CS3000: Algorithms & Data
Paul Hand

Lecture 18:

- Bellman-Ford Algorithm

Apr 1, 2019

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
  - $d[u]$ is the length of the shortest $s \rightsquigarrow u$ path
  - $p[u]$ is the final hop on shortest $s \rightsquigarrow u$ path
- **Running time:** $O(m \log n)$
  - Implement using **binary heaps**

What 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

Activity

- Suppose we try the following algorithm
  - Take a graph $G = (V, E, \{\ell(e)\})$ with negative lengths
  - Add some valueC to all lengths to make them non-negative
  - Run Dijkstra on the new graph
- Activity: Come up with a graph where this fails

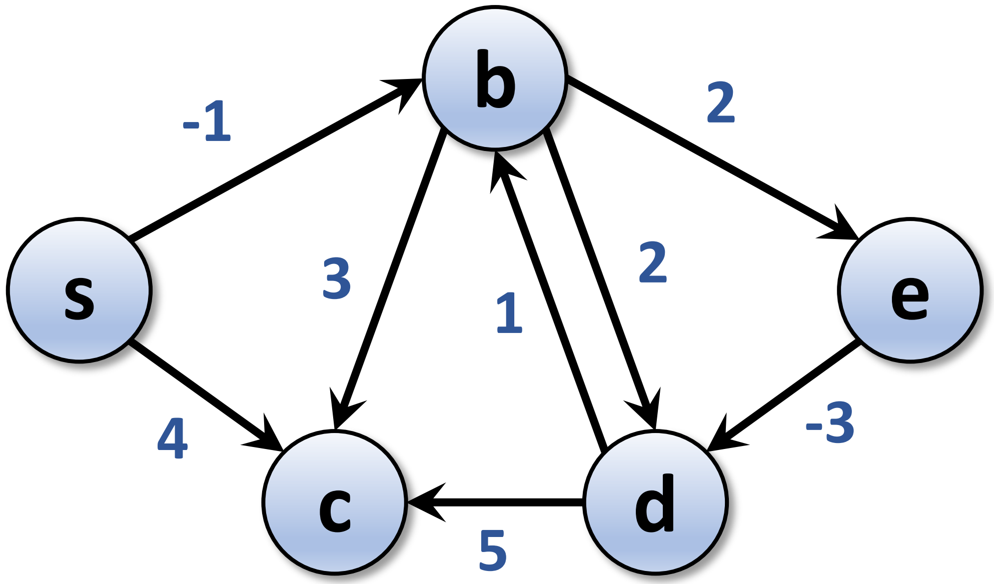
Structure of Shortest Paths

- 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

Bottom-Up Implementation?



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

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

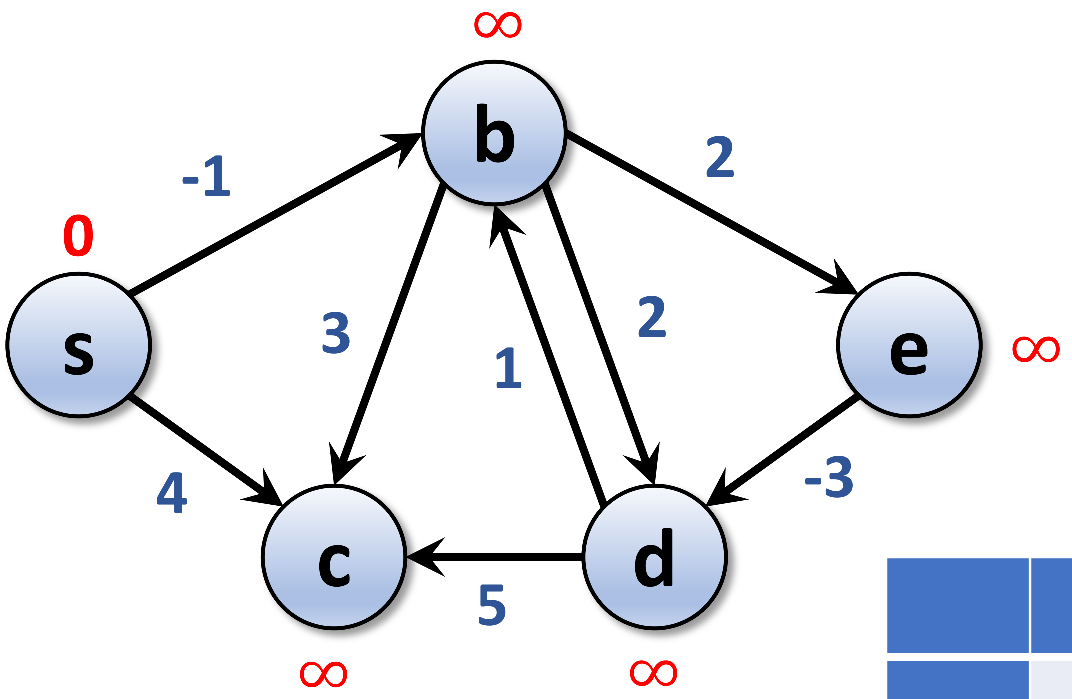
Finding the paths

- $\mathrm{OPT}(v, j)$ 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

Recurrence:

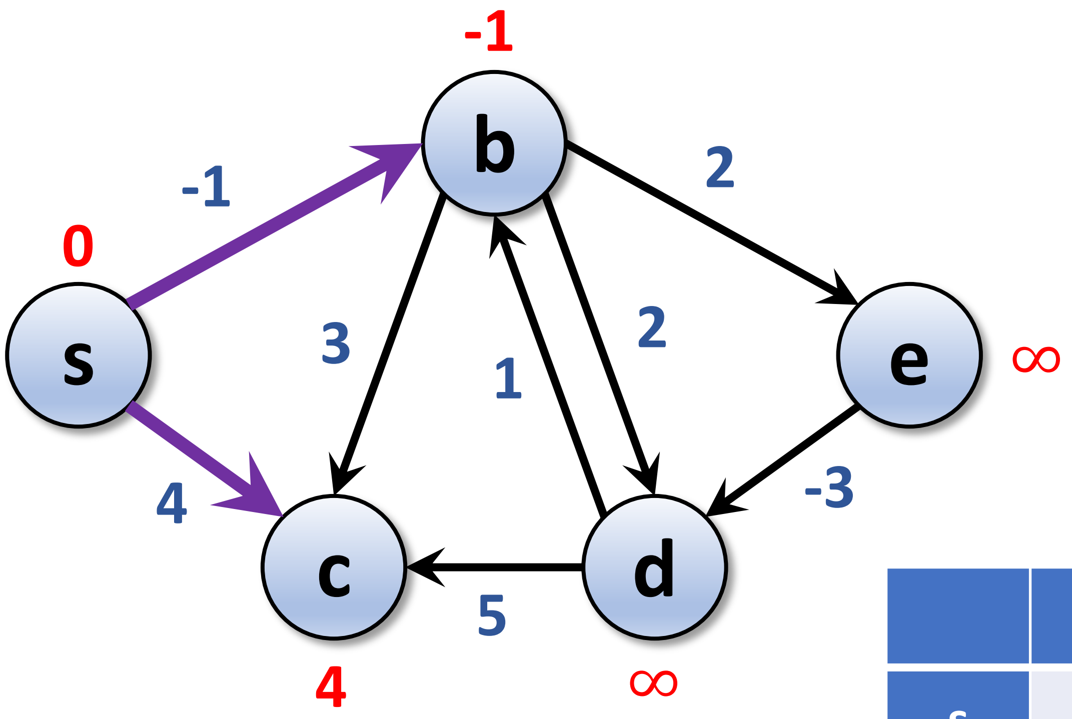
$$\mathrm{OPT}(v, j) = \min\left\{\mathrm{OPT}(v, i-1), \min_{(u,v)\in E}\{\mathrm{OPT}(u, i-1) + \ell_{u,v}\}\right\}$$

Example



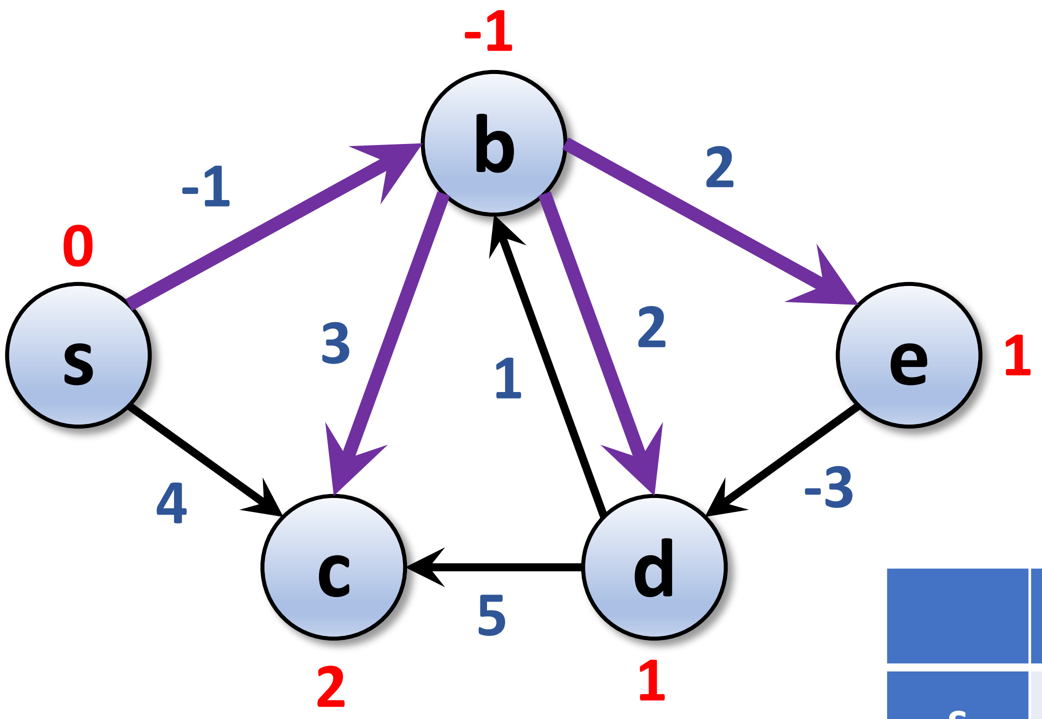
	0	1	2	3	4
s	0	0	0	0	0
b	∞				
c	∞				
d	∞				
e	∞				

Example



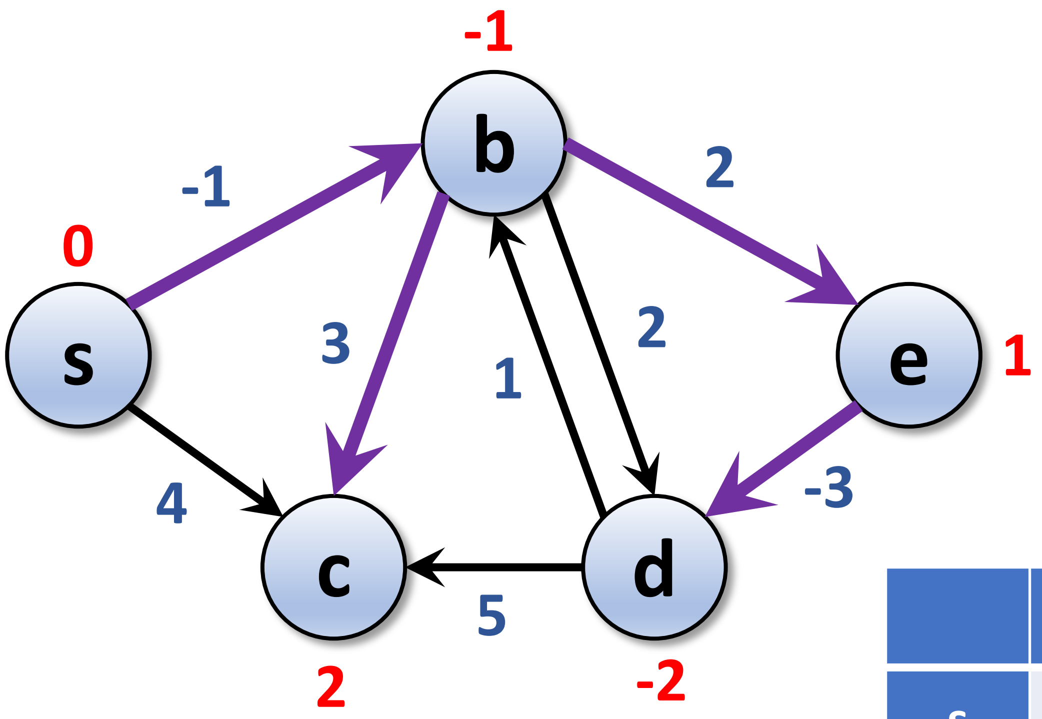
	0	1	2	3	4
s	0	0	0	0	0
b	∞	-1			
c	∞	4			
d	∞	∞			
e	∞	∞			

Example



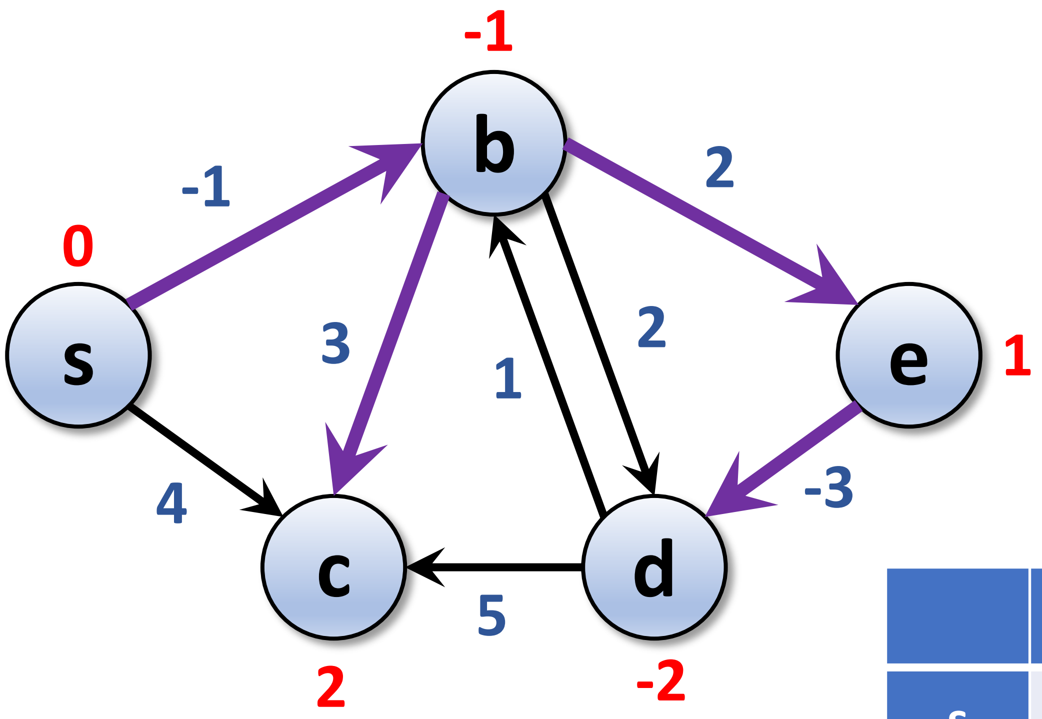
	0	1	2	3	4
s	0	0	0	0	0
b	∞	-1	-1		
c	∞	4	2		
d	∞	∞	1		
e	∞	∞	1		

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



	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)

```
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
```

Running time:
Space:

Optimizations

- 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

Implementation II

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

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