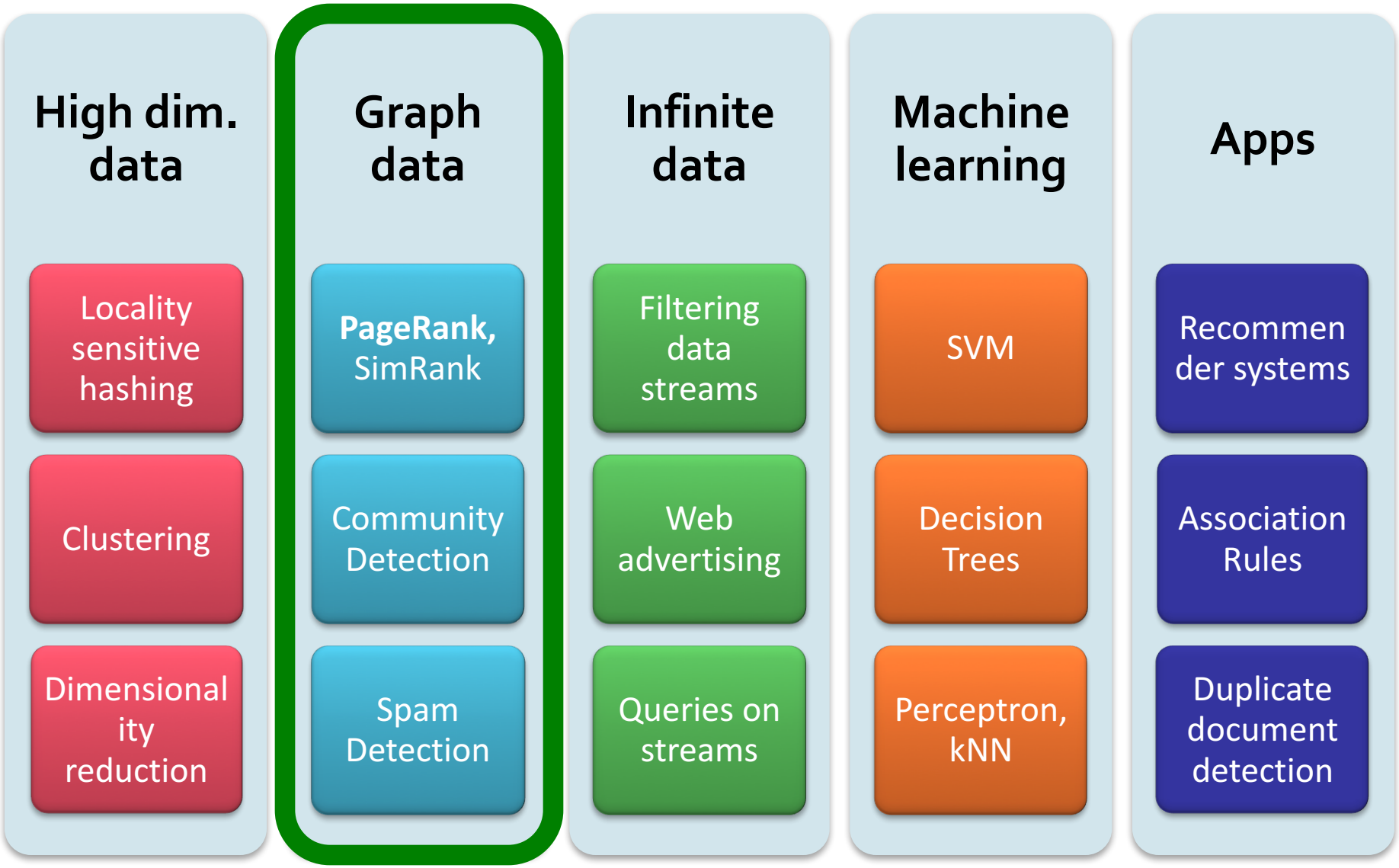
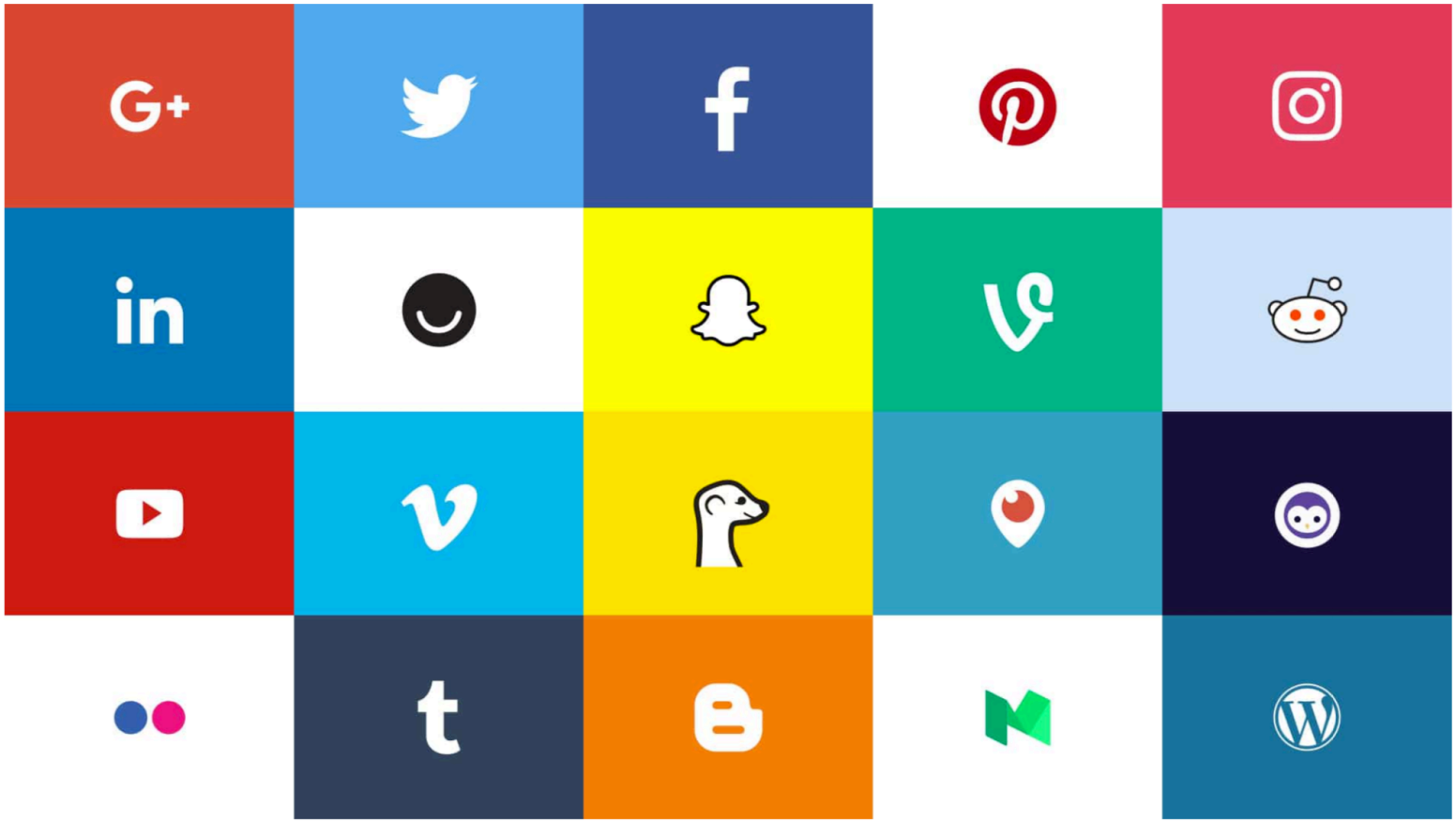
Mining Social Network Graphs: Community Detection

CSCD 430/530 Big Data Analytics

New Topic: Graph Data!



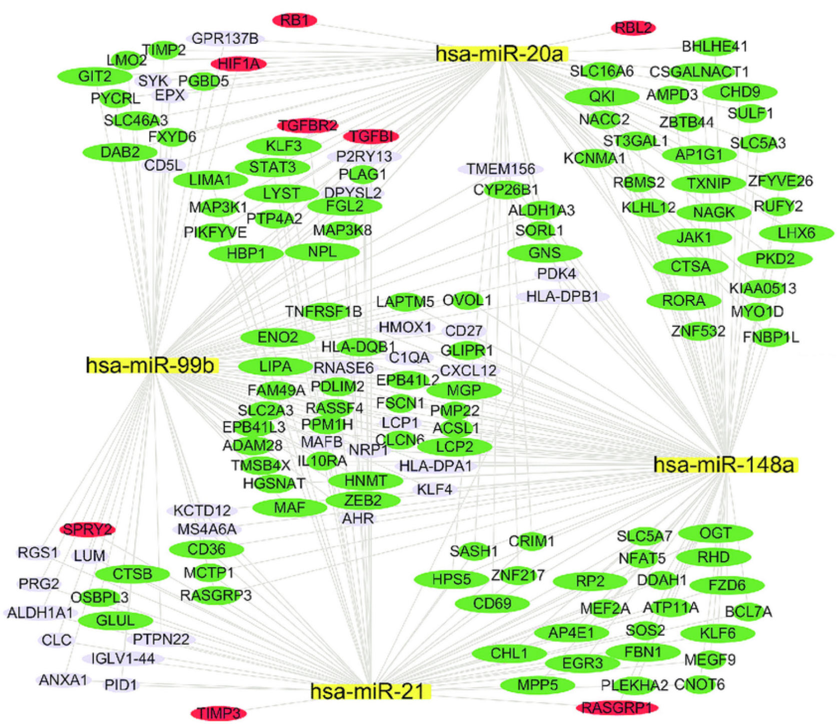
Social Networks



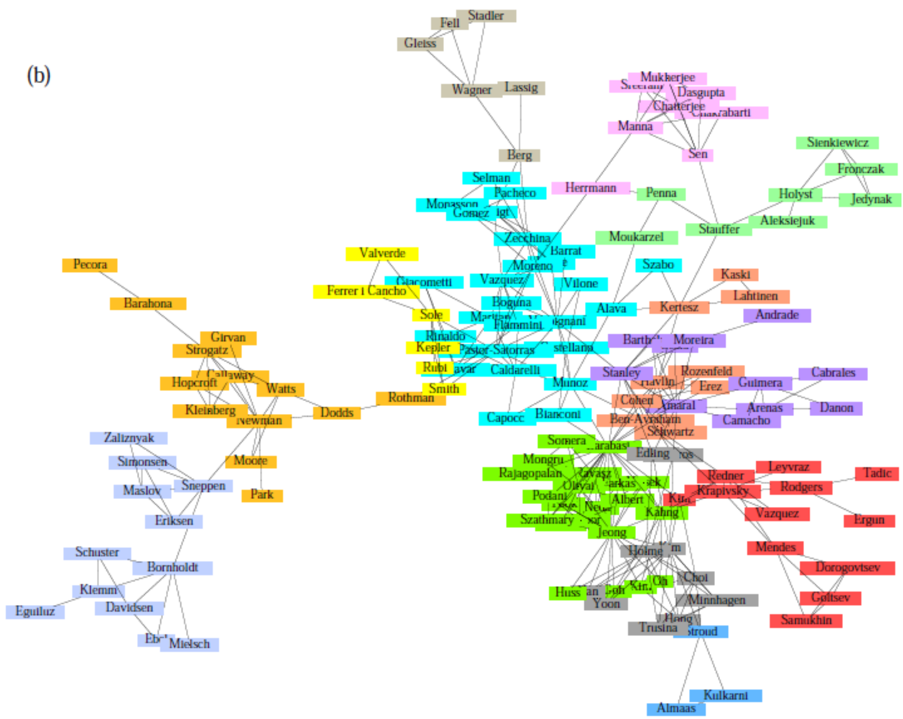
Social Networks (Obvious)

Social Networks (Less Obvious)

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



Interaction network of genes

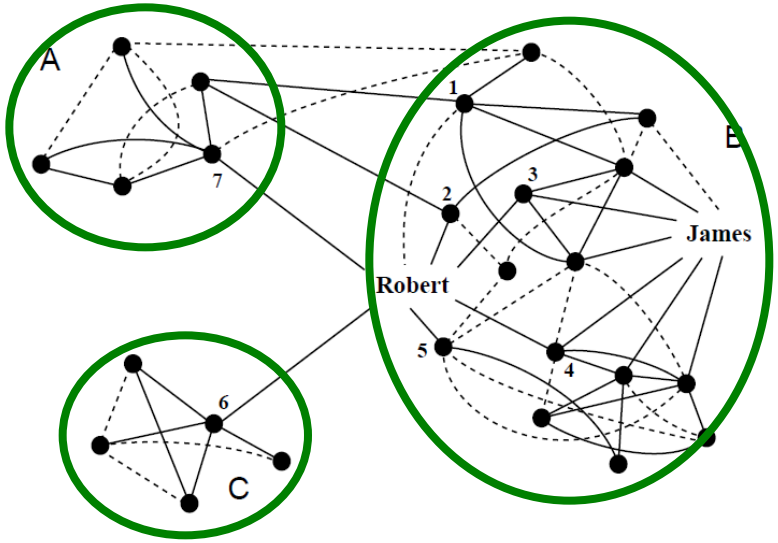


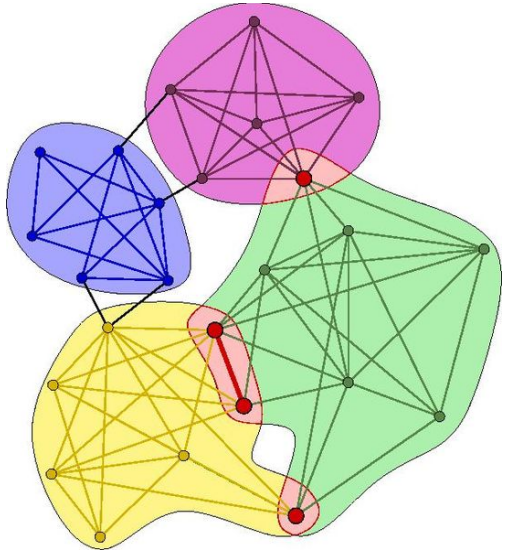
Communities in physics collaborations

Social Networks as Graphs

- Naturally modeled as graphs - social graphs
- **Entities** as nodes
- **Relationships** as edges
- **Degree** associated with the relationship as label of the edge
- Undirected or directed

Community Detection (10.1-2)





Goal: Community Dectection

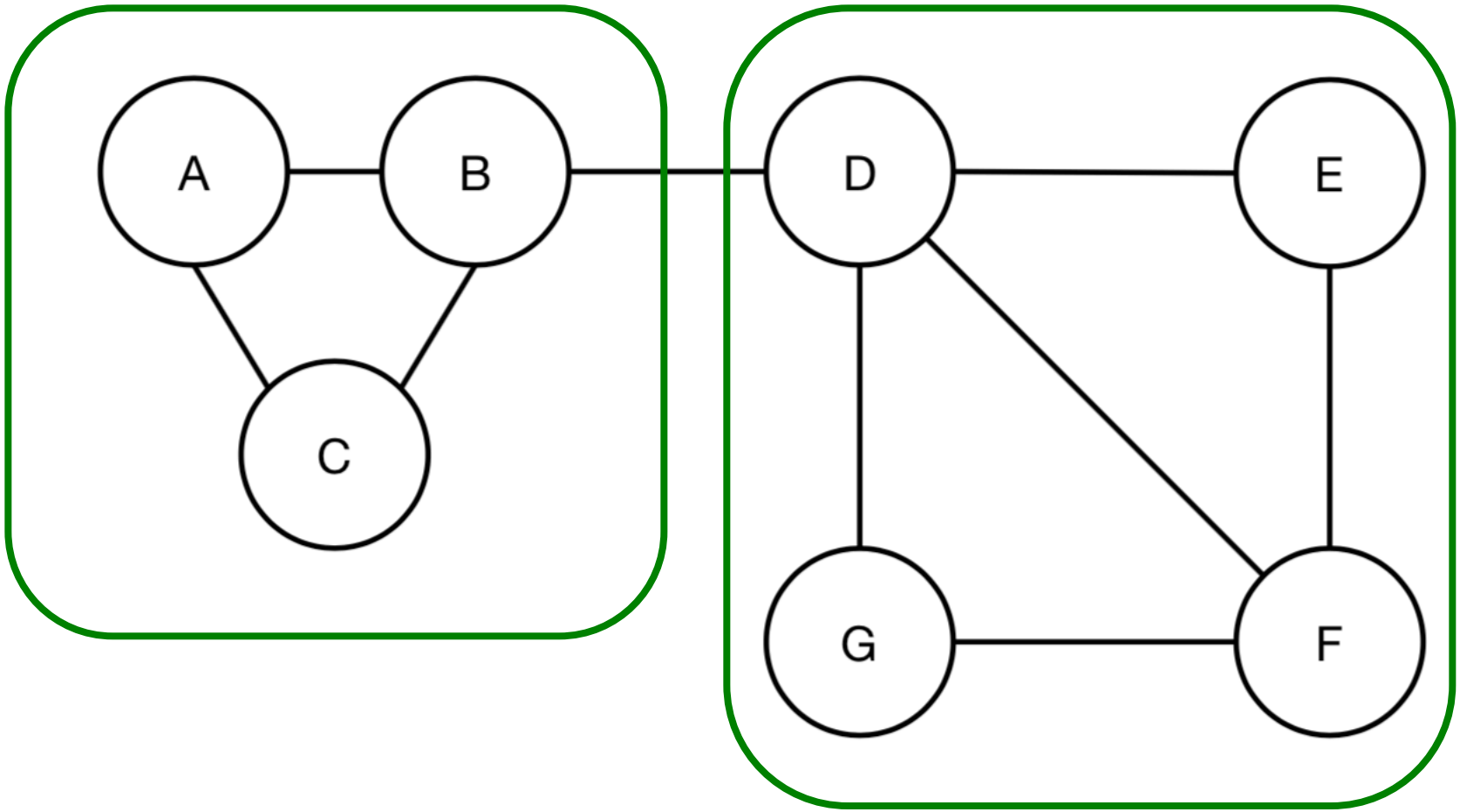
- Locality assumption
  - Typically assumed that if node A is connected to both B and C, then it is more likely than random that B and C are connected
- Our focus: **Community Detection**
  - Finding groups of densely connected nodes
  - Tightly related to the locality assumption – i.e. would not work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

A Question

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

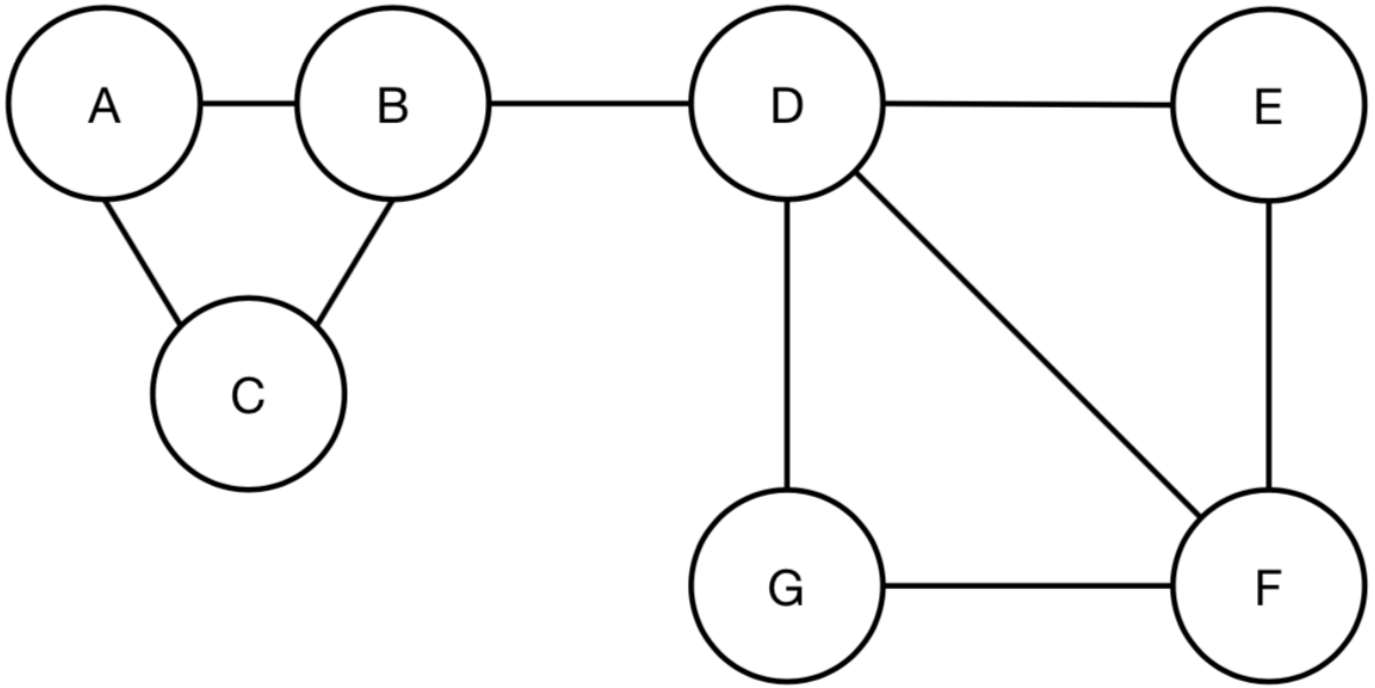
$$\binom{7}{2} = \frac{7!}{2!(7-2)!} = \frac{7(7-1)}{2} = 21$$

Example Graph



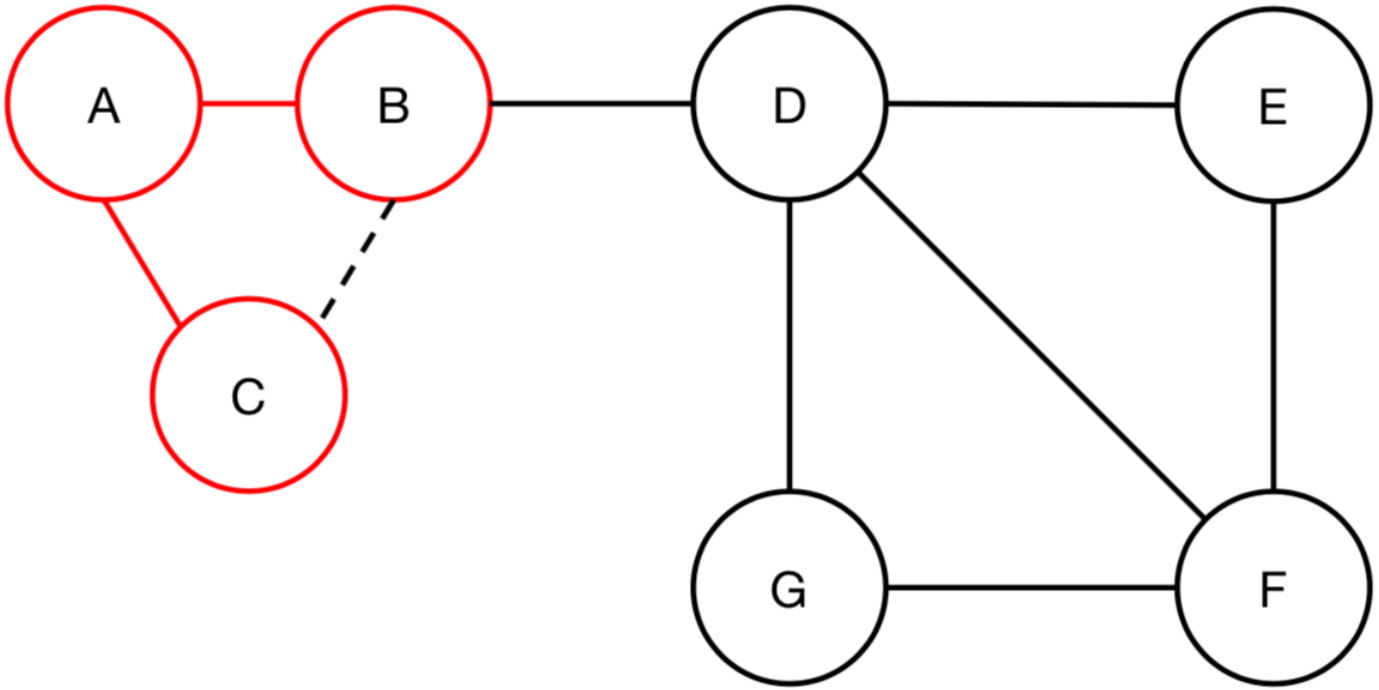
Intuitively, how many communities does this graph show?

Nodes, Edges, Average Connectivity?



- Nodes: 7
- Edges: 9
- Avg Connectivity: 9/21 ~ 0.43

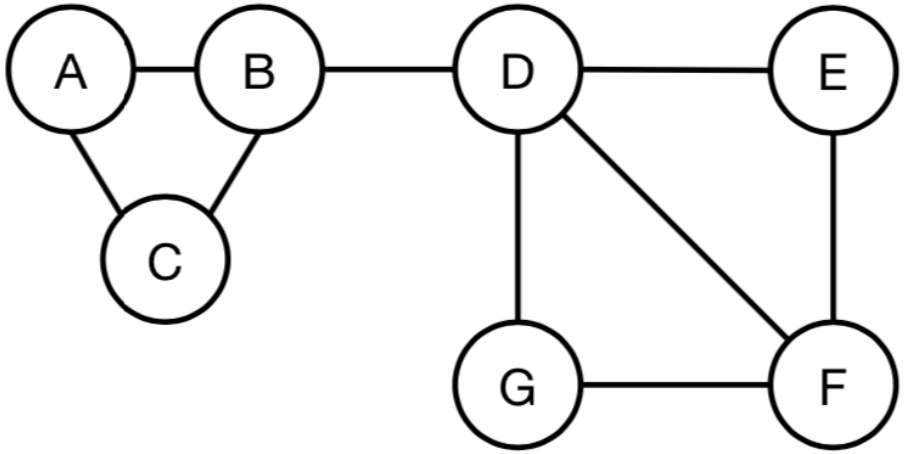
Given A-B, A-C; Expected B-C?



- Nodes: 7
- Edges: 9
- Avg Connectivity: 9/21 ~ 0.43
- Expected Locality: 7/19 ~ 0.37

Actual Locality: Evaluate Node Triples

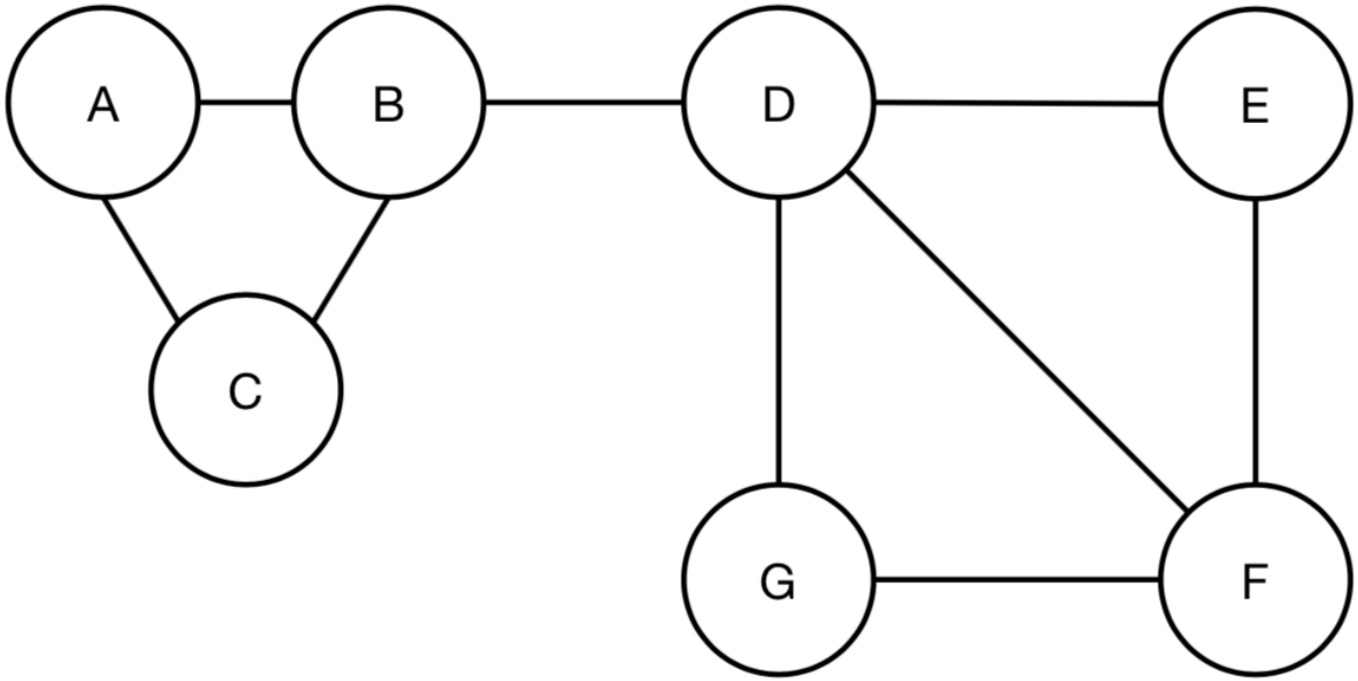
- **A**
  - BC ✅
- **B**
  - AC ✅, AD ❌
  - CD ❌
- **C**
  - AB ✅
- **D**
  - BE ❌, BF ❌, BG ❌
  - EF ✅, EG ❌
  - FG ✅
- **E**
  - DF ✅



- **F**
  - DE ✅, DG ✅
  - EG ❌
- **G**
  - DF ✅

- ✅ = 9, ❌ = 7
- ✅ / (✅ + ❌) = 9/16 ~ 0.56

Conclusion?



- Nodes: 7
- Edges: 9
- Avg Connectivity: 9/21 ~ 0.43
- Expected Locality: 7/19 ~ 0.37
- Actual Locality: 9/16 ~ 0.56 (**>> Expected!**)

Let's Cluster!

Clustering

- Given a set of data points, group them into a clusters so that:
  - points within each cluster are similar to each other
  - points from different clusters are dissimilar
- Usually, points are in a high dimensional space, and similarity is defined using a distance measure
  - Euclidean, Cosine, Jaccard, edit distance, ...

Distance Measures

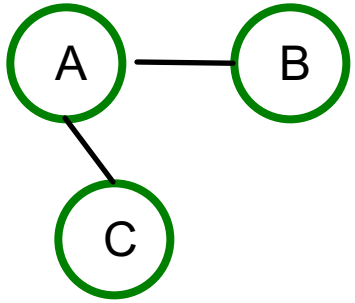
- $\boldsymbol{d}(\cdot)$ is a distance measure if it is a function from pairs of points **x, y** to real numbers such that:
  - $d\ x,y \geq 0$
  - $d(x,y) = 0\ if f\ x = y$
  - $d(x,y) = d(y,x)$
  - $d\ (x, y) \leq d(x, z) + d(z, y)$ (triangle inequality)

Clustering Methods

- Hierarchical (Agglomerative):
  - Initially, each point in cluster by itself.
  - Repeatedly combine the two “nearest” clusters into one.
  - Or the opposite way (divisive hierarchical clustering)
- Point Assignment:
  - Maintain a set of clusters.
  - Place points into their “nearest” cluster.

Distance Measures on Graphs

- Distance measures on social-network graphs can be tricky.
- Example: 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)



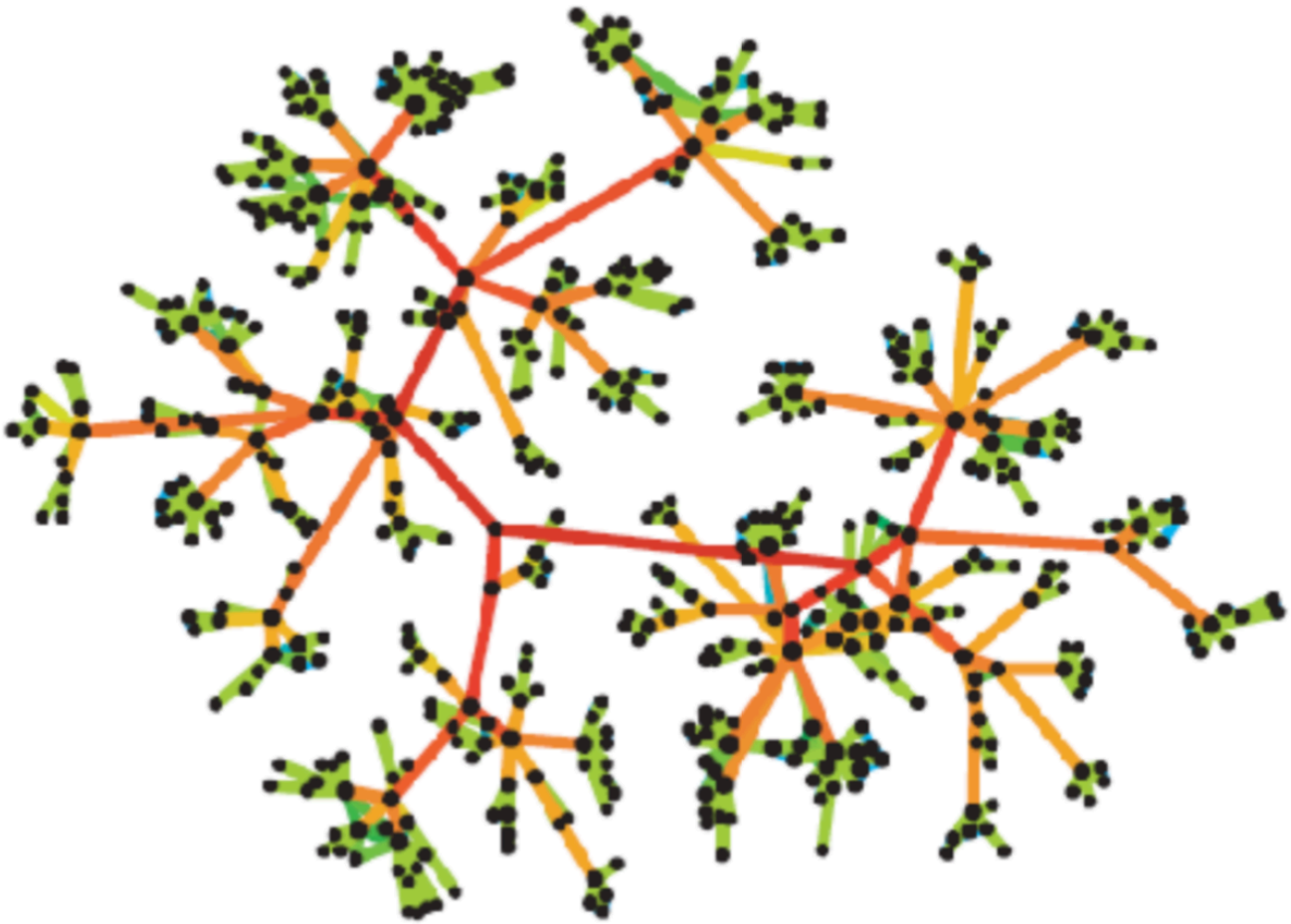
- Lead to unfortunate results in typical clustering algorithms

Betweenness as Distance Measure

- 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) = number of shortest paths passing through the edge *ab*

Betweenness Intuition



Edge betweenness in a real network

[Girvan-Newman '02]

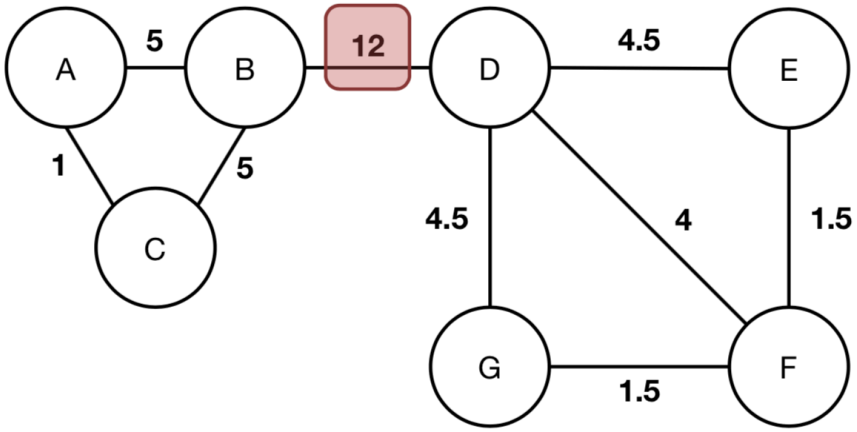
Compute Betweenness: Girvan-Newman

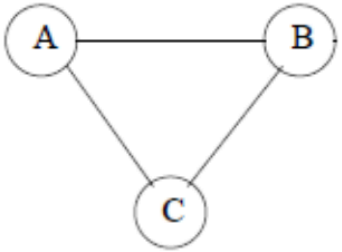
- Divisive hierarchical clustering based on the notion of edge **betweenness**:

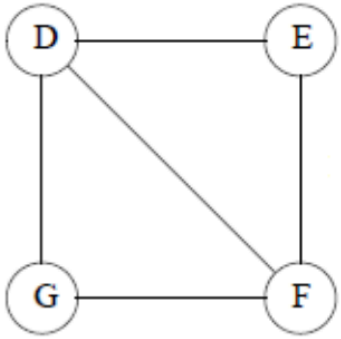
  Number of shortest paths passing through the edge

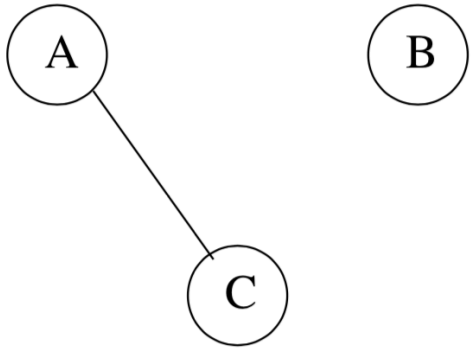
- **Girvan-Newman Algorithm:**
  - **Undirected unweighted networks**
  - **Repeat until no edges left or certain criteria met**
    - Calculate betweenness of edges
    - Remove edges with highest betweenness
  - Connected components are communities
  - Gives a hierarchical decomposition of the network

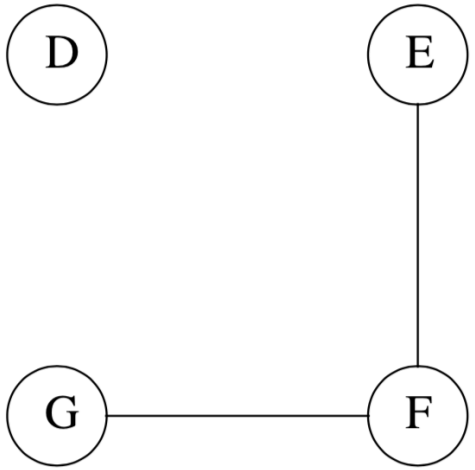
Divisive Hierarchical Clustering











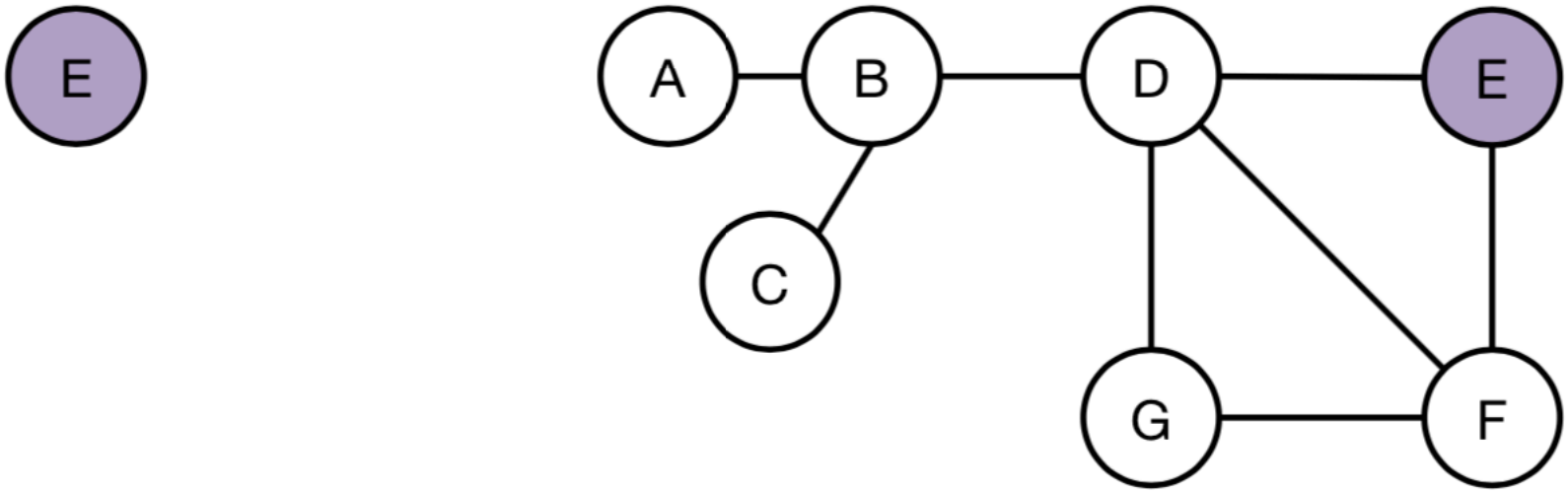
All the edges with betweenness 4 or more have been removed

Computing Betweenness

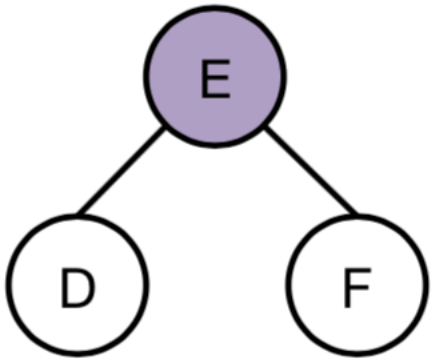
1. For each node
   a) Breadth-First Search (BFS)
      - WHY?
   b) For each edge, compute betweenness

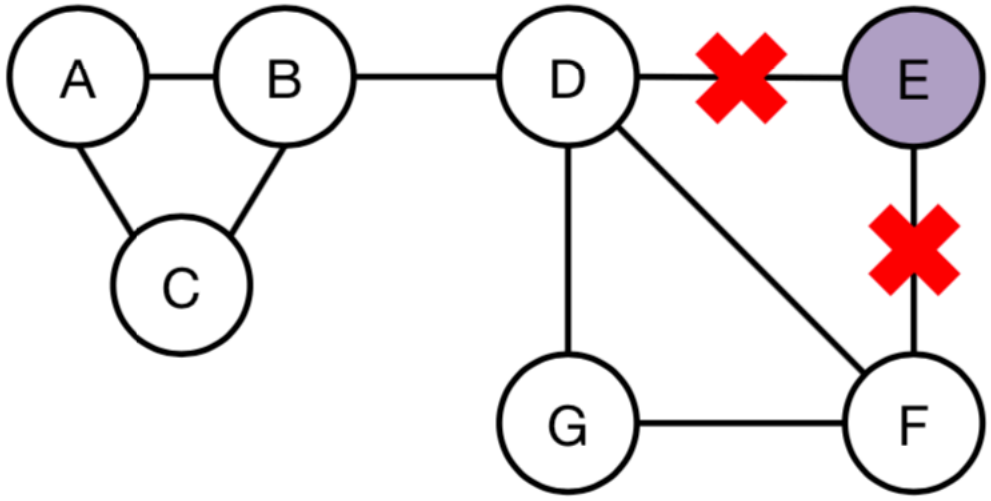
2. For each edge
   a) Sum contributions from trees above
   b) Divide by 2 (prevents duplication)

Example (E, BFS.o)

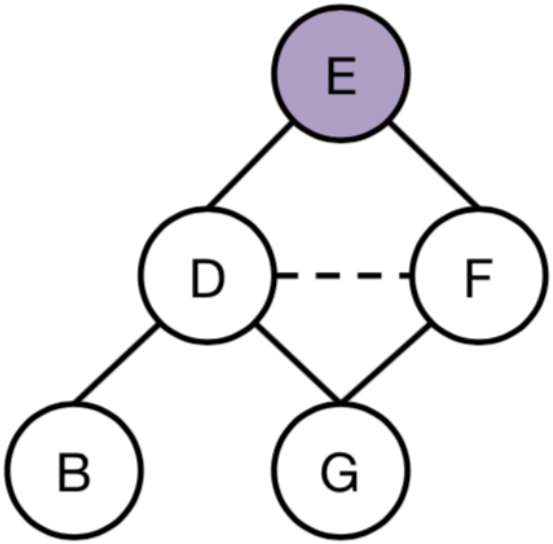


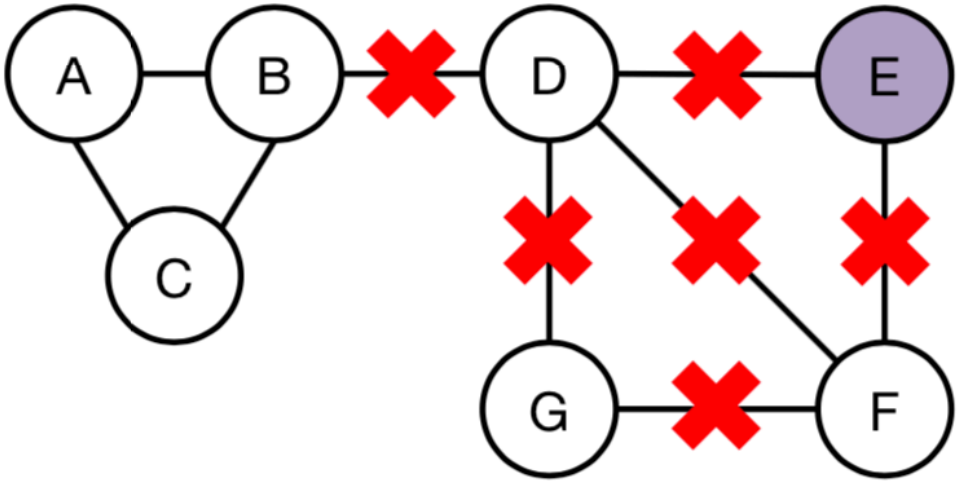
Example (E, BFS.1)



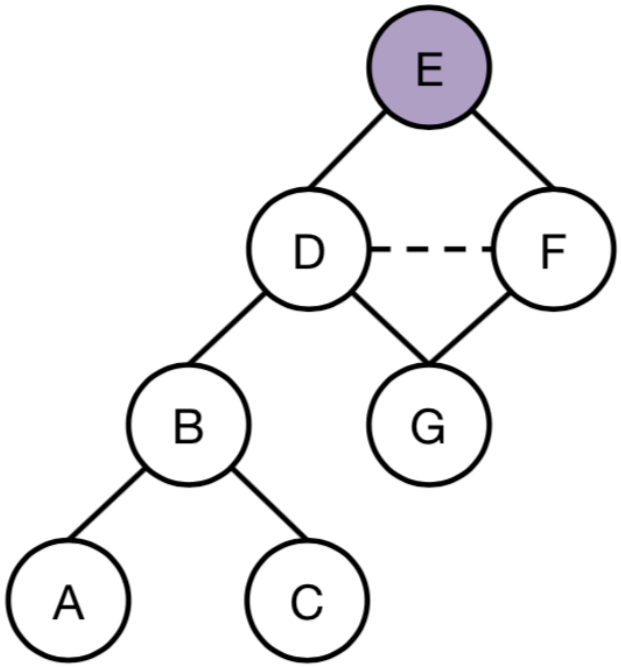


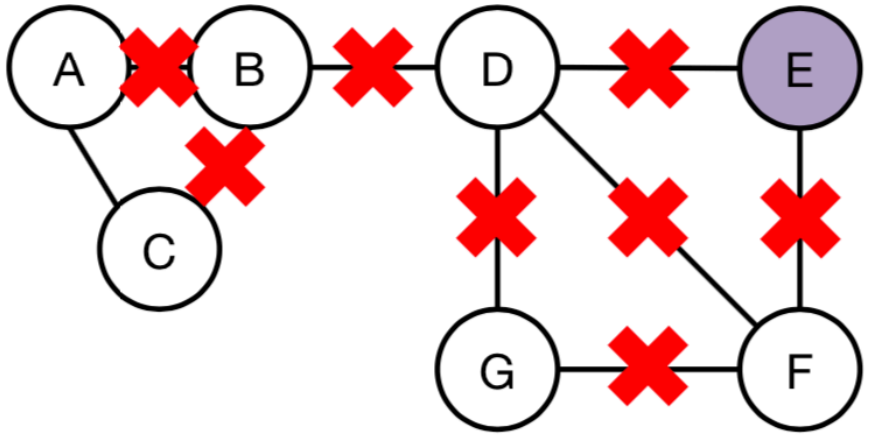
Example (E, BFS.2)



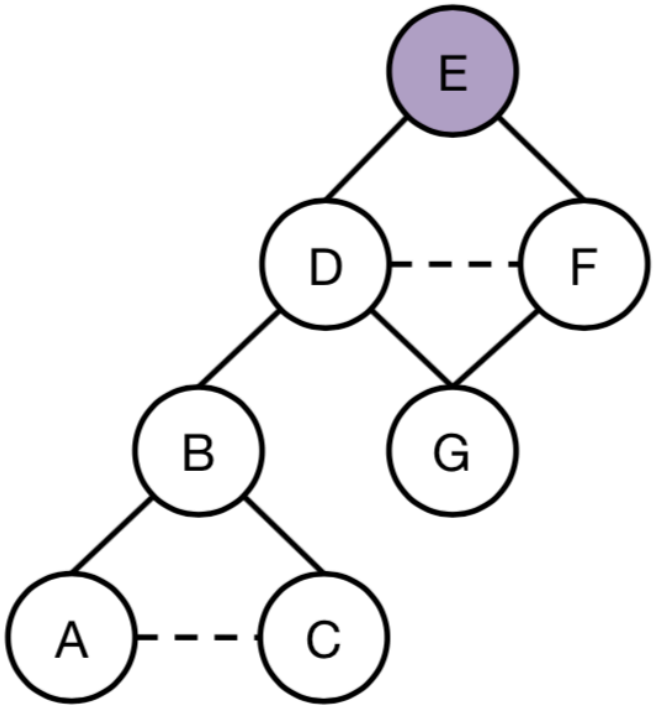


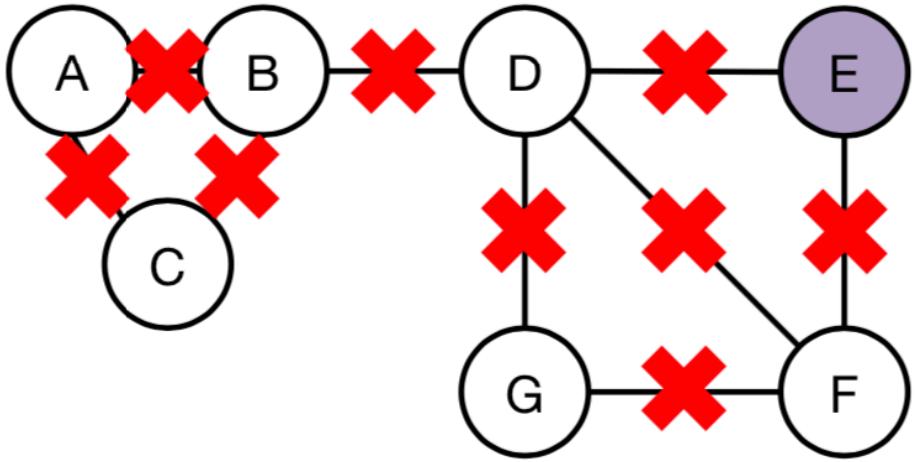
Example (E, BFS.3)





Example (E, BFS.Done)

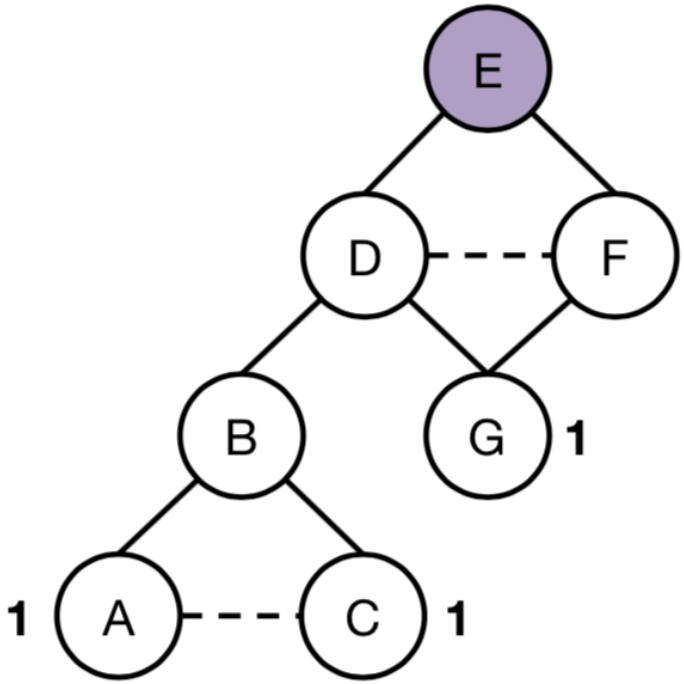


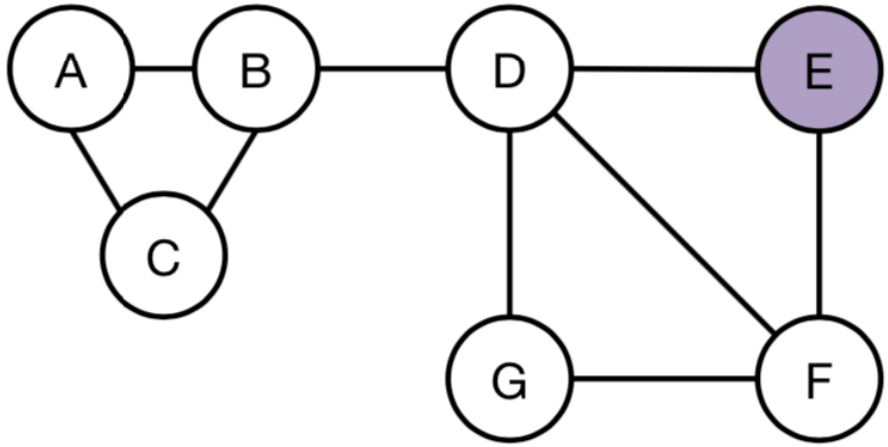


Computing BFS Betweenness

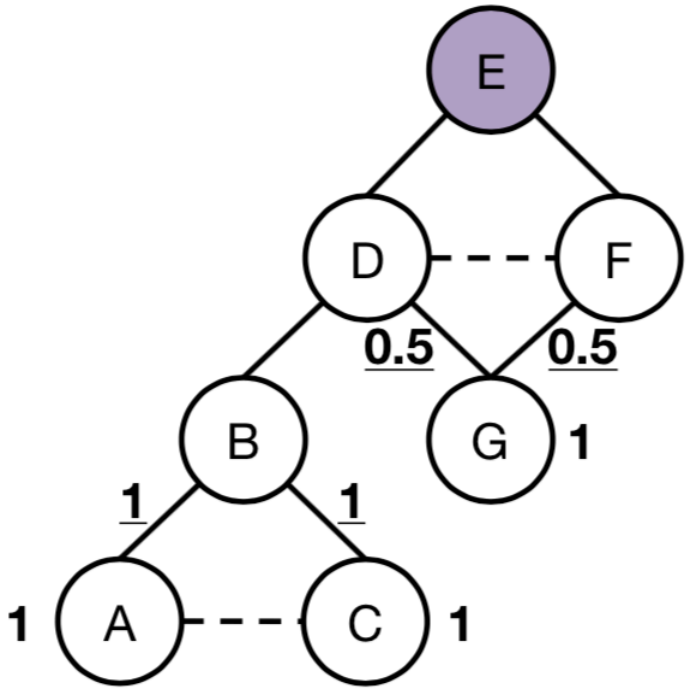
- Recursive credit definition
  - Node gets 1 + sum of edge credits below
  - Edge gets node below divided by outgoing
    - Ignore between nodes at the same level (never used for shortest paths)
- Start from leaves, move up

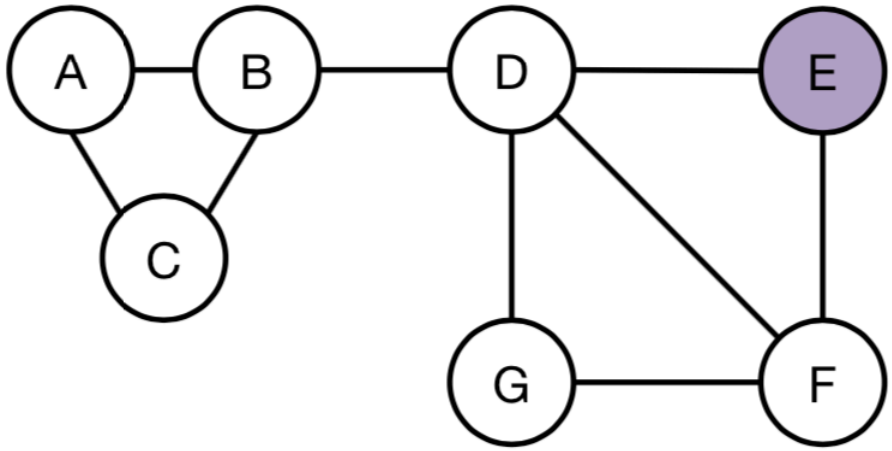
Example (E, Credits.1)



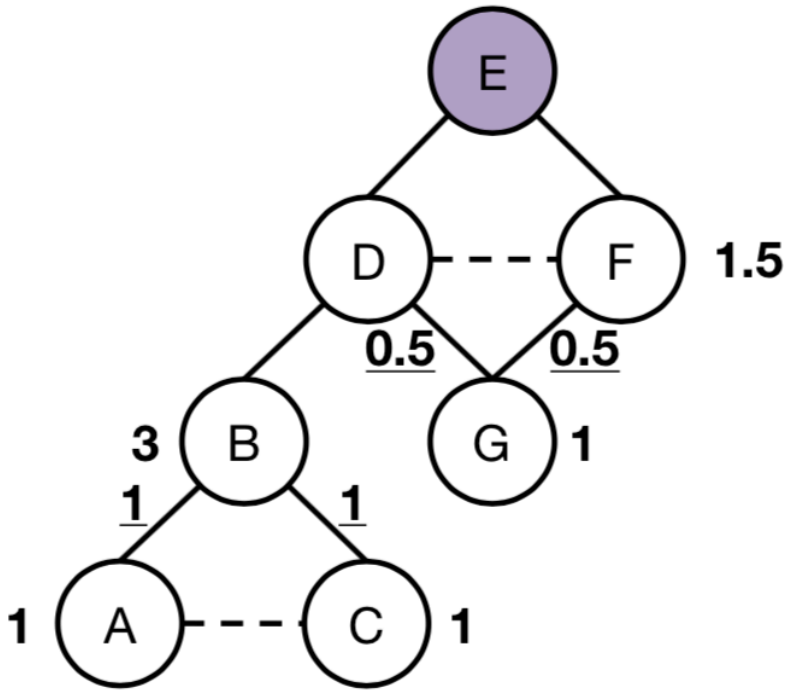


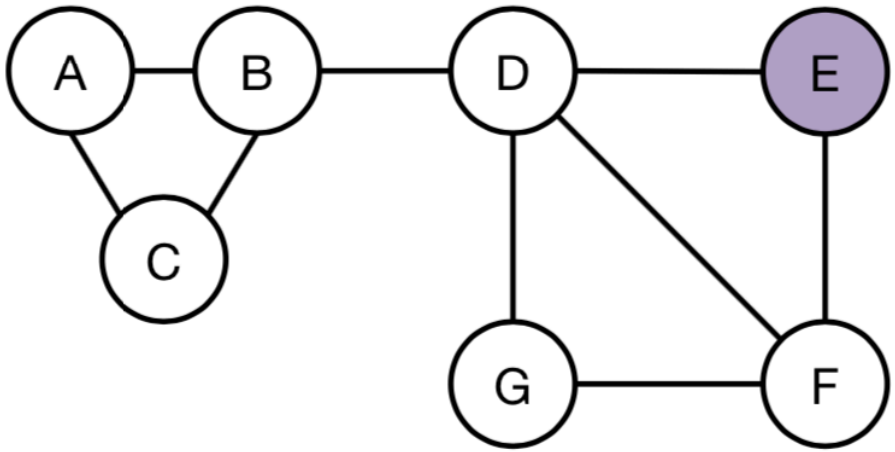
Example (E, Credits.2)



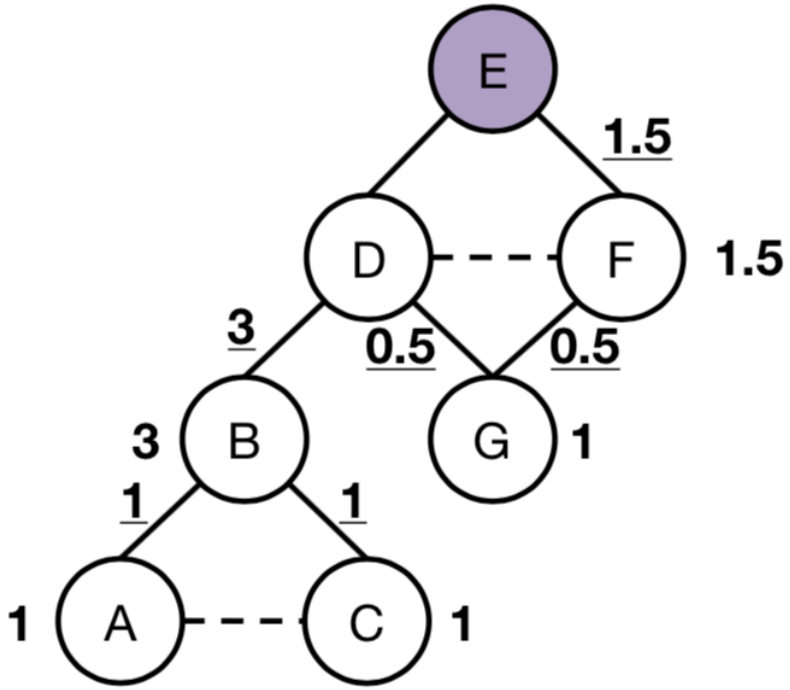


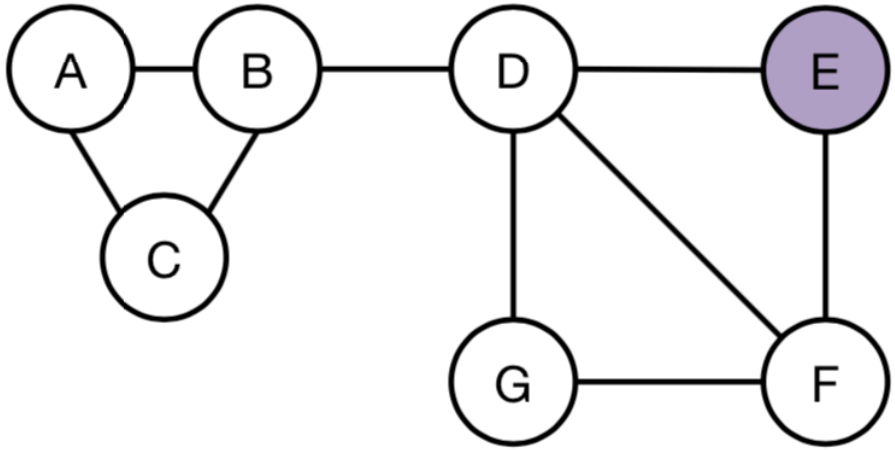
Example (E, Credits.3)



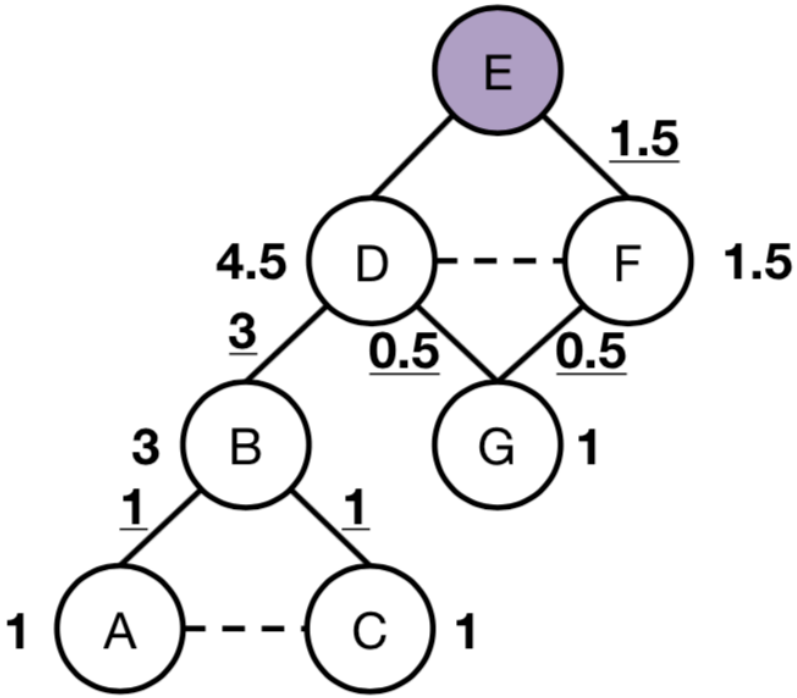


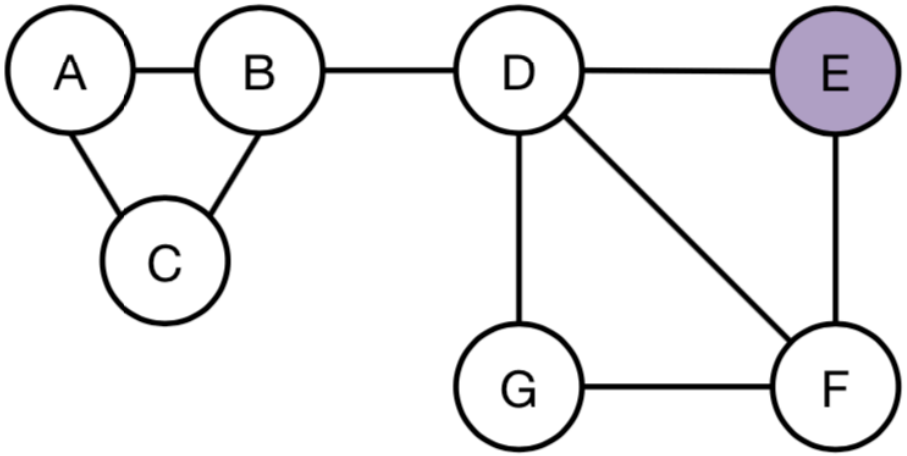
Example (E, Credits.4)



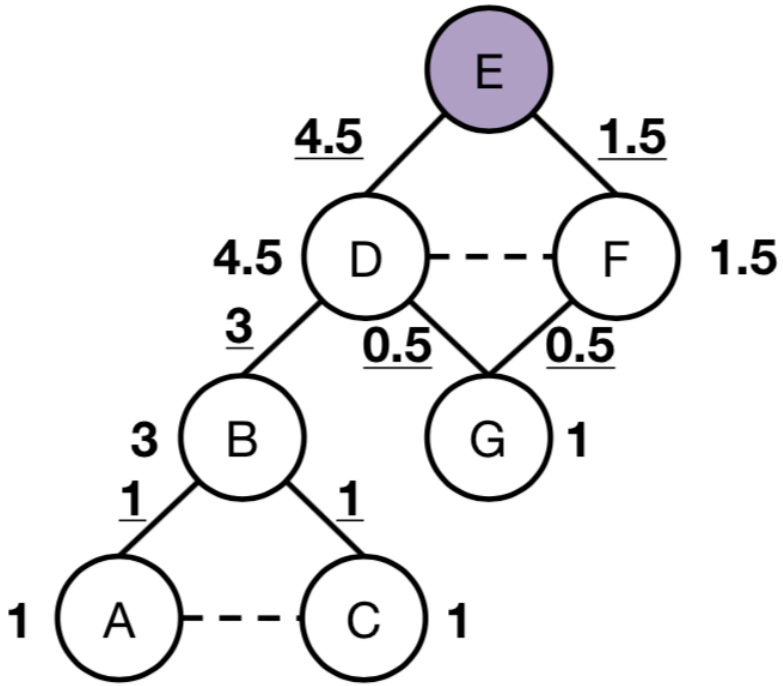


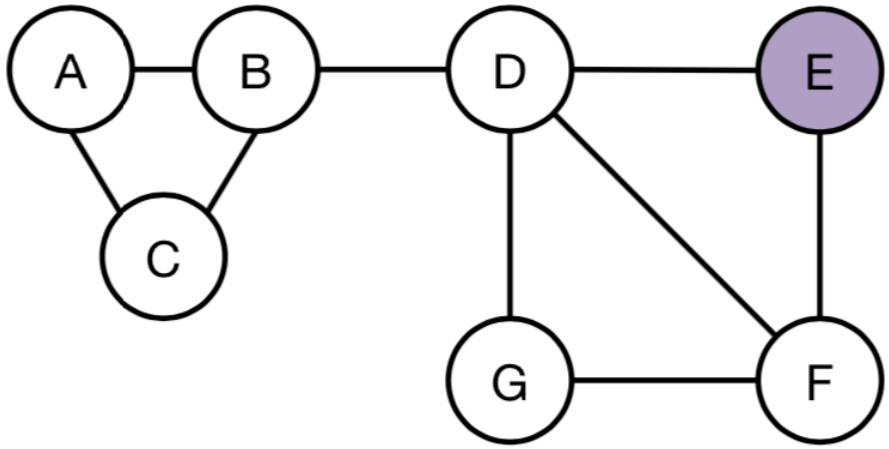
Example (E, Credits.5)



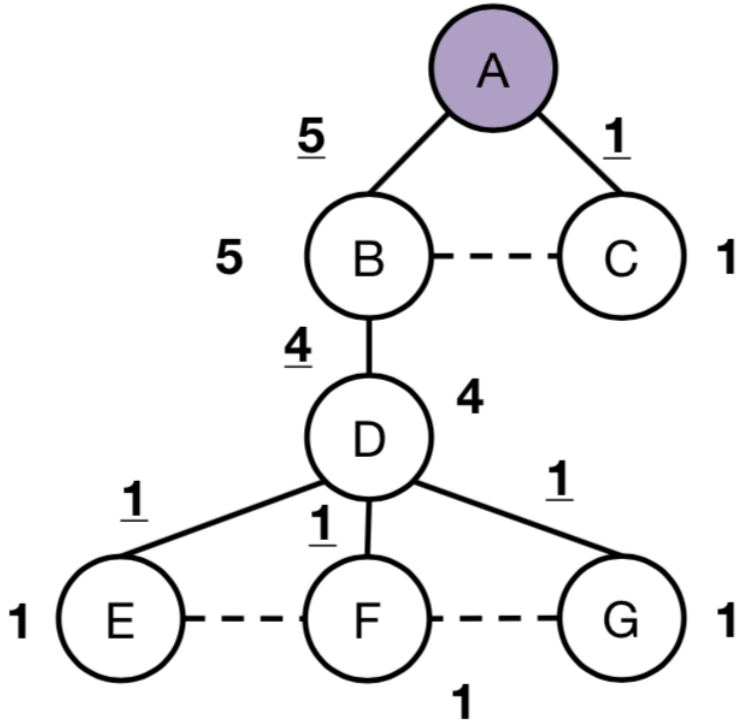


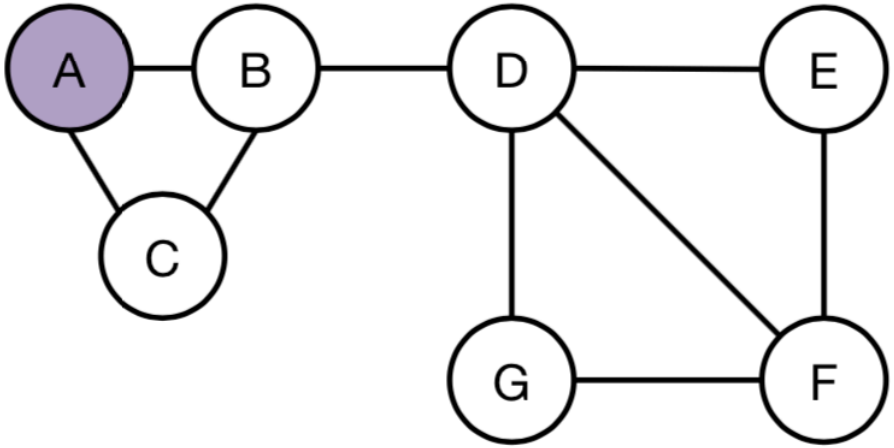
Example (E, Credits.6)



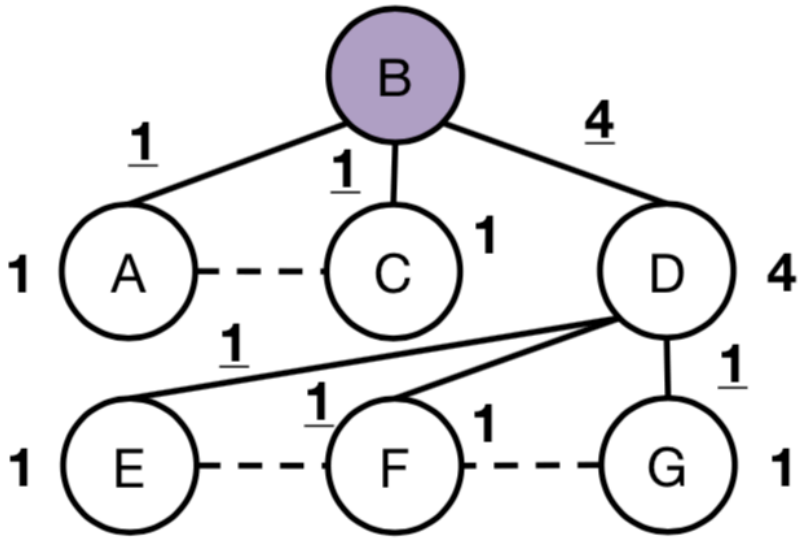


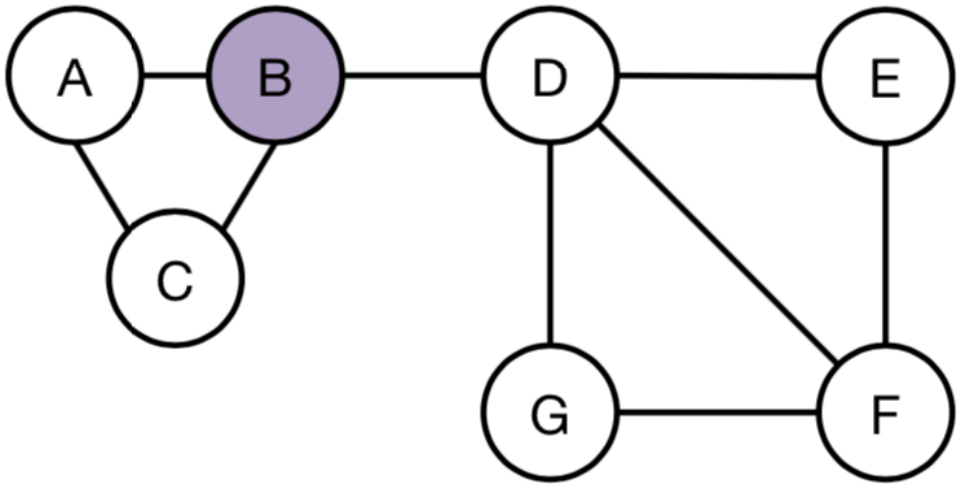
Example (A, Credits)



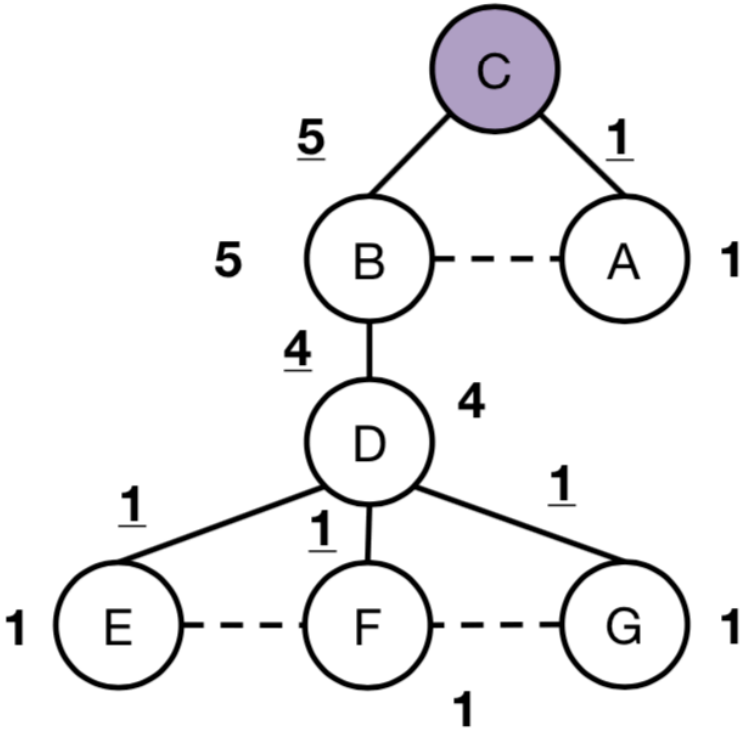


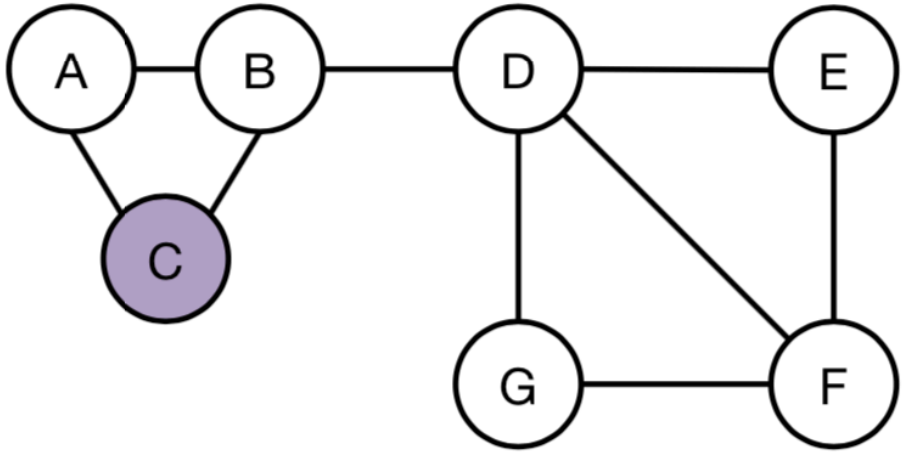
Example (B, Credits)



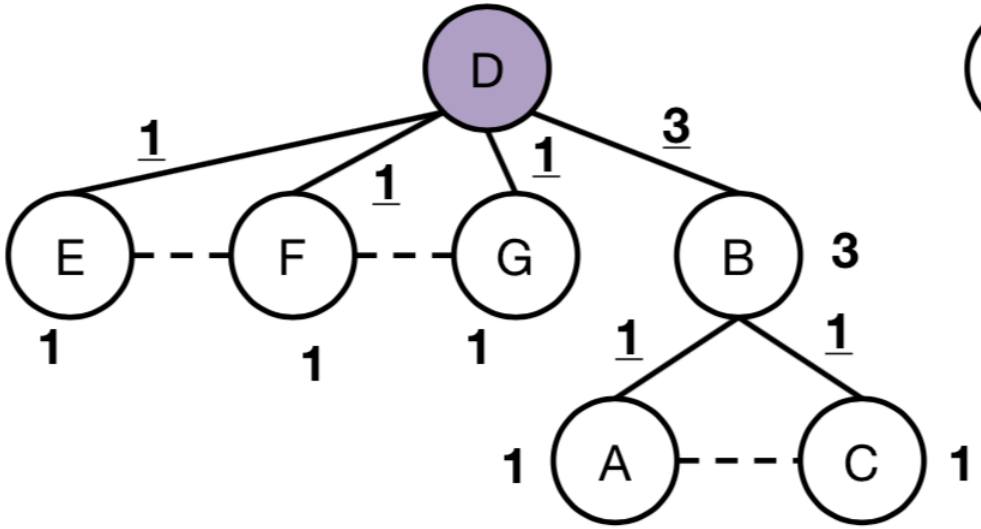


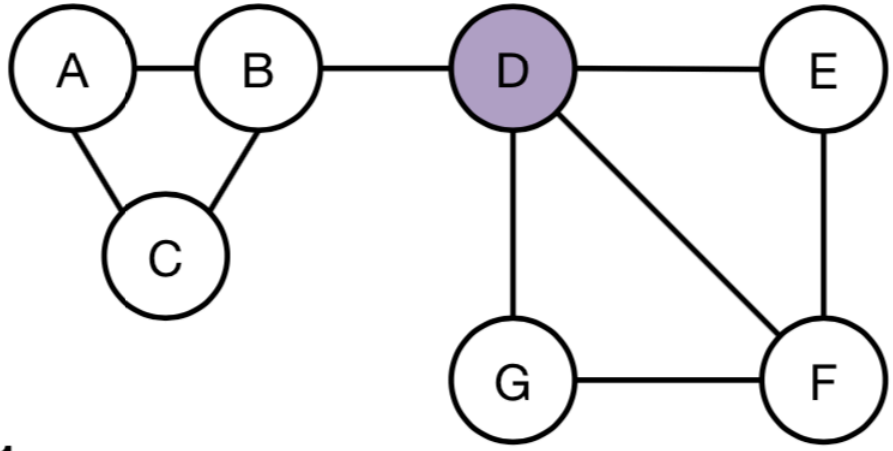
Example (C, Credits)



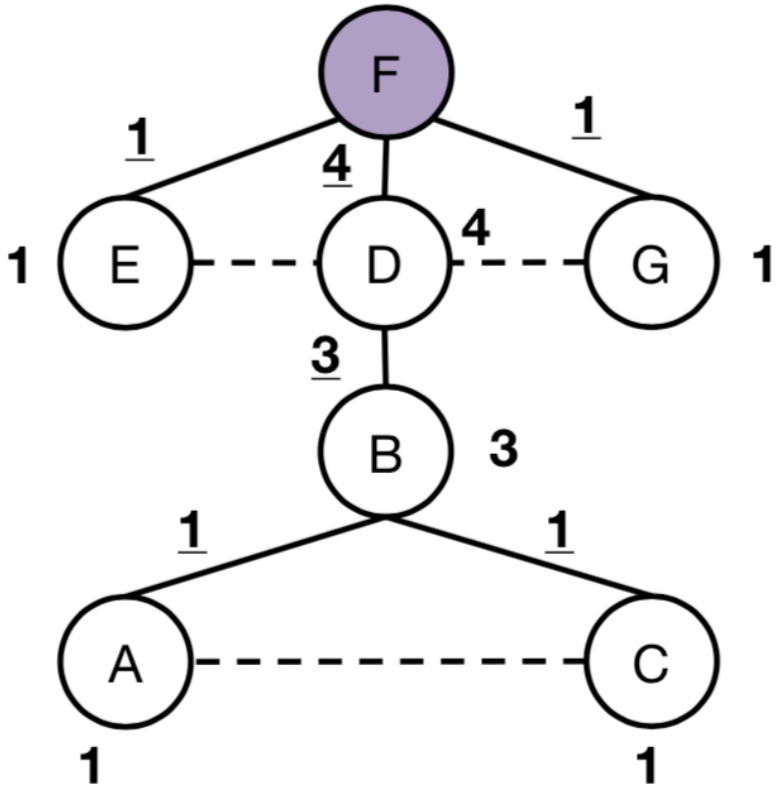


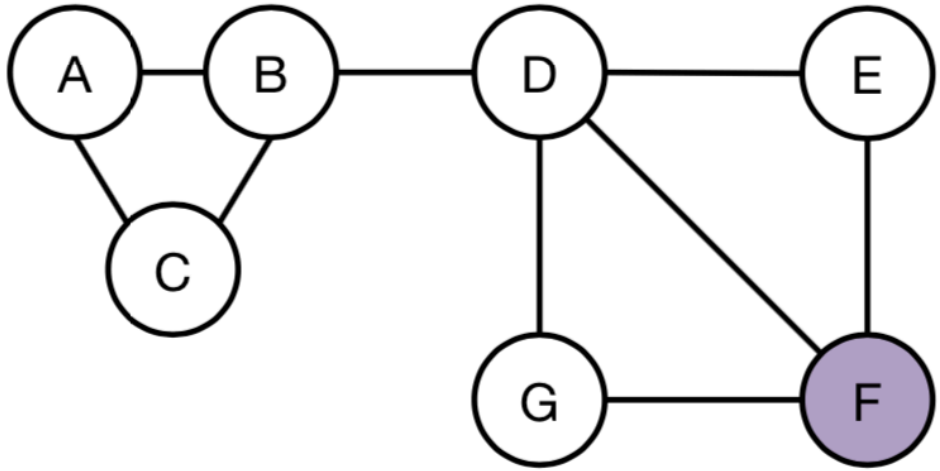
Example (D, Credits)



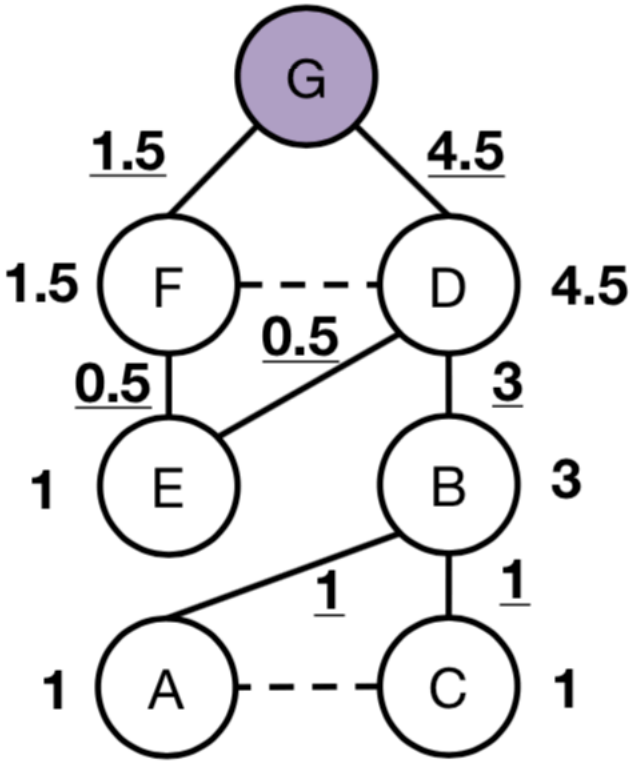


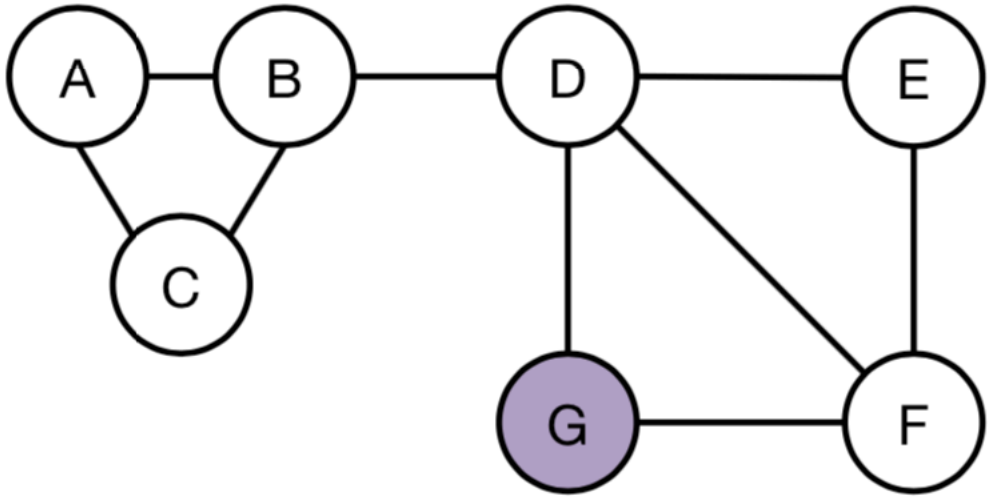
Example (F, Credits)





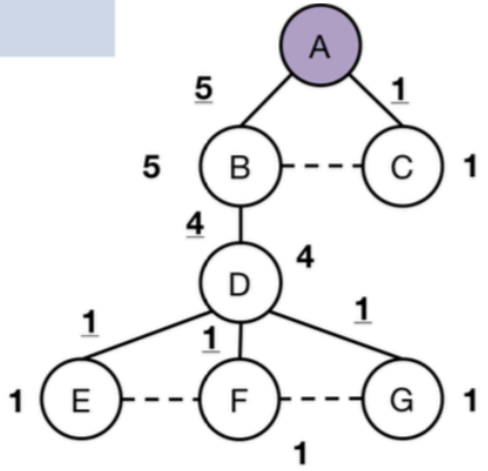
Example (G, Credits)





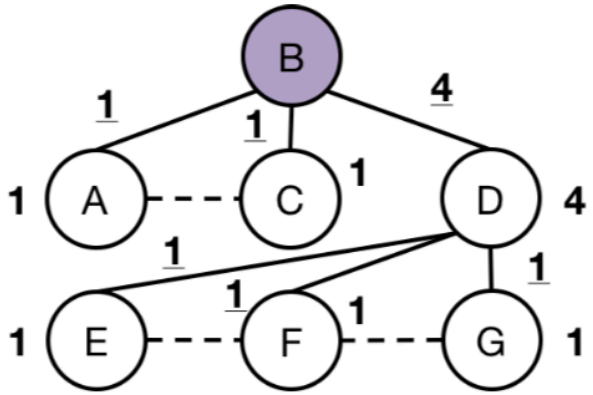
Sum Contributions (A)

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



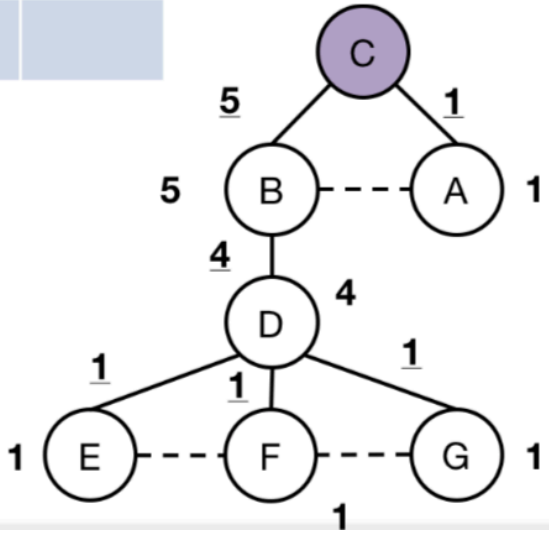
Sum Contributions (B)

	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									



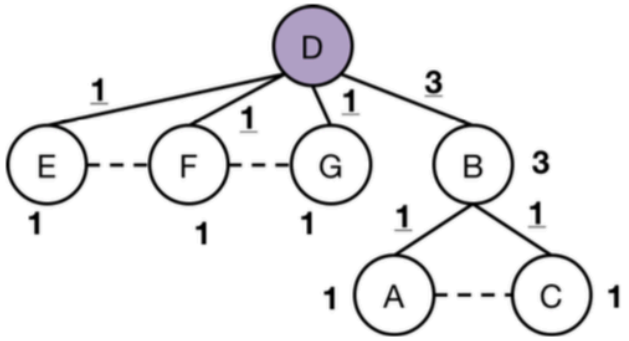
Sum Contributions (C)

	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									



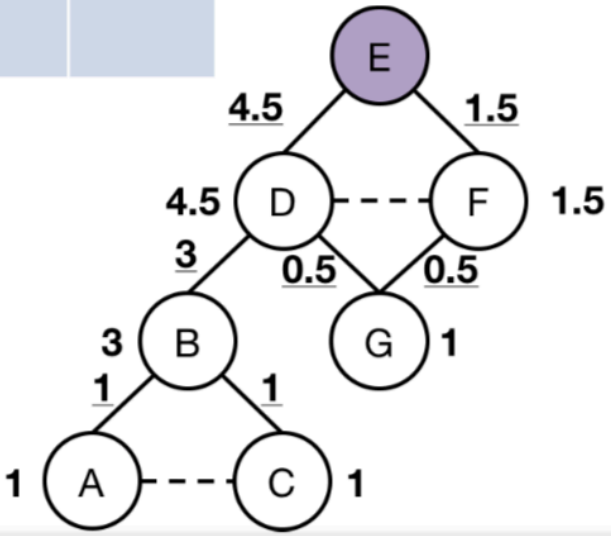
Sum Contributions (D)

	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									



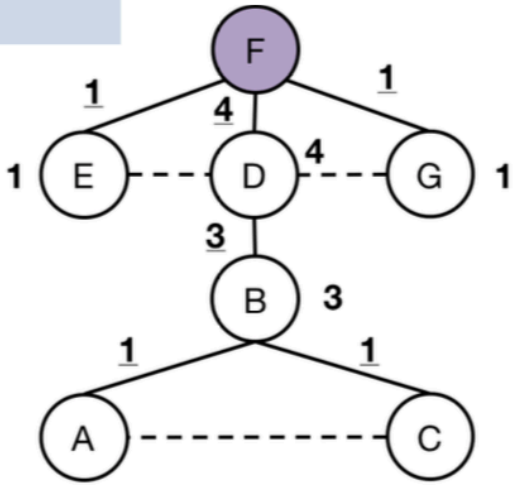
Sum Contributions (E)

	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									



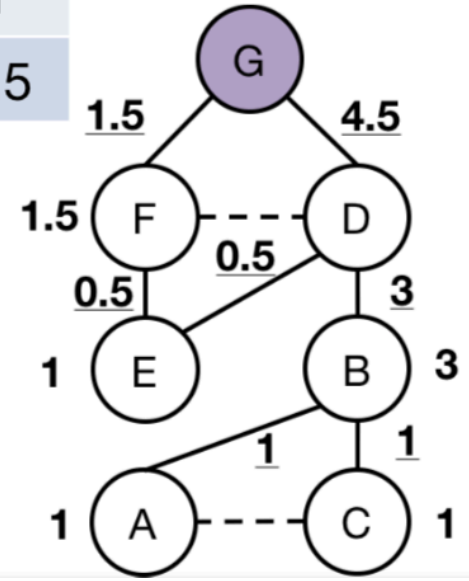
Sum Contributions (F)

	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)

	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



Sum Contributions (+)

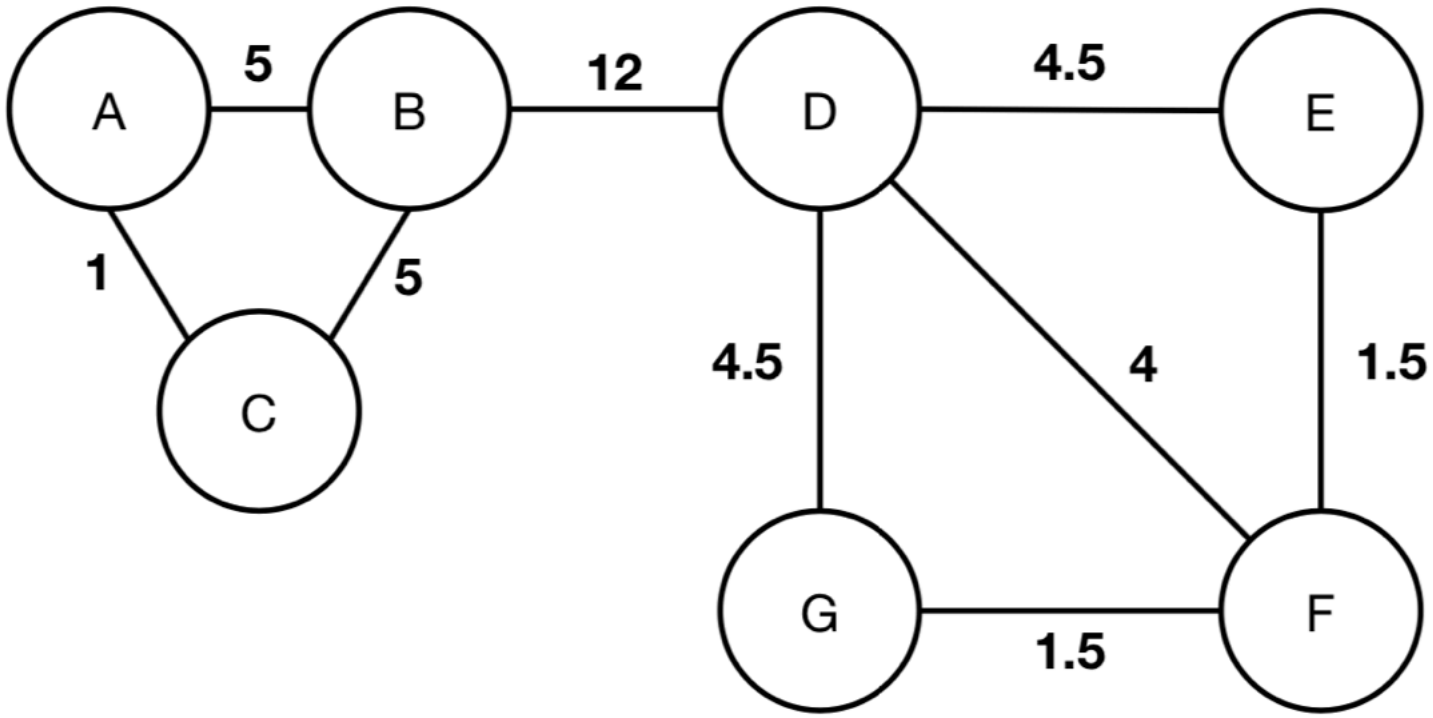
	AB	AC	BC	BD	DE	DG	DF	EF	GF
A	5	1		4	1	1	1		
B	1		1	4	1	1	1		
C		1	5	4	1	1	1		
D	1		1	3	1	1	1		
E	1		1	3	4.5	0.5		1.5	0.5
F	1		1	3			4	1	1
G	1		1	3	0.5	4.5		0.5	1.5
+	10	2	10	24	9	9	8	3	3

Sum Contributions (/2)

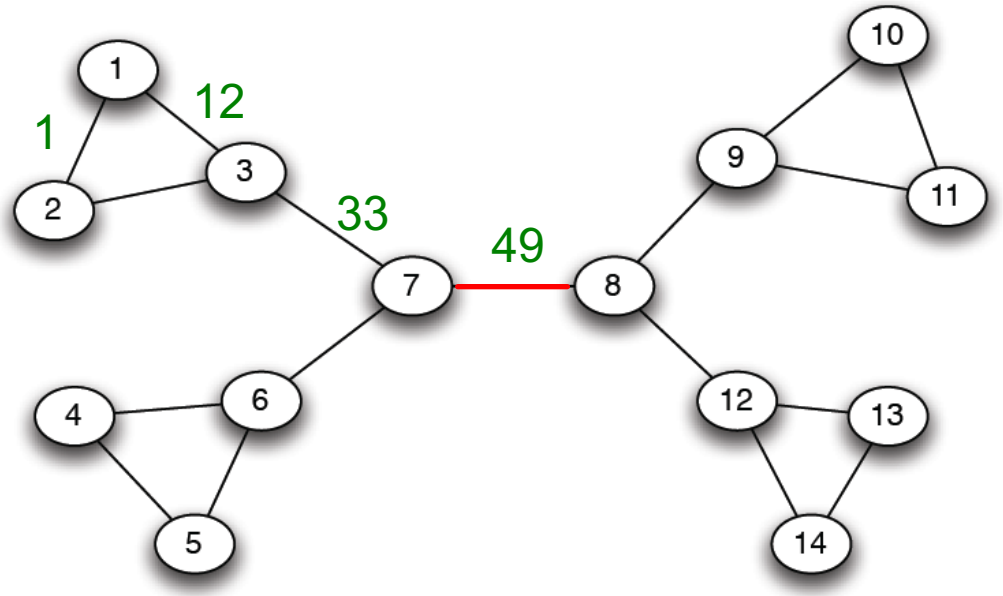
	AB	AC	BC	BD	DE	DG	DF	EF	GF
A	5	1		4	1	1	1		
B	1		1	4	1	1	1		
C		1	5	4	1	1	1		
D	1		1	3	1	1	1		
E	1		1	3	4.5	0.5		1.5	0.5
F	1		1	3			4	1	1
G	1		1	3	0.5	4.5		0.5	1.5
+	10	2	10	24	9	9	8	3	3
/2	5	1	5	12	4.5	4.5	4	1.5	1.5

Betweenness: Final Result

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

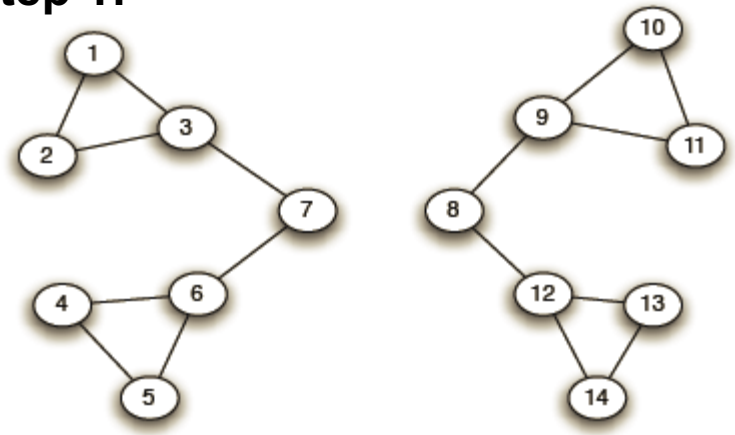


Girvan-Newman: Another Example

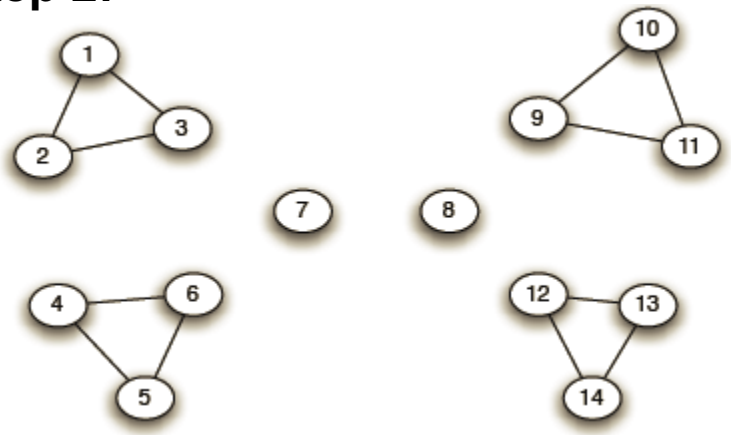


Girvan-Newman: Example

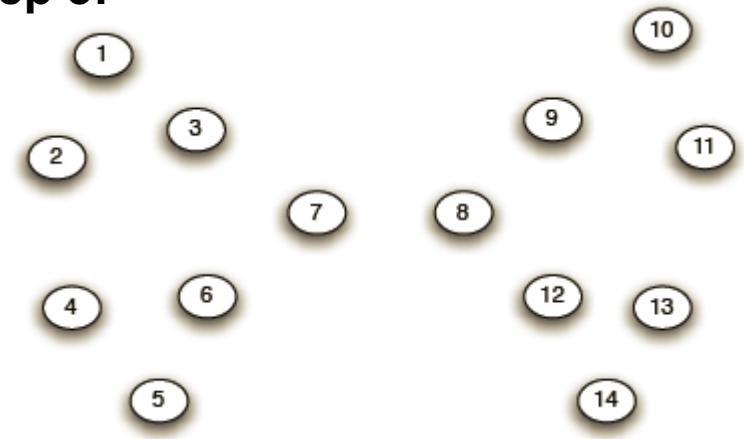
Step 1:



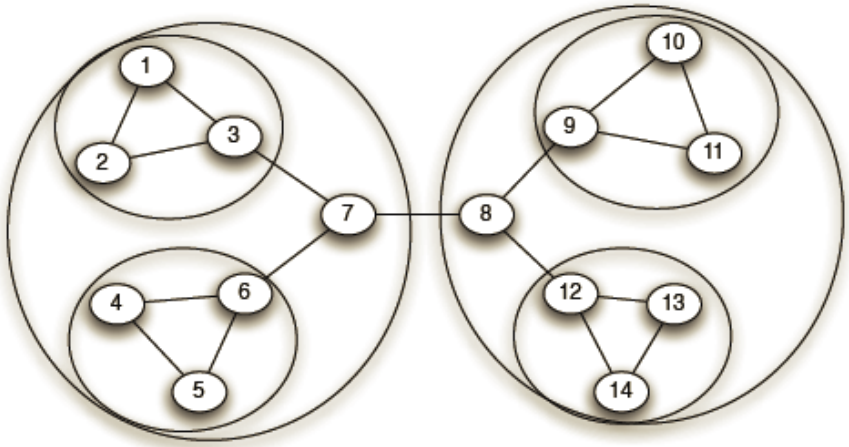
Step 2:



Step 3:



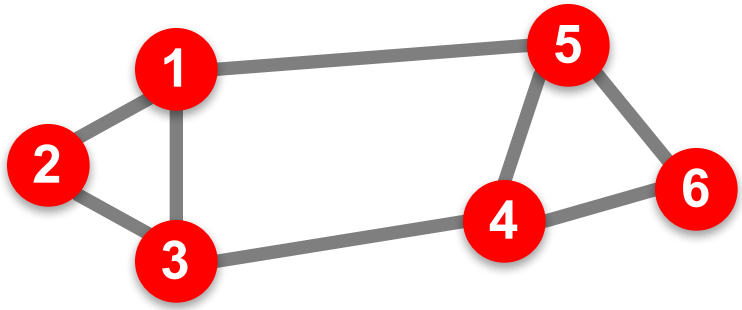
Hierarchical network decomposition:



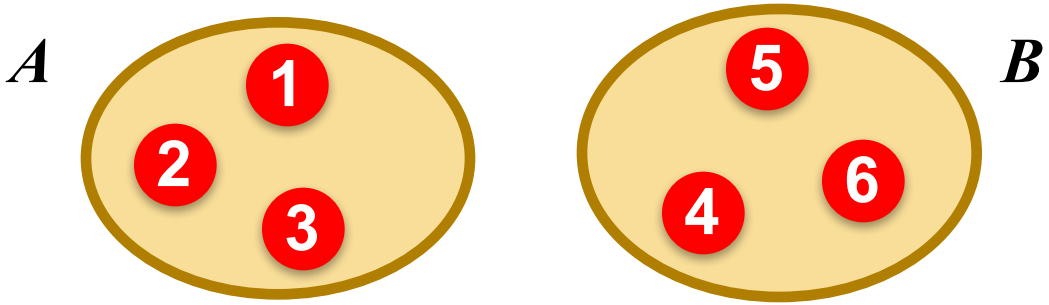
Spectral Clustering for Graph Partitioning (10.4)

Graph Partitioning

- **Undirected graph $G(V, E)$:**



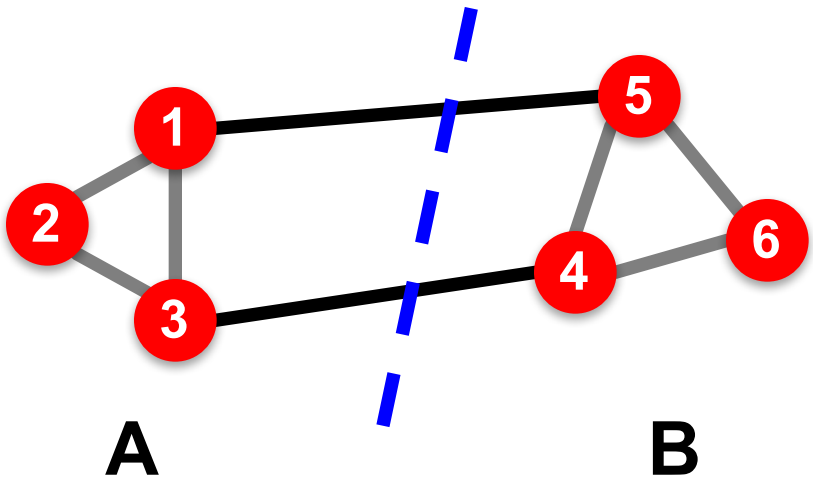
- **Bi-partitioning task:**
  - Divide vertices into two disjoint groups A, B



- **Questions:**
  - How can we define a "good" partition of G?
  - How can we efficiently identify such a partition?

Graph Partitioning

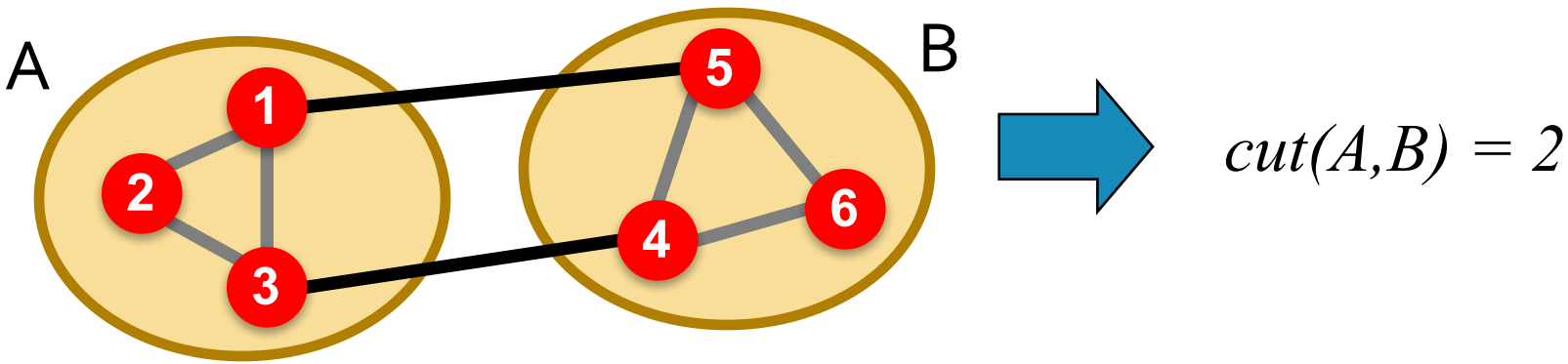
- **What makes a good partition?**
  - Maximize the number of within-group connections
  - Minimize the number of between-group connections



Graph Cuts

- **Express partitioning objectives as a function of the "edge cut" of the partition**
- **Cut:** Set of edges with only one vertex in a group:

$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij}$$

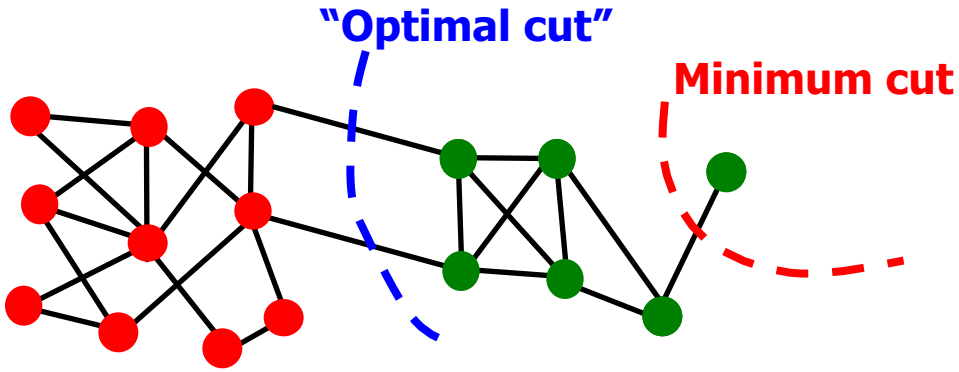


Graph Cut Criterion

- **Criterion: Minimum-cut**
  - Minimize weight of connections between groups

$$\arg\min_{A,B} cut(A,B)$$

- **Degenerate case:**



- **Problem:**
  - Only considers external cluster connections
  - Does not consider internal cluster connectivity

[Shi-Malik]

Graph Cut Criteria

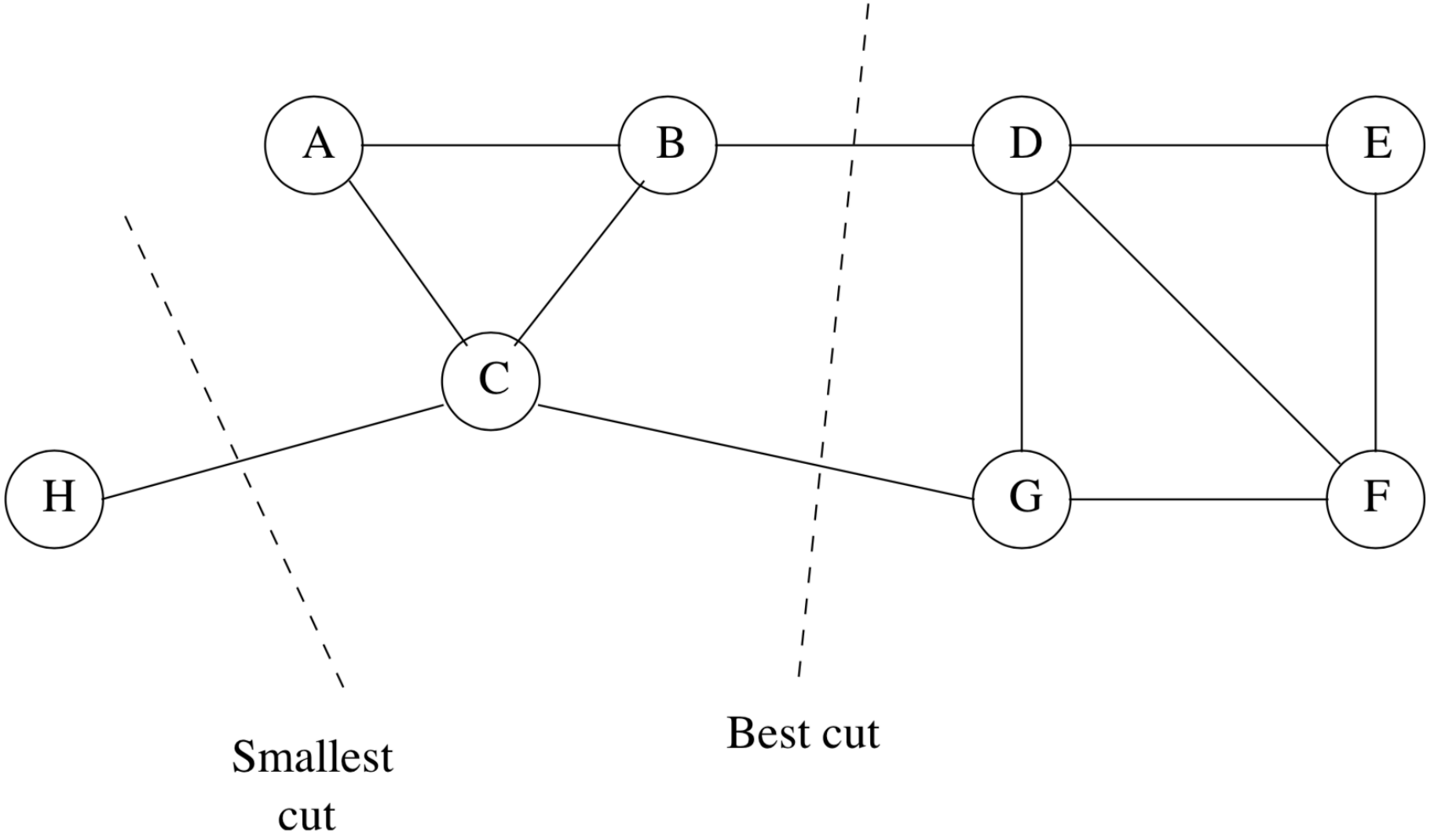
- **Criterion: Normalized-cut** [Shi-Malik, '97]
  - Connectivity between groups relative to the density of each group

$$ncut(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(A,B)}{vol(B)}$$

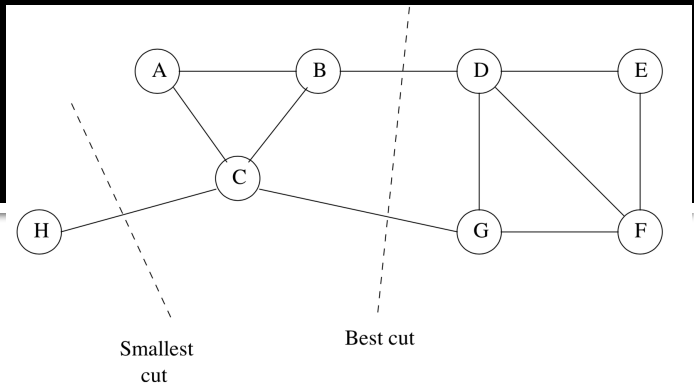
$\boldsymbol{vol(A)}$: total number/weight of the edges with at least one endpoint in $\boldsymbol{A}$: $\boldsymbol{vol(A) = \sum_{i \in A} k_i}$

- **Why use this criterion?**
  - Produces more balanced partitions

Normalized Cut: An Example

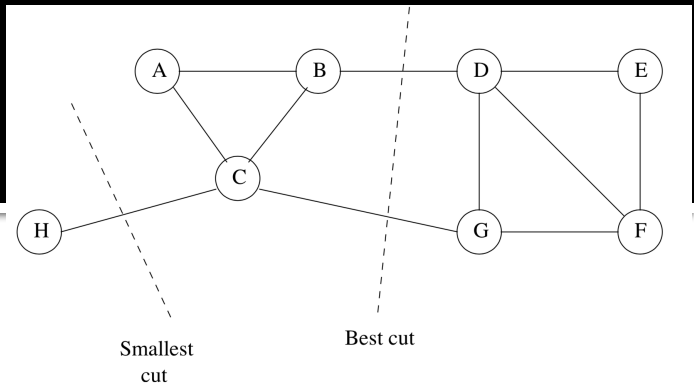


The Smallest Cut



- S = {H} and T = {A,B,C,D,E,F,G}
- Cut(S,T) = 1
- Vol(S) = 1, because there is only one edge connected to H.
- Vol(T) = 11, because all the edges have at least one end at a node of T.
- The normalized cut for this partition is 1/1 + 1/11 = 1.09.

The Best Cut



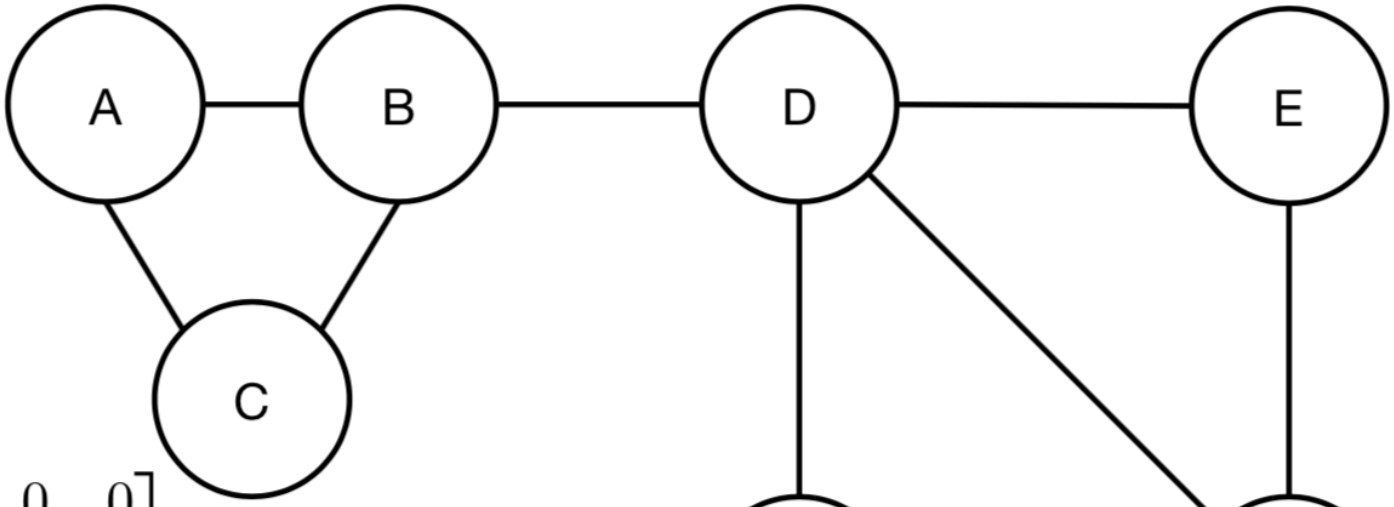
- S = {A,B,C,H} and T = {D,E,F,G}
- Cut(S,T) = 2
- Vol(S) = 6
- Vol(T) = 7
- The normalized cut for this partition is only 2/6 + 2/7 = 0.62.

Spectral Clustering

- Need three matrices
- Adjacency matrix *A*
- Degree matrix *D*
- Laplacian matrix *L*

Adjacency Matrix (A)

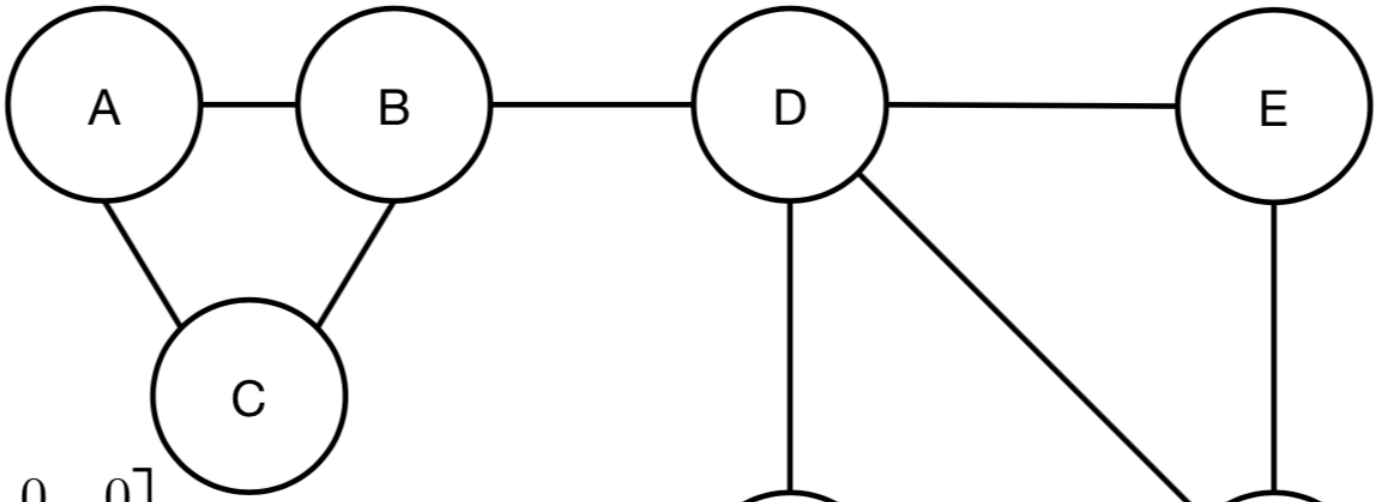
A_{ij} =1 if (i,j) is an edge, else 0



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

Degree Matrix (D)

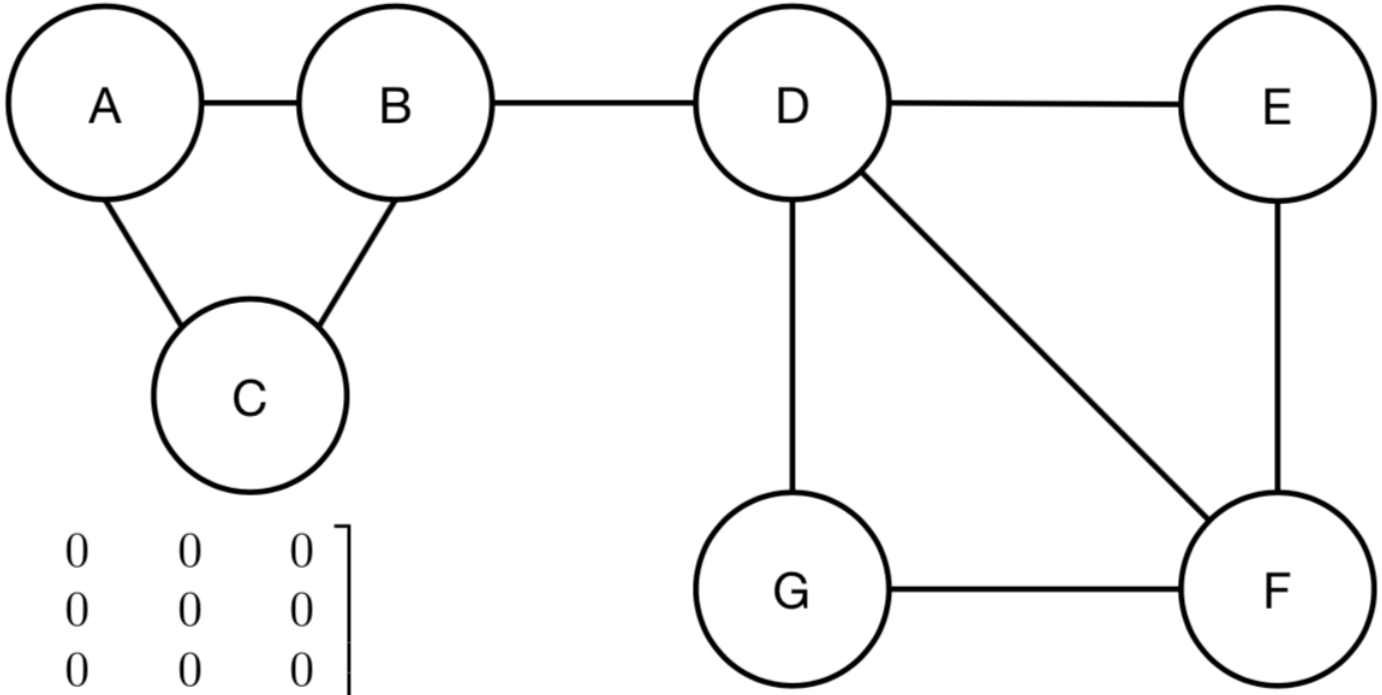
$$D_{ii} = deg(i), \text{ else } 0$$



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

Laplacian Matrix (L)

$$L = D - A$$



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

Laplacian Matrix: Findings

- 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)

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

Eigenvectors/Eigenvalues of *L*

- 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 (positive semi-definite), so $\lambda \geq 0$
    – when $\lambda = 0$: connectivity
    – when λ is the second smallest: partitioning

When λ=0

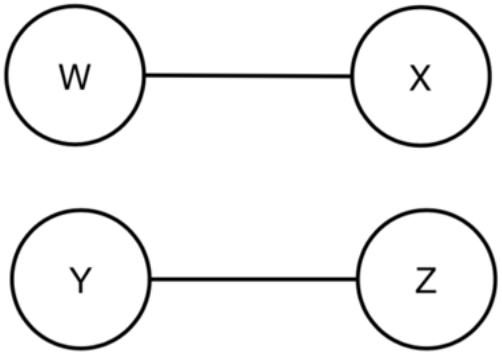
- What is a non-trivial solution to the equation Lv = λv for λ=0 (i.e. Lv=0)?

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

Solution: 1's (column vector of constants) or any constant *c*

Any other solutions?

When λ=0



What is the Laplacian matrix of this given graph?

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

What is the eigenvector when λ = 0?

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

This is called Null-space of *L*

Connectivity via o^{th} 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^{th} eigenvector is $[c\ c\ c\ \ldots\ ]^T$
- Otherwise, null-space of L provides connected components

Second-Smallest Eigenvalue of *L*

- Let's call this λ_1 (with corresponding eigenvetor v_1)
- If λ_1 is 0,what do we know?
  - That there's not much sense in bi-partitioning :)
- From here, we'll construct an objective function for a bi-partitioning, and it will turn out to be minimized precisely by v_1

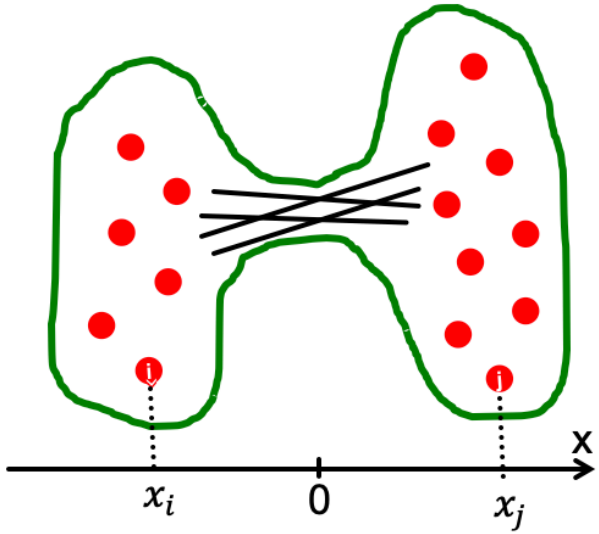
Set Up Objective Function

- 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 x_i
  - So, a vector, *X*, 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...

  $(x_i - x_j)^2$ for all edges (i, j)

Why This Objective Function Works?

- What is $(x_i - x_j)^2$ 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!



How does this relate to *L*?

- For the previously mentioned vector *x*, we have:
  - $\mathrm{x}^{\mathrm{T}} \mathrm{L}\, \mathrm{x} = \sum_{i,j=1}^{n} L_{ij}\, x_i x_j = \sum_{i,j=1}^{n} \left(D_{ij} - A_{ij}\right) x_i x_j$
  - $= \sum_i D_{ii} x_i^2 - \sum_{(i,j)\in E} 2 x_i x_j$
  - $= \sum_{(i,j)\in E} (x_i^2 + x_j^2 - 2 x_i x_j)$
  - $= \sum_{(i,j)\in E} \left(x_i - x_j\right)^2$ ⇐ Objective function!

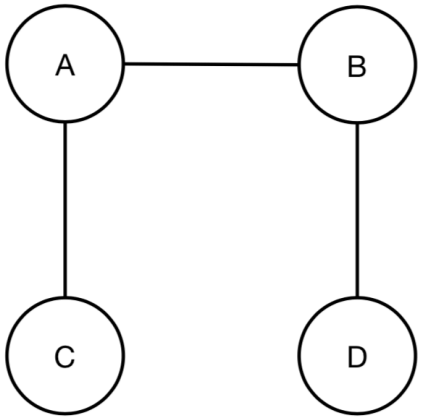
Node i has degree d_i. So, value x_i^2 needs to be summed up d_i times.
But each edge (i,j) has two endpoints so we need $x_i^2 + x_j^2$

Ready to Find Solutions

$$x^T L x = \sum_{(i,j) \in E} (x_i - x_j)^2$$

- To minimize the right-hand side, we just need to minimize the left-hand side
- What is the left-hand side?
  - Eigenvalue of *L*
  - So the second smallest eigenvalue is our solution!

Example 1



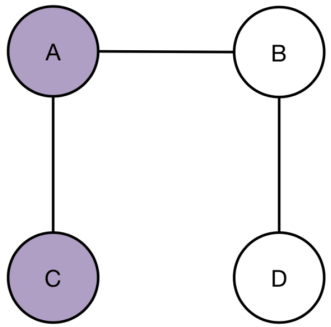
Laplacian

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

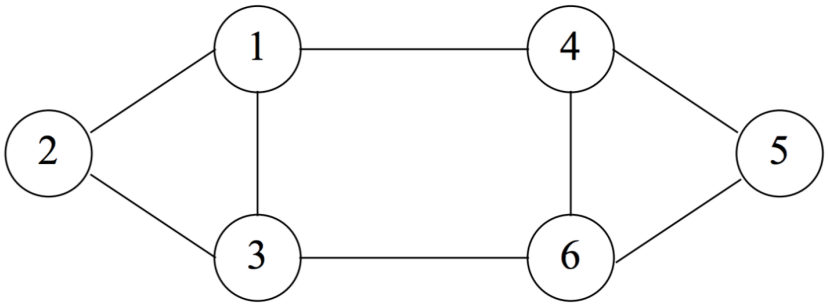
```
In [45]: np.linalg.eigh(A5)

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}

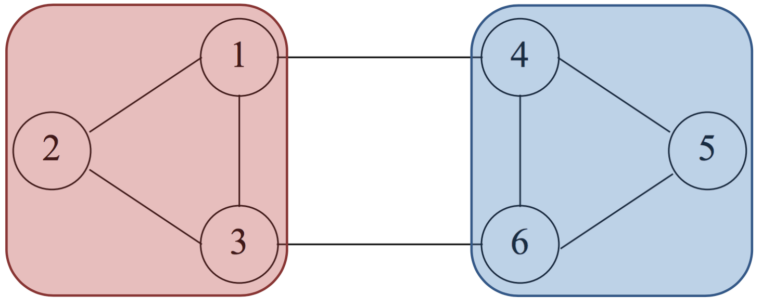


Example 2



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

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

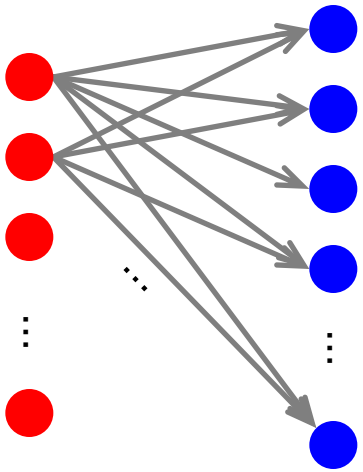


Search for Small Communities (10.3)

[Kumar et al. '99]

Trawling

- **Searching for small communities in the Web graph**
- **What is the signature of a community / discussion in a Web graph?**



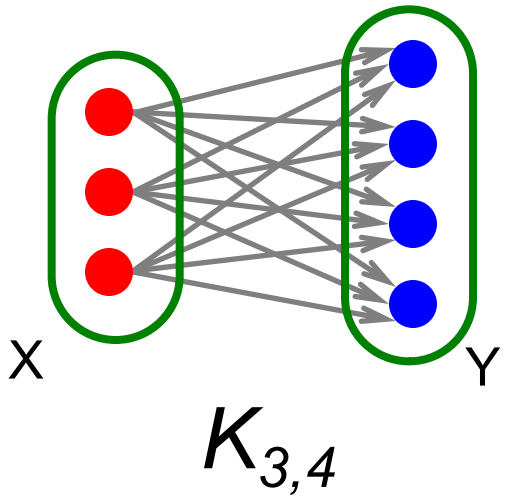
Dense 2-layer graph

Use this to define "topics":
What the same people on the left talk about on the right
Remember HITS!

Intuition: Many people all talking about the same things

Searching for Small Communities

- **A more well-defined problem:**
  Enumerate complete bipartite subgraphs $K_{s,t}$
  - Where $K_{s,t}$: s nodes on the "left" where each links to the same t other nodes on the "right"



$K_{3,4}$

|X| = s = 3
|Y| = t = 4

Fully connected

[Agrawal-Srikant '99]

Market Basket Analysis: Frequent Items

- itemset: A set of one or more items
- k-itemset $Y = \{y_1, \ldots, y_k\}$
- *support*, or, *support count* of *Y*: Frequency or occurrence of an itemset *Y*
- An itemset *Y* is *frequent* if *Y*'s support is no less than a *minimum sup* threshold *s*

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

Let $s = 3$, frequent itemsets found:
{Beer}:3, {Nuts}:3, {Diaper}:4, {Eggs}:3, {Beer, Diaper}:3

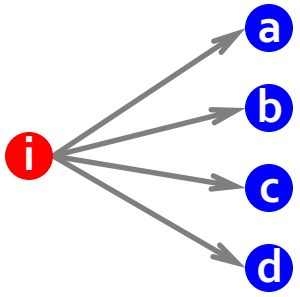
What's the connection between the itemsets and complete bipartite graphs?

[Kumar et al. '99]

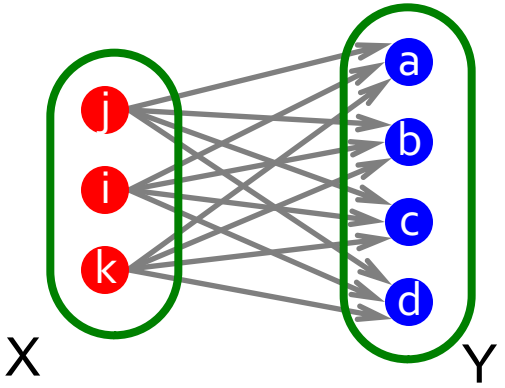
From Itemsets to Bipartite $K_{s,t}$

Frequent itemsets = complete bipartite graphs!

- **How?**
  - View each node on the left as a basket S_i.
  - View each node on the right as an item
  - $K_{s,t}$ = a set of t items that occurs in s buskets



S_i={a,b,c,d}

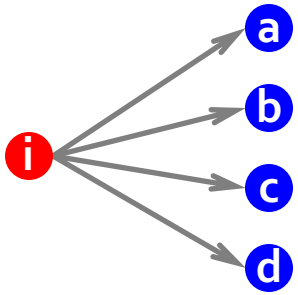


s … minimum support (|X|=s)
t … itemset size (|Y|=t)

[Kumar et al. '99]

From Itemsets to Bipartite $K_{s,t}$

View each node ***i*** as a set S_i of nodes ***i*** points to



S_i={a,b,c,d}

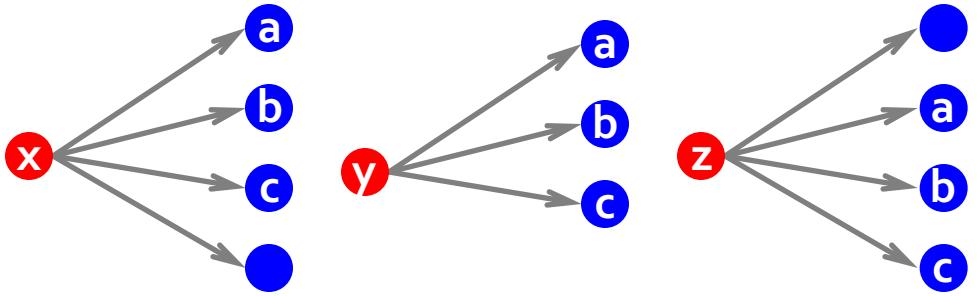
Find frequent itemsets:

s … minimum support

t … itemset size

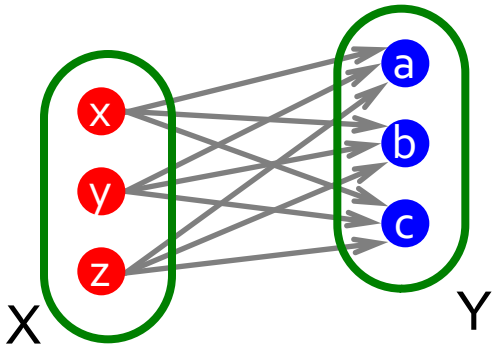
Say we find a **frequent itemset** ***Y={a,b,c}*** of supp ***s***

So, there are ***s*** nodes that link to all of **{a,b,c}**:

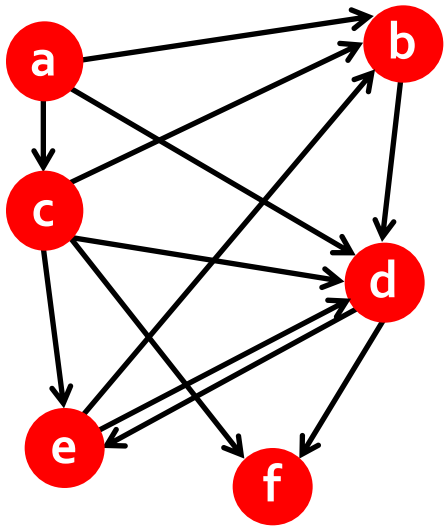


We found $K_{s,t}$!

$K_{s,t}$ = a set ***Y*** of size ***t*** that occurs in *s* sets S_i



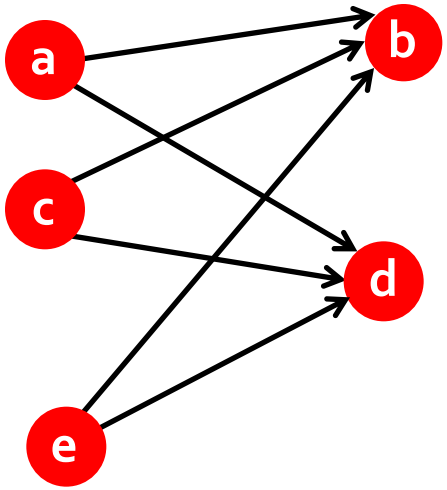
Example

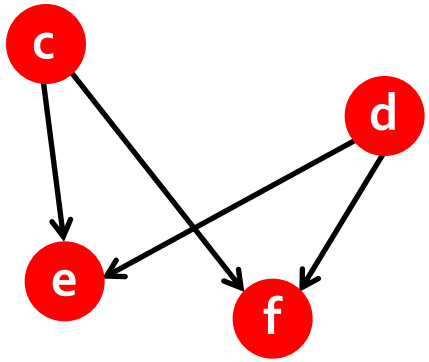


Itemsets:

a = {b,c,d}
b = {d}
c = {b,d,e,f}
d = {e,f}
e = {b,d}
f = {}

- **Support threshold s=2**
  - **{b,d}**: support 3
  - **{e,f}**: support 2
- **And we just found 2 bipartite subgraphs:**





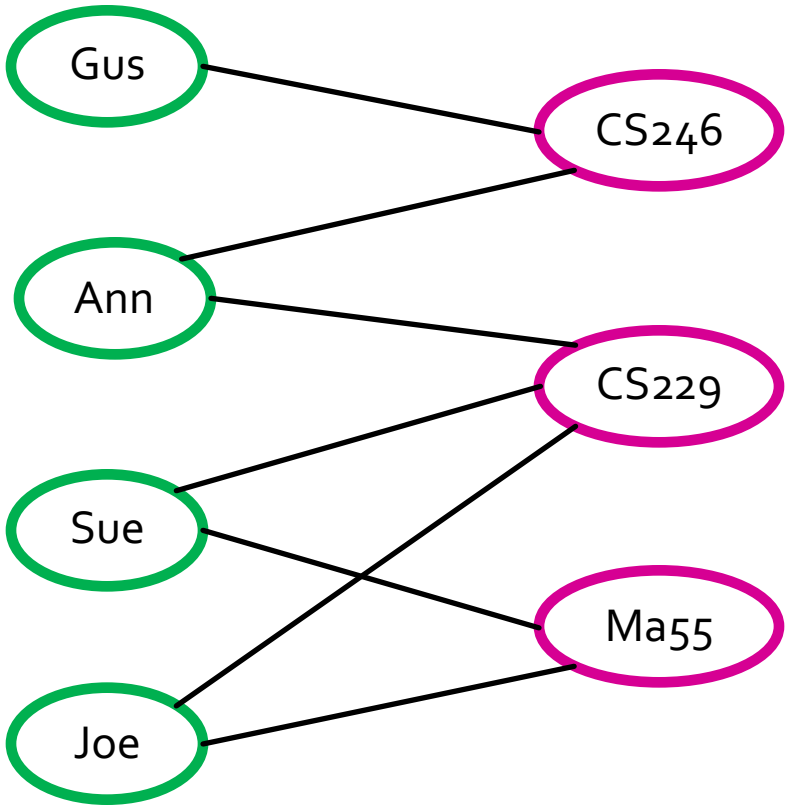
Finding Similar Entities by Random Walk (10.6)

Similarity in Networks

- Unlike similarity based on a distance measure, which we discussed with regard to LSH, we may instead wish to look for entities that play similar roles in a complex network.

- Example: Nodes represent students and classes; find students with similar interests, classes on similar subjects.

Example: Network

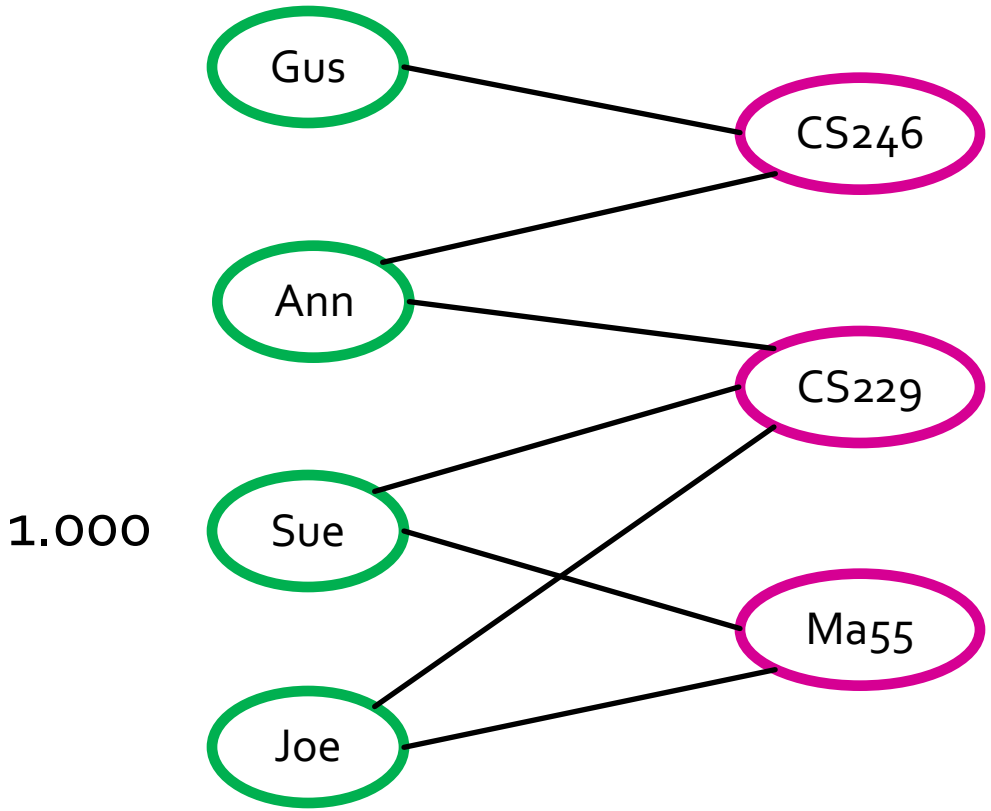


Solution: SimRank

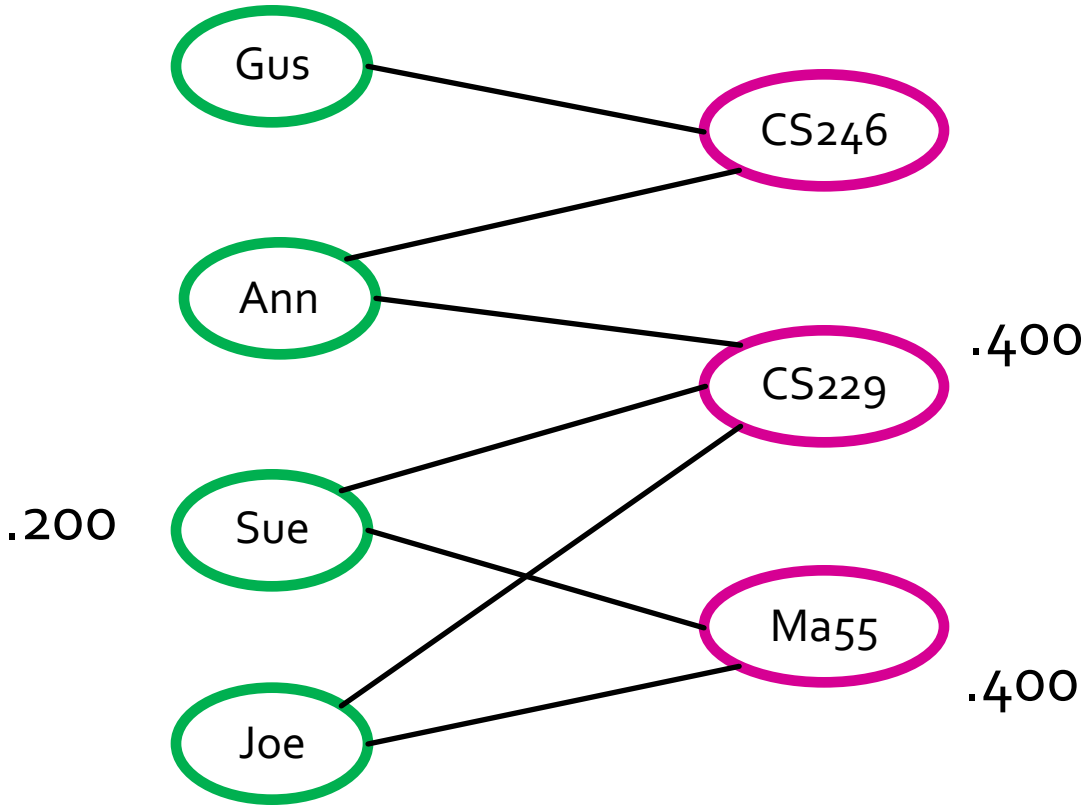
- Treat undirected edges as arcs or links in both directions.
- Goal: Find the entities similar to a single entity *S*
- Topic-specific PageRank idea:
  - We want to see where the random walker winds up on short walks from S
  - Modify the matrix of transition probabilities to have a small probability of transitioning back to S from any node, i.e. *S* becomes the only member of the teleport set.

Example: SimRank

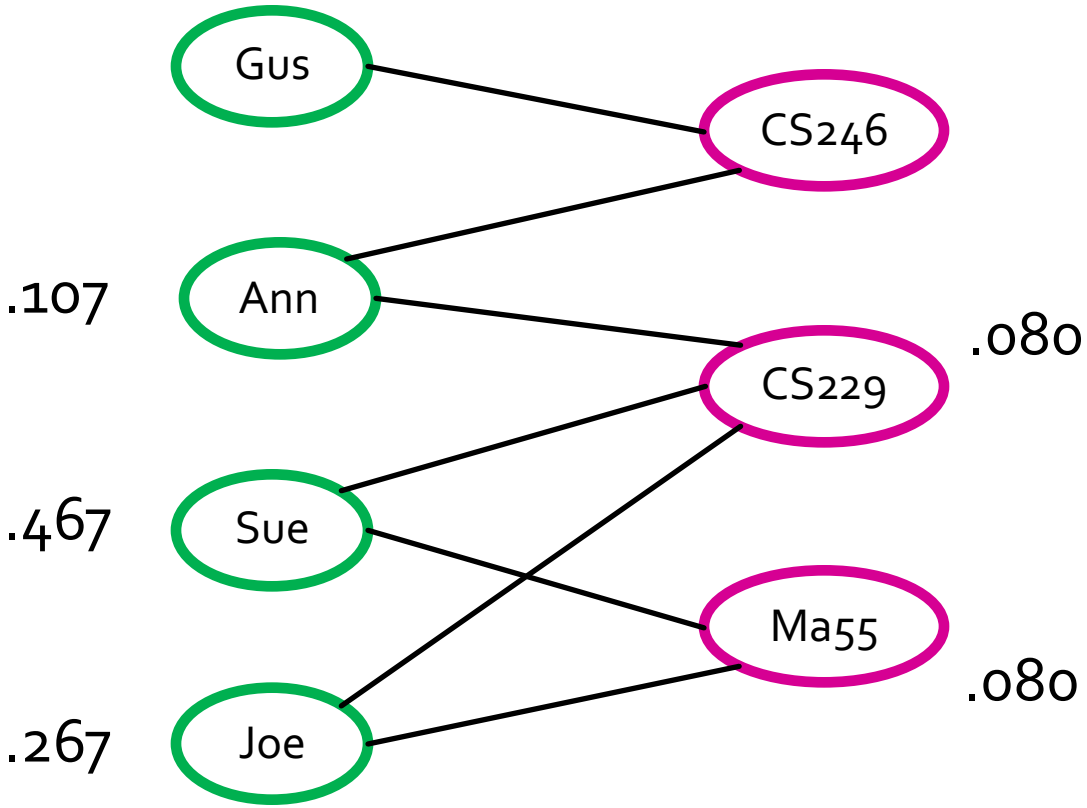
"Who is similar to Sue?"



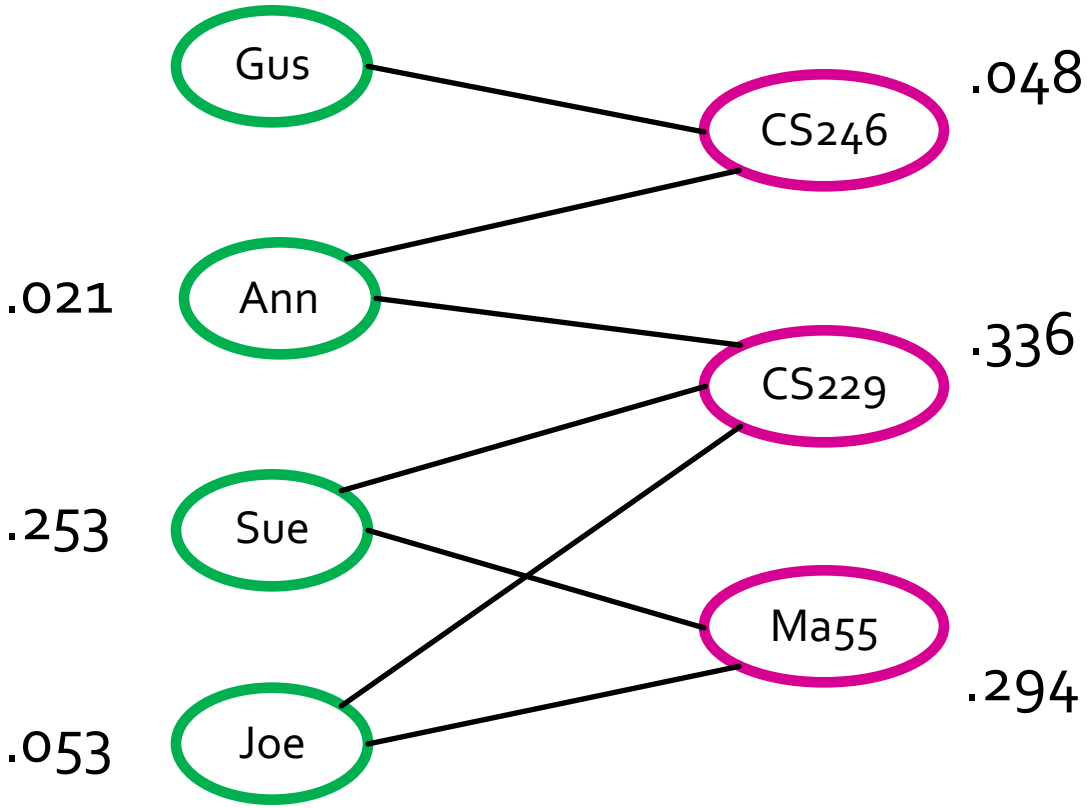
Example: SimRank (20% Tax)



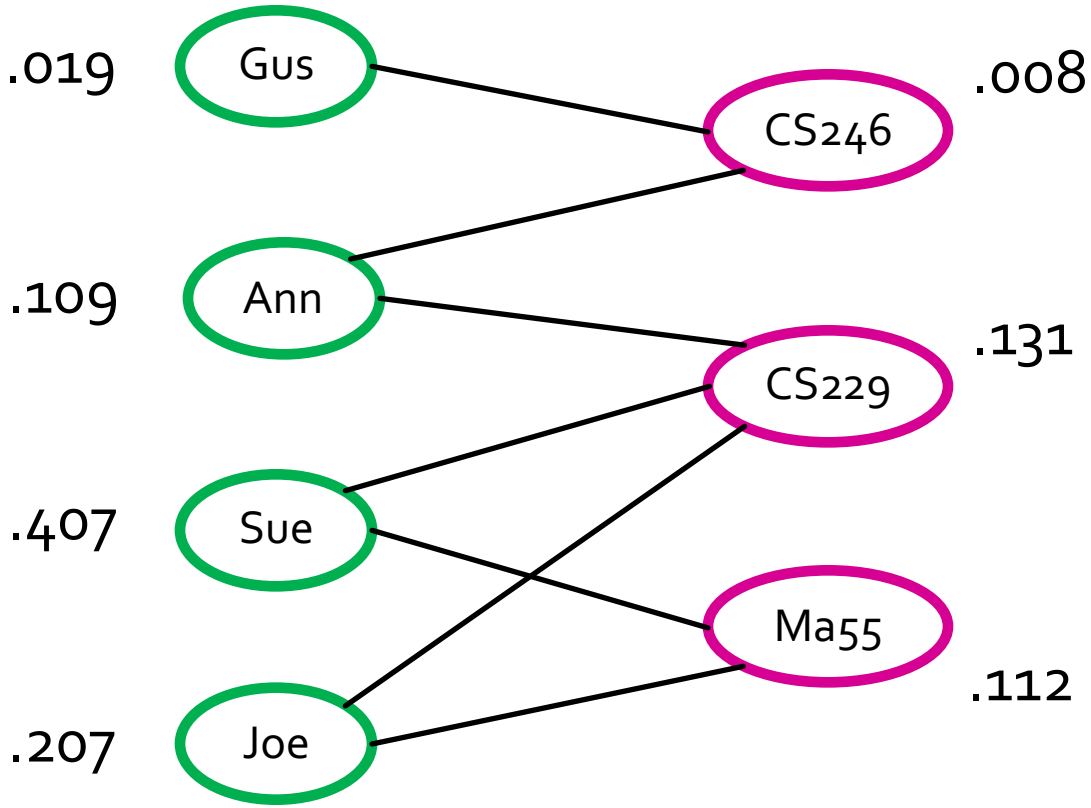
Example: SimRank



Example: SimRank



Example: SimRank



Social Graph Analysis: Summary

Summary

- Graphs to represent social networks
- How to identify communities?
  - Traditional distance measures do not work well
  - Need new definitions of distance/similarity
- Betweeness: number of shortest paths going through an edge (G-N algorithm)
- Cut and normalized cut (Laplacian matrix)
- Identify small communities (fully connected bi-partite subgraphs) using frequent itemset discovery
- Find similar entities using topic-specific PageRank