Social Media Analytics

Part III

Community detection

- Community: It is formed by individuals such that those within a group interact with each other **more frequently than with those outside the group**
  - a.k.a. group, cluster, cohesive subgroup, module in different contexts
- Community detection: discovering groups in a network where individuals' group memberships are not explicitly given

Community detection

- Why communities in social media?
  - Human beings are social
  - Easy-to-use social media allows people to extend their social life in unprecedented ways
  - Difficult to meet friends in the physical world, but much easier to find friend online **with similar interests**
  - Interactions between nodes can help determine communities

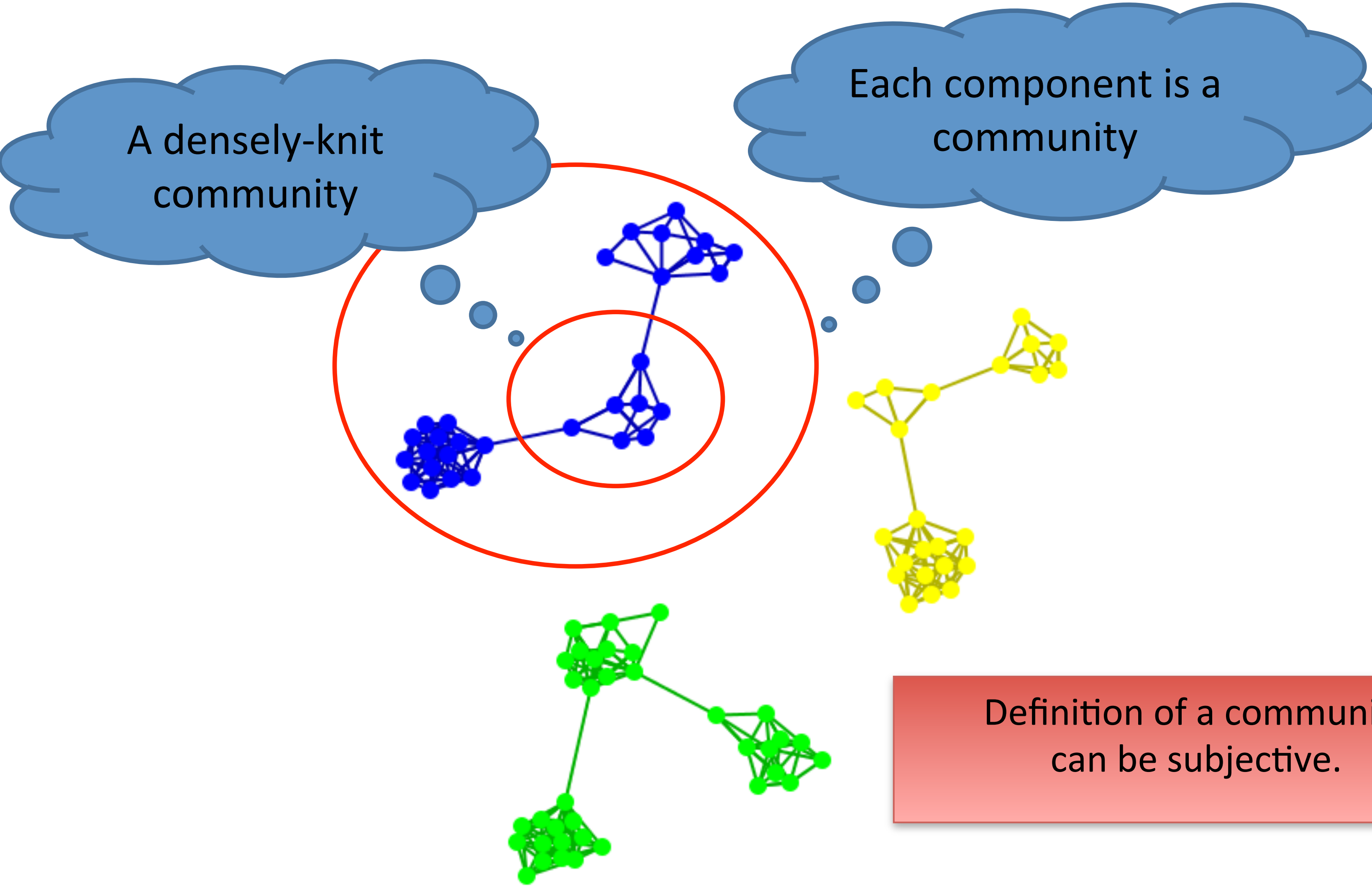
Communities in Social Media

- Two types of groups in social media
  - Explicit Groups: formed by user subscriptions (e.g. Google groups, Twitter lists)
  - Implicit Groups: implicitly formed by social interactions

- Some social media sites allow people to join groups, however it is still necessary to extract groups based on network topology
  - Not all sites provide community platform
  - Not all people want to make effort to join groups
  - Groups can change dynamically

- Network interaction provides rich information about the relationship between users
  - Can complement other kinds of information, e.g. user profile
  - Help network visualization and navigation
  - Provide basic information for other tasks, e.g. **recommendation**

Subjectivity of Community Definition

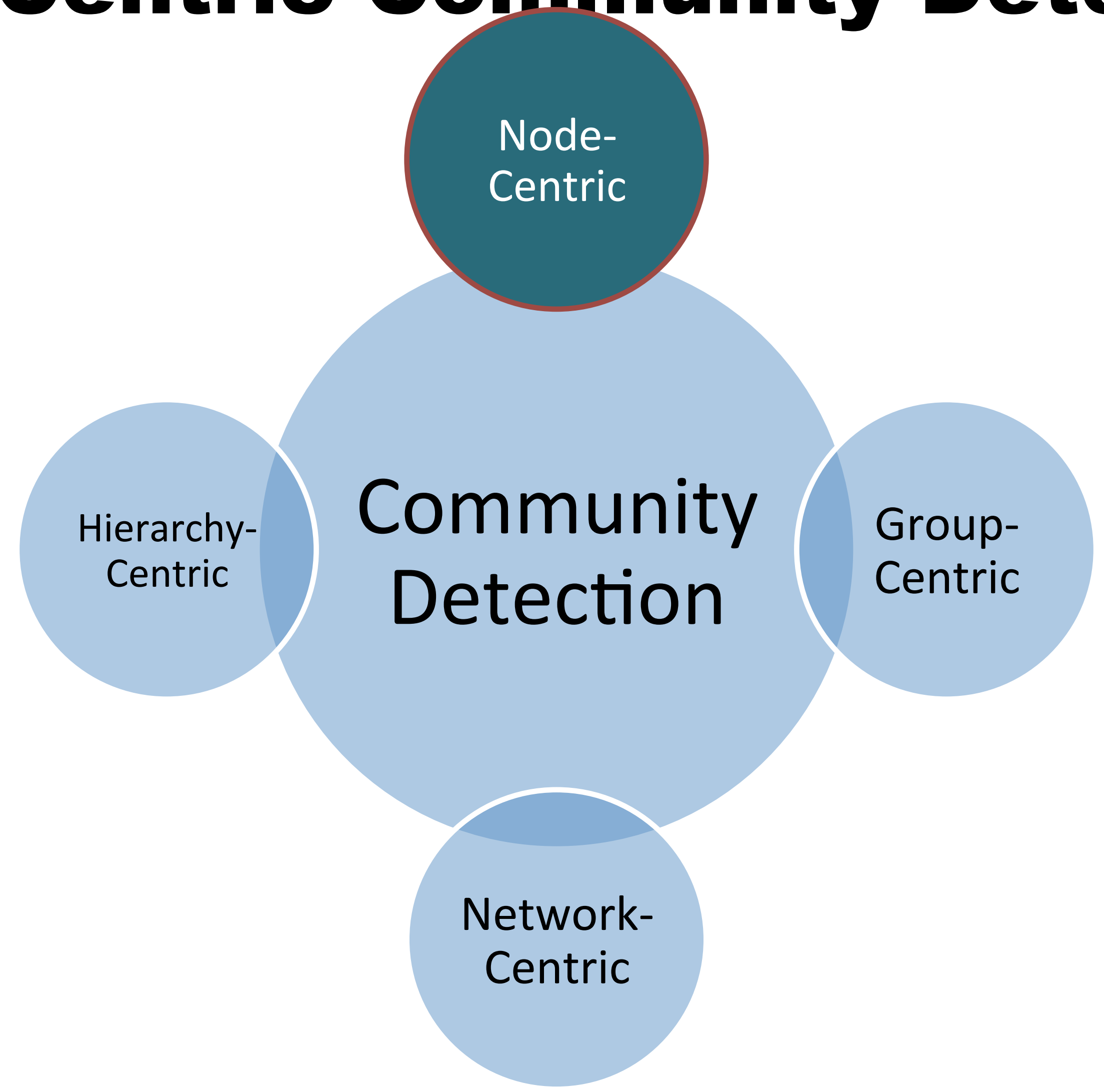


Definition of a community can be subjective.

Taxonomy of Community Detection Criteria

- Criteria vary depending on the tasks
- Roughly, community detection methods can be divided into 4 categories (not exclusive):
  - Node-Centric Community
  - Each node in a group satisfies certain properties
  - Group-Centric Community
  - Consider the connections within a group as a whole. The group has to satisfy certain properties without zooming into node-level
  - **Network-Centric Community**
  - Partition the whole network into several disjoint sets
  - Hierarchy-Centric Community
  - Construct a hierarchical structure of communities

Node-Centric Community Detection

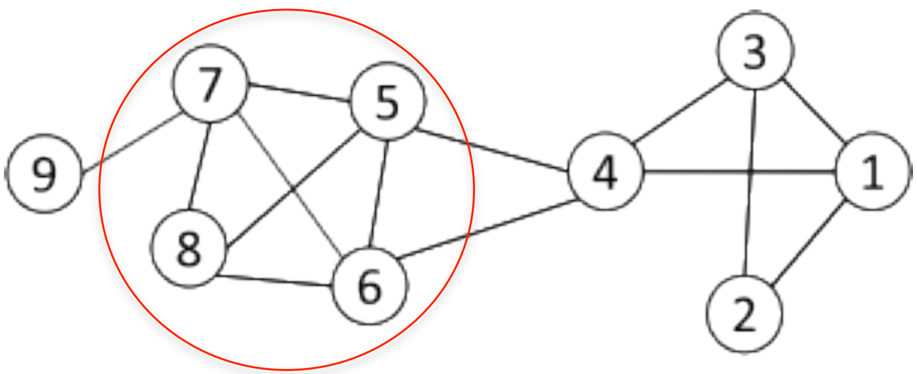


1. Node-Centric Community Detection

- Nodes in a community must satisfy specific properties, like:
  - Complete Mutuality
    - cliques
  - Reachability of members
    - k-clique, k-clan, k-club
  - Nodal degrees
    - k-plex, k-core
  - Relative frequency of Within-Outside Ties
    - LS sets, Lambda sets
- Commonly used in traditional social network analysis
- Here, we discuss only some of these properties

Complete Mutuality: Cliques

- Clique: a maximum complete subgraph in which all nodes are adjacent to each other



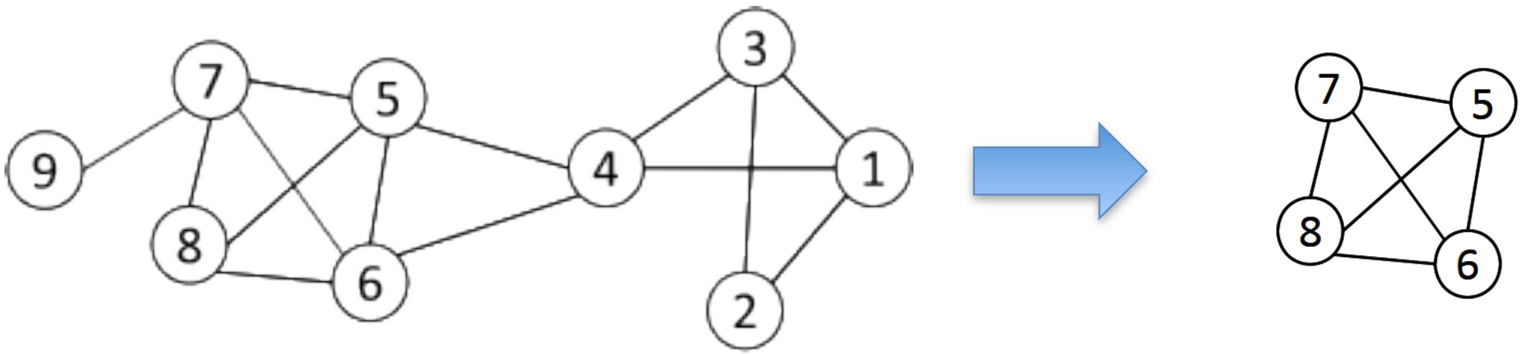
Nodes 5, 6, 7 and 8 form a clique

- **NP-hard** to find the maximum clique in a network
- Straightforward implementation to find cliques is very expensive in time complexity

Finding the Maximum Clique

- In a clique of size k, each node maintains degree >= k-1
  - Nodes with degree < k-1 will not be included in the maximum clique
- Recursively apply the following pruning procedure
  - Sample a sub-network from the given network, and find a clique in the sub-network, say, by a greedy approach
  - Suppose the clique above is size k, in order to find out a *larger* clique, **all nodes with degree <= k-1 should be removed**.
- Repeat until the network is small enough
- Many nodes will be pruned as social media networks follow a power law distribution for node degrees (Zipfian low, previous lessons)

Maximum Clique Example



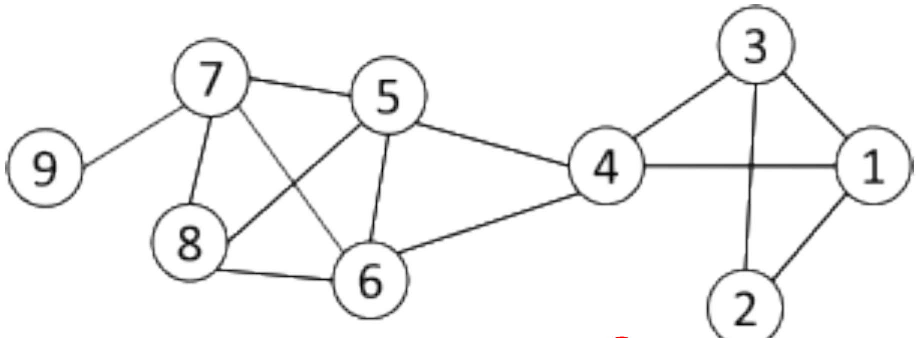
- Suppose we sample a sub-network with nodes {1-9} and find a clique {1, 2, 3} of size 3
- In order to find a clique >3, remove all nodes with degree <=3-1=2
  - Remove nodes 2 and 9
  - Remove nodes 1 and 3
  - Remove node 4

Clique Percolation Method (CPM)

- Clique is a very strict definition, unstable
- Normally use cliques as a core or a seed to find larger communities

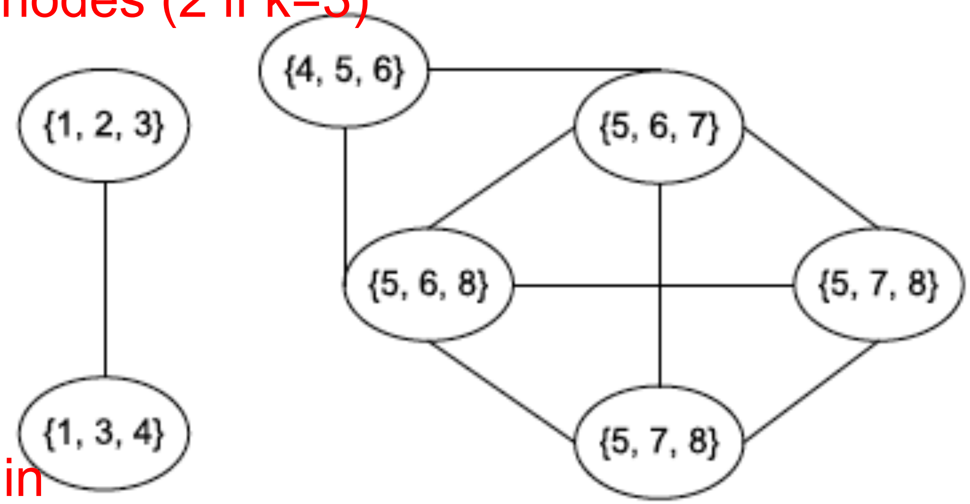
- CPM is such a method to find overlapping communities
  - **Input**
    - A parameter k, and a network
  - Procedure
    - Find out all cliques of size k in a given network
    - Construct a clique graph. Two cliques are adjacent if they share k-1 nodes
    - Each connected components in the clique graph form a community

CPM Example



Cliques of size 3:
{1, 2, 3}, {1, 3, 4}, {4, 5, 6}, {5, 6, 7}, {5, 6, 8}, {5, 7, 8}, {6, 7, 8}

Construct a clique graph. Two cliques are adjacent if they share k-1 nodes (2 if k=3)

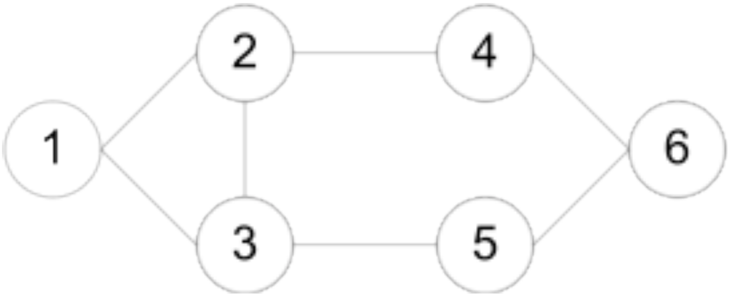


Communities:
{1, 2, 3, 4}
{4, 5, 6, 7, 8}

Each connected component in the clique graph forms a community

Reachability : k-clique, k-club

- Any node in a group should be reachable in k hops
- k-clique: a maximal subgraph in which the largest geodesic distance between any two nodes <= k
- k-club: a substructure of diameter <= k



Cliques: {1, 2, 3}

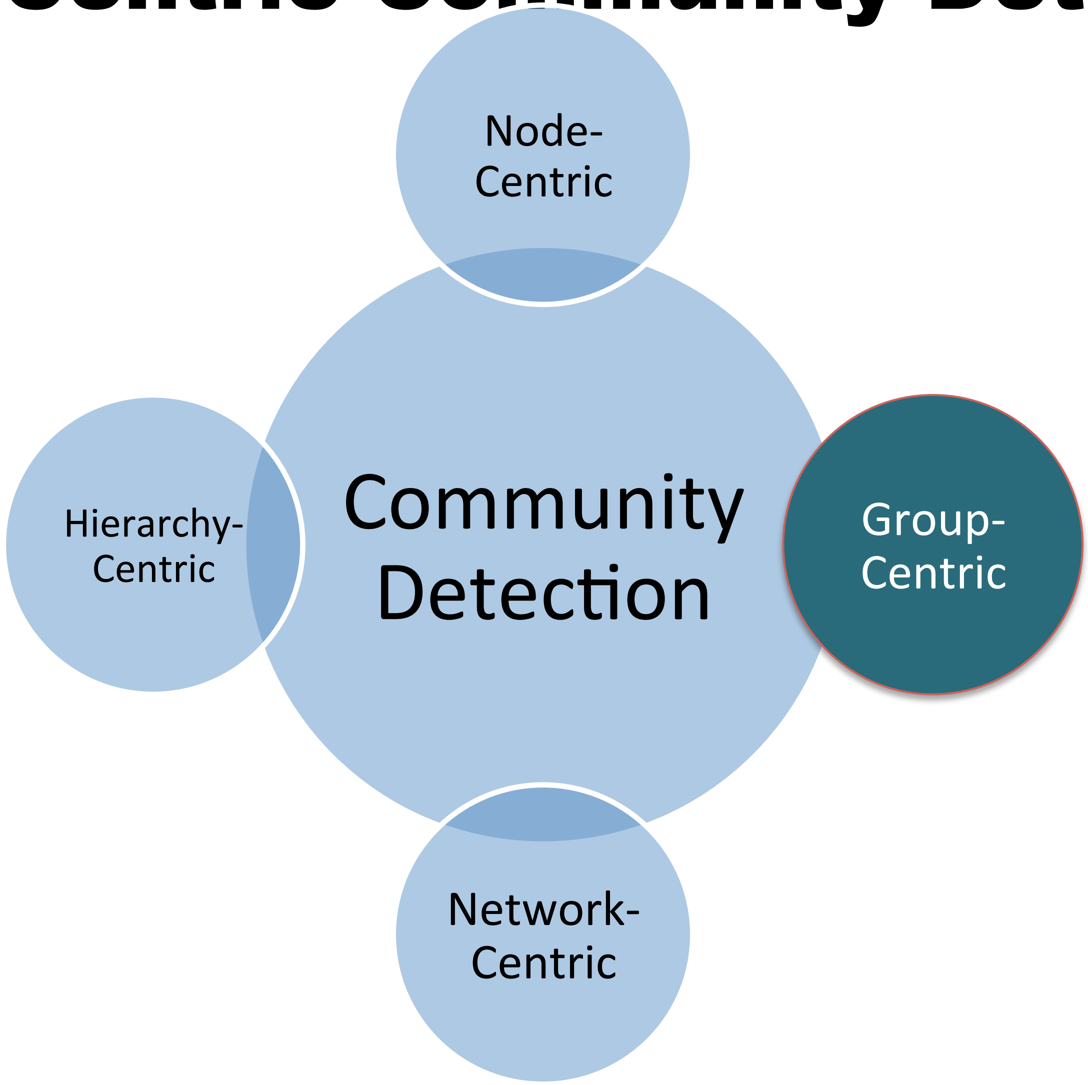
2-cliques: {1, 2, 3, 4,5}, {2, 3, 4, 5, 6}

2-clubs: {1,2,3,4}, {1, 2, 3, 5}, {2, 3, 4, 5, 6}

- A k-clique might have diameter larger than k in the subgraph
  - E.g. {1, 2, 3, 4, 5}
  - Commonly used in traditional SNA
- Often involves combinatorial optimization

Note that the path of length *k* or less linking a member of the k-clique to another member may pass through an intermediary who is not in the group (e.g. for nodes 4 and 5).

Group-Centric Community Detection



2. Group-Centric Community Detection: Density-Based Groups

- The group-centric criterion requires the **whole group** to satisfy a certain condition
  - E.g., the group density >= of a given threshold
- A subgraph $G_s(V_s, E_s)$ is a $\gamma - dense$ quasi-clique if

$$\frac{2|E_s|}{|V_s|(|V_s| - 1)} \geq \gamma$$

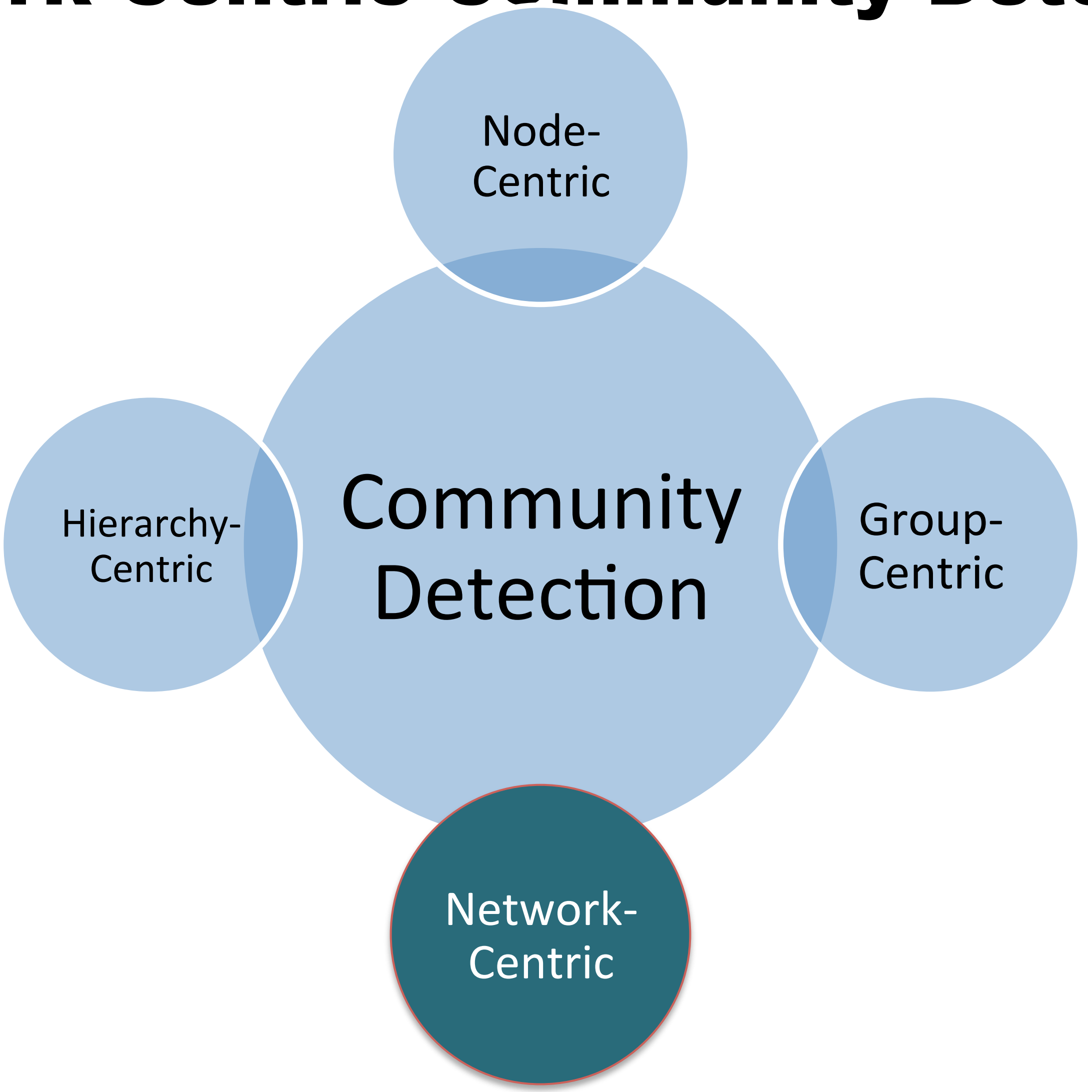
where the denominator is the maximum possible node degree (any node connected to any node).

- To detect quasi-cliques we can use a strategy similar to that of cliques
  - Sample a subgraph, and find a maximal $\gamma - dense$ quasi-clique (say, of size $|V_s|$)
  - Remove nodes with degree <u>less than</u> the average degree

$$|V_s|\gamma \leq \frac{2|E_s|}{|V_s| - 1}$$

  - iterate

Network-Centric Community Detection



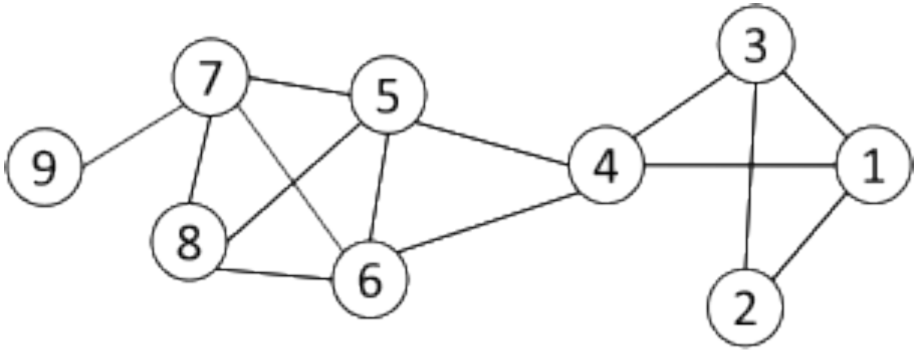
3. Network-Centric Community Detection

- Network-centric criterion needs to consider the connections within a network <u>globally</u>
- Goal: partition nodes of a network into <u>disjoint</u> sets such that members (i,j) of a set are more similar to each other than any to members (i,j) such that i belongs to a set and j to a different set.
- Many approaches to identify such sets, or CLUSTERS:
  - (1) Clustering based on vertex similarity
  - (2) Latent space models (multi-dimensional scaling)
  - (3) Block model approximation
  - (4) Spectral clustering
  - (5) Modularity maximization

Clustering based on Vertex Similarity

- Define a measure of vertex similarity
- Use an algorithm to group nodes based on similarity (e.g. **k-means**, see later)
- Vertex similarity is defined in terms of the similarity of their neighborhood
- Example of similarity measure: Structural equivalence
- Two nodes are structurally equivalent iff they are connecting to the same set of actors

Nodes 1 and 3 are structurally equivalent; So are nodes 5 and 6.



- Structural equivalence is too restricted for practical use.

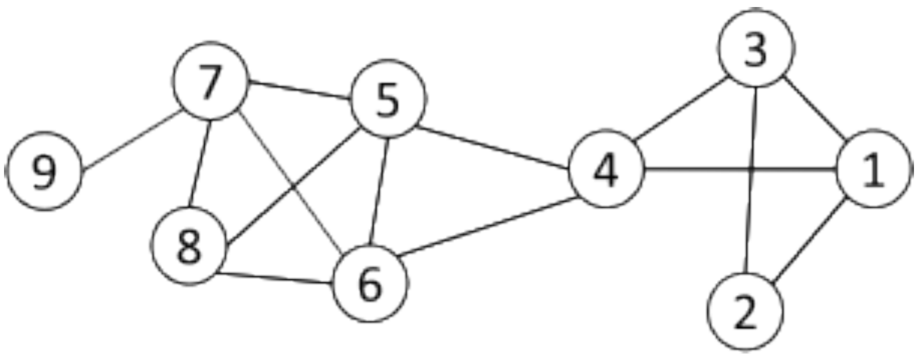
Clustering based on Vertex Similarity

- Jaccard Similarity

$$Jaccard(v_i, v_j) = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$$

- Cosine similarity

$$Cosine(v_i, v_j) = \frac{|N_i \cap N_j|}{\sqrt{|N_i| \cdot |N_j|}}$$



$$Jaccard(4, 6) = \frac{|\{5\}|}{|\{1, 3, 4, 5, 6, 7, 8\}|} = \frac{1}{7}$$

$$cosine(4, 6) = \frac{1}{\sqrt{4 \cdot 4}} = \frac{1}{4}$$

Clustering based on Vertex Similarity

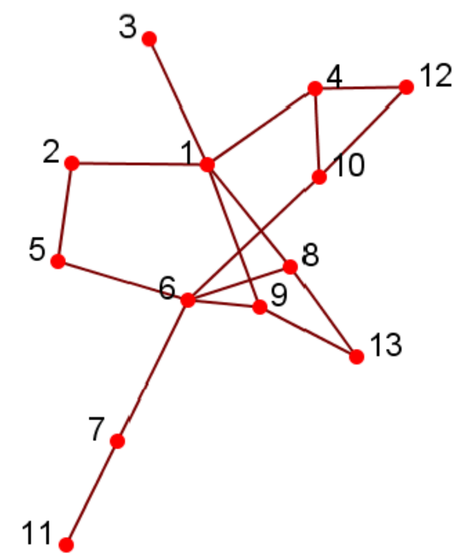
		1	2	3	4	5	6	7	8	9	10	11	12	13
a vector	5		1				1							
structurally equivalent	8	1					1							1
	9	1					1							1

Cosine Similarity: $\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}.$

$$sim(5,8) = \frac{1}{\sqrt{2} \times \sqrt{3}} = \frac{1}{\sqrt{6}}$$

Jaccard Similarity: $J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$

$$J(5,8) = \frac{|\{6\}|}{|\{1,2,6,13\}|} = 1/4$$



Clustering based on vertex similarity (K-means)

Given some similarity function (e.g. Jaccard)

K-Means Clustering:

1) Pick K objects as centers of K clusters and assign all the remaining objects to these centers
   - Each object will be assigned to the center that has minimal distance to it (distance= inverse of similarity)
   - Solve any ties randomly (if distance is the same, assign randomly)
2) In each cluster C, find a new center X_C so as to minimize the total sum of distances between X_C and all other elements in C
3) Reassign all elements to new centers as explained in step (1)
4) Repeat the previous two steps until the algorithm converges (clusters stay the same)

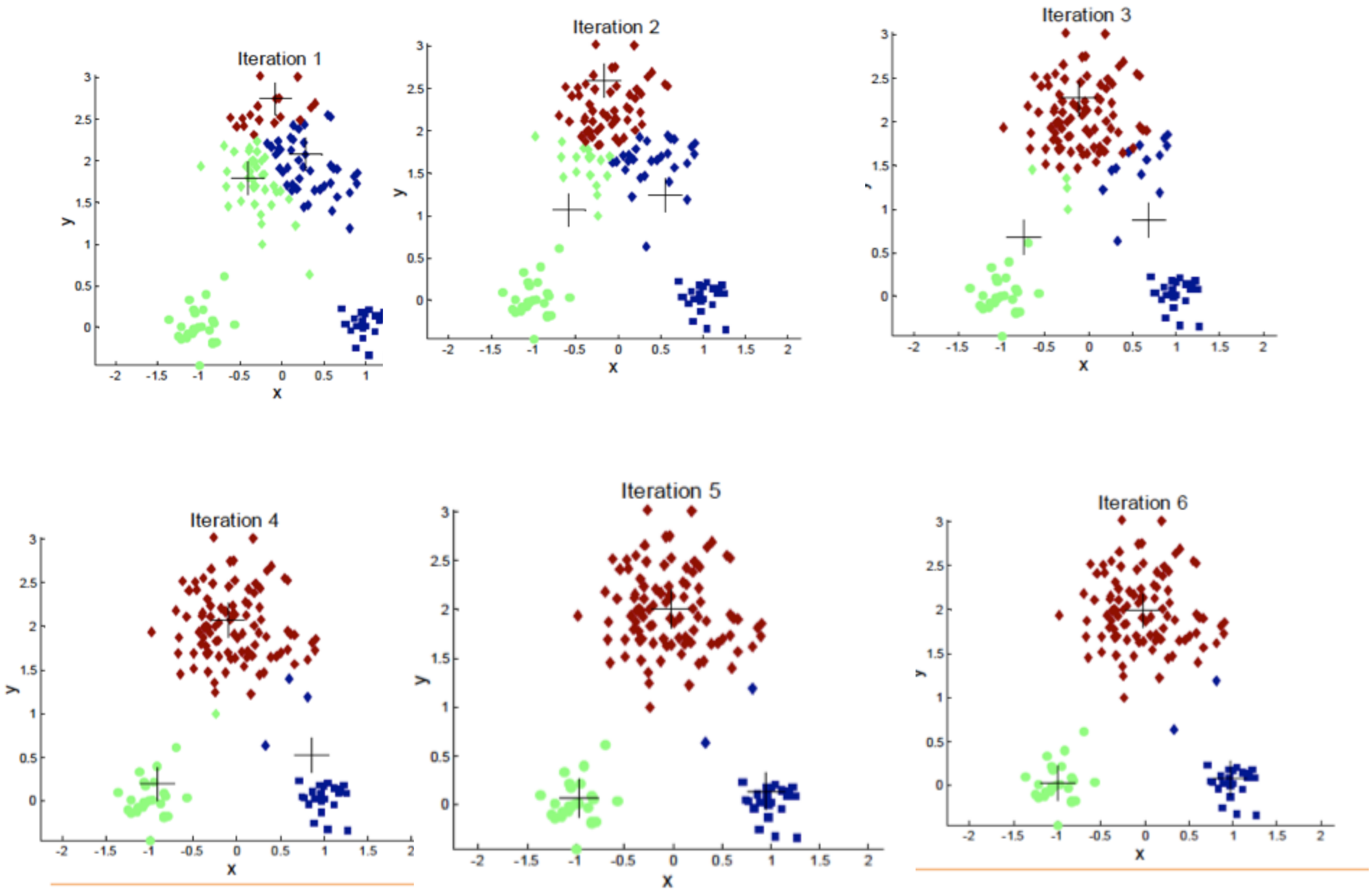
Clustering based on vertex similarity (K-means)

Algorithm 1 Basic K-means Algorithm.

Select K points as the initial centroids.
repeat
    Form K clusters by assigning all points to the closest centroid.
    Recompute the centroid of each cluster.
until The centroids don't change

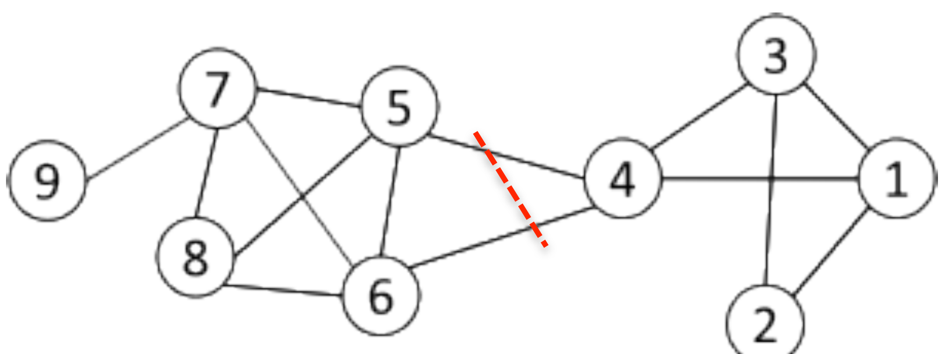
- Each cluster is associated with a centroid
- Each node is assigned to the cluster with the closest centroid

Illustration of k-means clustering



Clustering based on Min Cut

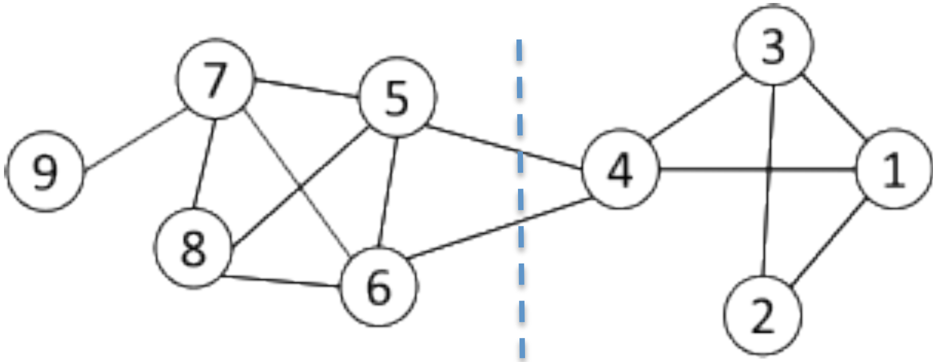
- Target: find clusters such that most interactions (edges) are within groups whereas interactions between members of different groups are fewer
- community detection → minimum cut problem
- Cut: A partition of vertices of a graph into two disjoint sets
- Minimum cut problem: find a graph partition such that the number of edges between the two sets is minimized
- (**http://en.wikipedia.org/wiki/Max-flow_min-cut_theorem**)

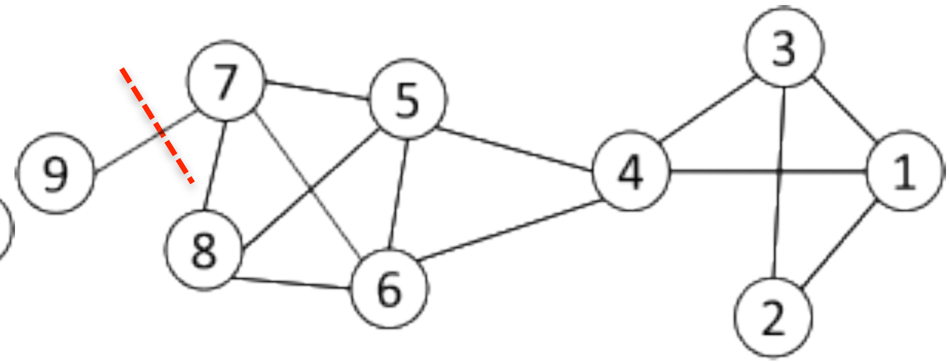


Cut Example

Cut: set of edges whose removal disconnects G

Min-Cut: a cut in G of minimum cost



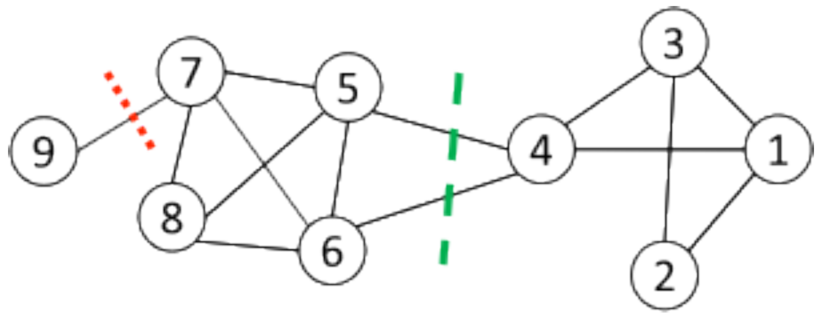


$$\min imize: \; cut(C_i, \overline{C}_i) = \sum_{i \in C, j \in \overline{C}_i} (i,j); \;\; where \;\; (i,j) = 1 \; if \;\; i \rightarrow j$$

Weight of this cut: 2

Weight of min cut: 1

Ratio Cut & Normalized Cut



- Minimum cut often returns an imbalanced partition, with one set being a singleton, e.g. node 9
- Change the objective function to consider community size (above formulas apply to a k-partition):

$$\text{Ratio Cut}(\pi) = \frac{1}{k} \sum_{i=1}^{k} \frac{cut(C_i, \bar{C}_i)}{|C_i|},$$

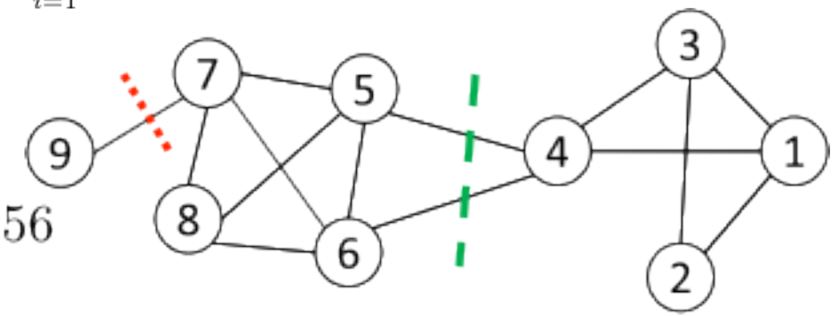
$$\text{Normalized Cut}(\pi) = \frac{1}{k} \sum_{i=1}^{k} \frac{cut(C_i, \bar{C}_i)}{vol(C_i)}$$

C_i: a community
$\bar{C}_i$: the remaining graph
$|C_i|$: number of nodes in C_i
$vol(C_i)$: sum of degrees in C_i

Typically, graph partition problems fall under the category of NP-hard problems. Practical solutions based on heuristics

Ratio Cut & Normalized Cut Example

$$\text{Ratio Cut}(\pi) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i,\bar{C}_i)}{|C_i|}, \quad \text{Normalized Cut}(\pi) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i,\bar{C}_i)}{vol(C_i)}$$



For partition in red: π_1

$$\text{Ratio Cut}(\pi_1) = \frac{1}{2}\left(\frac{1}{1}+\frac{1}{8}\right) = 9/16 = 0.56$$

$$\text{Normalized Cut}(\pi_1) = \frac{1}{2}\left(\frac{1}{1}+\frac{1}{27}\right) = 14/27 = 0.52$$

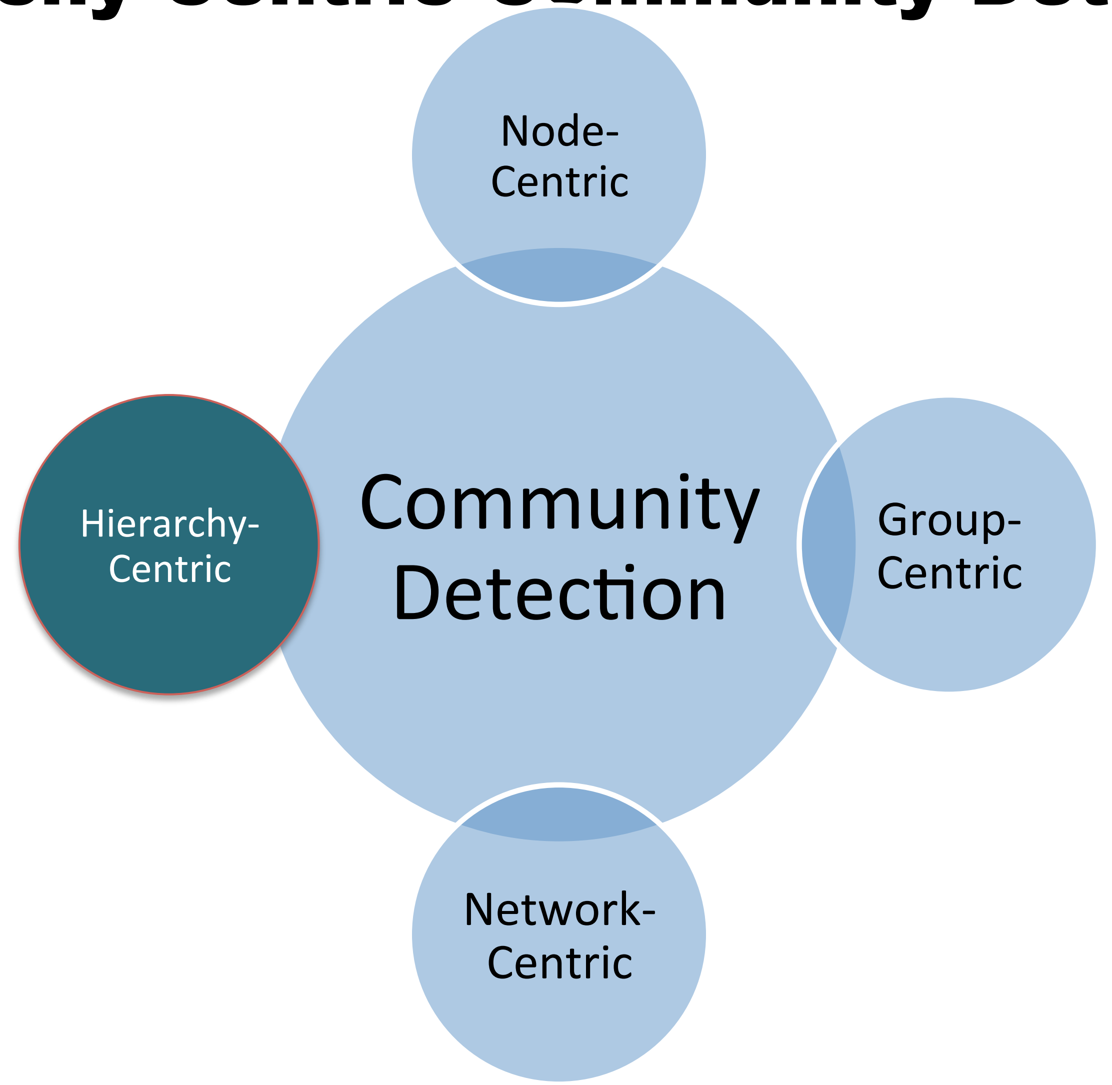
For partition in green: π_2

$$\text{Ratio Cut}(\pi_2) = \frac{1}{2}\left(\frac{2}{4}+\frac{2}{5}\right) = 9/20 = 0.45 < \text{Ratio Cut}(\pi_1)$$

$$\text{Normalized Cut}(\pi_2) = \frac{1}{2}\left(\frac{2}{12}+\frac{2}{16}\right) = 7/48 = 0.15 < \text{Normalized Cut}(\pi_1)$$

Both ratio cut and normalized cut prefer a <u>balanced</u> partition

Hierarchy-Centric Community Detection



4. Hierarchy-Centric Community Detection

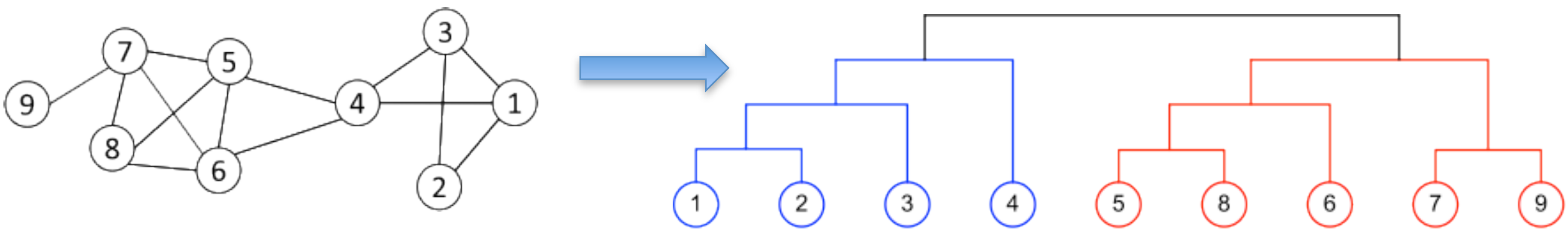
- Goal: build a hierarchical structure of communities based on network topology

- Allow the analysis of a network at different resolutions

- Representative approaches:
  - Divisive Hierarchical Clustering (top-down)
  - Agglomerative Hierarchical clustering (bottom-up)

Agglomerative Hierarchical Clustering

- Initialize each node as a community (singleton clusters)
- Merge communities successively into larger communities following a certain criterion
  - E.g., based on vertex similarity

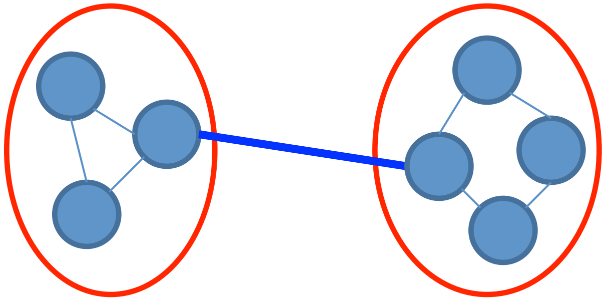


Dendrogram according to Agglomerative Clustering

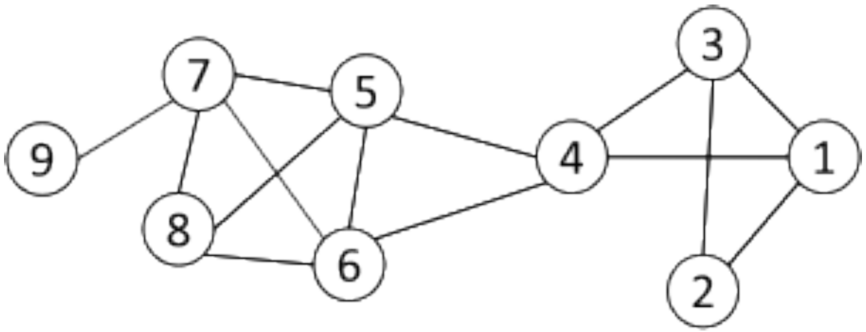
Divisive Hierarchical Clustering

- Divisive clustering
  - Partition nodes into several sets
  - Each set is further divided into smaller ones
  - Network-centric partition can be applied for the partition
- One particular example: recursively remove the "weakest" edge
  - Find the edge with the least strength
  - Remove the edge and update the corresponding strength of each edge (according to some measure of strength)
- Recursively apply the above two steps until a network is decomposed into desired number of connected components.
- Each component forms a community

Divisive clustering based on Edge Betweenness



- The strength of an edge can be measured by edge betweenness
- (remember) Edge betweenness: the number of shortest paths that pass along with the edge

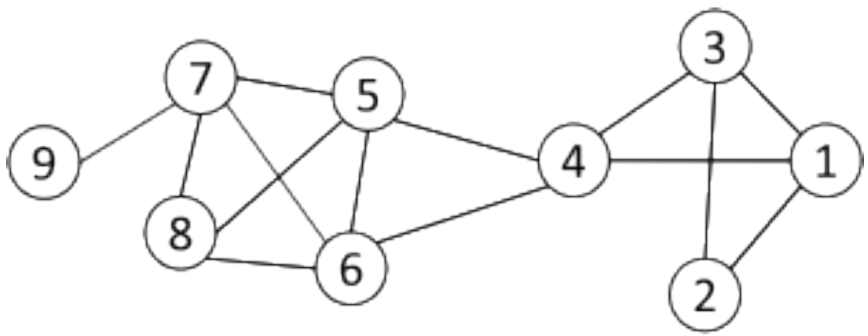


- The edges with higher betweenness tends to be the bridge between two communities.

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

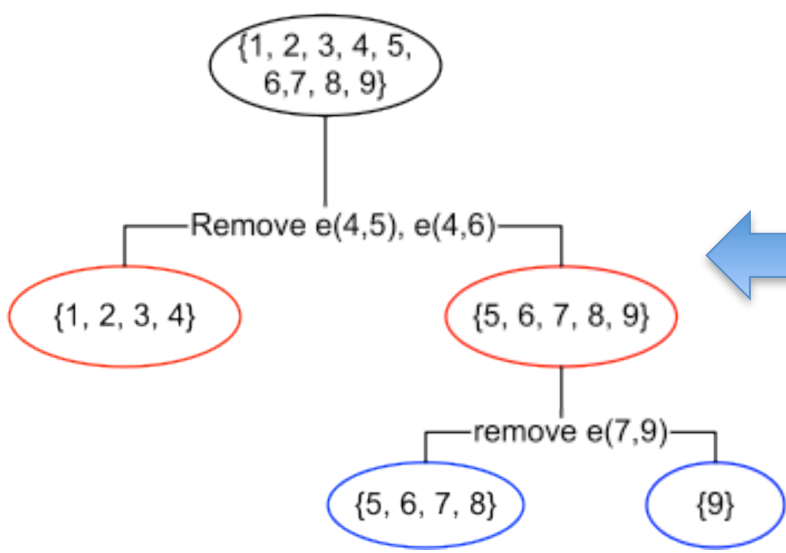
Divisive clustering based on edge betweenness



Initial betweenness value

Table 3.3: Edge Betweenness

	1	2	3	4	5	6	7	8	9
1	0	4	1	9	0	0	0	0	0
2	4	0	4	0	0	0	0	0	0
3	1	4	0	9	0	0	0	0	0
4	9	0	9	0	10	10	0	0	0
5	0	0	0	10	0	1	6	3	0
6	0	0	0	10	1	0	6	3	0
7	0	0	0	0	6	6	0	2	8
8	0	0	0	0	3	3	2	0	0
9	0	0	0	0	0	0	8	0	0

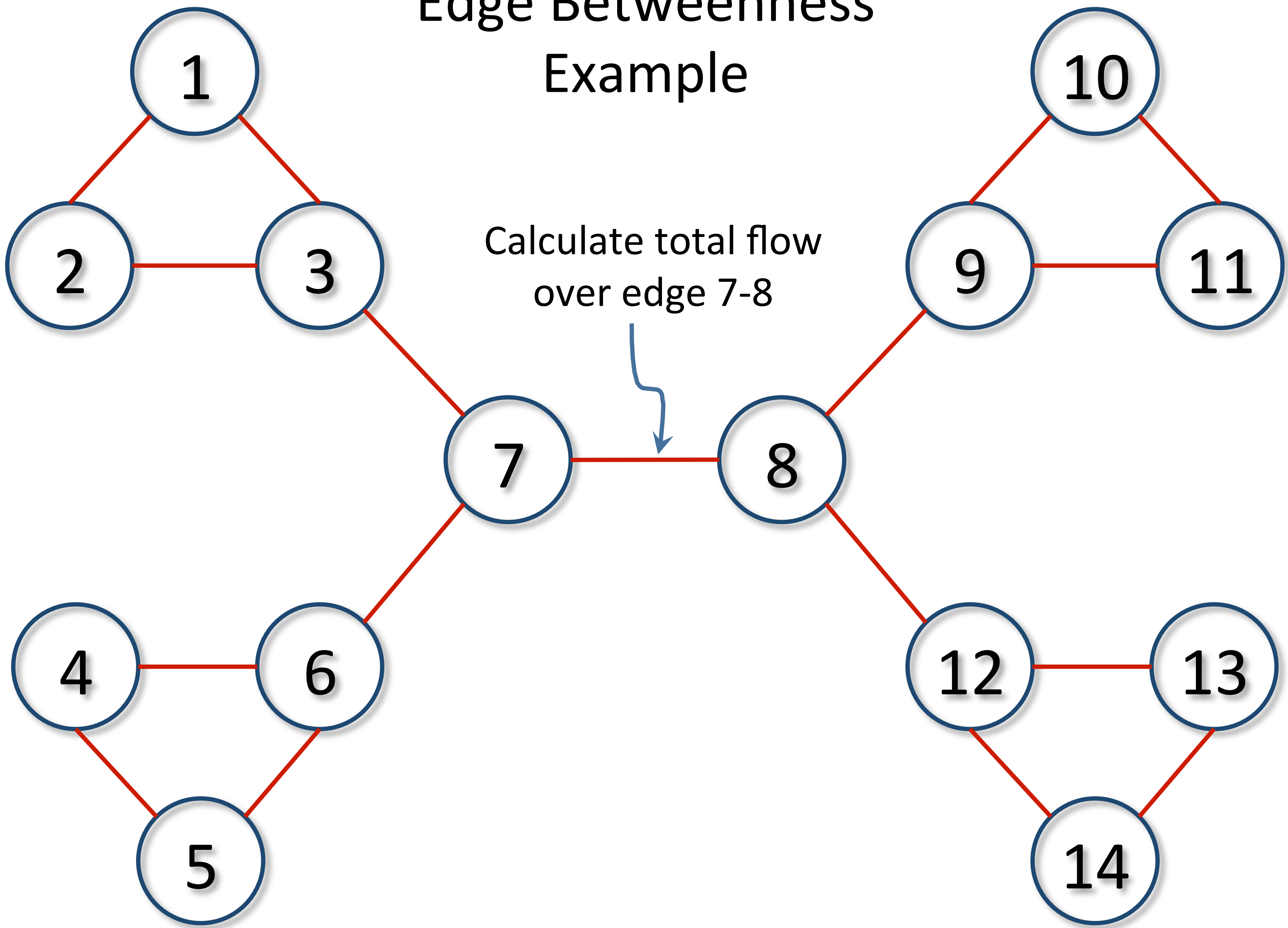


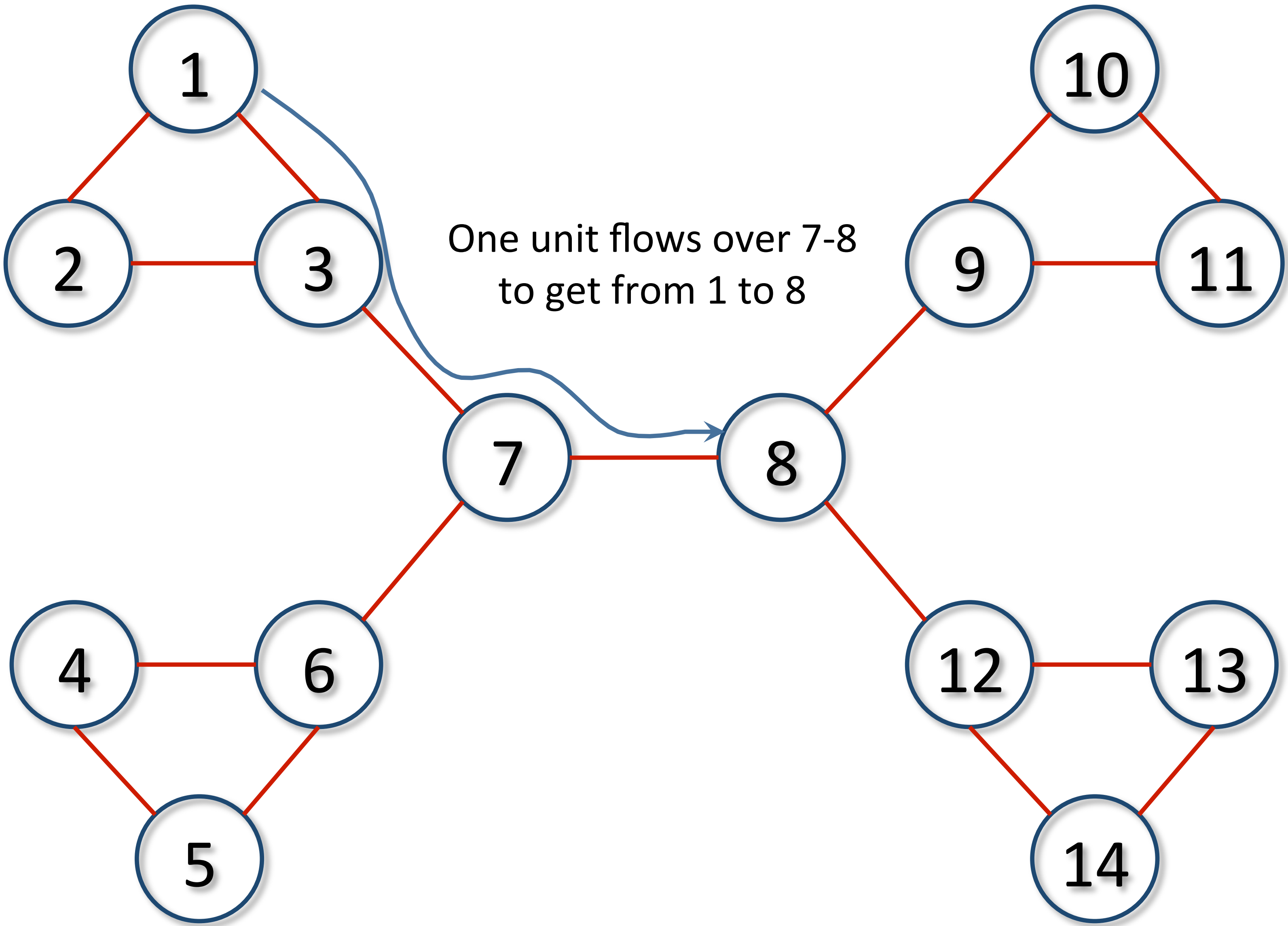
After removing e(4,5), the betweenness of e(4, 6) becomes 20, which is the highest;

After removing e(4,6), the edge e(7,9) has the highest betweenness value 4, and should be removed.

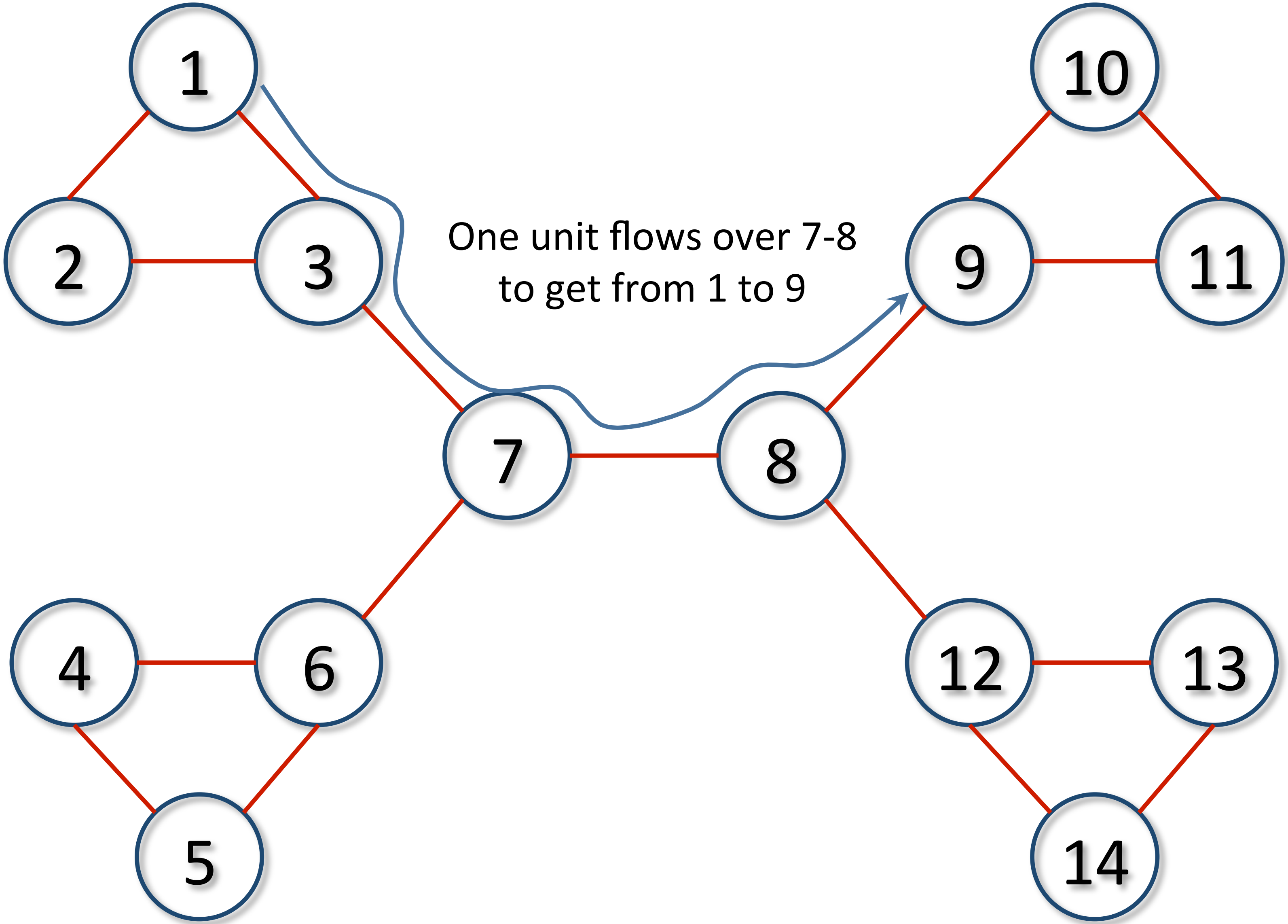
Idea: progressively removing edges with the highest betweenness

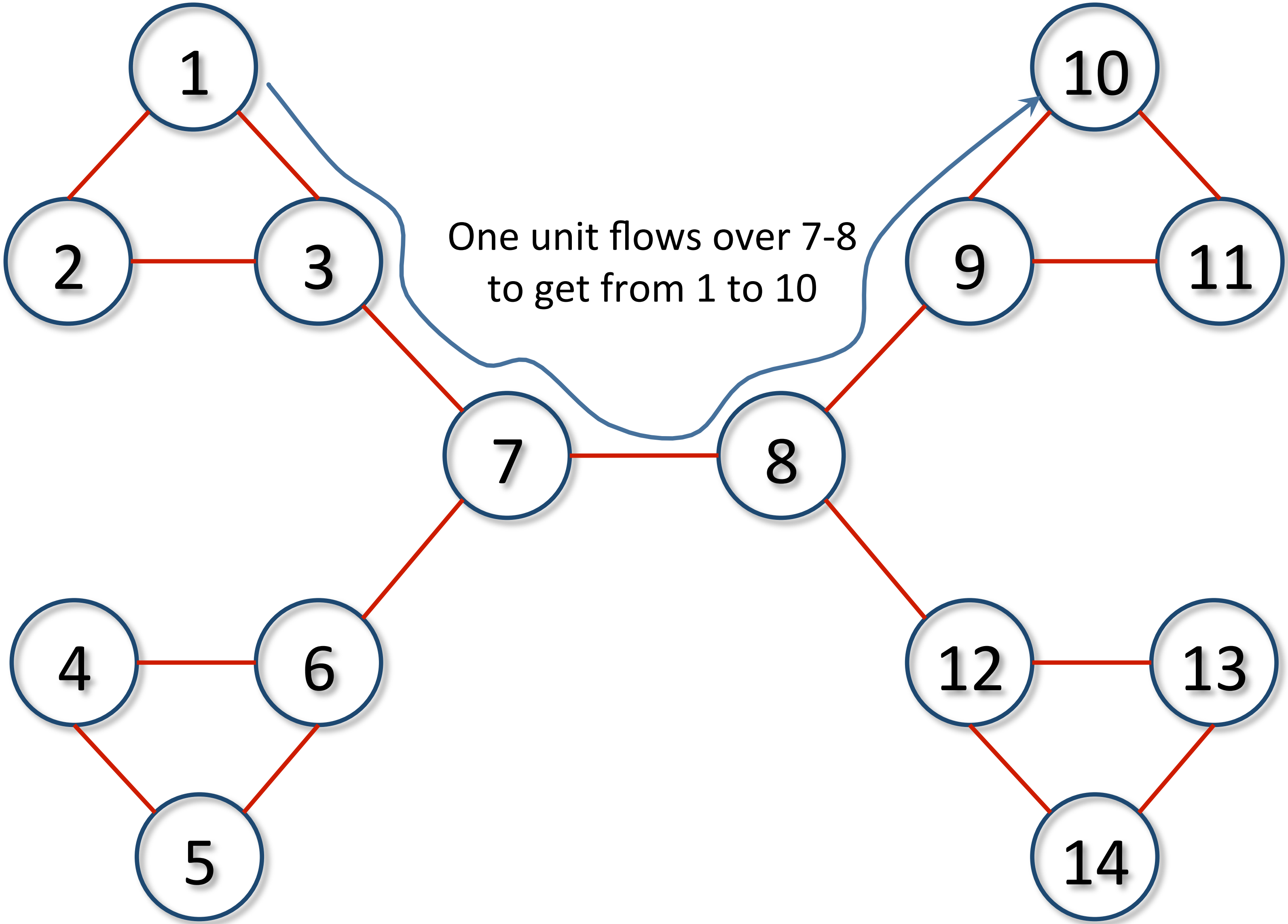
Edge Betweenness Example

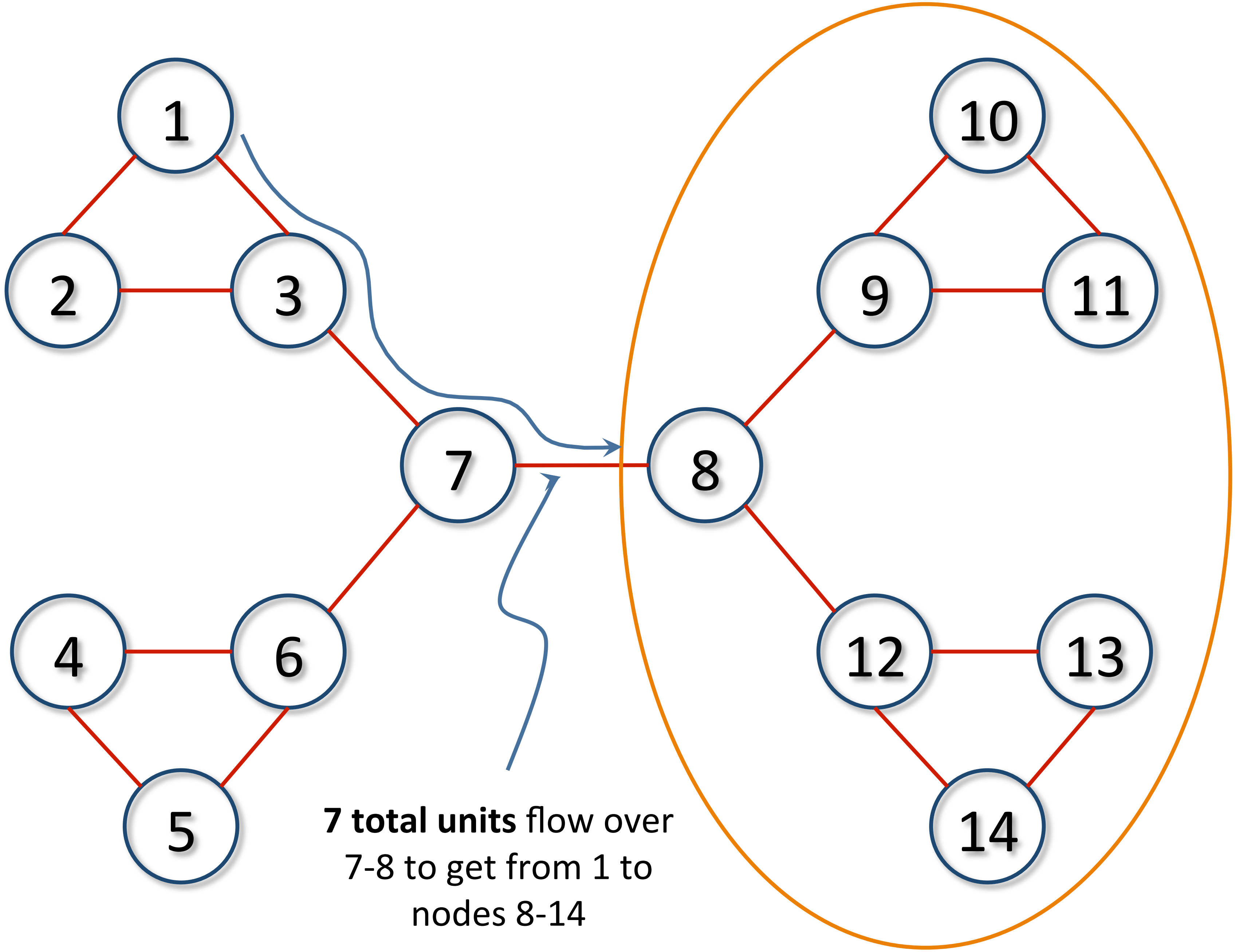


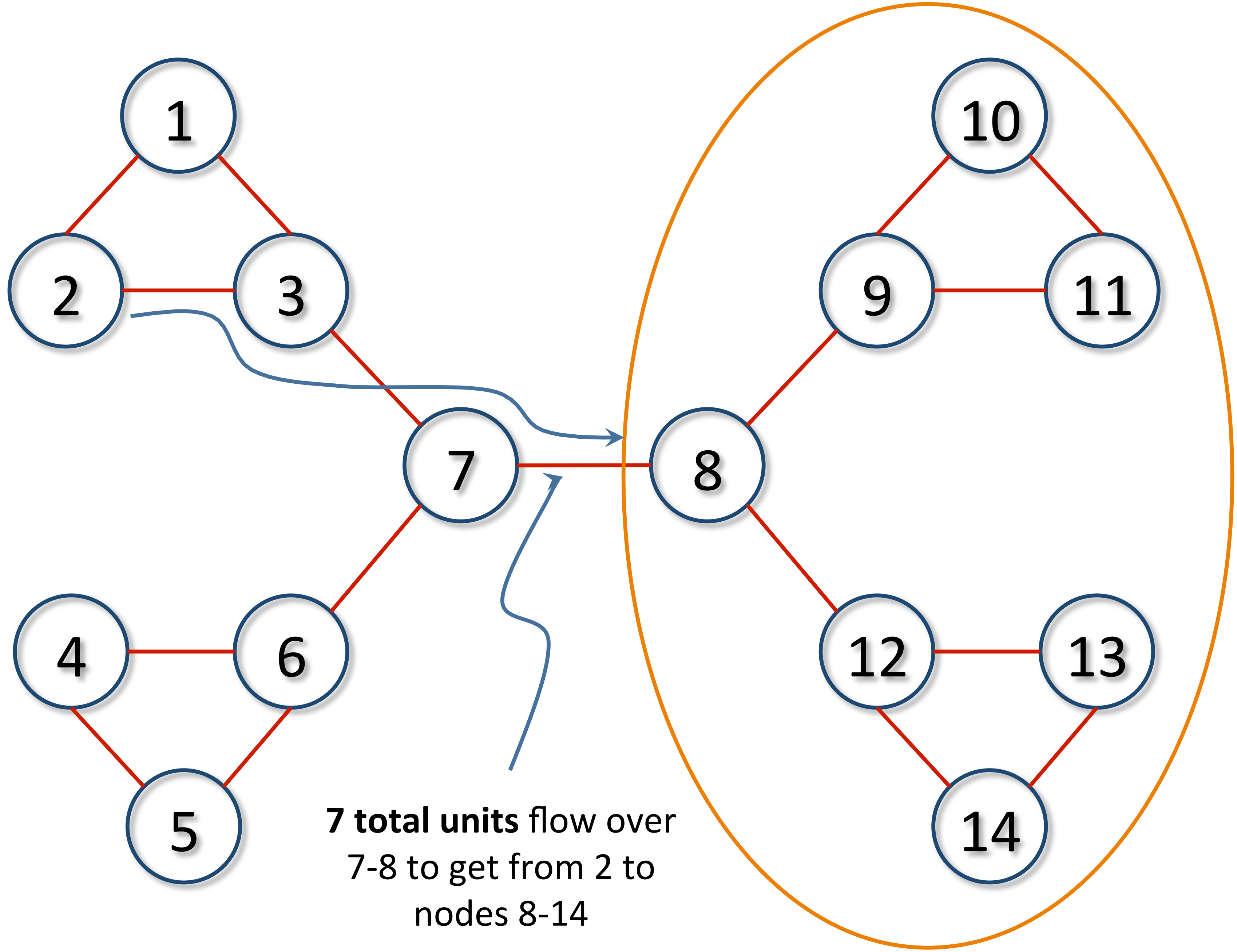


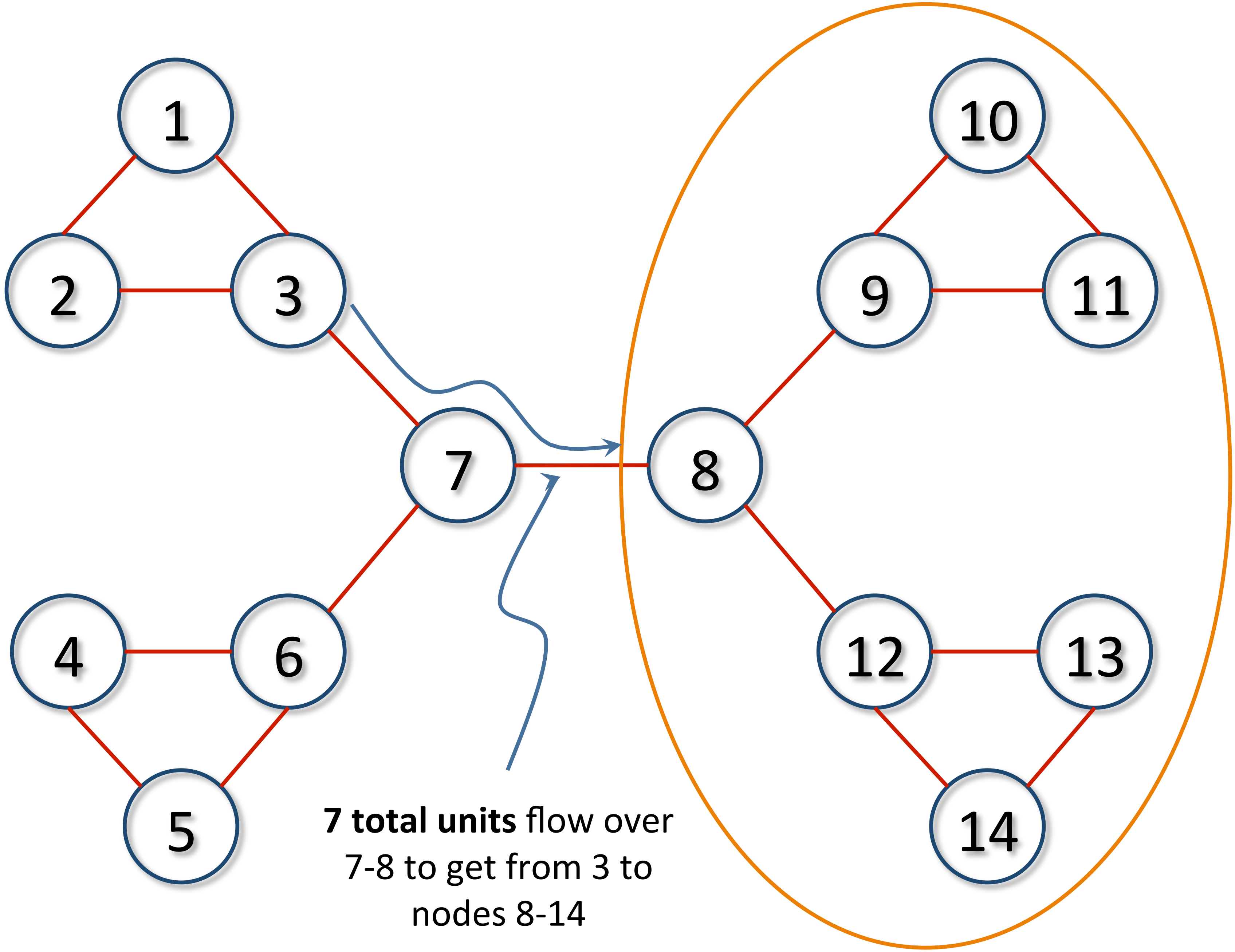
One unit flows over 7-8
to get from 1 to 8
1
2
3
4
5
6
7
8
9
10
11
12
13
14

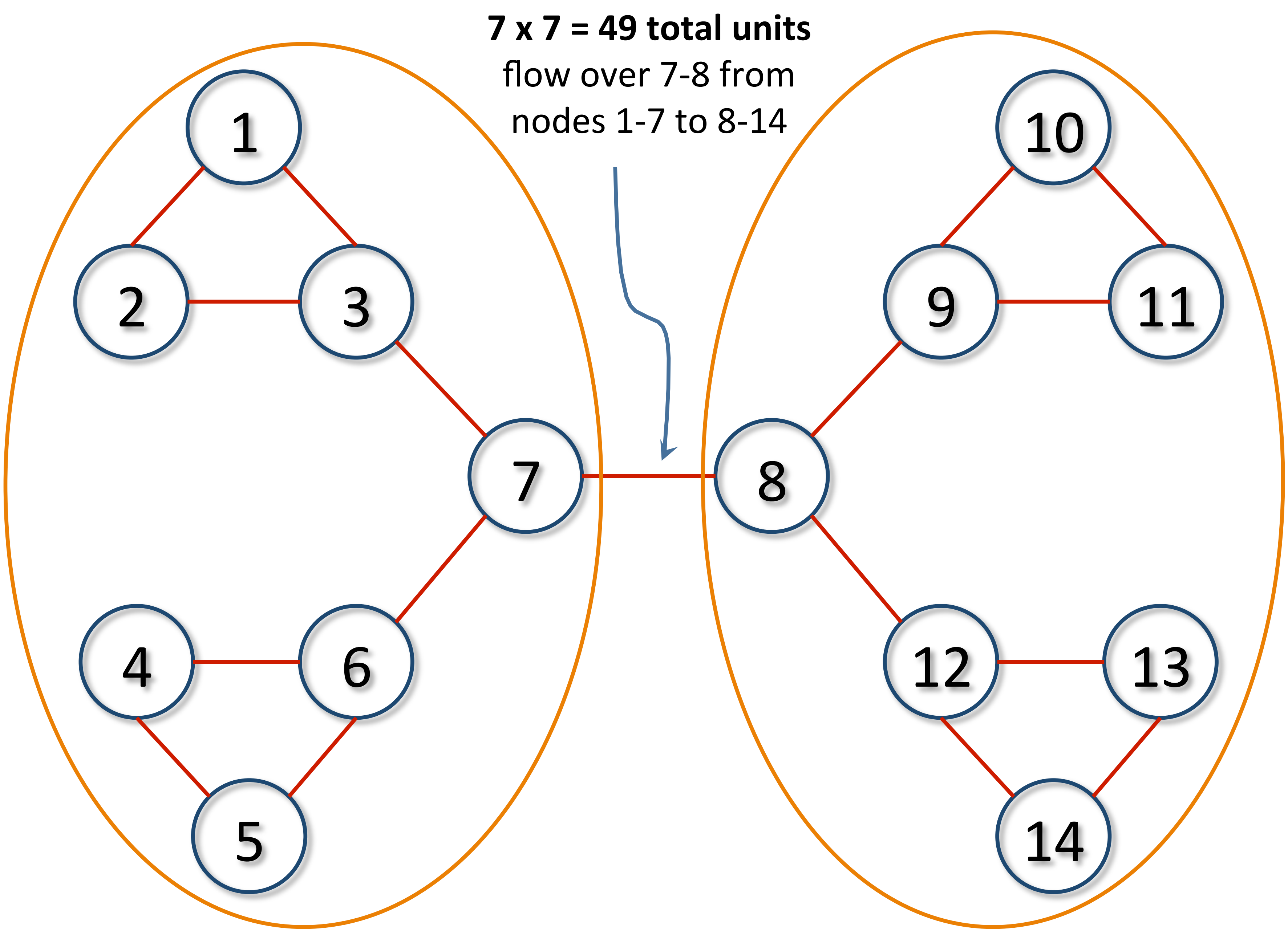

1
2
3
4
5
6
7
8
9
10
11
12
13
14
One unit flows over 7-8
to get from 1 to 9


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

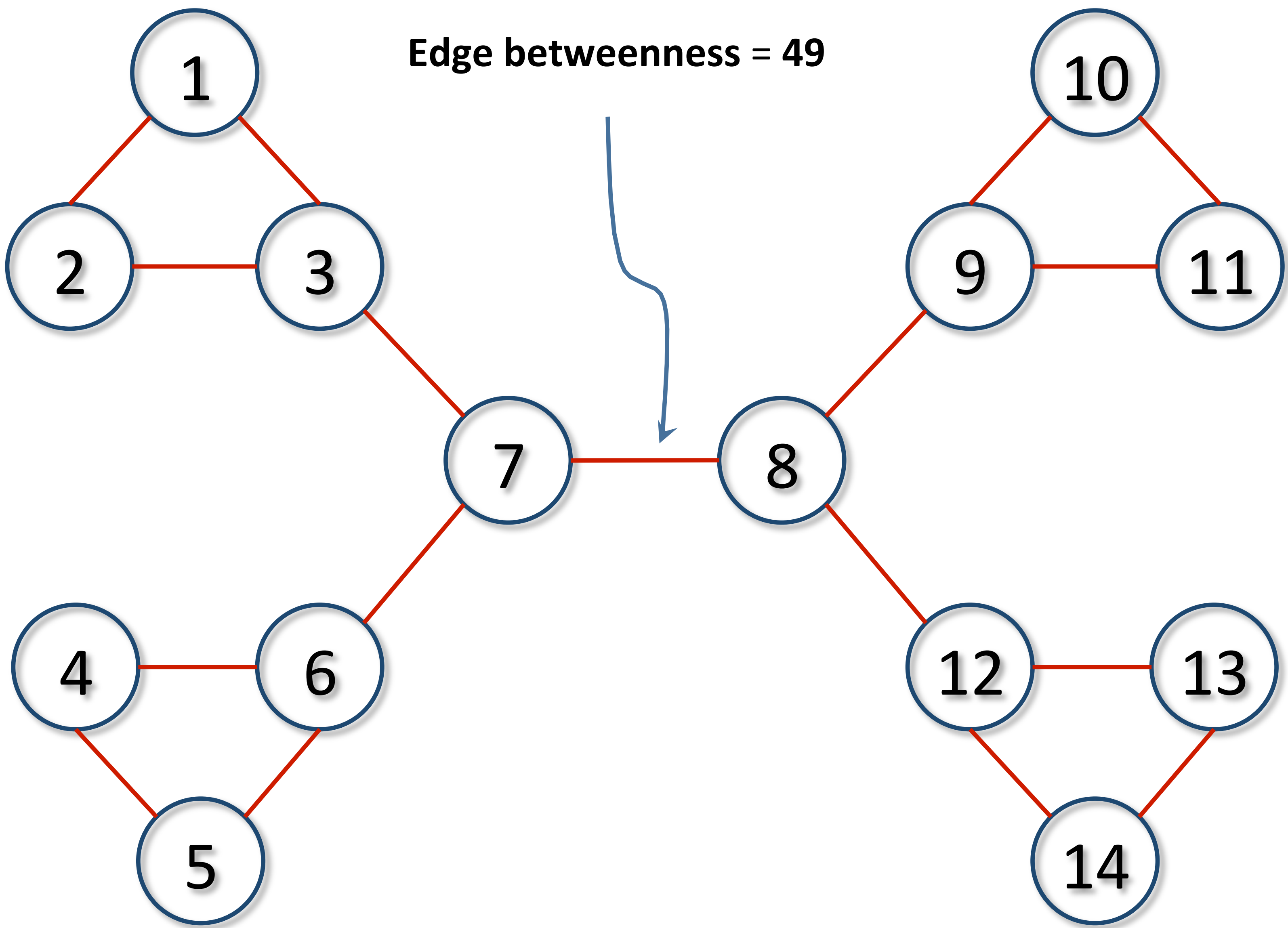

1
2
3
4
5
6
7
8
9
10
11
12
13
14
7 total units flow over 7-8 to get from 1 to nodes 8-14

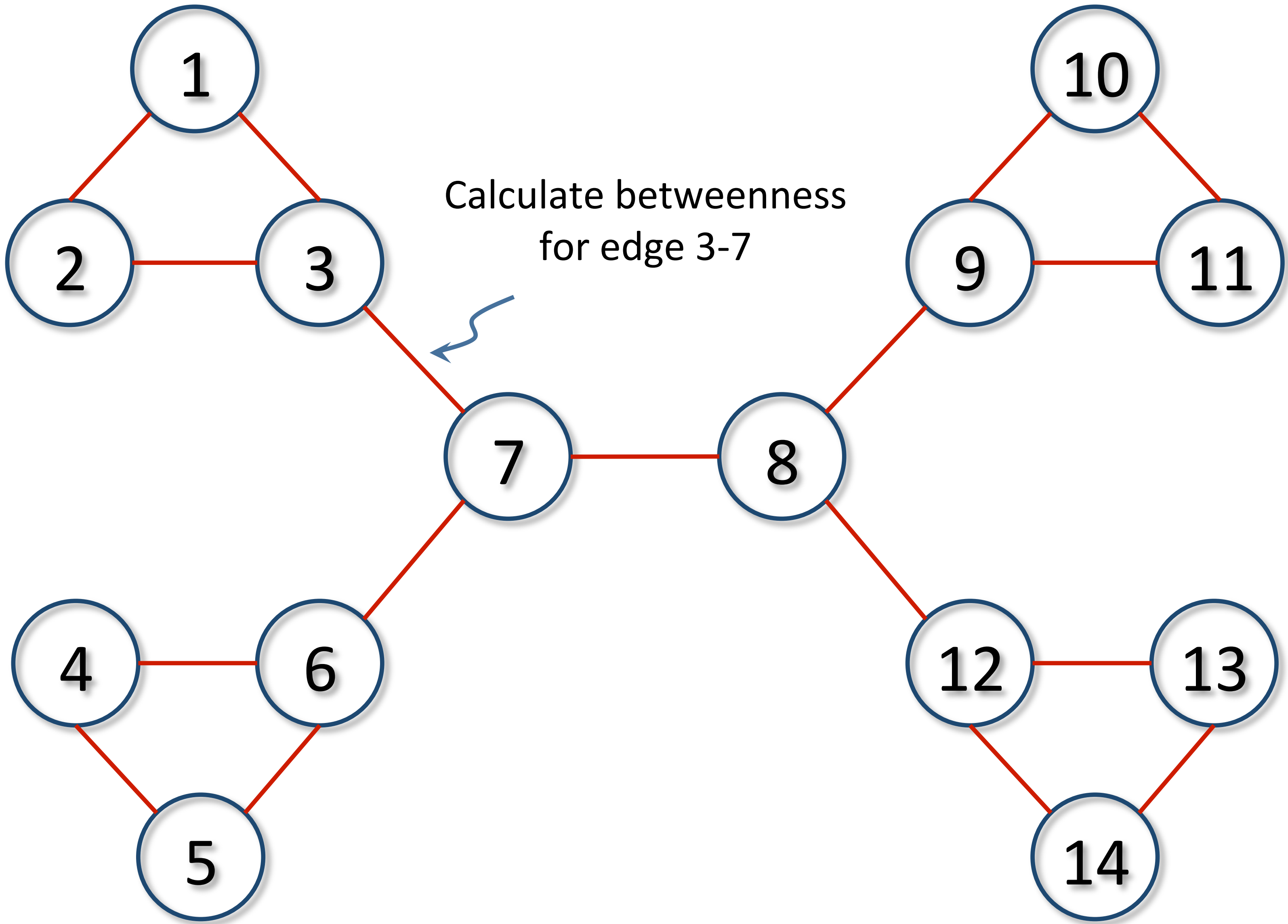

1
2
3
4
5
6
7
8
9
10
11
12
13
14
7 total units flow over 7-8 to get from 2 to nodes 8-14

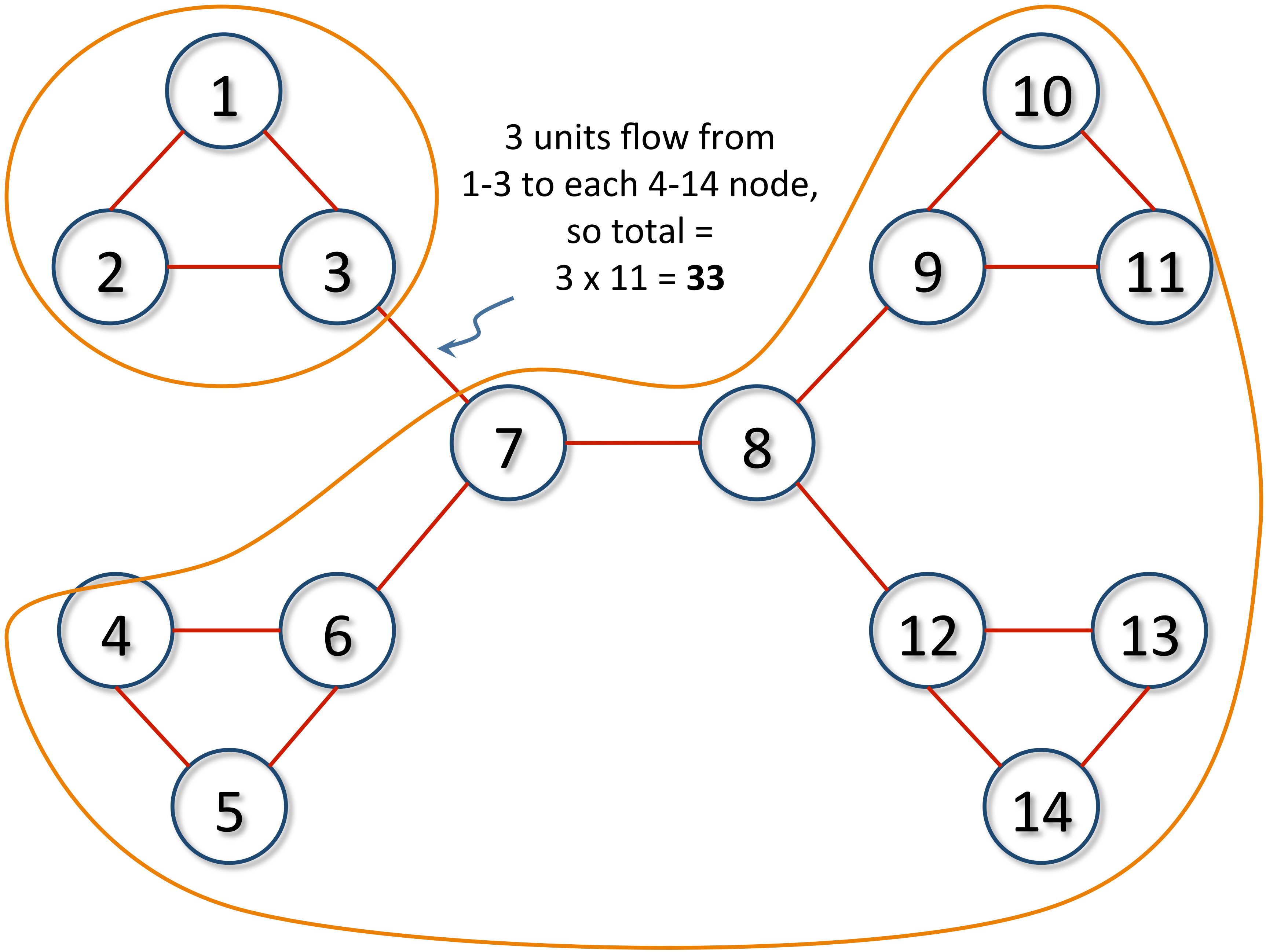

1
2
3
4
5
6
7
8
9
10
11
12
13
14
7 total units flow over 7-8 to get from 3 to nodes 8-14



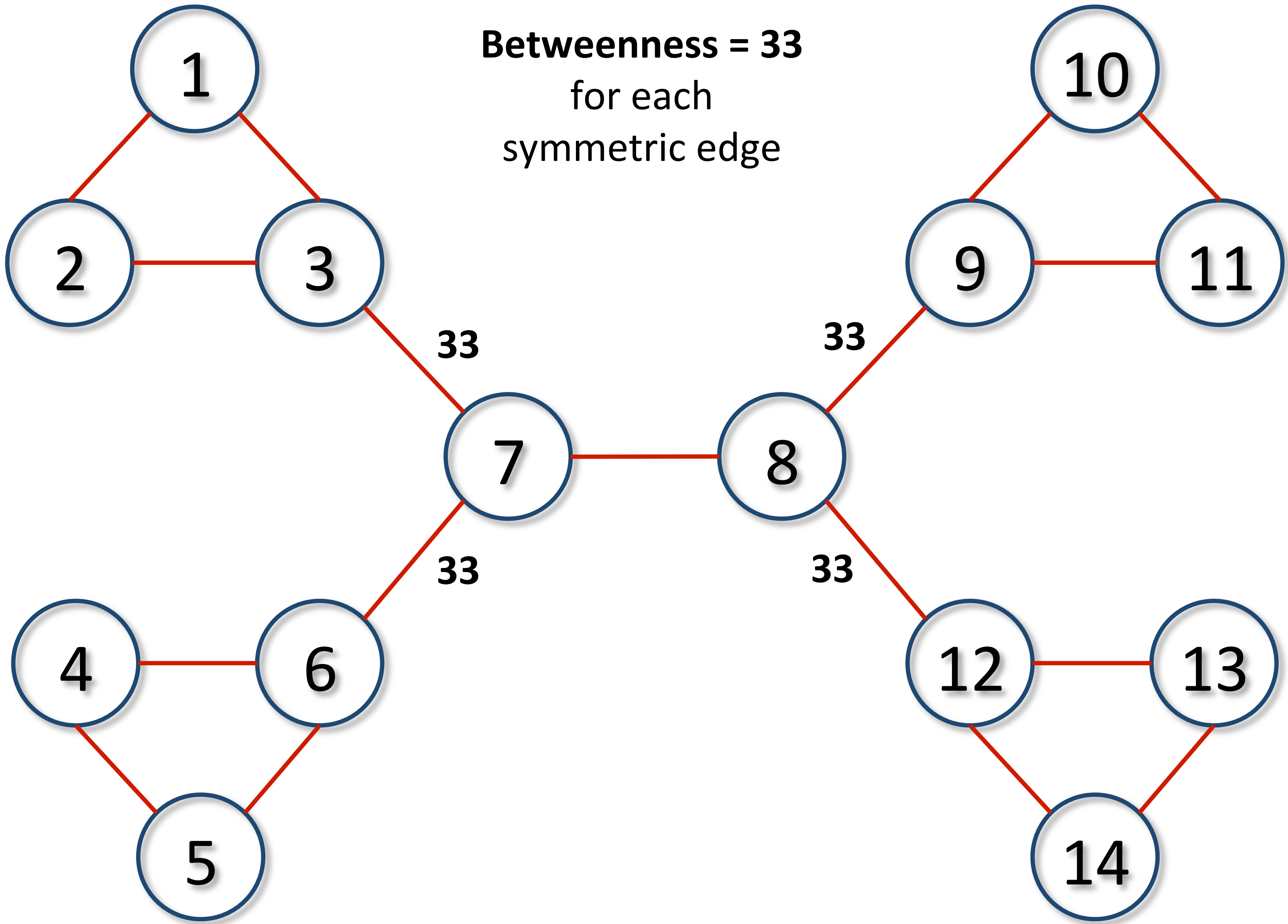
7 x 7 = 49 total units
flow over 7-8 from
nodes 1-7 to 8-14
1
2
3
7
4
6
5
10
9
11
8
12
13
14

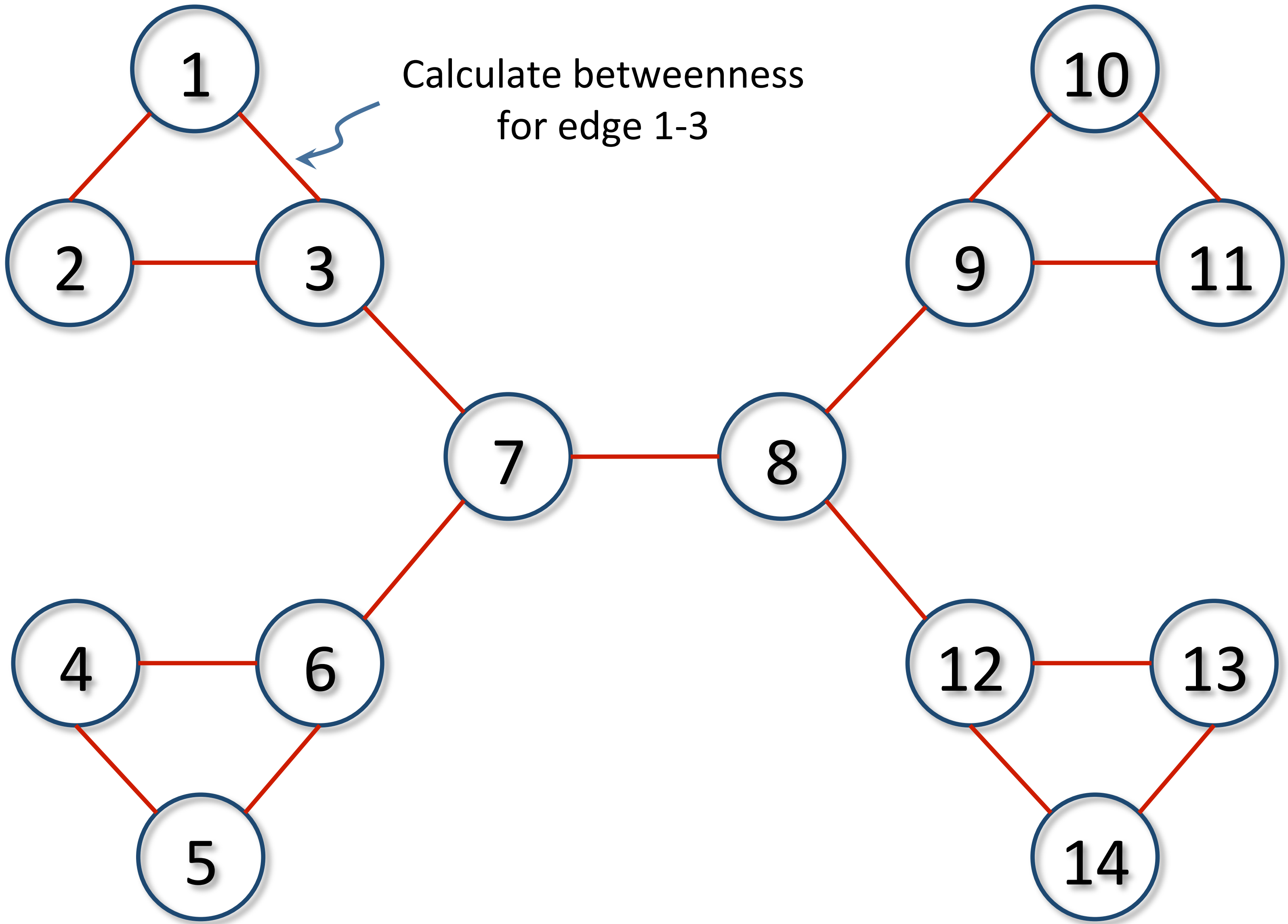

Edge betweenness = 49
1
2
3
4
5
6
7
8
9
10
11
12
13
14

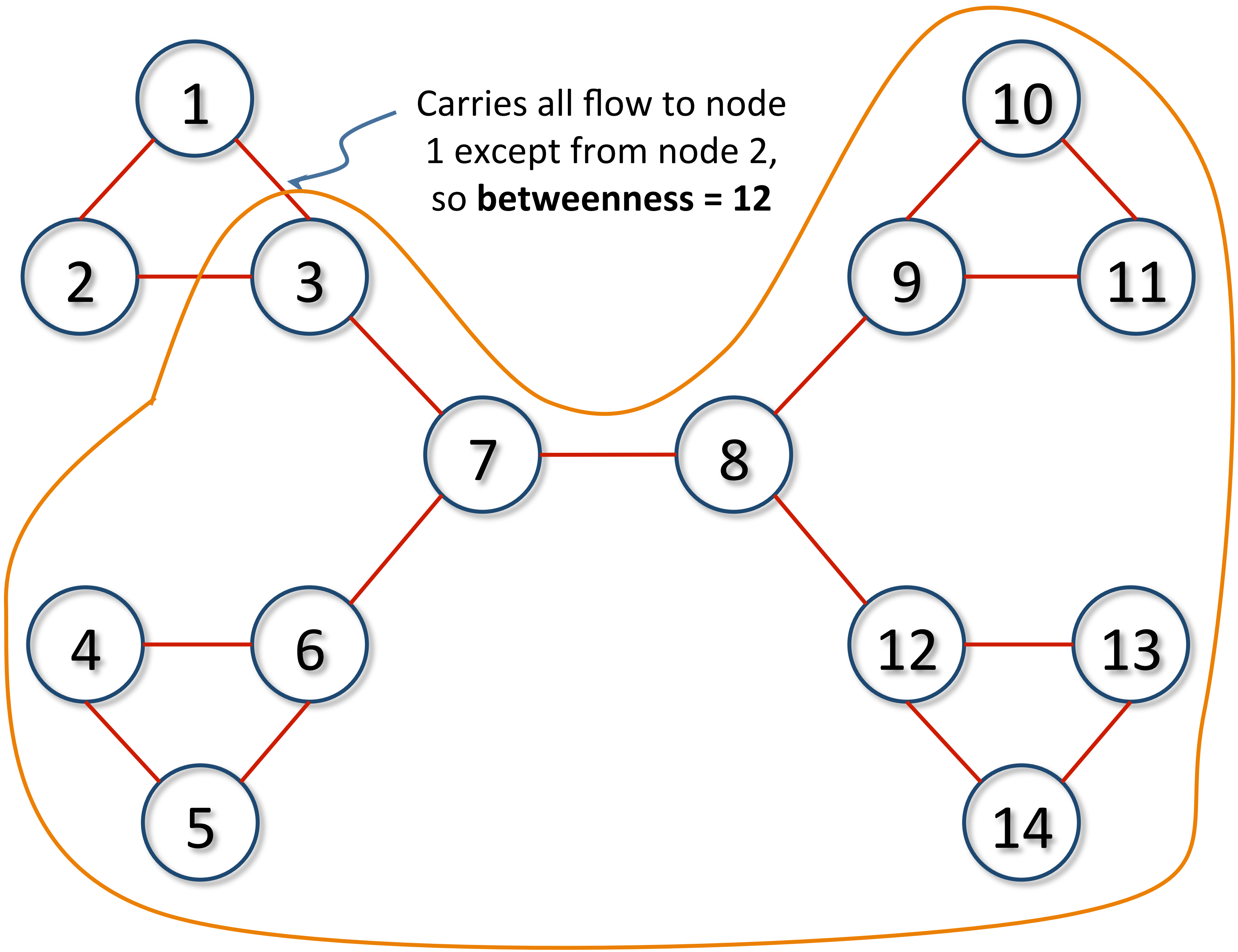

Calculate betweenness
for edge 3-7
1
2
3
4
5
6
7
8
9
10
11
12
13
14

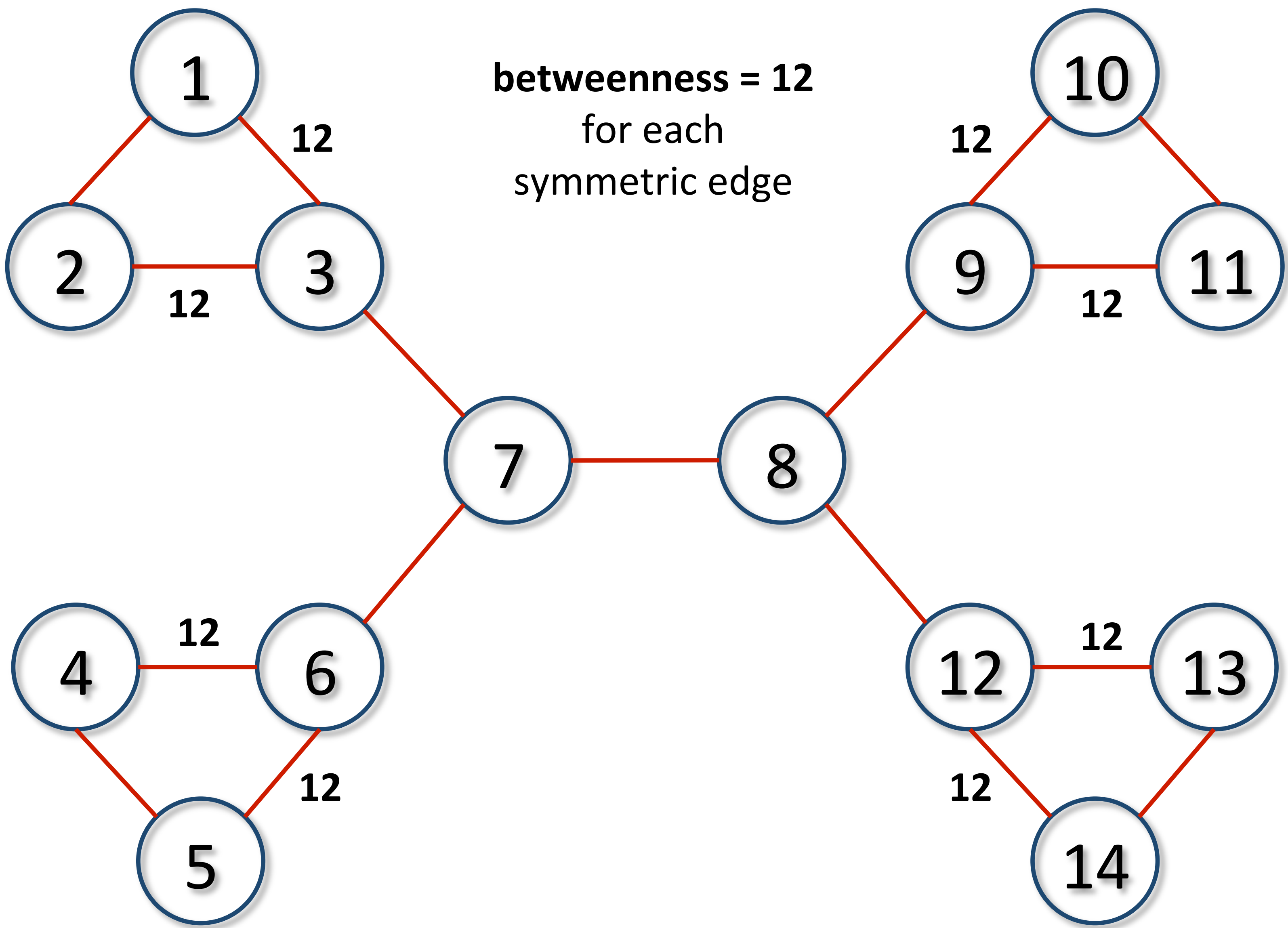


3 units flow from
1-3 to each 4-14 node,
so total =
3 x 11 = **33**
1
2
3
4
5
6
7
8
9
10
11
12
13
14

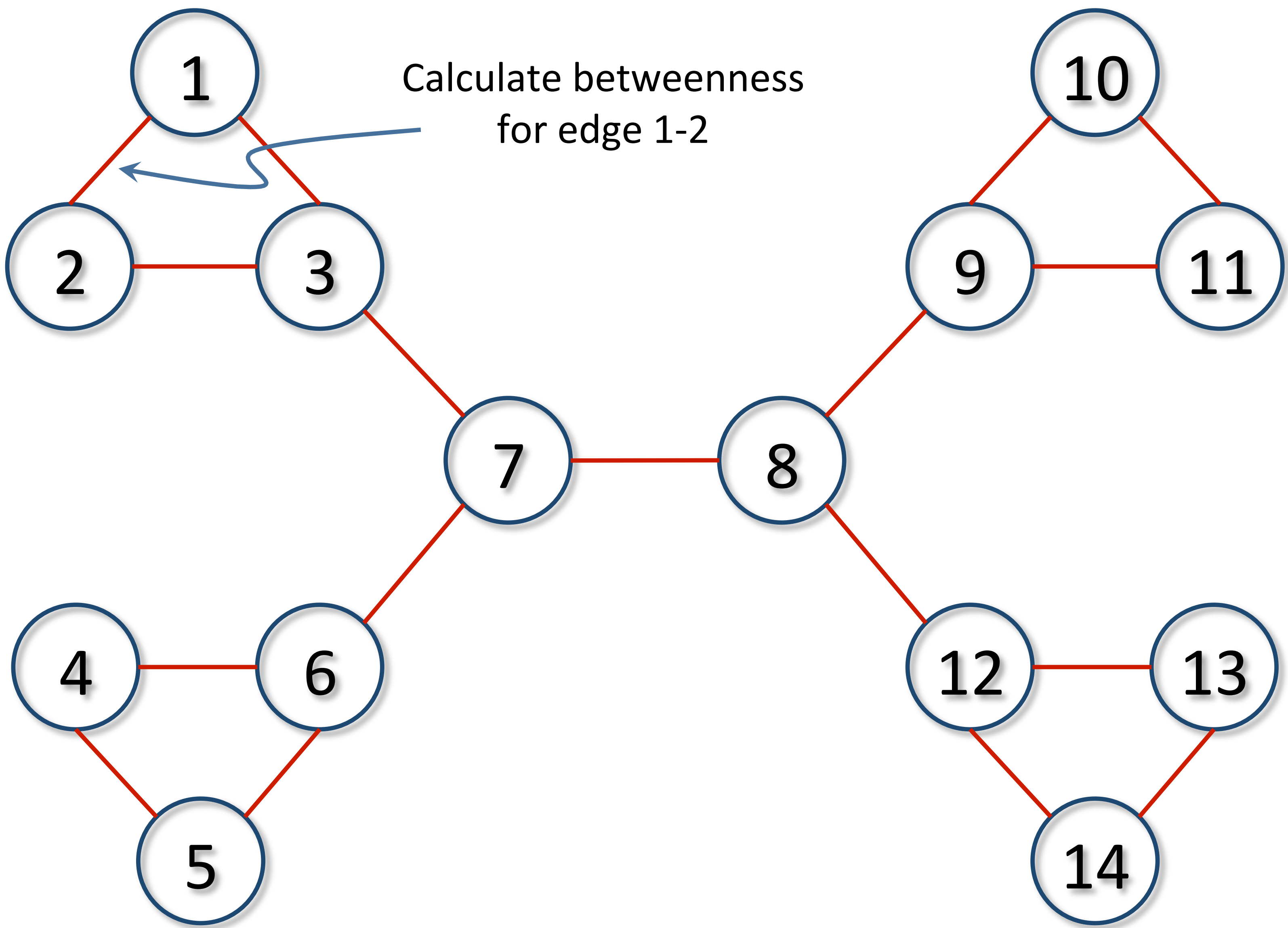

Betweenness = 33
for each
symmetric edge
1
2
3
33
7
8
33
9
10
11
33
33
4
5
6
12
13
14

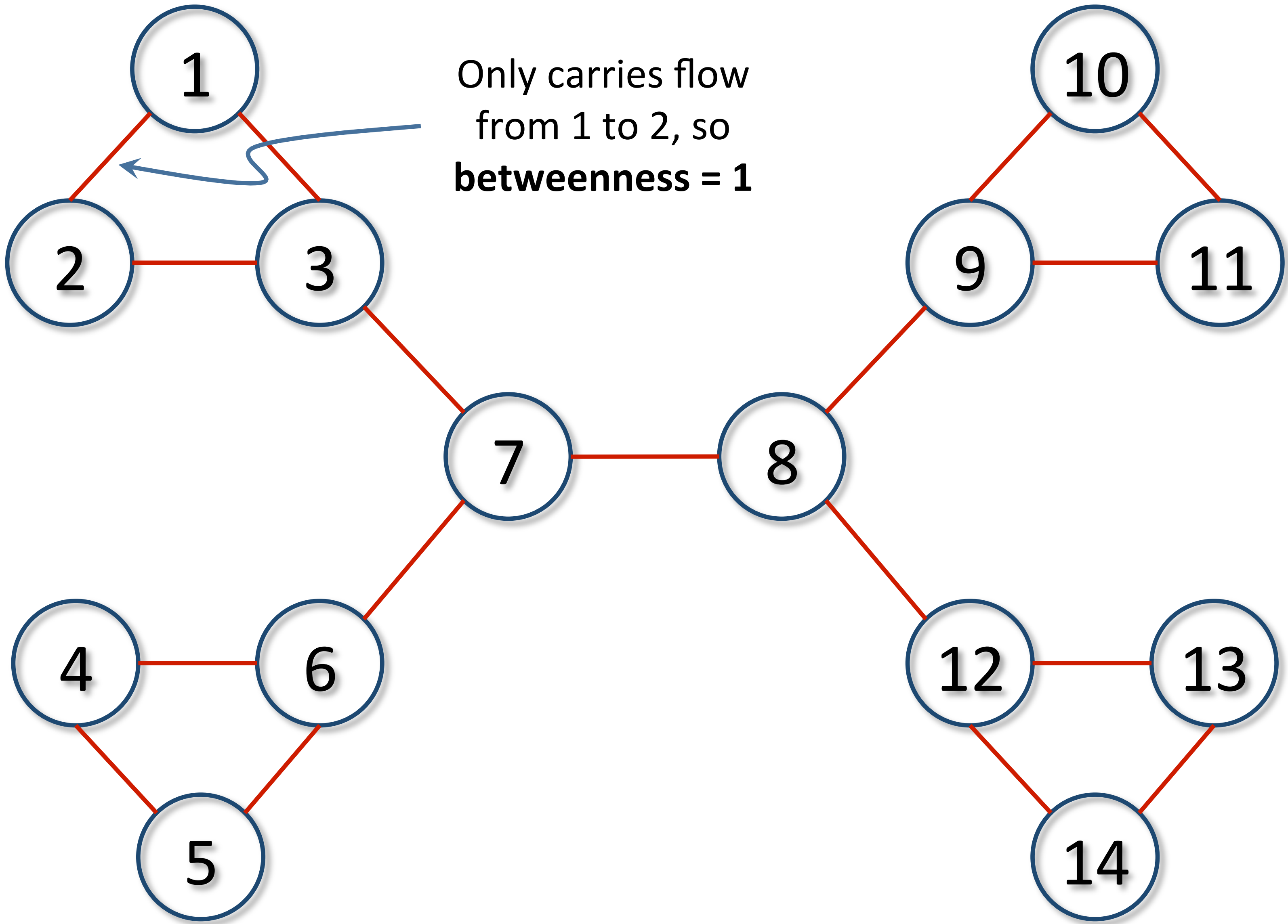

Calculate betweenness
for edge 1-3
1
2
3
4
5
6
7
8
9
10
11
12
13
14

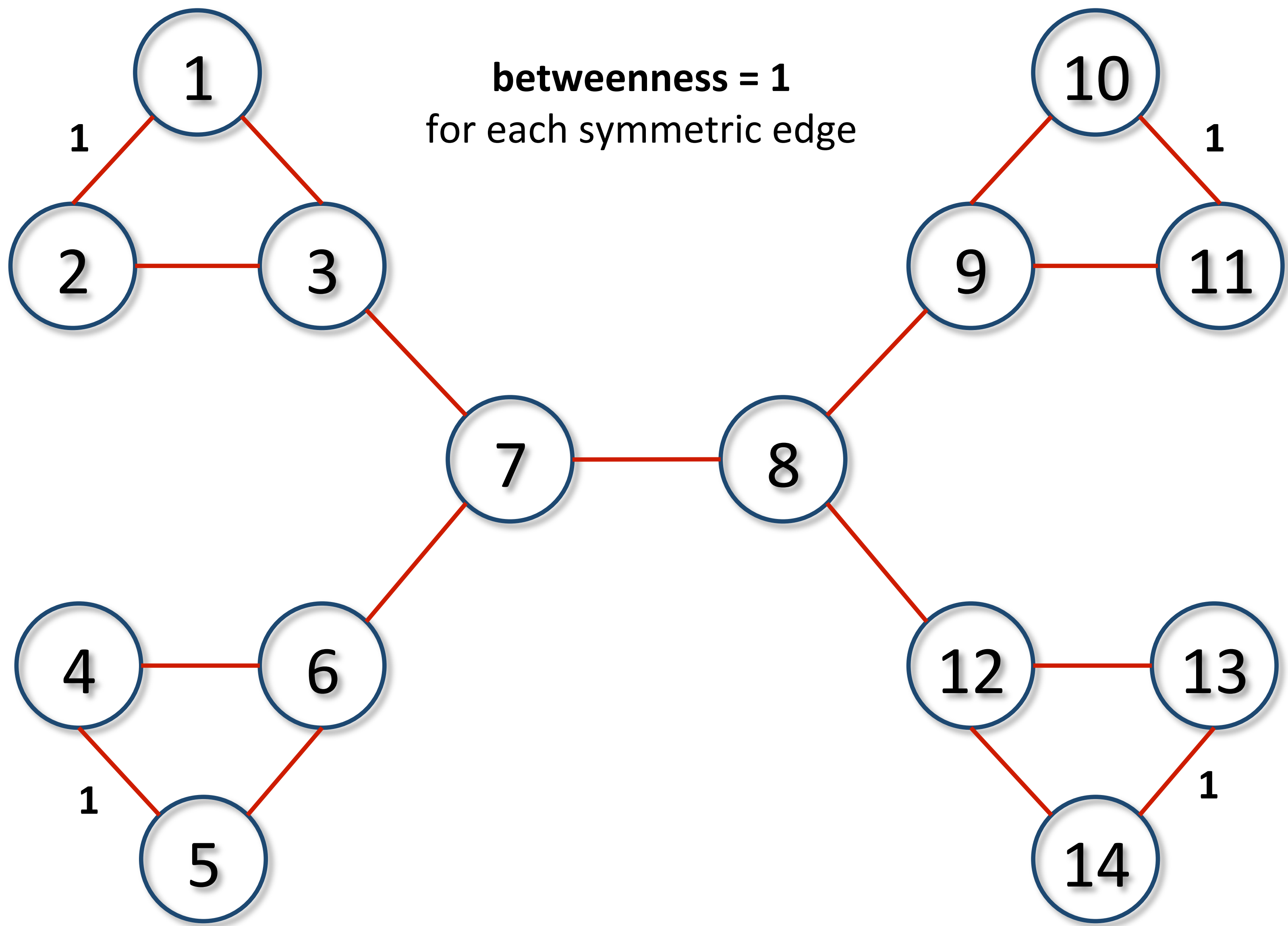

Carries all flow to node
1 except from node 2,
so **betweenness = 12**
1
2
3
4
5
6
7
8
9
10
11
12
13
14

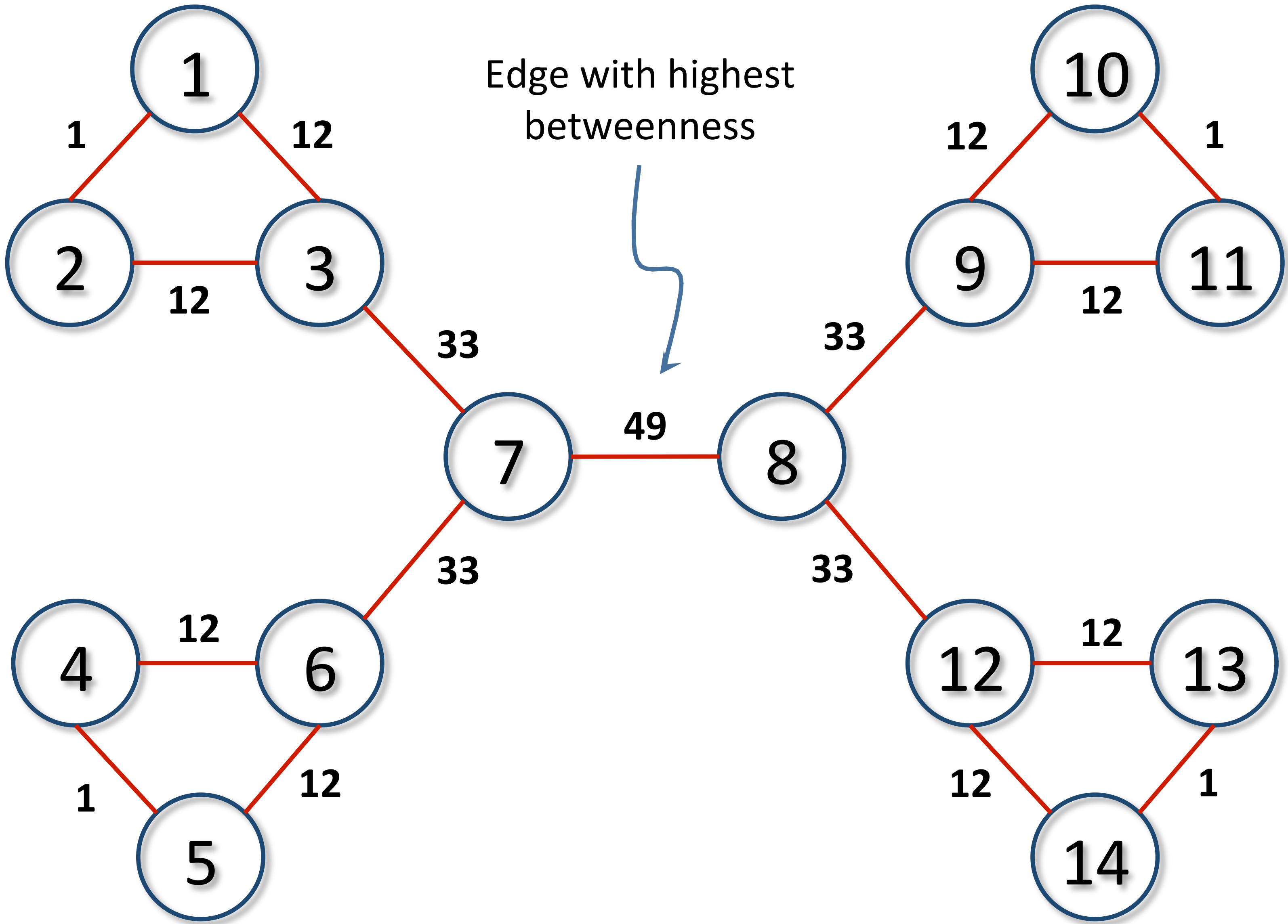


betweenness = 12
for each
symmetric edge
1
2
3
4
5
6
7
8
9
10
11
12
13
14
12
12
12
12
12
12
12
12

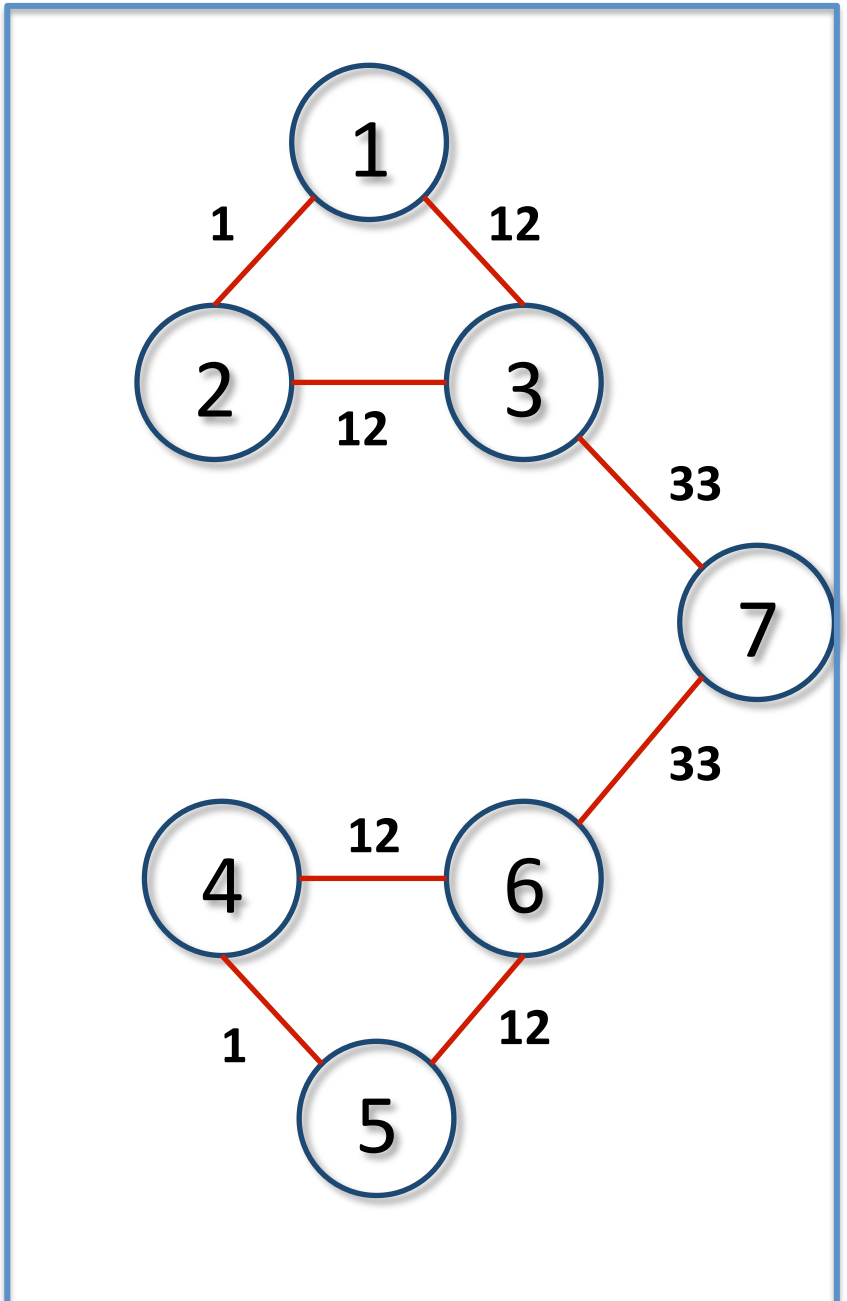

Calculate betweenness
for edge 1-2
1
2
3
4
5
6
7
8
9
10
11
12
13
14

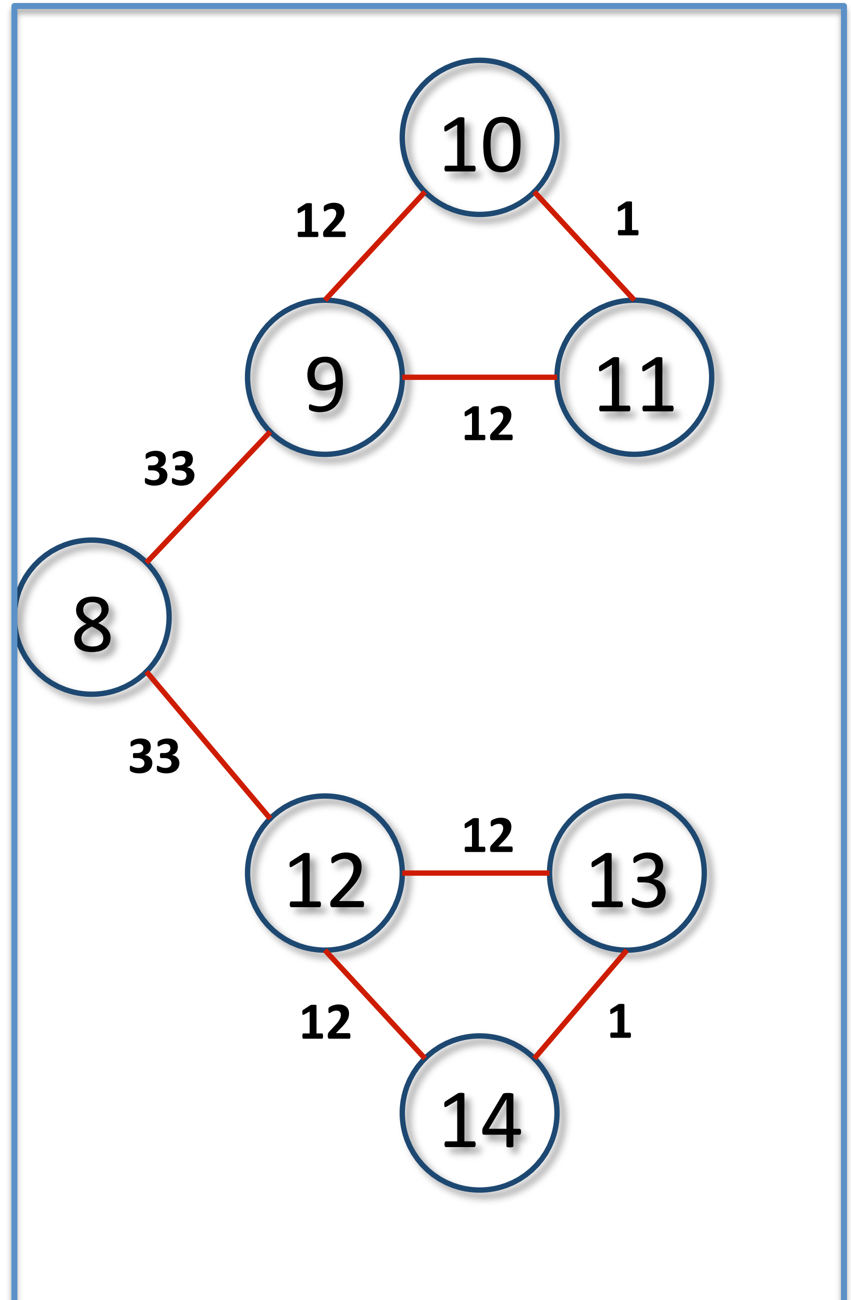

1
2
3
4
5
6
7
8
9
10
11
12
13
14
Only carries flow
from 1 to 2, so
betweenness = 1

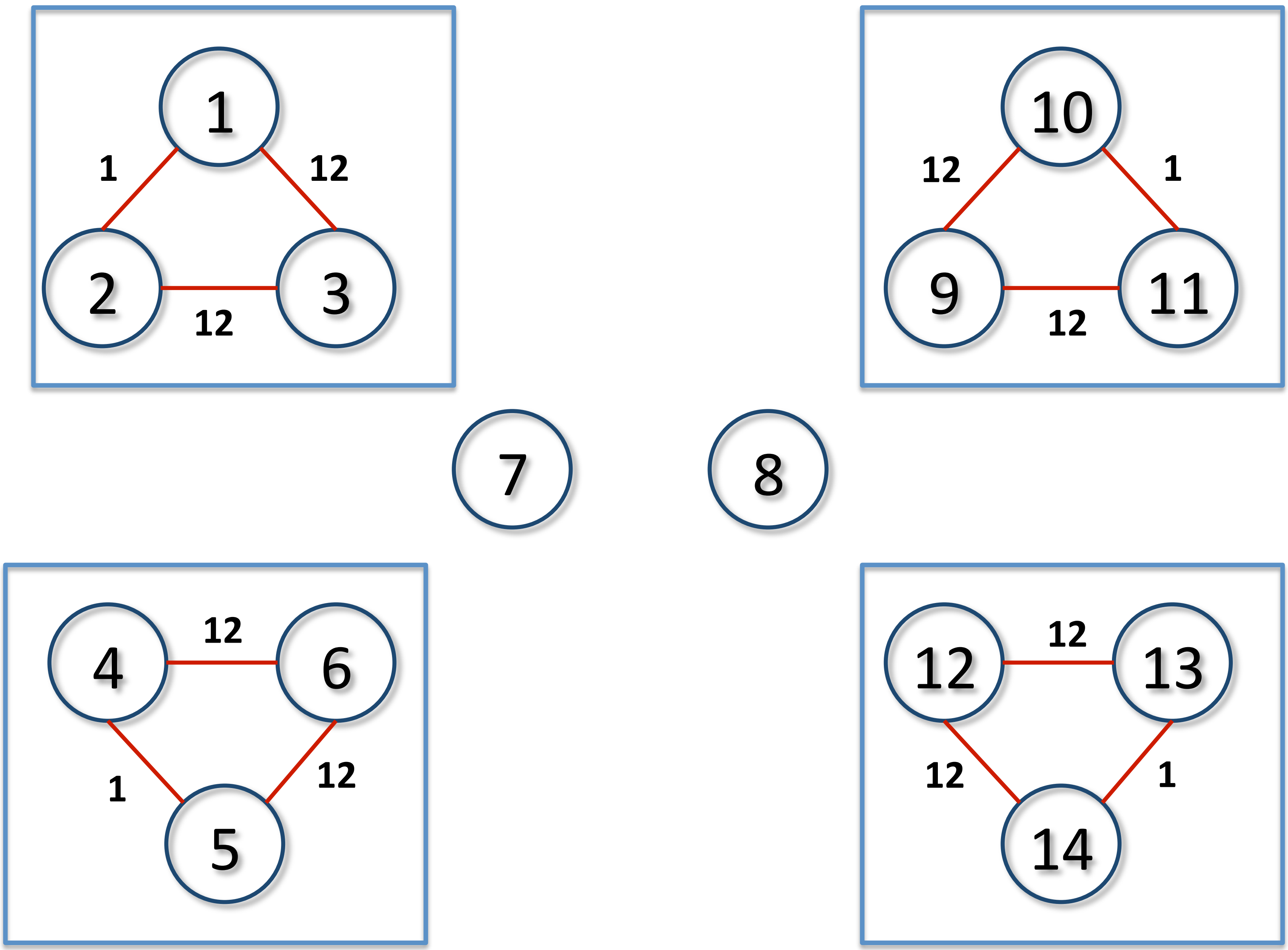

betweenness = 1
for each symmetric edge
1
2
3
4
5
6
7
8
9
10
11
12
13
14
1
1
1
1



Now progressively remove edges with highest betweenness


1
1
12
2
12
3
33
7
33
4
12
6
1
12
5


10
12
1
9
12
11
33
8
33
12
12
13
12
1
14


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

Complexity of Girvan-Newman

- 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

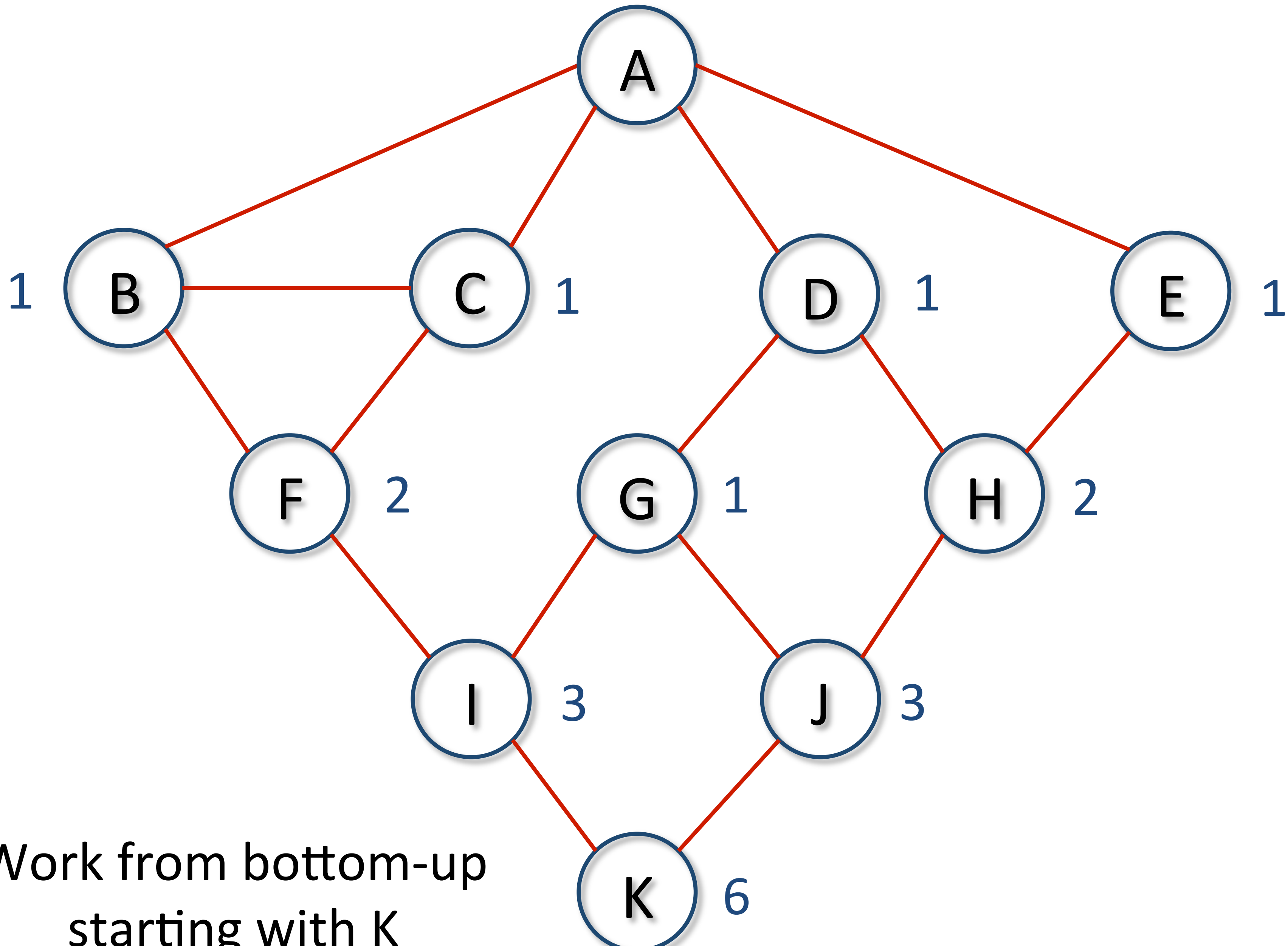
Computing Edge Betweenness Efficiently

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

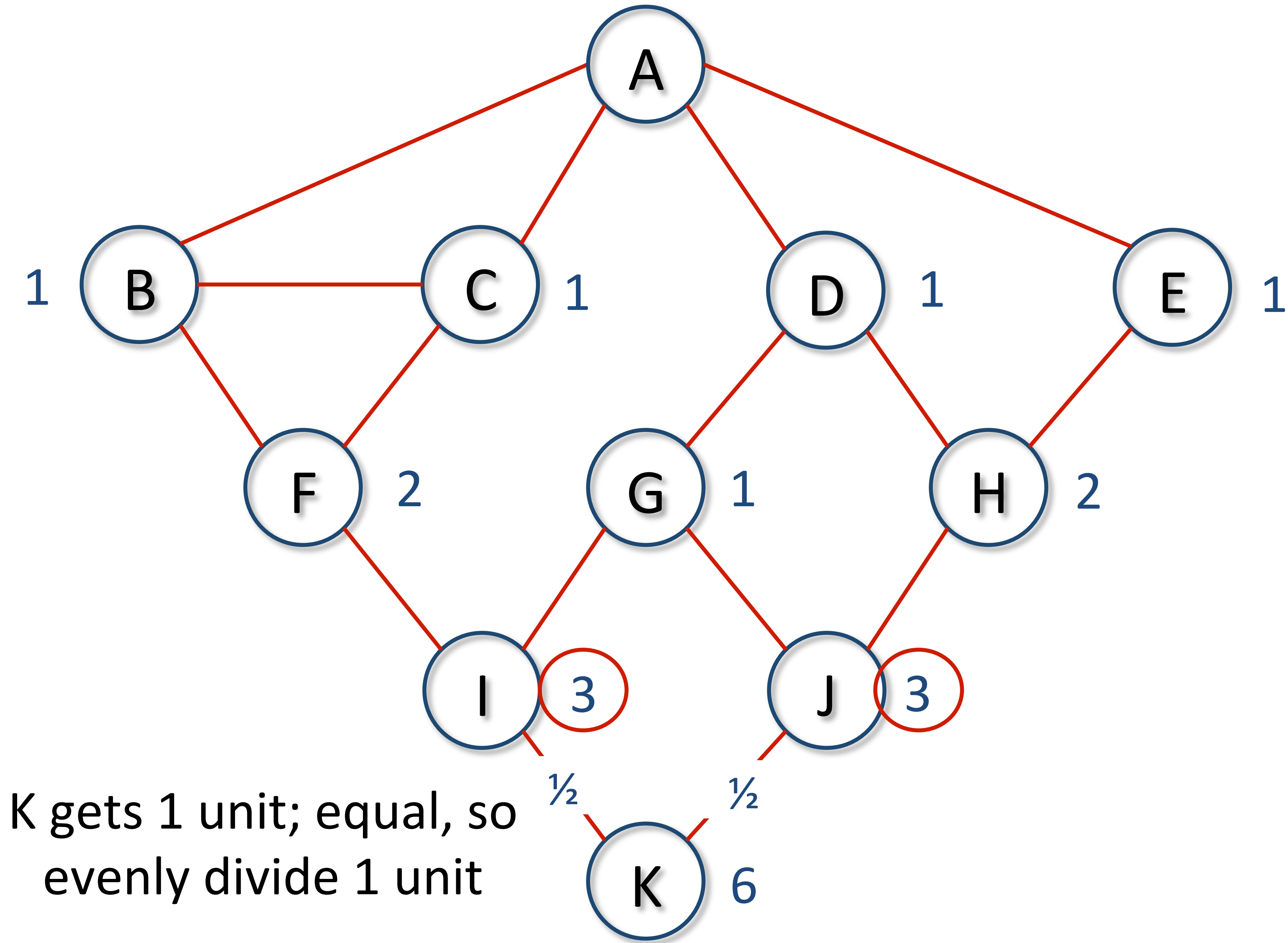
1. Perform breadth-first search of graph starting at node *N*
2. Determine the number of shortest paths from *N* to every other node
3. Based on these numbers, determine the amount of flow from *N* to all other nodes that use each edge

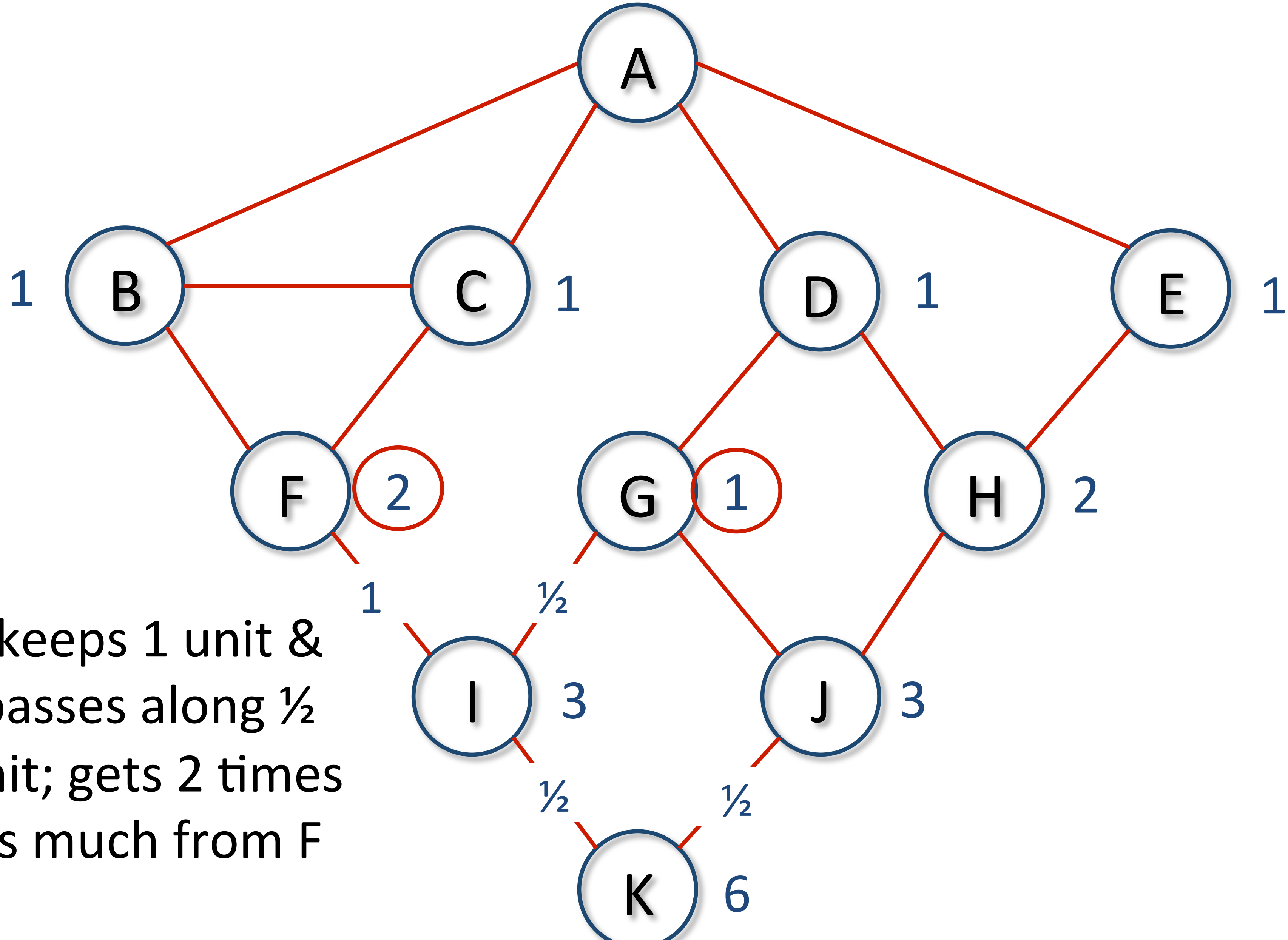
Divide sum of flow of all edges by 2

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

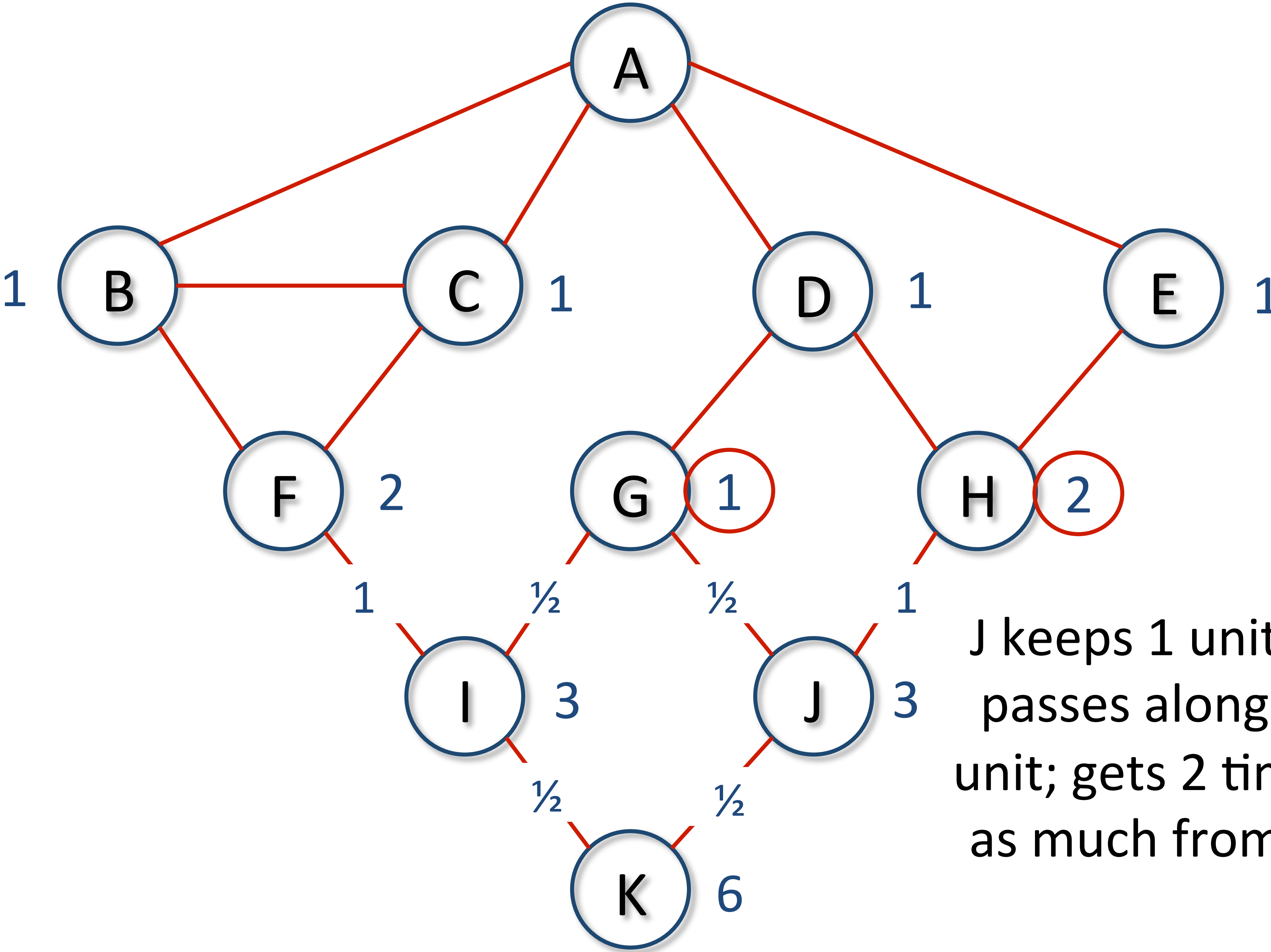


Work from bottom-up
starting with K

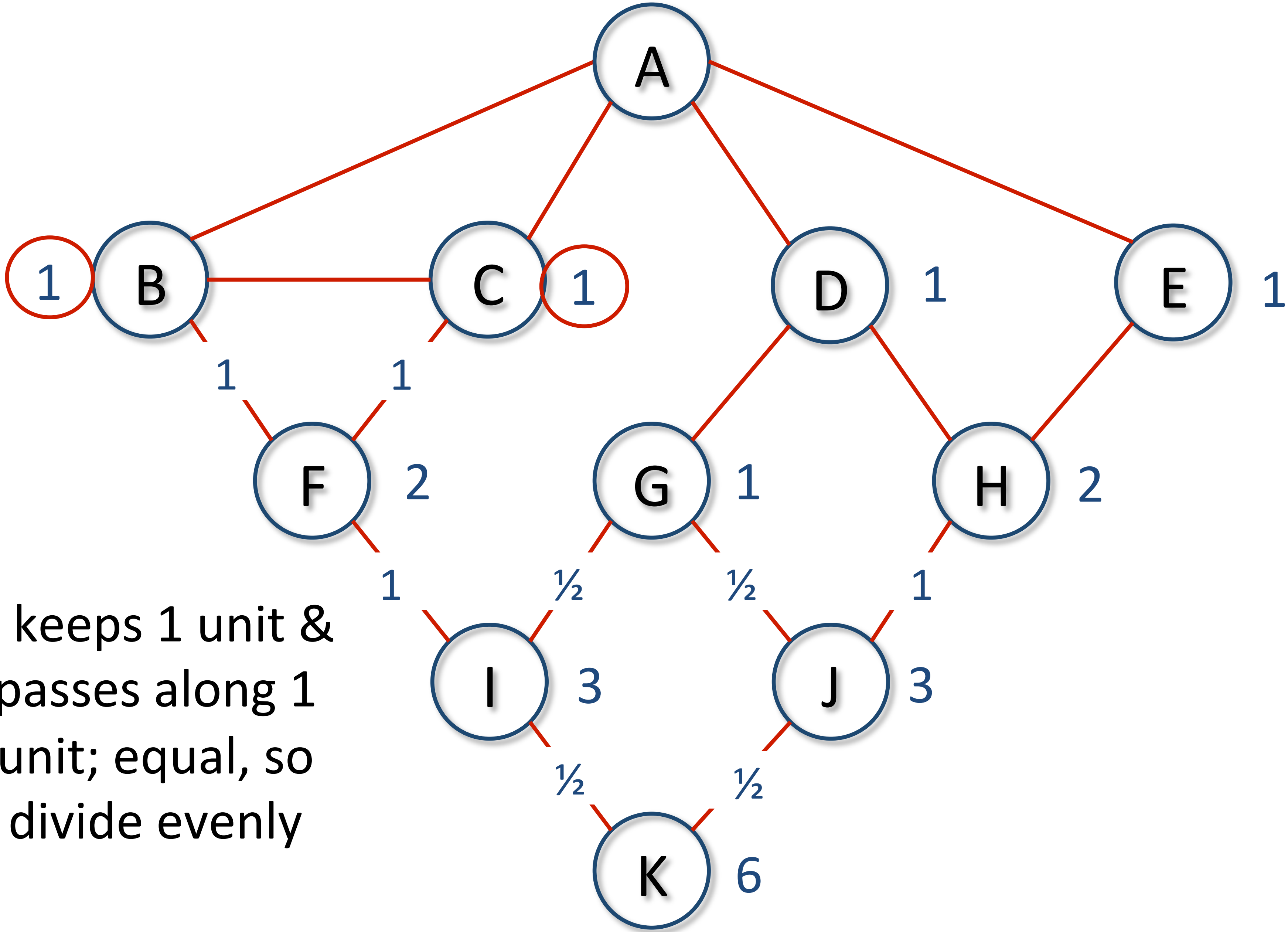

A
B
C
D
E
F
G
H
I
J
K
1
1
1
1
2
1
2
3
3
½
½
6
K gets 1 unit; equal, so evenly divide 1 unit



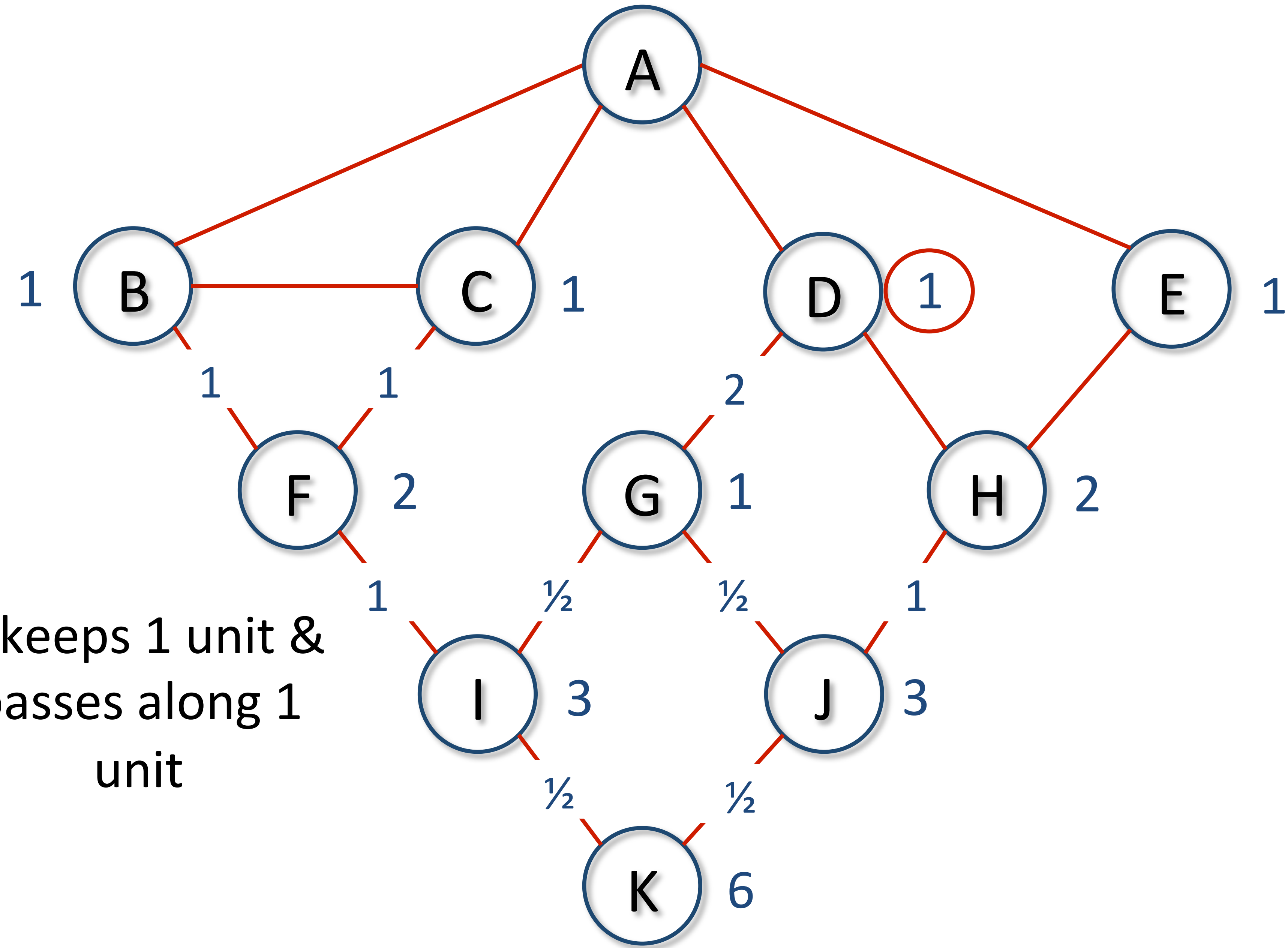
I keeps 1 unit & passes along ½ unit; gets 2 times as much from F



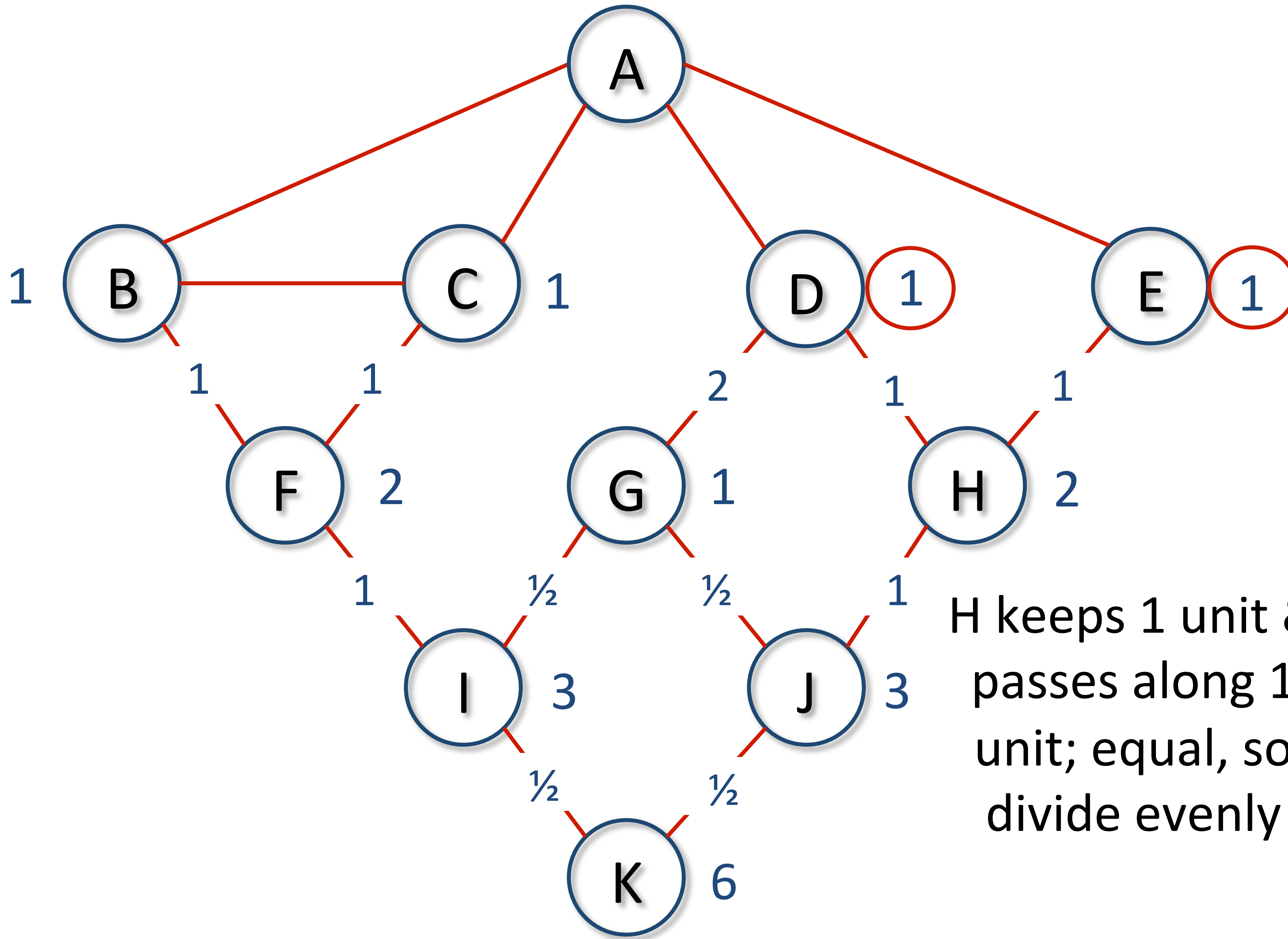
J keeps 1 unit & passes along ½ unit; gets 2 times as much from H



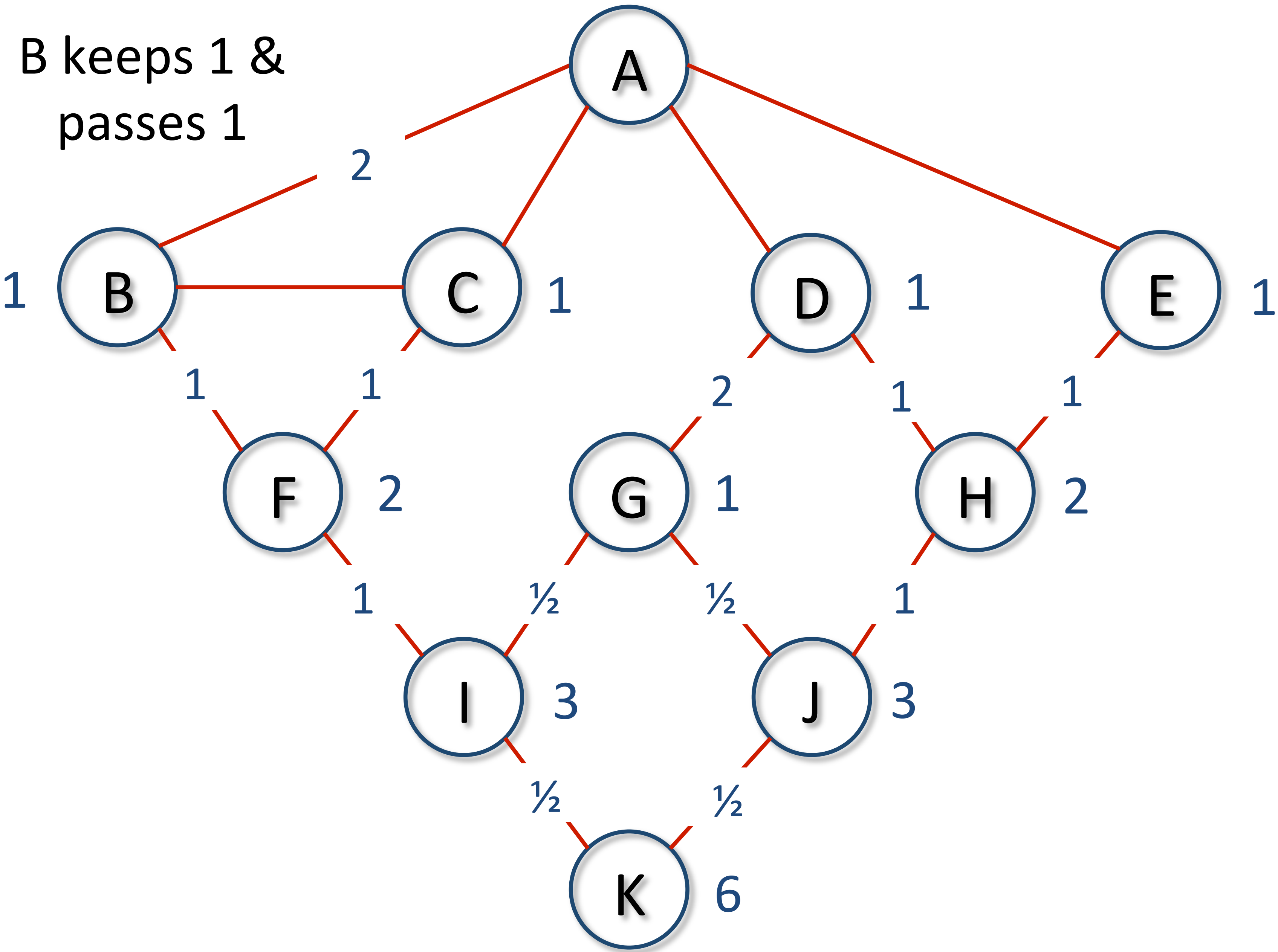
F keeps 1 unit & passes along 1 unit; equal, so divide evenly

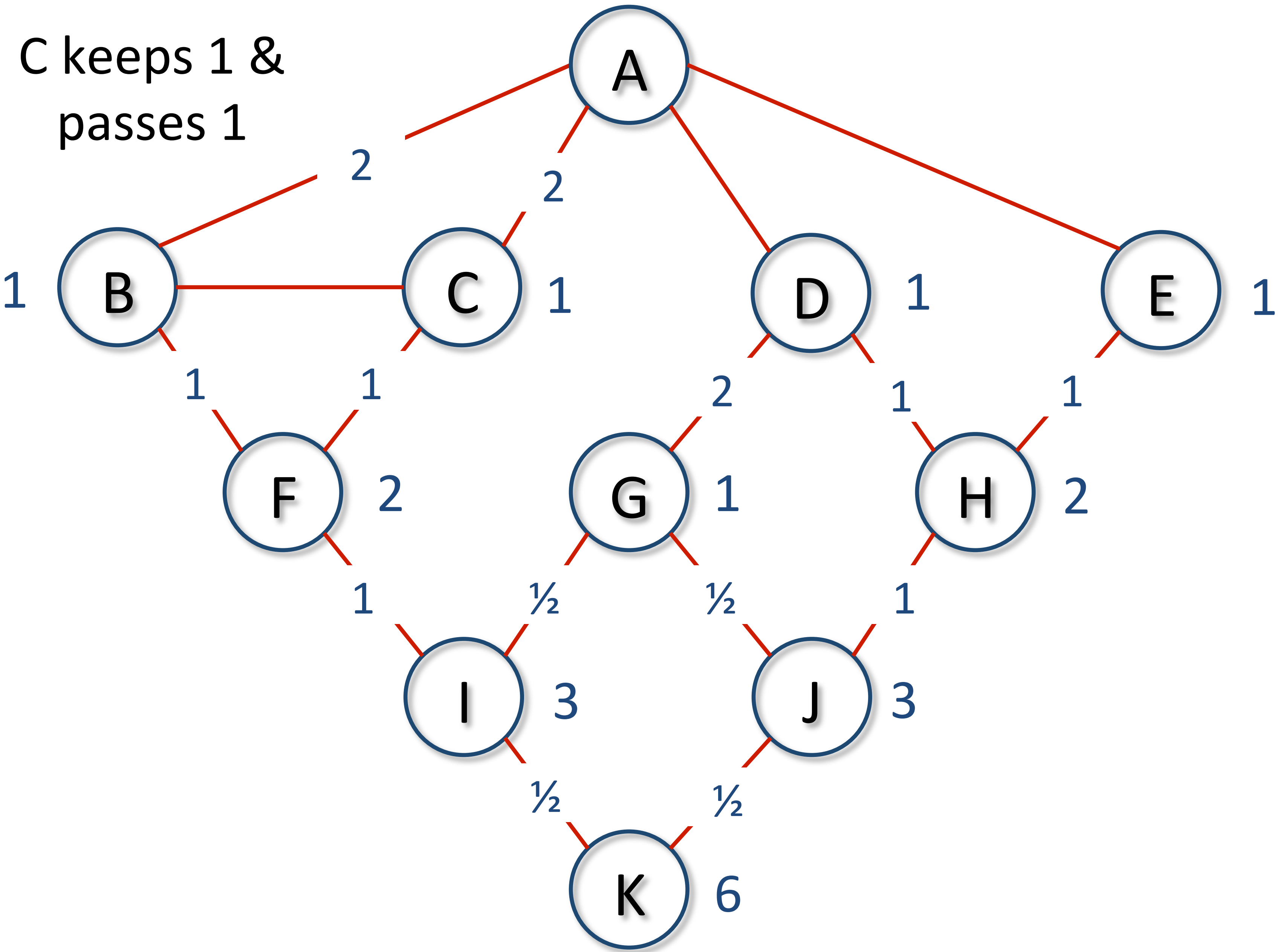


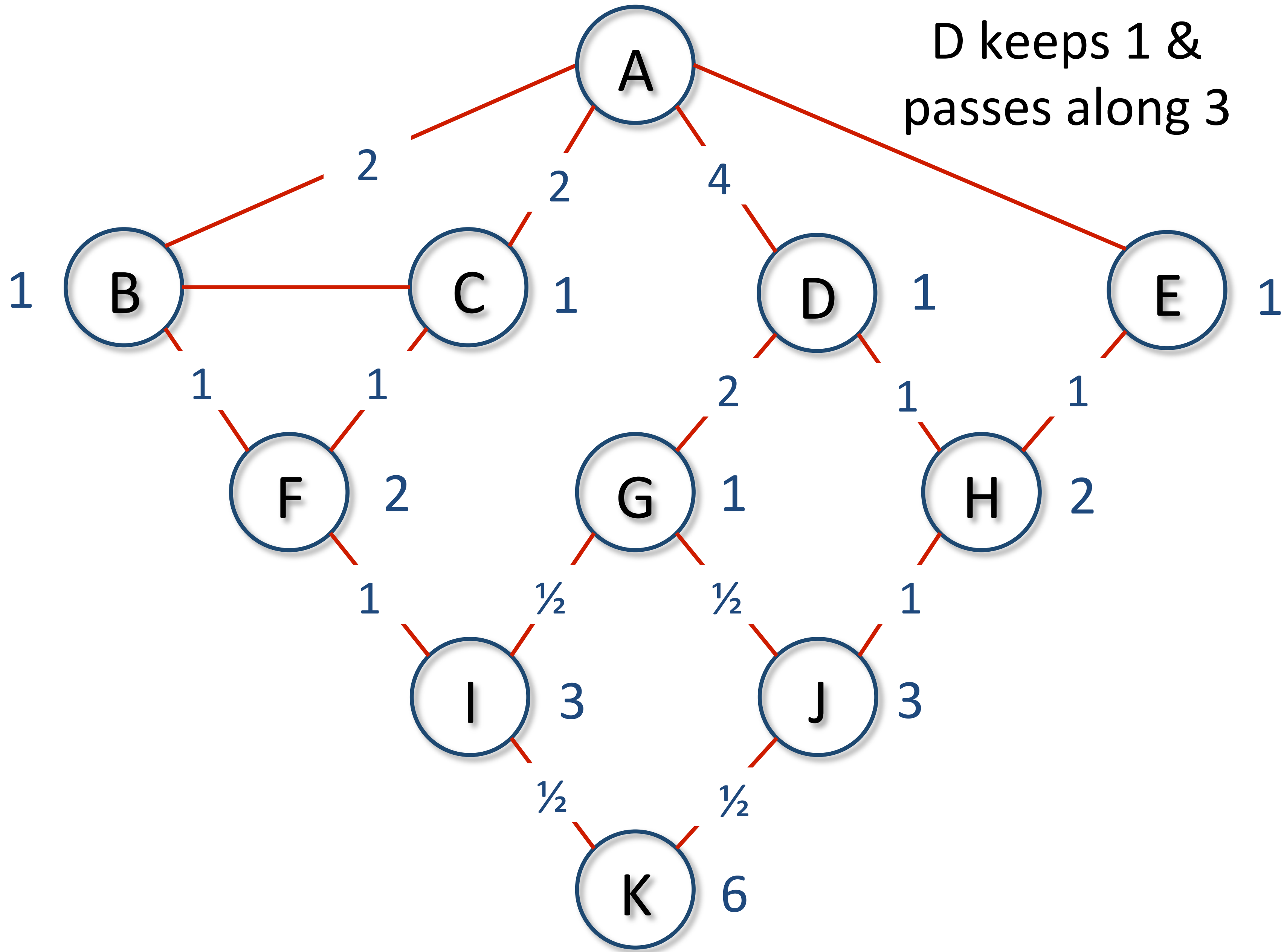
G keeps 1 unit & passes along 1 unit

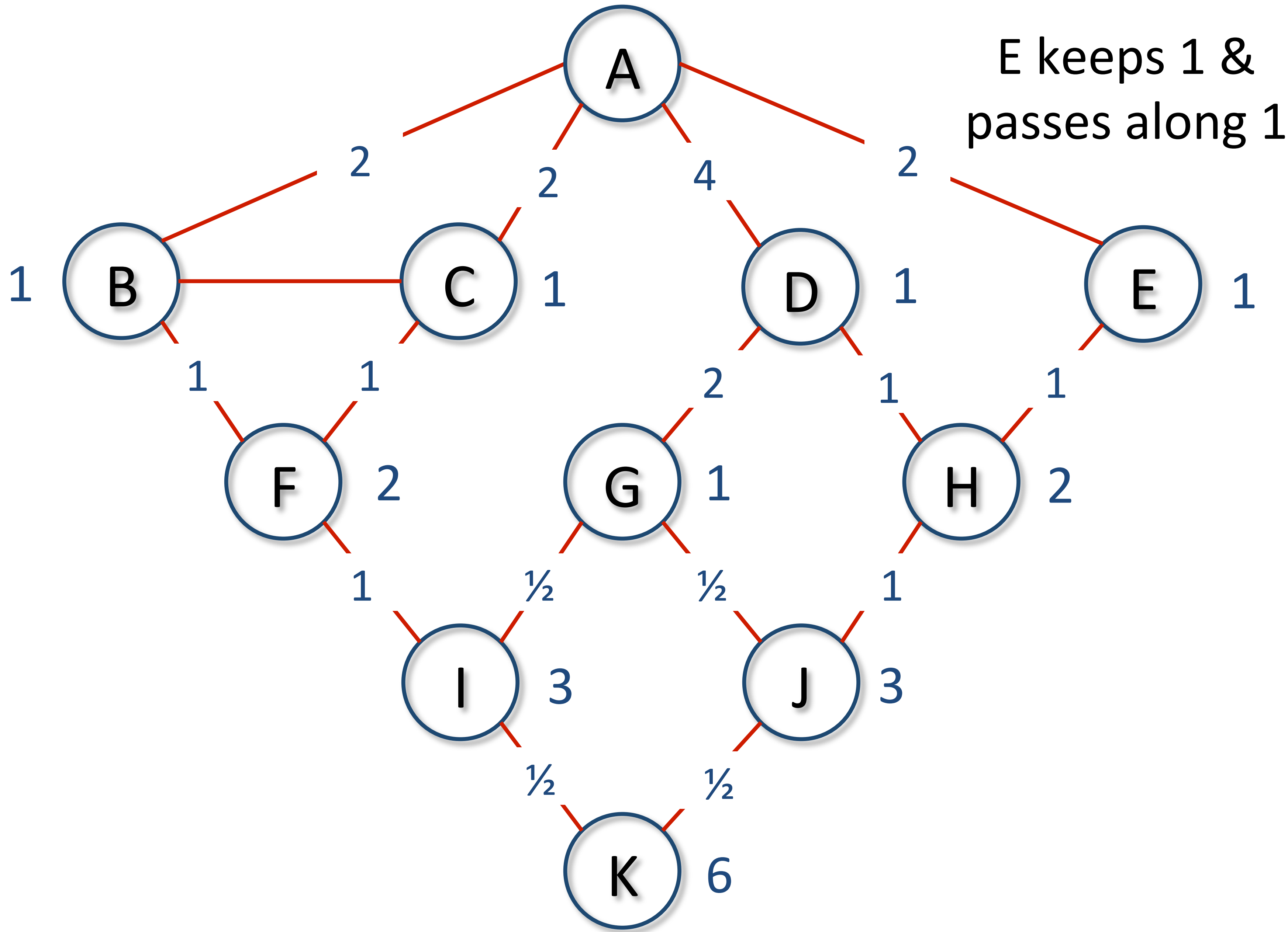


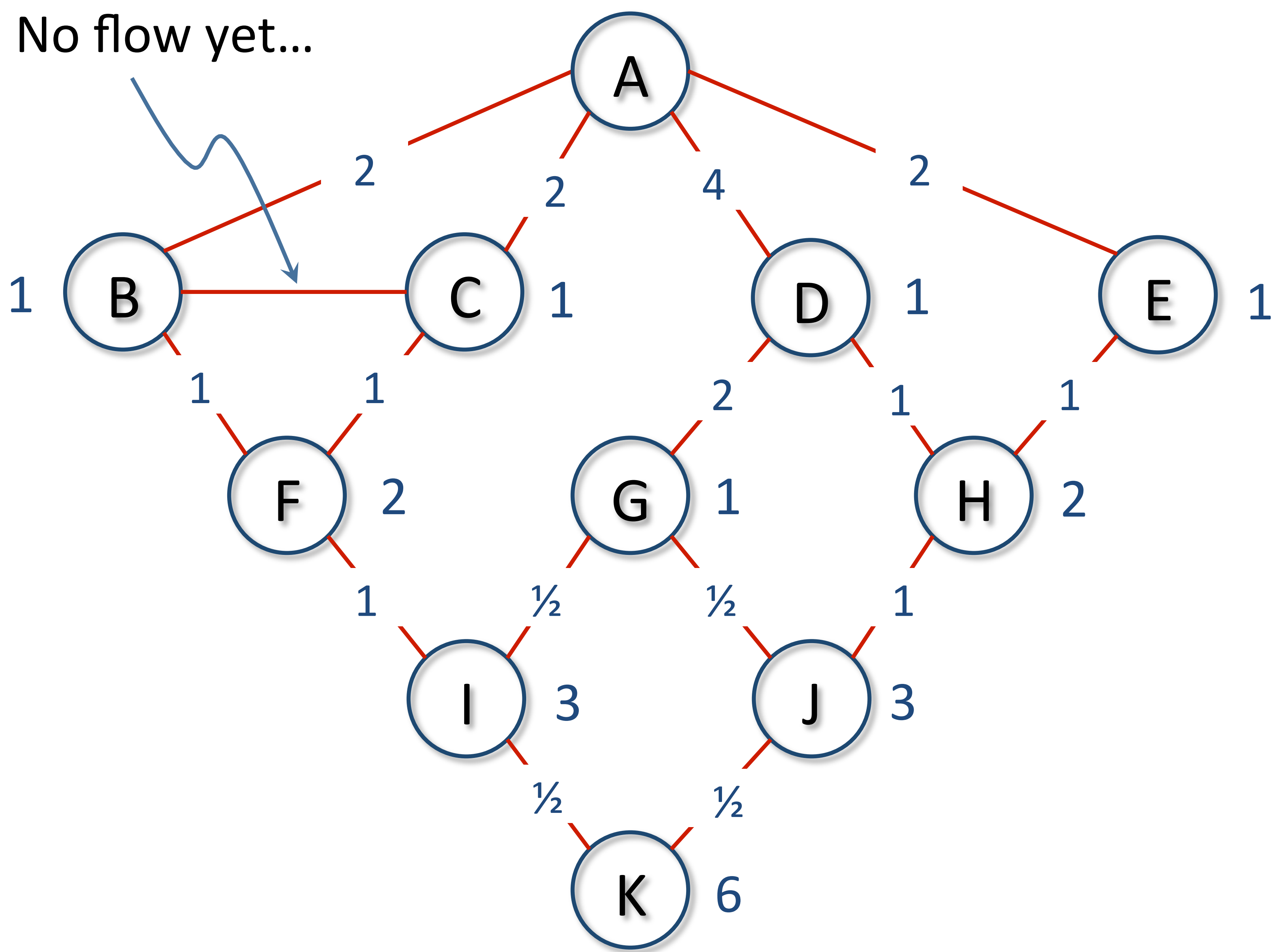
H keeps 1 unit & passes along 1 unit; equal, so divide evenly


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


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

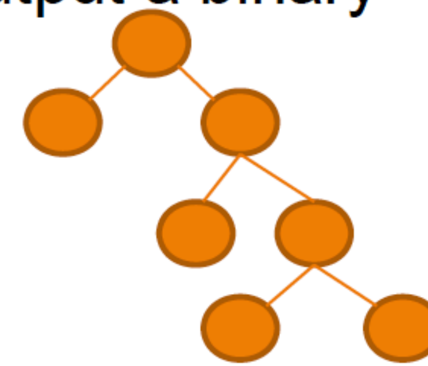

D keeps 1 &
passes along 3
A
B 1
C 1
D 1
E 1
F 2
G 1
H 2
I 3
J 3
K 6
2
2
4
1
1
2
1
1
1
½
½
1
½
½


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


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

Summary of Hierarchical Clustering

- Most hierarchical clustering algorithm output a binary tree
  - Each node has two children nodes
  - Might be highly imbalanced



- Agglomerative clustering can be very sensitive to the nodes processing order and merging criteria adopted.

- Divisive clustering is more stable, but generally more computationally expensive

Summary of Community Detection

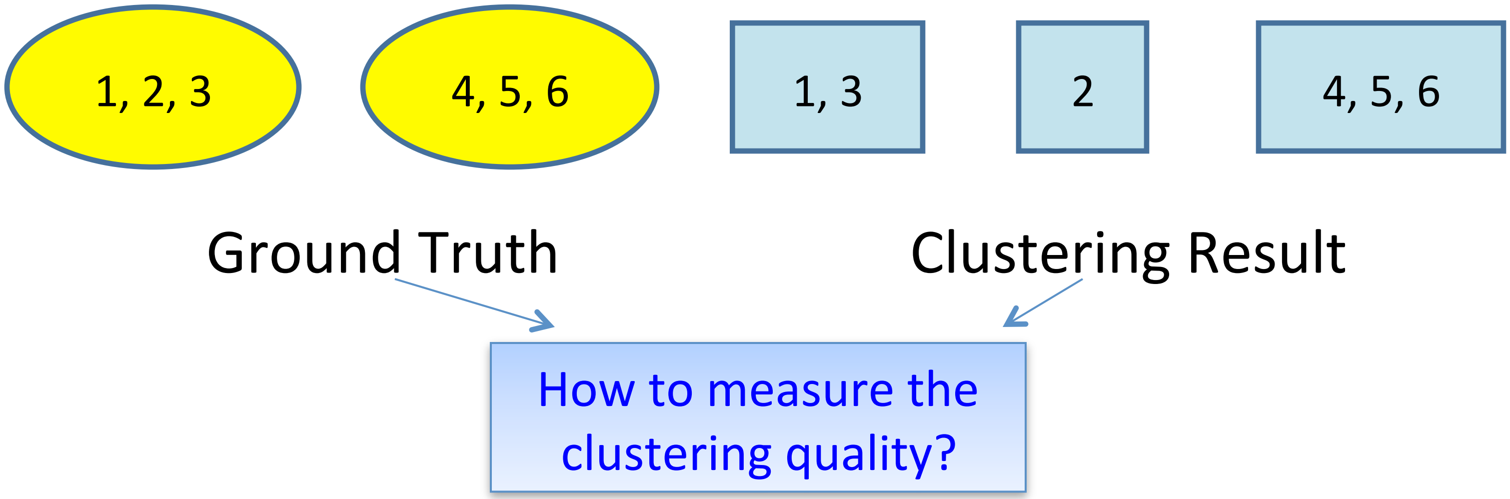
- Node-Centric Community Detection
  - *cliques, k-cliques, k-clubs*
- Group-Centric Community Detection
  - *quasi-cliques*
- Network-Centric Community Detection
  - *Clustering based on vertex similarity*
- Hierarchy-Centric Community Detection
  - *Divisive clustering*
  - *Agglomerative clustering*

COMMUNITY EVALUATION

Evaluating Community Detection (1)

- For groups with clear definitions
  - E.g., Cliques, k-cliques, k-clubs, quasi-cliques
  - Verify whether extracted communities satisfy the definition (e.g. if they are k-cliques etc.)
- For networks with ground truth information (e.g. we know already the communities)
  - Normalized mutual information
  - Accuracy of pairwise community memberships

Measuring a Clustering Result



- The number of communities after grouping can be different from the ground truth
- No clear community correspondence between clustering result and the ground truth

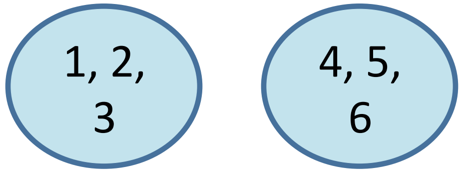
Accuracy of Pairwise Community Memberships

- Consider all the possible pairs of nodes and check whether they reside in the same community
- An error occurs *if*
  - Two nodes belonging to the same community are assigned to different communities after clustering
  - Two nodes belonging to different communities are assigned to the same community
- Construct a contingency table or confusion matrix

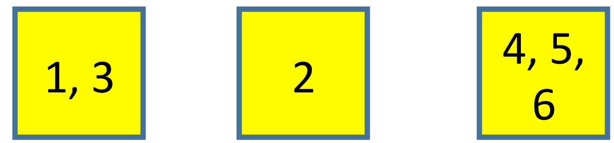
		Ground Truth	
		$C(v_i) = C(v_j)$	$C(v_i) \neq C(v_j)$
Clustering Result	$C(v_i) = C(v_j)$	a	b
	$C(v_i) \neq C(v_j)$	c	d

$$accuracy = \frac{a+d}{a+b+c+d} = \frac{a+d}{n(n-1)/2}$$

Accuracy Example



Ground Truth



Clustering Result

Pairs: (1,2) (1,3) (1,4) (1,5) (1,6) (2,3) (2,4) (2,5) (2,6) (3,4) (3,5) (3,6) (4,5) (4,6) (5,6)

		Ground Truth	
		$C(v_i) = C(v_j)$	$C(v_i) \neq C(v_j)$
Clustering Result	$C(v_i) = C(v_j)$	4	0
	$C(v_i) \neq C(v_j)$	2	9

Accuracy = (4+9)/ (4+2+9+0) = 13/15

Normalized Mutual Information

- Entropy: the information contained in a distribution

$$H(X) = \sum_{x \in X} p(x) \log p(x)$$

- Mutual Information: the shared information between two distributions

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_1(x) p_2(y)} \right)$$

- Normalized Mutual Information (between 0 and 1)

$$NMI(X;Y) = \frac{I(X;Y)}{\sqrt{H(X)H(Y)}}$$ or $$NMI(X;Y) = \frac{2I(X;Y)}{H(X)+H(Y)}$$

- Consider **a partition as a distribution** (probability of one node falling into one community), we can compute the matching between the clustering result and the ground truth

k^a, k^b = set of clusters generated by partitions π^a, π^b (e.g ground truth and output of clustering) *h* and ℓ are cluster indexes in partitions n_h^a dimension of cluster *h* in π^a, $n_{h,l}$ common nodes is two clusters of π^a, π^b

$$H(X) = \sum_{x \in X}$$

$\frac{n_h^a}{n}$ Is the ratio between the nodes in cluster h of partition a

$$H(\pi^a) = \sum_h^{k^{(a)}} \frac{n_h^a}{n} \log(\frac{n_h^a}{n})$$

$$H(\pi^b) = \sum_\ell^{k^{(b)}} \frac{n_\ell^b}{n} \log(\frac{n_\ell^b}{n})$$

$$I(X;Y) =$$

$\frac{n_{h,l}}{n}$ Are common nodes In cluster h of a and Cluster ℓ of b (sort of Jaccard between two Clusters)

$$I(\pi^a, \pi^b) = \sum_h \sum_\ell \frac{n_{h,\ell}}{n} \log\left(\frac{\frac{n_{h,\ell}}{n}}{\frac{n_h^a}{n}\frac{n_\ell^b}{n}}\right)$$

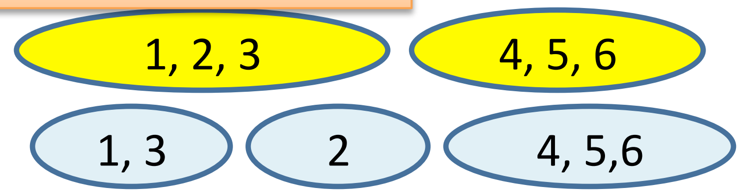
$$NMI(X;Y) =$$

$$NMI(\pi^a, \pi^b) = \frac{\sum_{h=1}^{k^{(a)}} \sum_{\ell=1}^{k^{(b)}} n_{h,\ell} \log\left(\frac{n \cdot n_{h,l}}{n_h^{(a)} \cdot n_\ell^{(b)}}\right)}{\sqrt{\left(\sum_{h=1}^{k^{(a)}} n_h^{(a)} \log \frac{n_h^a}{n}\right)\left(\sum_{\ell=1}^{k^{(b)}} n_\ell^{(b)} \log \frac{n_\ell^b}{n}\right)}}$$

NMI-Example

in a partition each node is assigned a number corresponding to its cluster

- Partition a: [1, 1, 1, 2, 2, 2]
- Partition b: [1, 2, 1, 3, 3, 3]



$$n = 6$$
$$k^{(a)} = 2$$
$$k^{(b)} = 3$$

k=# of clusters

	n_h^a
h=1	3
h=2	3

# of nodes in each cluster

	n_l^b
ℓ=1	2
ℓ=2	1
ℓ=3	3

$n_{h,l}$	**ℓ=1**	**ℓ=2**	**ℓ=3**
h=1	2	1	0
h=2	0	0	3

contingency table or confusion matrix

$$NMI(\pi^a, \pi^b) = \frac{\sum_{h=1}^{k^{(a)}} \sum_{\ell=1}^{k^{(b)}} n_{h,\ell} \log\left(\frac{n \cdot n_{h,l}}{n_h^{(a)} \cdot n_\ell^{(b)}}\right)}{\sqrt{\left(\sum_{h=1}^{k^{(a)}} n_h^{(a)} \log \frac{n_h^a}{n}\right)\left(\sum_{\ell=1}^{k^{(b)}} n_\ell^{(b)} \log \frac{n_\ell^b}{n}\right)}}$$

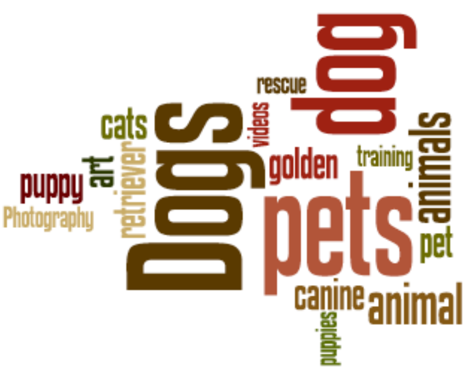
=0.8278

Reference: http://www.cse.ust.hk/~weikep/notes/NormalizedMI.m

Evaluation using Semantics

- For networks with semantics
  - Networks come with semantic or attribute information of nodes or connections
  - Human subjects can verify whether the extracted communities are coherent
- Evaluation is qualitative
- It is also intuitive and helps understand a community

An *animal* community



A *health* community

Next (and last) lesson

- Information Flow and maximization of Influence in social networks
- Sentiment analysis