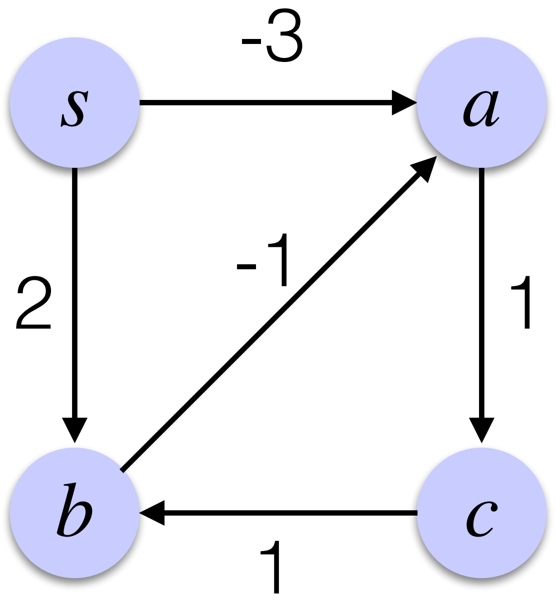
- $D[v,1] = \min\{D[v,0], \min_{u,v \in E} \{D[u,0] + w_{uv}\}$

	0	1	2	3
s	0	0	0	0
a	inf	-3		
b	inf	2		
c	inf			



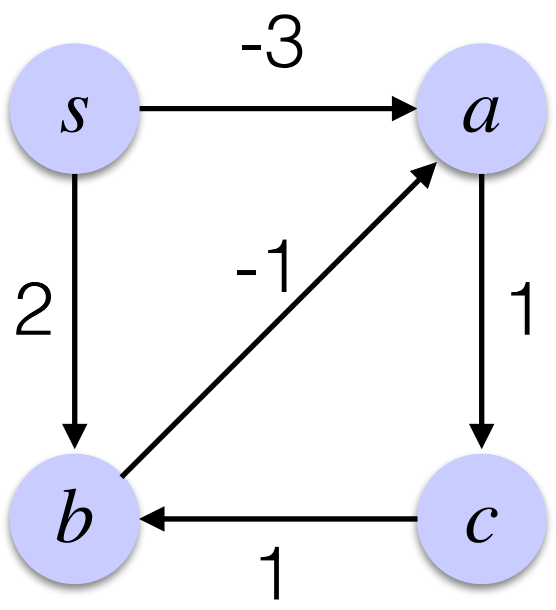
- $D[v,1] = \min\{D[v,0], \min_{u,v \in E} \{D[u,0] + w_{uv}\}$

	0	1	2	3
s	0	0	0	0
a	inf	-3		
b	inf	2		
c	inf	inf		



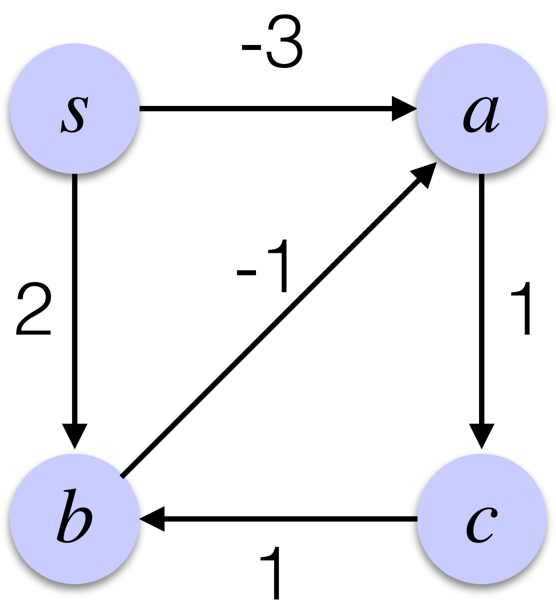
- $D[v,2] = \min\{D[v,1], \min_{u,v \in E}\{D[u,1] + w_{uv}\}$

	0	1	2	3
s	0	0	0	0
a	inf	-3		
b	inf	2		
c	inf	inf		



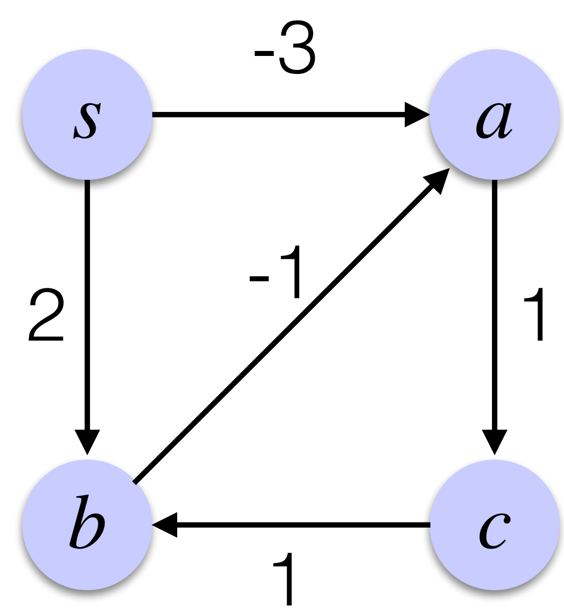
- $D[v,2] = \min\{D[v,1], \min_{u,v \in E} \{D[u,1] + w_{uv}\}$

	0	1	2	3
s	0	0	0	0
a	inf	-3	-3	
b	inf	2		
c	inf	inf		



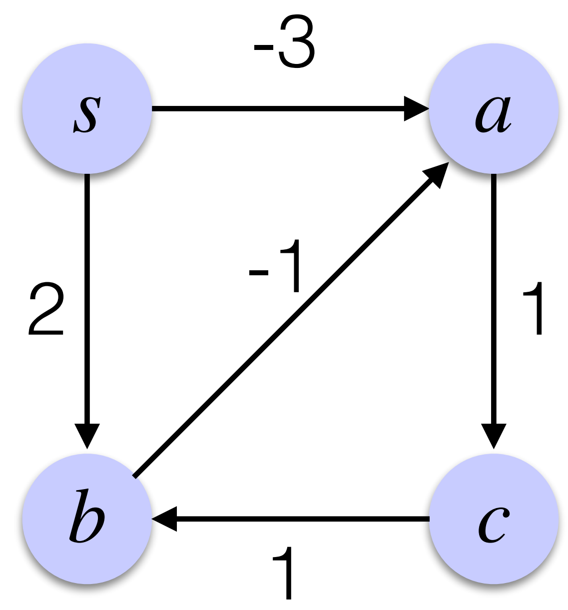
- $D[v,2] = \min\{D[v,1], \min_{u,v \in E} \{D[u,1] + w_{uv}\}$

	0	1	2	3
s	0	0	0	0
a	inf	-3	-3	
b	inf	2	2	
c	inf	inf		



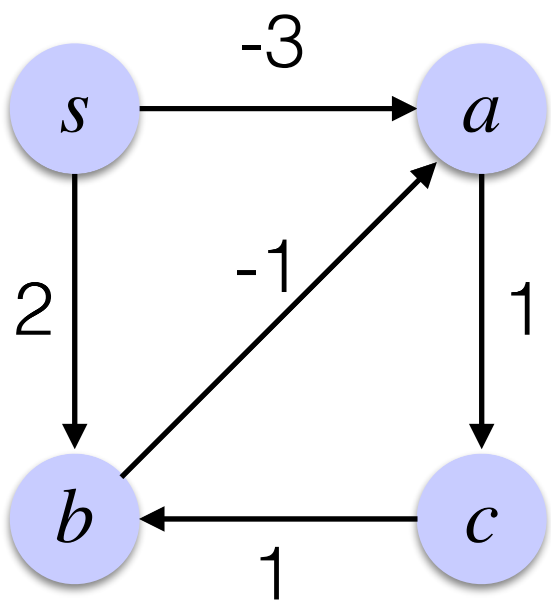
- $D[v,2] = \min\{D[v,1], \min_{u,v \in E} \{D[u,1] + w_{uv}\}$

	0	1	2	3
s	0	0	0	0
a	inf	-3	-3	
b	inf	2	2	
c	inf	inf	-2	



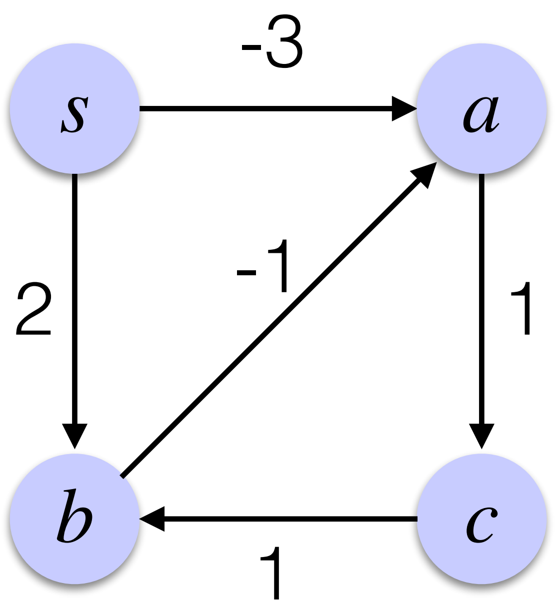
- $D[v,3] = \min\{D[v,2], \min_{u,v \in E} \{D[u,2] + w_{uv}\}$

	0	1	2	3
s	0	0	0	0
a	inf	-3	-3	-3
b	inf	2	2	
c	inf	inf	-2	



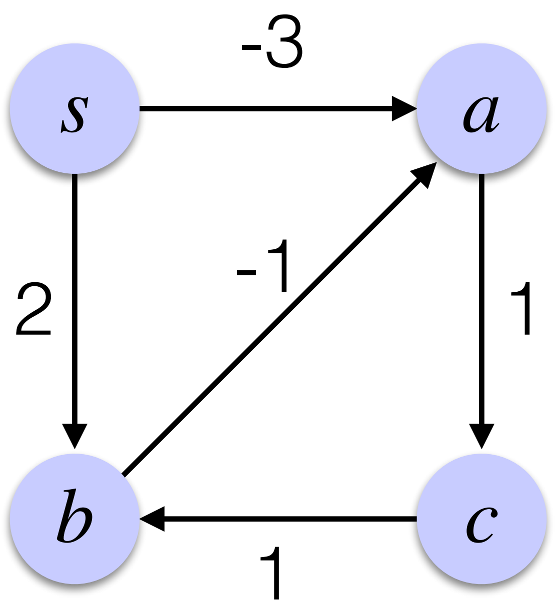
- $D[v,3] = \min\{D[v,2], \min_{u,v \in E} \{D[u,2] + w_{uv}\}$

	0	1	2	3
s	0	0	0	0
a	inf	-3	-3	-3
b	inf	2	2	-1
c	inf	inf	-2	



- $D[v,3] = \min\{D[v,2], \min_{u,v \in E}\{D[u,2] + w_{uv}\}$

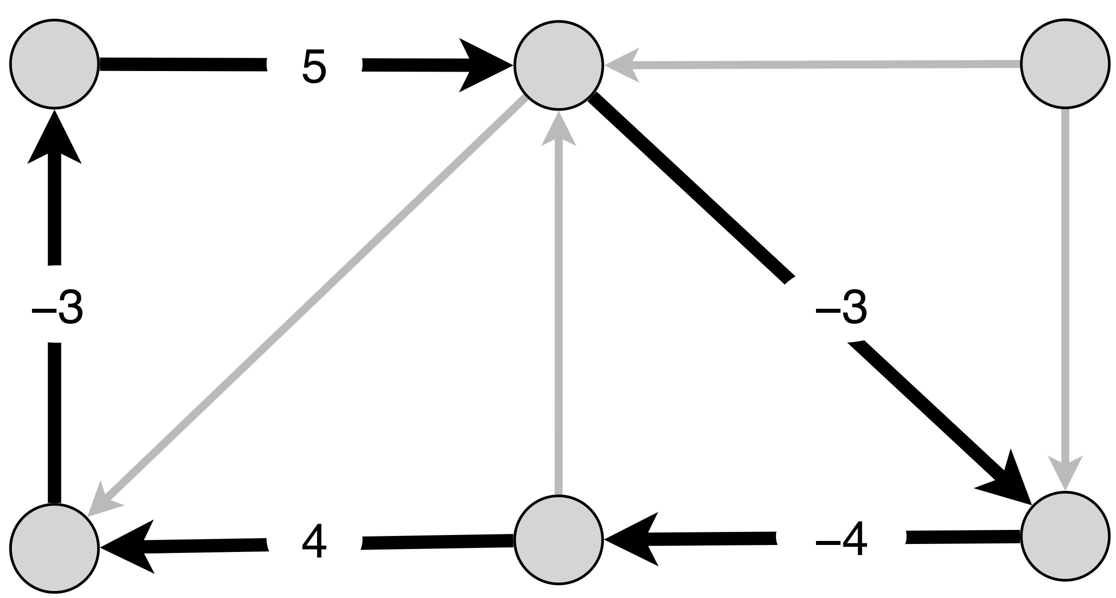
	0	1	2	3
s	0	0	0	0
a	inf	-3	-3	-3
b	inf	2	2	-1
c	inf	inf	-2	-2



Dynamic Programming Shortest Path: Detecting a Negative Cycle

Negative Cycle

- **Definition**. A negative cycle is a directed cycle C such that the sum of all the edge weights in C is less than zero
- **Claim.** If a path from s to some node v contains a negative cycle, then there does not exist a shortest path from s to v.



a negative cycle W : $\ell(W) = \sum_{e \in W} \ell_e < 0$

Recap and Problem

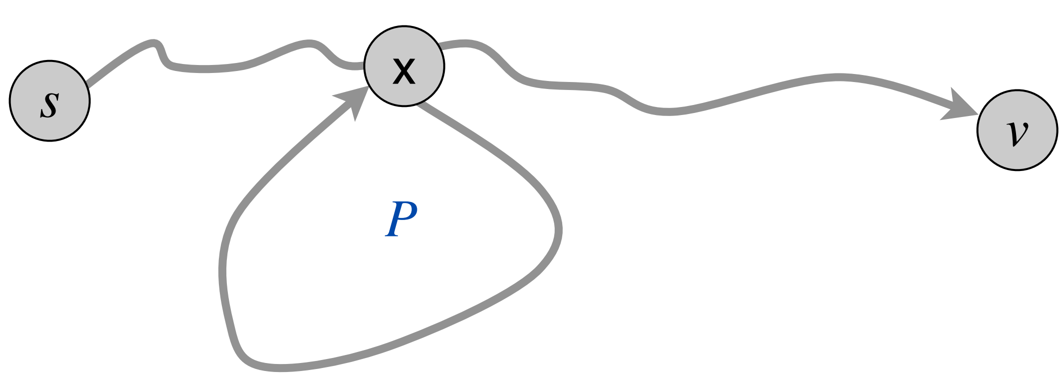
- **Summary.** Assuming there are no negative cycles in G, we can compute the shortest path from s to all nodes in G in $O(nm)$ time using the Bellman-Ford-Moore algorithm.
  - **Subproblem**. $D[v, i]$: Cost of shortest path from s to v using $\leq i$ edges
  - **Recurrence.**
    $$D[v, i] = \min\{D[v, i-1], \min_{(u,v) \in E} \{D[u, i-1] + w_{uv}\}\}$$
- **Question.** Given a directed graph $G = (V, E)$ with edge-weights w_e (can be negative), determine if G contains a negative cycle.
- We reduce this to a slightly different problem and will use Bellman-Ford-Moore algorithm to solve it

Detecting a Negative Cycle

- **Problem**. Given G and source s, find if there is negative cycle on a $s \rightsquigarrow v$ path for any node v.
- $D[v, i]$ is the cost of the shortest path from s to v of length at most i
- Suppose there is a negative cycle on a $s \rightsquigarrow v$ path
    - Then $\lim_{i \to \infty} D[v, i] = -\infty$
- If $D[v, n] = D[v, n-1]$ for every node v then G has no negative cycles exists! Why?
    - Table values converge, no further improvements possible

Detecting a Negative Cycle

- **Lemma.** If $D[v, n] < D[v, n-1]$ then any shortest $s \rightsquigarrow v$ path contains a negative cycle.
- **Proof**. [By contradiction] Suppose G does not contain a negative cycle
- Since $D[v, n] < D[v, n-1]$, the shortest $s \rightsquigarrow v$ path has exactly n edges
- By pigeonhole principle, path must contain a repeated node, let the cycle between two successive visits to the node be P
- If P has non-negative weight, removing it would give us a shortest path with less than n edges $\Rightarrow\!\Leftarrow$



Problem Reduction

- Now we know how to detect negative cycles on a shortest path from s to some node v.
- **Reduction**. Given graph G, add a source s and connect it to all vertices in G with edge weight 0. Let the new graph be G'
- **Claim**. G has a negative cycle iff G' has a negative cycle from s to some node v.
- **Proof**. $\Rightarrow$ If G has a negative cycle, then this cycle lies on the shortest path from s to a node on the cycle in G'
- $\Leftarrow$ If G' has a negative cycle on a shortest path from s to some node, then that node is on a negative cycle in G

Acknowledgments

- Some of the material in these slides are taken from
  - Kleinberg Tardos Slides by Kevin Wayne (https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/04GreedyAlgorithmsI.pdf)
  - Jeff Erickson's Algorithms Book (http://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf)