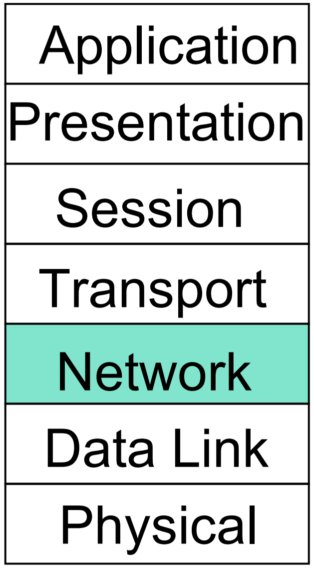
CSE/EE 461: Introduction to Computer Communications Networks
Winter 2010

Module 7
Routing Overview

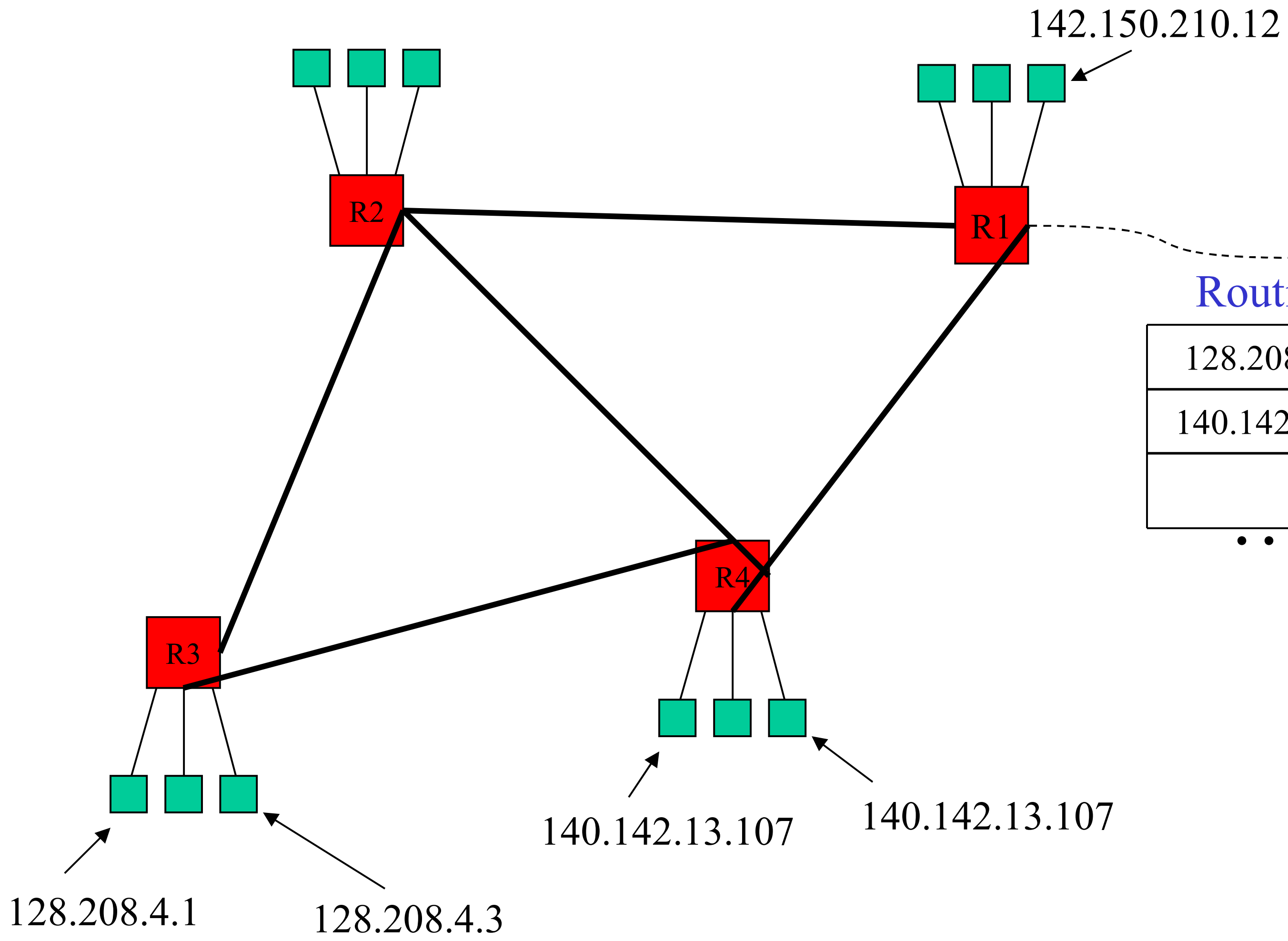
John Zahorjan
zahorjan@cs.washington.edu
534 Allen Center

This Module

- Review of forwarding
- Overview of approaches
    - Distance Vector Routing
    - Link State Routing



Forwarding

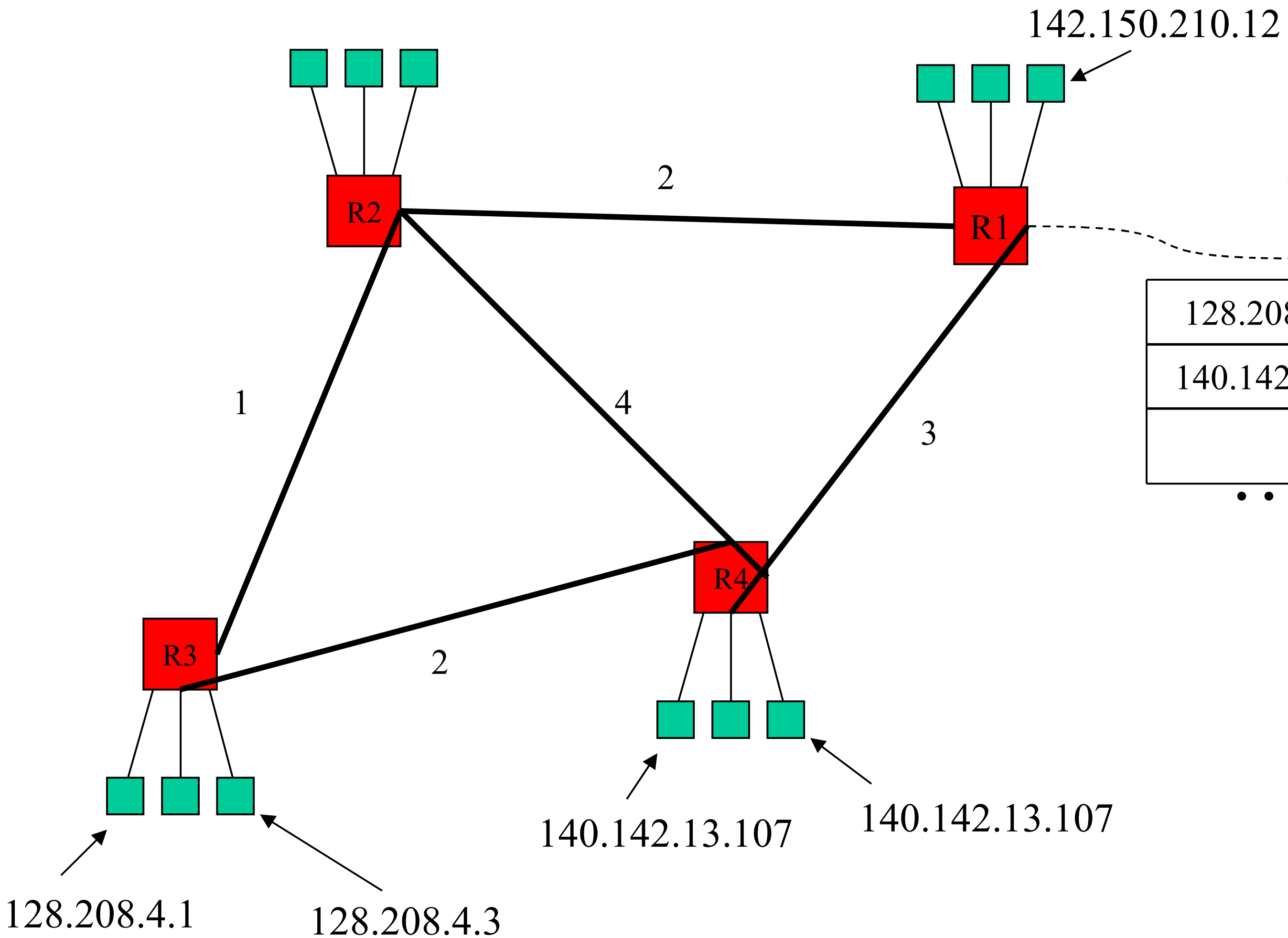


Routing table

128.208.4.*	R2
140.142.13.*	R4

• • • • • •

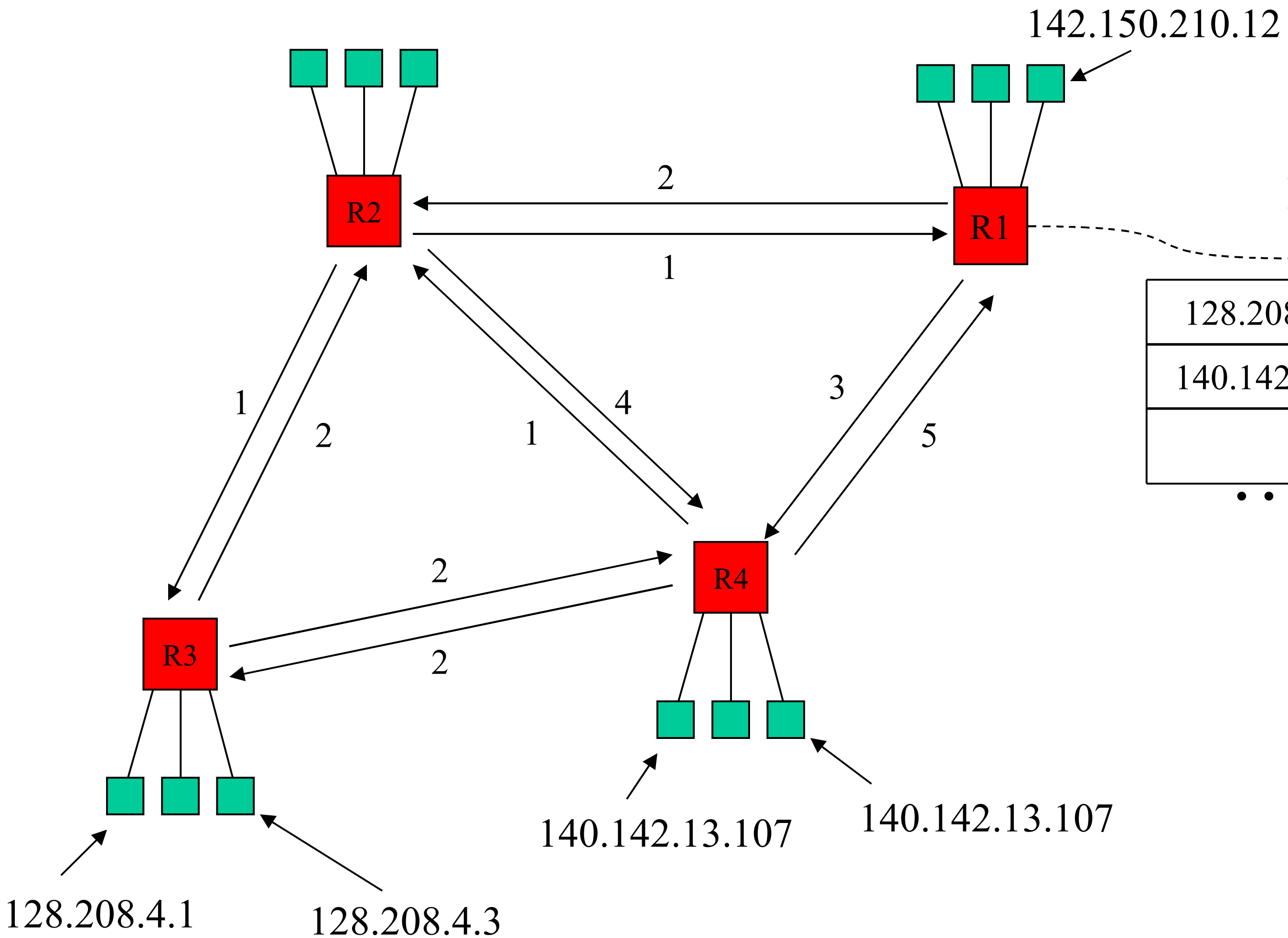
Routing: Link Costs



Routing table

128.208.4.*	R2
140.142.13.*	R4
…	…

Routing: Full Duplex Links



Routing table

128.208.4.*	R2
140.142.13.*	R4
…	…

Routing as a Shortest Path Problem

- Routing table entries: [destination network, next hop router]
- To decide which router is on the next hop, want to find the shortest path from the router to the destination network's router
- We'll first look at sequential solutions, then distributed
  - "Sequential": full network topology information is available
  - "Distributed": must distribute information and perform computation on each router
- We'll first look at the single-destination / all-sources problem, then all-destinations / all-sources
- One thing to look for:
  - each router obtains a consistent view
    - forwards on shortest path
    - shortest paths don't have loops!

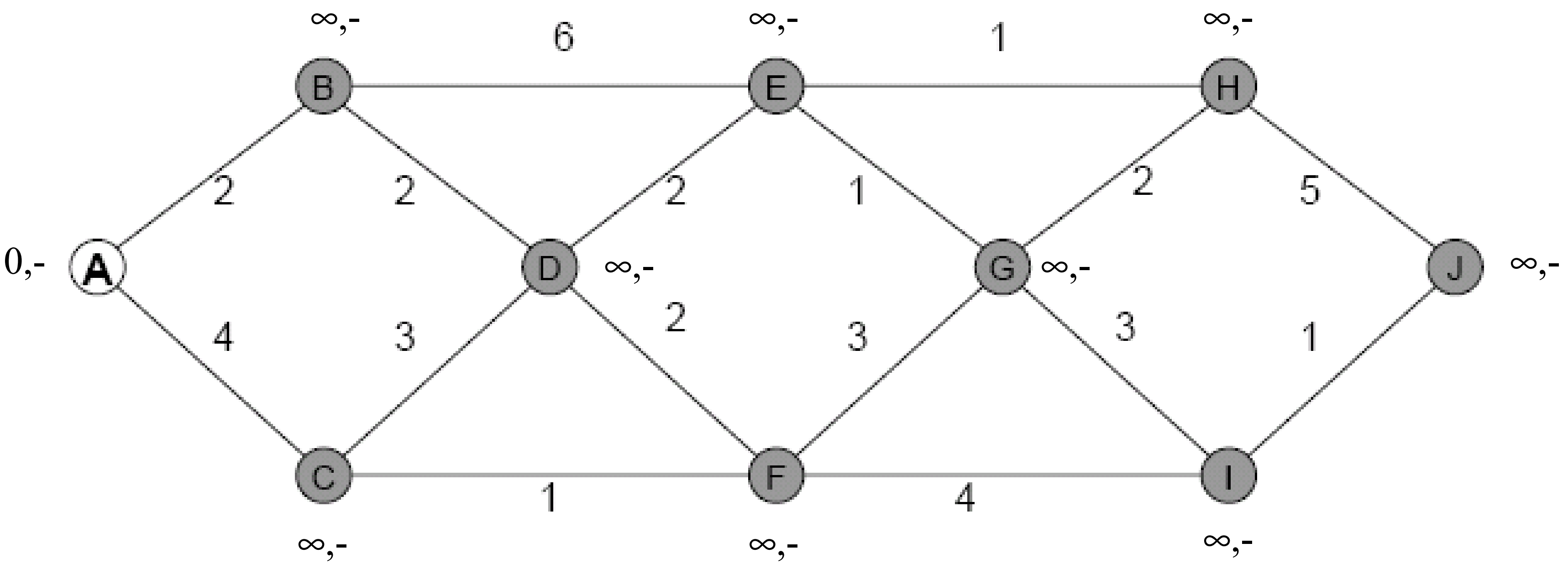
First Approach: Greedy

- Dijkstra's Algorithm

- Greedy:
  - Build the spanning tree by adding routers to the current spanning tree one at a time
  - Choose next the as-yet-unadded router whose distance to the destination is minimal
  - Starting conditions:
    - [0,-] at destination
    - [∞, -] at every other router
    - Spanning tree is the destination router alone

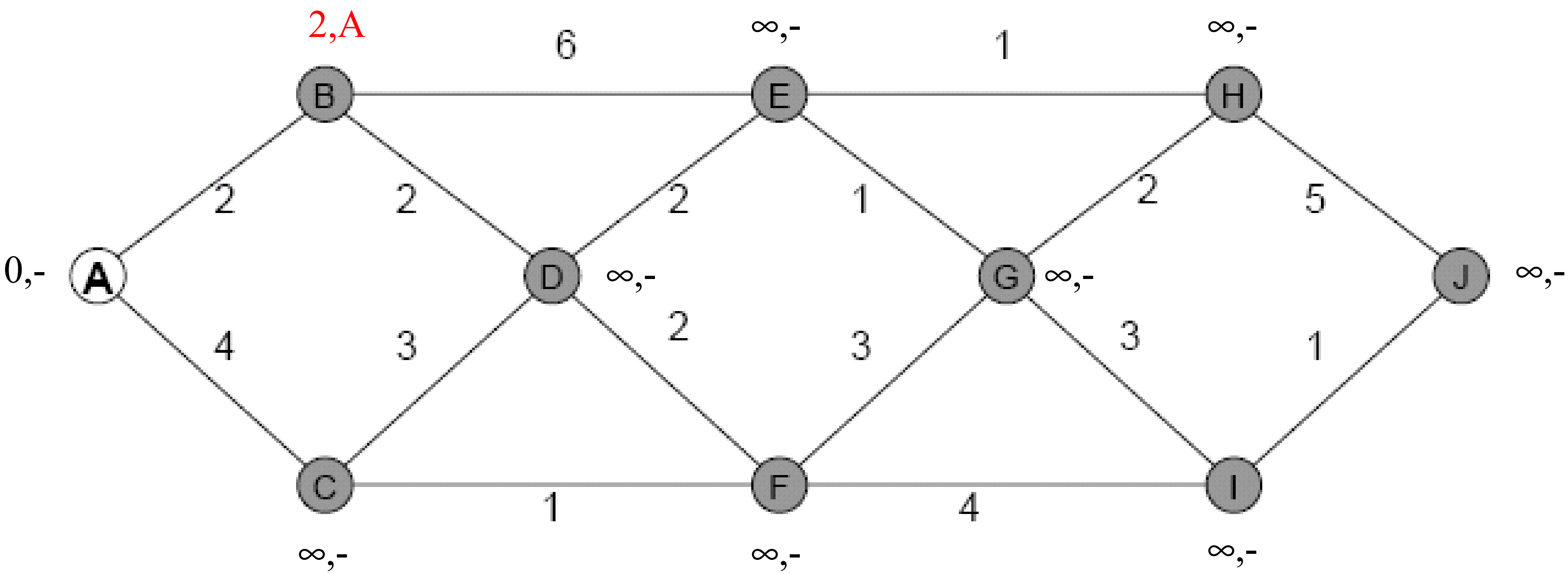
- Running time: O(E logV)

Dijkstra Example



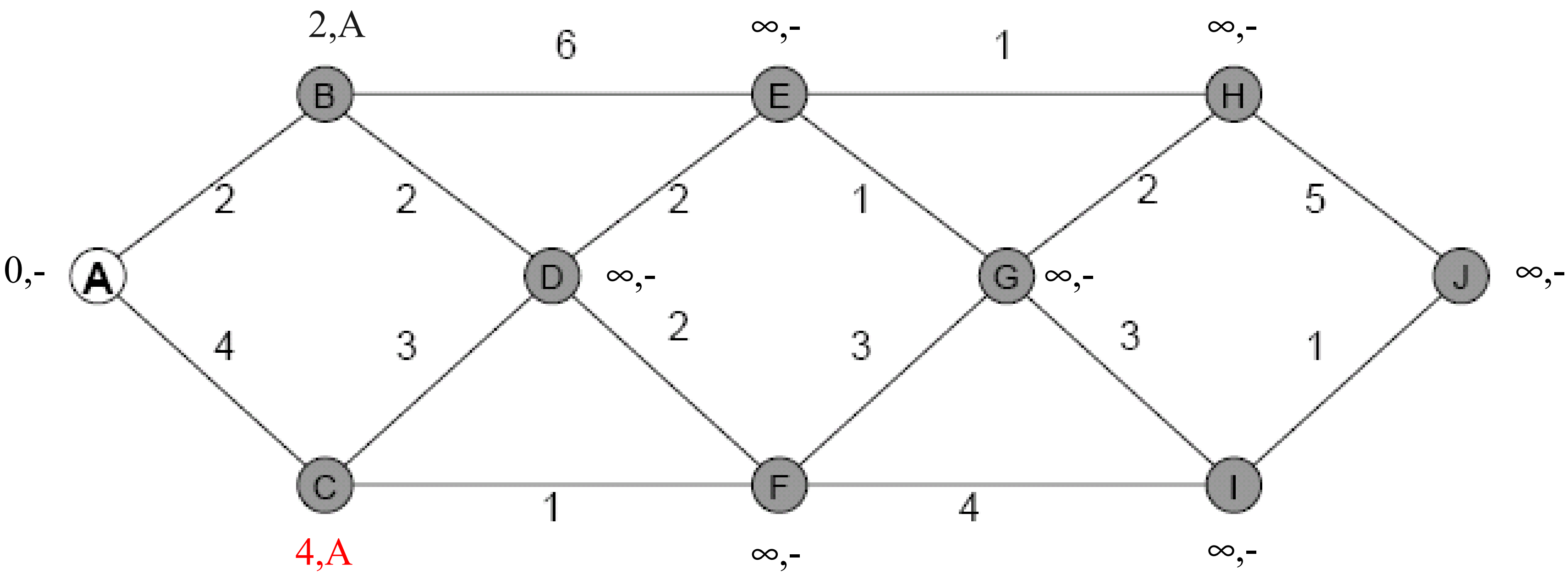
How do we know this works?

After One Step



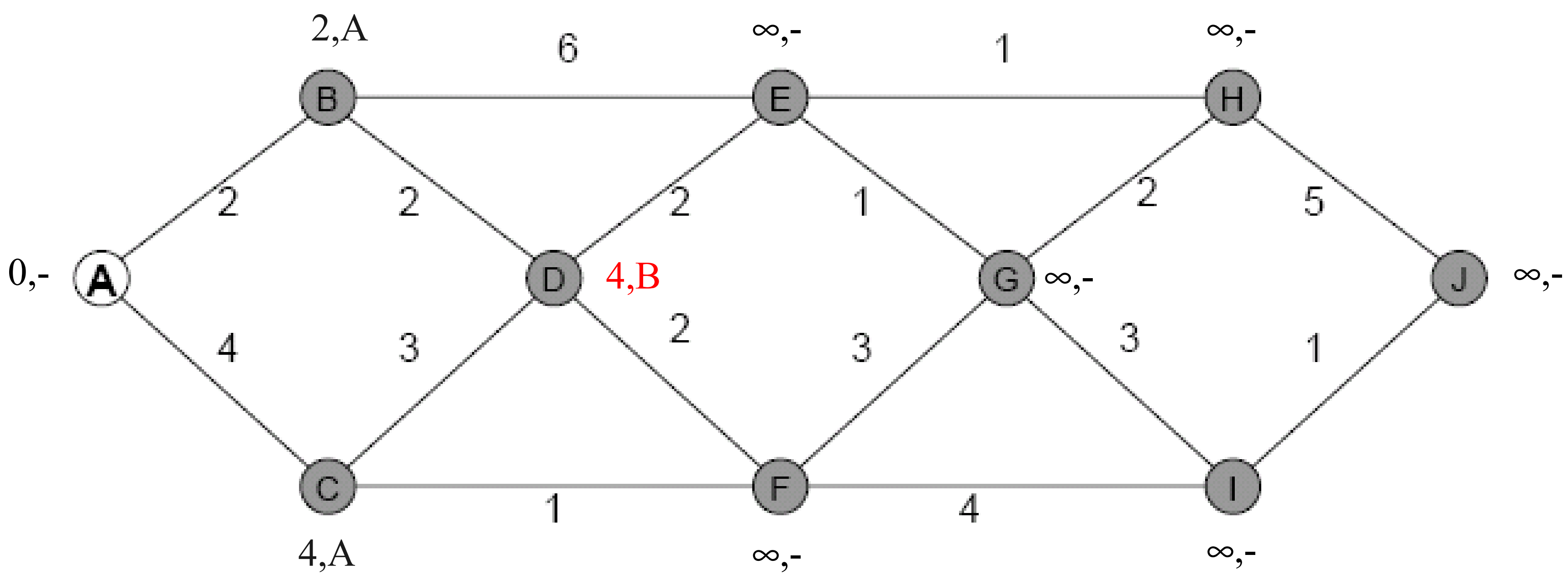
How do we know this works?

After Two Steps



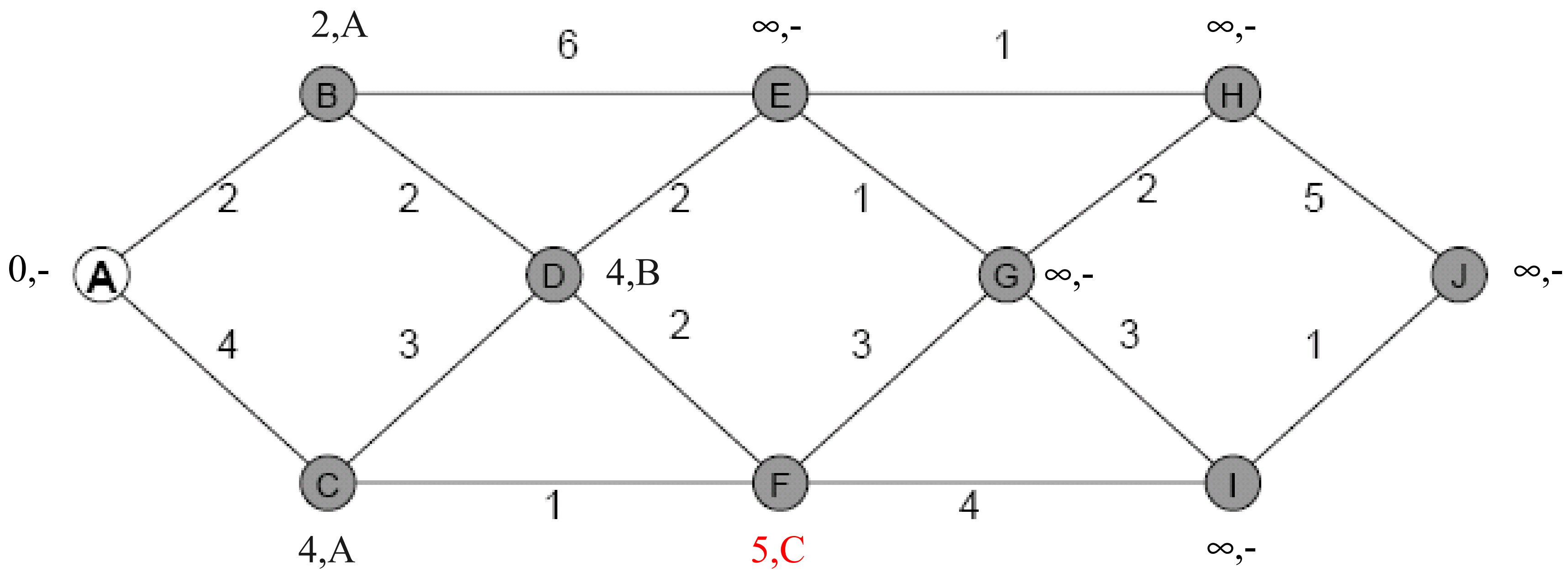
How do we know this works?

After Three Steps



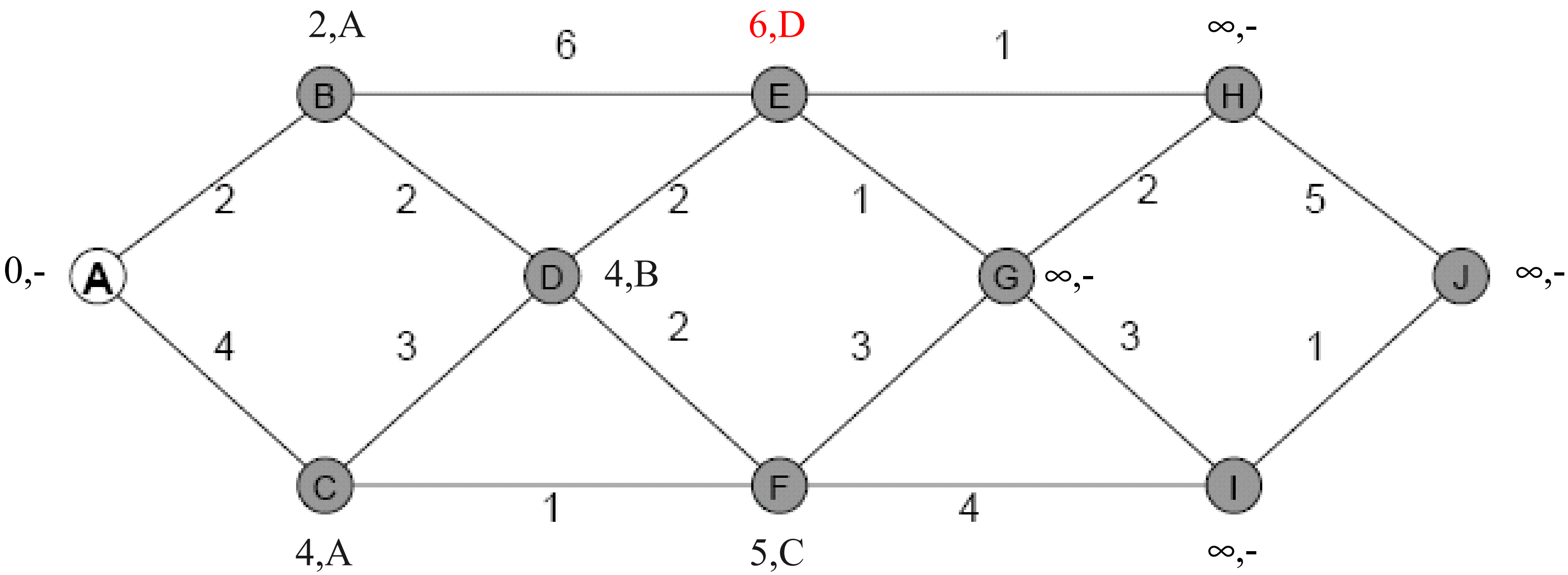
How do we know this works?

After Four Steps



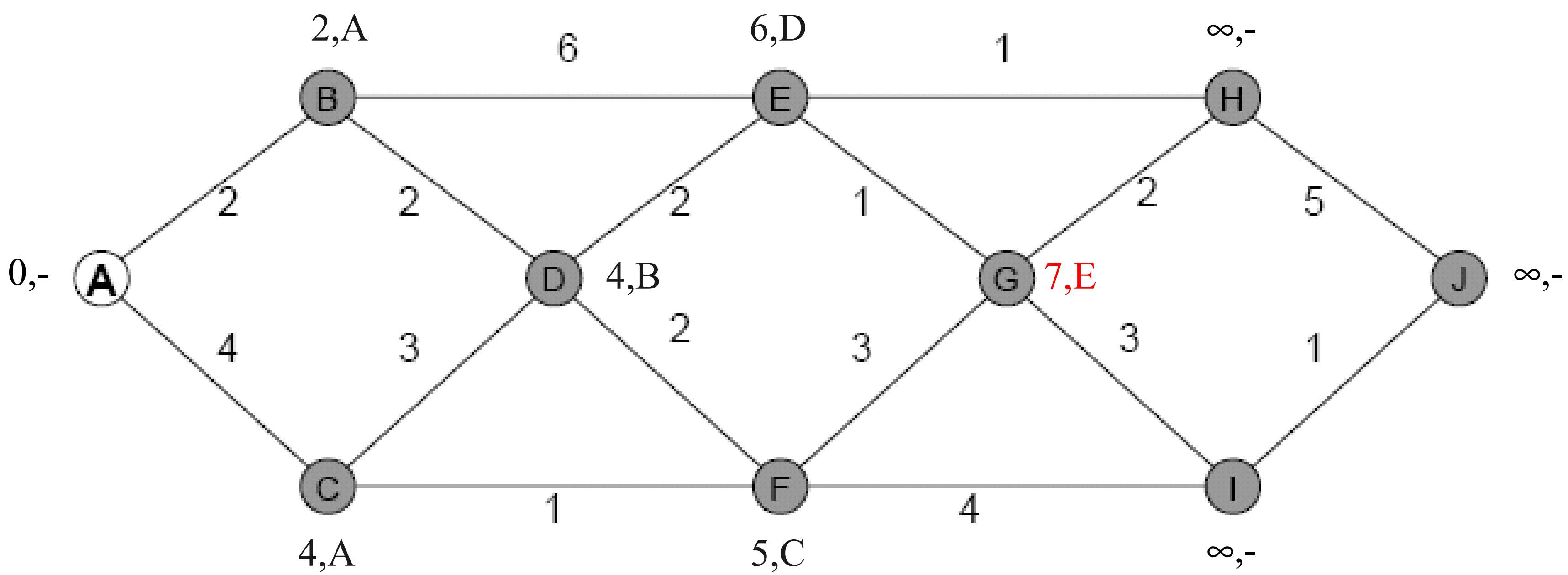
How do we know this works?

After Five Steps



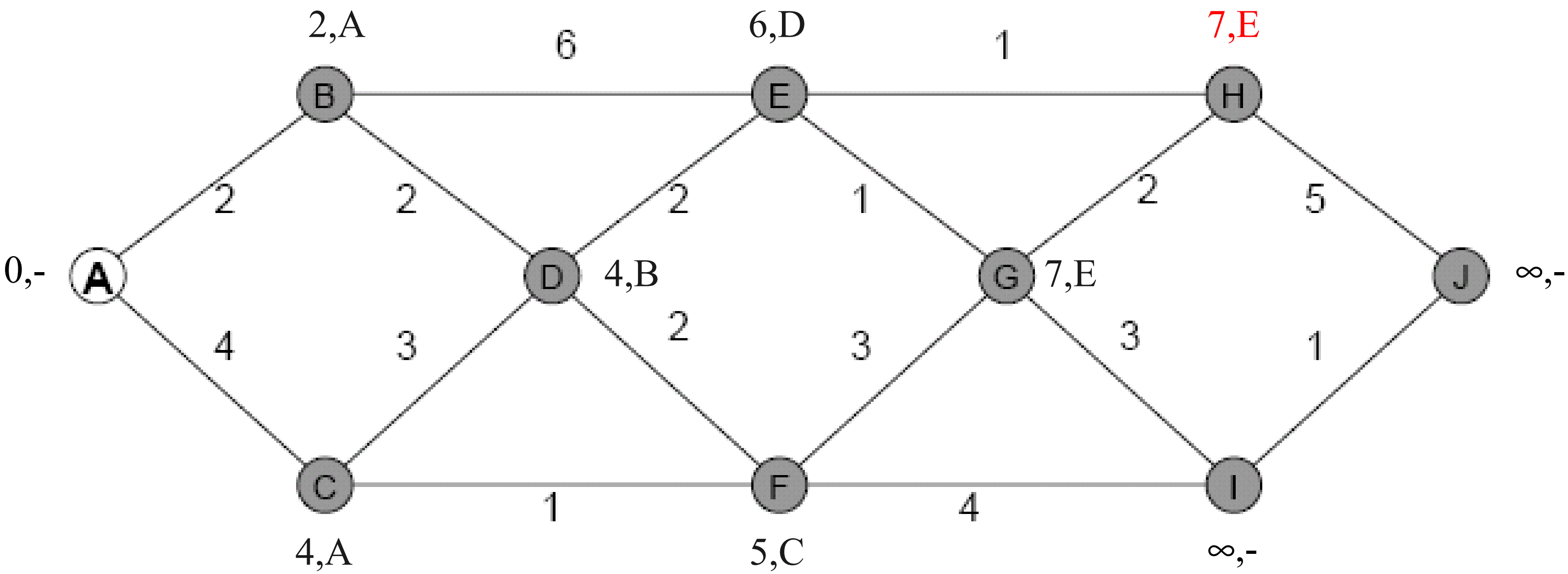
How do we know this works?

After Six Steps



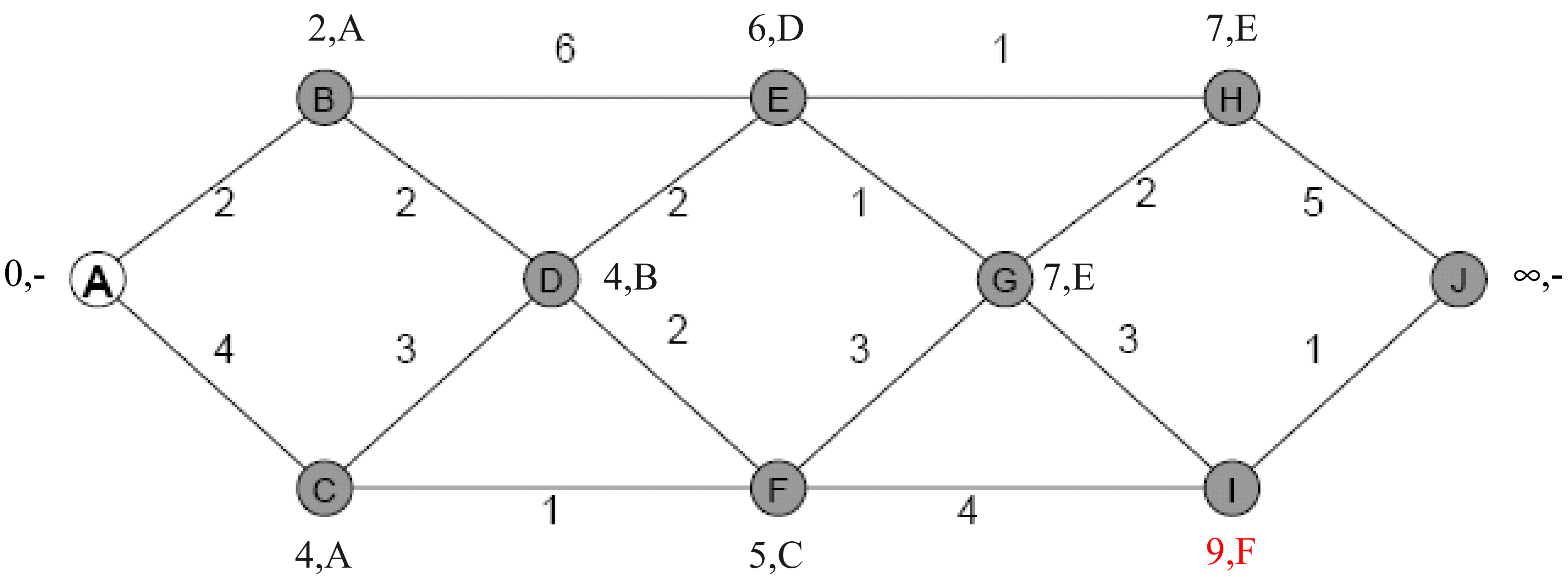
How do we know this works?

After Seven Steps



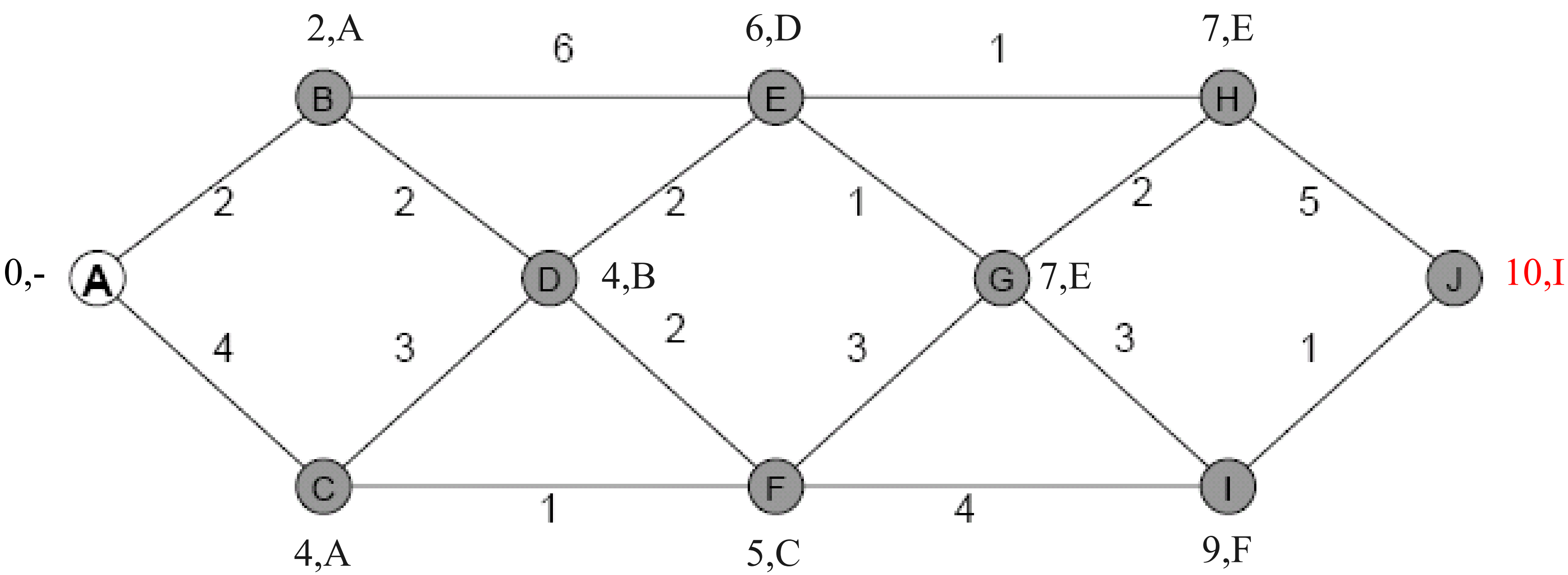
How do we know this works?

After Eight Steps



How do we know this works?

After Nine Steps



How do we know this works?

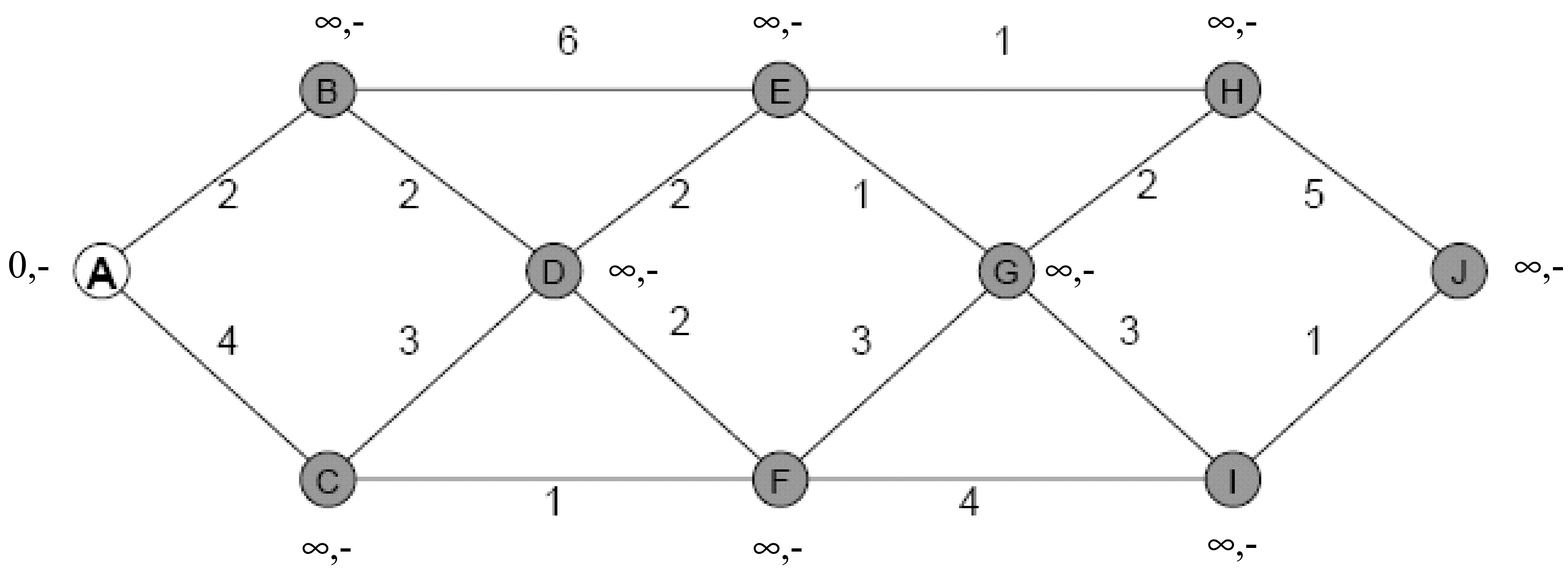
Second Approach: Iterative

- Bellman-Ford Algorithm

- Iterative:
  - At each step, update [cost, next hop] for every router based on [cost] at neighbors
  - Starting conditions:
    - [0,-] at destination
    - [∞, -] at every other router

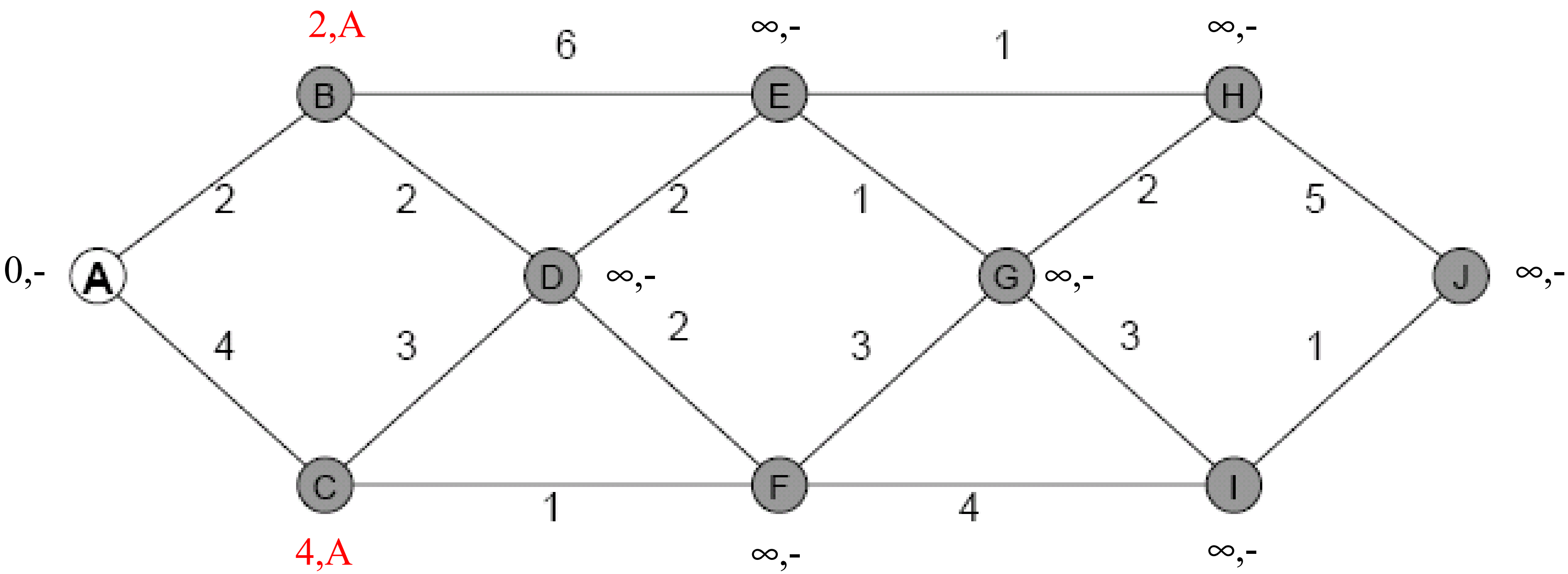
- Running time: O(VE)
  - V: number of vertices (routers)
  - E: number of edges (links)

Bellman-Ford Example



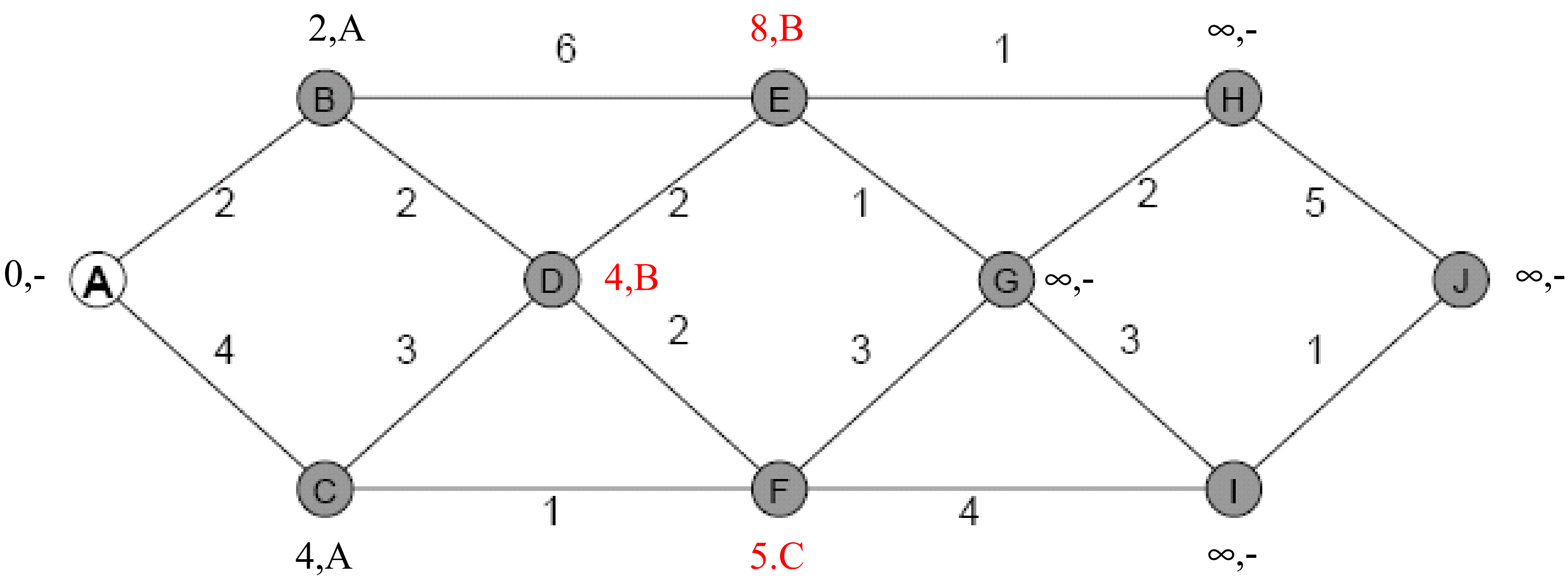
How long can it take to converge?

After One Iteration



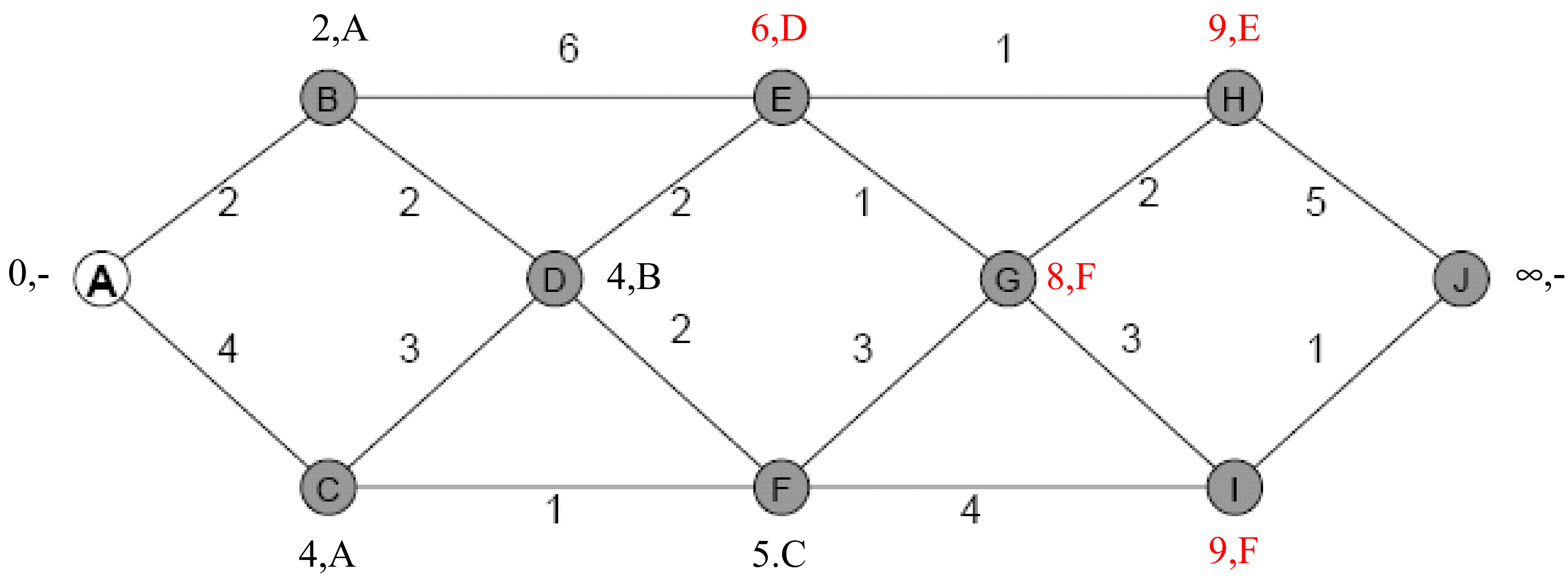
How long can it take to converge?

After Two Iterations



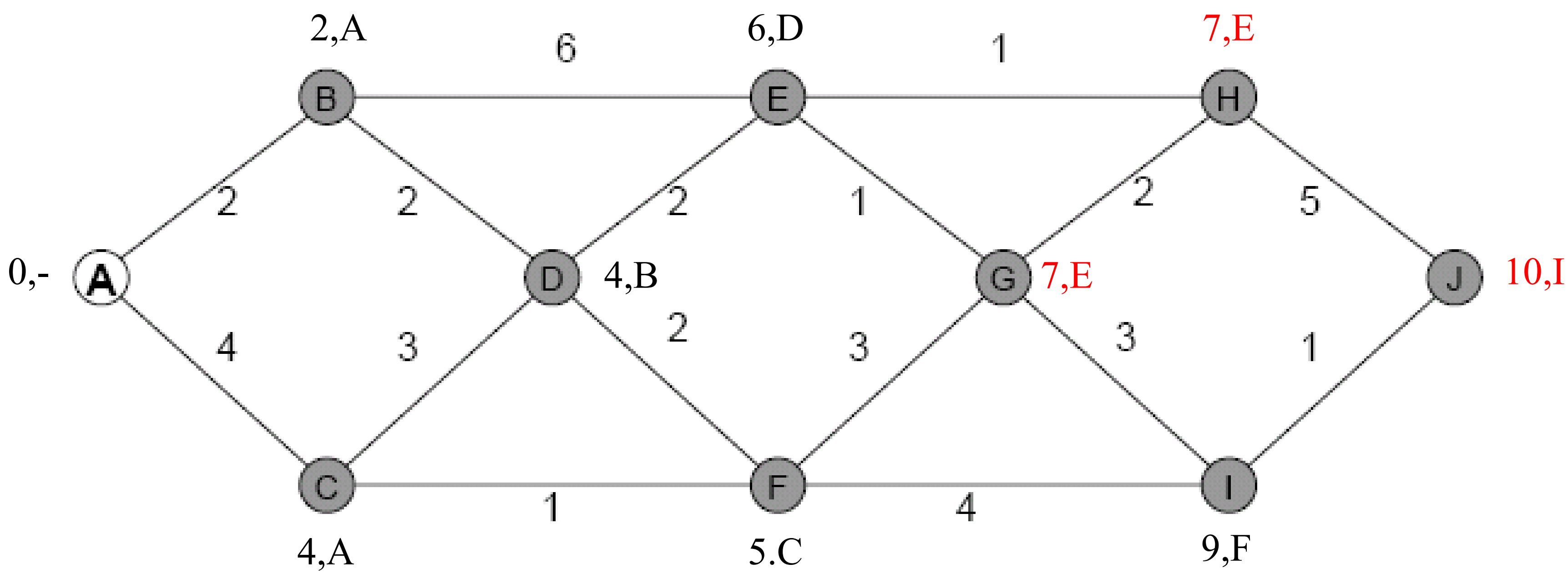
How long can it take to converge?

After Three Iterations



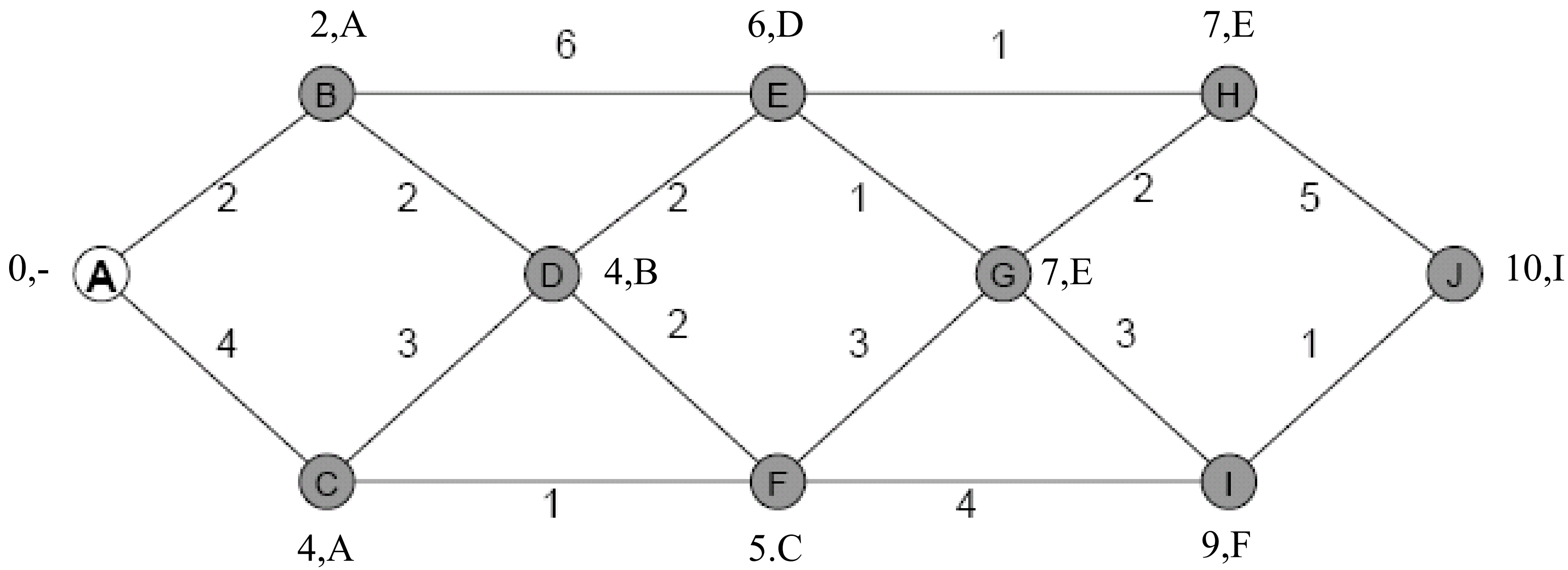
How long can it take to converge?

After Four Iterations



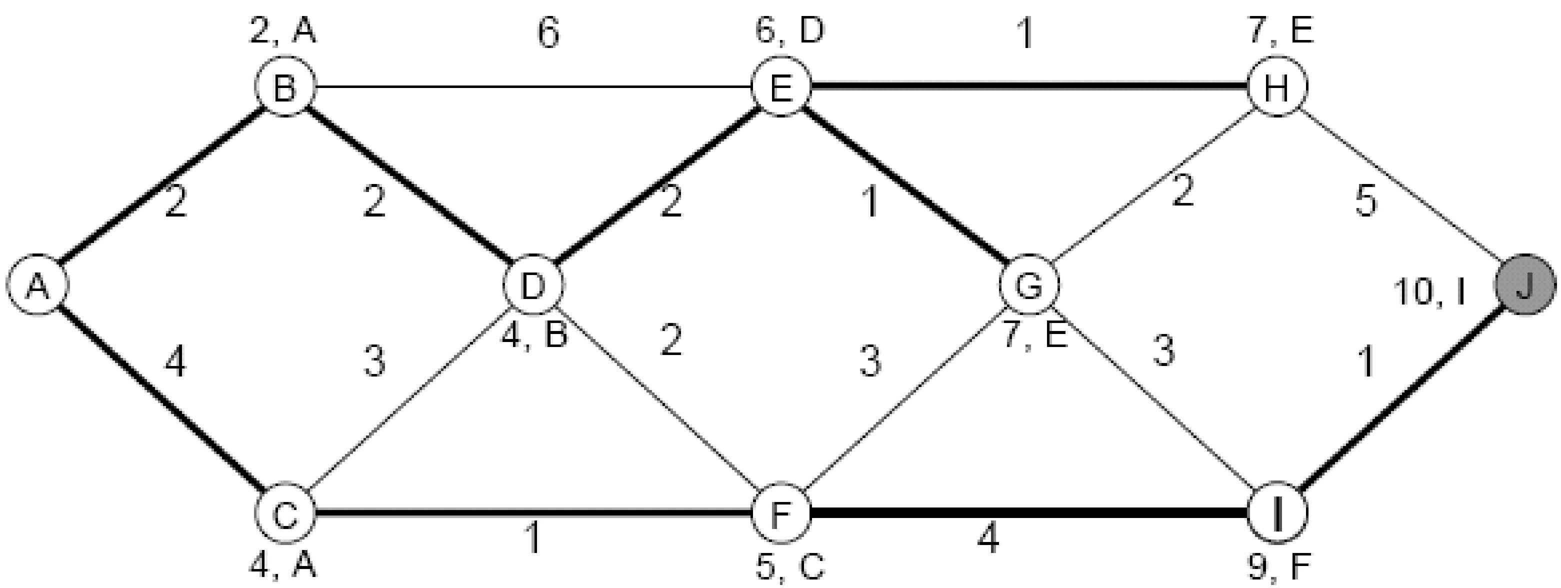
How long can it take to converge?

After Five Iterations



How long can it take to converge?

Result



Note: The result is a spanning tree rooted at the destination

Moving to the Internet

- Routing table reflects spanning tree from source to every destination
  - Not really a big change
    - Bellman-Ford: every destination is engaged in the procedure
    - Dykstra: make the source the root, rather than the destination
- Have to distribute information
  - Bellman-Ford: neighbor information about current costs to each destination
  - Dijkstra: full topology/cost information
- The process is on-going
  - Not all routers boot at once
- Router/link failures can occur
  - Link cost data isn't static