Graph Theory and Social Networks - part 4

EE599: Social Network Systems

Keith M. Chugg
Fall 2014

USC University of Southern California

Overview

- Continuation of graph theory for social networks
  - Community detection (partitioning)
  - Betweenness computation
  - Homophily (segregation, polarization)
  - Examples

References

- Easley & Kleinberg, Ch 3-4
  - Focus on relationship to social nets with little math
- Jackson, Ch 2-3, 13.2
  - Social network focus with more formal math

Motivation

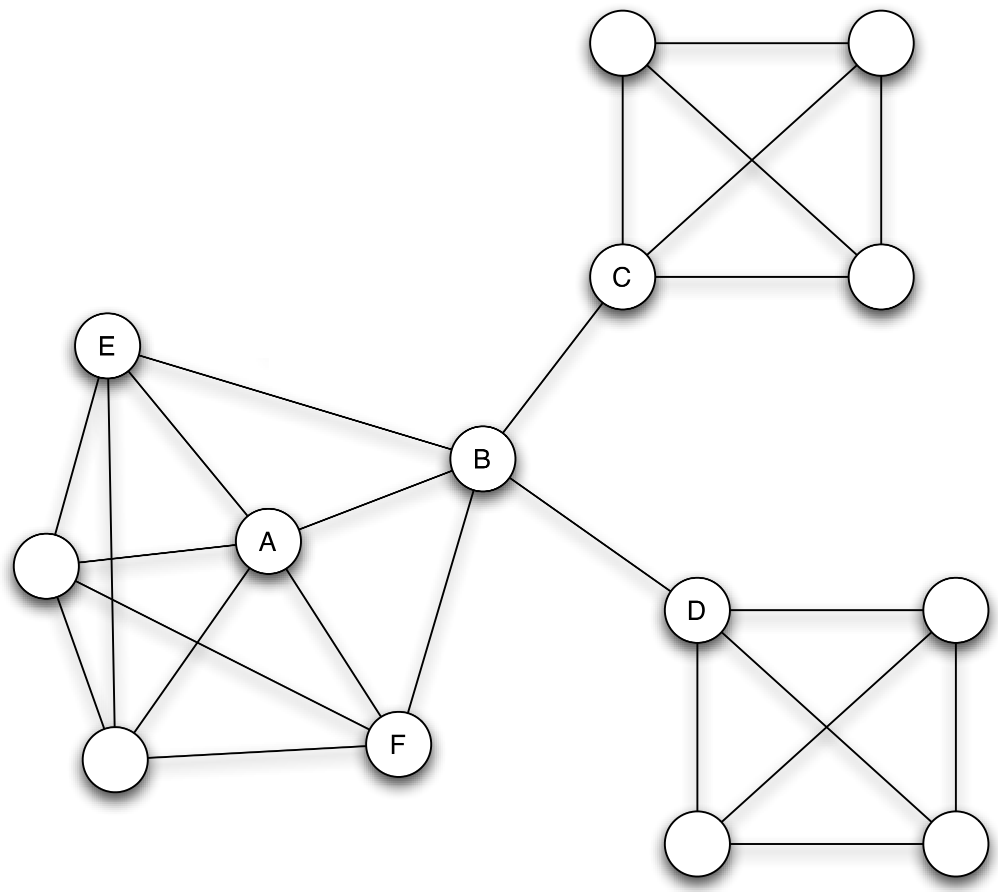
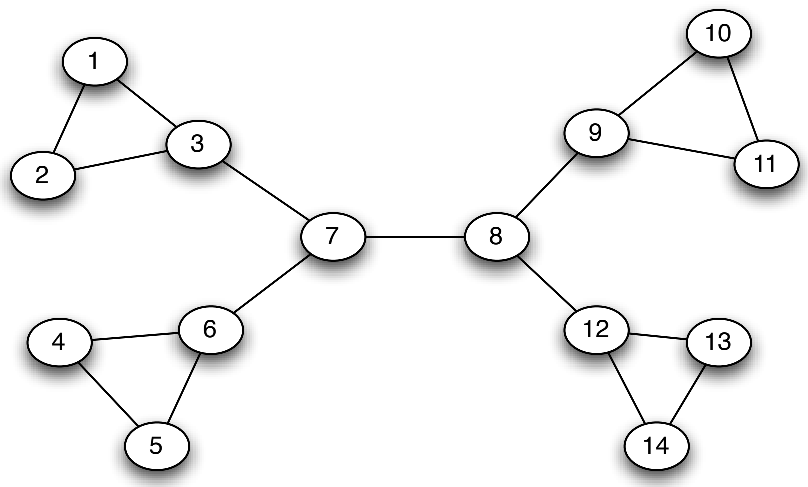


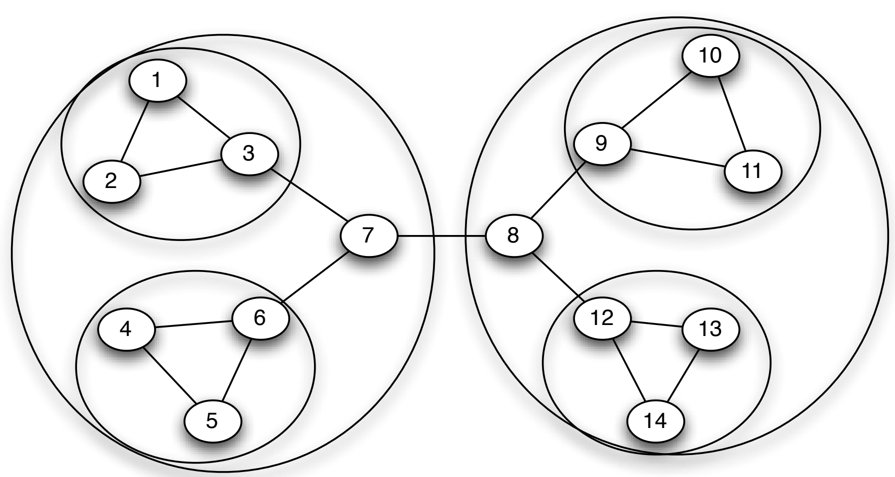
Figure 3.11: The contrast between densely-knit groups and boundary-spanning links is reflected in the different positions of nodes A and B in the underyling social network.
Easley & Kleinberg

- Not all nodes and edges are "equal"
  - A vs. B

Network Partitioning



(a) *A sample network*



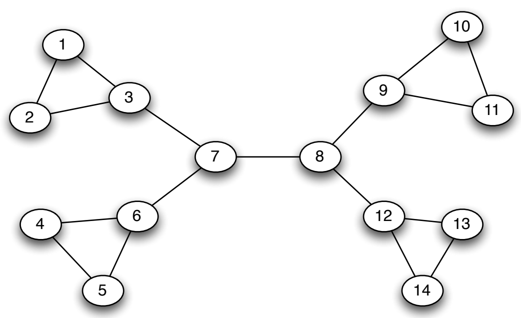
(b) *Tightly-knit regions and their nested structure*

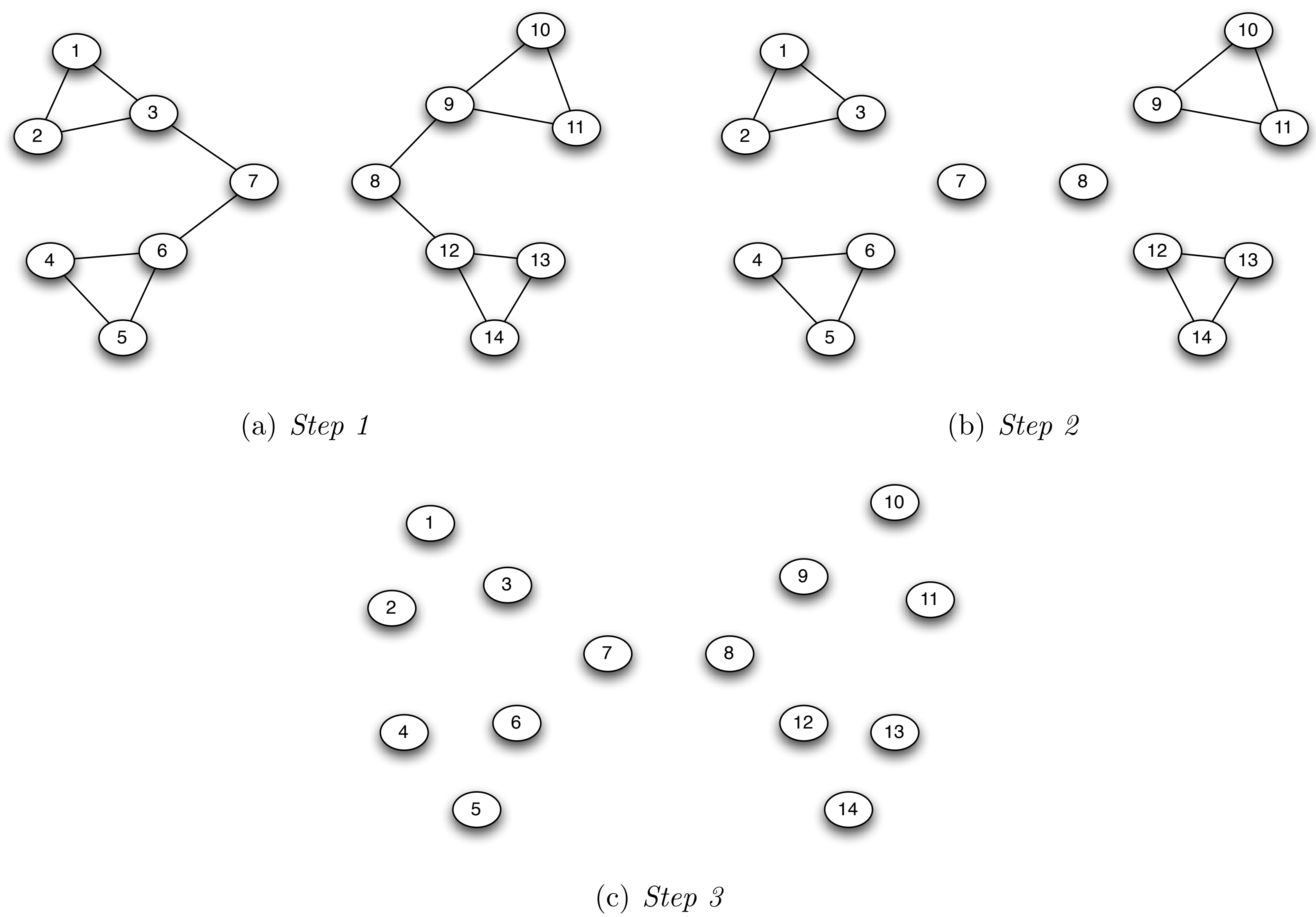
Figure 3.14: In many networks, there are tightly-knit regions that are intuitively apparent, and they can even display a *nested* structure, with smaller regions nesting inside larger ones.

Easley & Kleinberg

- How to identify tightly knight regions in a social network from the graph structure?
  - Assume we have a method of identifying the most "central" edges
  - Remove these edges to break the graph into components
  - Repeat this process on the components as they arise
- Girvan-Newman Algorithm

Network Partitioning





Easley & Kleinberg

Figure 3.16: The steps of the Girvan-Newman method on the network from Figure 3.14(a).

Network Partitioning

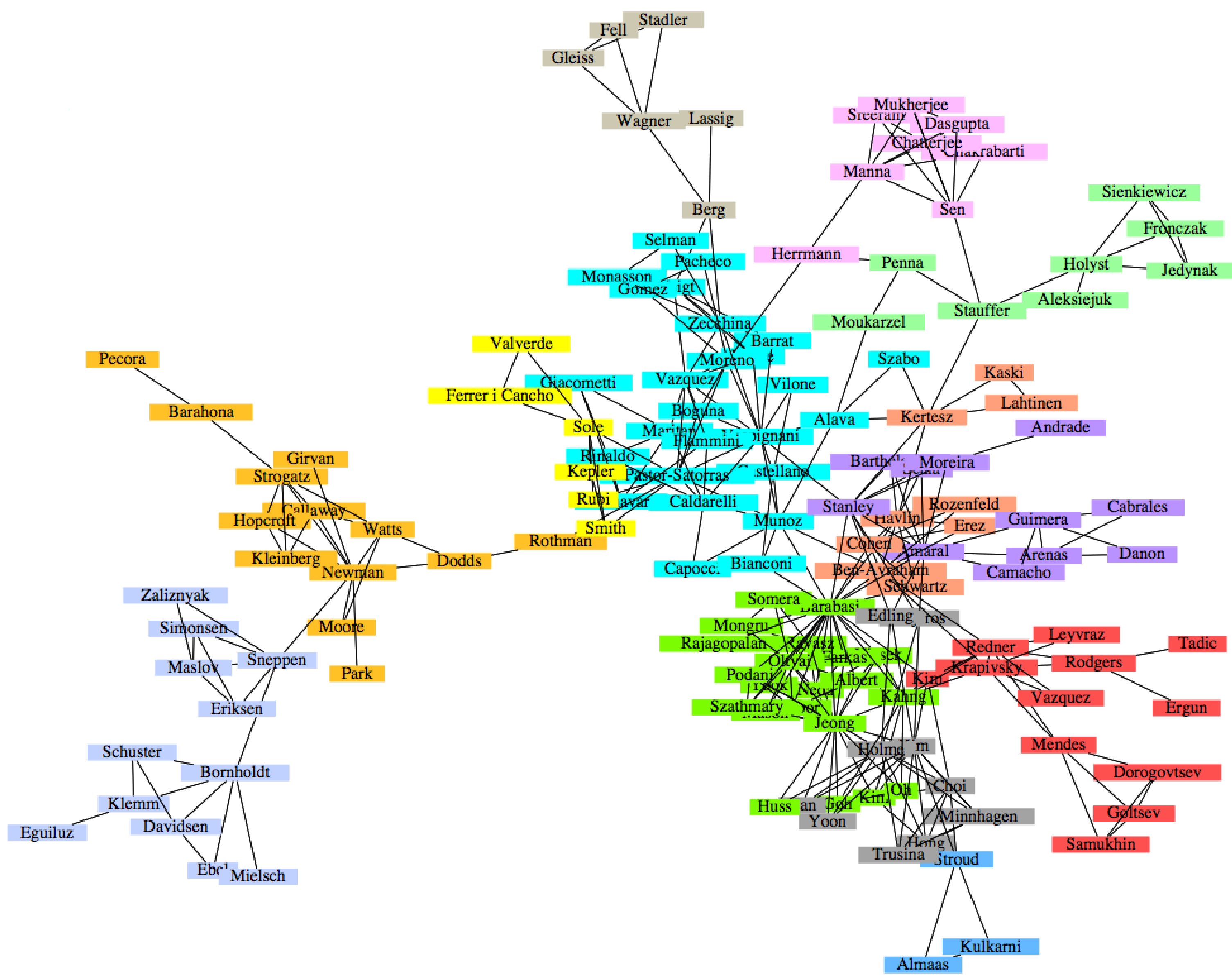


Figure 3.12: A co-authorship network of physicists and applied mathematicians working on networks [322]. Within this professional community, more tightly-knit subgroups are evident from the network structure.

Easley & Kleinberg

Network Partitioning

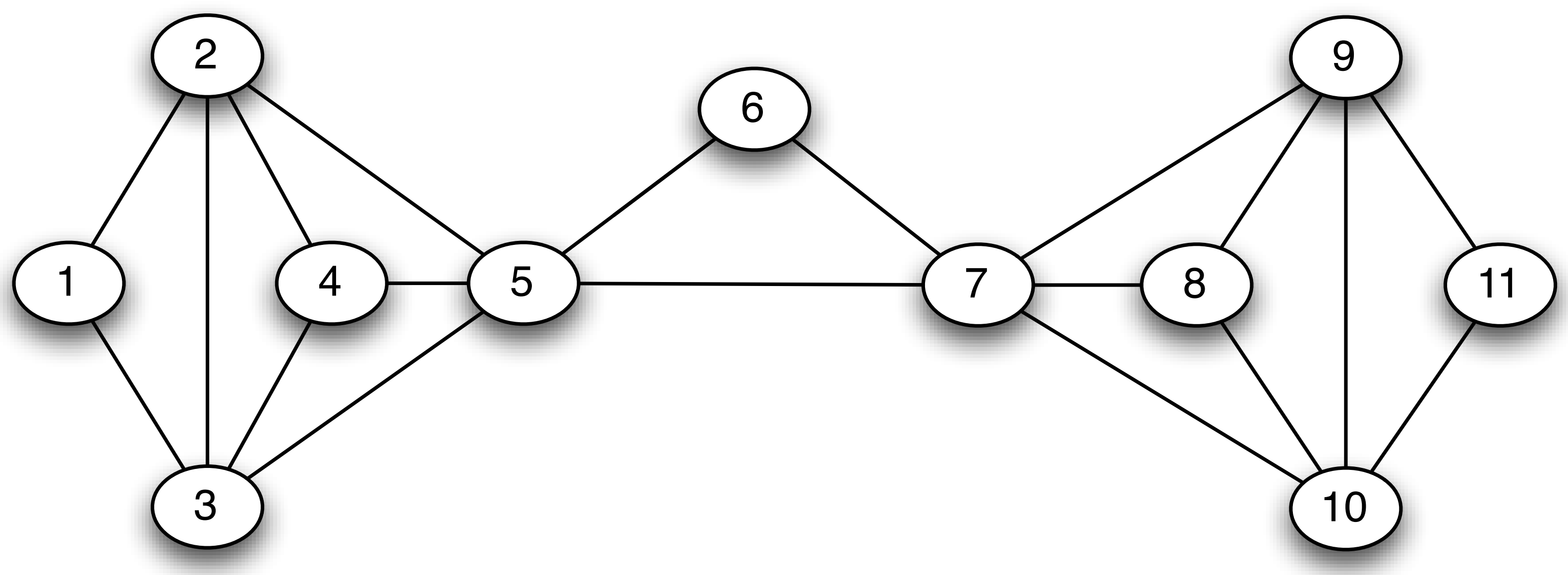


Figure 3.15: A network can display tightly-knit regions even when there are no bridges or local bridges along which to separate it.

Easley & Kleinberg

Network Partitioning

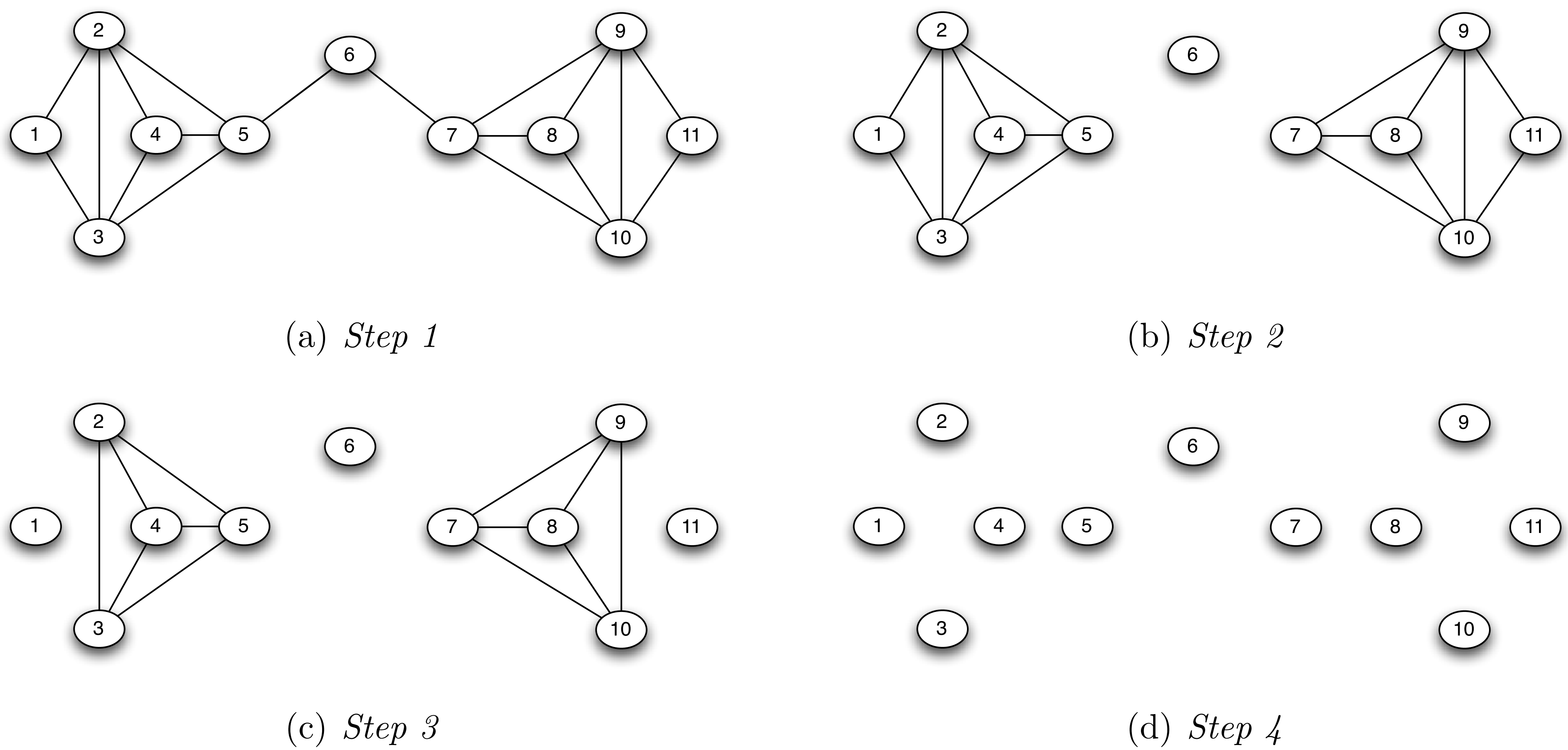


Figure 3.17: The steps of the Girvan-Newman method on the network from Figure 3.15.

Easley & Kleinberg

Centrality Measures (nodes)

- Betweenness Centrality:
    - Fraction of shortest paths in the network that pass through node i

$$B(i) = \sum_{(j,k), i \notin \{j,k\}} \frac{P_i(j,k)}{P(j,k)} = \frac{\text{number of shortest paths between j \& k, passing through i}}{\text{number of shortest paths between j \& k}}$$

Often normalized by: $\binom{N-1}{2}$ (number of pairs excluding node i)

Centrality Example

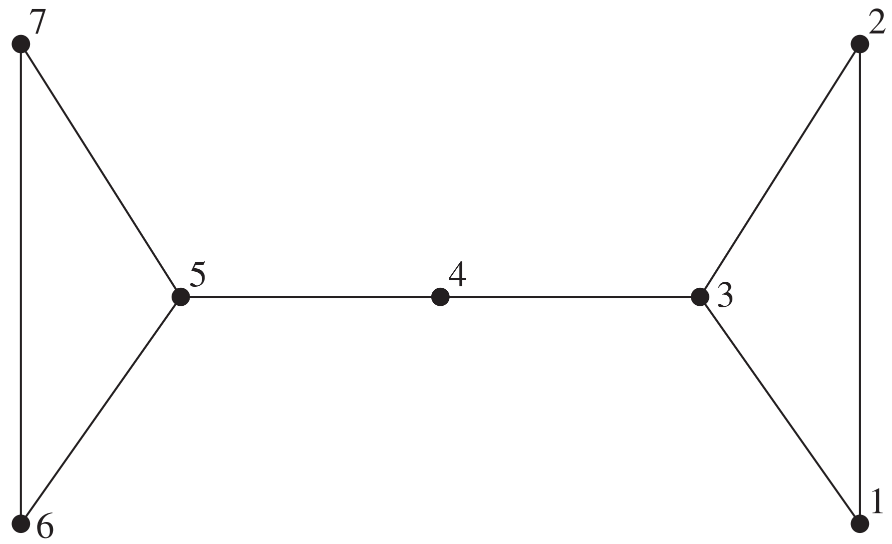


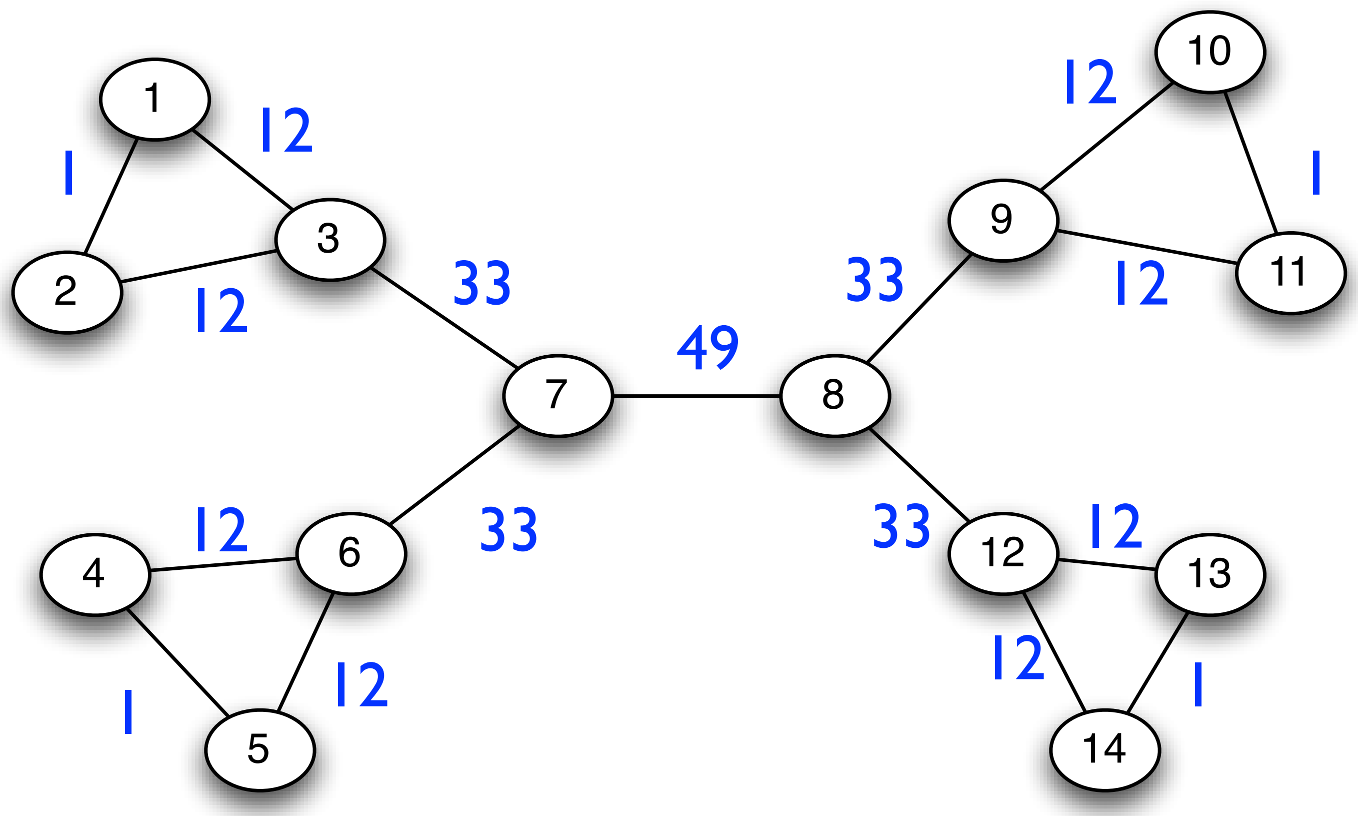
FIGURE 2.13 A central node with low degree centrality.

TABLE 2.1
Centrality comparisons for Figure 2.13

Measure of centrality	Nodes 1, 2, 6, and 7	Nodes 3 and 5	Node 4
Degree (and Katz prestige P^K)	.33	.50	.33
Closeness	.40	.55	.60
Decay centrality ($\delta = .5$)	1.5	2.0	2.0
Decay centrality ($\delta = .75$)	3.1	3.7	3.8
Decay centrality ($\delta = .25$)	.59	.84	.75
★ Betweenness	.0	.53	.60
Eigenvector centrality	.47	.63	.54
Katz prestige-2 P^{K2}, $a = 1/3$	3.1	4.3	3.5
Bonacich centrality $b = 1/3$, $a = 1$	9.4	13.0	11.0
Bonacich centrality $b = 1/4$, $a = 1$	4.9	6.8	5.4

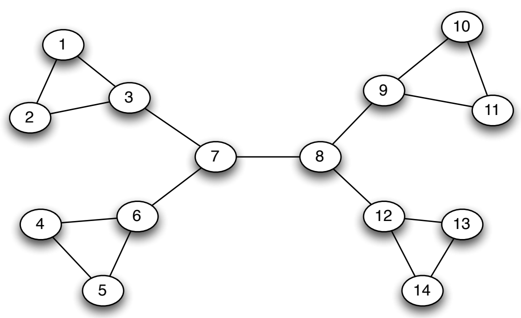
Jackson

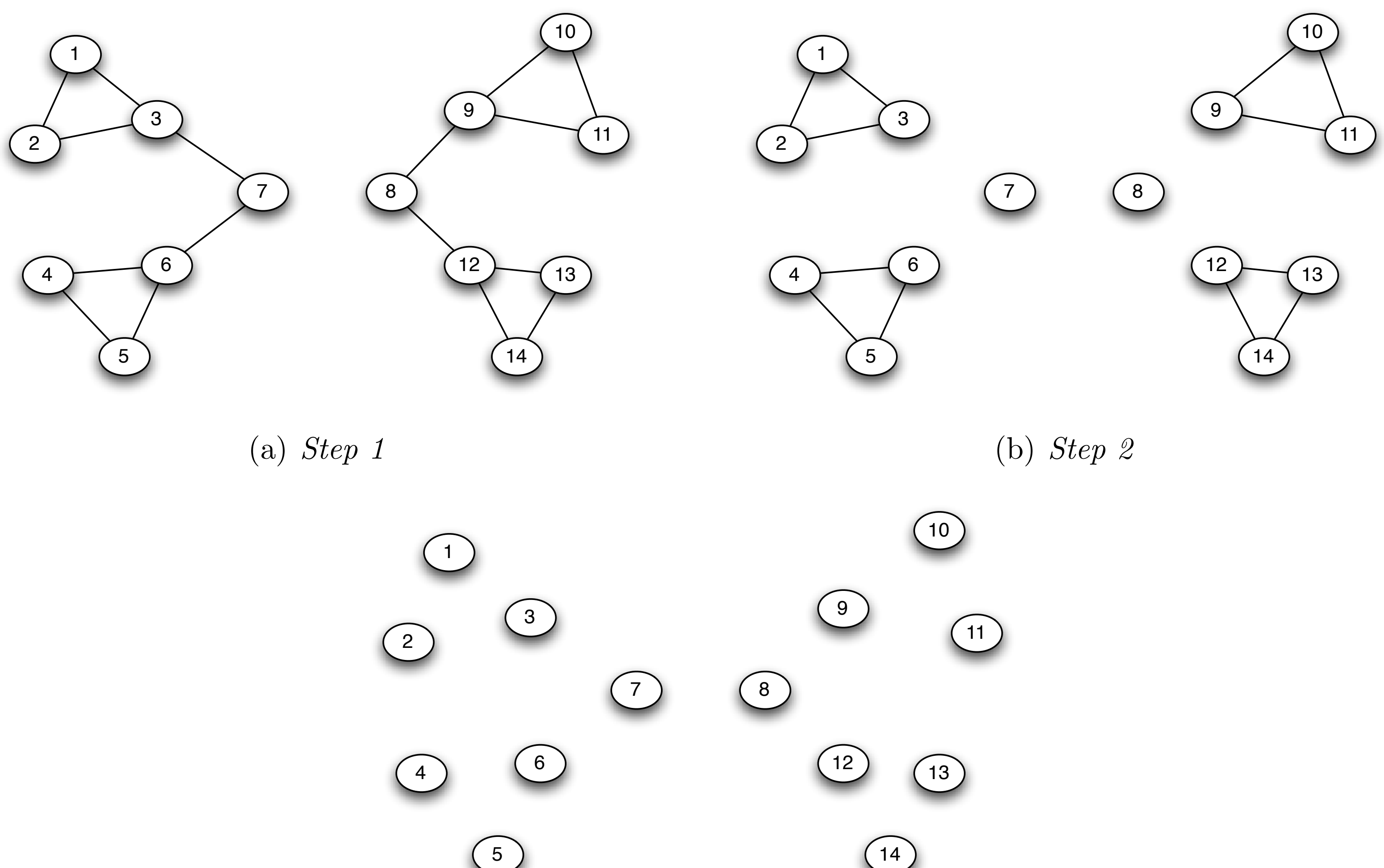
Computing Betweenness (edge)



Easley & Kleinberg

Network Partitioning





(a) *Step 1*

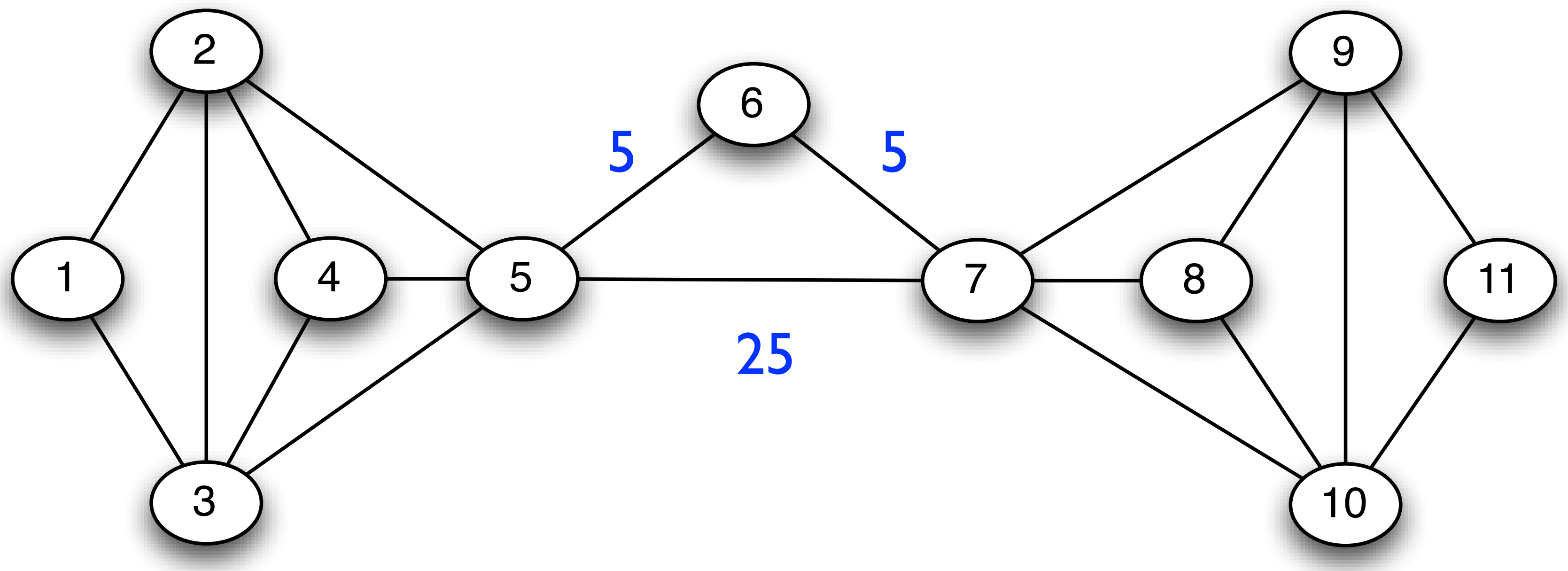
(b) *Step 2*

(c) *Step 3*

Figure 3.16: The steps of the Girvan-Newman method on the network from Figure 3.14(a).

Easley & Kleinberg

Computing Betweenness (edge)



Easley & Kleinberg

Network Partitioning

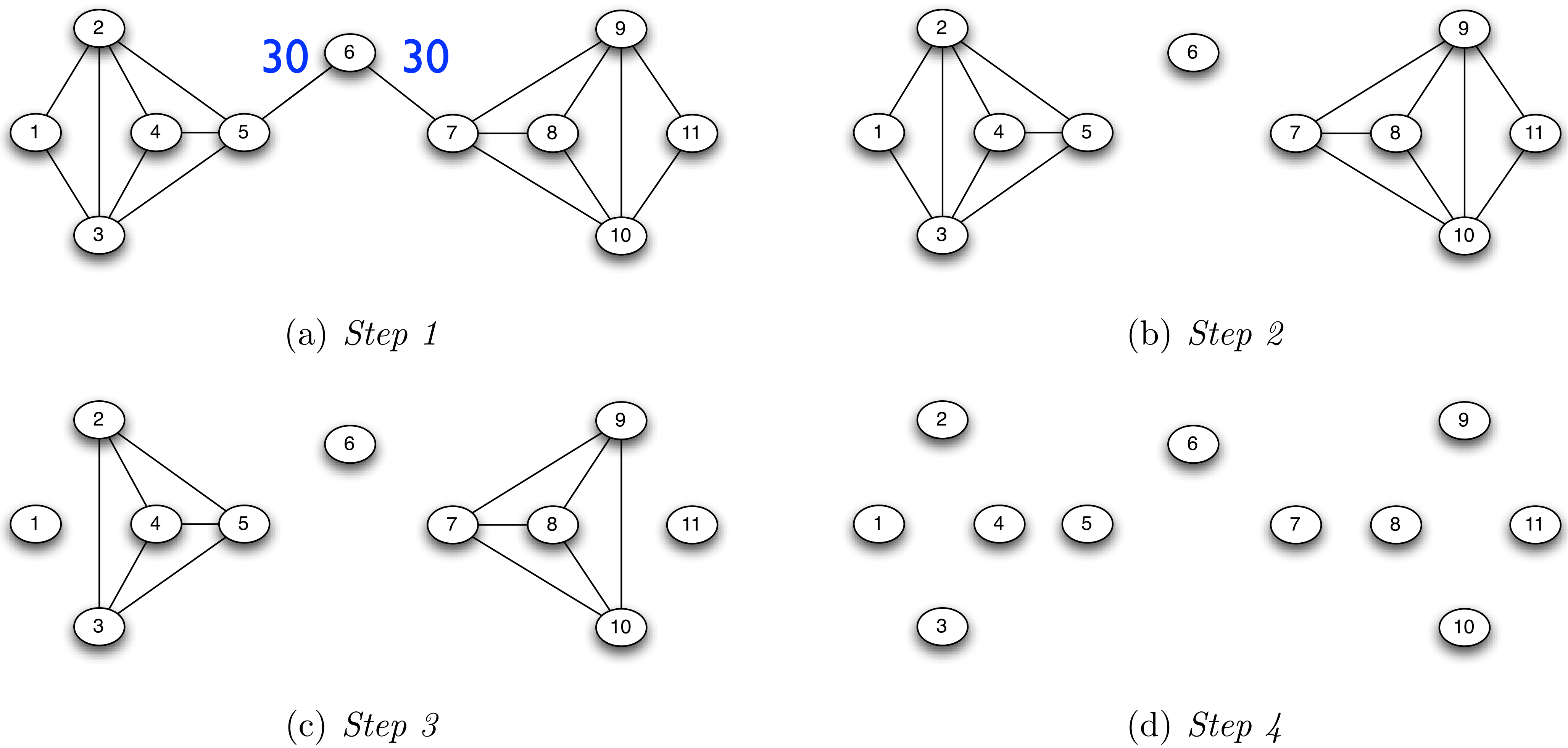
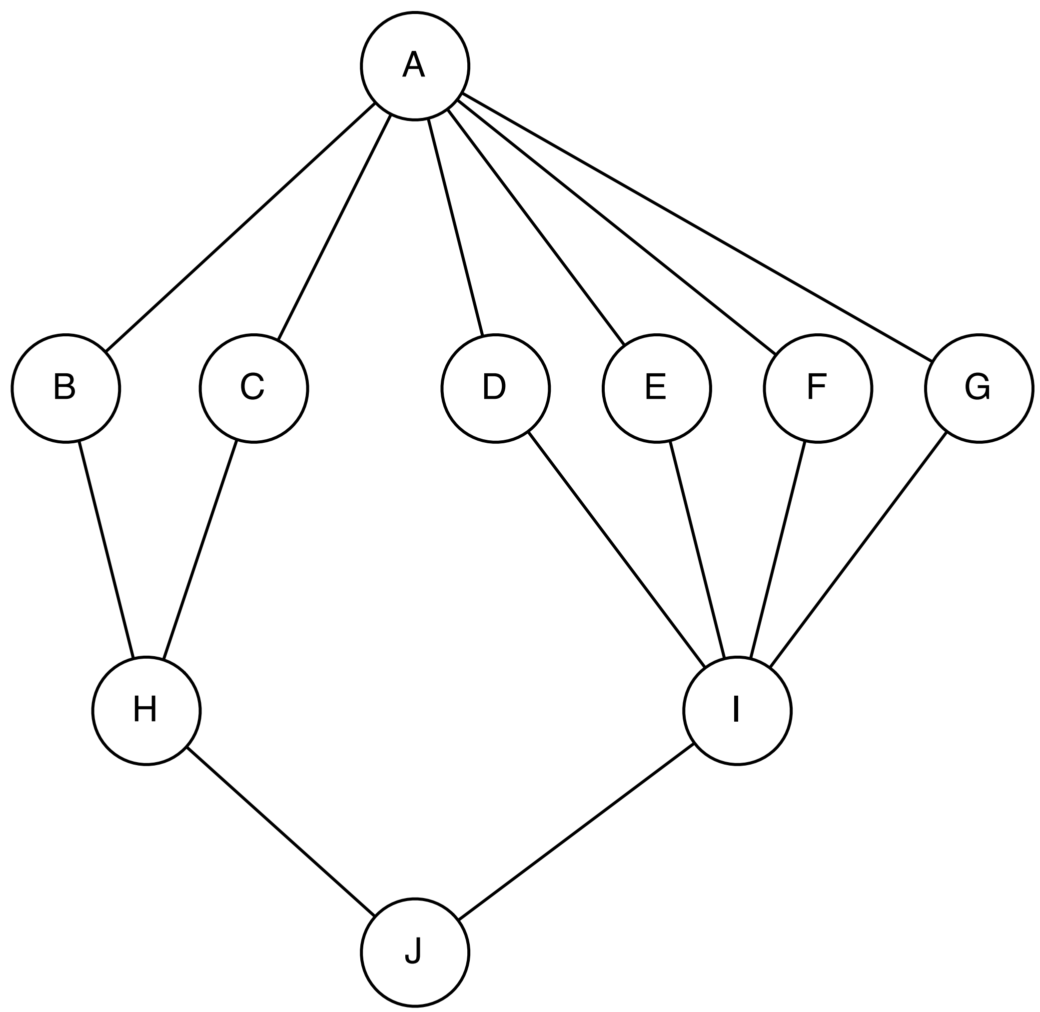


Figure 3.17: The steps of the Girvan-Newman method on the network from Figure 3.15.

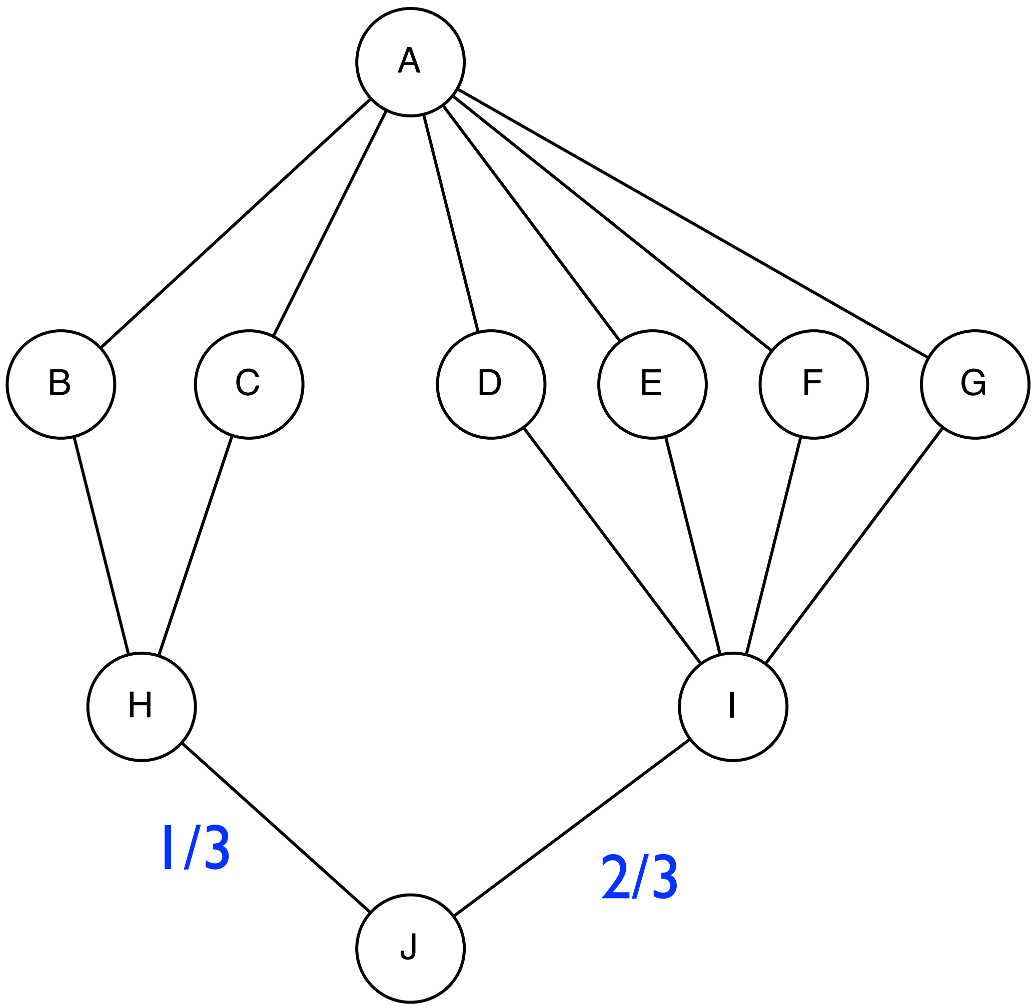
Easley & Kleinberg

Computing Betweenness



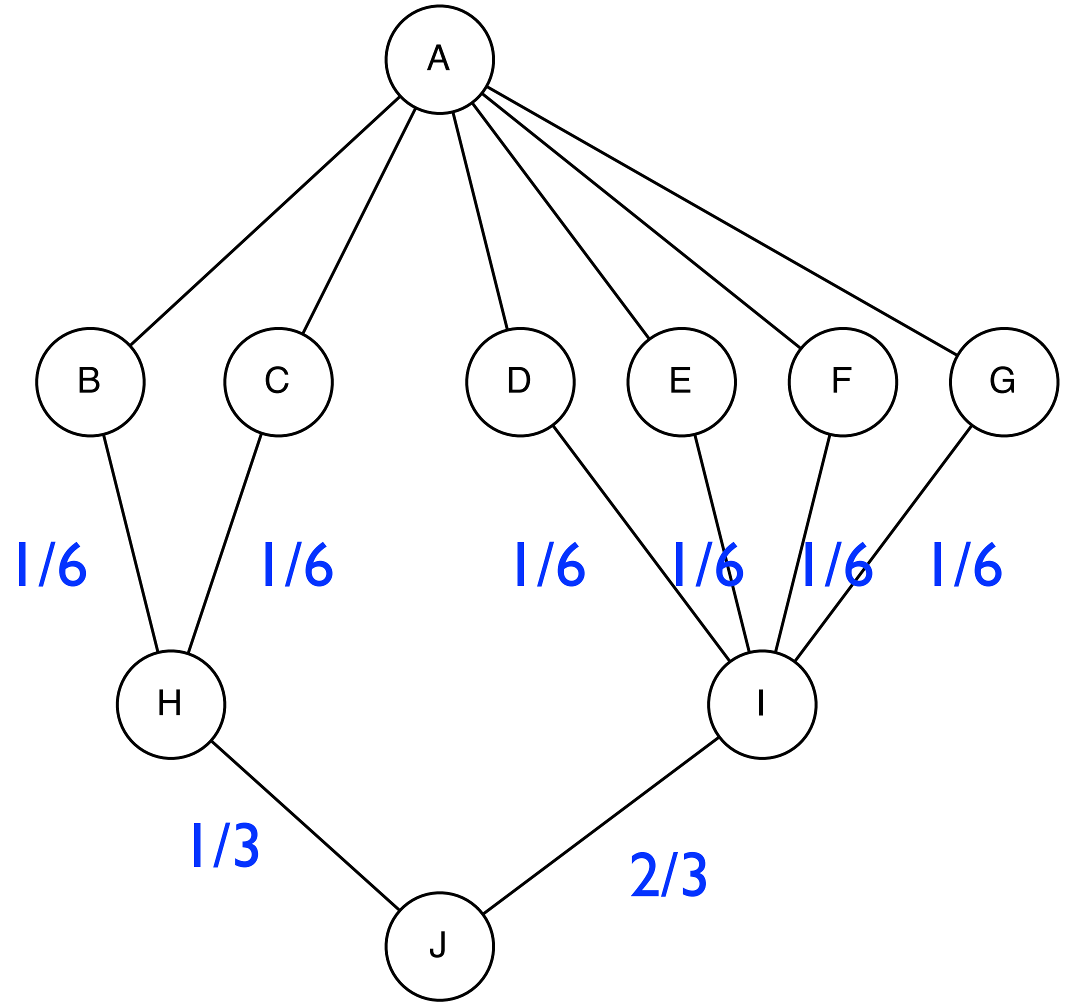
- What fraction of geodesics from A to J go thru each edge?

Computing Betweenness



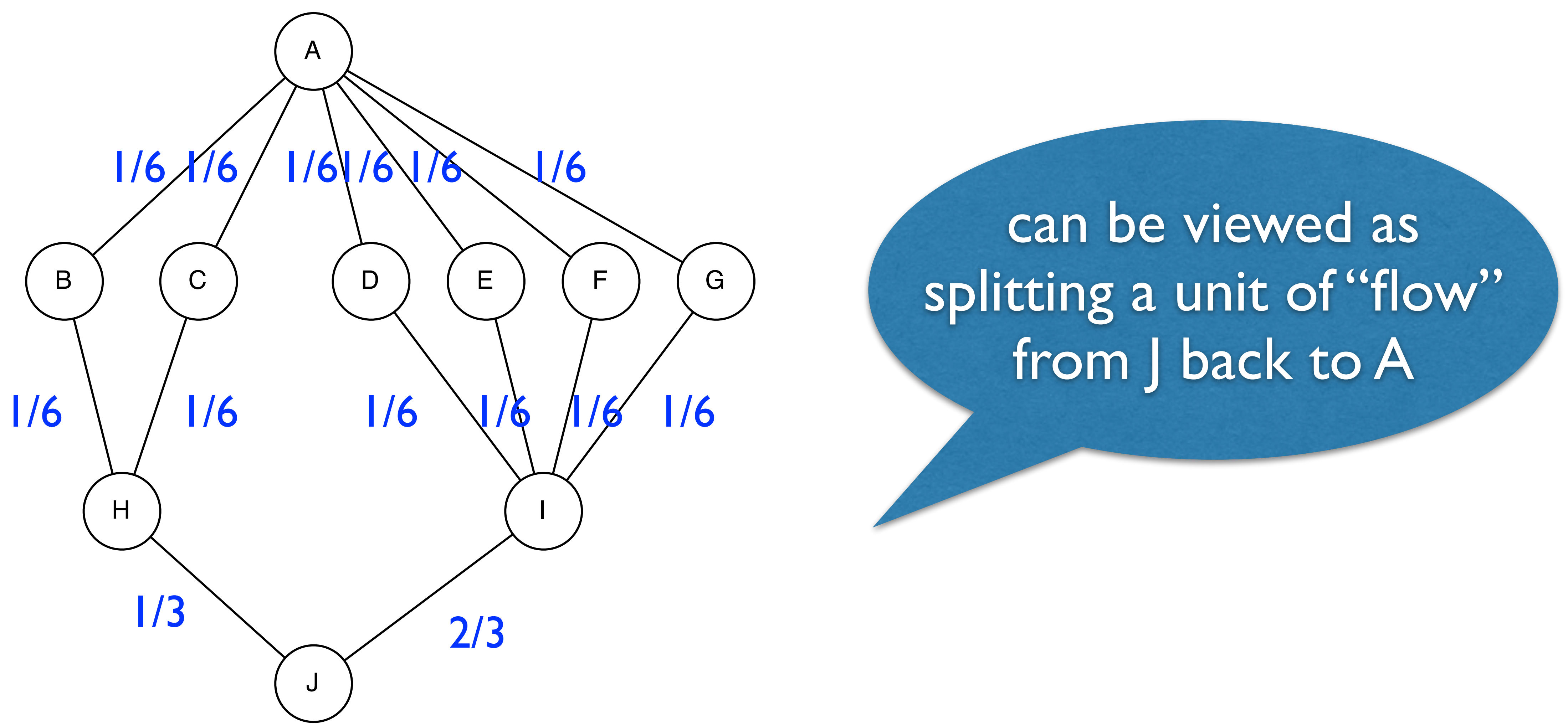
- What fraction of geodesics from A to J go thru each edge?

Computing Betweenness



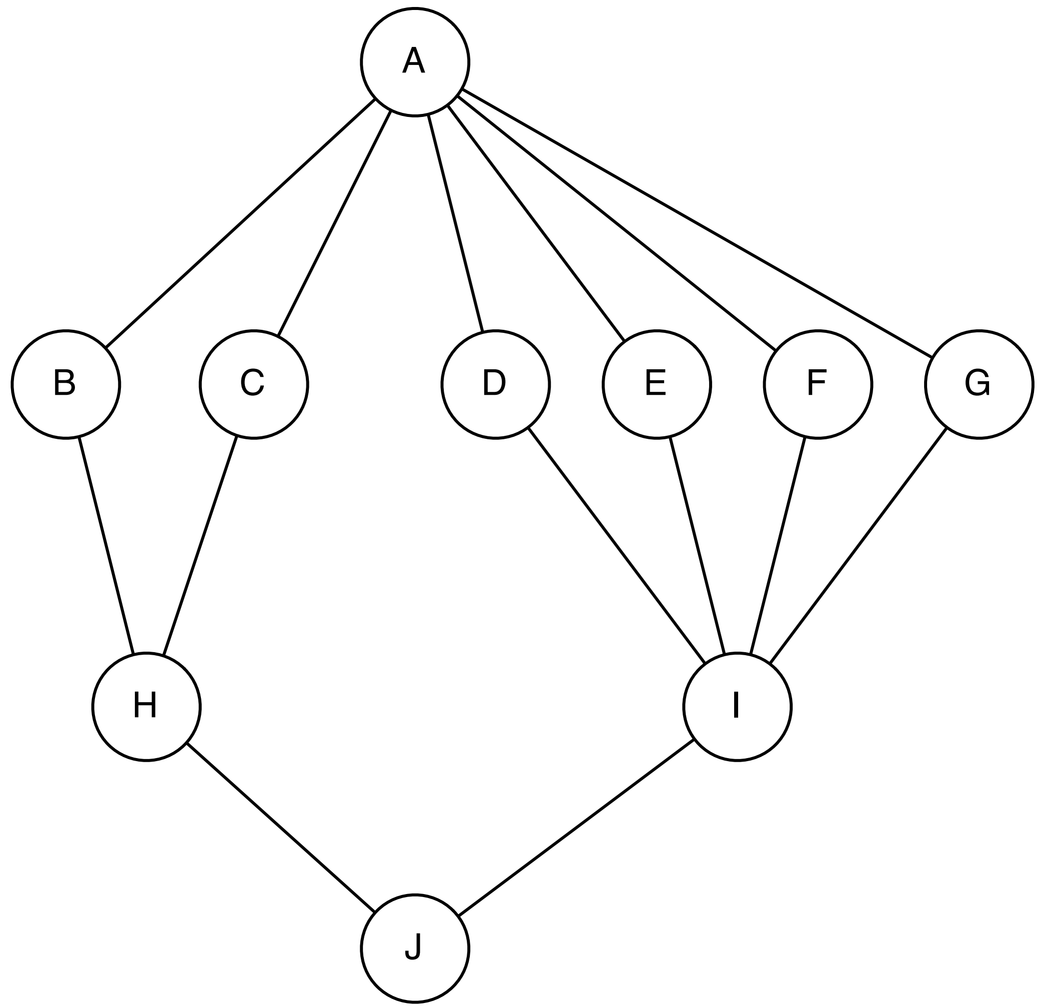
- What fraction of geodesics from A to J go thru each edge?

Computing Betweenness



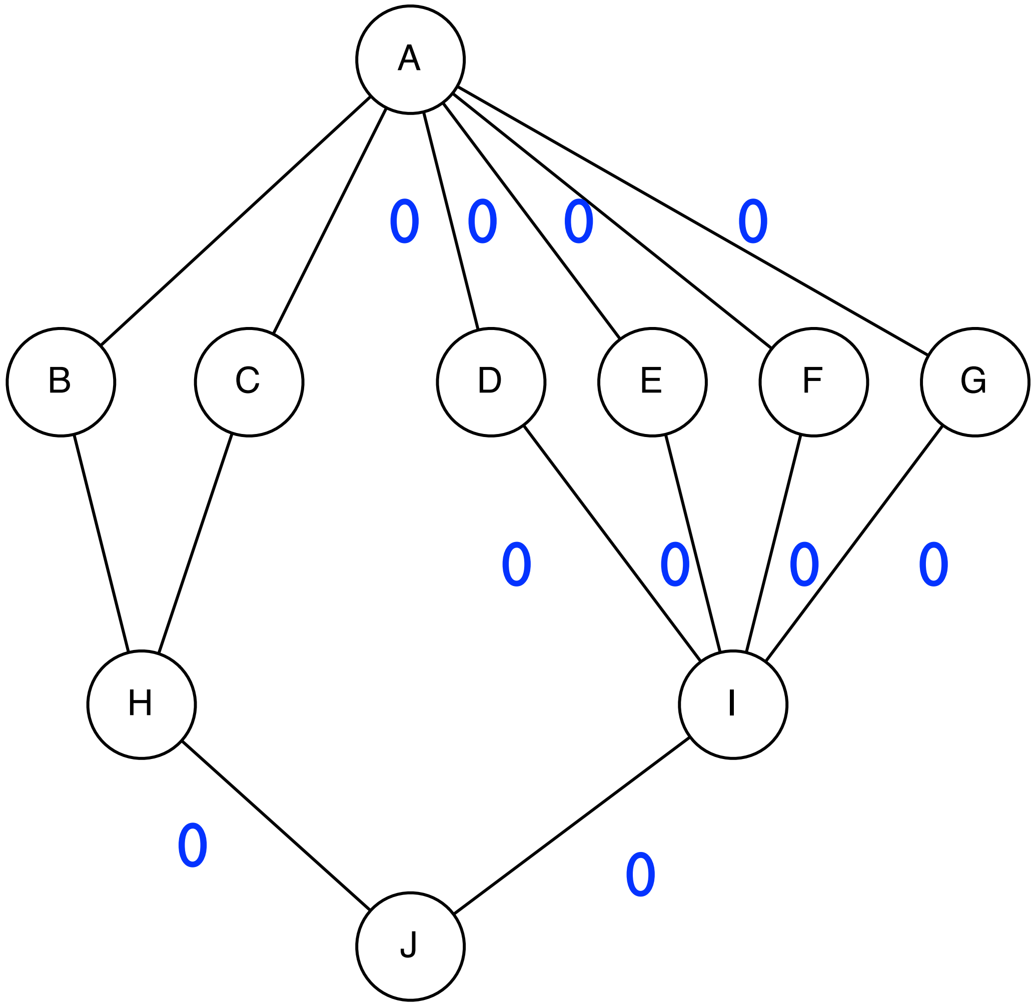
- What fraction of geodesics from A to J go thru each edge?

Computing Betweenness



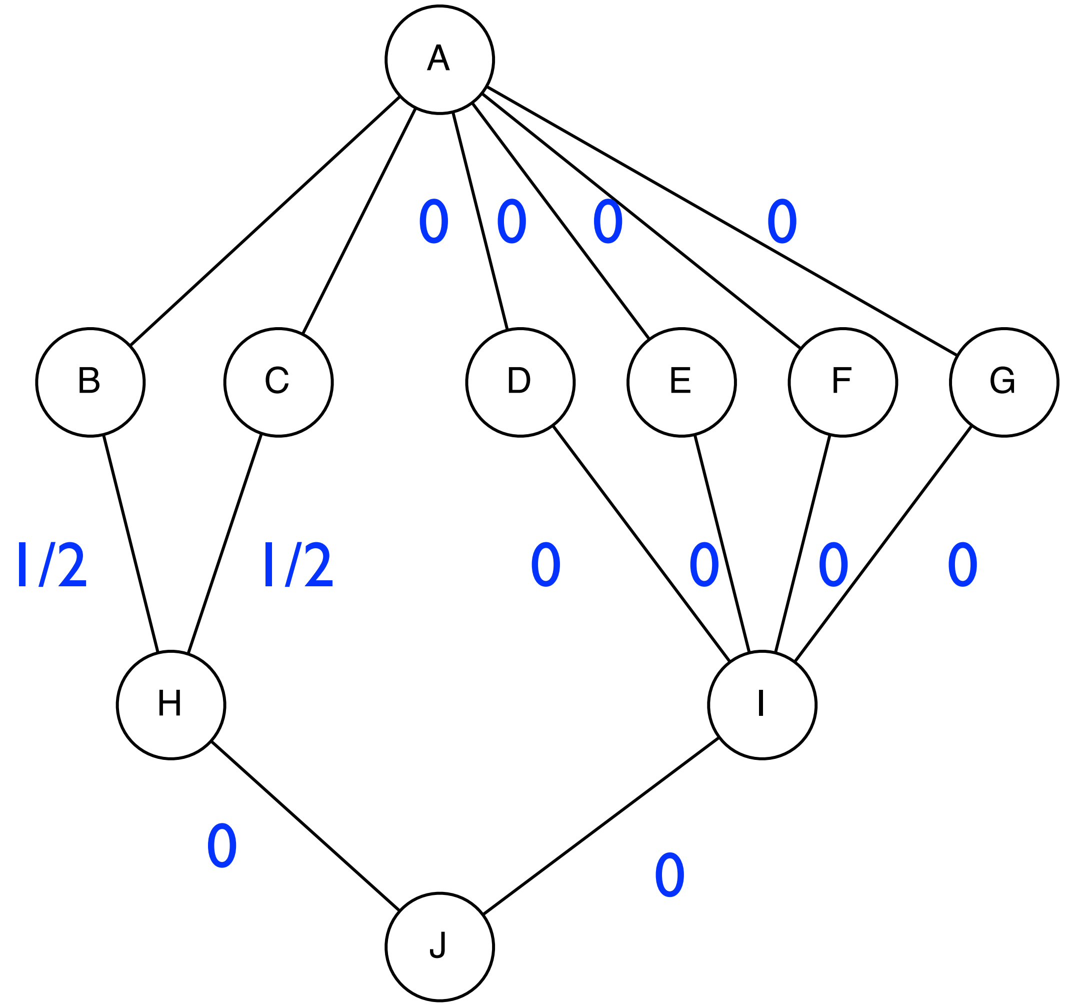
- What fraction of geodesics from A to H go thru each edge?

Computing Betweenness



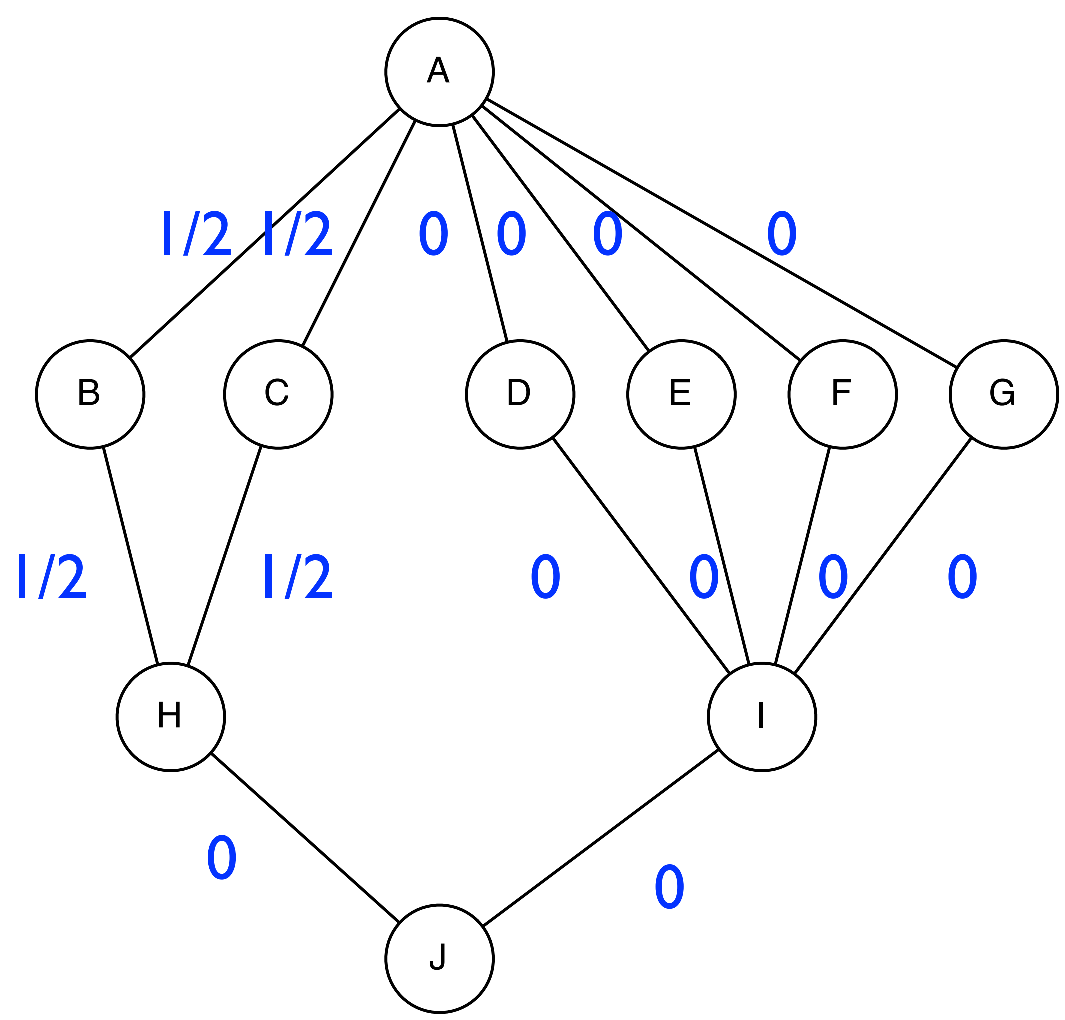
- What fraction of geodesics from A to H go thru each edge?

Computing Betweenness



- What fraction of geodesics from A to H go thru each edge?

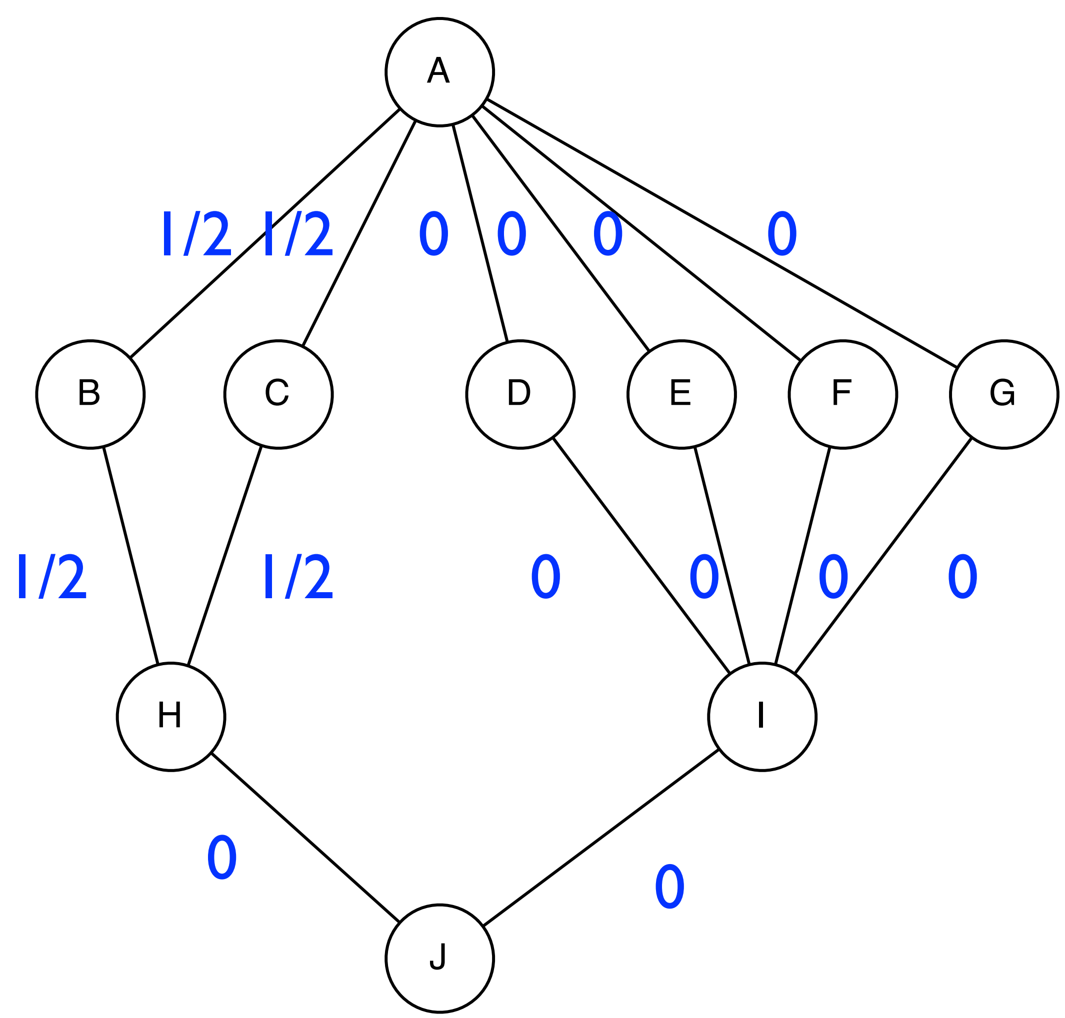
Computing Betweenness



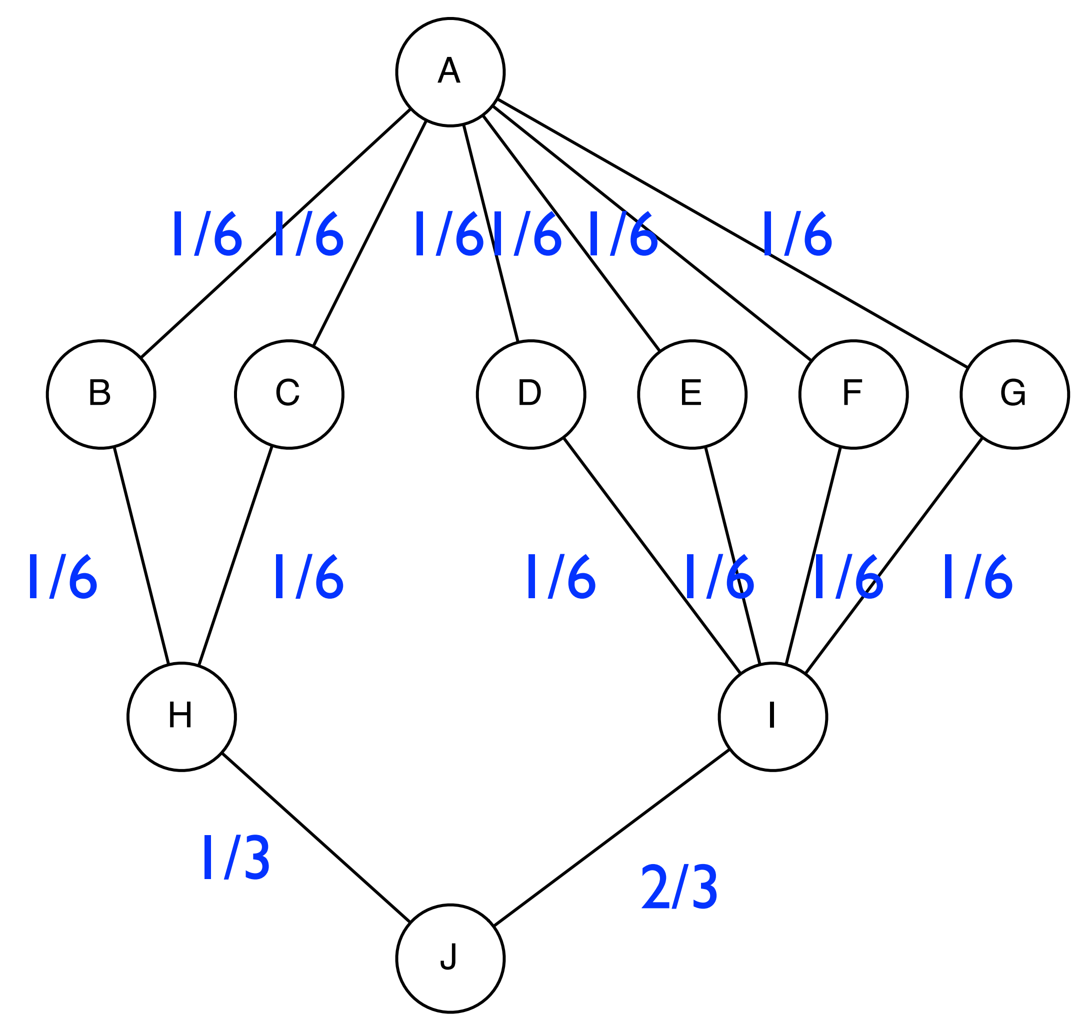
- What fraction of geodesics from A to J go thru each edge?

Computing Betweenness

A~H betweenness contribution



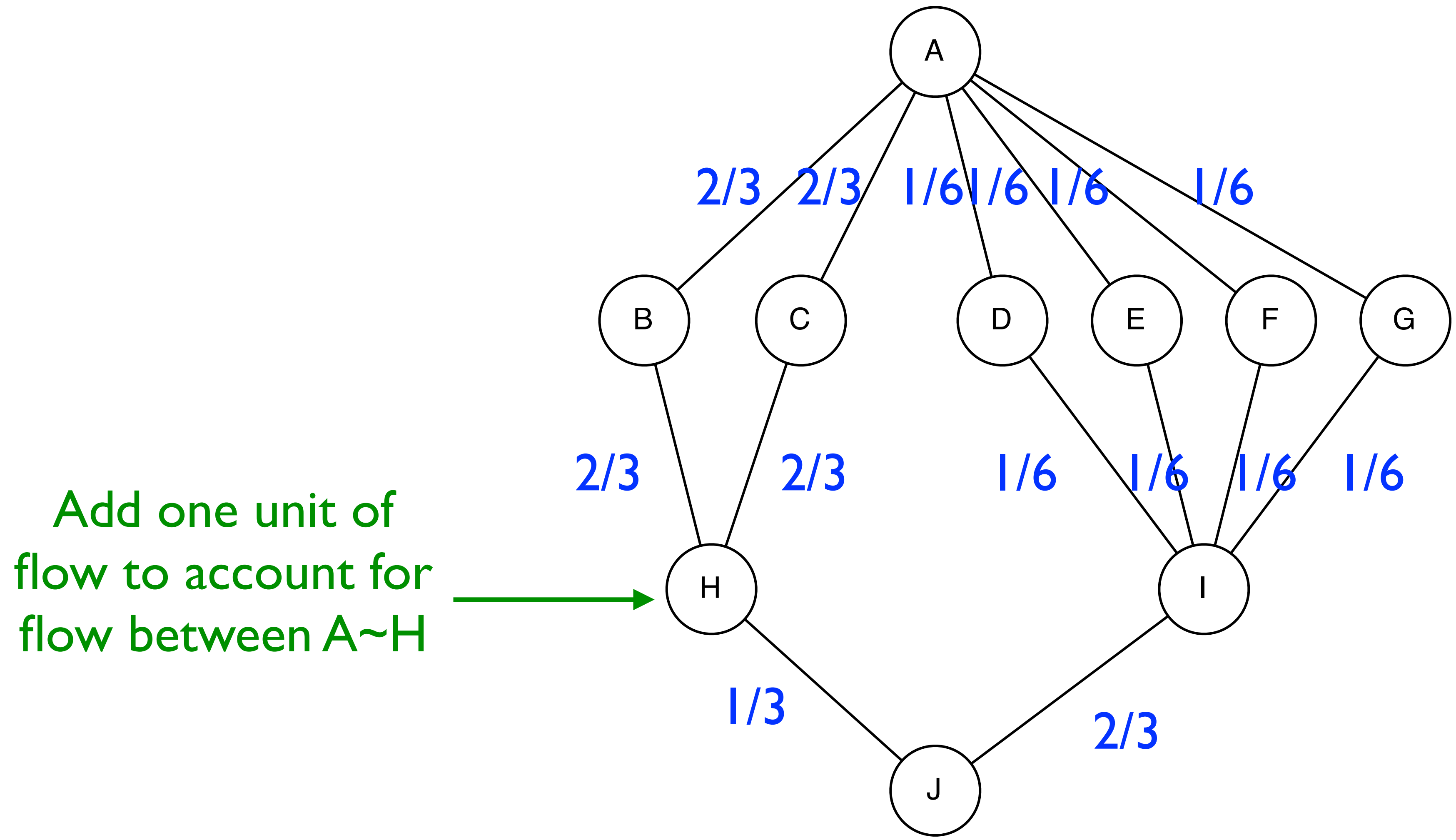
A~J betweenness contribution



- Can we combine the computation for A-to-J and and A-to-H flow computations?

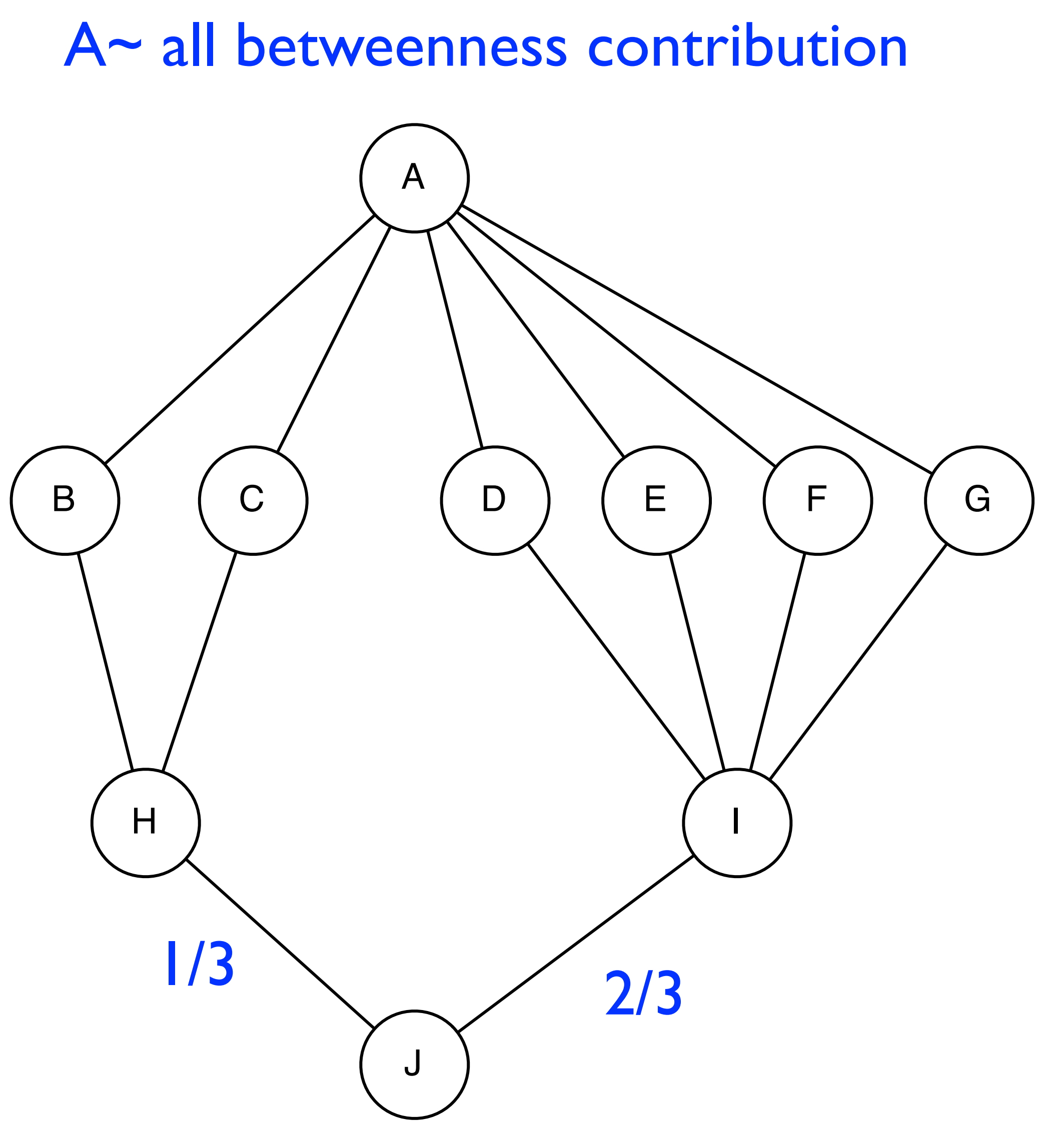
Computing Betweenness

A~J & A~H betweenness contribution



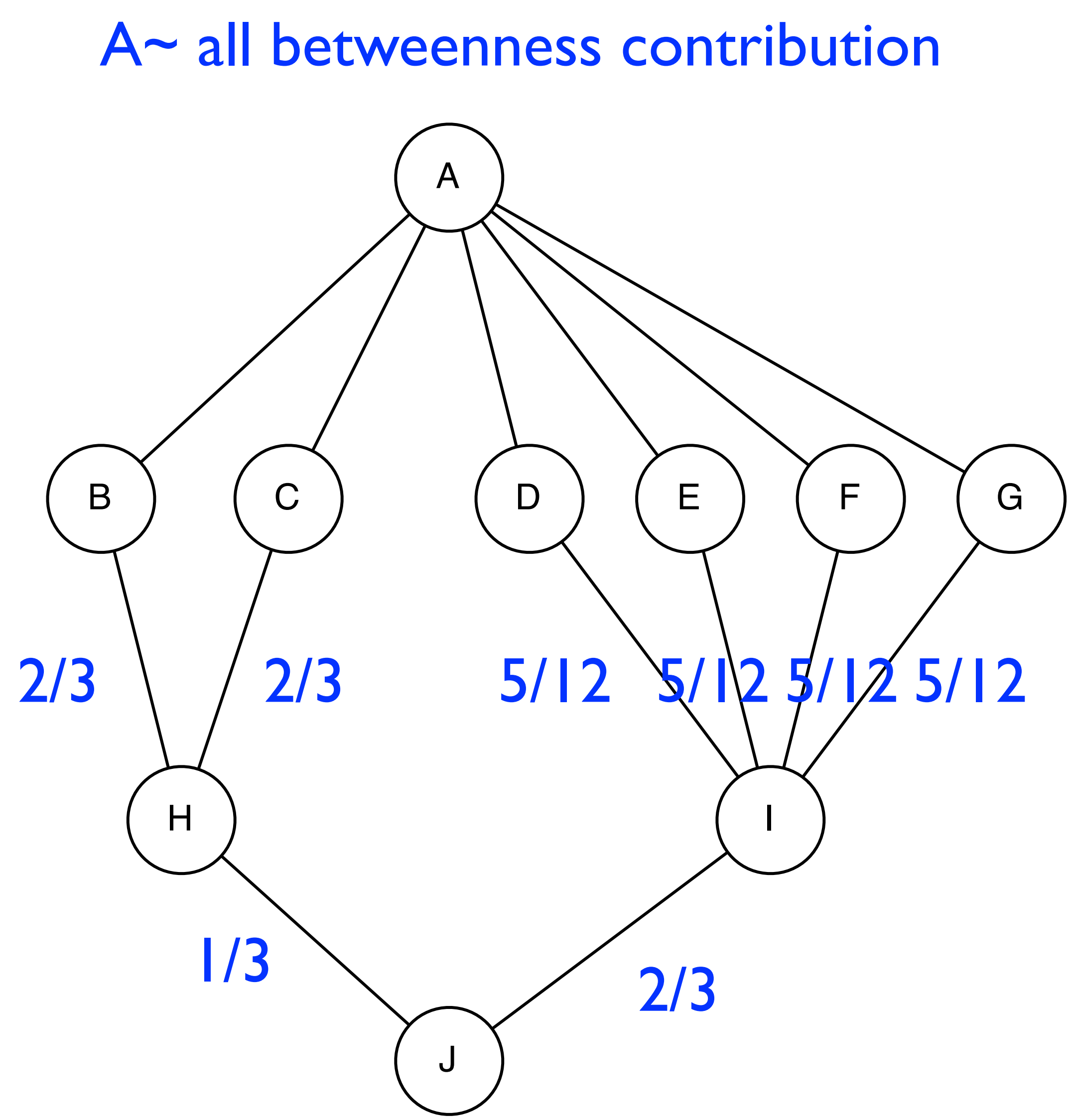
- Yes, we can combine the computation for A-to-J and and A-to-H flow computations?

Computing Betweenness



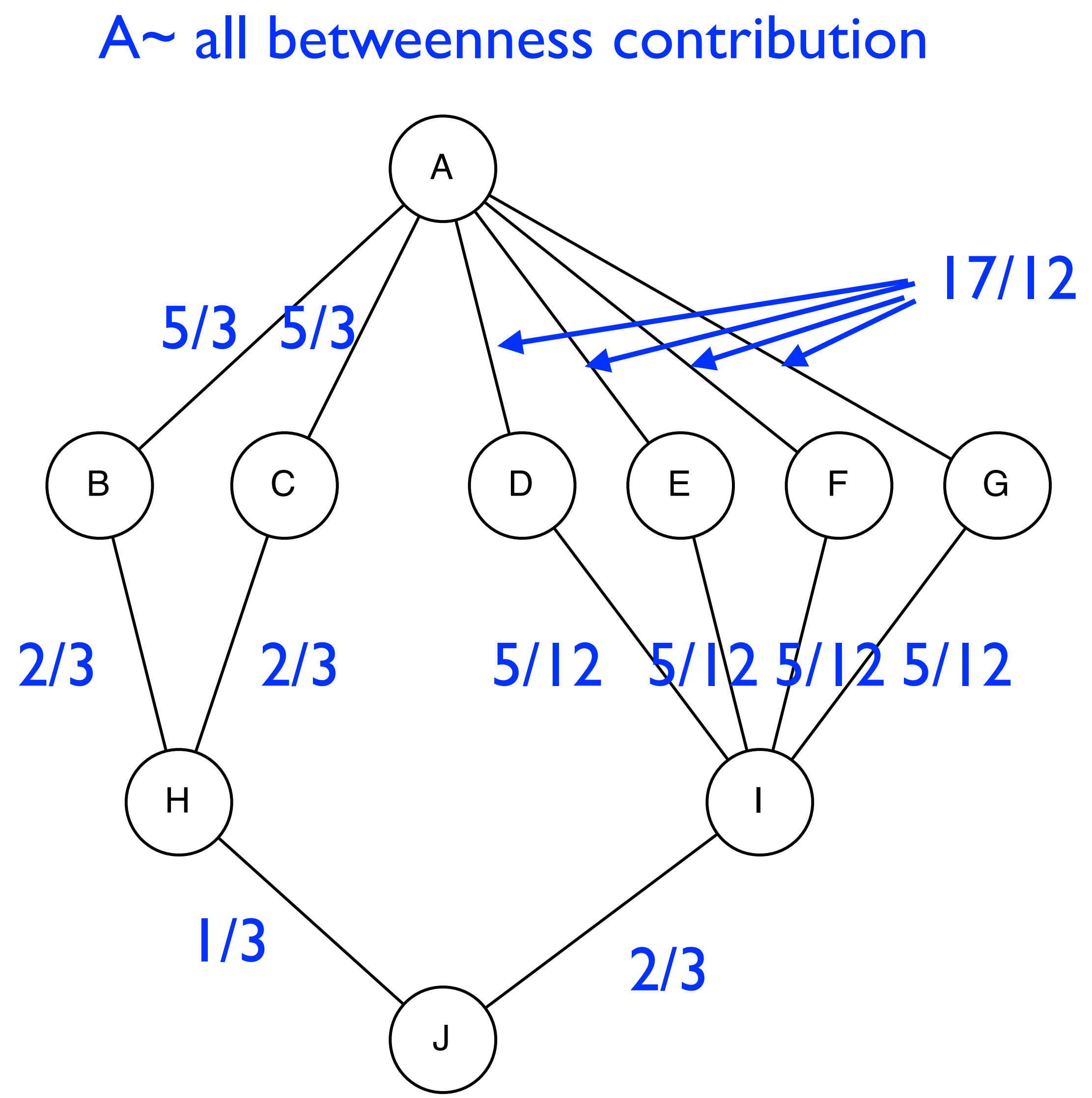
- We can combine (sum) all flows through edges — i.e., A~m flows for m=B,C,…J

Computing Betweenness



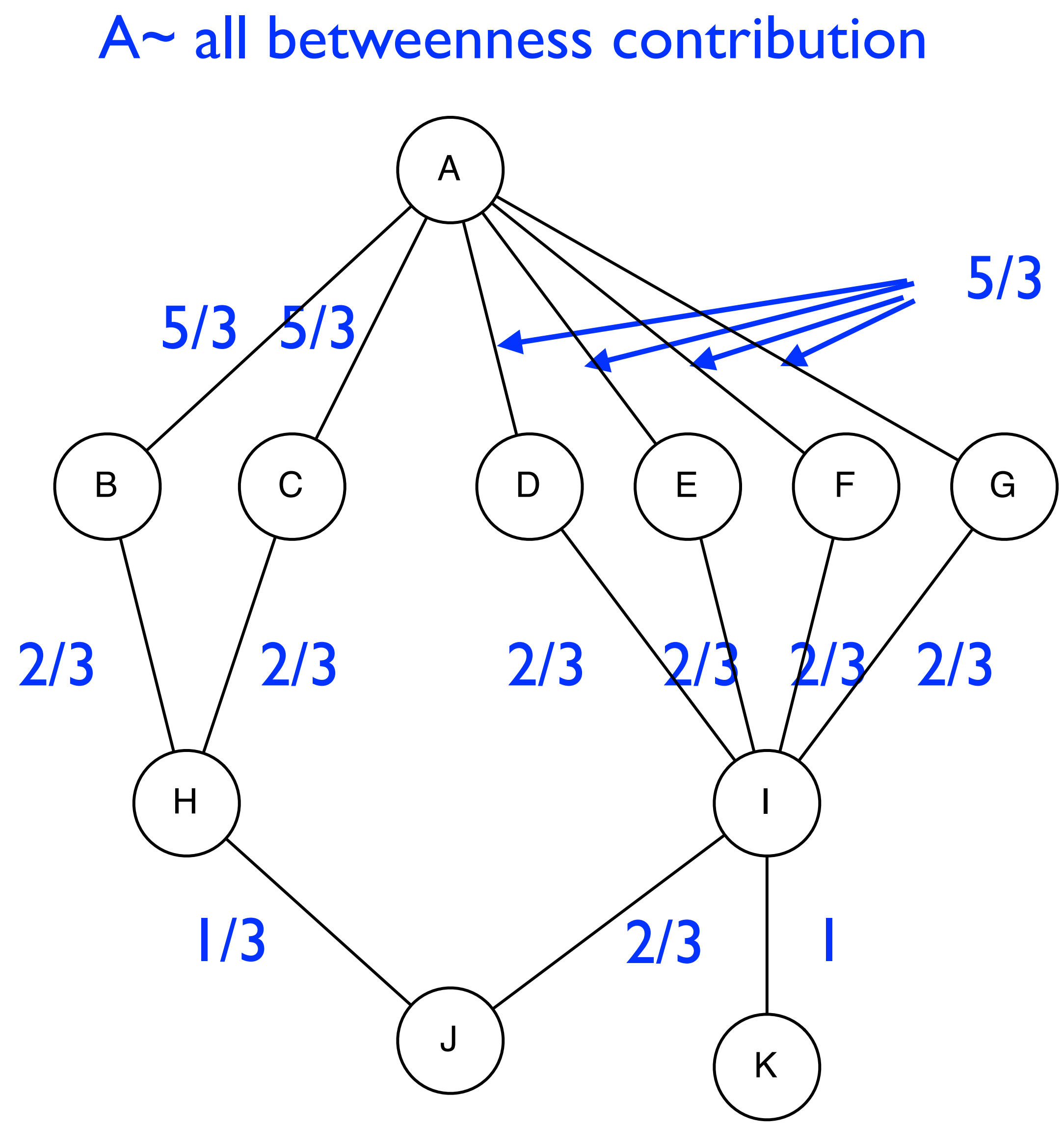
- We can combine (sum) all flows through edges — i.e., A~m flows for m=B,C,…J

Computing Betweenness



- We can combine (sum) all flows through edges — i.e., A~m flows for m=B,C,…J

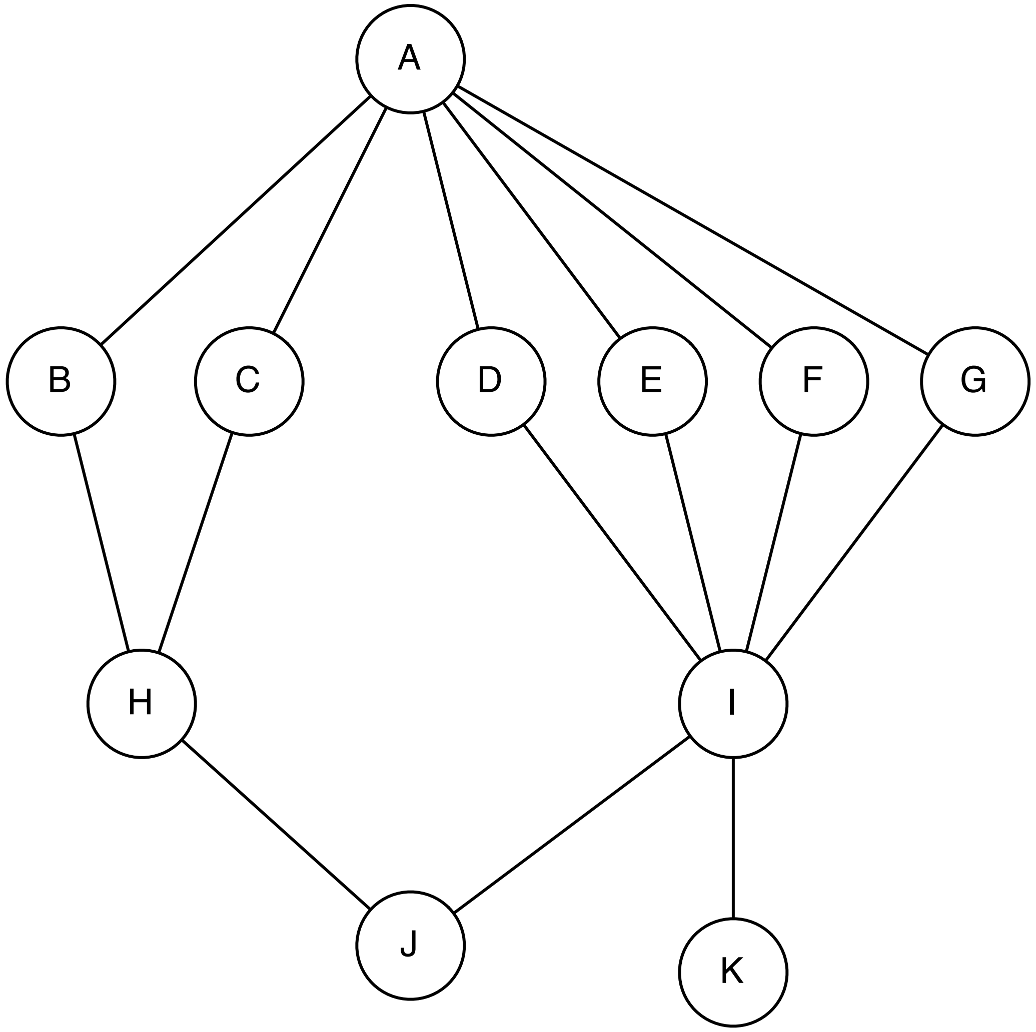
Computing Betweenness



- How does another node at the deepest level affect this?

Computing Betweenness

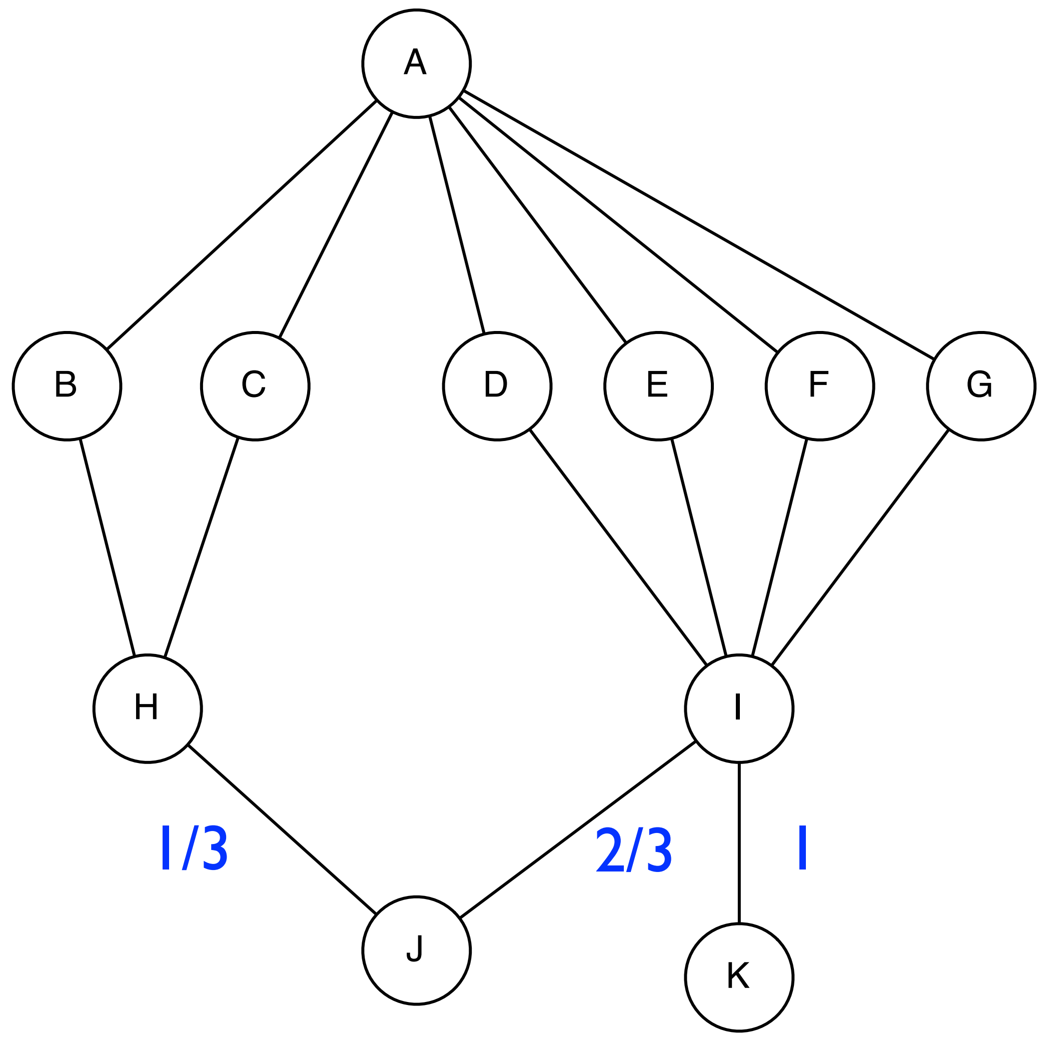
A~ all betweenness contribution



- How does another node at the deepest level affect this?

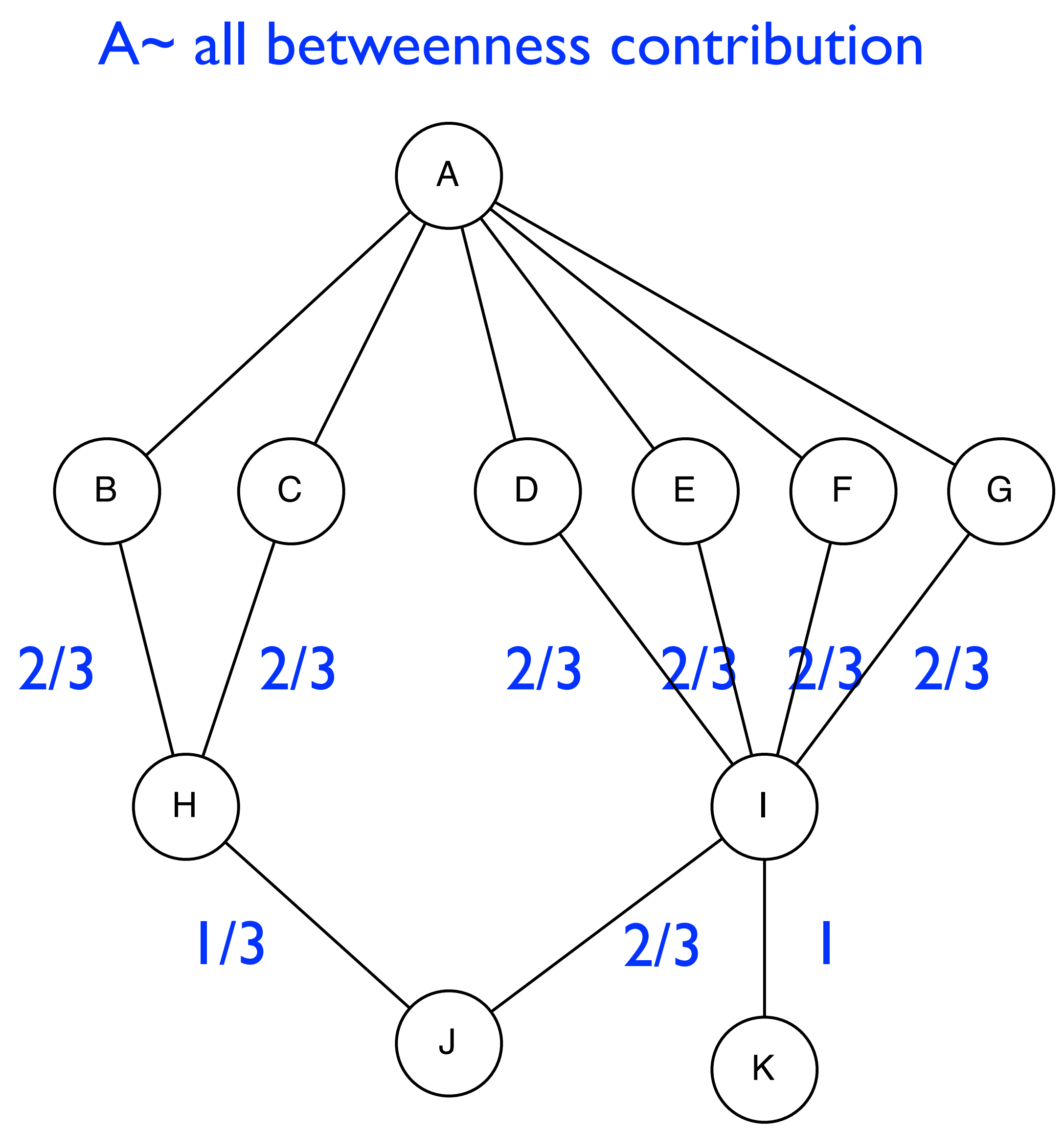
Computing Betweenness

A~ all betweenness contribution



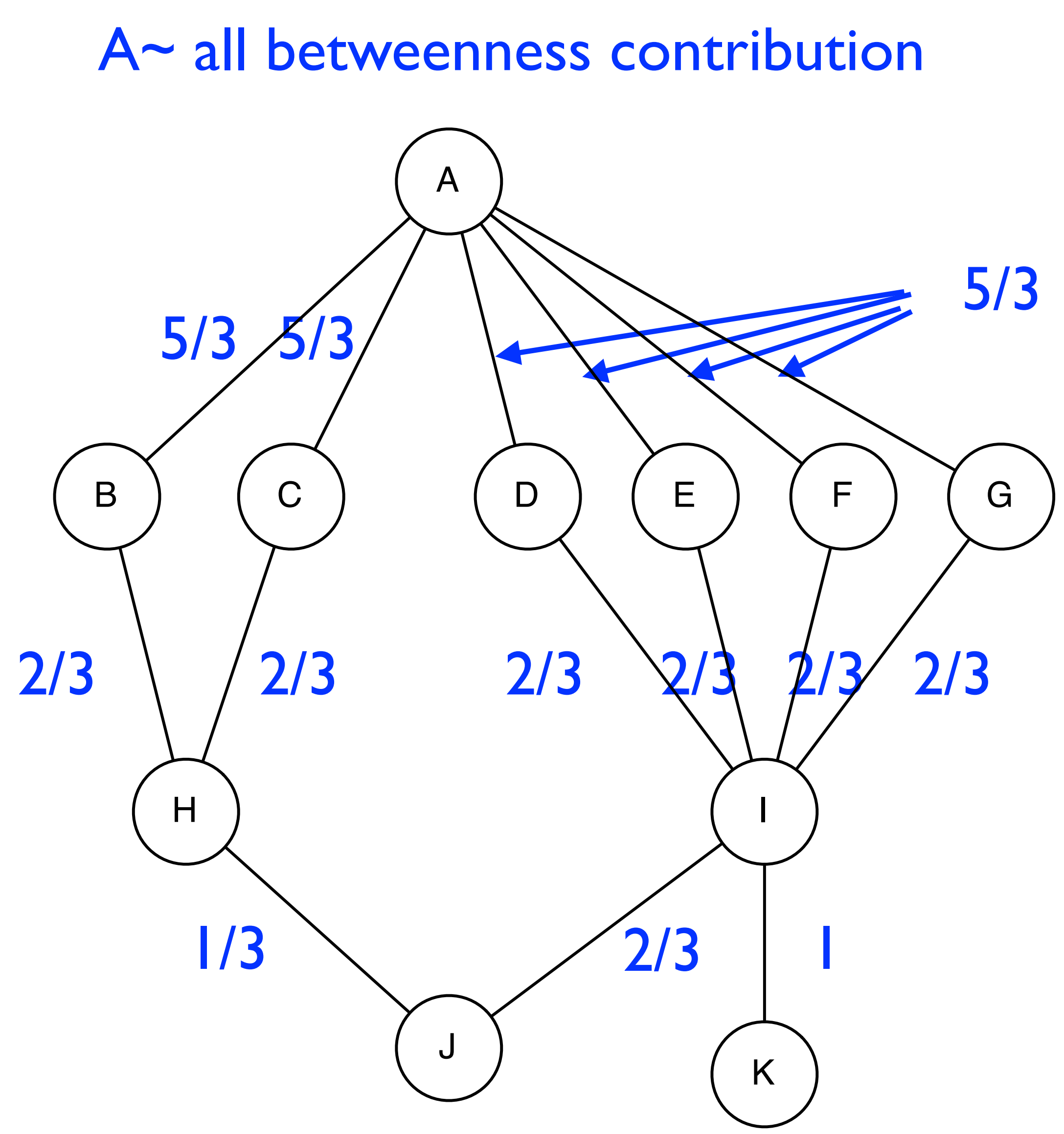
- How does another node at the deepest level affect this?

Computing Betweenness



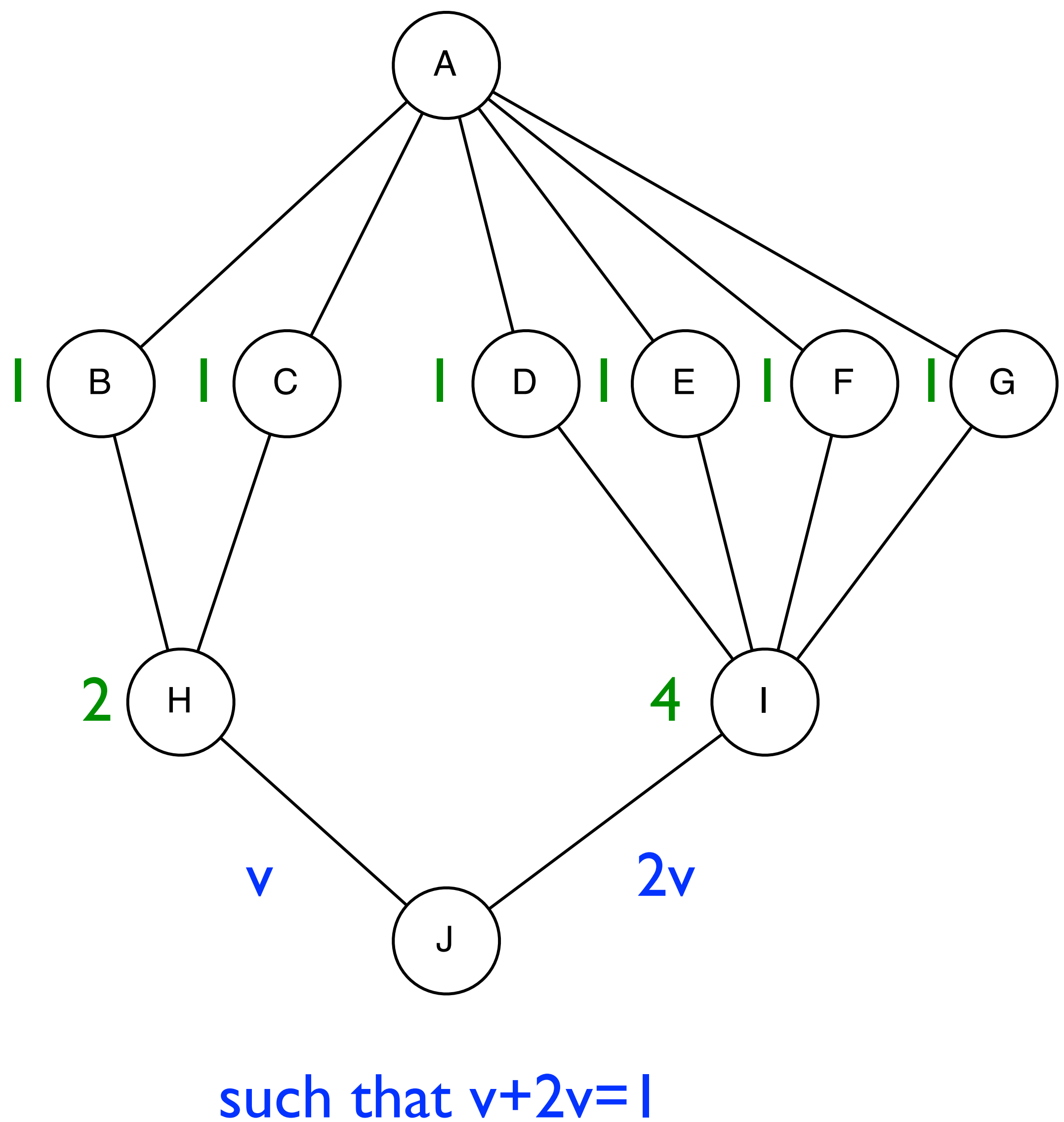
- How does another node at the deepest level affect this?

Computing Betweenness



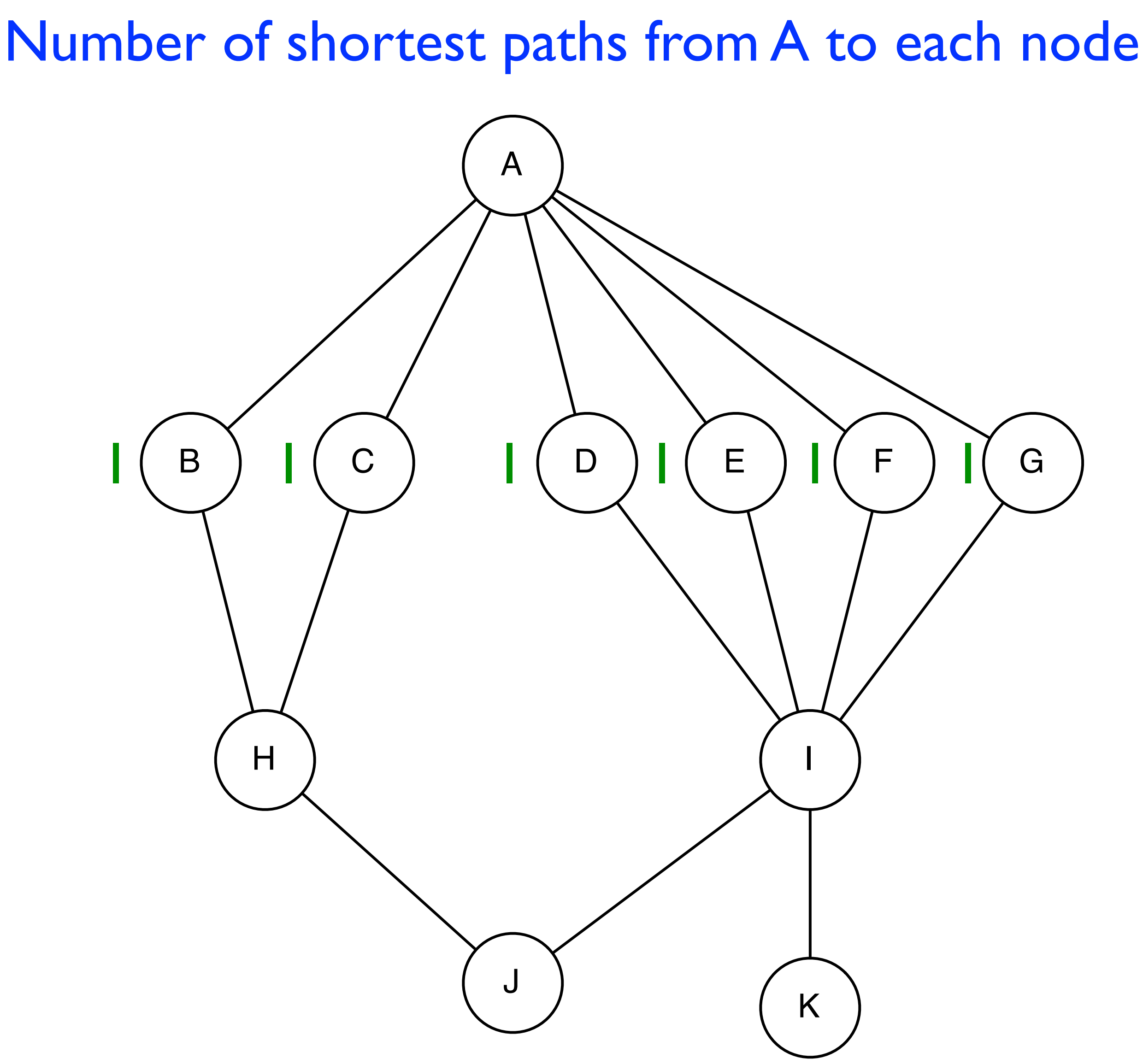
- How does another node at the deepest level affect this?

Computing Betweenness



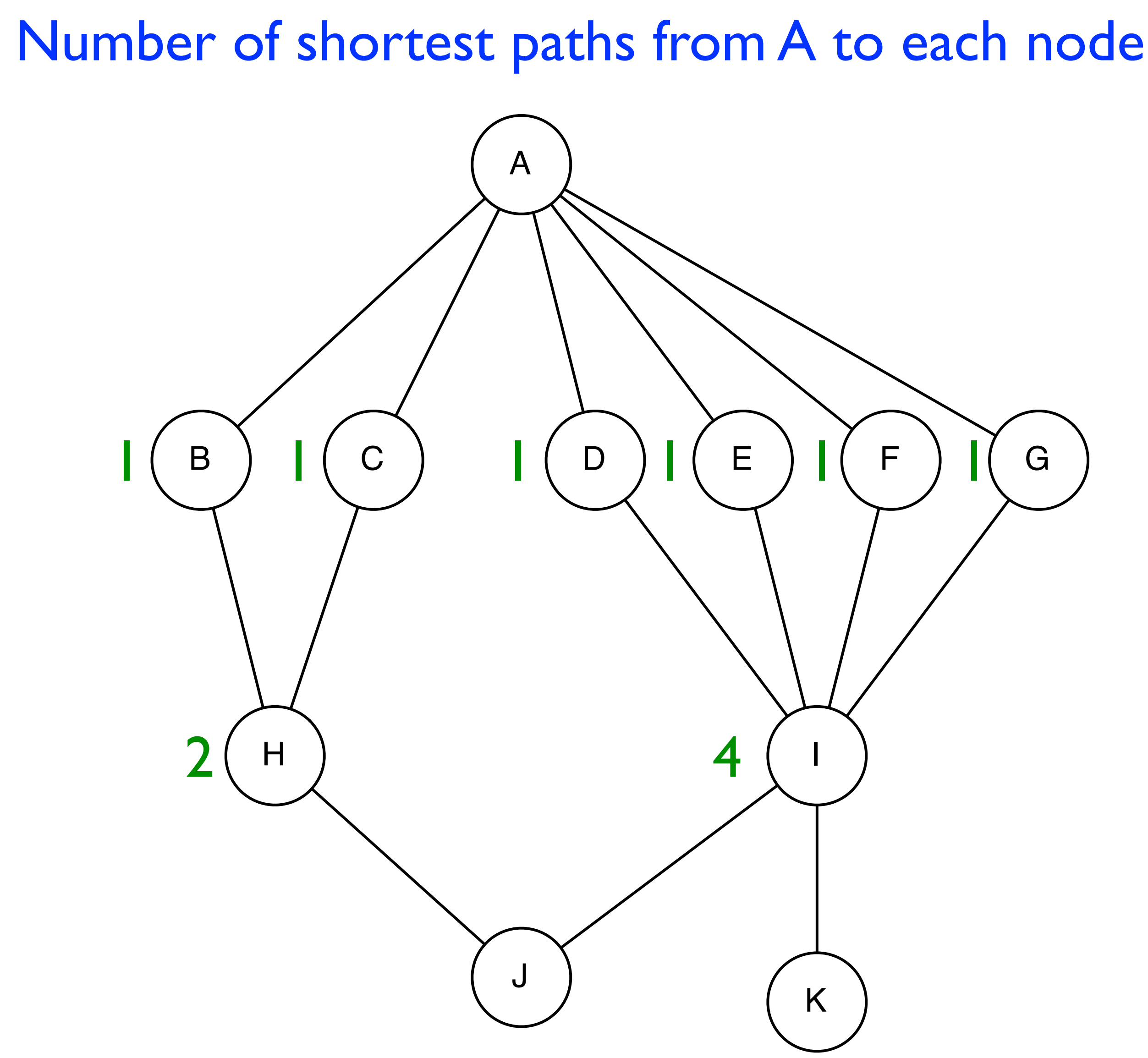
- We split the flow proportional to the number of shortest paths as we progressed up the graph from the bottom
- Can we compute the number of shortest paths from A to each node efficiently?

Computing Betweenness



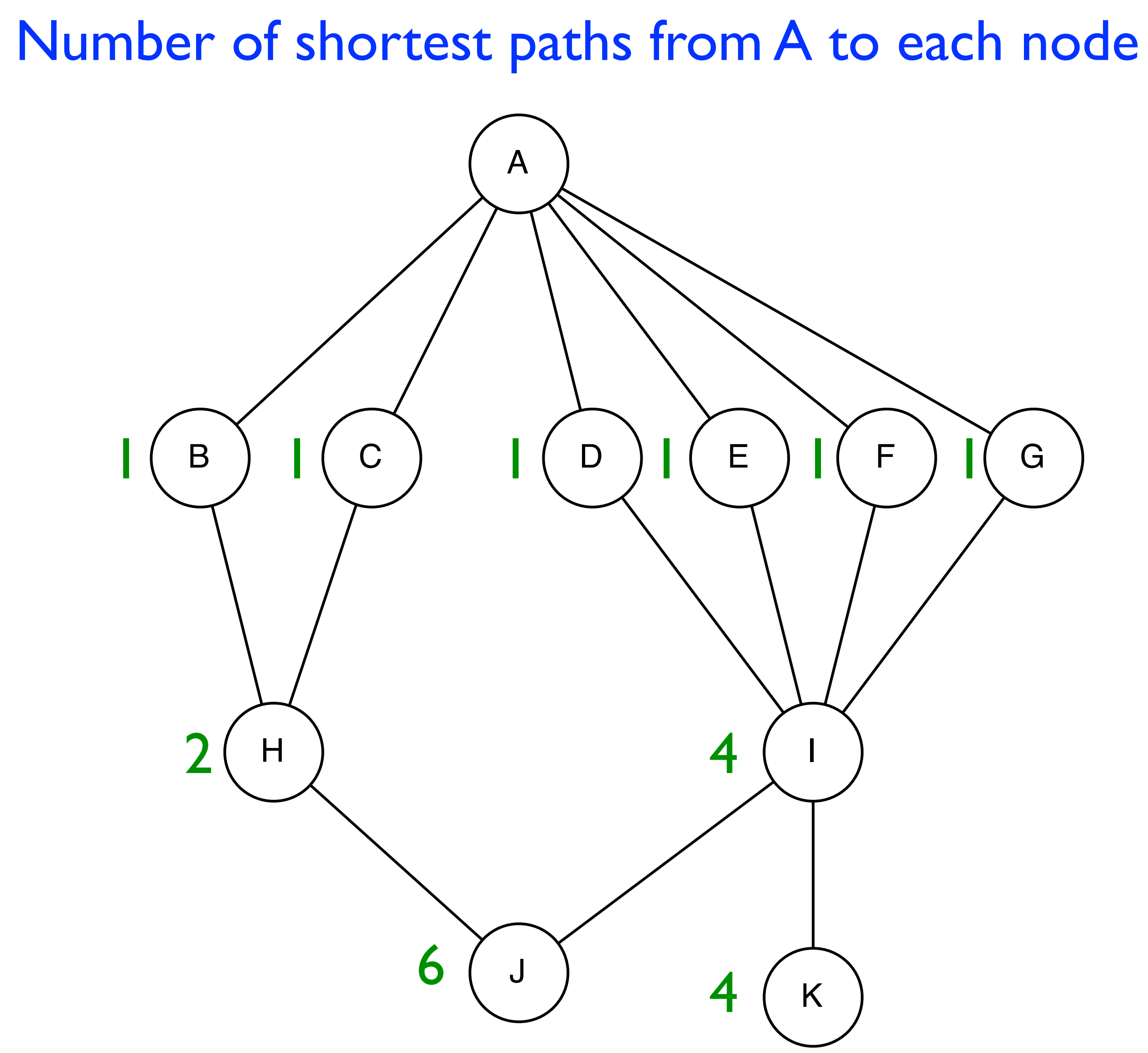
- Recursively computing the number of shortest paths from A to each node

Computing Betweenness



- Recursively computing the number of shortest paths from A to each node

Computing Betweenness

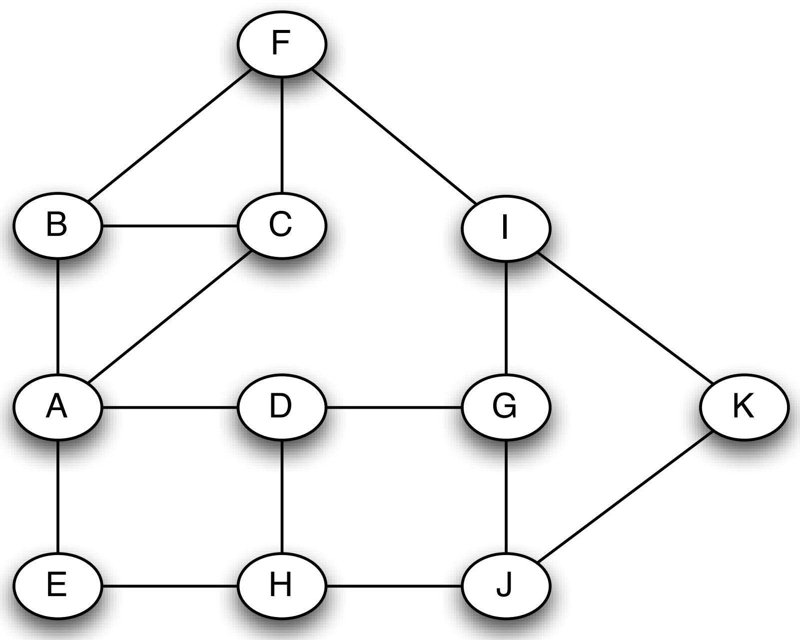


- Recursively computing the number of shortest paths from A to each node

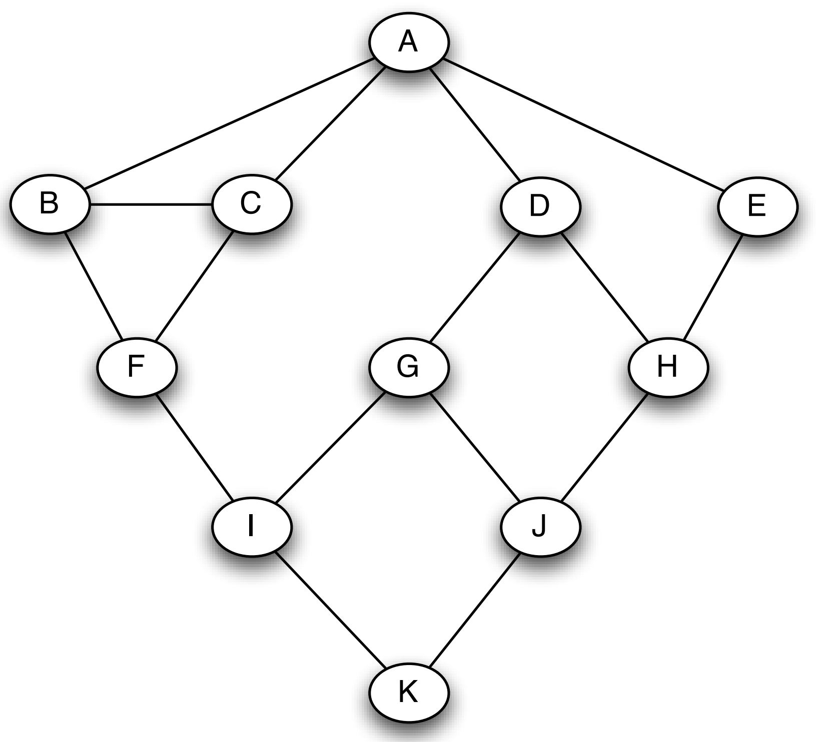
Computing Betweenness

- Efficient algorithm for computing betweenness (Brandes algorithm, 2001)
  - For each node i
    - Outward recursion: compute number of geodesics (shortest paths) from i to every other node
    - Inward recursion: go back towards node i splitting the "flow"(fraction of geodesics) proportionally according to number of shortest paths
  - Sum the flow on each edge over all of these N BFSes and divide by 2 (Divide by (N-1) choose 2 optional)

Computing Betweenness



(a) *A sample network*

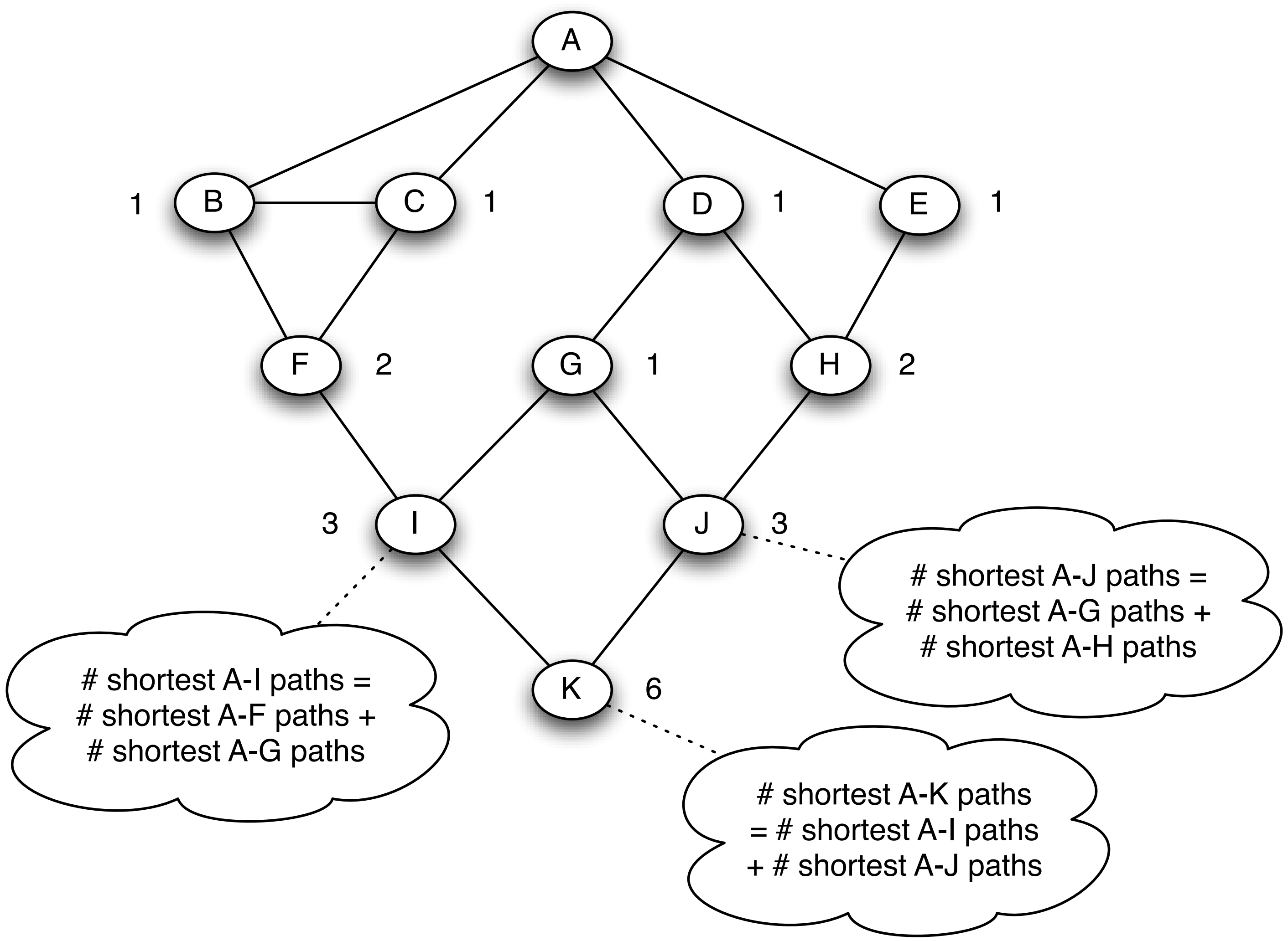


(b) *Breadth-first search starting at node A*

Figure 3.18: The first step in the efficient method for computing betweenness values is to perform a breadth-first search of the network. Here the results of breadth-first from node A are shown; over the course of the method, breadth-first search is performed from each node in turn.

Easley & Kleinberg

Computing Betweenness

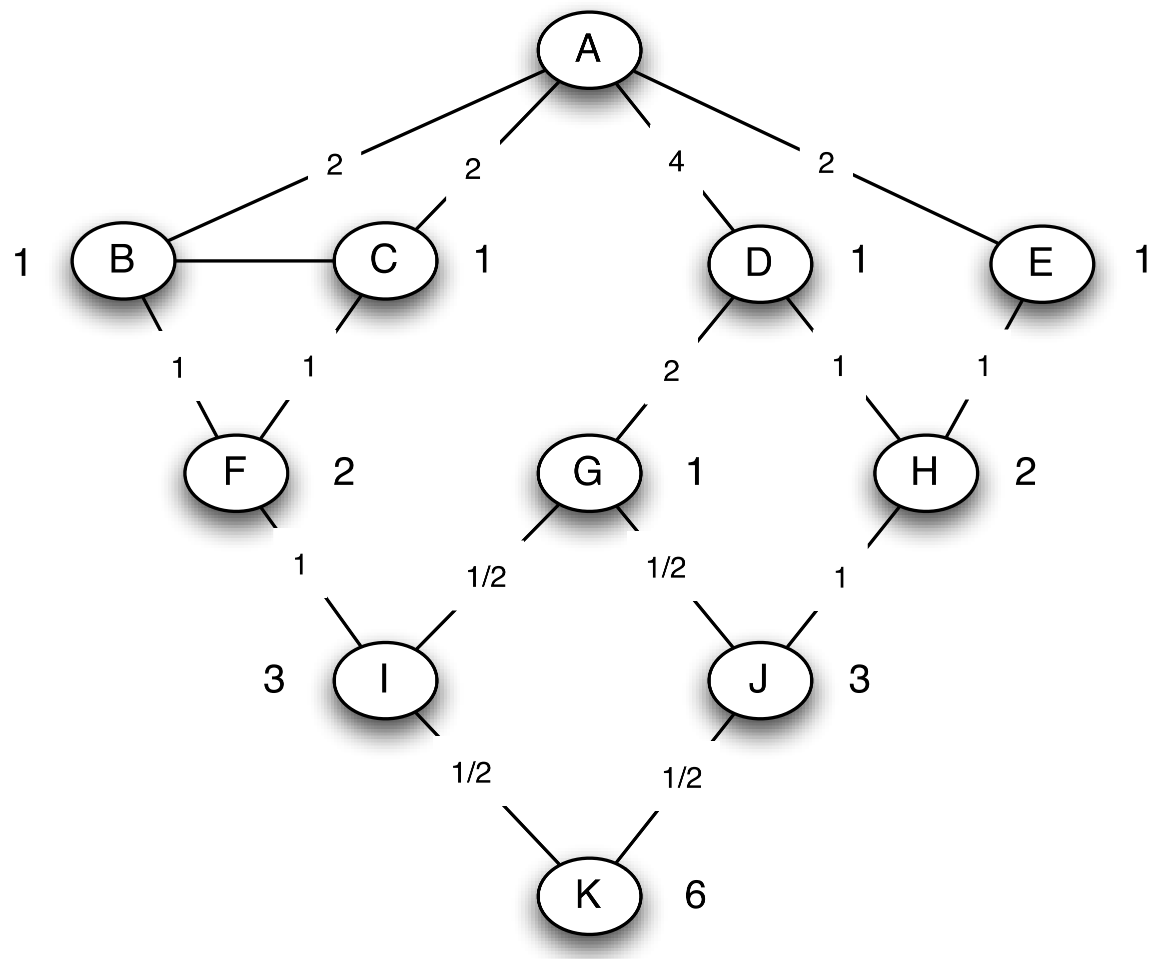


outward recursion

Figure 3.19: The second step in computing betweenness values is to count the number of shortest paths from a starting node A to all other nodes in the network. This can be done by adding up counts of shortest paths, moving downward through the breadth-first search structure.

Easley & Kleinberg

Computing Betweenness

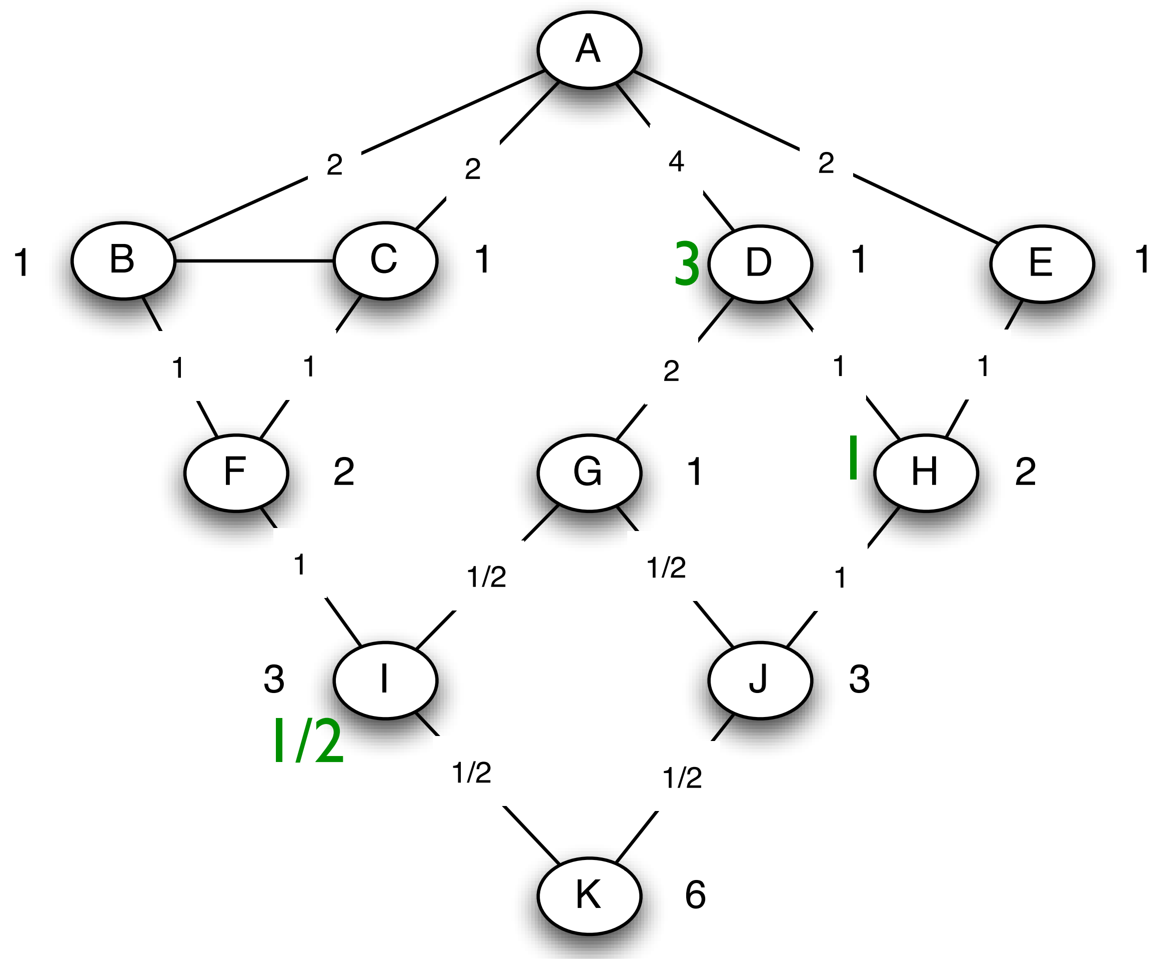


inward recursion

Figure 3.20: The final step in computing betweenness values is to determine the flow values from a starting node A to all other nodes in the network. This is done by working up from the lowest layers of the breadth-first search, dividing up the flow above a node in proportion to the number of shortest paths coming into it on each edge.

Easley & Kleinberg

Computing Betweenness

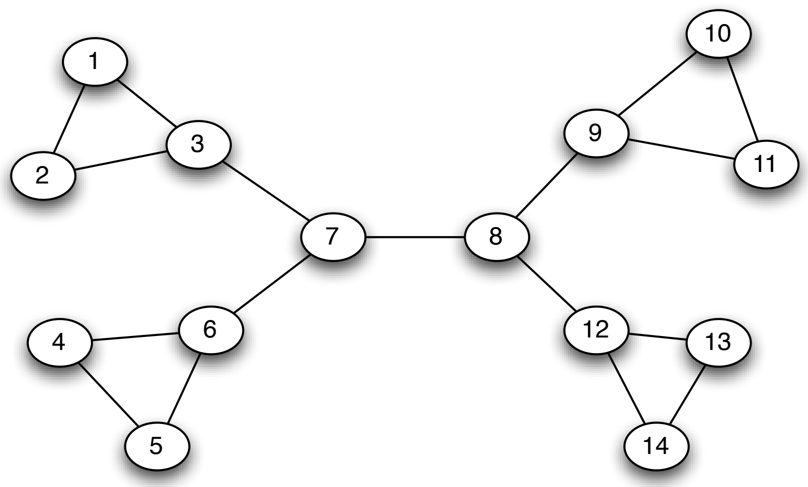


Same method for node betweenness

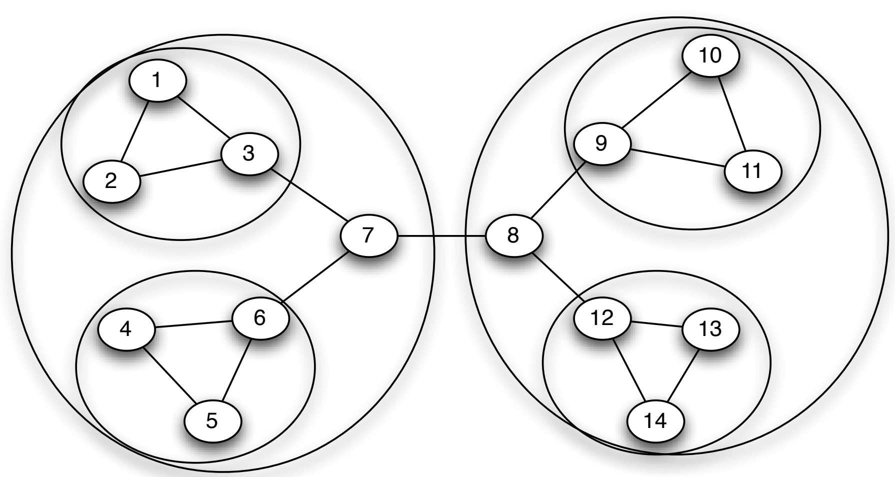
Figure 3.20: The final step in computing betweenness values is to determine the flow values from a starting node A to all other nodes in the network. This is done by working up from the lowest layers of the breadth-first search, dividing up the flow above a node in proportion to the number of shortest paths coming into it on each edge.

Easley & Kleinberg

Network Partitioning



(a) *A sample network*



(b) *Tightly-knit regions and their nested structure*

Figure 3.14: In many networks, there are tightly-knit regions that are intuitively apparent, and they can even display a *nested* structure, with smaller regions nesting inside larger ones.

Easley & Kleinberg

- How to identify tightly knight regions in a social network from the graph structure?
  - Use a metod for determining edge centrality (e.g., betweenness)
  - Remove these edges to break the graph into components
  - Repeat this process on the components as they arise
- Girvan-Newman Algorithm

Network Partitioning

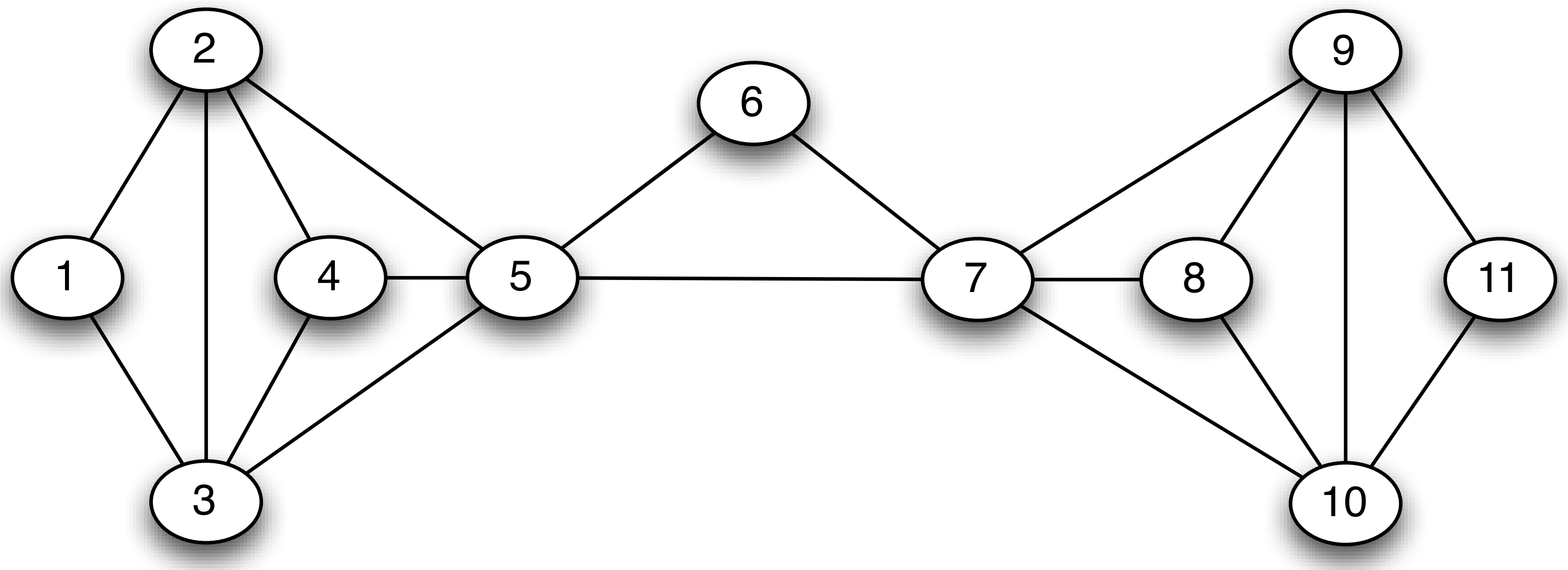


Figure 3.15: A network can display tightly-knit regions even when there are no bridges or local bridges along which to separate it.

Easley & Kleinberg

Network Partitioning

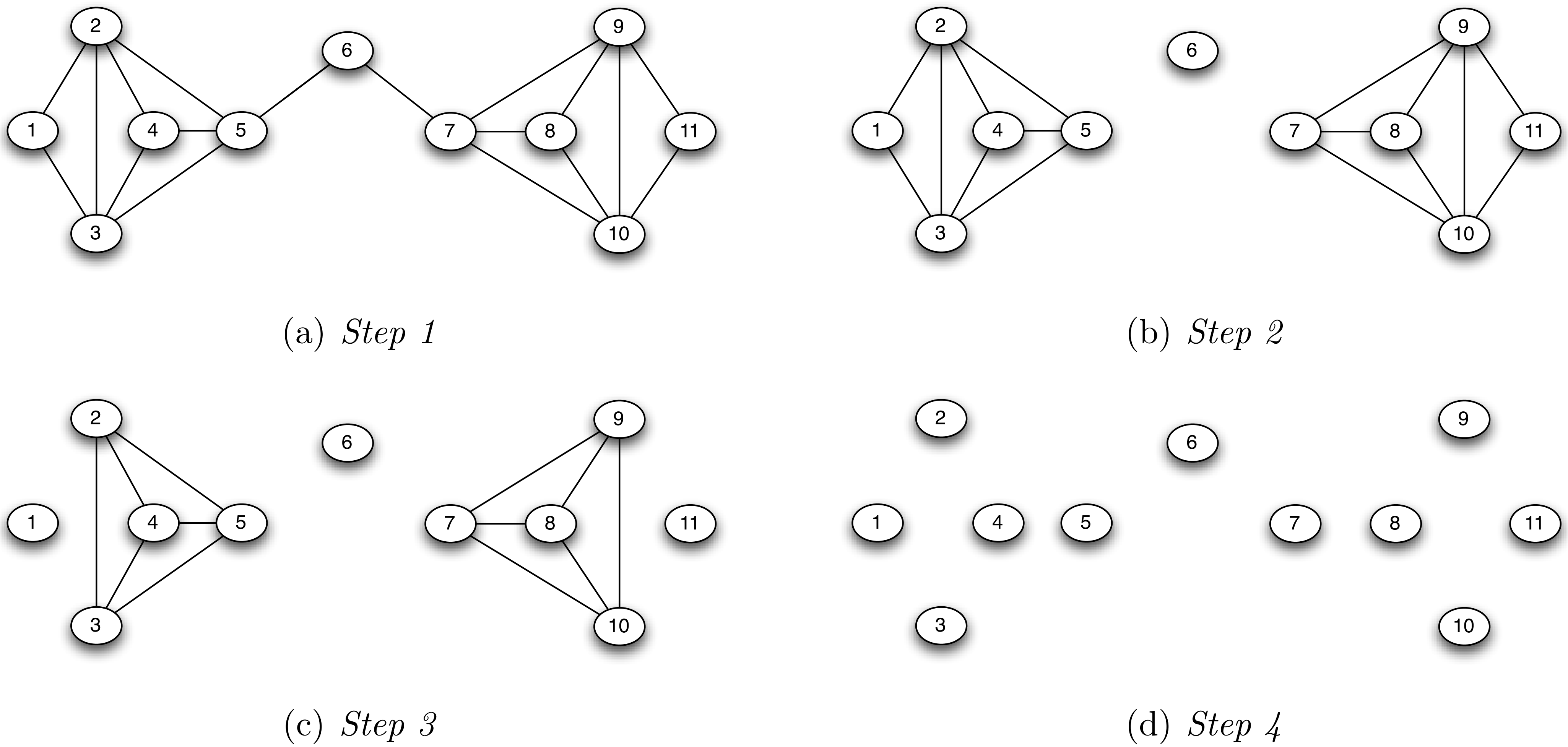


Figure 3.17: The steps of the Girvan-Newman method on the network from Figure 3.15.

Easley & Kleinberg

Network Partitioning

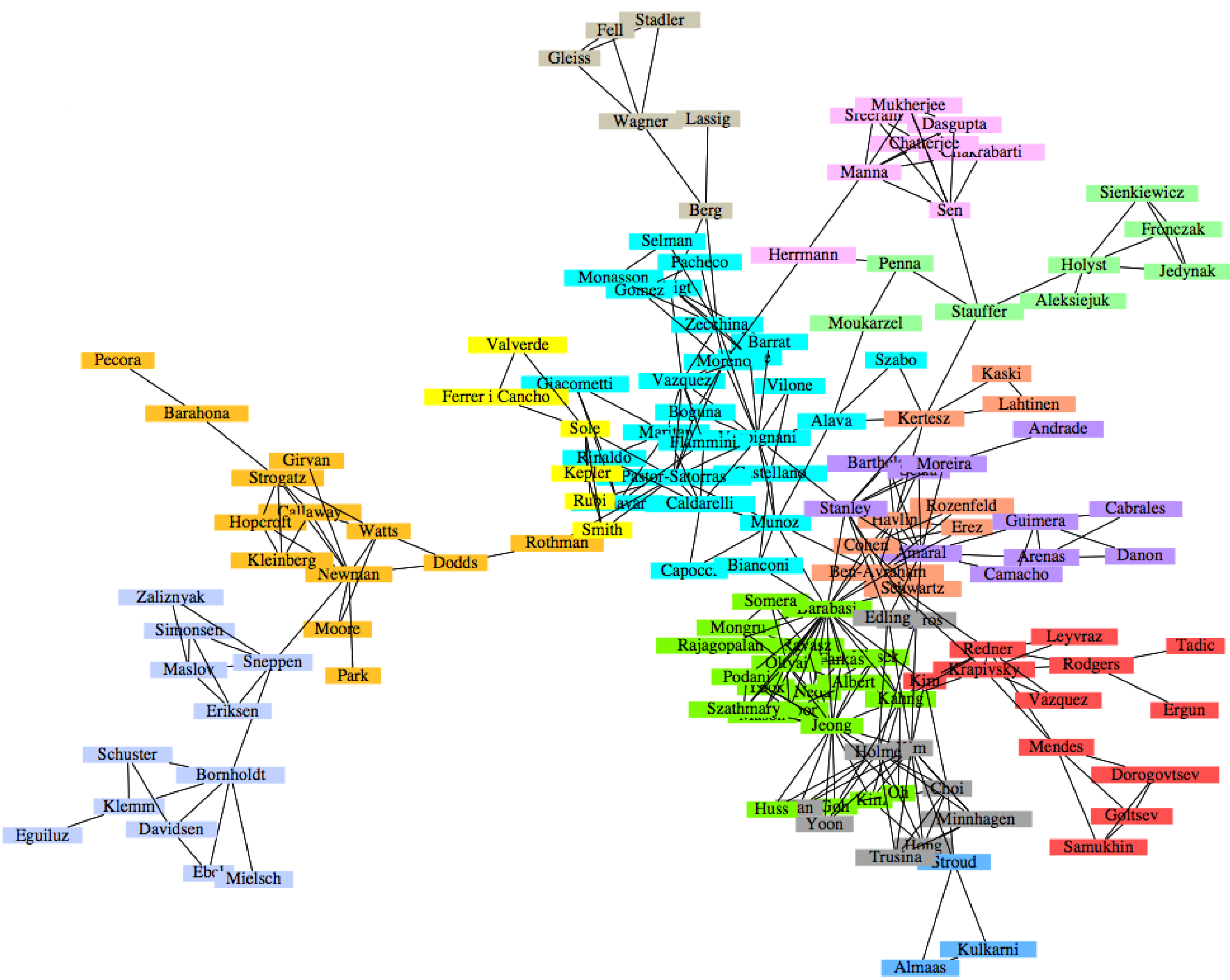


Figure 3.12: A co-authorship network of physicists and applied mathematicians working on networks [322]. Within this professional community, more tightly-knit subgroups are evident from the network structure.

Easley & Kleinberg