Girvan-Newman Algorithm

Slides from Dr. Frank McCown

Intro to Web Science

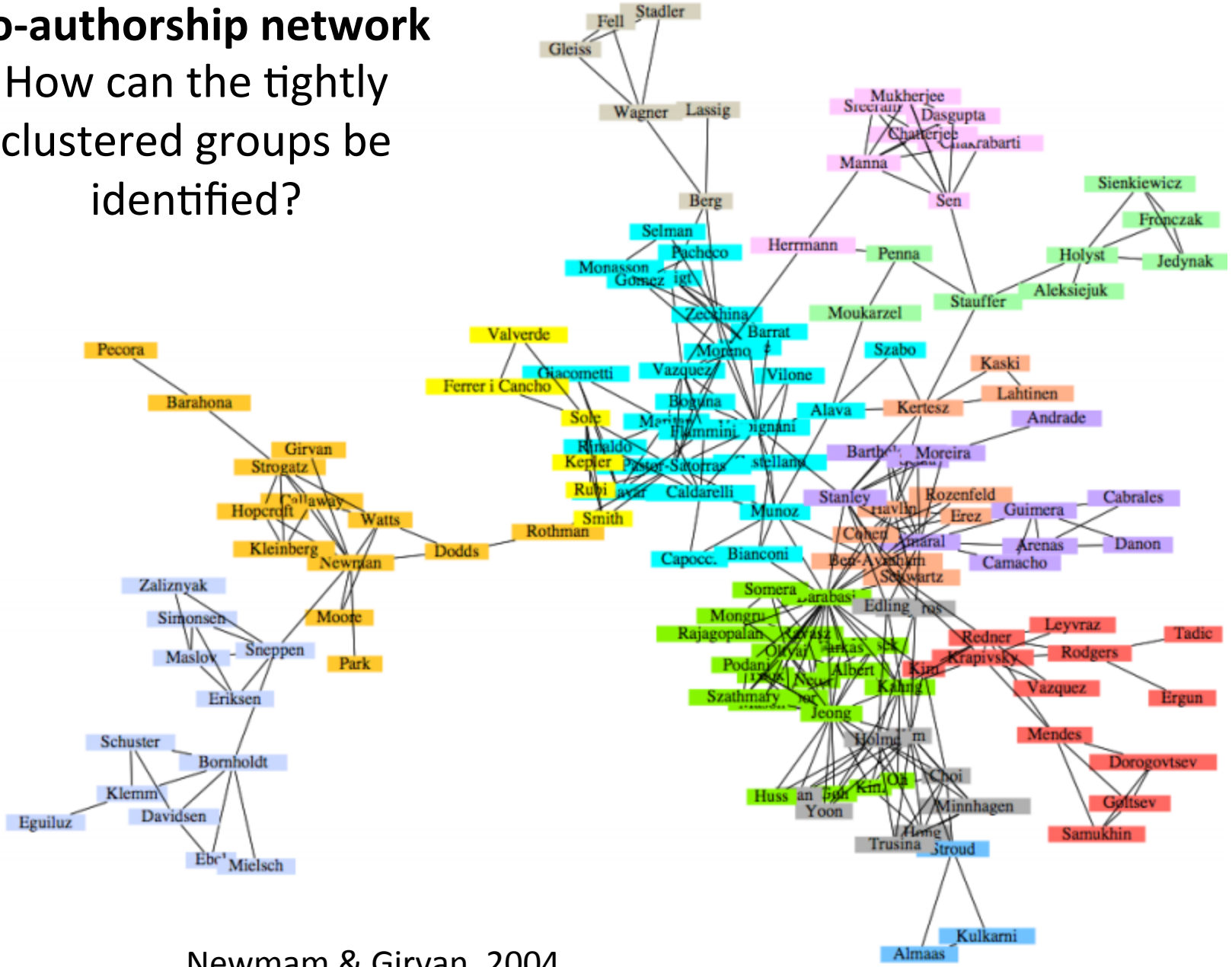
Harding University

Slides use figures from Ch 3.6 of *Networks, Crowds and Markets* by Easley & Kleinberg (2010)

http://www.cs.cornell.edu/home/kleinber/networks-book/

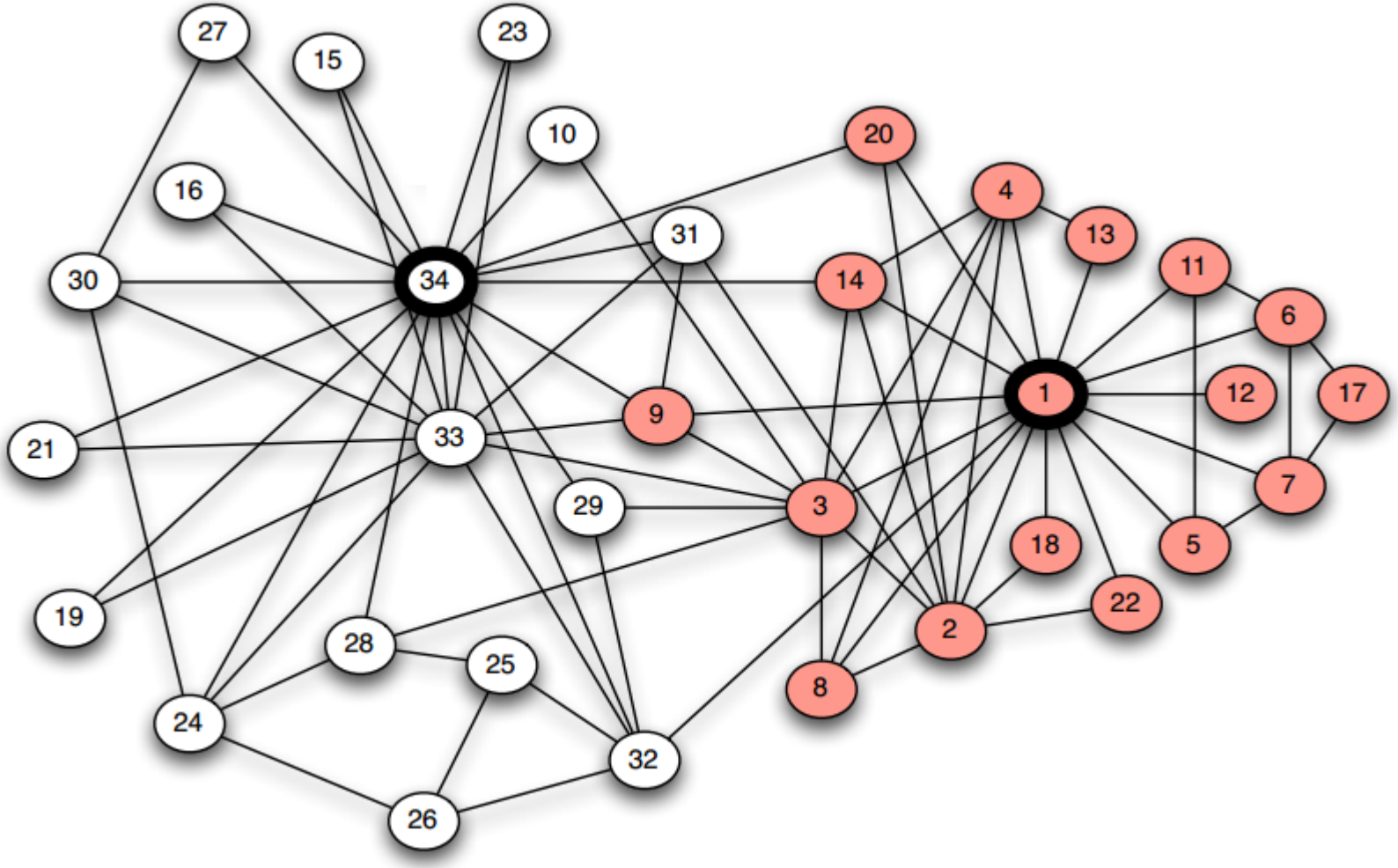
Co-authorship network

How can the tightly clustered groups be identified?



Newmam & Girvan, 2004

Karate Club splits after a dispute. Can new clubs be identified based on network structure?

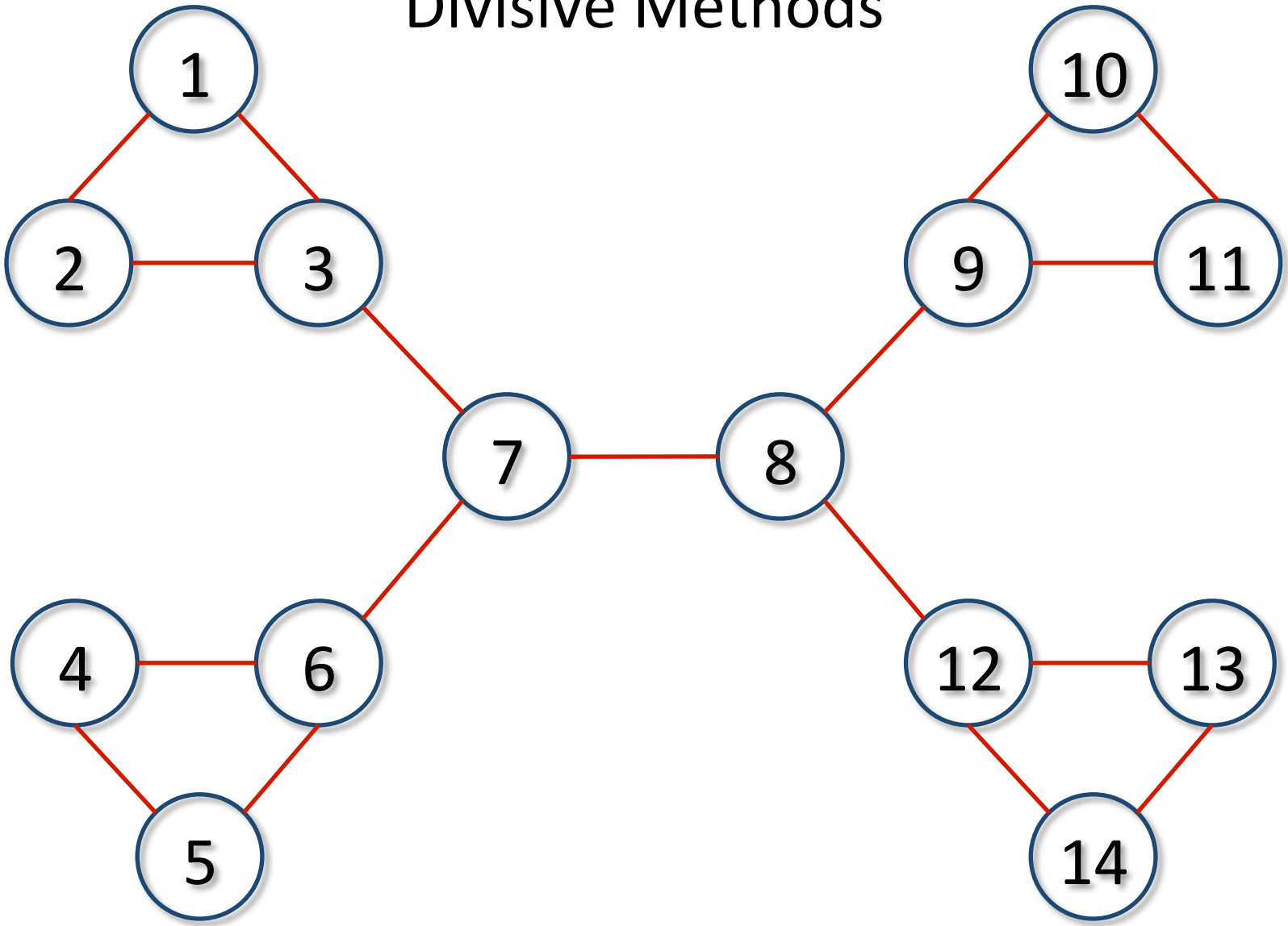


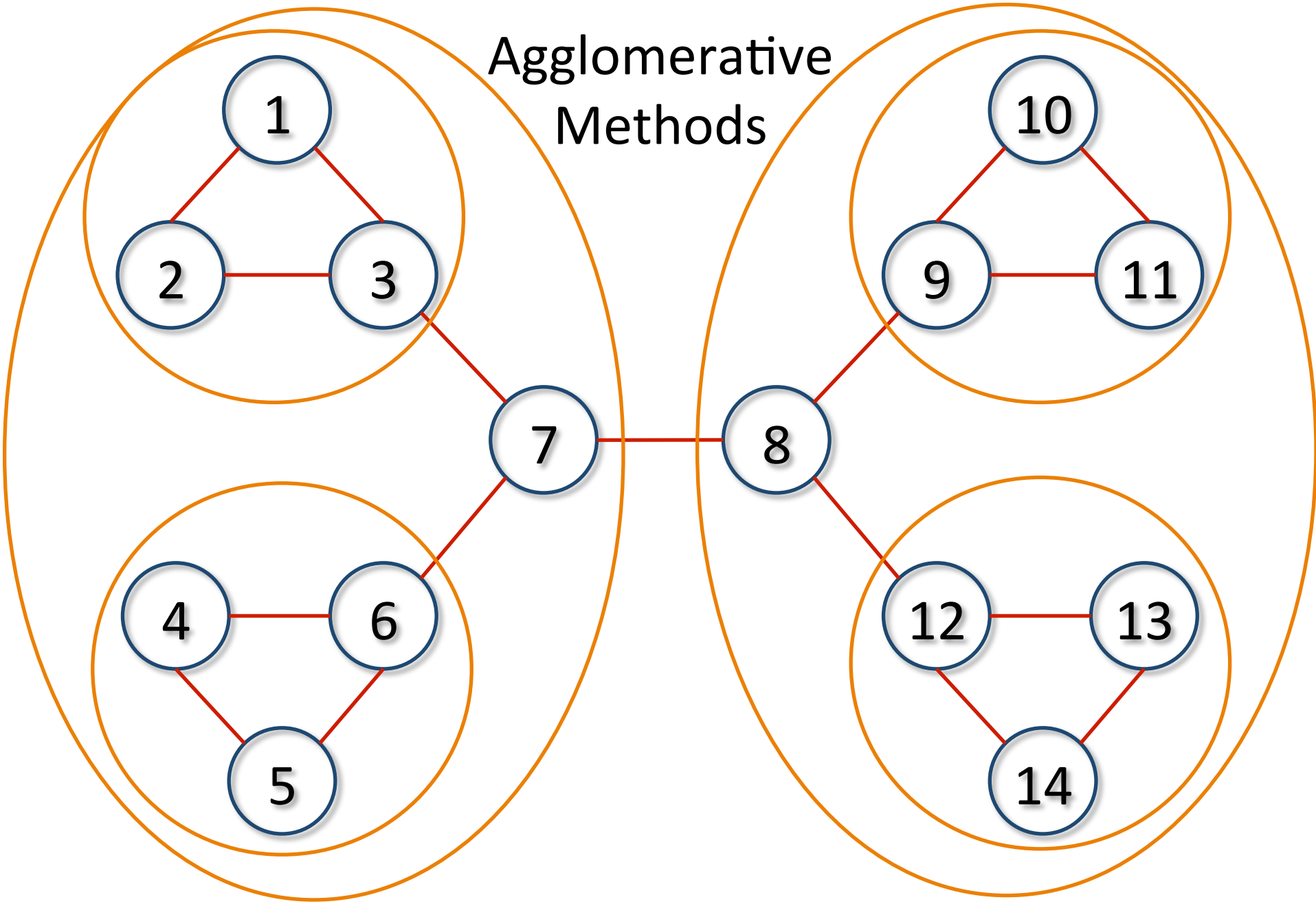
Zachary, 1977

Community Discovery

- Methods to break a network into sets of connected components called *regions*
- Many general approaches
  - **Divisive methods**: Repeatedly identify and remove edges connecting densely connected regions
  - **Agglomerative methods**: Repeatedly identify and merge nodes that likely belong in the same region

Divisive Methods

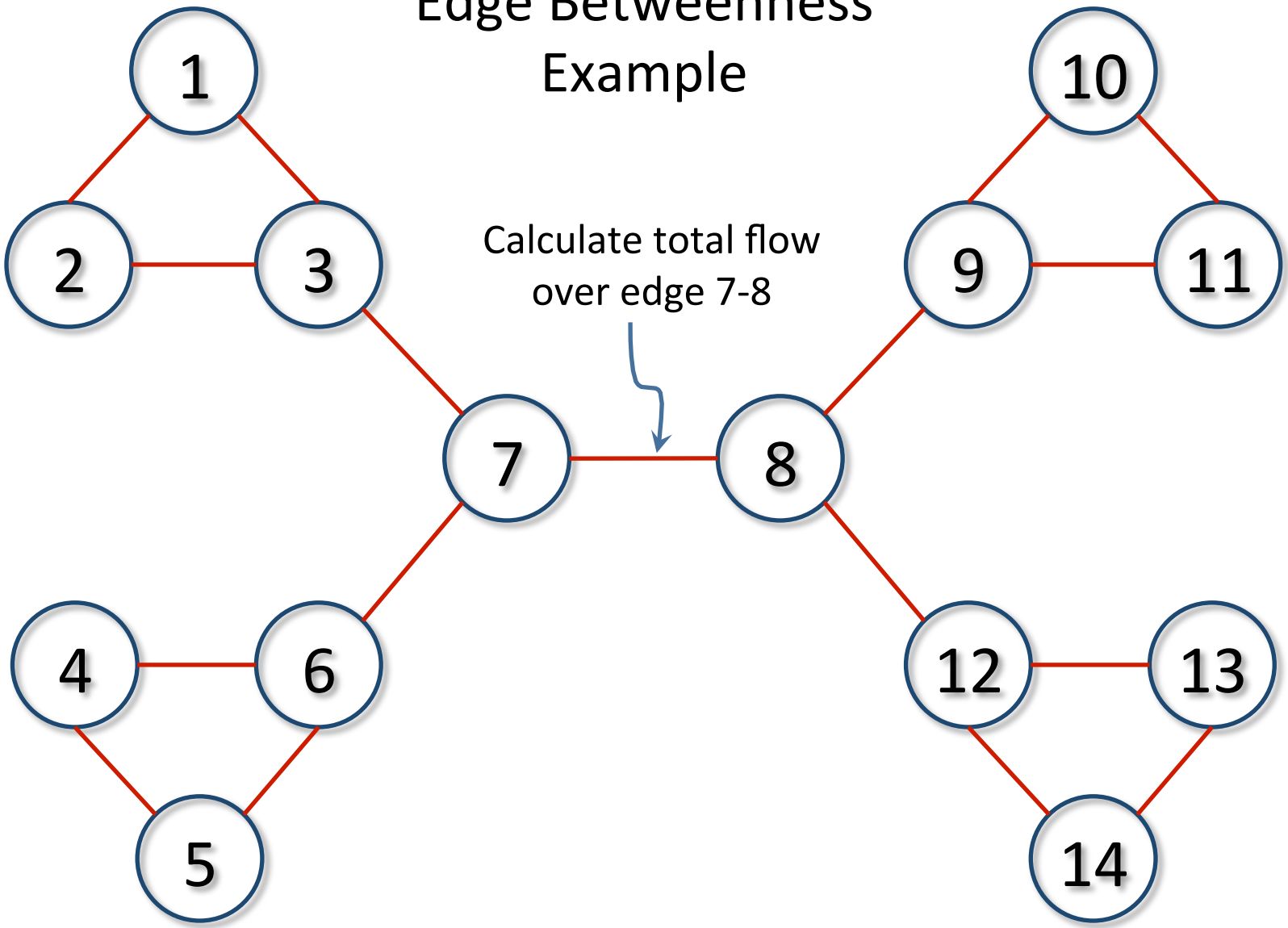


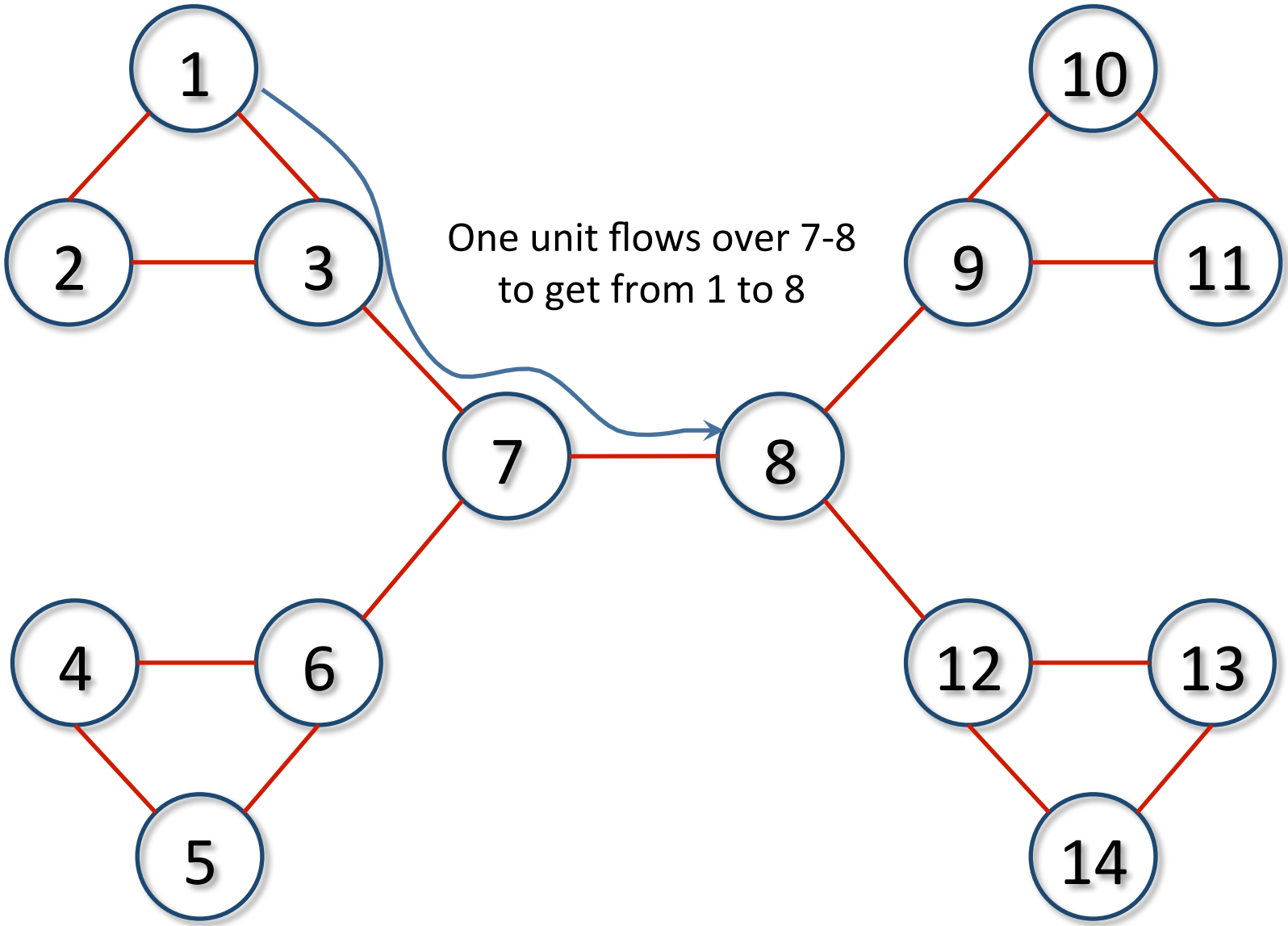

Agglomerative
Methods
1
2
3
4
5
6
7
8
9
10
11
12
13
14

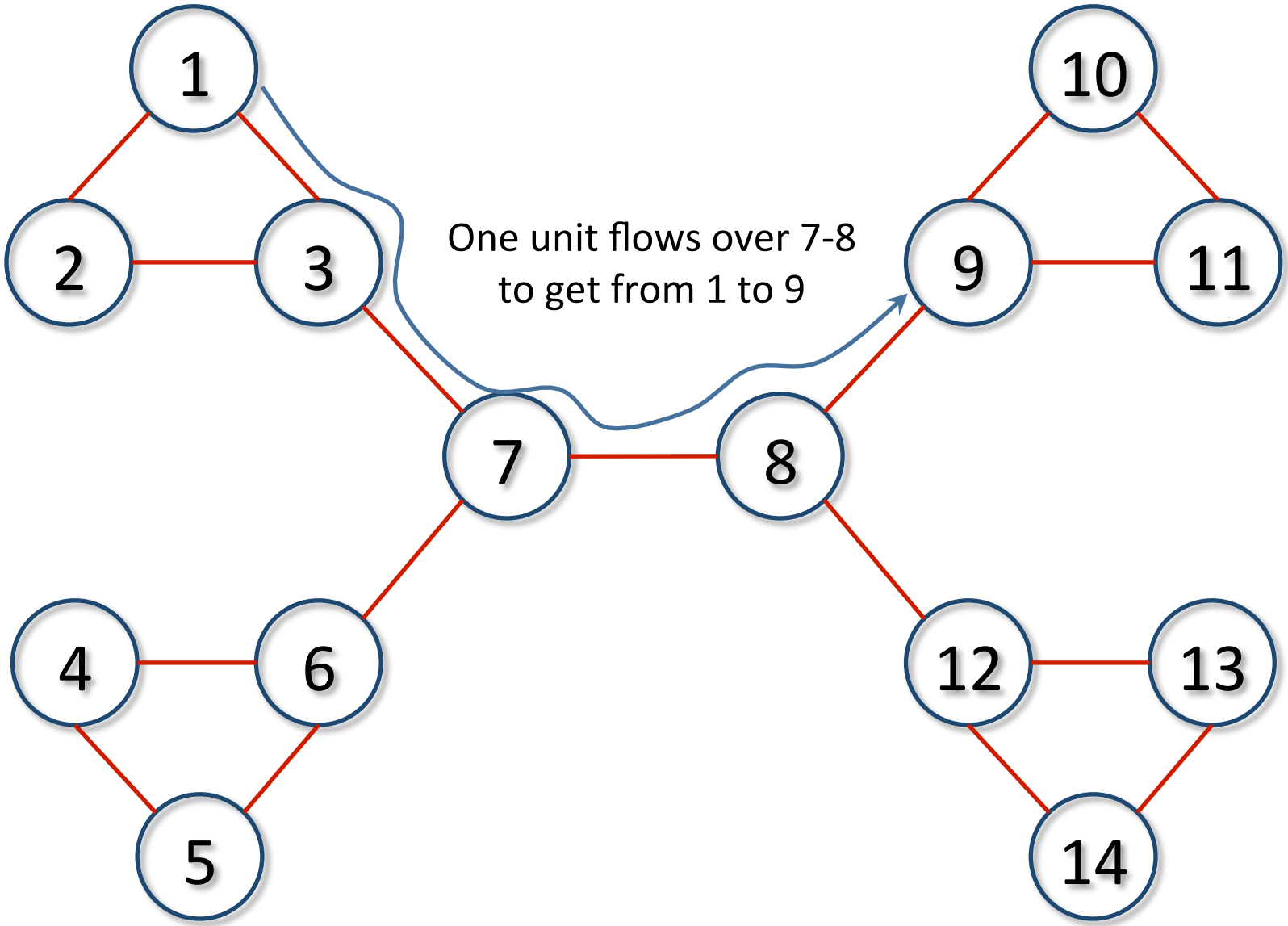
Girvan-Newman Algorithm

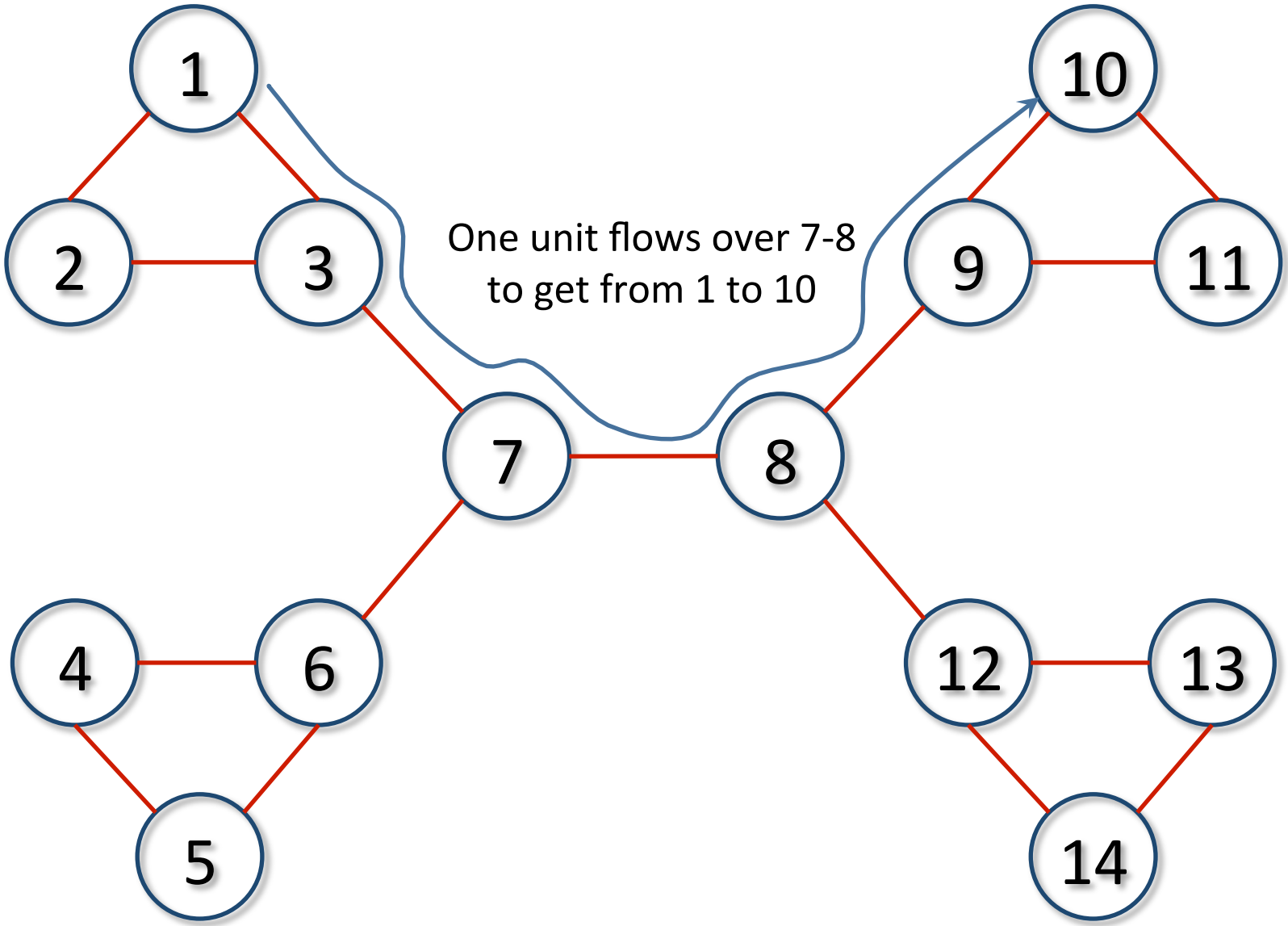
- Divisive method Proposed by Girvan and Newman in 2002
- Uses *edge betweenness* to identify edges to remove
- **Edge betweenness of edge *e*: (simple version)** The number of shortest paths (between all pair of vertices) that passes through edge *e*.

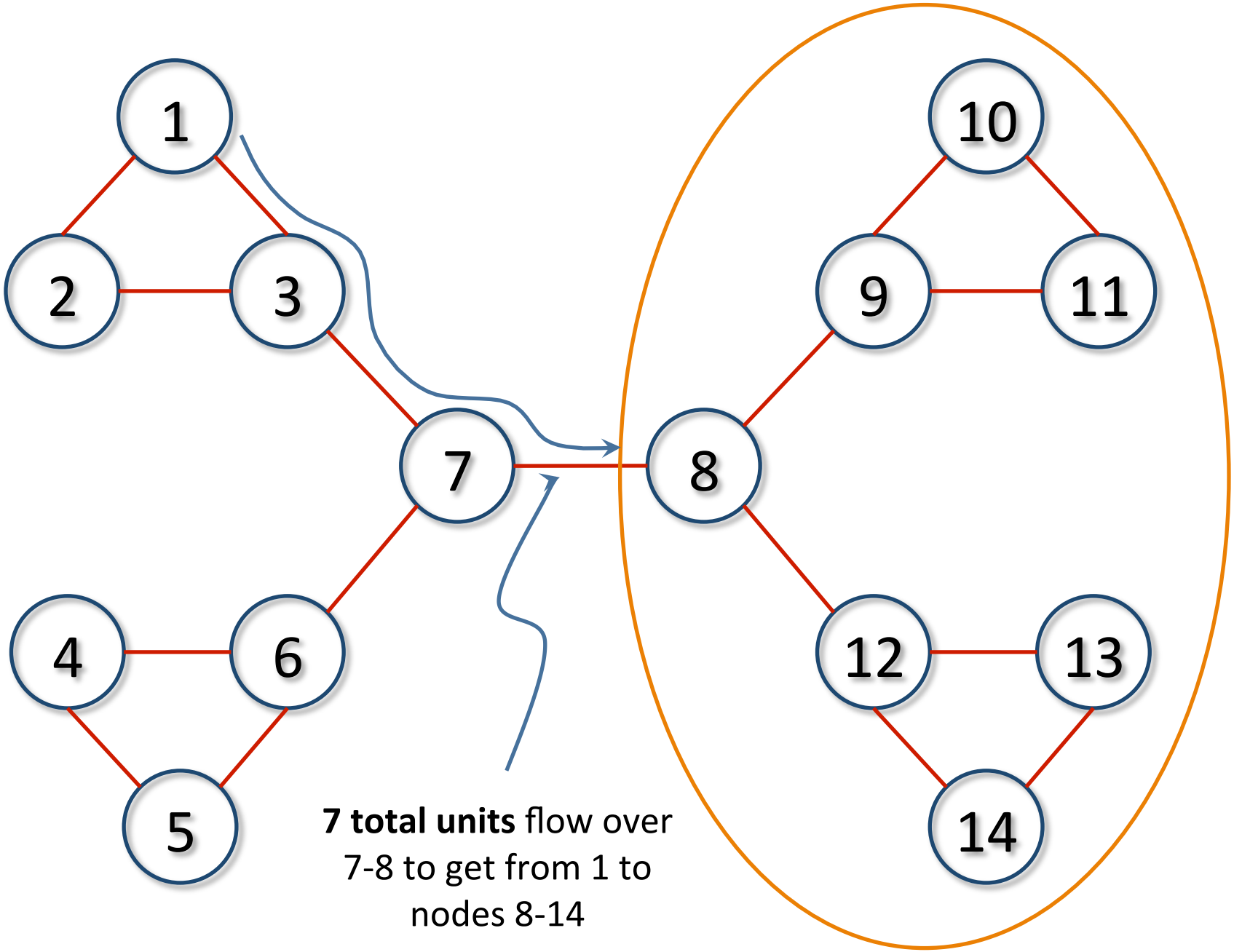
Edge Betweenness Example

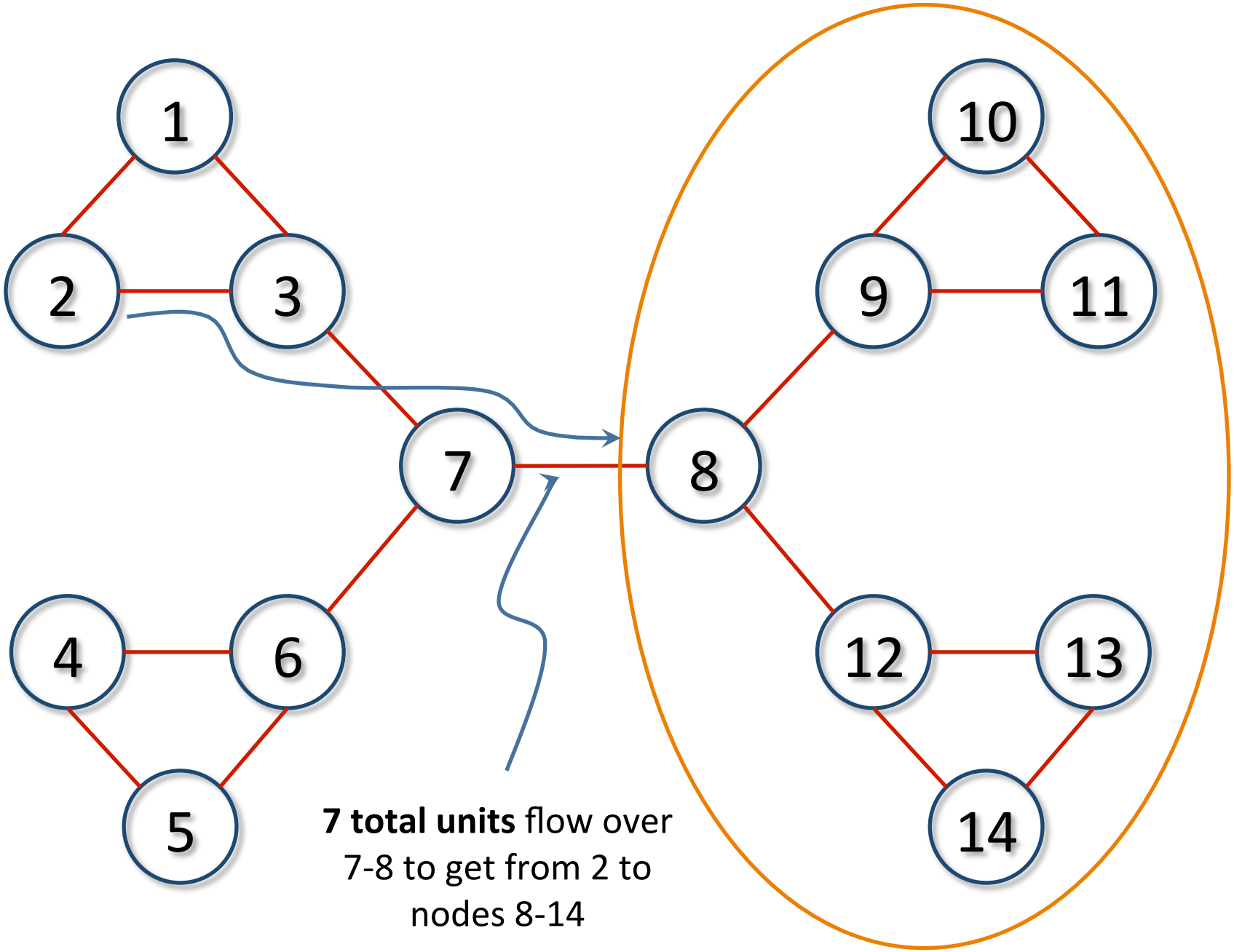


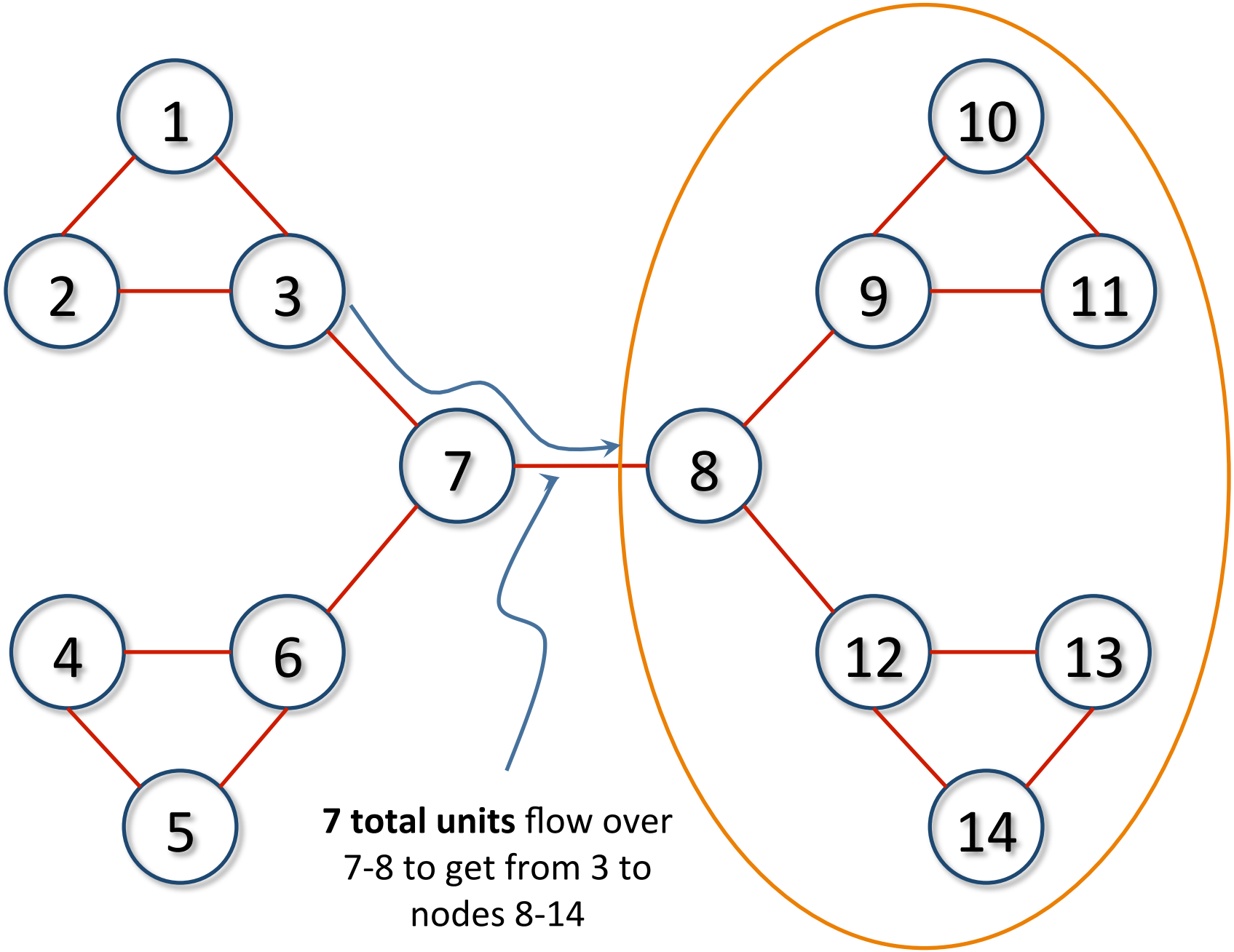

1
2
3
4
5
6
7
8
9
10
11
12
13
14
One unit flows over 7-8
to get from 1 to 8

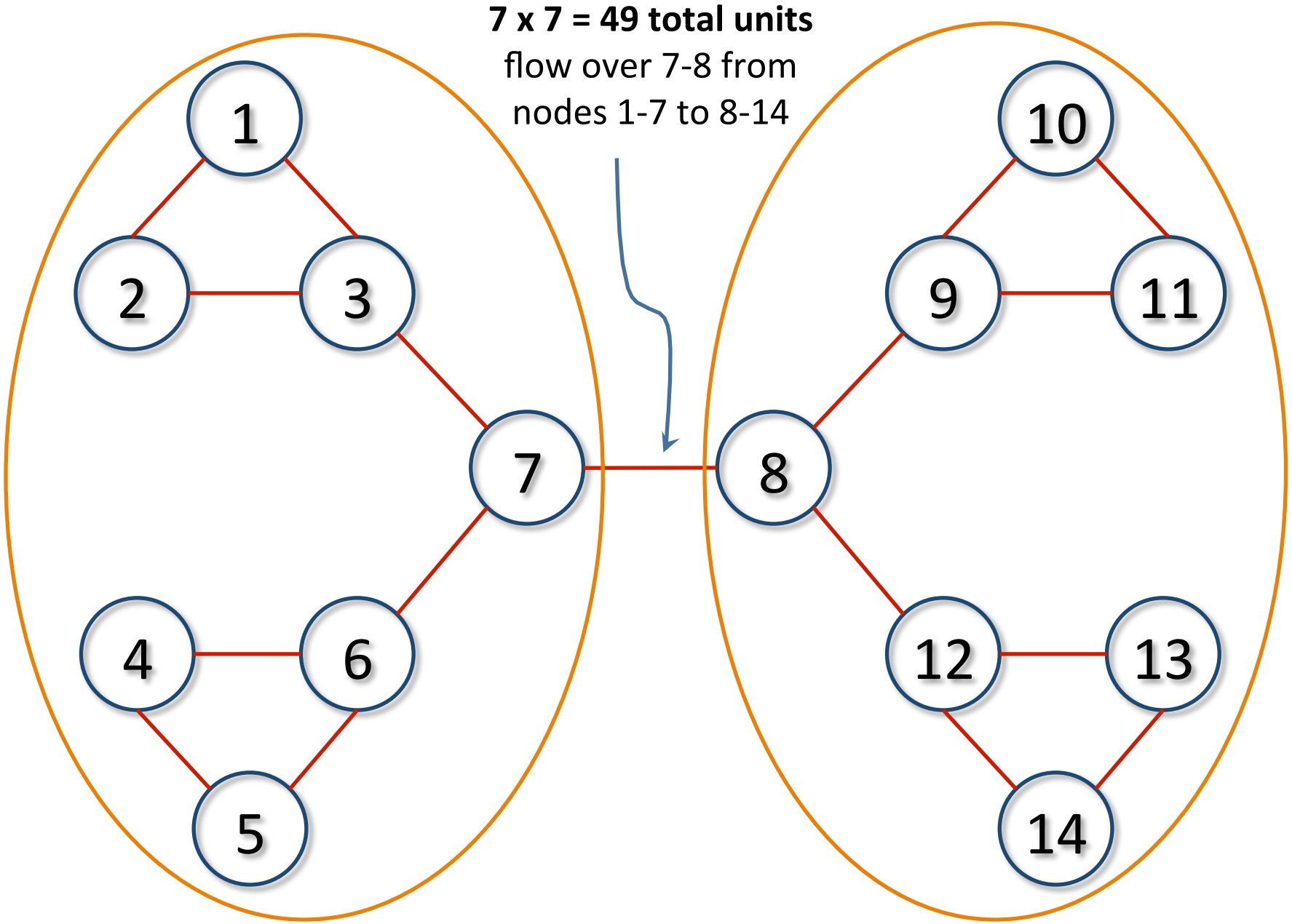

1
2
3
4
5
6
7
8
9
10
11
12
13
14
One unit flows over 7-8
to get from 1 to 9

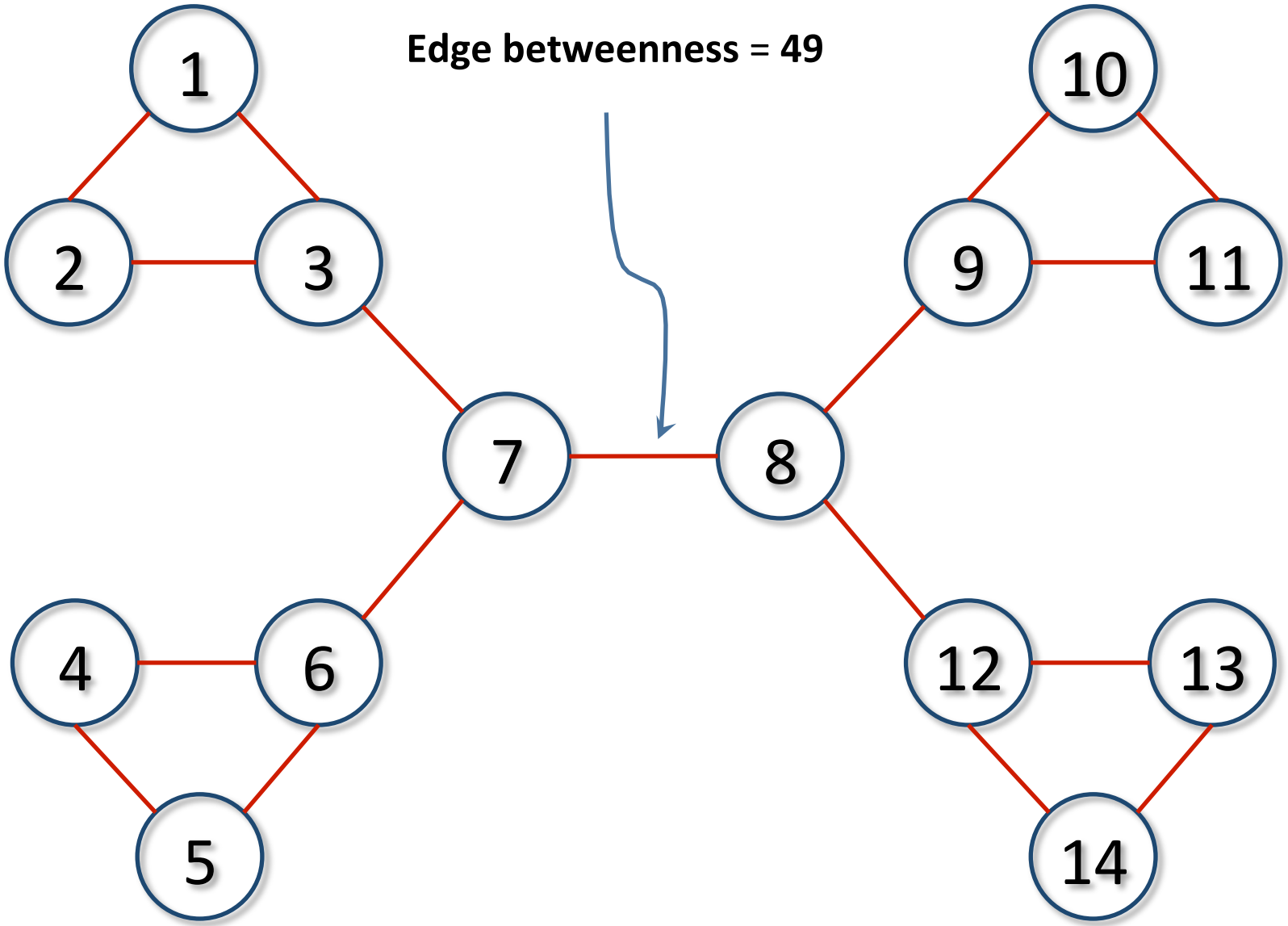

1
2
3
4
5
6
7
8
9
10
11
12
13
14
One unit flows over 7-8
to get from 1 to 10

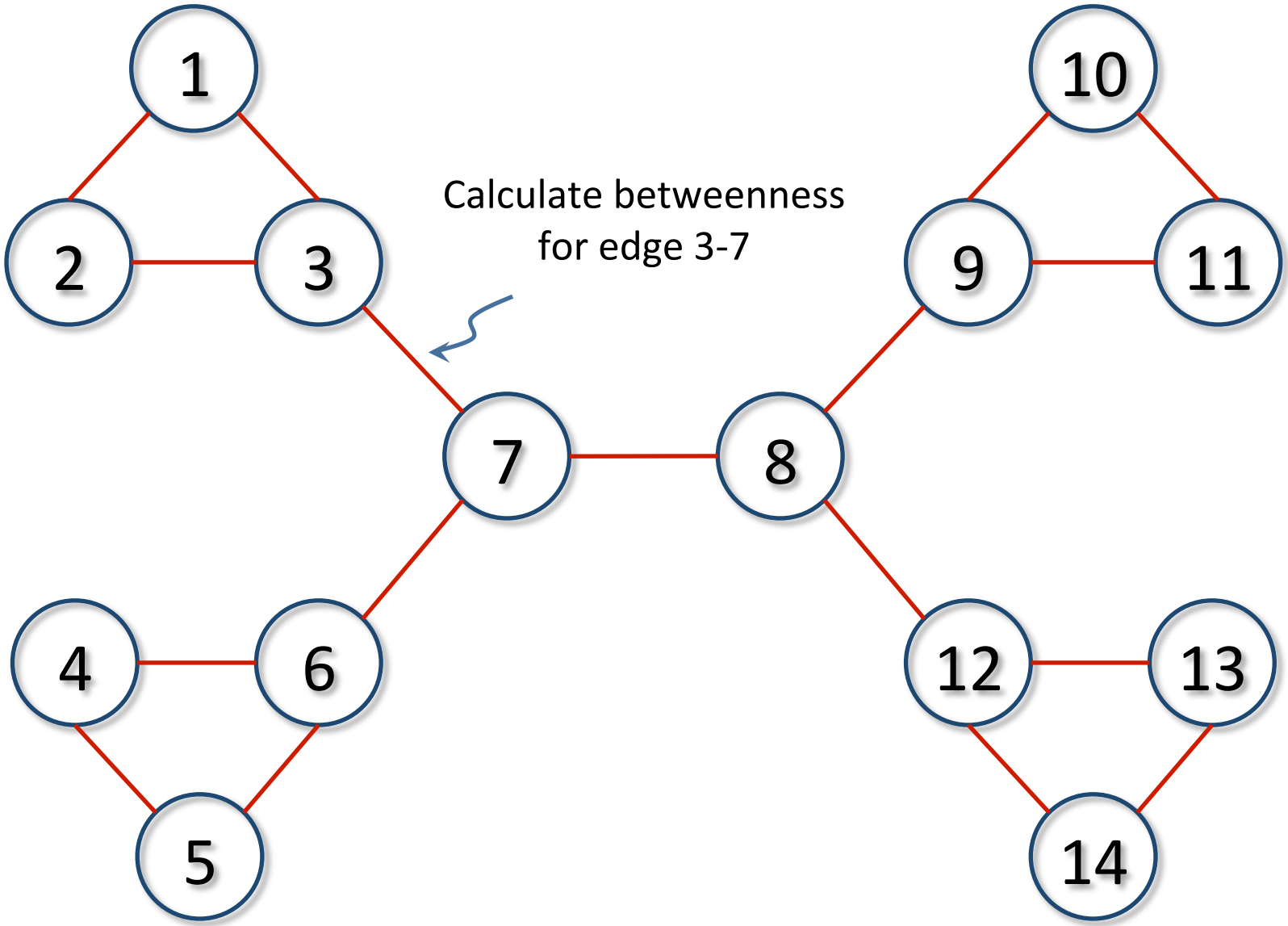

1
2
3
4
5
6
7
8
9
10
11
12
13
14
7 total units flow over 7-8 to get from 1 to nodes 8-14

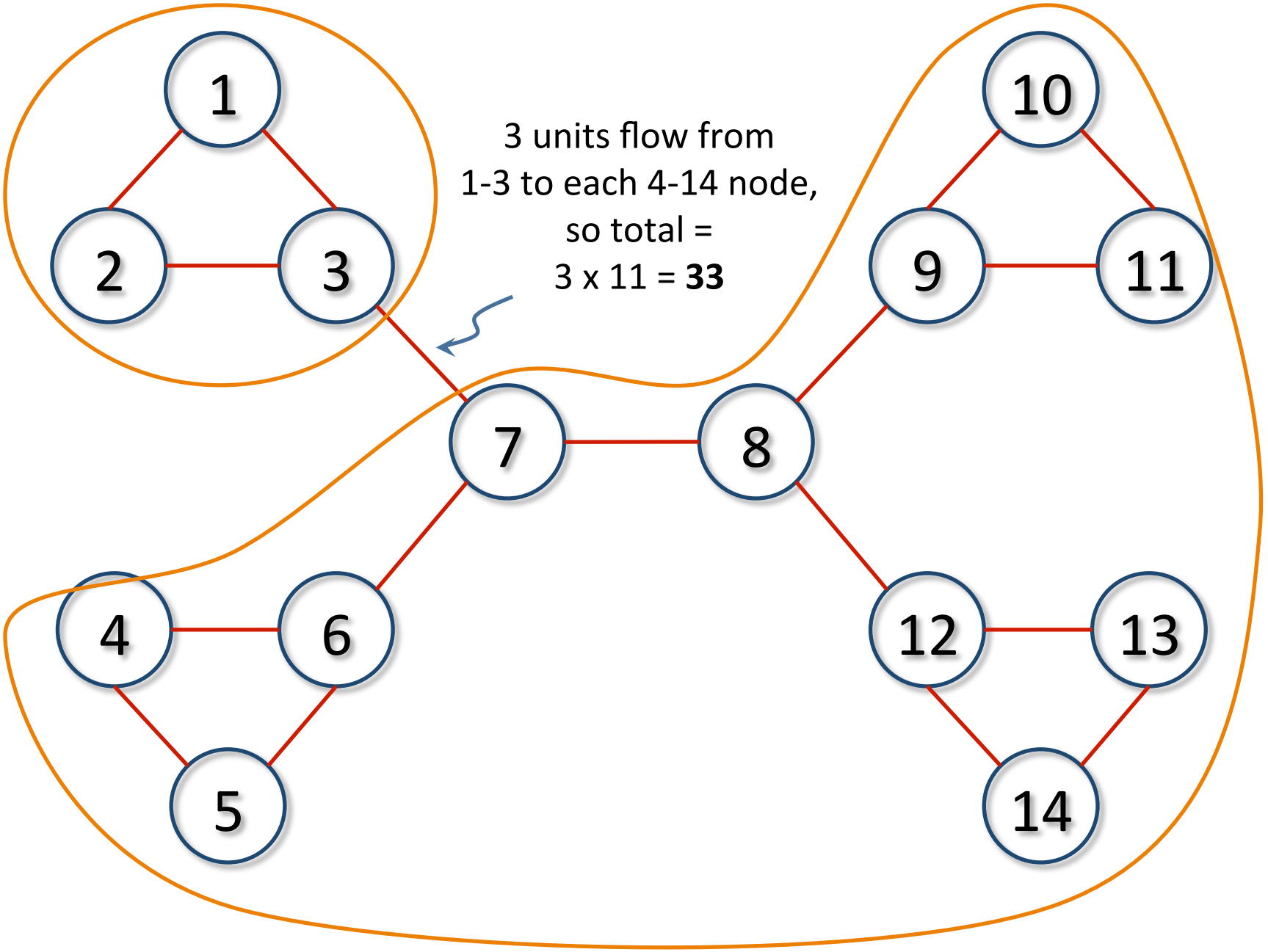

1
2
3
4
5
6
7
8
9
10
11
12
13
14
7 total units flow over 7-8 to get from 2 to nodes 8-14

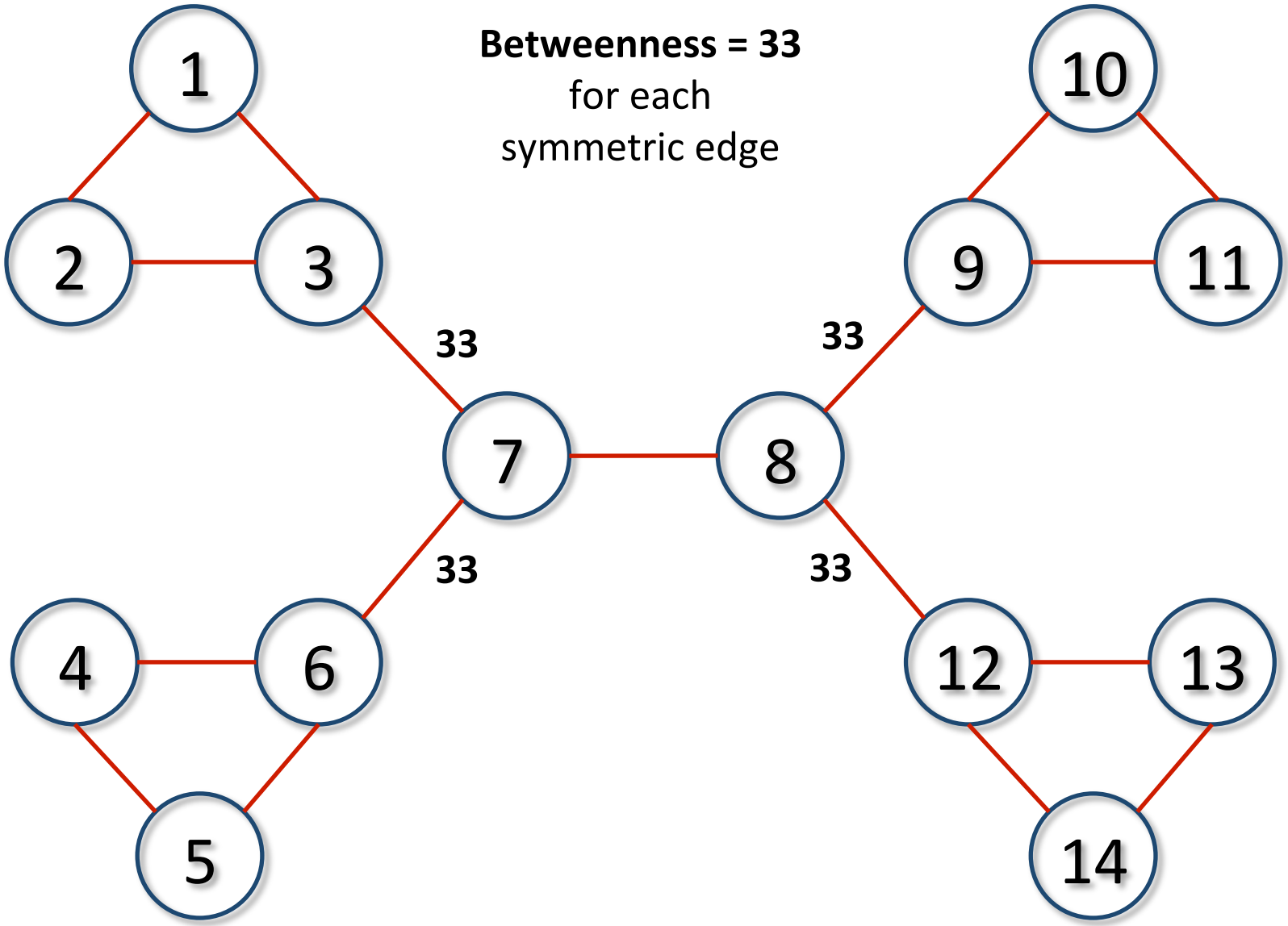

1
2
3
4
5
6
7
8
9
10
11
12
13
14
7 total units flow over 7-8 to get from 3 to nodes 8-14

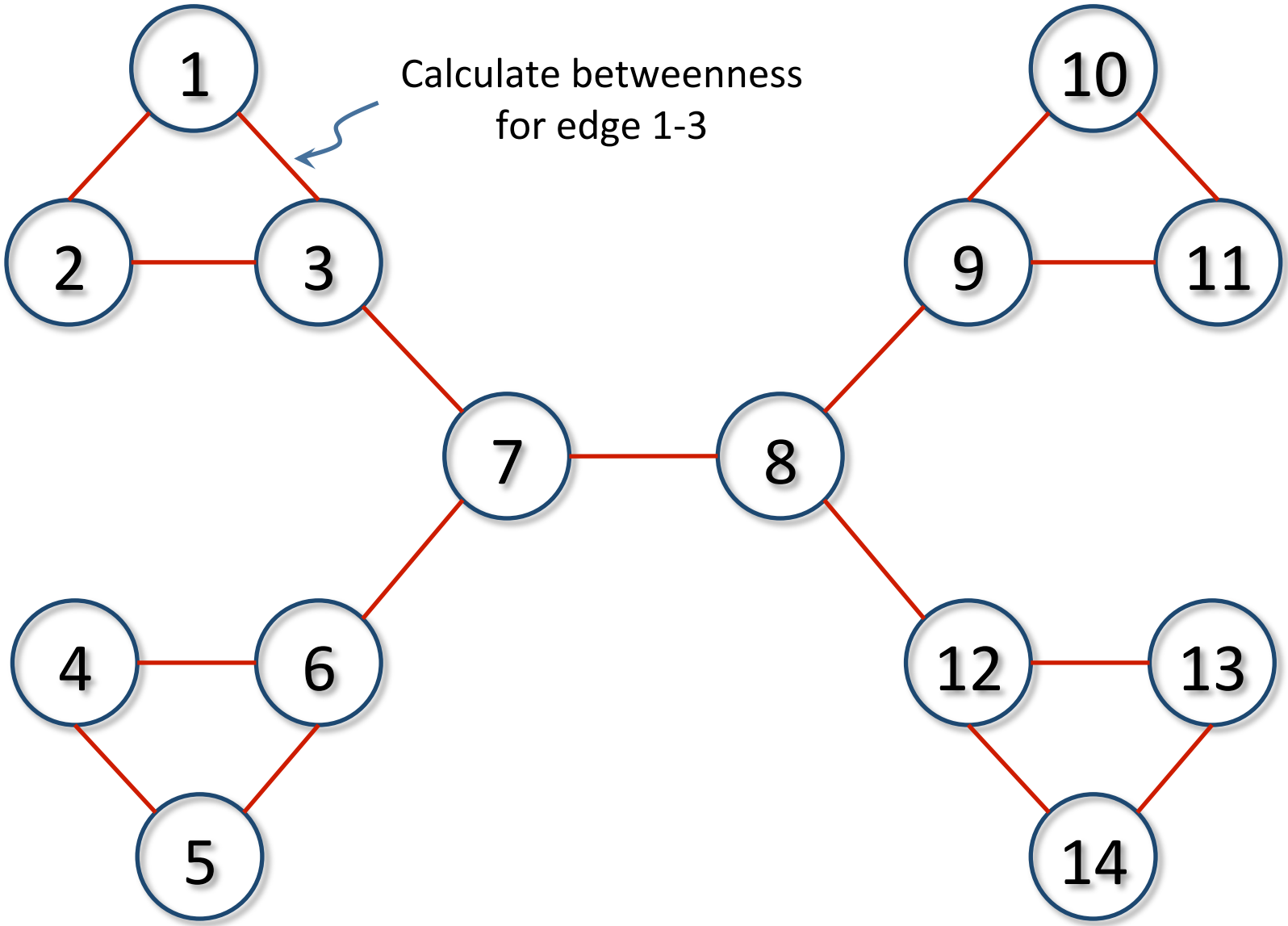

7 x 7 = 49 total units
flow over 7-8 from
nodes 1-7 to 8-14
1
2
3
4
5
6
7
8
9
10
11
12
13
14

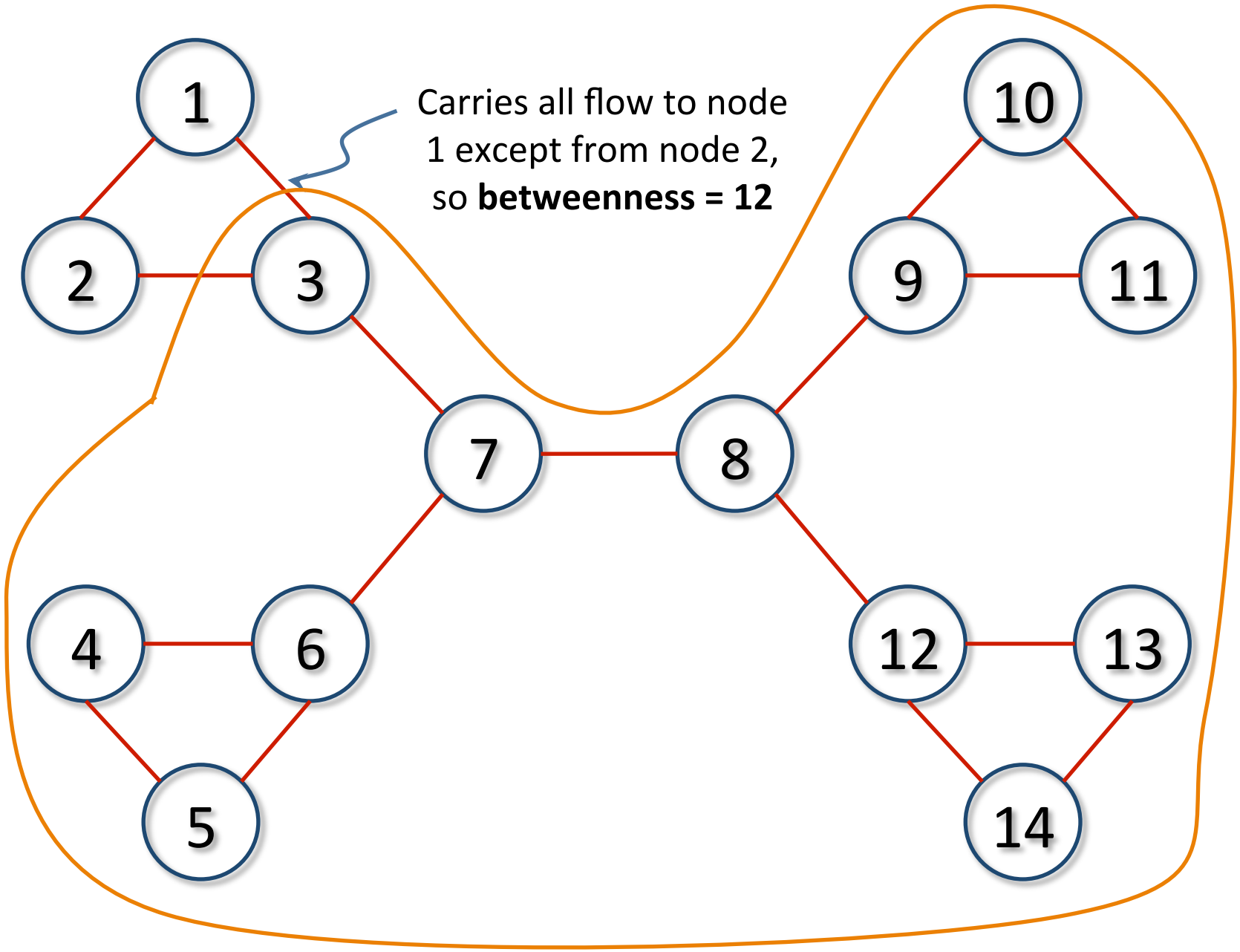

Edge betweenness = 49
1
2
3
4
5
6
7
8
9
10
11
12
13
14

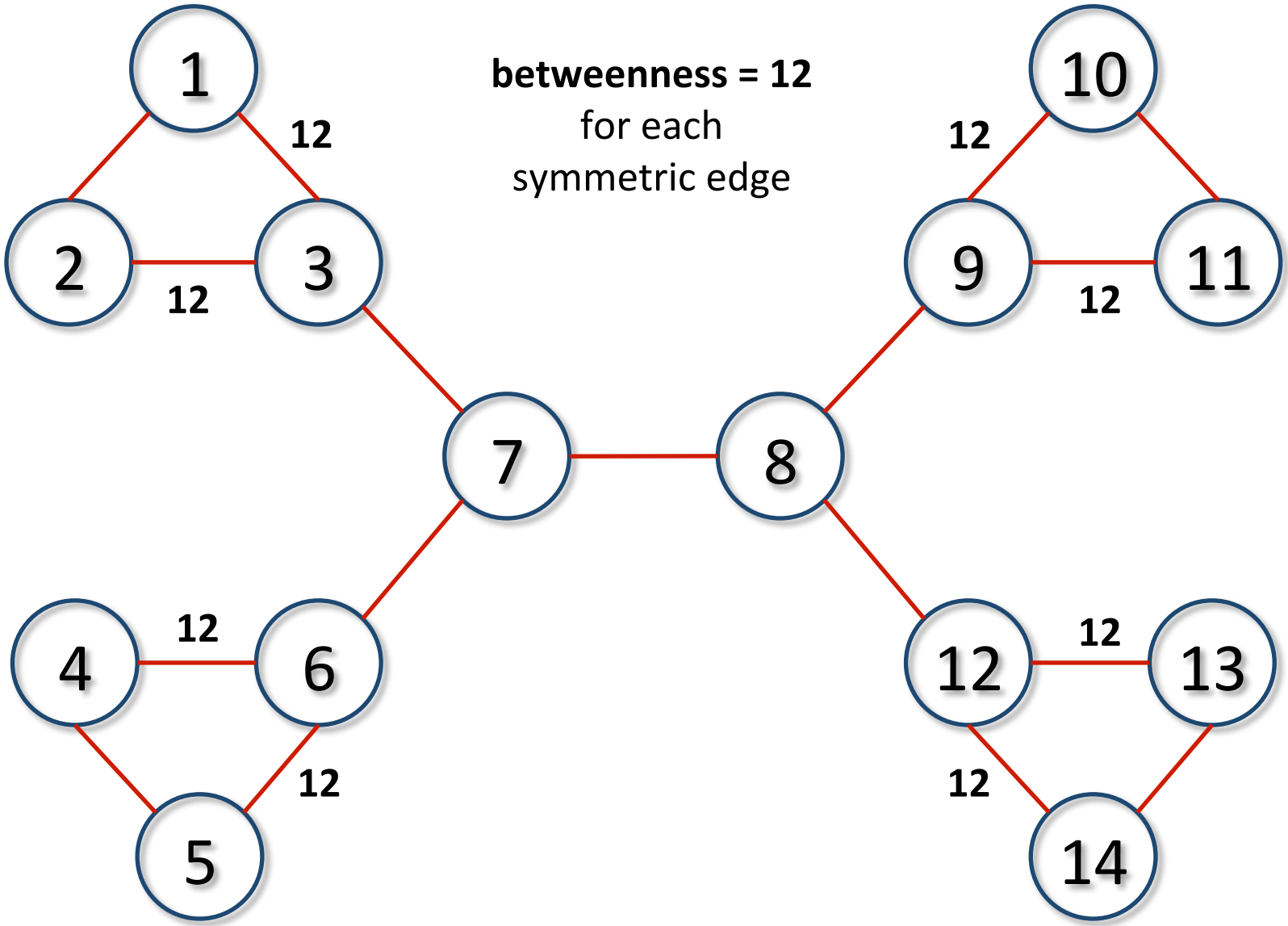

Calculate betweenness
for edge 3-7
1
2
3
4
5
6
7
8
9
10
11
12
13
14

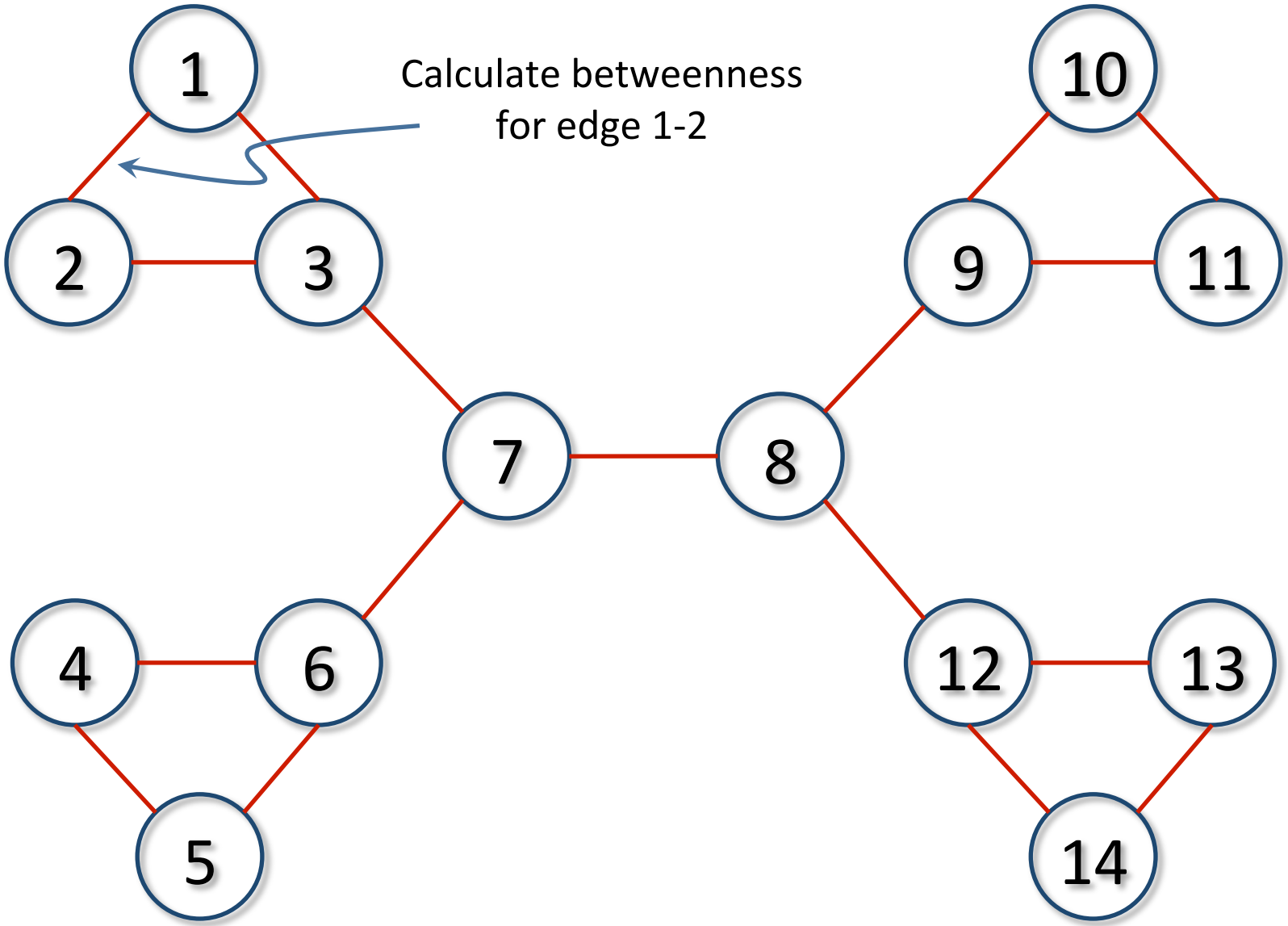

1
2
3
10
9
11
7
8
4
6
12
13
5
14
3 units flow from
1-3 to each 4-14 node,
so total =
3 x 11 = **33**

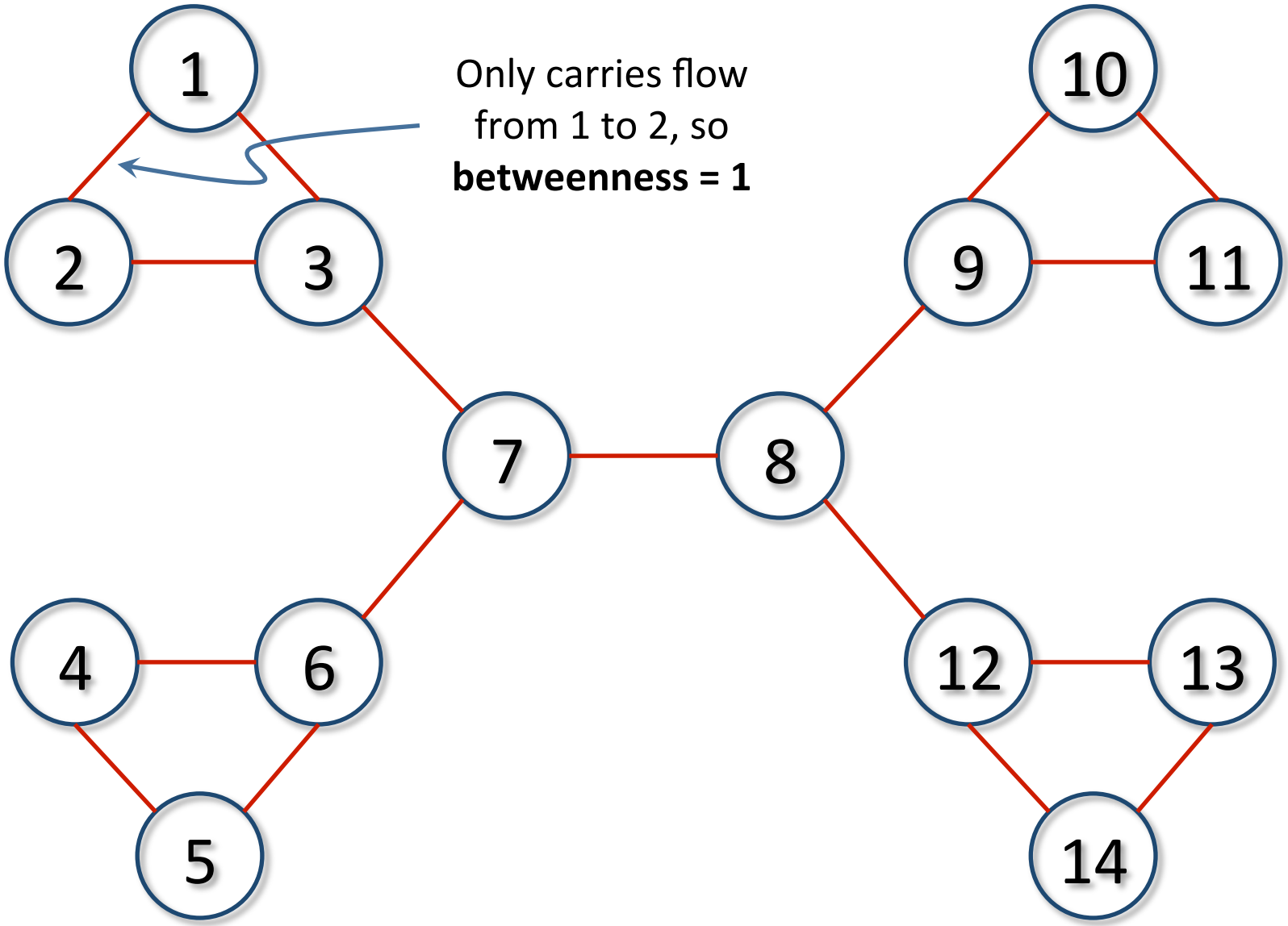

Betweenness = 33
for each
symmetric edge
1
2
3
10
9
11
33
33
7
8
33
33
4
6
12
13
5
14

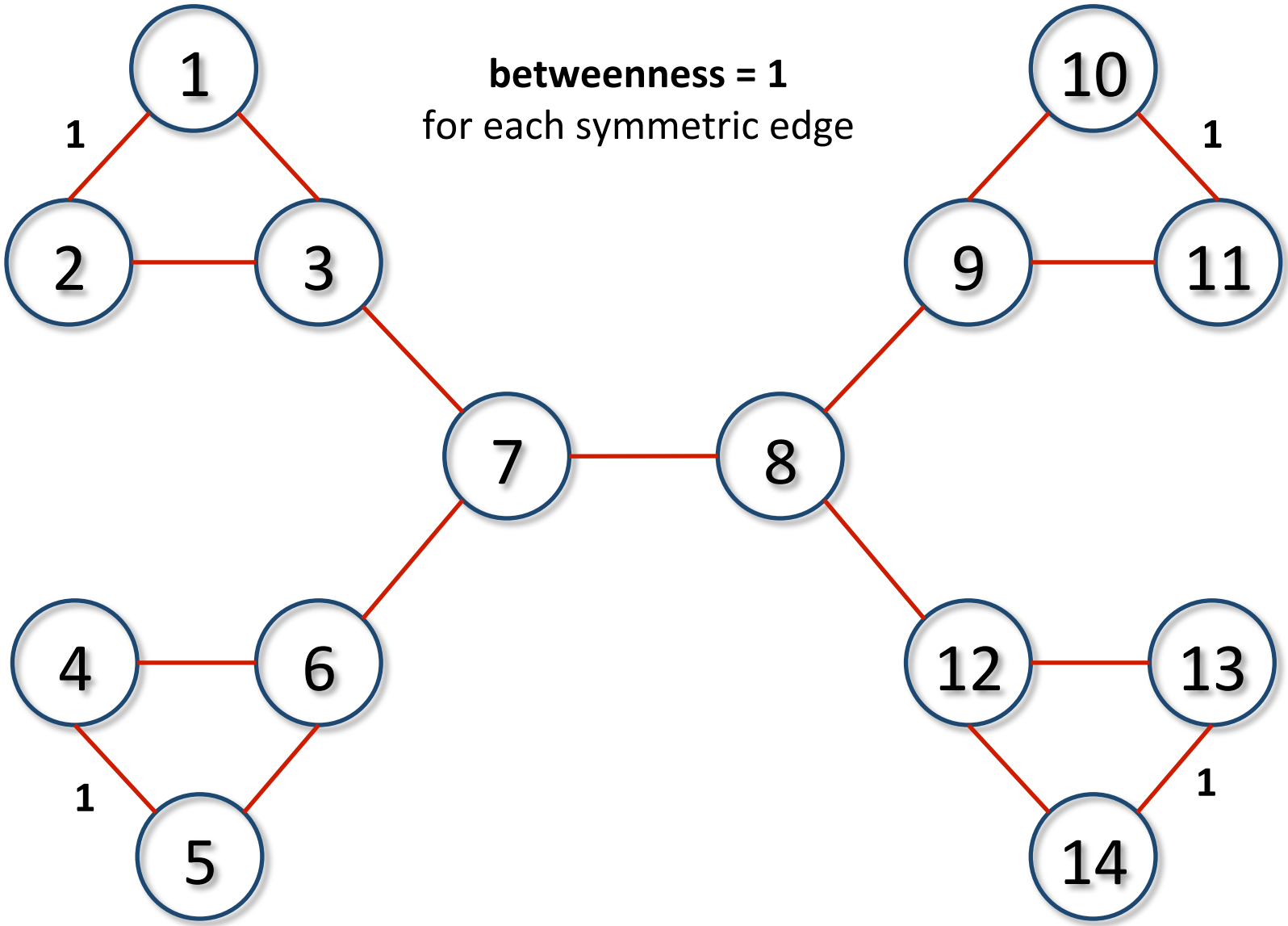

Calculate betweenness
for edge 1-3
1
2
3
4
5
6
7
8
9
10
11
12
13
14

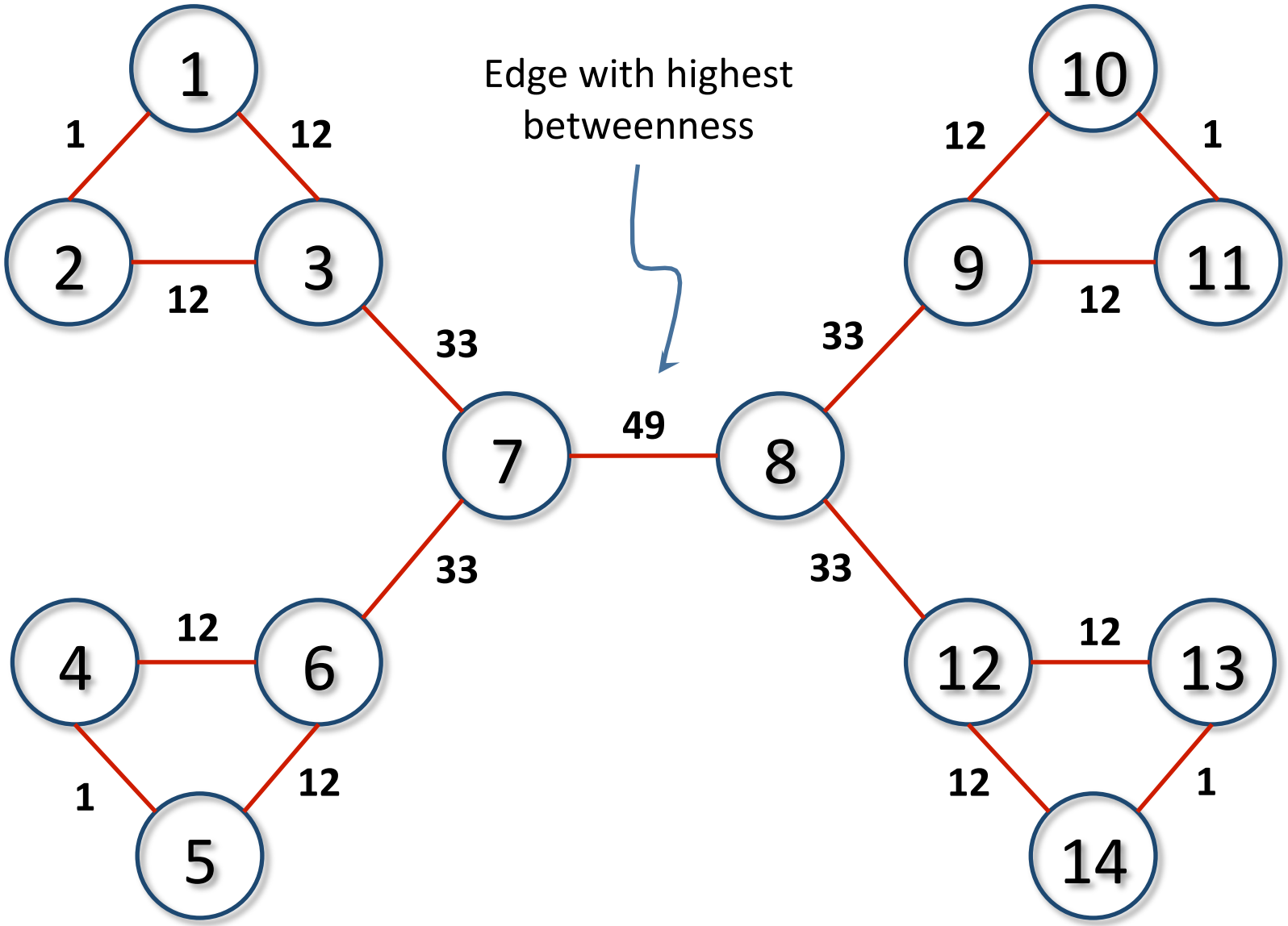

Carries all flow to node 1 except from node 2, so **betweenness = 12**
1
2
3
4
5
6
7
8
9
10
11
12
13
14


betweenness = 12
for each
symmetric edge
1
2
3
4
5
6
7
8
9
10
11
12
13
14
12
12
12
12
12
12
12


Calculate betweenness
for edge 1-2
1
2
3
4
5
6
7
8
9
10
11
12
13
14

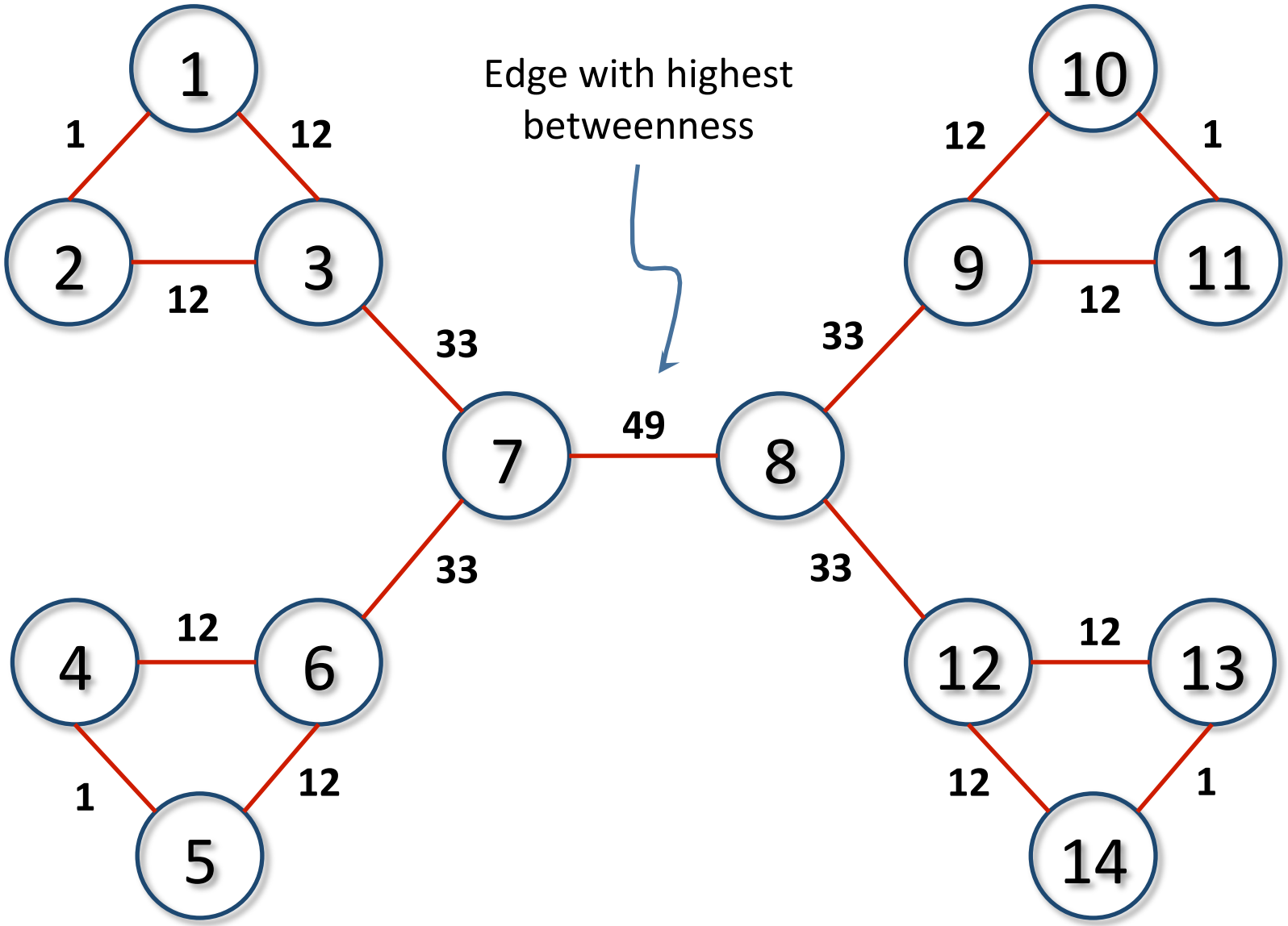

Only carries flow
from 1 to 2, so
betweenness = 1
1
2
3
4
5
6
7
8
9
10
11
12
13
14


betweenness = 1
for each symmetric edge
1
2
3
4
5
6
7
8
9
10
11
12
13
14
1
1
1
1


Edge with highest betweenness
1
2
3
4
5
6
7
8
9
10
11
12
13
14
1
12
12
33
33
49
33
33
12
12
1
12
12
1
12
12
1

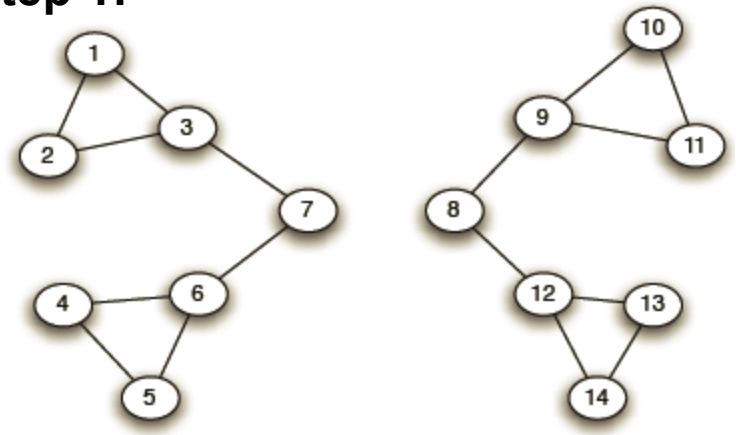
Girvan-Newman Algorithm

1. Calculate betweenness of all edges
2. Remove the edge(s) with highest betweenness
3. Repeat steps 1 and 2 until graph is partitioned into as many regions as desired

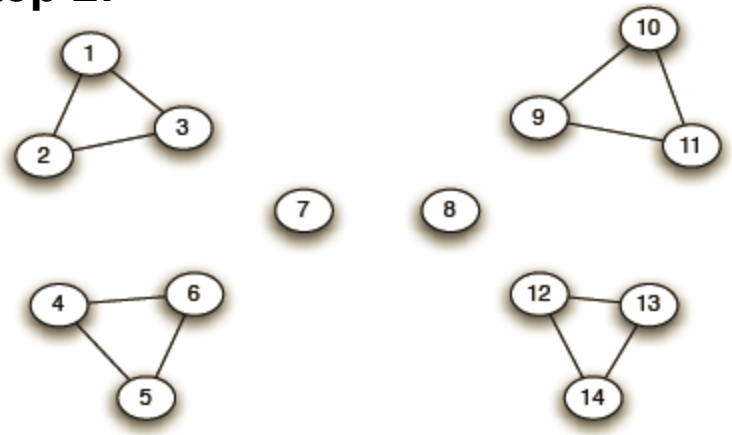

Edge with highest betweenness
1
2
3
4
5
6
7
8
9
10
11
12
13
14
1
12
12
33
33
49
33
33
12
12
1
12
12
1
12
12
1

Girvan-Newman: Example

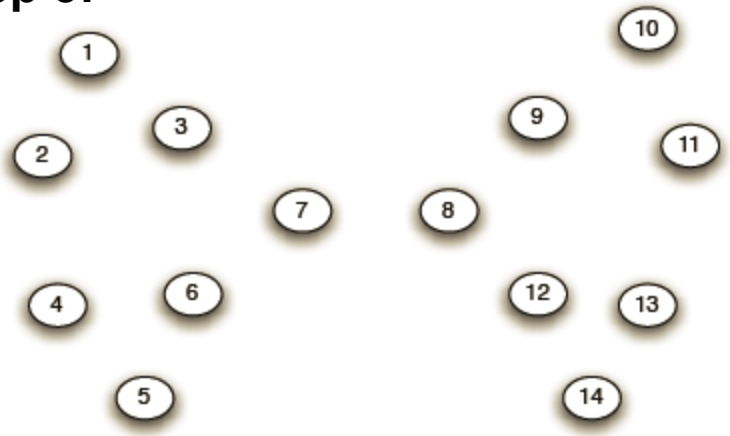
Step 1:



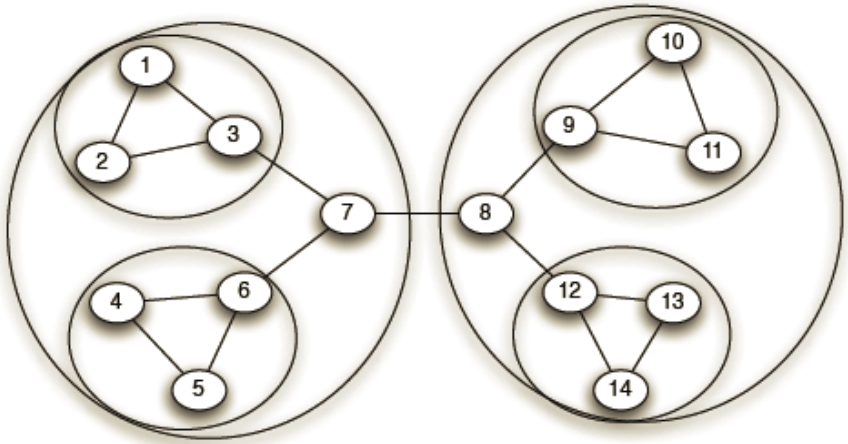
Step 2:



Step 3:



Hierarchical network decomposition:



Girvan-Newman Algorithm

- In general, there can be many shortest path between any two nodes

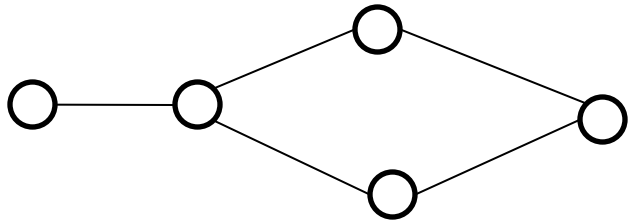
- **Edge betweenness of edge *e*: (full version)**
  Total amount of "flow" an edge *e* carries between all pairs of nodes where a single unit of flow between two nodes divides itself evenly among *all shortest paths* between the nodes (1/*k* units flow along each of *k* shortest paths)

Edge Betweenness

- Edge betweenness centrality
  - Number of shortest paths that pass through a given edge
  - “If there are k (>1) shortest paths between a pair of vertices, each path is given equal weight 1/k (total weight of all the paths is 1).”

Girvan-Newman Algorithm

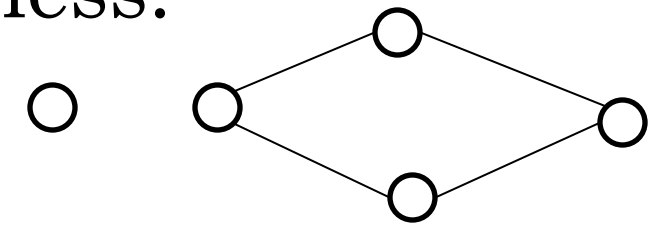
- 1. Calculate the betweenness for all edges in the network.



ab	**4**
bc	**3**
bd	**3**
ce	**3**
de	**3**

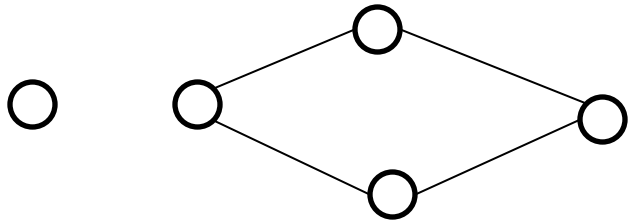
Girvan-Newman Algorithm

- 2. Remove the edge with the highest betweenness.



Girvan-Newman Algorithm

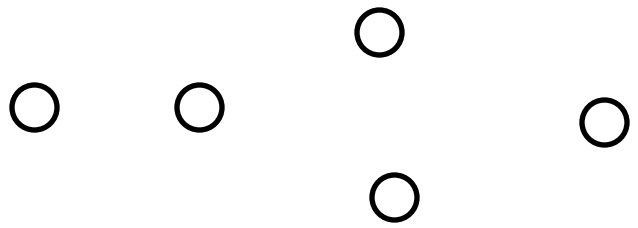
- 3. Recalculate betweennesses for all edges affected by the removal.



bc	**2**
bd	**2**
ce	**2**
de	**2**

Girvan-Newman Algorithm

- 4. Repeat from step 2 until no edges remain.



Calculating edge-betweenness

Algorithm for all-edge betweenness

- Choose two vertices
  - Calculate all shortest paths between these two vertices; (suppose the are *k* such path)
  - For each such shortest path

    Increment the betweenness of every edge on the path by 1/k
- Repeat for every pair of vertices

Girvan-Newman Complexity

Complexity:

- Original algorithm is $O(mn^2)$

- Faster algorithm

  $O(mn)$, or $O(n^2)$ on sparse graph

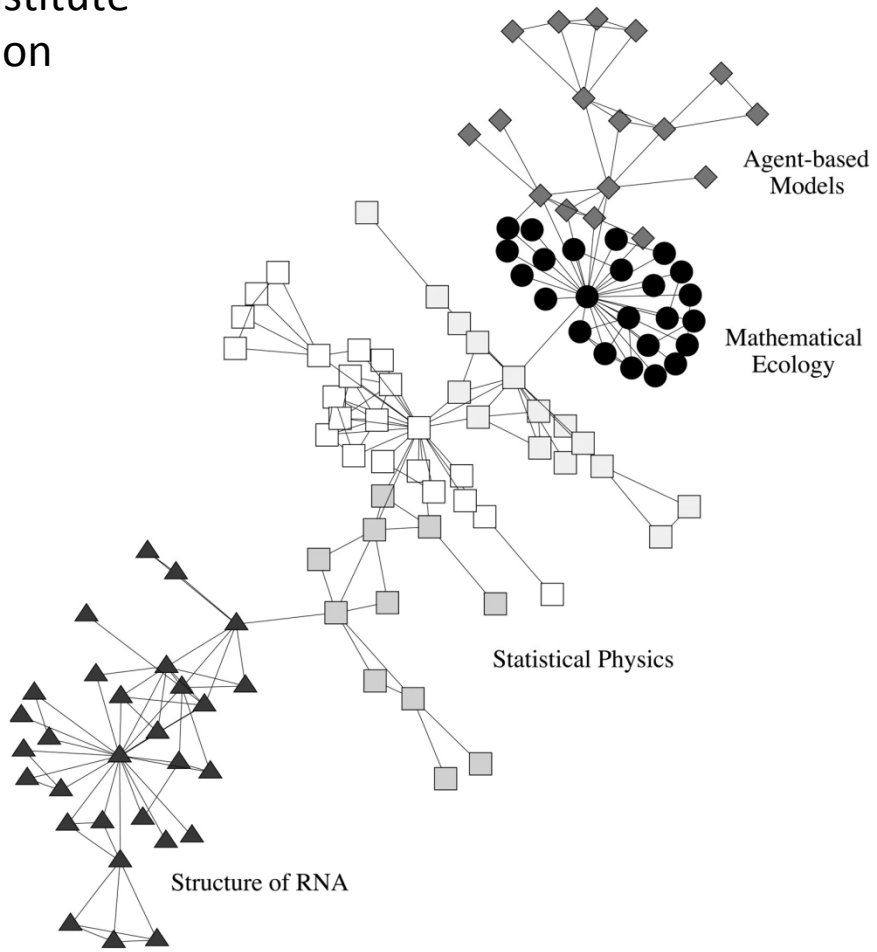
Girvan-Newman Algorithm

- Intuitively, why should this work? Analogy:
  - Network of N nodes: nodes are towns, edges are roads
  - Place N-1 cars on the nodes; each one to a town
  - Each road gets a point when a car drives on it
  - Remove the highest ranked road – interstate highway
  - Repeat the process

  - First we'll remove all interstates (leaving state roads)
  - Then state roads will be removed, leaving county roads, then suburban roads, etc
  - After we each set of levels, we get a more fine-grained division of communities

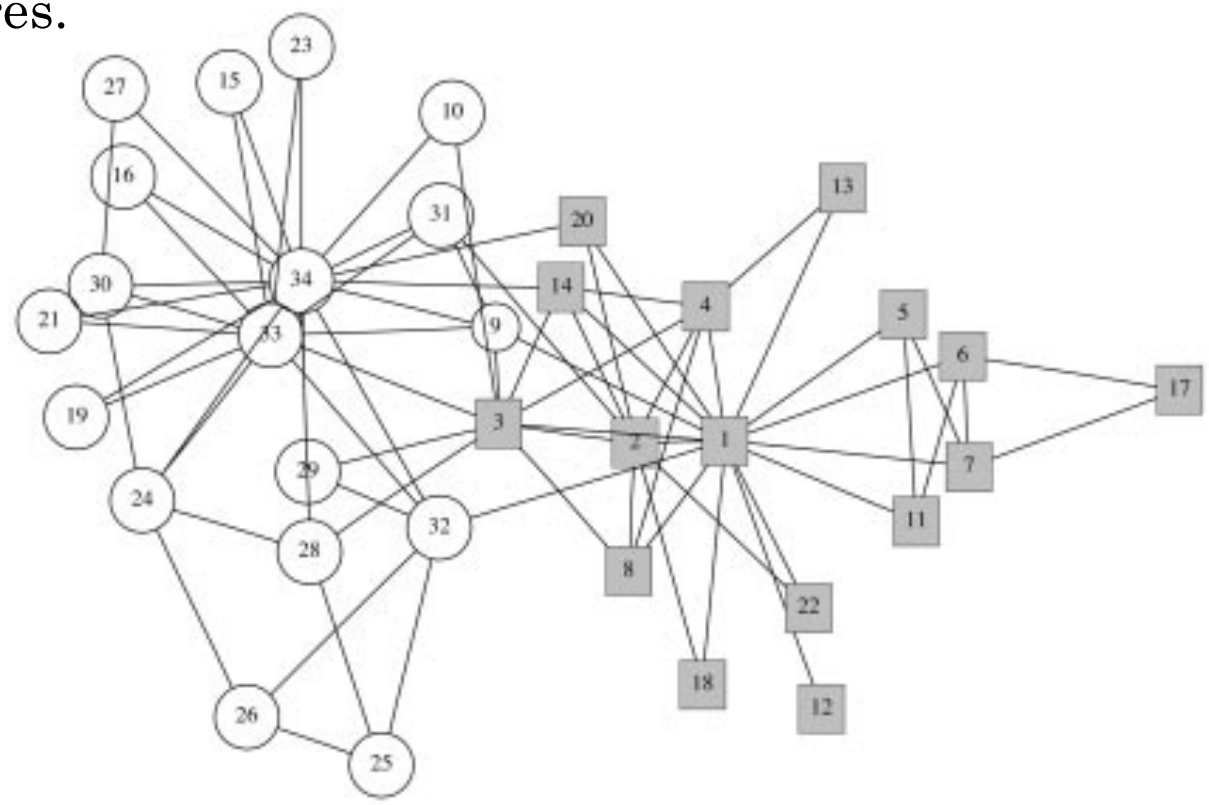
Girvan-Newman Output

Santa Fe Institute Collaboration Network



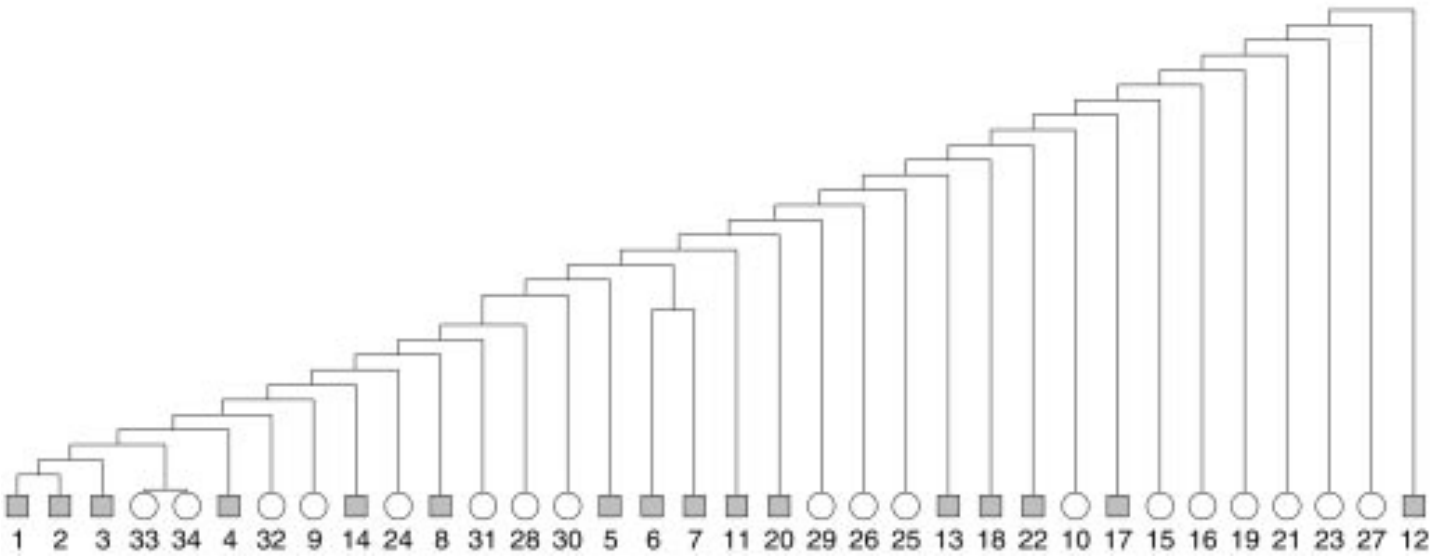
TESTS OF THE METHODS

- The friendship network : nodes associated with the club administrator's faction are drawn as circles, those associated with the instructor's faction are drawn as squares.



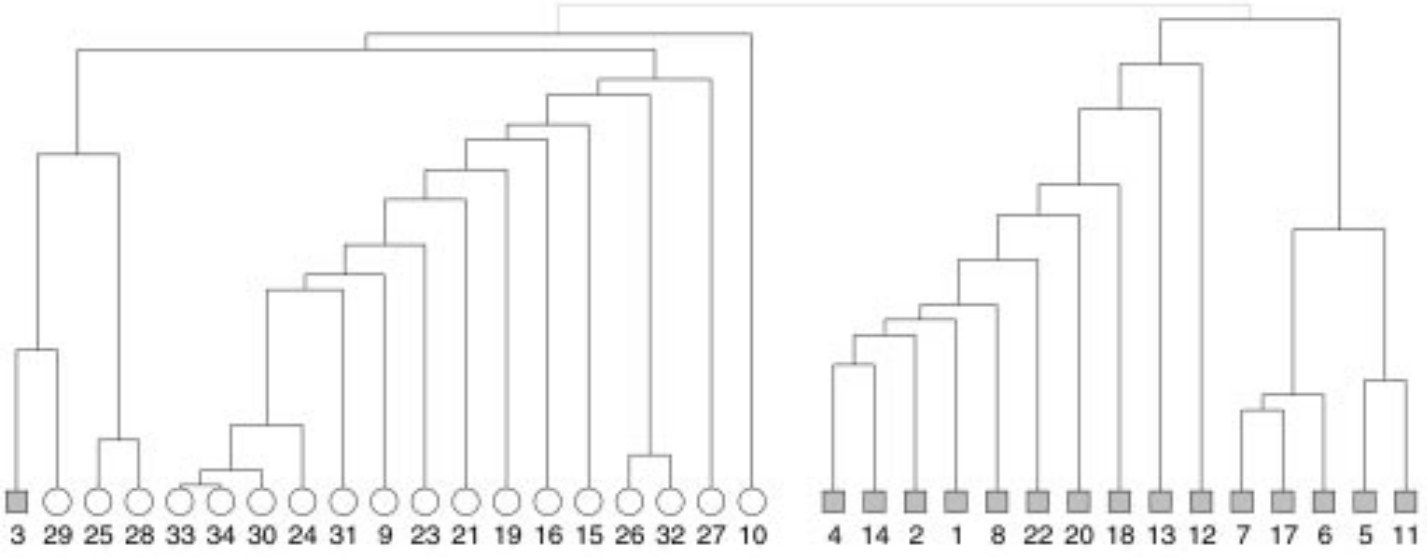
TESTS OF THE METHODS(CONT.)

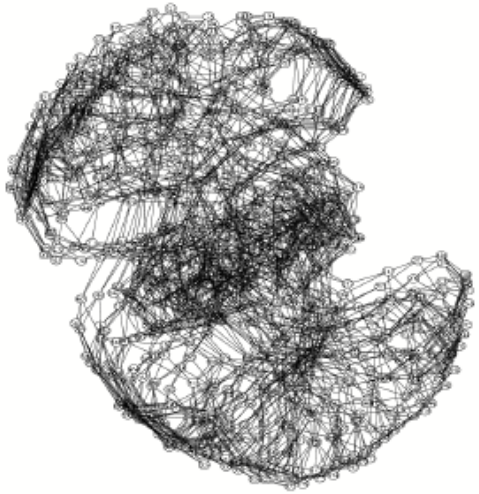
- Hierarchical tree calculated by using edge-independent path counts, which fails to extract the known community structure of the network



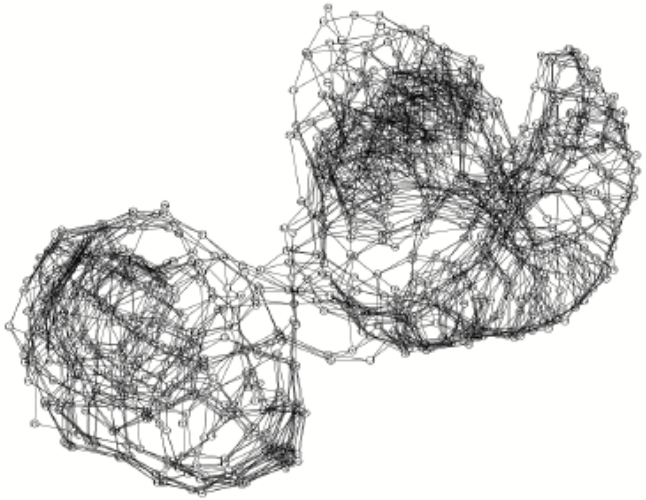
TESTS OF THE METHODS(CONT.)

- Hierarchical tree showing the complete community structure for the network calculated by using the algorithm presented in this article. The initial split of the network into two groups is in agreement with the actual factions observed by Zachary, with the exception that node 3 is misclassified.

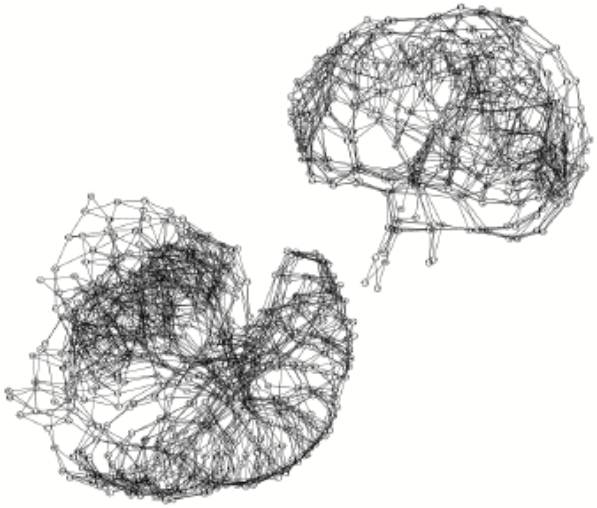




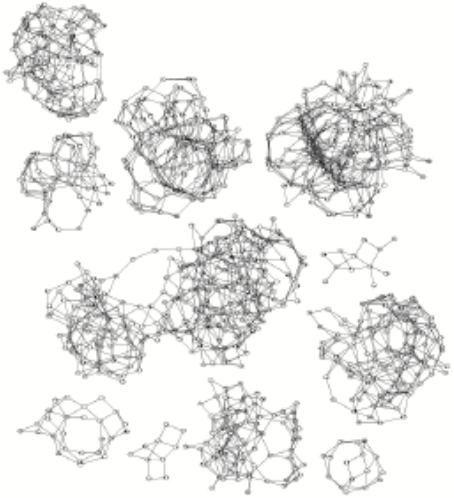
0 cuts



100 cuts



120 cuts



500 cuts

When do you stop cutting?

Modularity

Newman, Girvan (2004)

e_{ij} is equal to the number of links between community i and community j.

$$Q = \sum_i e_{ii} - \sum_{ijk} e_{ij} e_{ki} = \mathrm{Tr}\, \mathbf{e} - \| \mathbf{e}^2 \|, \qquad (4)$$

where $\| \mathbf{x} \|$ indicates the sum of all elements of $\mathbf{x}$. Physically, Q is the fraction of all edges that lie within communities minus the expected value of the same quantity in a graph in which the vertices have the same degrees but edges are placed at random without regard for the

It is important to recalculate

Newman, Girvan (2004)

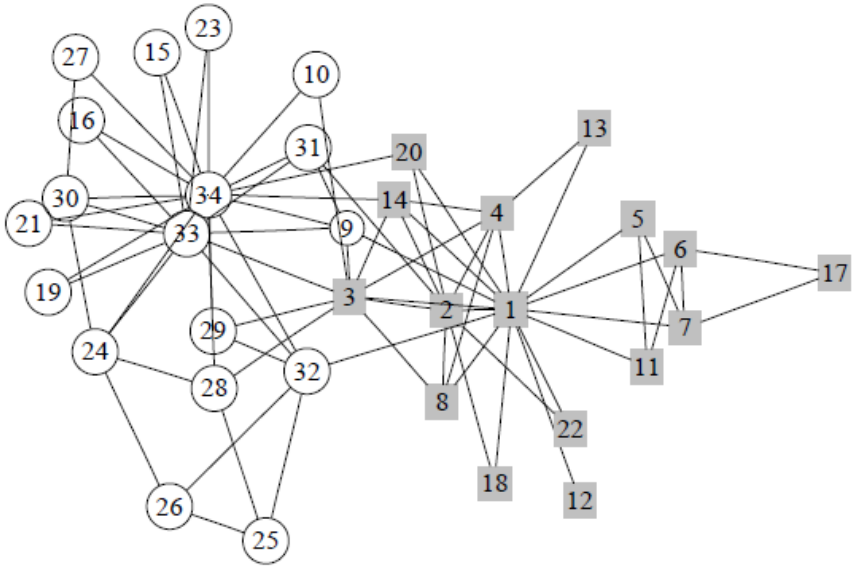
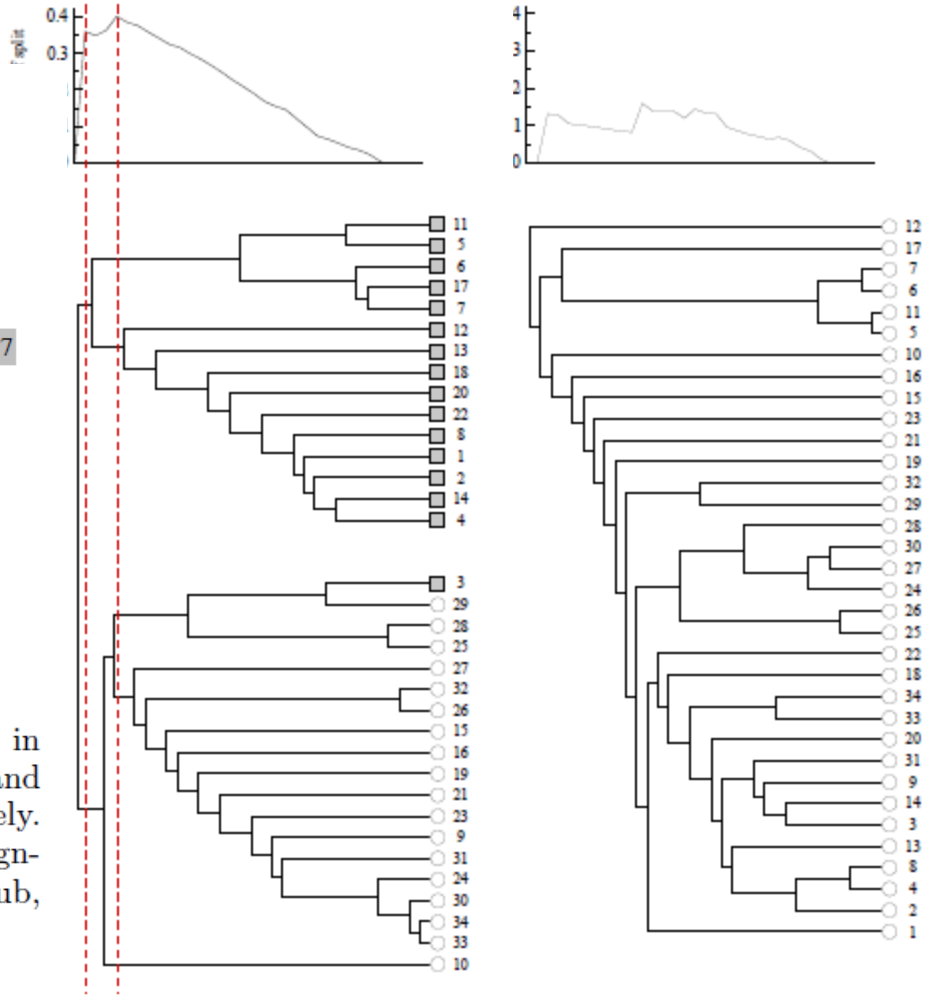


FIG. 8: The network of friendships between individuals in the karate club study of Zachary [35]. The administrator and the instructor are represented by nodes 1 and 33 respectively. Shaded squares represent individuals to who ended up aligning with the club's administrator after the fission of the club, open circles those who aligned with the instructor.



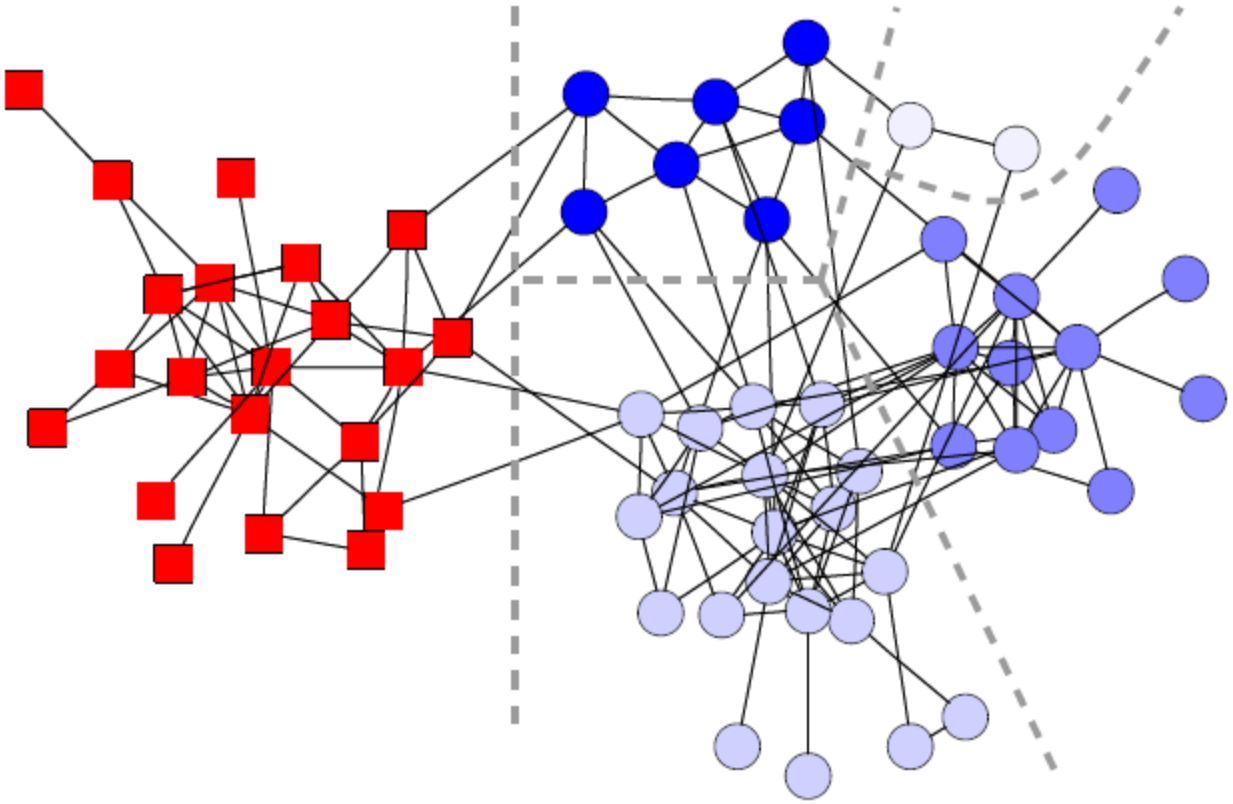


FIG. 4: Community structure in the social network of bottlenose dolphins assembled by Lusseau *et al.* [36, 37], ex-

Newman, Girvan (2004)

NUS
National University
of Singapore

School of Computing

Thank you.

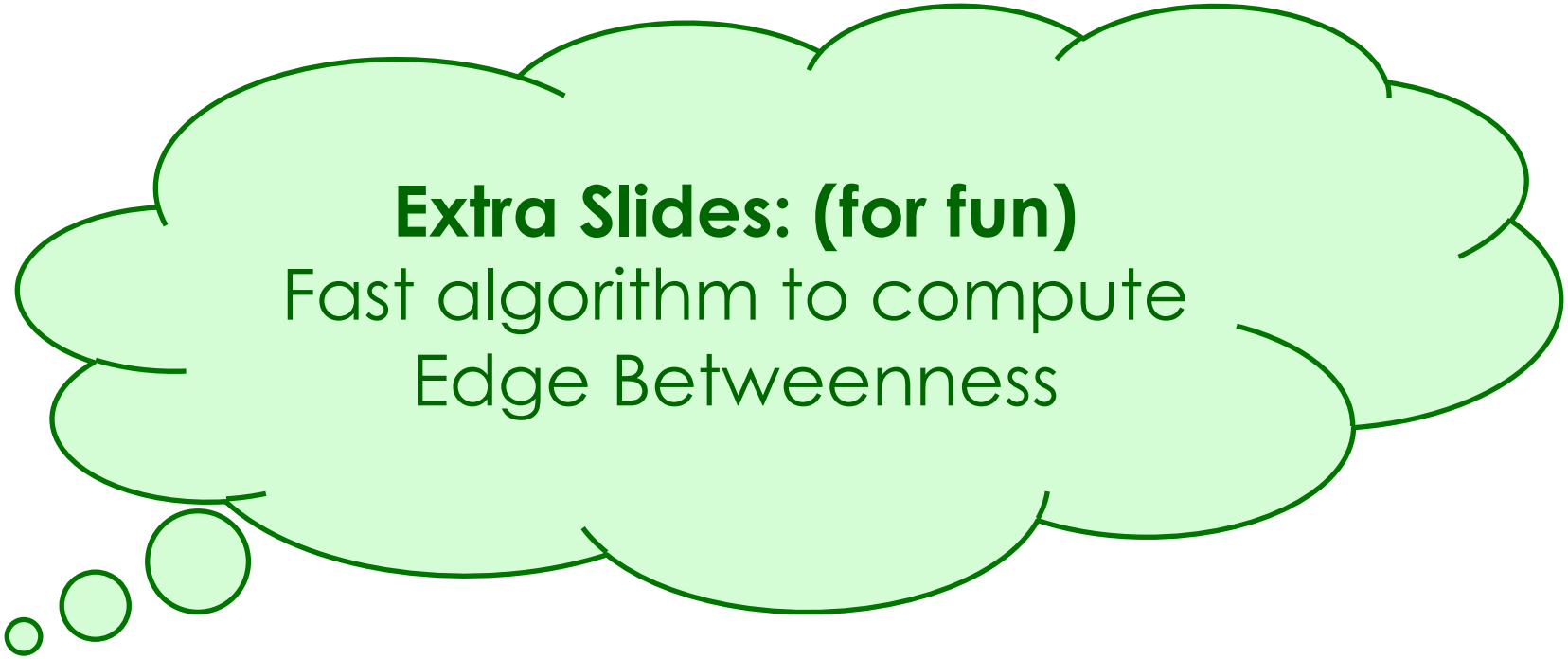
Q & A

Contact:

Hon Wai Leong（梁汉槐）

FB, email: leonghw@comp.nus.edu.sg

http://www.comp.nus.edu.sg/~leonghw/


Extra Slides: (for fun)
Fast algorithm to compute
Edge Betweenness

Girvan-Newman Algorithm

1. **Calculate betweenness of all edges**
2. Remove the edge(s) with highest betweenness
3. Repeat steps 1 and 2 until graph is partitioned into as many regions as desired

How much computation does this require?

Newman (2001) and Brandes (2001) independently developed similar algorithms that reduce the complexity from $O(mn^2)$ to $O(mn)$ where m = # of edges, n = # of nodes

Computing Edge Betweenness Efficiently

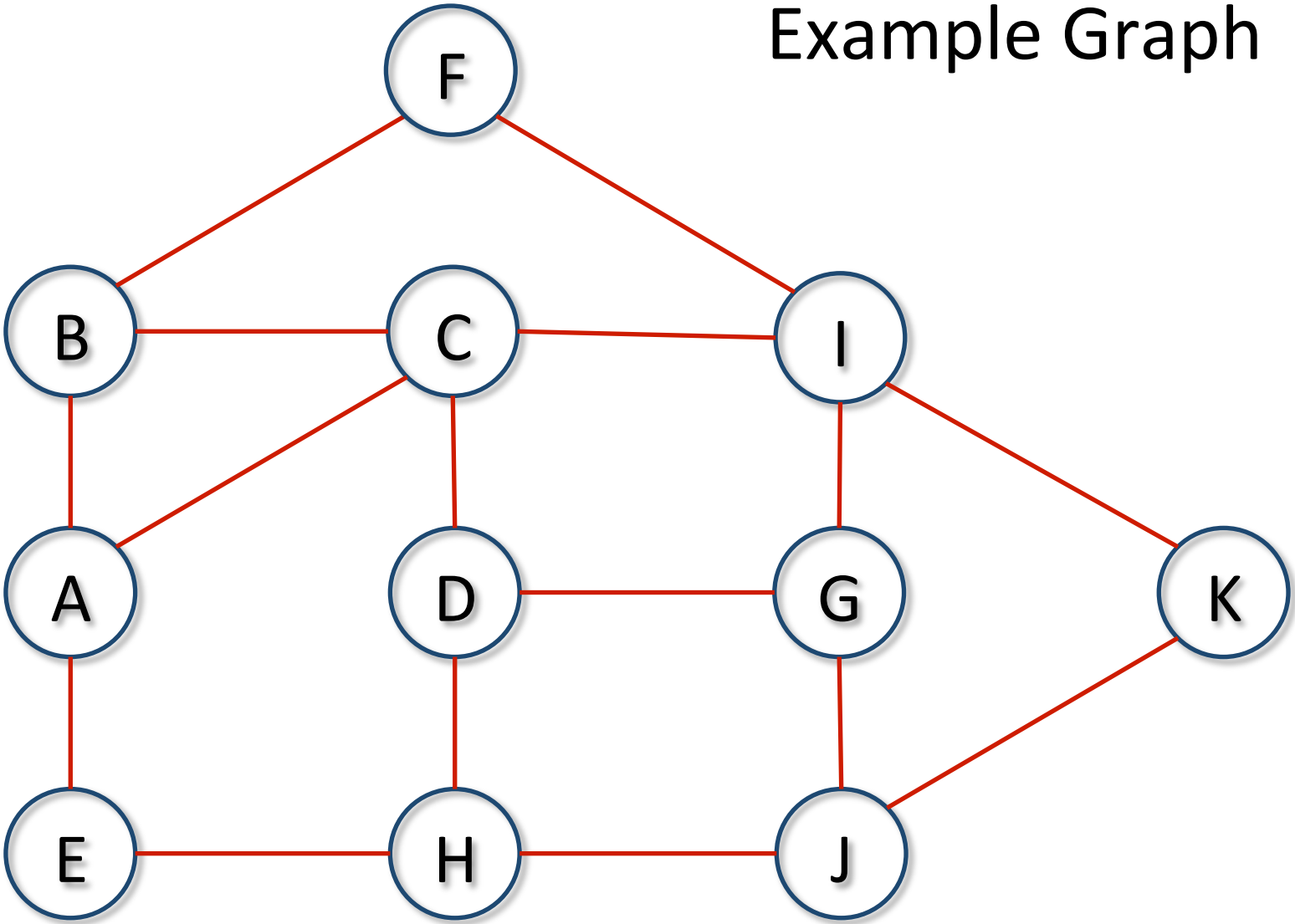
For each node *N* in the graph

1. Perform breadth-first search of graph starting at node *N*
2. Determine the number of shortest paths from *N* to every other node
3. Based on these numbers, determine the amount of flow from *N* to all other nodes that use each edge

Divide sum of flow of all edges by 2

Method developed by Brandes (2001) and Newman (2001)

Example Graph



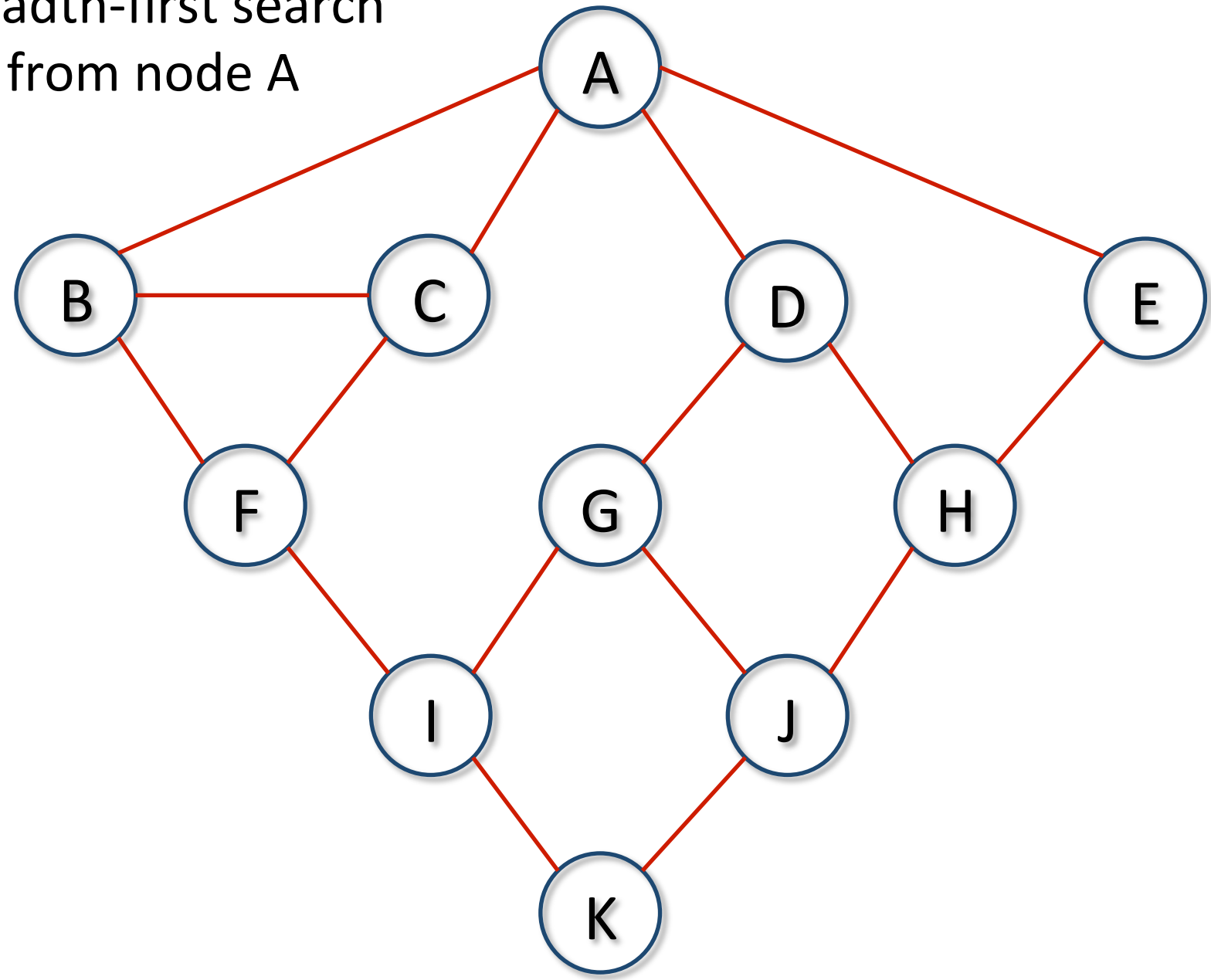
Computing Edge Betweenness Efficiently

For each node *N* in the graph

1. Perform breadth-first search of graph starting at node *N*
2. Determine the number of shortest paths from *N* to every other node
3. Based on these numbers, determine the amount of flow from *N* to all other nodes that use each edge

Divide sum of flow of all edges by 2

Breadth-first search
from node A

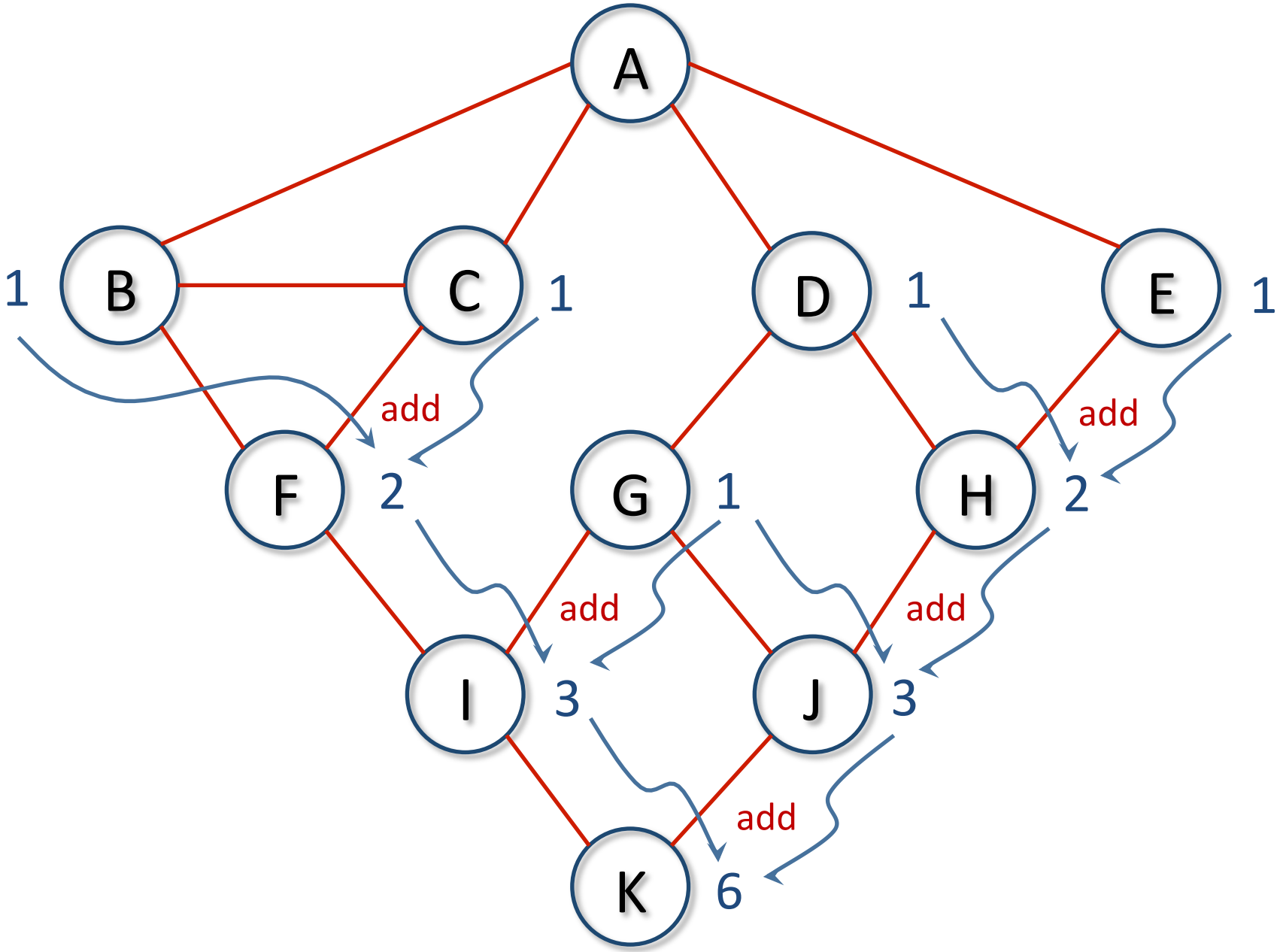


Computing Edge Betweenness Efficiently

For each node *N* in the graph

1. Perform breadth-first search of graph starting at node *N*
2. Determine the number of shortest paths from *N* to every other node
3. Based on these numbers, determine the amount of flow from *N* to all other nodes that use each edge

Divide sum of flow of all edges by 2

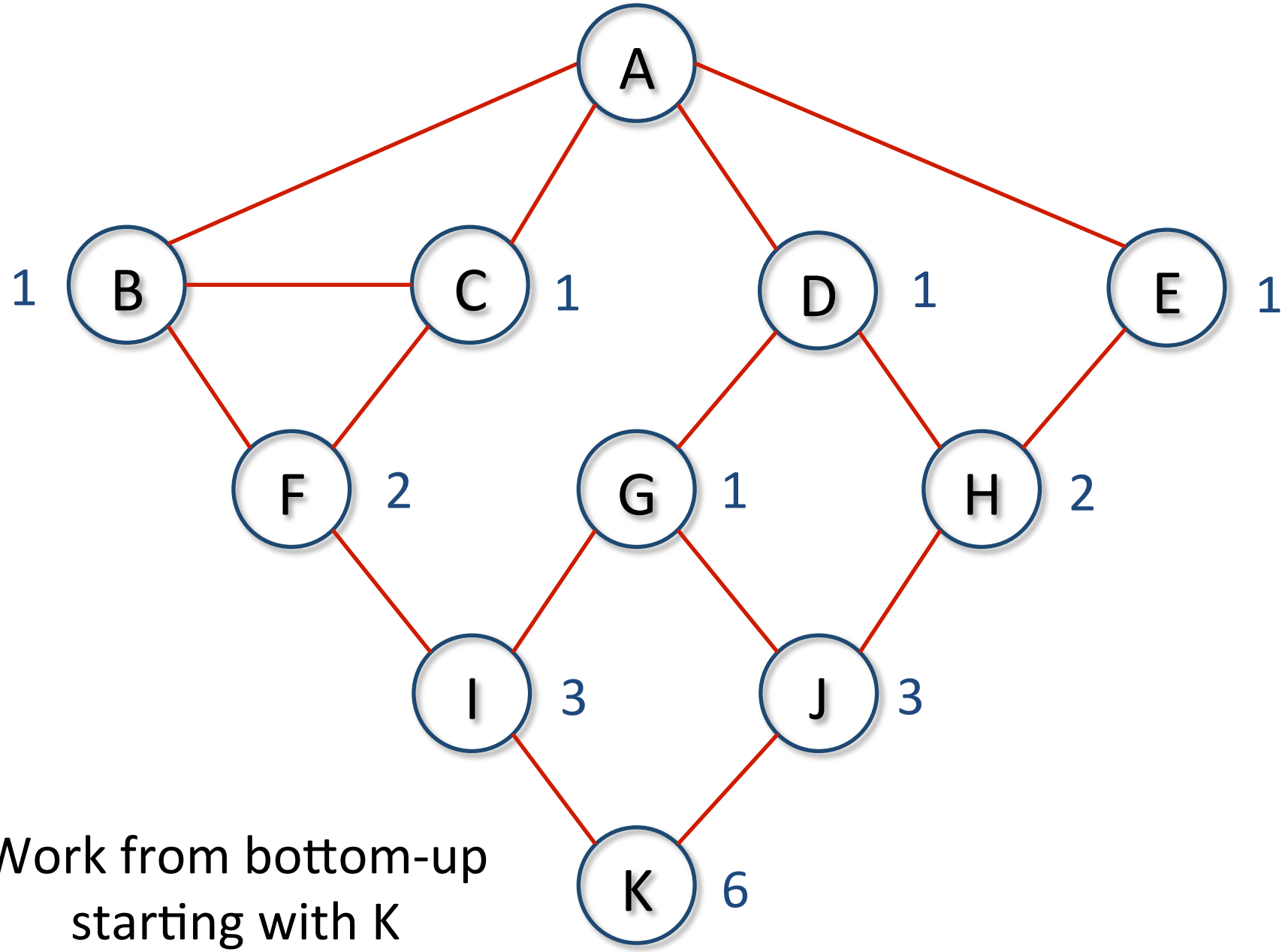

A
B
C
D
E
F
G
H
I
J
K
1
1
1
1
add
2
add
2
1
add
add
3
3
add
6

Computing Edge Betweenness Efficiently

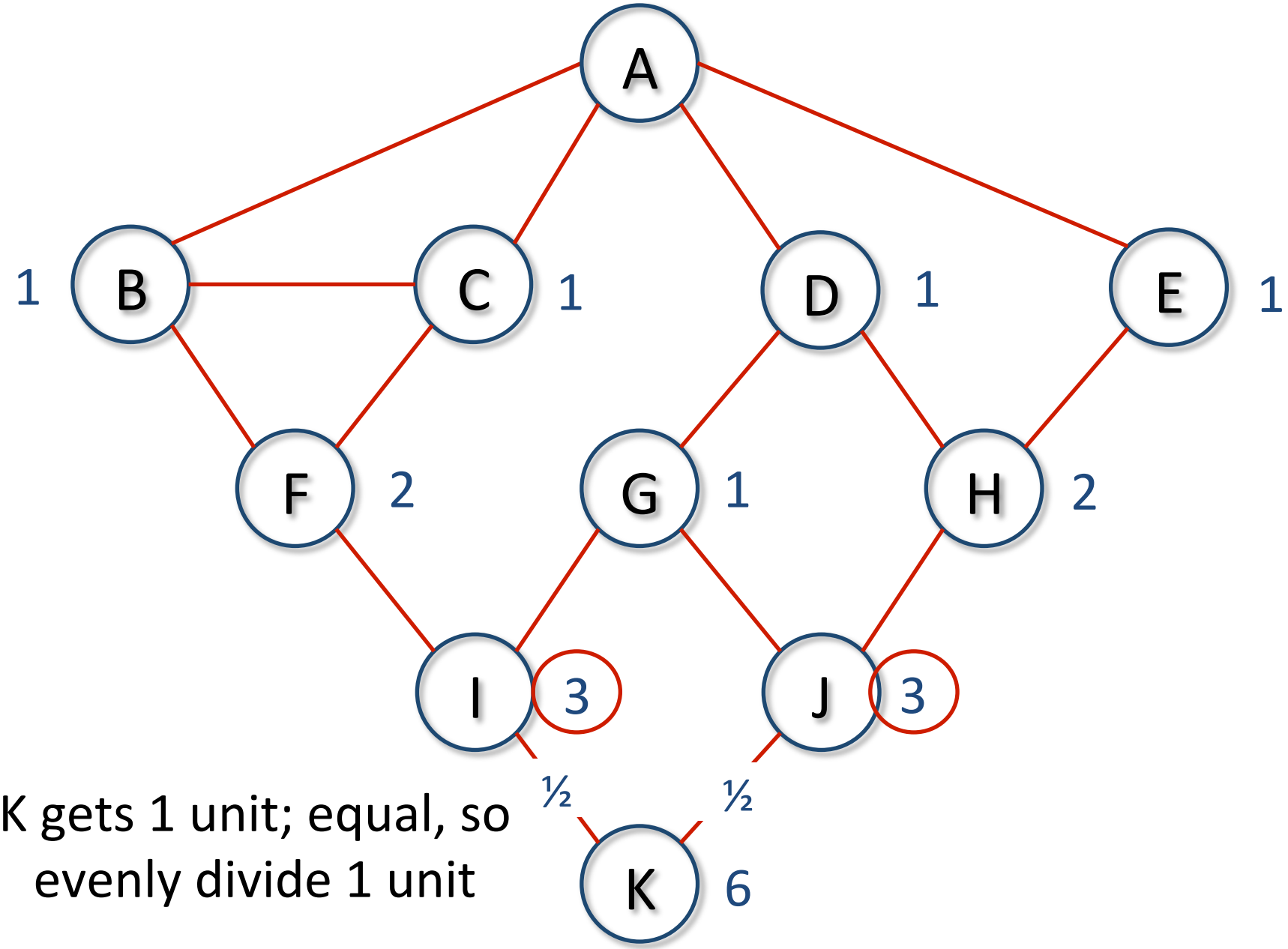
For each node *N* in the graph

1. Perform breadth-first search of graph starting at node *N*
2. Determine the number of shortest paths from *N* to every other node
3. Based on these numbers, determine the amount of flow from *N* to all other nodes that use each edge

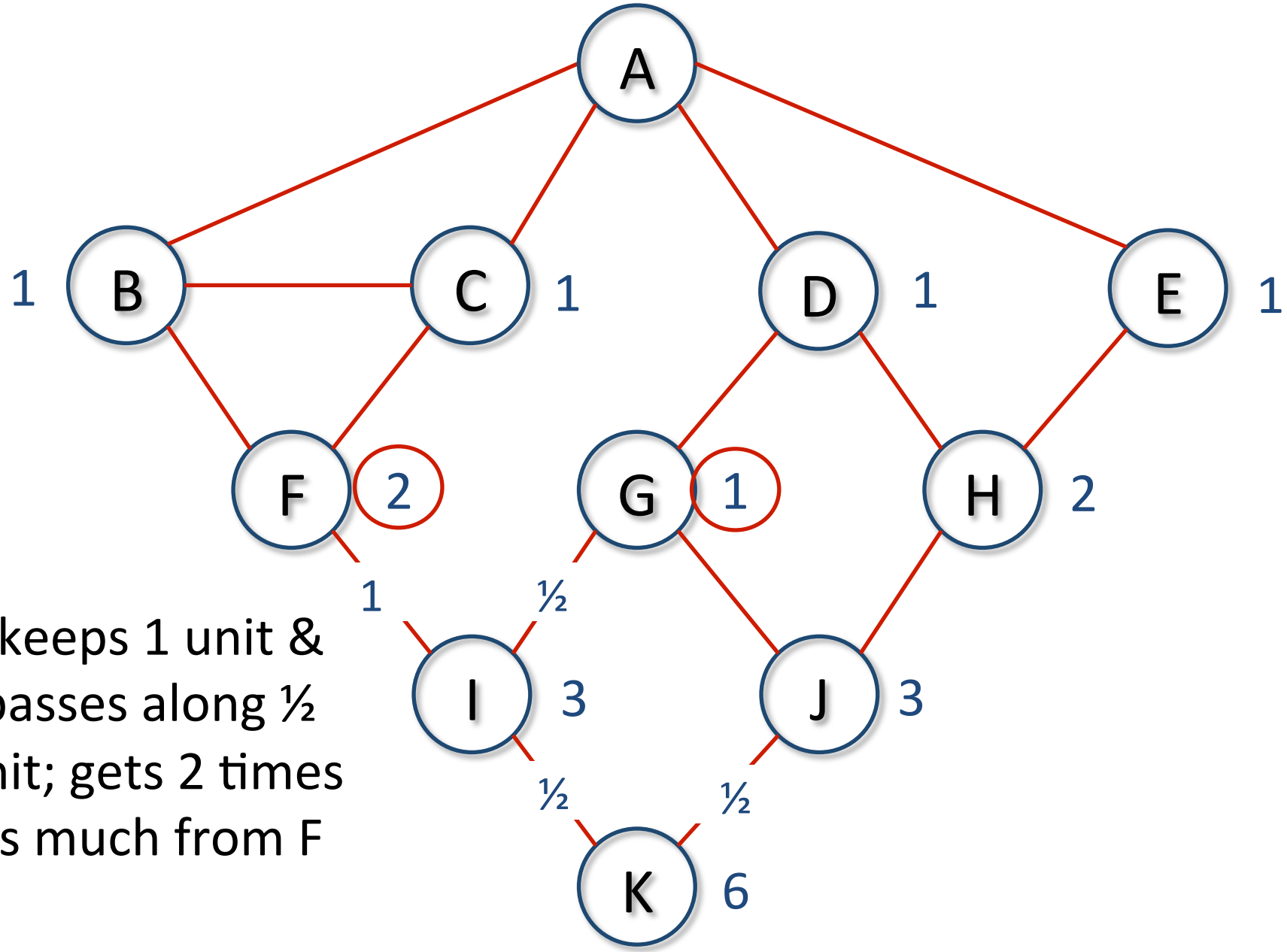
Divide sum of flow of all edges by 2



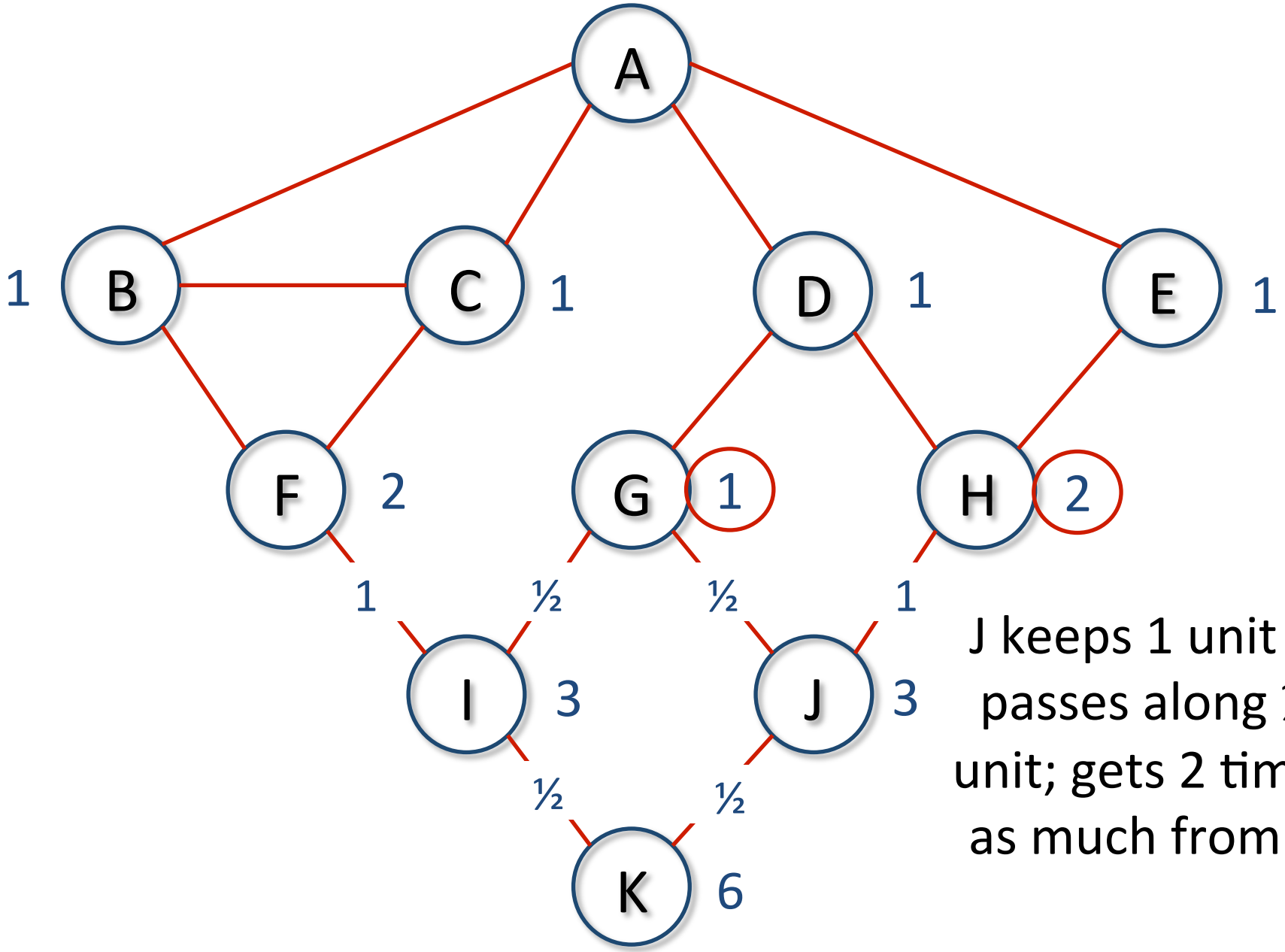
Work from bottom-up starting with K



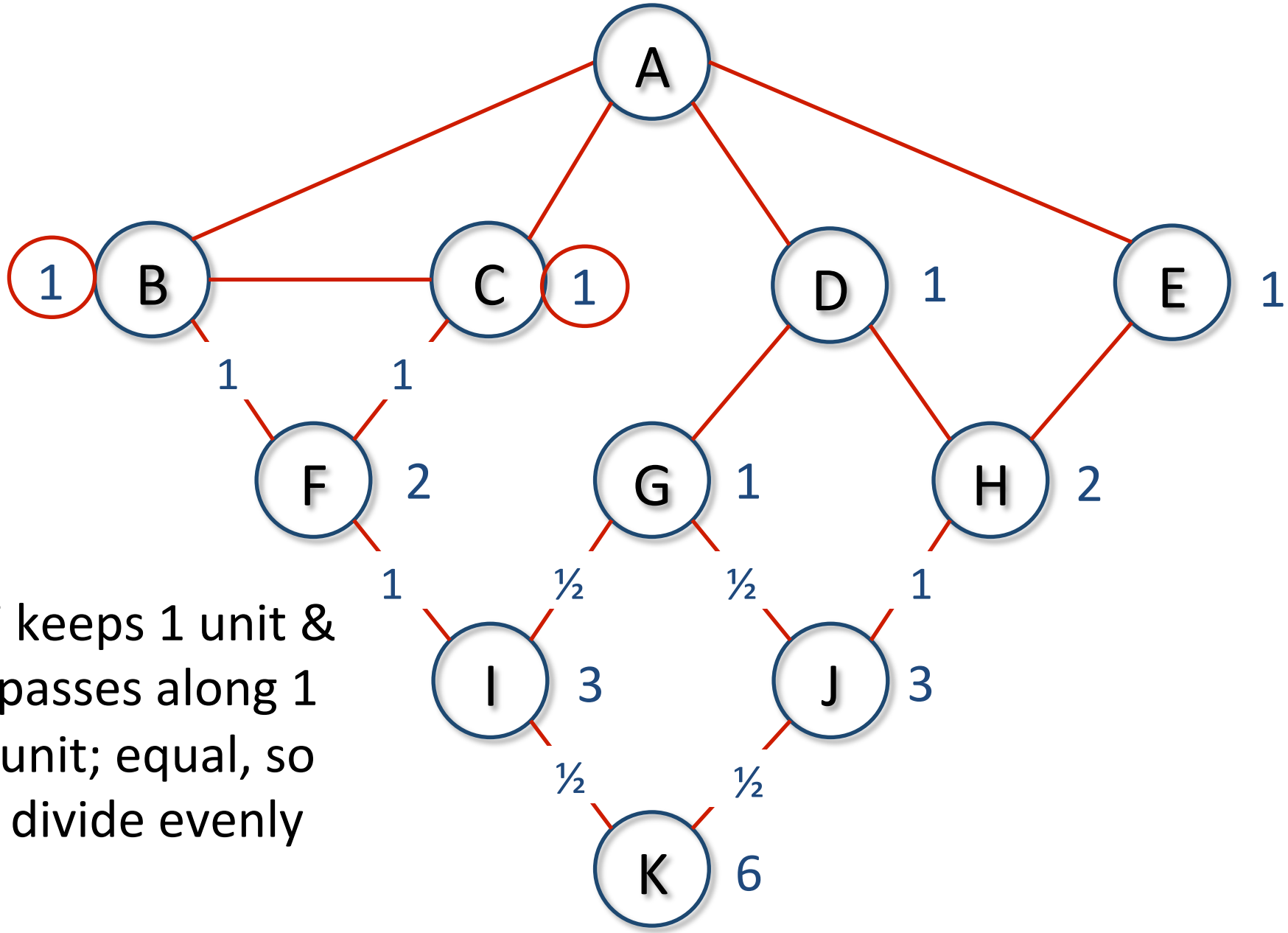
K gets 1 unit; equal, so evenly divide 1 unit



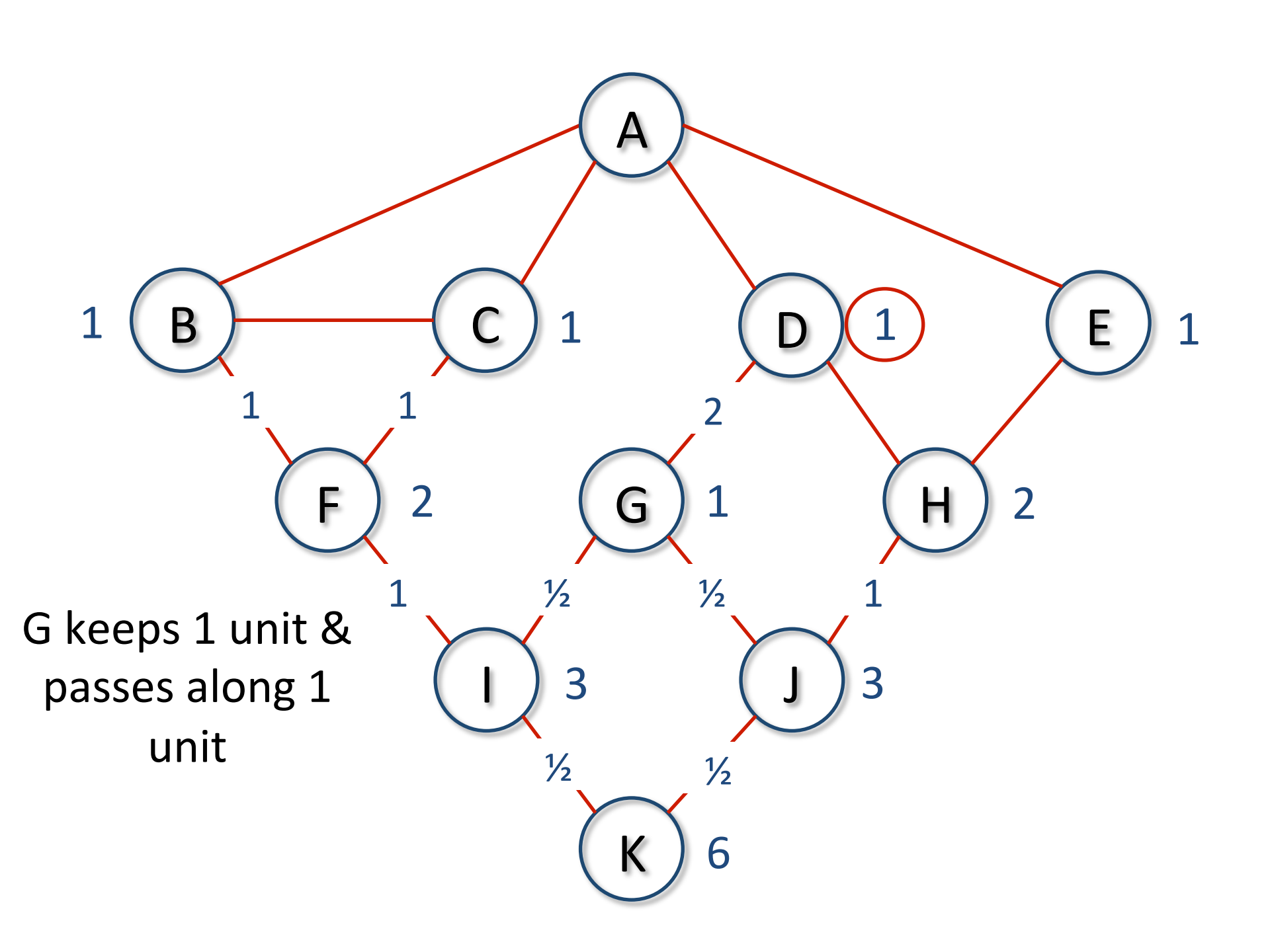
I keeps 1 unit & passes along ½ unit; gets 2 times as much from F

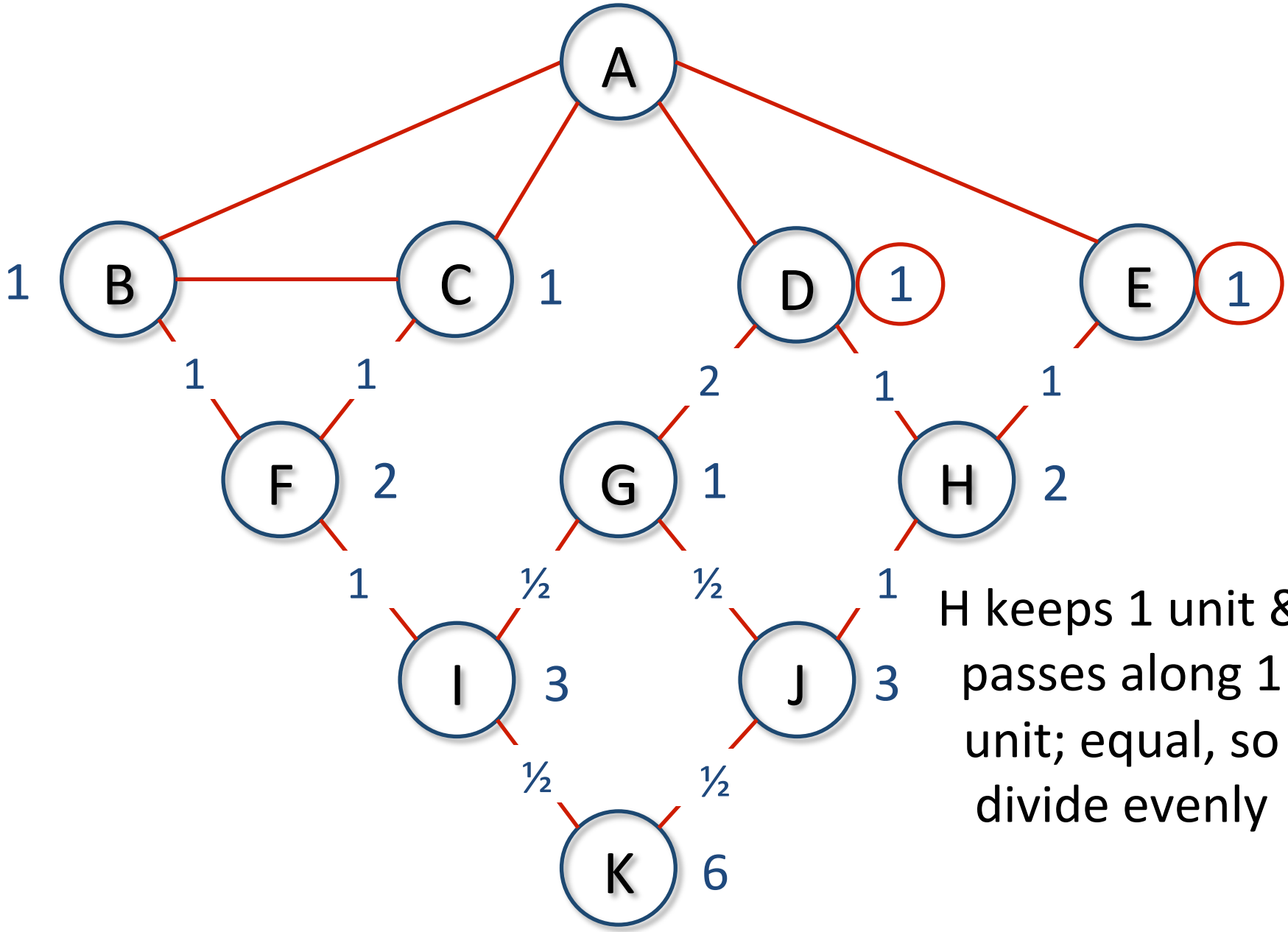


J keeps 1 unit & passes along ½ unit; gets 2 times as much from H

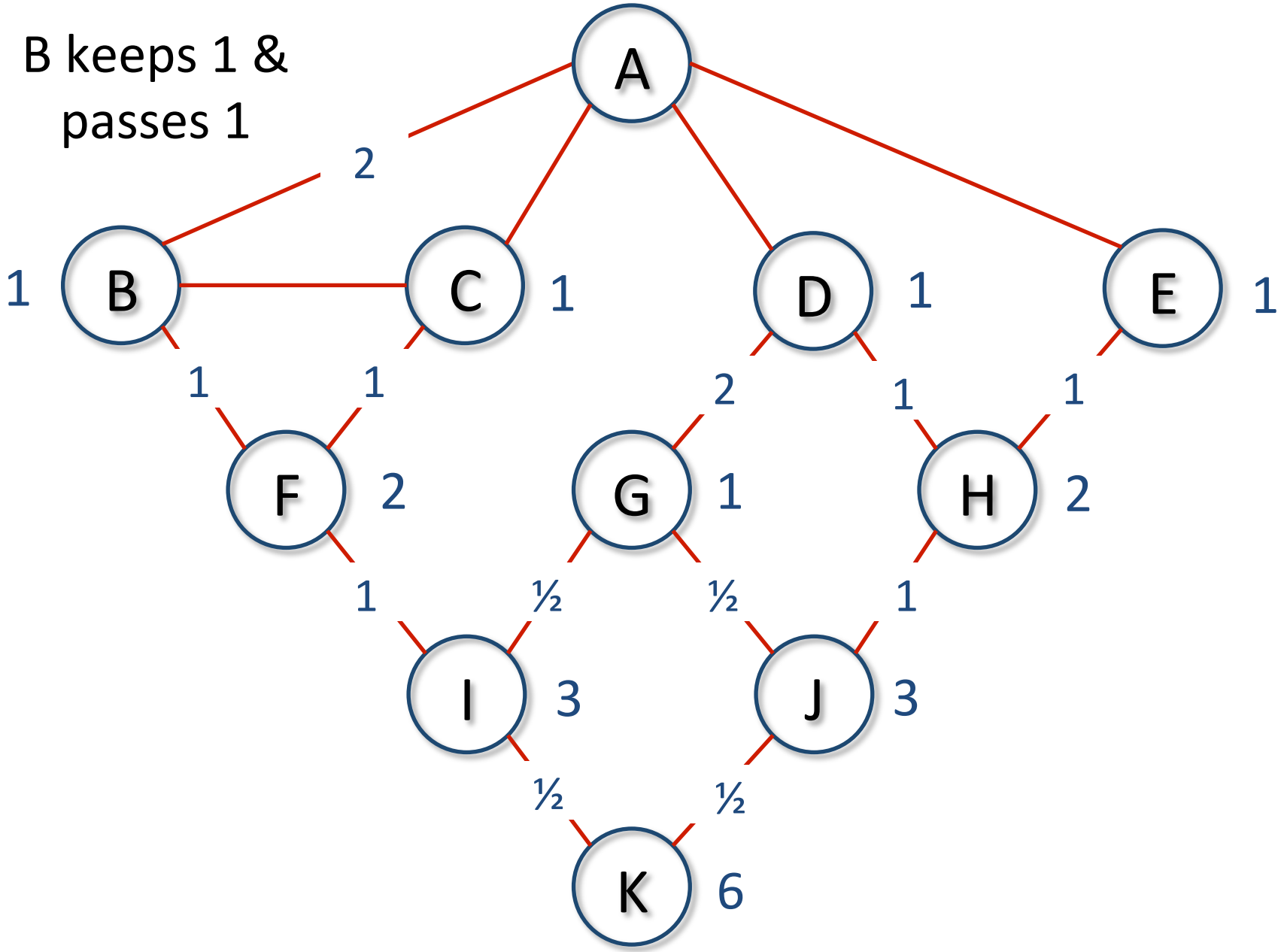


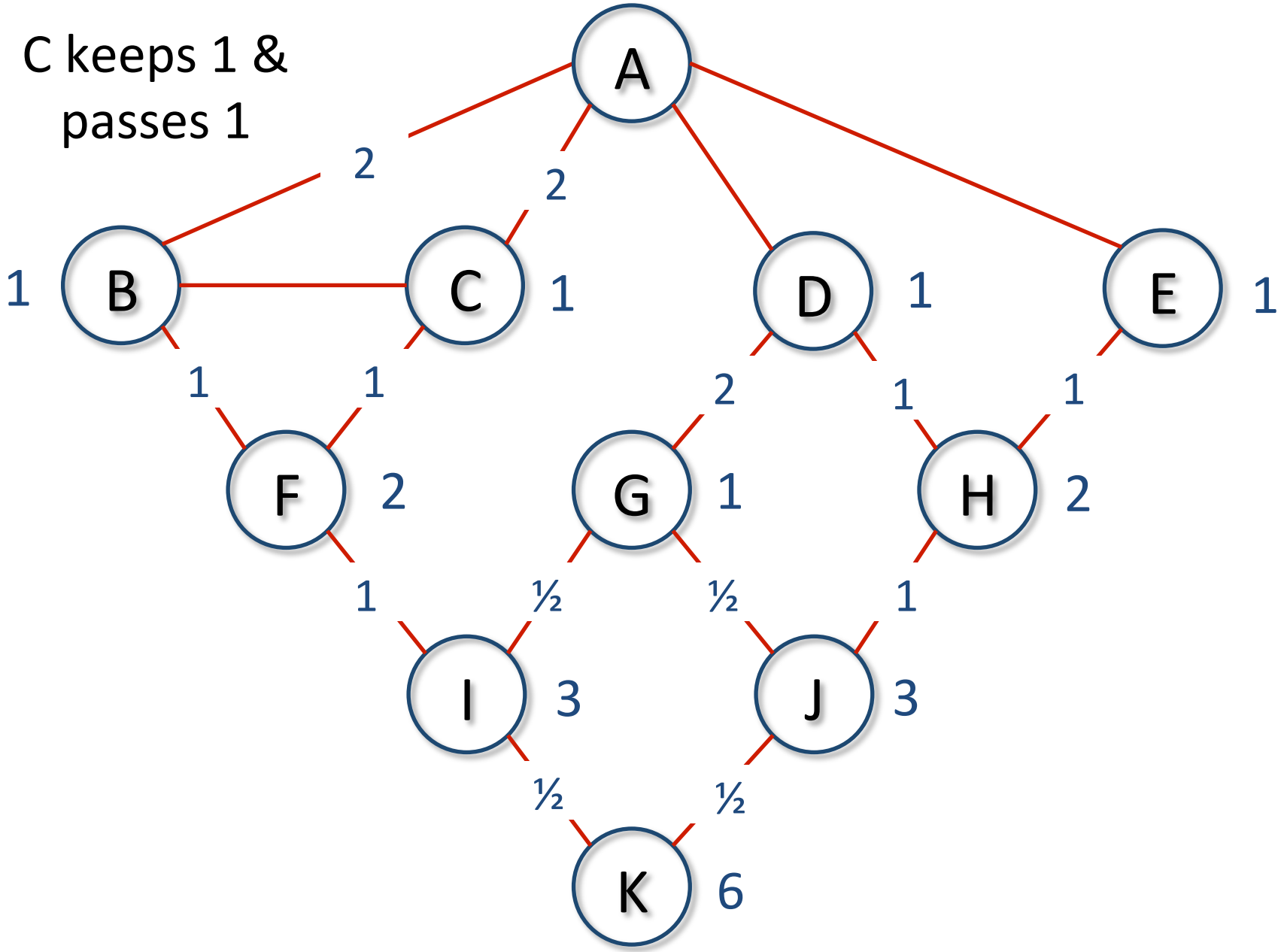
F keeps 1 unit & passes along 1 unit; equal, so divide evenly

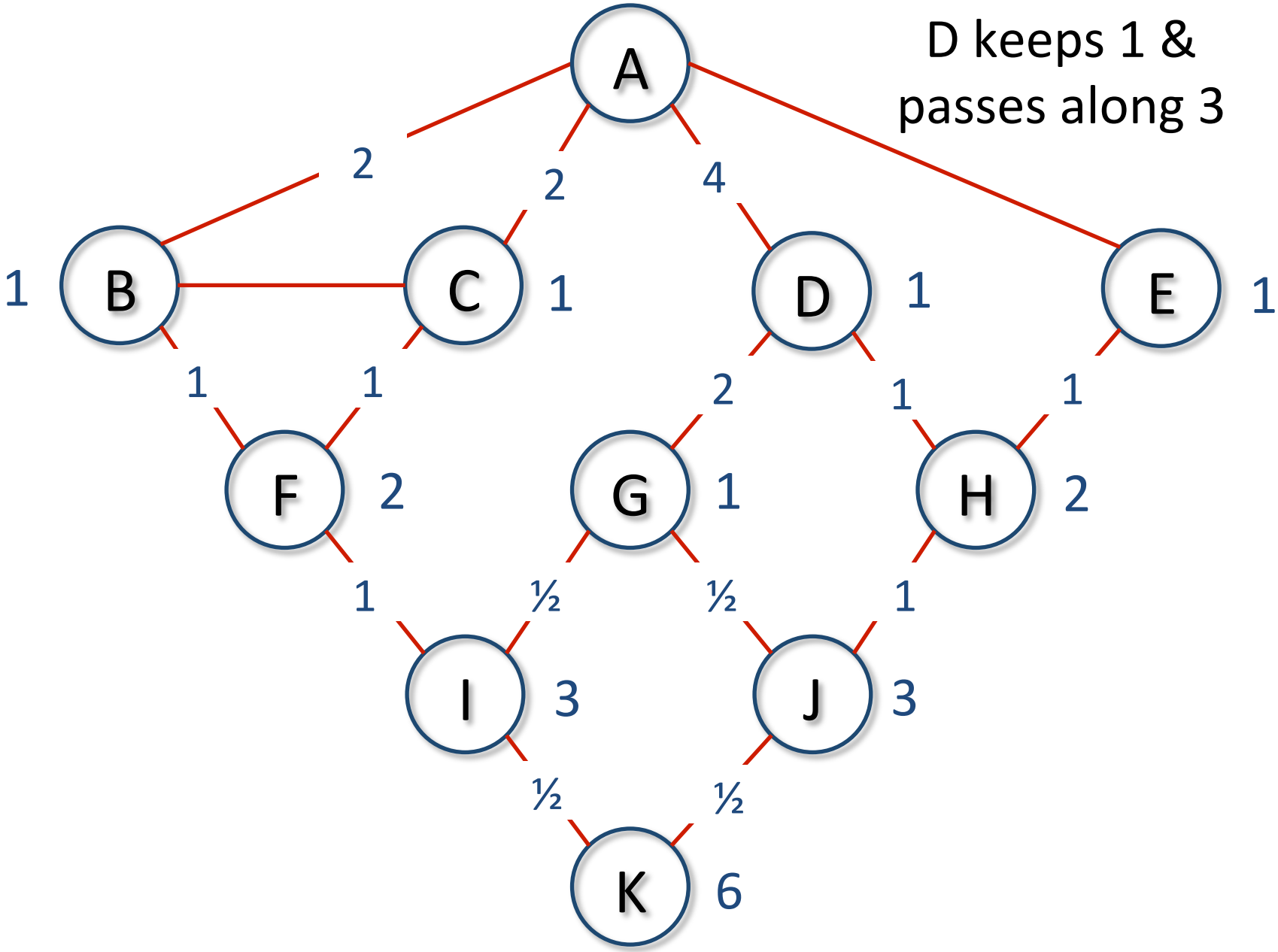

A
B
C
D
E
F
G
H
I
J
K
1
1
1
1
1
1
2
2
1
1
½
½
1
3
3
½
½
6
G keeps 1 unit & passes along 1 unit

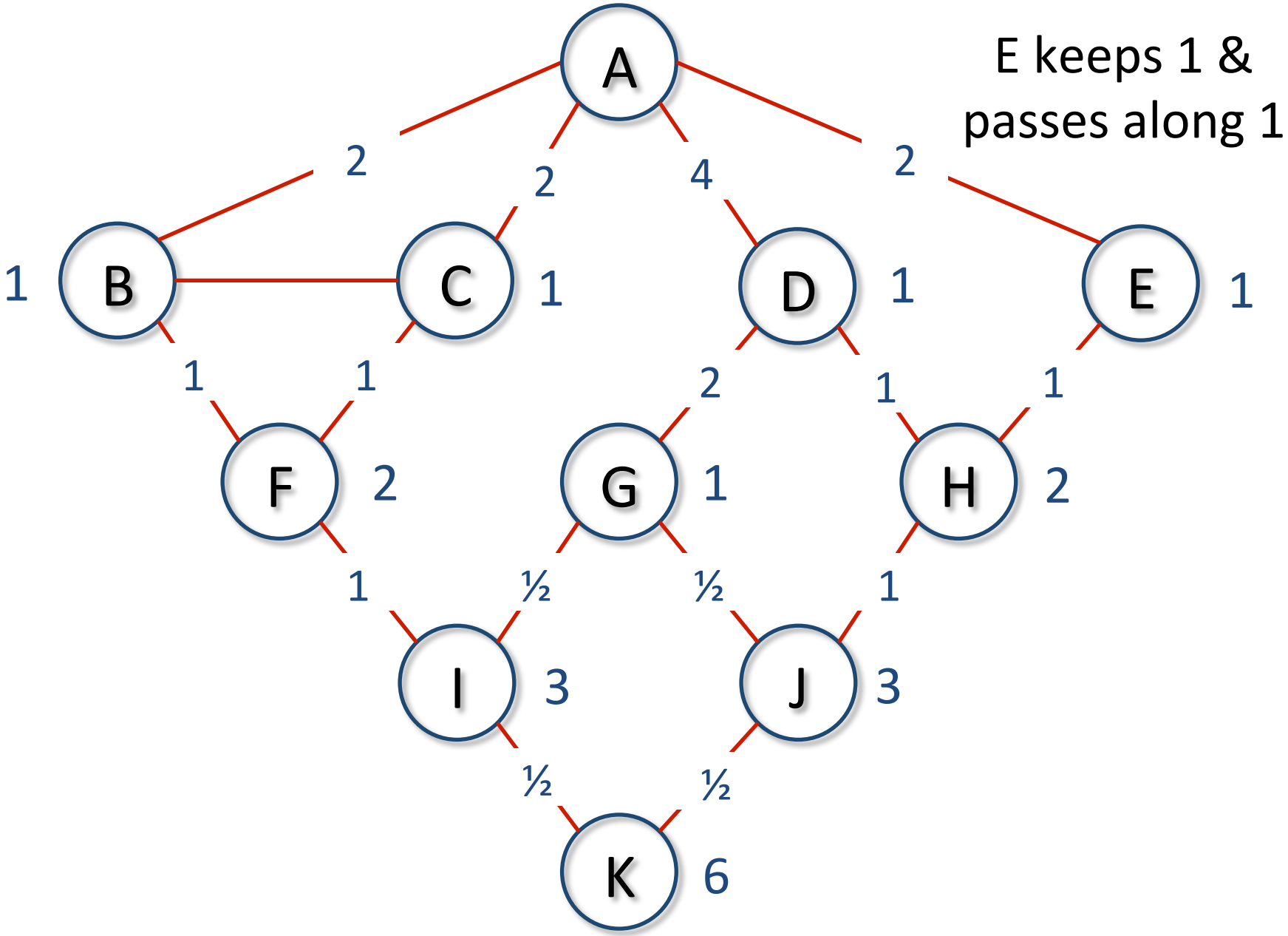


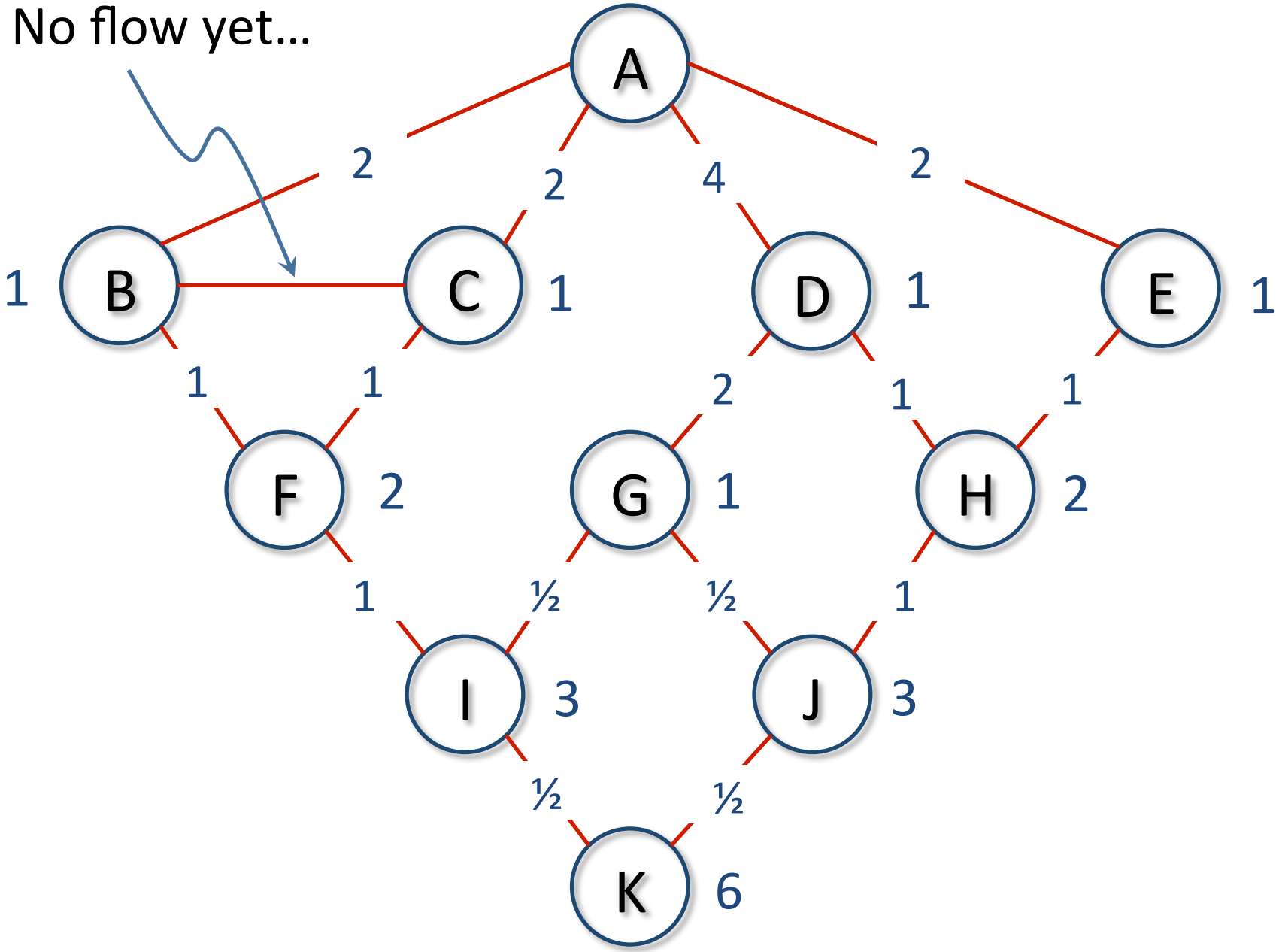
H keeps 1 unit & passes along 1 unit; equal, so divide evenly


B keeps 1 &
passes 1
A
2
1
B
C
1
D
1
E
1
1
1
2
1
1
F
2
G
1
H
2
1
½
½
1
I
3
J
3
½
½
K
6


C keeps 1 &
passes 1
A
B
C
D
E
F
G
H
I
J
K
2
2
1
1
1
1
1
1
2
1
1
2
1
½
½
1
3
3
½
½
6


A
B
C
D
E
F
G
H
I
J
K
D keeps 1 &
passes along 3
2
2
4
1
1
1
1
1
1
1
2
2
1
1
1
2
1
½
½
1
3
3
½
½
6


A
B
C
D
E
F
G
H
I
J
K
E keeps 1 &
passes along 1
2
2
4
2
1
1
1
1
1
1
2
1
1
2
1
1
½
½
1
3
3
½
½
6


No flow yet…
A
B
C
D
E
F
G
H
I
J
K
2
2
4
2
1
1
1
1
1
1
2
1
1
2
1
½
½
1
3
3
½
½
6

Computing Edge Betweenness Efficiently

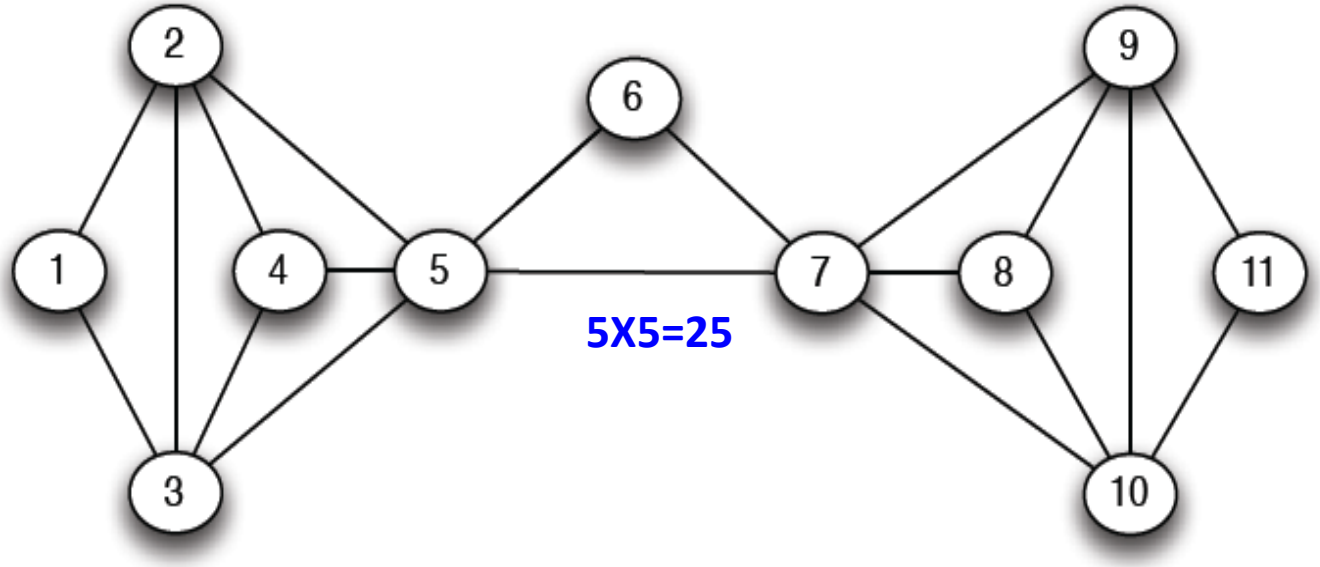
For each node *N* in the graph ← Repeat for B, C, etc.

1. Perform breadth-first search of graph starting at node *N*
2. Determine the number of shortest paths from *N* to every other node
3. Based on these numbers, determine the amount of flow from *N* to all other nodes that use each edge

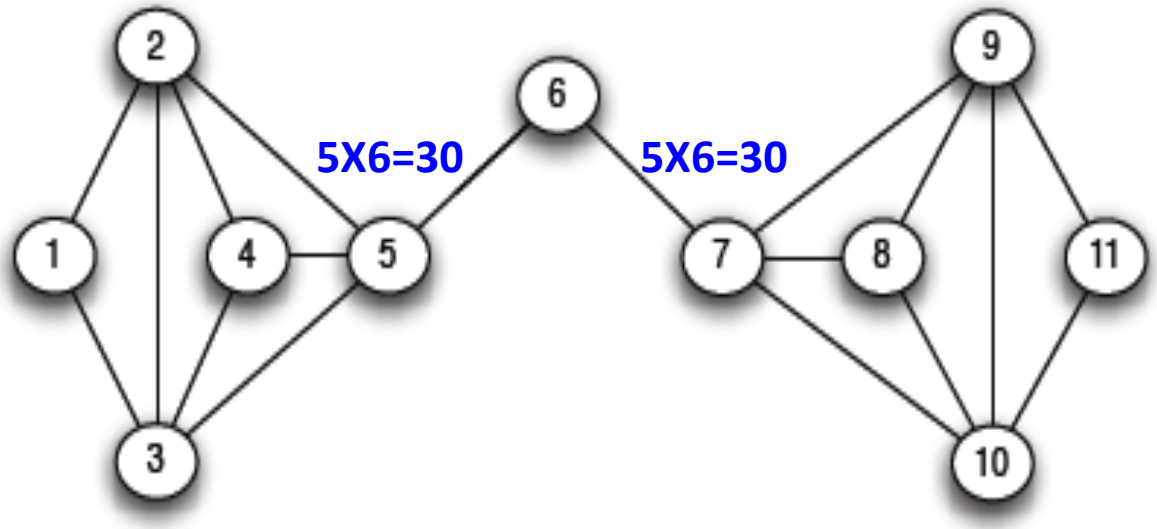
Divide sum of flow of all edges by 2

↶ Since sum includes flow from A → B and B → A, etc.

Another example

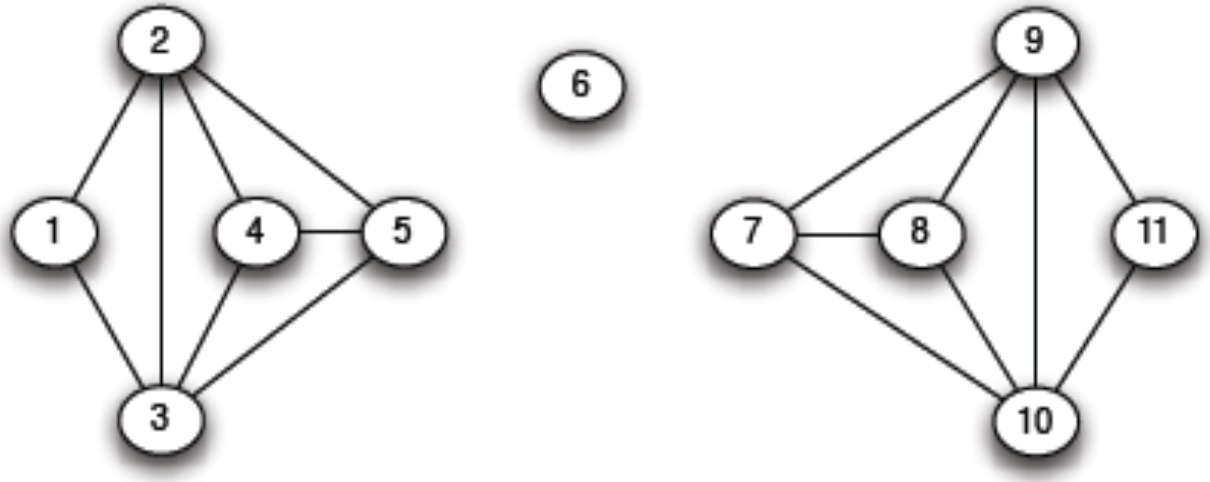


Another example



(a) *Step 1*

Another example



(b) *Step 2*

A Related Concept: Node Betweenness

- Betweenness also defined for nodes
- **Node betweenness**: Total amount of "flow" a node carries when a unit of flow between each pair of nodes is divided up evenly over shortest paths
- Nodes and edges of high betweenness perform critical roles in the network structure

Girvan-Newman Algorithm

- Edge Betweenness example - EA
  - D-E 0
  - D-A +1.0
  - D-C 0
  - D-B +0.5
  - E-A +1.0
  - E-C 0
  - E-B +0.5
  - … …