

Betweenness

Social Networks Analysis and Graph Algorithms

Prof. Carlos Castillo — <https://chato.cl/teach>



Universitat
Pompeu Fabra
Barcelona

Sources

- D. Easley and J. Kleinberg (2010). Networks, Crowds, and Markets – [Section 3.6B](#)
- A. L. Barabási (2016). Network Science – [Section 9.3](#)
- P. Boldi and S. Vigna (2014). [Axioms for Centrality](#) in *Internet Mathematics*
- Esposito and Pesce: [Survey of Centrality](#) 2015.
- URLs cited in the footer of slides

Types of centrality measure

- **Non-spectral**
 - Degree
 - Closeness and harmonic closeness
 - Betweenness
- Spectral
 - HITS
 - PageRank

Betweenness

Definitions

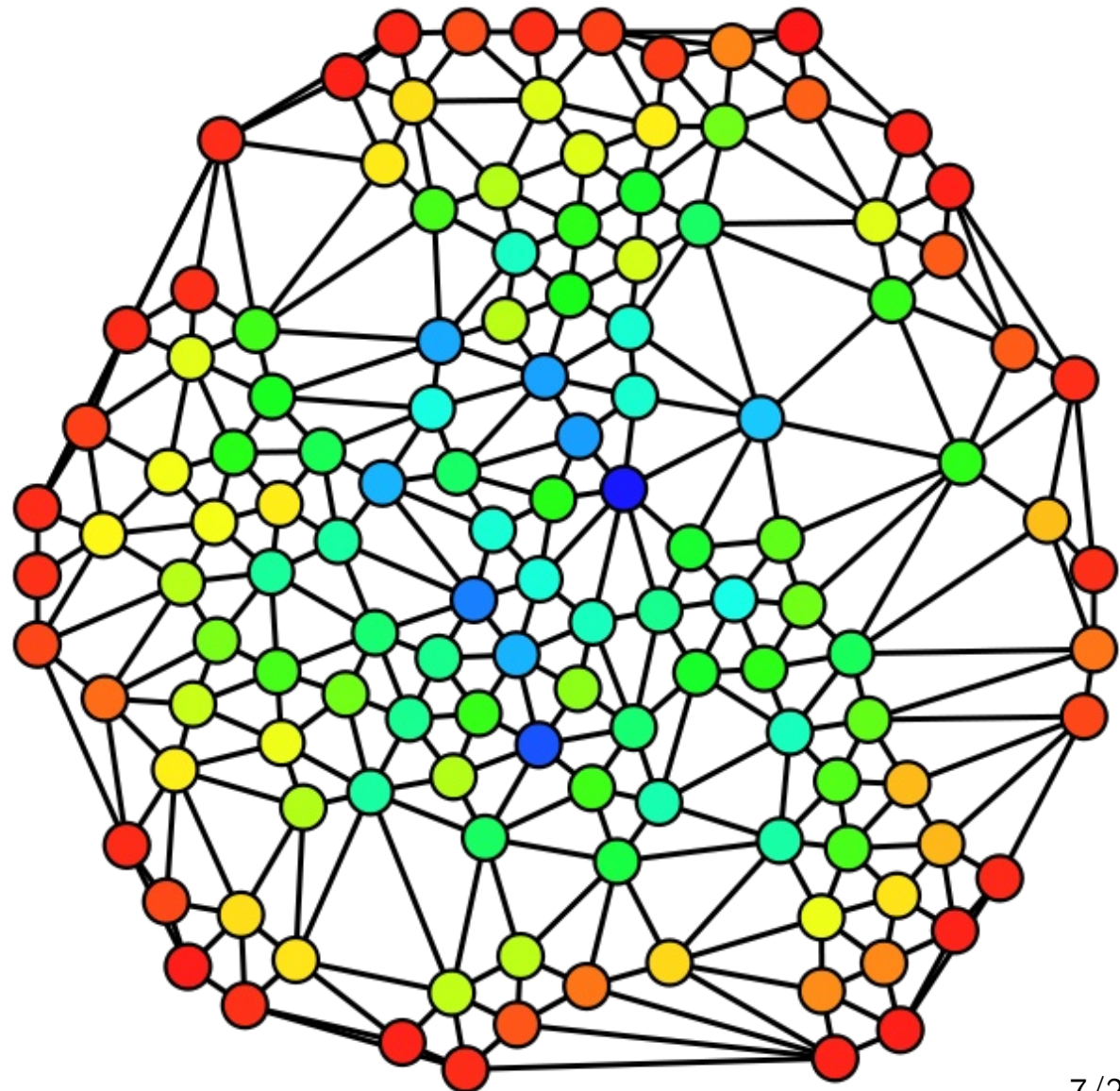
The **betweenness of a node** is the number of shortest paths that cross that node

The **betweenness of an edge** is the number of shortest paths that cross that edge

Node Betweenness

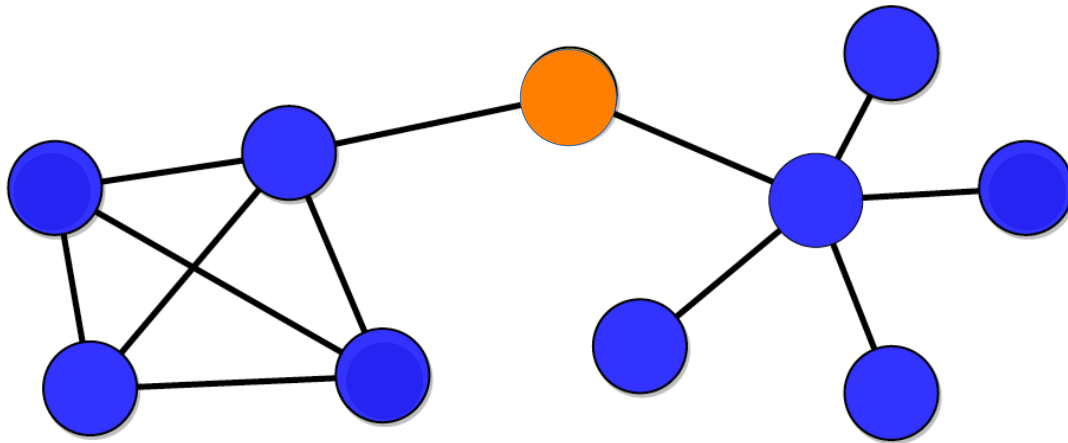
Graph with nodes
colored according to
node betweenness

red=low, blue=high



[Wikipedia: Betweenness centrality]

Example 1

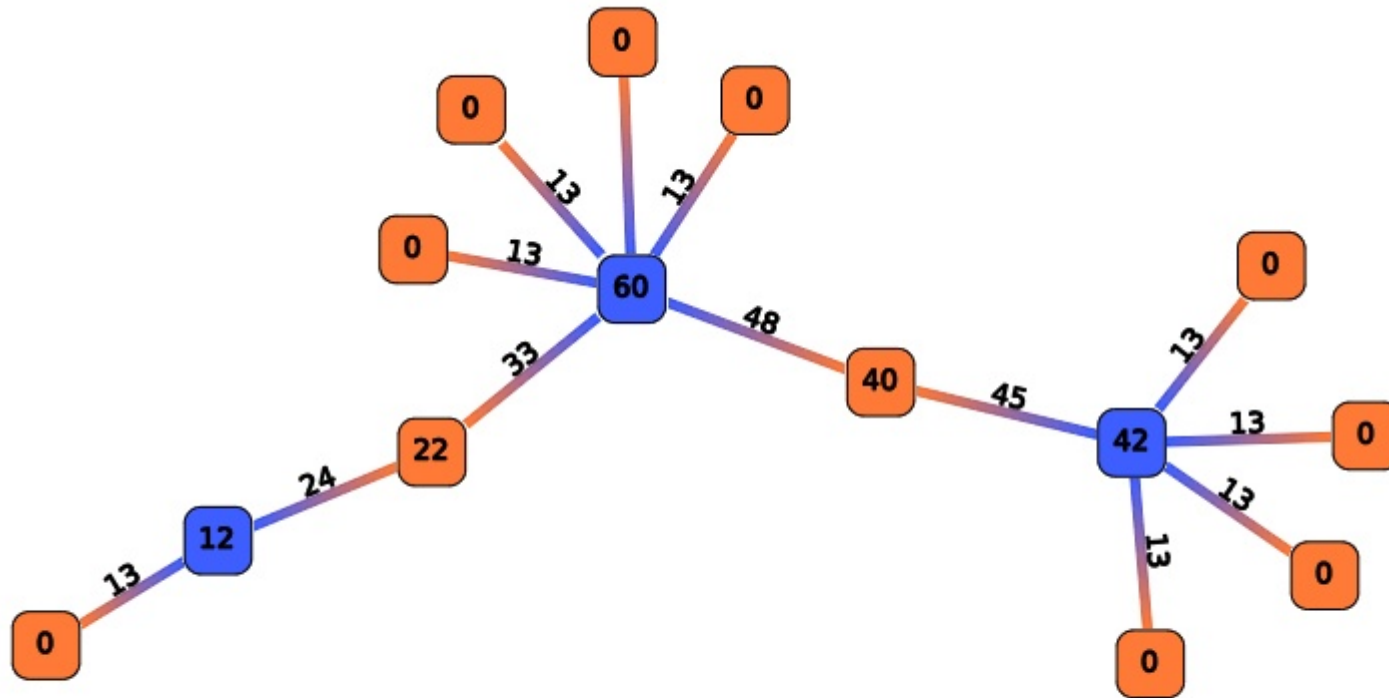


There are 20 shortest paths that cross through node 3. Why?

The shortest path between nodes 1 and 2 does not cross node 3, but the shortest path between nodes 1 and 4 does cross node 3.

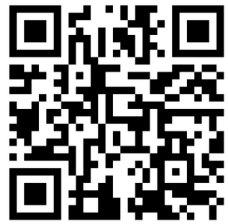
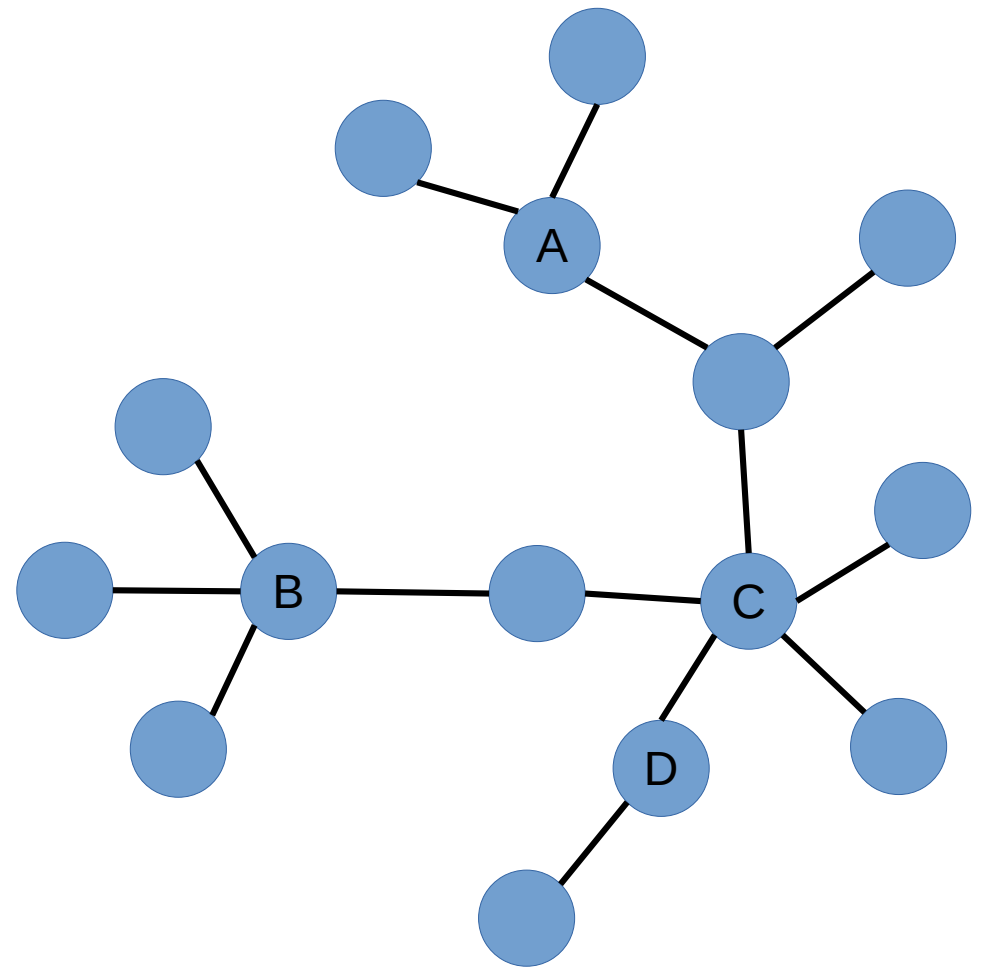
Example 2

Here, nodes and edges are labeled with their betweenness.



Exercise

Compute the node betweenness of the nodes marked with letters.

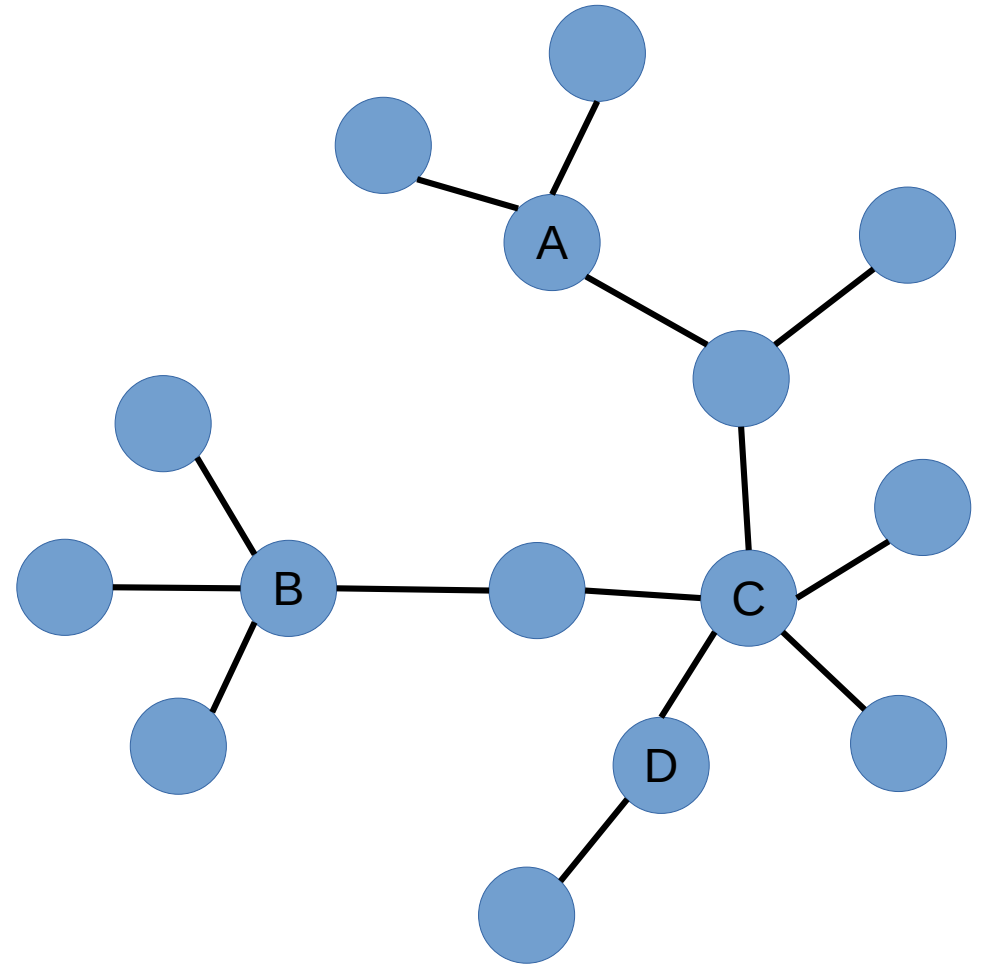


Pin board: <https://upfbarcelona.padlet.org/chato/asfs154waxnnkhgo>

Exercise (cont.)

What is a good algorithm to compute node betweenness of all nodes?

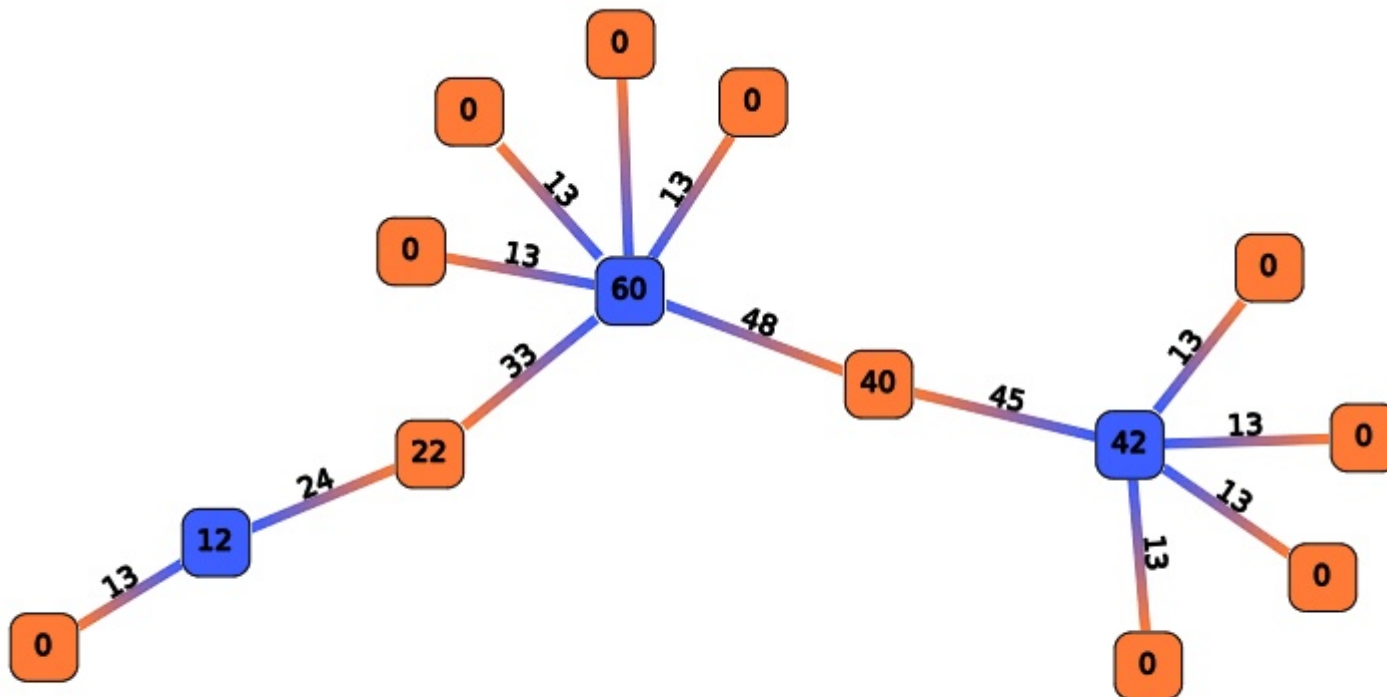
What limitations does your algorithm have?



Edge Betweenness

Edge Betweenness

An **edge** has high betweenness if it is part of many shortest-paths.



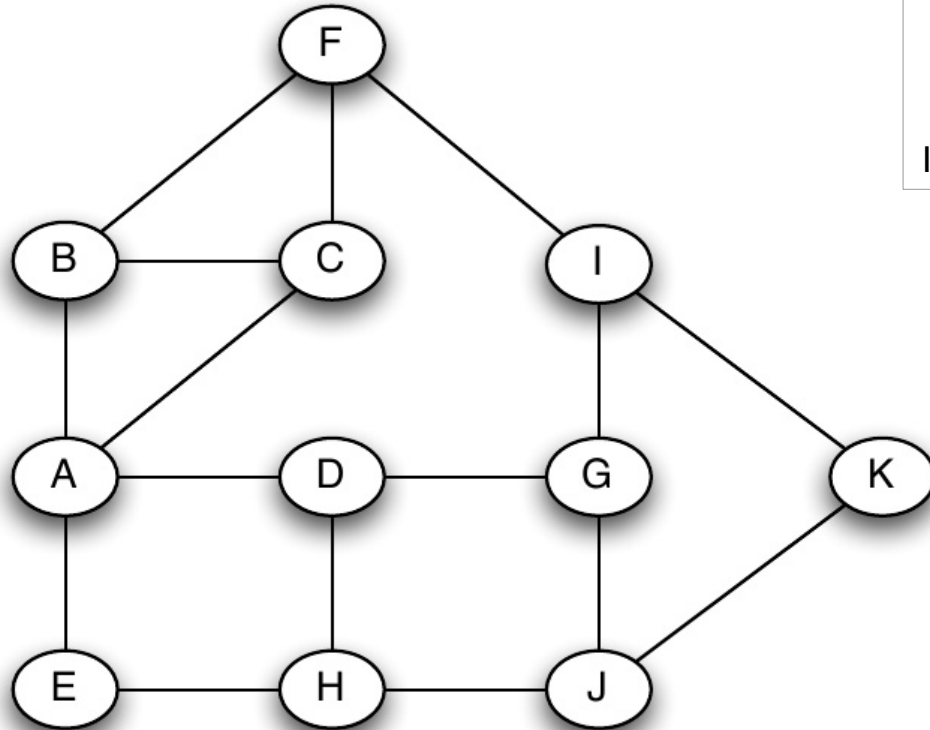
Approximate method [sampling]

- Label all edges e with $b(e) = 0$
- Repeat K times:
 - Pick a random pair of nodes (u,v)
 - Compute shortest path between u and v
 - $b(e) \leftarrow b(e) + 1$ for all edges e along the path
- $b(e)$ is a lower bound for betweenness (e)
- Useful if we only care about finding the edge with the highest betweenness, or finding the top- k edges with the highest betweenness \rightarrow an early stopping criterion is possible

Exact algorithm [Brandes, Newman]

- For every node u in V
 - Layer the graph performing a BFS from u
 - For every node v in V , $v \neq u$, sorted by layer
 - Assign to v a number $s(v)$ indicating how many shortest paths from u arrive to v
 - For every node v in V , $v \neq u$, sorted by reverse layer
 - Score to distribute = $1 + \text{score from children}$
 - Add score to parent edges in proportion to $s(v)$
- In the end divide all edge scores by two

Example

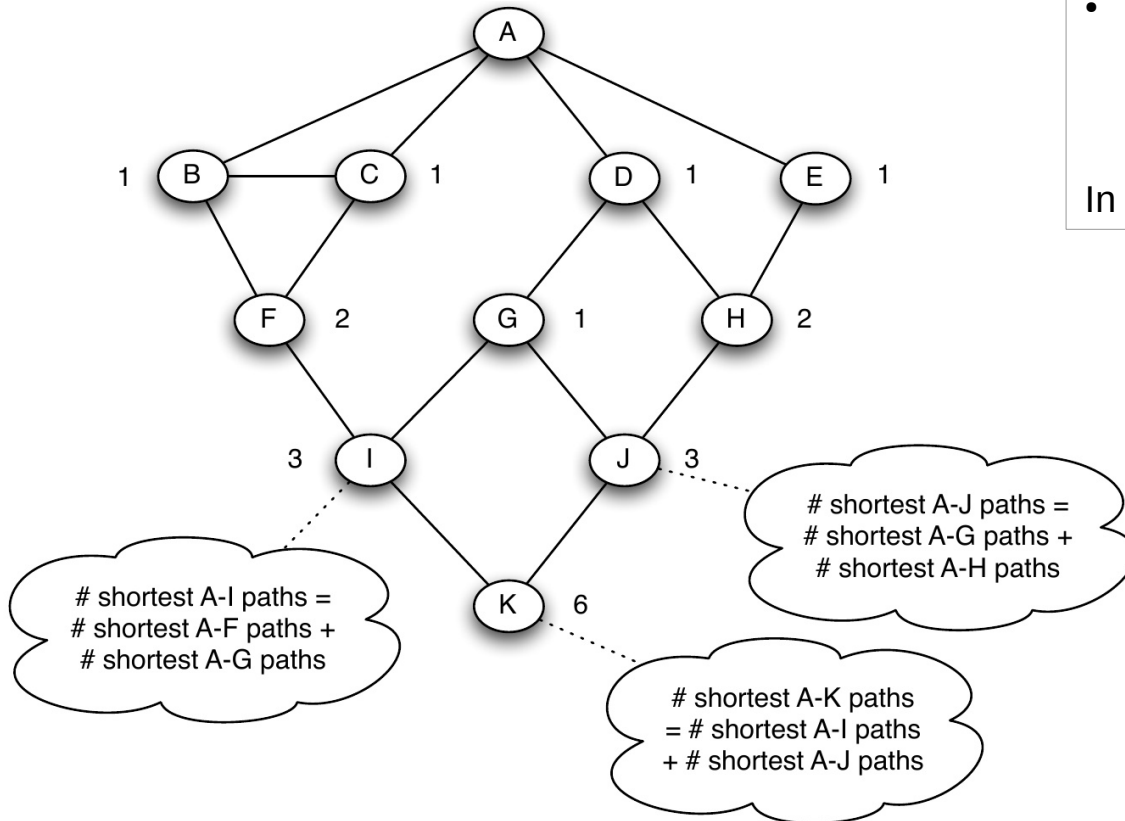


For every node u in V

- Layer the graph performing a BFS from u
- For every node v in V , $v \neq u$, sorted by layer
 - Assign to v a number $s(v)$ indicating how many shortest paths from u arrive to v
- For every node v in V , $v \neq u$, sorted by reverse layer
 - Score to distribute = $1 + \text{score from children}$
 - Add score to distribute to parent edges in proportion to $s(v)$

In the end divide all edge scores by two

Example



For every node u in V

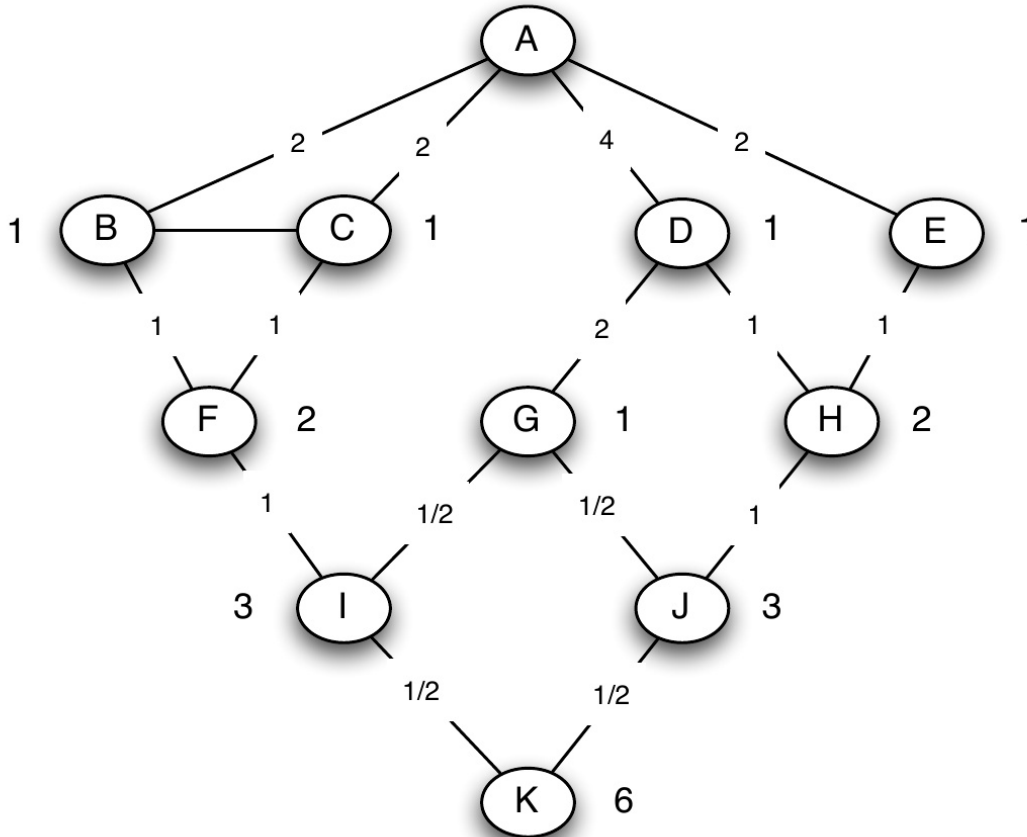
- Layer the graph performing a BFS from u
- **For every node v in V , $v \neq u$, sorted by layer**
 - **Assign to v a number $s(v)$ indicating how many shortest paths from u arrive to v**
- For every node v in V , $v \neq u$, sorted by reverse layer
 - Score to distribute = 1 + score from children
 - Add score to distribute to parent edges in proportion to $s(v)$

In the end divide all edge scores by two

All nodes in layer 1 get $s(v)=1$

Remaining nodes: simply add $s(\cdot)$ of their parents

Example



For every node u in V

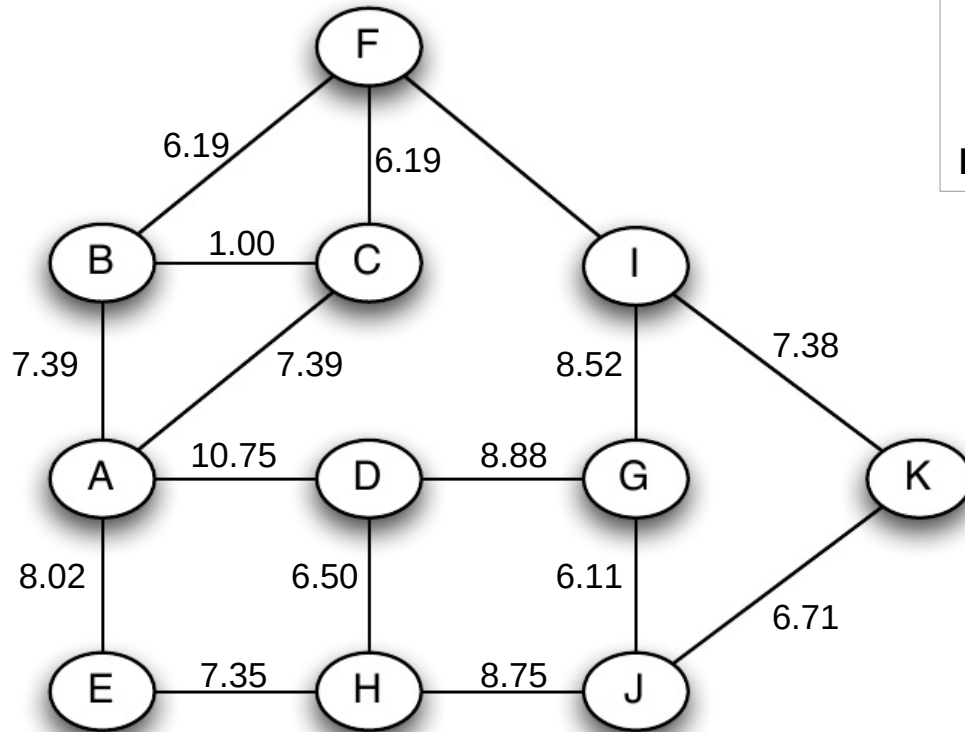
- Layer the graph performing a BFS from u
- For every node v in V , $v \neq u$, sorted by layer
 - Assign to v a number $s(v)$ indicating how many shortest paths from u arrive to v
- **For every node v in V , $v \neq u$, sorted by rev. layer**
 - **Score to distribute = 1 + score from children**
 - **Add score to distribute to parent edges in proportion to $s(v)$**

In the end divide all edge scores by two

Nodes without children distribute a score of 1

Other nodes distribute 1 + whatever they receive from their children

Result



For every node u in V

- Layer the graph performing a BFS from u
- For every node v in V , $v \neq u$, sorted by layer
 - Assign to v a number $s(v)$ indicating how many shortest paths from u arrive to v
- For every node v in V , $v \neq u$, sorted by reverse layer
 - Score to distribute = $1 + \text{score from children}$
 - Add score to distribute to parent edges in proportion to $s(v)$

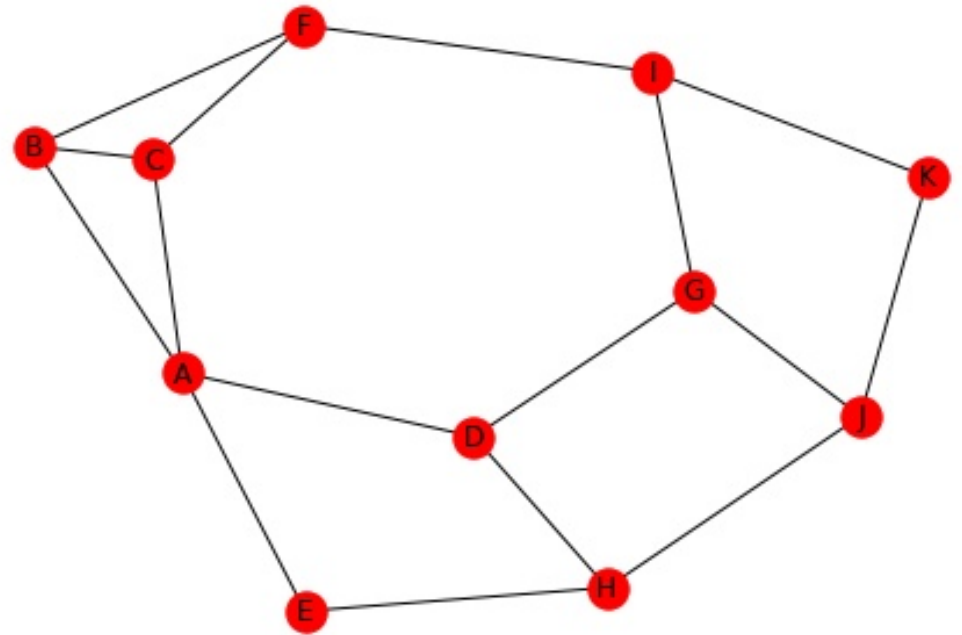
In the end divide all edge scores by two

Computed using NetworkX
(edge betweenness)

NetworkX code

```
import networkx as nx
g = nx.Graph()
g.add_edge("A", "B")
g.add_edge("A", "C")
g.add_edge("A", "D")
g.add_edge("A", "E")
g.add_edge("B", "C")
g.add_edge("B", "F")
g.add_edge("C", "F")
g.add_edge("D", "G")
g.add_edge("D", "H")
g.add_edge("E", "H")
g.add_edge("F", "I")
g.add_edge("G", "I")
g.add_edge("G", "J")
g.add_edge("H", "J")
g.add_edge("I", "K")
g.add_edge("J", "K")
nx.edge_betweenness(g, normalized=False)
```

`nx.draw_spring(g, with_labels=True)`



Exercise

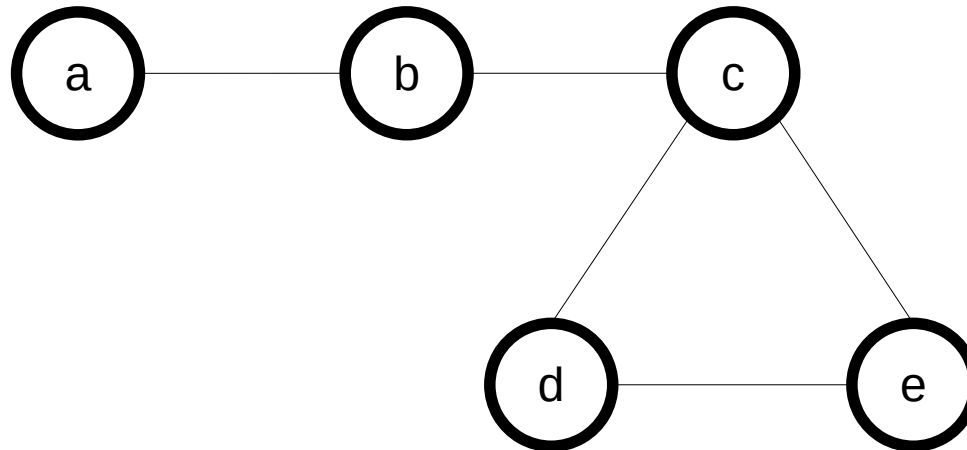
Try to compute it by inspection first

*Then use the algorithm;
you should get the same results*

For every node u in V

- Layer the graph performing a BFS from u
- For every node v in V , $v \neq u$, sorted by layer
 - Assign to v a number $s(v)$ indicating how many shortest paths from u arrive to v
- For every node v in V , $v \neq u$, sorted by reverse layer
 - Score to distribute = $1 + \text{score from children}$
 - Add score to distribute to parent edges in proportion to $s(v)$

In the end divide all edge scores by two



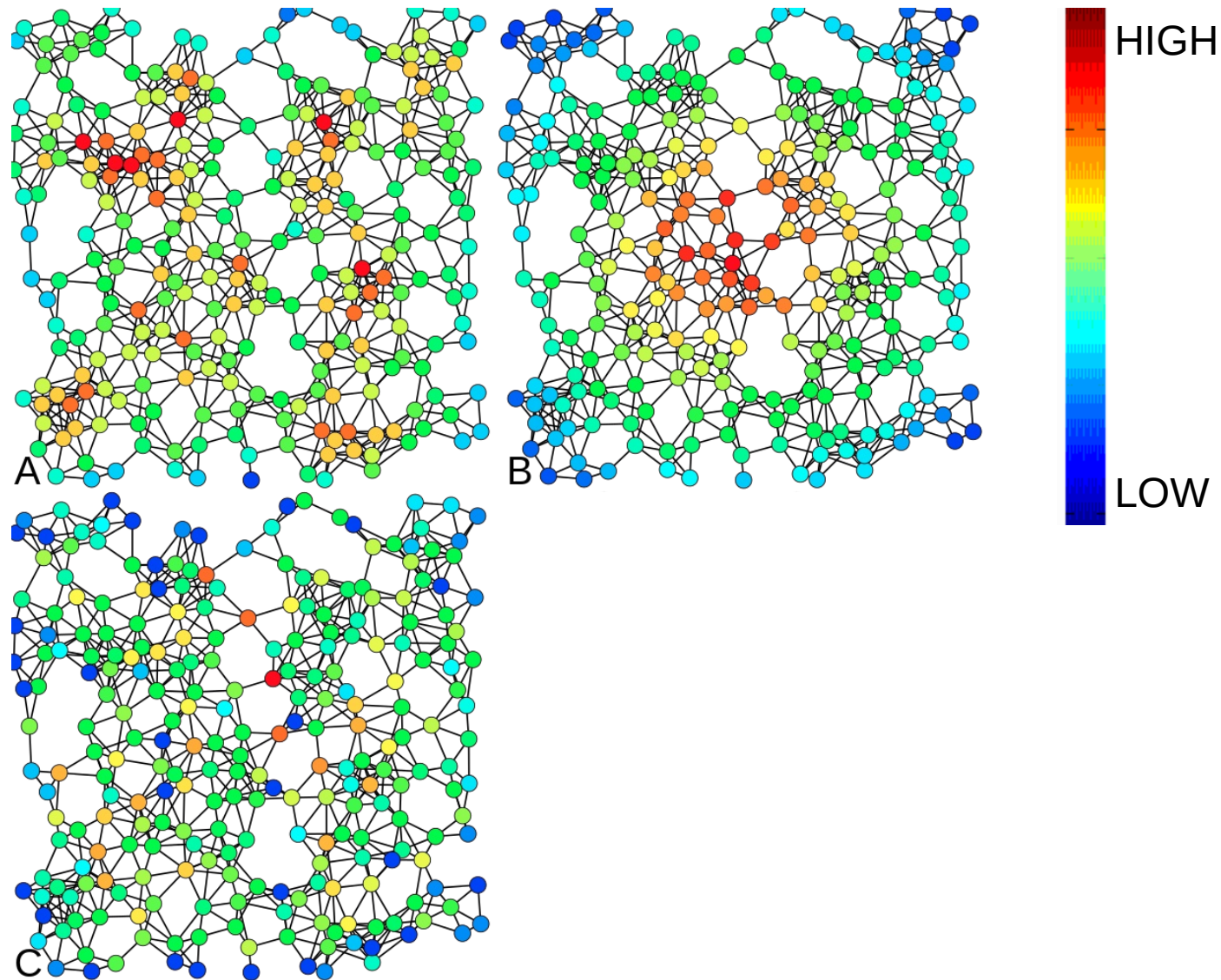
Fractional values?

- In a graph with cycles, you may get **fractional values** of the edge betweenness for an edge

A: Degree

B: Closeness

C: Betweenness



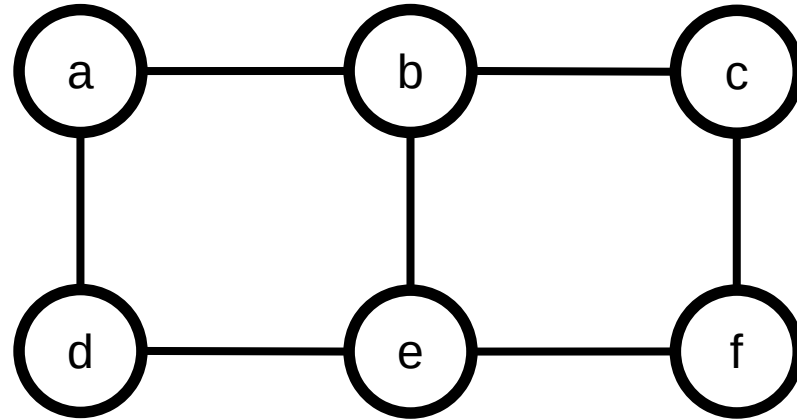
Summary

Things to remember

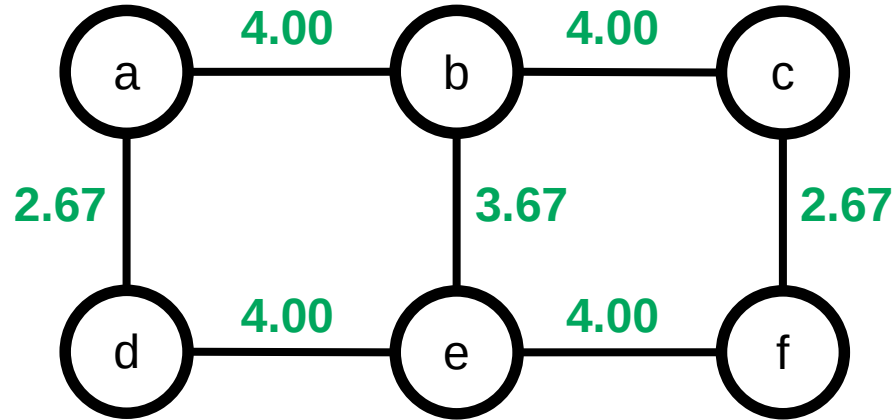
- Closeness and harmonic closeness
- Node and edge betweenness
- Practice running the Brandes-Newman algorithm on small graphs
- Write code to execute the Brandes-Newman algorithm

Practice on your own

- Compute edge betweenness on this graph



Practice on your own (cont.)



If you don't get this result, check:

<https://www.youtube.com/watch?v=uYjWbp8VC7c>

Two constructive problems

1. Sketch a graph of N nodes in which a node, which you should mark with an asterisk (*), should have betweenness approximately equal to N and closeness approximately $1/N$ for large N . Explain briefly.
2. Sketch a graph of N nodes in which a node, which you should mark with an asterisk (*), should have betweenness approximately equal to N and closeness approximately $2/N^2$ for large N . Explain briefly.

Do not use a concrete N . Use a general N , for instance by using the ellipsis (...) to denote multiple nodes.