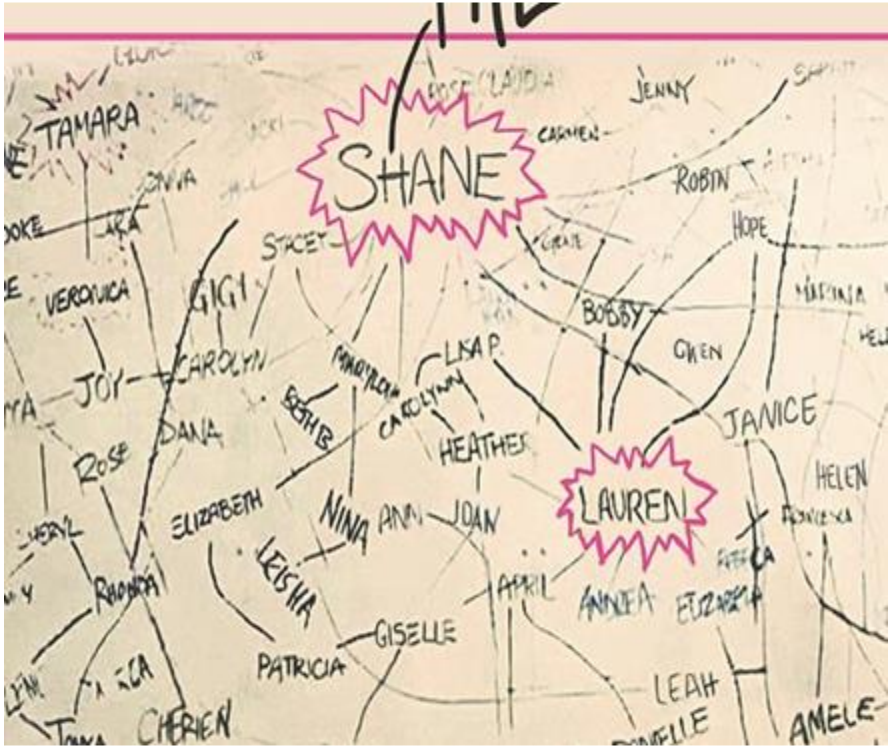
Graph Theory: Centrality

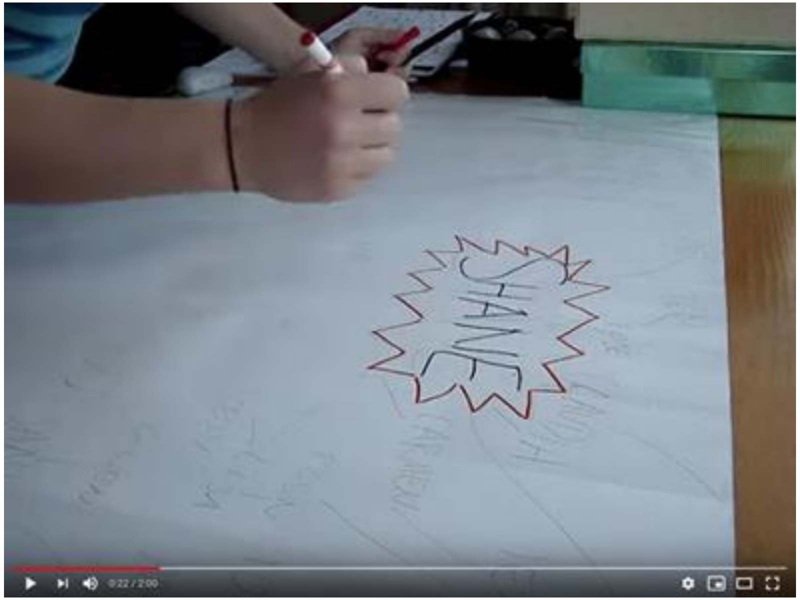
Introduction to Network Science

Instructor: Michele Starnini — https://github.com/chatox/networks-science-course

Universitat Pompeu Fabra Barcelona

A *central* question in networks is determining who is more ... *central*





https://youtu.be/wQ3TX65MnjM?t=22

"We are all connected through love, loneliness, or one tiny lamentable lapse of judgment"

Types of centrality measure

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

Is *u* a well-connected person?

- **Degree**: *u* has many connections
- **Closeness:** *u* is close to many people
    - Average distance from *u* is small
- **Betweenness**: many connections pass through *u*
    - Large number of shortest paths pass through *u*
- **PageRank**: *u is connected to the well-connected*

Closeness

Closeness

- Distance between two nodes is *d(u,v)*

- **Closeness** is the reciprocal of the sum of distances

$$\text{closeness}(u) = \frac{1}{\sum_{v \in V, v \neq u} d(u,v)}$$

- Some graphs are not connected, in that case *d(u,v)* can be ∞; assuming *1/∞ = 0* one can define the **harmonic closeness**:

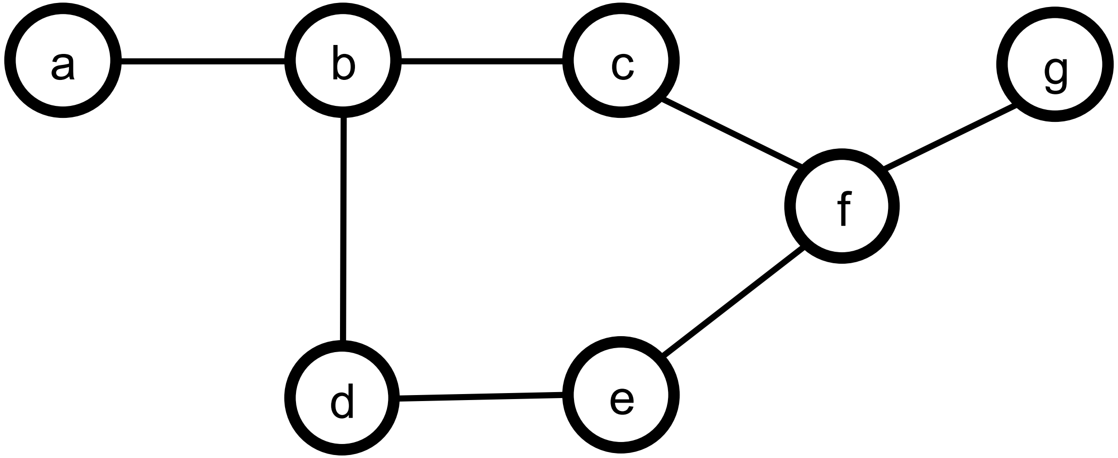
$$\text{hcloseness}(u) = \sum_{v \neq u} \frac{1}{d(u,v)}$$

Exercise

Compute closeness and harmonic closeness for all the nodes; d(u,v) = 1 if v is a neighbor of u

$$\text{closeness}(u) = \frac{1}{\sum_{v \in V, v \neq u} d(u,v)}$$

$$\text{hcloseness}(u) = \sum_{v \in V, v \neq u} \frac{1}{d(u,v)}$$



Spreadsheet links: https://upfbarcelona.padlet.org/chato/shyq9m6f2g2dh1bw

Betweenness

Definitions

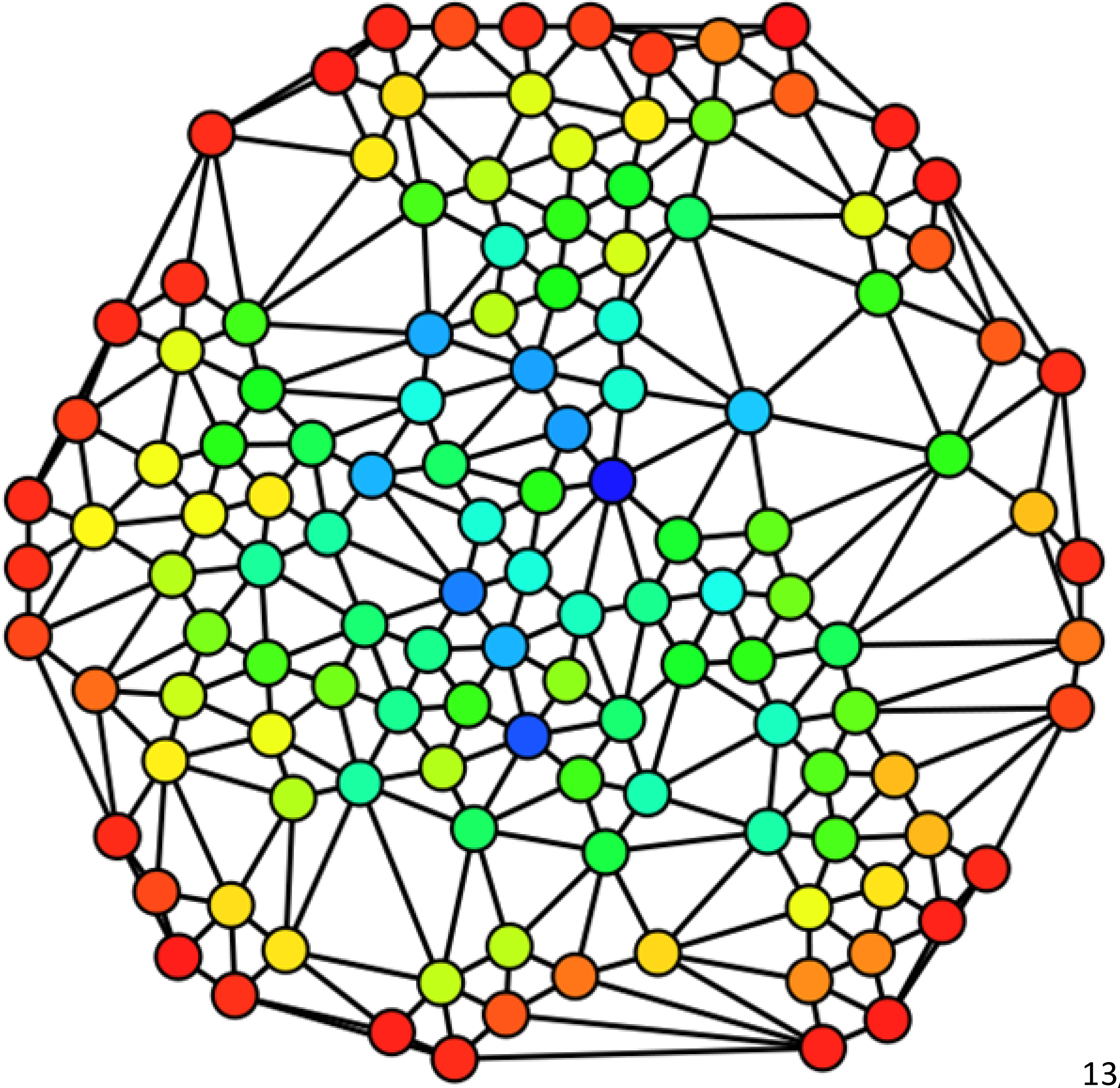
The **betweenness of a node** is the number of shortest paths that cross that node

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

Node Betweenness

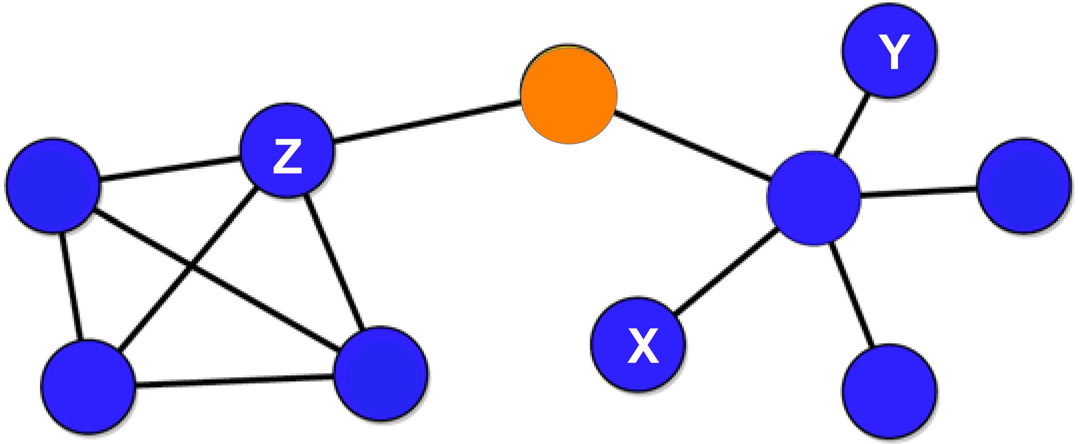
Graph with nodes colored according to node betweenness

red=low, blue=high



[Wikipedia: Betweenness centrality]

Example 1



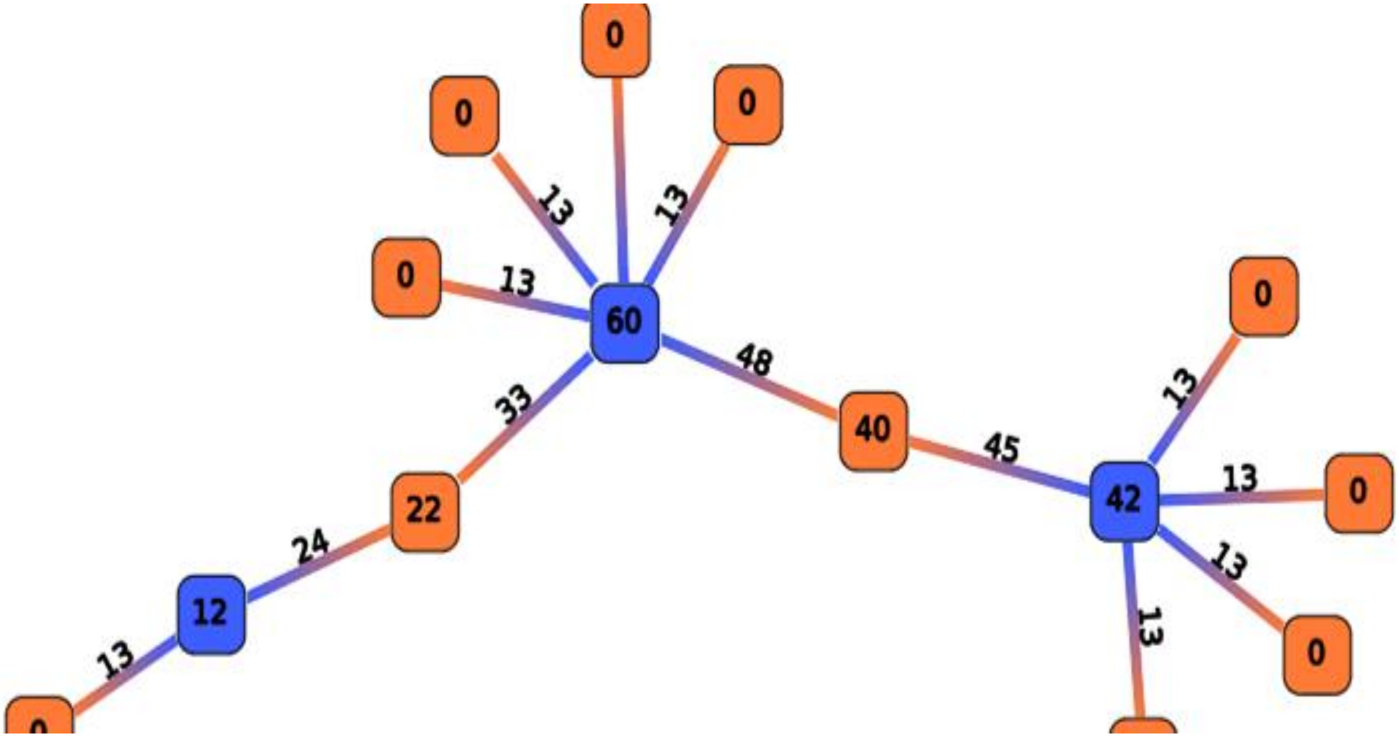
There are 20 shortest paths that cross through the orange node. Why?

The shortest path between nodes X and Y does not cross the orange node, but the shortest path between nodes X and Z does cross the orange node.

Source: Menczer, Fortunato, Davis: A First Course on Networks Science. Cambridge, 2020.

Example 2

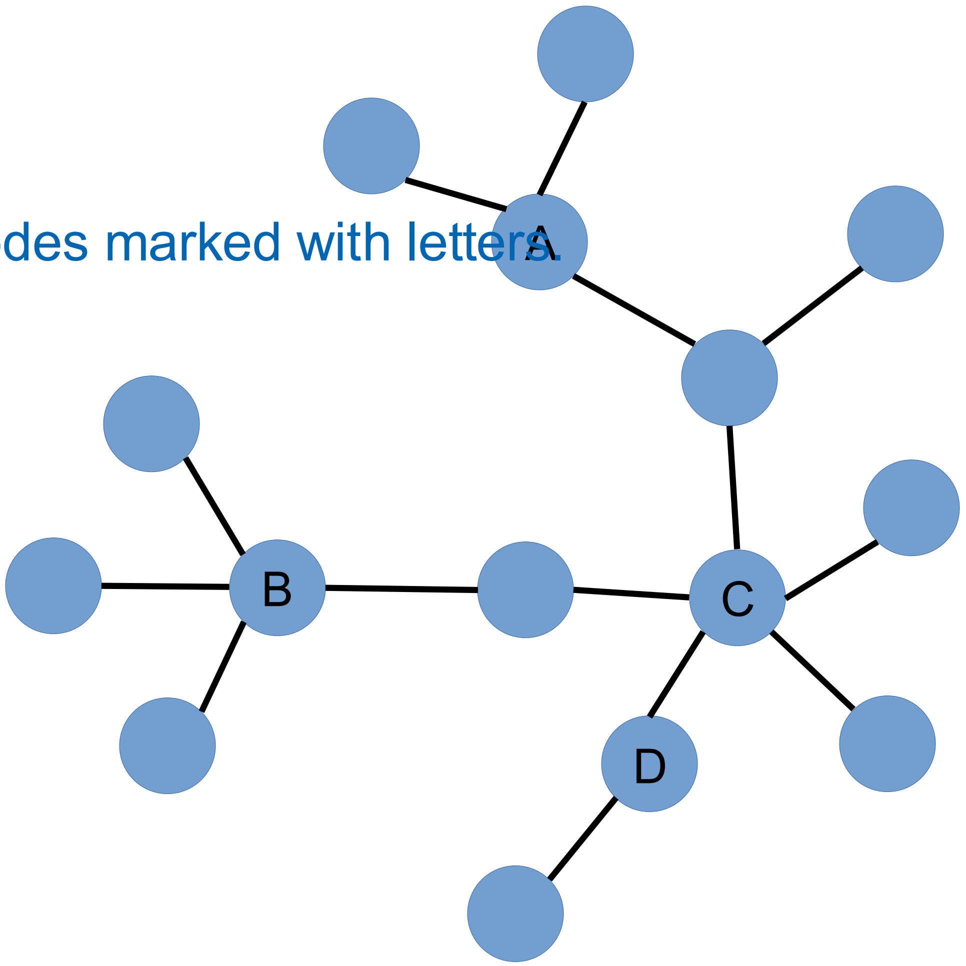
Here, nodes and edges are labeled with their betweenness.



http://rnav.labri.fr/rNAV2.0_public_version/online_doc/_images/ComputeBetweennessCentrality.png

Exercise

Compute the node betweenness of the nodes marked with letters.



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

Algorithms

Floyd–Warshall algorithm

SP(i,j,k) = SP from i to j, using nodes {1,2,...,k}

SP(i,j) = SP(i,j, N)

SP(i,j,k) smaller or equal to SP(i,j,k-1)

IF SP(i,j,k) < SP(i,j,k-1) THEN I passed through node k

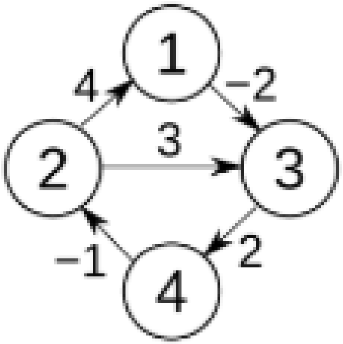
IF SP(i,j,k) < SP(i,j,k-1) THEN   SP(i,j,k) = SP(i,k,k-1) + SP(k,j,k-1)

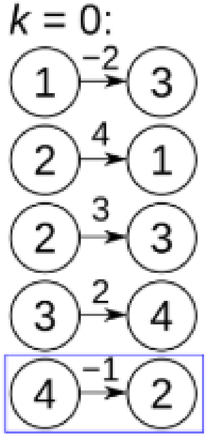
SP(i,j,k) = min{ SP(i,j,k-1), SP(i,k,k-1) + SP(k,j,k-1)}

Recursive algorithm

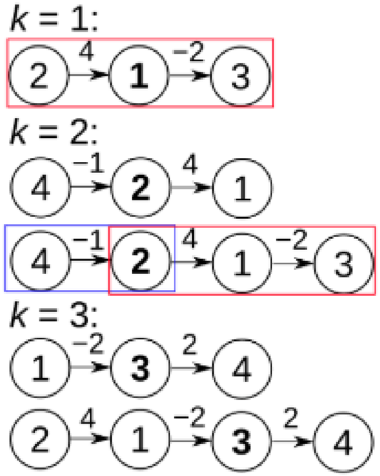
Floyd–Warshall algorithm

SP(i,j,k) = min{ SP(i,j,k-1), SP(i,k,k-1) + SP(k,j,k-1)}   (No negative cycles)





$k=0$		j			
		1	**2**	**3**	**4**
i	**1**	0	∞	−2	∞
	2	4	0	3	∞
	3	∞	∞	0	2
	4	∞	−1	∞	0



$k=1$		j			
		1	**2**	**3**	**4**
i	**1**	0	∞	−2	∞
	2	4	0	**2**	∞
	3	∞	∞	0	2
	4	∞	−1	∞	0

Complexity: O(N^3) = O(N^2) [SP(i,j,k), for all (I,j)] \times N (k=1,2,...,N)

Betweenness centrality

$$C_B(v) = \sum_{s \in V} \sum_{t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

$sigma_{st}(v)$ = number SP from s to t, passing through v

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$$

$delta_{st}(v)$ = pair dependency of s,t on v
(proportion of SP(st) through v)

$$\delta_s(v) = \sum_{t \in V} \delta_{st}(v),$$

$delta_s(v)$ = (single) dependency on v wrt origin s
(SP originated at s, which involve v)

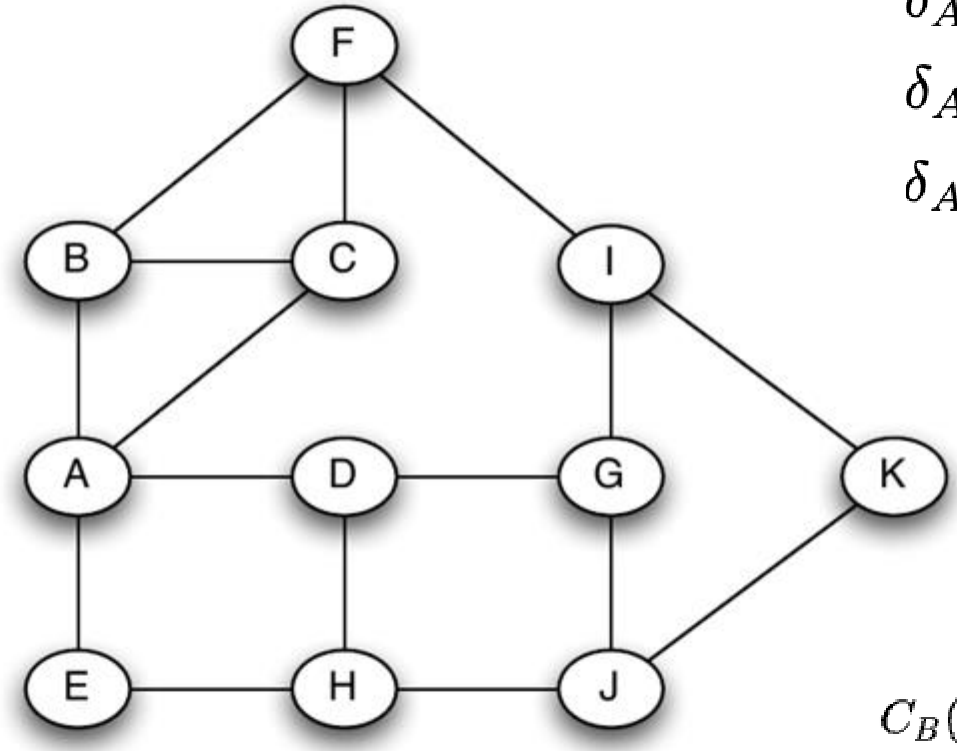
$$C_B(v) = \sum_{s \in V} \delta_s(v).$$

BC(v) is the sum of the dependencies
on wrt all paths

Example

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$$

pair dependency of s,t on v
(proportion of SP(st) through v)



$$\delta_{AF}(B) = 1/2 \qquad \delta_{Ax}(B) = 0,$$

$$\delta_{AI}(B) = 1/3 \qquad x \in \{A, B, C, D, E, G, H, J\}$$

$$\delta_{AK}(B) = 1/6$$

(single)dependency on v wrt origin s
(SP originated at s, which involve v)

$$\delta_s(v) = \sum_{t \in V} \delta_{st}(v), \qquad \delta_A(B) = 1$$

$$C_B(v) = \sum_{s \in V} \delta_s(v).$$

BC(v) is the sum of the dependencies on all sources

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*≠*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*≠*u*, sorted by reverse layer
- Score to distribute = 1 + score from children
- Add score to parent edges in proportion to *s(v)*
- In the end divide all edge scores by two

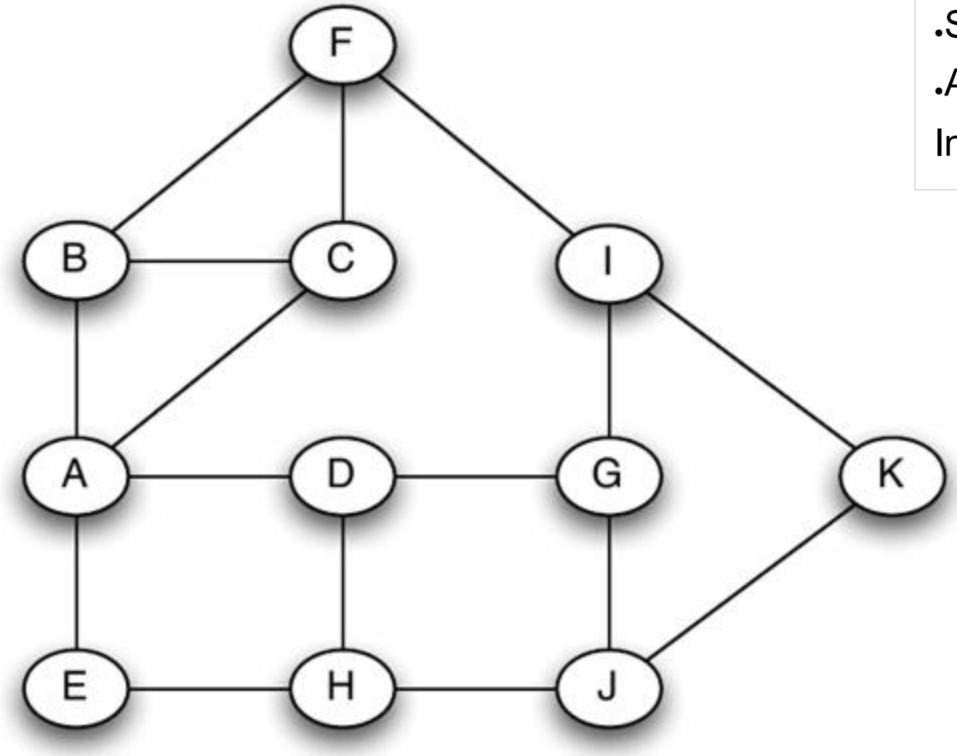
Ulrich Brandes. A faster algorithm for betweenness centrality. Journal of Mathematical Sociology, 25:163–177, 2001.
Mark E. J. Newman. Scientific collaboration networks: II. Shortest paths, weighted networks, and centrality. Physical Review E, 64:016132, 2001.

Example

For every node u in V

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

In the end divide all edge scores by two

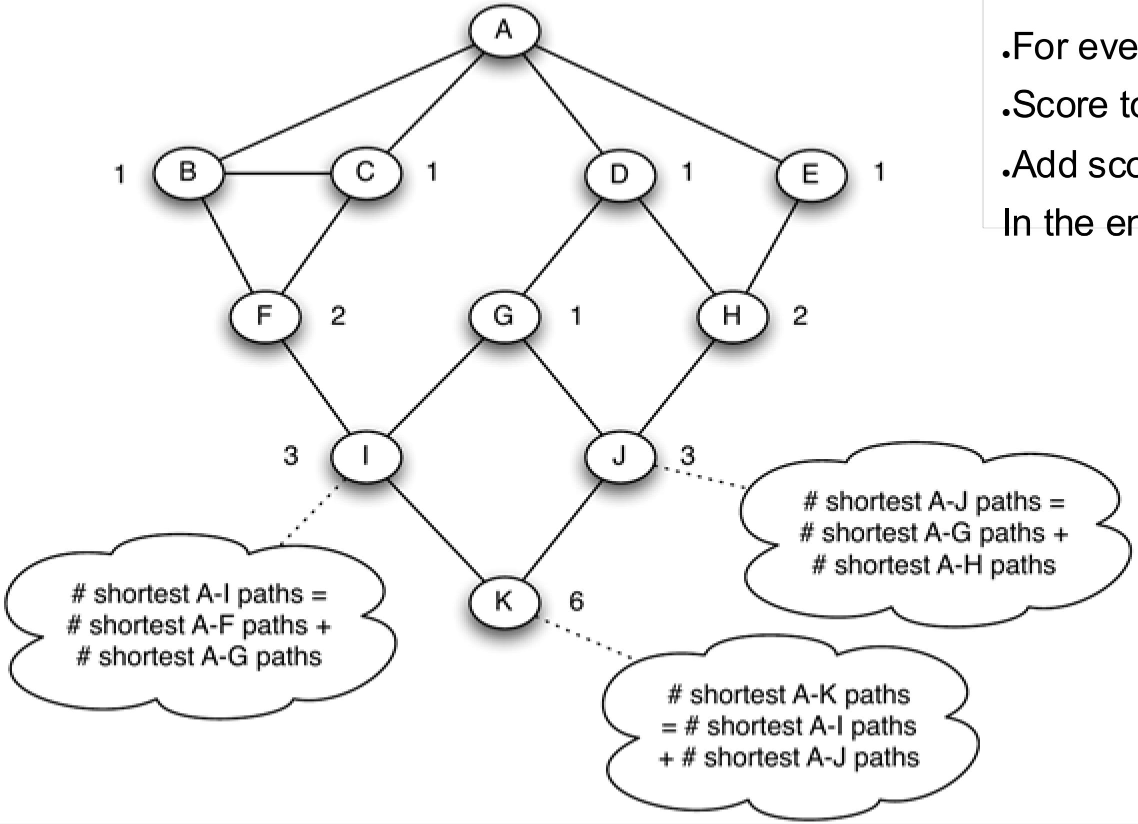


Example

For every node u in V

- Layer the graph performing a BFS from u
- **For every node v in V, v≠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≠u, sorted by reverse layer
- Score to distribute = 1 + score from children
- Add score to distribute to parent edges in proportion to s(v)

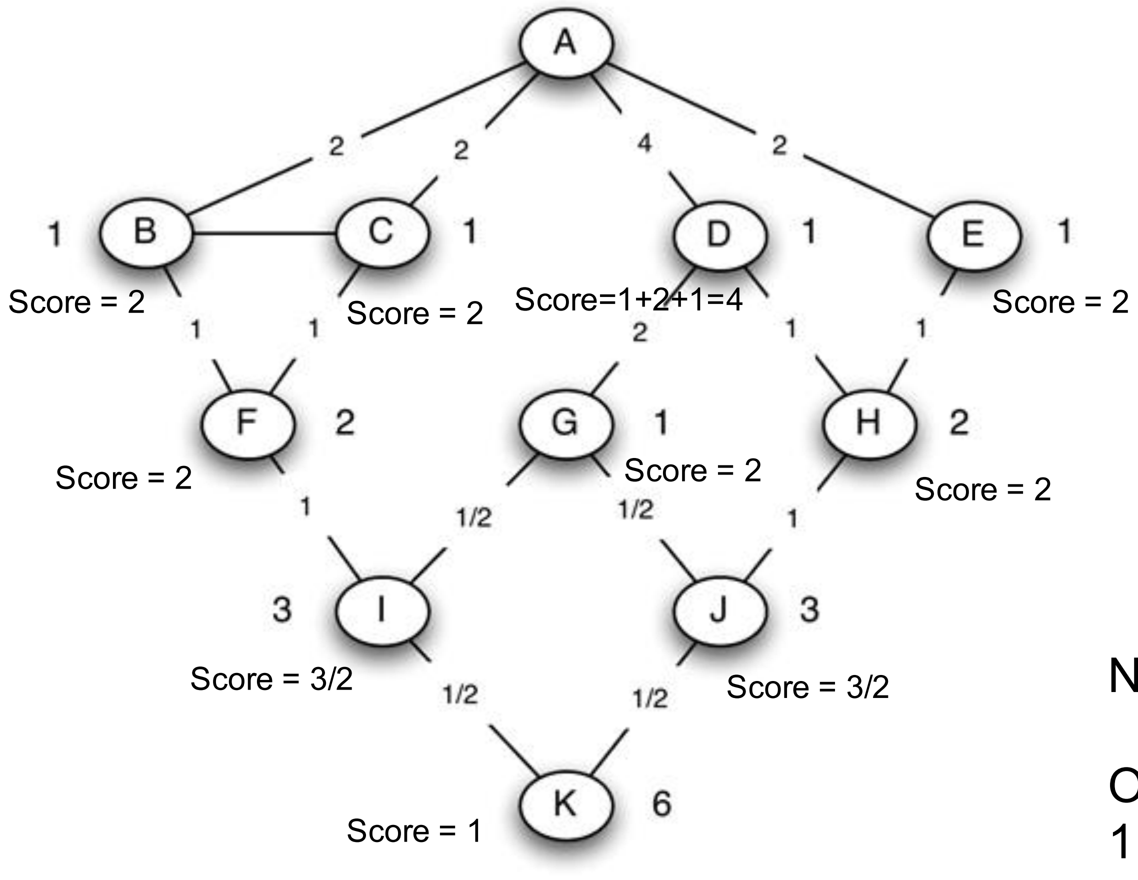
In the end divide all edge scores by two



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

Remaining nodes: simply add s(.) 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≠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≠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

From children to parents:

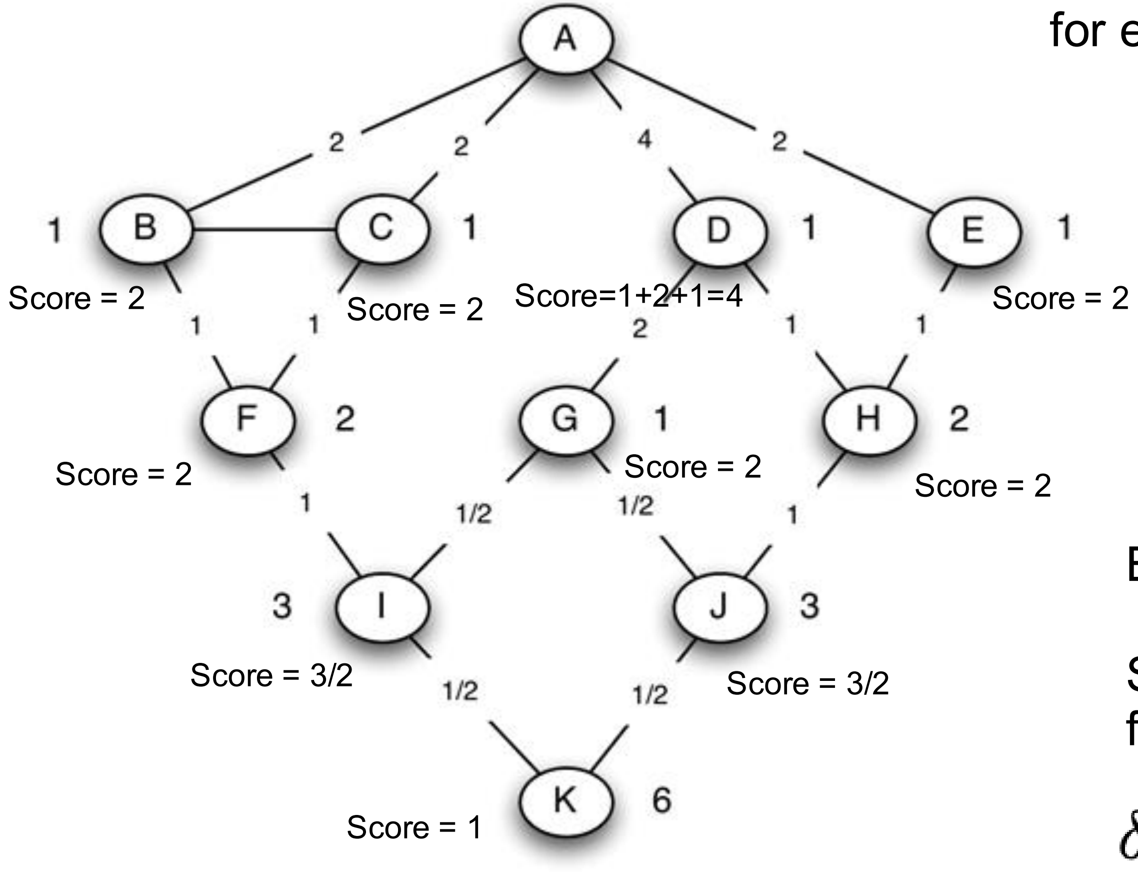
Nodes without children distribute a score of 1

Other nodes distribute:
1 + whatever they receive from their children

Example

Betweenness of edge A-B =

Sum of the scores obtained by edge A-B,
for each source (with source = A, score = 2)



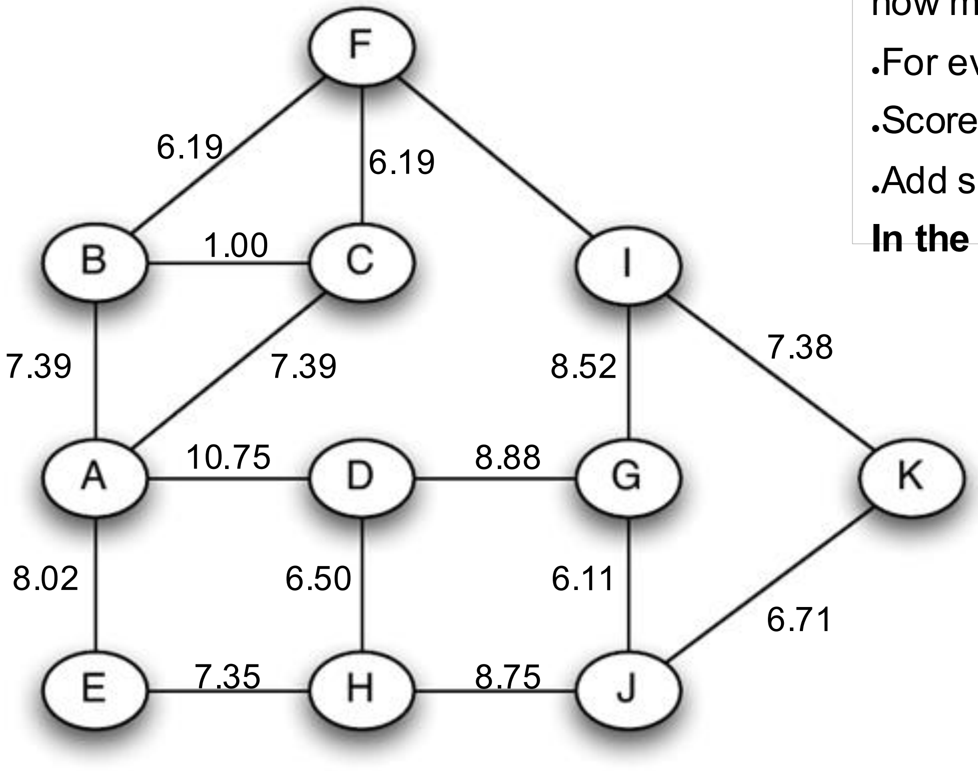
$$B(v) = \sum_{s \in V} \delta_s(v)$$

Betweenness of node B =

Sum of 1 - scores obtained by node B,
for each source (source = A, score = 2-1=1)

$\delta_A(B) = 1$ $\delta_A(D) = 3$

Result



For every node u in V

- Layer the graph performing a BFS from u
- For every node v in V, v≠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≠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

Computed using NetworkX (edge betweenness)

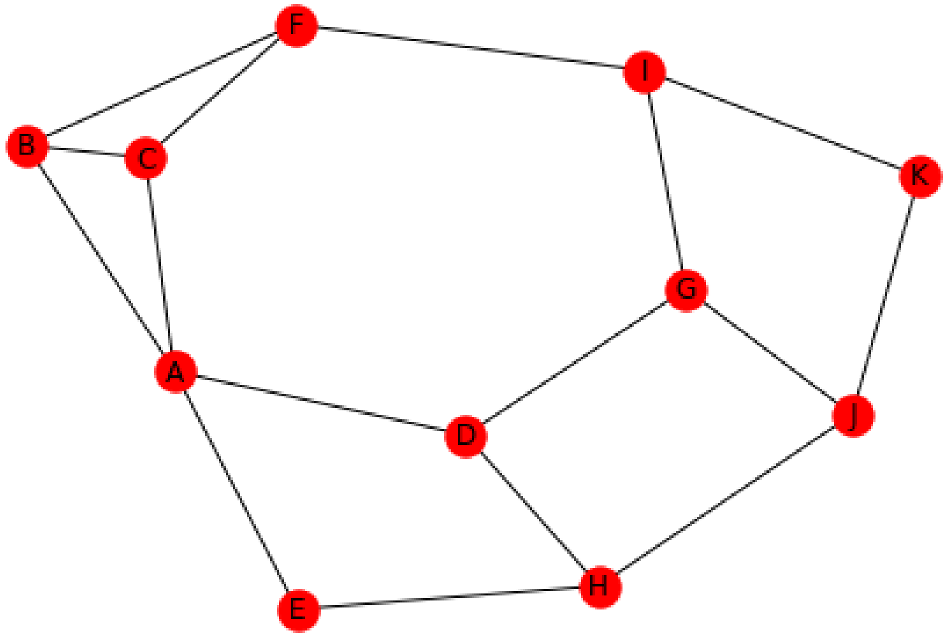
$$B(v) = \sum_{s \in V} \delta_s(v)$$

Complexity: O(N E) = (O(N + E) [BFS] + O(E) [backpropagation]) \times N (s=1,2,...,N)

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")
…
nx.edge_betweenness(g, normalized=False)
```

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



Exercise

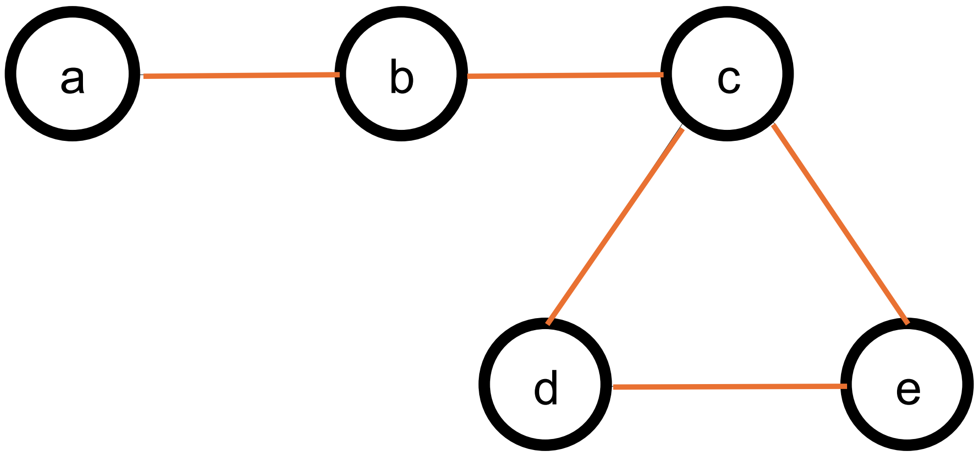
*Try to compute **edge betweenness** by inspection first*

Then use the Brandes-Newman 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≠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≠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



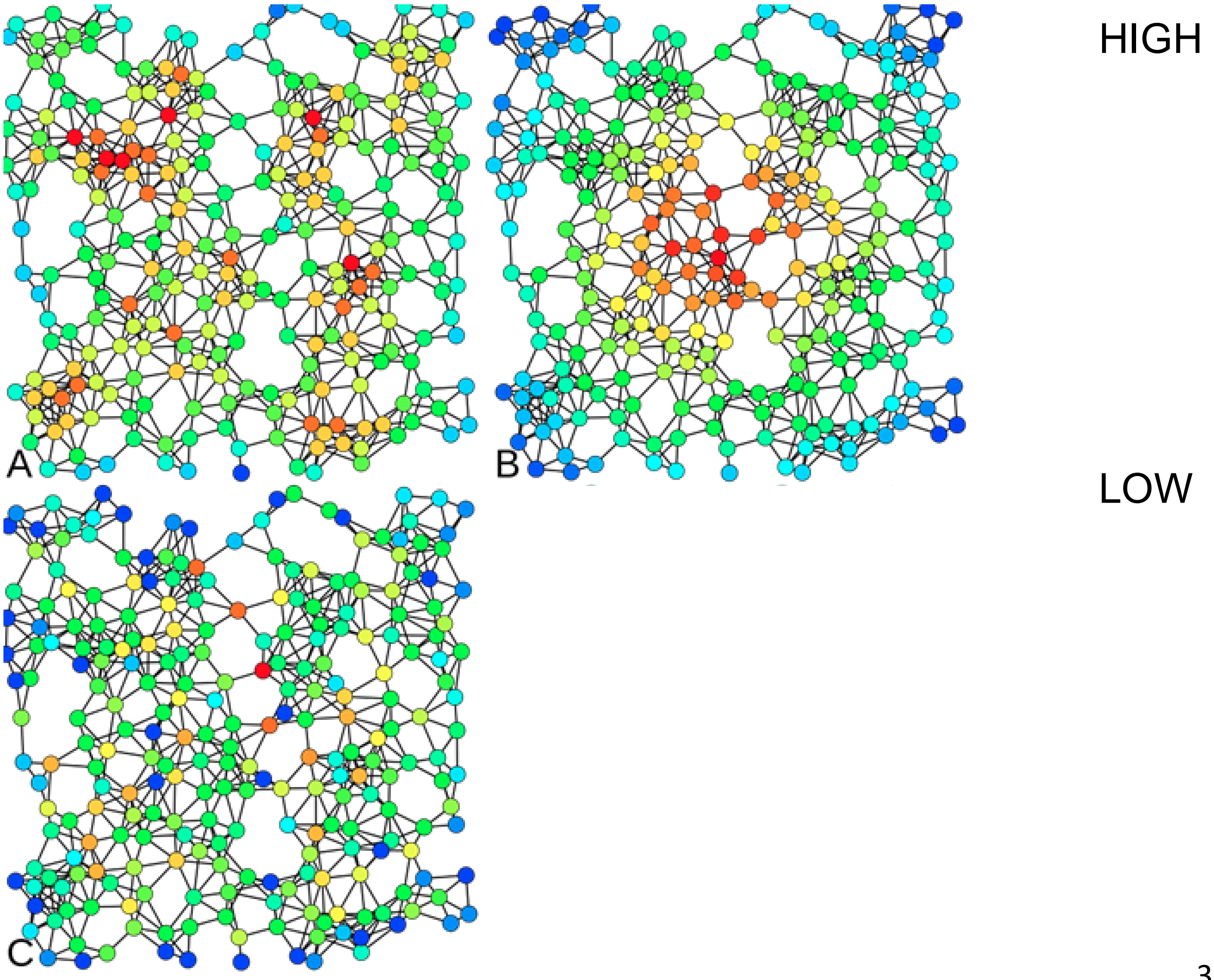
Fractional values?

- In a graph with cycles, you may get **fractional values** of the edge betweenness for an edge
- Conceptually, this is because in a graph with cycles there might be s>1 shortest paths between two nodes, each of them counts 1/s

A: Degree

B: Closeness

C: Betweenness



Summary

Things to remember

- Closeness and harmonic closeness
- Node and edge betweenness
- Floyd–Warshall & Brandes-Newman algorithms
- Practice running the BN algorithm on small graphs
- Write code to execute the BN algorithm

Constructive problems

- Practice drawing examples of graphs in which a chosen node has high degree but low closeness, or viceversa

- Can you find a graph in which there is a node that has the maximum degree and the minimum closeness? If not, why?

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.

Sources

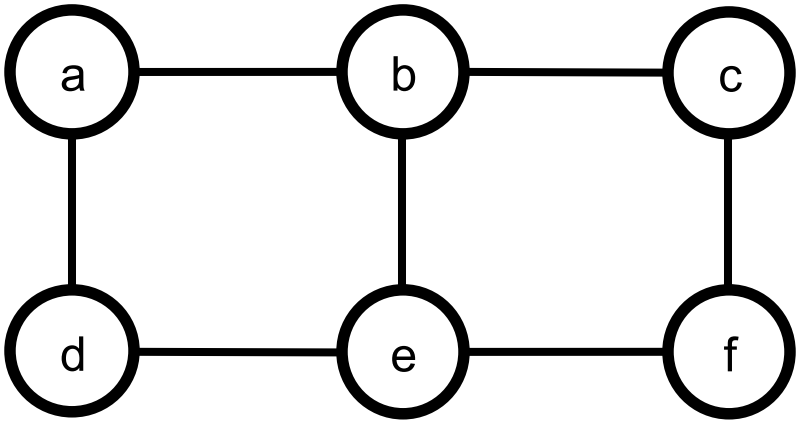
- D. Easly and J. Kleinberg (2010). Networks, Crowds, and Markets – Section 3.6B
- A. L. Barabási (2016). Network Science – Section 9.3
- P. Boldi and S. Vigna (2014). Axioms for Centrality in *Internet Mathematics*
- Esposito and Pesce: Survey of Centrality 2015.
- URLs cited in the footer of slides

Sources

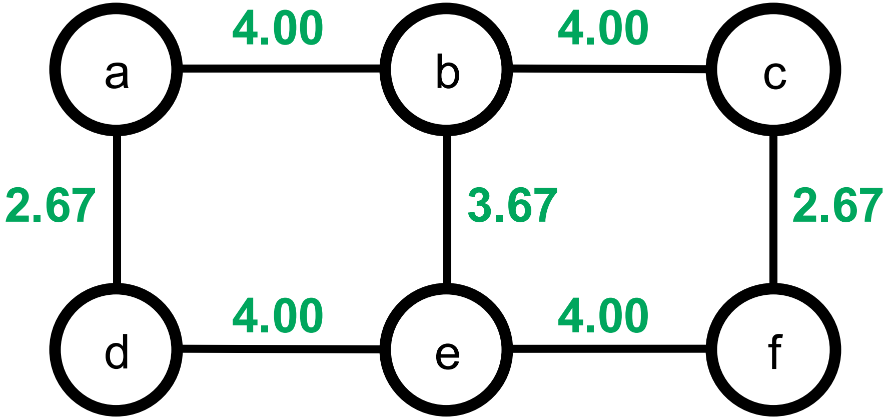
- D. Easley and J. Kleinberg (2010). Networks, Crowds, and Markets – Section 3.6B
- P. Boldi and S. Vigna (2014). Axioms for Centrality in *Internet Mathematics*.
- Esposito and Pesce (2015): Survey of Centrality.
- F. Menczer, S. Fortunato, C. A. Davis (2020). A First Course in Network Science – Chapter 02

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