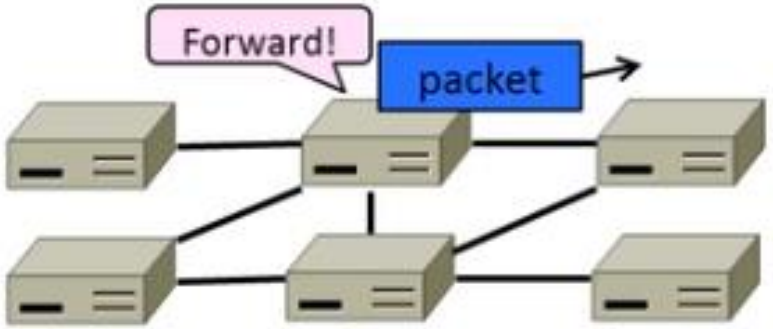
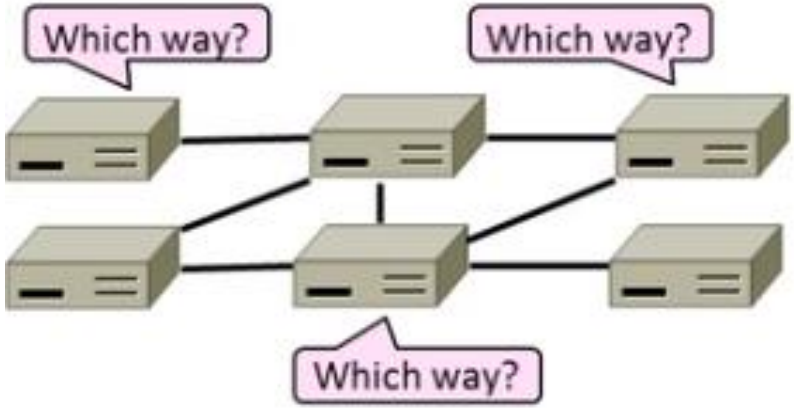
Routing

Routing versus Forwarding

- Forwarding is the process of sending a packet on its way



- Routing is the process of deciding in which direction to send traffic



Goals of Routing Algorithms

- We want several properties of any routing scheme:

Property	Meaning
Correctness	Finds paths that work
Efficient paths	Uses network bandwidth well
Fair paths	Doesn't starve any nodes
Fast convergence	Recovers quickly after changes
Scalability	Works well as network grows large

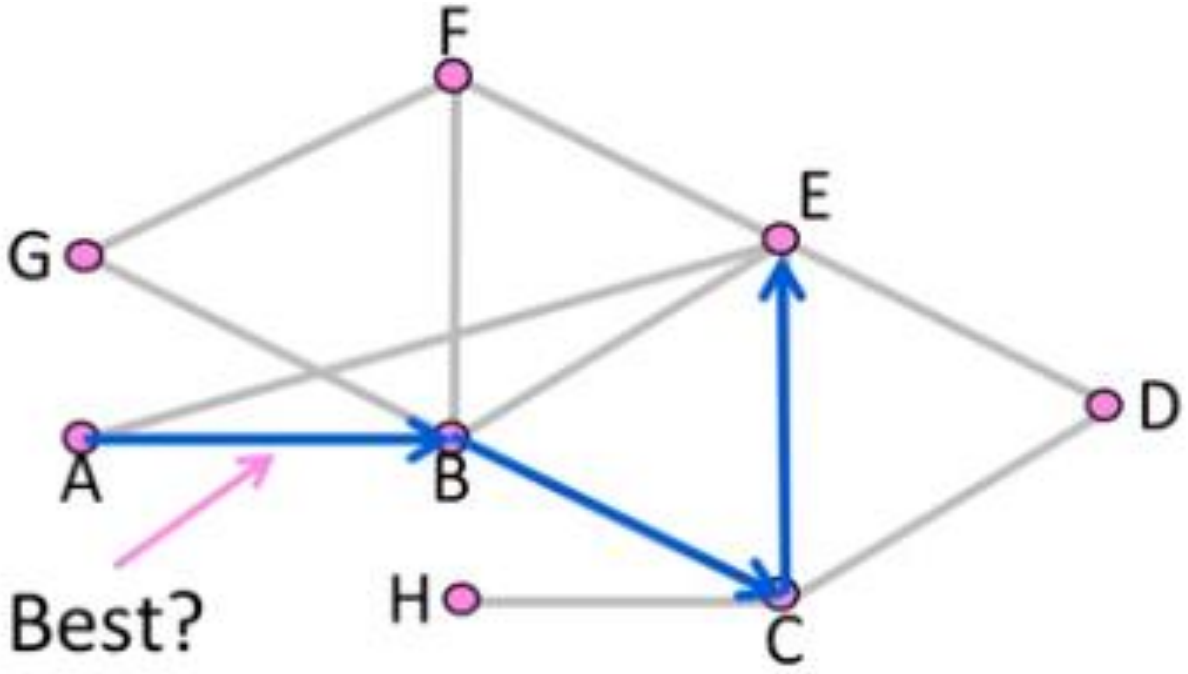
Rules for Routing Algorithms

- Decentralized, distributed setting
- All nodes are alike; no controller
- Nodes only know what they learn by exchanging messages with neighbors
- Nodes operate concurrently
- May be node/link/message failures

Shortest Path Routing

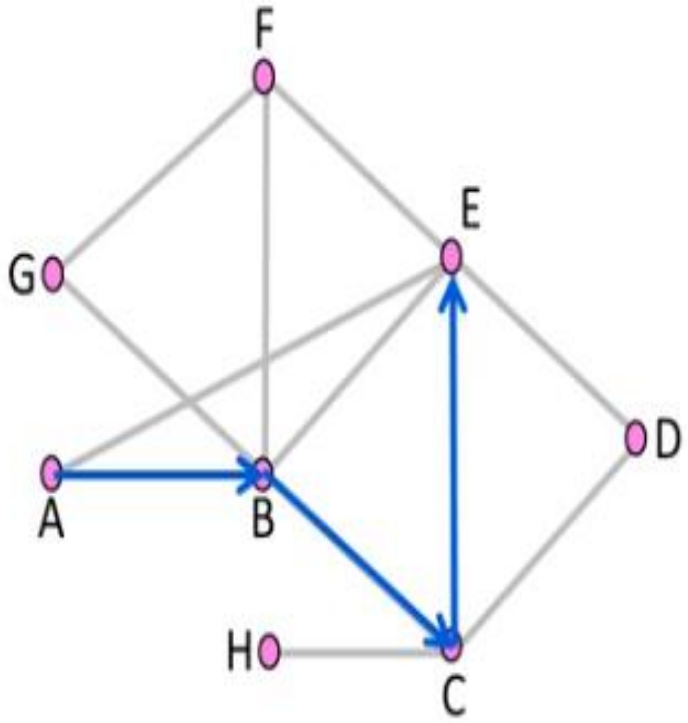
Overview

- Defining "best" paths with link costs
  - —These are shortest path routes



What are “Best” path anyhow?

- Many possibilities:
  - —Latency, avoid circuitous paths
  - —Bandwidth, avoid slow links
  - —Money, avoid expensive links
  - —Hops, to reduce switching
- But only consider topology
  - —Ignore workload, e.g., hotspots



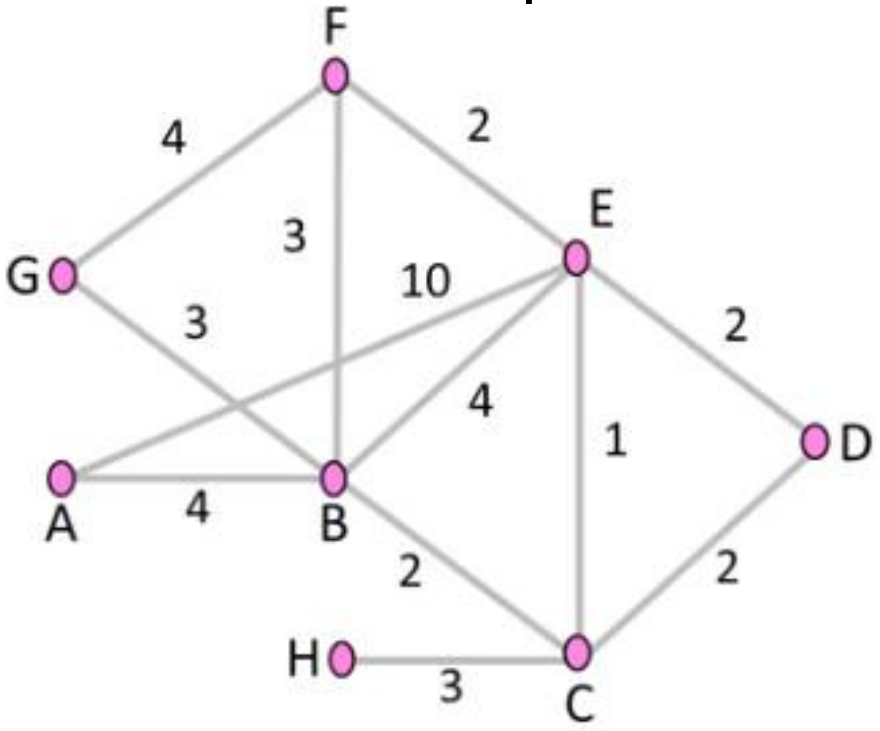
Shortest Paths

- We'll approximate "best" by a cost function that captures the factors
  - —Often called lowest "shortest"

1. Assign each link a cost (distance)
2. Define best path between each pair of nodes as the path that has the lowest total cost (or is shortest)
3. Pick randomly to any break ties

Shortest Paths (2)

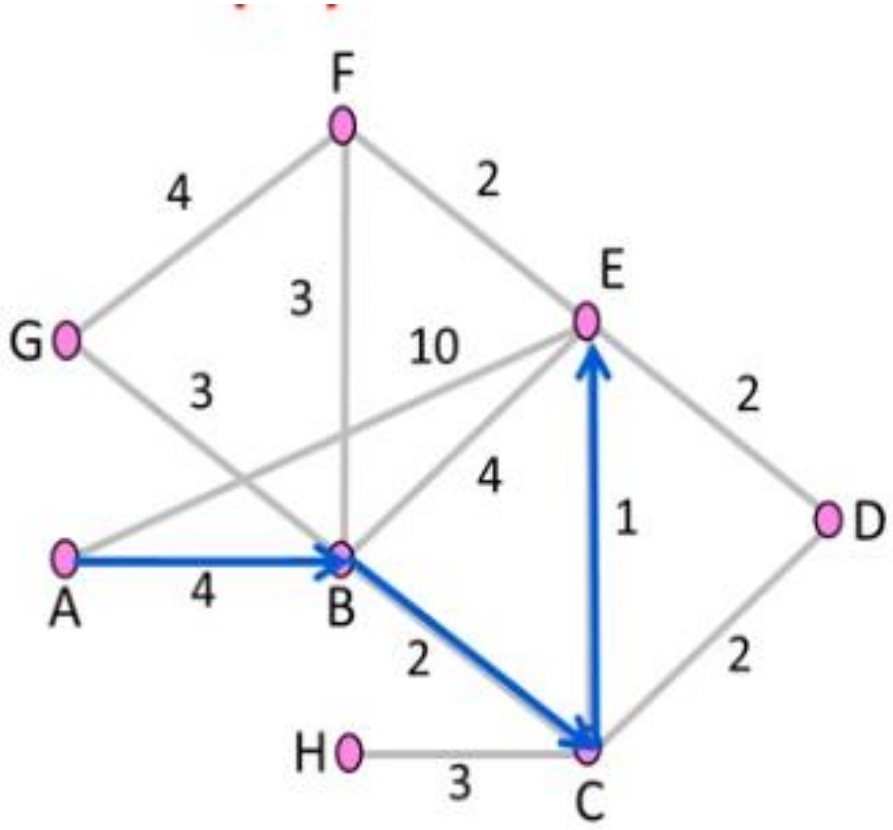
- Find the shortest path A ⟶ E
- All links are bidirectional, with equal costs in each direction
  - Can extend model to unequal costs if needed



Shortest Paths (3)

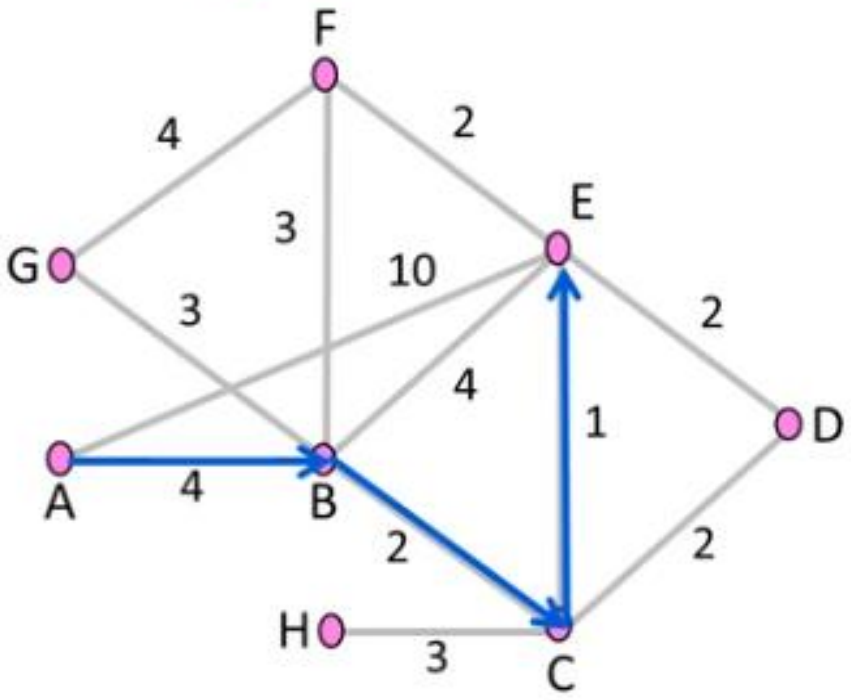
- ABCE is the shortest path
- dist(ABCE) = 4 + 2+ 1 = 7

- This is less than
- dist(ABE) = 8
- dist(ABFE) = 9
- dist(AE) = 10
- dist(ABCDE) = 10



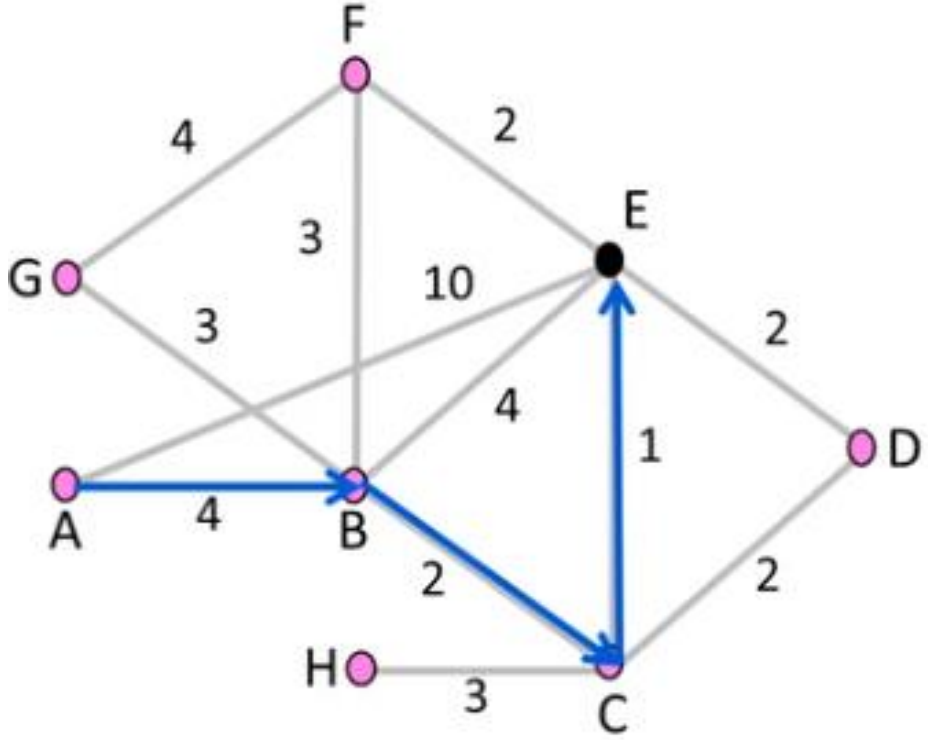
Shortest Paths (4)

- Optimality property:
  - —Subpaths of shortest paths are also shortest paths
- ABCE is a shortest path
  - —So are ABC, AB, BCE, BC, CE



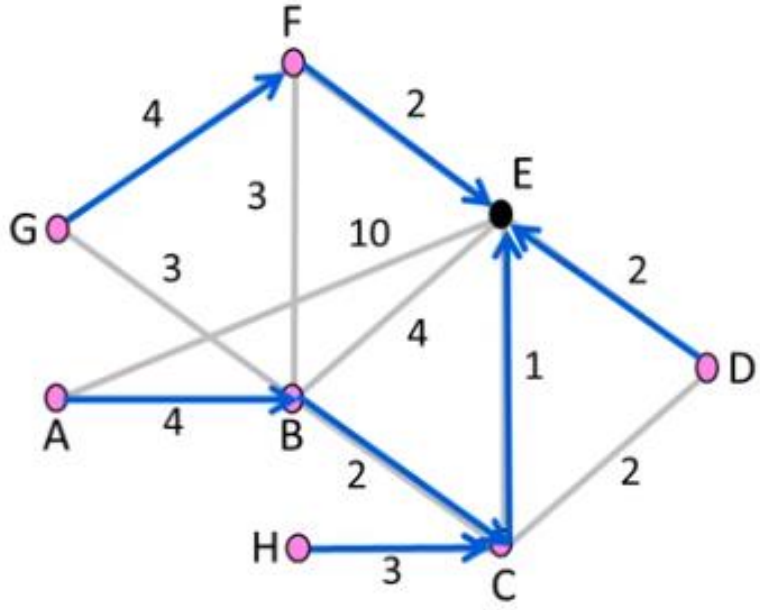
Sink Trees

- Sink tree for a destination is the union of all the shortest paths towards the destination
  - —Similarly source tree
- Find the sink tree for E



Sink Trees (2)

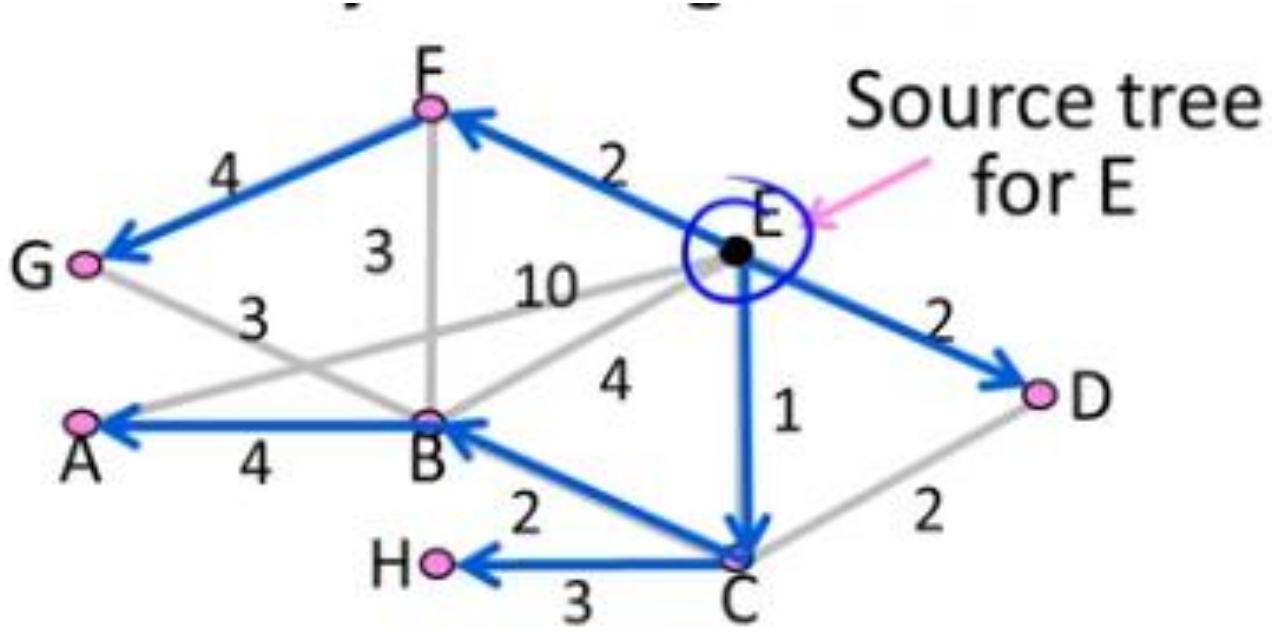
- Implications:
  - —Only need to use destination to follow shortest paths
  - —Each node only need to send the next hop
- Forwarding table at a node
  - —Lists next hop for each destination
  - —Routing table may know more



Computing Shortest Paths with Dijkstra

Overview

- How to compute the shortest paths given the network topology
  - With Dijkstra's algorithm



Edsger W. Dijkstra (1930-2002)

- Famous computer scientist
  - —Programming languages
  - —Distributed algorithms
  - —Program verification

- Dijkstra's algorithm, 1959
  - —Single-source paths, given network with non-negative link costs

By Hamilton Richards, CC-BY-SA-3.0, via Wikimedia Commons

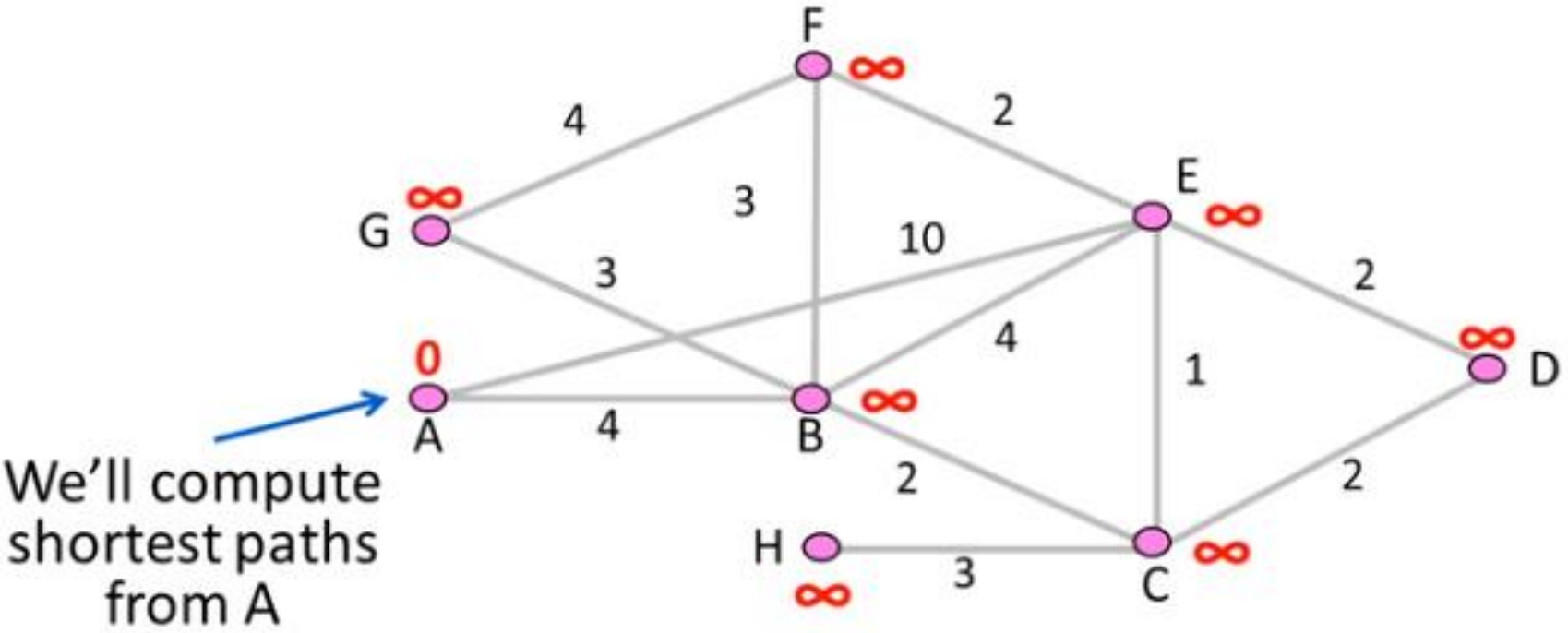
Dijkstra's Algorithm

- Mark all nodes tentative, set distances from source to 0 (zero) for source, and ∞ (infinity) for all other nodes.

- While tentative node remain:
  - Extract N, a node with lowest distance
  - Add link to N to the shortest path tree
  - Relax the distances of neighbors of n by lowering any better distance estimates

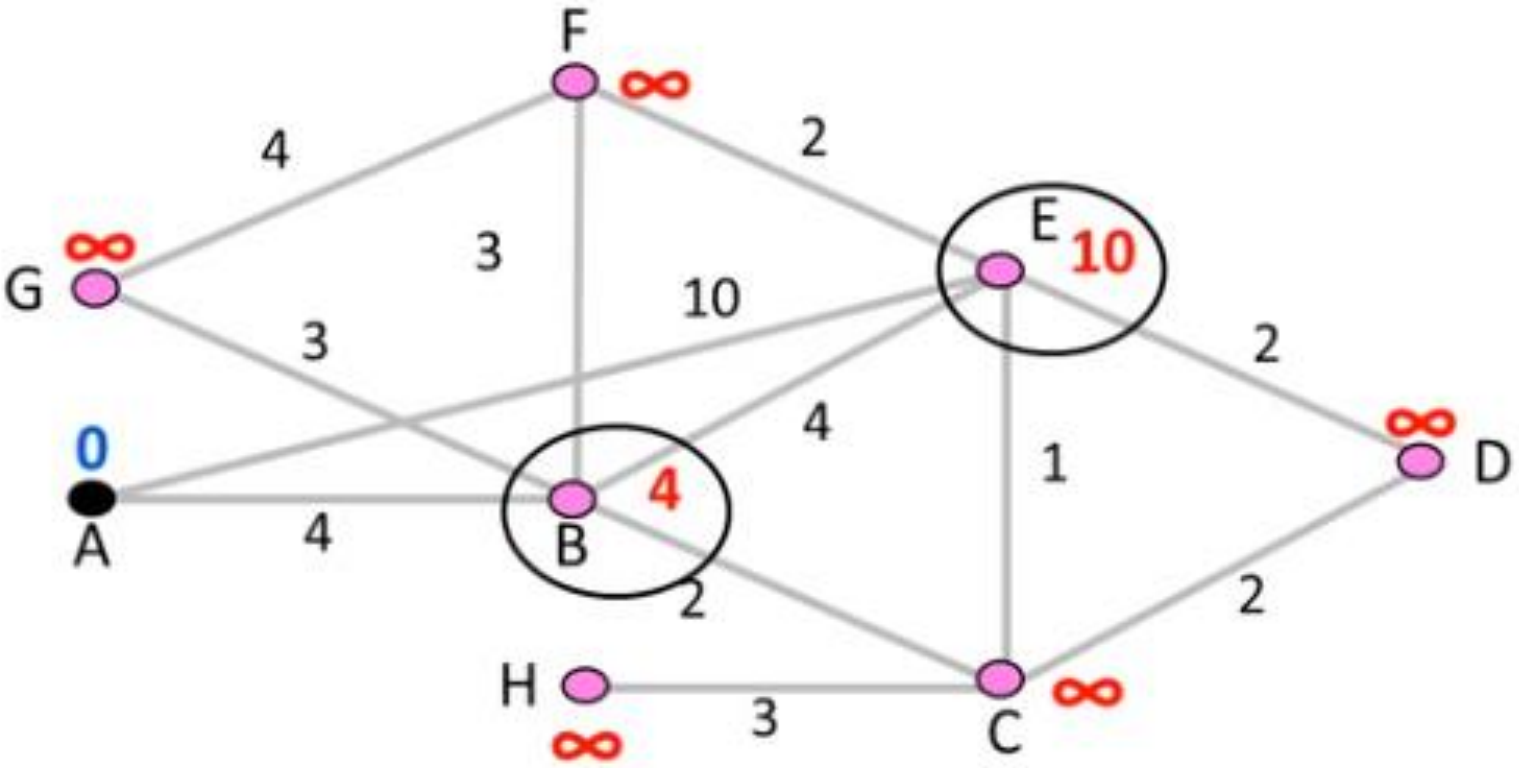
Dijkstra's Algorithm (2)

- Initialization



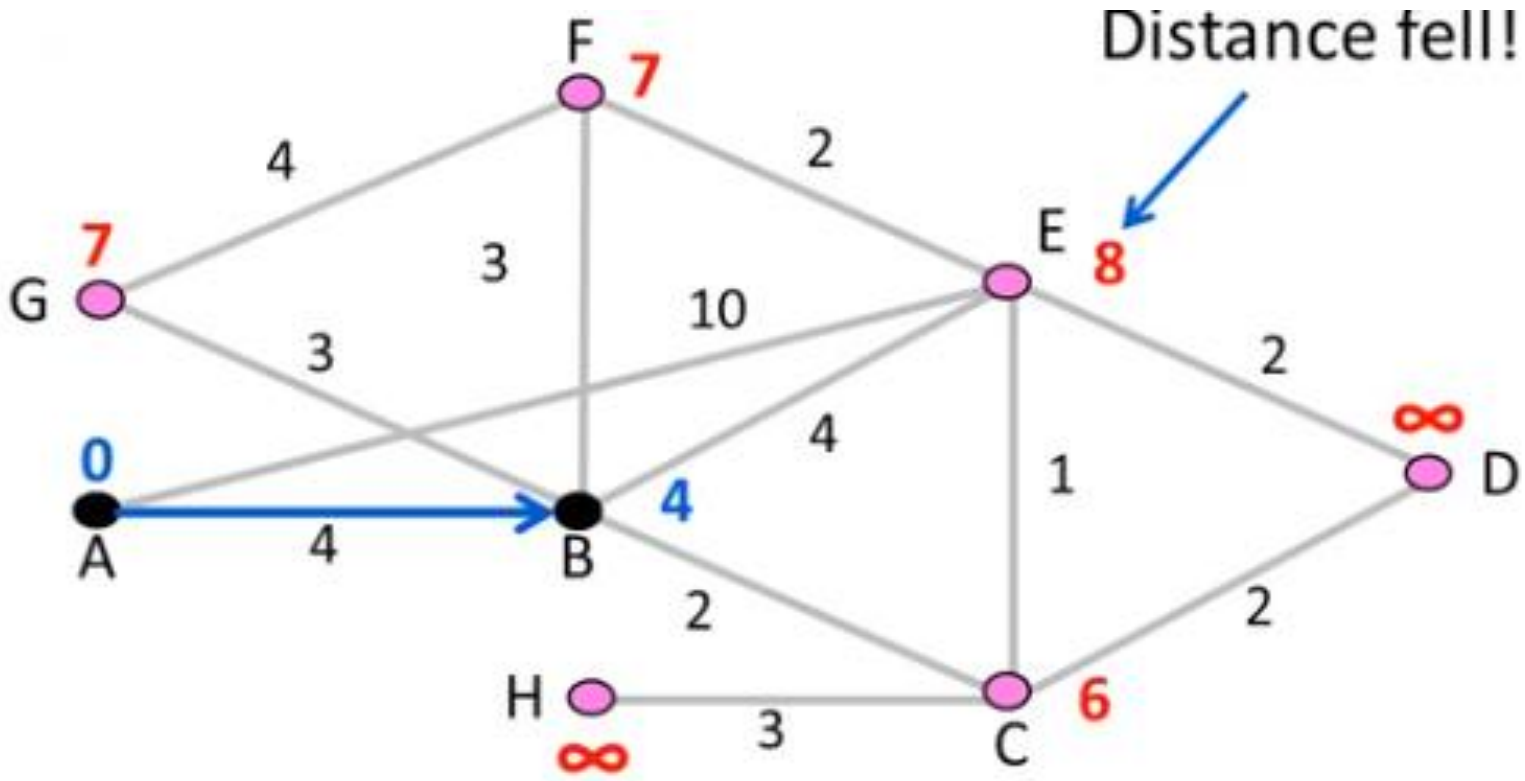
Dijkstra's Algorithm (3)

- Relax around A



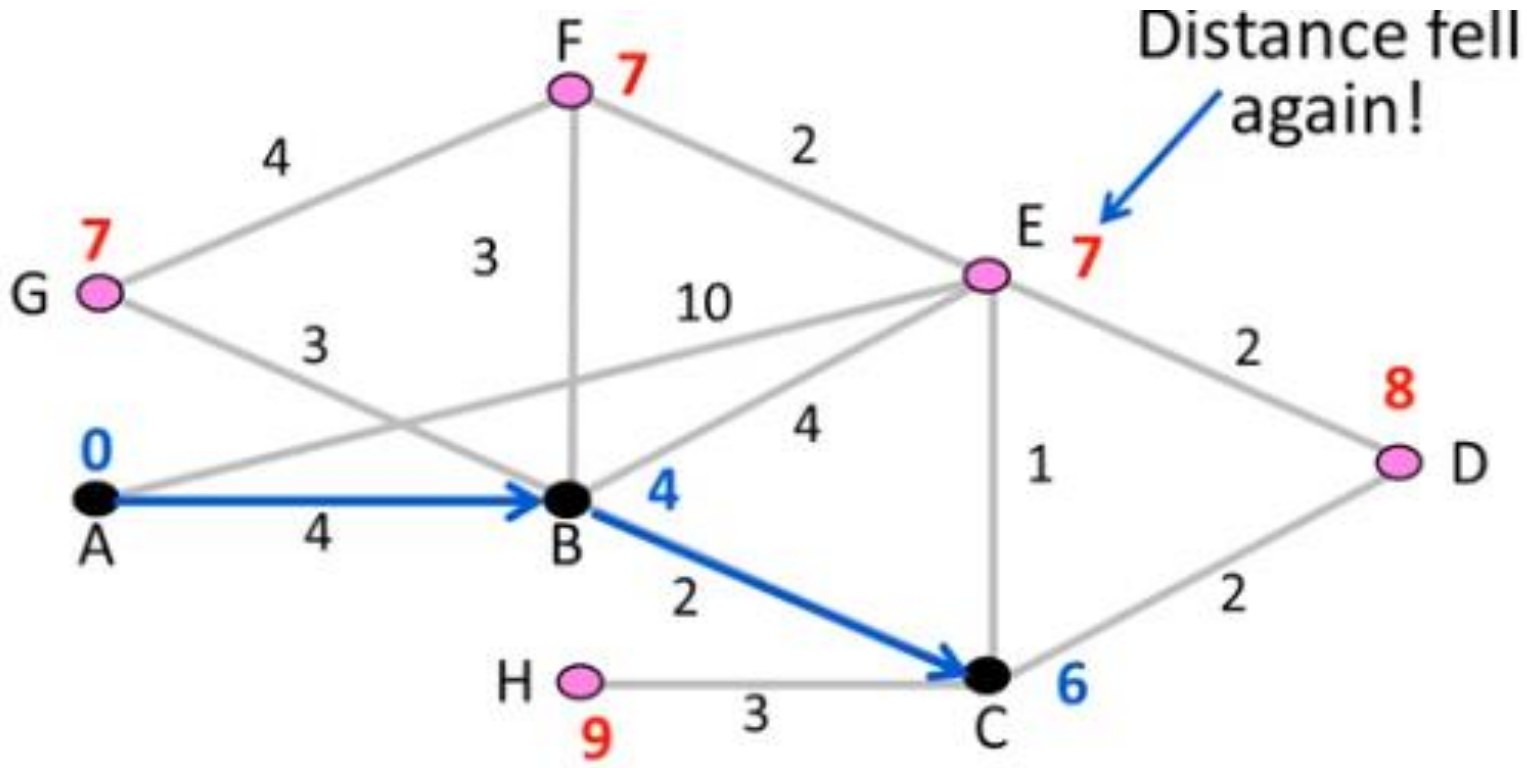
Dijkstra's Algorithm (4)

- Relax around B



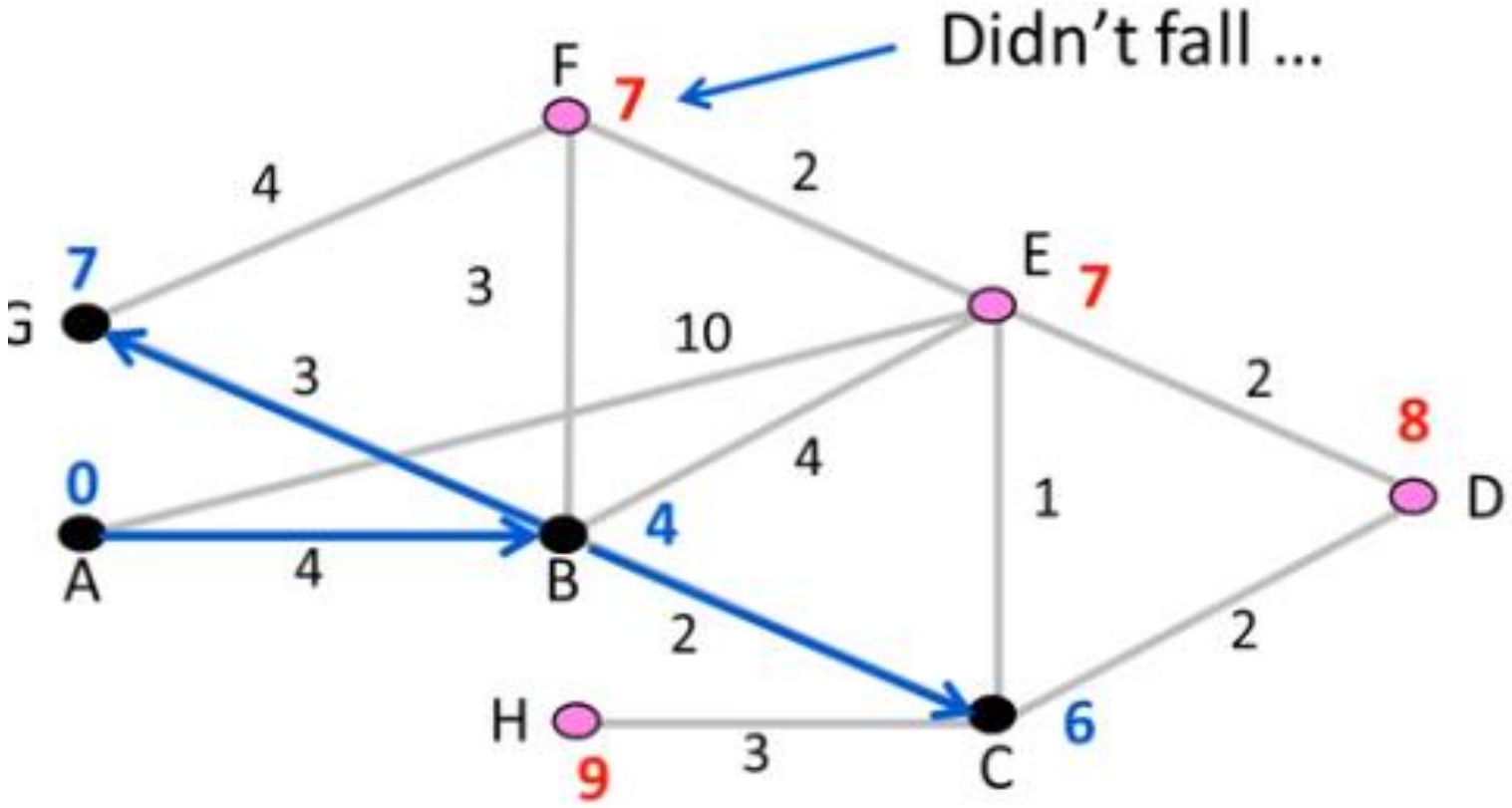
Dijkstra's Algorithm (5)

- Relax around C



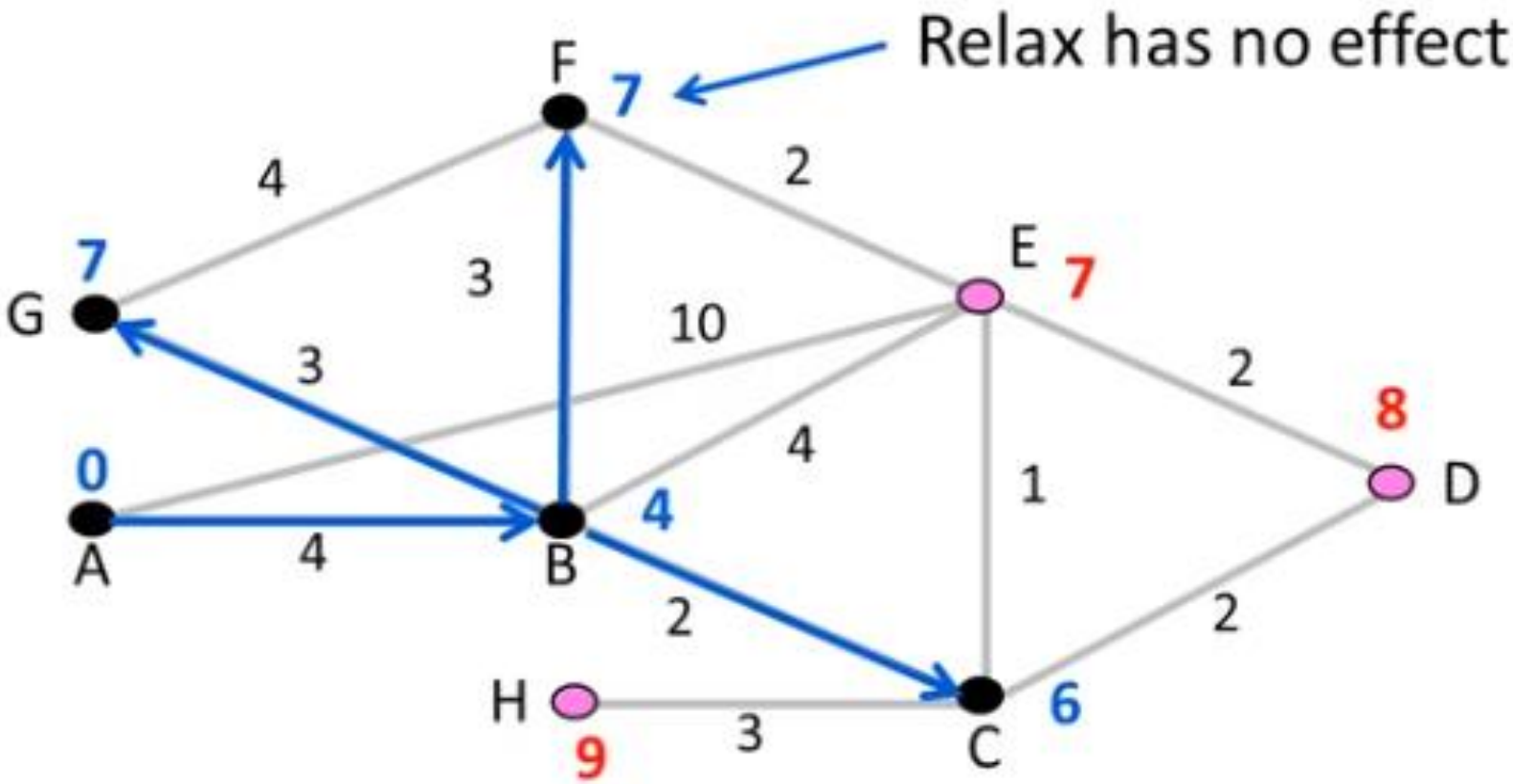
Dijkstra's Algorithm (6)

- Relax around G (say)



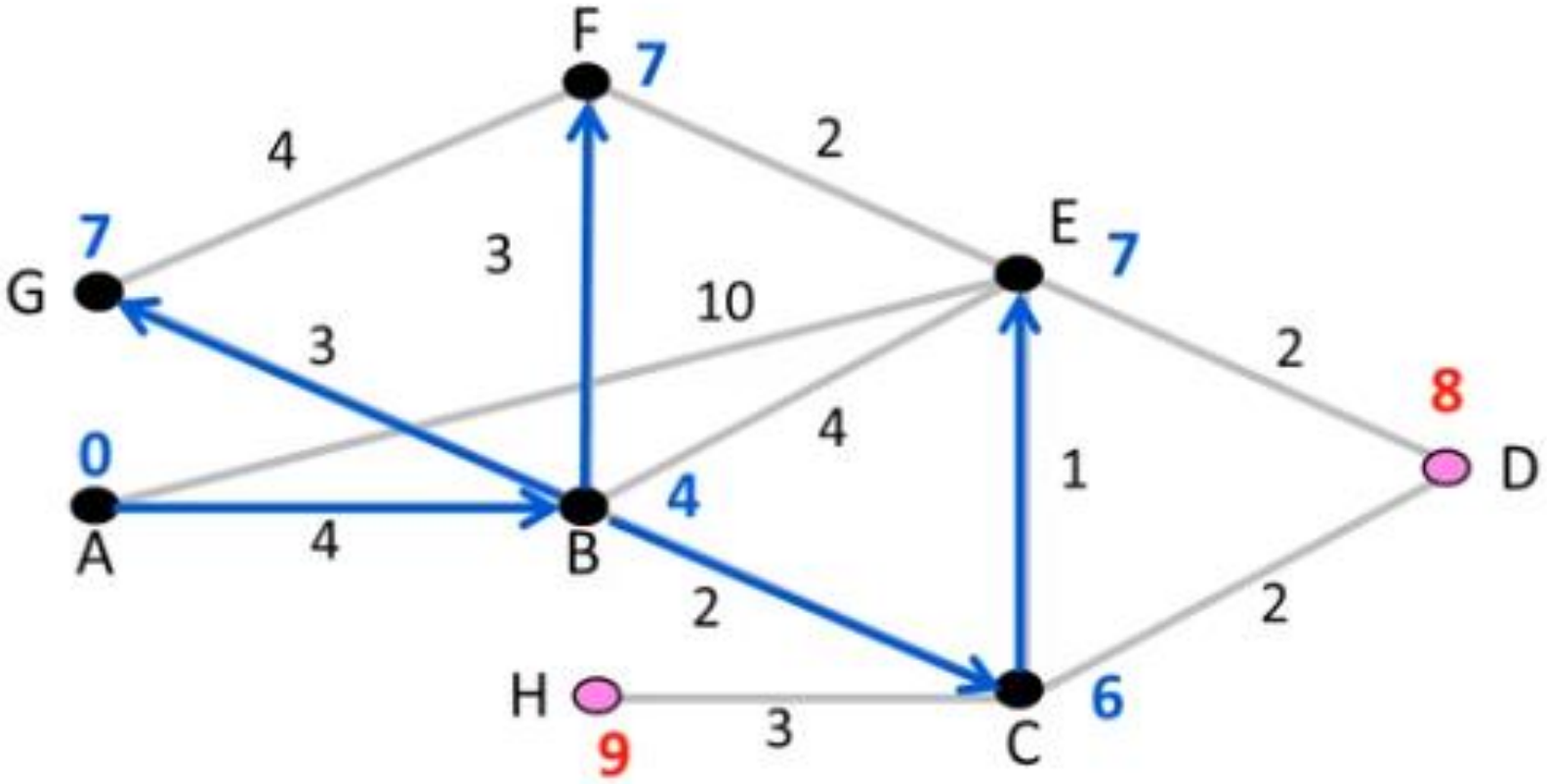
Dijkstra's Algorithm (7)

- Relax around F (say)



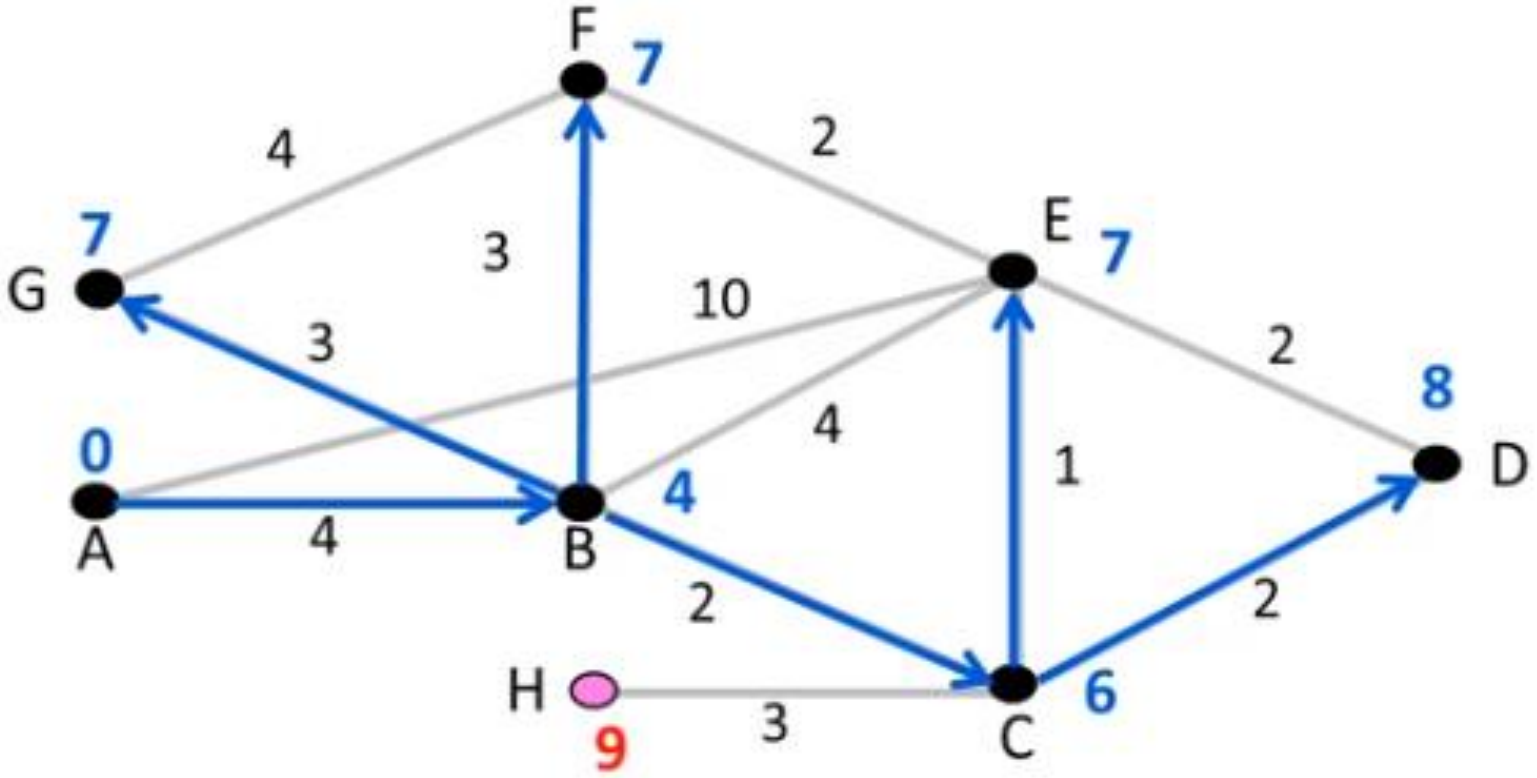
Dijkstra's Algorithm (8)

- Relax around E



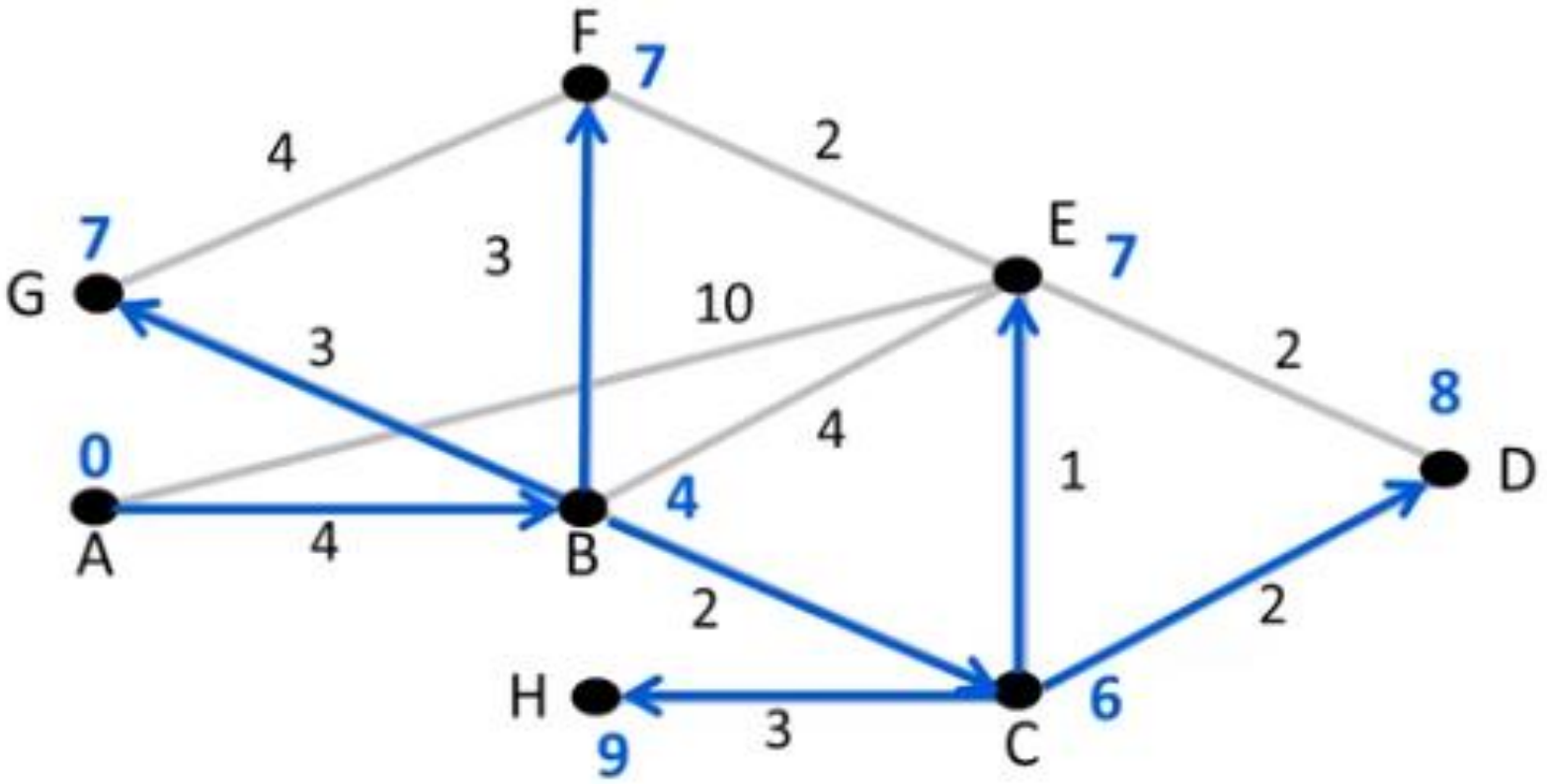
Dijkstra's Algorithm (9)

- Relax around D



Dijkstra's Algorithm (10)

- Finally, H



Dijkstra Comments

- Find shortest path in order of increasing distance from source
  - —Leverages optimality property
- Runtime depends on efficiency of extracting min-cost node
  - —Superlinear in Network size (grows fast)
- Gives complete source/sink tree
  - —More than needed for forwarding!
  - —But requires complete topology