

# Mining Social-Network Graphs

## Lecture 8



# Outline

## 1. Social Networks as Graphs

- a) Examples
- b) Representation
- c) Properties

## 2. Clustering

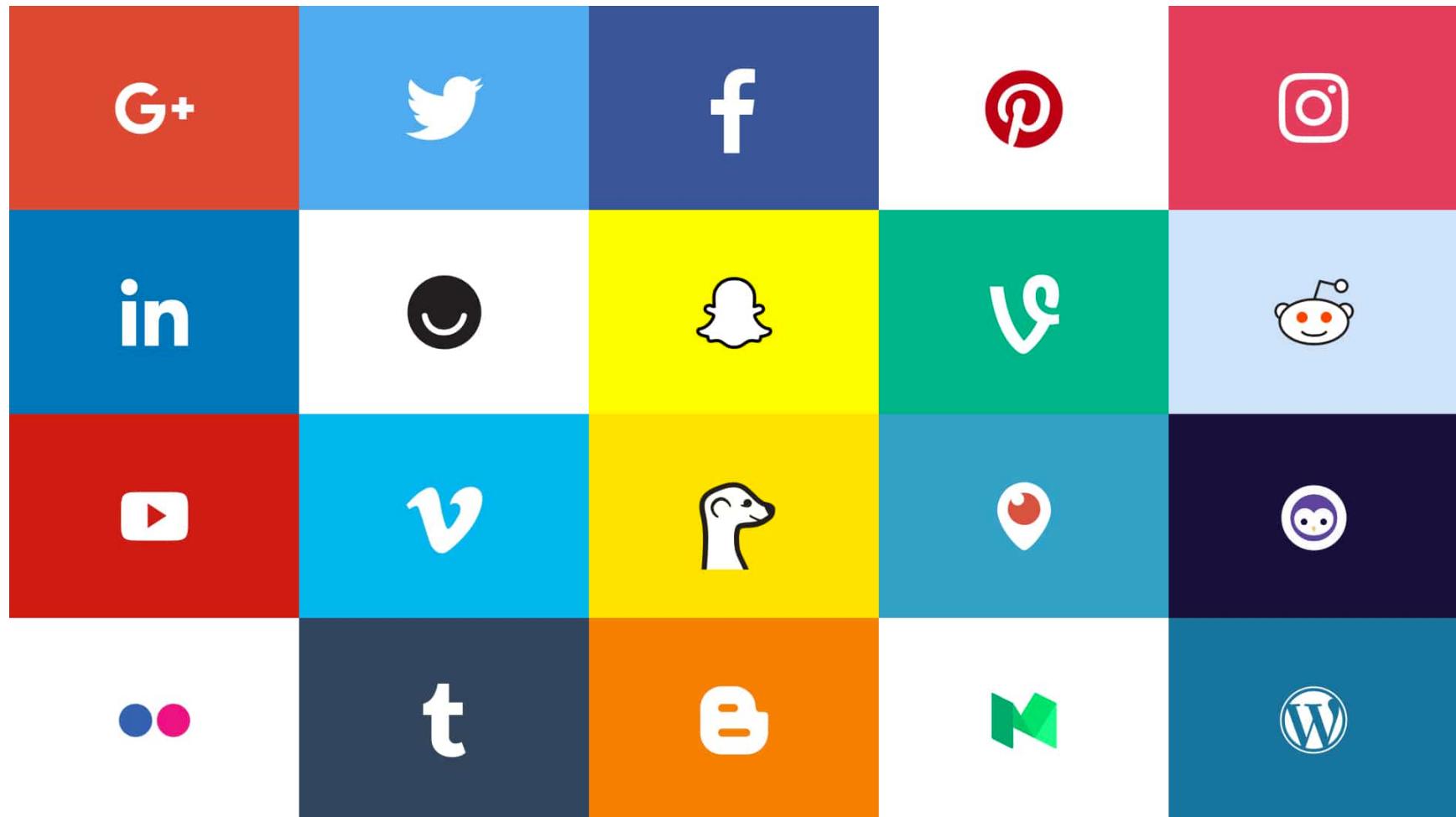
- a) Distance Measures
- b) Girvan-Newman

## 3. Spectral Analysis

- a) Networks as Matrices
- b) Connectivity
- c) Partitioning
- d) Clustering



# Social Networks (obvious)

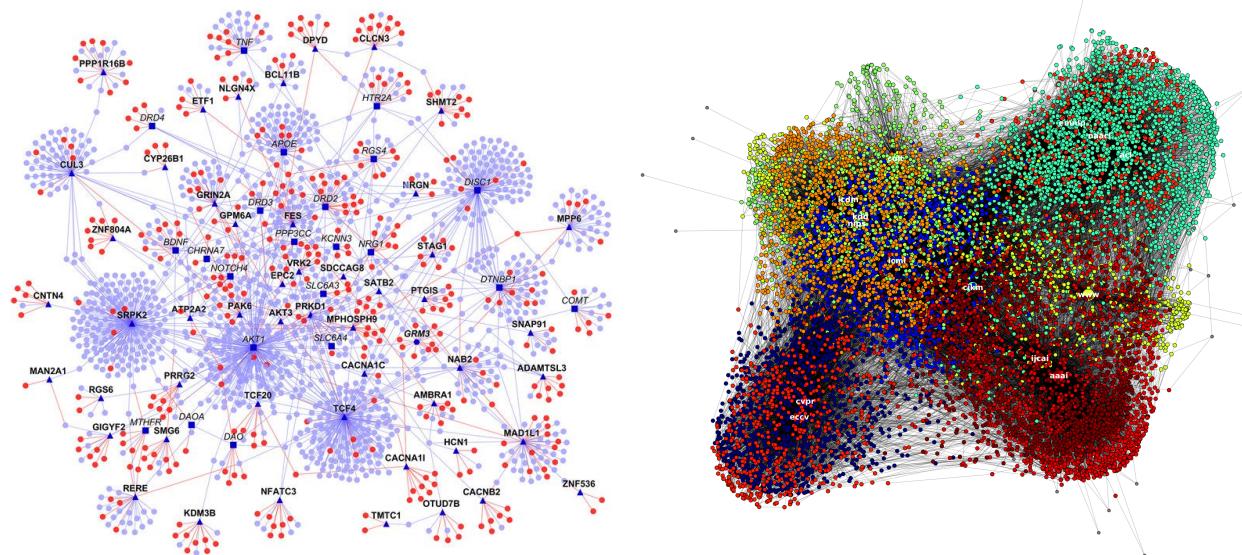


# Social Networks (obvious)



# Less Obvious

- Phone/E-mail (within K units of time)
- Collaboration (academic papers, patents)
- Biological (proteins, genes)



# Representation

- Nodes = entities
  - People, papers, ...
  - **Single** vs. multiple
    - del.icio.us: people, websites, tags
- Edges = relationship
  - **Discrete** vs continuous (i.e. weighted)
  - Directed (e.g. following) vs **undirected**



# Checkup

- Given a graph of 7 nodes (A-G), how many undirected edges are possible?



# Answer

- Given a graph of 7 nodes (A-G), how many undirected edges are possible?

$$\binom{7}{2} = \frac{7!}{2!(7-2)!} = \frac{7(7-1)}{2} = 21$$

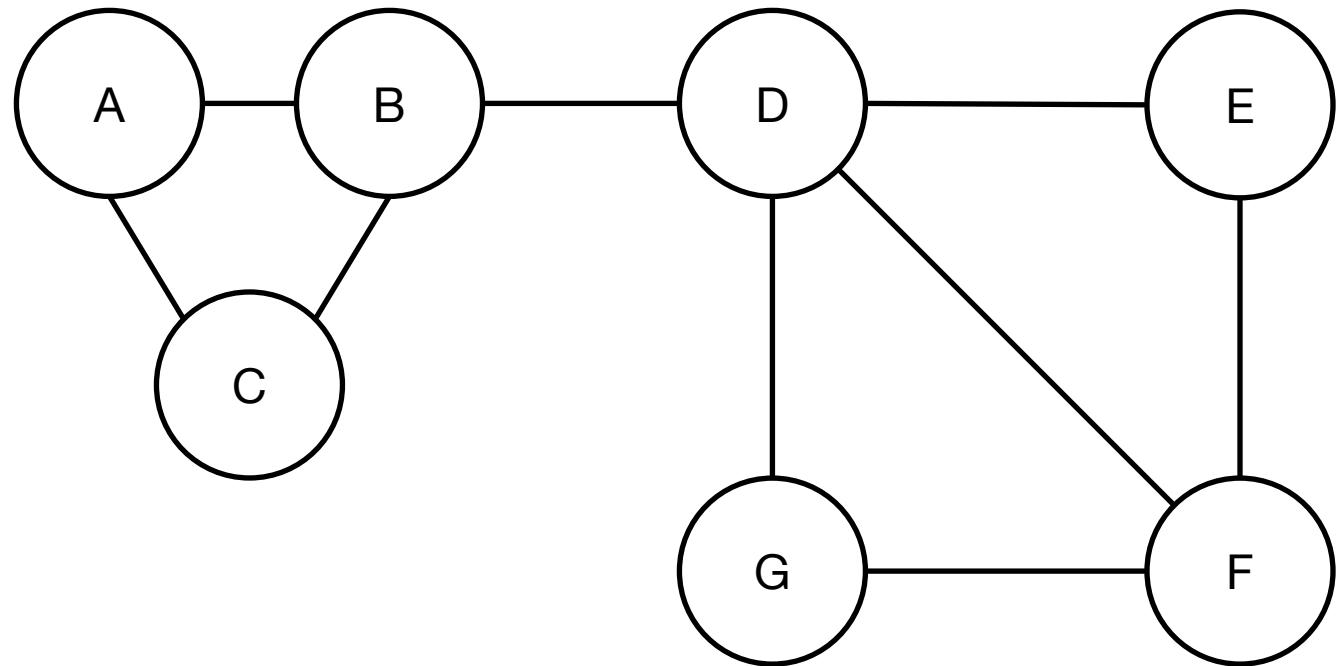


# Locality

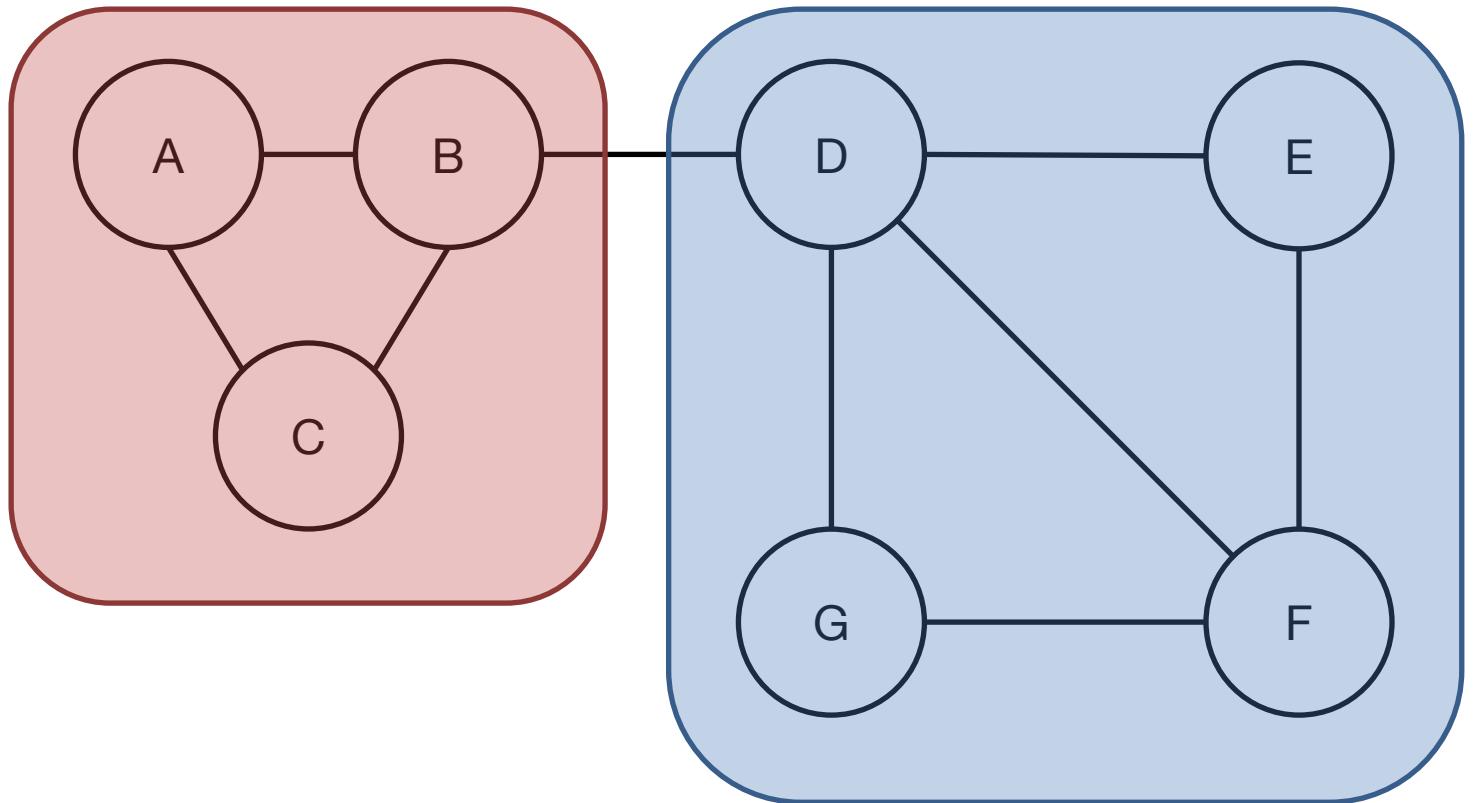
- Typically assumed that if node A is connected to both B and C, then it is more likely than random that B and C are connected
- Our focus: **Community Detection**
  - Finding groups of densely connected nodes
  - (Tightly related to the locality assumption – i.e. would not work well on random graphs)



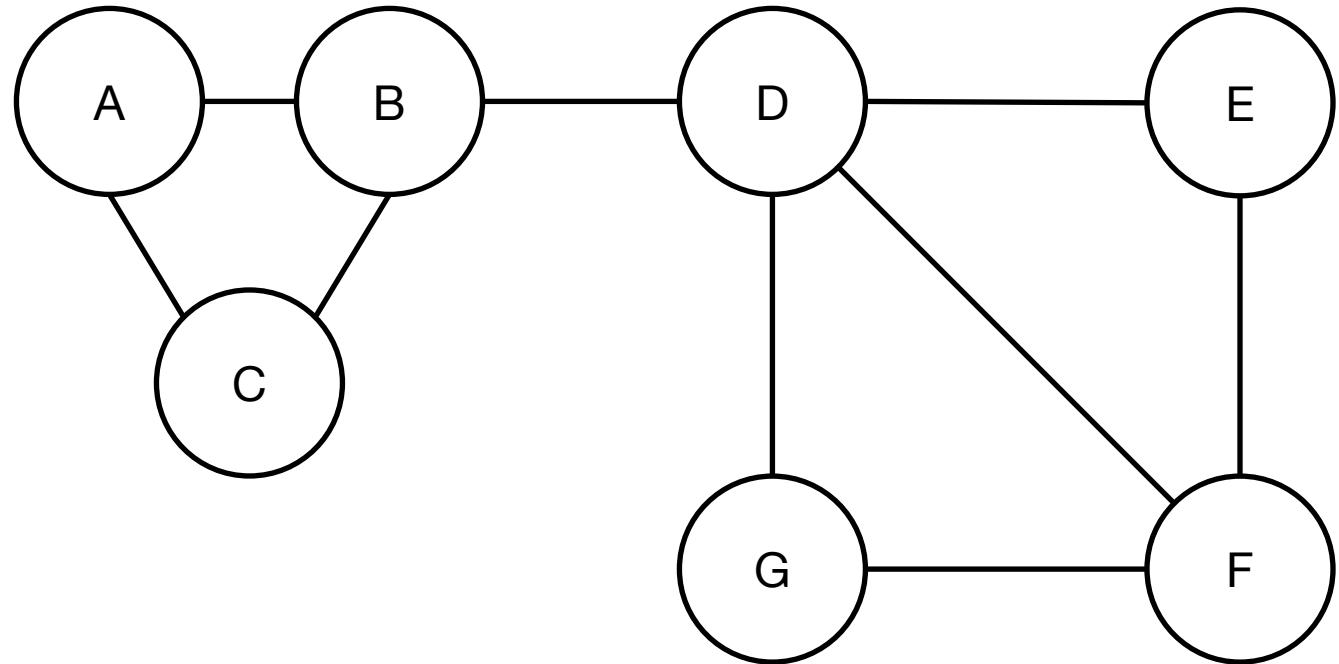
# Example Graph



# Communities to Be Detected



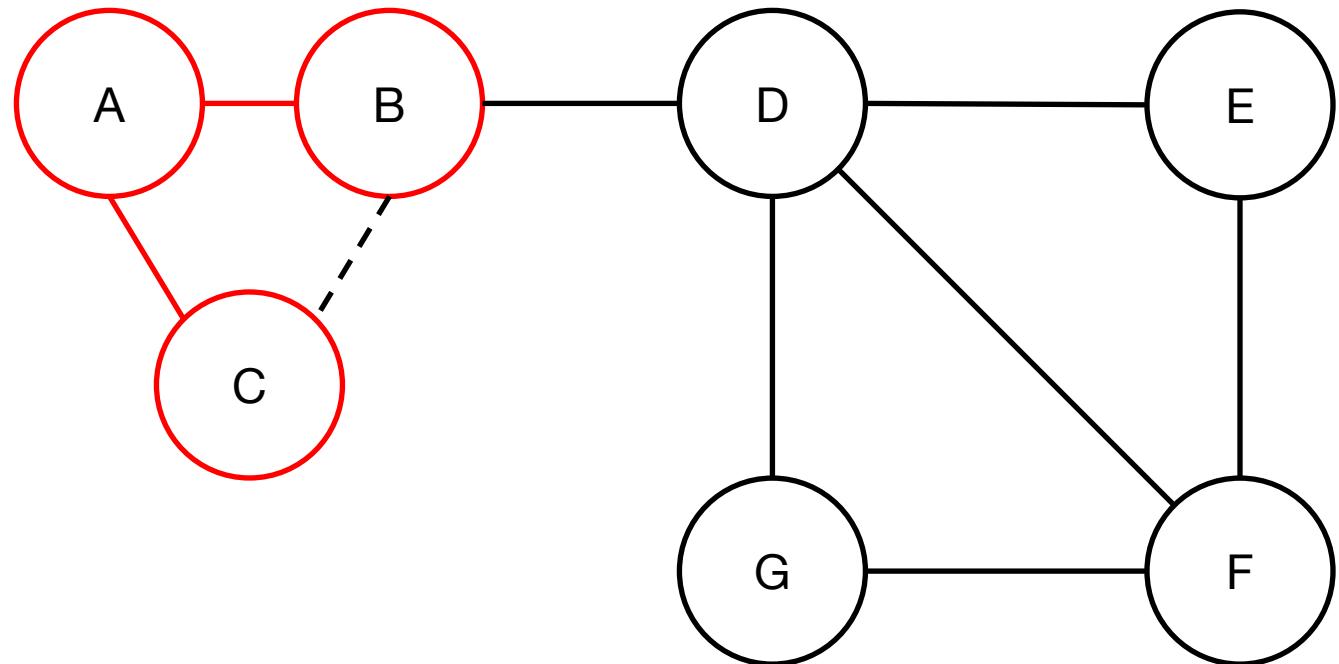
# Nodes, Edges, Average Connectivity?



- Nodes: 7
- Edges: 9
- Avg Connectivity:  $9/21 \sim 0.43$



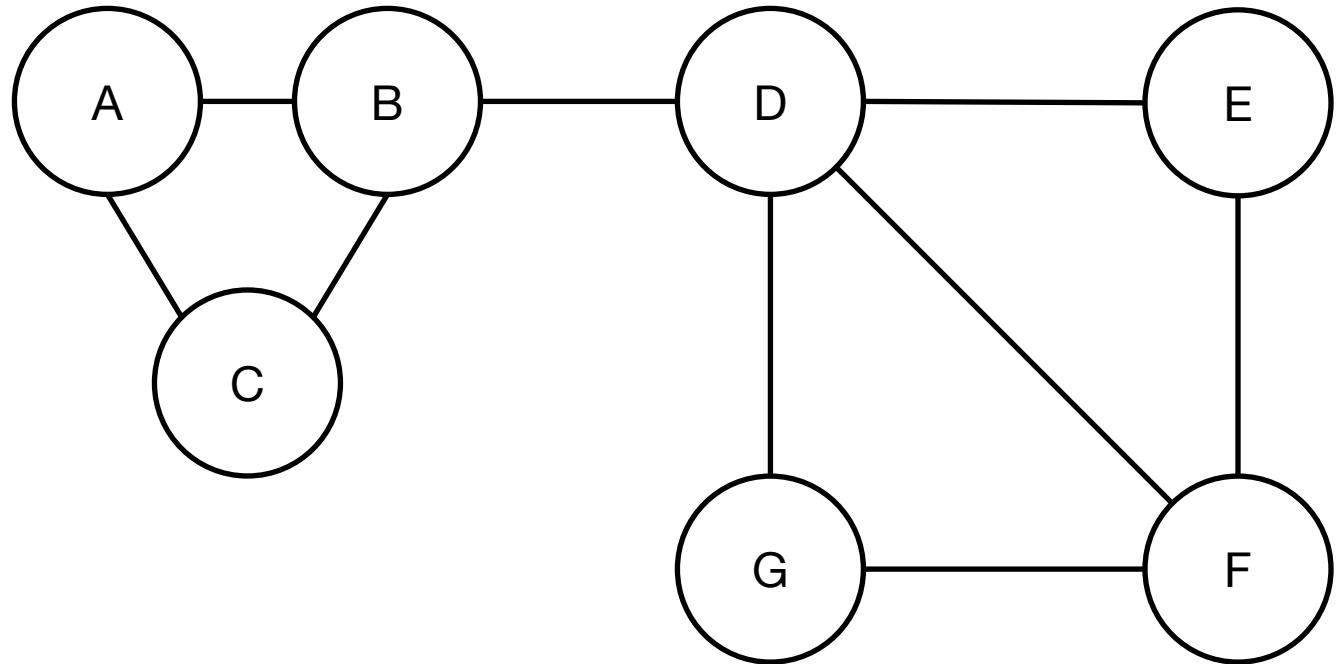
# Given A-B, A-C; Expected B-C?



- Nodes: 7
- Edges: 9
- Avg Connectivity:  $9/21 \sim 0.43$
- Expected Locality:  $7/19 \sim 0.37$



# Actual?

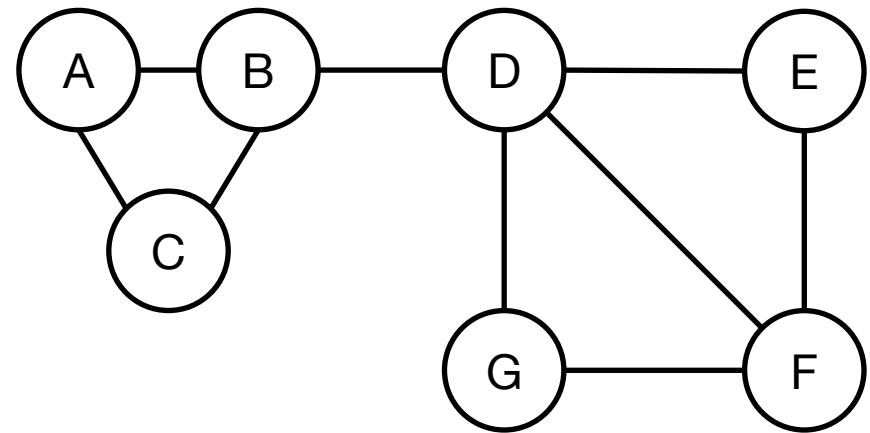


- Nodes: 7
- Edges: 9
- Avg Connectivity:  $9/21 \sim 0.43$
- Expected Locality:  $7/19 \sim 0.37$
- *What are all the triples?*



# Evaluate Node Triples

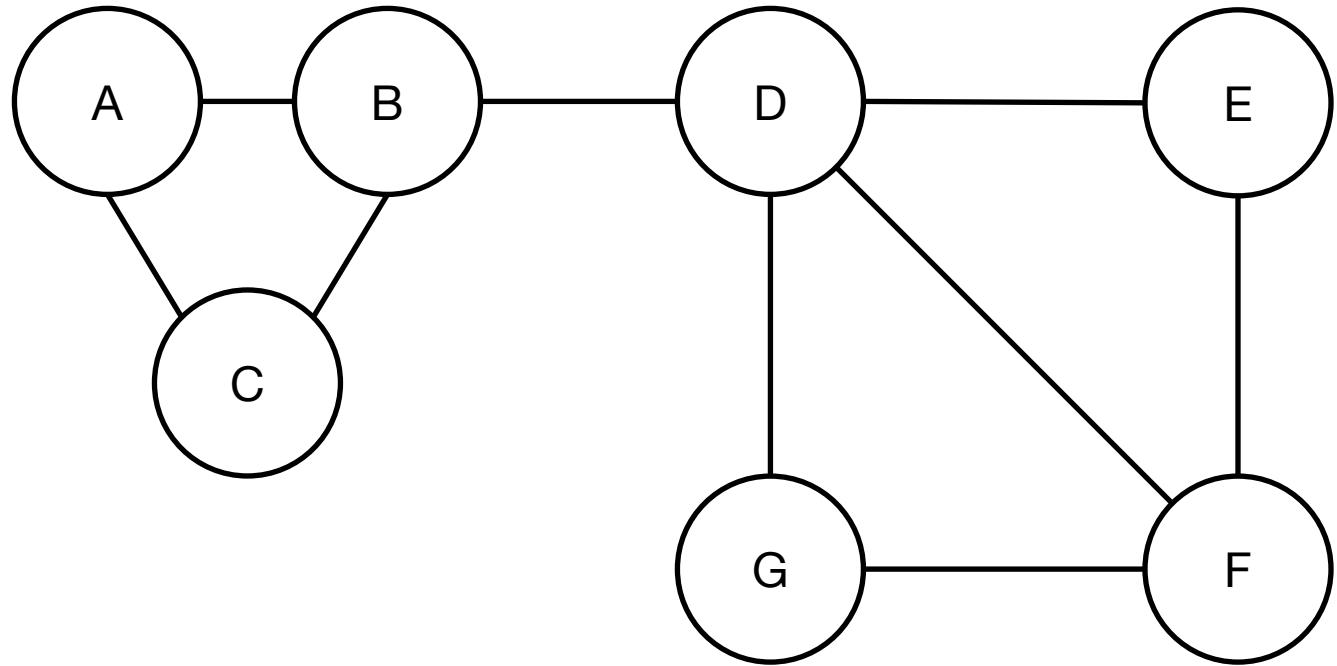
- **A**
  - BC ✓
- **B**
  - AC ✓, AD ✗
  - CD ✗
- **C**
  - AB ✓
- **D**
  - BE ✗, BF ✗, BG ✗
  - EF ✓, EG ✗
  - FG ✓
- **E**
  - DF ✓



- **F**
  - DE ✓, DG ✓
  - EG ✗
- **G**
  - DF ✓
- ✓ = 9, ✗ = 7
- ✓ / (✓ + ✗) = 9/16 ~ 0.56



# Local!



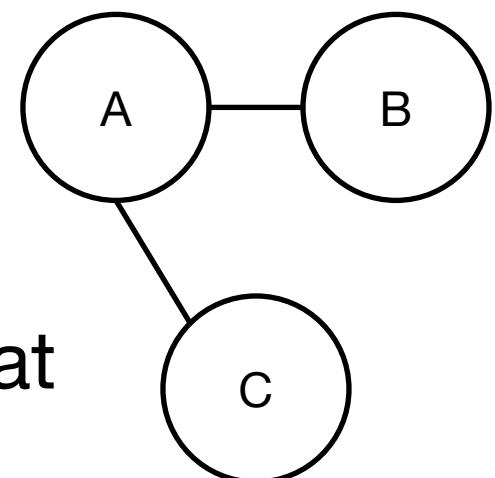
- Nodes: 7
- Edges: 9
- Avg Connectivity:  $9/21 \sim 0.43$
- Expected Locality:  $7/19 \sim 0.37$
- Actual Locality:  $9/16 \sim 0.56 (>> \text{Expected!})$

Let's Cluster!



# Distance Measure?

- Distance measures on social-network graphs can be tricky
- KISS ex:  $D(x,y)=0$  if edge, 1 otherwise
  - $D(A, B)=0$
  - $D(A, C)=0$
  - $D(B, C) = 1 > D(A, B) + D(A, C)$
- Also leads to unfortunate ties that lead to unfortunate results in typical clustering algorithms

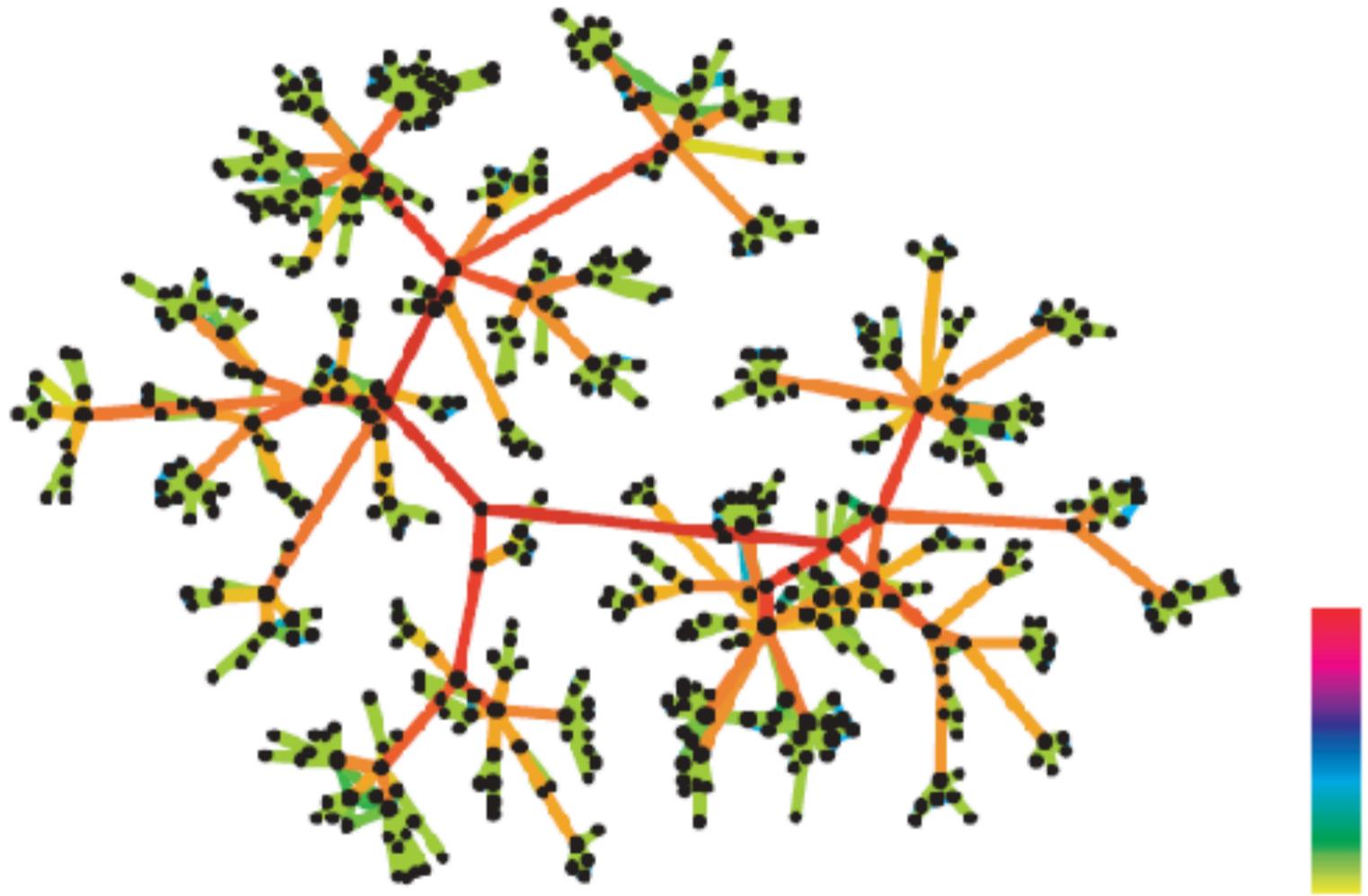


# Betweenness

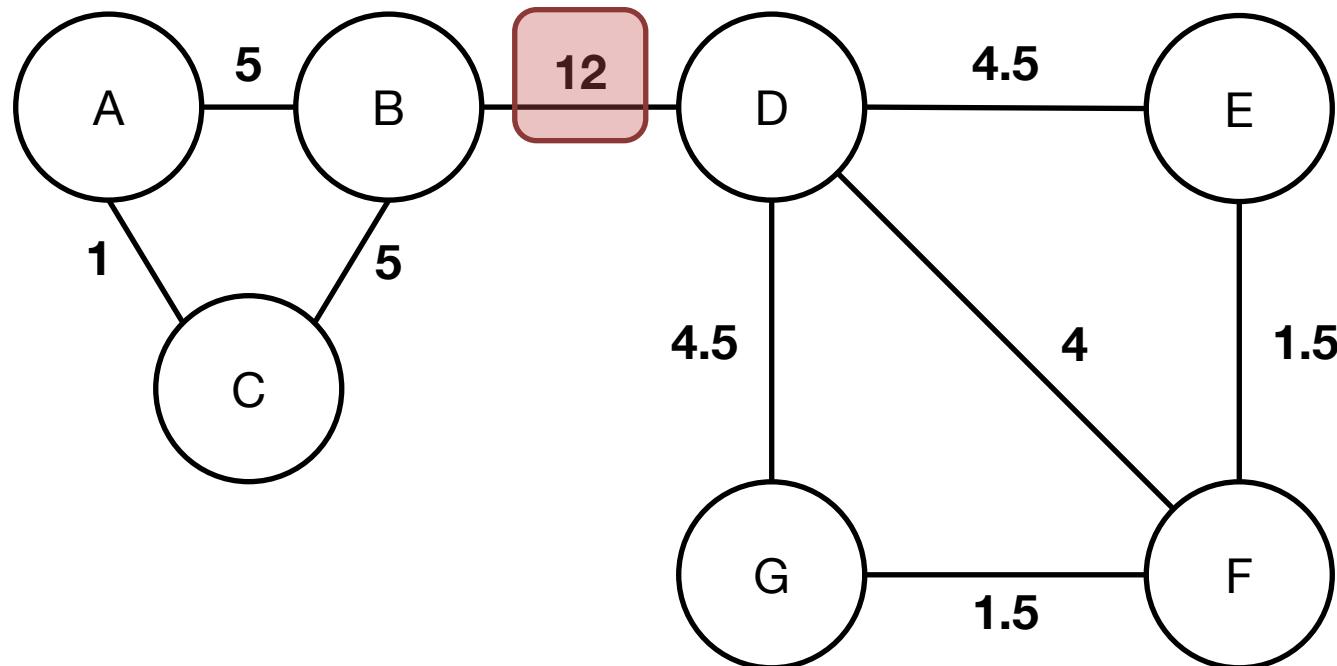
- One of the simplest measures, based on finding the edges that are the *least likely* to be inside a community
  - Clustering: remove high betweenness first!
- $B(a, b) =$  for all pairs of nodes  $(x, y)$ , fraction of shortest paths that include edge  $ab$



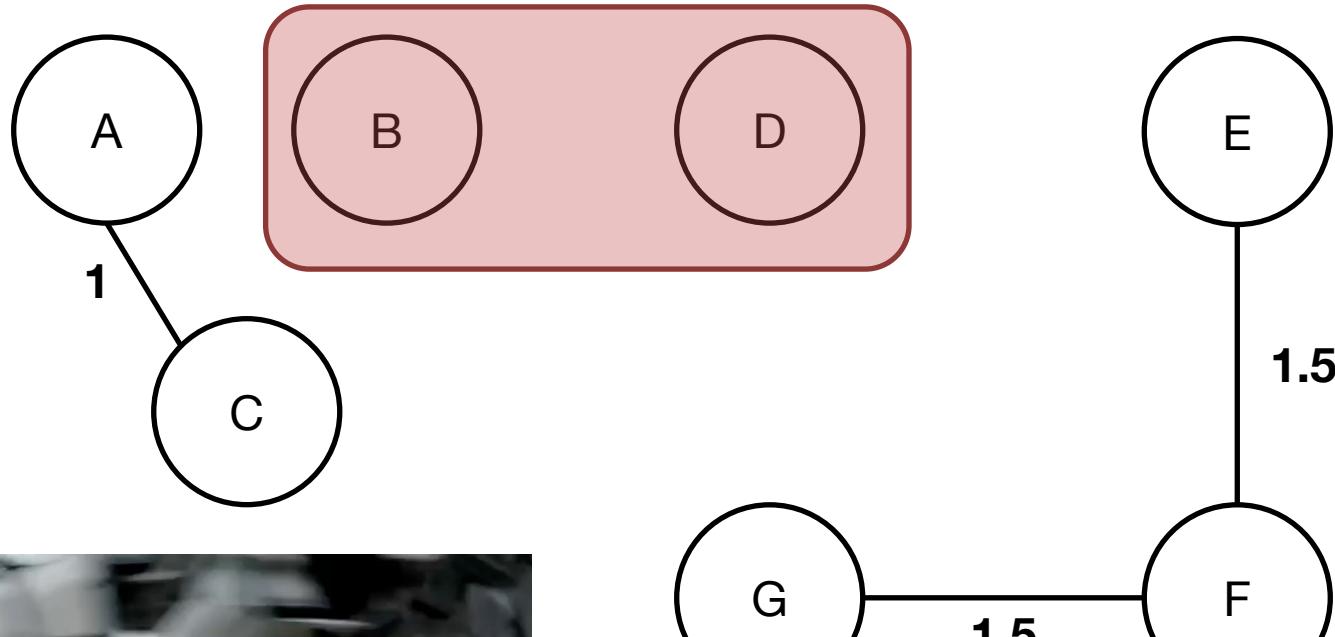
# Betweenness Intuition (1)



# Betweenness Intuition (2)



# Betweenness Intuition (3)



# Computing Graph Betweenness

1. For each node

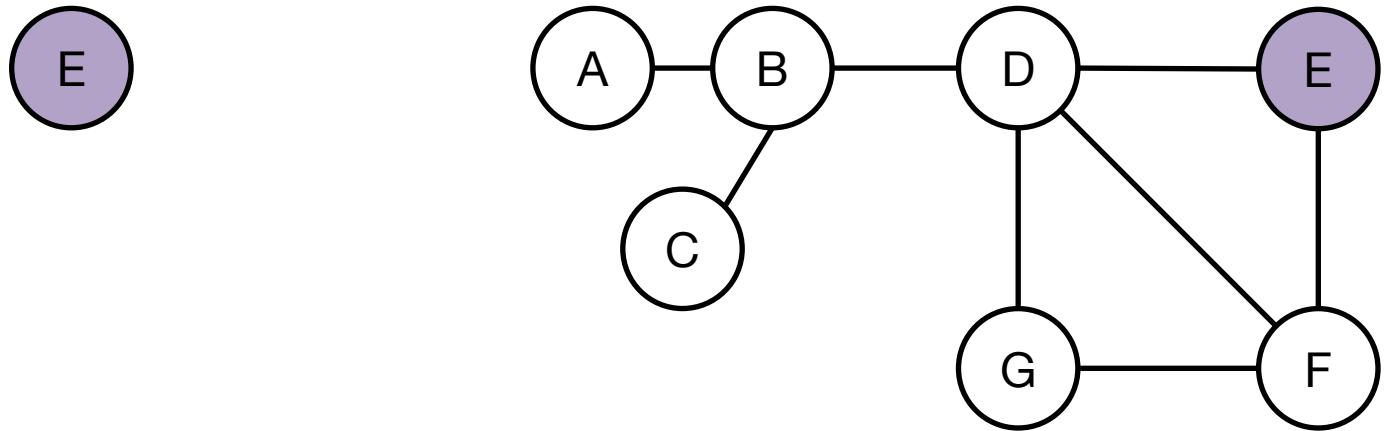
- a) Breadth-First Search (BFS)
  - WHY?
- b) For each edge, compute betweenness

2. For each edge

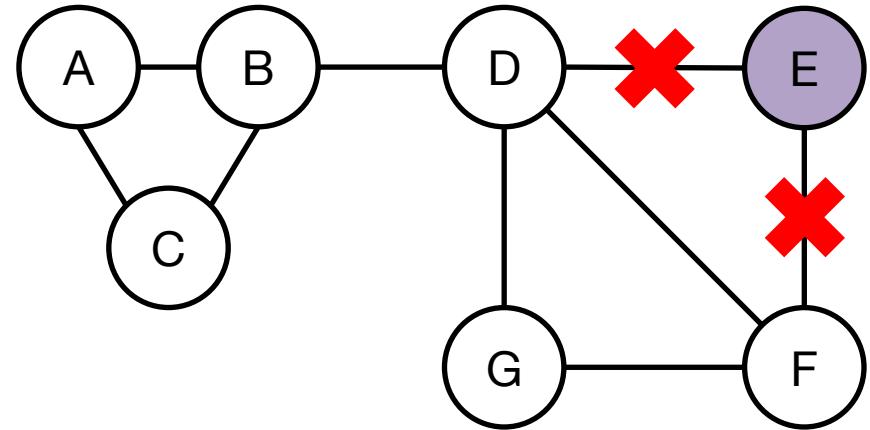
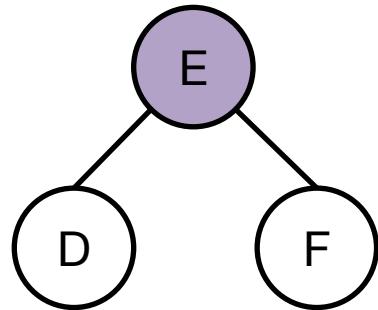
- a) Sum contributions from trees above
- b) Divide by 2 (prevents duplication)



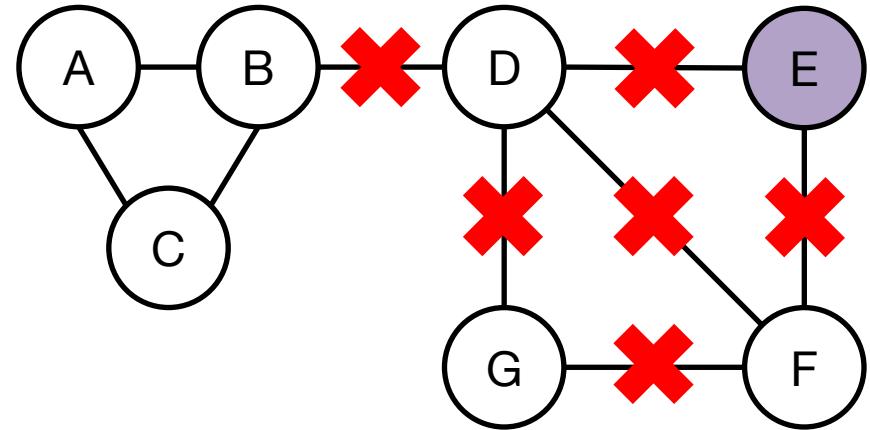
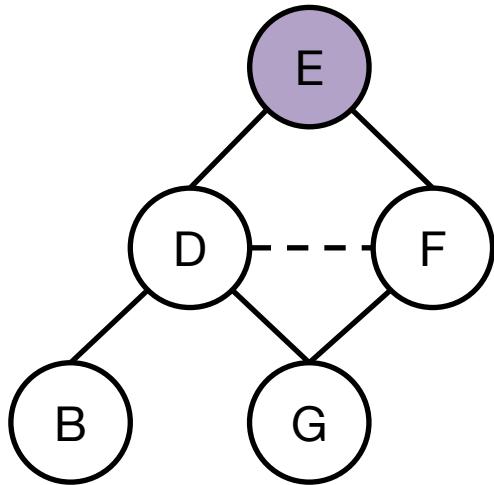
# Example (E, BFS.0)



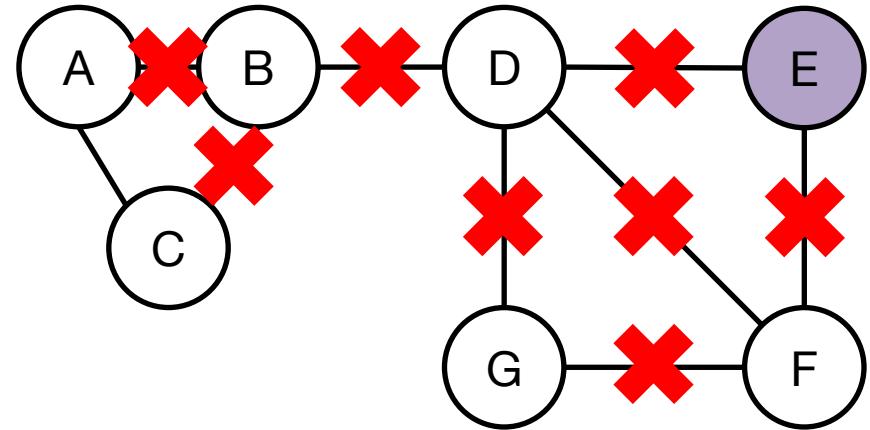
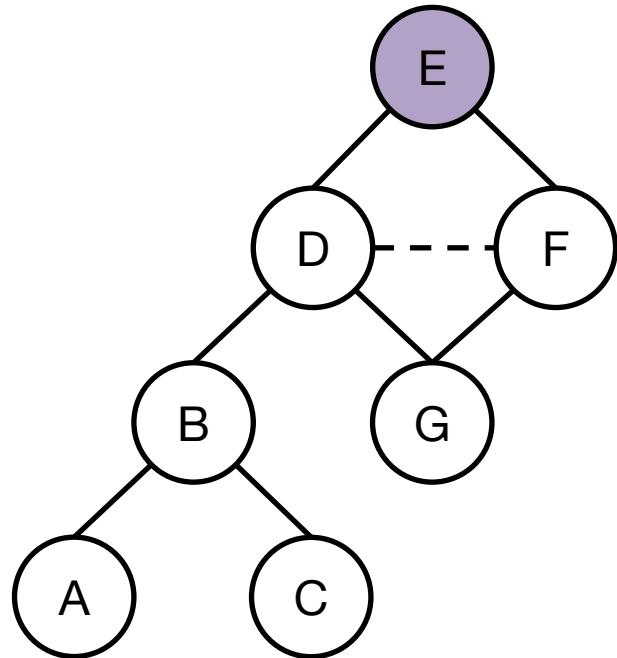
# Example (E, BFS.1)



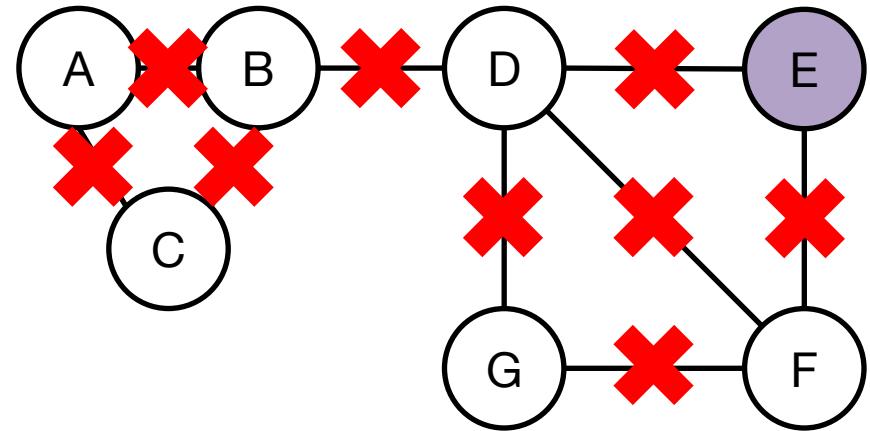
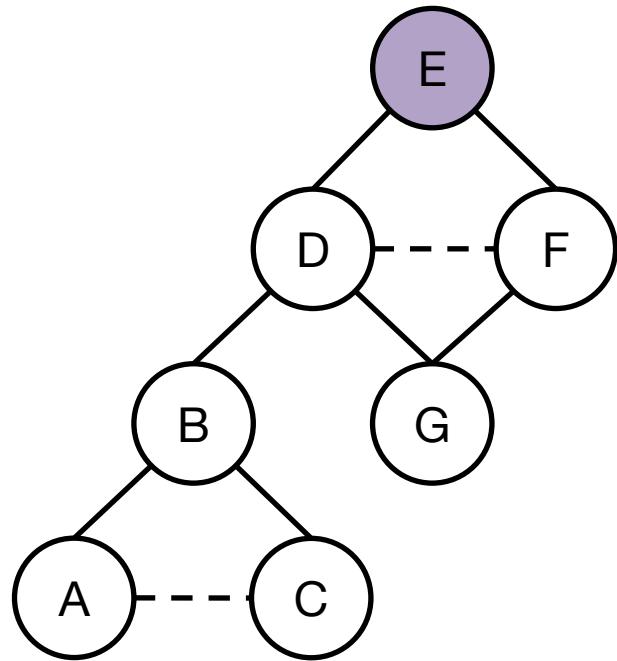
# Example (E, BFS.2)



# Example (E, BFS.3)



# Example (E, BFS.Done)

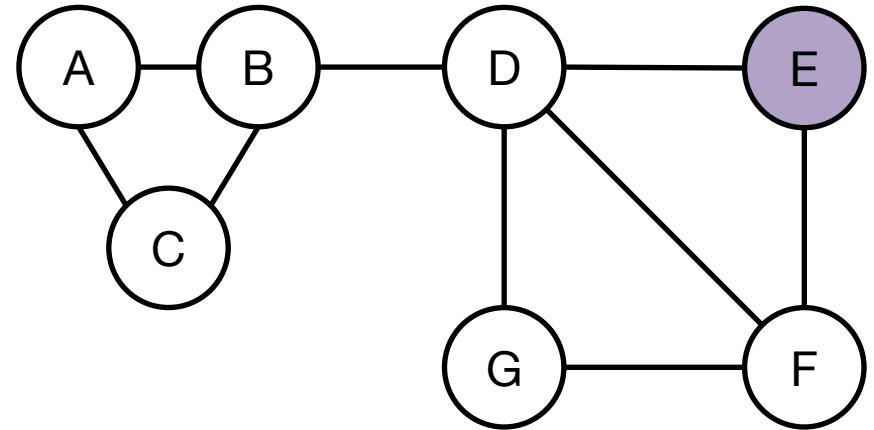
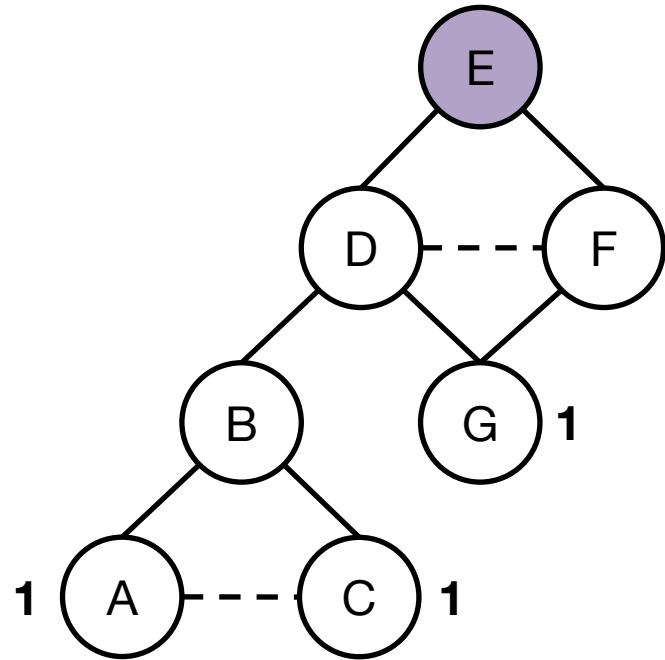


# Computing BFS Betweenness

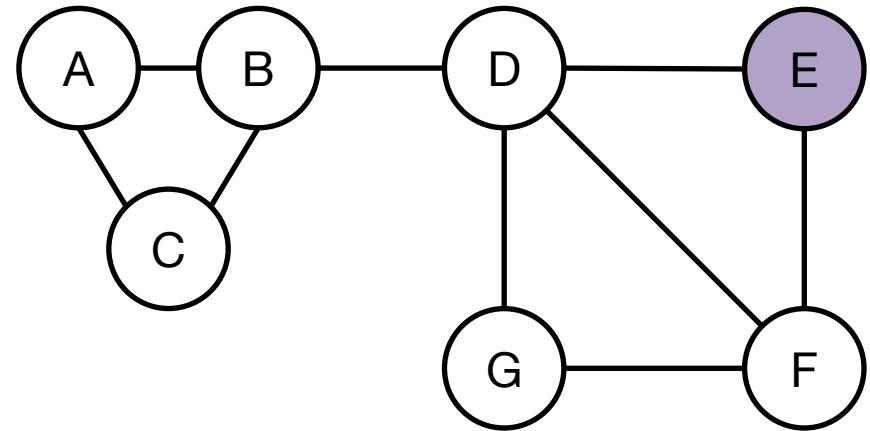
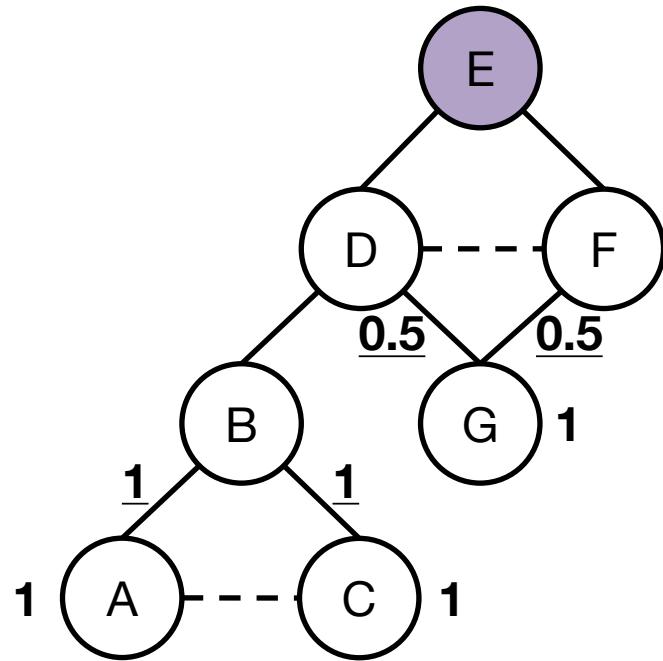
- Recursive credit definition
  - Node gets  $1 + \text{sum of edge credits below}$
  - Edge gets node below divided by outgoing
    - Ignore between nodes at the same level  
(never used for shortest paths)
- Start from leaves, move up



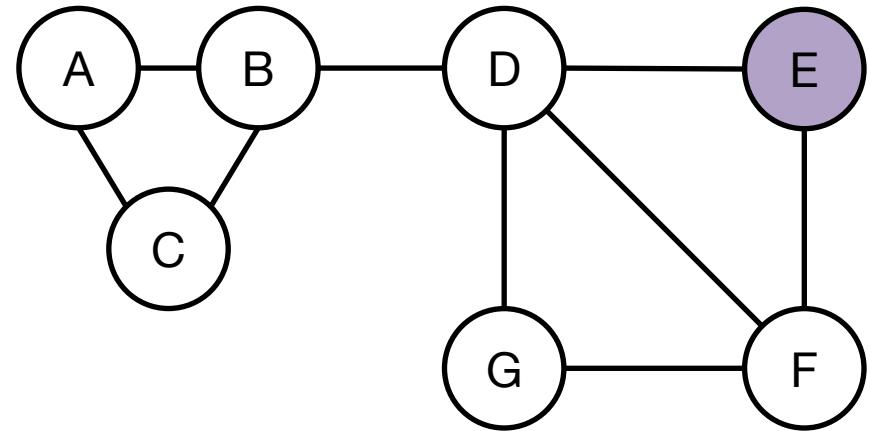
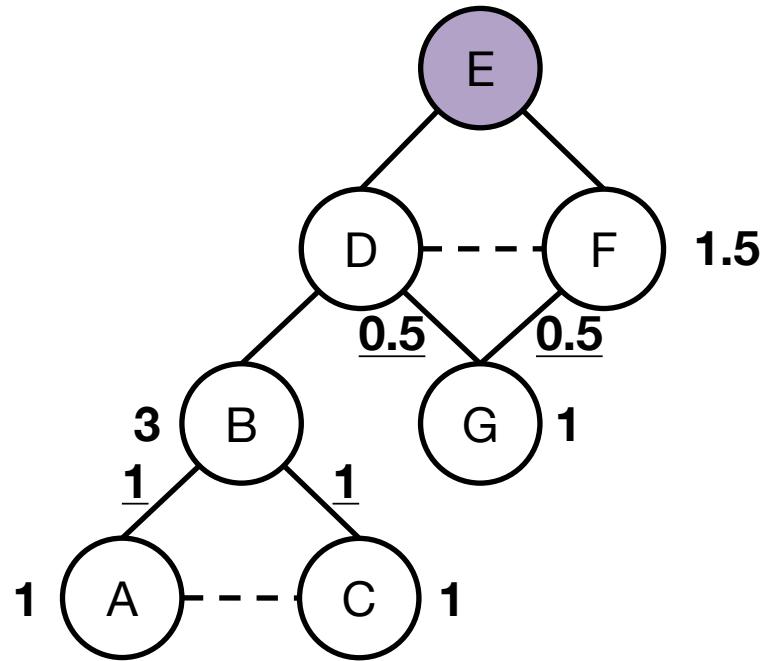
# Example (E, Credits.1)



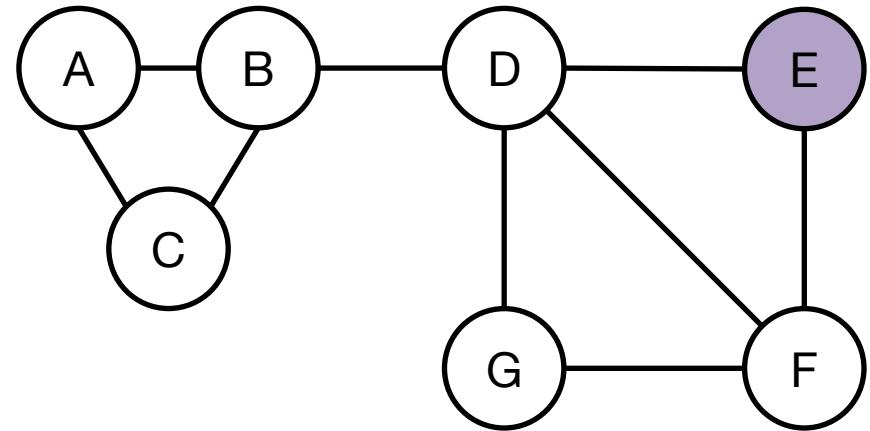
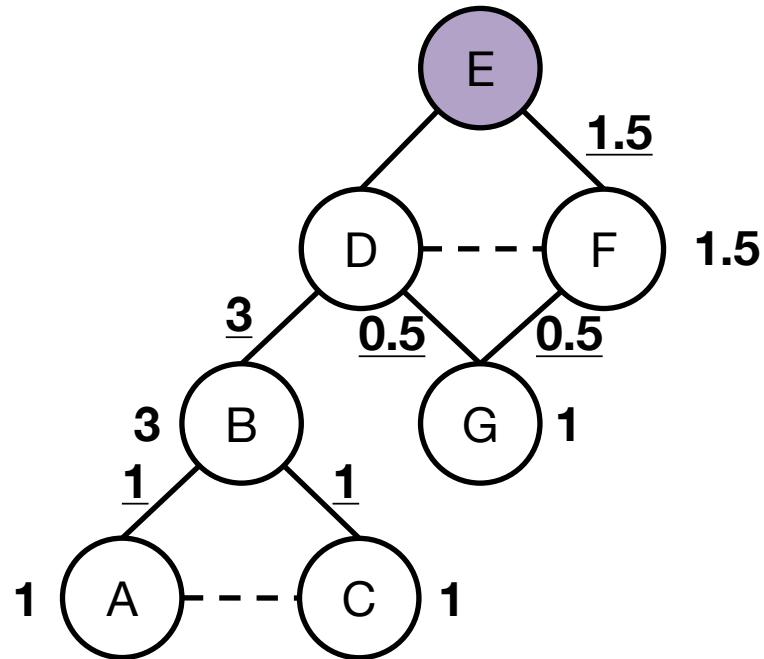
# Example (E, Credits.2)



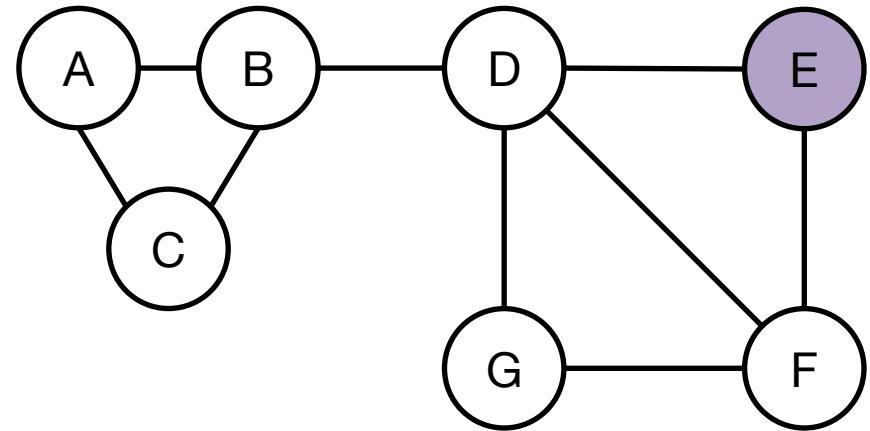
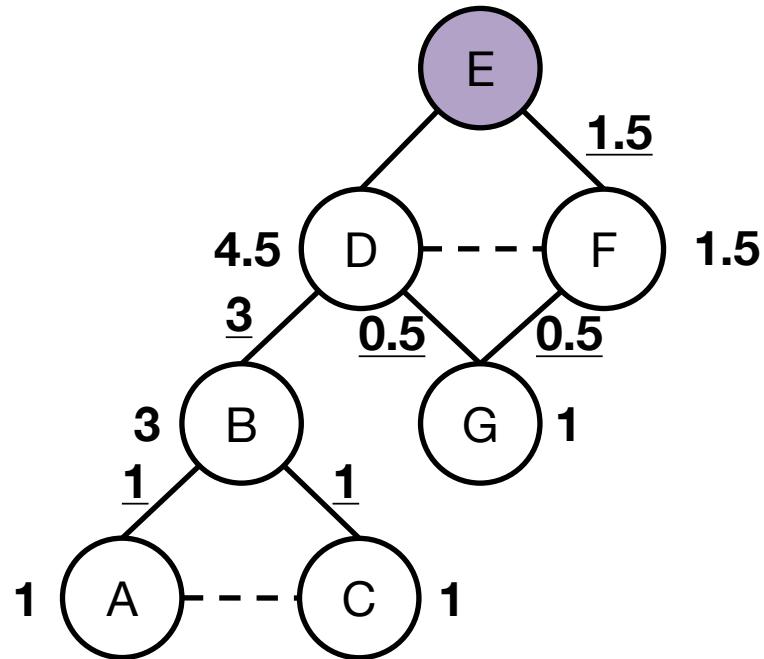
# Example (E, Credits.3)



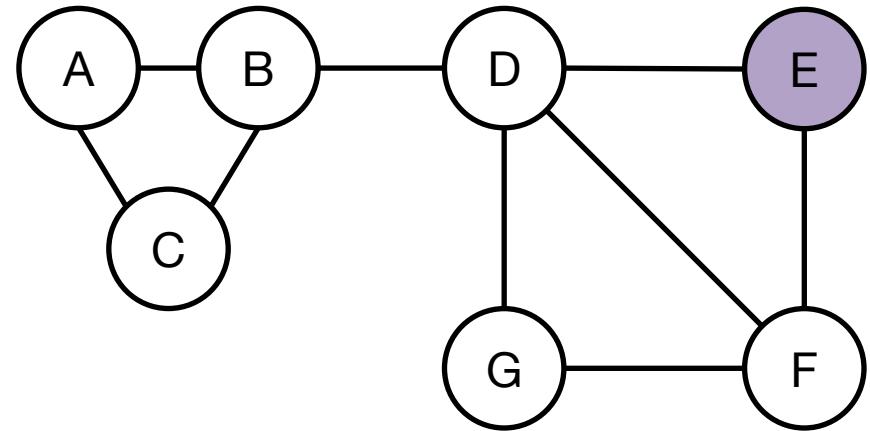
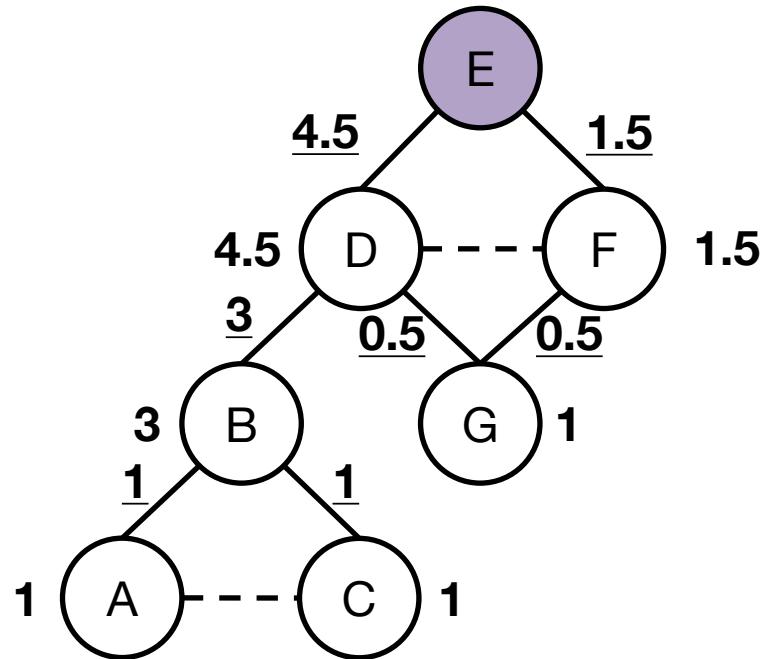
# Example (E, Credits.4)



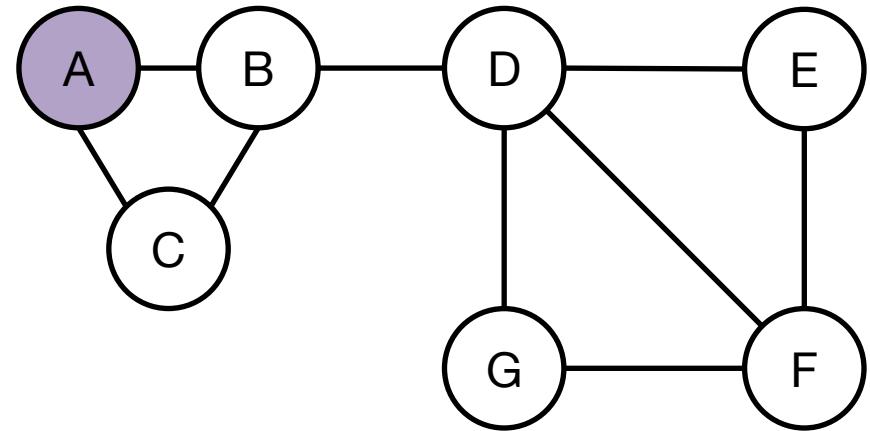
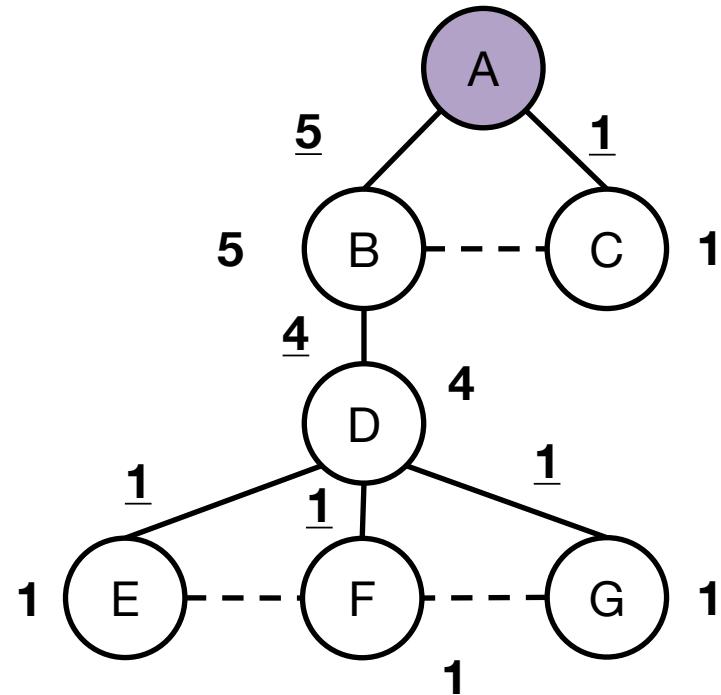
# Example (E, Credits.5)



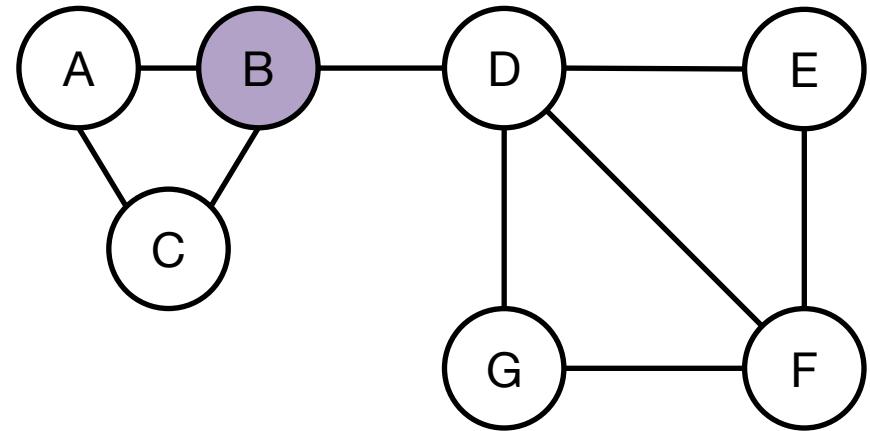
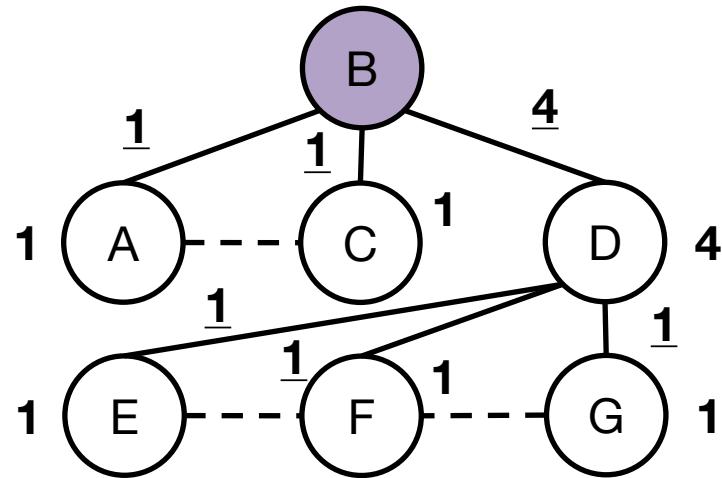
# Example (E, Credits.6)



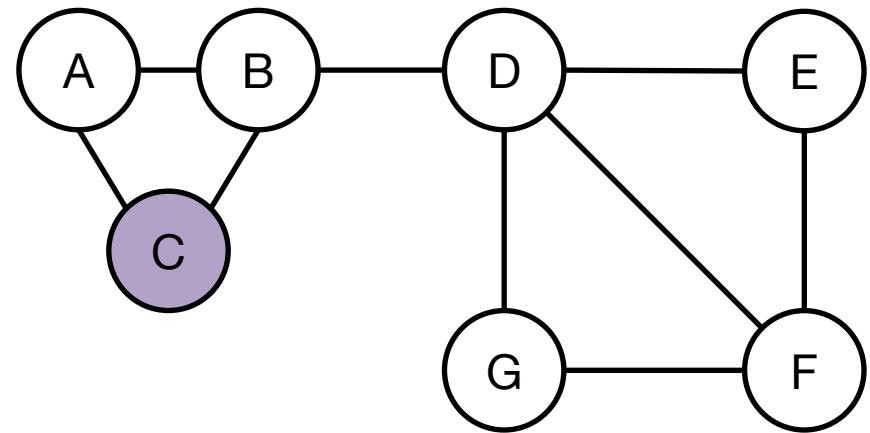
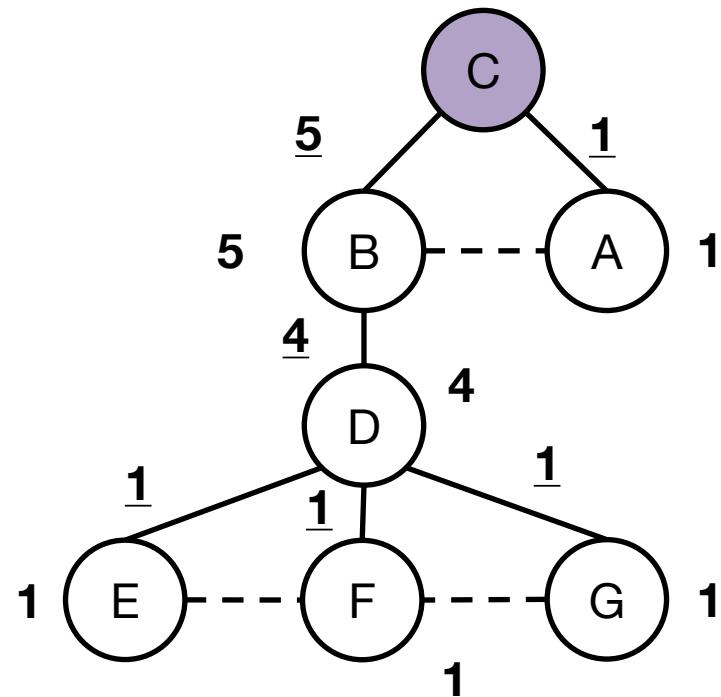
# Example (A, Credits)



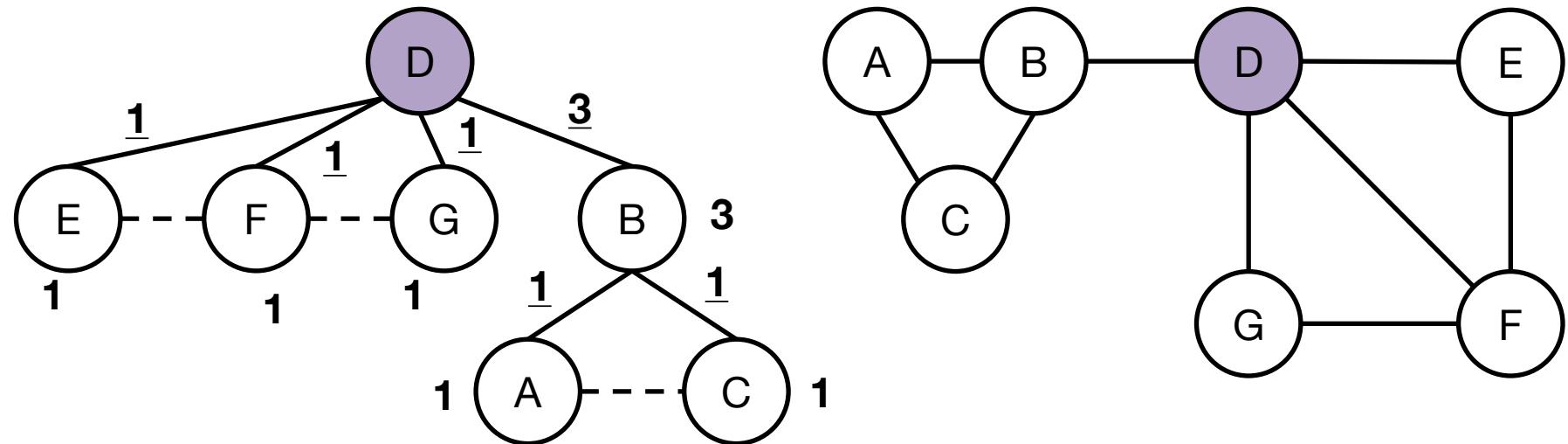
# Example (B, Credits)



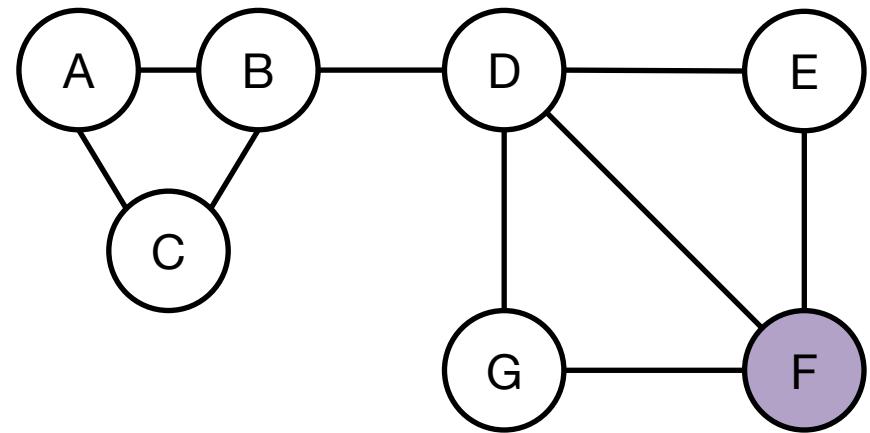
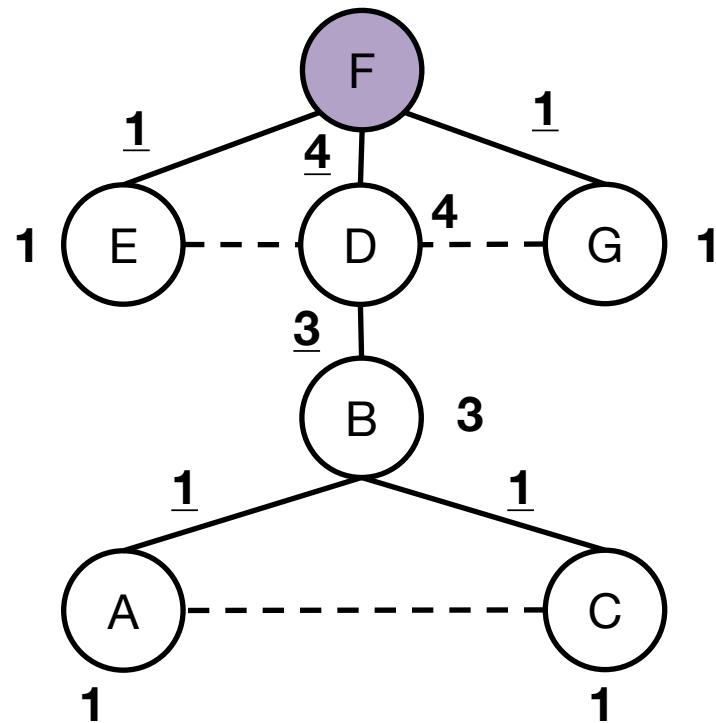
# Example (C, Credits)



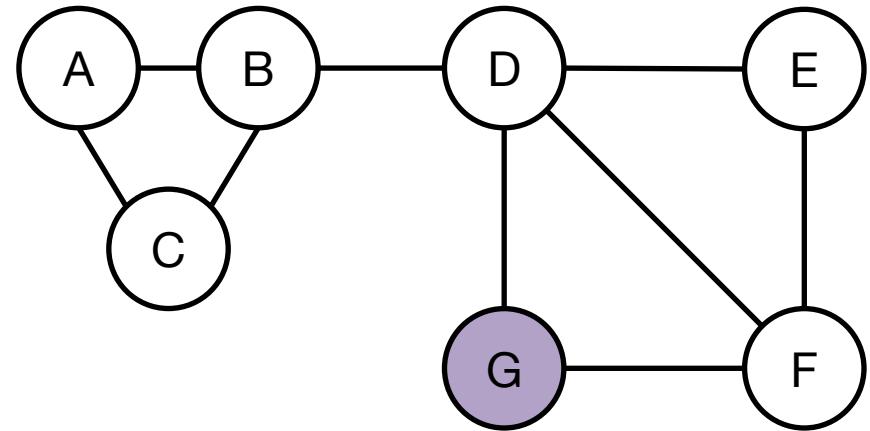
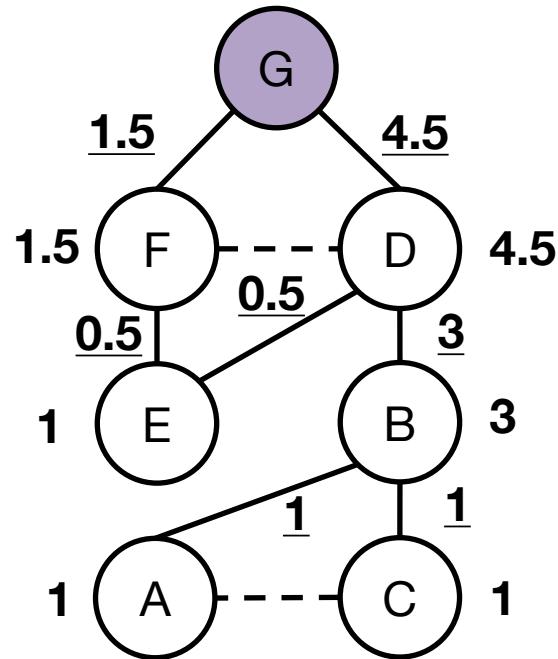
# Example (D, Credits)



# Example (F, Credits)

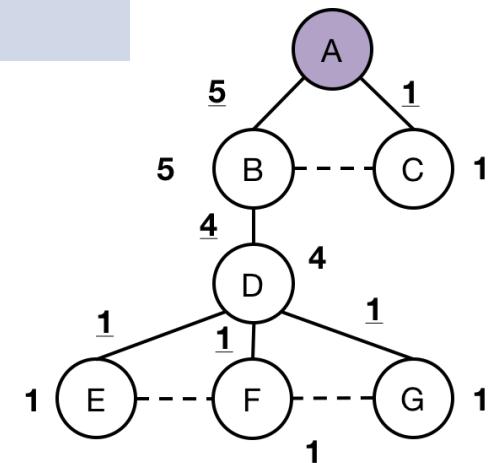


# Example (G, Credits)



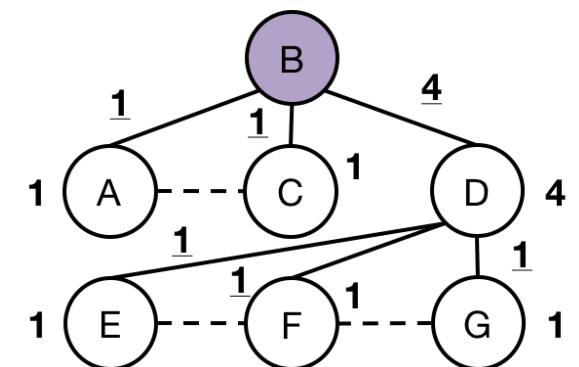
# Sum Contributions (A)

	AB	AC	BC	BD	DE	DG	DF	EF	GF
A	5	1		4	1	1	1		
B									
C									
D									
E									
F									
G									



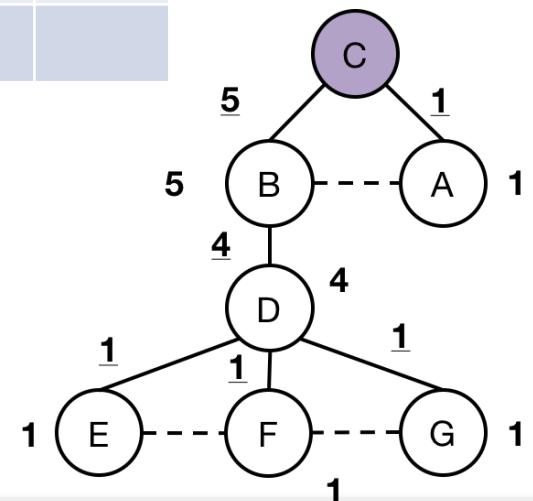
# Sum Contributions (B)

	AB	AC	BC	BD	DE	DG	DF	EF	GF
A	5	1		4	1	1	1		
B	1		1	4	1	1	1		
C									
D									
E									
F									
G									



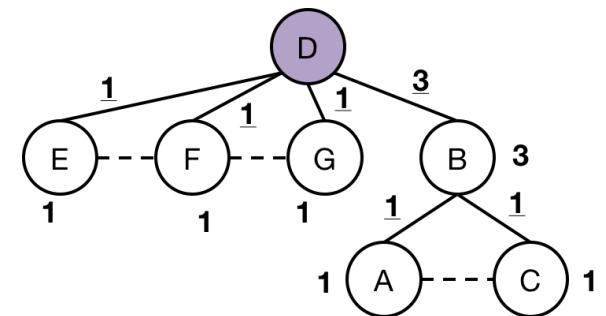
# Sum Contributions (C)

	AB	AC	BC	BD	DE	DG	DF	EF	GF
A	5	1		4	1	1	1		
B	1		1	4	1	1	1		
C		1	5	4	1	1	1		
D									
E									
F									
G									



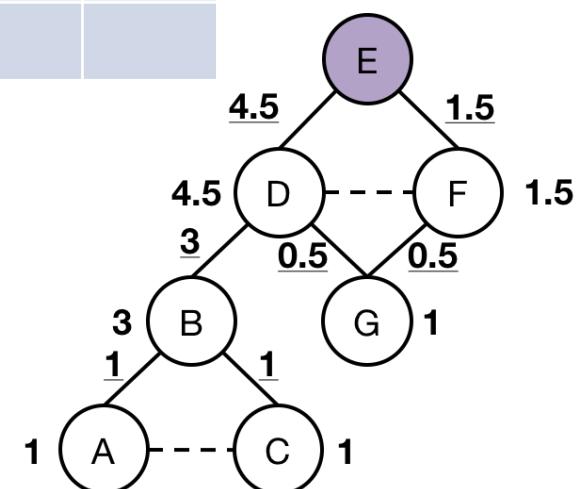
# Sum Contributions (D)

	AB	AC	BC	BD	DE	DG	DF	EF	GF
A	5	1		4	1	1	1		
B	1		1	4	1	1	1		
C		1	5	4	1	1	1		
D	1		1	3	1	1	1		
E									
F									
G									



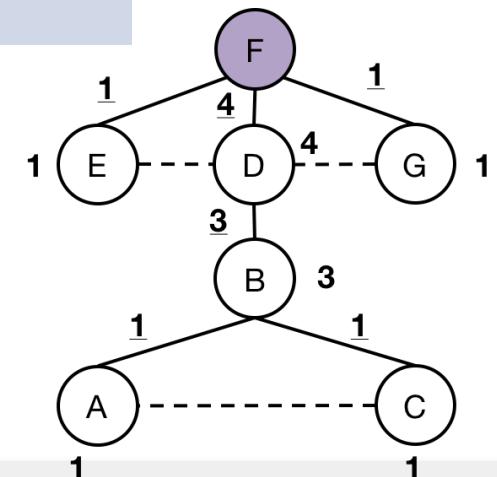
# Sum Contributions ( $E$ )

	AB	AC	BC	BD	DE	DG	DF	EF	GF
A	5	1		4	1	1	1		
B	1		1	4	1	1	1		
C		1	5	4	1	1	1		
D	1		1	3	1	1	1		
E	1		1	3	4.5	0.5		1.5	0.5
F									
G									



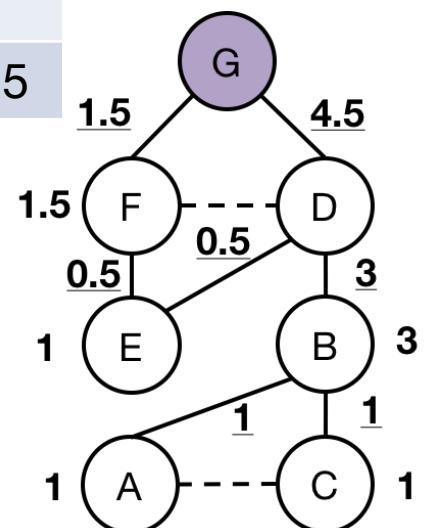
# Sum Contributions (F)

	AB	AC	BC	BD	DE	DG	DF	EF	GF
A	5	1		4	1	1	1		
B	1		1	4	1	1	1		
C		1	5	4	1	1	1		
D	1		1	3	1	1	1		
E	1		1	3	4.5	0.5		1.5	0.5
F	1		1	3			4	1	1
G									



# Sum Contributions (G)

	<b>AB</b>	<b>AC</b>	<b>BC</b>	<b>BD</b>	<b>DE</b>	<b>DG</b>	<b>DF</b>	<b>EF</b>	<b>GF</b>
<b>A</b>	5	1		4	1	1	1		
<b>B</b>	1		1	4	1	1	1		
<b>C</b>		1	5	4	1	1	1		
<b>D</b>	1		1	3	1	1	1		
<b>E</b>	1		1	3	4.5	0.5		1.5	0.5
<b>F</b>	1		1	3			4	1	1
<b>G</b>	1		1	3	0.5	4.5		0.5	1.5



# Sum Contributions (+)

	AB	AC	BC	BD	DE	DG	DF	EF	GF
A	5	1		4	1	1	1		
B	1		1	4	1	1	1		
C		1	5	4	1	1	1		
D	1		1	3	1	1	1		
E	1		1	3	4.5	0.5		1.5	0.5
F	1		1	3			4	1	1
G	1		1	3	0.5	4.5		0.5	1.5
+	10	2	10	24	9	9	8	3	3



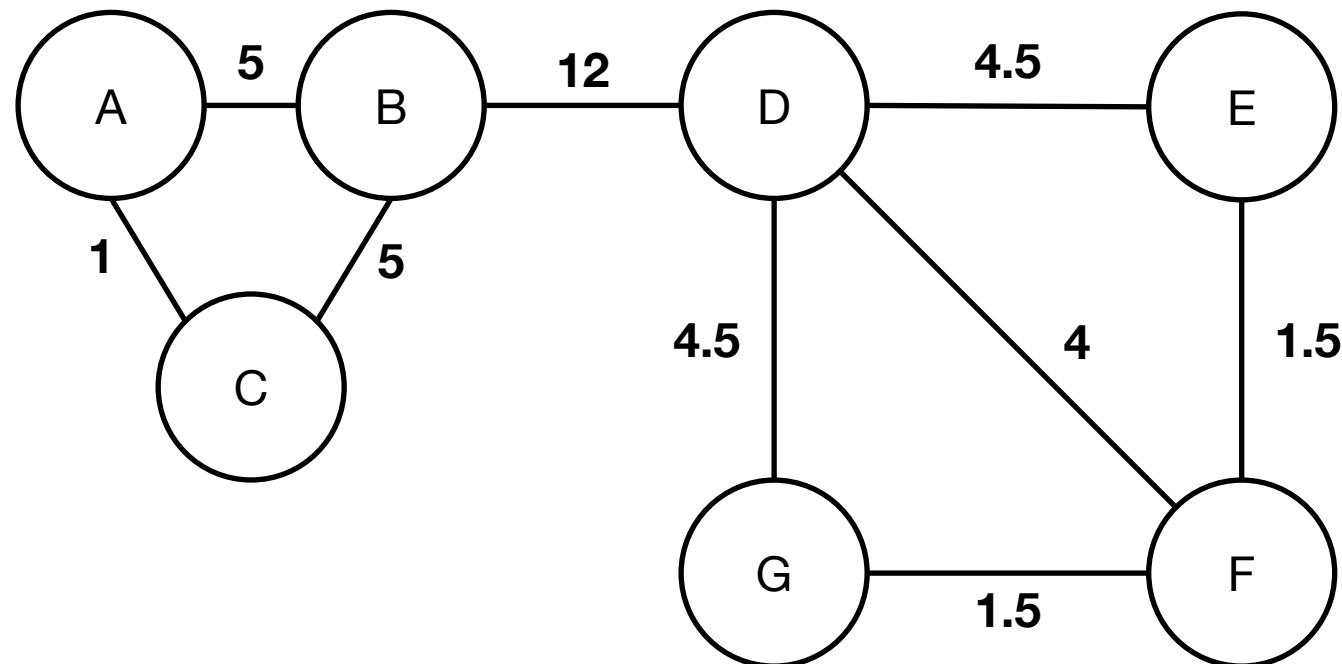
# Sum Contributions (/2)

	AB	AC	BC	BD	DE	DG	DF	EF	GF
A	5	1		4	1	1	1		
B	1		1	4	1	1	1		
C		1	5	4	1	1	1		
D	1		1	3	1	1	1		
E	1		1	3	4.5	0.5		1.5	0.5
F	1		1	3			4	1	1
G	1		1	3	0.5	4.5		0.5	1.5
+	10	2	10	24	9	9	8	3	3
/2	5	1	5	12	4.5	4.5	4	1.5	1.5



# Edge Contributions

AB	AC	BC	BD	DE	DG	DF	EF	GF
5	1	5	12	4.5	4.5	4	1.5	1.5

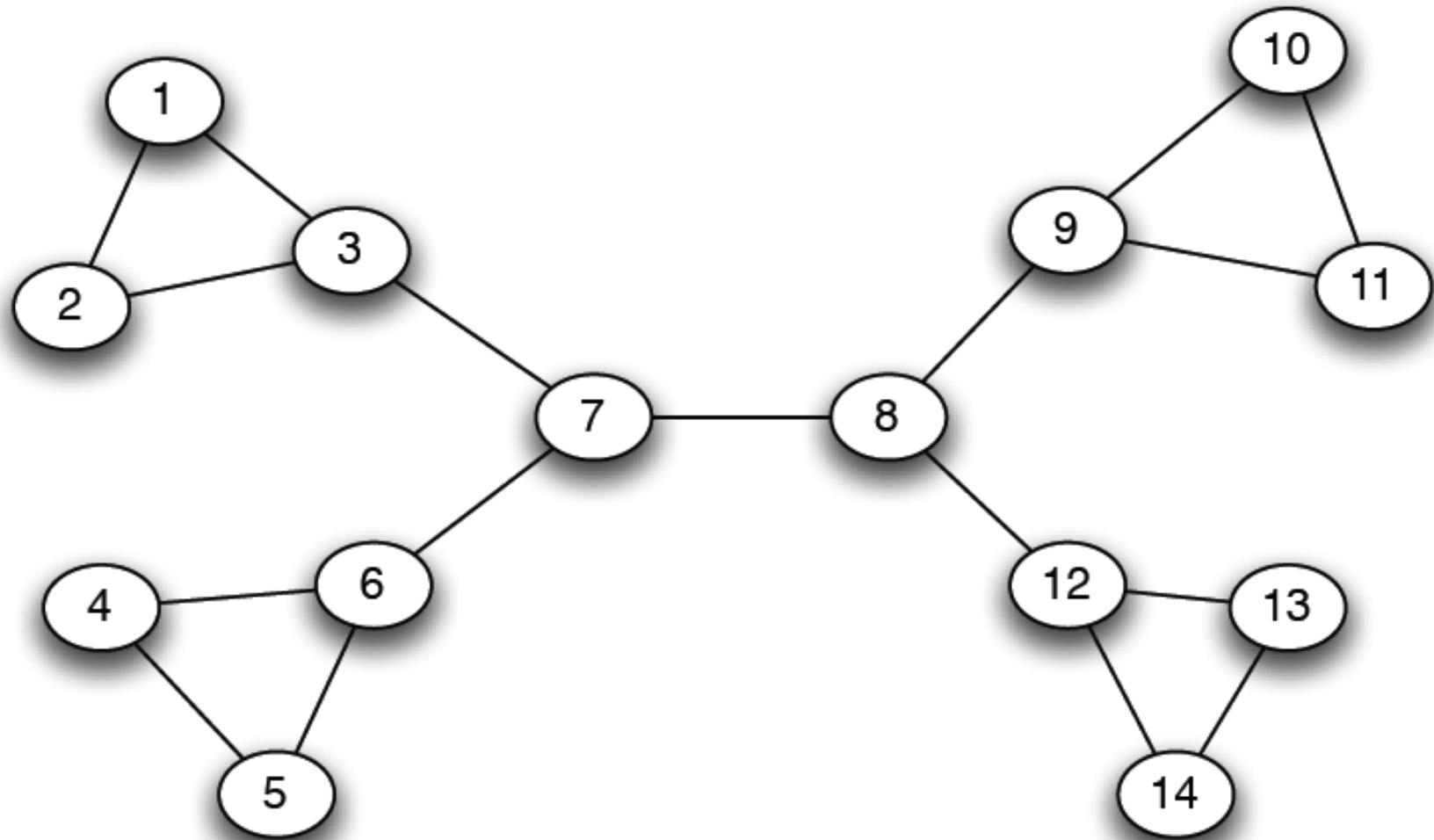


# Girvan-Newman

1. Repeat until no edges left
  - a) Calculate betweenness of edges
  - b) Remove edge(s) with highest betweenness

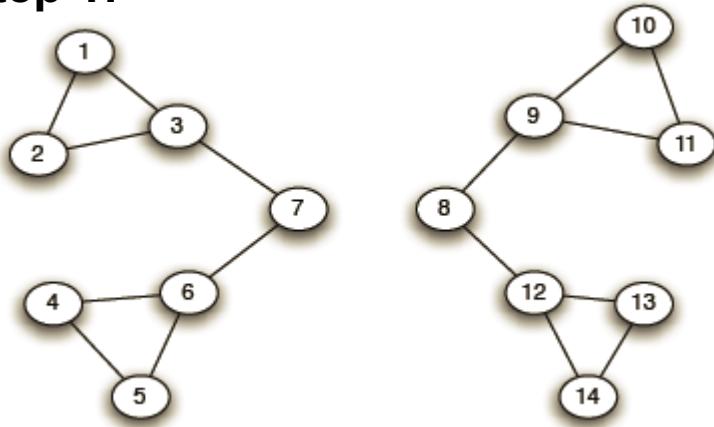


# Example (0)

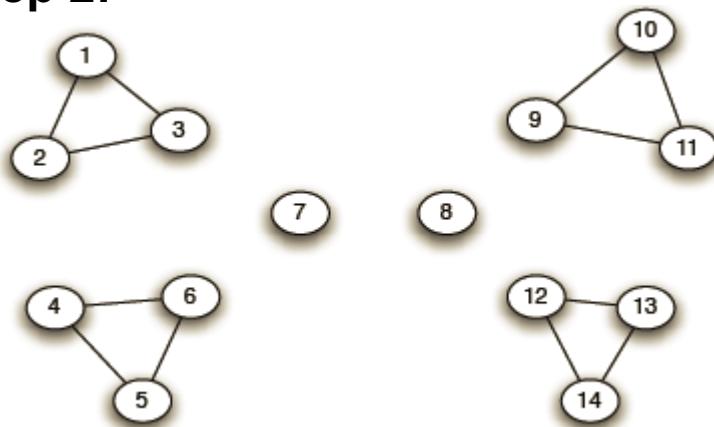


# Example (1)

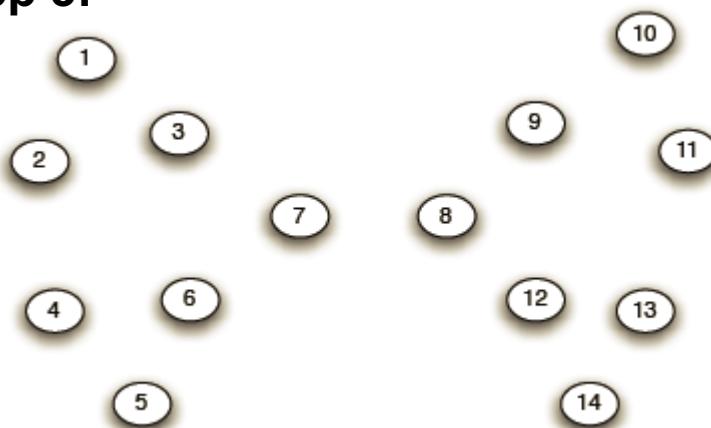
**Step 1:**



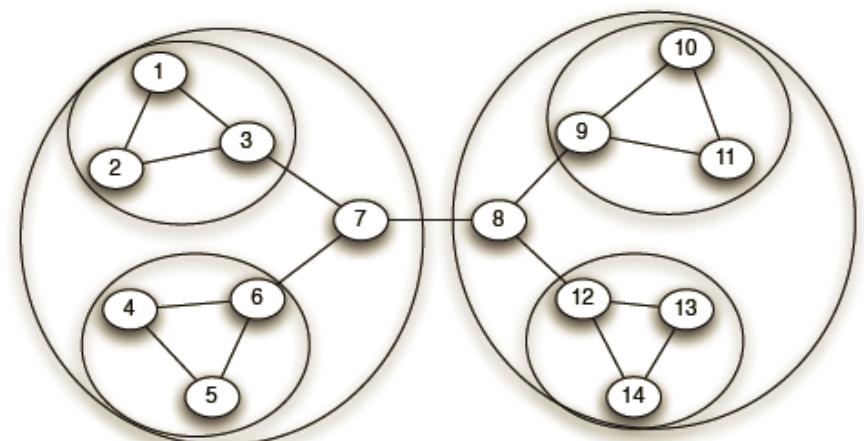
**Step 2:**



**Step 3:**



**Hierarchical network decomposition:**

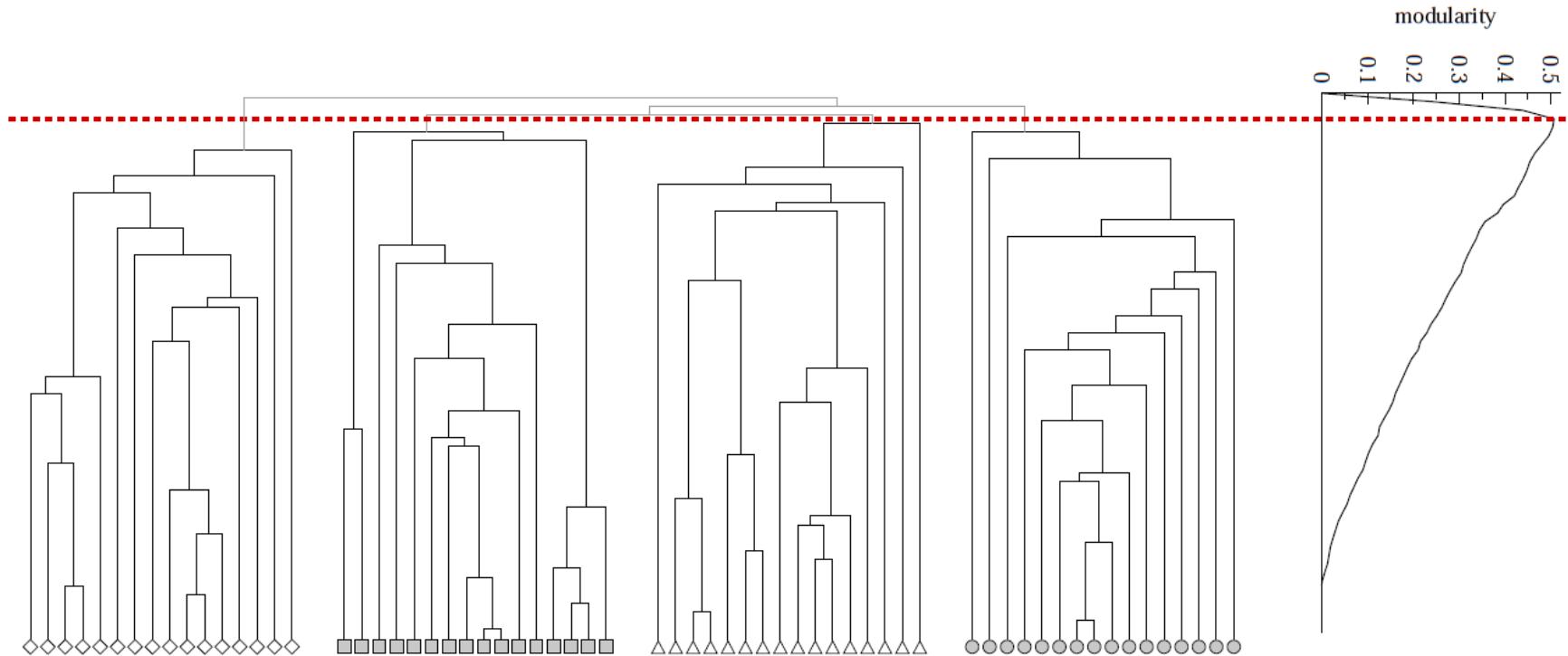


# How to Select # of Clusters?

- Similar to agglomerative, with a different metric: **Modularity ( $Q$ ; [-1, 1])**
- Idea: compare fraction of edges within a group to the fraction that would be observed for random connections



# Selecting # of Clusters



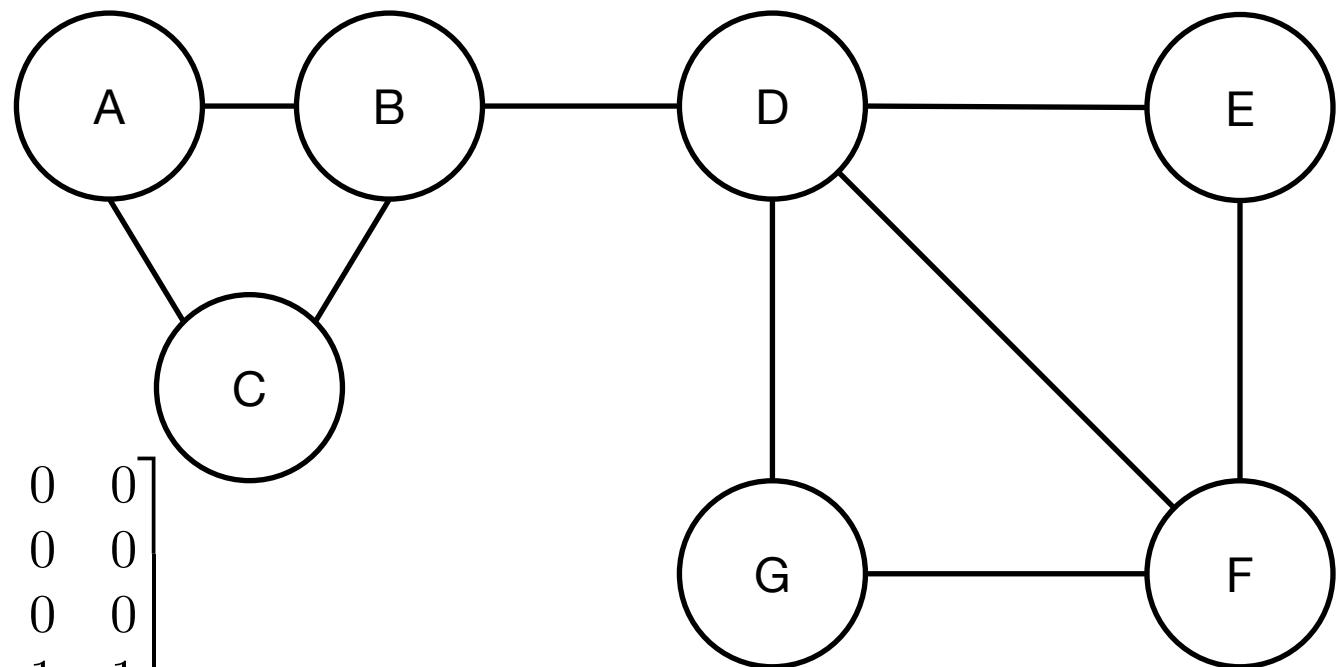
# Direct Partitioning

- We now look at an approach to divide a graph into two disjoint groups, the task of **bi-partitioning**, via **spectral analysis**
- To do so, we must express the graph as a matrix



# Adjacency Matrix (A)

$A_{ij} = 1$  if  $(i,j)$  is an edge, else 0

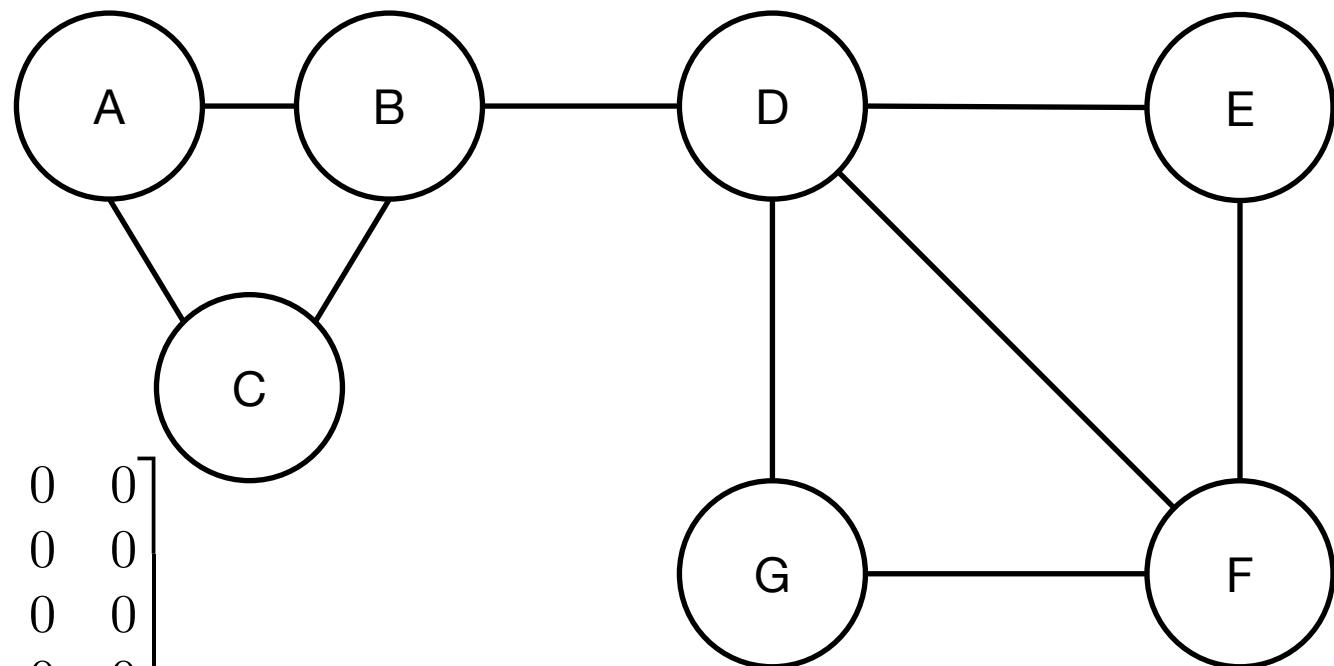


$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$



# Degree Matrix (D)

$$D_{ii} = \deg(i), \text{ else } 0$$

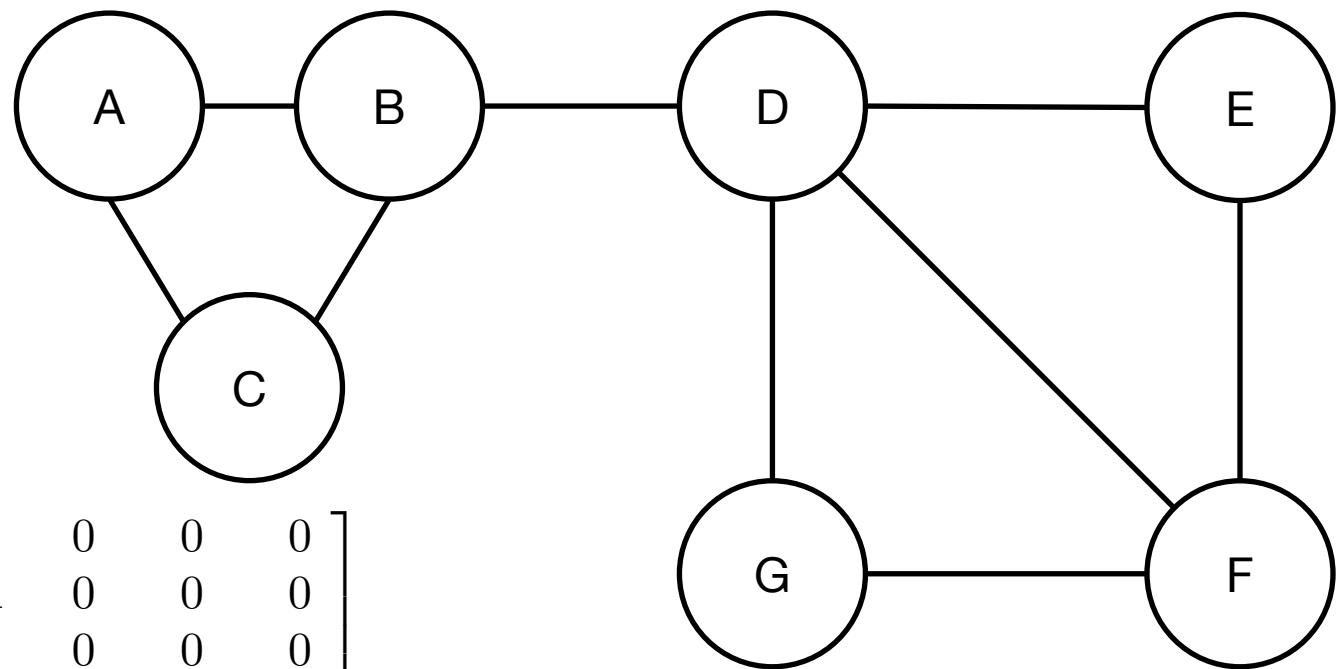


$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$



# Laplacian ( $L$ )

$$L = D - A$$



$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$



# Checkup

- Rows sum to... ?  
0 (degree – edges)
- Columns sum to... ?  
0 (degree – edges)
- Symmetric?  
Yes! (D is diagonal, A is symmetric for **undirected** graph)
- Diagonal Dominant?  
Yes! (edge case = lonely)

$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$



# The Plan

- We will now analyze the eigen-decomposition ( $Lv = \lambda v$ ) of the Laplacian
- It turns out that the smallest eigenvalues/vectors can tell us a lot!
  - Note:  $L$  is PSD, so  $\lambda \geq 0$
  - 0: connectivity
  - Next smallest: partitioning



# Checkup

- What is a non-trivial solution to the equation  $Lv = \lambda v$  for  $\lambda=0$  (i.e.  $Lv=0$ )?

$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$



# Answer

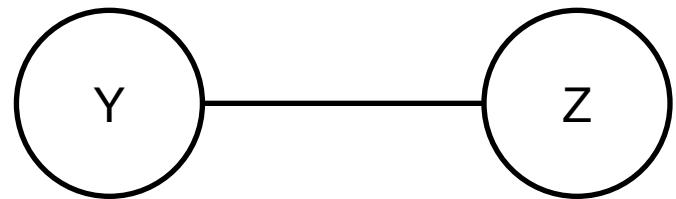
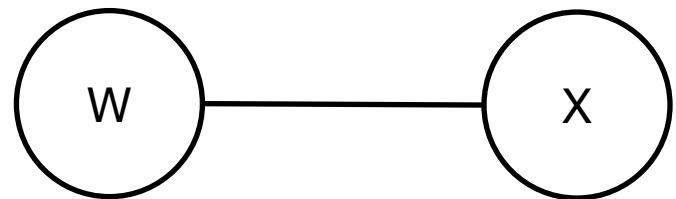
- What is a non-trivial solution to the equation  $Lv = \lambda v$  for  $\lambda=0$  (i.e.  $Lv=0$ )?
- You already solved it: 1's (column vector of constants)
- ACTUALLY, not the whole story...

$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$



# Checkup

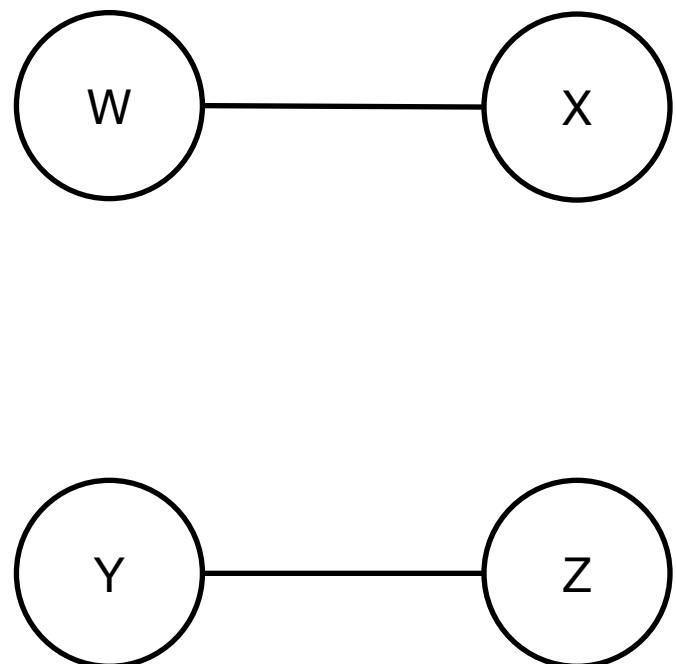
- Laplacian of the following graph?



# Answer

- Laplacian of the following graph?

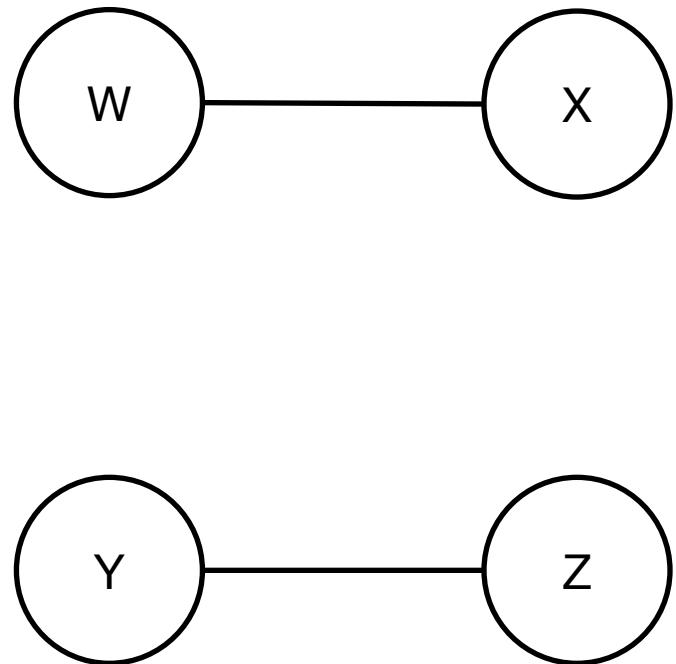
$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$



# Checkup

- Null-space of  $L$ ? (i.e.  $Lv=0$ )

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$



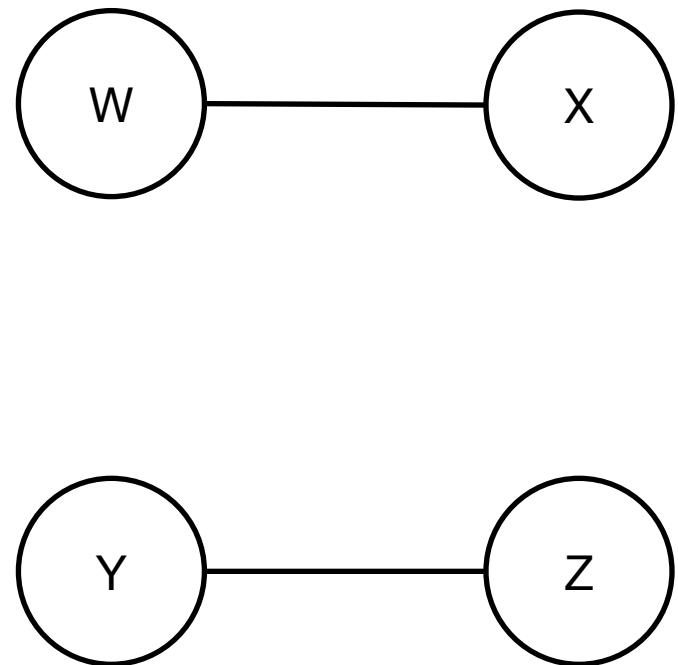
# Answer

- Null-space of  $L$ ? (i.e.  $Lv=0$ )

$$[1 \ 1 \ 0 \ 0]^T$$

$$[0 \ 0 \ 1 \ 1]^T$$

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$



# Connectivity via 0<sup>th</sup> Eigenvalue

- If nodes i and j are connected, their values in the corresponding eigenvector must be equal
  - This “cancels out” across all rows
- So if the graph is connected, the 0<sup>th</sup> eigenvector  
 $[c \ c \ c \ \dots]^T$ 
  - Transitivity!
- Otherwise, null-space of L provides connected components
  - Could be detected by a simple clustering algorithm (e.g. k-Means)



# Second-Smallest Eigenvalue

- Let's call this  $\lambda_1$  (with corresponding  $v_1$ )
- If  $\lambda_1$  is 0, what do we know?  
That there's not much sense in bi-partitioning :)
- Foreshadowing: from here, we'll construct an objective function for a bi-partitioning, and it will turn out to be minimized precisely by  $v_1$



# Setup

- For a given graph, we want to assign each node to one of two groups
  - Let's represent this assignment, per node, as the value -1 or +1 for variable  $n_i$
  - So, a vector,  $n$ , of length  $|V|$  of either -1 or 1
- And do so in a way that minimizes the number of graph edges between the groups - so let's minimize...
$$(n_i - n_j)^2 \text{ for all edges } (i, j)$$



# Checkup

What is  $(n_i - n_j)^2$  for two connected nodes that ...

- Are in the same group?

$$(1 - 1)^2 = 0; (-1 - -1)^2 = 0$$

- Are in different groups?

$$(1 - -1)^2 = 4; (-1 - 1)^2 = 4$$

So minimizing the sum across edges also minimizes cross-group edges – woohoo!



# Converting to Matrices

- So we have  $\sum(n_i - n_j)^2$ 
  - Or:  $\sum (n_i^2 - 2n_i n_j + n_j^2)$
- The way to express this in terms of the Laplacian is...  $n^T L n$ 
  - Each node will have  $n_i^2 d(n_i)$  term (diagonal), so one  $n_i^2$  per edge =  $n_i^2 + n_j^2$
  - Each edge will have two  $(-1)(n_i n_j)$  terms (symmetric) =  $-2n_i n_j$
  - Each non-existent edge will multiply by zero from  $A_{ij}$  being zero



# Checkup

- What is the easiest way to minimize the following function:  $n_i^2 - 2n_i n_j + n_j^2$



# Answer

- What is the easiest way to minimize the following function:  $n_i^2 - 2n_i n_j + n_j^2$
- $n = [0 \ 0 \dots 0]$ 
  - So we need to force values...



# Forcing Non-Trivial Solutions

- Easy:  $\sum n^2 = |V|$ 
  - The sum of the assignments must equal the total number of nodes
- Or, in vector notation  $n^T n = |V|$



# Objective Function – Minimize!

$$n^\top L n + \lambda(n^\top n - |V|)$$



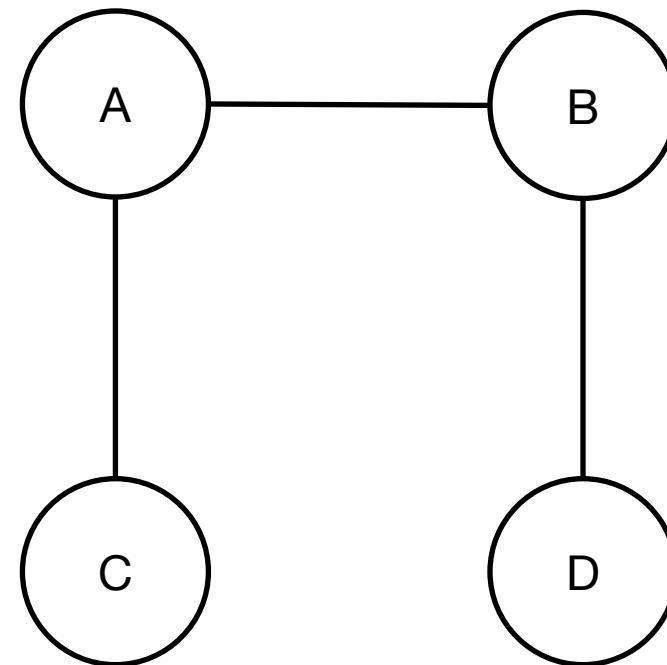
$$Ln - \lambda n = 0$$

SO, to find a good partitioning we need to...

- Find an eigenvector...
- with the second-smallest eigenvalue
  - Why doesn't 0 work?



# Example (1): Graph



# Example (1): Laplacian

$$\begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$



# Example (1): Eigendecomposition

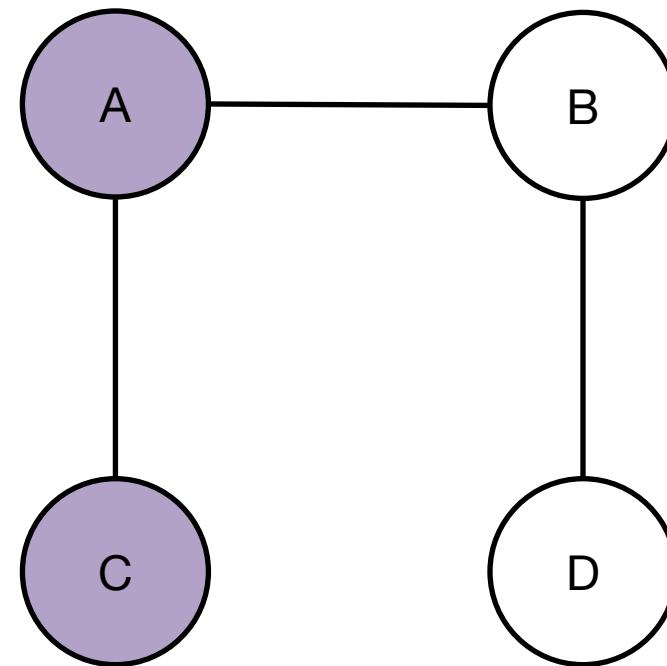
```
In [45]: np.linalg.eigh(A5)
```

```
Out[45]: (array([-1.76298406e-16,  5.85786438e-01,  2.00000000e+00,
                   3.41421356e+00]),
           array([[[-0.5        , -0.27059805,  0.5        , -0.65328148],
                  [-0.5        ,  0.27059805,  0.5        ,  0.65328148],
                  [-0.5        , -0.65328148, -0.5        ,  0.27059805],
                  [-0.5        ,  0.65328148, -0.5        , -0.27059805]])))
```

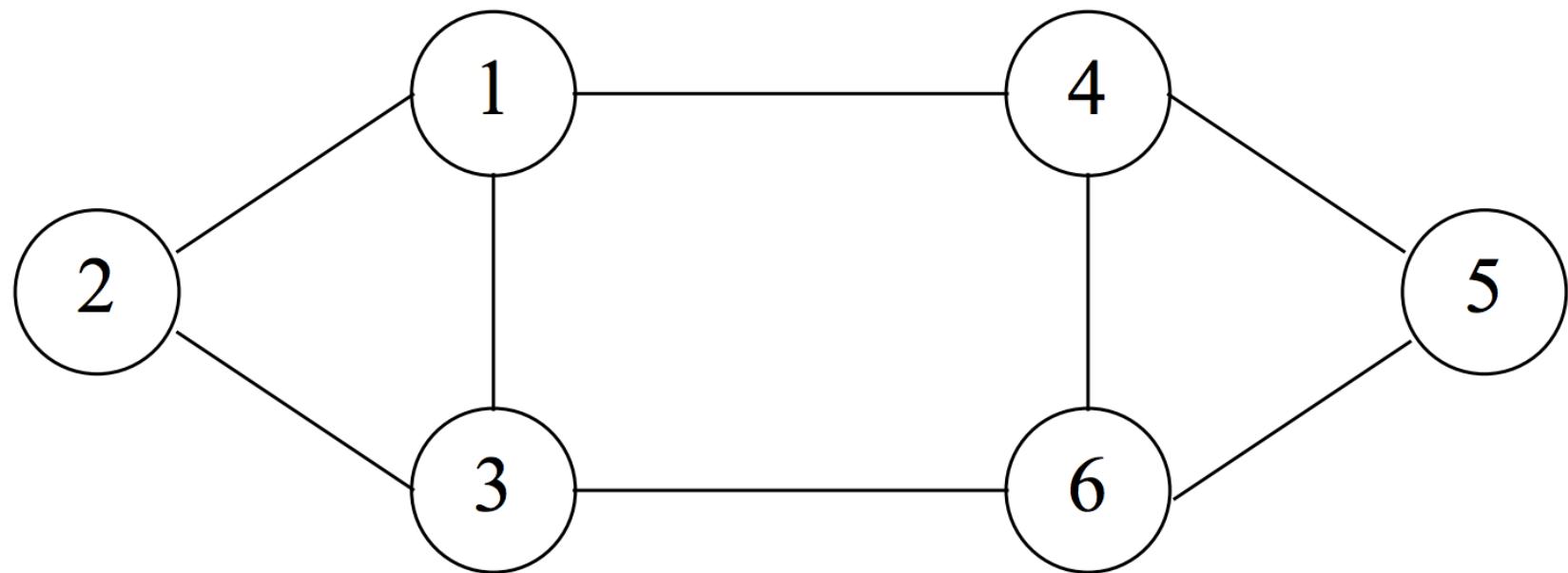
Sign = Partitioning (+ vs -)  
+ {B, D}  
- {A, C}



# Example (1): Partitioning



# Example (2): Graph



# Example (2): Laplacian

$$\begin{bmatrix} 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & 0 & 0 & -1 \\ -1 & 0 & 0 & 3 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & -1 & -1 & 3 \end{bmatrix}$$

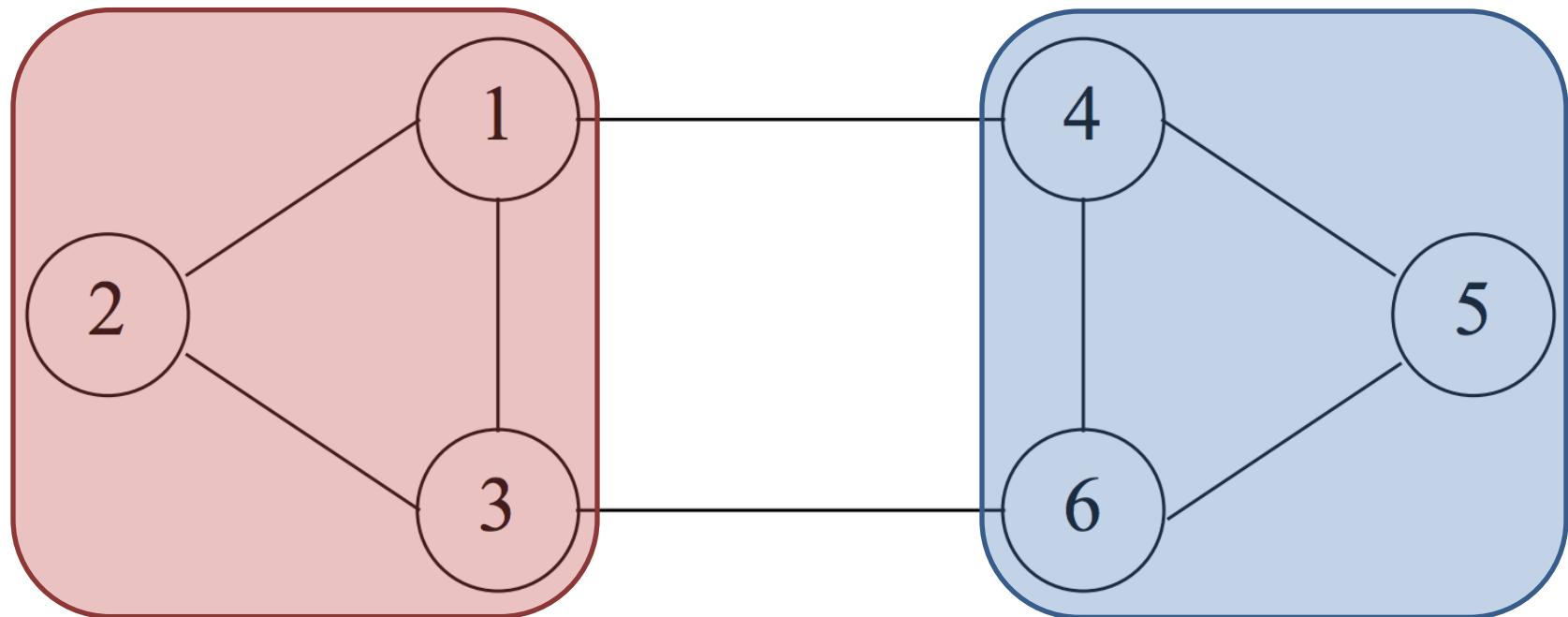


# Example (2): Eigendecomposition

Eigenvalue	0	1	3	3	4	5
Eigenvector	1	1 2 1 -1 -2 -1	-5 4 1 -5 4 1	-1 -2 3 -1 -2 3	-1 1 -1 1 -1 1	-1 0 1 1 0 -1



# Example (2): Partition



# Many Other Topics

- Overlapping communities
- Network/Node properties
  - Triangles, neighborhoods
  - Centrality, influence
- Network formation
- Problems
  - Link prediction

