MOD-3 Honours

Graph Analytics

Social Network

Social Graph

A **social graph** is a way to visualize and analyze social networks, where individuals (entities) are represented as nodes and their relationships are represented as edges.

Essential Characteristics of a Social Network

1. **Collection of Entities**: The network consists of various participants, such as people, organizations, or accounts.
2. **Relationships**: At least one type of relationship exists among the entities, which is crucial for defining the network.
3. **Non-randomness or Locality**: There's an assumption that connections aren't random; individuals tend to connect with others nearby in some sense, either geographically or socially.

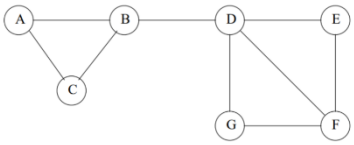
Modeling as Graphs

- **Nodes and Edges**:
    - **Nodes**: Represent the entities (people or accounts).
    - **Edges**: Connect nodes based on their relationships (e.g., friendship, following).
- **Degrees of Relationships**: If a relationship has a weight or significance (like the strength of a friendship), this can be represented by labeling the edges (e.g., a scale of closeness).

Types of Social Graphs

- **Undirected Graphs**:
    - These graphs show mutual relationships (like friendships on Facebook), where the connection goes both ways.
- **Directed Graphs**:
    - These represent one-way relationships (like following someone on Twitter or Instagram), where one node follows another, but the reverse isn't necessarily true.

Example of a social graph



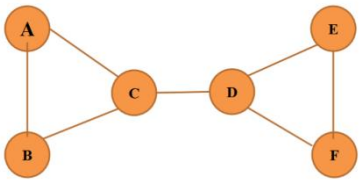
Betweenness Centrality

- The Vertex betweenness centrality BC(v) of a vertex v is defined as follow

$$BC(v) = \sum_{u,v \in V} \frac{\sigma_{uw}(v)}{\sigma_{uw}}$$

- σ_{uw} = Total Number of shortest path between node u and w
- $\sigma_{uw}(v)$=total number of shortest paths between node u and w that pass through v

Lets find Betweenness Centrality for C in the below diagram



	σ_{uw}	$\sigma_{uw}(v)$	$\sigma_{uw}(v)/\sigma_{uw}$
(A,B)	1	0	0
(A,D)	1	1	1
(A,E)	1	1	1
(A,F)	1	1	1
(B,D)	1	1	1
(B,E)	1	1	1
(B,F)	1	1	1
(D,E)	1	0	0
(D,F)	1	0	0
(E,F)	1	0	0

Here all possible edges are their

If Connection between A & B Exist or not

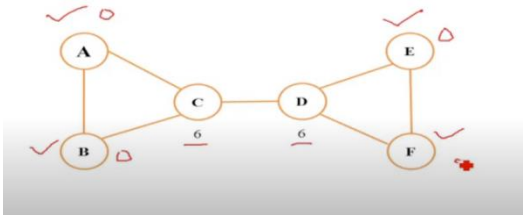
1 if it exist
0 if it does not exit

If the node C comes in between A & B (Example for first)

1 if C Comes in between the connection
0 if C does not come in between the connection

	σ_{uw}	$\sigma_{uw}(v)$	$\sigma_{uw}(v)/\sigma_{uw}$
(A, B)	1	0	0
(A, D)	**1**	**1**	**1**
(A, E)	**1**	**1**	**1**
(A, F)	**1**	**1**	**1**
(B, D)	**1**	**1**	**1**
(B, E)	**1**	**1**	**1**
(B, F)	**1**	**1**	**1**
(D, E)	1	0	0
(D, F)	1	0	0
(E, F)	1	0	0

Betweenness centrality for C = 6

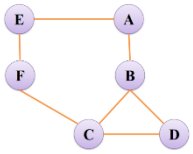


D will also have 6 Betweenness Centrality becaz structure of D is same like C

A,B,E,F Will have 0 Betweenness Centrality becaz no nodes are between them.
They are side node's.

<u>Always consider the shortest path.</u>

Example #2

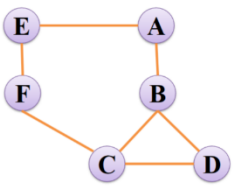


Find Betweenness Centrality of A

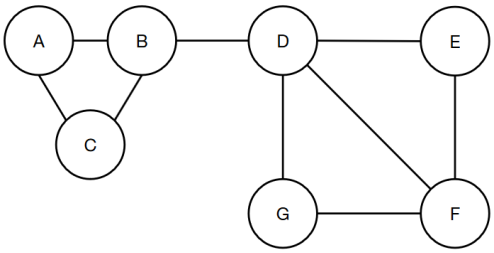
Betweenness Centrality for A= ?

- BC = 0/1 = 0
- BD = 0/1 = 0
- BE = 1/1 = 1
- BF = 0/1 = 0
- CD = 0/1 = 0
- CE = 0/1 = 0
- CF = 0/1 = 0
- DE = 1/2 =0.5
- DF = 0/1 = 0
- EF = 0/1 = 0

Betweenness Centrality for A= ?



Therefore betweenness Centrality of A is 1.5



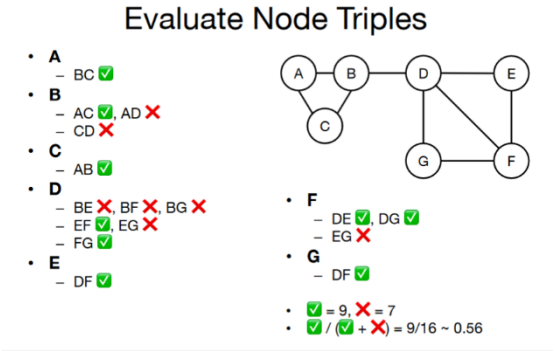
- Nodes: 7
- Edges: 9
- Avg Connectivity: 9/21 ~ 0.43

3. **Maximum possible edges**: In an undirected graph, the maximum number of edges that can exist between n nodes is given by the combination formula $\binom{n}{2} = \frac{n(n-1)}{2}$. For 7 nodes:

$$\binom{7}{2} = \frac{7 \times 6}{2} = 21$$

4. **Average connectivity** (edge density) is the ratio of actual edges to the maximum possible edges:

$$\text{Avg Connectivity} = \frac{\text{Number of Edges}}{\text{Max Possible Edges}} = \frac{9}{21} \approx 0.43$$



The image explains the calculation of **Actual Locality** by evaluating **node triples**—groups of three nodes that form possible connections. Here's a breakdown of what's happening:

Node Triples Evaluation

Each node in the graph is checked for its connections (edges) with two other nodes to form a triangle (a fully connected subgraph of three nodes). The goal is to determine if the neighbors of each node are connected (i.e., if there is an edge between them).

- **Green checkmarks (✓)** indicate that the connection exists, meaning the two nodes are connected.
- **Red crosses (❌)** indicate that the connection between those two nodes does not exist.

Breakdown by Node:

Node A:

- **BC (✓)**: Nodes B and C are connected.

Node B:

- **AC (✓)**: Nodes A and C are connected.
- **AD (❌)**: Nodes A and D are not connected.
- **CD (❌)**: Nodes C and D are not connected.

Node C:

- **AB (✓)**: Nodes A and B are connected.

Node D:

- **BE (❌)**: Nodes B and E are not connected.
- **BF (❌)**: Nodes B and F are not connected.
- **BG (❌)**: Nodes B and G are not connected.
- **EF (✓)**: Nodes E and F are connected.
- **EG (❌)**: Nodes E and G are not connected.
- **FG (✓)**: Nodes F and G are connected.

Node E:

- **DF (✓)**: Nodes D and F are connected.

Node F:

- **DE (✓)**: Nodes D and E are connected.
- **DG (✓)**: Nodes D and G are connected.
- **EG (❌)**: Nodes E and G are not connected.

Node G:

- **DF (✓)**: Nodes D and F are connected.

Total Evaluation:

- **✓ = 9**: There are 9 connections (local edges) that exist.
- **❌ = 7**: There are 7 connections (local edges) that do not exist.

Formula for Actual Locality:

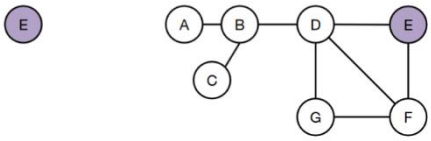
The formula for **Actual Locality** is the ratio of the number of existing connections (✓) to the total number of possible connections (✓ + ❌):

$$\text{Actual Locality} = \frac{\checkmark}{\checkmark + \times} = \frac{9}{9+7} = \frac{9}{16} \approx 0.56$$

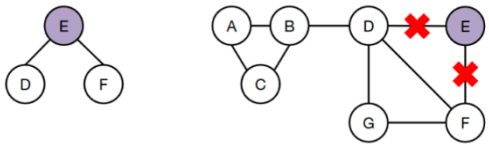
This gives the **Actual Locality** of 0.56, meaning 56% of the possible local connections (node triples) in the graph are actually connected. This shows that the graph has a higher degree of clustering than what would be expected by random chance.

Computing Graph Betweenness

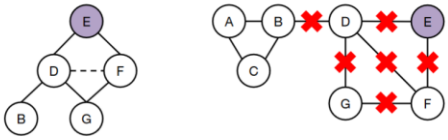
1) Example (E, BFS.0)



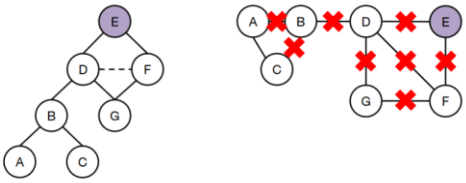
2)



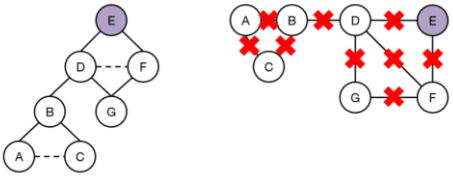
Example (E, BFS.2)



Example (E, BFS.3)



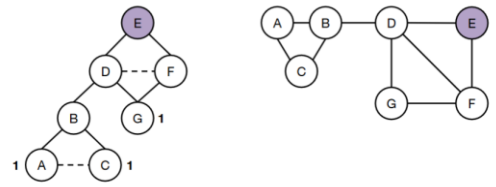
Example (E, BFS.Done)



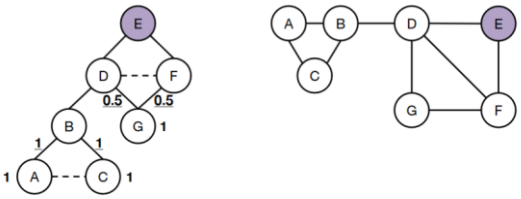
Computing BFS Betweenness

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

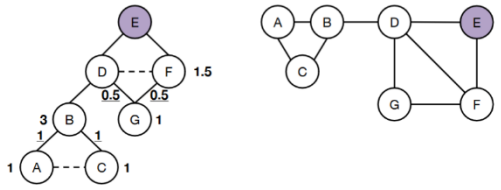
Example (E, Credits.1)



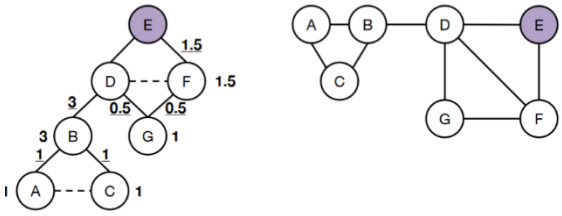
Example (E, Credits.2)



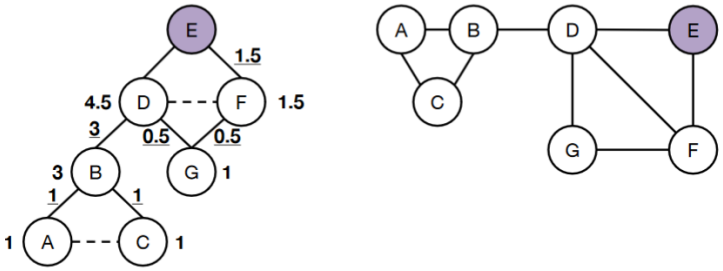
Example (E, Credits.3)



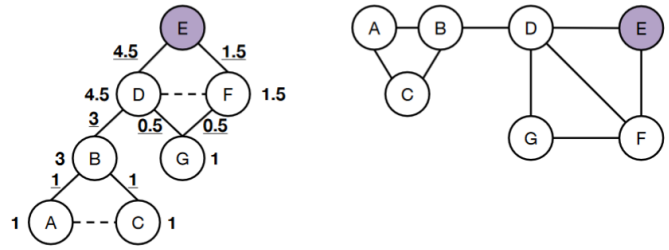
Example (E, Credits.4)



Example (E, Credits.5)

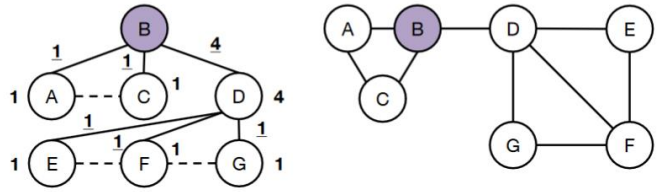


Example (E, Credits.6)

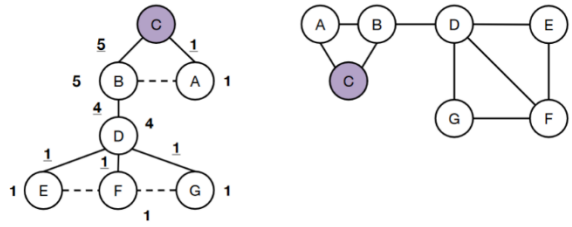


We can do this for A,B,C,D

Example (B, Credits)

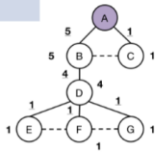


Example (C, Credits)



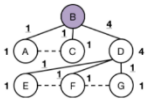
Sum Contributions (A)

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



Sum Contributions (B)

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



<u>We do for all A,B,C,D,E,F then we all them</u>

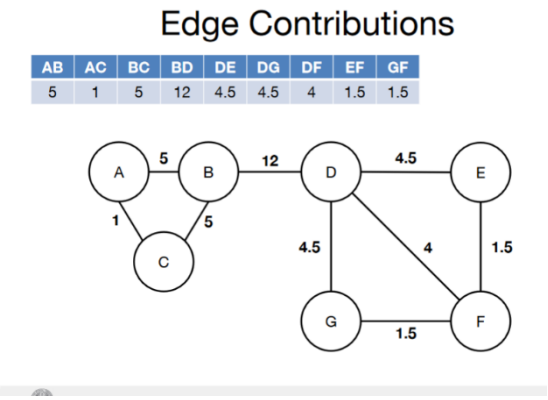
Sum Contributions (+)

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

Then we divide by 2

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



Girvan Newman

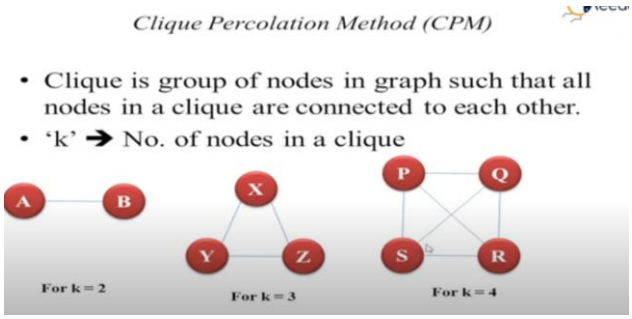
The **Girvan-Newman algorithm** is a method for detecting community structure in networks by iteratively removing edges with the highest betweenness centrality, which measures how many shortest paths pass through each edge. This process helps to identify and separate groups of densely connected nodes.

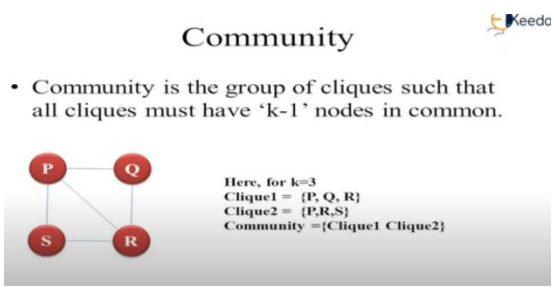
Repeat until no edge is left

1. Calculate edge betweenness for every edge in the graph.
2. Remove the edge with highest edge betweenness.
3. Calculate edge betweenness for remaining edges.
4. Connected components are communities.

After finding The edge contributions like above we will eliminate the edge with highest weightage.

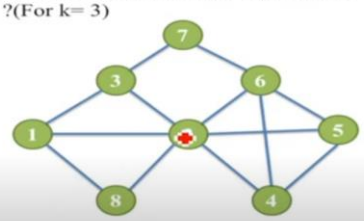
Short Cut

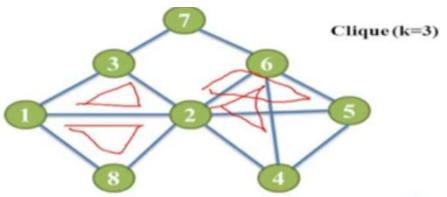




Example

Use clique percolation method (CPM) and find the communities ?(For k= 3)





Clique (k=3)

- {1,2,3}
- {1,2,8}
- {2,6,5}
- {2,6,4}
- {2,5,4}
- {4,5,6}

Simrank is to be added

$$\mathbf{v}' = \beta M \mathbf{v} + (1 - \beta)\mathbf{e}/n$$

- β is a chosen constant, usually in the range 0.8 to 0.9,
- e is a vector of all 1's with the appropriate number of components, and
- n is the number of nodes in the Web graph
- v is link matrix represented form web graph
- v' is new, calculated page rank matrix

Bipartite Simrank (mutually-reinforcing rule)

1. **Rule 1:** People are similar if they purchase similar items
2. **Rule 2:** Items are similar if they are purchased by similar people

→ Rule 1 reinforces Rule 2, and vice versa

Example:

1. If frosting and eggs are similar, then A and B also similar.
2. If A and B are similar then frosting and eggs are similar.

Observation: We can magically see the similar of **sugar** and **flour**, even though there is no common customer.