# 6.852 Lecture 3

- Algorithms in general synchronous networks (continued)
  - breadth-first search
  - broadcast, convergecast
  - shortest paths
  - minimum-weight spanning tree

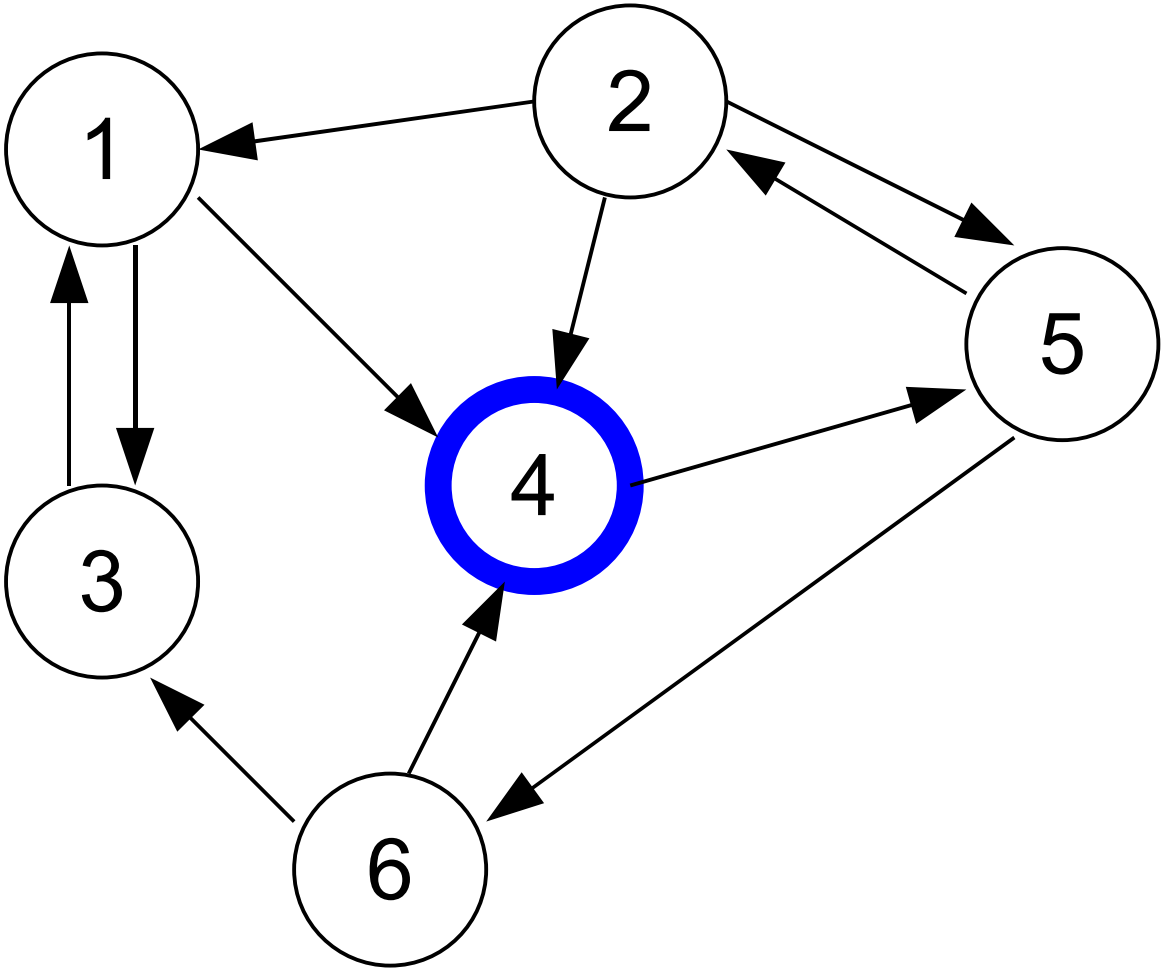
# Last lecture

- Lower bound for comparison-based leader election in a ring
- Leader election in general synchronous networks
  - flooding
  - reducing message complexity
  - simulations

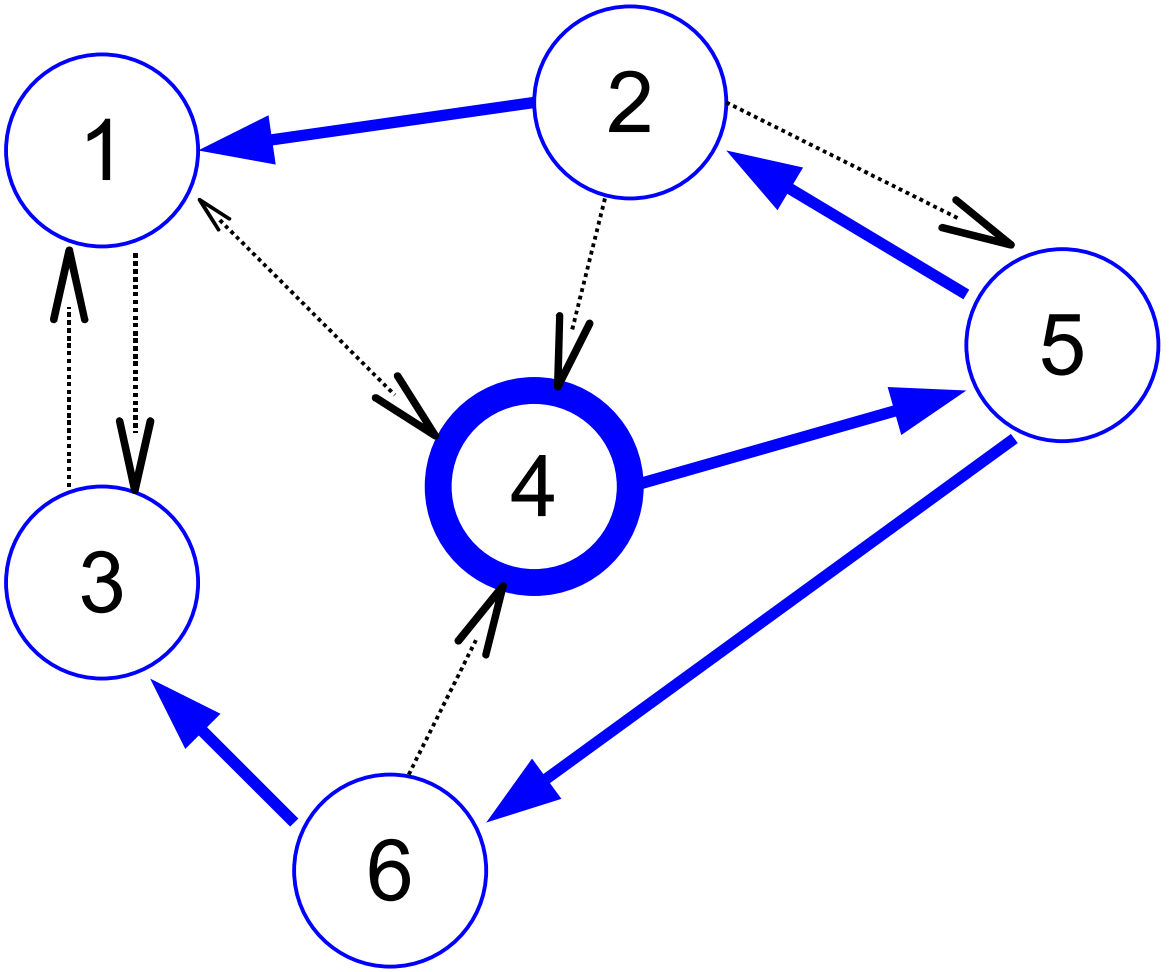
# Breadth-first search

- Assume
  - strongly connected digraph, UIDs
  - no knowledge of size, diameter of network
  - distinguished source node $i_0$
- Required: breadth-first spanning tree
  - spanning: contains every node
  - breadth-first: node at distance d from $i_0$ appears at depth d in tree
  - output: parent of each node (except $i_0$)

# Breadth-first search



# Breadth-first search



# Breadth-first search

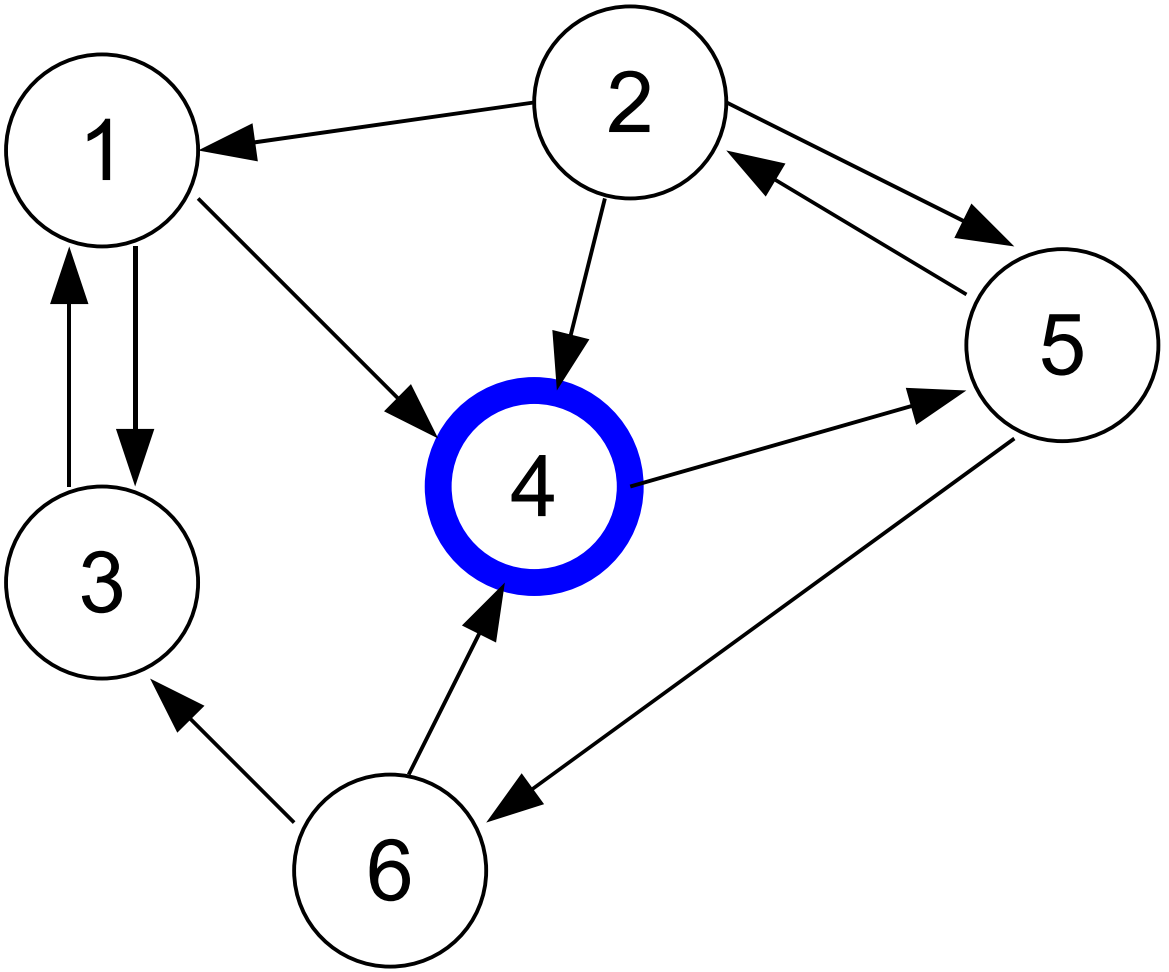
- "Mark" nodes as they get incorporated into tree
  - initially only $i_0$ is marked
  - round 1: $i_0$ sends "search" to out-nbrs
  - every round: unmarked nodes that receive "search"
    - marks self
    - designates one process that sent "search" as parent
    - send "search" to out-nbrs next round

# Breadth-first search

- "Mark" nodes as they get incorporated into tree
  - initially only $i_0$ is marked
  - round 1: $i_0$ sends "search" to out-nbrs
  - every round: unmarked nodes that receive "search"
    - marks self
    - designates one process that sent "search" as parent
    - send "search" to out-nbrs next round

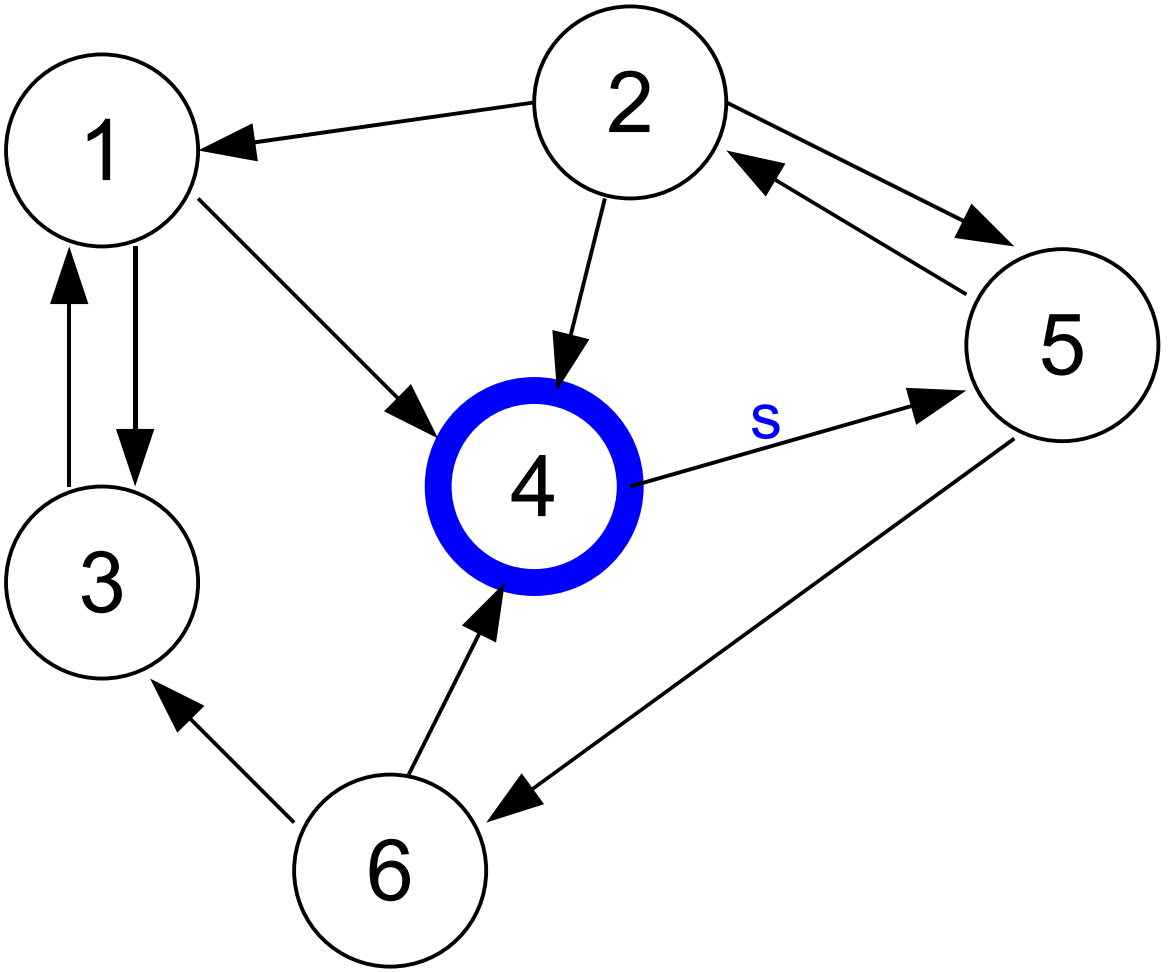
What state variables do we need?

# Breadth-first search



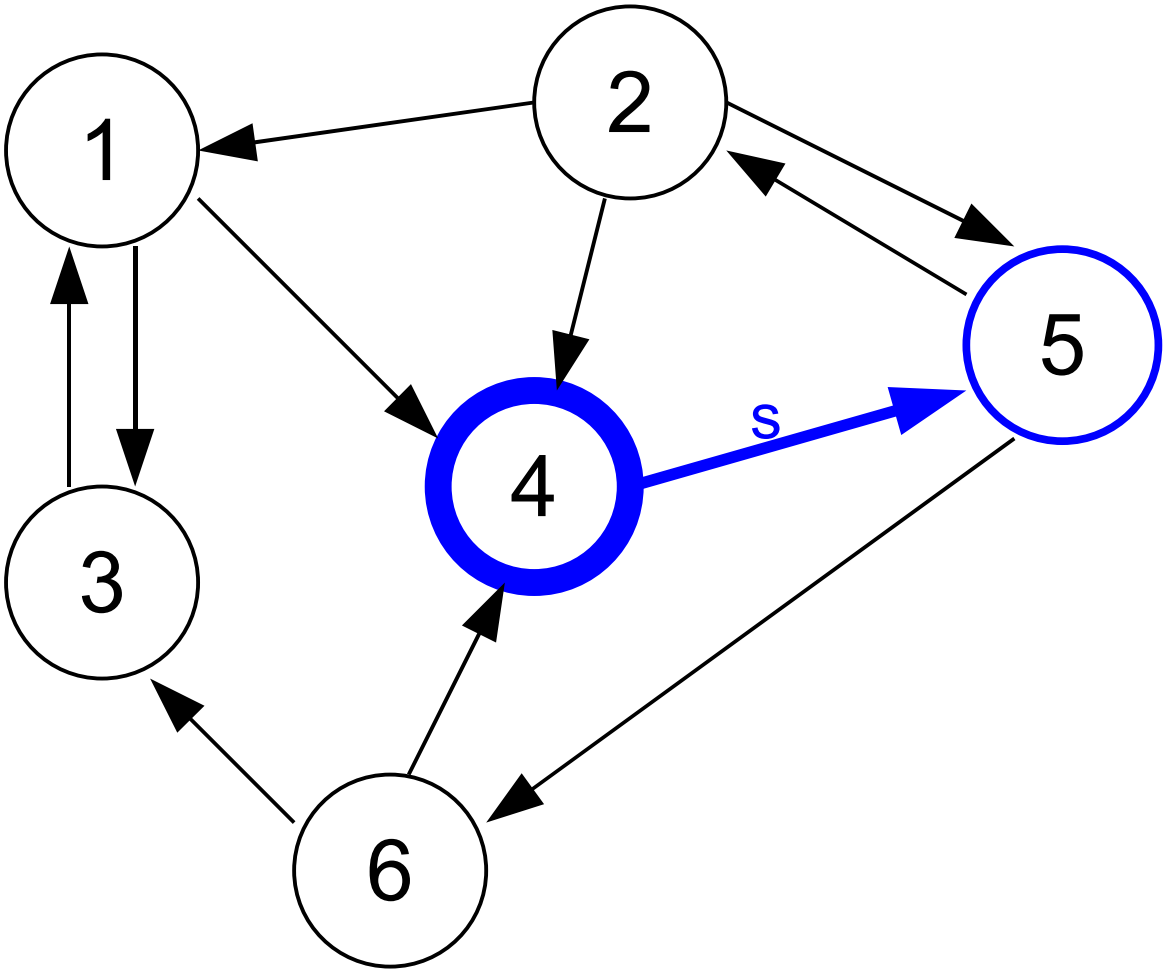

Round 1 (start)

# Breadth-first search



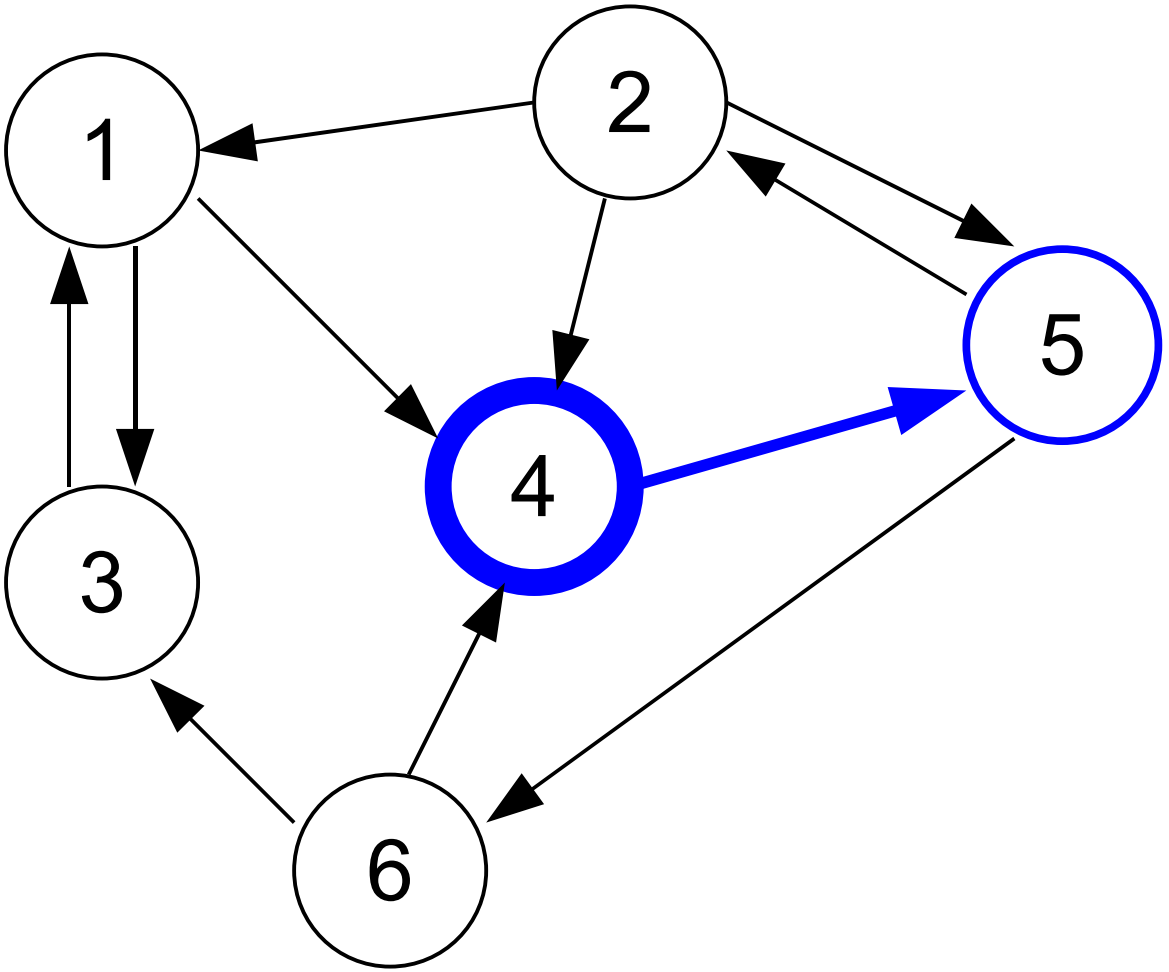

Round 1 (msgs)

# Breadth-first search



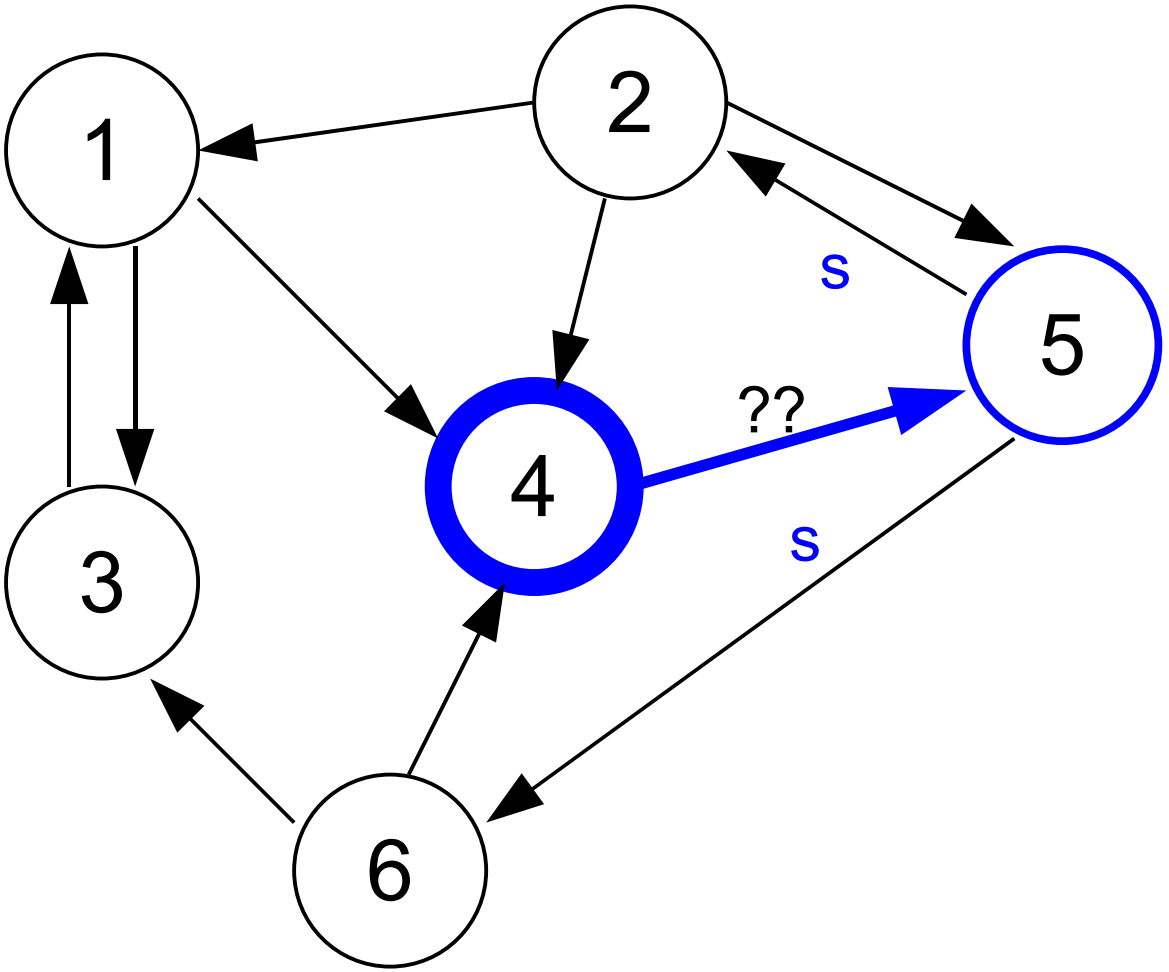

Round 1 (trans)

# Breadth-first search



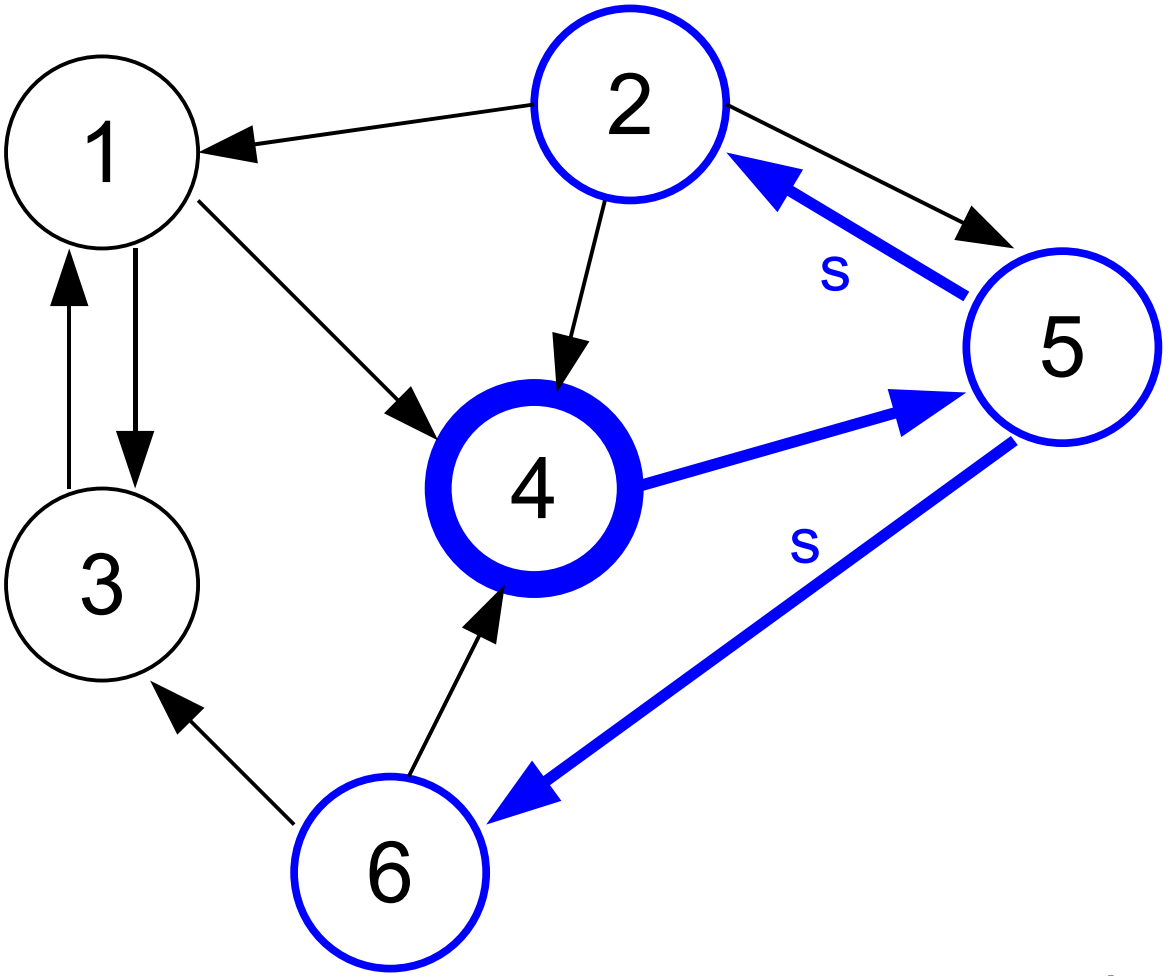

Round 2 (start)

# Breadth-first search



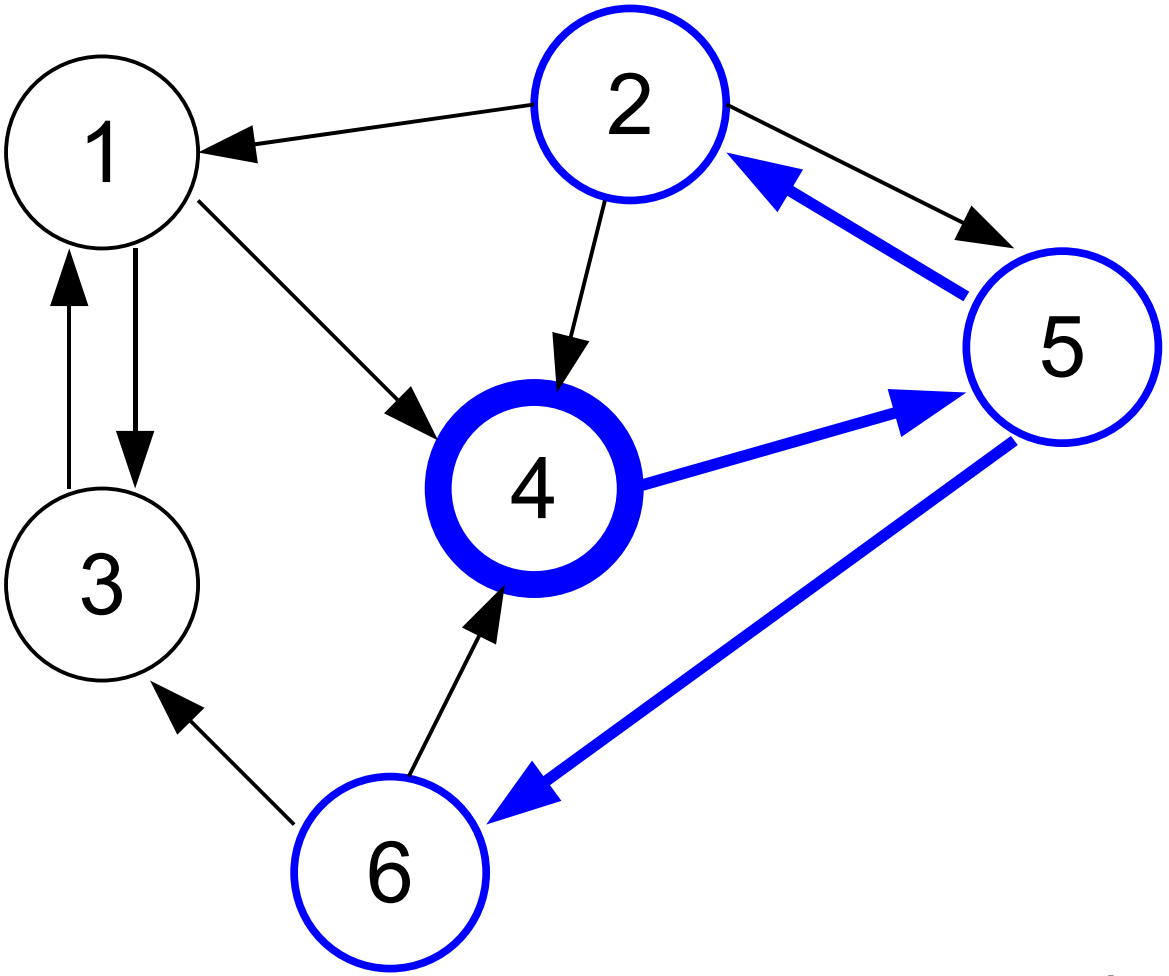

Round 2 (msgs)

# Breadth-first search



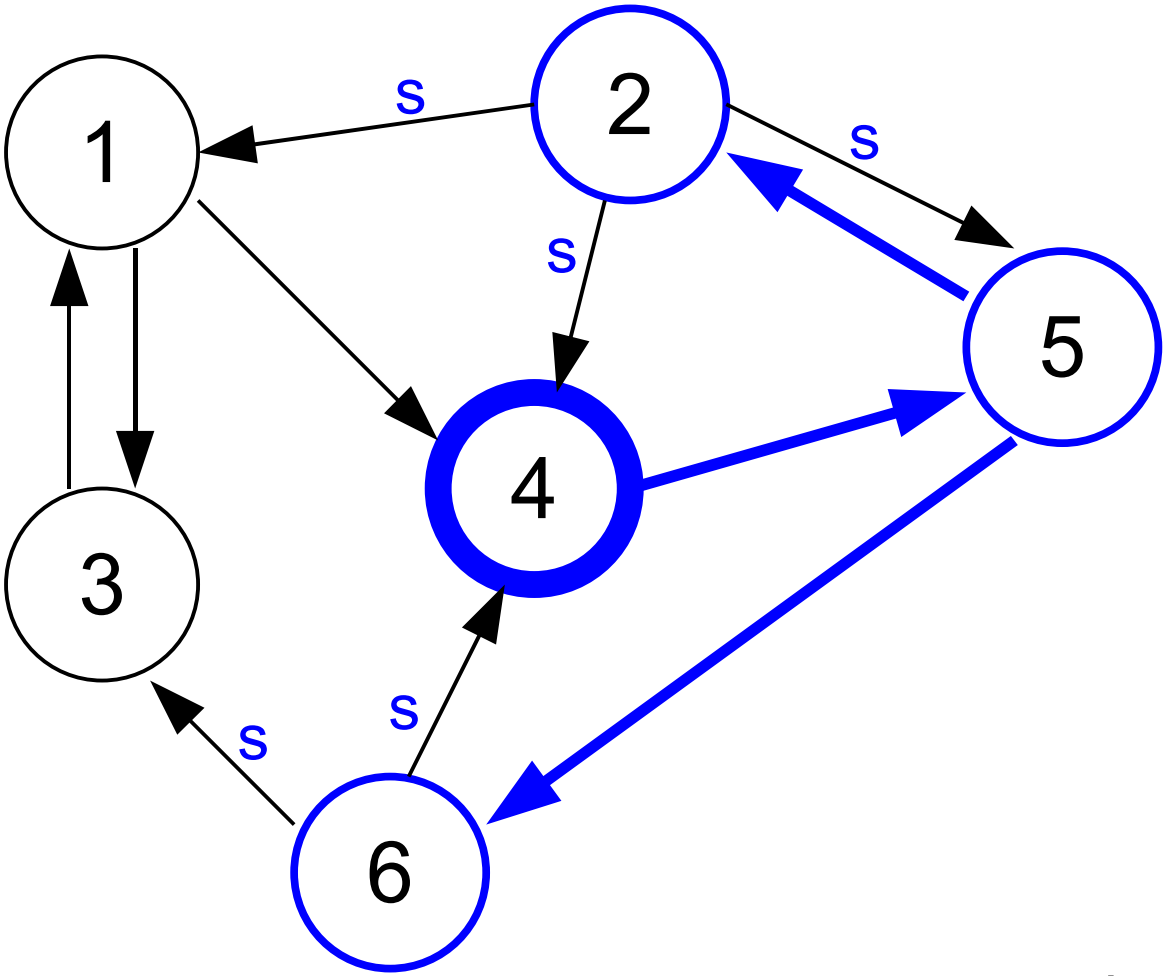

Round 2 (trans)

# Breadth-first search



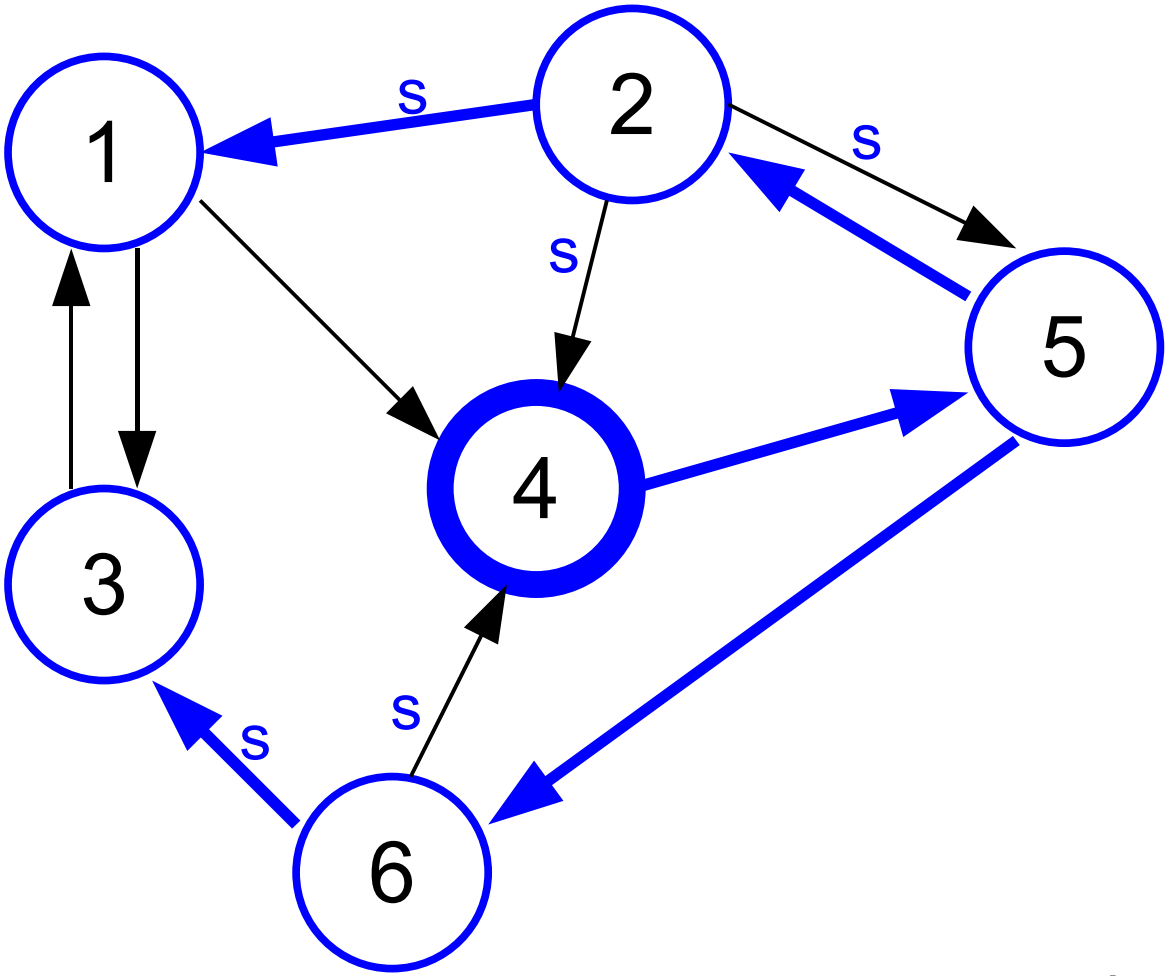

Round 3 (start)

# Breadth-first search



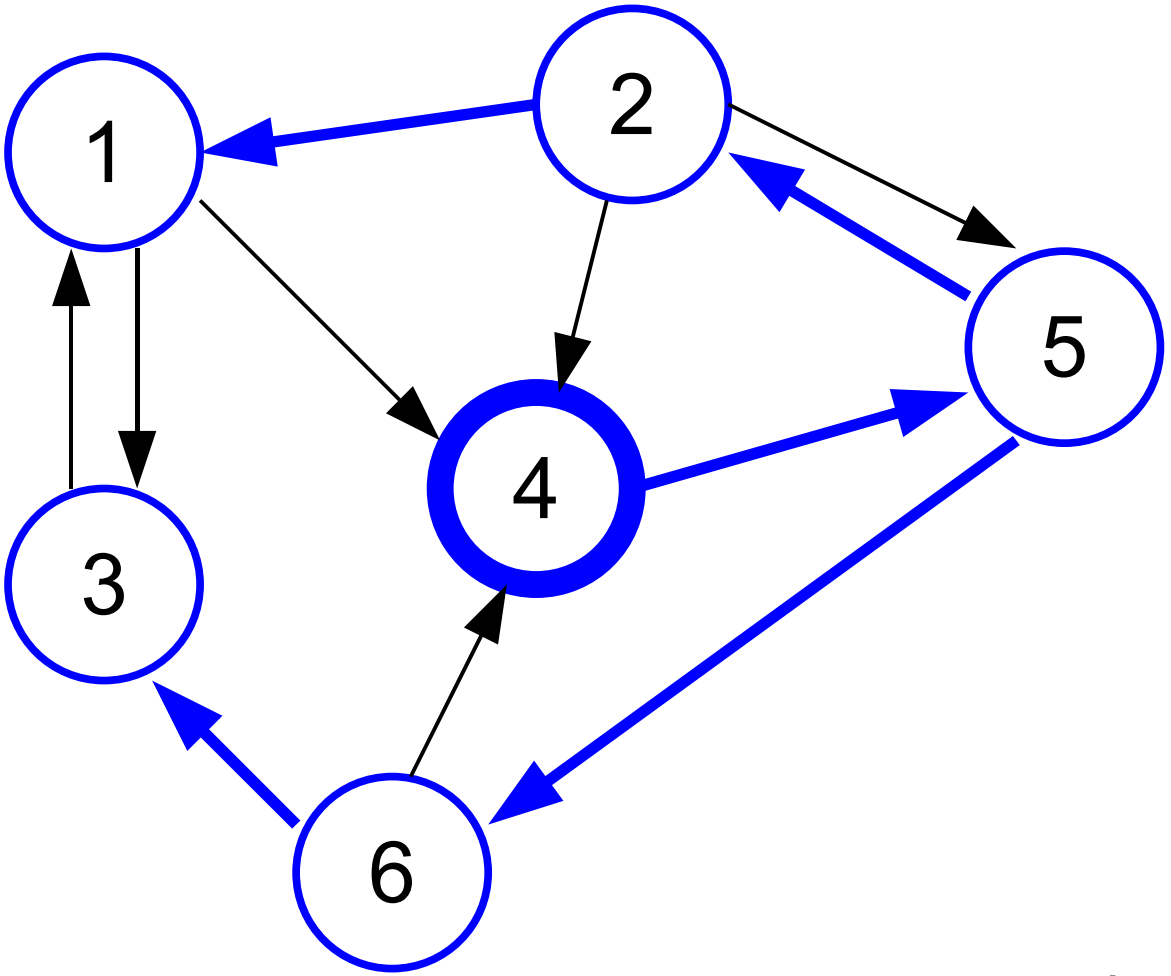

Round 3 (msgs)

# Breadth-first search



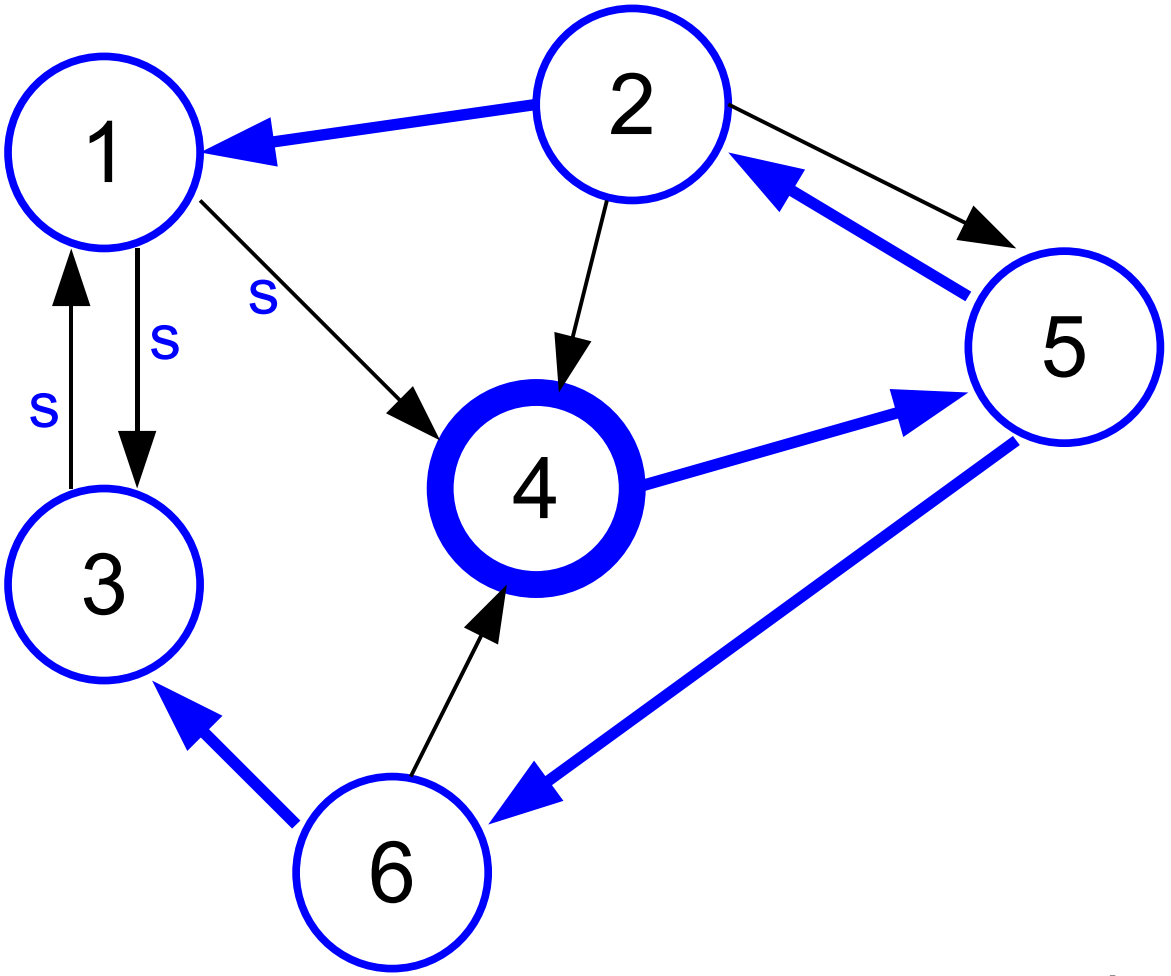

Round 3 (trans)

# Breadth-first search



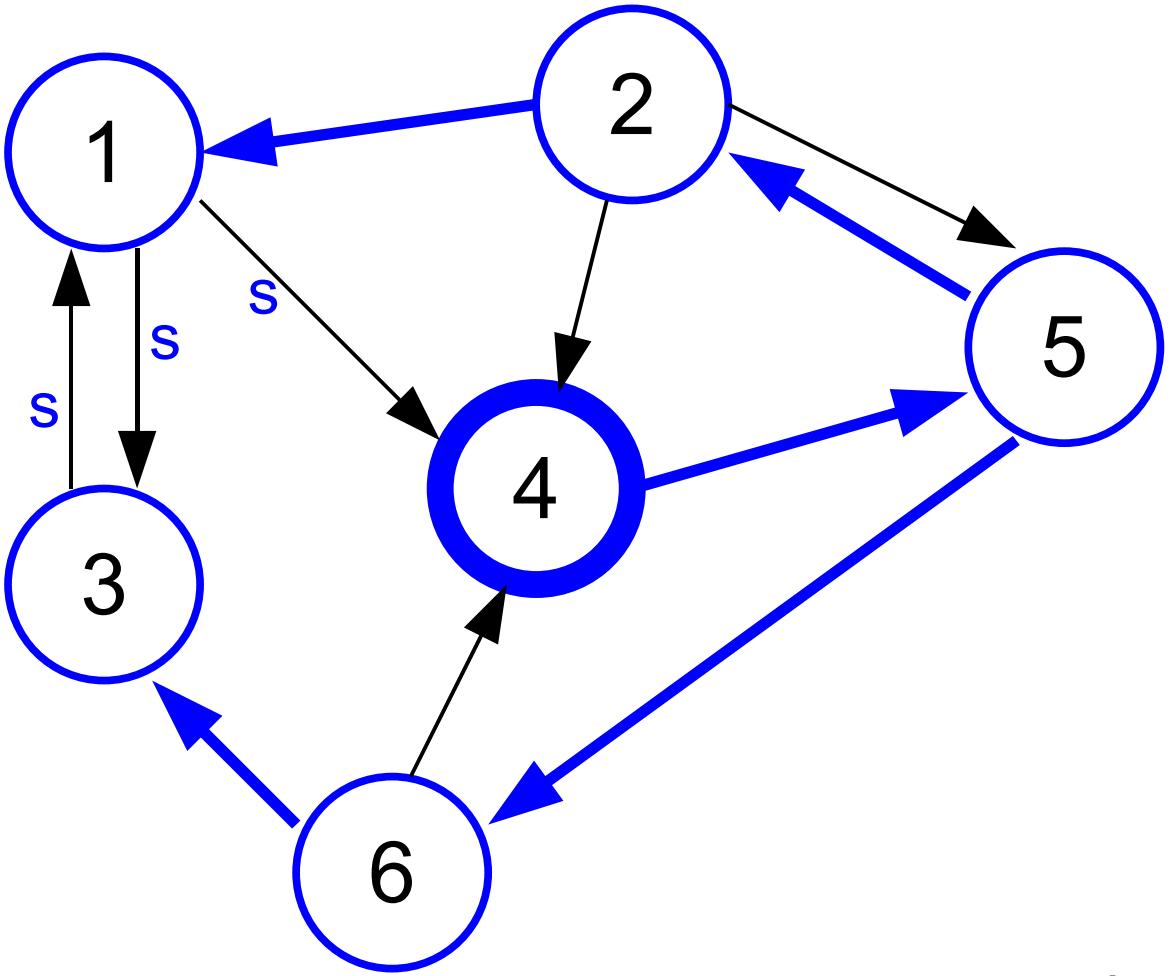

Round 4 (start)

# Breadth-first search



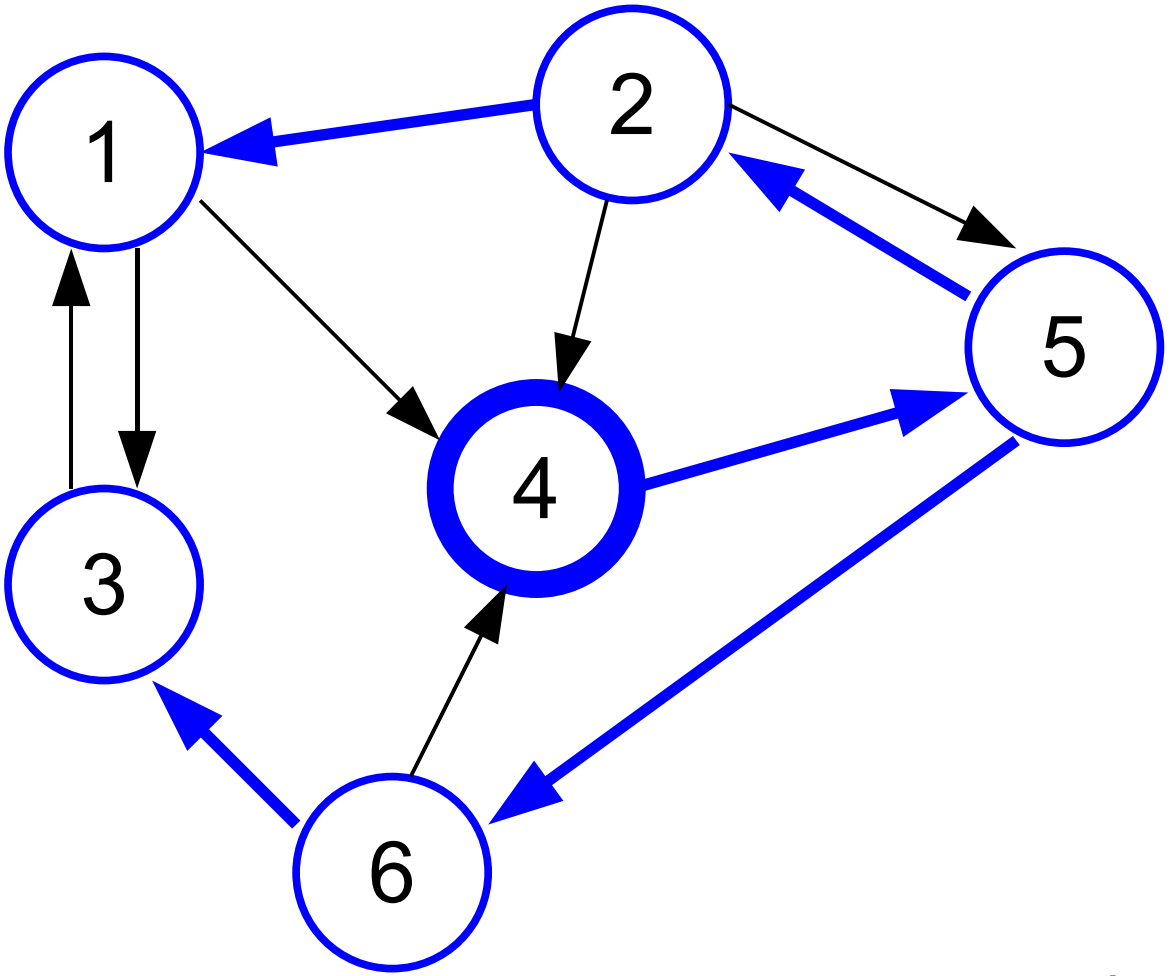

Round 4 (msgs)

# Breadth-first search



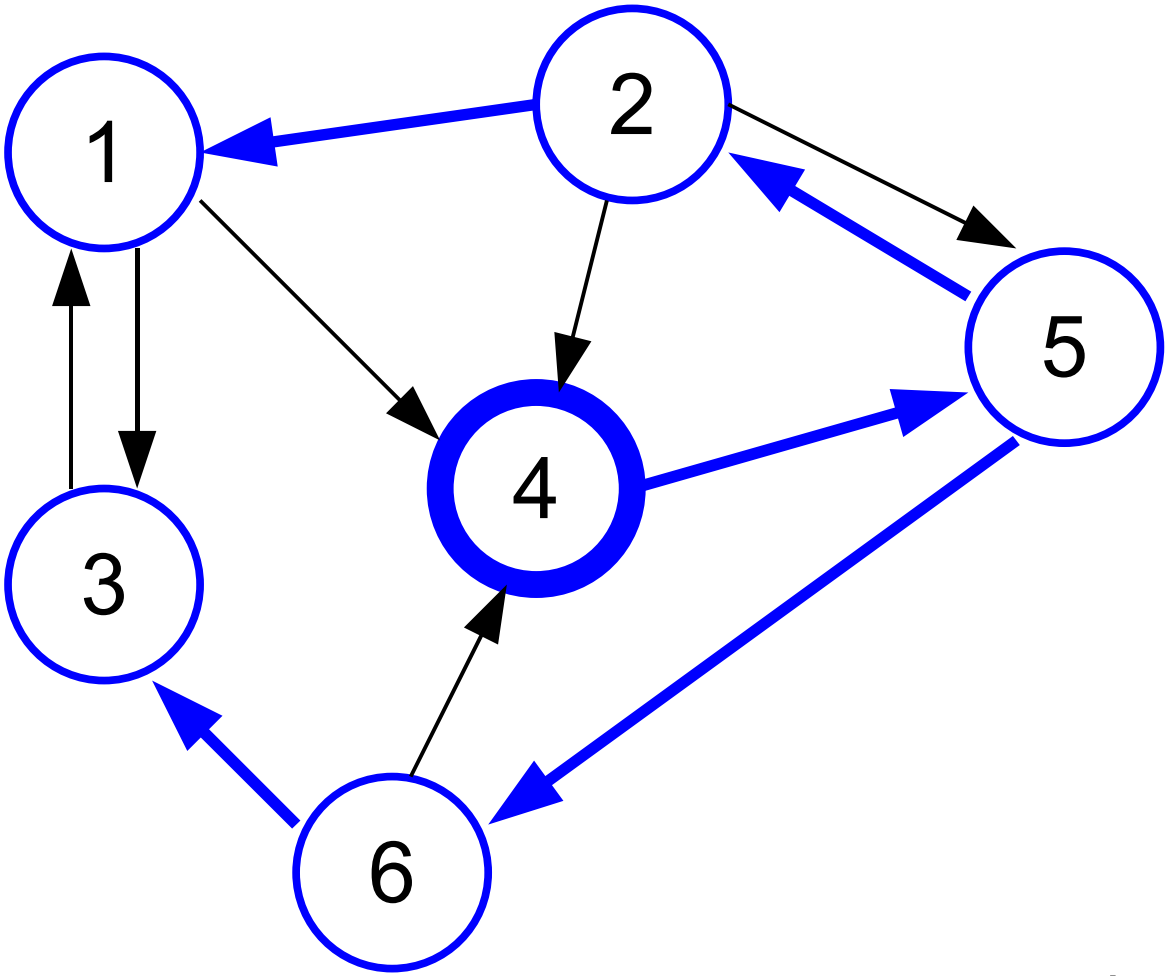

Round 4 (trans)

# Breadth-first search



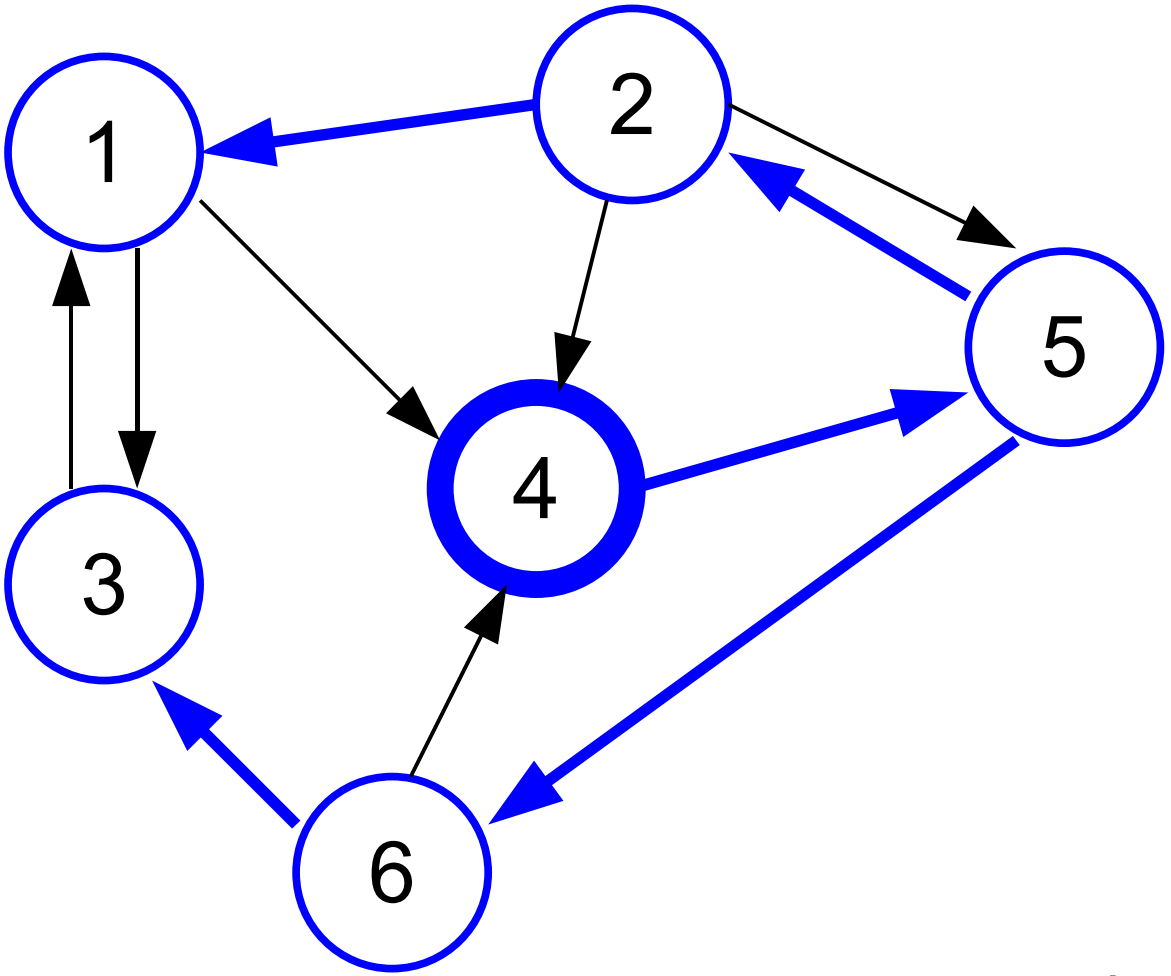

Round 5 (start)

# Breadth-first search




Round 5 (msgs)

# Breadth-first search




Round 5 (trans)

# Breadth-first search

- "Mark" nodes as they get incorporated into tree
  - initially only $i_0$ is marked
  - round 1: $i_0$ sends "search" to out-nbrs
  - every round: unmarked nodes that receive "search"
    - marks self
    - designates one process that sent "search" as parent
    - send "search" to out-nbrs next round
      - need flag to keep track of when to send
- Complexity: time = diameter+1; msg = |E|

# Breadth-first search

- Child pointers?
  - easy with bidirectional communication
  - what if not?
    - message bit complexity
- Termination?
  - with bidirectional communication?
    - “convergecast”
  - with unidirectional communication?

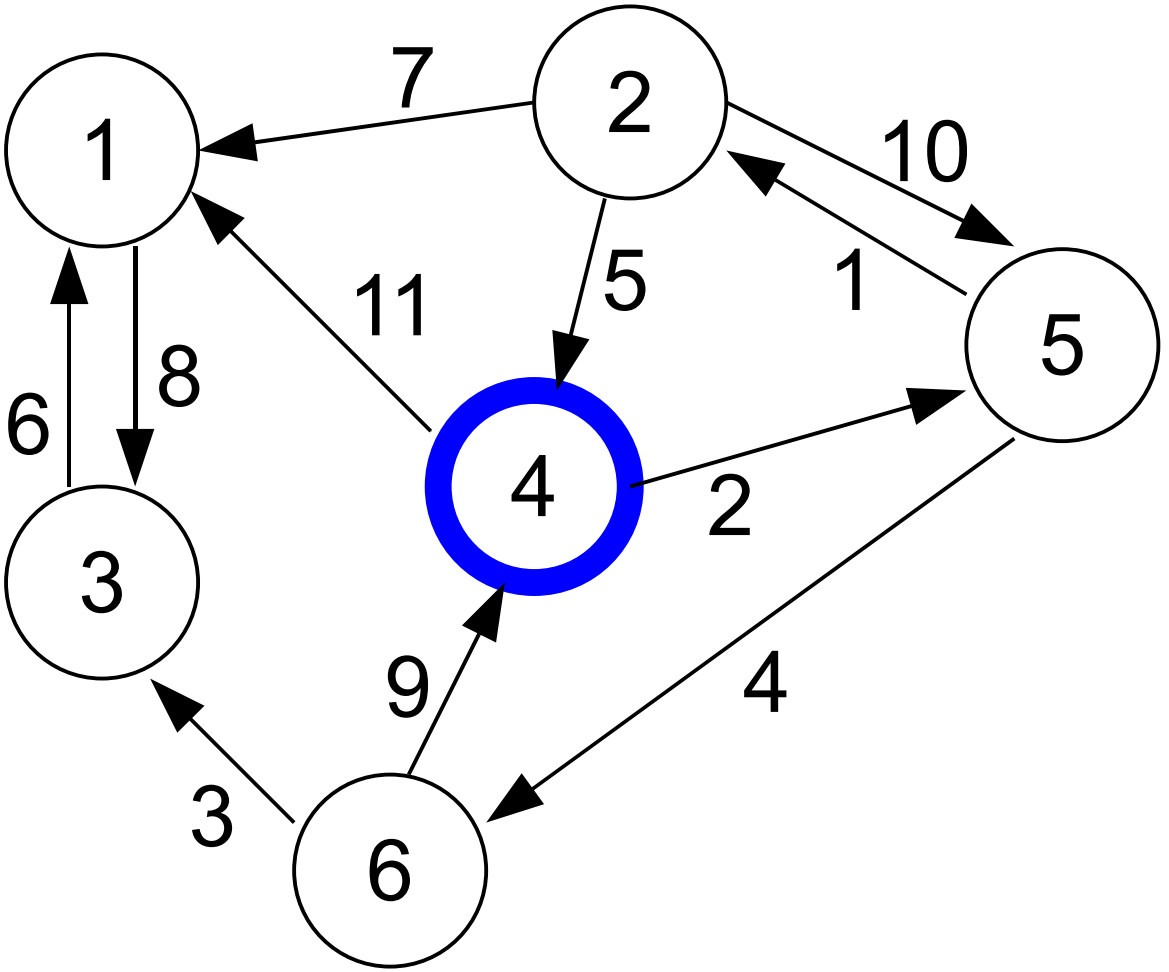
# Applications of BFS

- Message broadcast
  - “piggyback” (watch message bit complexity)
  - complexity: time = O(diameter); msg = O(n)
- Global computation
  - sum, or any accumulation: convergecast
  - complexity: time = O(diameter); msg = O(n)
- Leader election (without knowing diameter)
  - everyone start BFS, finds max UID
  - complexity: time = O(diam); msg = O(n |E|) or O( diam |E|)
- Compute diameter
  - all do BFS; convergecast to find height of each BFS tree; convergecast to find max of all heights

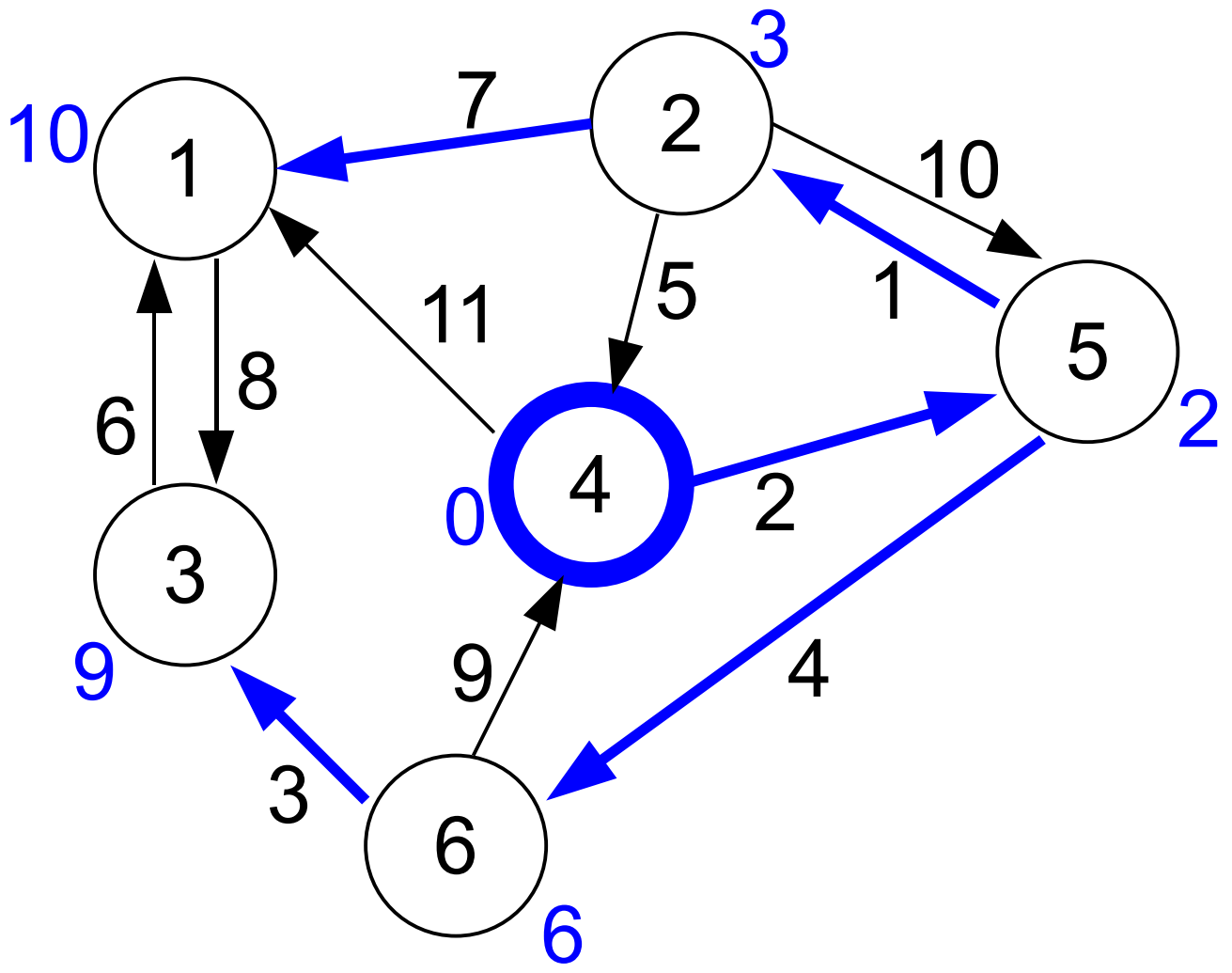
# Shortest paths

- Generalization of BFS
  - assume **weighted** digraph, UIDs, $i_0$
    - weights represent some (communication) cost (known)
    - all nodes know n (need for termination!)
  - require shortest-paths tree rooted at $i_0$
    - paths should have min weight
    - output parent, “distance” from root (by weight)

# Shortest paths



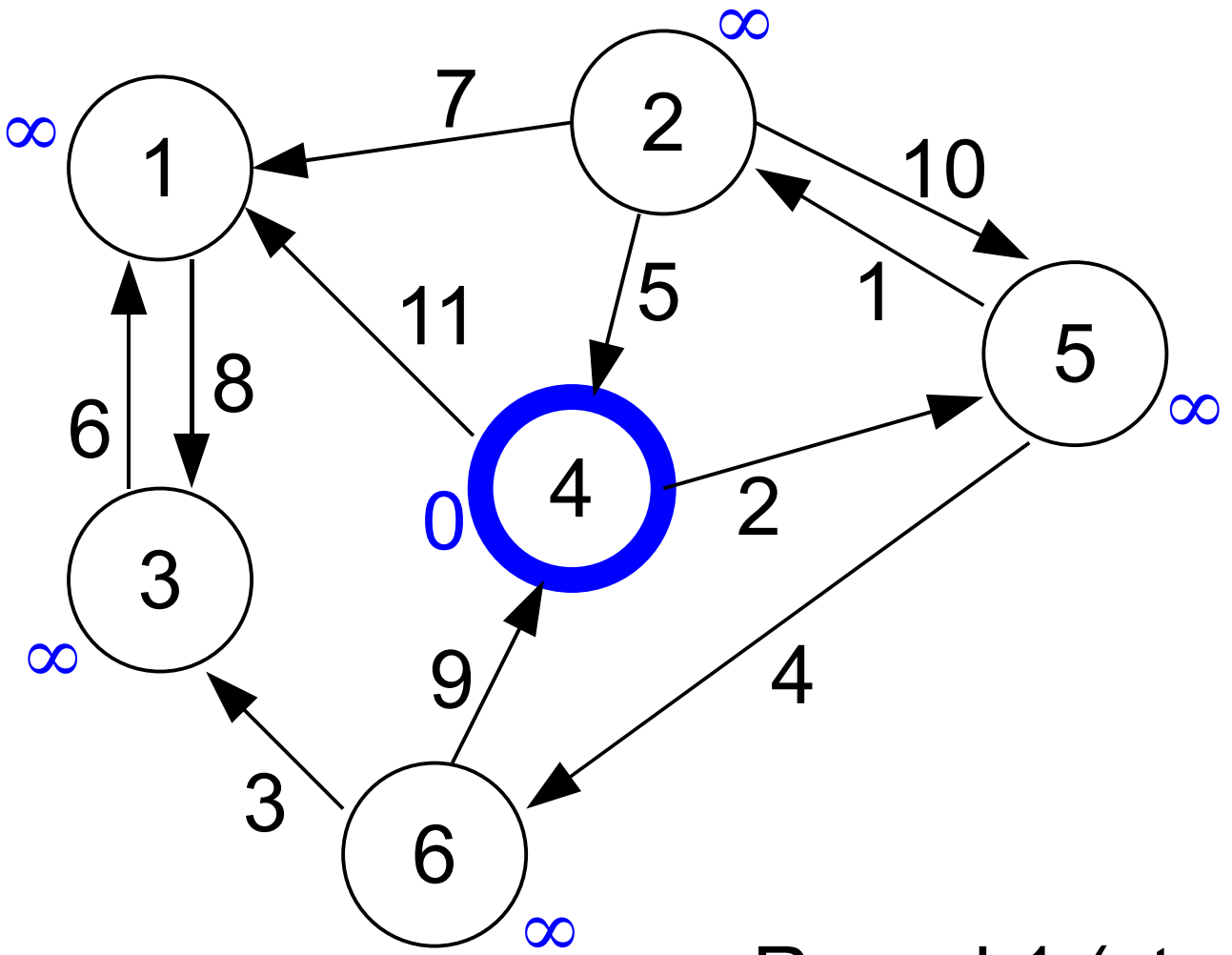
# Shortest paths



# Shortest paths

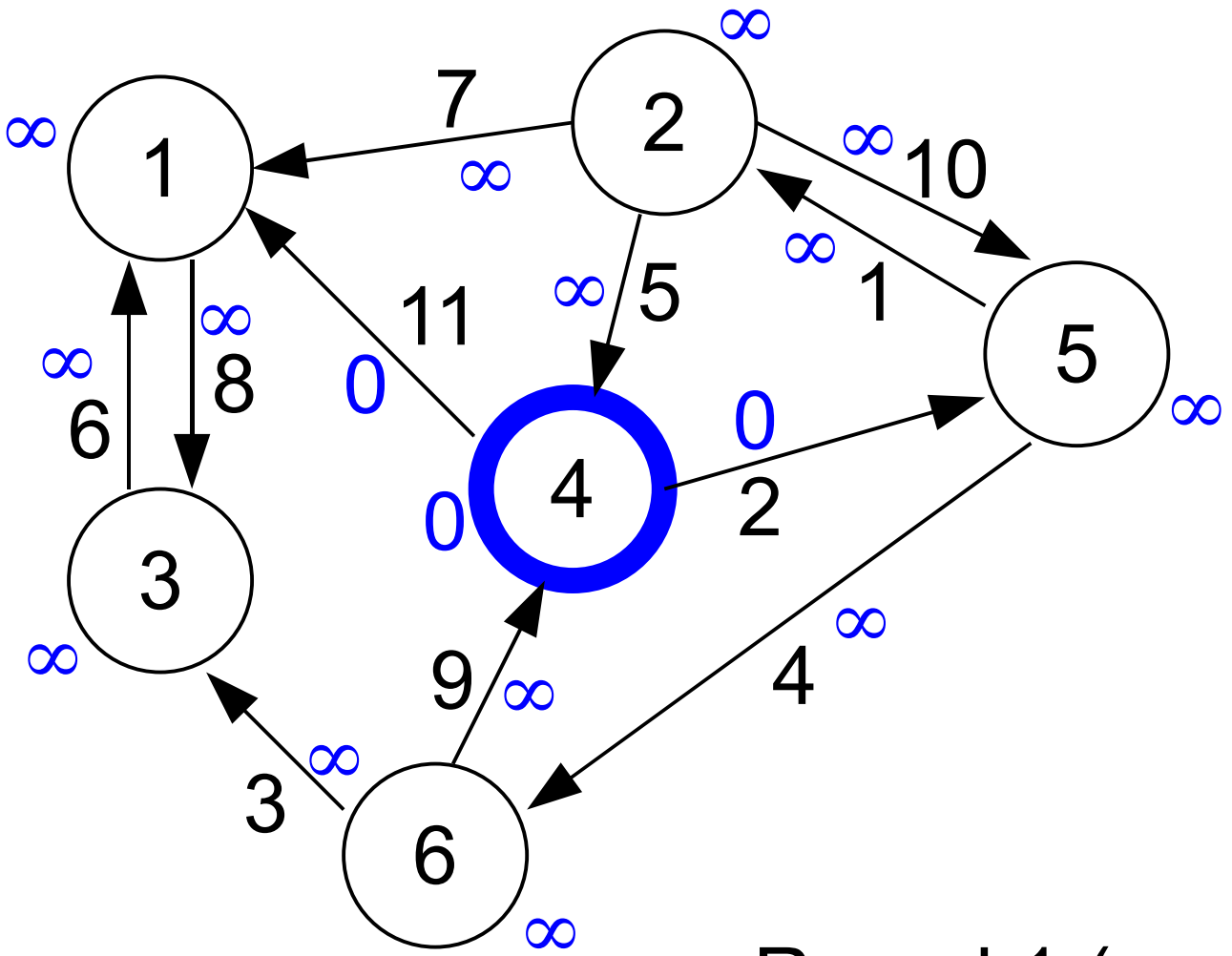
- Bellman-Ford (adapted from sequential alg)
  - "relaxation algorithm"
  - nodes maintain: dist, parent (best so far), round#
    - initially i0 has dist 0, all other ∞; parents all null
  - each round all nodes:
    - send dist to all out-nbrs
    - relaxation: compute new dist = min(dist, $\min_j(d_j+w_{ji})$)
      - update parent if dist changes
  - stop after n-1 rounds

# Shortest paths



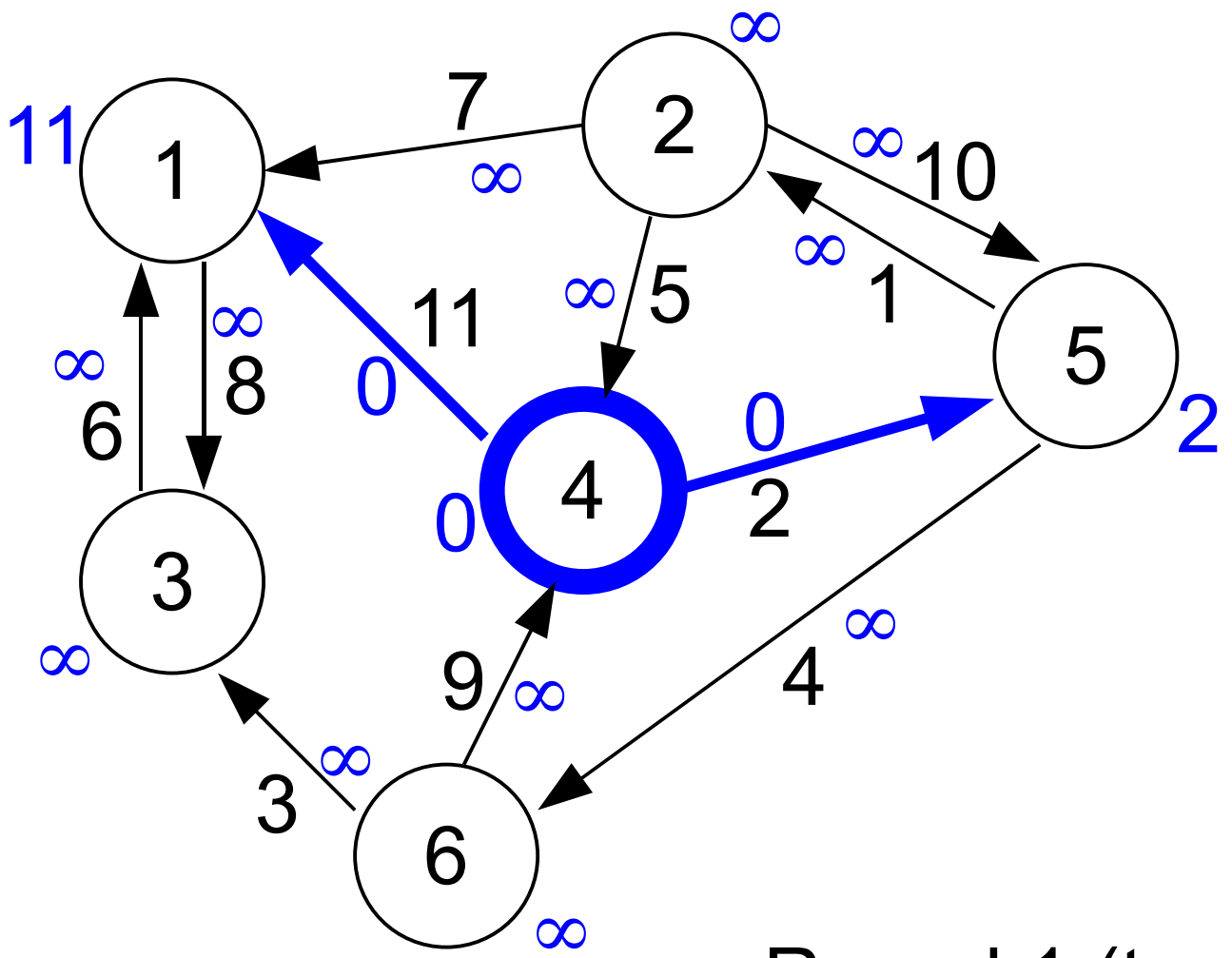

Round 1 (start)

# Shortest paths



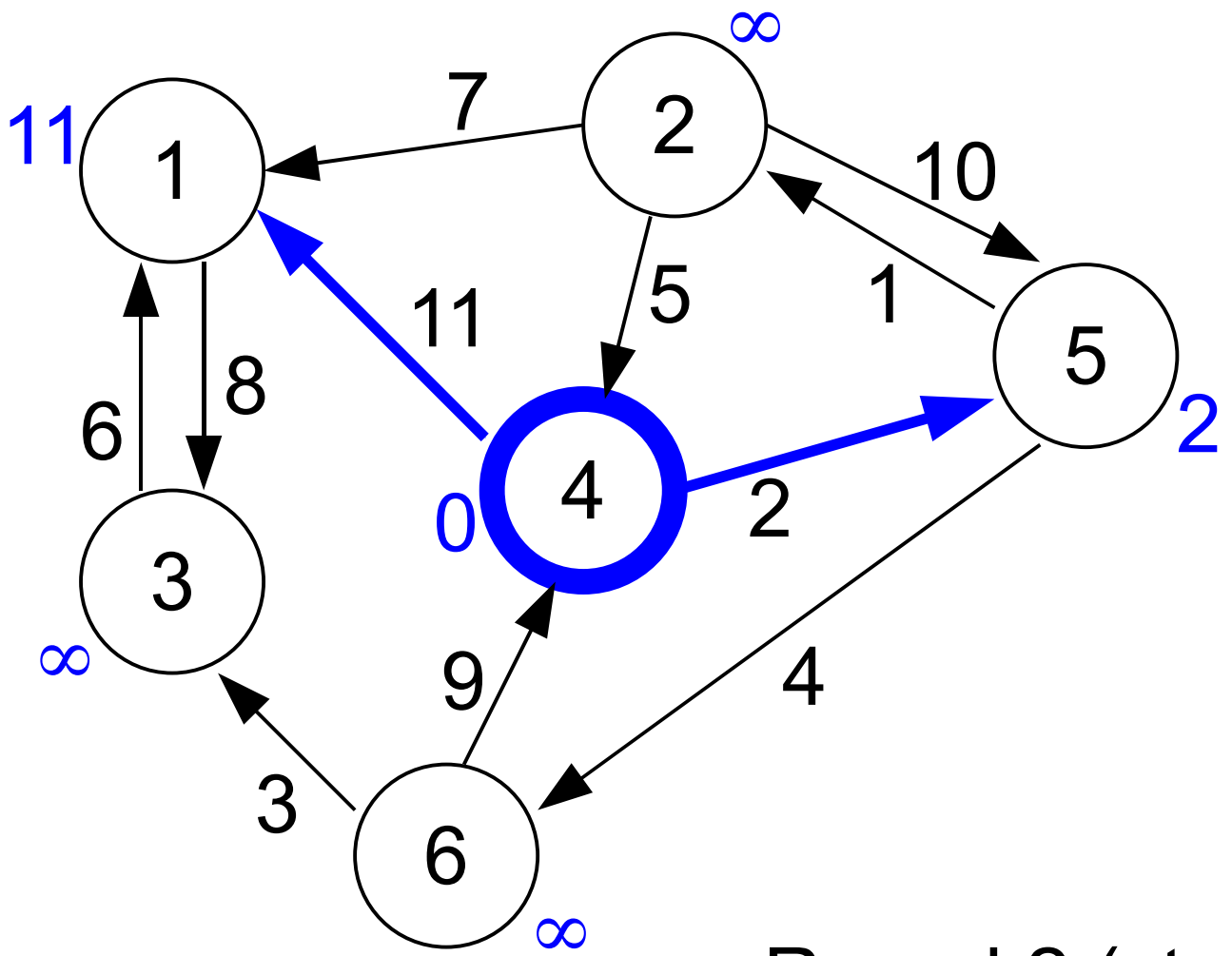

Round 1 (msgs)

# Shortest paths



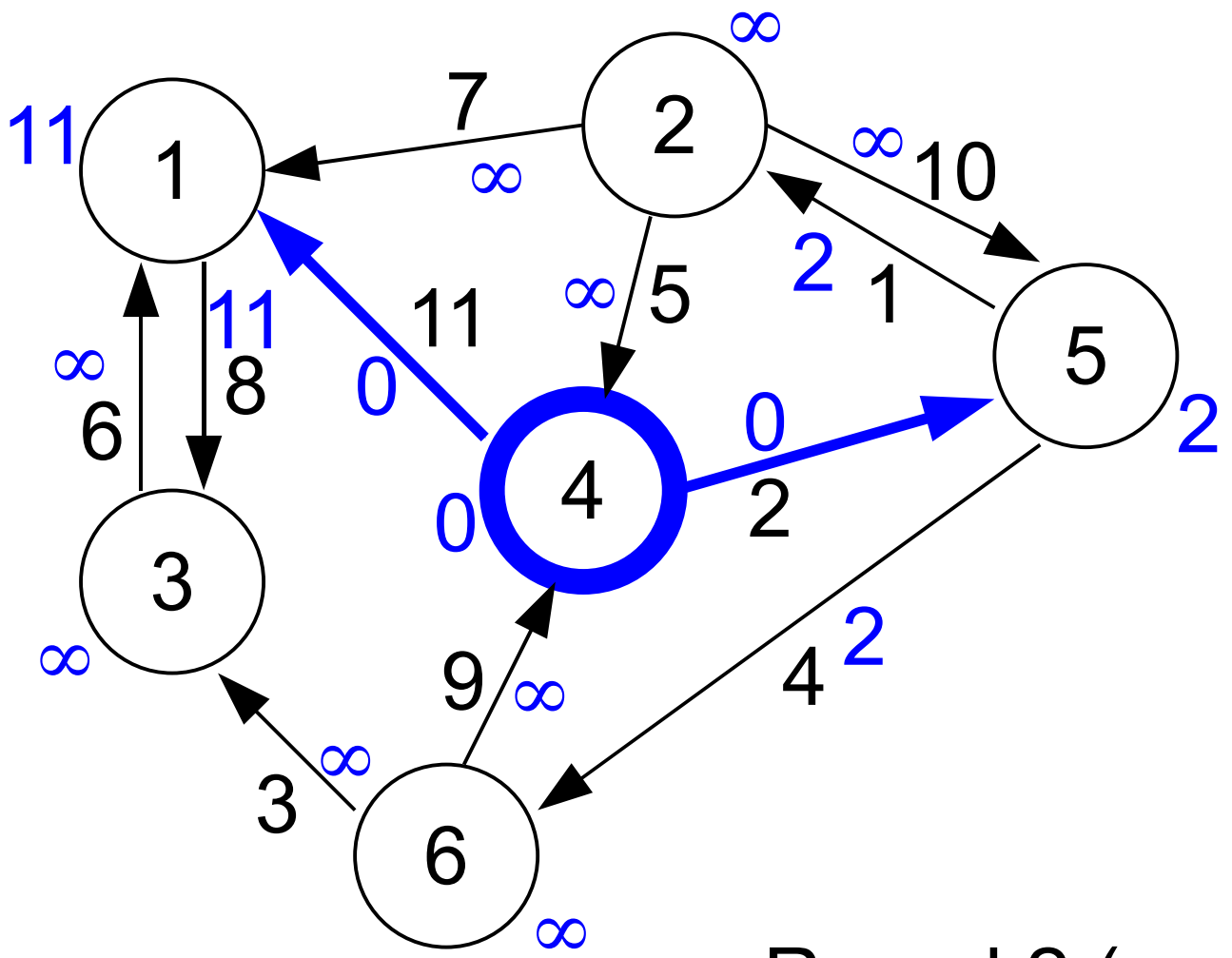

Round 1 (trans)

# Shortest paths



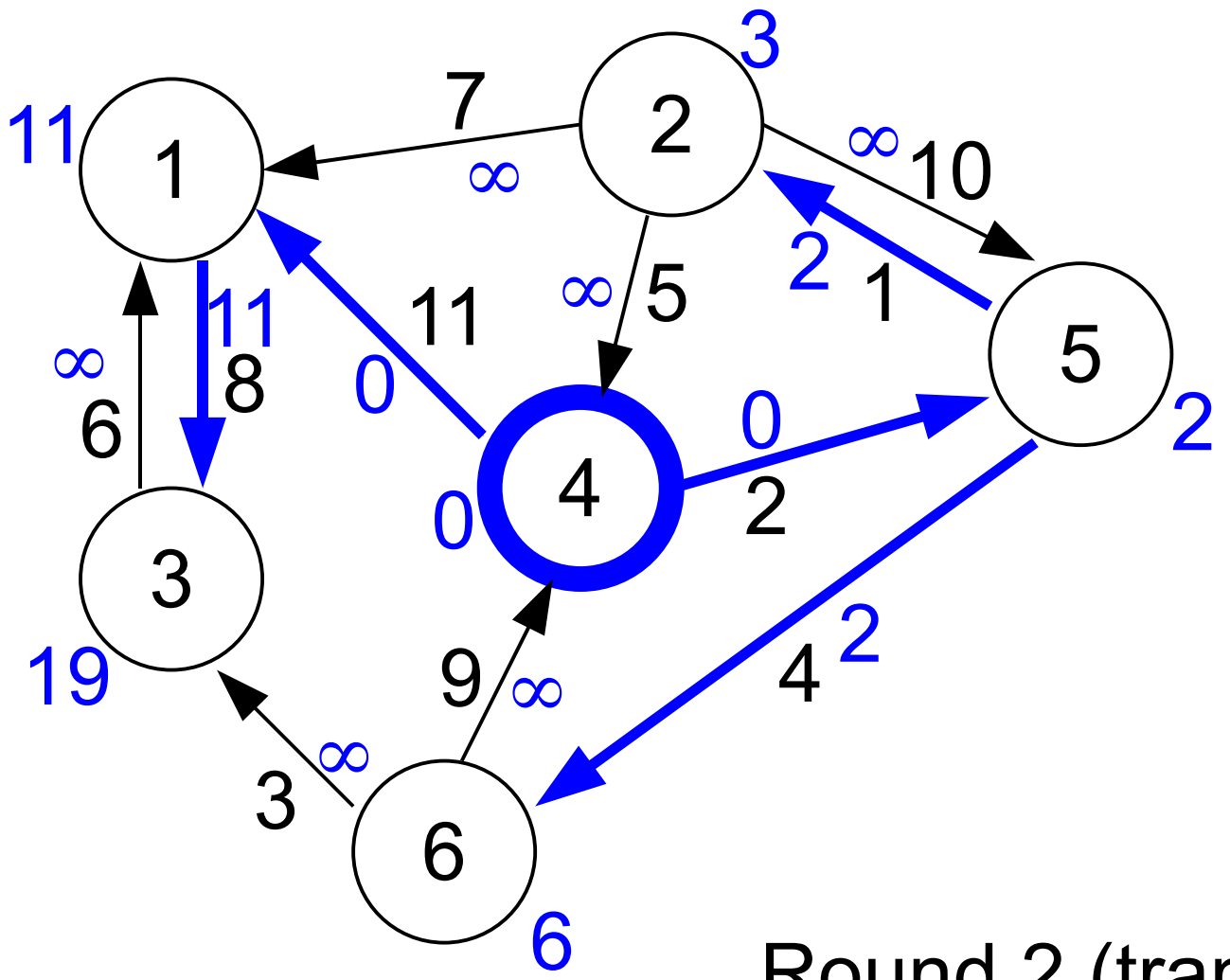

Round 2 (start)

# Shortest paths



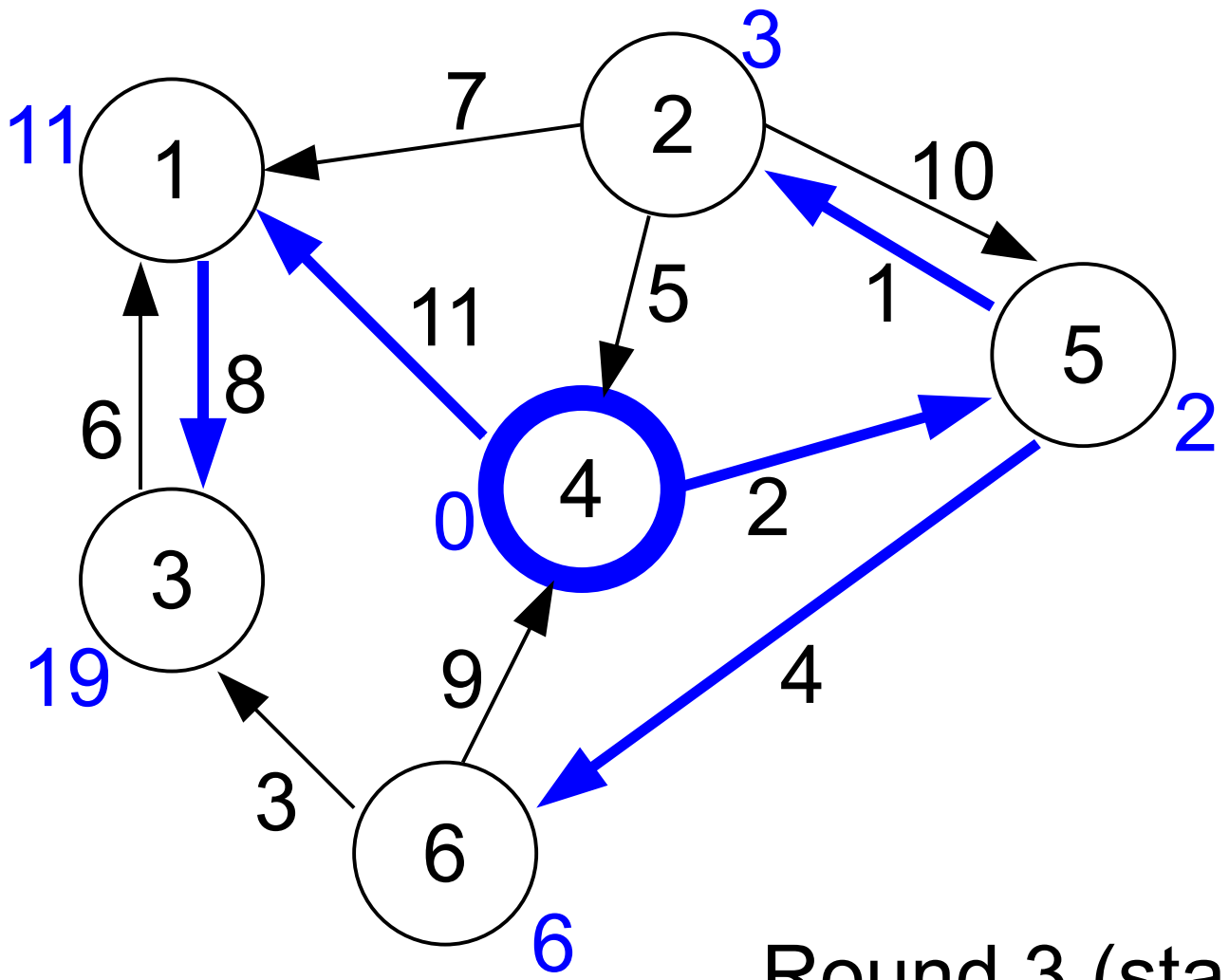

Round 2 (msgs)

# Shortest paths



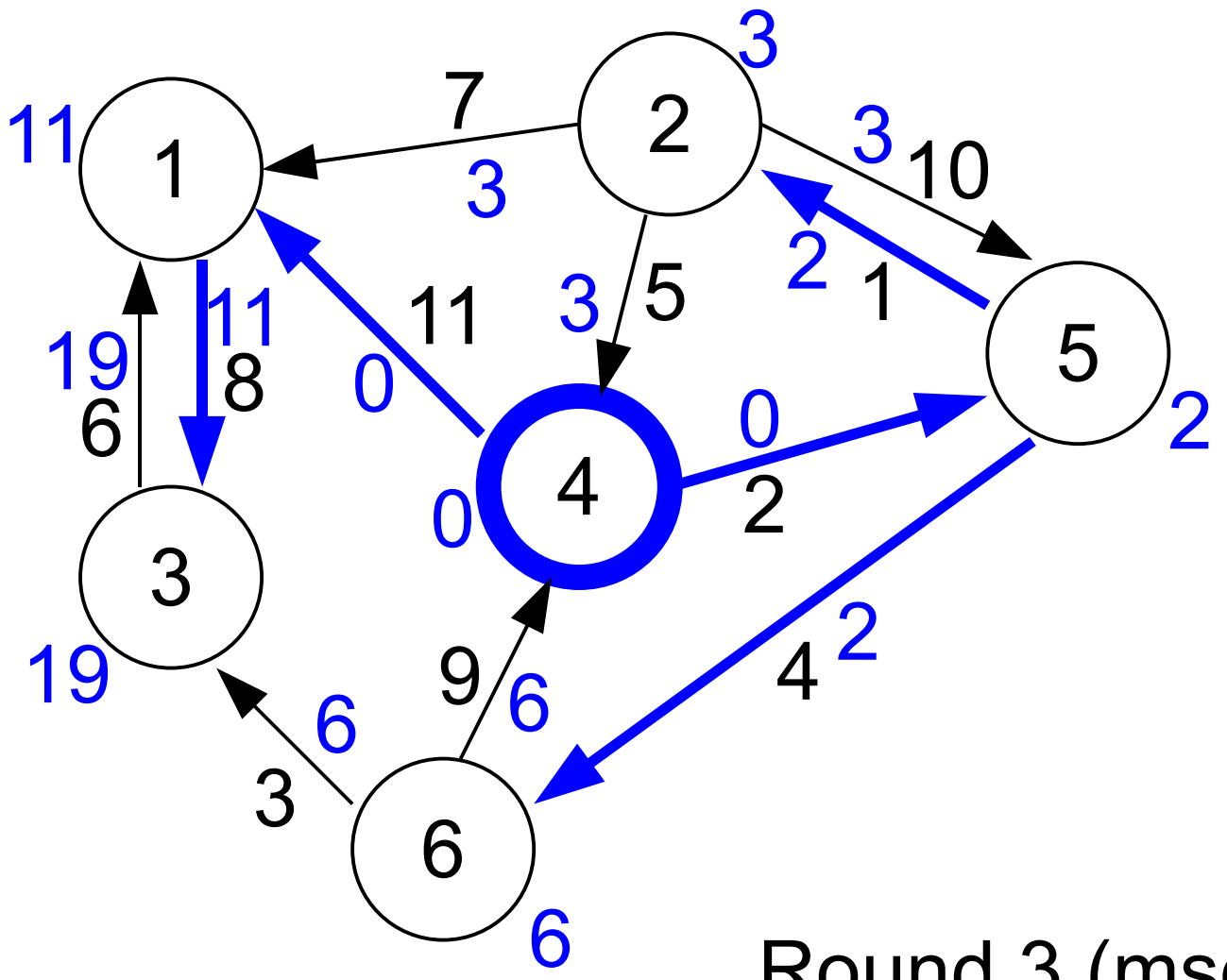

Round 2 (trans)

# Shortest paths



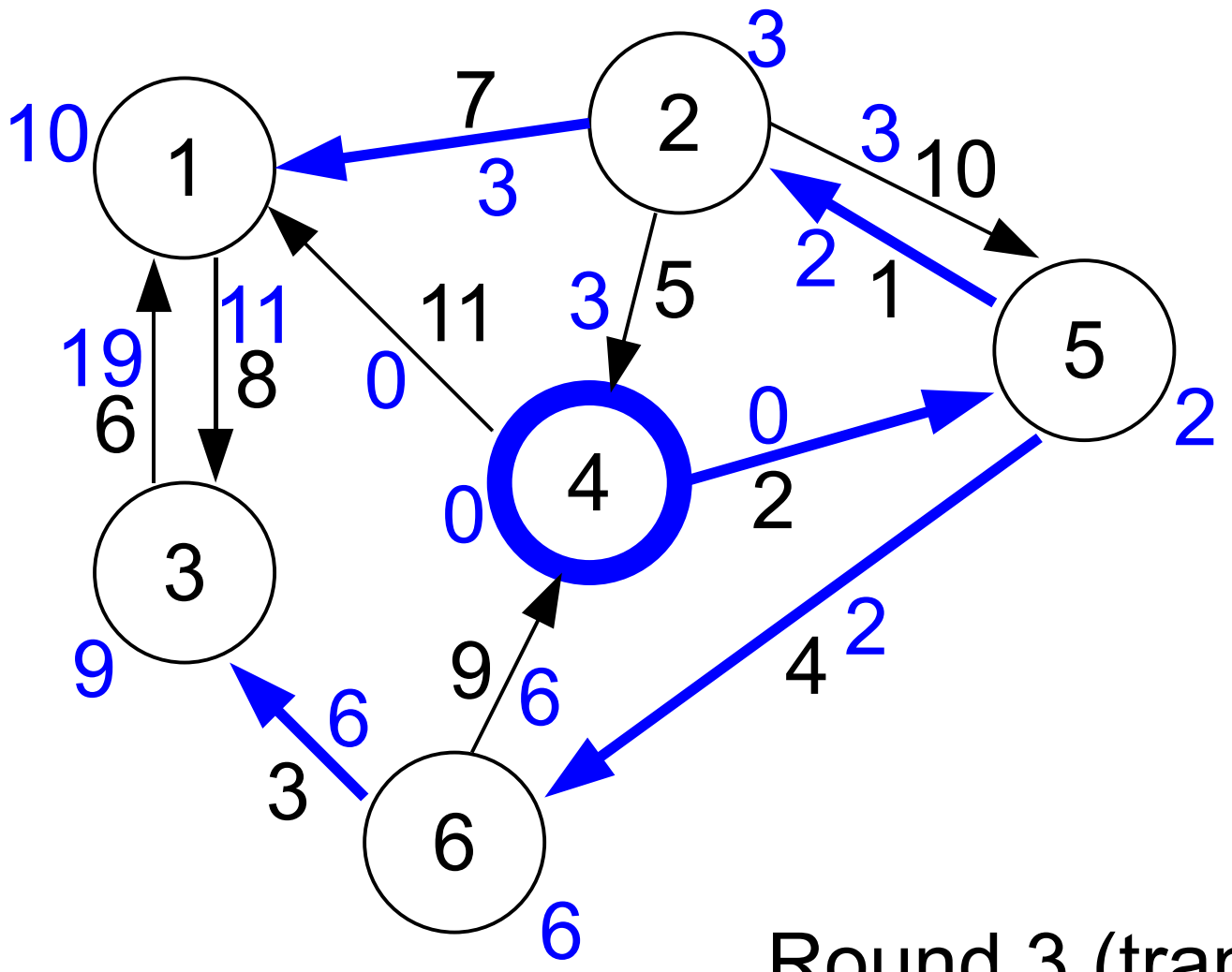

Round 3 (start)

# Shortest paths



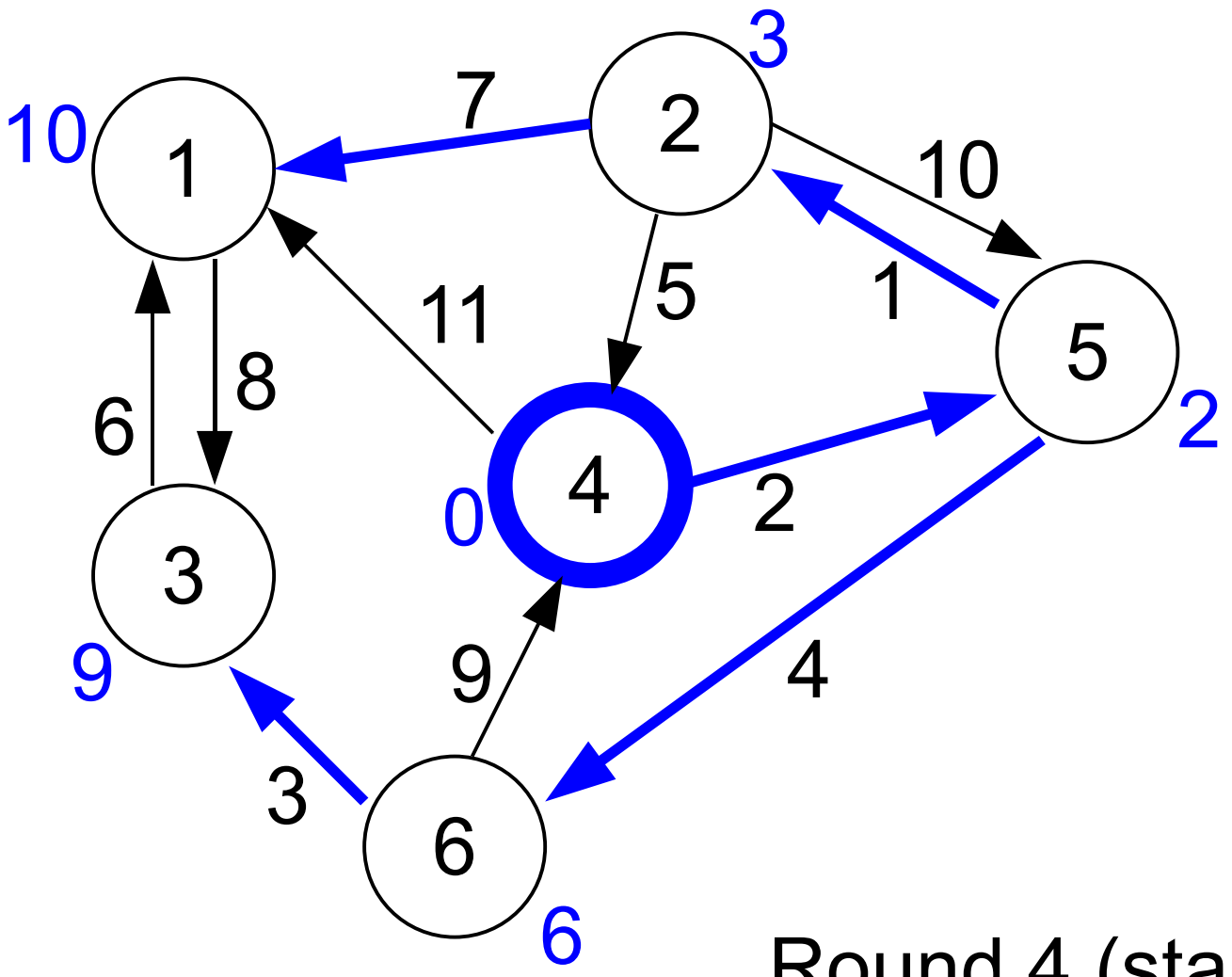

Round 3 (msgs)

# Shortest paths



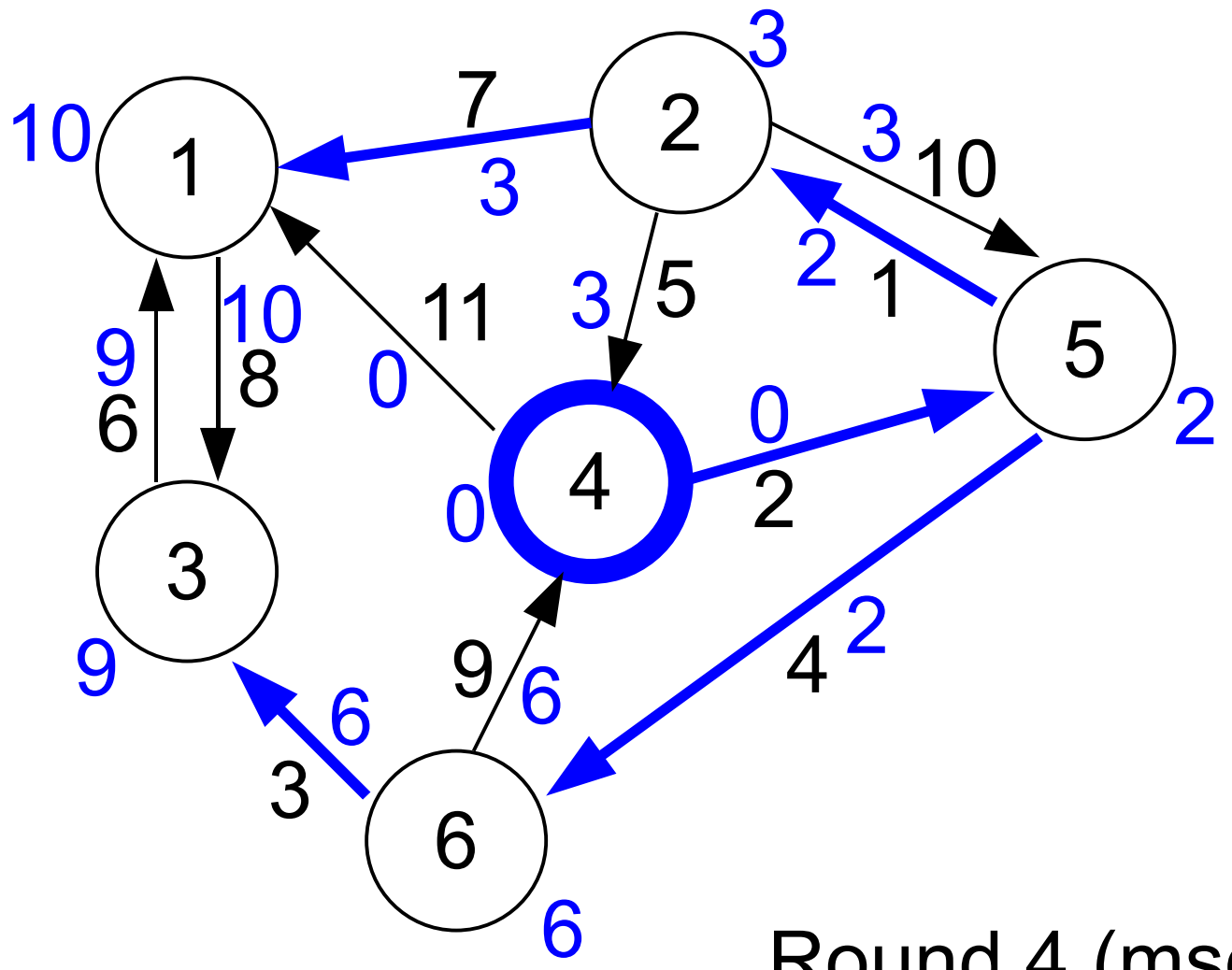

Round 3 (trans)

# Shortest paths



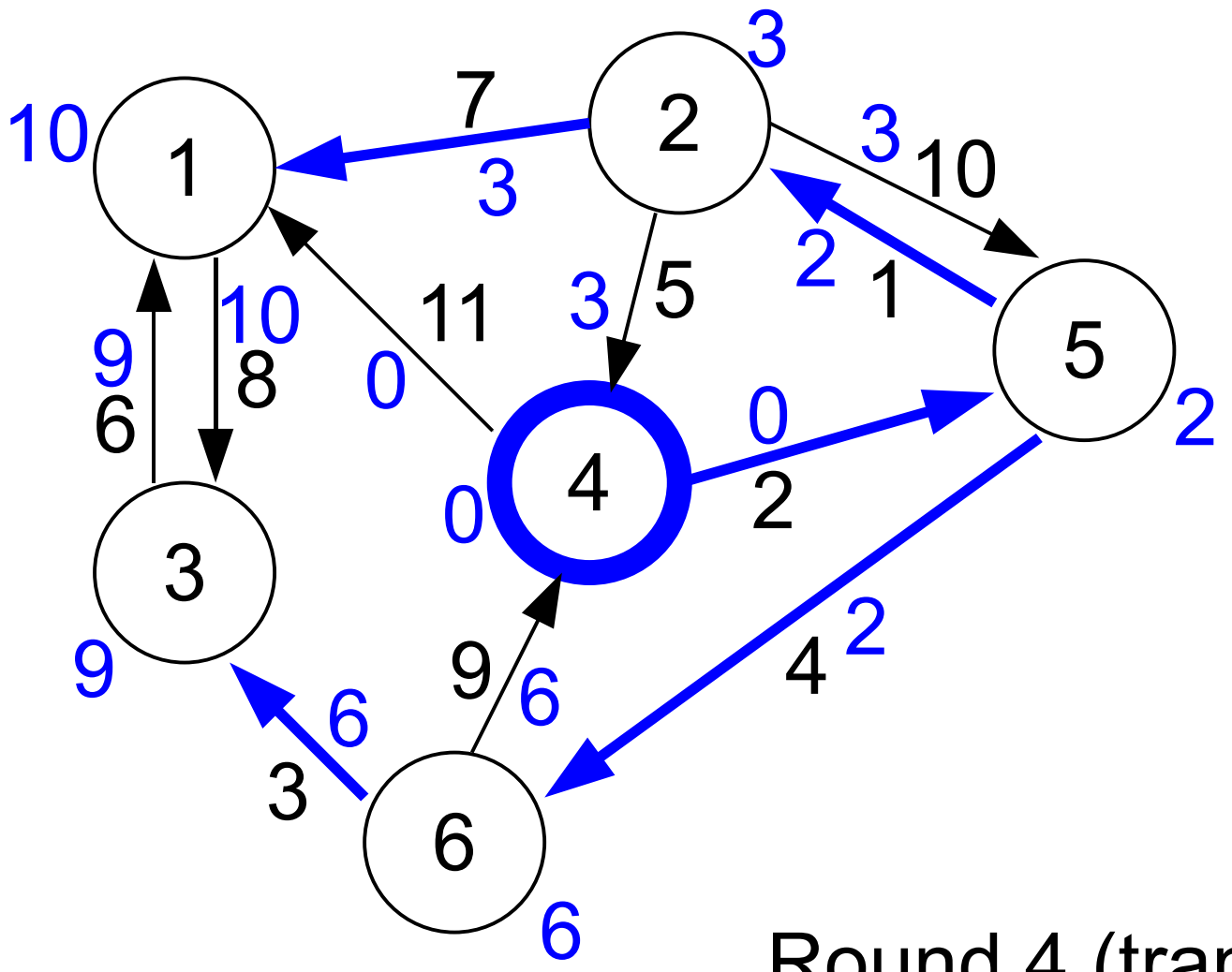

Round 4 (start)

# Shortest paths



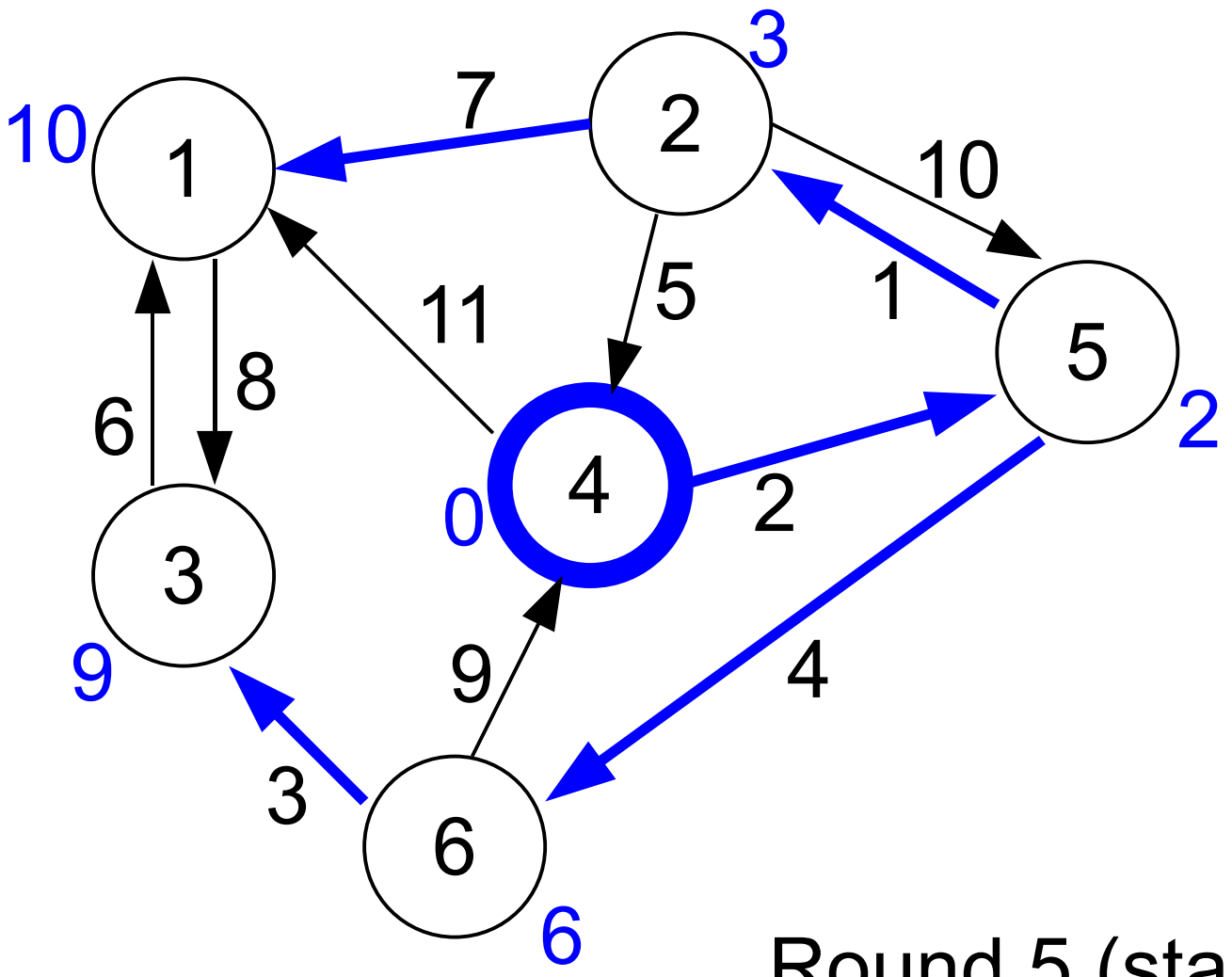

Round 4 (msgs)

# Shortest paths



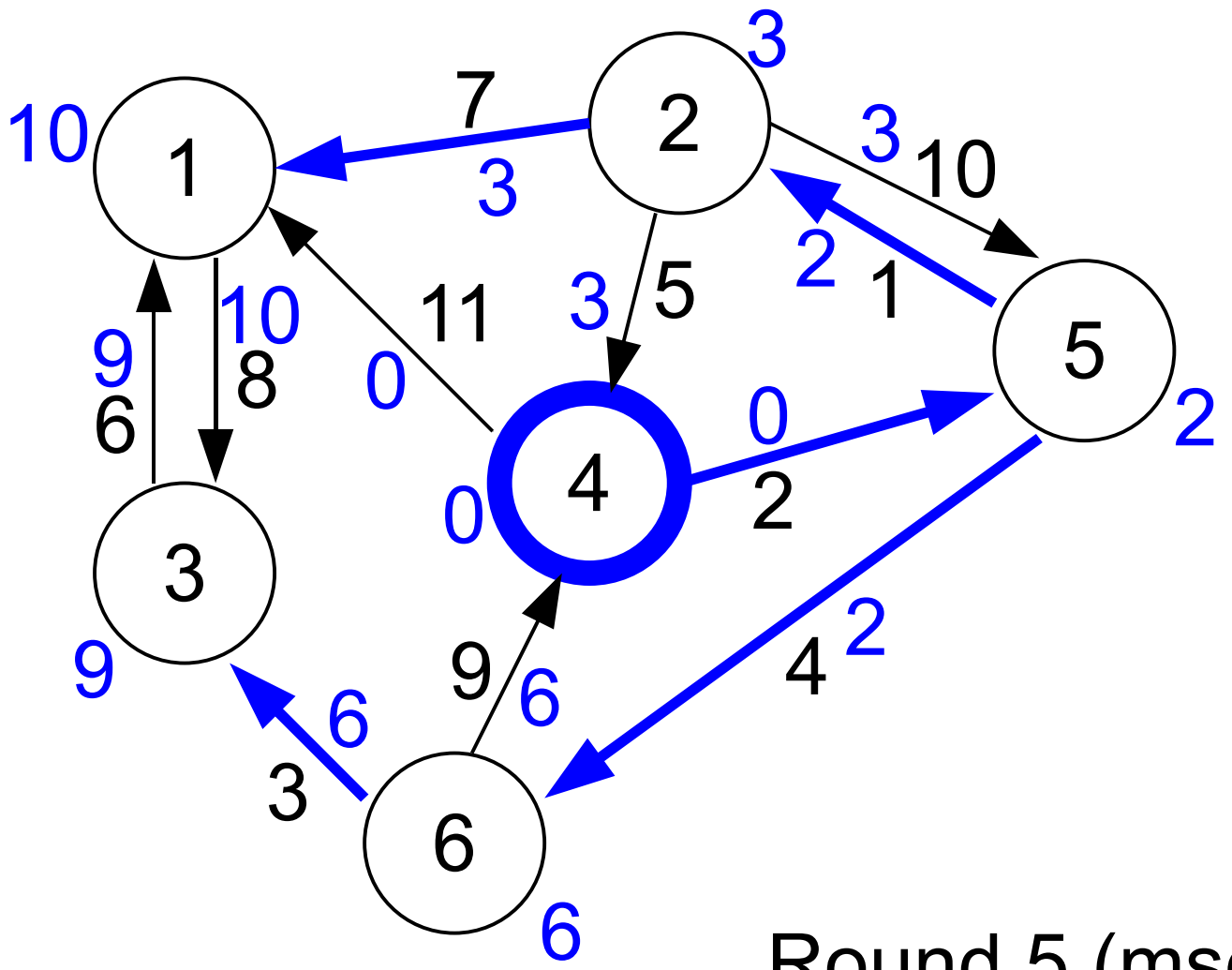

Round 4 (trans)

# Shortest paths

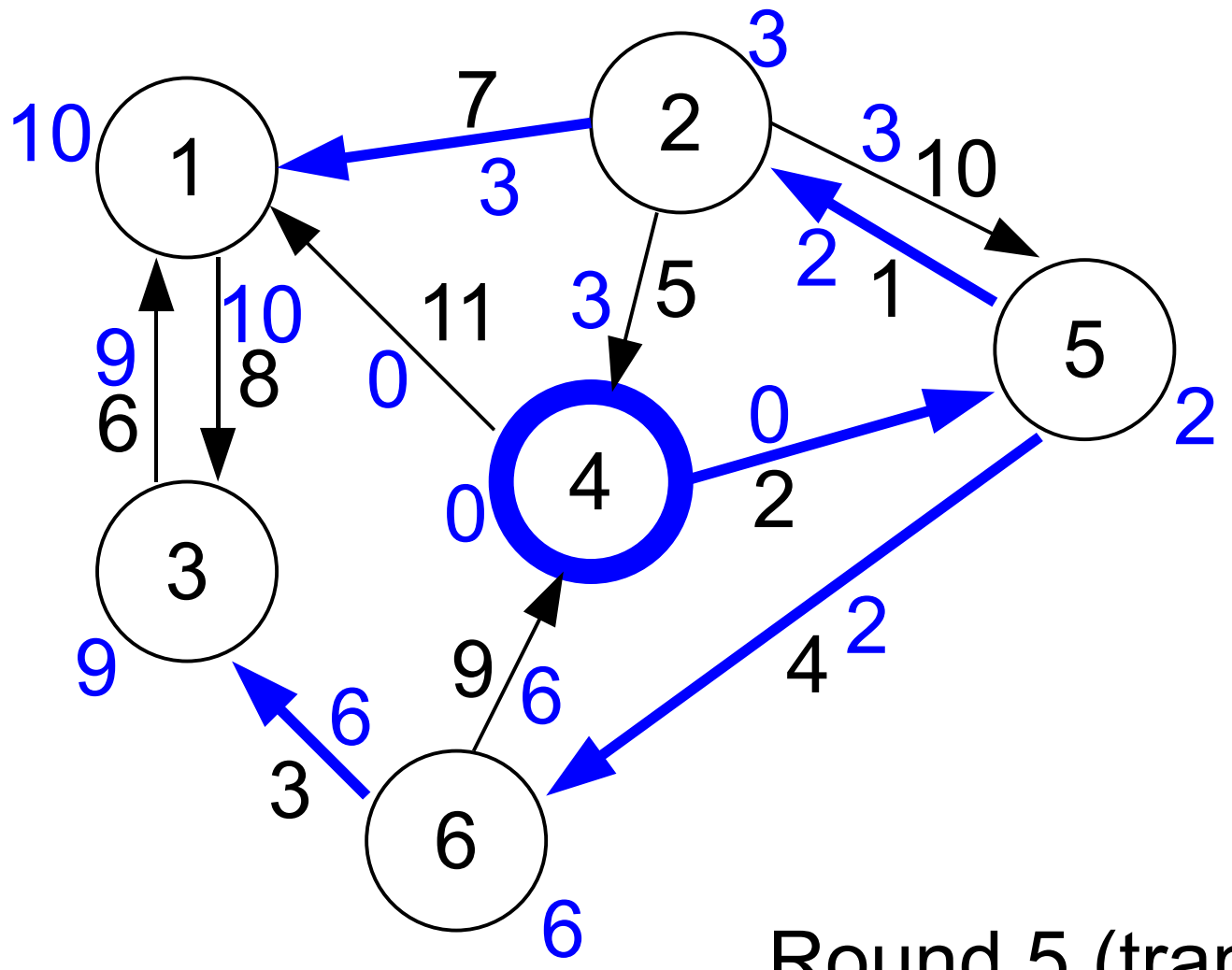



Round 5 (start)

# Shortest paths

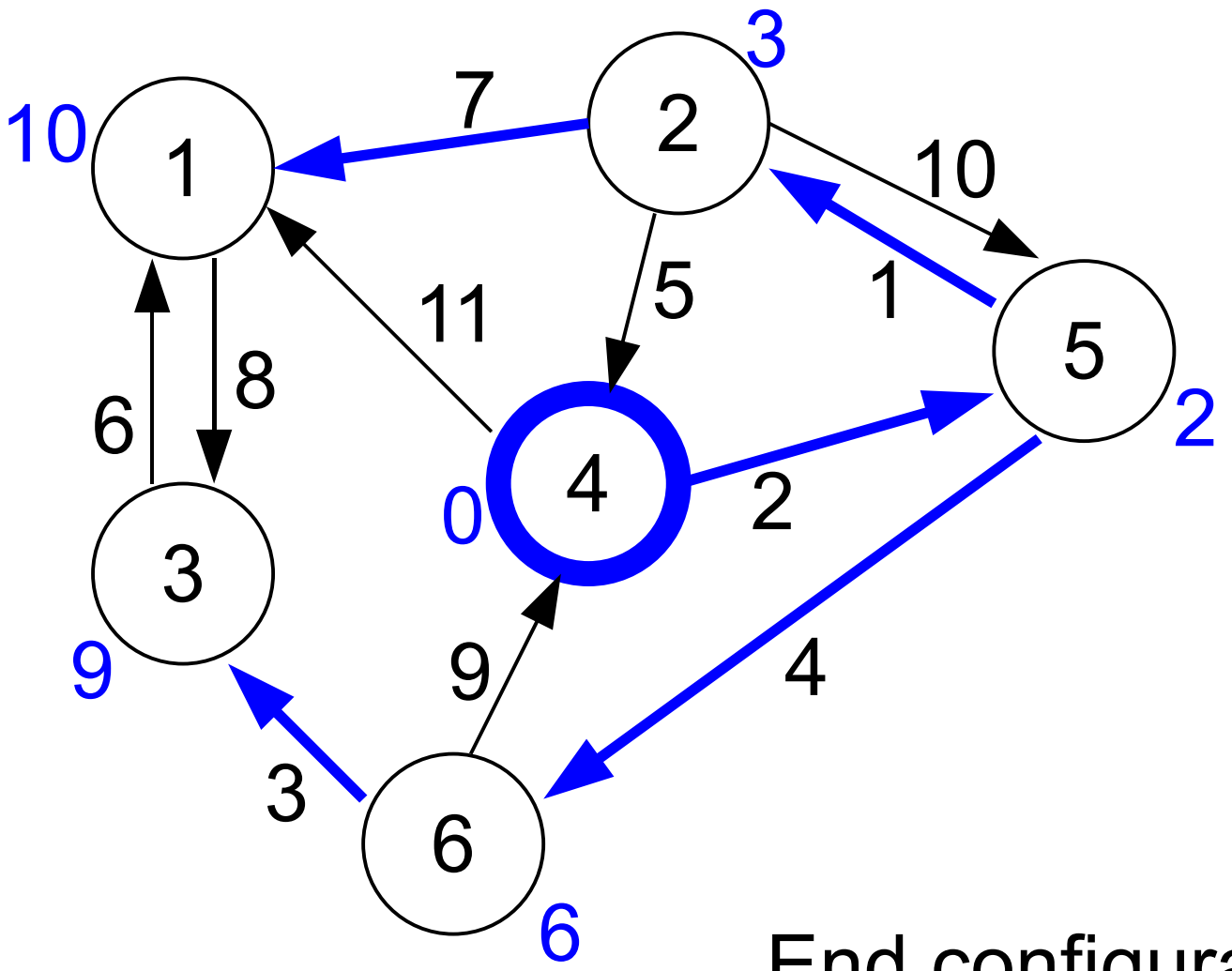


# Shortest paths




Round 5 (trans)

# Shortest paths




End configuration

# Shortest paths

- Complexity: time = n-1; msg = (n-1) |E|
  - can we reduce time complexity? diameter?
  - what about message complexity?
- Proof?

# Shortest paths

- Complexity: time = n-1; msg = (n-1) |E|
  - can we reduce time complexity? diameter?
  - what about message complexity?
- Proof?
- Correctness condition?

# Shortest paths

- Complexity: time = n-1; msg = (n-1) |E|
  - can we reduce time complexity? diameter?
  - what about message complexity?
- After round n-1, for each process i
  - $dist_i$ = shortest distance from $i_0$
  - $parent_i$ = predecessor on shortest path from $i_0$
- Proof?

# Shortest paths

- Invariant: after r rounds:
  - every process i has its dist (& parent) corresp to shortest path from $i_0$ to i with at most r edges
- Proof (by induction):
  - base case: trivial for r = 0
  - inductive step:
    - fix i, let p be pred on shortest path from $i_0$ with ≤ r edges
    - by ind hyp, after round r-1, $dist_p$ and $parent_p$ corresp to shortest path from $i_0$ to p with at most r-1 edges
    - $dist_i(r) = dist_p(r-1) + w_{pi}$ correct by “optimal substructure”

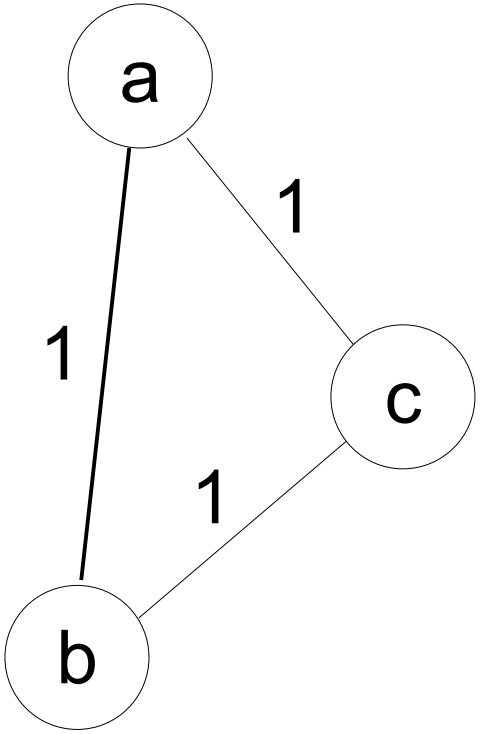
# Minimum spanning tree

- Another classic problem (lots of seq algs)
- Assume
  - weighted **undirected** graph (bidirectional comm)
    - all weights nonnegative
  - processes have UIDs
  - know weights of incident edges, bound on n
- Require
  - each process knows which incident edge is in MST

# Minimum spanning tree

- Graph theory definitions (for undirected graphs)
  - **tree**: connected acyclic graph
  - **spanning**: property of a subgraph that it includes all nodes of a graph
  - **forest**: an acyclic graph (not necessarily connected)
  - **component**: a maximal connected subgraph
- Common strategy for computing MST:
  - start with trivial spanning forest (n isolated nodes)
  - repeatedly (n-1 times): for any component, add the minimum-weight *outgoing* edge (MWOE) of that component to E
  - all components can choose simultaneously, except...

# Minimum spanning tree



# Minimum spanning tree

- Assume for now that weights are unique
  - implies  there is a unique MST
  - components can choose concurrently
- GHS (Gallager Humblet Spira) algorithm
  - very influential (Dijkstra prize)
  - designed for asynchronous setting: simplified here
  - we will revisit it in asynchronous networks

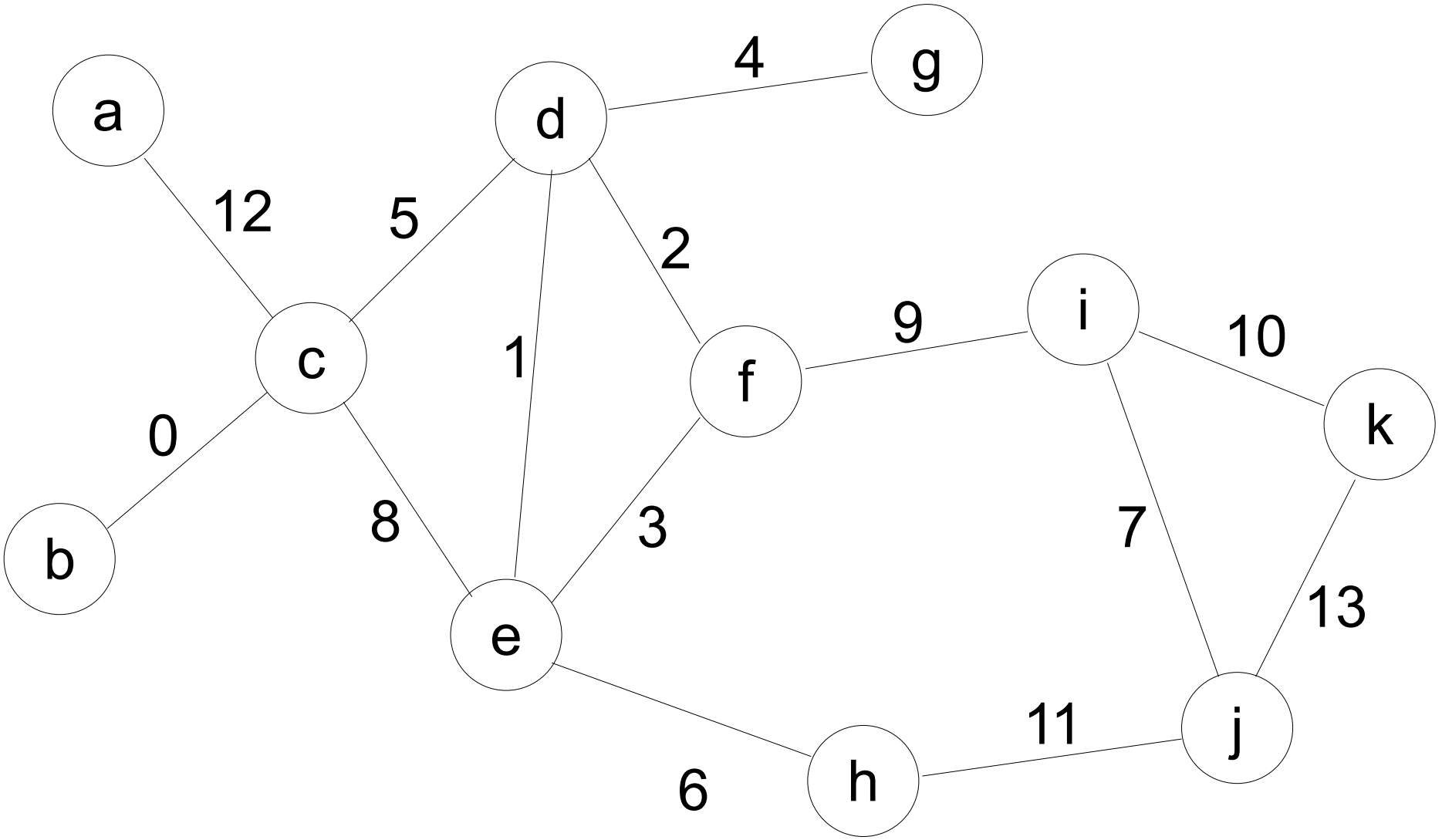
# Minimum spanning tree

- GHS
  - proceeds in phases, each with O(n) rounds
    - length of phases is fixed; this is what n is used for
  - in each phase, graph is partitioned into components
    - phase k component has size at least $2^k$
    - components identified by UID of leader
    - each component is a tree rooted at leader
    - every phase k+1 component contains of two or more phase k components

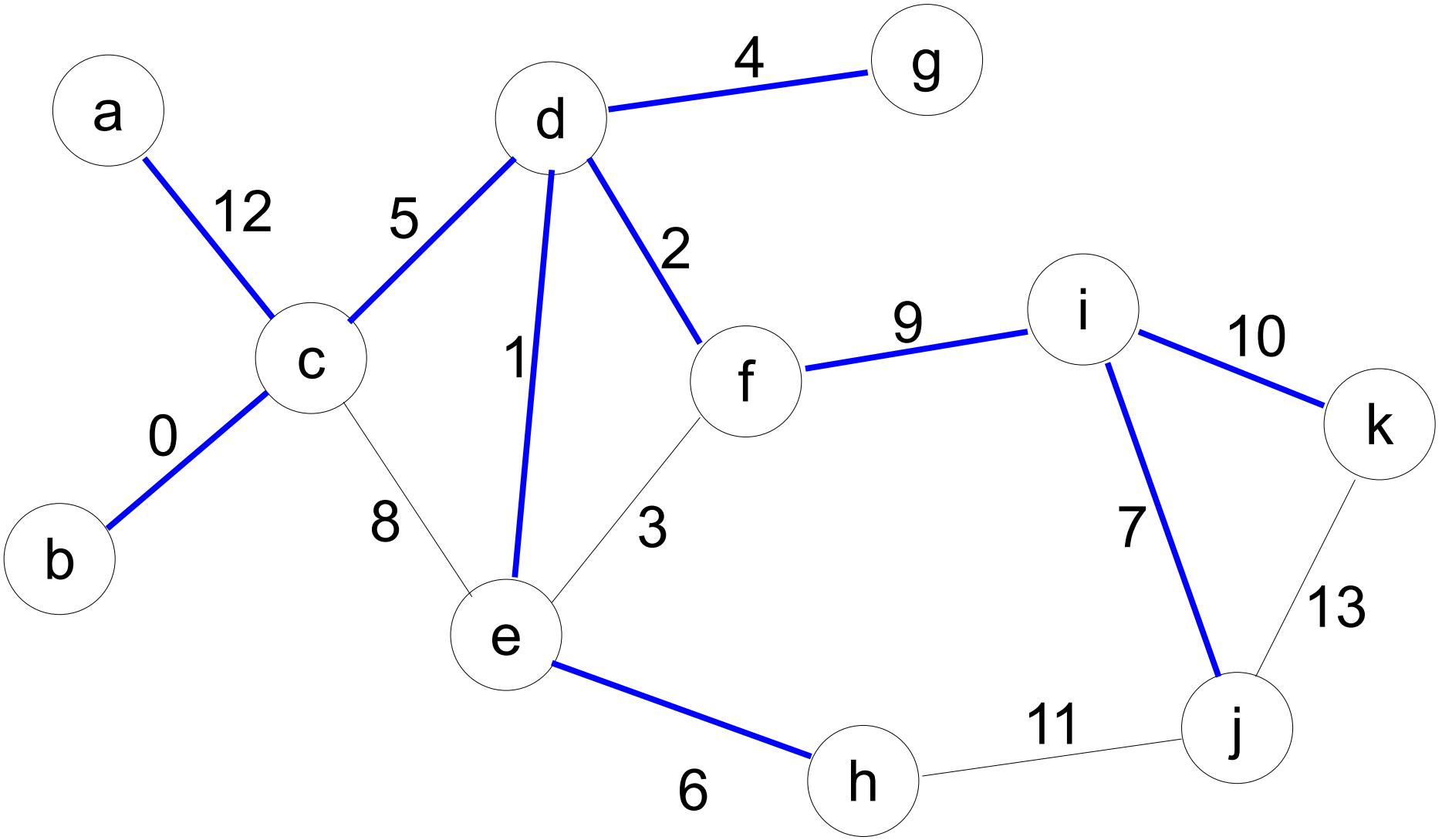
# Minimum spanning tree

- GHS phases consists of multiple stages
  - leader finds MWOE of its component
    - broadcast search (via tree edges)
    - convergecast MWOE (via tree edges)
    - leader chooses minimum weight edge
  - combine components joined by MWOEs
    - inform nodes at either end of MWOEs of merger
    - choose new leader
      - larger UID adjacent to “shared” MWOE
    - broadcast to new (merged) component
- GHS terminates when no more outgoing edges

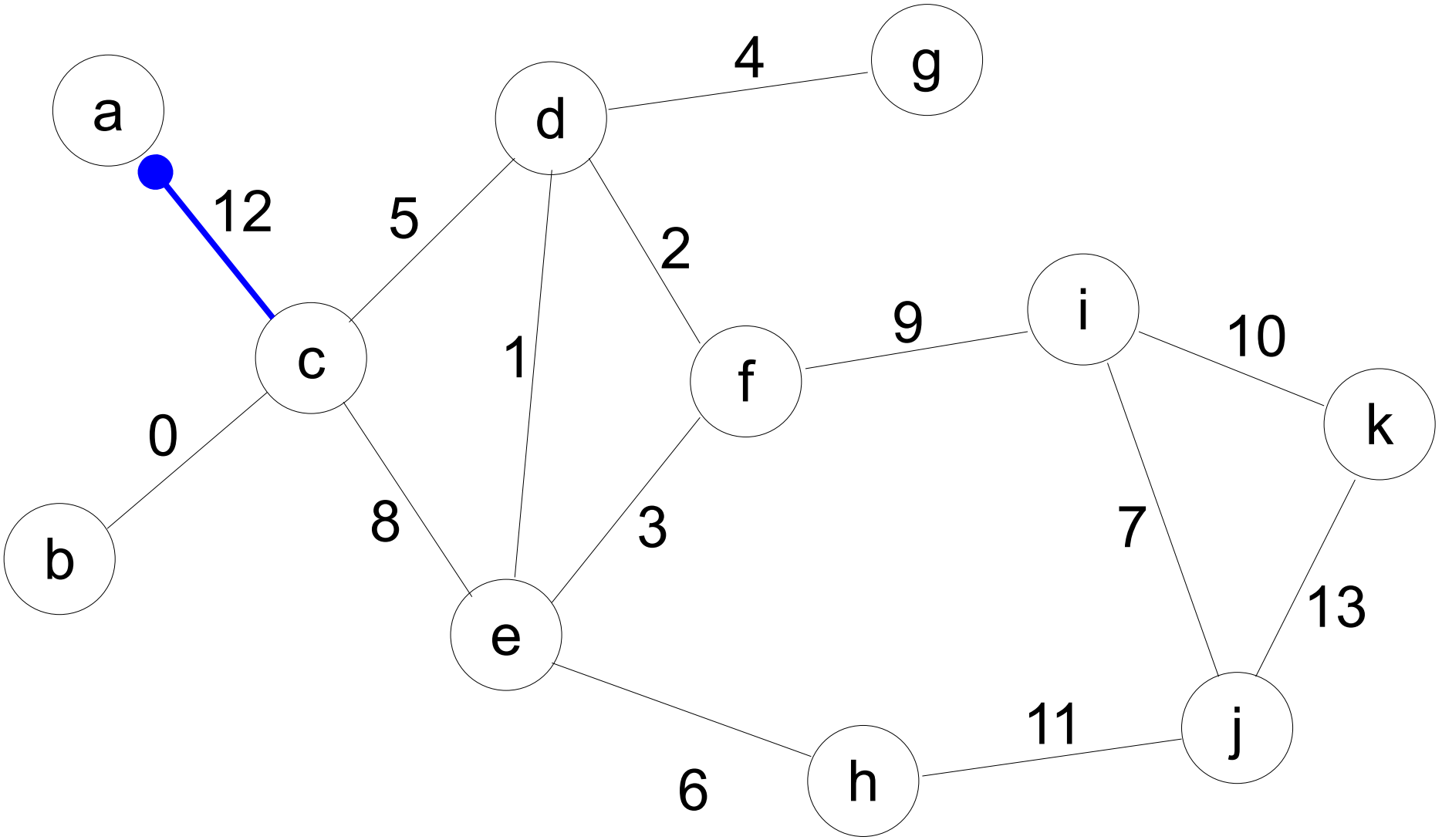
# Minimum spanning tree



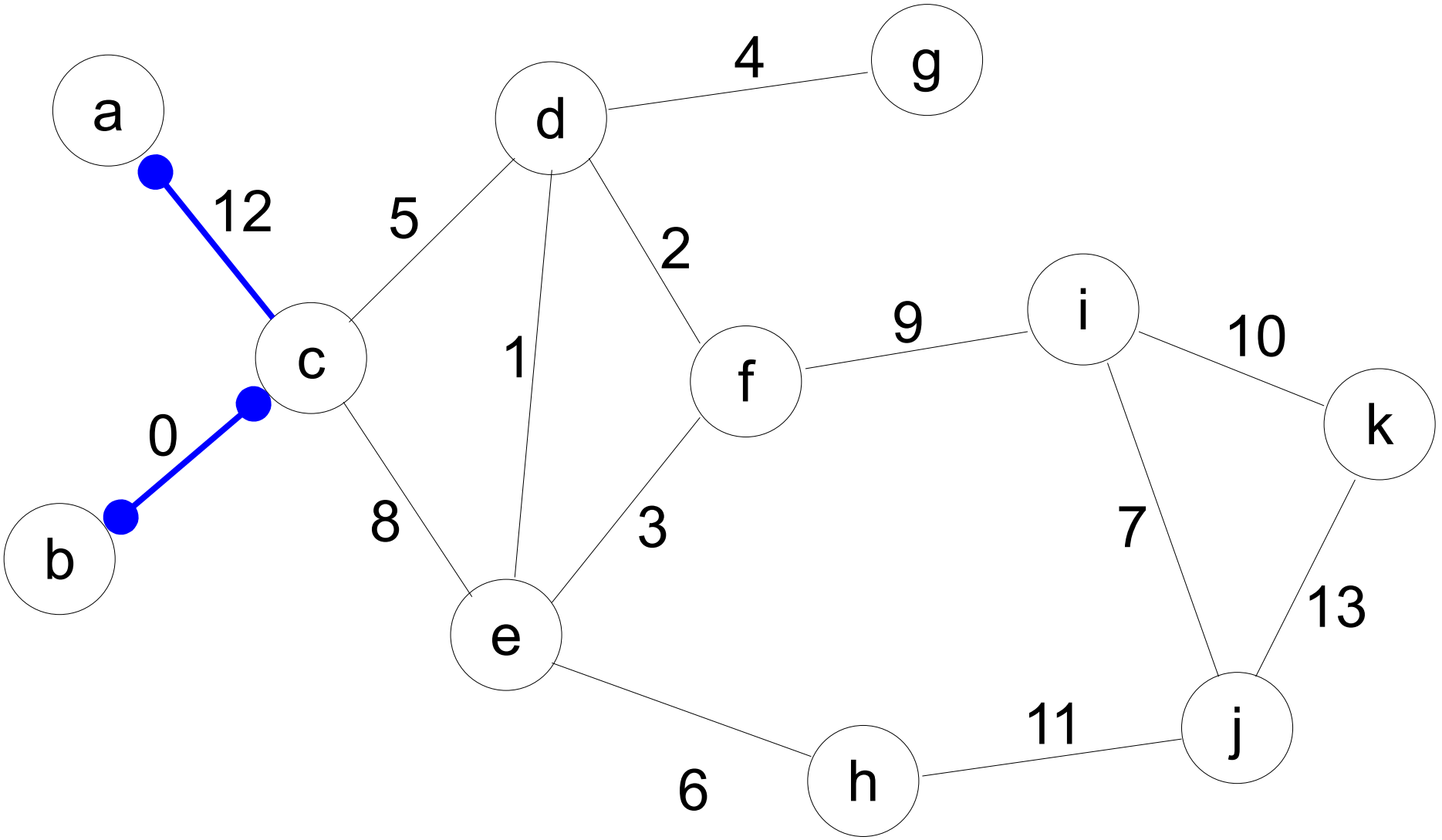
# Minimum spanning tree



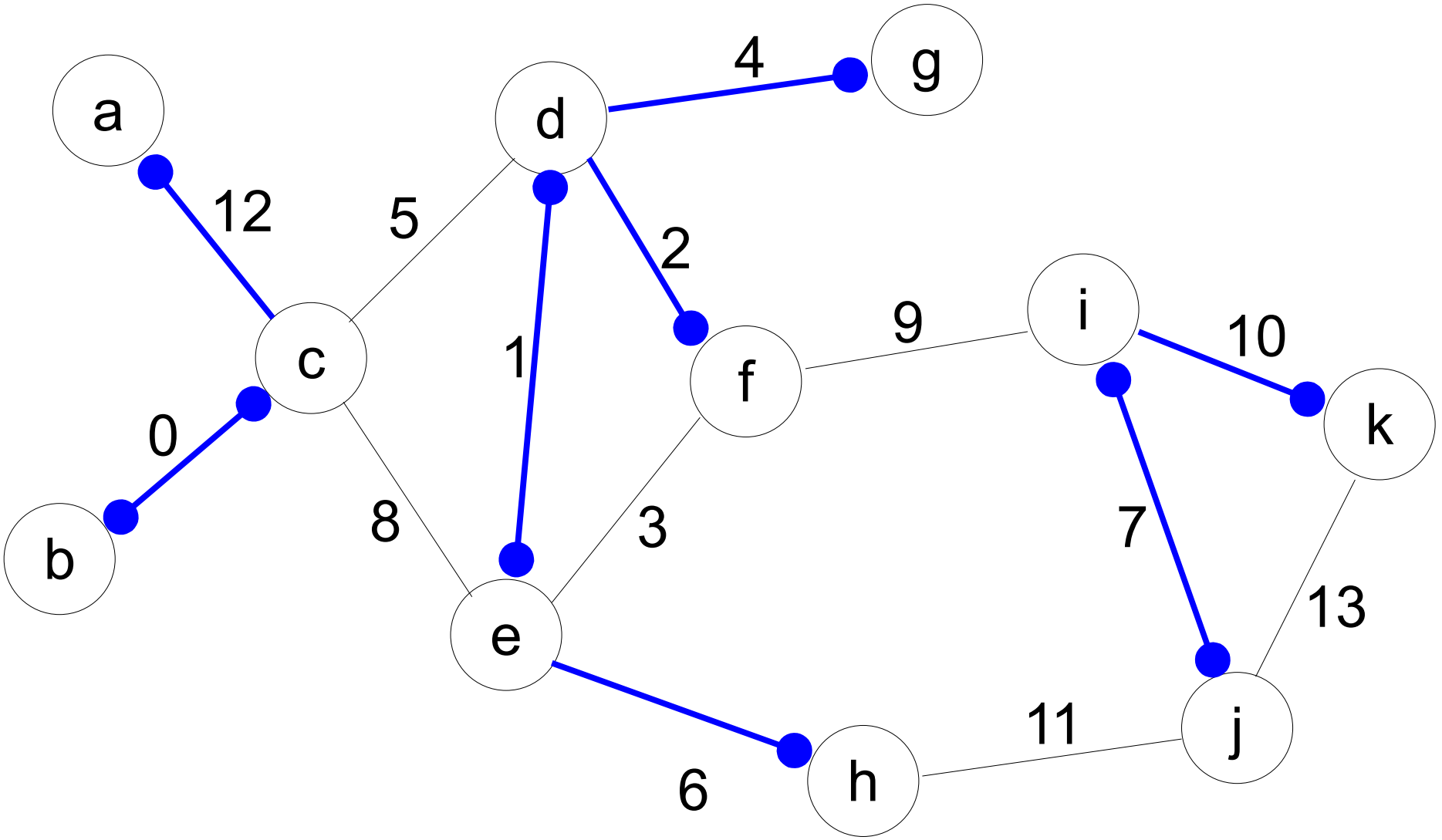
# Minimum spanning tree



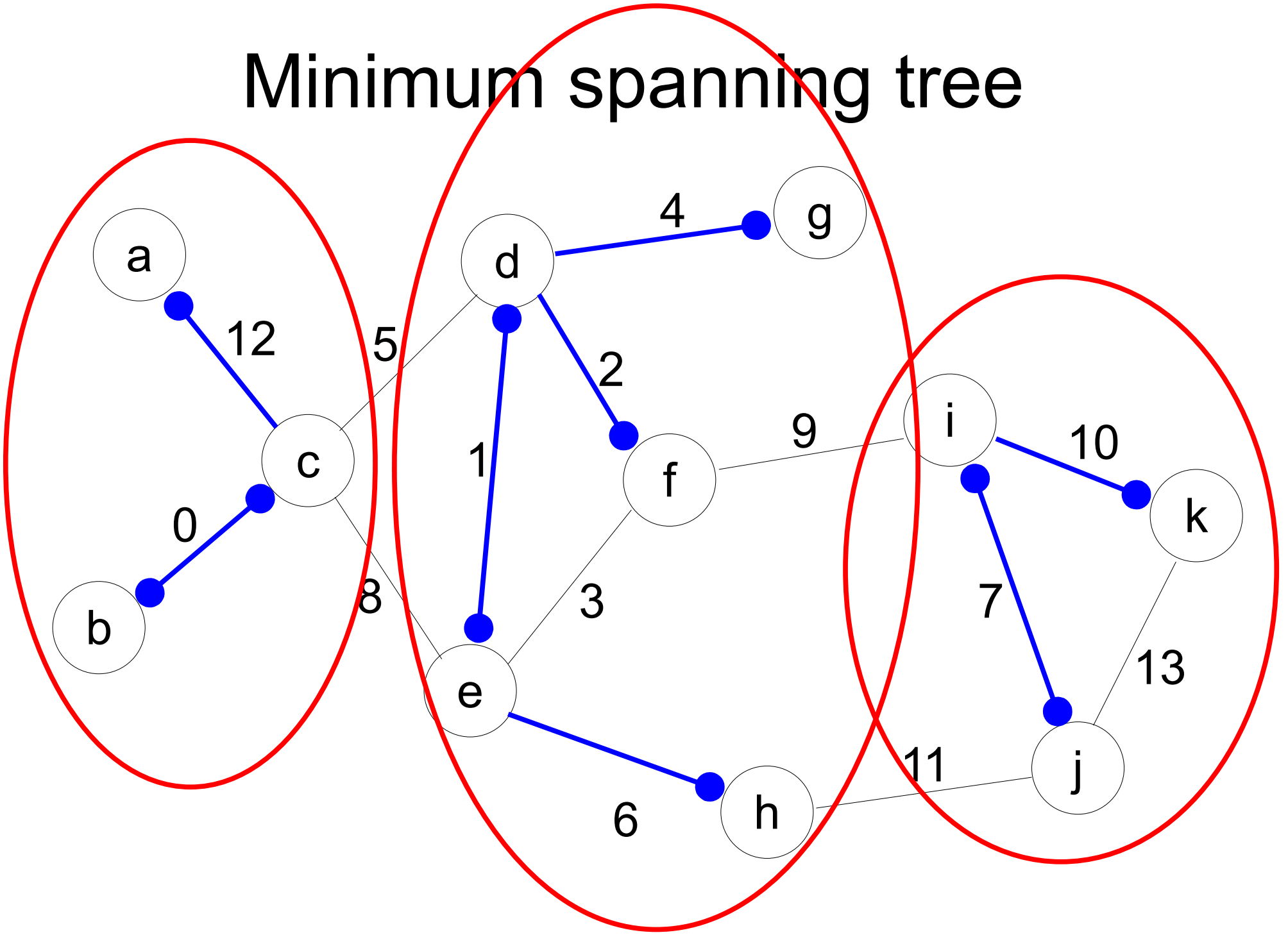
# Minimum spanning tree



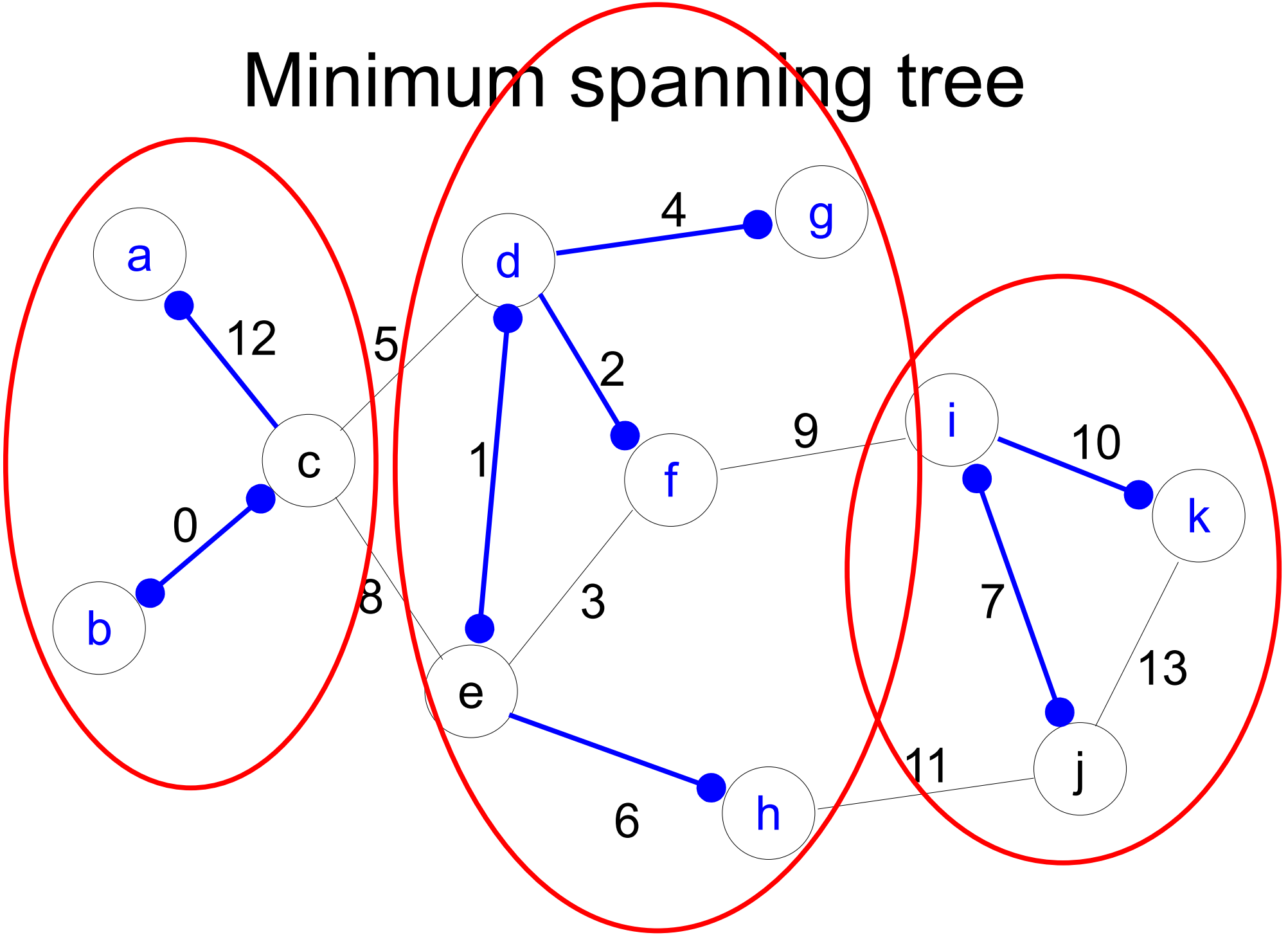
# Minimum spanning tree



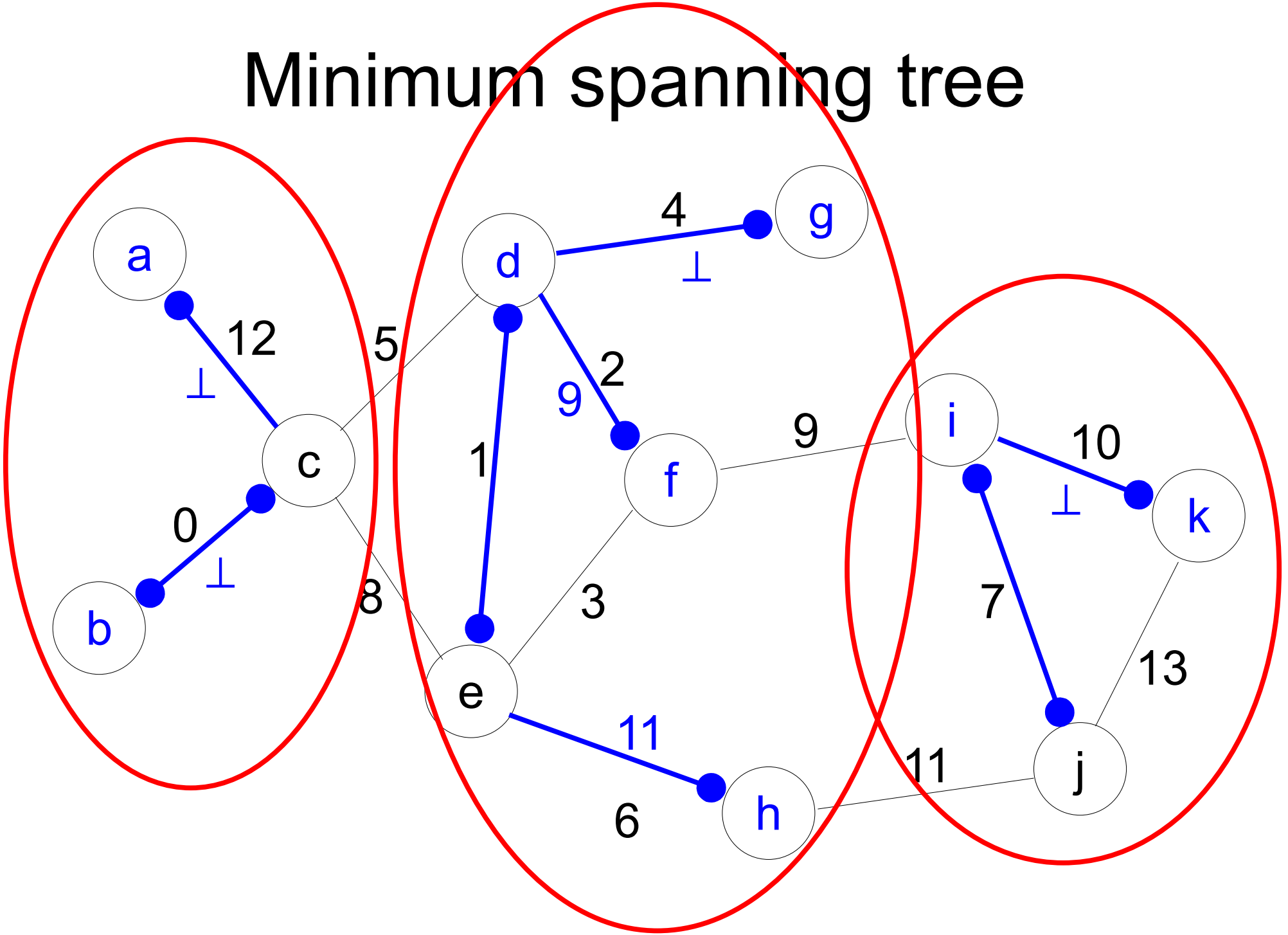
# Minimum spanning tree



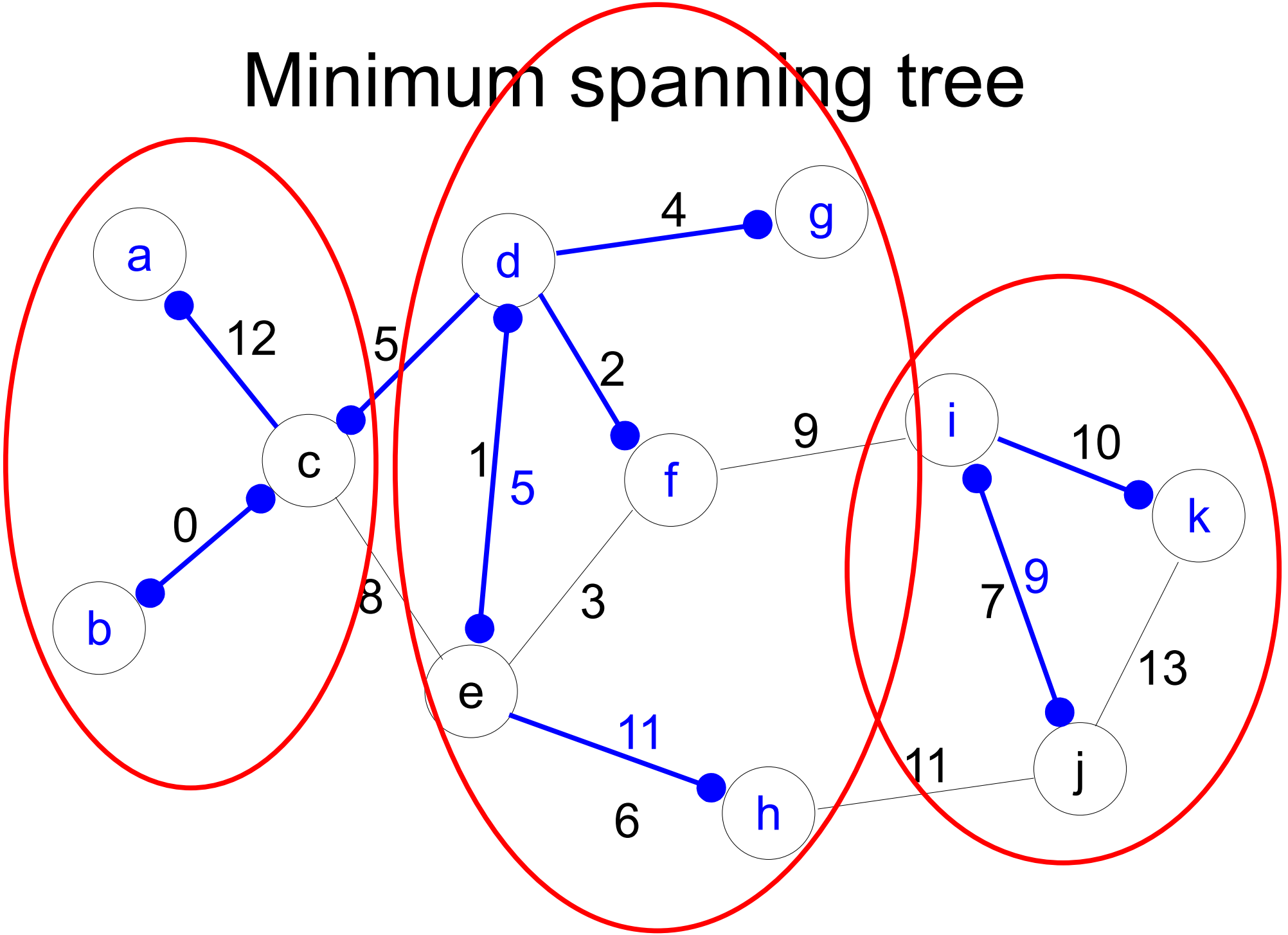
# Minimum spanning tree



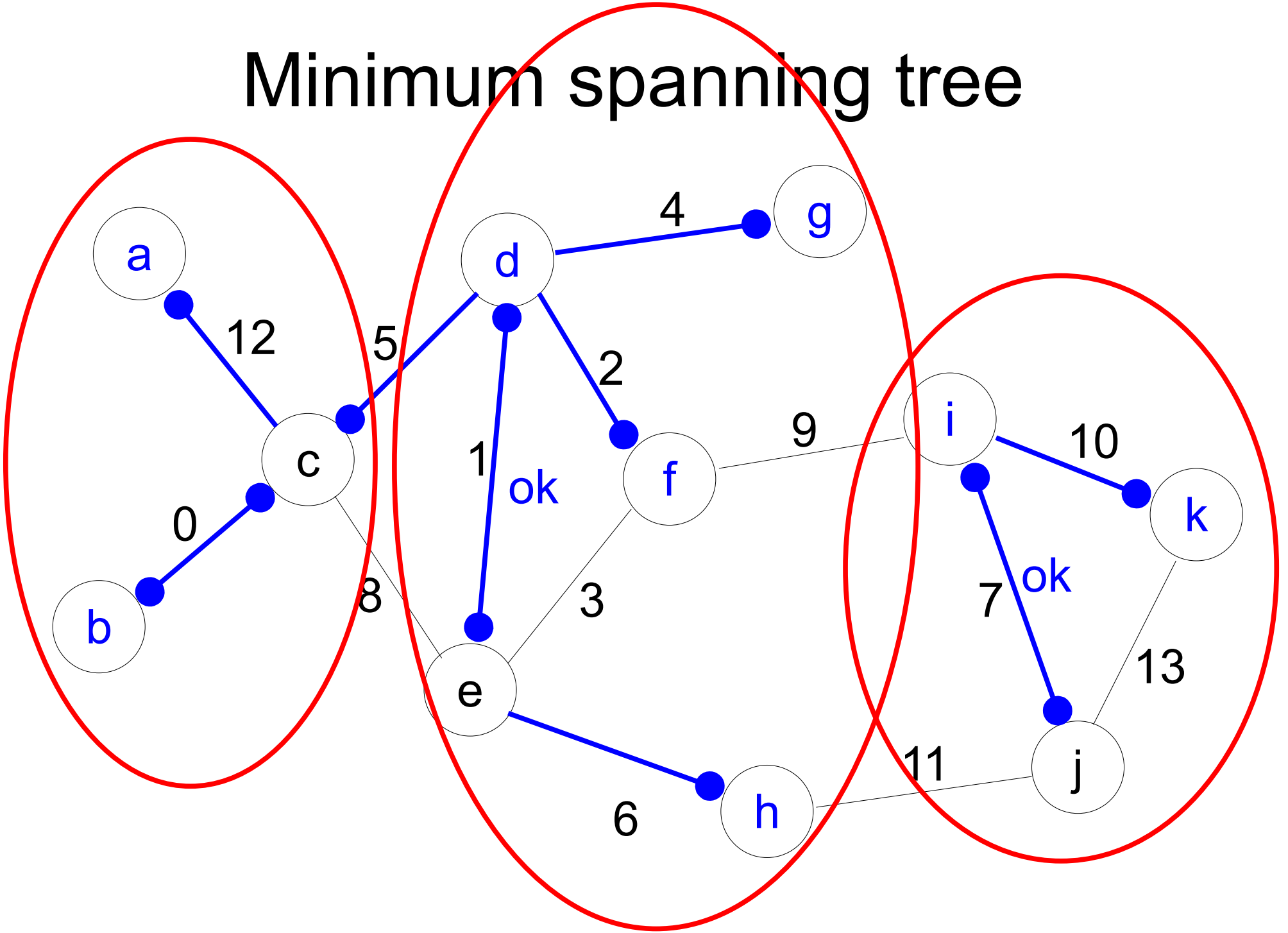
# Minimum spanning tree



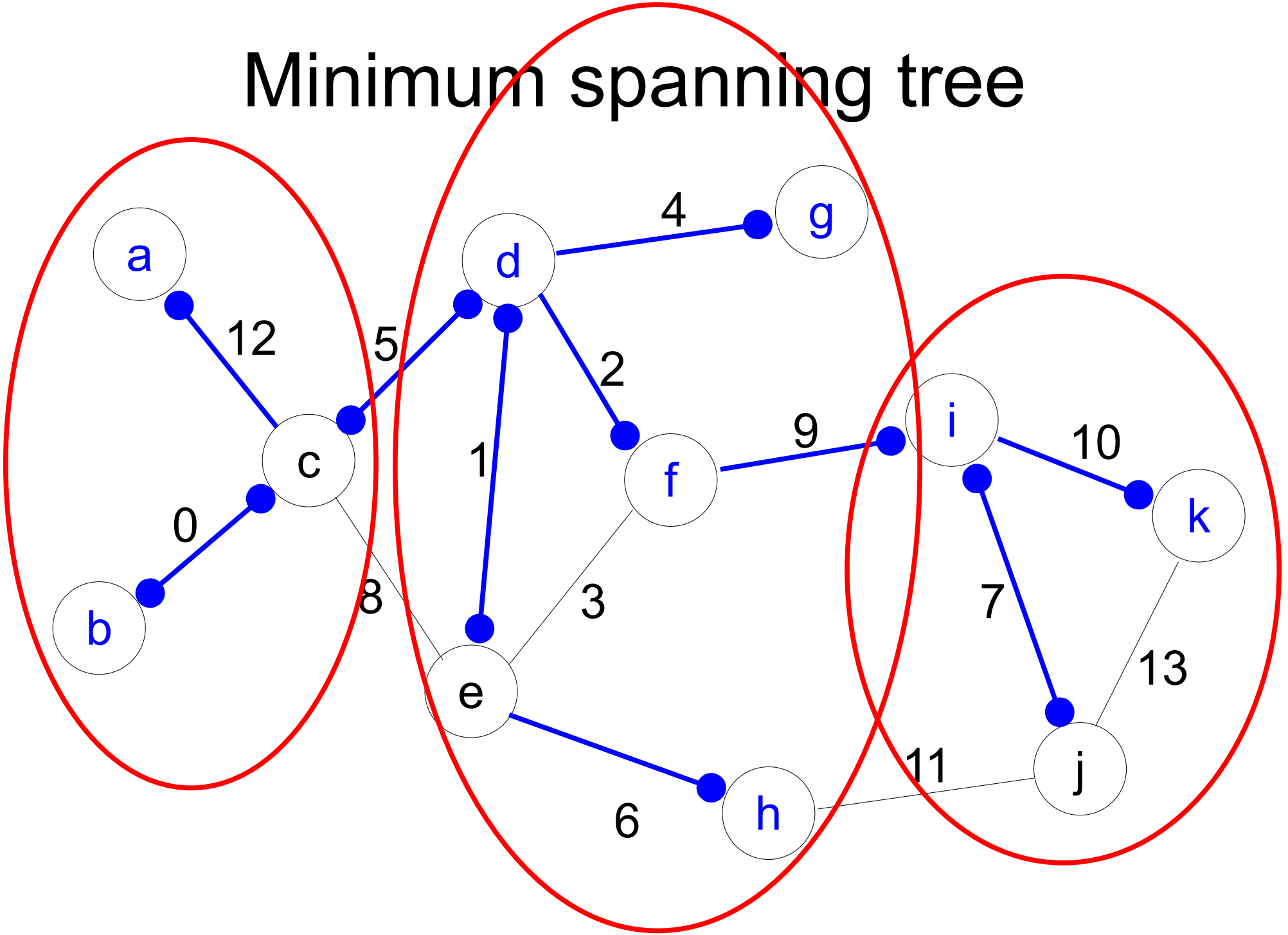
# Minimum spanning tree



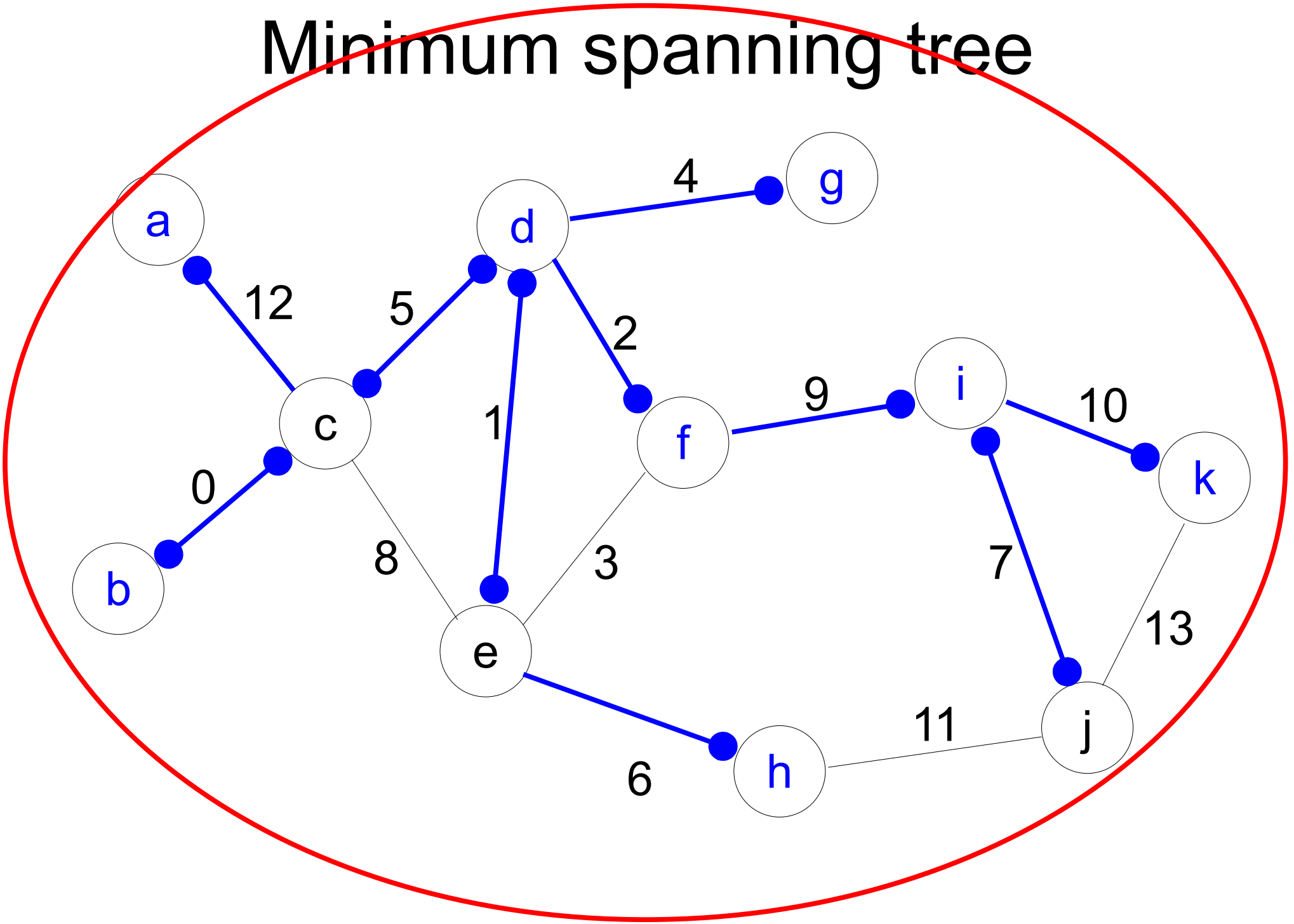
# Minimum spanning tree



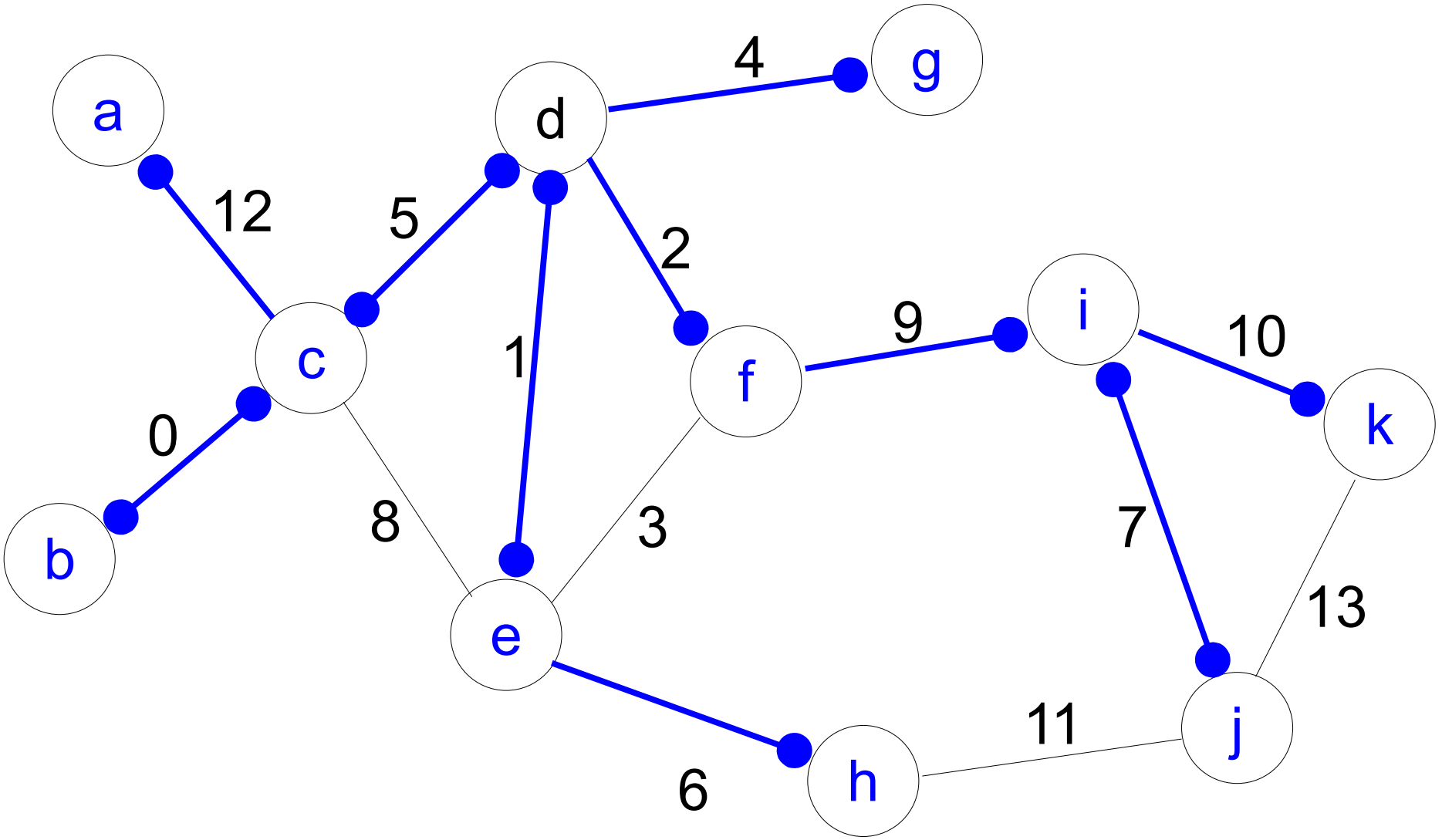
# Minimum spanning tree



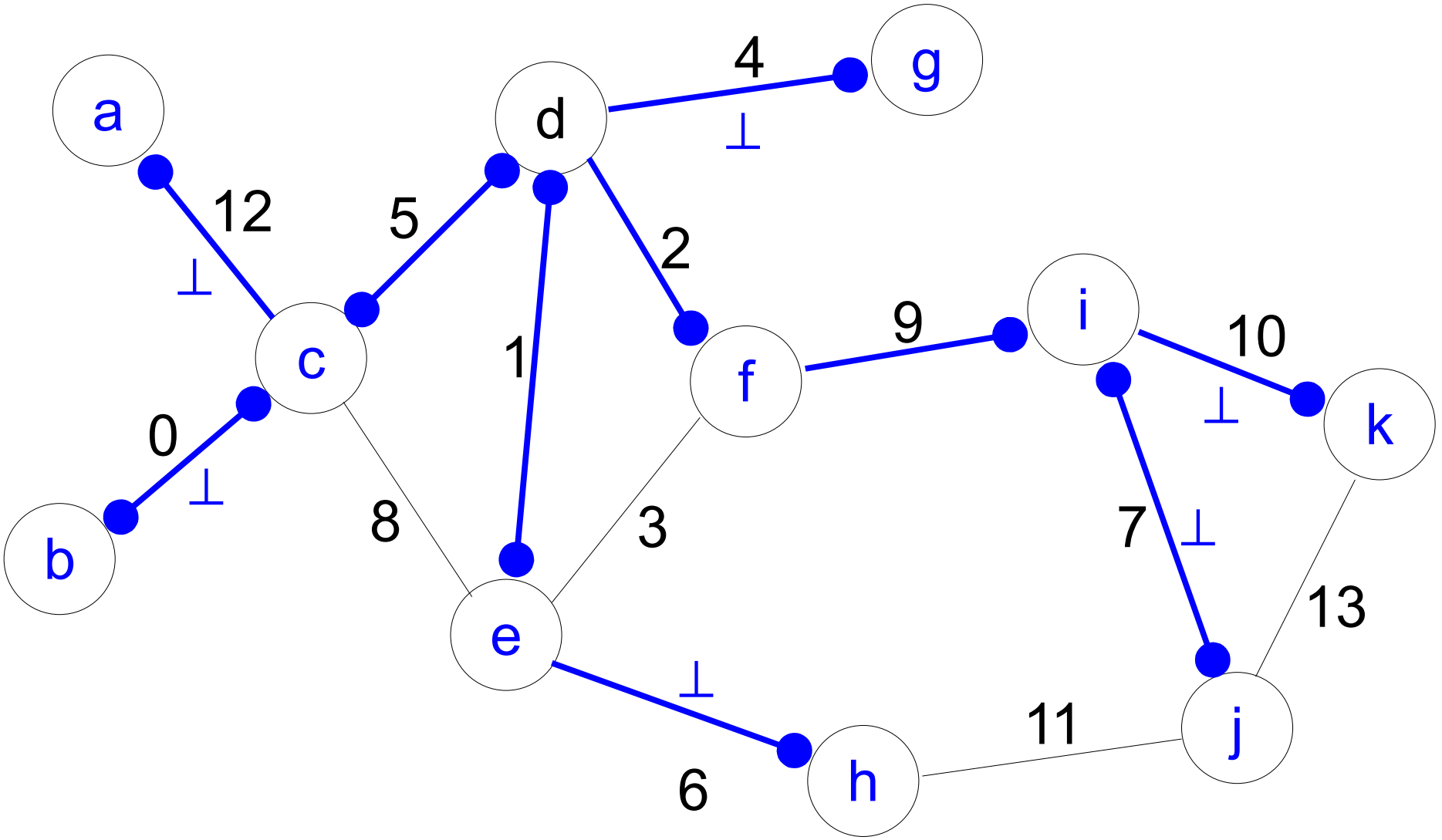
# Minimum spanning tree



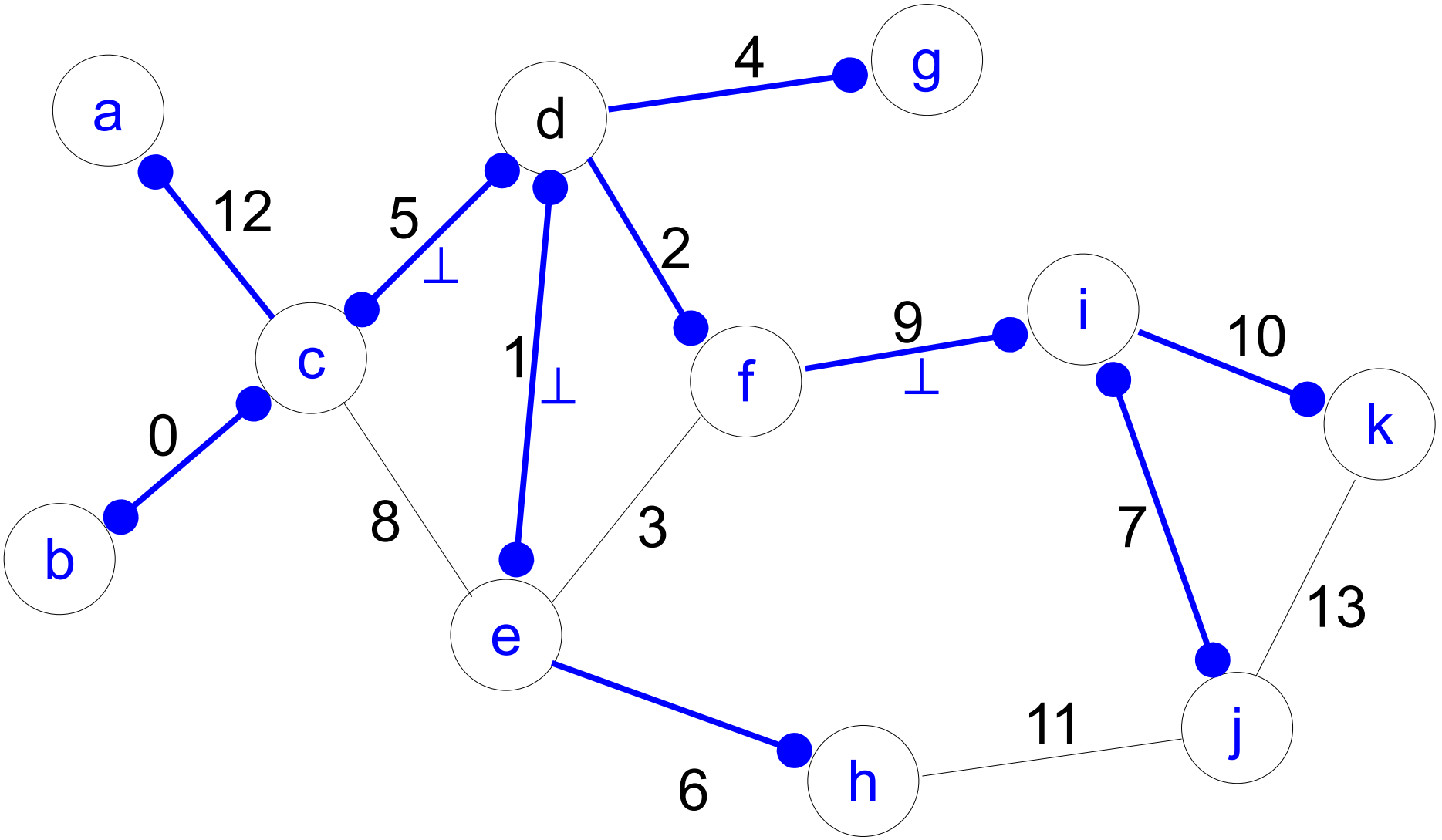
# Minimum spanning tree



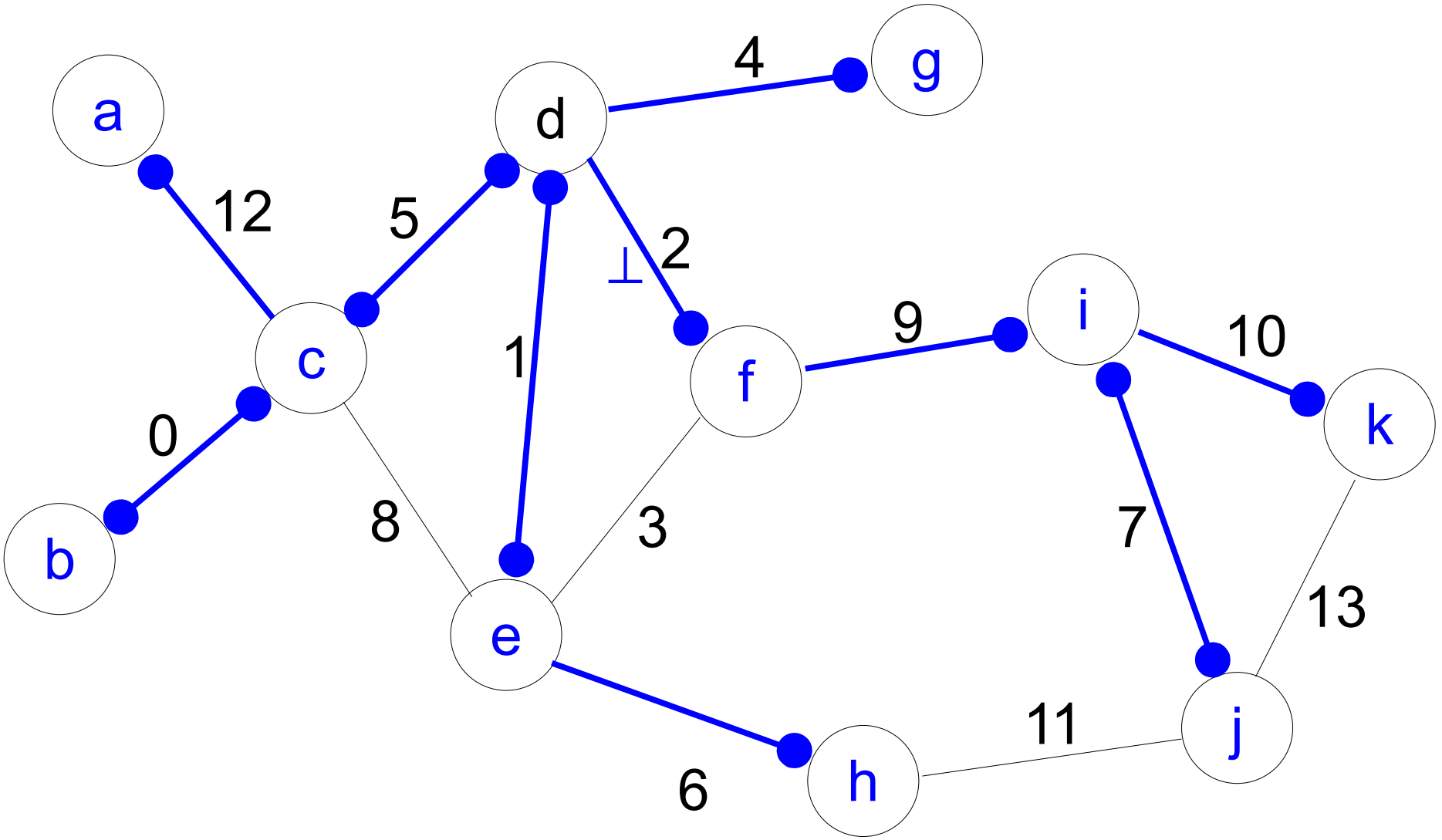
# Minimum spanning tree



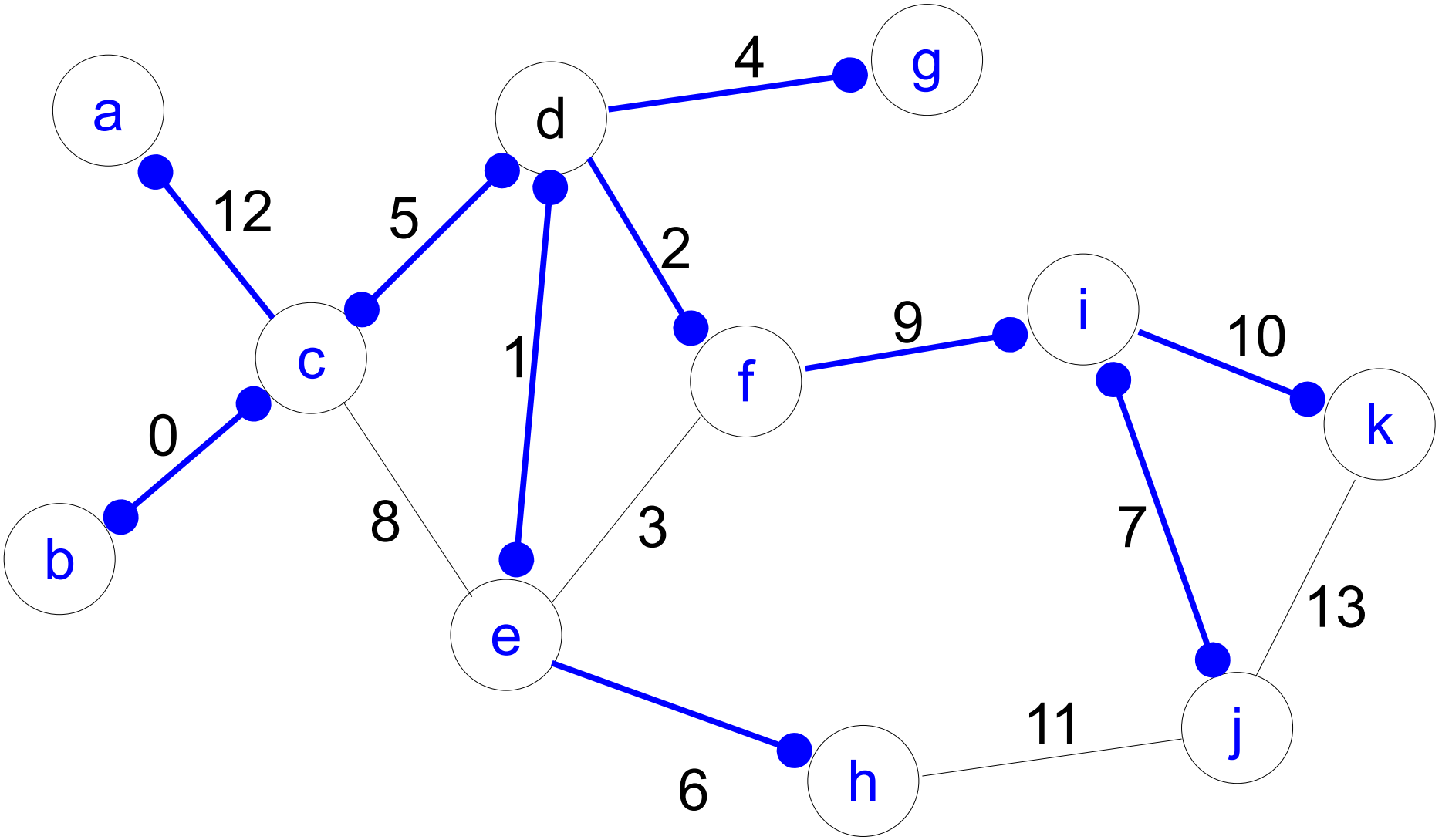
# Minimum spanning tree



# Minimum spanning tree



# Minimum spanning tree



# Minimum spanning tree

- GHS algorithm simplifed for synchronous setting
- Proof?
- Complexity?
  - time: O(n log n)
  - msg: O( (n + |E|) log n)
    - actually O(n log n + |E|)

# Where did we use synchrony?

- Leader election
- Breadth-first search
- Shortest paths
- Minimum spanning tree

We will see these algorithms again
in the asynchronous setting.