CS3000: Algorithms & Data
Paul Hand

Lecture 16:

- Shortest Paths and Dijkstra's Algorithm
- Correctness of Dijkstra's Algorithm

Mar 18, 2019

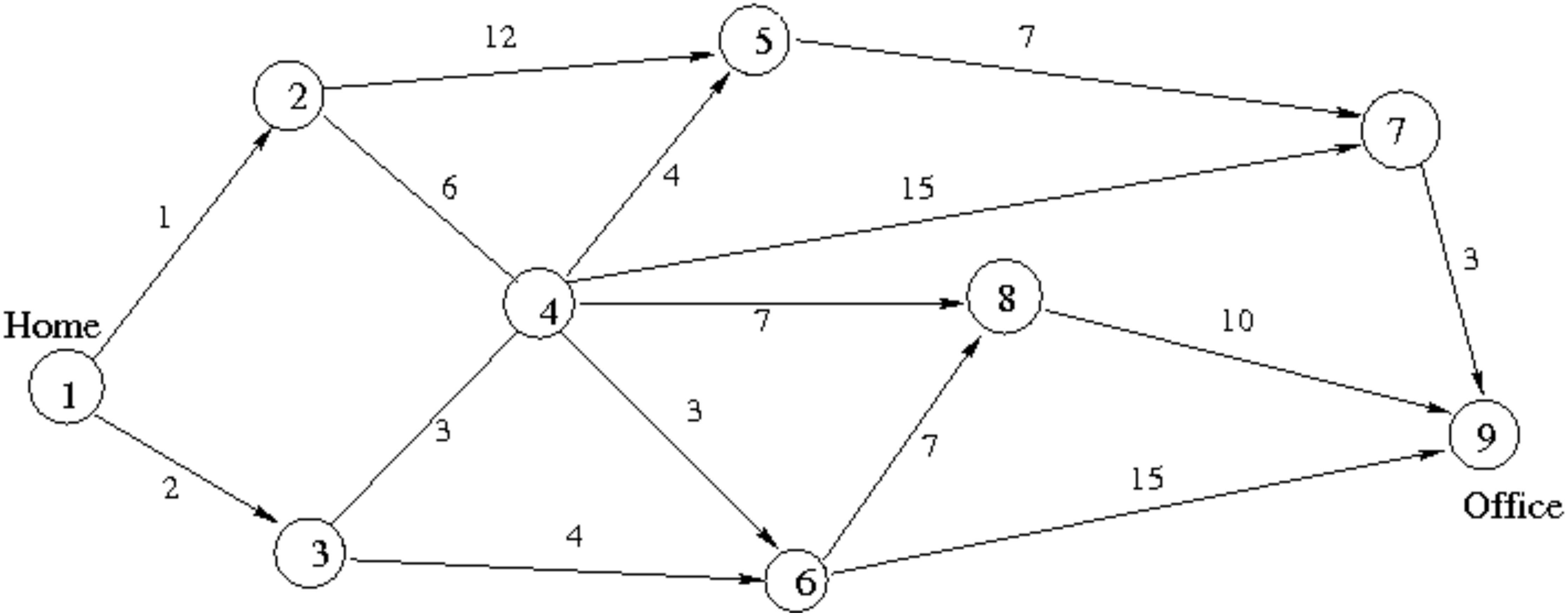
Shortest Paths

Weighted Graphs

- **Definition:** A weighted graph $G = (V, E, \{w(e)\})$
  - V is the set of vertices
  - $E \subseteq V \times V$ is the set of edges
  - $w_e \in \mathbb{R}$ are edge weights/lengths/capacities
  - Can be directed or undirected

- **Today:**
  - Directed graphs (one-way streets)
  - Strongly connected (there is always some path)
  - Non-negative edge lengths ($\ell(e) \geq 0$)

Activity: Find the shortest path



Shortest Paths

- The length of a path $P = v_1 - v_2 - \cdots - v_k$ is the sum of the edge lengths

- The distance $d(s, t)$ is the length of the shortest path from s to t
- **Shortest Path:** given nodes $s, t \in V$, find the shortest path from s to t
- **Single-Source Shortest Paths:** given a node $s \in V$, find the shortest paths from s to **every** $t \in V$

Structure of Shortest Paths

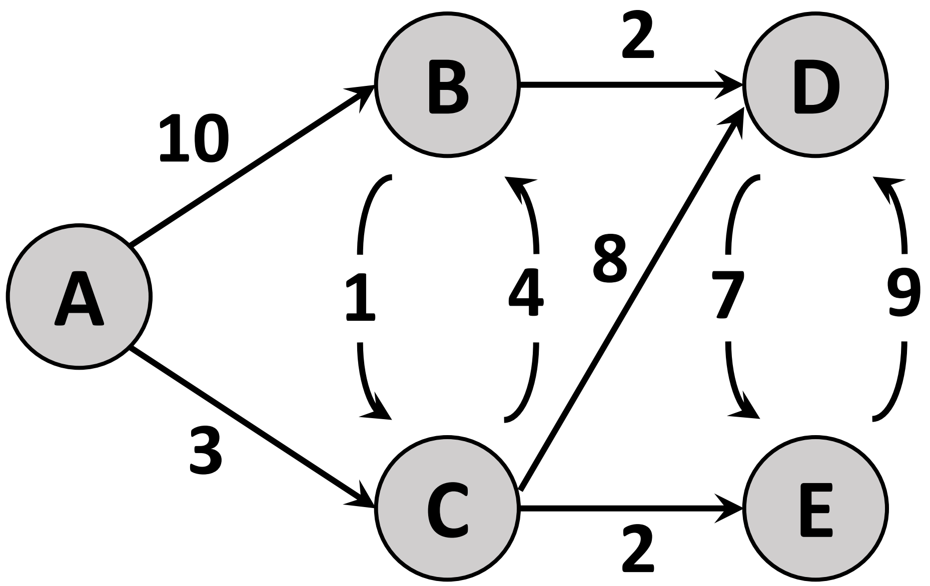
- If $(u, v) \in E$, then $d(s, v) \leq d(s, u) + \ell(u, v)$ for every node $s \in V$

- 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)

Dijkstra’s Algorithm

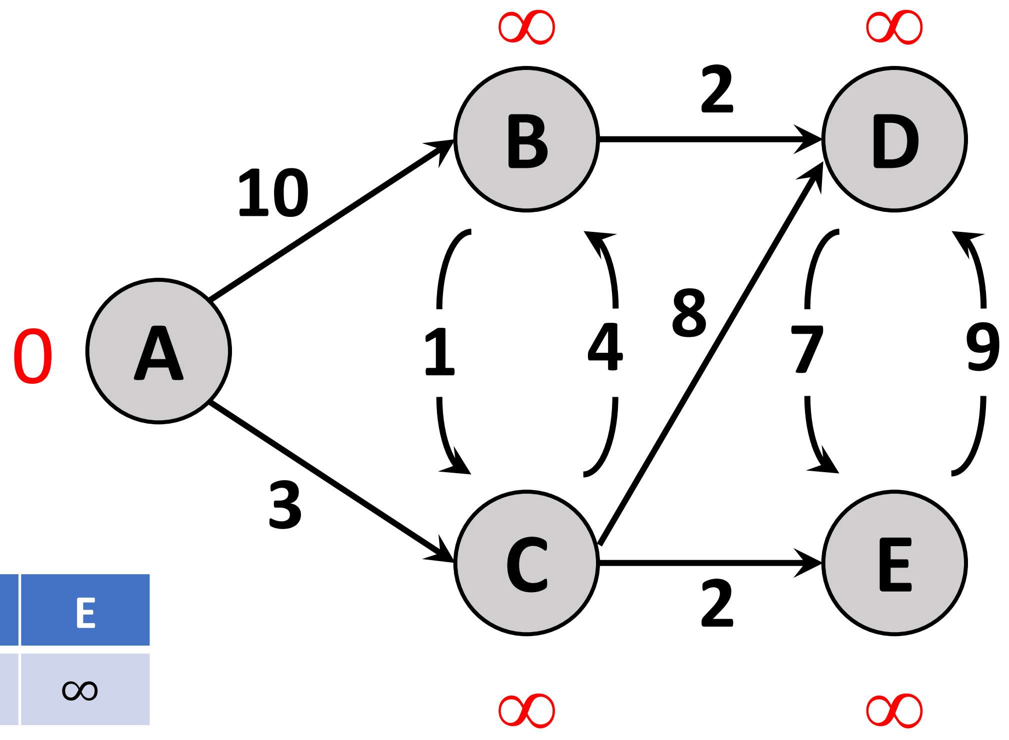
- **Dijkstra’s Shortest Path Algorithm** is a modification of BFS for non-negatively weighted graphs
- **Informal Version:**
    - **Maintain a set S of explored nodes**
    - **Maintain an upper bound on distance**
        - If u is explored, then we know $d(u)$ (Key Invariant)
        - If u is explored, and (u, v) is an edge, then we know $d(v) \leq d(u) + \ell(u, v)$
    - **Explore the “closest” unexplored node**
    - **Repeat until we’re done**

Dijkstra's Algorithm: Demo



Dijkstra’s Algorithm: Demo

Initialize

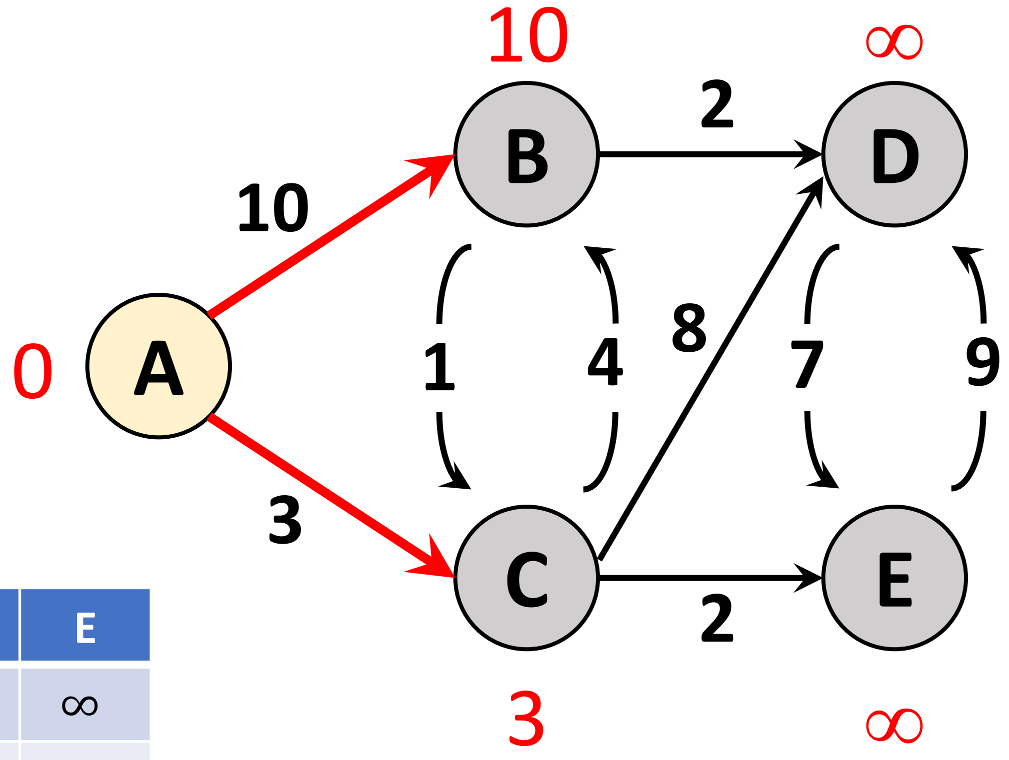


	A	B	C	D	E
$d_0(u)$	**0**	∞	∞	∞	∞

$$S = \{\}$$

Dijkstra’s Algorithm: Demo

Explore A

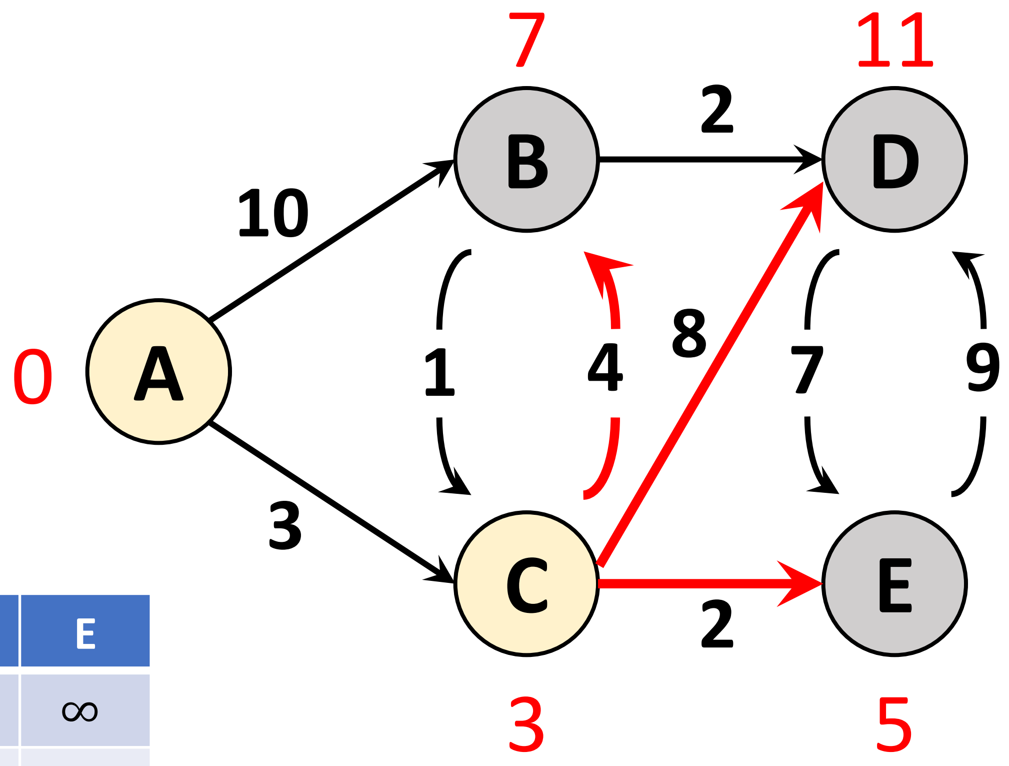


	A	B	C	D	E
$d_0(u)$	**0**	∞	∞	∞	∞
$d_1(u)$	**0**	**10**	**3**	∞	∞

$S = \{A\}$

Dijkstra's Algorithm: Demo

Explore C

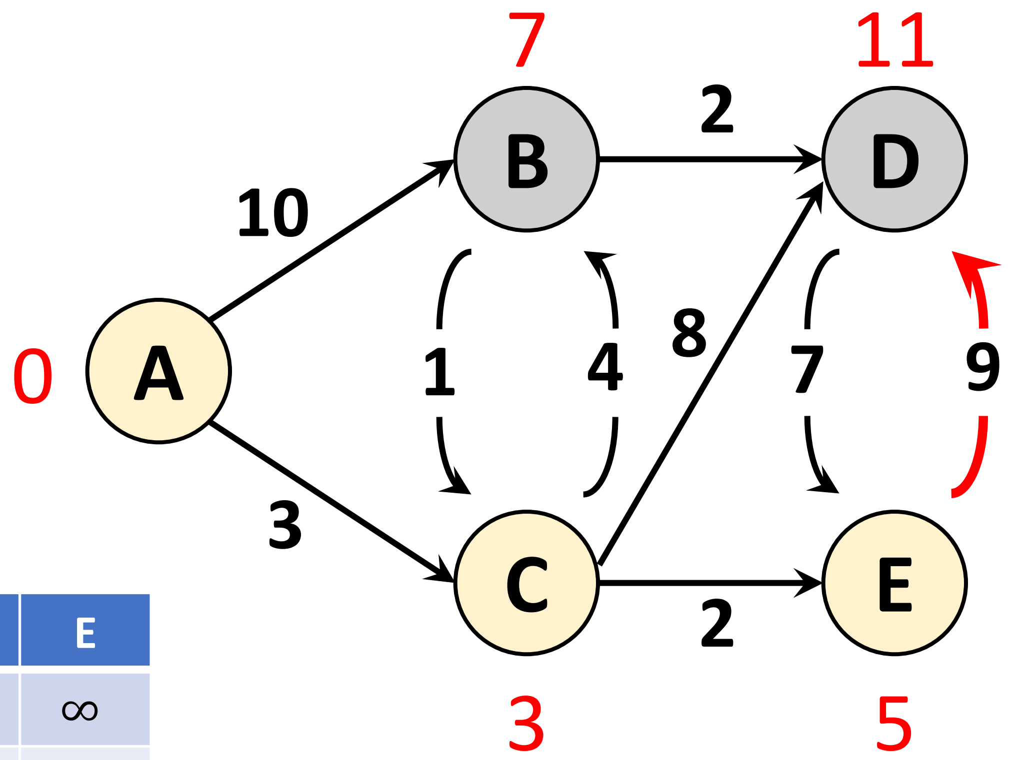


	A	B	C	D	E
$d_0(u)$	0	∞	∞	∞	∞
$d_1(u)$	0	10	3	∞	∞
$d_2(u)$	0	7	3	11	5

$$S = \{A, C\}$$

Dijkstra's Algorithm: Demo

Explore E

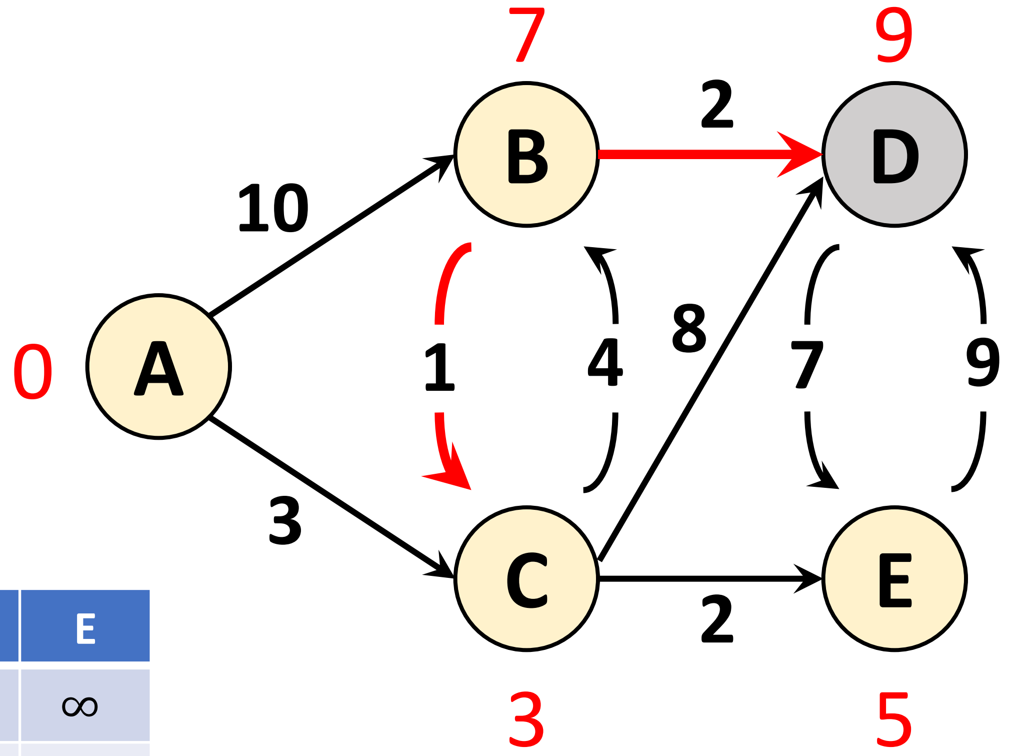


	A	B	C	D	E
$d_0(u)$	0	∞	∞	∞	∞
$d_1(u)$	0	10	3	∞	∞
$d_2(u)$	0	7	3	11	5
$d_3(u)$	0	7	3	11	5

$$S = \{A, C, E\}$$

Dijkstra's Algorithm: Demo

Explore B

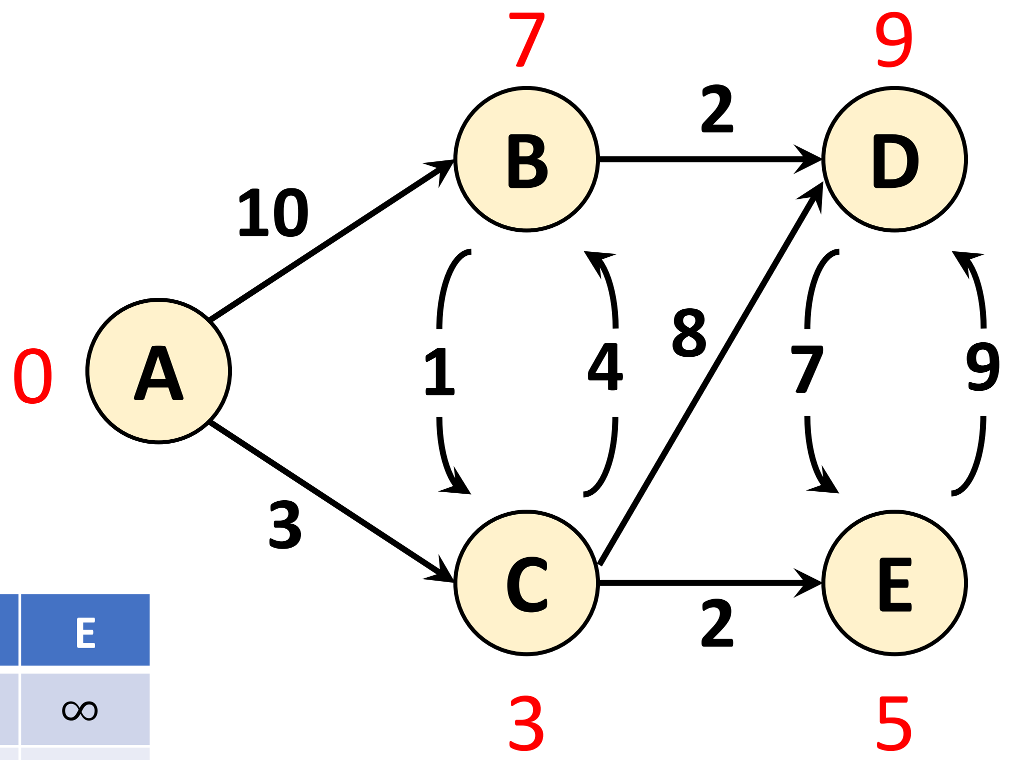


	A	B	C	D	E
$d_0(u)$	0	∞	∞	∞	∞
$d_1(u)$	0	10	3	∞	∞
$d_2(u)$	0	7	3	11	5
$d_3(u)$	0	7	3	11	5
$d_4(u)$	0	7	3	9	5

$$S = \{A, C, E, B\}$$

Dijkstra’s Algorithm: Demo

Don’t need to explore D

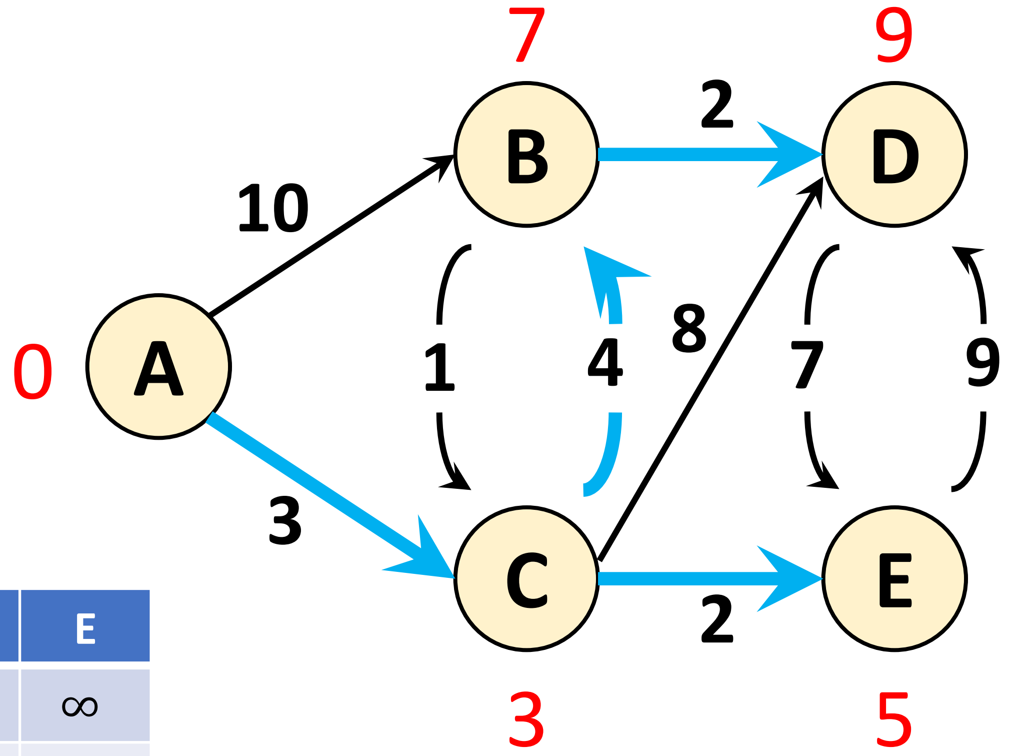


	A	B	C	D	E
$d_0(u)$	0	∞	∞	∞	∞
$d_1(u)$	0	10	3	∞	∞
$d_2(u)$	0	7	3	11	5
$d_3(u)$	0	7	3	11	5
$d_4(u)$	0	7	3	9	5

$$S = \{A, C, E, B, D\}$$

Dijkstra's Algorithm: Demo

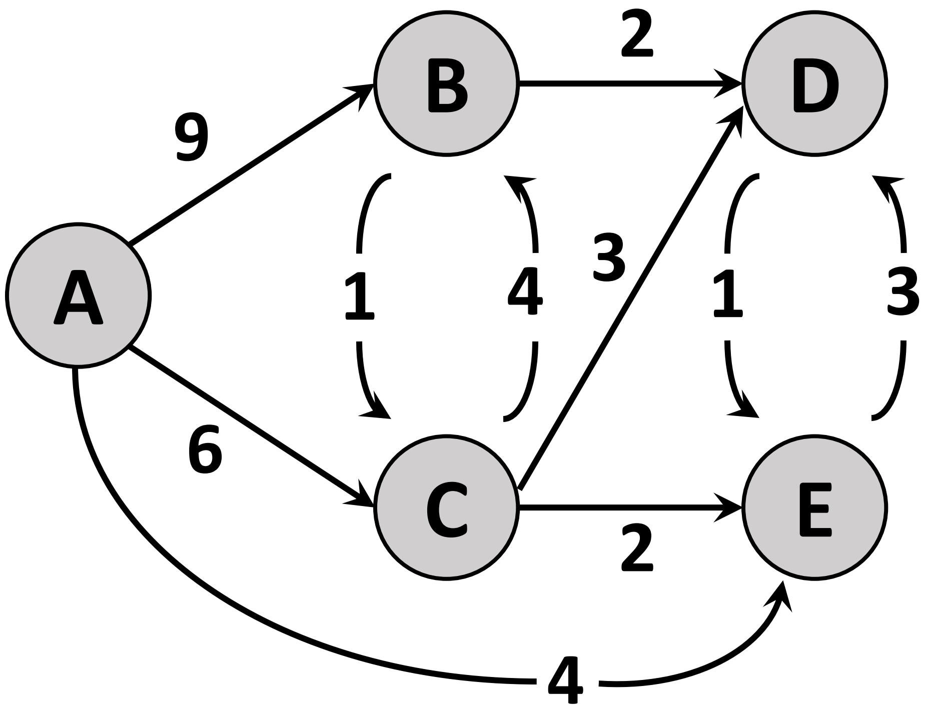
Maintain parent pointers so we can find the shortest paths



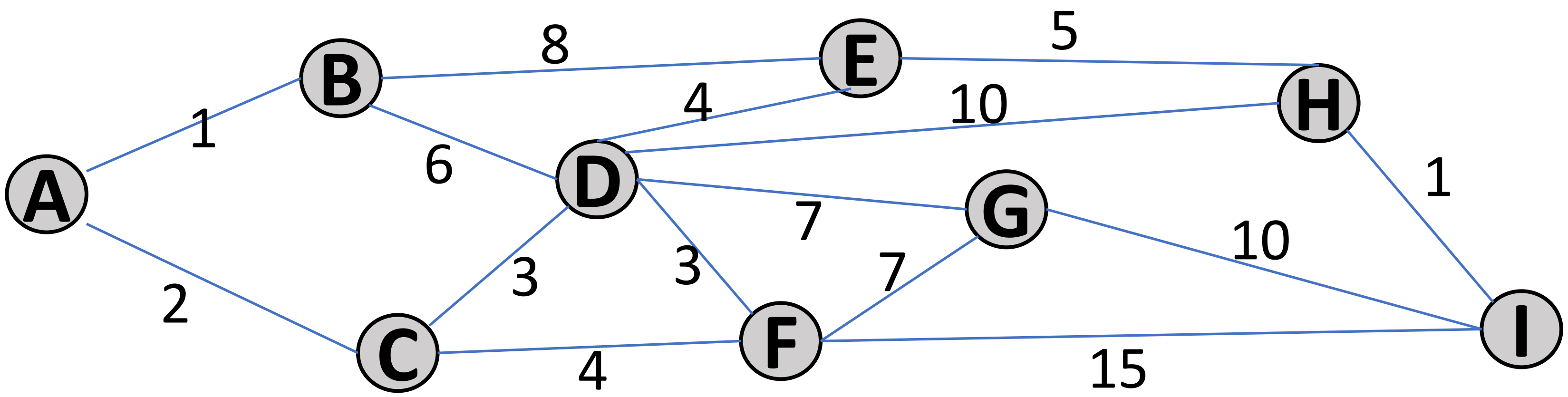
	A	B	C	D	E
$d_0(u)$	0	∞	∞	∞	∞
$d_1(u)$	0	10	3	∞	∞
$d_2(u)$	0	7	3	11	5
$d_3(u)$	0	7	3	11	5
$d_4(u)$	0	7	3	9	5

Execute Dijkstra's Algorithm: Activity

	A	B	C	D	E
$d_0(u)$	0	∞	∞	∞	∞
$d_1(u)$	0				
$d_2(u)$	0				
$d_3(u)$	0				
$d_4(u)$	0				
$d_5(u)$	0				



Execute Dijkstra's Algorithm: Activity



	A	B	C	D	E	F	G	H	I
$d_0(u)$	**0**	∞	∞	∞	∞	∞	∞	∞	∞
$d_1(u)$	**0**								
$d_2(u)$	**0**								
$d_3(u)$	**0**								
$d_4(u)$	**0**								
$d_5(u)$	**0**								
$d_6(u)$	**0**								
$d_7(u)$	**0**								
$d_8(u)$	**0**								
$d_9(u)$	**0**								

Correctness of Dijkstra

- **Warmup 0:** initially, $d_0(s)$ is the correct distance

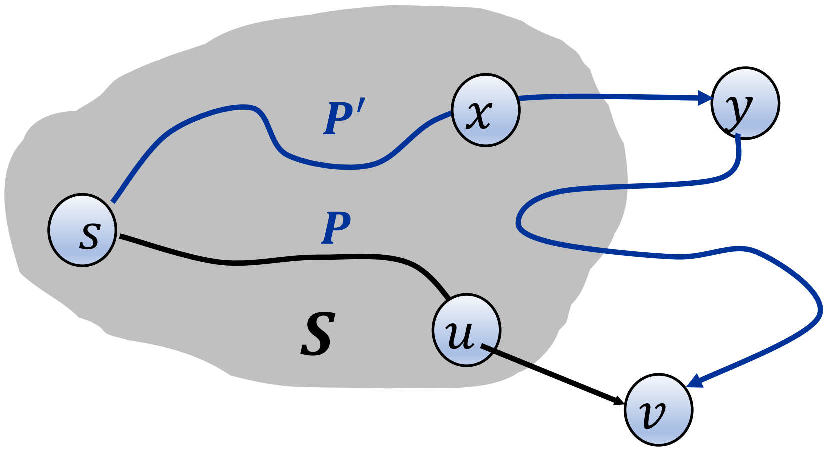
- **Warmup 1:** after exploring the second node v, $d_1(v)$ is the correct distance

Correctness of Dijkstra

- **Invariant:** after we explore the i-th node, $d_i(v)$ is correct for every $v \in S$

- We just argued the invariant holds after we've explored the 1st and 2nd nodes

Correctness of Dijkstra



- **Invariant:** after we explore the i-th node, $d_i(v)$ is correct for every $v \in S$
- **Proof:**

- There is some path P of length $\ell(P) = d_i(v)$
- Consider any other path P' from s to v
  - P' leaves S using some edge $x \to y$
  - The part of P' from s to x is a shortest path, has length $d_i(x)$

$$\ell(P') \geq (\text{distance to } x) + \ell(x \to y)$$
$$= d_i(x) + \ell(x \to y)$$
$$\geq d_i(y) \quad \text{[Because x was explored]}$$
$$\geq d_i(v) \quad \text{[Because we explored v, not y]}$$

- $\ell(P') \geq \ell(P)$

$\therefore$ P is a shortest path

$\therefore$ $d_i(v) = d(s, v)$ □

Implementing Dijkstra

```
Dijkstra(G = (V,E,{ℓ(e)}, s):
  d[s] ← 0, d[u] ← ∞ for every u != s
  parent[u] ←⊥ for every u
  Q ← V                    // Q holds the unexplored nodes

  While (Q is not empty):
    u ← argmin_{w∈Q} d[w]      //Find closest unexplored
    Remove u from Q

    // Update the neighbors of u
    For ((u,v) in E):
      If (d[v] > d[u] + ℓ(u,v)):
        d[v] ← d[u] + ℓ(u,v)
        parent[v] ← u

  Return (d, parent)
```

Implementing Dijkstra Naively

- Need to explore all n nodes
- Each exploration requires:
  - Finding the unexplored node u with smallest distance
  - Updating the distance for each neighbor of u
    - Lookup current distance
    - Possibly decrease distance