

Data Science

UNIT-V

Prescriptive Analysis

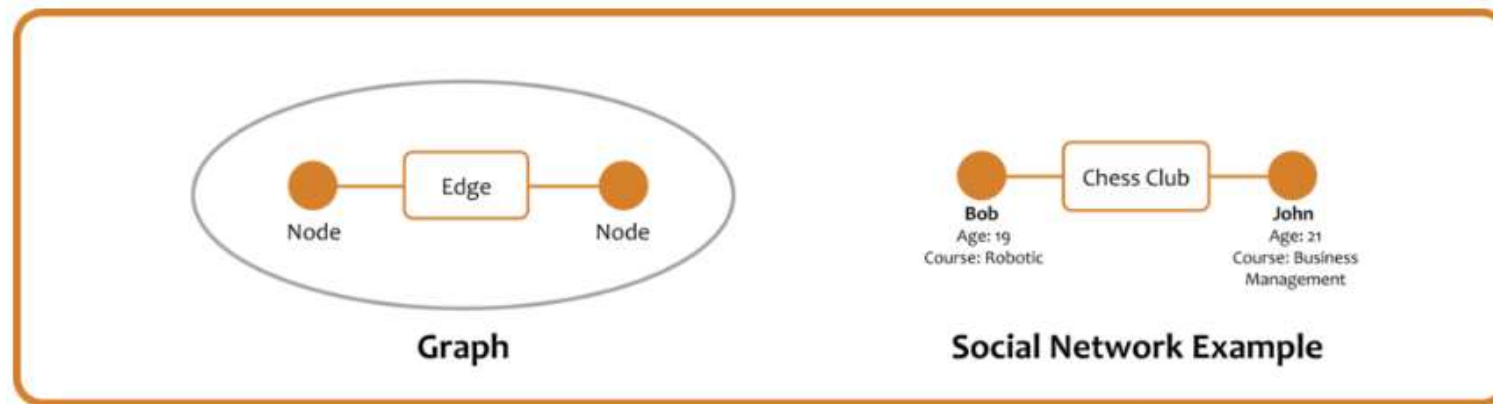
Dr. N. Subhash Chandra
Professor CSE

UNIT-1: Contents

- **UNIT V – Prescriptive Analysis**
- **Forecasting Techniques:** Time series data, Techniques, and accuracy, Moving average method, Single, double, triple exponential smoothing, Regression model for forecasting, Auto-Regression models, ARIMA Process.
- **Graph Analytics:** Path analysis, Connectivity analysis, Community analysis, Centrality analysis, Social-Network Graphs, Communities and Clusters, Betweenness, The Girvan-Newman Algorithm.

Graph Analytics

- Graph analytics, also known as graph algorithms or network analysis, is the analysis of data in a graph format using data points as nodes and relationships as edges.
- Nodes signify points or entities Every node and edge hold specific properties that define its characteristics.
- Graph analytics is used to determine the strength and the direction of connections between objects in a graph.
- Graph analytics enables you to store, manage, and query data in the form of a graph. It allows you to find the relationships between people, places, things, events, locations, and others.



Graph Analytics

- Graph analytics tends to provide a pairwise relationship between the objects and represent the structural characteristics of a graph.
- Graphs are visually appealing and make it easier to understand how complex networks behave.
- The best part about graph analytics is the ease with which you can add information to a graph. It is not necessary to understand everything about your data to start storage and investigation.
- Some of the most popular tools for graph analytics include Neo4j, OrientDB, Amazon Neptune, and several others.

Types of Graph Analytics

Graph analytics can be categorized into four main types. Let's have a quick look at them.

1. Path analysis

Path analysis tends to focus on the relationships between two nodes in a graph. It **determines the shortest distance between two nodes** and analyzes similar shapes and distances from different paths that connect entities within the graph.

2. Connectivity analysis

Connectivity analysis is used to determine **the strength of two interconnected nodes**. It helps you identify whether the connection between the nodes is strong or weak. Connectivity analysis also determines the number of edges flowing into the node and the ones flowing out of the node.

3. Centrality analysis

It tends to determine the **importance of a present node within the graph network** and its connectivity to others. Centrality analysis enables you to understand the most influential node and the connection it accesses.

4. Community Analysis (Network analysis)

Network analysis, also known as, community analysis tends to **define the density of relationships between nodes**. It helps you learn about the nodes that frequently interact with each other in a graph network. Network analysis also enables you to determine if the connecting edges are transient. It tends to predict whether the created graph network will grow or not.

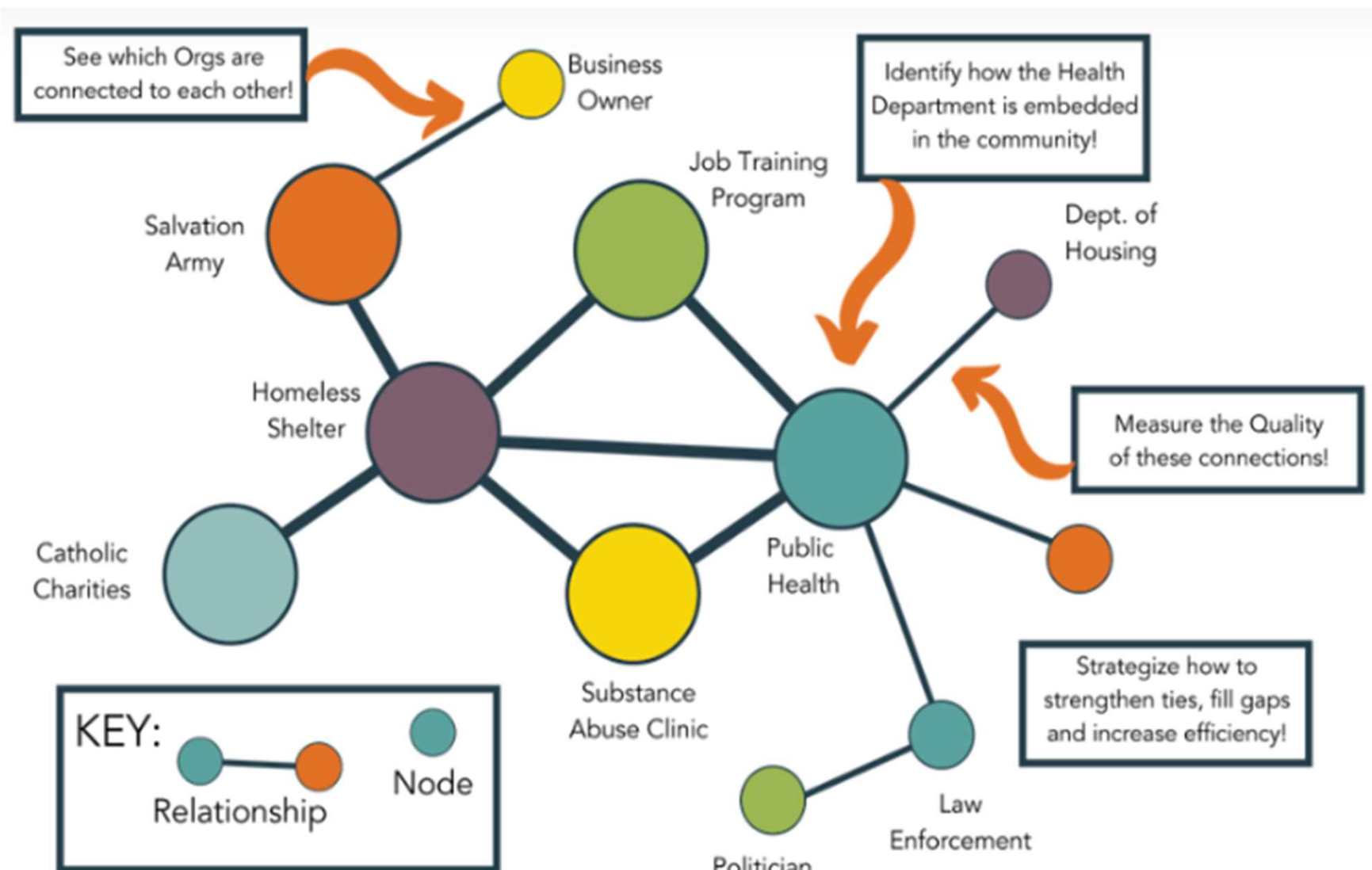
What is a Social Network?

Social networks are websites and apps that allow users and organizations to connect, communicate, share information and form relationships.

Ex: Facebook, Twitter, Google+, or another website that is called a “social network,” and indeed this kind of network is representative of the broader class of networks called “social.”

The essential characteristics of a social network are:

1. There is a collection of entities that participate in the network. Typically, these **entities are people**, but they could be something else entirely.
2. There is **at least one relationship** between entities of the network. On Facebook or its ilk, this relationship is called friends. Sometimes the relationship is all-or-nothing; two people are either friends or they are not.
3. There is an assumption of **non randomness or locality**. This condition is the hardest to formalize, but the intuition is that relationships tend to cluster. That is, if entity A is related to both B and C, then there is a higher probability than average that B and C are related.



1. Nodes (or Vertices)

These are the individual **entities** within the network, which could be people, groups, or other organizations. They are the basic unit of analysis in a social network. For example, in a friendship network, each person would be a node.

2. Edges (or Links or Ties)

These represent **relationships or interactions between nodes**. For instance, an edge may represent a friendship between two people or a business transaction between two companies.

3. Degree

This is the **number of connections a node has**. A person with a high degree in a social network could be considered popular or influential as they have many connections to others.

4. Density

A measure of how many connections exist in the network compared to how many could possibly exist. High density networks suggest that **many members are connected to each other**, promoting quick information or disease spread.

5. Degree Centrality

This quantifies the **importance of a node within the network**. Nodes with high centrality are influential within the network due to their connections.

6. Betweenness Centrality

This is a type of centrality that quantifies the **number of times a node acts as a bridge along the shortest path between two other nodes**. It identifies nodes that connect different parts of a network.

7. Closeness Centrality

Another type of centrality that measures **how close a node is to all other nodes** in the network. Nodes with high closeness centrality can quickly interact with all others.

9. Clusters (or Communities)

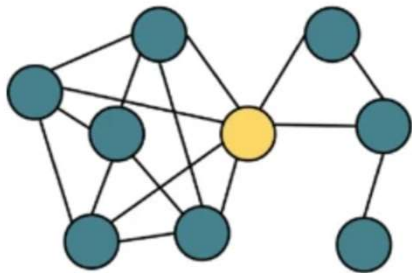
These are **groups of nodes that have a higher density** of ties among themselves than with nodes outside the group. For instance, groups of friends within a larger social network.

12. Degree Distribution

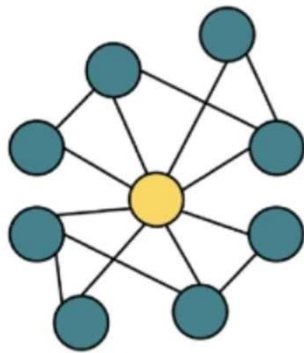
The **probability distribution of degrees** over the entire network. It's often used in network analysis to identify the type of network (e.g., random, scale-free).

13. Modularity

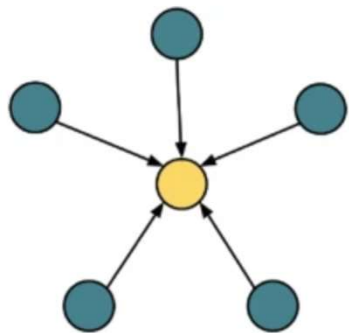
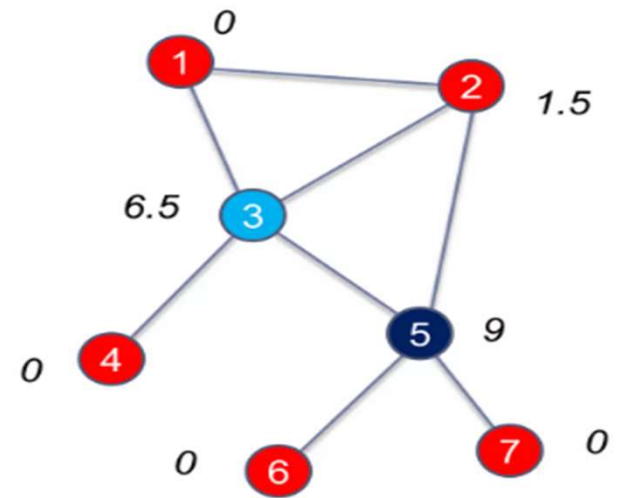
A **measure that quantifies the strength of division of a network into modules** (also called groups, clusters or communities). Networks with high modularity have dense connections between nodes within modules but sparse connections between nodes in different modules.



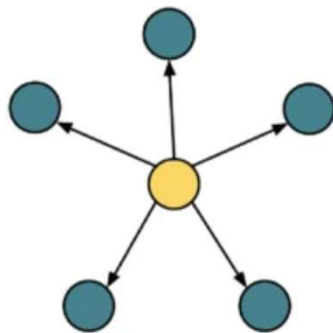
Betweenness



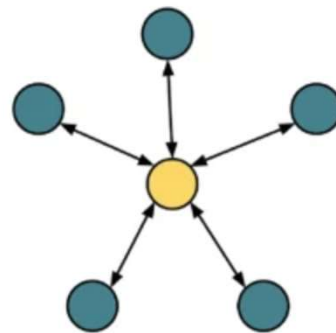
Closeness



In-degree

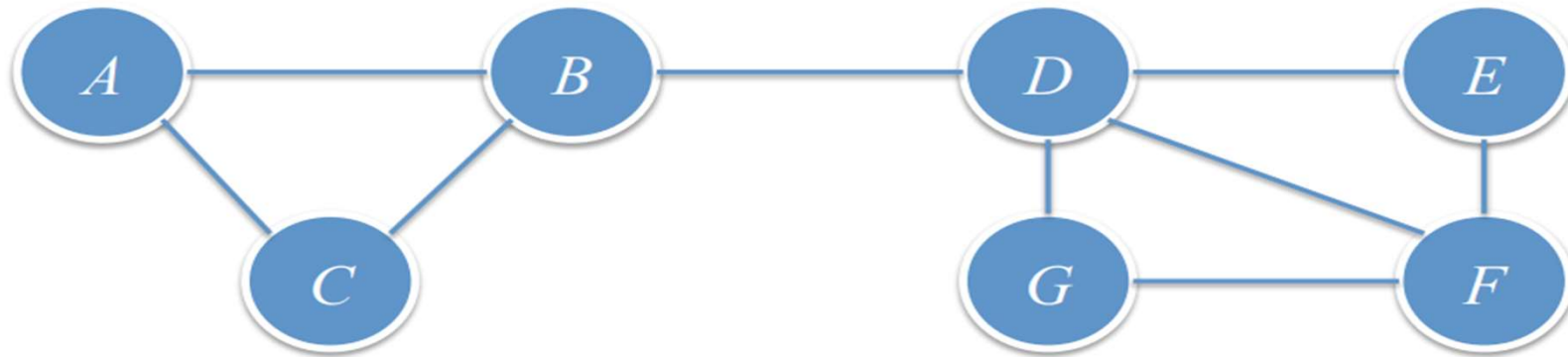


Out-degree



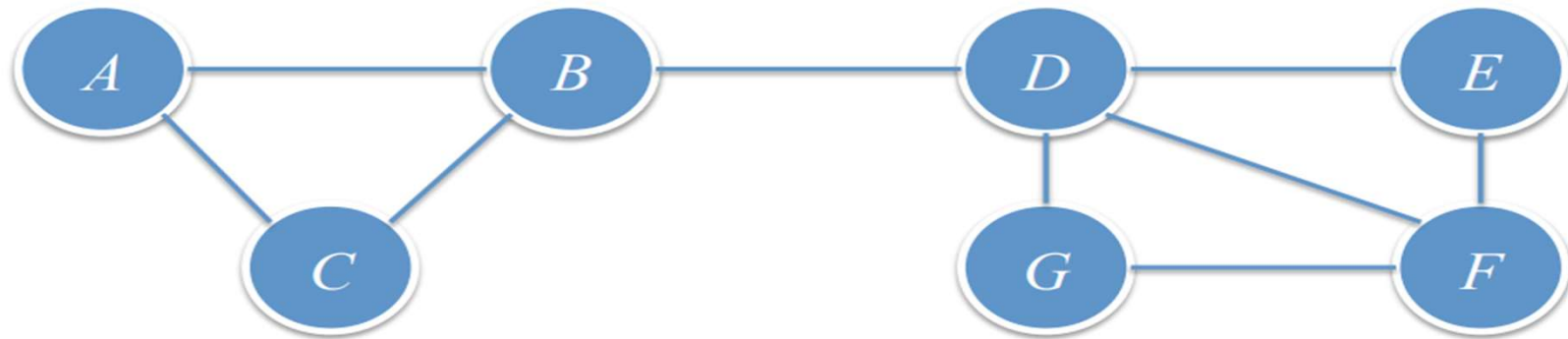
Degree

Types of Social (or Professional) Networks



- Of course, the “social network”. But also several other types
- Telephone network
- Nodes are phone numbers
- AB is an edge if A and B talked over phone within the last one week, or month, or ever
- Edges could be weighted by the number of times phone calls were made, or total time of conversation

Types of Social (or Professional) Networks



- Email network: nodes are email addresses
- AB is an edge if A and B sent mails to each other within the last one week, or month, or ever
 - One directional edges would allow spammers to have edges
- Edges could be weighted
- Other networks: collaboration network – authors of papers, jointly written papers or not
- Also networks exhibiting locality property

Social Networks as Graphs

Social networks are naturally modeled as graphs, which we sometimes refer to as a social graph. The entities are the nodes, and an edge connects two nodes if the nodes are related by the relationship that characterizes the network.

If there is a degree associated with the relationship, this degree is represented by labeling the edges. Often, social graphs are undirected, as for the Facebook friends graph.

But they can be directed graphs, as for example the graphs of followers on Twitter or Google+.

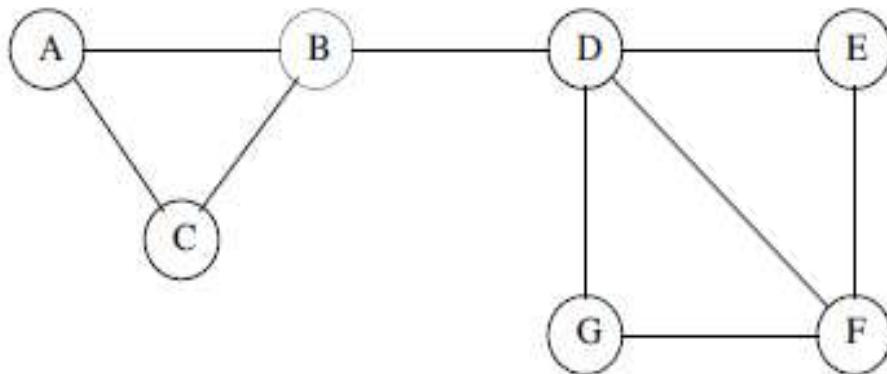
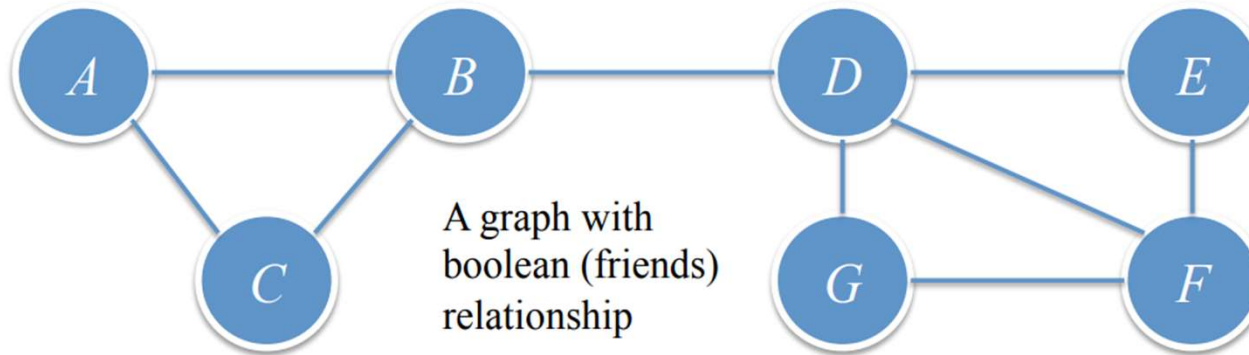


Figure 10.1 is an example of a tiny social network. The entities are the nodes A through G. The relationship, which we might think of as “friends,” is represented by the edges. For instance, B is friends with A, C, and D.

Figure 10.1 Example of a small social network

Social Networks as Graphs



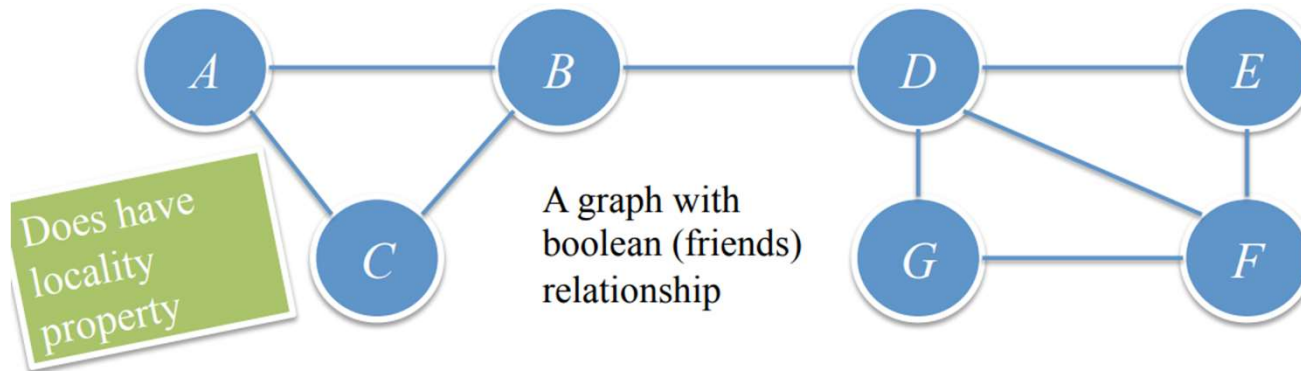
- Given a graph of 7 nodes (A-G), how many undirected edges are possible?

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

- Check for the non-randomness criterion
- In a random graph (V,E) of 7 nodes and 9 edges, if XY is an edge, YZ is an edge, what is the probability that XZ is an edge?
 - For a large random graph, it would be close to $|E|/\binom{V}{2} = 9/21 \sim 0.43$
 - Small graph: XY and YZ are already edges, so compute within the rest
 - So the probability is $(|E|-2)/(\binom{V}{2}-2) = 7/19 = 0.37$
- Now let's compute what is the probability for this graph in particular

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

Social Networks as Graphs



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

- For each X , check possible YZ and check if YZ is an edge or not
- Example: if $X = A$, $YZ = \{BC\}$, it is an edge

| $X=$ | $YZ=$ | <i>Yes/Total</i> |
|------|--------------------------|------------------|
| A | BC | 1/1 |
| B | AC, AD, CD | 1/3 |
| C | AB | 1/1 |
| D | BE, BG, BF, EF, EG, FG | 2/6 |

| $X=$ | $YZ=$ | <i>Yes/Total</i> |
|--------------|--------------|---------------------------|
| E | DF | 1/1 |
| F | DE, DG, EG | 2/3 |
| G | DF | 1/1 |
| <i>Total</i> | | $9/16 \sim \mathbf{0.56}$ |

Clustering of Social-Network Graphs

- Clustering of the graph is considered as a way to identify communities. Clustering of graphs involves the following steps:

1. Distance Measures for Social-Network Graphs

- Locality property → there are clusters
- Clusters are communities
 - People of the same institute, or company
 - People in a photography club
 - Set of people with “Something in common” between them
- Need to define a distance between points (nodes)
- In graphs with weighted edges, different distances exist
- For graphs with “friends” or “not friends” relationship
 - Distance is 0 (friends) or 1 (not friends)
 - Or 1 (friends) and infinity (not friends)
 - Both of these violate the triangle inequality
 - Fix triangle inequality: distance = 1 (friends) and 1.5 or 2 (not friends) or length of shortest path

2. Applying Standard Clustering Methods

Hierarchical clustering of a social-network graph starts by combining some two nodes that are connected by an edge. Successively, edges that are not between two nodes of the same cluster would be chosen randomly to combine the clusters to which their two nodes belong. The choices would be random, because all distances represented by an edge are the same.

EXAMPLE:

EXAMPLE 10.3 Consider again the graph of Fig. 10.1, repeated here as Fig. 10.3. First, let us agree on what the communities are. At the highest level, it appears that there are two communities $\{A, B, C\}$ and $\{D, E, F, G\}$. However, we could also view $\{D, E, F\}$ and $\{D, F, G\}$ as two subcommunities of $\{D, E, F, G\}$; these two subcommunities overlap in two of their members, and thus could never be identified by a pure clustering algorithm. Finally, we could consider each pair of individuals that are connected by an edge as a community of size 2, although such communities are uninteresting.

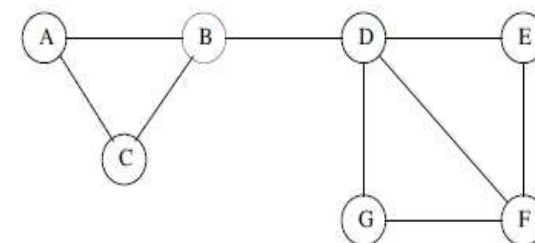
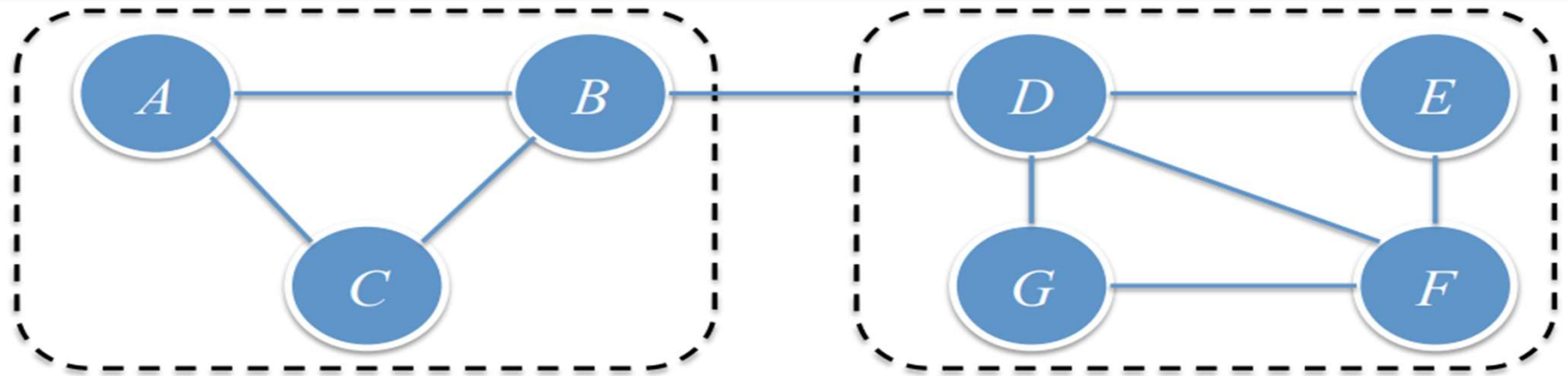


Figure 10.3 Repeat of Fig. 10.1

The problem with hierarchical clustering of a graph like that of Fig. 10.3 is that at some point we are likely to choose to combine B and D, even though they surely belong in different clusters. The reason we are likely to combine B and D is that D, and any cluster containing it, is as close to B and any cluster containing it, as A and C are to B. There is even a $1/9$ probability that the first thing we do is to combine B and D into one cluster.

Traditional Clustering



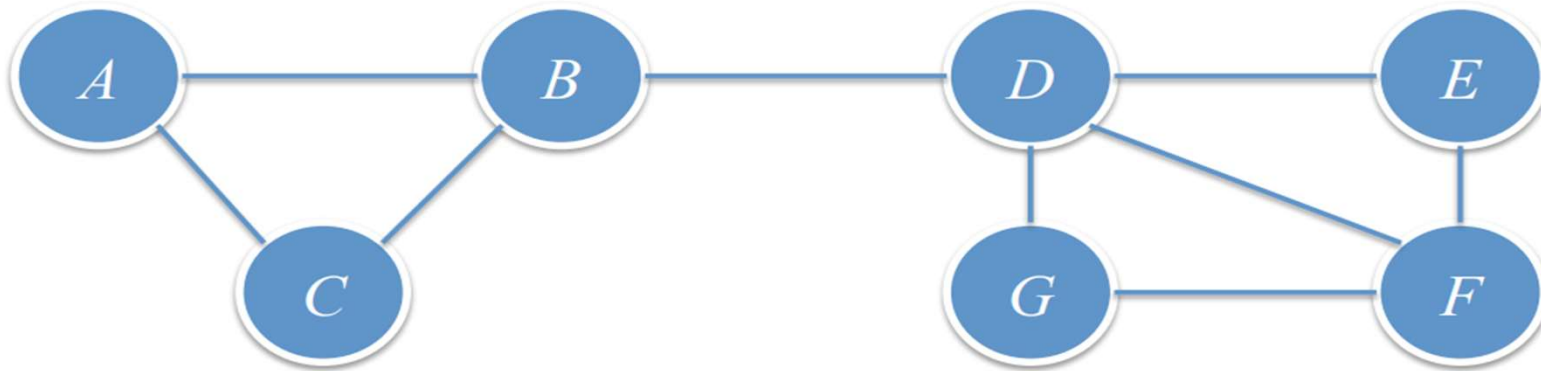
- Intuitively, two communities
- Traditional clustering depends on the distance
 - Likely to put two nodes with small distance in the same cluster
 - Social network graphs would have cross-community edges
 - Severe merging of communities likely
- May join *B* and *D* (and hence the two communities) with not so low probability

3. Betweenness:

Since there are problems with standard clustering methods, several specialized clustering techniques have been developed to find communities in social networks. The simplest one is based on finding the edges that are least likely to be inside the community.

Define the betweenness of an edge (a, b) to be the number of pairs of nodes x and y such that the edge (a, b) lies on the shortest path between x and y . To be more precise, since there can be several shortest paths between x and y , edge (a, b) is credited with the fraction of those shortest paths that include the edge (a, b) . As in golf, a high score is bad. It suggests that the edge (a, b) runs between two different communities; that is, a and b do not belong to the same community

Betweenness:



- Betweenness of an edge AB : #of pairs of nodes (X,Y) such that AB lies on the shortest path between X and Y
 - There can be more than one shortest paths between X and Y
 - Credit AB the fraction of those paths which include the edge AB
- High score of betweenness means?
 - The edge runs “between” two communities
- Betweenness gives a better measure
 - Edges such as BD get a higher score than edges such as AB
- Not a distance measure, may not satisfy triangle inequality. Doesn't matter!

4. The Girvan-Newman Algorithm:

In order to exploit the betweenness of edges, we need to calculate the number of shortest paths going through each edge. We shall describe a method called the Girvan-Newman (GN) Algorithm, which visits each node X once and computes the number of shortest paths from X to each of the other nodes that go through each of the edges. The algorithm begins by performing a breadth-first search (BFS) of the graph, starting at the node X . Note that the level of each node in the BFS presentation is the length of the shortest path from X to that node. Thus, the edges that go between nodes at the same level can never be part of a shortest path from X .

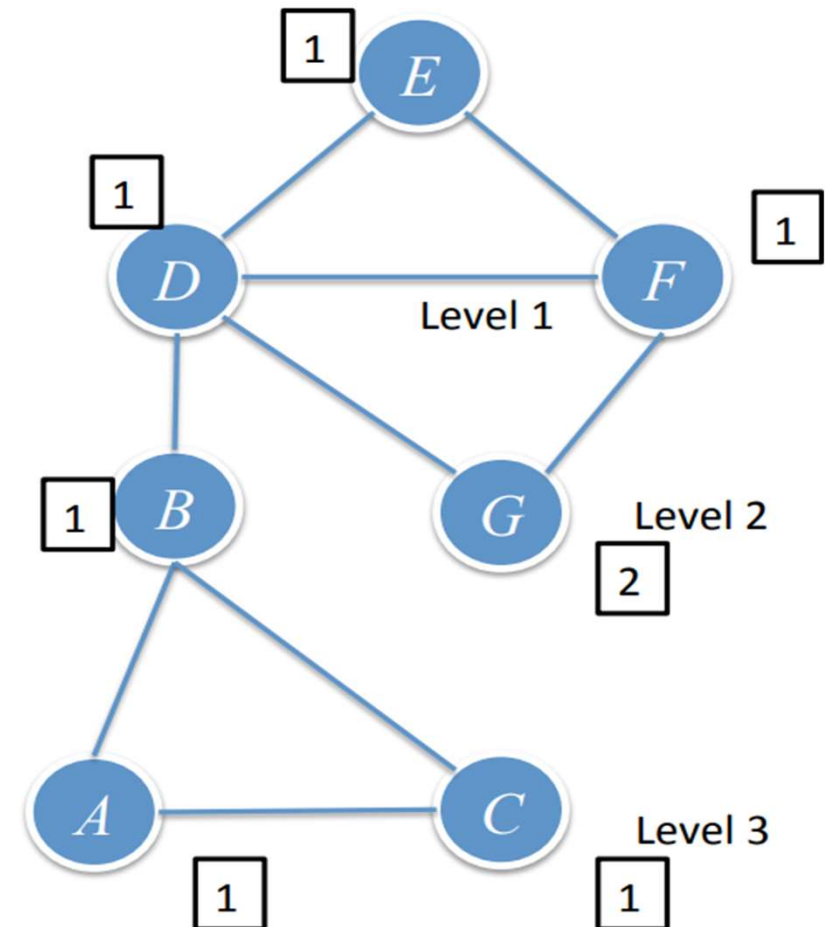
Edges between levels are called DAG edges (“DAG” stands for directed, acyclic graph). Each DAG edge will be part of at least one shortest path from root X . If there is a DAG edge (Y, Z) , where Y is at the level above Z (i.e., closer to the root), then we shall call Y a parent of Z and Z a child of Y , although parents are not necessarily unique in a DAG as they would be in a tree.

Girvan Newman Algorithm

- Step 1 – BFS: Start at a node X , perform a BFS with X as root
- Observe: level of node Y = length of shortest path from X to Y
- Edges between level are called “DAG” edges
 - Each DAG edge is part of at least one shortest path from X
- Step 2 – Labeling: Label each node Y by the number of shortest paths from X to Y

A directed acyclic graph (DAG) is a graph with nodes connected by directed edges that don't form closed loops

Calculate *betweenness* of edges



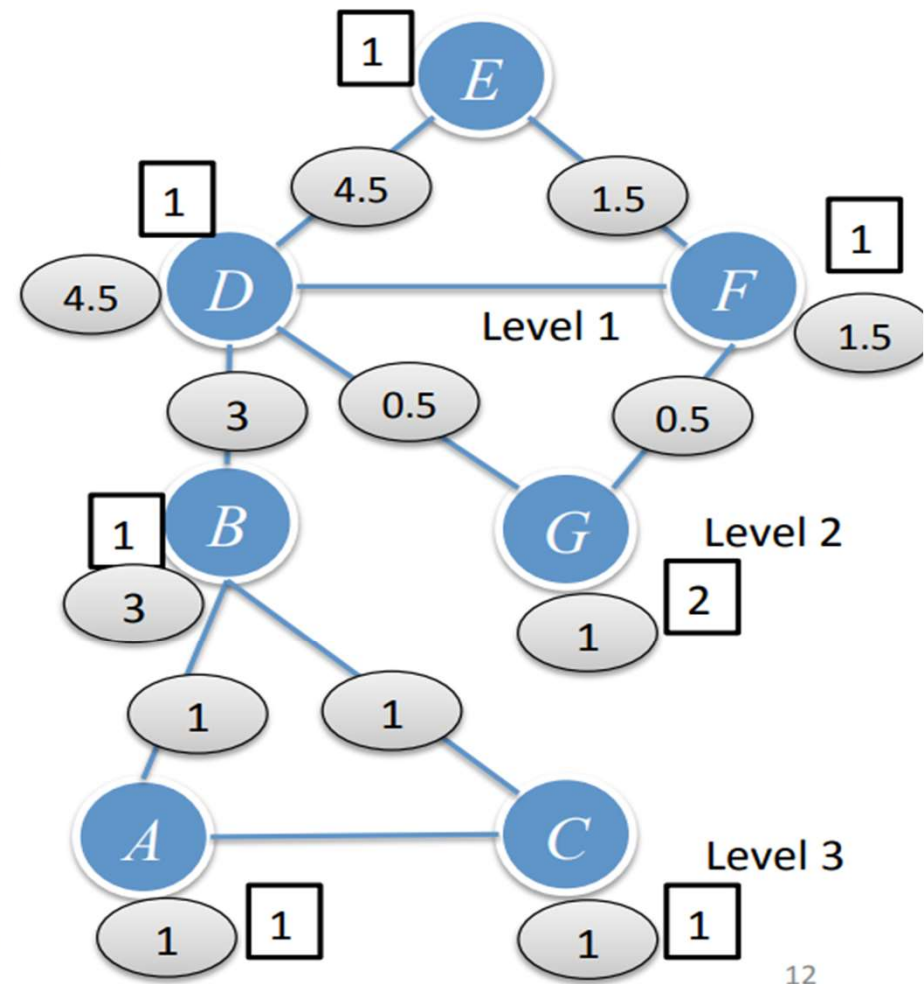
Step 3 – credit sharing:

- Each leaf node gets credit 1
- Each non-leaf node gets $1 + \text{sum}(\text{credits of the DAG edges to the level below})$
- Credit of DAG edges: Let Y_i ($i=1, \dots, k$) be parents of Z , $p_i = \text{label}(Y_i)$

$$\text{credit}(Y_i, Z) = \frac{\text{credit}(Z) \times p_i}{(p_1 + \dots + p_k)}$$
- Intuition: a DAG edge $Y_i Z$ gets the share of credit of Z proportional to the #of shortest paths from X to Z going through $Y_i Z$

Finally: Repeat Steps 1, 2 and 3 with each node as root. For each edge, $\text{betweenness} = \text{sum credits obtained in all iterations} / 2$

Calculate betweenness of edges



5. Using betweenness to find communities:

The betweenness scores for the edges of a graph behave something like a distance measure on the nodes of the graph. It is not exactly a distance measure, because it is not defined for pairs of nodes that are unconnected by an edge, and might not satisfy the triangle inequality even when defined. However, we can cluster by taking the edges in order of increasing betweenness and add them to the graph one at a time. At each step, the connected components of the graph form some clusters. The higher the betweenness we allow, the more edges we get, and the larger the clusters become.

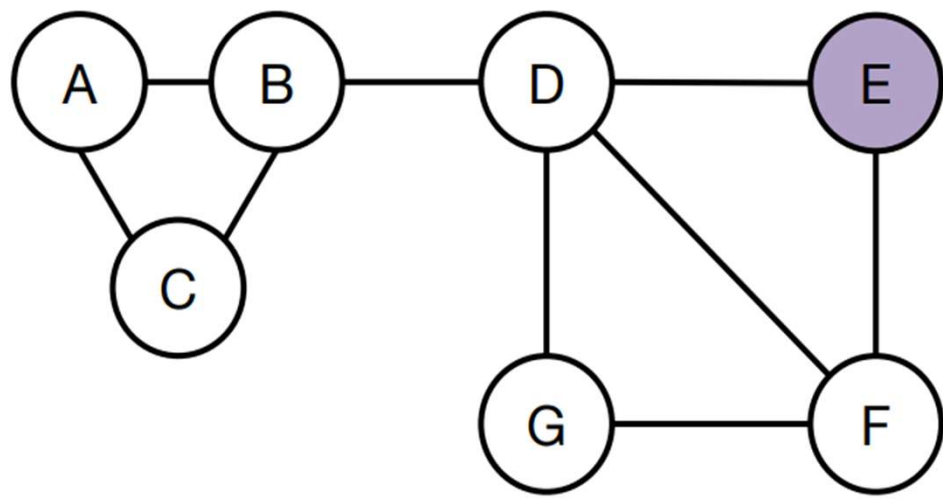
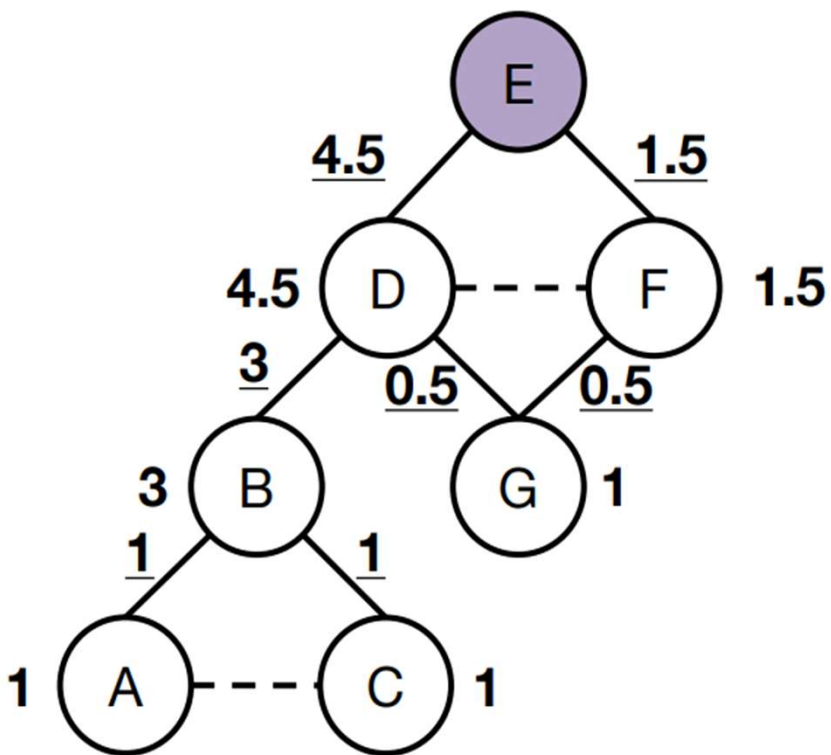
More commonly, this idea is expressed as a process of edge removal. Start with the graph and all its edges; then remove edges with the highest betweenness, until the graph has broken into a suitable number of connected components.

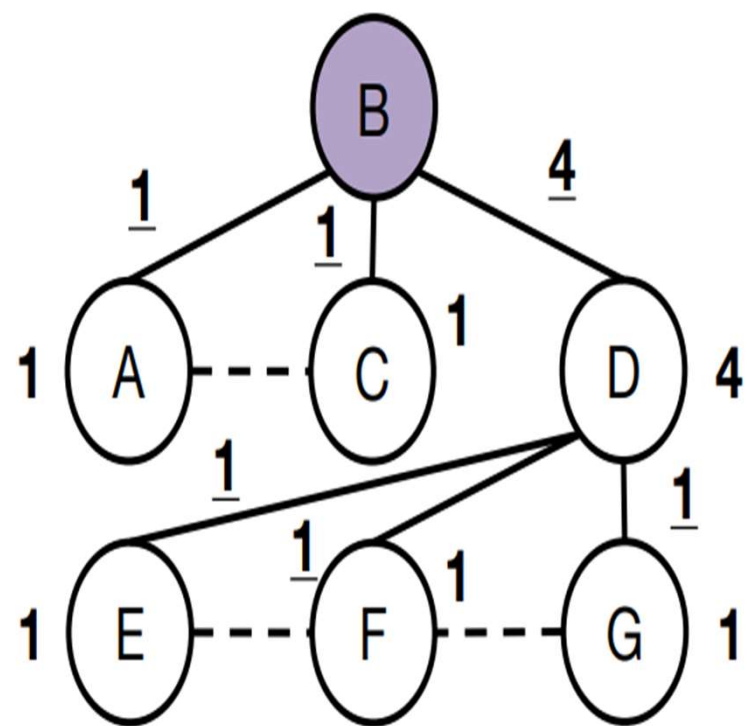
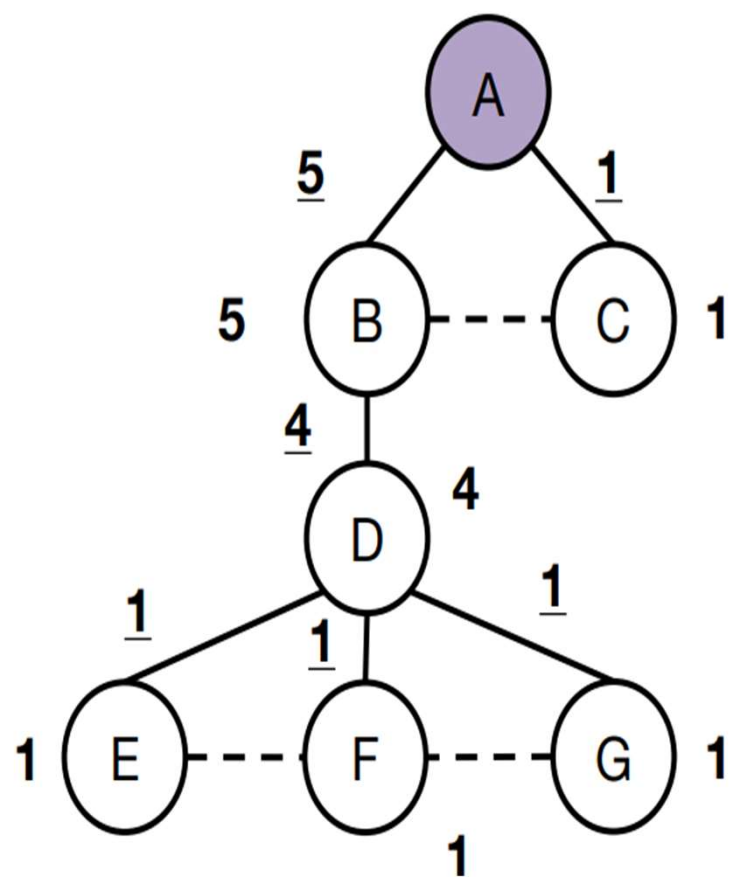
Computation in practice

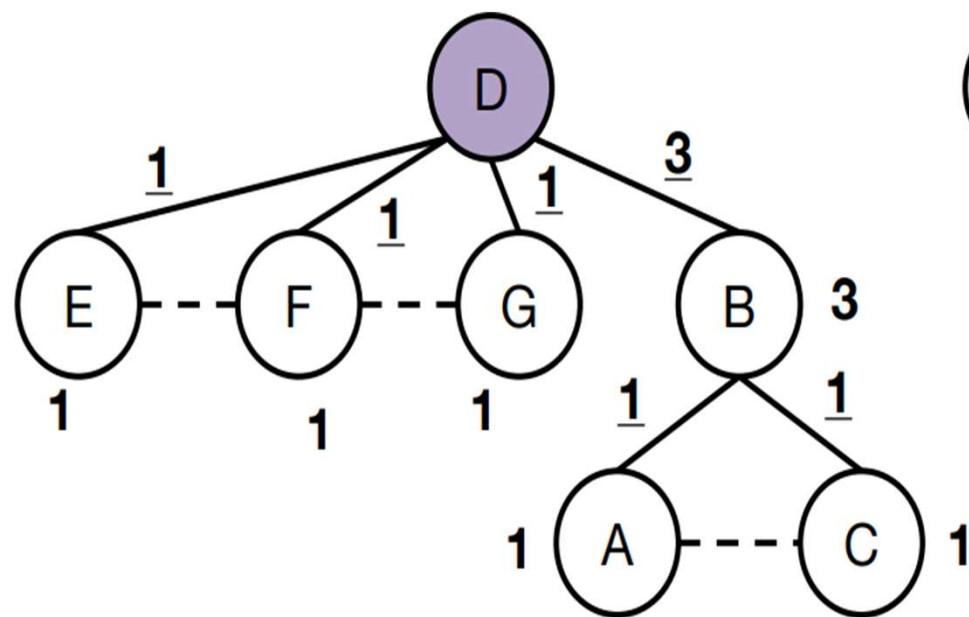
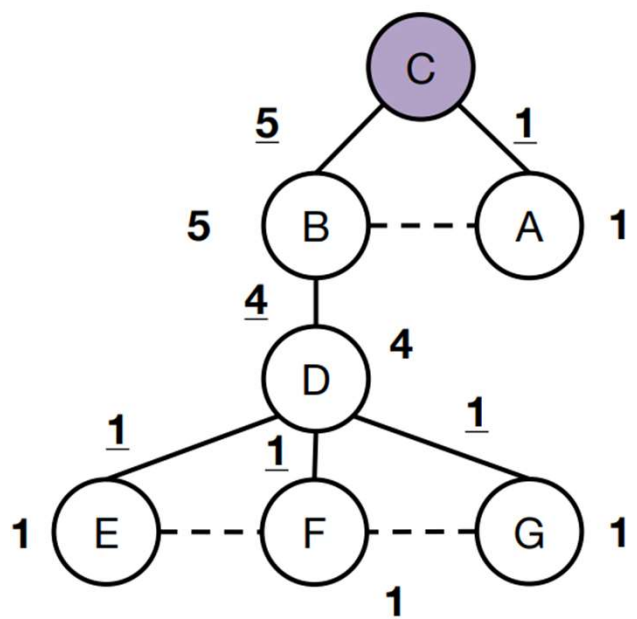
- Complexity: n nodes, e edges
 - BFS starting at each node: $O(e)$
 - Do it for n nodes
 - Total: $O(ne)$ time
 - Very expensive
- Method in practice
 - Choose a random subset W of the nodes
 - Compute credit of each edge starting at each node in W
 - Sum and compute betweenness
 - A reasonable approximation

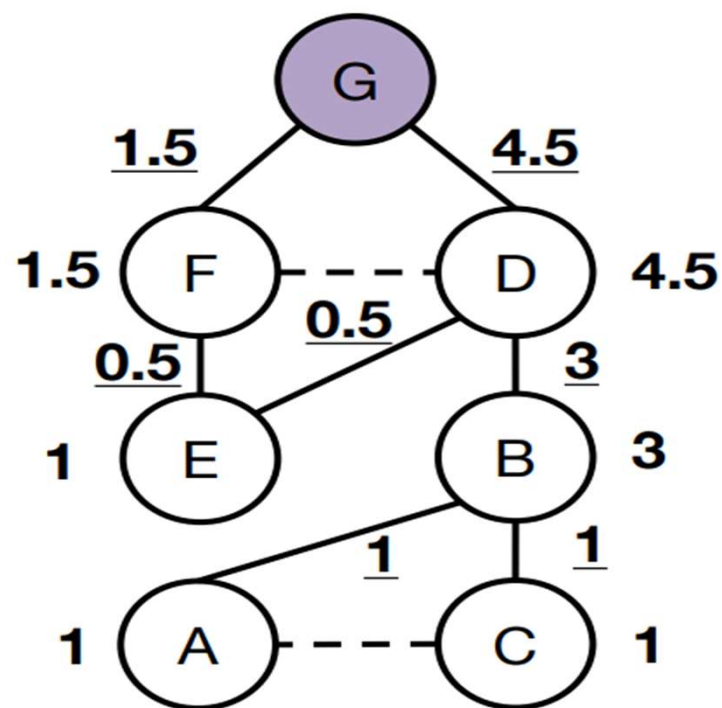
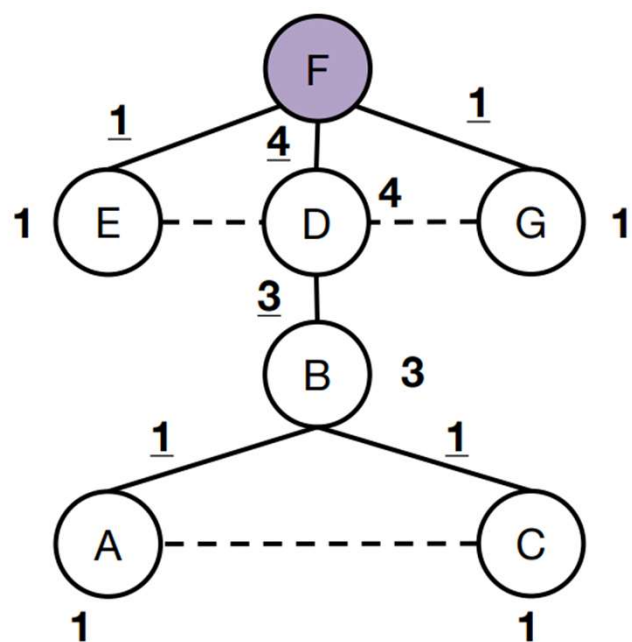
Example: Find Betweenness of edges using Girvan-Newman algorithm. Generate communities of SNG

Example (E, Credits.6)







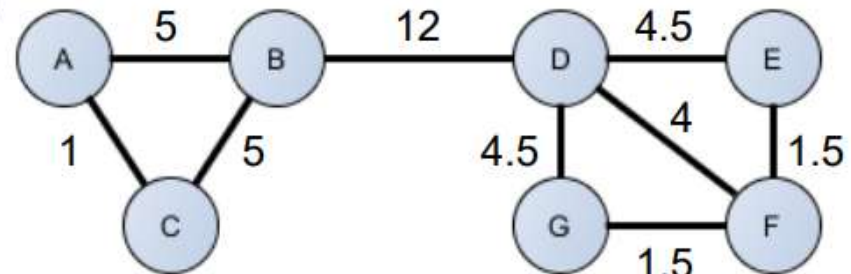


Sum Contributions (/2)

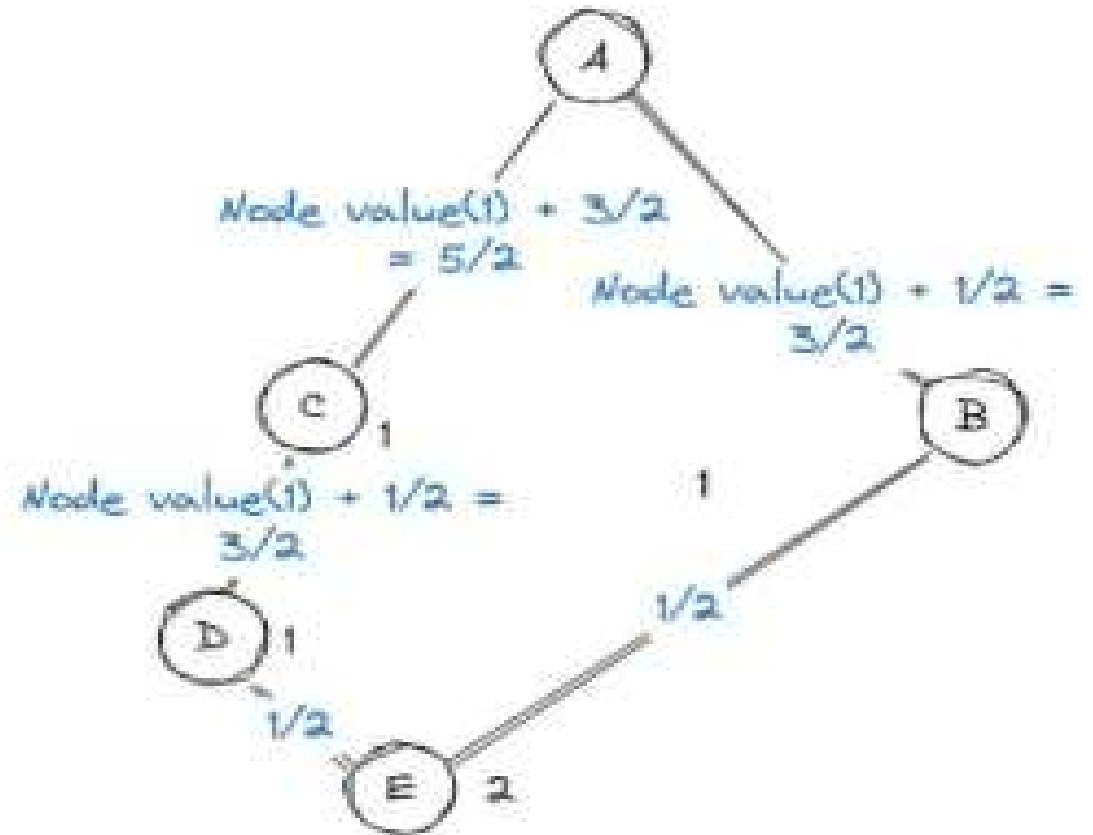
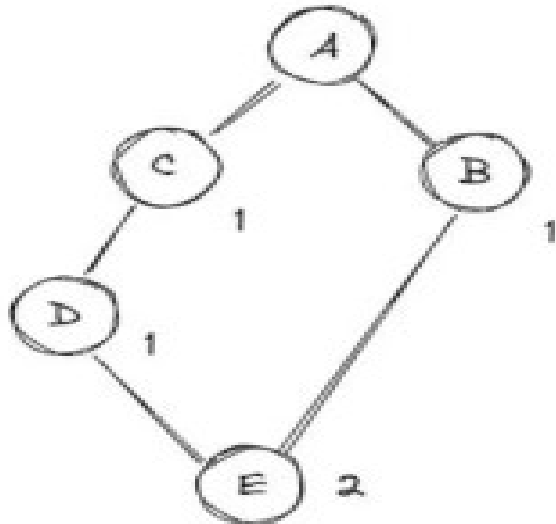
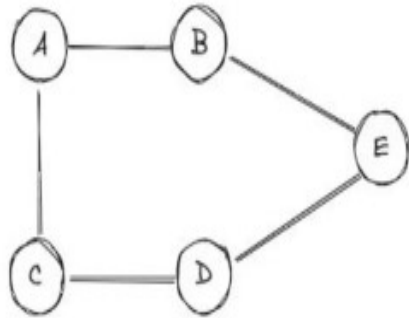
| | AB | AC | BC | BD | DE | DG | DF | EF | GF |
|----|----|----|----|----|-----|-----|----|-----|-----|
| A | 5 | 1 | | 4 | 1 | 1 | 1 | | |
| B | 1 | | 1 | 4 | 1 | 1 | 1 | | |
| C | | 1 | 5 | 4 | 1 | 1 | 1 | | |
| D | 1 | | 1 | 3 | 1 | 1 | 1 | | |
| E | 1 | | 1 | 3 | 4.5 | 0.5 | | 1.5 | 0.5 |
| F |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

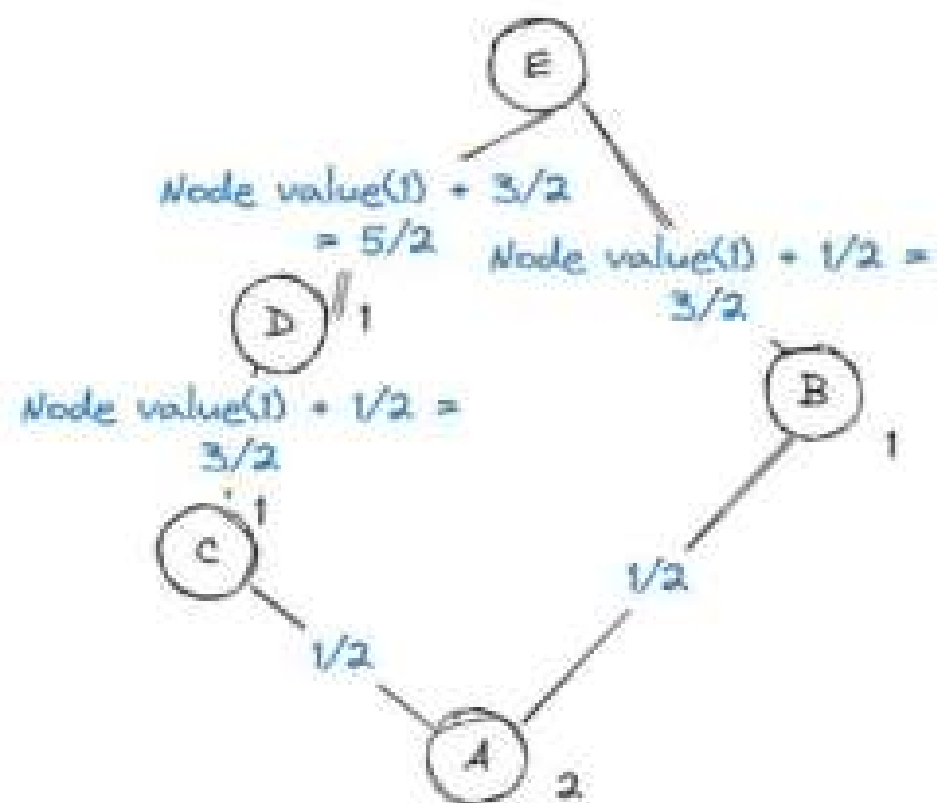
Remove edges, starting by highest betweenness:

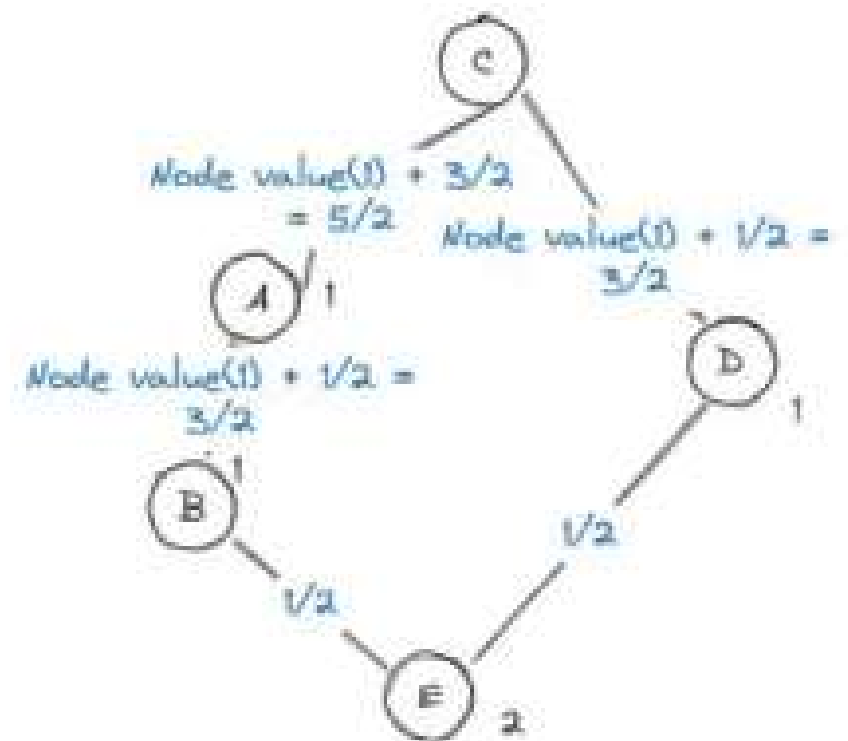
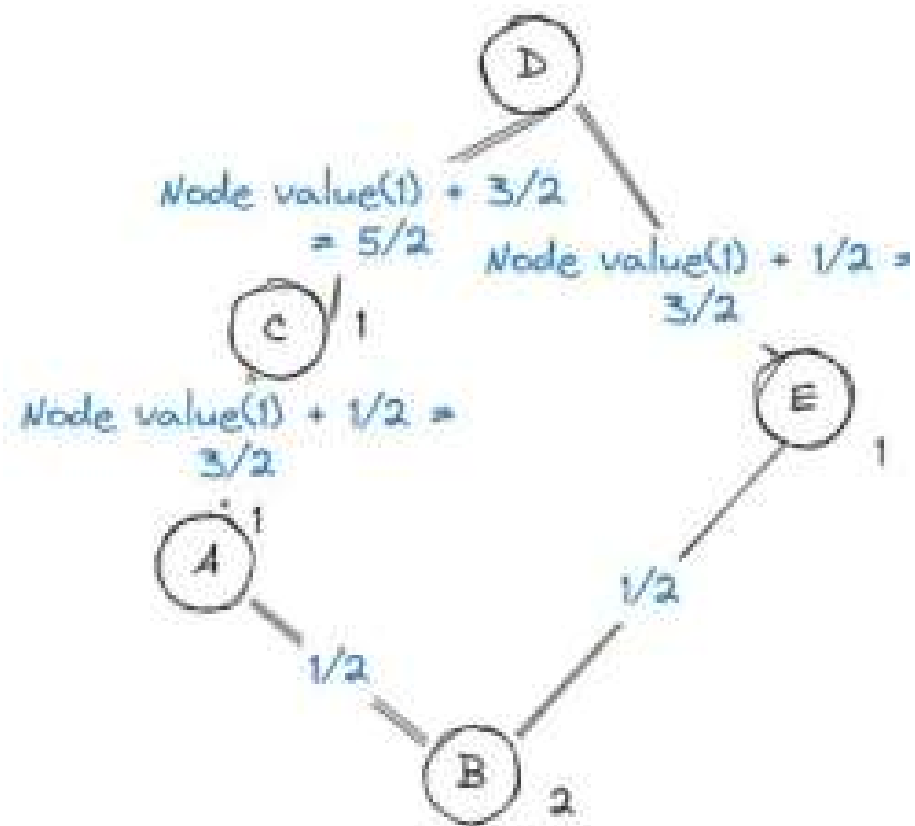
- 1. Remove (B,D)
→ Communities {A,B,C} and {D,E,F,G}
- 2. Remove (A,B), (B,C), (D,G), (D,E), (D,F)
→ Communities {A,C} and {E,F,G}
Node B and D are encapsulated as ,traitors' of communities


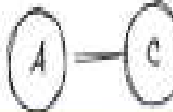


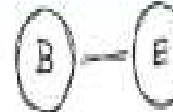


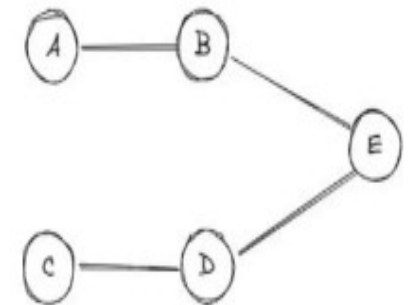
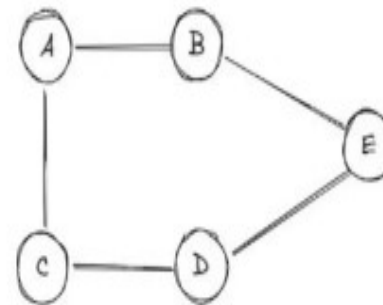
Example: Find community detection for given graph



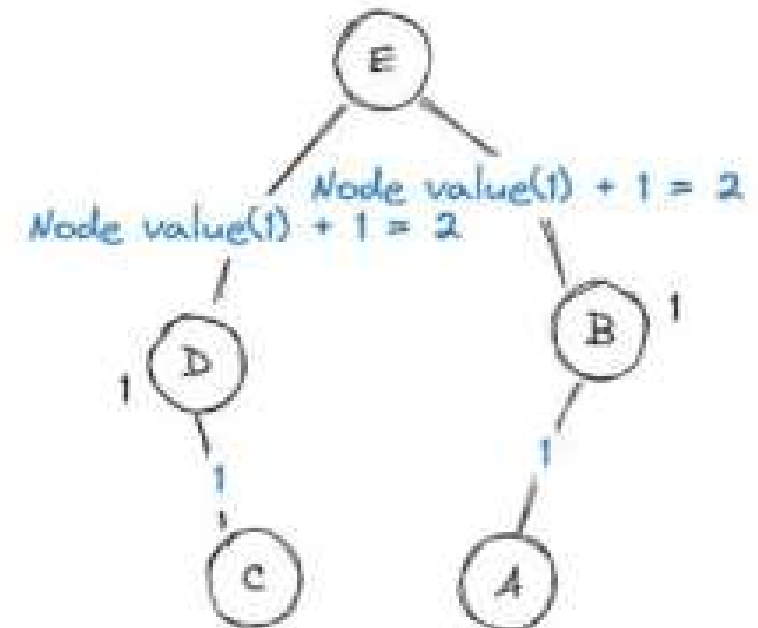
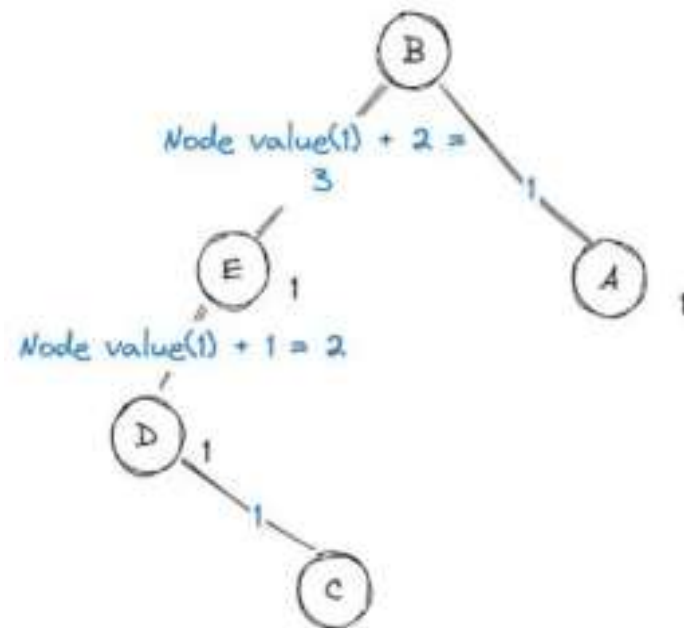
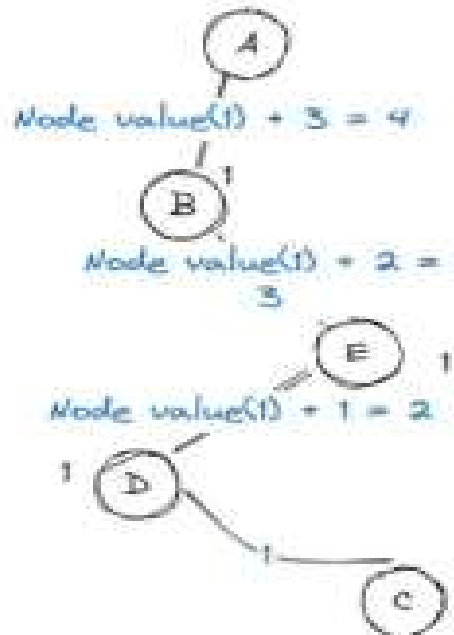


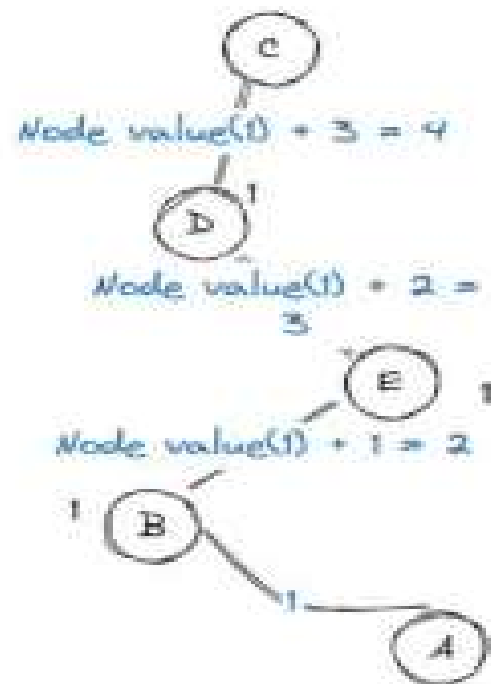
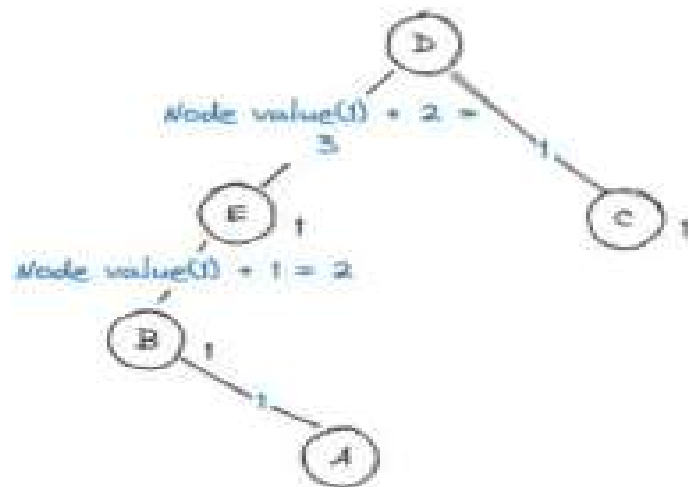


| | | |
|--|---|---------|
|  | $\frac{3}{2} + \frac{5}{2} + \frac{1}{2} + \frac{1}{2} + \frac{3}{2}$ | $= 6.5$ |
|  | $\frac{5}{2} + \frac{3}{2} + \frac{1}{2} + \frac{3}{2} + \frac{5}{2}$ | $= 8.5$ |
|  | $\frac{3}{2} + \frac{1}{2} + \frac{3}{2} + \frac{5}{2} + \frac{3}{2}$ | $= 7.5$ |
|  | $\frac{1}{2} + \frac{1}{2} + \frac{5}{2} + \frac{3}{2} + \frac{3}{2}$ | $= 6.5$ |
|  | $\frac{1}{2} + \frac{3}{2} + \frac{3}{2} + \frac{1}{2} + \frac{1}{2}$ | $= 4.5$ |



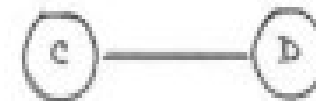
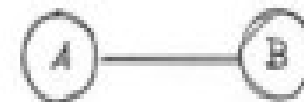
IInd- iteration



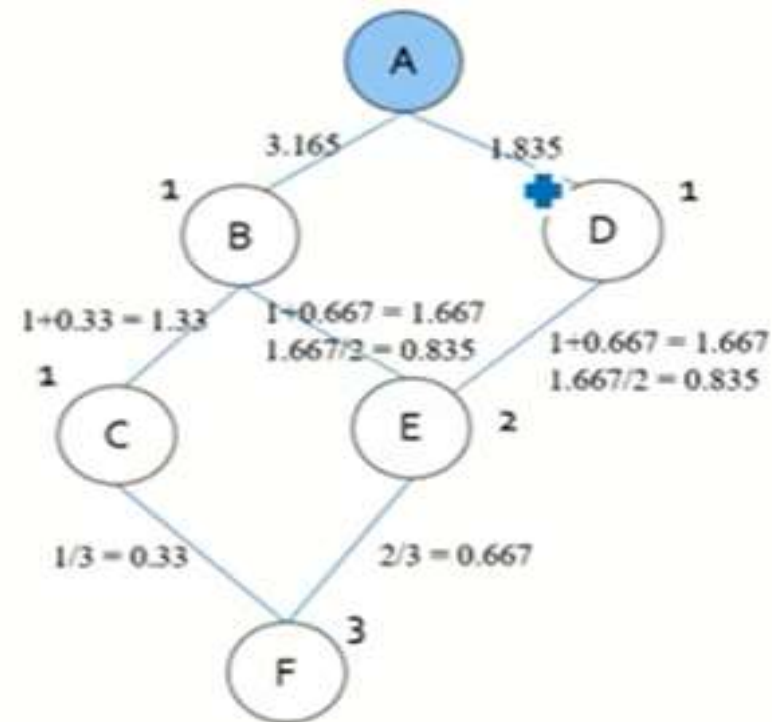
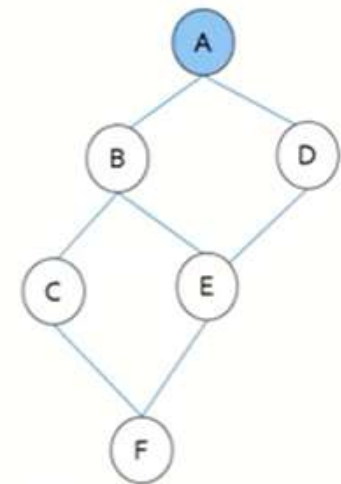
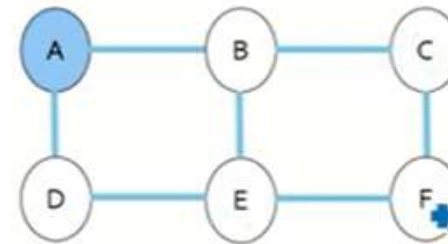


Edge Betweenness Value (2st iter)

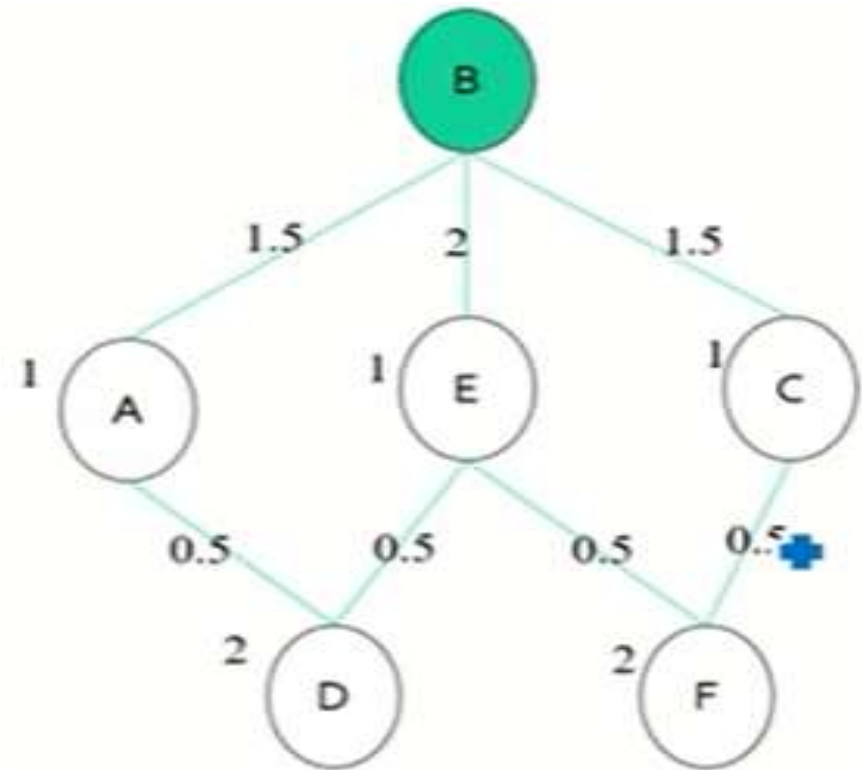
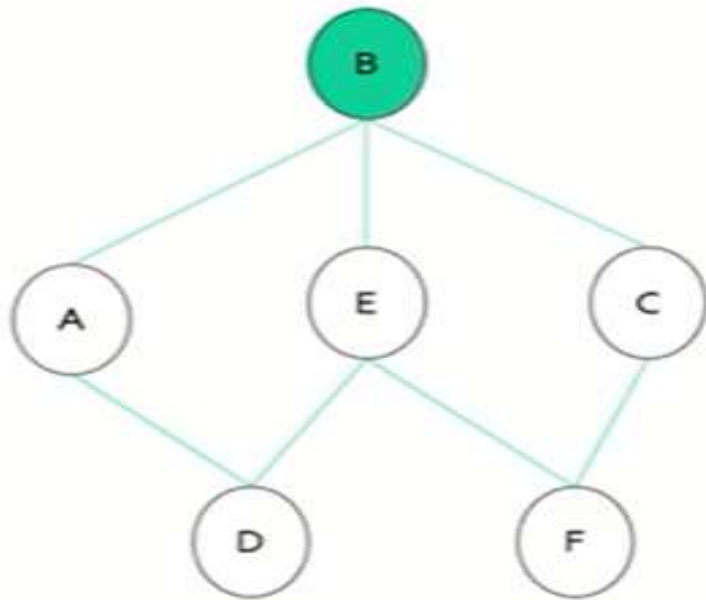
| | | |
|--|---------------------|------|
| | $4 + 1 + 1 + 1 + 1$ | = 8 |
| | $3 + 1 + 1 + 1 + 4$ | = 8 |
| | $2 + 2 + 2 + 3 + 3$ | = 12 |
| | $1 + 3 + 2 + 2 + 2$ | = 12 |



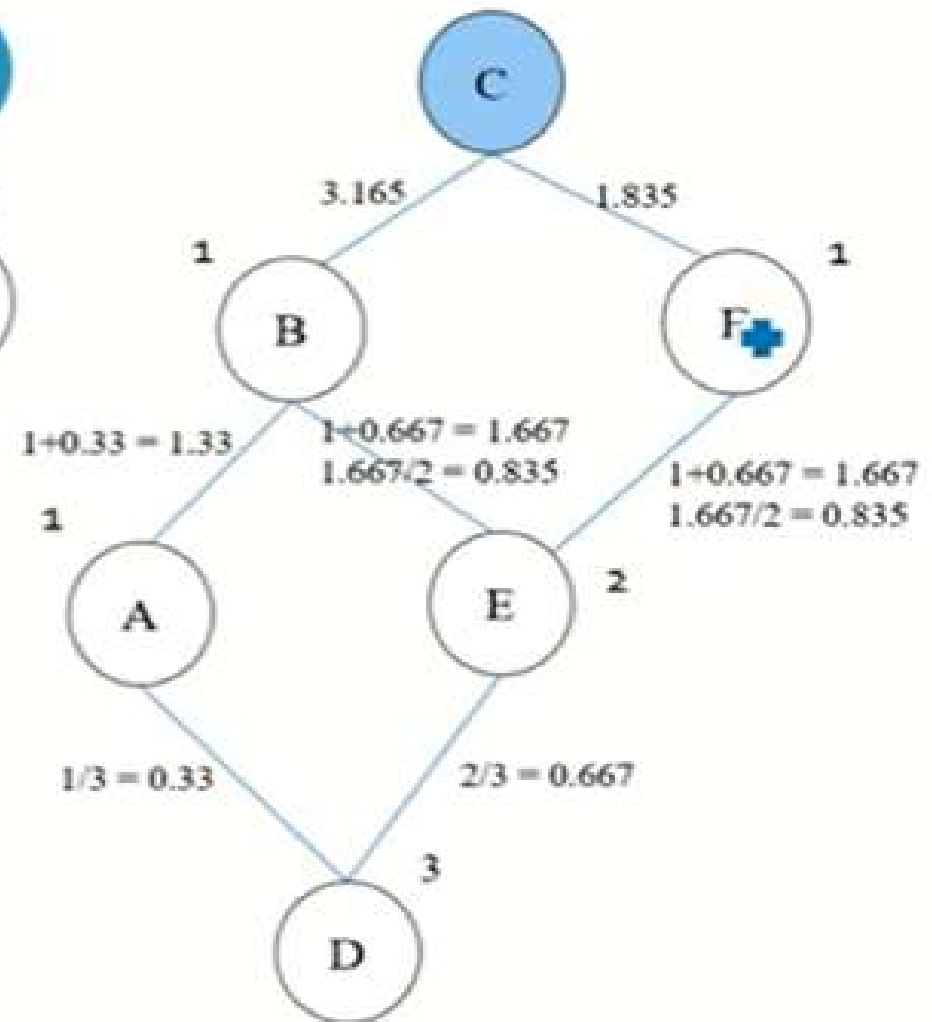
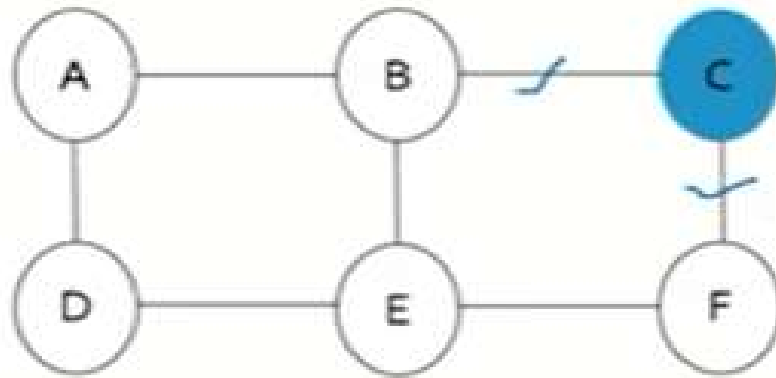
Example : For the given graph calculate the betweenness centrality for the edges, solve using Girvan Newman.



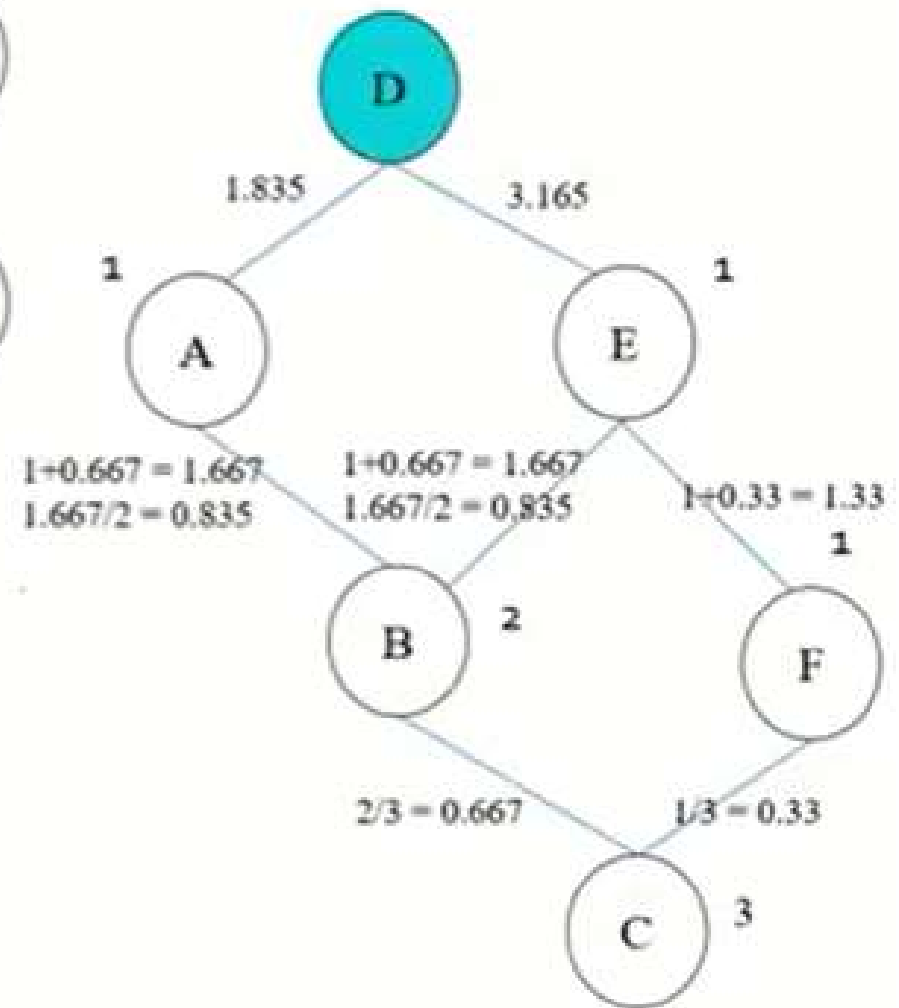
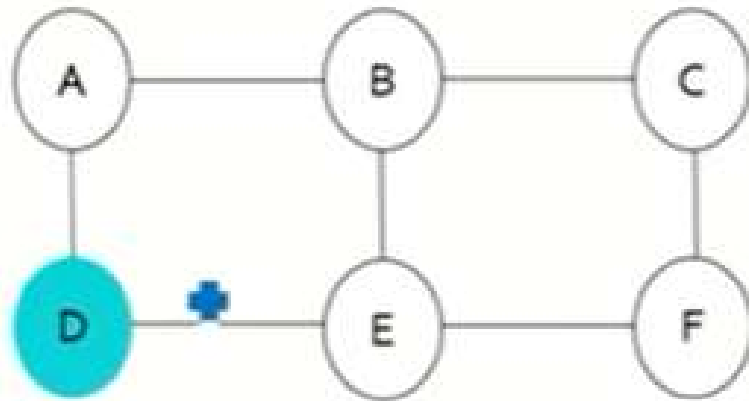
Example: cont.....

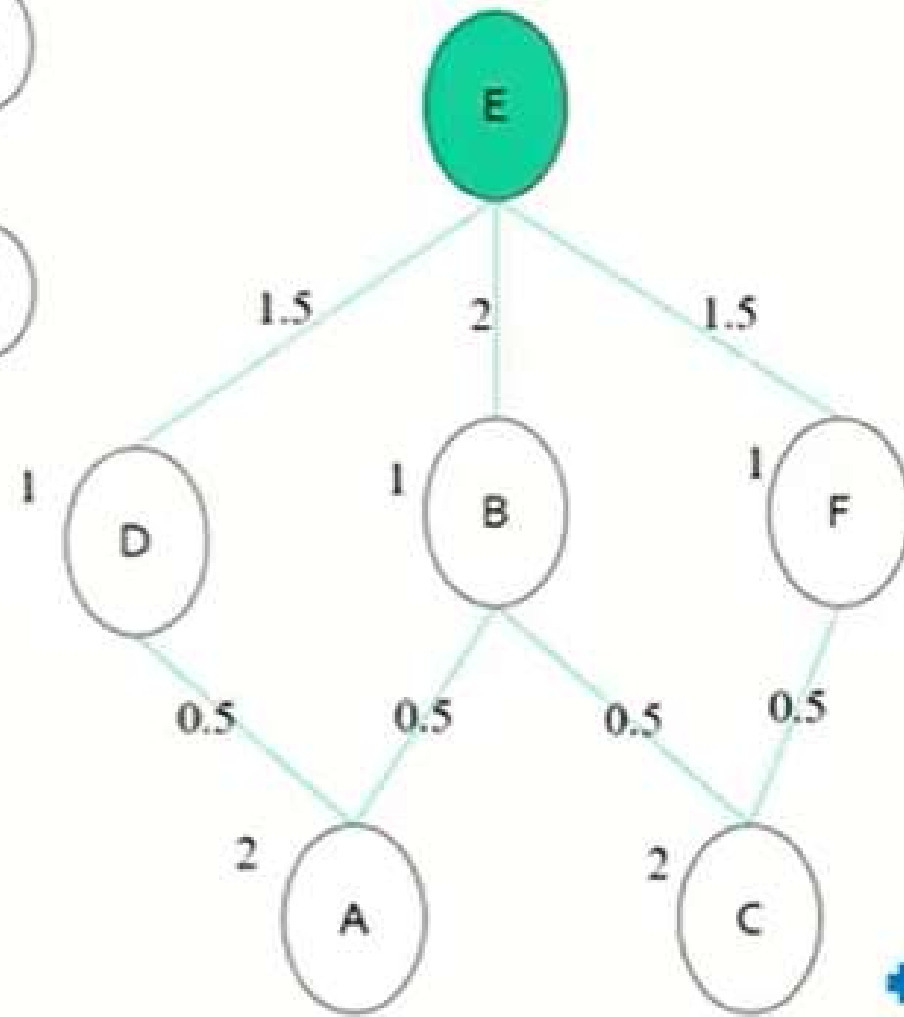
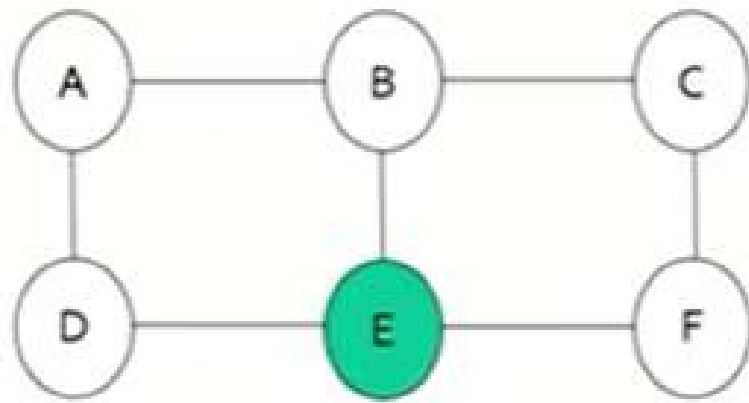


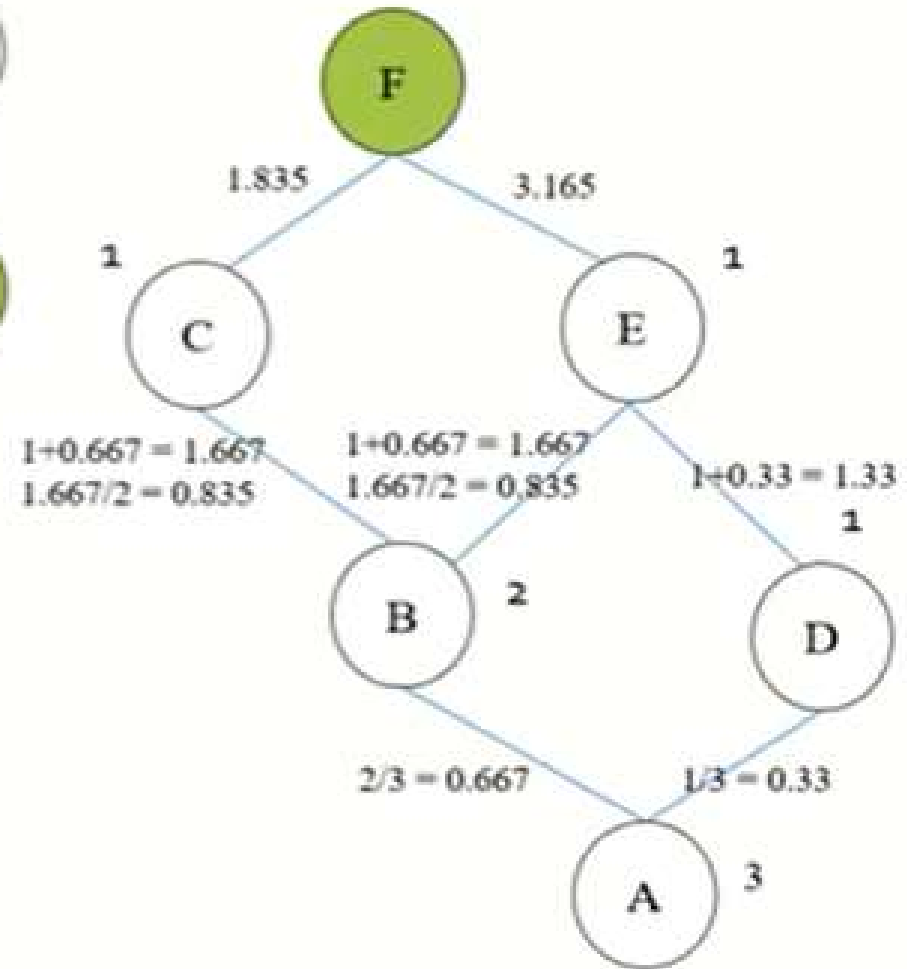
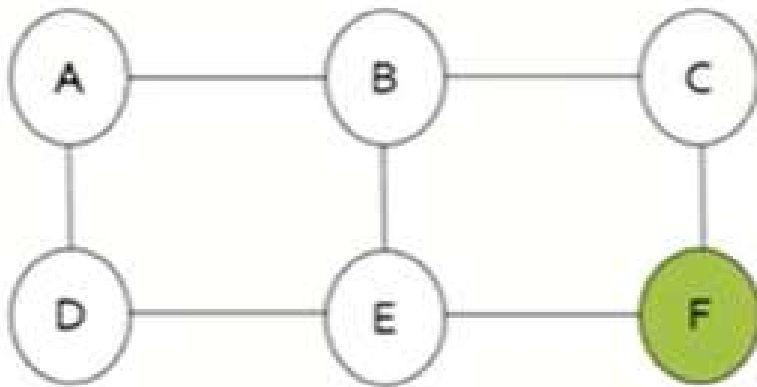
Example Node C as root

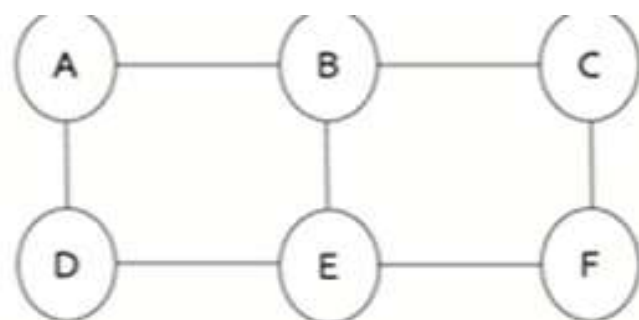


Example

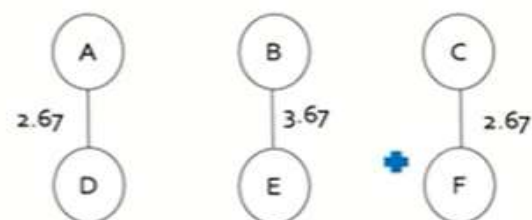
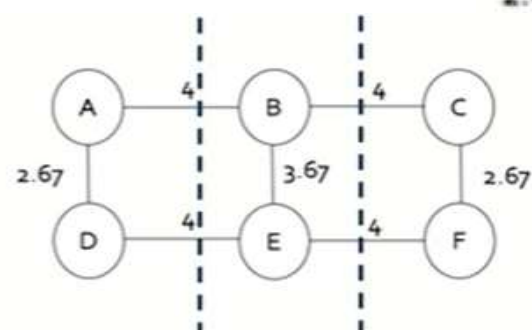
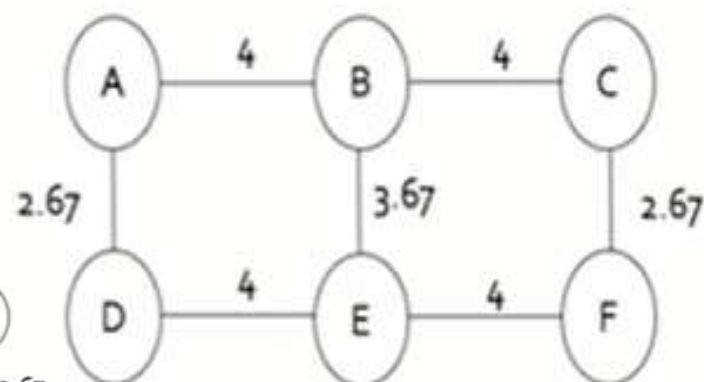
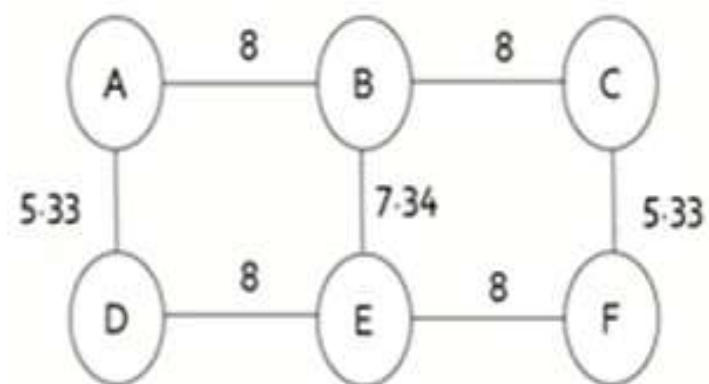








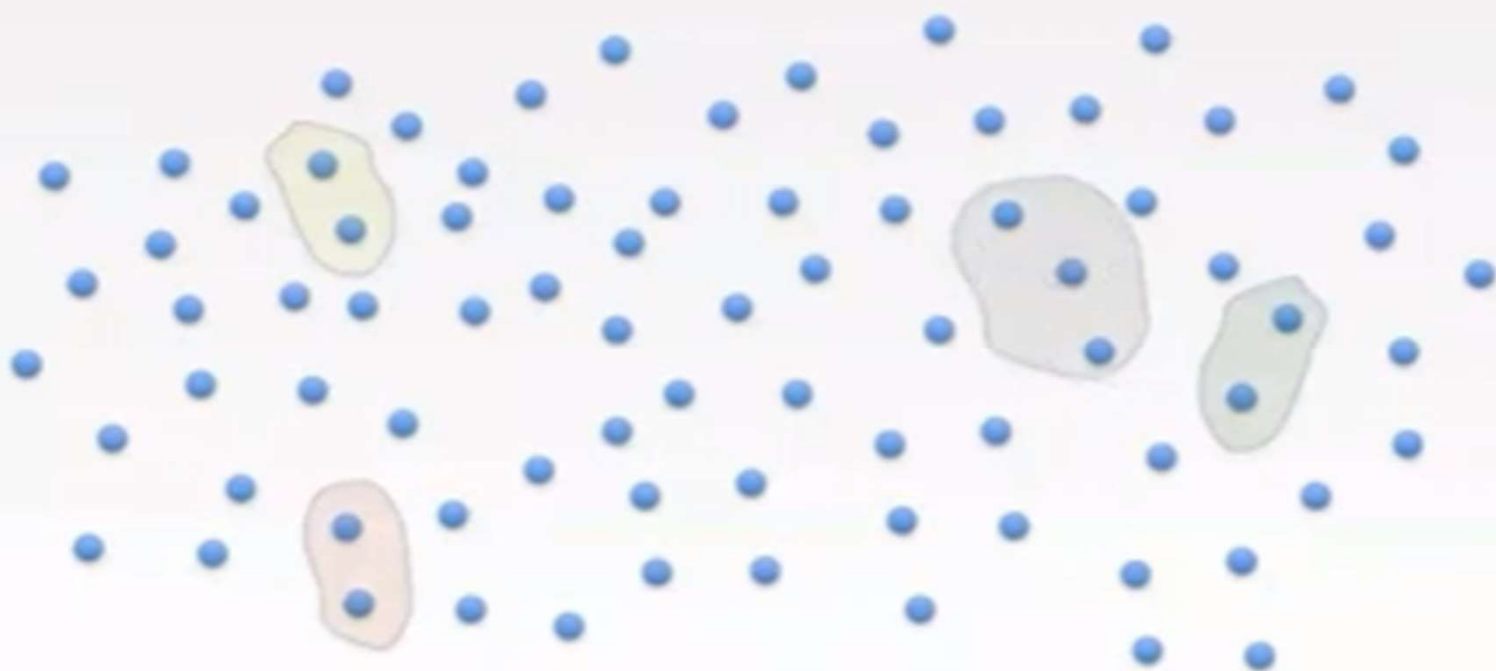
| EDGES | EDGE BETWEENNESS |
|-------|--|
| AB | $3.165+1.5+1.33+0.835+0.5+0.667 = 8$ |
| AD | $1.835+0.5+0.33+1.835+0.5+0.33 = 5.33$ |
| BC | $3.165+1.5+1.33+0.835+0.5+0.667 = 8$ |
| BE | $0.835+2+0.835++0.835+2+0.835 = 7.34$ |
| CF | $1.835+0.5+0.33+1.835+0.5+0.33 = 5.33$ |
| DE | $3.165+1.5+1.33+0.835+0.5+0.667 = 8$ |
| EF | $3.165+1.5+1.33+0.835+0.5+0.667 = 8$ |



Finding Communities using Betweenness

Method 1: (bottom - up)

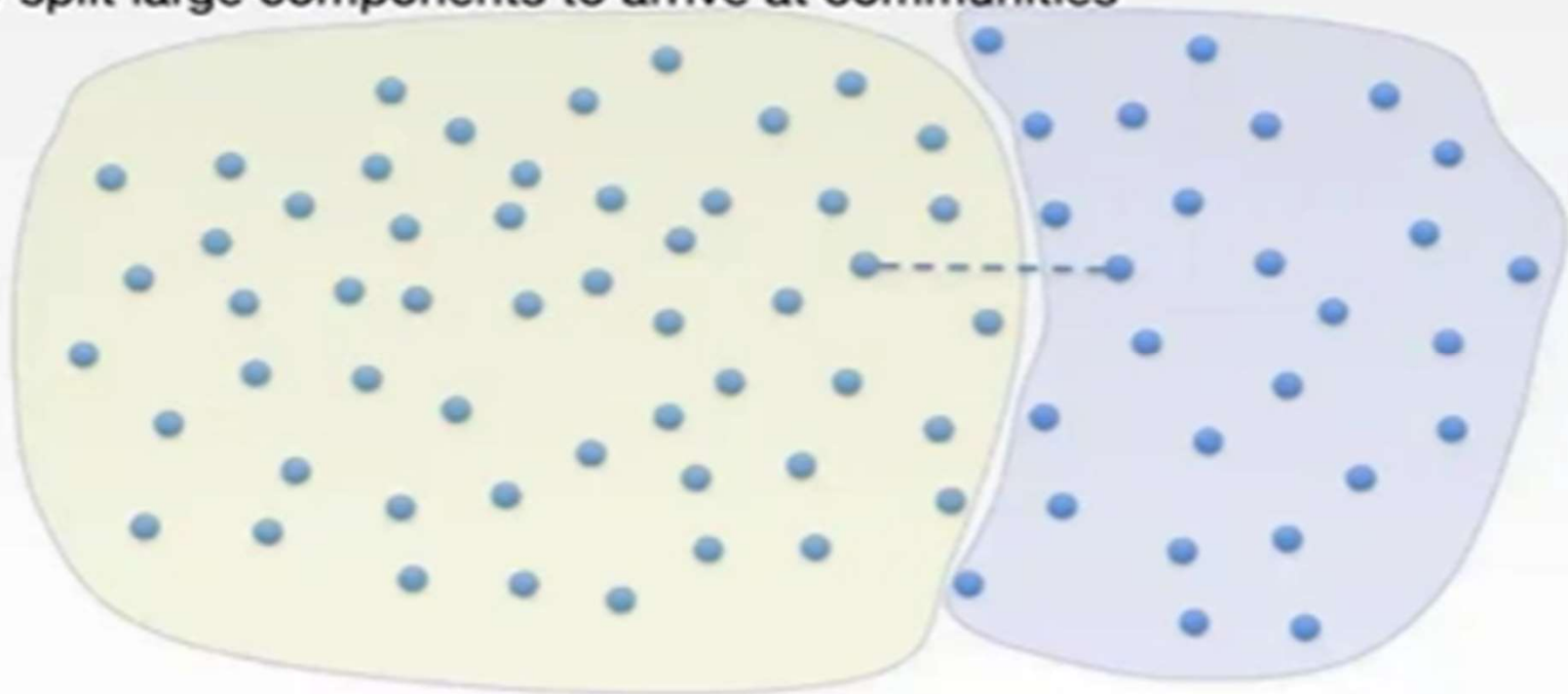
- Keep adding edges (among existing ones) starting from lowest betweenness
- Gradually join small components to build large connected components



Finding Communities using Betweenness

Method 2: (top - down)

- Start from all existing edges. The graph may look like one big component.
- Keep removing edges starting from highest betweenness
- Gradually split large components to arrive at communities



Bottom - up vs top - down

- Consider our desired clustering, where we only have edges within clusters and not across
- Let E_c be set of edges within clusters, E_o be the set of edges across clusters and E be the set of actually existing edges
- Then, we have $|E| = |E_c| + |E_o|$
- Question: is $|E_c| > |E_o|$, or otherwise?
- In other words, to get to our desired clustering, are we likely to *add* more edges starting from none, or *delete* more edges starting from all?
- If the number of clusters is not too high (that's the usual approach for clustering), E_c is likely to be much bigger
- Then top-down approach will be more efficient

