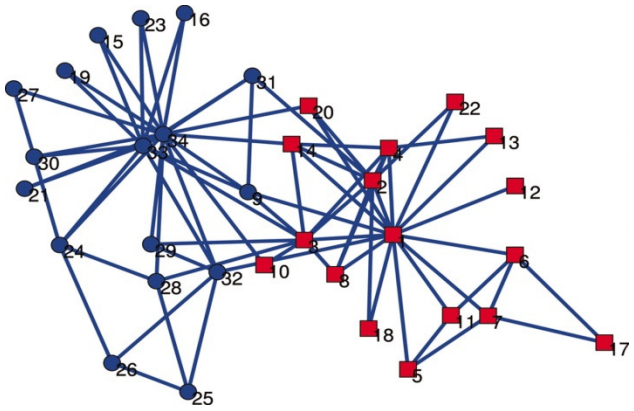
Community Detection Algorithms

Dr. Natarajan Meghanathan
Associate Professor of Computer Science
Jackson State University, Jackson, MS
E-mail: natarajan.meghanathan@jsums.edu

Community

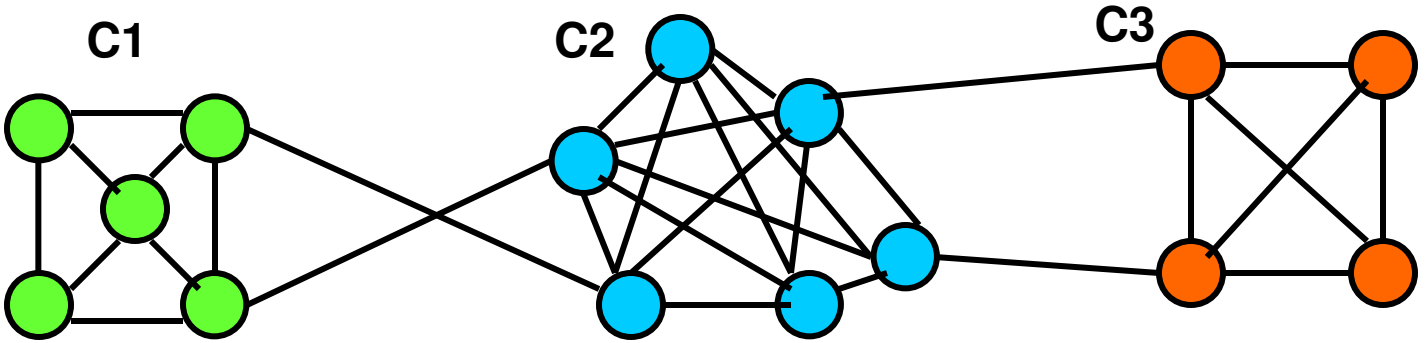
- Community: It is formed by individuals such that those within a group interact with each other more frequently than with those outside the group.
- Community detection: discovering groups in a network where individuals' group memberships are not explicitly given.
  - Interactions (edges) between nodes can help determine communities
- Community structures are quite common in real networks. Social networks include community groups based on common location, interests, occupation, etc.
- Metabolic networks have communities based on functional groupings.
- Citation networks form communities by research topic.
- Identifying the community sub structures within a network can provide insight into how network function and topology affect each other.



There is most likely a path from one vertex to another vertex within a community through the vertices that are also part of the same community.

For the Karate Club network (to the left), the internal densities of the two communities are 0.26 and 0.24; the external densities are 0.035; the overall network density is 0.14.

Internal and External Community Densities



- Let C be a subset of nodes (V) that form a community.
- For every node i in C, let k_i^{int} and k_i^{ext} be the # links connecting node i to a node in C and outside C respectively.

$$\delta_{\text{int}}(C) = \frac{\sum_i k_i^{\text{int}}}{n_C(n_C - 1)}$$

$$\delta_{ext}(C) = \frac{\sum_i k_i^{ext}}{2n_C(n_C - 1)}$$

The internal density of every cluster is significantly larger than the external density as well as the total density of the network.

Internal Densities

C1	(4*3 + 1*4) / (5*4) = 0.8
C2	(6*5)/(6*5) = 1.0
C3	(4*3)/(4*3) = 1.0

External Densities

C1	(1 + 1) / (2*5*4) = 0.05
C2	(1 + 1 + 1 + 1)/(2*6*5) = 0.067
C3	(1 + 1)/(2*4*3) = 0.083

Schemes for Identifying Communities

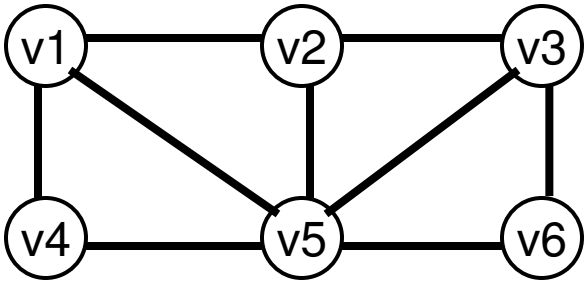
- The number of communities within a network is typically unknown and the communities are often of unequal size and/or density.
- Schemes:
  - Clique-based
  - Hierarchical Clustering
    - Bottom-up and Top-down
  - Neighborhood Overlap based
  - Homophily
  - Eigen Vector based
- Evaluation:
  - Modularity Maximization
  - Silhouette Index

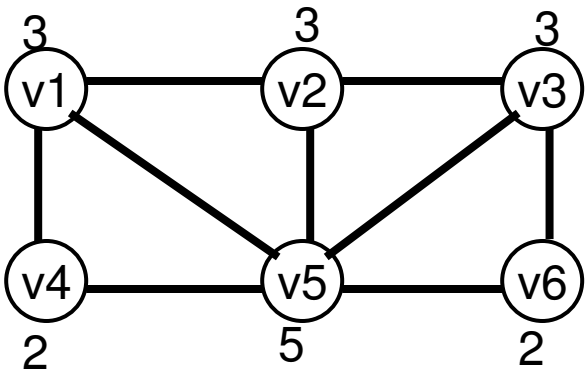
Clique-based Schemes

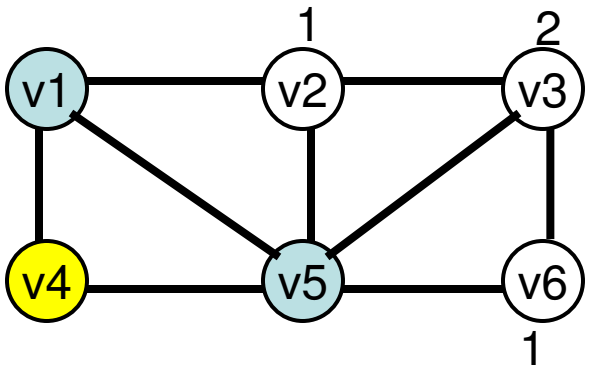
Clique (Complete Mutuality)

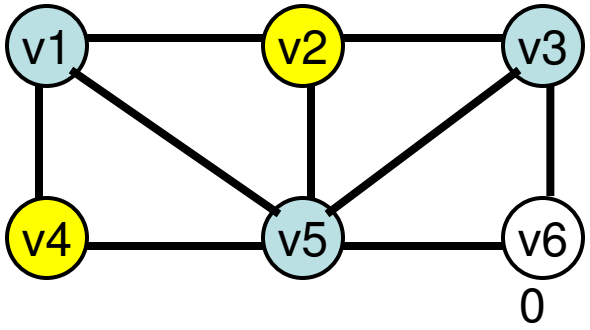
- A clique in a graph is a sub graph in which all the constituent nodes are directly reachable from one another.
- It is a NP-hard problem to find the maximum-sized clique in a graph.
  - Independent Set-based Minimum Neighbors Heuristic
    - Could find more than one clique of different sizes
  - Repeated Vertex and Edge Removal Heuristic
    - To find a clique of maximum size (depends on an underlying heuristic)
  - Clique Percolation methods (could find overlapping communities)
- We will use the notion of Independent Sets to find a clique in a graph
  - Independent Set: A subset of vertices such that there is no edge in the graph between any two vertices in the subset
  - For a given graph G, we will find a complement graph G*
    - The vertices in G and G* are the same.
    - If an edge (u, v) is in G, there is no edge (u, v) in G*
    - If an edge (u, v) is not in G, there is an edge (u, v) in G*
  - An independent set in the complement graph G* is a clique in the original graph G.

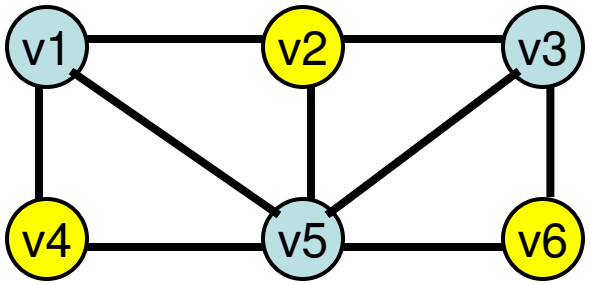
Example to Find Independent Set and Clique: Minimum Neighbors Heuristic











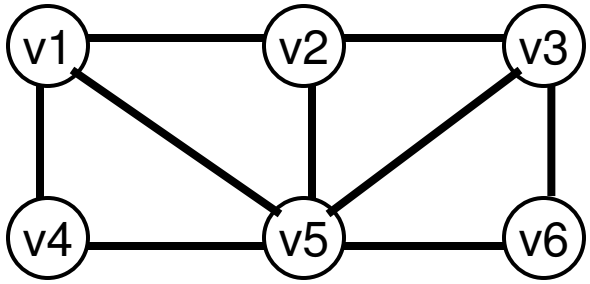
Idea: Give preference to vertices with minimal number of (uncovered) neighbors to be part of the Independent Set. A vertex is said to be covered if itself or any of its neighbors in the Independent Set.

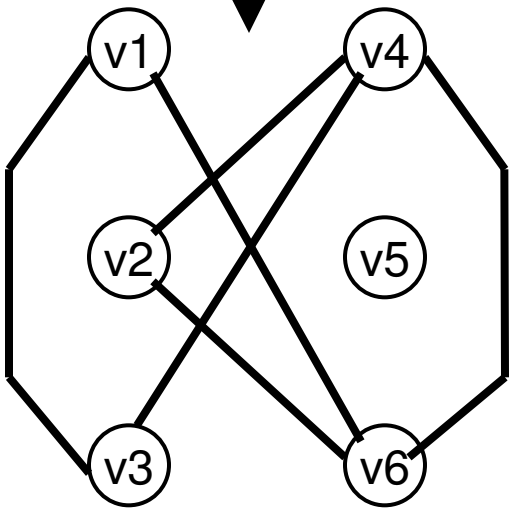
Independent Set for the above graph = {v2, v4, v6}

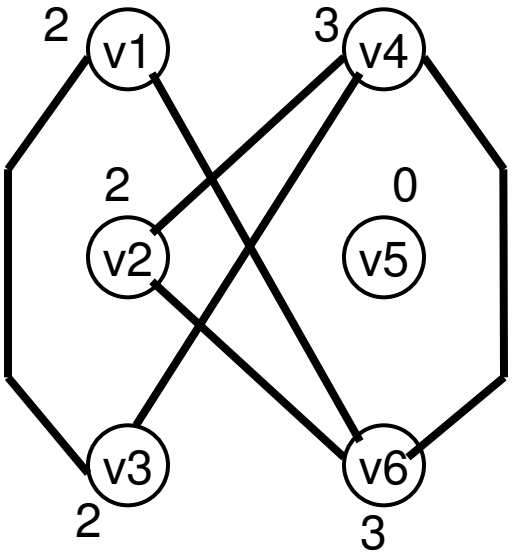
This is also the Maximal Independent Set (i.e., there exists no Independent Set of size 4 or more for the above graph). However, the heuristic is not guaranteed in general to give a maximal Independent set.

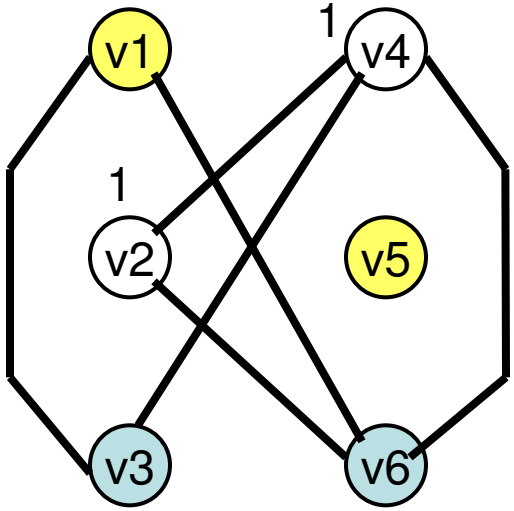
Given G ------->

Find G*, complement of G



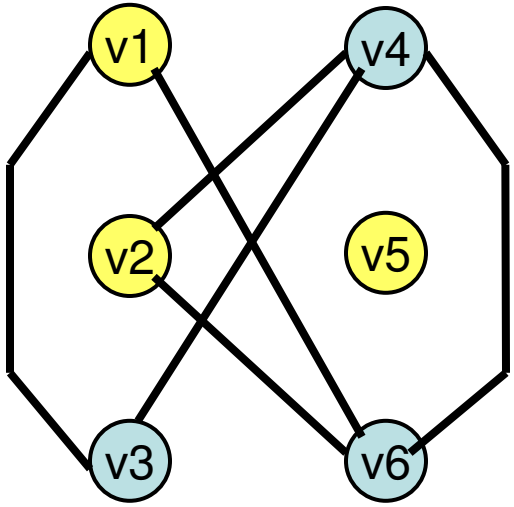




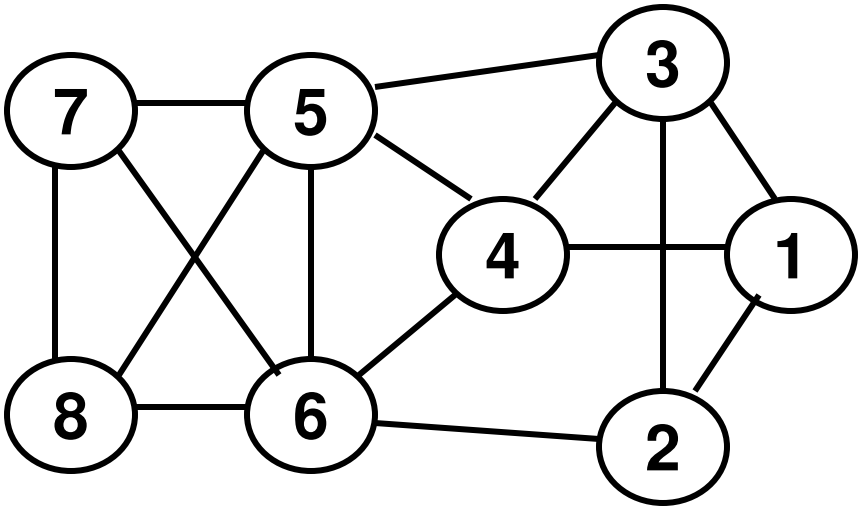


Example 1 to Determine a Clique Using the Minimum Neighbors Heuristic to Approximate an Independent Set

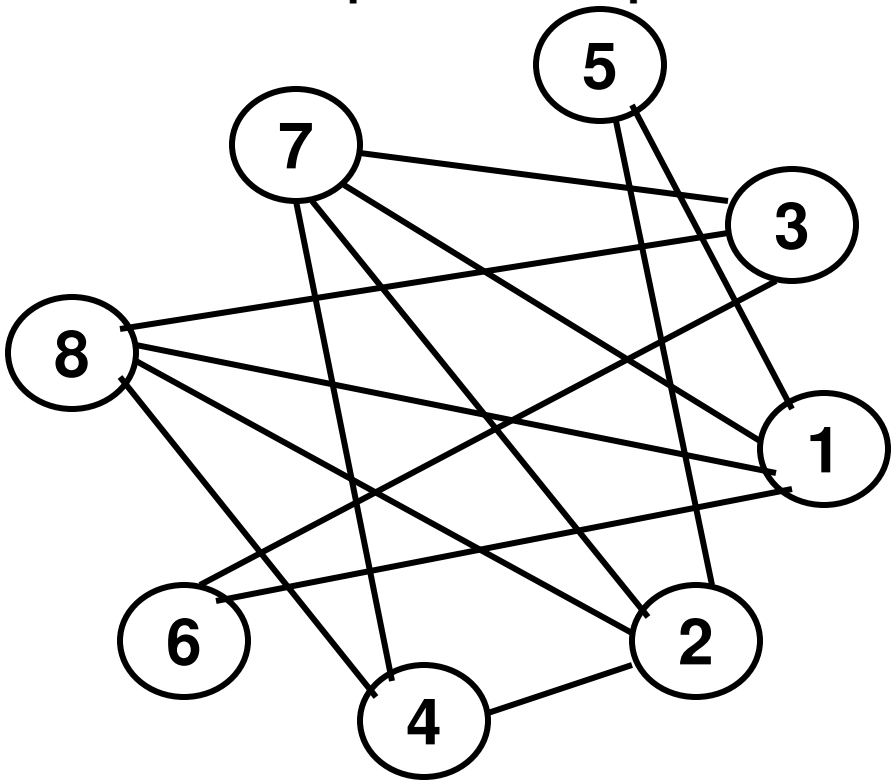
{v1, v2, v5} is an Independent Set in G* and it is a clique in G.



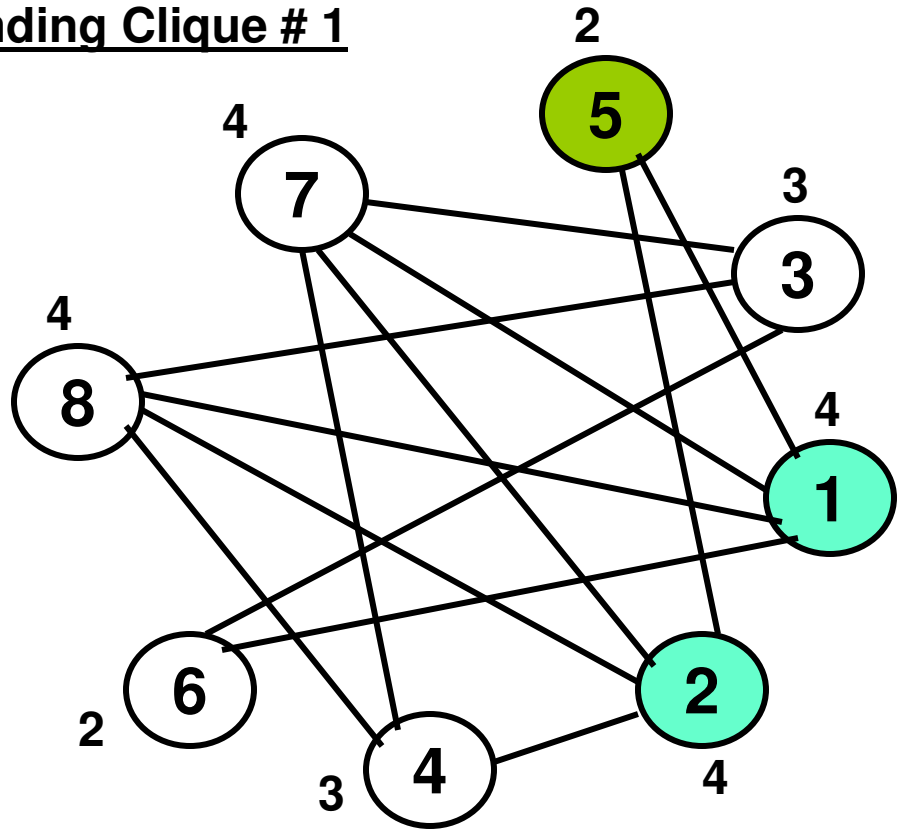
Example to Find Several Cliques In a Graph

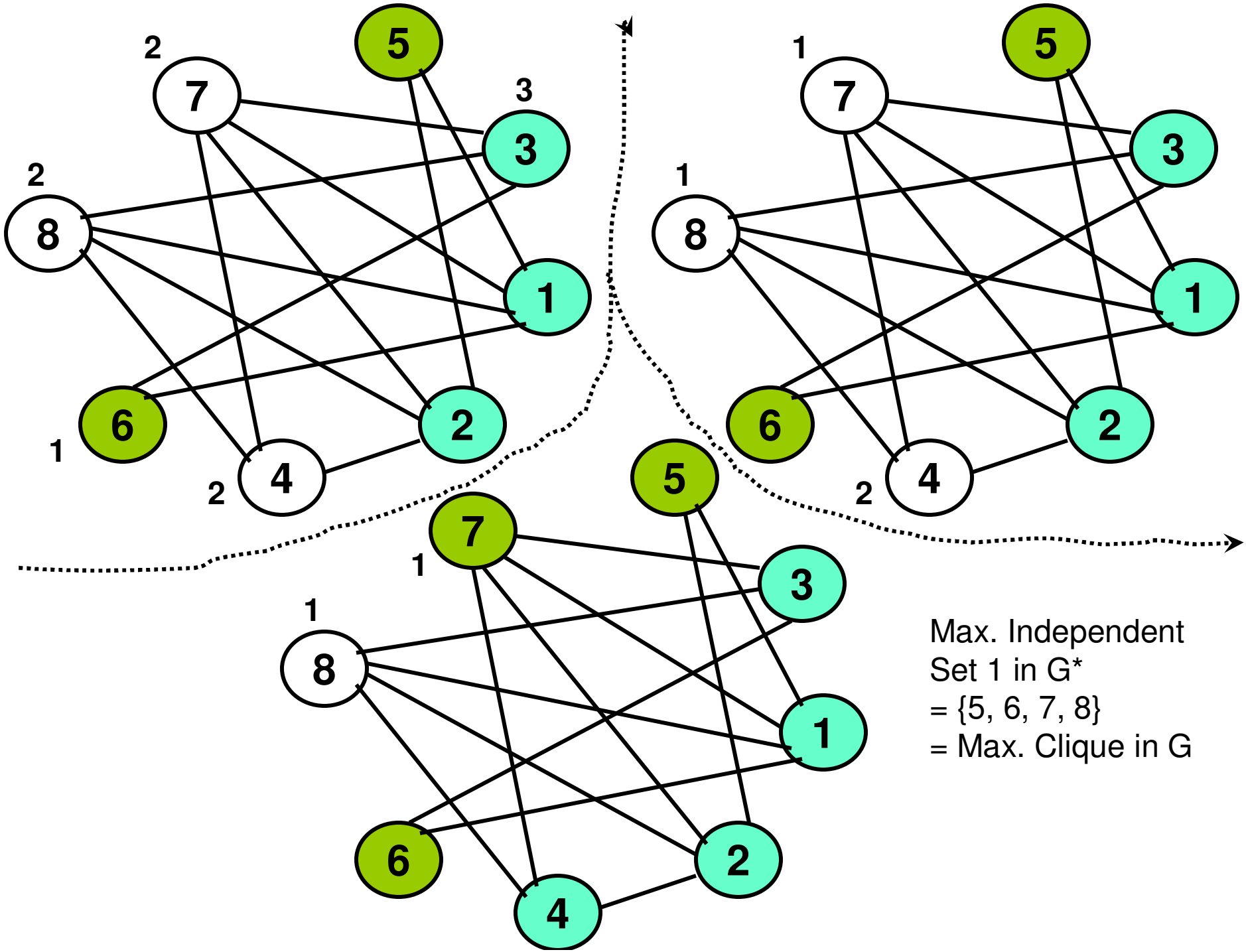


Find the Complement Graph G*



Finding Clique # 1



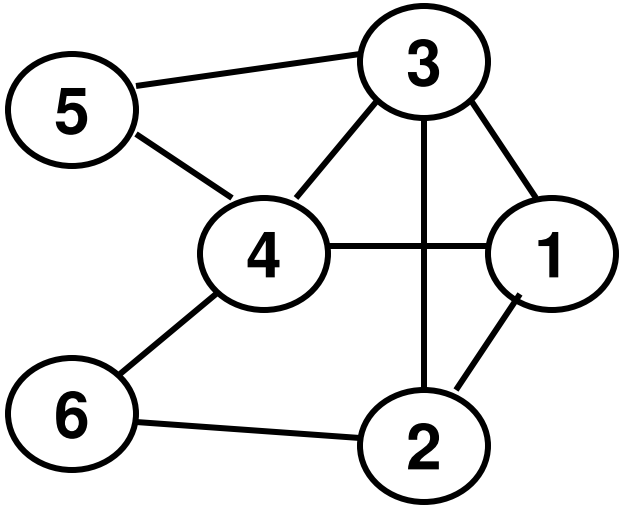

5
2
7
3
3
2
8
1
6
2
1
4
2
5
1
7
3
1
8
1
6
2
2
4
5
7
1
3
1
8
1
6
2
4
Max. Independent
Set 1 in G*
= {5, 6, 7, 8}
= Max. Clique in G

Removing the edges associated with 5, 6, 7, 8
from the original graph G and generate a new graph G'
Find a new complement Graph G (for G')**

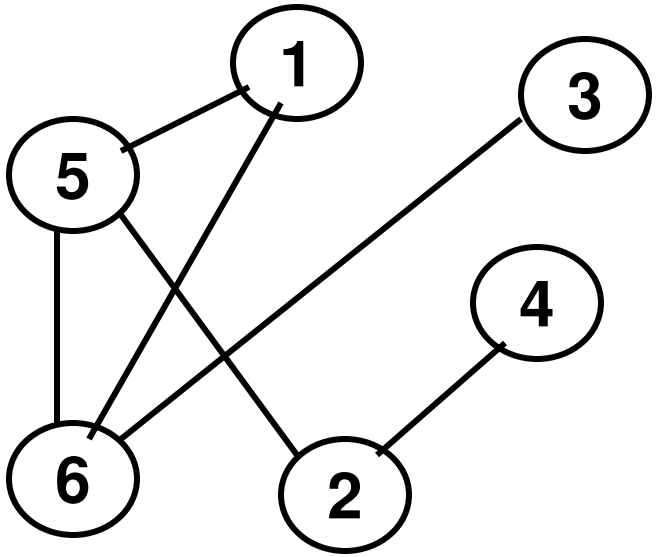
[5, 6, 7, 8] is Clique # 1

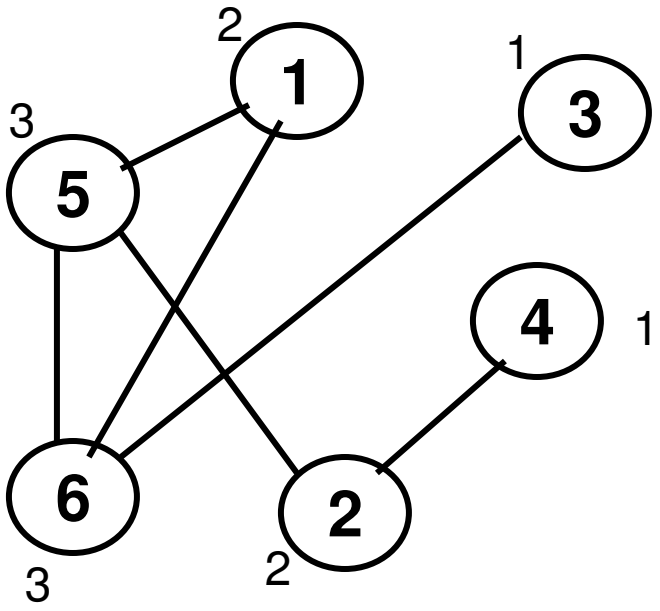
Remove stub nodes and isolated nodes
(nodes with only one edge or no edge)

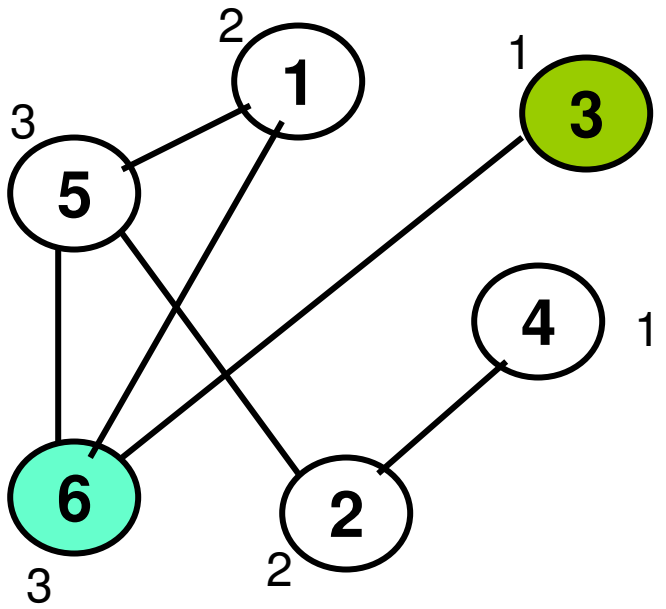
Reduced graph G'

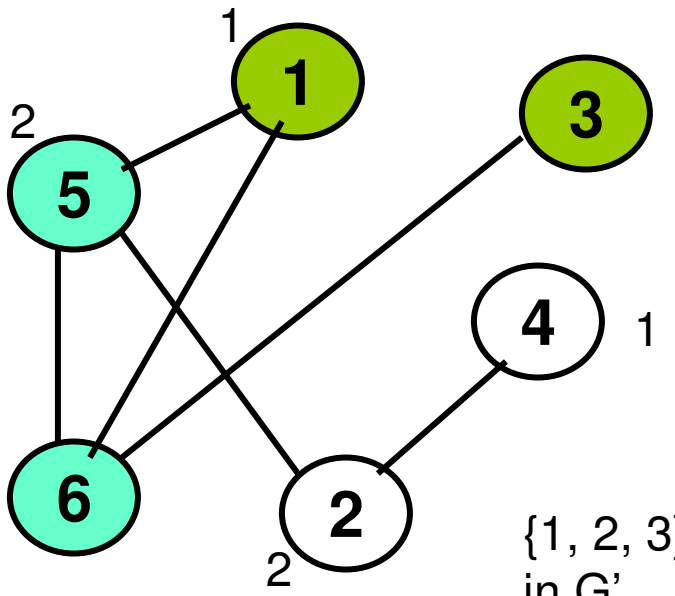


New Complement Graph G**

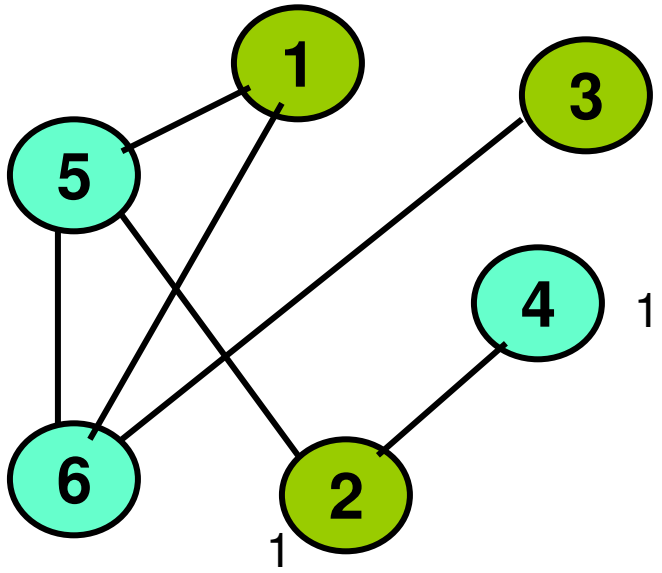


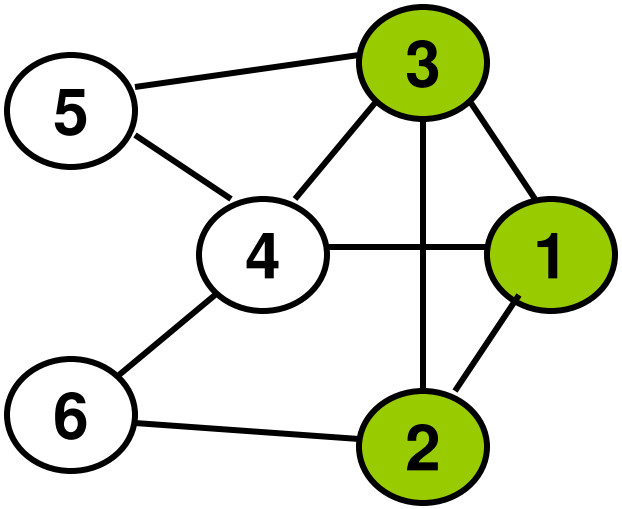






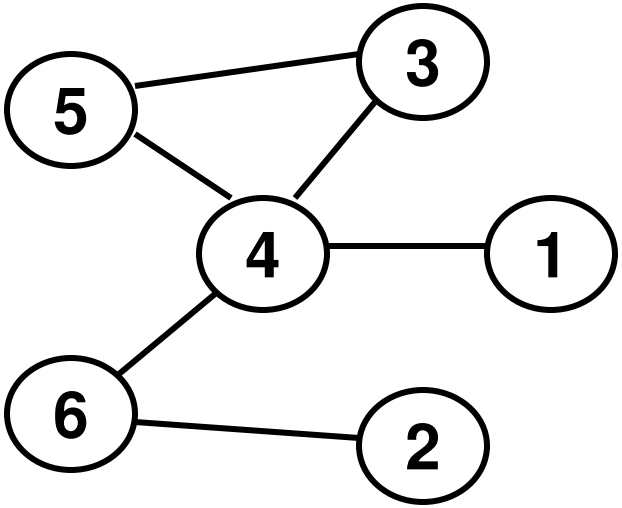
{1, 2, 3} is a clique in G'

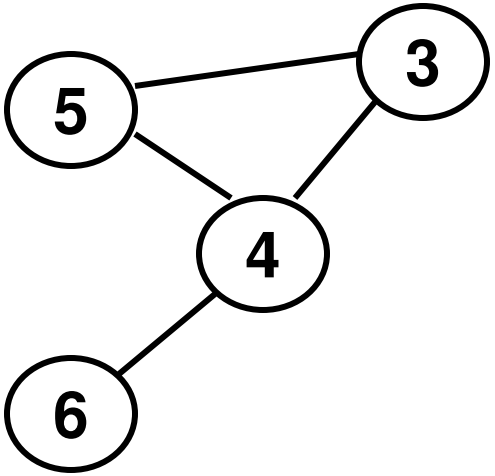




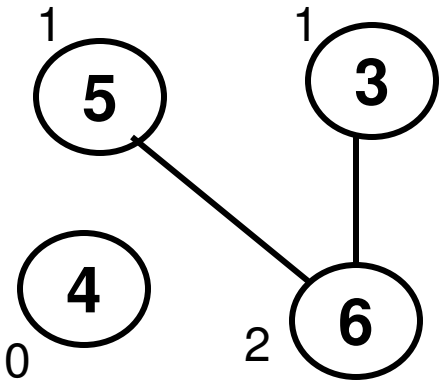
[1, 2, 3] is Clique # 2

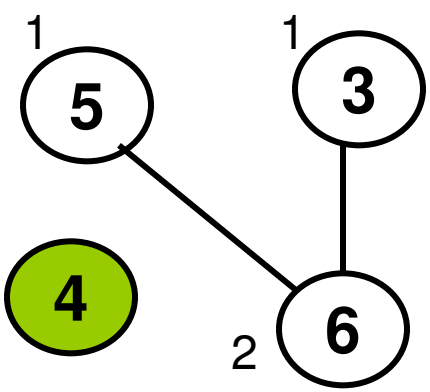
Removing edges associated with 1, 2, 3

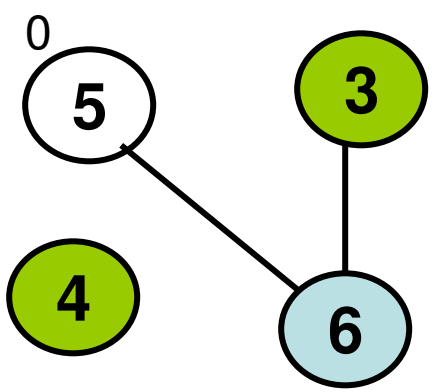




Graph G''

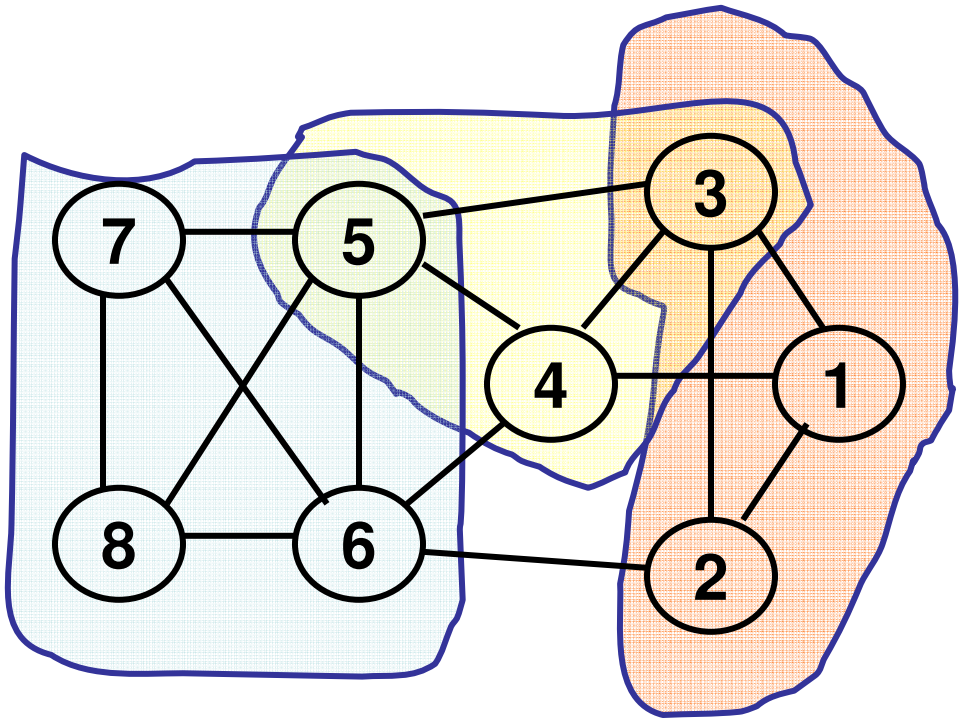






Complement Graph
G* for G''**

[3, 4, 5] is a clique in G''

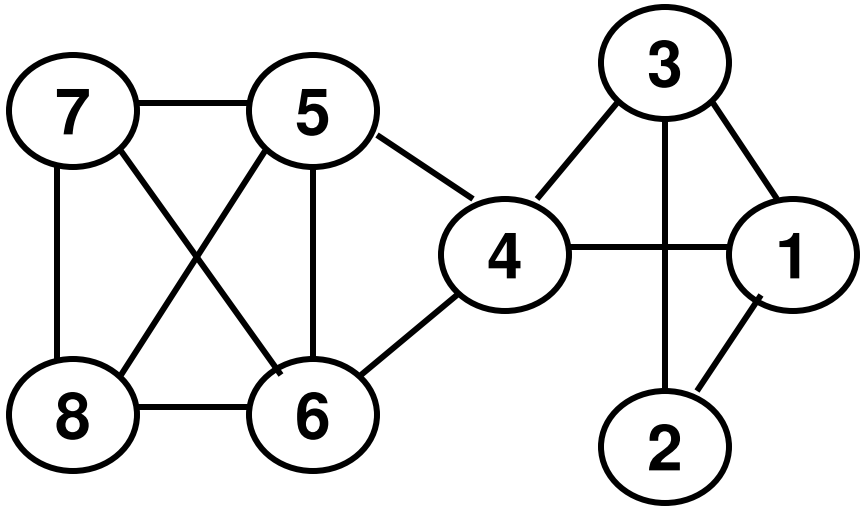


The three cliques

[5, 6, 7, 8]
[1, 2, 3]
[3, 4, 5]

Pruning Technique: Maximum Clique

- Lets say, we are interested to find a clique of size k.
  - All vertices in such clique must have degree at least k-1.
    - Repeat the following until we only have vertices with degree k-1 or above in the graph
      - Step 1: Remove vertices that have degree less than k-1
      - Step 2: Because of the removal of the vertices in Step 1, the degree of some other vertices would have become less than k-1.
        - » If any such vertices exist, Go to Step 1.
        - » Otherwise, exit from the loop
    - If the reduced graph obtained from the above has one or more components in which each component has vertices with degree k-1 or above, run any heuristic to find clique (say, the Independent Set-based heuristic) on the reduced graph

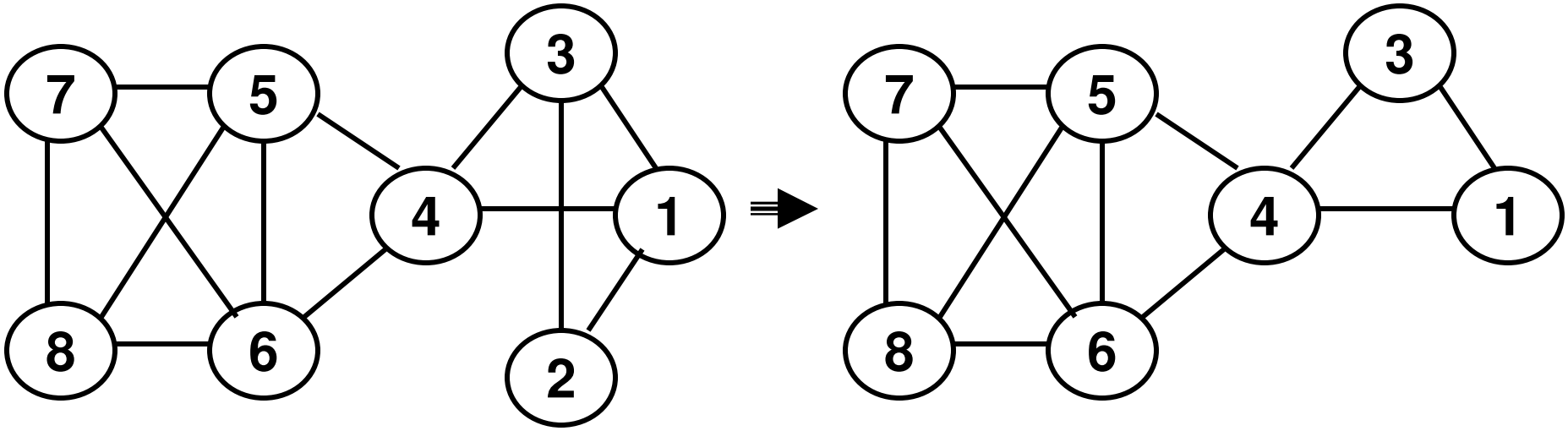


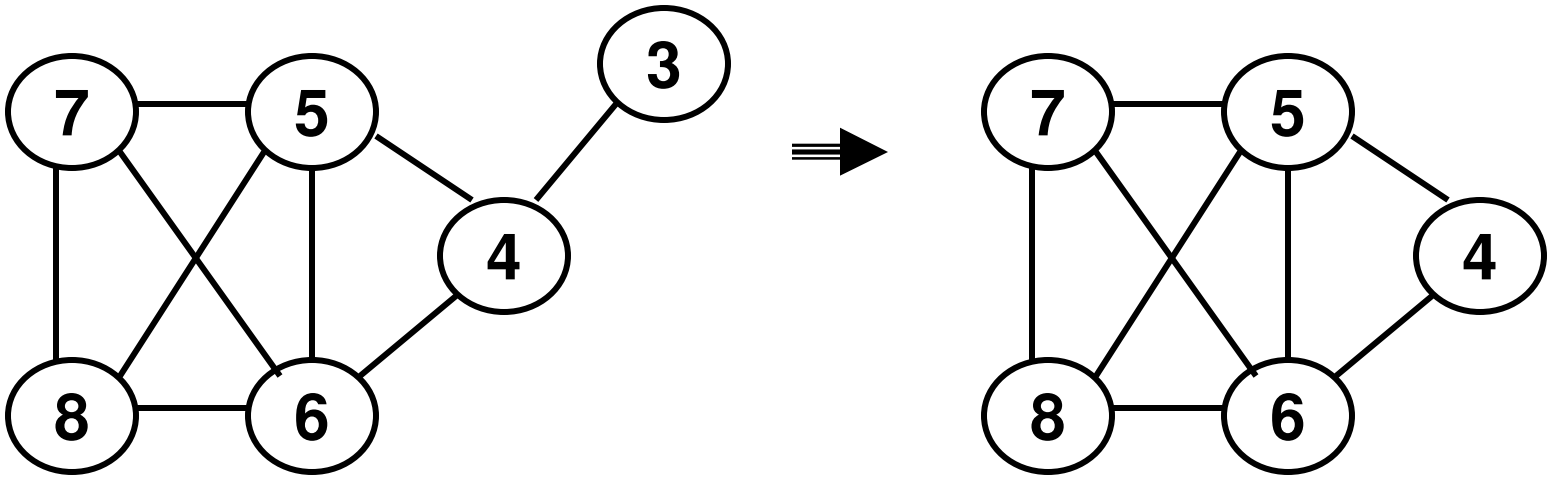
Example to Find The Maximum Clique Using Pruning Technique

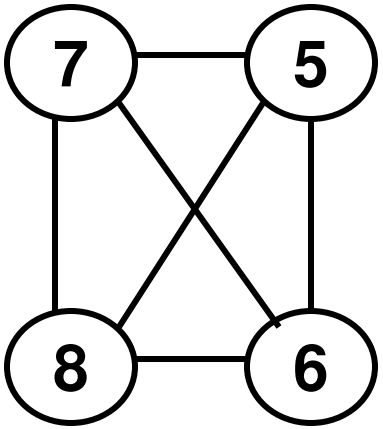
Finding the Maximum Clique

Anticipating a Clique of Size 4 (i.e., each vertex in the clique has degree 3)
Remove from the graph all vertices with degree less than 3.
Recursively remove all the vertices and associated edges until each vertex has degree 3 or above.

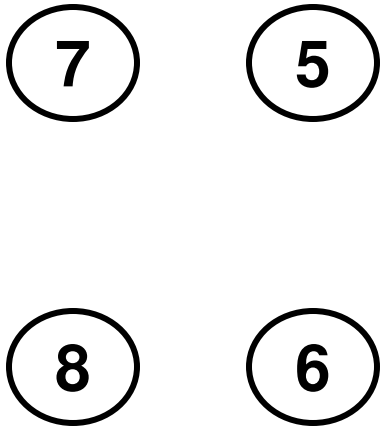
May not be effective all the time







Apply the Independent Set Heuristic



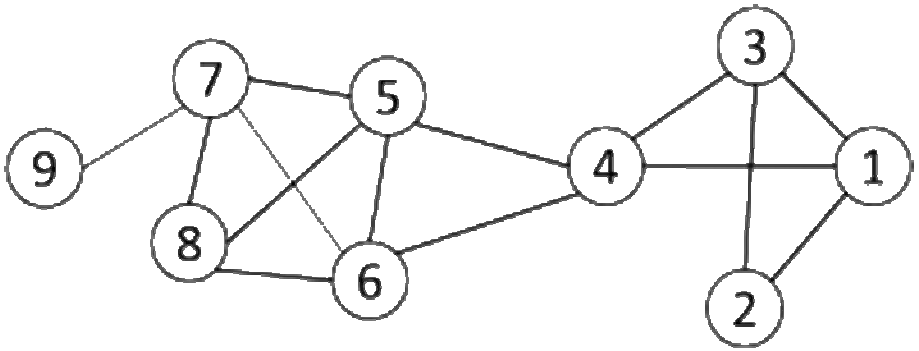
[5, 6, 7, 8] form an independent set in the complement graph

Hence [5, 6, 7, 8] form a clique in the original graph

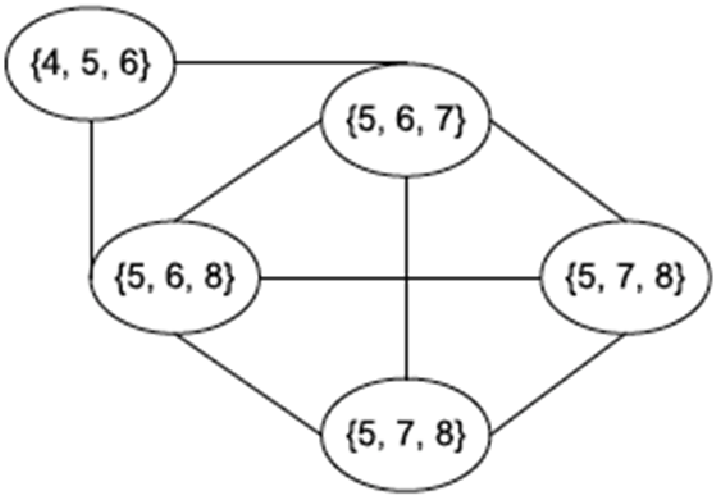
Clique Percolation Method

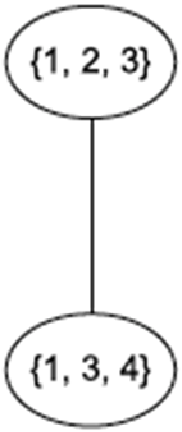
- Used to find overlapping communities
  - **Input**
    - A parameter k, and a network
  - **Procedure**
    - Find out all cliques of size k in a given network
    - Construct a clique graph. Two cliques are adjacent if they share k-1 nodes
    - Each connected component in the clique graph forms a community

Example 1: Clique Percolation Method



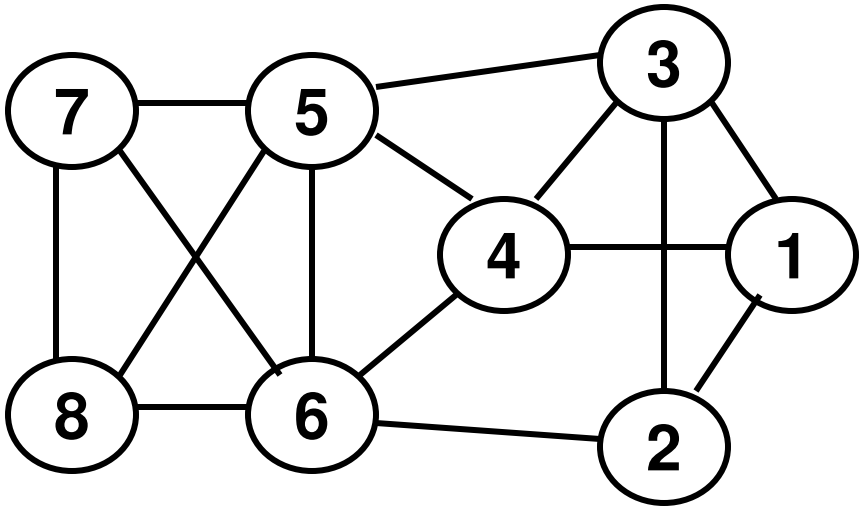
Cliques of size 3:
{1, 2, 3}, {1, 3, 4}, {4, 5, 6},
{5, 6, 7}, {5, 6, 8}, {5, 7, 8},
{6, 7, 8}





Communities:
{1, 2, 3, 4}
{4, 5, 6, 7, 8}

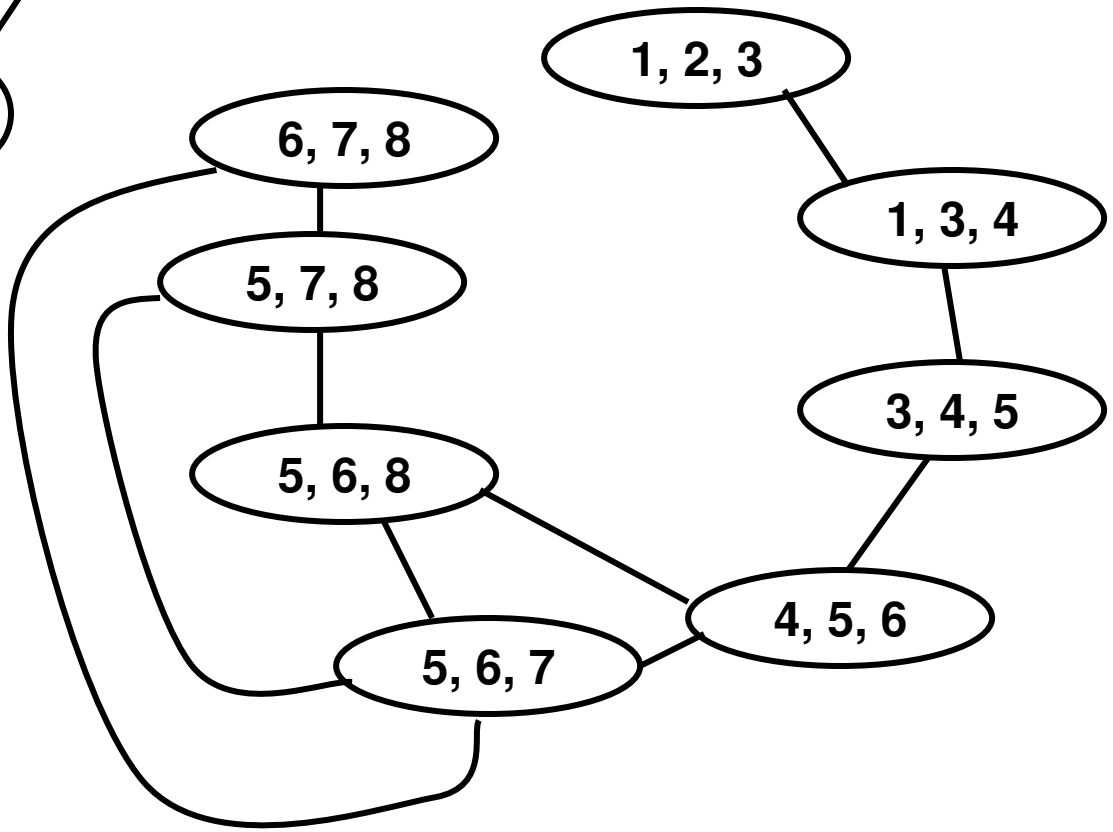
Example 2: Clique Percolation Method



The following is the clique graph. All the cliques of size 3 are connected. Hence, all the vertices in the given graph are said to be in one single community.

Cliques of Size 3

[1, 2, 3]
[1, 3, 4]
[3, 4, 5]
[4, 5, 6]
[5, 6, 7]
[5, 6, 8]
[5, 7, 8]
[6, 7, 8]



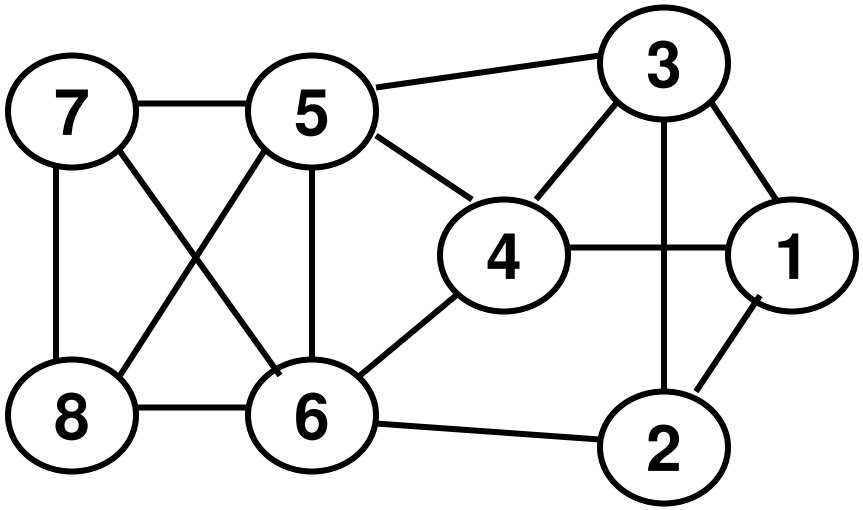
CINET: Clique

- CINET can be used for the following:
  - To count the number of cliques of a given size in a graph
  - To find the maximal cliques containing a particular vertex
  - To find the size of the largest clique in a graph and to find its constituent nodes
- Example: The following results are obtained for the American Football Network (114 nodes; 613 edges)
  - # Cliques of Size 3 (# triangles): 810
  - Largest Clique is of Size 9 and it comprises of nodes 25, 33, 109, 45, 1, 103, 89, 37, 105
  - The maximal clique that contains vertex 0 is of size 8 and the constituent nodes are: 0, 23, 16, 9, 93, 4, 104, 41

Modularity Maximization

Modularity Maximization

- Modularity measures the strength of a community partition by taking into account the degree distribution.
- Given a network with *m* edges, the expected number of edges between two nodes *i* and *j* with degrees d_i and d_j respectively is $d_i^* d_j$ / 2m.



Expected number of edges between nodes 1 and 2 is (3)(2) / (2*15) = 0.20

Strength of a Community, C

$$\sum_{i \in C,\, j \in C} A_{i,j} - \frac{d_i d_j}{2m}$$

For a network with *k* communities and a total of *m* edges

Modularity:

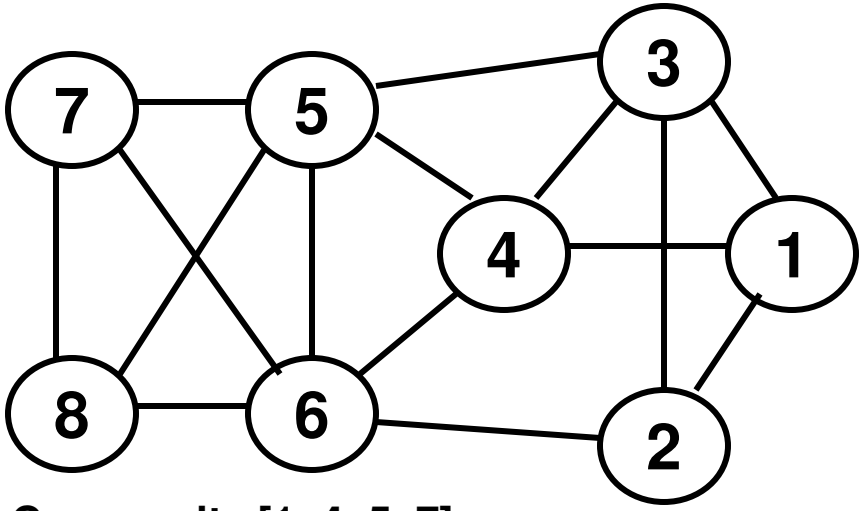
$$Q = \sum_{l=1}^{k} \sum_{i \in C_l,\, j \in C_l} A_{i,j} - \frac{d_i d_j}{2m}$$

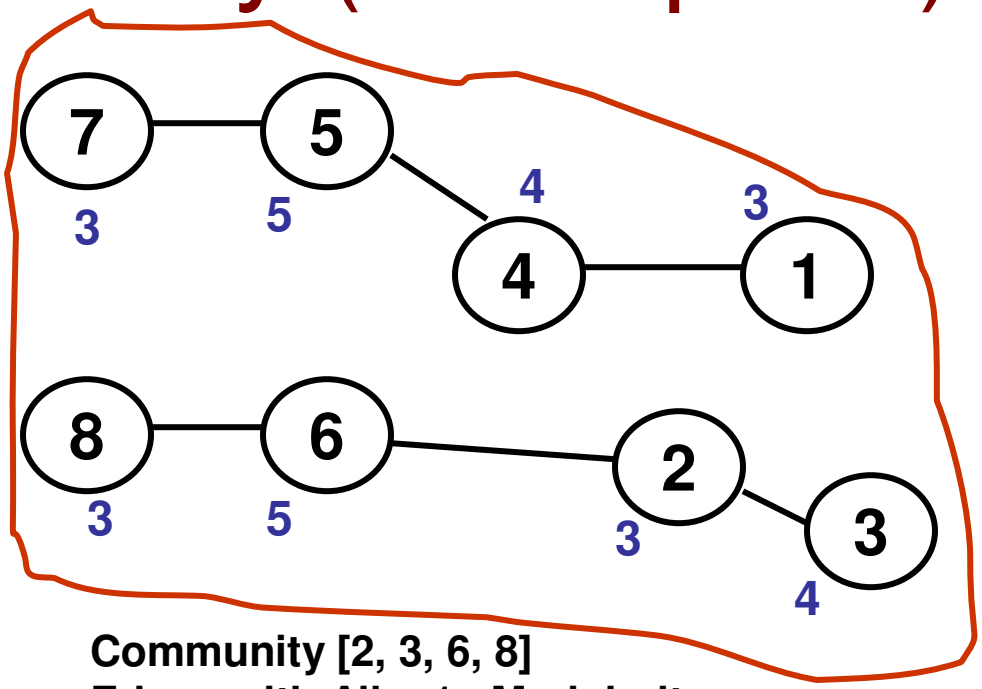
A larger value for Q indicates a good community structure

Modularity Maximization

- The intuition behind the idea of modularity is that a community is a structural element of a network that has been formed in a manner far from a random process.
- If we consider the actual density of links in a community, it should be significantly larger than the density we would expect if the links in the network were formed by a random process.
  - In other words, if two nodes i and j are end up being in the same community, there should be more likely a link between them (i.e., Aij = 1, leading to an overall high value for Q).
  - If i and j end up being in a community such that the chances of having a link between them is just as the same as between any two nodes in the network (i.e., a random network), then the value of Q is more likely to be low (because there could be some Aij = 0 that will bring down the value of Q).

Evaluating Modularity (Example 1)





Community [1, 4, 5, 7]

Edges with Aij = 1	Modularity
1 – 4	1 – (3)(4)/(2*15) = 0.60
4 – 5	1 – (4)(5)/(2*15) = 0.33
5 – 7	1 – (3)(5)/(2*15) = 0.50
Edges with Aij = 0	
1 – 5	0 – (3)(5)/(2*15) = -0.50
1 – 7	0 – (3)(3)/(2*15) = -0.30
4 – 7	0 – (4)(3)/(2*15) = -0.40

Total Modularity Score for Community [1, 4, 5, 7] 0.23

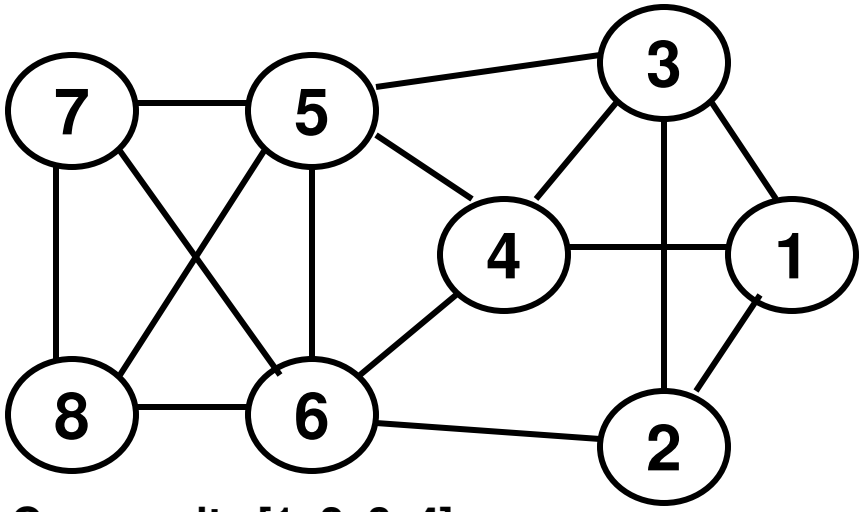
Community [2, 3, 6, 8]

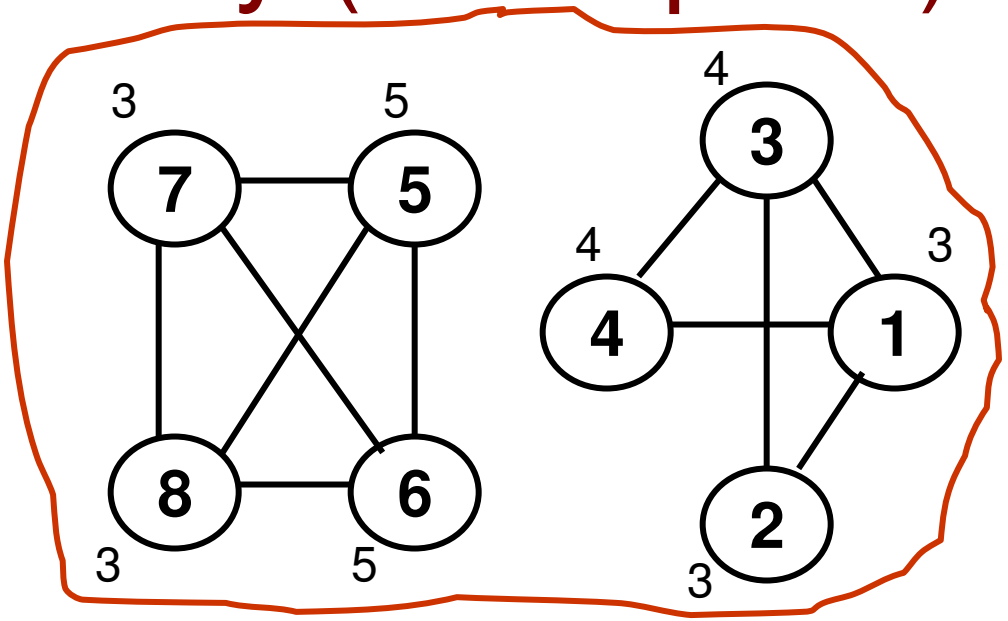
Edges with Aij = 1	Modularity
2 – 3	1 – (3)(4)/(2*15) = 0.60
2 – 6	1 – (3)(5)/(2*15) = 0.50
6 – 8	1 – (3)(5)/(2*15) = 0.50
Edges with Aij = 0	
2 – 8	0 – (3)(3)/(2*15) = -0.30
3 – 6	0 – (4)(5)/(2*15) = -0.67
3 – 8	0 – (4)(3)/(2*15) = -0.40

Total Modularity Score for Community [2, 3, 6, 8] 0. 23

Total Modularity for the two Communities: 0.23 + 0.23 = 0.46

Evaluating Modularity (Example 2)





Community [1, 2, 3, 4]

Edges with Aij = 1	Modularity
1 – 2	1 – (3)(3)/(2*15) = 0.70
1 – 3	1 – (3)(4)/(2*15) = 0.60
1 – 4	1 – (3)(4)/(2*15) = 0.60
2 – 3	1 – (3)(3)/(2*15) = 0.70
3 – 4	1 – (4)(4)/(2*15) = 0.47
Edges with Aij = 0	
2 – 4	0 – (3)(4)/(2*15) = -0.40

Total Modularity Score for Community [1, 2, 3, 4] **2.67**

Community [5, 6, 7, 8]

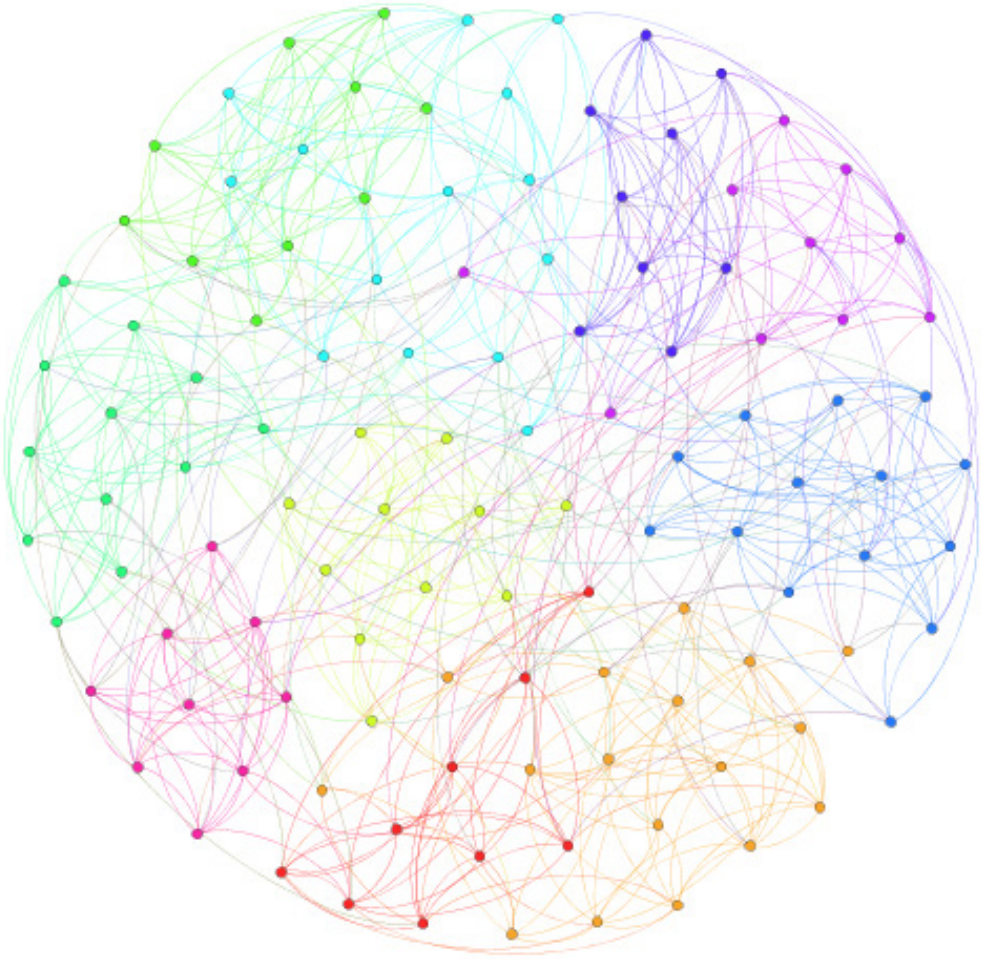
Edges with Aij = 1	Modularity
5 – 6	1 – (5)(5)/(2*15) = 0.17
5 – 7	1 – (3)(5)/(2*15) = 0.50
5 – 8	1 – (3)(5)/(2*15) = 0.50
6 – 7	1 – (3)(5)/(2*15) = 0.50
6 – 8	1 – (3)(5)/(2*15) = 0.50
7 – 8	1 – (3)(3)/(2*15) = 0.70

Total Modularity Score for Community [2, 3, 6, 8] **2.87**

Total Modularity for the two Communities: 2.67 + 2.87 = 5.54

CINET: Cliques and Communities

- The following communities could be predicted in the American College Football network based on the cliques identified by CINET. The modularity score of the communities is 0.6.



Evaluating Clusters with Silhouette Index

Silhouette Index

- Let there be *r* communities (clusters) C1, C2, …, Cr.
- For every node i, determine the shortest path distances (# hops) to every other node j.
- Find the average distance of node i to the nodes in each of the *r* clusters.
- For a node i in cluster Ck, find the minimum average distance ($\overline{d}\min$) to a cluster Cj (j ≠ k).

Silhouette Index of a node i:

$$s(i) = \frac{\overline{d}\min - \overline{d}_{i,C_k}}{\max\{\overline{d}\min, \overline{d}_{i,C_k}\}}$$

Silhouette Index of a cluster C_k

$$s(C_k) = \frac{1}{n_{C_k}} \sum_{i=1}^{nCk} s(i)$$

Silhouette Index of a graph G

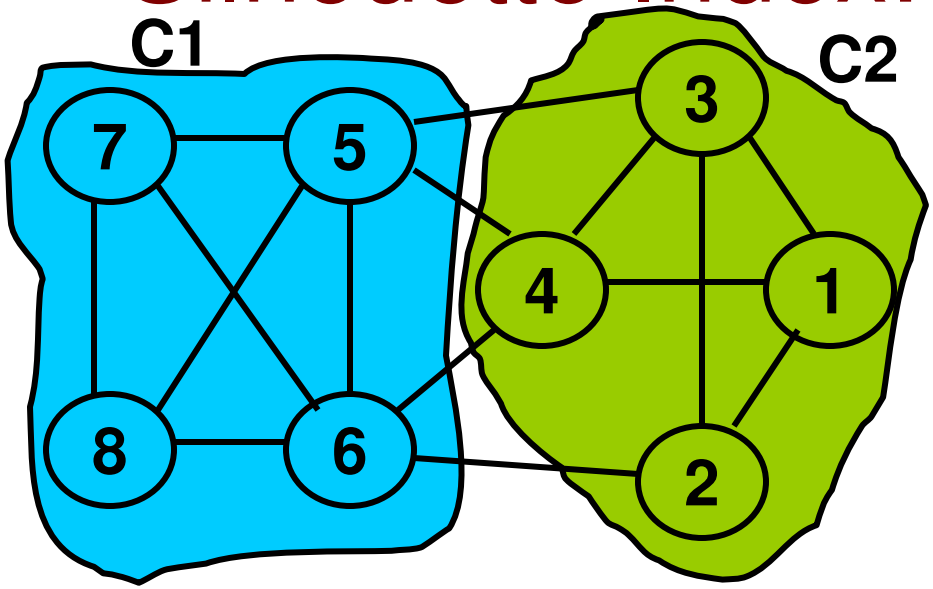
$$s(G) = \frac{1}{r} \sum_{j=1}^{r} s(C_r)$$

s(i) → 1 implies the node is in the best possible cluster

s(i) → -1 implies the node is in the wrong Cluster

s(i) → 0, implies it would be possible to move the node to an adjacent cluster

Silhouette Index: Correct Clustering



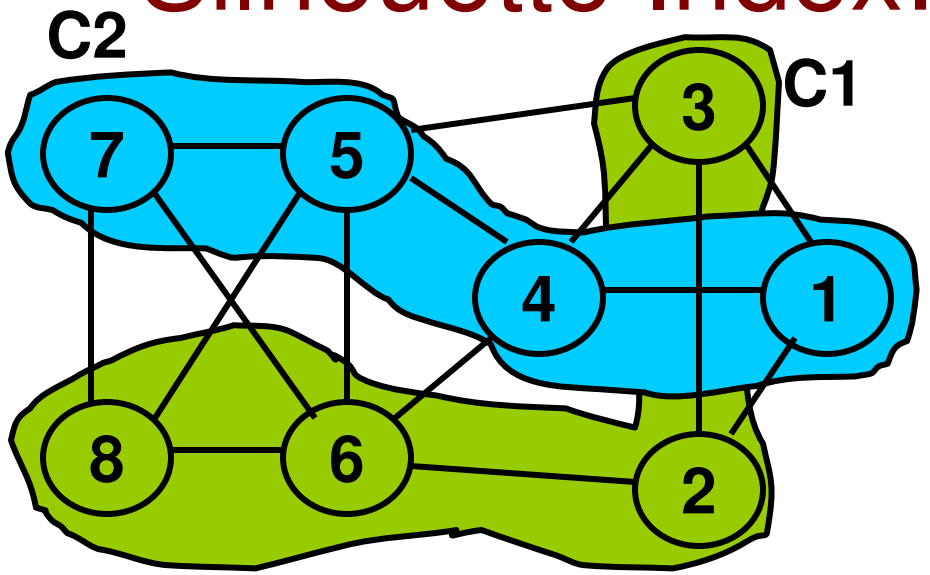
	1	2	3	4	5	6	7	8
1	**0**	**1**	**1**	**1**	**2**	**2**	**3**	**3**
2	**1**	**0**	**1**	**2**	**2**	**1**	**3**	**2**
3	**1**	**1**	**0**	**1**	**1**	**2**	**2**	**3**
4	**1**	**2**	**1**	**0**	**1**	**1**	**2**	**2**
5	**2**	**2**	**1**	**1**	**0**	**1**	**1**	**1**
6	**2**	**1**	**2**	**1**	**1**	**0**	**1**	**1**
7	**3**	**2**	**2**	**2**	**1**	**1**	**0**	**1**
8	**3**	**2**	**2**	**2**	**1**	**1**	**1**	**0**

$$s(i) = \frac{\bar{d}\,\min - \bar{d}_{i,C_k}}{\max\{\bar{d}\,\min, \bar{d}_{i,C_k}\}}$$

	Mem. cluster	C1 5, 6, 7, 8	C2 1, 2, 3, 4	$\bar{d}\,\min$ (outside own cluster)	$\bar{d}_{i,C_k}$ (Mem.cluster)		
1	C2	2.5	0.75	2.5	0.75	0.7	s(C2) Avg(si) = 0.54
2	C2	2.0	1.0	2.0	1.0	0.5	
3	C2	2.0	0.75	2.0	0.75	0.625	
4	C2	1.5	1.0	1.5	1.0	0.333	
5	C1	0.75	1.5	1.5	0.75	0.5	s(C1) Avg(si) = 0.59
6	C1	0.75	1.5	1.5	0.75	0.5	
7	C1	0.75	2.25	2.25	0.75	0.67	
8	C1	0.75	2.25	2.25	0.75	0.67	

Overall Silhouette Index of the Network = Average of the index of all clusters = **0.565**

Silhouette Index: Wrong Clustering



	1	2	3	4	5	6	7	8
1	**0**	**1**	**1**	**1**	**2**	**2**	**3**	**3**
2	**1**	**0**	**1**	**2**	**2**	**1**	**3**	**2**
3	**1**	**1**	**0**	**1**	**1**	**2**	**2**	**3**
4	**1**	**2**	**1**	**0**	**1**	**1**	**2**	**2**
5	**2**	**2**	**1**	**1**	**0**	**1**	**1**	**1**
6	**2**	**1**	**2**	**1**	**1**	**0**	**1**	**1**
7	**3**	**2**	**2**	**2**	**1**	**1**	**0**	**1**
8	**3**	**2**	**2**	**2**	**1**	**1**	**1**	**0**

$$s(i) = \frac{\bar{d}\,\min - \bar{d}_i, C_k}{\max\{\bar{d}\,\min, \bar{d}_i, C_k\}}$$

	Mem. cluster	C1 2, 3, 6, 8	C2 1, 4, 5, 7	$\bar{d}\,\min$ (outside own cluster)	$\bar{d}_i, C_k$ (Mem.cluster)		
1	C2	1.75	1.5	1.75	1.5	0.143	s(C2) Avg(si) = 0.17
2	C1	1.0	2.0	2.0	1.0	0.5	
3	C1	1.5	1.25	1.25	1.5	-0.2	
4	C2	1.5	1.0	1.5	1.0	0.333	
5	C2	1.25	1.0	1.25	1.0	0.2	s(C1) Avg(si) = 0.197
6	C1	1.0	1.25	1.25	1.0	0.2	
7	C2	1.5	1.5	1.5	1.5	0	
8	C1	1.25	1.75	1.75	1.25	0.286	

Overall Silhouette Index of the Network = Average of the index of all clusters = **0.183**

Hierarchical Clustering (Complete Linkage Clustering)

Bottom-Up Approach

(Agglomerative)

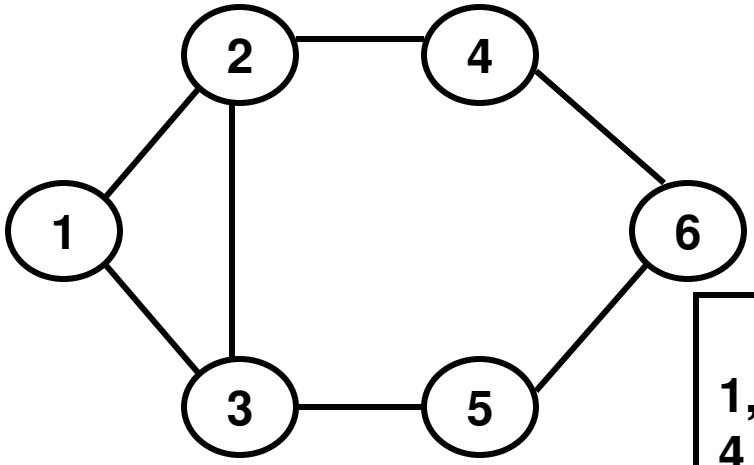
Complete Linkage Clustering

- Compute the "pair-wise" distance matrix P between any two vertices.
- Initially, start with each vertex in its own cluster.
- Merge the two "closest" vertices (clusters)
  - In case of a tie (between two or more pairs of clusters), choose the pair with the minimum value for the total pair-wise distance / sum of the two pair sizes
- Remove the entries from P, for the two vertices (clusters) merged, and add an entry corresponding to the merged vertex (cluster).
  - Update this entry with the longest distance between any vertex in the merged cluster with the vertices in the other clusters in P.

$$\text{Distance}(C_i, C_j) = Max\left[\forall_{u \in C_i} \forall_{v \in C_j} MinHops(u, v)\right]$$

- Repeat the above step of merging and removing/adding entries to P until there is only one cluster.

Complete Linkage Clustering (Example 1)

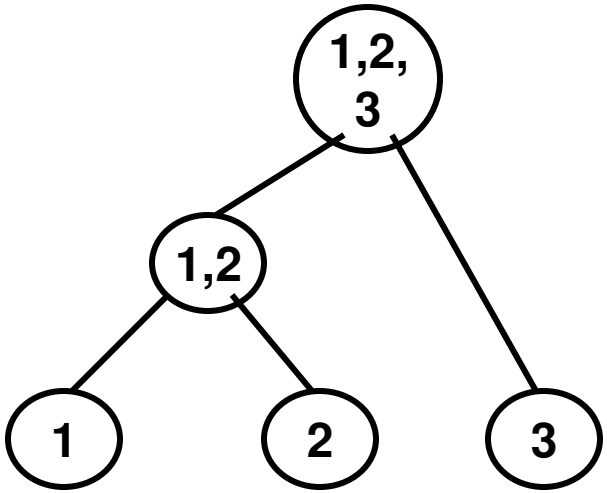


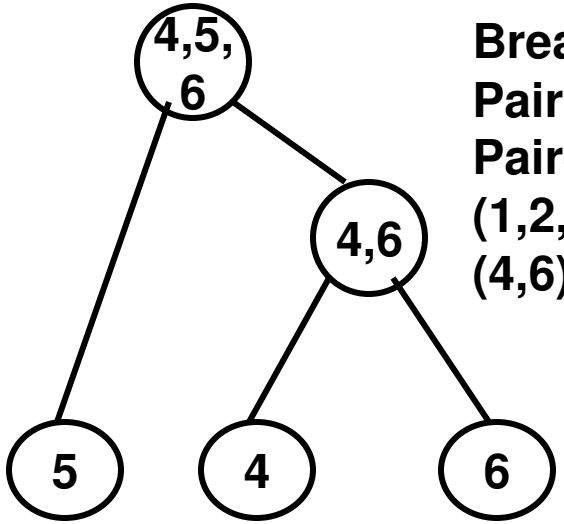
	1	2	3	4	5	6
1	0	1	1	2	2	3
2		0	1	1	2	2
3			0	2	1	2
4				0	2	1
5					0	1
6						0

	1,2	3	4	5	6
1,2	0	1	2	2	3
3		0	2	1	2
4			0	2	1
5				0	1
6					0

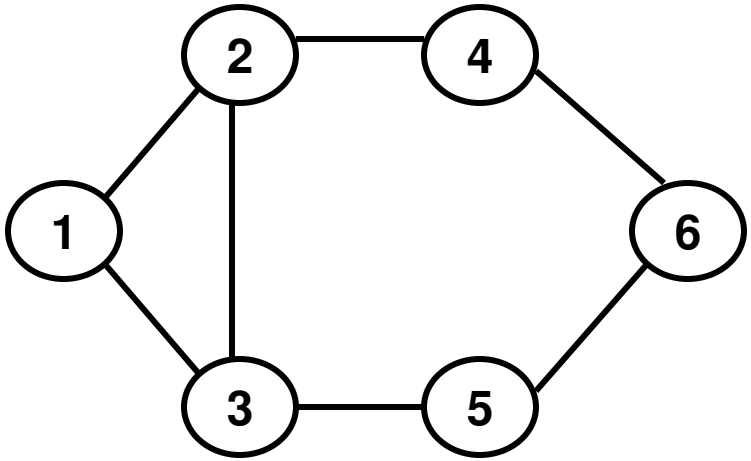
	1,2,3	4	5	6
1,2,3	0	2	2	3
4		0	2	1
5			0	1
6				0

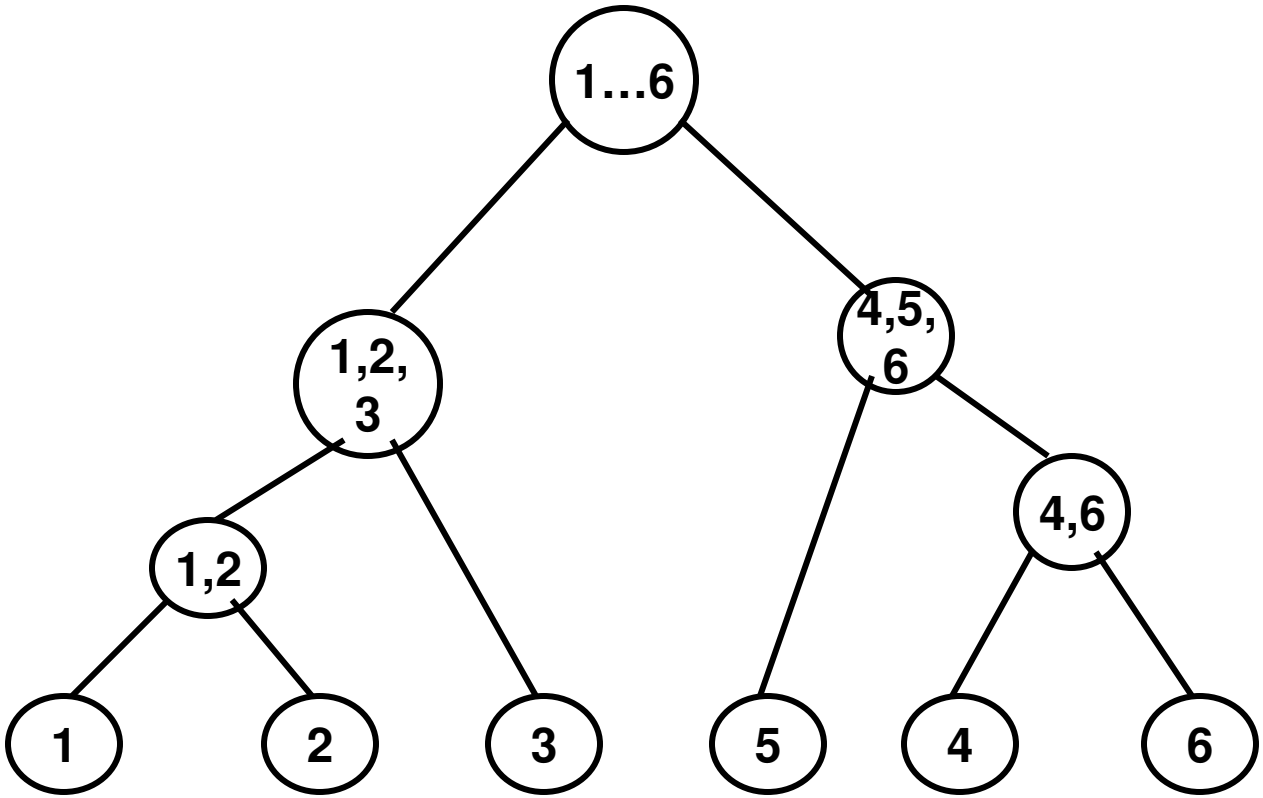
	1,2,3	4,6	5	
1,2,3	0	3	2	(5)
4,6		0	2	(3)
5			0	



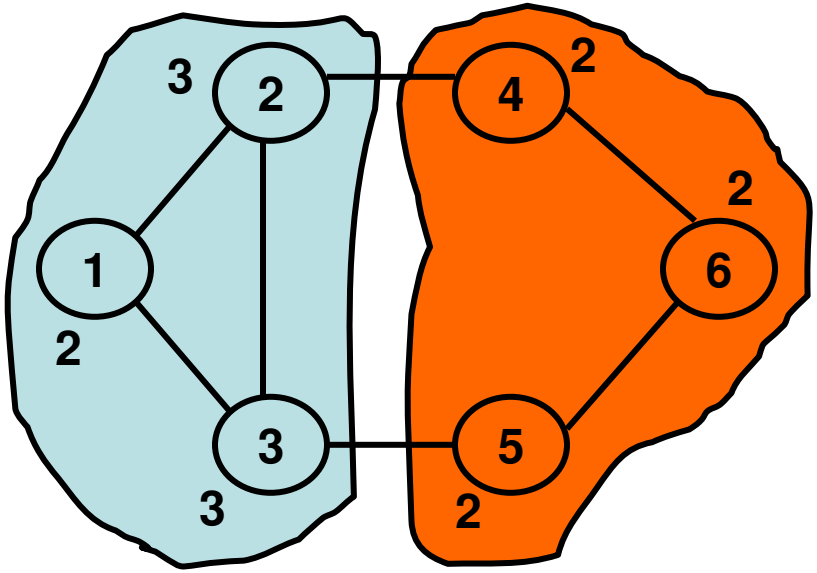


Break the tie by choosing the Pair with the minimum total Pair-wise distance / Pair size
(1,2,3) and (5): 5/4 = 1.25
(4,6) and (5): 3/3 = 1.0



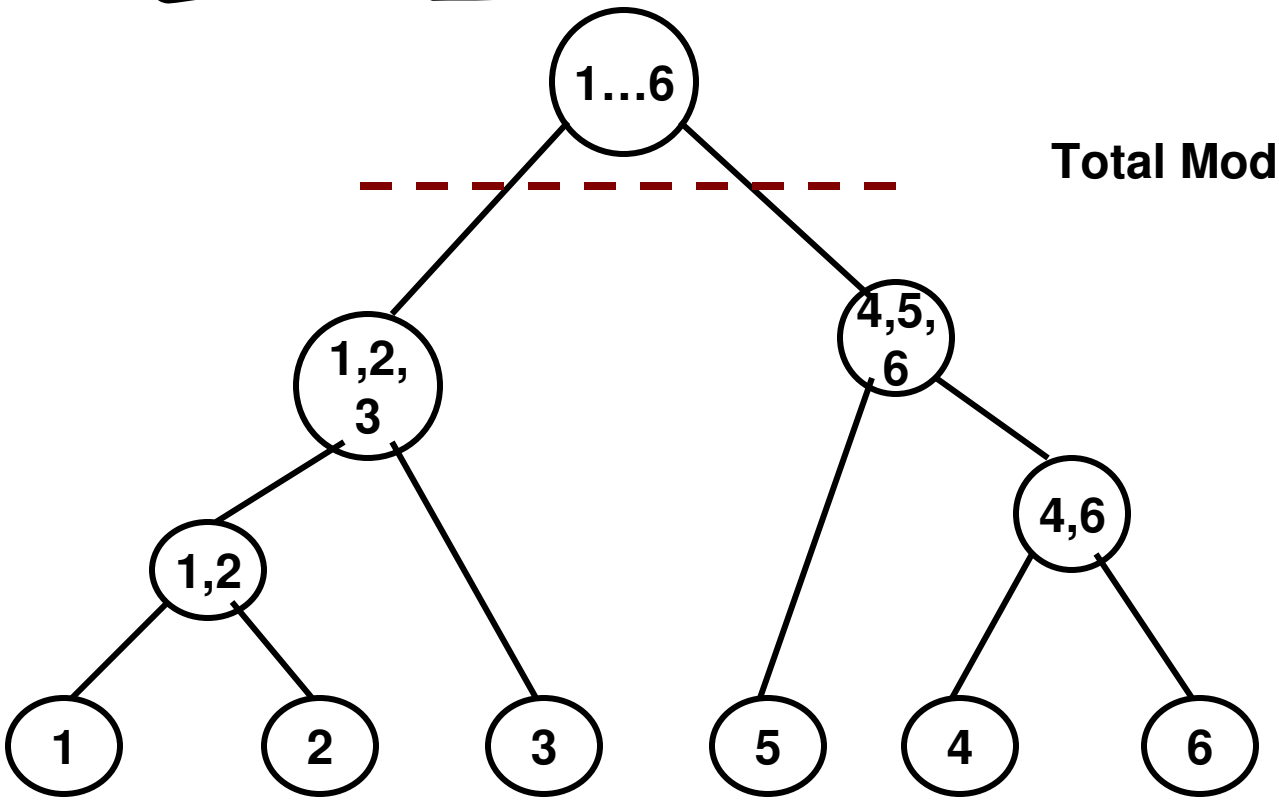


Complete Linkage Clustering



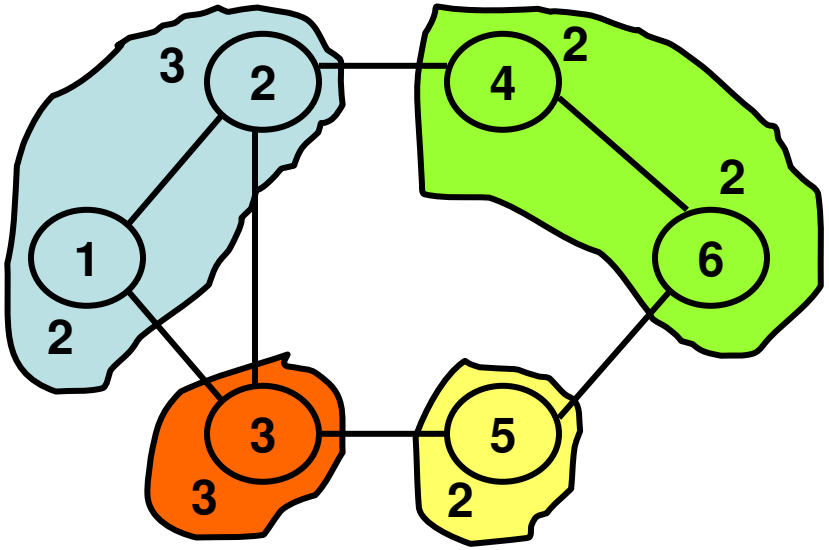
Modularity(1,2,3)
Mod(1,2) = 1 – (2*3)/(2*7) = 0.571
Mod(1,3) = 1 – (2*3)/(2*7) = 0.571
Mod(2,3) = 1 – (3*3)/(2*7) = 0.357

Modularity(4,5,6)
Mod(4,5) = 0 – (2*2)/(2*7) = -0.286
Mod(4,6) = 1 – (2*2)/(2*7) = 0.714
Mod(5,6) = 1 – (2*2)/(2*7) = 0.714

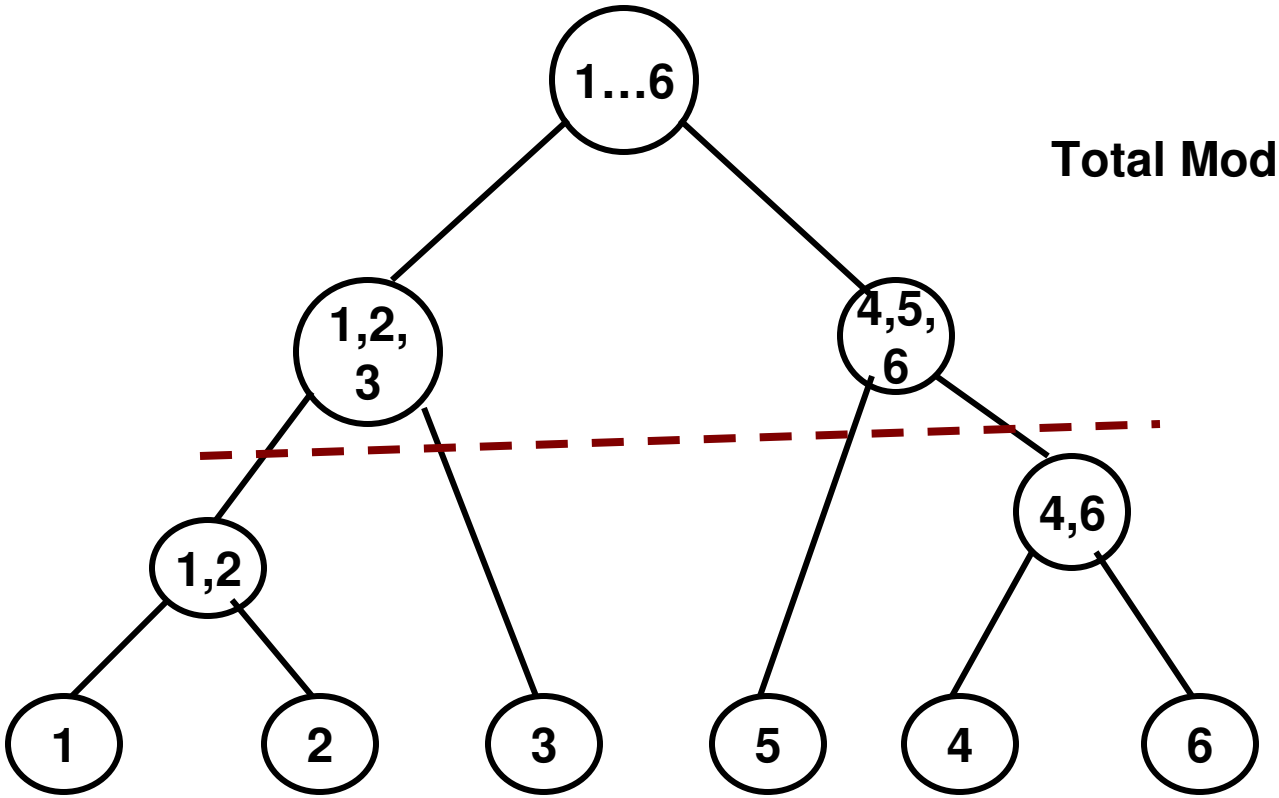


Total Modularity = 2.641

Complete Linkage Clustering



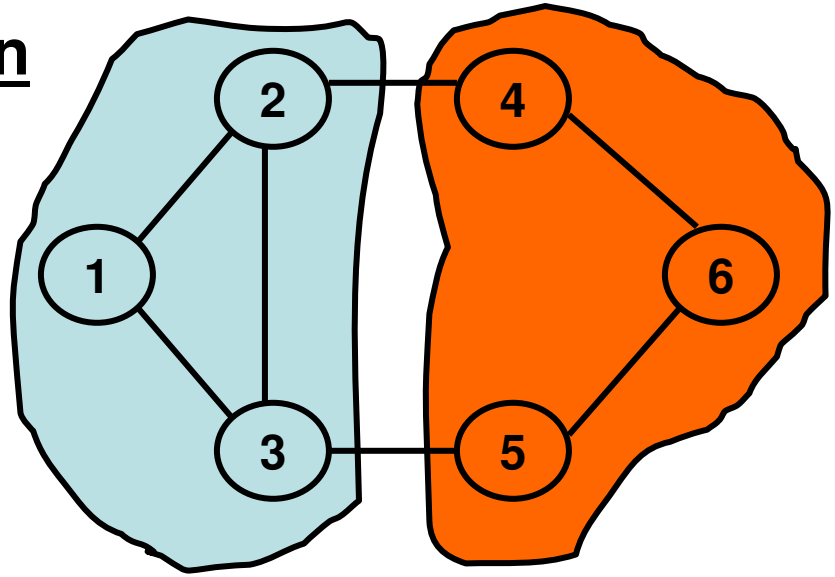
Mod(1,2) = 1 – (2*3)/(2*7) = 0.571
Mod(3) = 0
Mod(5) = 0
Mod(4,6) = 1 – (2*2)/(2*7) = 0.714



Total Modularity = 1.285

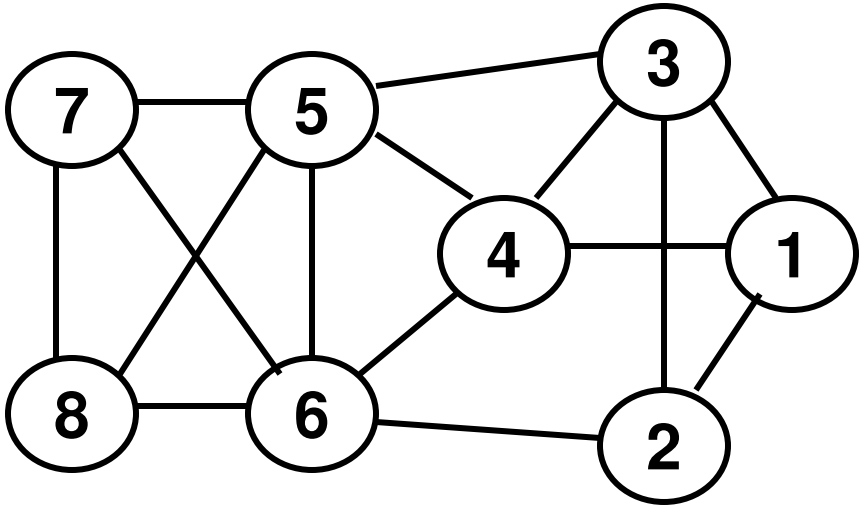
Complete Linkage
Clustering

Final Partition



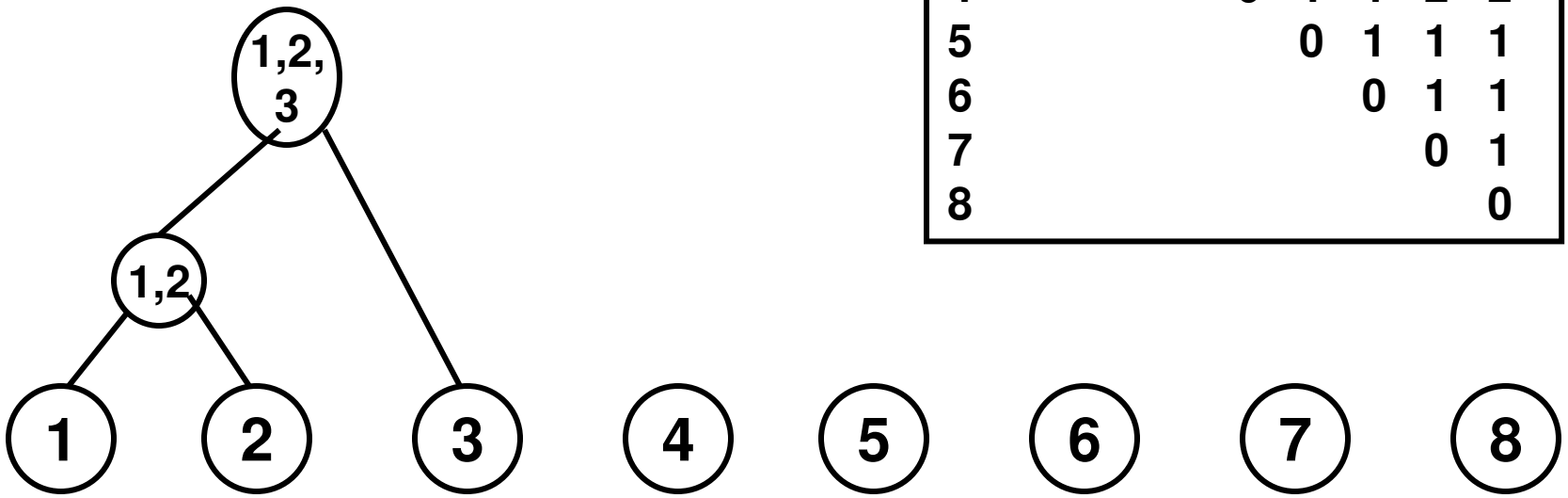
Total Modularity = 2.641

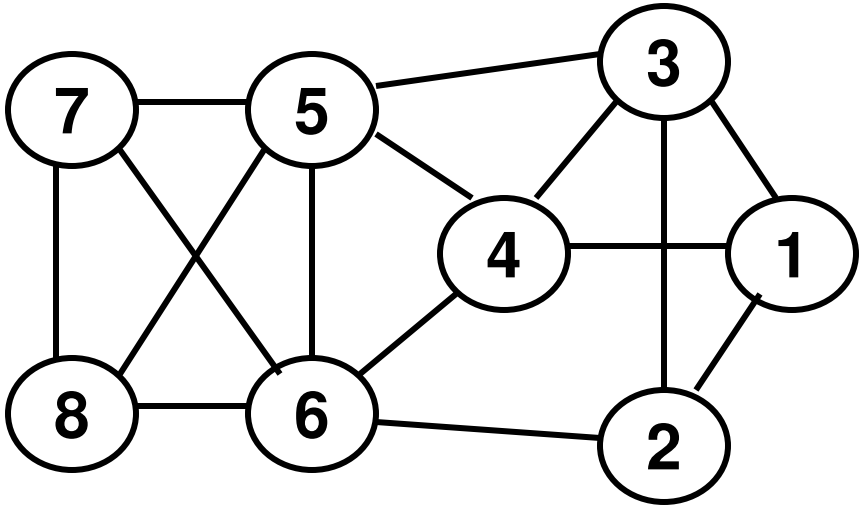
Complete Linkage Clustering (Example 2)



	1	2	3	4	5	6	7	8
1	0	1	1	1	2	2	3	3
2		0	1	2	2	1	2	2
3			0	1	1	2	2	3
4				0	1	1	2	2
5					0	1	1	1
6						0	1	1
7							0	1
8								0

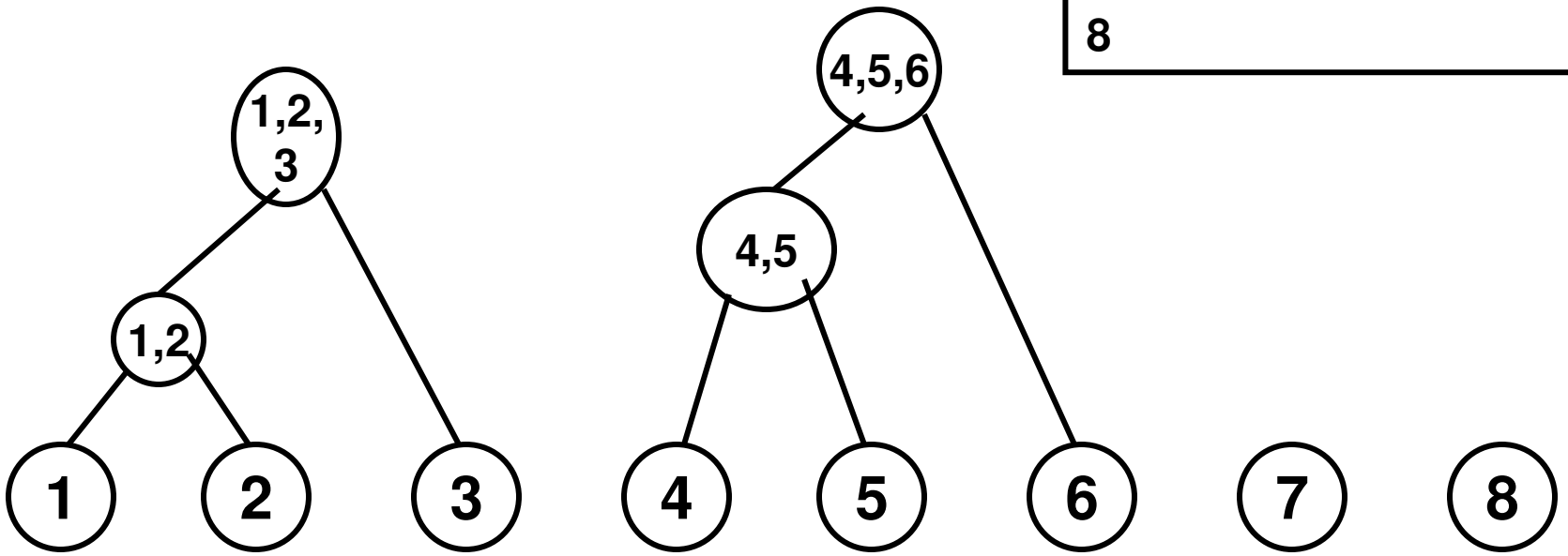
	1,2	3	4	5	6	7	8
1,2	0	1	2	2	2	3	3
3		0	1	1	2	2	3
4			0	1	1	2	2
5				0	1	1	1
6					0	1	1
7						0	1
8							0

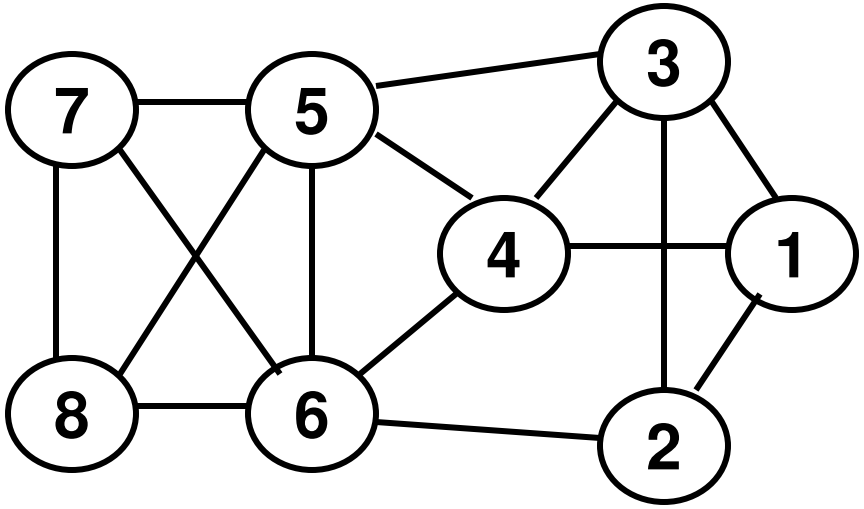




	1,2,3	4	5	6	7	8
1,2,3	0	2	2	2	3	3
4		0	1	1	2	2
5			0	1	1	1
6				0	1	1
7					0	1
8						0

	1,2,3	4,5	6	7	8
1,2,3	0	2	2	3	3
4,5		0	1	2	2
6			0	1	1
7				0	1
8					0

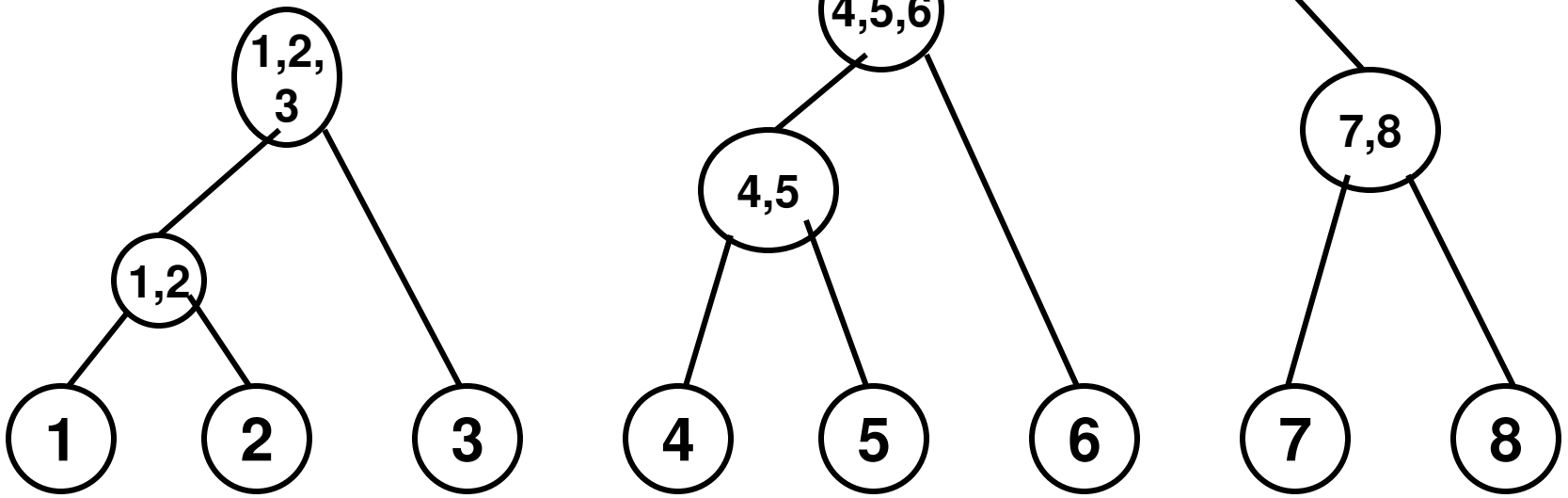


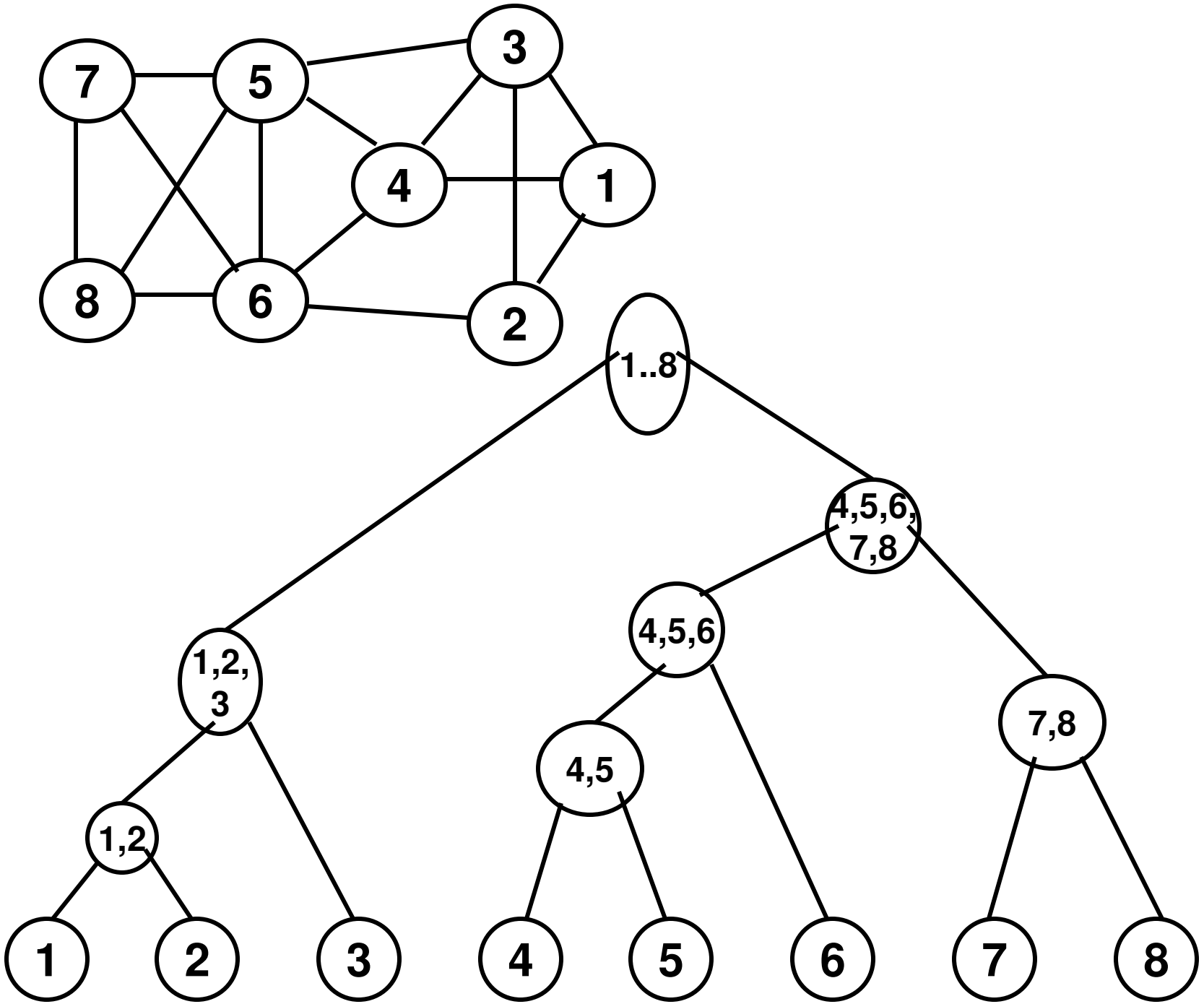


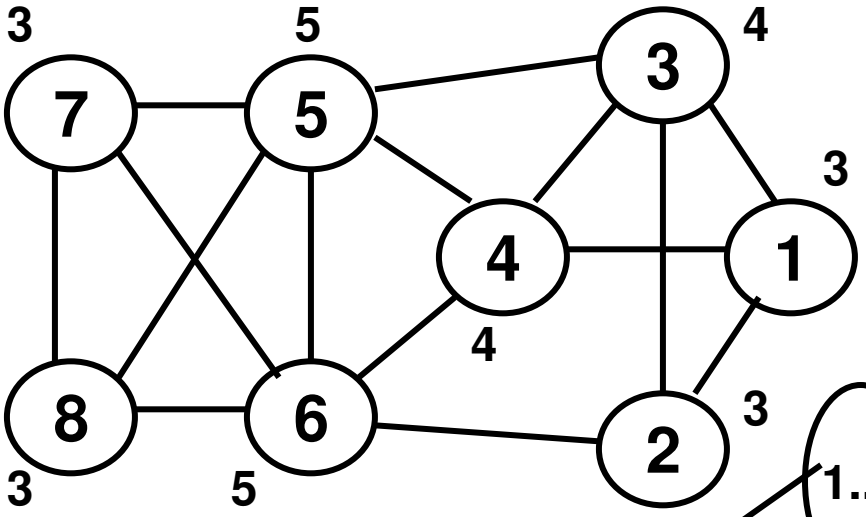
	1,2,3	4,5,6	7	8
1,2,3	0	2	3	3
4,5,6		0	2	2
7			0	1
8				0

	1,2,3	4,5,6	7,8
1,2,3	0	2 (14)	3
4,5,6		0	2 (8)
7,8			0

Break the tie by choosing the Pair with the minimum total Pair-wise distance / Sum of pair size
(1,2,3) and (4,5,6): 14/6 = 2.33
(4,5,6) and (7,8): 8/5 = 1.6

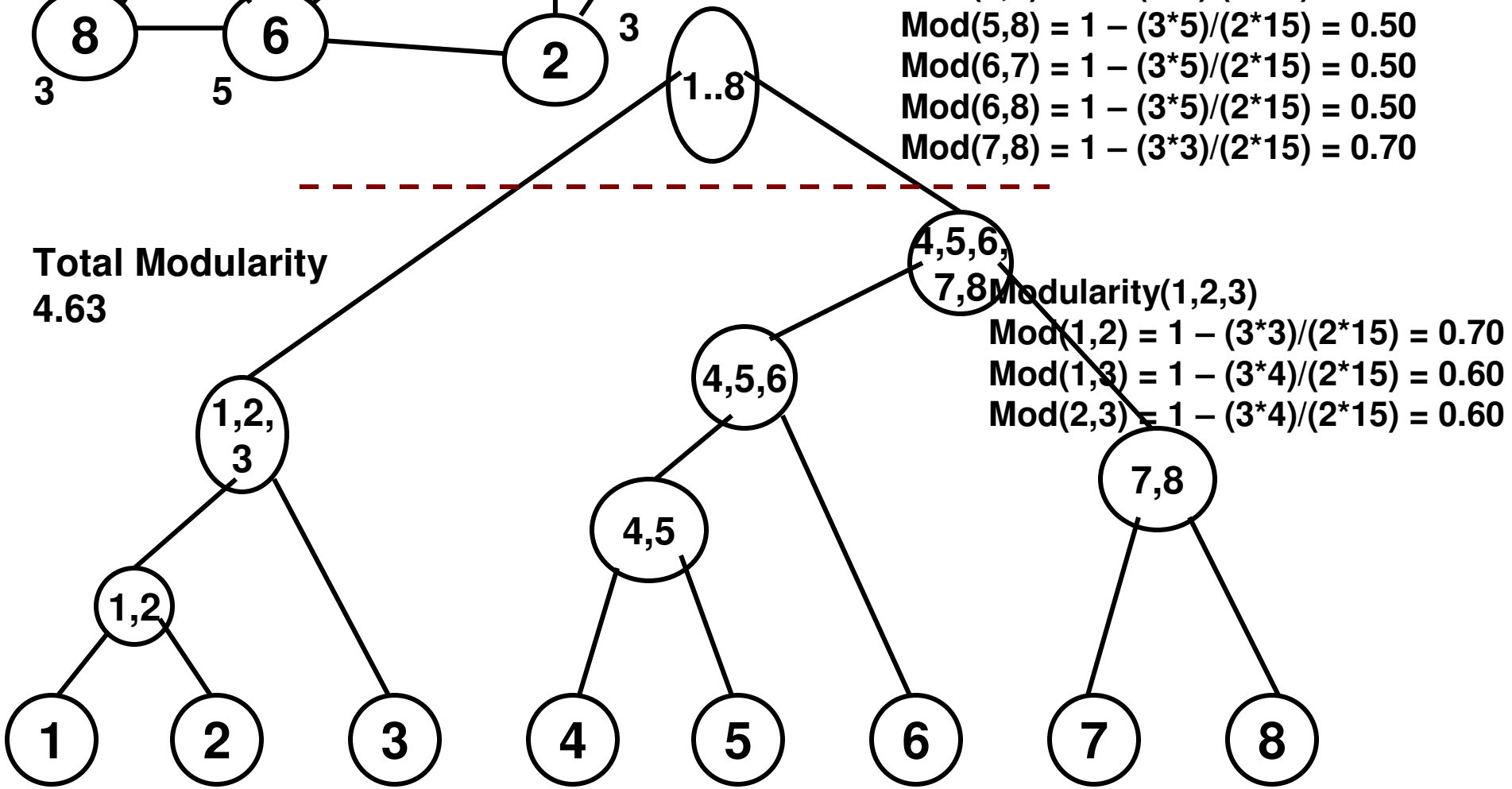



7
5
3
4
1
8
6
2
1..8
4,5,6,
7,8
1,2,
3
4,5,6
7,8
4,5
1,2
1
2
3
4
5
6
7
8



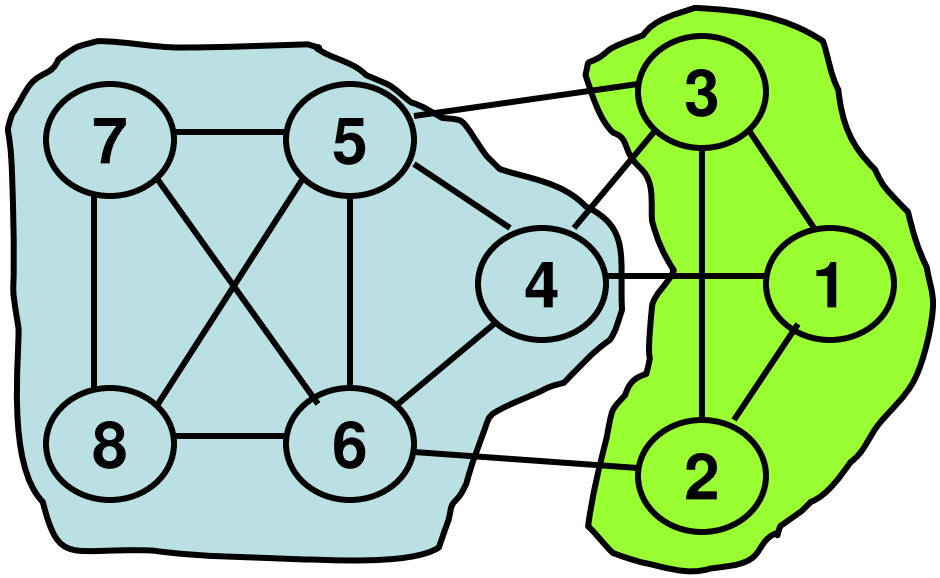
Modularity(4,5,6,7,8)
Mod(4,5) = 1 – (4*5)/(2*15) = 0.33
Mod(4,6) = 1 – (4*5)/(2*15) = 0.33
Mod(4,7) = 0 – (3*4)/(2*15) = -0.4
Mod(4,8) = 0 – (3*4)/(2*15) = -0.4
Mod(5,6) = 1 – (5*5)/(2*15) = 0.17
Mod(5,7) = 1 – (3*5)/(2*15) = 0.50
Mod(5,8) = 1 – (3*5)/(2*15) = 0.50
Mod(6,7) = 1 – (3*5)/(2*15) = 0.50
Mod(6,8) = 1 – (3*5)/(2*15) = 0.50
Mod(7,8) = 1 – (3*3)/(2*15) = 0.70

Total Modularity
4.63

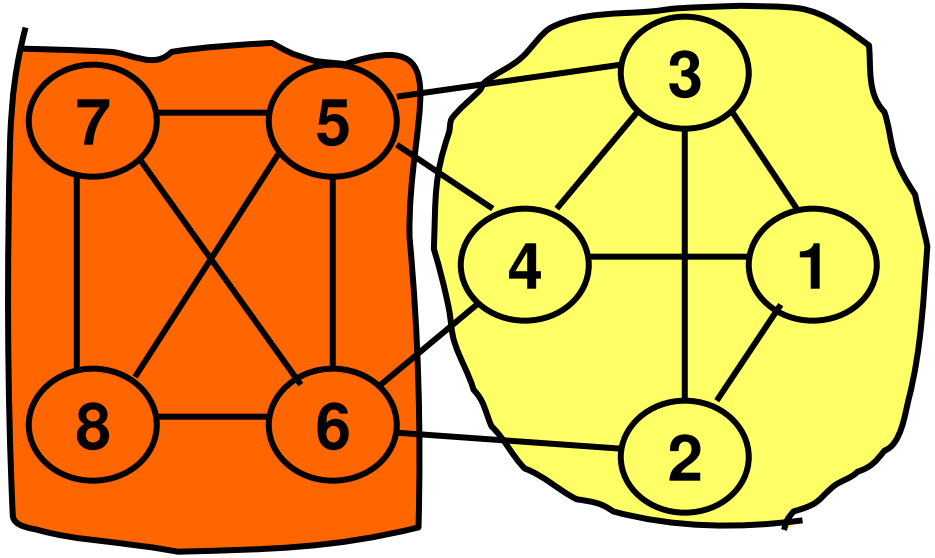


Modularity(1,2,3)
Mod(1,2) = 1 – (3*3)/(2*15) = 0.70
Mod(1,3) = 1 – (3*4)/(2*15) = 0.60
Mod(2,3) = 1 – (3*4)/(2*15) = 0.60

Final Partition



From the previous slides,
We know that the optimal
Partitioning of the graph is:



Total Modularity = 4.63

Modularity (1, 2, 3, 4) +
Modularity (5, 6, 7, 8) = 5.54

Complete Linkage Clustering

Thus, complete linkage clustering need not always give the optimal solution.

Hierarchical Clustering
Edge Betweenness

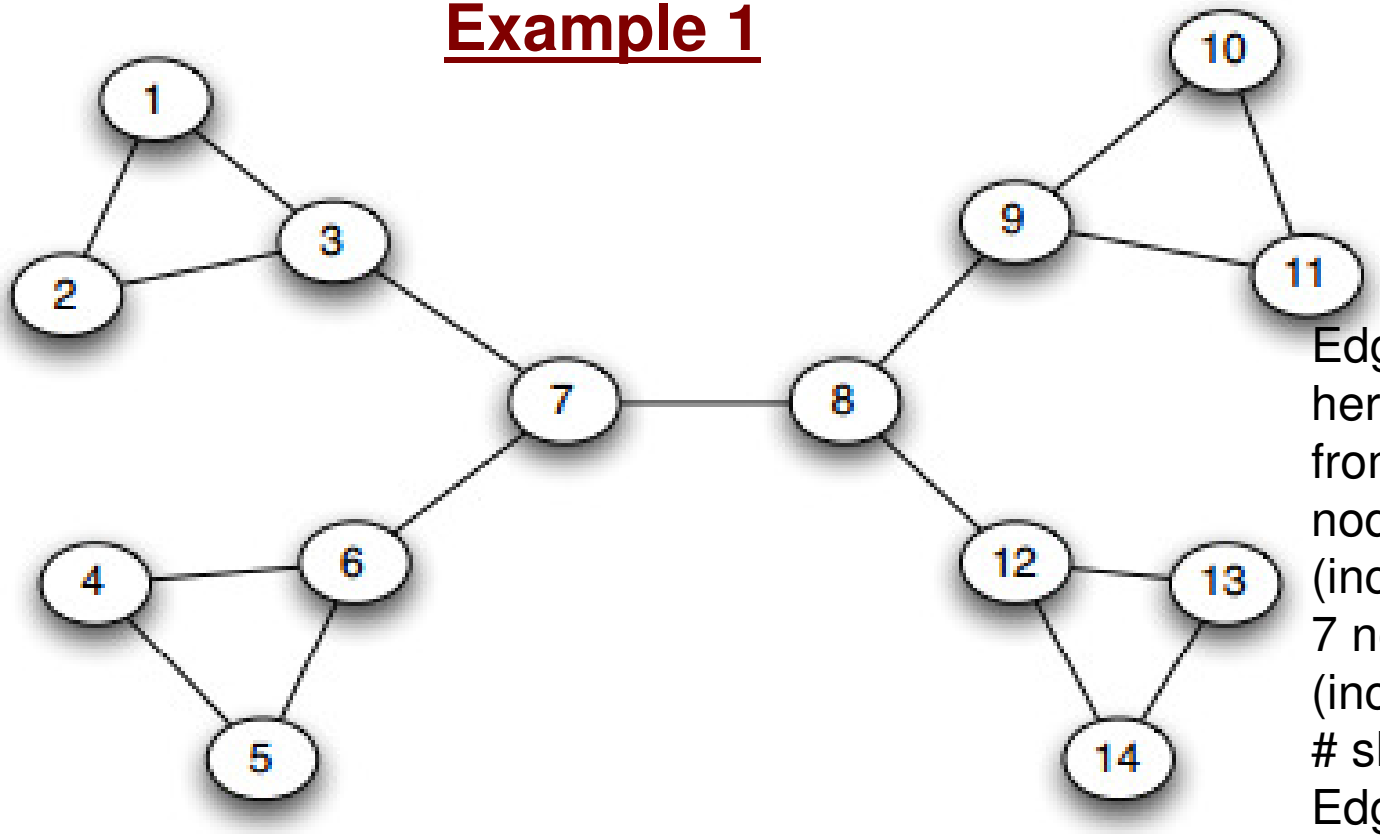
Top-Down Approach

(Divisional)

Edge Betweenness as Flow along the Edge

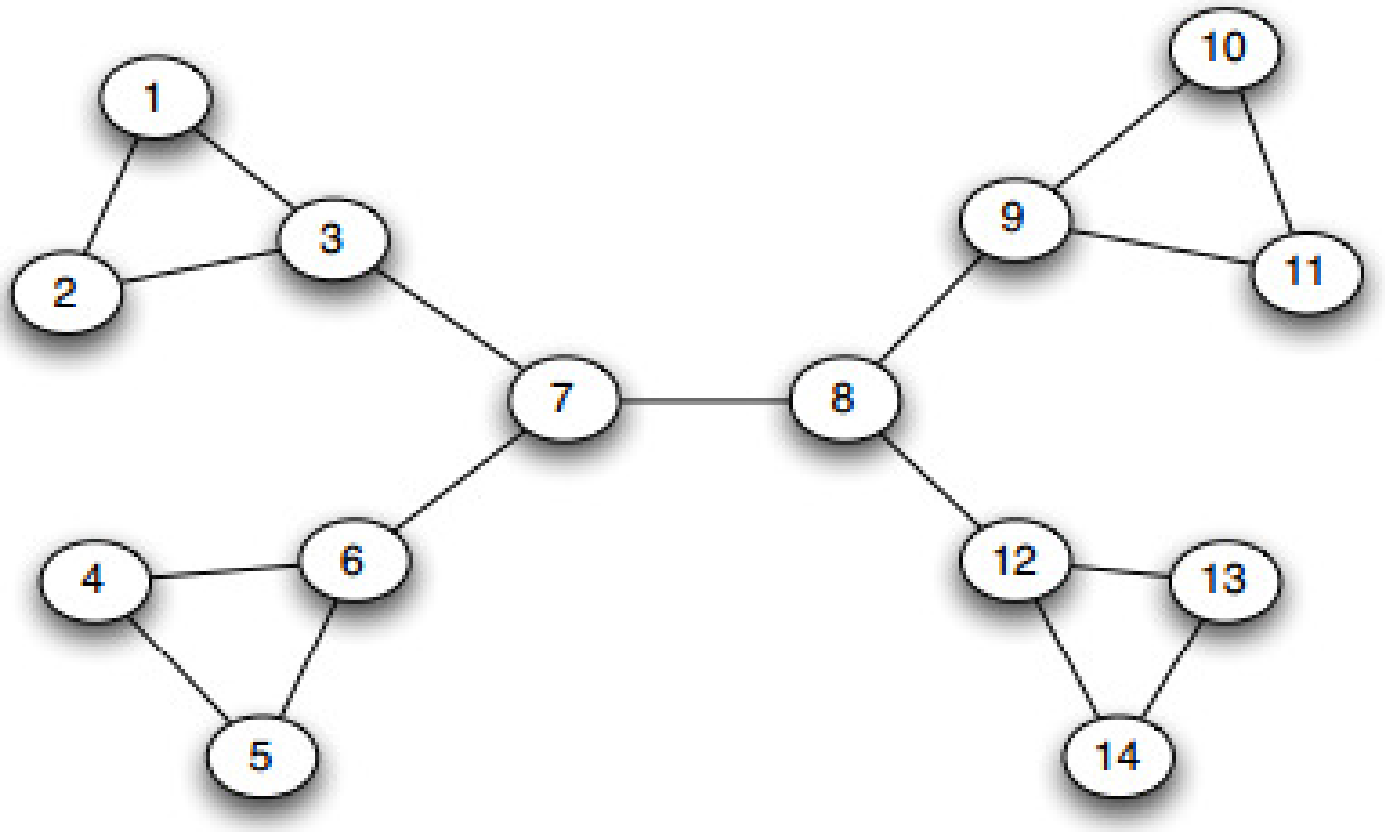
- Here, we model edge betweenness as a measure of the total amount of flow (proportional to the number of shortest paths the edge is part of) it carries, counting flow between all pairs of nodes using this edge.
- Note that in this graph below, there is only one shortest path between any two nodes. Hence, the total amount of flow through an edge is the # shortest paths through that edge

Example 1



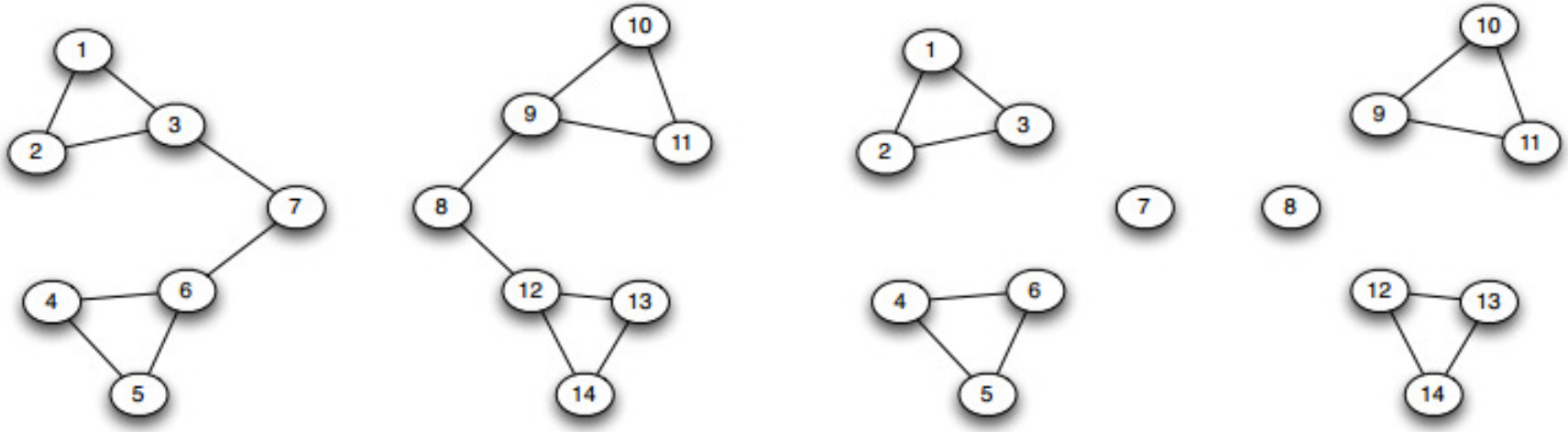
Edge 7-8 in the graph here carries flow from each of the 7 nodes on the left (incl. node 7) and the 7 nodes on the right (incl. node 8)
shortest paths through Edge 7-8 is 7*7 = 49

Edge Betweenness: Motivating Example



Shortest Paths through edge 3 – 7 is: 3 * 11 = 33
Similarly, the # Shortest Paths through edges 6 – 7, 8 – 9 and 8 – 12 are 33 each.
Shortest Paths through edge 1 – 3 is 12. Similarly, the # Shortest Paths through edges 2 – 3, 4 – 6, 5 – 6, 9 – 10, 9 – 11, 12 – 13 and 12 – 14 are 12 each.
Shortest Paths through edge 1 – 2, 4 – 5, 10 – 11 and 13 – 14 are 1 each.

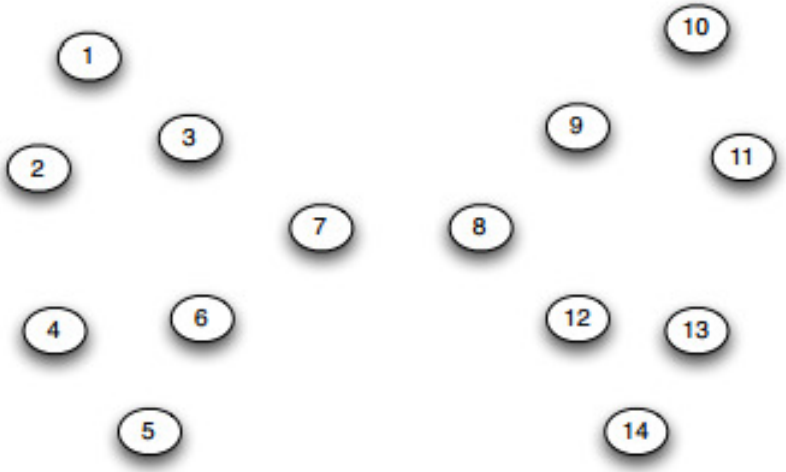
Edge Betweenness: Motivating Example 1



(a) *Step 1*

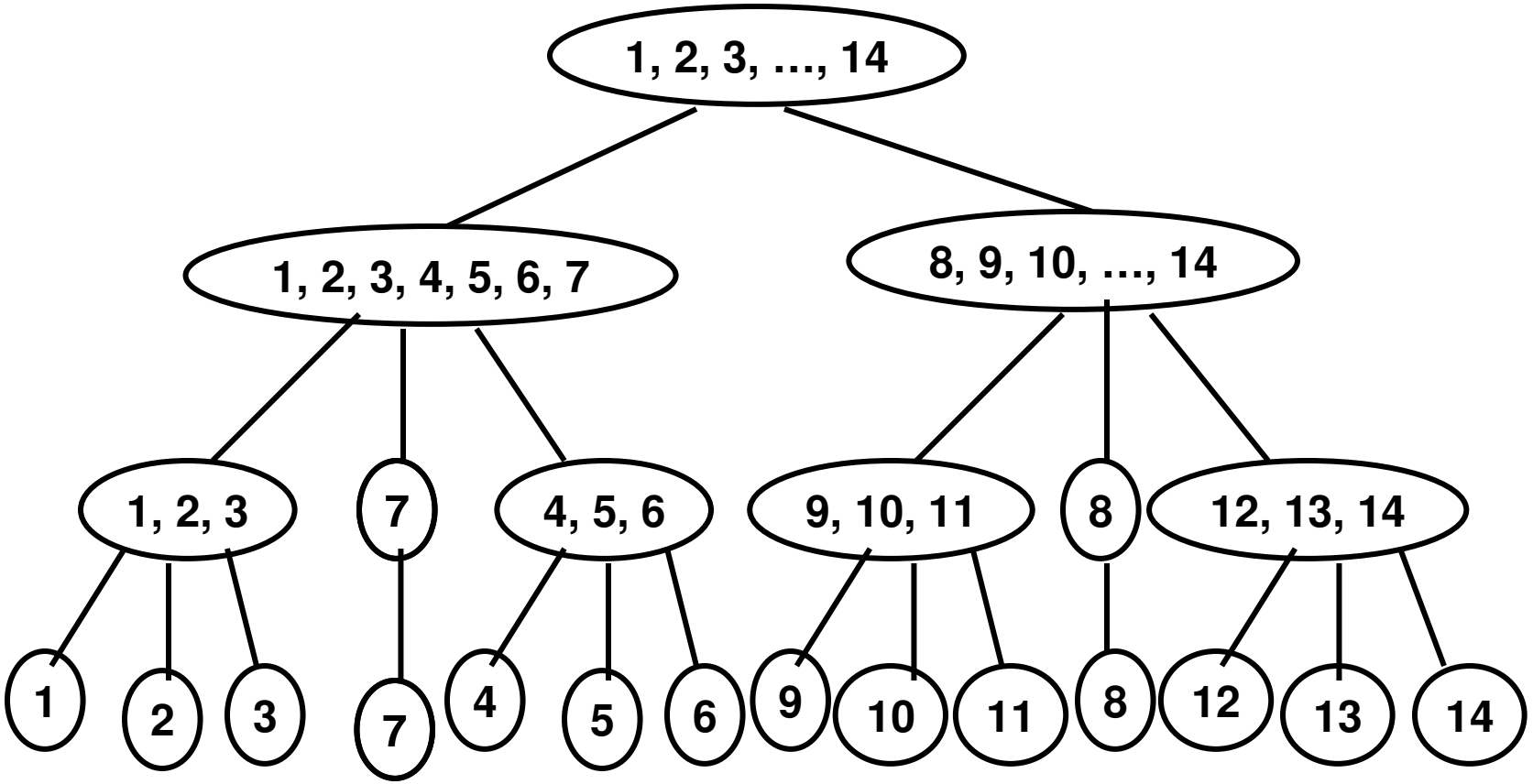
(b) *Step 2*

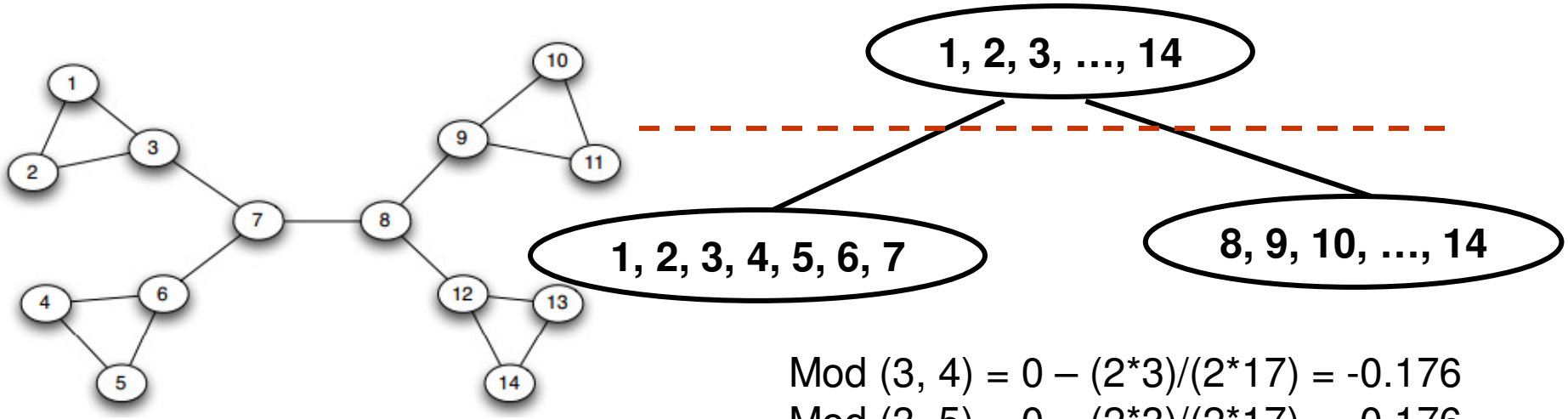
Step 2
Shortest Paths
3 – 7: 3*4 = 12
6 – 7: 3*4 = 12
1 – 2: 1
4 – 5: 1
1 – 3: 5
5 – 6: 5
2 – 3: 5
4 – 6: 5



(c) *Step 3*

Partition Tree

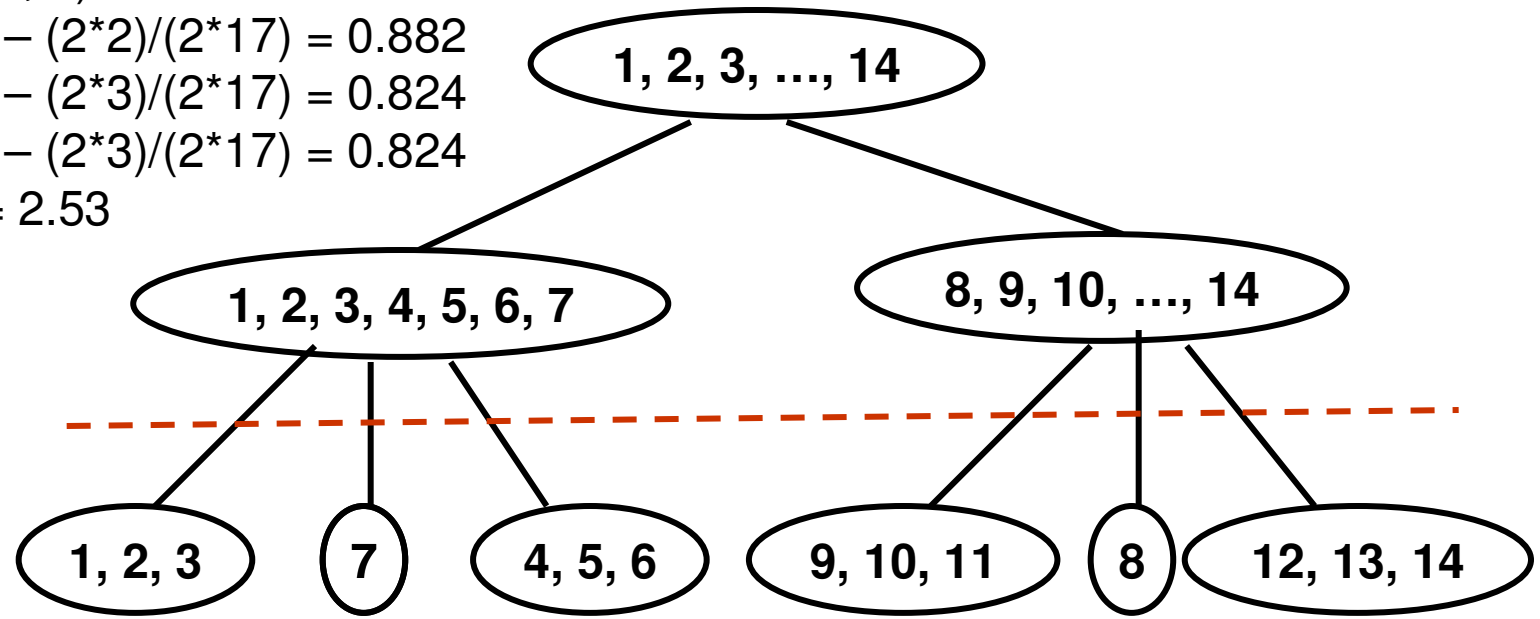


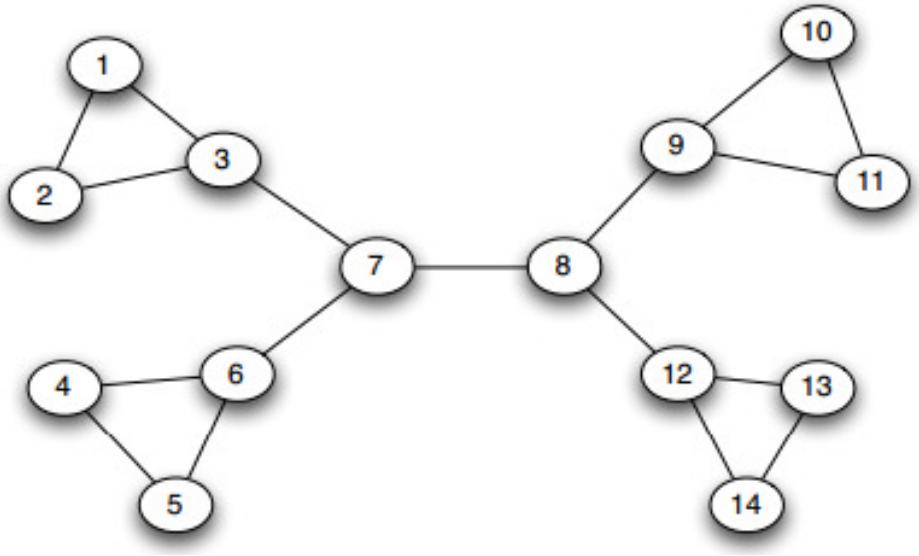


Modularity (1, 2, 3,4 ,5 6, 7)
Mod (1, 2) = 1 – (2*2)/(2*17) = 0.882
Mod (1, 3) = 1 – (2*3)/(2*17) = 0.824
Mod (1, 4) = 0 – (2*2)/(2*17) = -0.118
Mod (1, 5) = 0 – (2*2)/(2*17) = -0.118
Mod (1, 6) = 0 – (2*3)/(2*17) = -0.176
Mod (1, 7) = 0 – (2*3)/(2*17) = -0.176
Mod (2, 3) = 1 – (2*3)/(2*17) = 0.824
Mod (2, 4) = 0 – (2*2)/(2*17) = -0.118
Mod (2, 5) = 0 – (2*2)/(2*17) = -0.118
Mod (2, 6) = 0 – (2*3)/(2*17) = -0.176
Mod (2, 7) = 0 – (2*3)/(2*17) = -0.176
Mod (3, 4) = 0 – (2*3)/(2*17) = -0.176
Mod (3, 5) = 0 – (2*3)/(2*17) = -0.176
Mod (3, 6) = 0 – (3*3)/(2*17) = -0.265
Mod (3, 7) = 1 – (3*3)/(2*17) = 0.735
Mod (4, 5) = 1 – (2*2)/(2*17) = 0.882
Mod (4, 6) = 1 – (2*3)/(2*17) = 0.824
Mod (4, 7) = 0 – (2*3)/(2*17) = -0.176
Mod (5, 6) = 1 – (2*3)/(2*17) = 0.824
Mod (5, 7) = 0 – (2*3)/(2*17) = -0.176
Mod (6, 7) = 1 – (3*3)/(2*17) = 0.735

Modularity(1, 2, 3, 4, 5 , 6, 7) = 4.385
Modularity(8, 9, 10, …, 14) = 4.385
Total Modularity = 8.77

Modularity (1, 2, 3)
Mod (1, 2) = 1 – (2*2)/(2*17) = 0.882
Mod (1, 3) = 1 – (2*3)/(2*17) = 0.824
Mod (2, 3) = 1 – (2*3)/(2*17) = 0.824
Mod (1, 2, 3) = 2.53



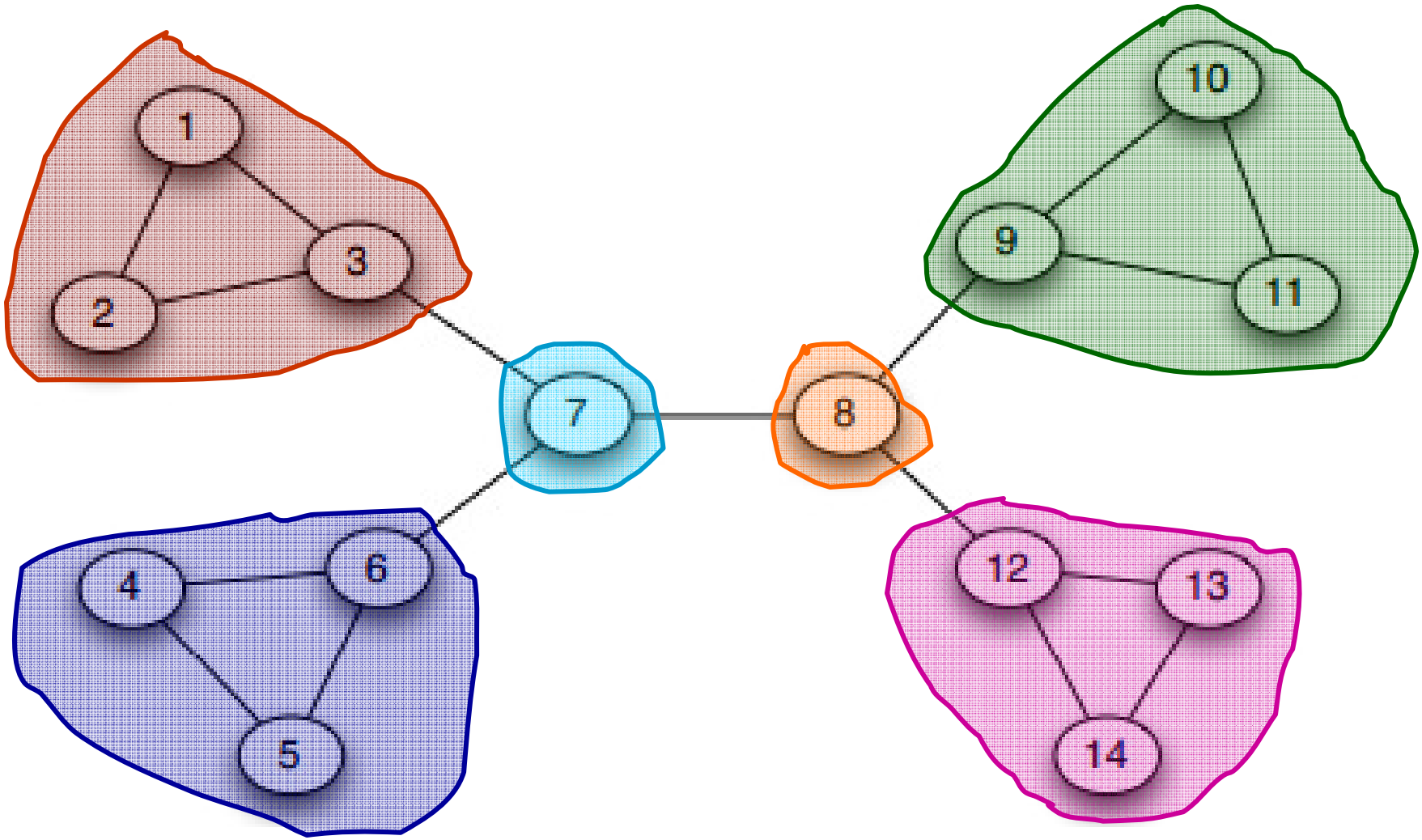


Due to symmetry,
Modularity (4, 5, 6) = 2.53
Modularity (9, 10, 11) = 2.53
Modularity (12, 13, 14) = 2.53

Total Modularity = 10.12

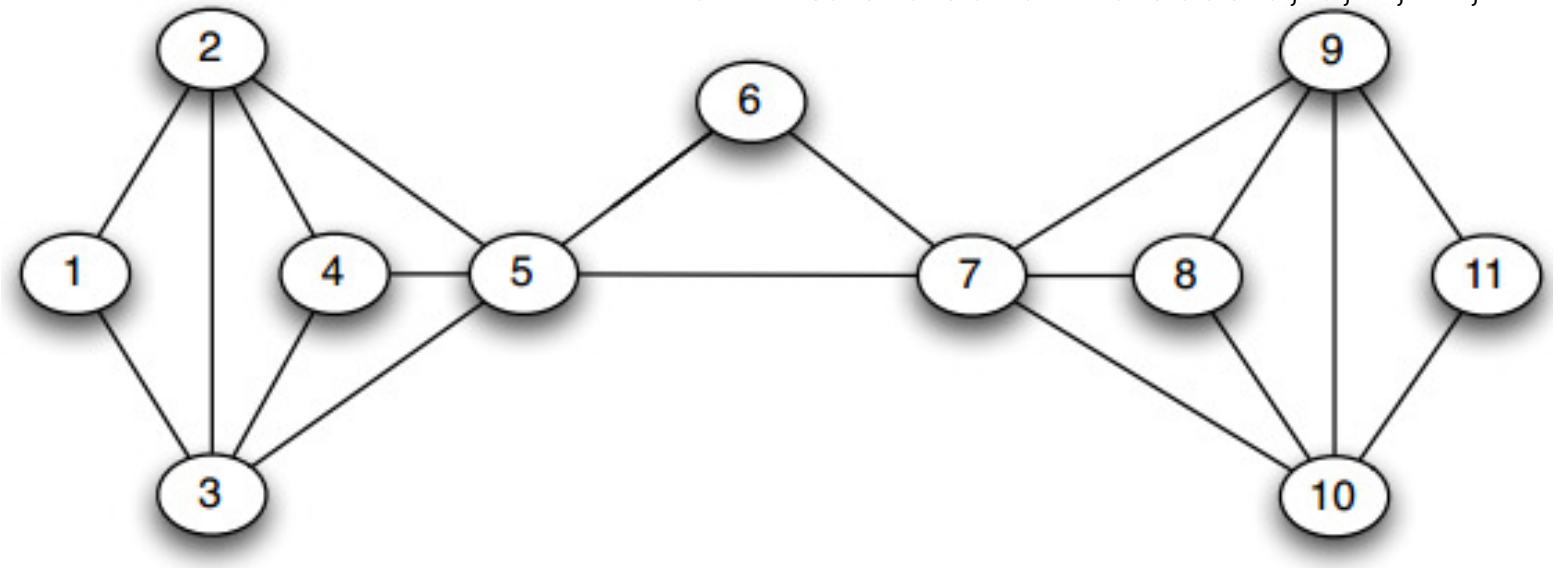
Example 1

Final Partitioning into Communities

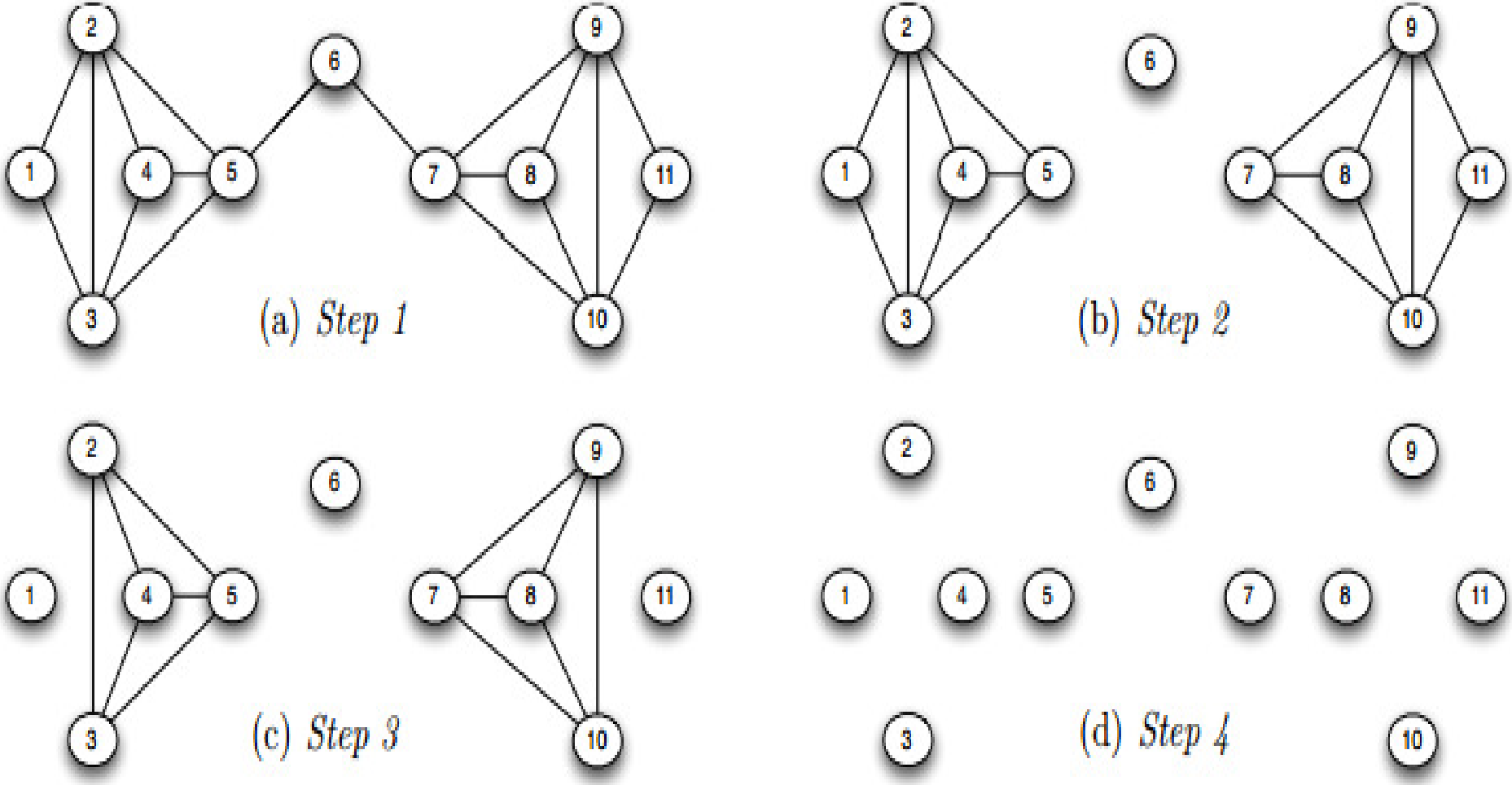


Example 2

Note: For an edge like 2 – 5, it could be part of shortest paths from 1 to 5, 6, 7, …, 11. But, there are also shortest paths from 1 to 5, 6, , …, 11 through edge 3 – 5. Hence, the net # shortest paths on 2 – 5 from 1 to the seven vertices 5, 6, 7, …, 11 is 7/2 = 3.5



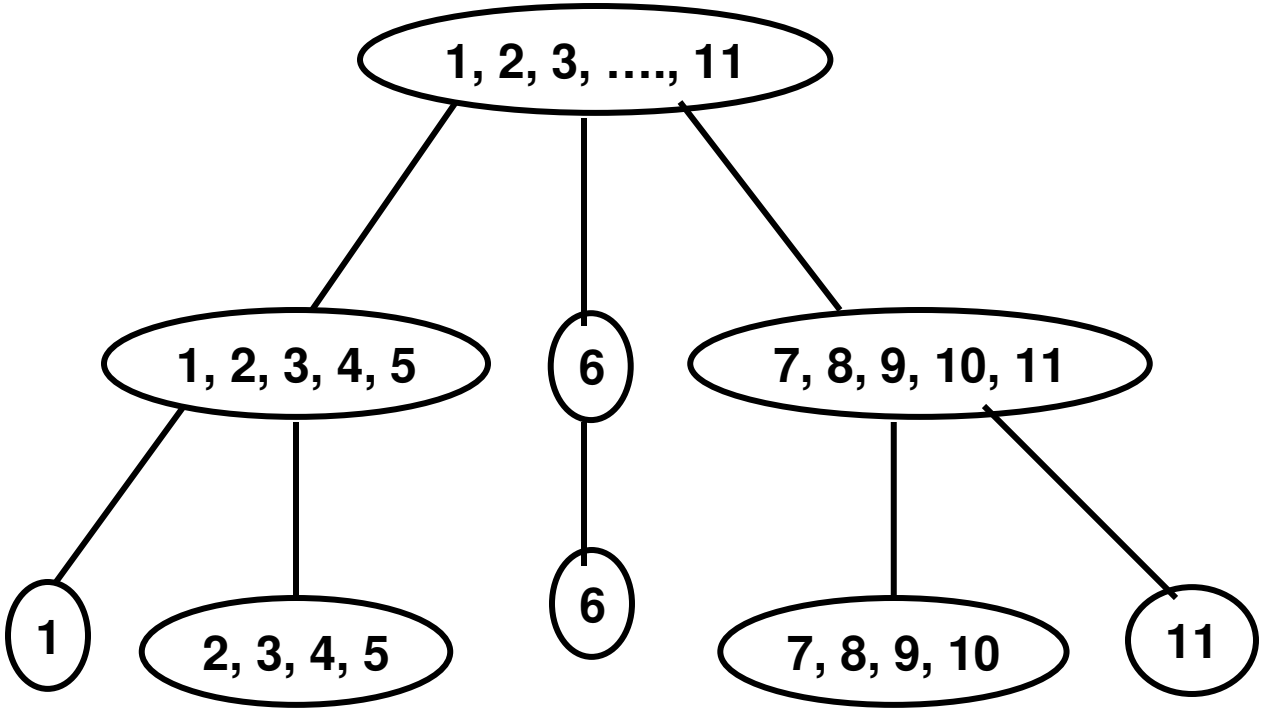
Shortest Paths through edge 5 – 7 is: 5 * 5 = 25
Shortest Paths through edge 5 – 6 (from 1, 2, .., 5 to 6) and 6 – 7 is only 5 each
Shortest Paths through edge 2 – 5 is: 7 (from 2 to 5..11) + 3.5 (from 1 to 5…11) = 10.5
Similarly, # Shortest Paths through edge 3 – 5, 7 – 9 , 7 – 10 are 10.5 each.
Shortest Paths through 4 – 5 (4 to 5, 6, …, 11) and 7 – 8 are 7 each
Shortest Paths through 2 – 4, 3 – 4, 8 – 9 and 8 – 10 are only 1 each
Shortest Paths through 2 – 3 and 9 – 10 are only 1 each
Shortest Paths through 1 – 2 is 1 + 4.0 (1/2 for each of 4, 5, …, 11) = 5.0
Shortest Paths through 1 – 3, 9 – 11, 10 – 11 are also 5 each.

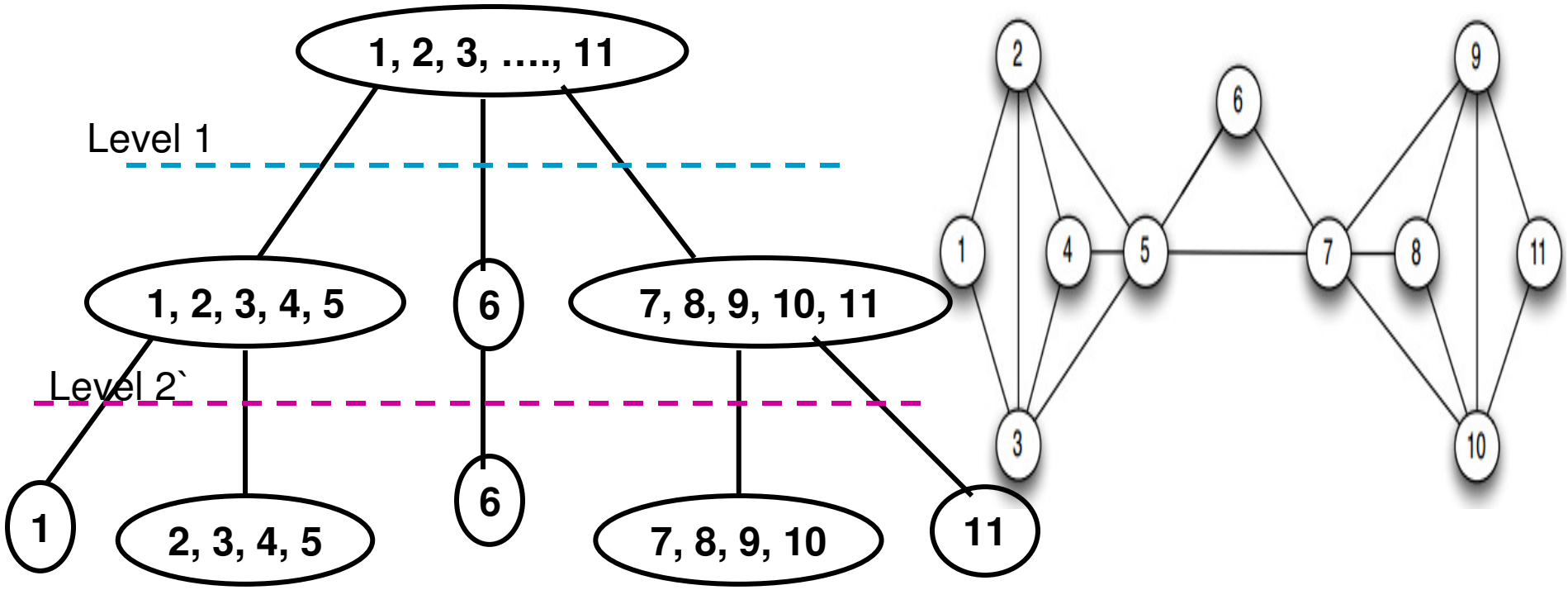


Note that after Step 1, the # shortest paths through edge 5 – 6 becomes: 5 * 6 = 30
Likewise, the # shortest paths through edge 6 – 7 becomes: 5 * 6 = 30

After Step 2, the # shortest paths through edge 1 – 2 is: 1 (1 to 2) + 1 (1/2 from 1 to 4 and 5) = 2.0. Similarly, the # Shortest paths through edge 1 – 3 is 2.0
The # shortest paths through edge 2 – 4 is 1 + ½ (from 1 to 4) = 1.5
Hence, edges 1 – 2 and 1 – 3 have high betweenness (2 each) than
edges 2 – 4 and 3 – 4 (1.5 each)

Partition Tree





Modularity(1, 2, 3, 4, 5)
Mod (1, 2) = 1 – (2*4)/(2*19) = 0.789
Mod (1, 3) = 1 – (2*4)/(2*19) = 0.789
Mod (1, 4) = 0 – (2*3)/(2*19) = -0.158
Mod (1, 5) = 0 – (2*5)/(2*19) = -0.263
Mod (2, 3) = 1 – (4*4)/(2*19) = 0.579
Mod (2, 4) = 1 – (4*3)/(2*19) = 0.684
Mod (2, 5) = 1 – (4*5)/(2*19) = 0.474
Mod (3, 4) = 1 – (3*4)/(2*19) = 0.684
Mod (3, 5) = 1 – (4*5)/(2*19) = 0.474
Mod (4, 5) = 1 – (3*5)/(2*19) = 0.605

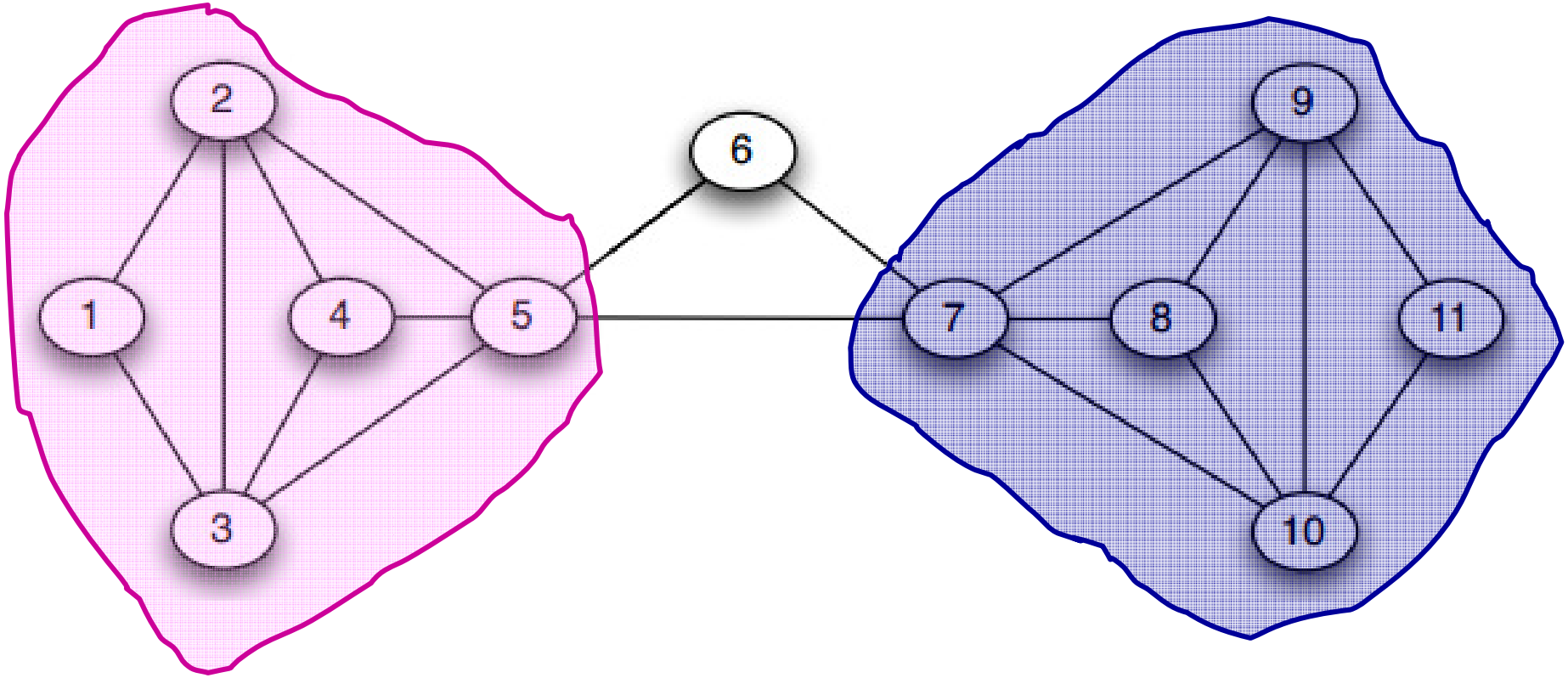
Modularity(1, 2, 3, 4, 5) = 4.657
Similarly, Modularity(7, 8, 9, 10, 11) = 4.657
Total Modularity (Level 1) = 9.314

Modularity (2, 3, 4, 5) = 3.5
Modularity (7, 8, 9, 10) = 3.5
Total Modularity (Level 2) = 7.0

Total Modularity (Level 1) > Total Modularity (Level 2)

Hence, we will go with Level 1 partitioning

Final Partitioning

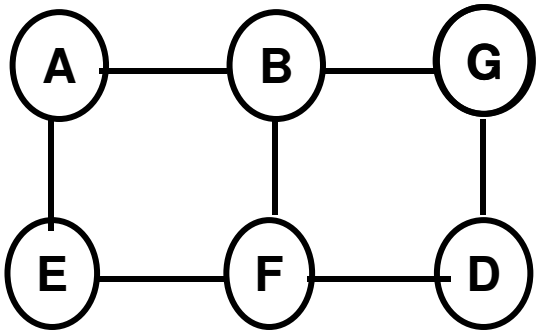


Finding the # Shortest Paths through an Edge

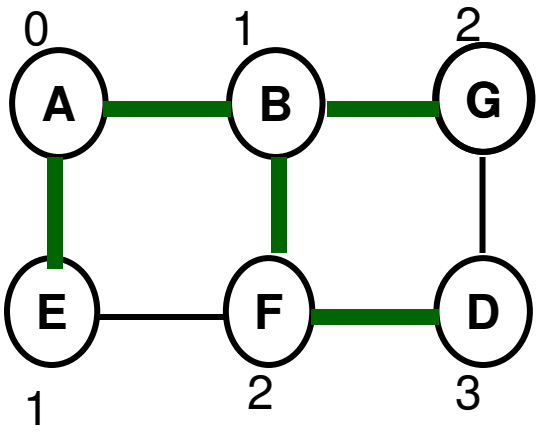
- For graphs in which there is more than one paths between one or more pair of vertices, the total flow through an edge is not equal to the total # shortest paths through the edge.

- We will now see an algorithm proposed by Girvan and Newman to determine the total flow through an edge.
- Repeat the following for every vertex
  - Perform a Breadth First Search (BFS) of the graph, starting from the first vertex, say A.
  - Determine the # shortest paths from A to each other node using the BFS levels of the nodes
  - Based on the above numbers of shortest paths, determine the amount of flow from A to all the other vertices that uses each edge.

- The total flow through an edge is the sum (for directed graph) or half of the sum (for undirected graph) of the flows determined through that edge when BFS is run from every vertex in the graph.
  - For undirected graph, we divide by the total sum of the flows by 2 because an edge is counted twice on the shortest path between any two vertices.
  - For example A – B – C; the edge A – B is counted twice (once on the shortest path from A to C and once on the shortest path from C to A



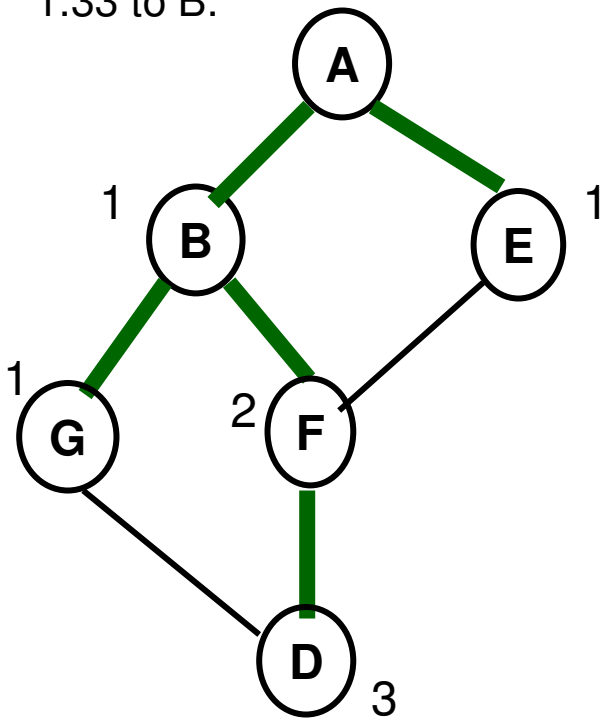
BFS run on A



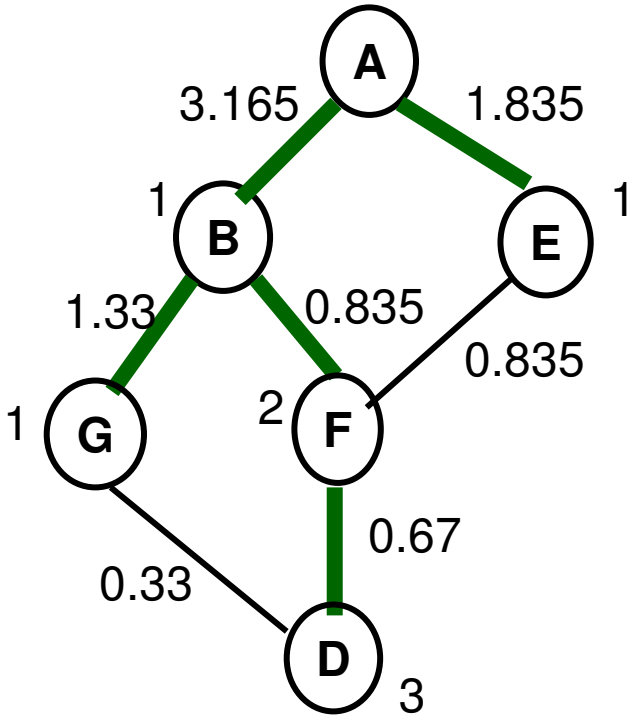
Node Levels

Computing the Flow Values

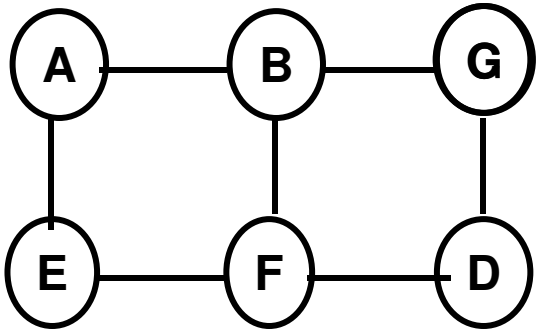
We assume one unit of flow originates at each node. We start with the node at the bottom most level. Let 1 unit of flow start from node D. Node D gets 2 of its Shortest paths to node A through F and 1 through G. So, node A sends 2/3 of the flow to F and 1/3 of the flow to G. Node F adds 2/3 flow received to 1 unit of flow originating at itself and splits the resulting 1.67 equally and sends 0.835 to each of B and E. G merely adds the 0.33 flow units to the 1 units of flow originating at itself and sends 1.33 to B.



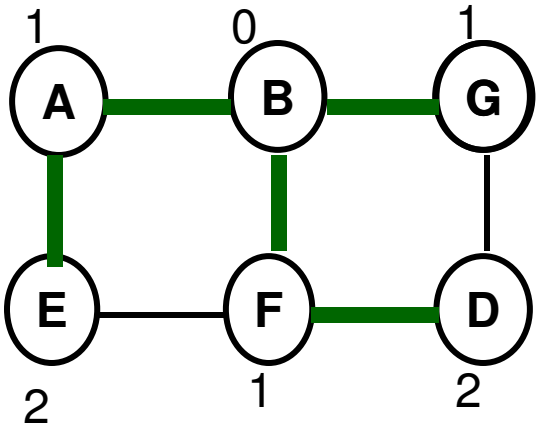
 # Shortest Paths from Node A to every other Node



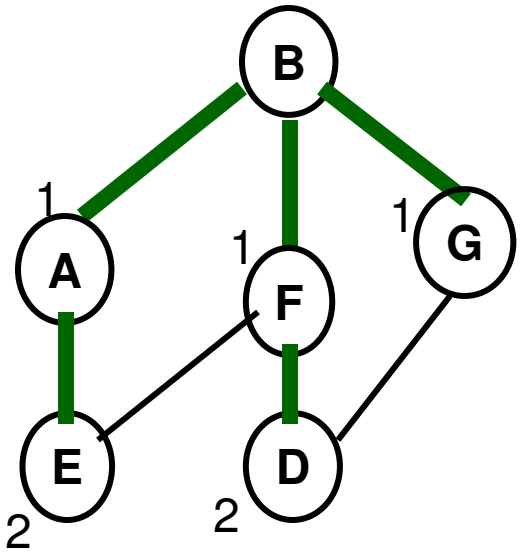
Flow on each edge



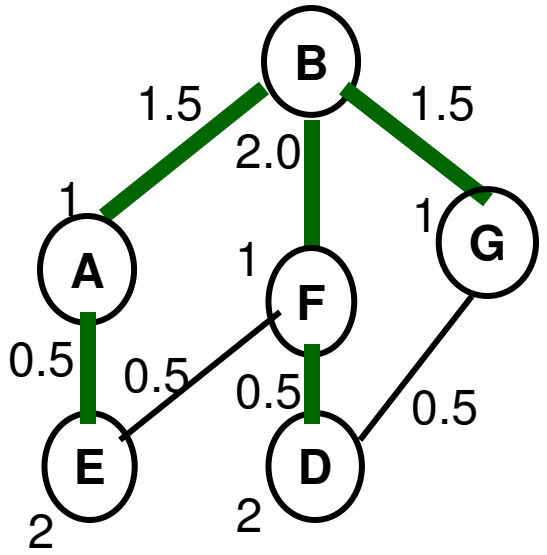
BFS run on B



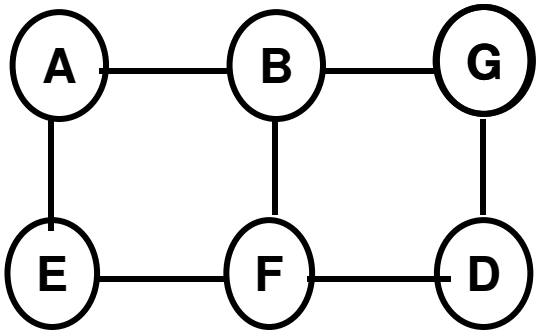
Node Levels



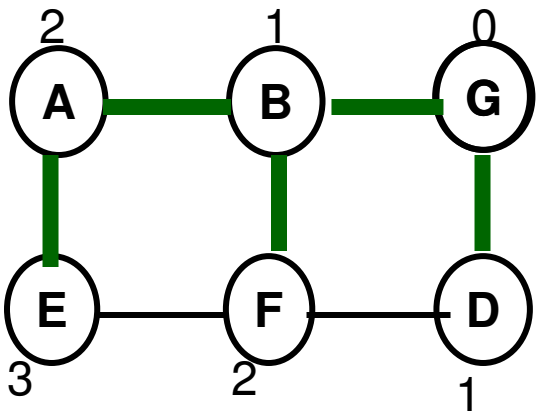
 # Shortest Paths from Node B to every other Node



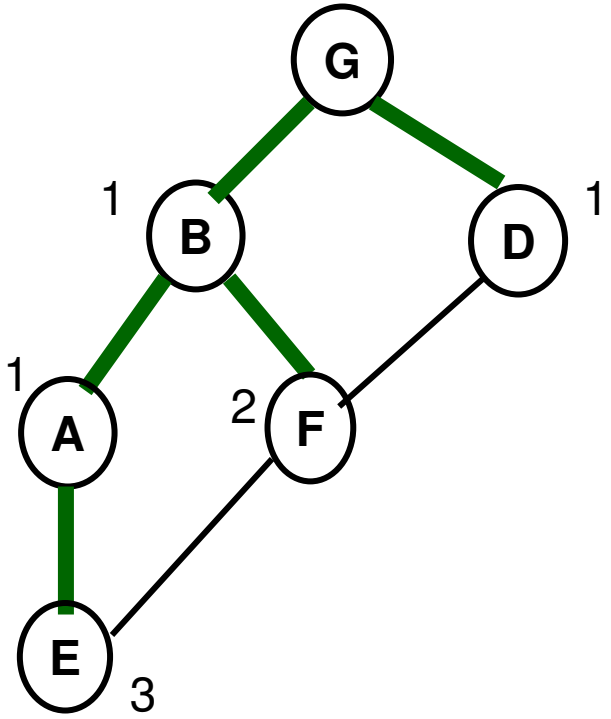
Flow on each edge



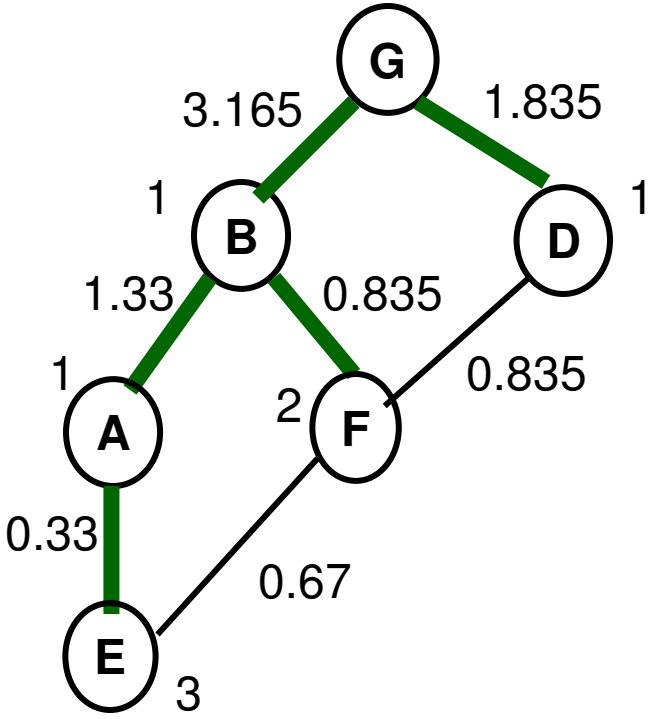
BFS run on G



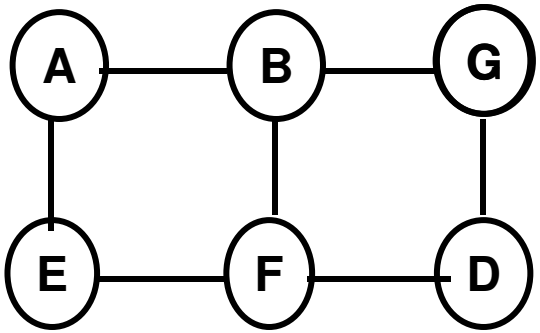
Node Levels



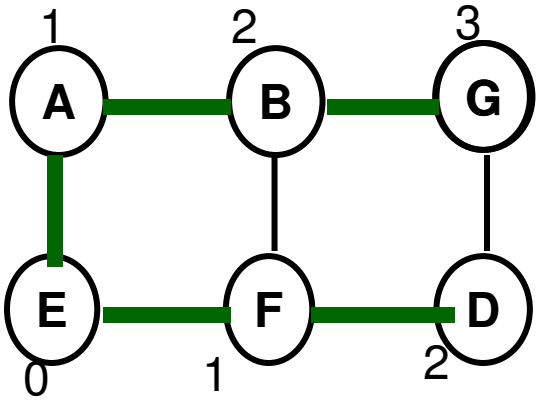
 # Shortest Paths from Node G to every other Node



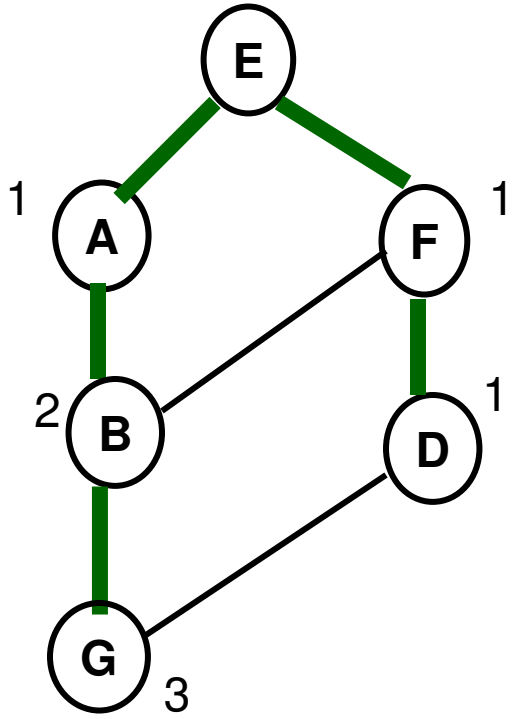
Flow on each edge



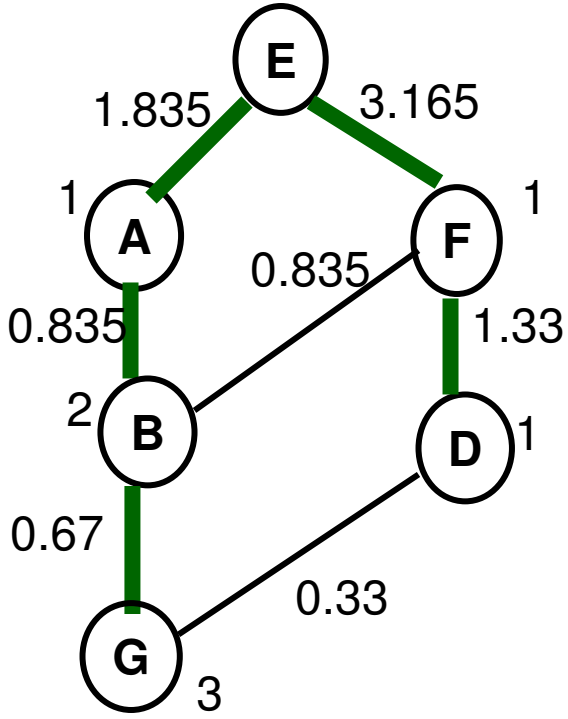
BFS run on E



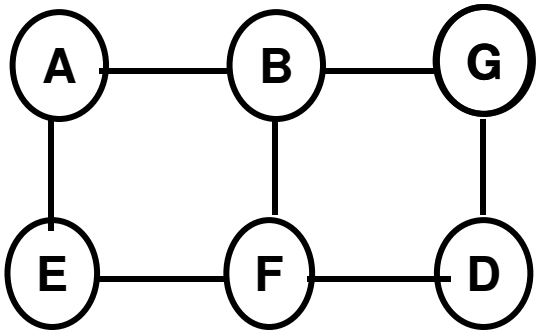
Node Levels



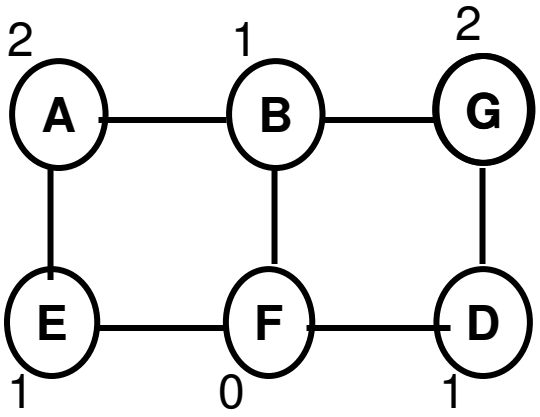
 # Shortest Paths from Node E to every other Node



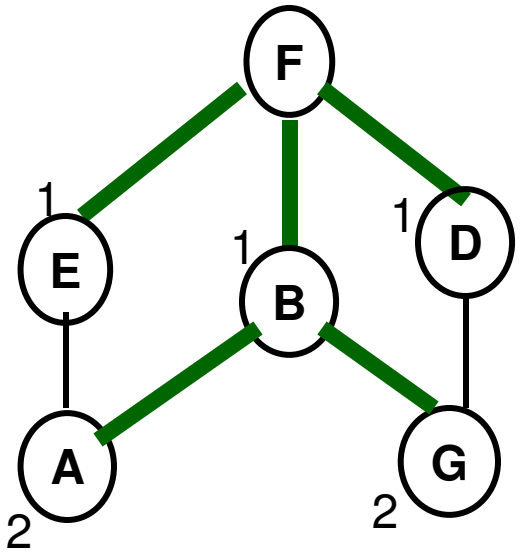
Flow on each edge



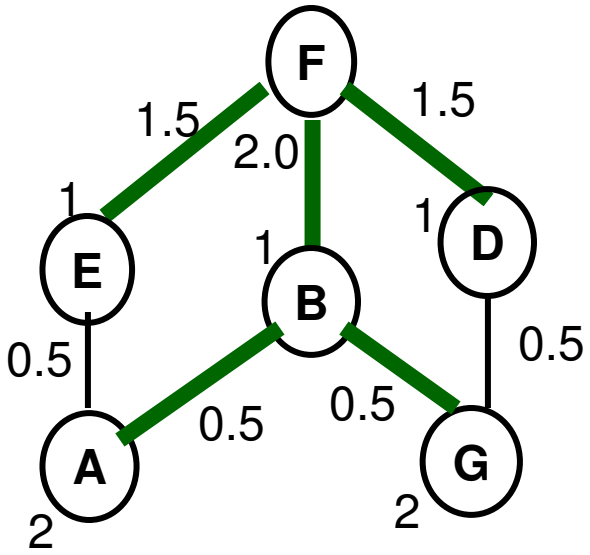
BFS run on F



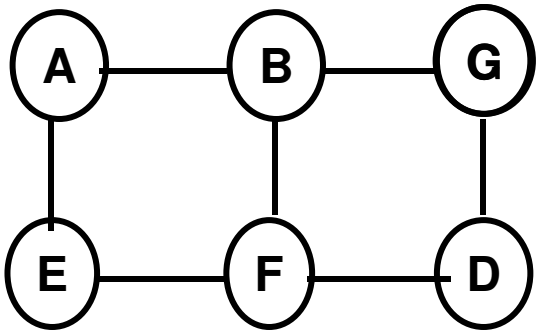
Node Levels



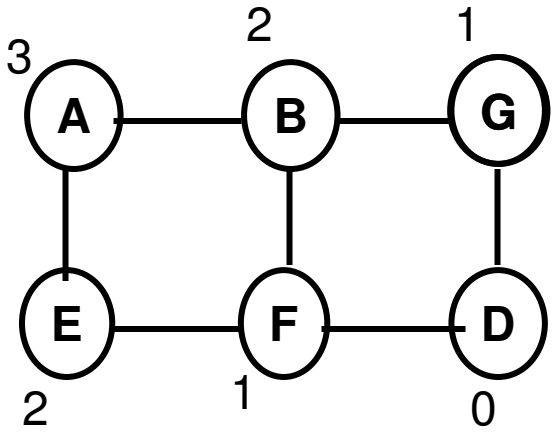
 # Shortest Paths from Node F to every other Node



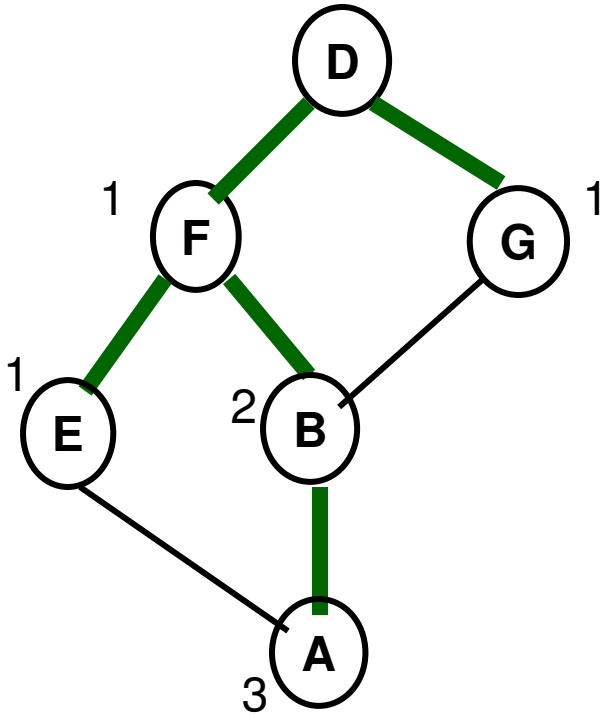
Flow on each edge



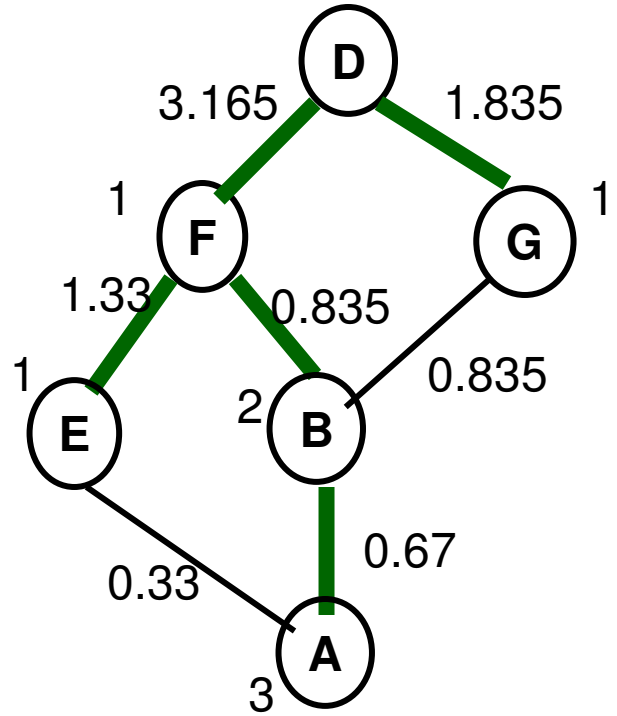
BFS run on D



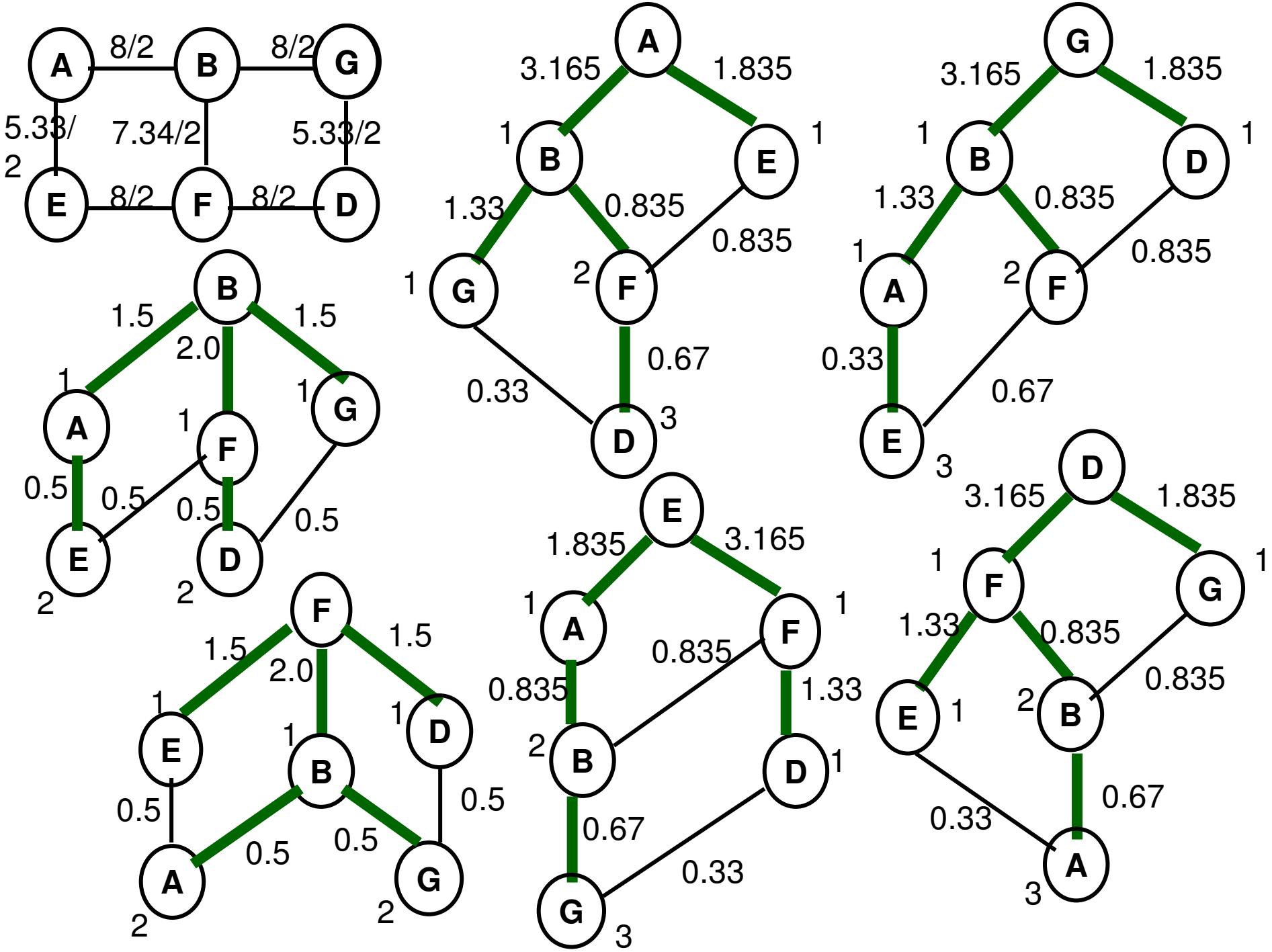
Node Levels



 # Shortest Paths from Node D to every other Node



Flow on each edge


A
B
G
E
F
D
8/2
8/2
5.33/
2
7.34/2
5.33/2
8/2
8/2
3.165
1.835
1
1
1.33
0.835
0.835
1
2
0.67
0.33
3
1.5
1.5
2.0
1
1
1
0.5
0.5
0.5
0.5
2
2
3.165
1.835
1
1
1.33
0.835
0.835
1
2
0.33
0.67
3
1.835
3.165
1
1
0.835
0.835
1.33
2
1
0.67
0.33
3
1.5
1.5
2.0
1
1
1
0.5
0.5
0.5
0.5
2
2
3.165
1.835
1
1
1.33
0.835
0.835
1
2
0.67
0.33
3

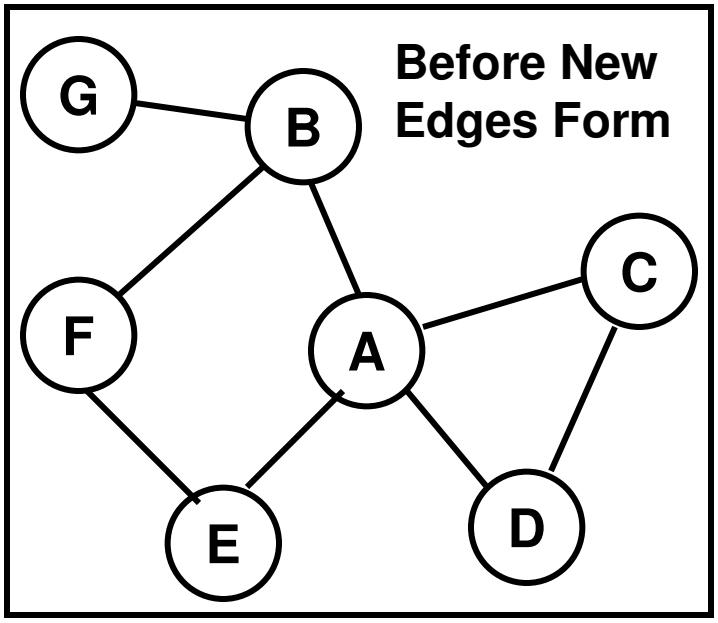
Analysis of: Girvan and Newman Algorithm

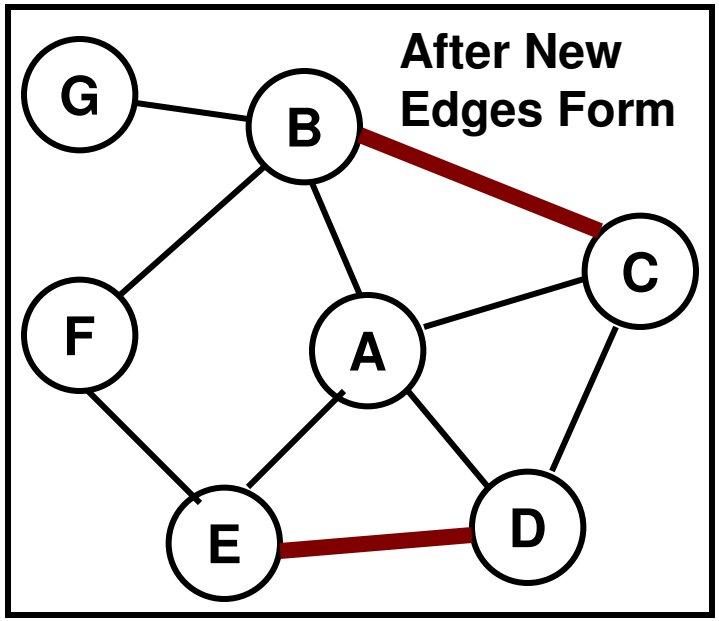
- After we get the total flow through each edge (betweenness of the edges), we remove the edge with the largest betweenness.
- We re-run BFS on each vertex and find the betweenness of every edge and remove the edge with the largest betweenness henceforth.
- We repeat this process until we divide the graph into individual vertices.
- We keep track of the communities that get generated with each edge removal and then decide on the level of partition (to stop the edge removal process) by evaluating the modularity scores of the community scores formed at different levels.
- The Girvan and Newman algorithm, though effective in delineating communities with high modularity scores, is very inefficient as it requires BFS (of time complexity $\Theta(E+V)$) to be run on each vertex for every edge removal.
  - For a graph with E edges and V vertices, the overall time complexity will be $\Theta(EV(E+V))$

Neighborhood Overlap based Approach

Principle of Triadic Closure

- If two people in a social network have a friend in common, then there is an increased likelihood that they will become friends themselves at some point in the future.
- If we observe snapshots of a social network at two distinct points in time, then in the later snapshot, we generally find a significant number of new edges that have formed through this triangle-closing operation, between two people who had a common neighbor in the earlier snapshot.





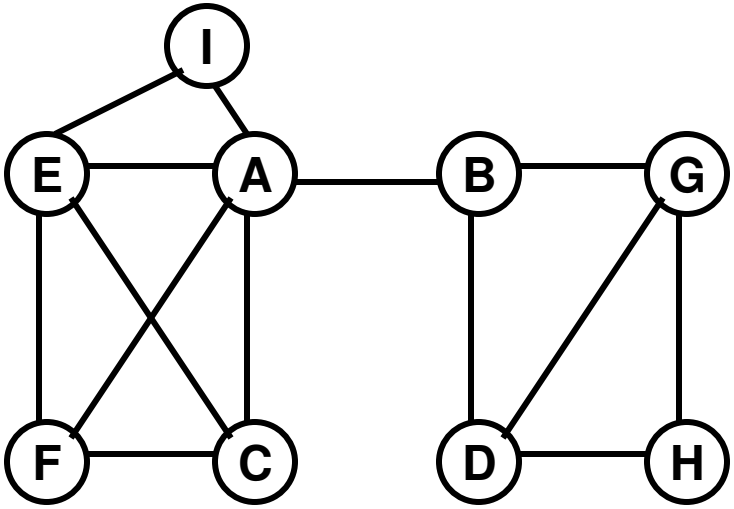
Neighborhood Overlap: Strong/Weak Tie

Neighborhood Overlap = $$\frac{\text{number of nodes who are neighbors of } \textit{both } A \text{ and } B}{\text{number of nodes who are neighbors of } \textit{at least one of } A \text{ or } B}$$

Note that one should not count neither A nor B as part of the neighbors in the denominator

For every edge, determine its neighborhood overlap. If it is above a threshold, then the edge could be classified to be of "strong tie", otherwise, we say weak tie.

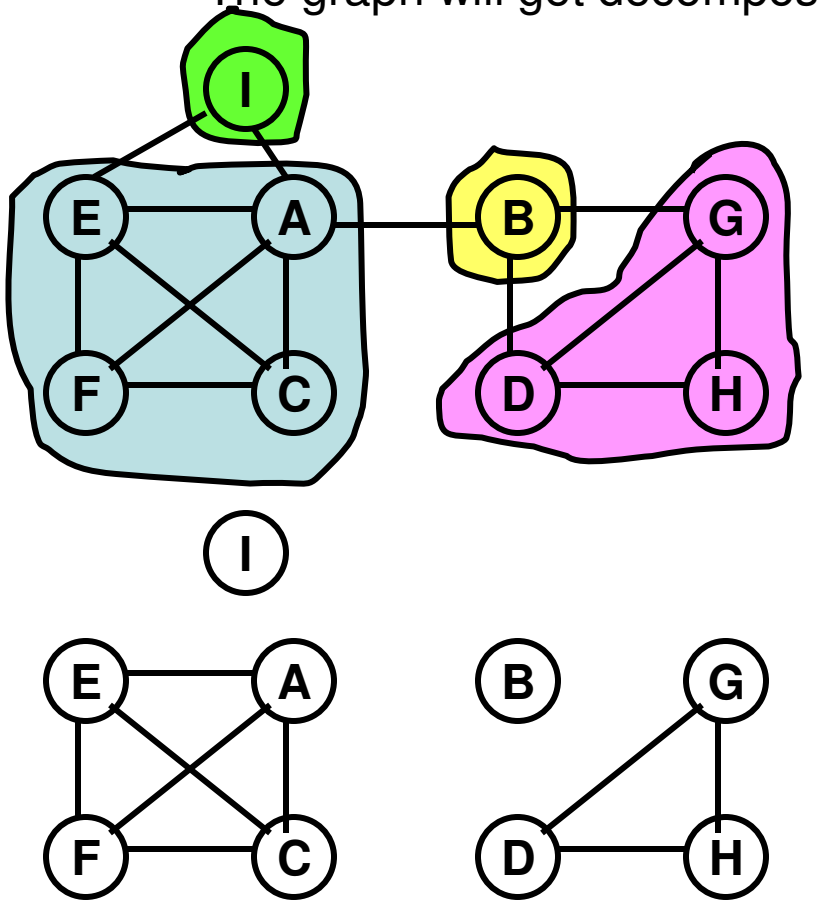
Let threshold = 0.5



Edge	Neighborhood Overlap	Tie
A – B	0	Weak
B – G	1/(2+1) = 0.33	Weak
B – D	1/(2+1) = 0.33	Weak
G – D	2/(2+1) = 0.66	Strong
D – H	1/(1+1) = 0.5	Strong
A – I	1/(4) = 0.25	Weak
A – E	3/(4) = 0.75	Strong
E – I	1/3 = 0.33	Weak
E – F	2/4 = 0.5	Strong
F – C	2/2 = 1.0	Strong
A – F	2/4 = 0.5	Strong

Weak Ties for Community Detection

- Weak ties serve to link together different tightly-knit communities that each contain a large number of stronger ties.
- Remove the Weak ties in the increasing order of their neighborhood overlap value.
  - The graph will get decomposed into several components (communities).



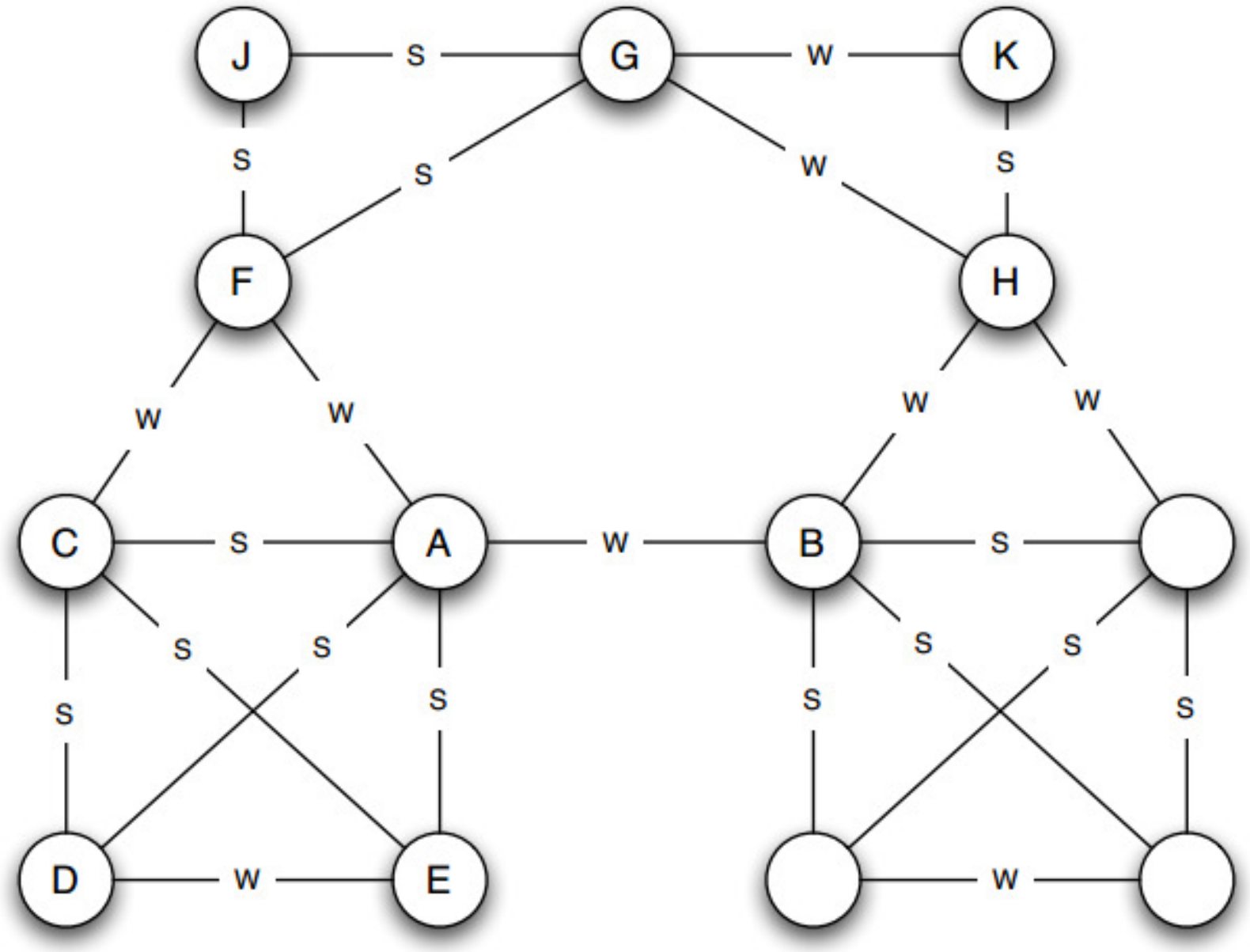
Edge	Neighborhood Overlap	Tie
A – B	0	Weak
B – G	1/(2+1) = 0.33	Weak
B – D	1/(2+1) = 0.33	Weak
G – D	2/(2+1) = 0.66	Strong
D – H	1/(1+1) = 0.5	Strong
A – I	1/(4) = 0.25	Weak
A – E	3/(4) = 0.75	Strong
E – I	1/3 = 0.33	Weak
E – F	2/4 = 0.5	Strong
F – C	2/2 = 1.0	Strong
A – F	2/4 = 0.5	Strong

Strength of an Edge and Strong Triadic Closure Property

- We classify edges to be either a strong tie (more frequent, trusted, incentive, opportunity) or a weak tie (not much frequent interaction, less known)
- In the context of social networks,
  - Strong tie → Friend
  - Weak tie → Acquaintance

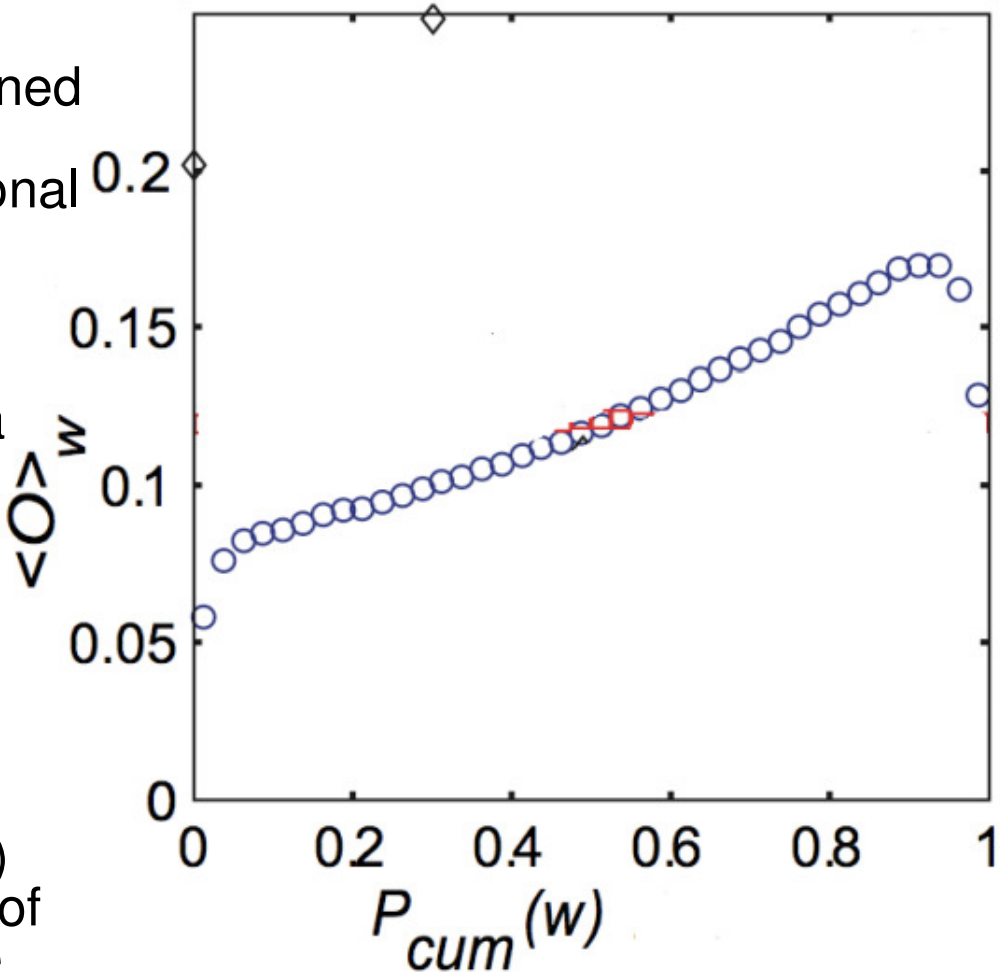
- Strong Triadic Closure Property: If a node A has strong ties to two neighbor nodes B and C, then there should be an edge between B and C (at least a weak tie).
  - A node is said to violate the strong triadic closure property if there is no edge between any two of its neighbor nodes with which it has strong ties.

Example: Strong Triadic Closure



Real World Networks: Strong/Weak Ties and Neighborhood Overlap

- Onnela et al. studied the who-talks-to-whom network maintained by a cell-phone provider that covered roughly 20% of a national population.
- Nodes – cell phone users
- Edges – If two nodes made a phone call to each other over a 18-week observation period
- Assigned edge weights that indicate the total number of minutes spent on phone calls between the two ends of the edge.
- The edges are sorted by the weights (indicating tie strength) and the neighborhood overlap of these edges in the network are also measured. We can notice a positive correlation in the plot.



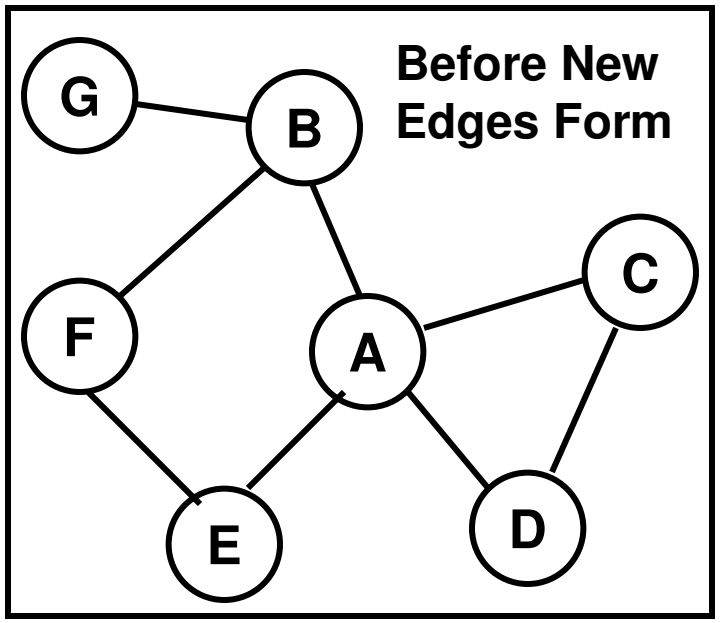
Percentile of the sorted order of the edges based on their strength

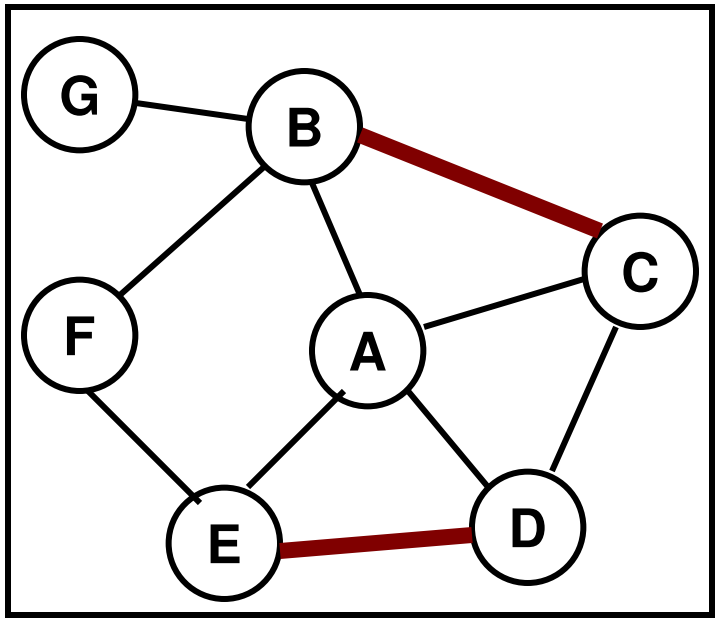
Strength of Weak Ties: Motivating Study

- Granovetter Interviews
  - 54 people who found their jobs via social tie:
    - 16.7% via strong tie (at least two interactions/ week)
    - 55.7% via medium tie (at least one interaction/ year)
    - 27.6% via a weak tie (less than one interaction/ year)
  - Weak ties are like bridges, used to transfer less redundant (but more useful) information.

Clustering Coefficient

- A measure used to capture the prevalence of triadic closure in networks
- The clustering coefficient of a node is the probability that two randomly selected neighbors of the node are directly connected to each other.
- The clustering coefficient of a node ranges from 0 (none of the neighbors are connected to each other) to 1 (all of the neighbors are connected to each other).
- The more strongly triadic closure is operating in the neighborhood of a node, the higher will tend to be its clustering coefficient.
- In the figure, the clustering coefficient of node A has increased from 1/6 to ½ after certain triadic closures.

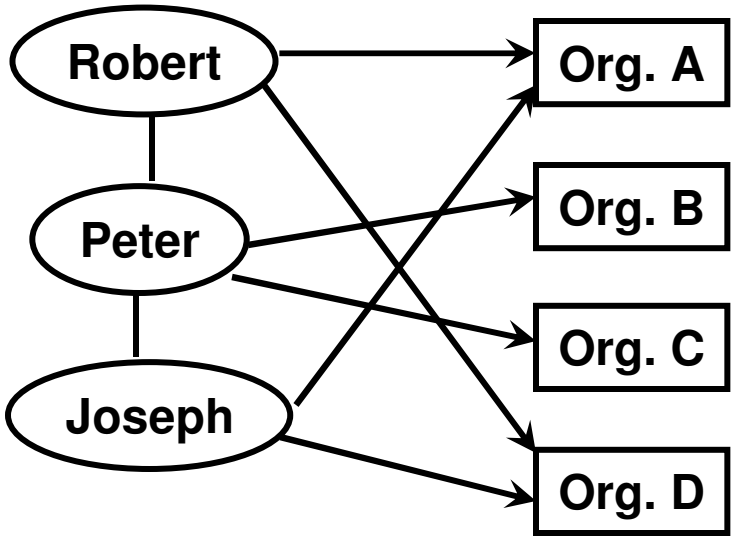




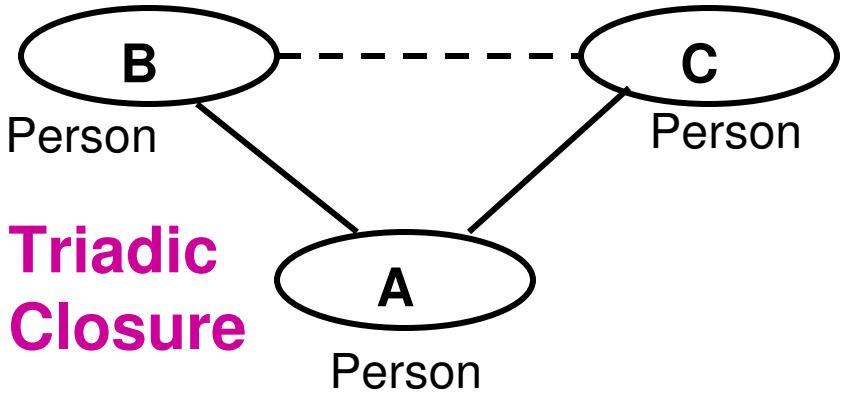
After New Edges Form

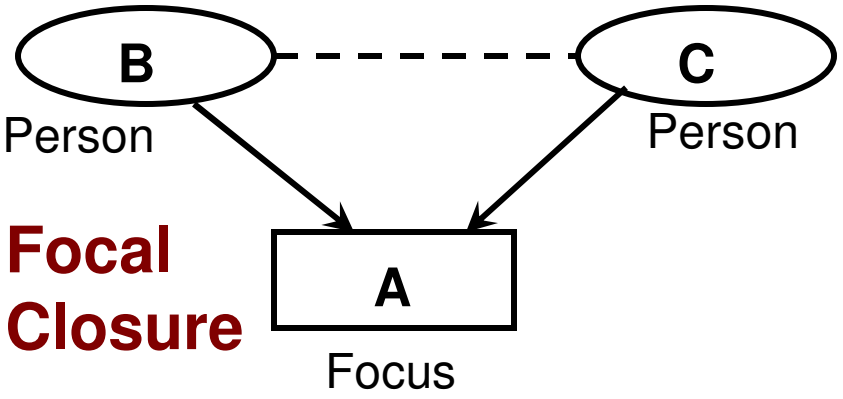
Social Affiliation Network

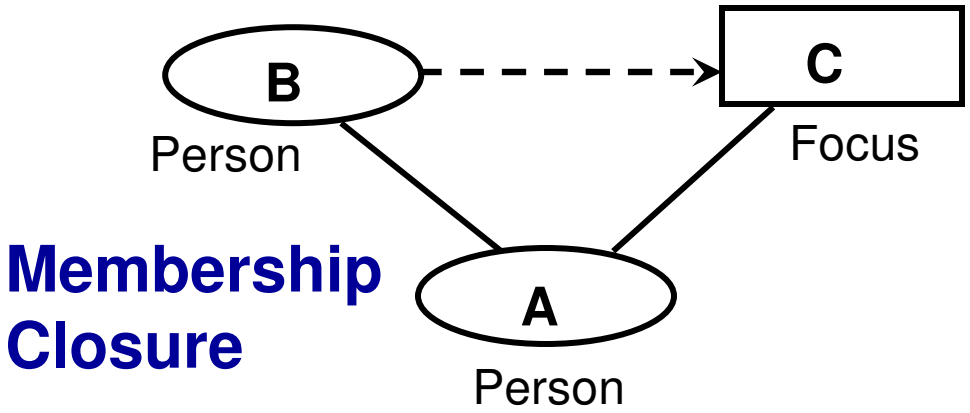
- Social Affiliation Network: A network that connects two people, and people with their foci (groups, activities, etc) they are associated with.
- As the network evolves,
  - Two people are likely to be linked to each other if they are affiliated to one or more common foci or common people.
  - A person is likely to be linked to a foci, because of one or more common neighbors who are already linked to the foci.



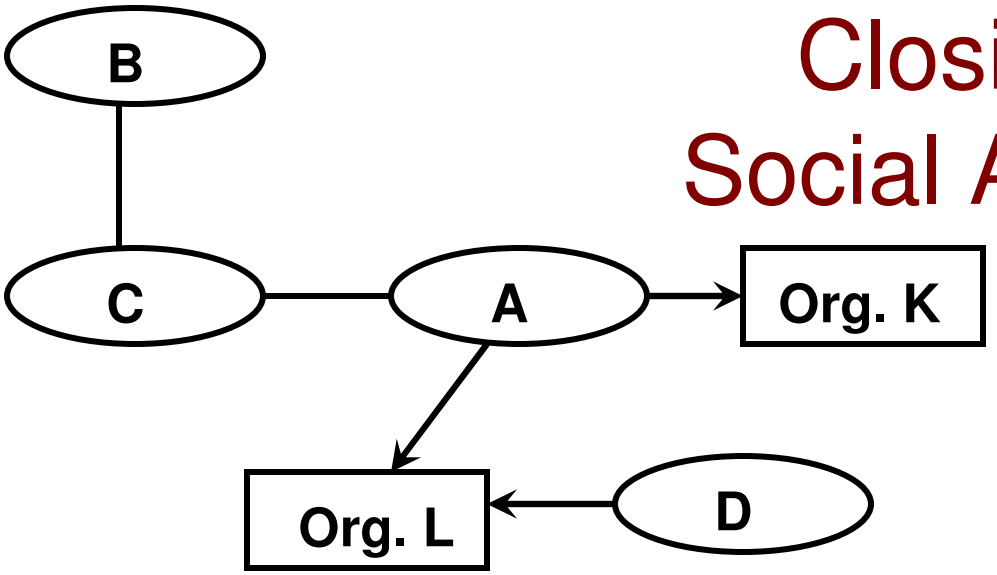
Closing of Triangle in Social Affiliation Networks



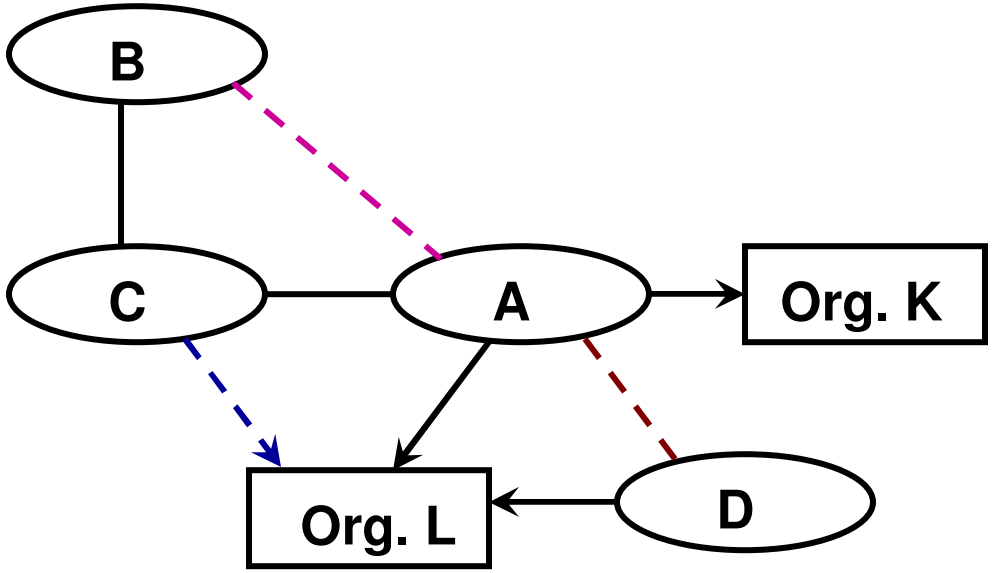




Closing of Triangle in Social Affiliation Networks: Example



Before triangle Closing

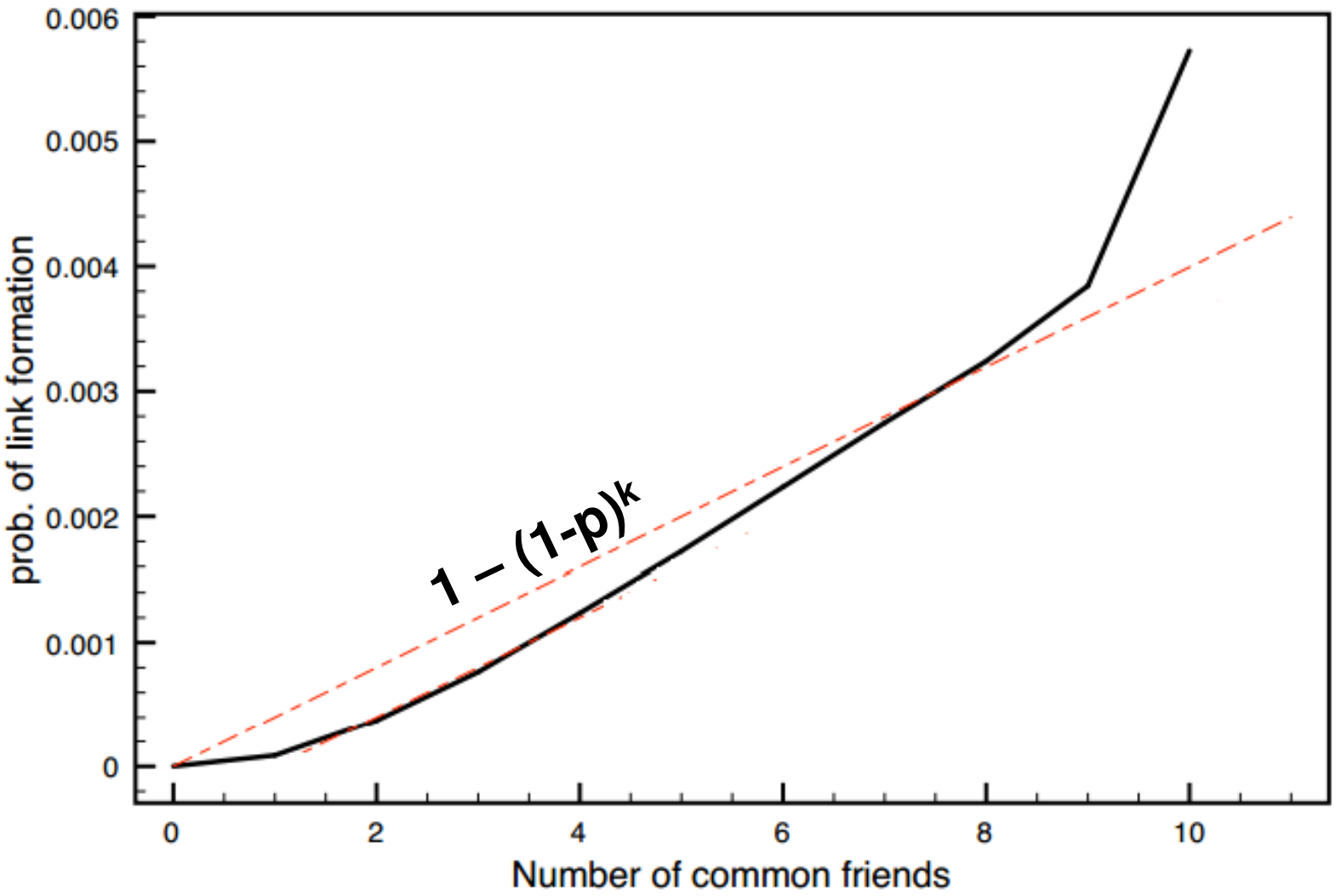


After triangle Closing

Triangle Closure: Probabilistic Analysis

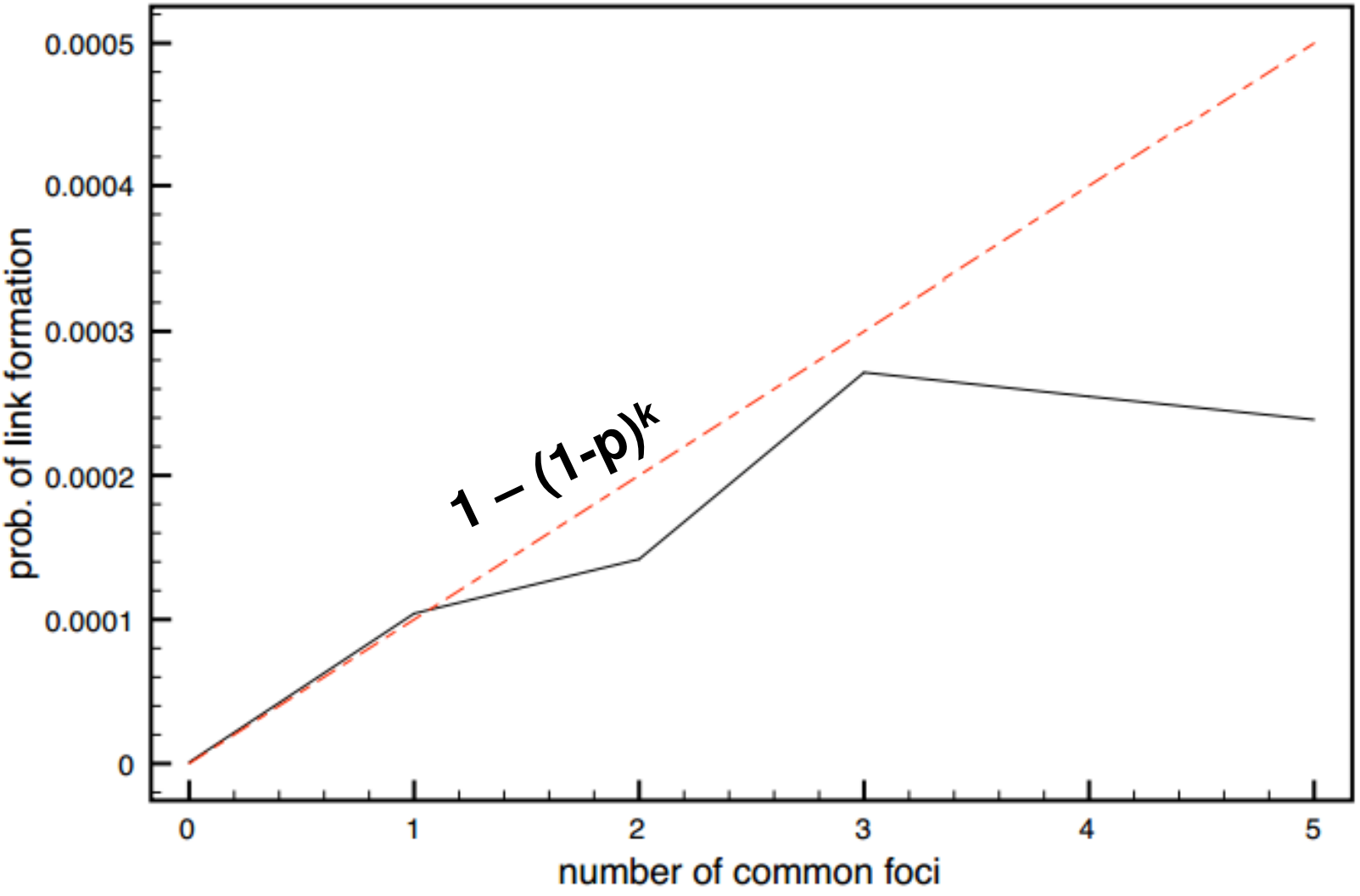
- For any two nodes, the larger the number of common neighbors, the larger is the probability of a link to be formed between the two.
- Let *p* be the probability that a common neighbor node could trigger the formation of a link between two particular nodes.
- If there are *k* such common neighbors,
  - the probability that a link is still not formed is given by: $(1-p)^k$
  - the probability that a link will be formed (due to triangle closure) is: $1 – (1-p)^k$
  - As *k* increases, the probability that a link will be formed due to triangle closure increases.
- On real-time data, to measure the probability that a link is formed due to triangle closure,
  - we take snapshots of the network at an earlier time (before triangle closure sets in) and at a later time.
  - For each possible value of the number of common neighbors (k), we measure the fraction of the new links added due to triangle closure

Triadic Closure on Real-time Data



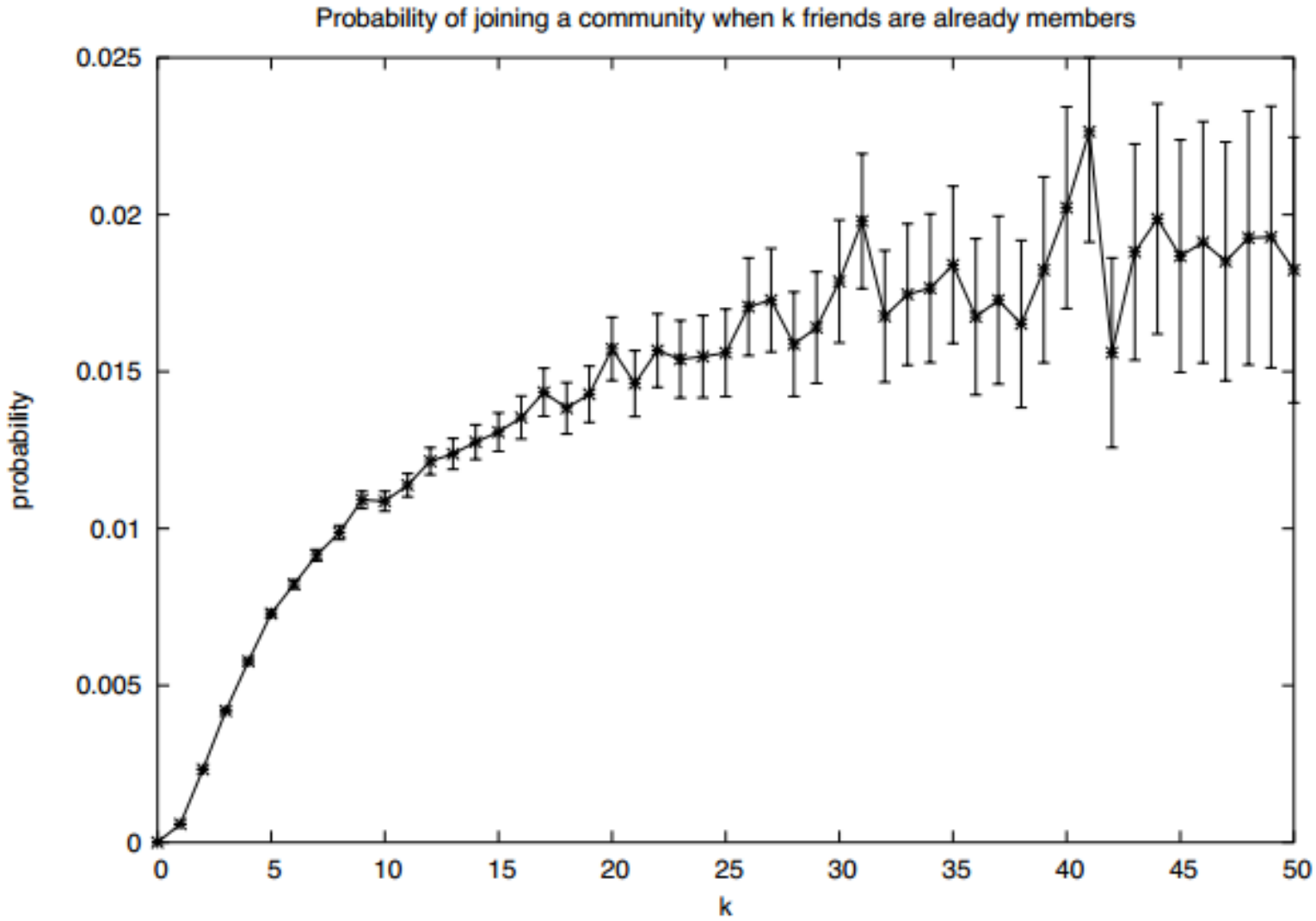
Source (Easley and Kleinberg): Figure 4.9: E-mail Dataset

Focal Closure on Real-time Data



Source (Easley and Kleinberg): Figure 4.10: E-mail Dataset

Membership Closure on Real-time Data

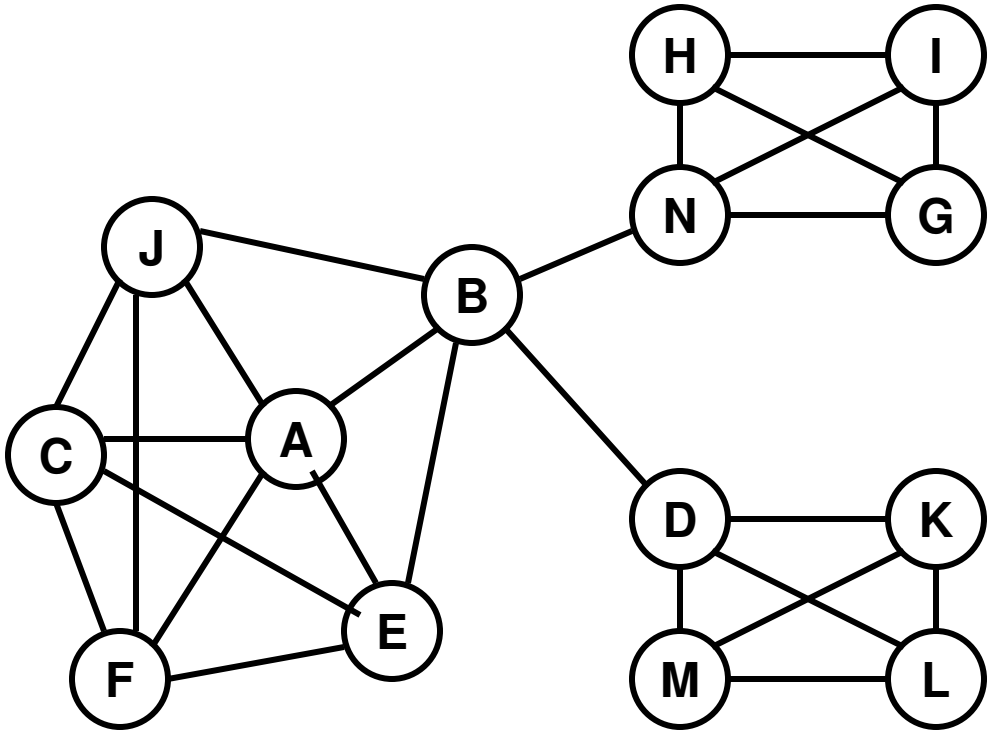


Source (Easley and Kleinberg): Figure 4.11: Membership to a LiveJournal Community

Edge and Node Embeddedness

- The “embeddedness of an edge” is the number of common neighbors the two endpoints have.
- The “embeddedness of a node” is the average of the embeddedness of its associated edges.
- A network could be partitioned into communities around nodes with lower embeddedness
- The “embeddedness of a community” is the average of the embeddedness of the edges forming the community.
- Structural Hole: The node with lower embeddedness is said to constitute a structural hole, especially when it connects two or more communities with larger embeddedness.
- A split typically occurs along a weak interface between two densely connected regions.

Edge and Node Embeddedness



A – B	2	C – E	2
A – E	3	H – I	2
A – C	3	H – N	2
A – F	3	N – G	2
A – J	3	I – G	2
B – E	1	H – G	2
E – F	2	N – I	2
C – F	3	D – K	2
C – J	2	D – M	2
F – J	2	M – L	2
B – J	1	K – L	2
B – N	0	D – L	2
B – D	0	M – K	2

A	2.8	I	2.0
B	0.8	J	2.0
C	2.5	K	2.0
D	1.5	L	2.0
E	2.0	M	2.0
F	2.5	N	1.5
G	2.0		
H	2.0		

B	0.8
D	1.5
N	1.5

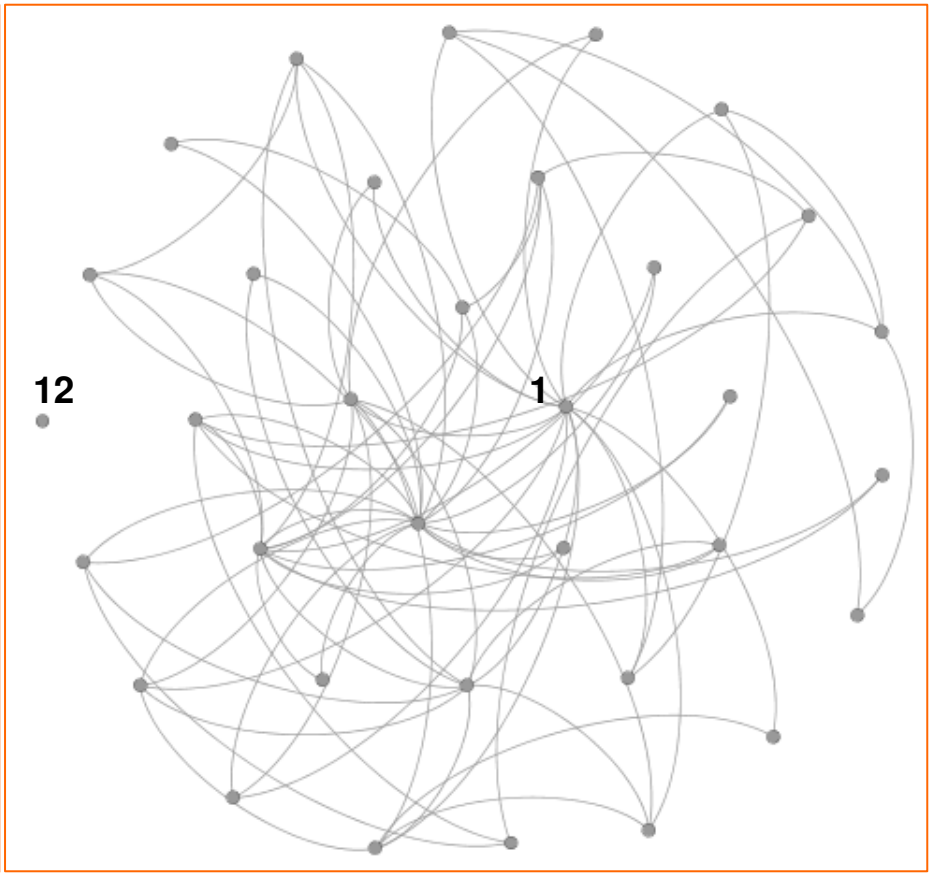
Community (A, B, C, E, F, J)	1.93
Community (D, K, L, M)	2.0
Community (H, I, N, G)	2.0

Nodes B, D and N are structural holes in the above network.

CINET: Bridge Edges

- CINET can identify the bridge edges in a network graph (removing the bridge edges disconnects the graph).

Karate Network: One component



Karate Network: Two components (after the removal of bridge edge 1-12)

Homophily: Networks and their Surrounding Contexts

Homophily

- Homophily refers to the notion that we tend to be similar to our friends.
- Typically, our friends are not like a random sample of the underlying population; rather, when viewed collectively, our friends are generally similar to us along
  - some immutable characteristics (things that cannot be changed) such as race and ethnicity
  - some mutable characteristics (things the could change over time) such as places we live, we work, beliefs, interests, etc.
- Homophily can produce a division of a network into densely connected homogeneous parts that are weakly connected to each other.

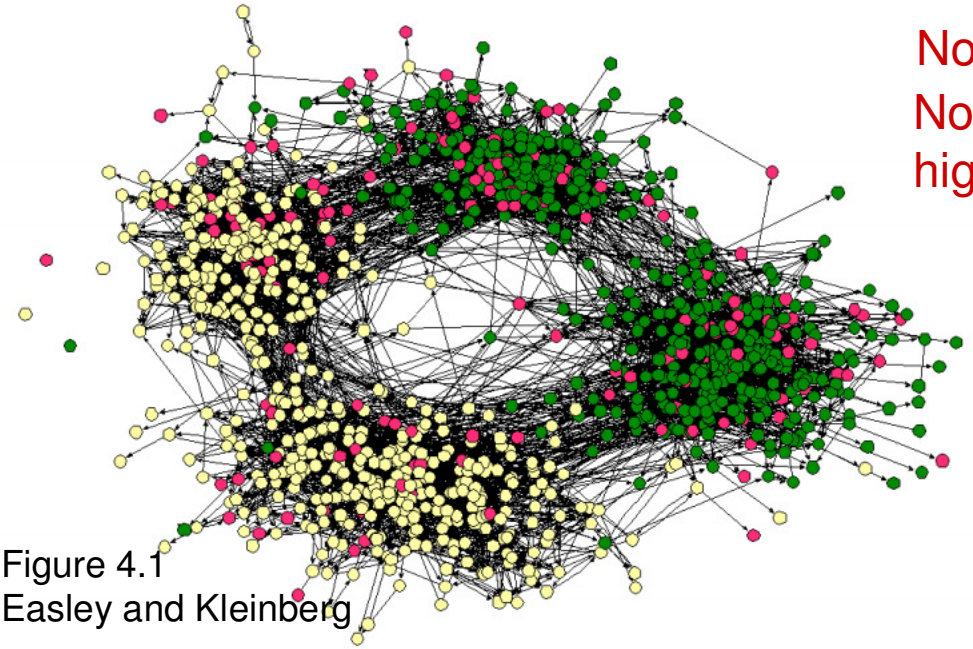


Figure 4.1
Easley and Kleinberg

Nodes are colored according to their race

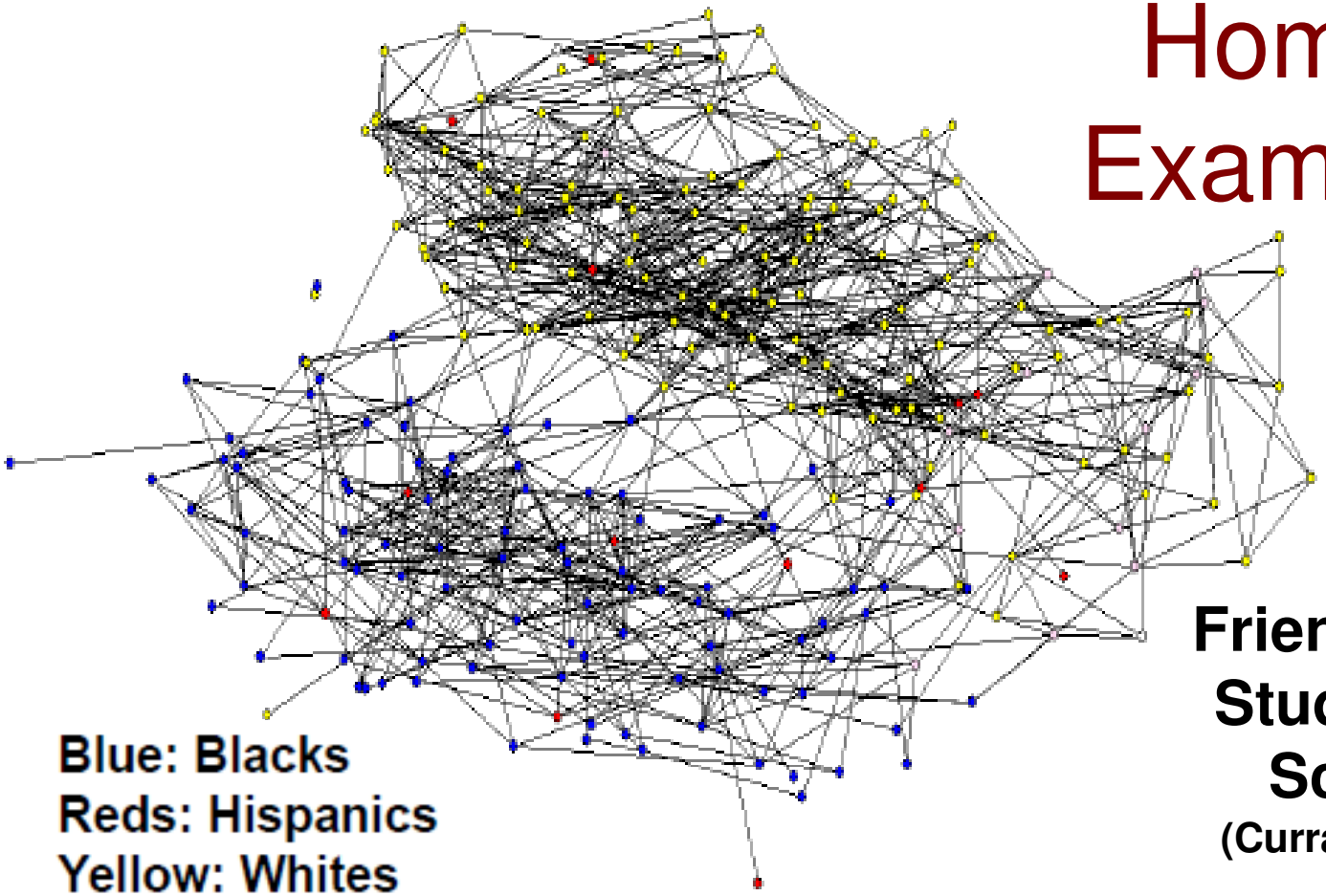
Nodes are students in middle school and high schools of a particular region.

The left and right regions of the network represent communities of nodes belonging to the same race.
The top and bottom regions of the network represent communities of nodes in the middle school and high school respectively.

Homophily Examples (1)

- National Sample: only 8% of people have *any* people of another race that they ``discuss important matters'' with (Marsden 87)
- Interracial marriages U.S.: 1% of white marriages, 5% of black marriages, 14% of Asian marriages (Fryer 07)
- In middle school, less than 10% of ``expected'' cross-race friendships exist (Shrum et al 88)
- Closest friend: 10% of men name a woman, 32% of women name a man (Verbrugge 77)
  - Far less than 50% if the connections were random

Homophily Examples (2)



Blue: Blacks
Reds: Hispanics
Yellow: Whites
White: Other

Friendship among Students in High School in US
(Currarini, Jackson, Pin; 2009, 2010)

Smaller groups tend to have different characteristics than larger groups.

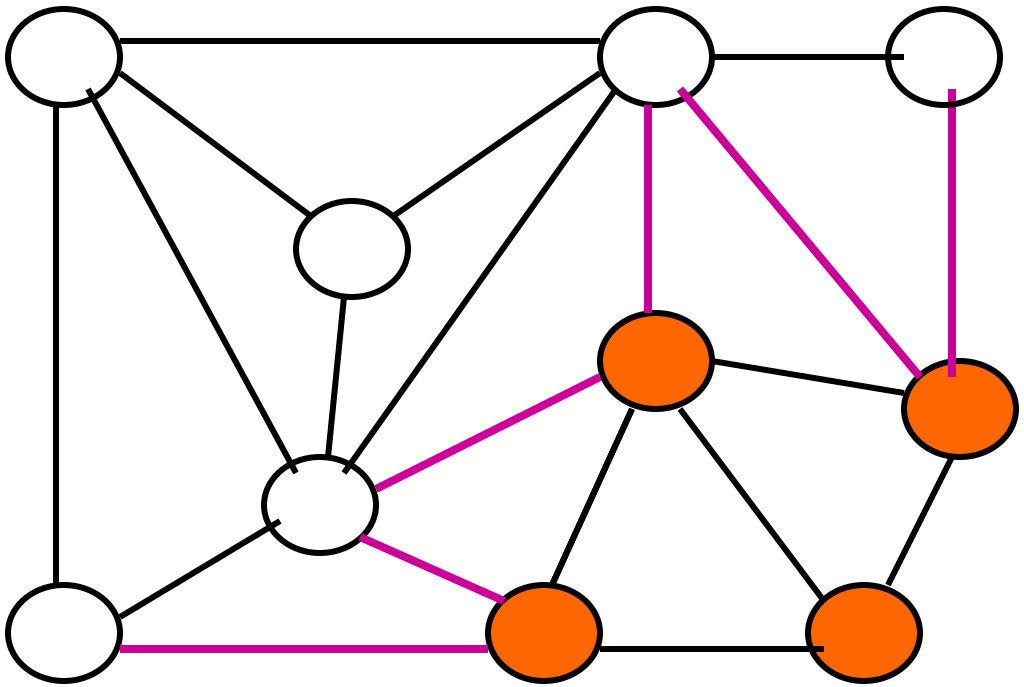
Segregation is more dominant as and when we impose more stringent definition on friendships

Percent	52	38	5	5
	White	Black	Hispanic	Other
White	86	7	47	74
Black	4	85	46	13
Hispanic	4	6	2	4
Other	6	2	5	9
	100	100	100	100

Measuring Homophily

- We will see a technique to find out whether a network exhibits homophily according to a particular characteristic of interest (like race or age).
- If *p* and *q* are the fraction of nodes of two particular types (say, male and female), then on average, the probability that a cross-community link is randomly formed between two nodes of different types is *2pq* (considering bi-directionality).
- If the fraction of cross-community links in the network is significantly less than *2pq*, then the network exhibits ***homophily***.
  - That is, a majority of the links are between people belonging to the same type.
  - The definition for "significantly less" is subjective, say 25% of 2pq
- If the fraction of cross-community links is in the vicinity of 2pq, then there is no homophily [the links are more random, not based on node types].
- If the fraction of cross-community links in the network is far greater than *2pq*, then the network exhibits ***inverse homophily***
  - A network of romantic relationships
- We can also extend the analysis to finding homophily involving nodes of more than two types. We have to just estimate the fraction of cross-community links that could be formed involving any two different node types.

Example: Measuring Homophily



Consider a network of classmates
4 – female and 6 – male students

Fraction of male students = 0.6
Fraction of female students = 0.4
Expected fraction of cross-gender links = 2 * 0.4 * 0.6 = 0.48

A total of 20 links are in the network
Of these, 6 are cross-gender links
Fraction of cross-gender links is 6/20 = 0.3.

If the threshold for cross-community links is 100%, then 0.3 < 0.48
The network is said to exhibit homophily.

Removing all the cross-community links would lead us to identify the nodes forming the different communities

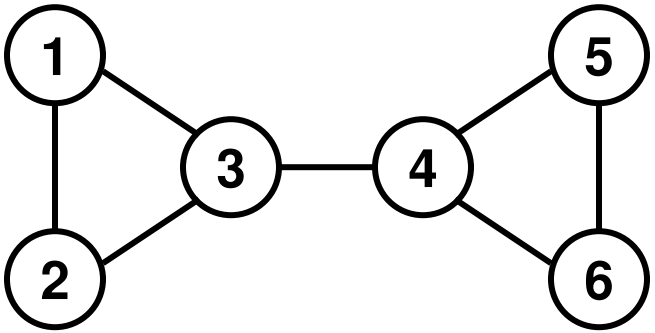
If the threshold for cross-community links is 25%, then 0.3 > (25%)(0.48)
The network does not exhibit homophily.

Eigen vectors based approach

Using Eigenvectors to Identify Components

- Compute the Eigenvalues and Eigenvectors of the Adjacency matrix A
- The principal Eigenvector is the one that corresponds to the largest Eigenvalue.
- If all the entries in the “principal Eigenvector” are positive, then it implies that all the nodes are in one component.
  - Else, the vertices with the positive entries are in one component and those with the negative entries are in another component. (Note: 0 is considered positive).
- We apply the above interpretation to all the subsequent Eigenvectors (in the decreasing order of the corresponding Eigenvalues) and identify the smaller communities within the larger components.

Example: Eigenvectors to Identify Communities (1)

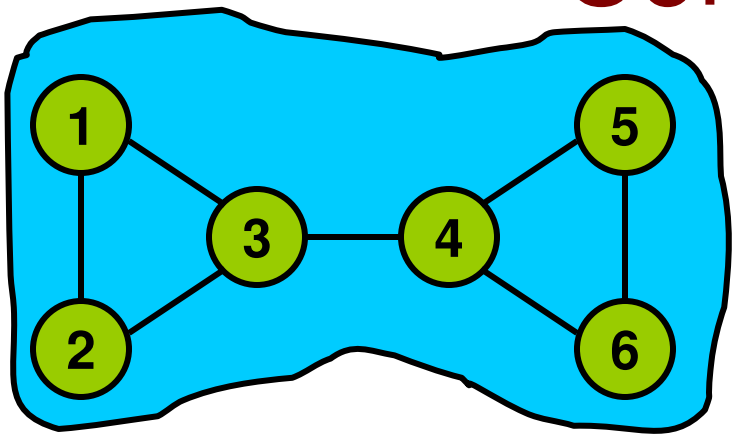


$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

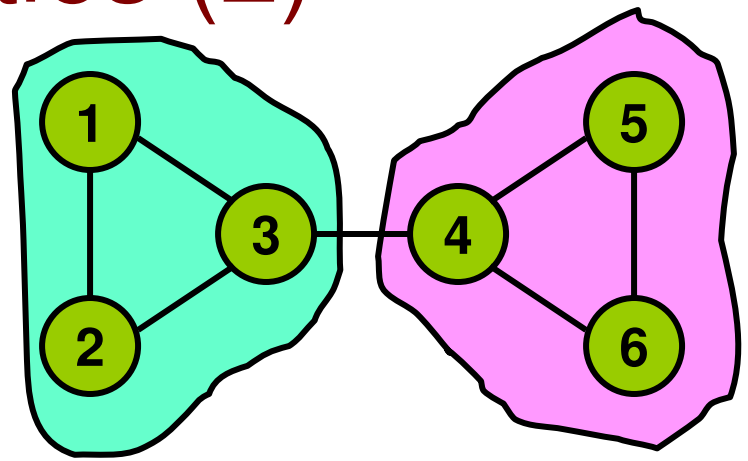
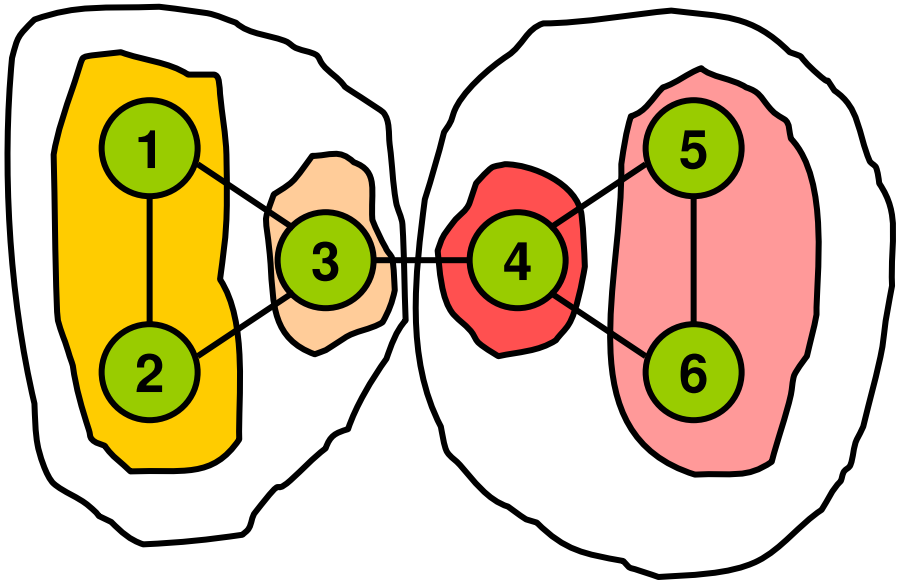
The Eigenvalues in the decreasing order and their corresponding Eigenvectors are:

Eigenvalue	Eigenvector	
2.4142	[1; 1; 1.4142; 1.4142; 1; 1]	All entries are +ve; hence all vertices are in one single component
1.7321	[-1; -1; -0.7321; 0.7321; 1; 1]	Vertices 1, 2, 3 form one community Vertices 4, 5, 6 form another comm.
-0.4142	[1; 1; -1.4142; -1.4142; 1; 1]	Within 1-2-3; 3 is in one comm. Within 4-5-6; 4 is in one comm.
-1	[-1; 1; 0; 0; 0; 0]	
-1	[0; 0; 0; 0; -1; 1]	
-1.7321	[-1; -1; 2.7321; -2.7321; 1; 1]	

Example: Eigenvectors to Identify Communities (2)



2.4142 [1; 1; 1.4142; 1.4142; 1; 1]



1.7321 [-1; -1; -0.7321; 0.7321; 1; 1]