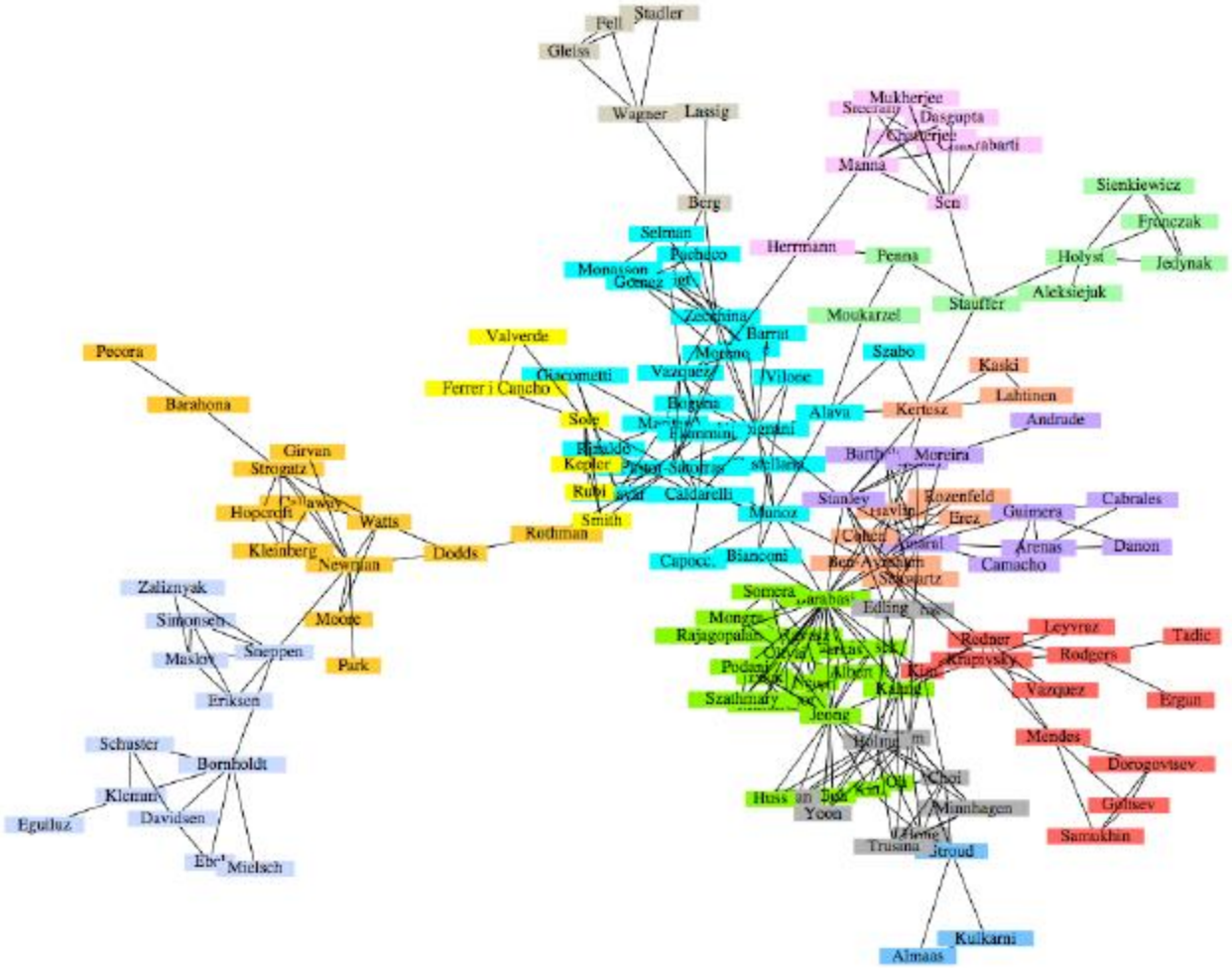
Betweenness Measures and Graph Partitioning

Objectives

- Define densely connected regions of a network
- Graph partitioning
  - Algorithm to identify densely connected regions
  - breaking a network into a set of nodes densely connected with each other with edges
  - having sparser interconnections between the regions

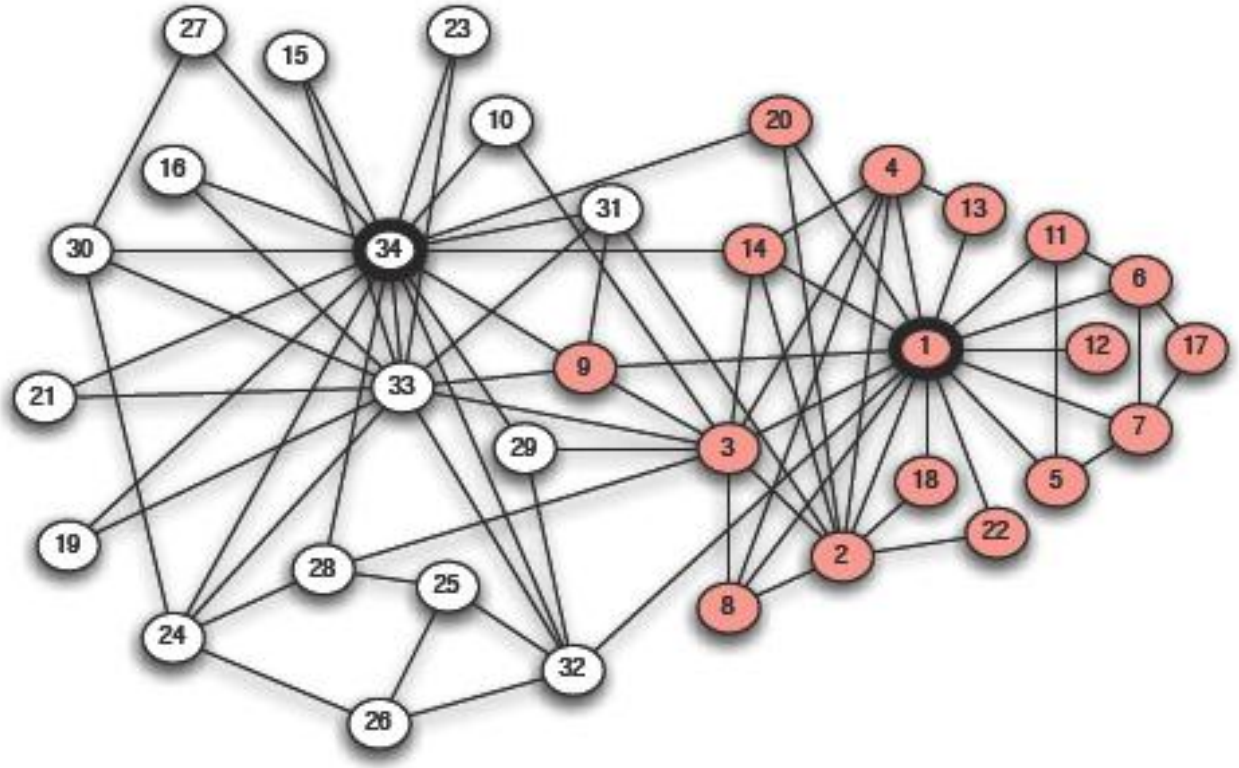
Graph partitioning example



A co-authorships network among a set of physicists

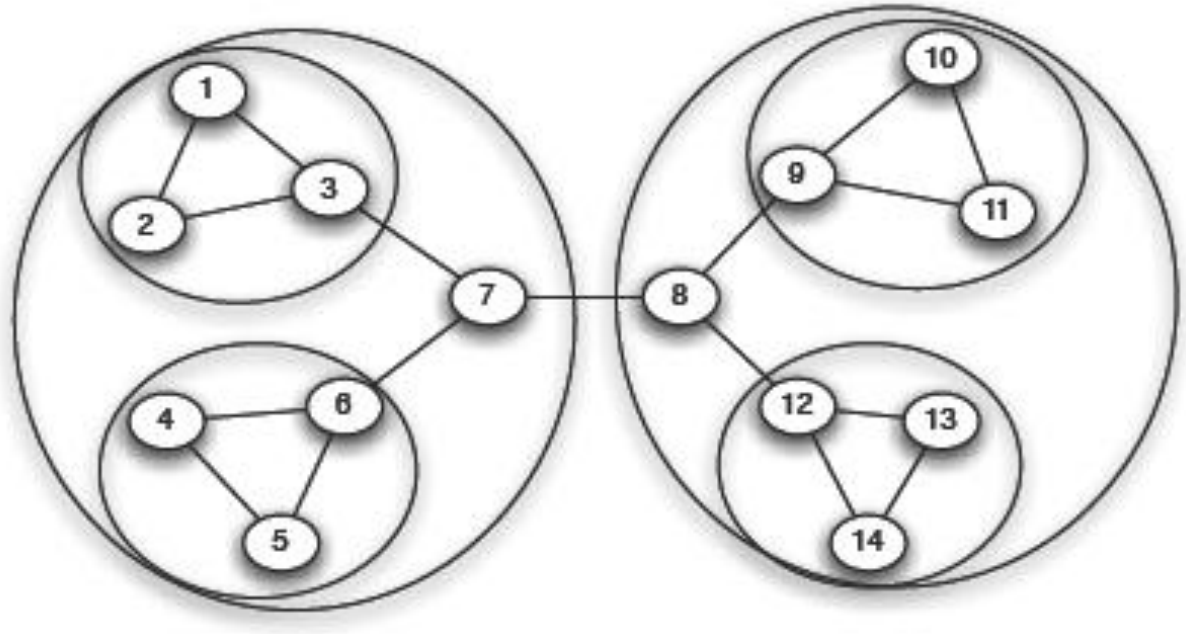
Graph partitioning example

the 2 conflicting groups are still heavily interconnected
Need to look how edges between groups occur at lower “density” than edges within the groups



social network of a karate club

Nesting of regions



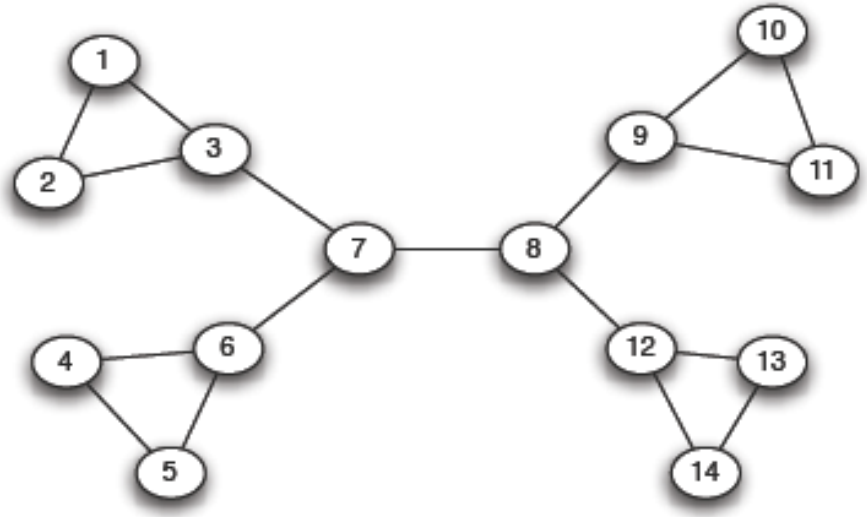
Larger regions containing several smaller

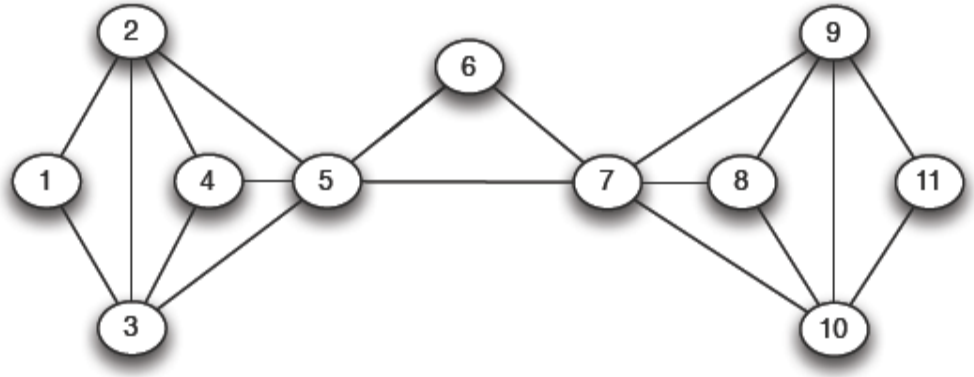
Divisive methods: breaking first at the 7-8 edge, and then the nodes into nodes 7 and 8

Agglomerative methods: merge the 4 triangles and then pairs of triangles (via nodes 7 and 8)

Divisive removal of Bridges

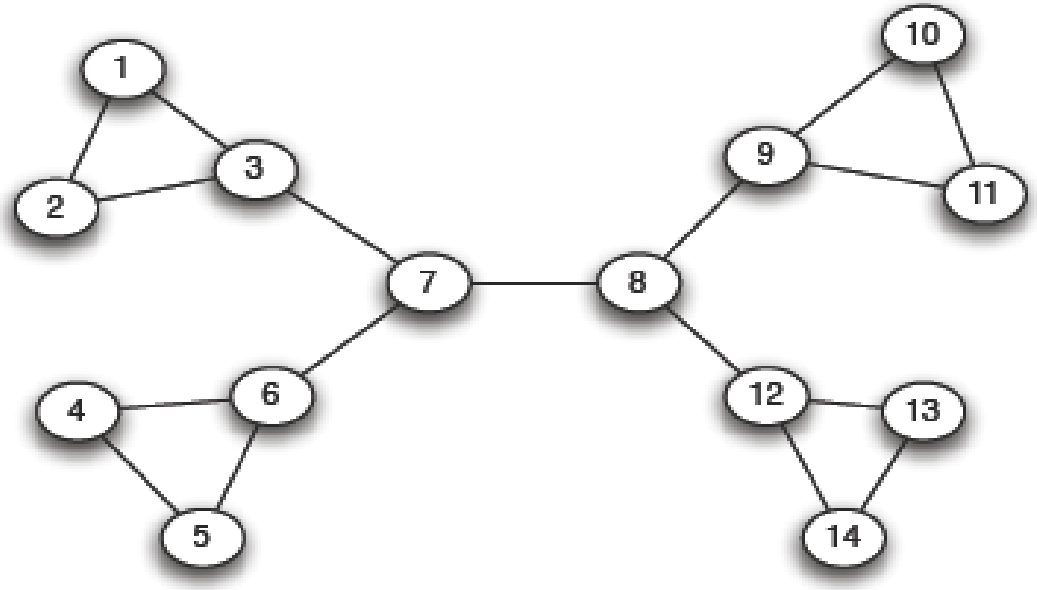
- Simple idea:
  - remove bridges and local bridges
- Problems:
  - which when several? (ex: in fig up 5 bridges)
  - what if none (ex: in fig down nodes 1-5 and 7-11)





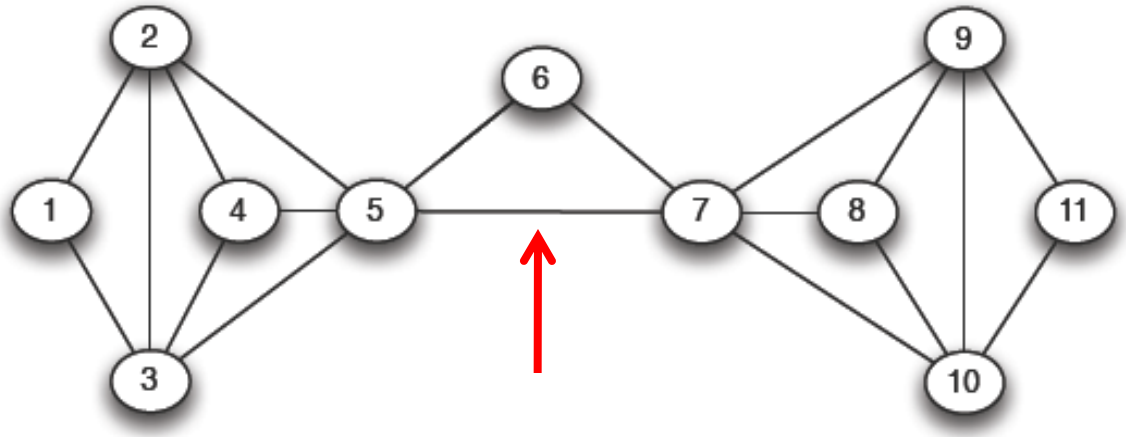
The role of Bridges

- Q: What bridges and local bridges are doing?
- A: They form part of the shortest path between pairs of nodes in different parts of the network



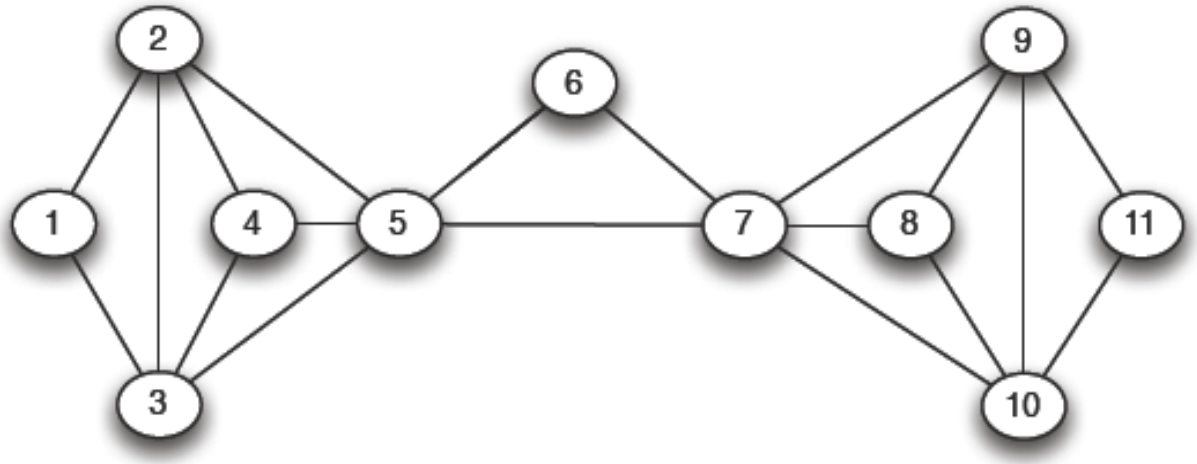
Generalize the Role of Bridges

- Look for the edges that carry the most of "traffic" in a network
  - without the edge, paths between many pairs of nodes may have to be "re-routed" a longer way
  - edges to link different densely-connected regions
  - good candidates for removal in a divisive method
  - generalize the (local) bridges



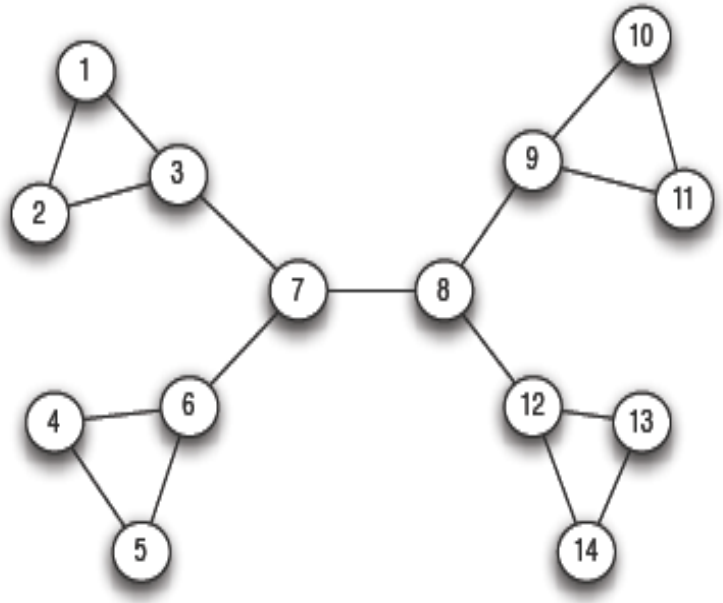
Traffic in a Network

- For nodes A and B connected by a path assume 1 unit of "flow"
  - (If A and B in different connected components, flow = 0)
- Divide flow evenly along all possible shortest paths from A to B
  - if k shortest paths from A and B, then 1/k units of flow pass along each
- Ex: 2 shortest paths from 1 to 5, each with 1/2 units of flow



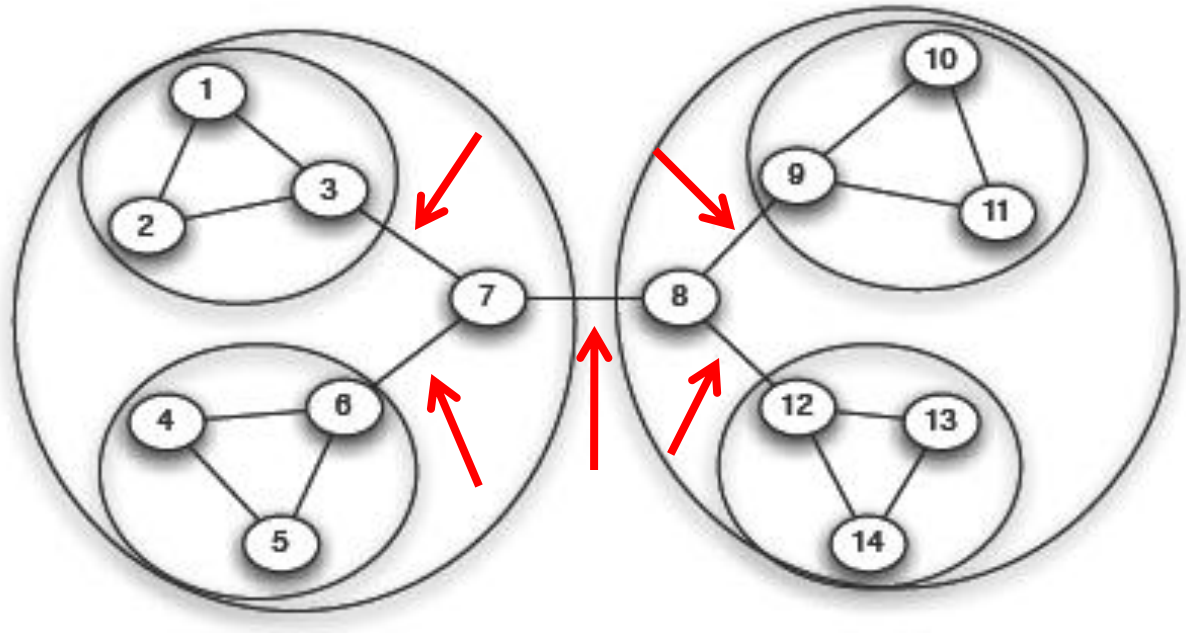
Edge Betweenness

- Betweenness of an edge: the total amount of flow it carries
  - counting flow between all pairs of nodes using this edge
- Ex:
  - Edge **7-8**: each pair of nodes between [1-7] and [8-14]; each pair with traffic = 1; total 7 x 7 = 49
  - Edge **3-7**: each pair of nodes between [1-3] and [4-14]; each pair with traffic = 1; total 3 x 11 = 33
  - Edge **1-3**: each pair of nodes between [1] and [3-14] (not node 2); each pair with traffic = 1; total 1 x 12 = 12
    - similar for edges 2-3, 4-6, 5-6, 9-10, 9-11, 12-13, and 12-14
  - Edge **1-2**: each pair of nodes between [1] and [2] (no other); each pair with traffic = 1; total 1 x 1 = 1
    - similar for edges 4-5, 10-11, and 13-14



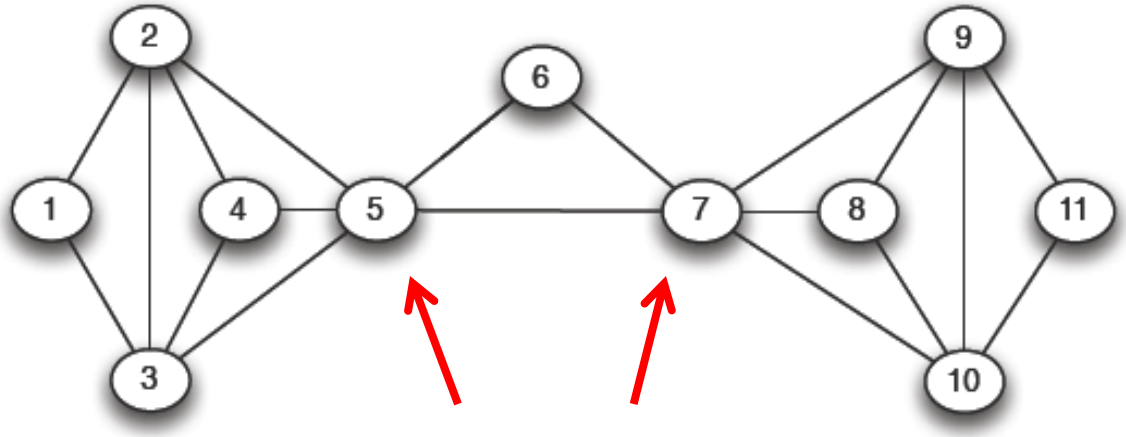
Betweenness for Partitioning

- Divisive: remove edges with high betweenness



Betweenness of Nodes

- Betweenness of a node: total amount of flow that it carries, when a unit of flow between each pair of nodes is divided up evenly over shortest paths (same as for edges)
  - nodes of high betweenness occupy critical roles in the network ("gatekeepers")



Girvan-Newman Partitioning Alg.

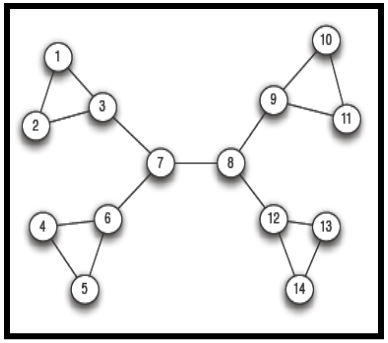
Successively Deleting Edges of High Betweenness

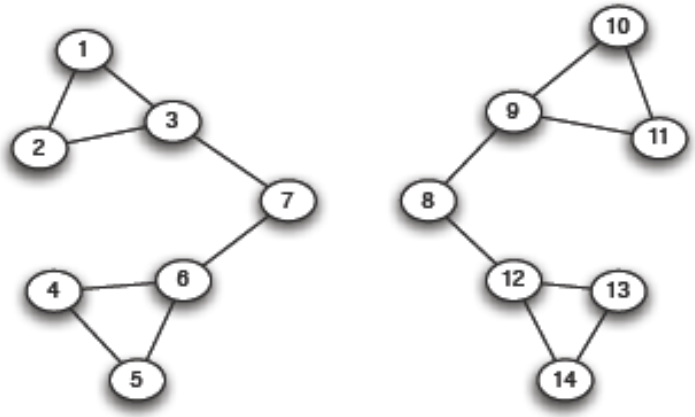
(1) Find the edge of highest betweenness — or multiple edges of highest betweenness, if there is a tie — and remove these edges from the graph. This may cause the graph to separate into multiple components. If so, this is the first level of regions in the partitioning of the graph.

(2) Now recalculate all betweennesses, and again remove the edge or edges of highest betweenness. This may break some of the existing components into smaller components; if so, these are regions nested within the larger regions.

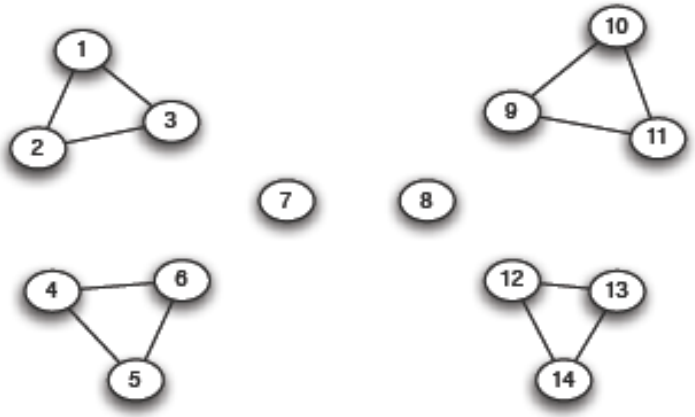
(...) Proceed in this way as long as edges remain in graph, in each step recalculating all betweennesses and removing the edge or edges of highest betweenness.

Example 1

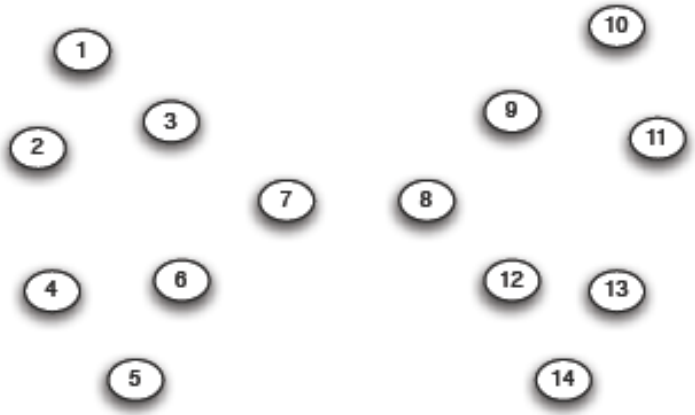




(a) *Step 1*

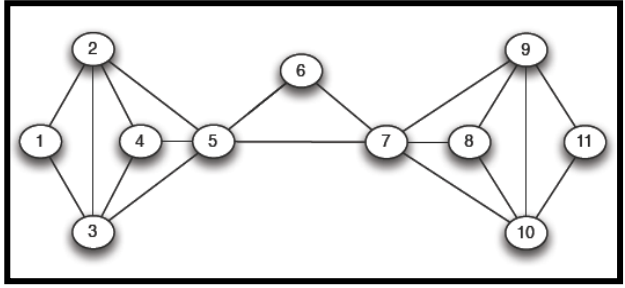


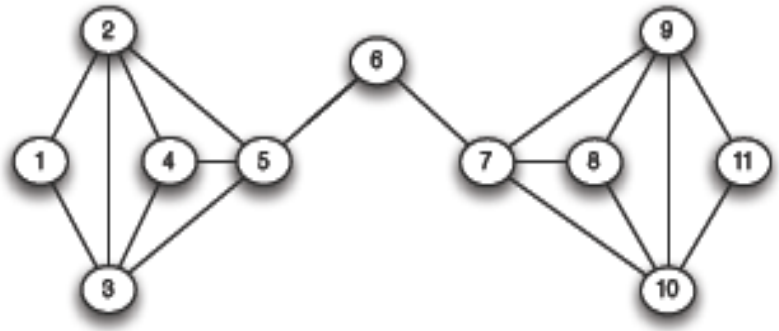
(b) *Step 2*



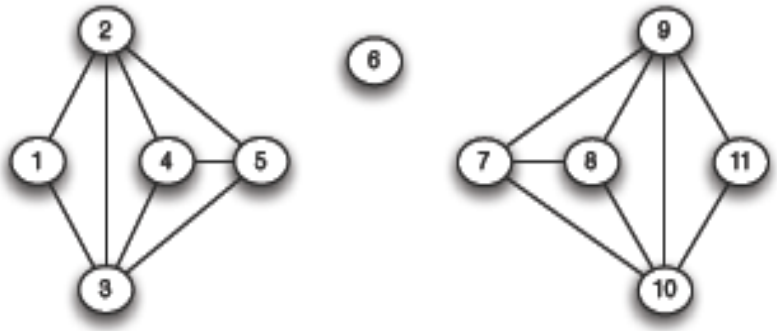
(c) *Step 3*

Example 2

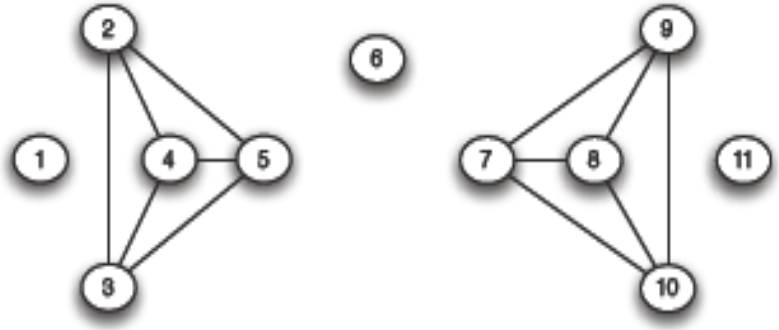




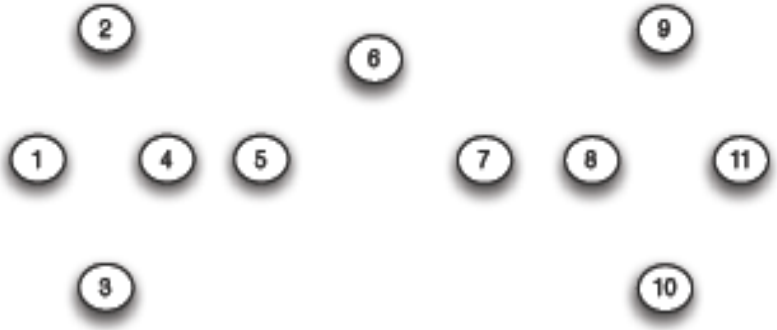
(a) *Step 1*



(b) *Step 2*



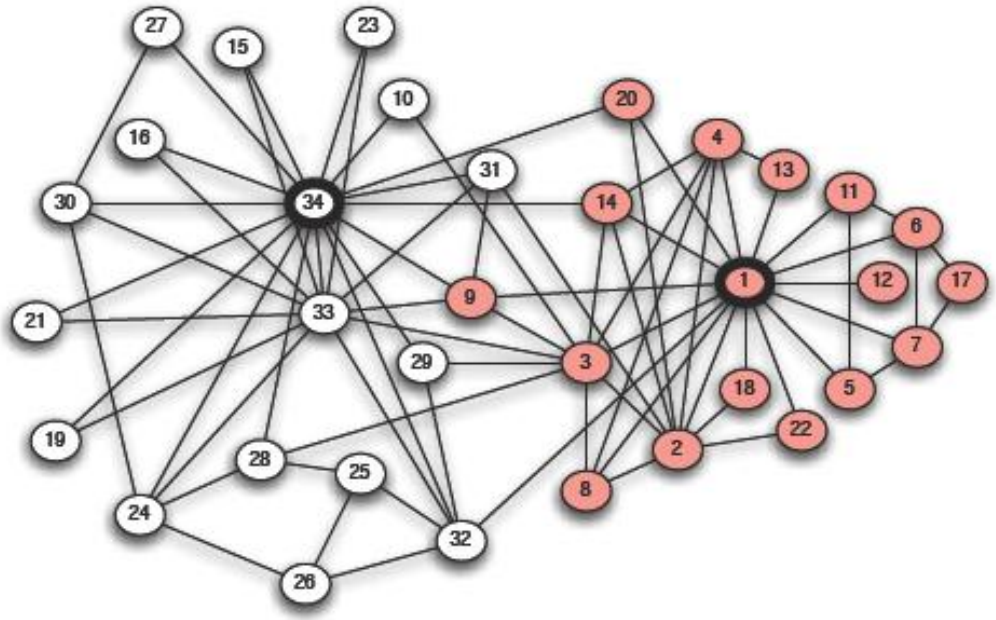
(c) *Step 3*



(d) *Step 4*

Example 3

- Girvan-Newman partitions correctly
  - exception: node 9 assigned to region of 34 (left part)
  - at the time of conflict, node 9 was completing a four-year quest to obtain a black belt, which he could only do with the instructor (node 1)



Partitioning large Social Networks

- In real social network data, partitioning is easier when network is small (at most a few hundred nodes)
- In large networks, nodes become much more "inextricable"
- Open research problem

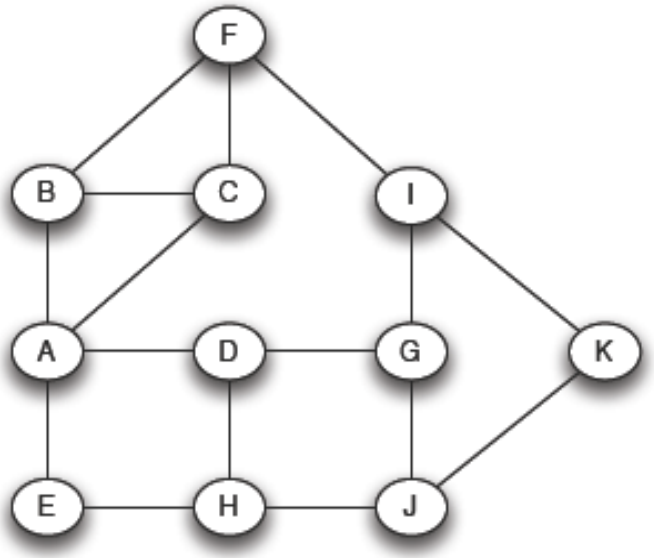
Computing Betweenness Values

- According to definition: consider all the shortest paths between all pairs of nodes
- Computationally expensive
- How to compute betweenness without listing out all such shortest paths?
- Method based on BFS

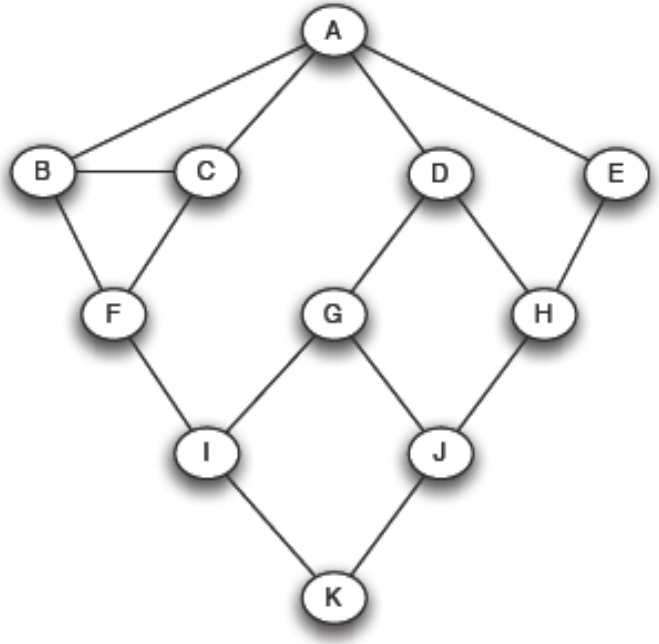
Method

- For each node A:
  1. BFS starting at A
  2. Count the number of shortest paths from A to each other node
  3. Based on this number, determine the amount of flow from A to all other nodes

Step 1: Example



(a) *A sample network*



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

Layer 1

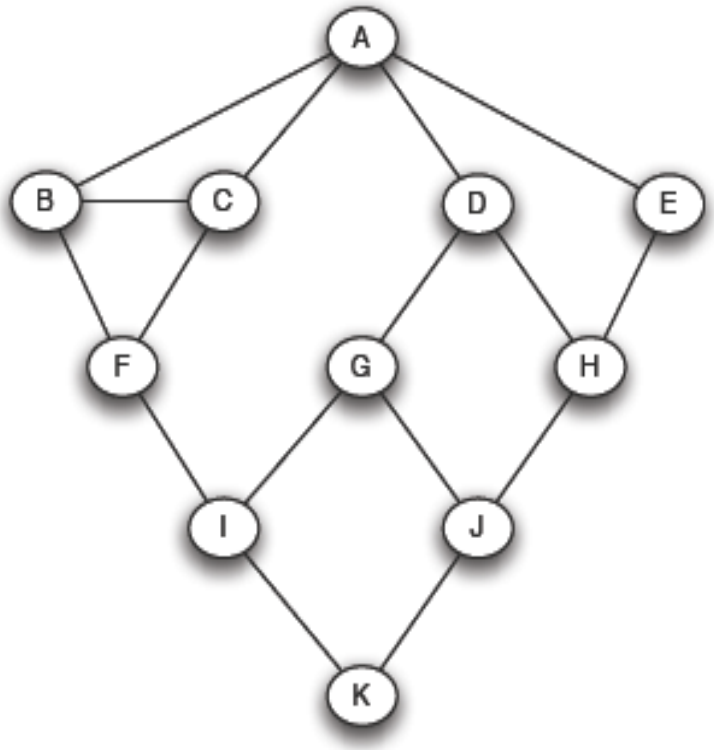
Layer 2

Layer 3

Layer 4

Step 2: Example

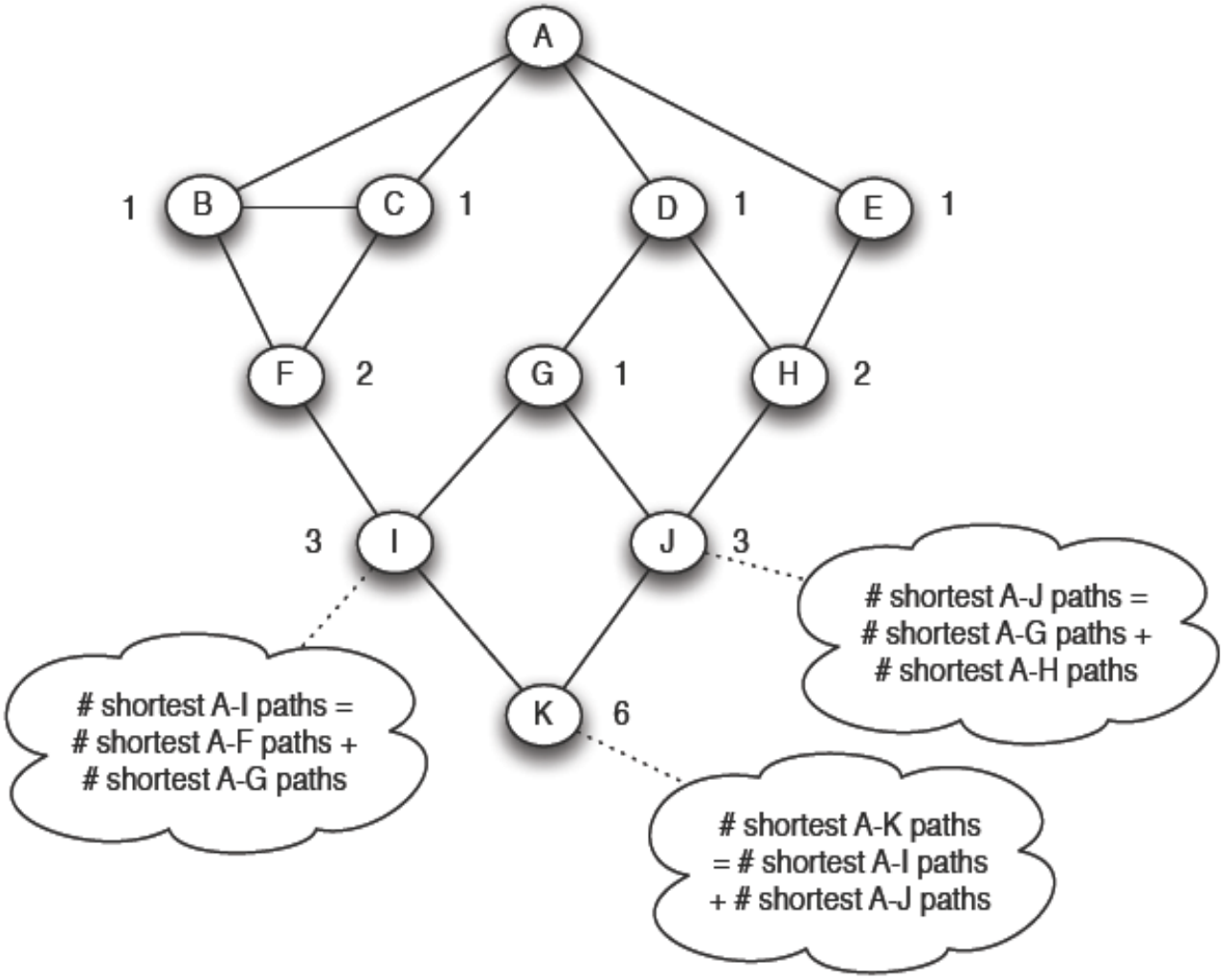
- F and G are above I
- All shortest-paths from A to I must take their last step through either F or G
- To be a shortest path to I, a path must first be a shortest path to one of F or G, and then take this last step to I
- The number of shortest paths from A to I is the number of shortest paths from A to F, plus
- the number of shortest paths from A to G



a node *X is above a node Y in the breadth-first search if X is in the layer immediately preceding Y , and X has an edge to Y*

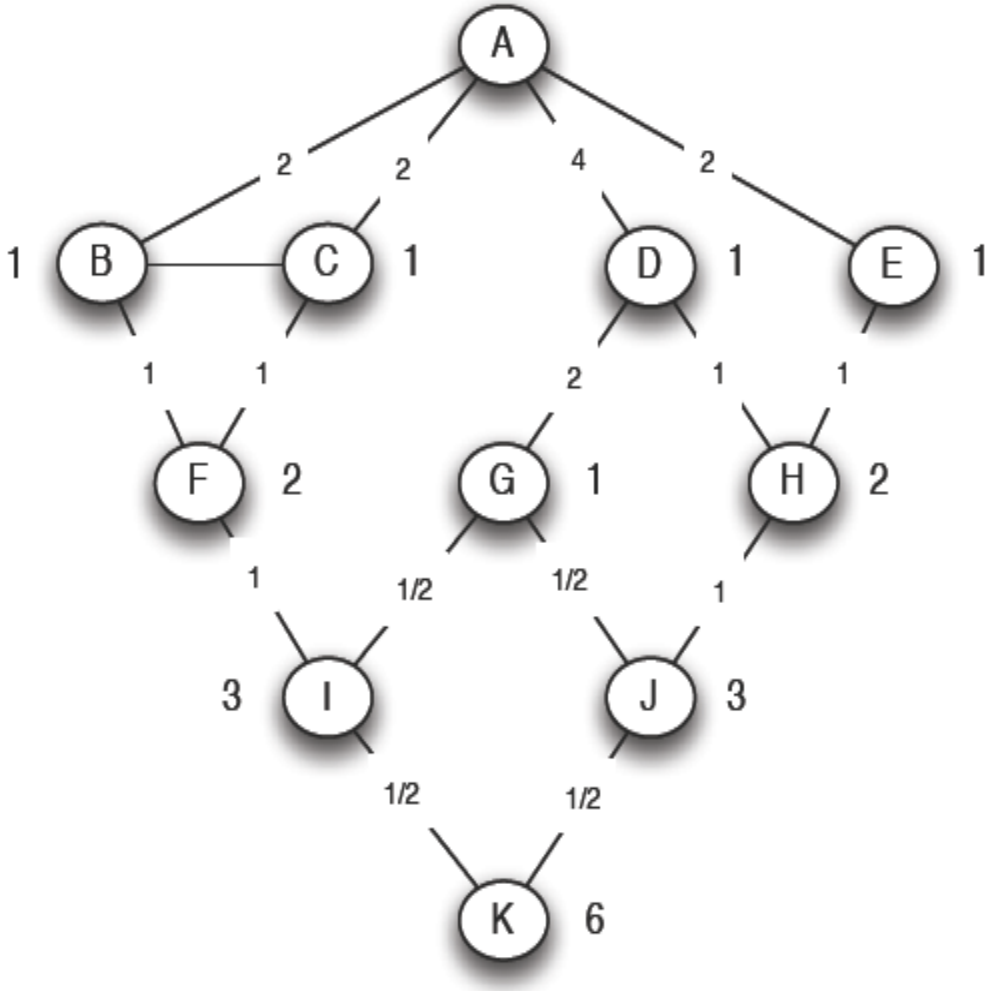
Step 2: Example

- Each node in the first layer has only 1 shortest path from A
- The number of shortest paths to each other node is the sum of the number of shortest paths to all nodes directly above it
- Avoid finding the shortest paths themselves!



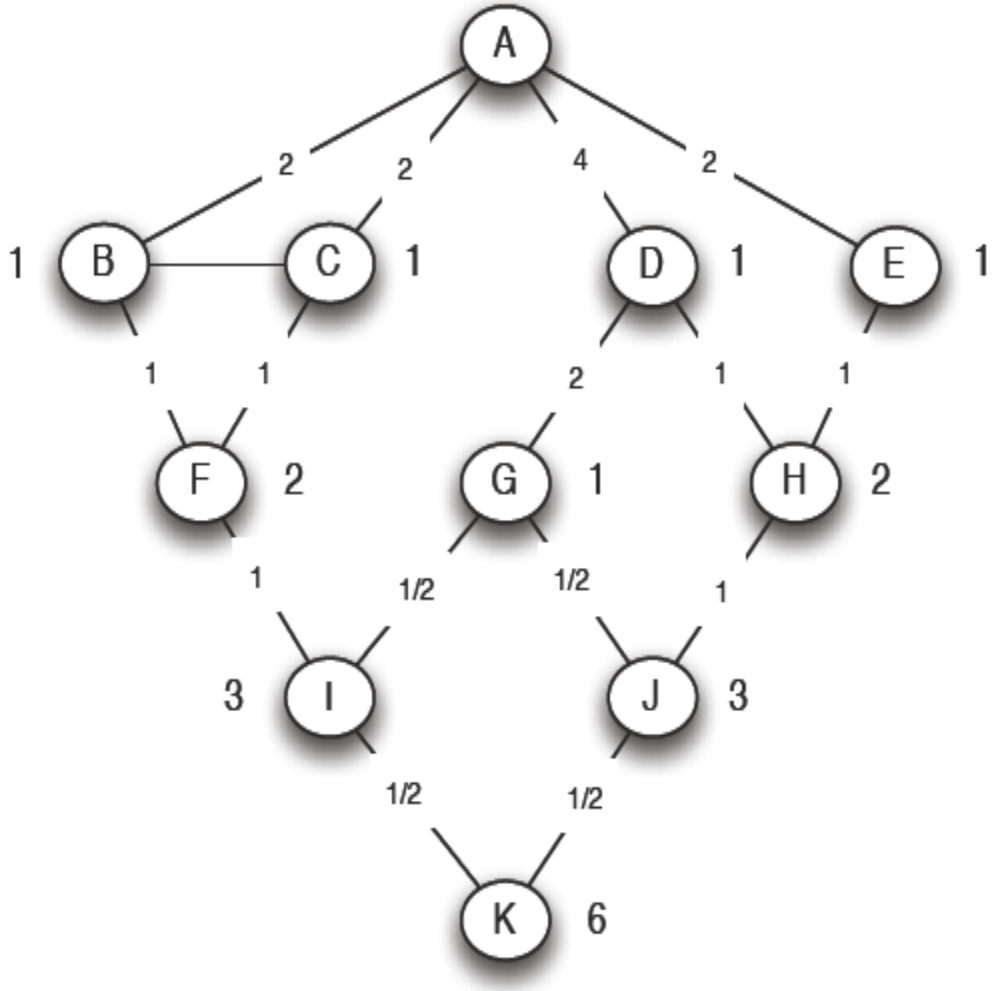
Step 3: Example

- How the flow from A to all other nodes spreads out across the edges?
- Working up from the lowest layers
  - **1** unit of flow arrives at K and an equal number of the shortest paths from A to K come through nodes I and J => **1/2**-unit of flow on each of these edges
  - **3/2** units of flow arriving at I (**1** unit destined for I plus the **1/2** passing through to K). These **3/2** units are divided in proportion **2** to **1** between F and G => **1** unit to F and **1/2** to G



Step 3: Method

- Move bottom up
- At each node X
  - add up all flow arriving from edges directly below X, plus 1 for the flow destined for X itself
  - Divide this up over the edges leading upward from X, in proportion to the number of shortest paths coming through each



Summary

- Build one BFS structures for each node
- Determine flow values for each edge using the previous procedure and (3 steps)
- Sum up the flow values of each edge in all BFS structures to get its betweenness value
- Notice: we are counting the flow between each pair of nodes X and Y twice (once when BFS from X and once when BFS from Y)
  - at the end we divide everything by two
- Usie these betweenness values to identify the edges of highest betweenness for purposes of removing them in the Girvan-Newman method

Computing Betweennes of Nodes

- Same procedure
- Compute the outgoing (upwords) sum of flow from node
  - or downards sum + 1