Dynamic and Social Network Analysis

Lecture 4

Miray Kas
Bilkent University
Computer Engineering Department

Network Topology and Metrics

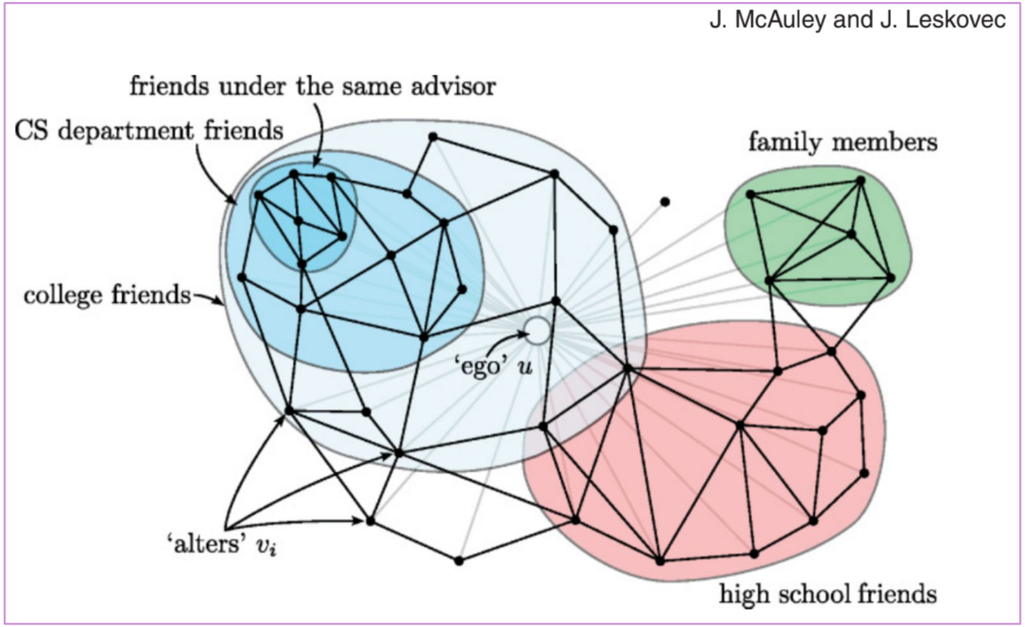
Commonly Discussed Network Topologies/Structures

- Ego Networks
- Hierarchical Networks
- Cliques
- Complete Graphs (Not common)
- Sparse Networks
- Small-World Networks

Ego Networks

- **Ego:** A focal person connected to everyone in a network.

- **Ego Network:** Social network from the ego's point of view.
    - Ego is surrounded by the **alters**
    - Each alter is nominated only by the **ego**
    - Egocentric **networks** are often referred to "perceived" or "cognitive" **networks**



(https://www-cs.stanford.edu/~jure/pubs/circles-tkdd14.pdf)

Ego Networks

Social Relations

Role-based

Kinship

Father of
Brother of

Other

Manager of
Colleague of
Friend of
Teacher of

Cognitive/Affective

Likes
Dislikes
Knows
Admires

Actions

Gives aids to
Sells to
Talks to

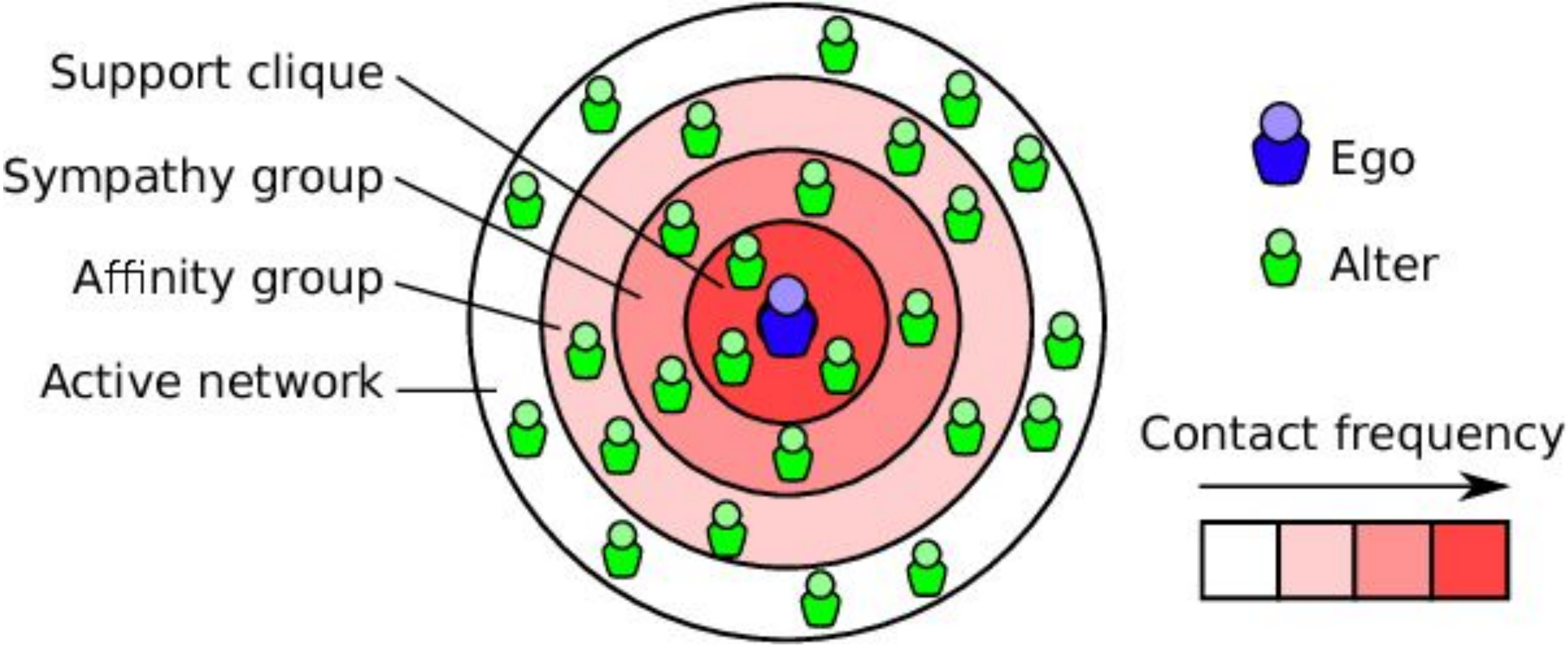
Observations on the Characteristics of Ego Networks

- **Strong ties are homophilous**
  - People feel/get closer with people that are like them

- **People with heterogeneous networks are "better off"**
  - They are more likely to find/get what they need
  - Particularly relevant for entrepreneurs

- **Sharing weak ties is important**
  - The stronger the tie between EGO and two of her alters, the greater the likelihood that the alters enjoy at least a weak tie.

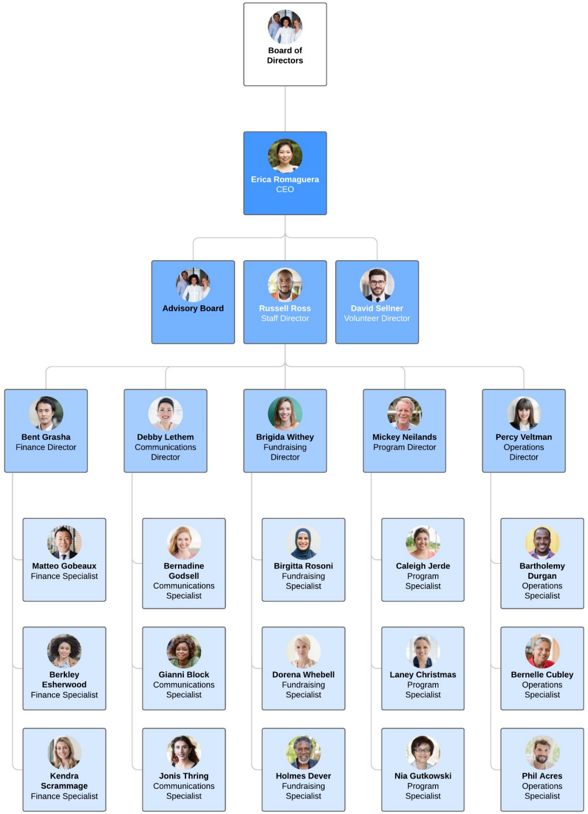
Another way of looking at EgoCentric Networks



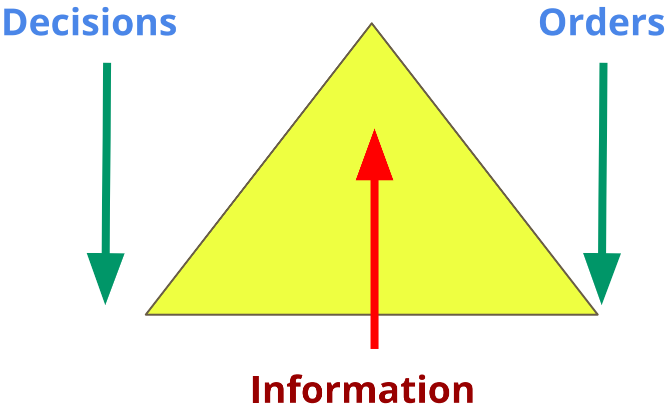
Hierarchical Networks

- Defines usually based on **authority**
- The network is divided into **discrete layers**
  - Each layer provides specific functions that define its role
- Relationships are close, **dyadic ties**
- Observed in traditional companies/organizations
  - **Information flow** is tricky and monopolized**!!!**
  - Potentially reduced efficiency

Hierarchical Networks

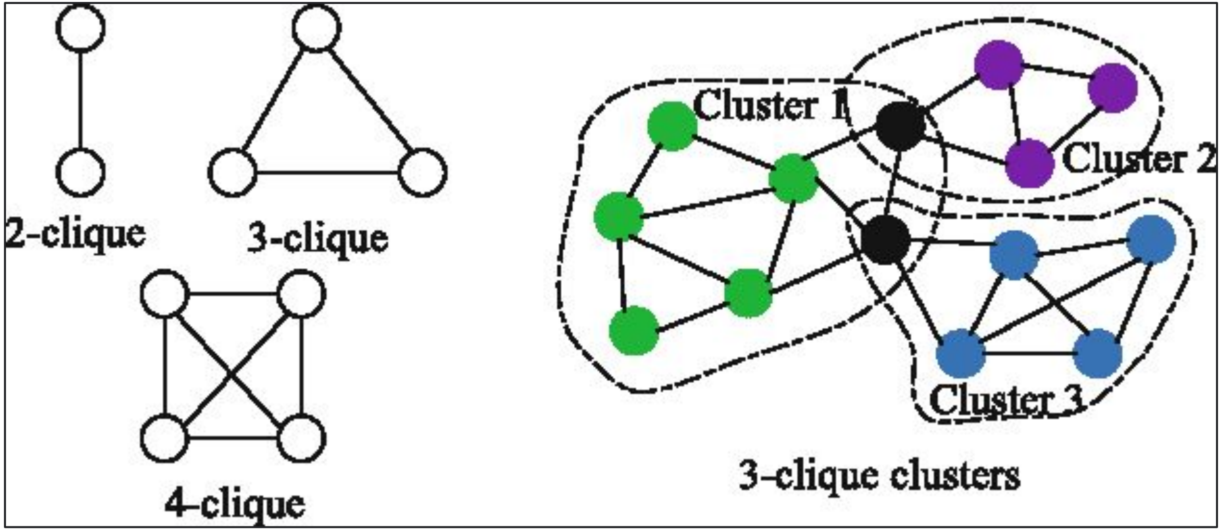


- Pyramid level organizational shape
- Each level has clear responsibilities
- Each node has a supervisor/manager
- Top-down chain of commands/decision



Cliques

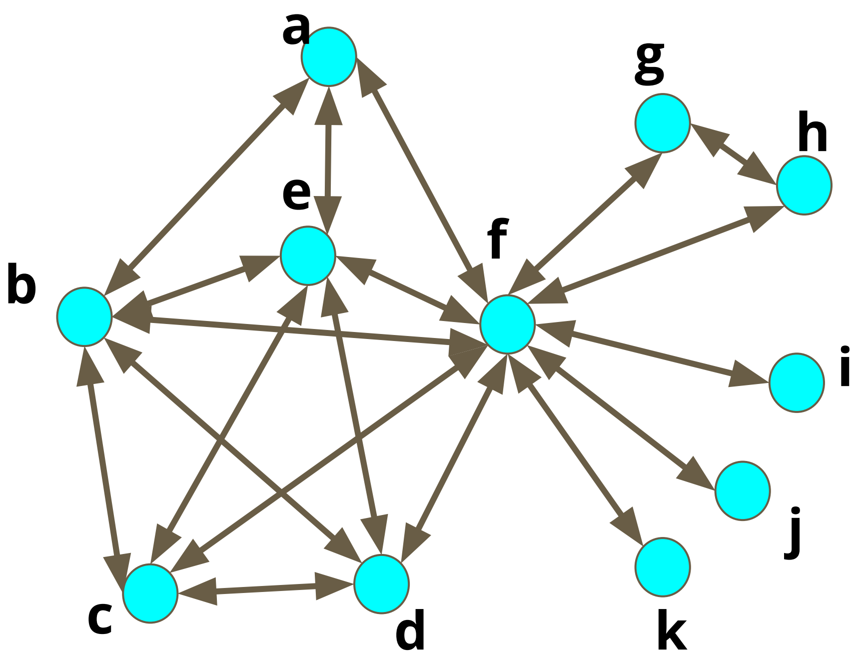
Cliques can make people feel left out!



- **Clique:** Max subset where all nodes are connected
- **K-Clique:** Clique with K members

- Cliques can overlap

How Many Cliques [Pen & Paper]



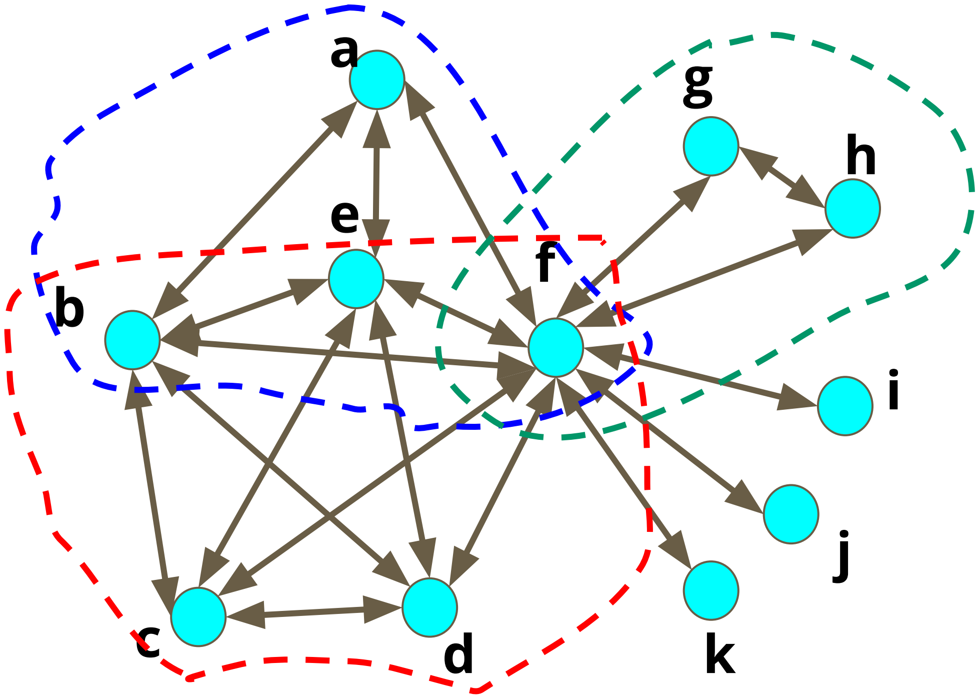
Write down the clique sizes and sets you spot!

N-Clique = {x, y, z}

…

…

How Many Cliques [Pen & Paper]



6 Cliques!

5-Clique: {b, e, f, c, d}

4-Clique: {a, b, e, f }

3-Clique: {g, h, f}

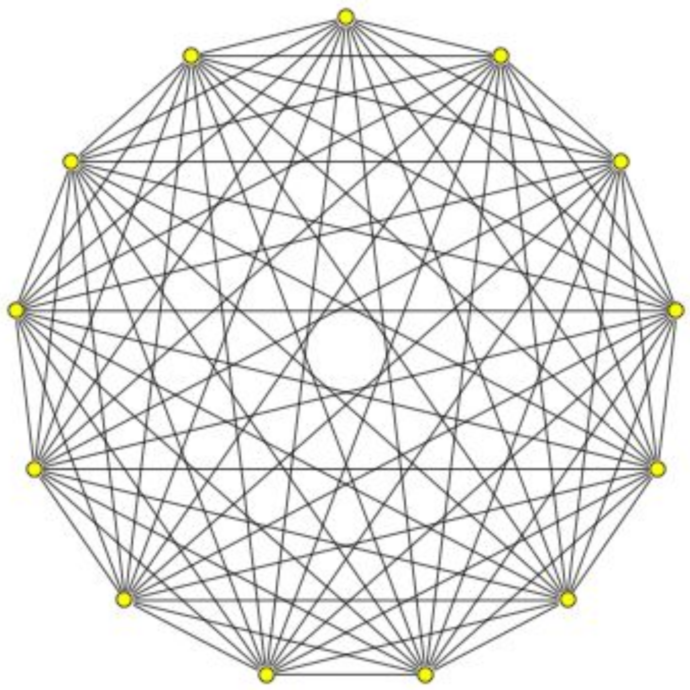
2-Clique: {f, i}

2-Clique: {f, j}

2-Clique: {f, k}

a-e-b would be a 3-clique, but it is counted within a-b-e-f 4-clique.
Here, it is counted within the maximal set you can count it in.

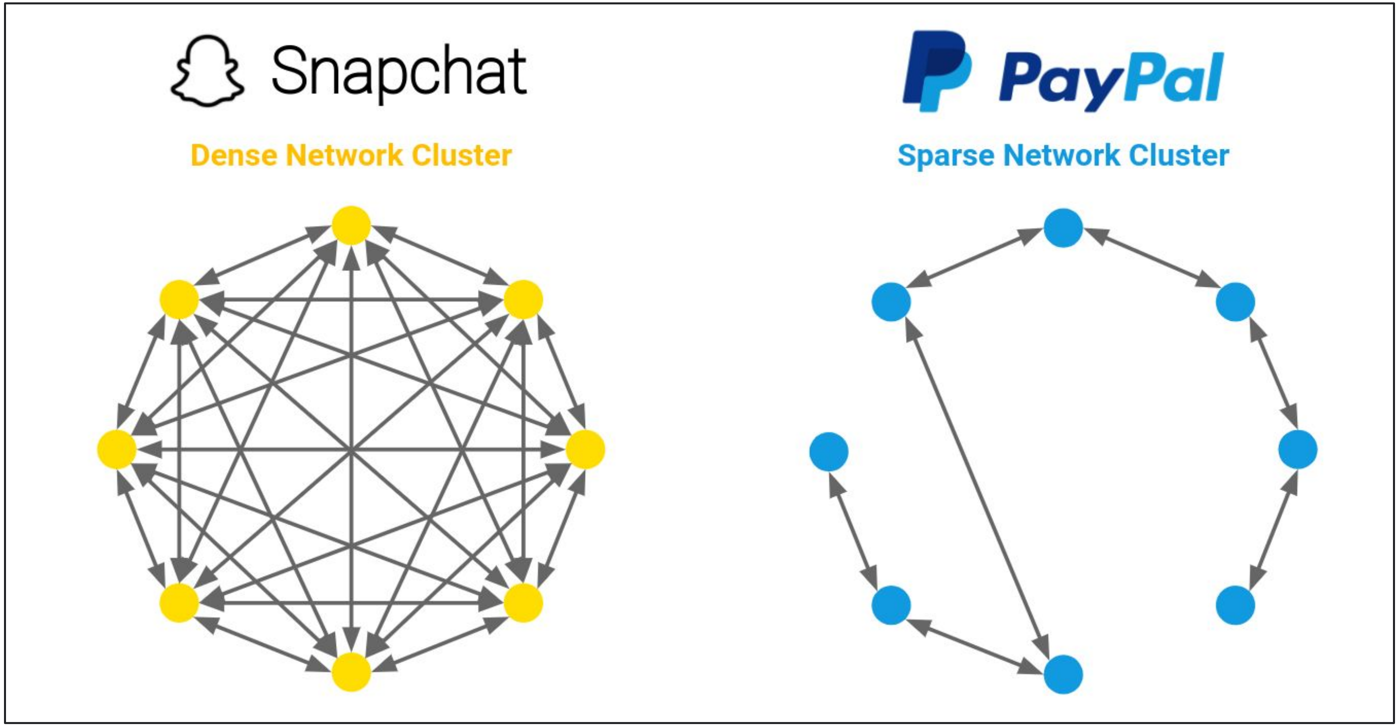
Complete Graphs (Not very common)



Complete Graph for N=13

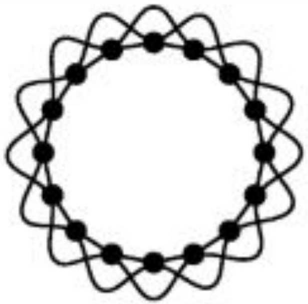
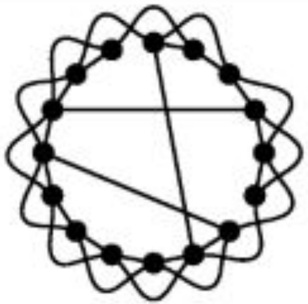
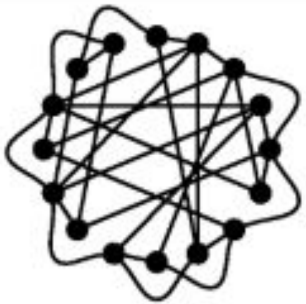
- All possible links are present (All nodes are adjacent to one another)
- All nodes' degrees are equal to N-1
- A clique is a complete graph
- Not common at all in real life networks at scale!
- Has maximum density

Dense versus Sparse Networks

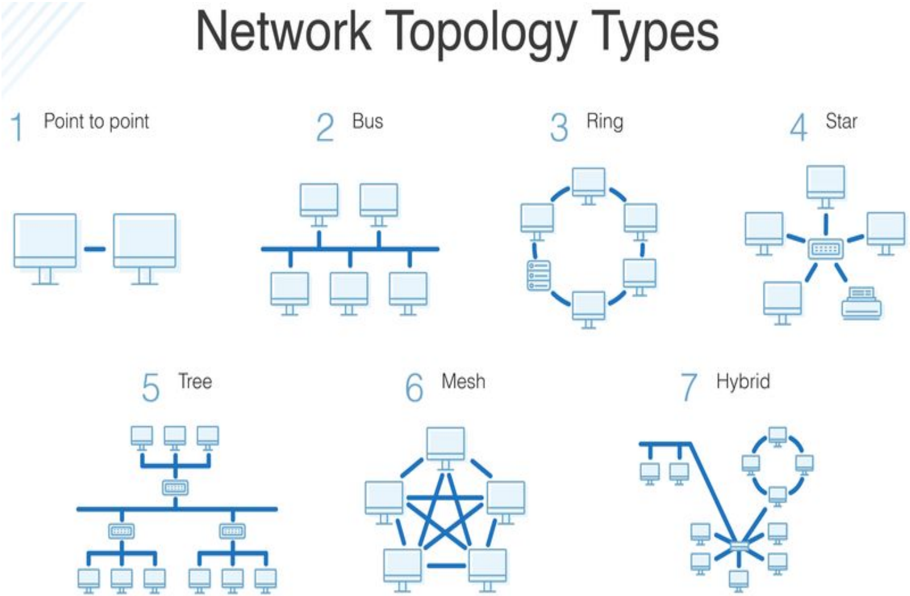


When density is lower, clusters and bridging gains importance!

Small World Networks

			
Network	Lattice, Ordered	Small World	Random, Disordered
Clustering Coefficient	High	High	Low
Mean Path Length	Long	Short	Short

Computer vs Social/Organizational Network Topologies



Impacts the communication patterns:

- Which node is centrally gathering information
- How much latency from point a to b
- Point of failures
- Which nodes are the bridges

Network vs Individual Level Analysis

Network (whole graph) level

– E.g. density

– Is it easier to disrupt a cellular or hierarchical structure?

– **Use:** Characterizing topology, comparing groups, high level change

Node level

– E.g., centralities

– Who has the power? Who has the ability to influence?

– **Use:** Identifying key actors, events, resources, influencers

These two types of analysis complete one another:

- Individual behaviors are not independent of the network they are in.
- Individual’s network position is not independent of the network structure.

Commonly Used Network Level Metrics

Metric	Value
Size	10
Link Count	21
Density	0.233
Isolate Count	0
Component Count	1.0
Reciprocity	0.1667
Characteristic Path Length	5.3699
Clustering Coefficient	0.325

Sample network reference values

Size

- Number of nodes in the network
- In real life networks;
  - Density decreases as size increases
  - Clustering increases
- Should always be included in network analyses as a covariate

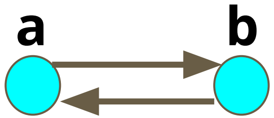
(Covariate: a variable that is possibly predictive of the outcome under study.)

Density

Number of links, expressed as a ratio of existing links over all possible links.

Density = #(Links) ÷ #(All possible links)

Directed Networks



$$D = \frac{L}{N * (N-1)}$$

Undirected Networks

$$D = \frac{L}{N * (N-1) \mathbf{/\ 2}}$$

Density

Metric	Value
Size	10
Link Count	21
Density	0.233
Isolate Count	0
Component Count	1.0
Reciprocity	0.1667
Characteristic Path Length	5.3699
Clustering Coefficient	0.325

Is the density here for a directed or undirected network?

Size & Density Correlation

- Size and density are negatively correlated.
- Network density decreases as the size increases.

Reciprocity (Mutuality, Symmetry)

- Relevant for directed networks
- Mutual ties: Both A → B and B → A exist

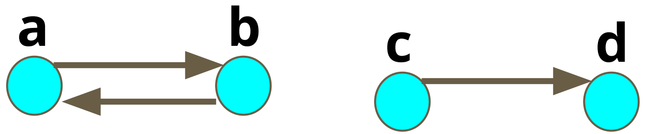
- Some relations are inherently symmetric or asymmetric
  - Who did you have lunch with? (Symmetric)
  - Who do you go to for advice (Asymmetric)

- Depending on how the question is phrased same relation can be represented symmetrically or asymmetrically
  - Who is whose immediate family? (Symmetric)
  - Who is whose father? (Asymmetric)

Reciprocity (Mutuality, Symmetry)

- Reciprocity = Ratio of ties that are reciprocated
  - Note: Not the ratio of mutual ties over all possible ties

$$\text{Reciprocity} = \frac{A_{ij} \text{ and } A_{ji}}{A_{ij} \text{ or } A_{ji}}$$



Characteristic Path Length

- Also called as average path length
- Geodesic distances between nodes are used in its computation
  - Shortest path between two nodes
  - Simply the number of edges on the path if unweighted
- Steps to compute characteristic path length:
  - **Step-1:** Average geodesic distance from node *i* to all other nodes
  - **Step-2:** Average of values computed in Step-1 for all nodes

Characteristic Path Length

- **Step-1:** The average distance from a specific node *i* to all other nodes

$$d_i = \frac{\sum_{j=1}^{n} d(i,j)}{(n-1)}$$

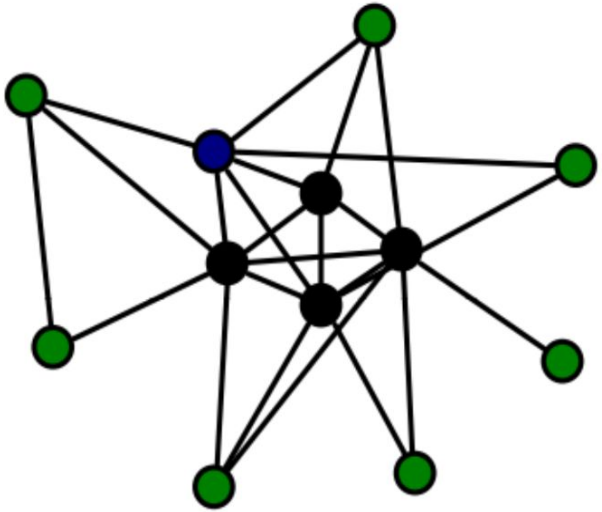
($d(i, j)$ is the geodesic distance from *i* to *j*)

- **Step-2:** The average of distances from Step-1 over all nodes

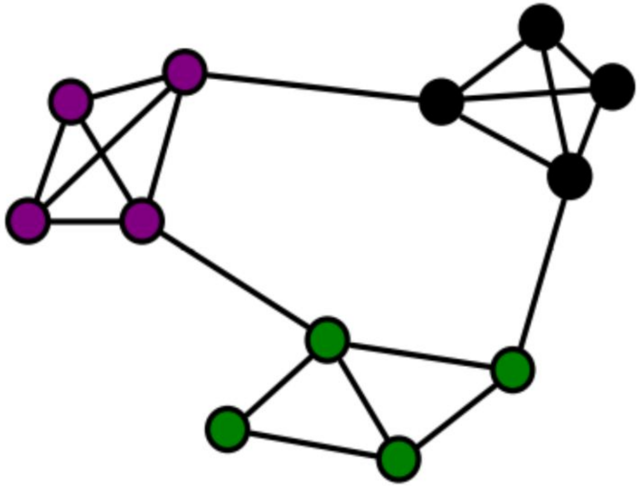
$$d = \frac{\sum_{i=1}^{n} d_i}{n}$$

Diameter

- Maximum geodesic distance between any pair of nodes



Diameter = 3



Diameter = 4

Steve Borgatti - 2004

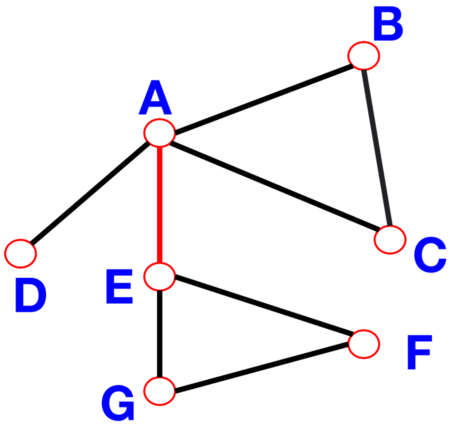
Connectivity and Components

- Handled differently for directed and undirected networks

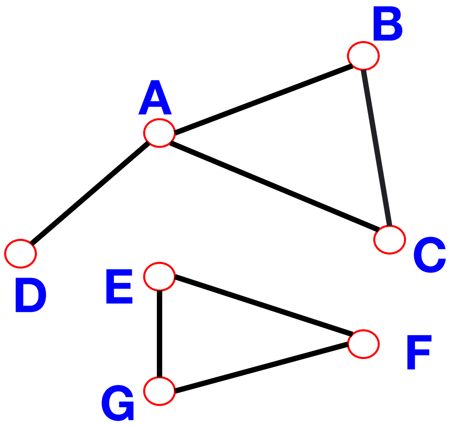
- Breadth-first traversal is commonly used

Connectivity and Components (Undirected Graphs)

- **Connected (undirected) graph:** any two vertices can be joined by a path.
- **Disconnected graph:** made up by two or more connected components.



Bridge: if we erase it, the graph becomes disconnected.

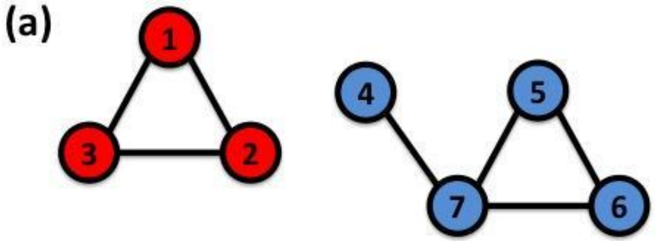


Largest Component:
Giant Component

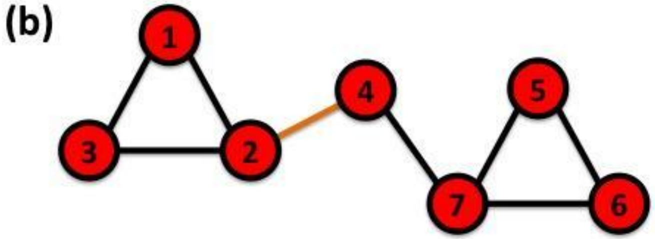
The rest: **Isolates**

Connectivity and Components (Undirected Graphs)

- The adjacency matrix of a network with several components can be written in a block-diagonal form
- Non-zero elements are confined to squares, with all other elements being zero:



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

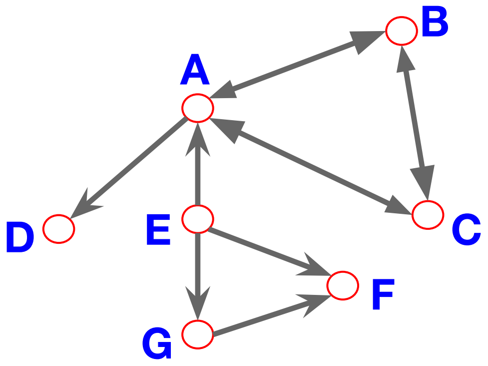


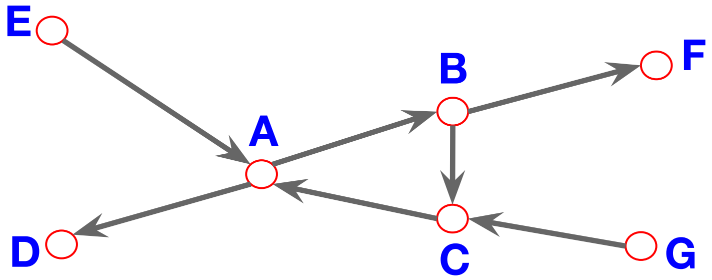
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Connectivity and Components (Directed Graphs)

- **Strongly connected directed graph:** has a path from each node to every other node and vice versa (e.g. AB path and BA path).

- **Weakly connected directed graph:** it is connected if we disregard the edge directions.





- **In-component**: nodes that can reach the scc,
- **Out-component**: nodes that can be reached from the scc.

Finding Connected Components of a Network

1. Start from a random node i and perform BFS.
2. Label all nodes with $n = 1$
3. If the number of nodes labeled is equal to N (e.g. number of nodes), then the graph is connected.
4. If the number of nodes labeled is less than N, then there are multiple components. Proceed to find them.
5. Increase the label $n \rightarrow n + 1$
6. Choose an unmarked node j, and label it with n.
7. Perform BFS starting with j and mark all reachable nodes with n.
8. Return to Step-3.

Clustering Coefficient

- A measure of degree to which nodes in a graph tend to cluster together
  - Find clustering coefficient of each node and then take their average to find the clustering coefficient of the network
- Clustering coefficient of a node
  - What fraction of your neighbors are connected

Clustering Coefficient

- **Clustering coefficient of a Node *i*:** It is a local measurement
- **Reasoning:** Nodes create tight knit groups and have higher density of ties in real networks (e.g. higher probability than randomly created ties)

C_i = Clustering coefficient of node *i*

$$C_i = \frac{2L_i}{k_i(k_i - 1)}$$

L_i = Number of edges between neighbors of node *i*

k_i = Degree of node *i*

- **Clustering coefficient of a Graph *G*:** Average of Clustering coefficient of all nodes

$$C_G = \frac{\sum_{i=1}^{N} C_i}{N}$$

Clustering Coefficient

Consider graph shown in Figure 1. Now we calculate clustering coefficient for each node.

For Node 1: $K_1 = 2$, $L_1 = 1$, $C_1 = \frac{2(1)}{2(2-1)} \Rightarrow C_1 = \frac{2}{2} \Rightarrow C_1 = 1$

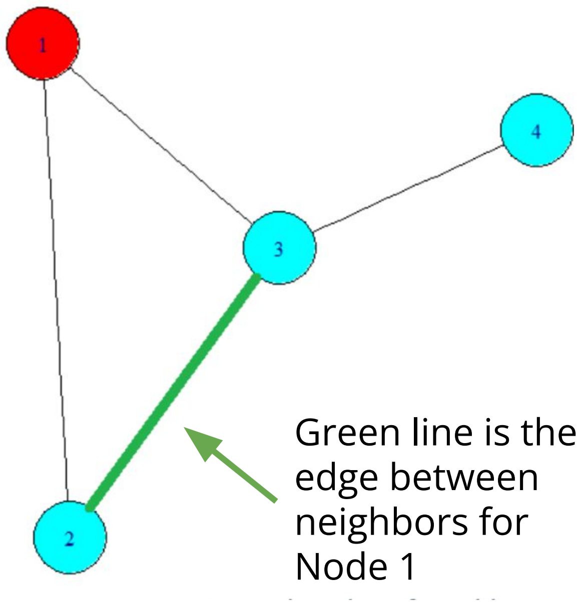
For Node 2: $K_2 = 2$, $L_2 = 1$, $C_2 = \frac{2(1)}{2(2-1)} \Rightarrow C_2 = \frac{2}{2} \Rightarrow C_2 = 1$

For Node 3: $K_3 = 3$, $L_2 = 1$, $C_3 = \frac{2(1)}{3(3-1)} \Rightarrow C_3 = \frac{2}{6} \Rightarrow C_3 = 0.33$

For Node 4: $K_4 = 1$, $L_4 = 0$, $C_4 = \frac{2(0)}{1(1-1)} \Rightarrow C_4 = \frac{0}{0} \Rightarrow C_4 = 0$

For average clustering coefficient

$$\langle C \rangle = \frac{1}{N}\sum_{i=1}^{N} C_i \Rightarrow \frac{1}{4}(1 + 1 + 0.33 + 0) \Rightarrow 0.58$$



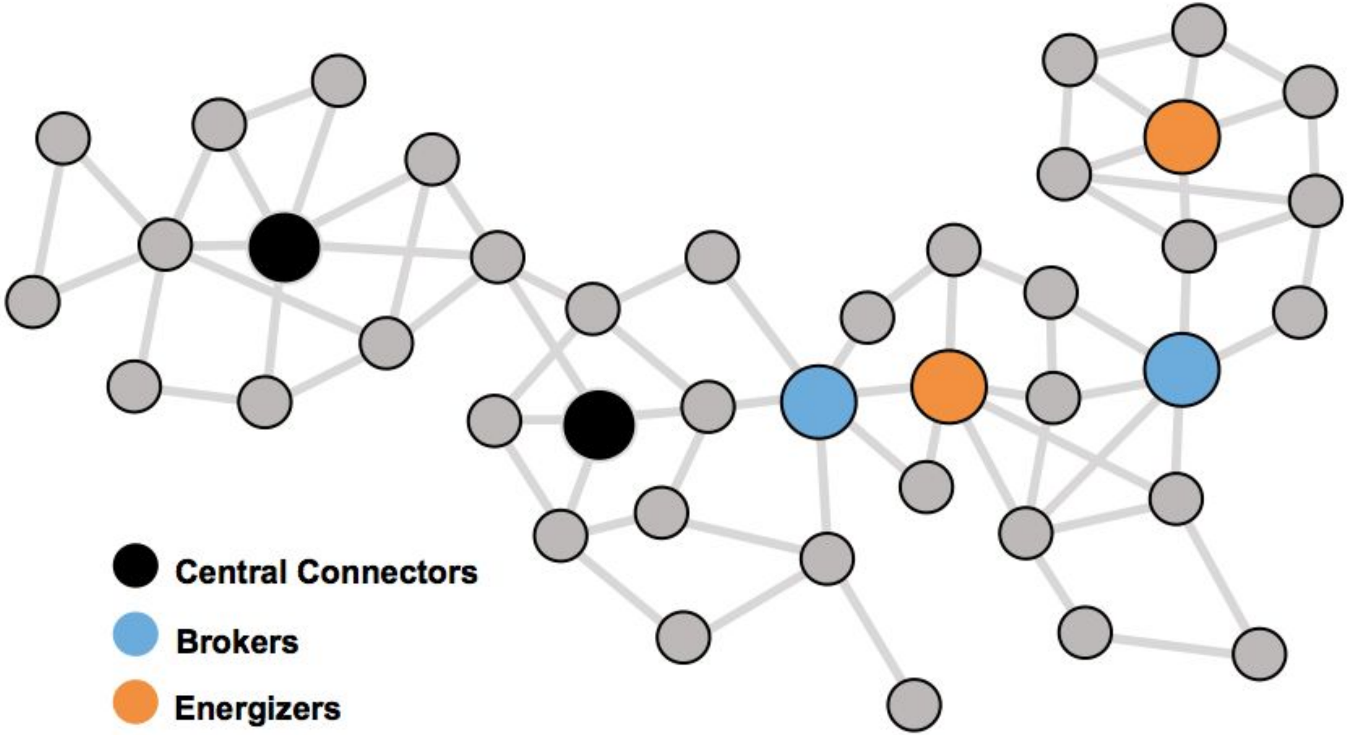
Centrality and Prestige

Centrality Measures - Node Level Analysis

- Reveals who are key actors
  - Helps identify network elites, brokers, energizers, influencers
  - Reveals structural imbalances

- Node level analysis but very much dependent on the overall network topology
  - Centrality measures for a network are highly correlated

Social Power and Centrality



Most Commonly Used Centrality Measures

Centrality Measure	Definition	Meaning	Usage
Degree Centrality	Nodes with most connections	In the know, direct access to more information	• Identifying resources for intel • Identifying who to remove to reduce information flow
Closeness Centrality	Nodes that are closest to other nodes	Rapid access to information	• Identifying who is good for rapid information acquisition/dissemination
Betweenness Centrality	Nodes that are on the best paths	Connects otherwise disconnected groups	• Identifying who can be brokers, go-betweens, or gate-keepers. • Identifying who can reduce activity by disconnecting groups
Eigenvector Centrality	Nodes most connected to other highly connected nodes.	Strong social capital	• Identifying who can mobilize others

Degree Centrality

- Number of edges in and out of a node
- Sum (degrees of all nodes) = 2 * #(edges in graph)
- Commonly thought of as a measure of influence or importance
  - Nodes with high degree centrality have the opportunity to influence & be influenced directly

For Node A

- **Total Degree Centrality:** Total number of incoming + outgoing edges for Node A
- **In Degree Centrality:** Number of edges incoming towards Node A
  - Column A in adjacency matrix)
  - **Sink:** 0 in degree
- **Out Degree Centrality:** Number of edges outgoing from Node A
  - Row A in adjacency matrix
  - **Source:** 0 out degree

Closeness Centrality

- Not necessarily the node with the highest number of friends, but...
  - In the middle of the network by being close to many friends
  - In gossip networks: central player who hears things first

- **Usage:** Efficiency of a node in reaching out to everyone quickly
  - Spreading news or virus

$$C_i = \frac{1}{\sum_{j=1}^{N} d(i,j)}$$

Distance between i and j

- **Drawback:** Computed only for/within the component

Betweenness Centrality

- How often a node lies along the shortest path between two other nodes
- **Usage:** Potential for gate-keeping, brokering, controlling the flow, and bridging/connecting otherwise separate parts of the network

Betweenness of node v

$$b(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Number of shortest paths between s and t that pass through v

Number of shortest paths between s and t

- **Drawback:** Very “expensive” to compute

Eigenvector Centrality

- A node has high eigenvector centrality if it is connected to many nodes are themselves well connected
- "A node is important if it is connected to important nodes."
  - Does not necessarily mean many connections, few but important connections can boost the eigenvector centrality
- **Usage:** Measures popularity and influence. Tends to identify centers of large cliques (e.g. leader of a self-contained group)
- **Recursive Definition:**

$$E_i \propto \sum_j E_j$$

Eigenvector Centrality

$$x_v = \frac{1}{\lambda} \sum_{t \in M(v)} x_t = \frac{1}{\lambda} \sum_{t \in G} a_{v,t} x_t$$

- M(v) is neighbors of v, lambda is a constant
- Rearrange this equation, you arrive at Eigenvector definition

$$Ax = \lambda x$$

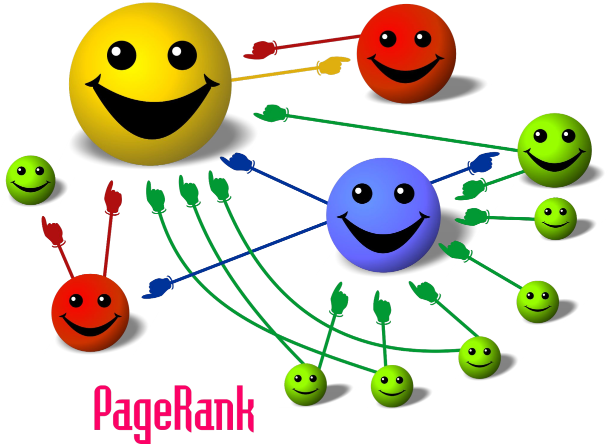
- There will be many different eigenvalues for which a non-zero eigenvector solution exists.
- Non negative values for the centrality is desired, take the highest Eigenvalue.

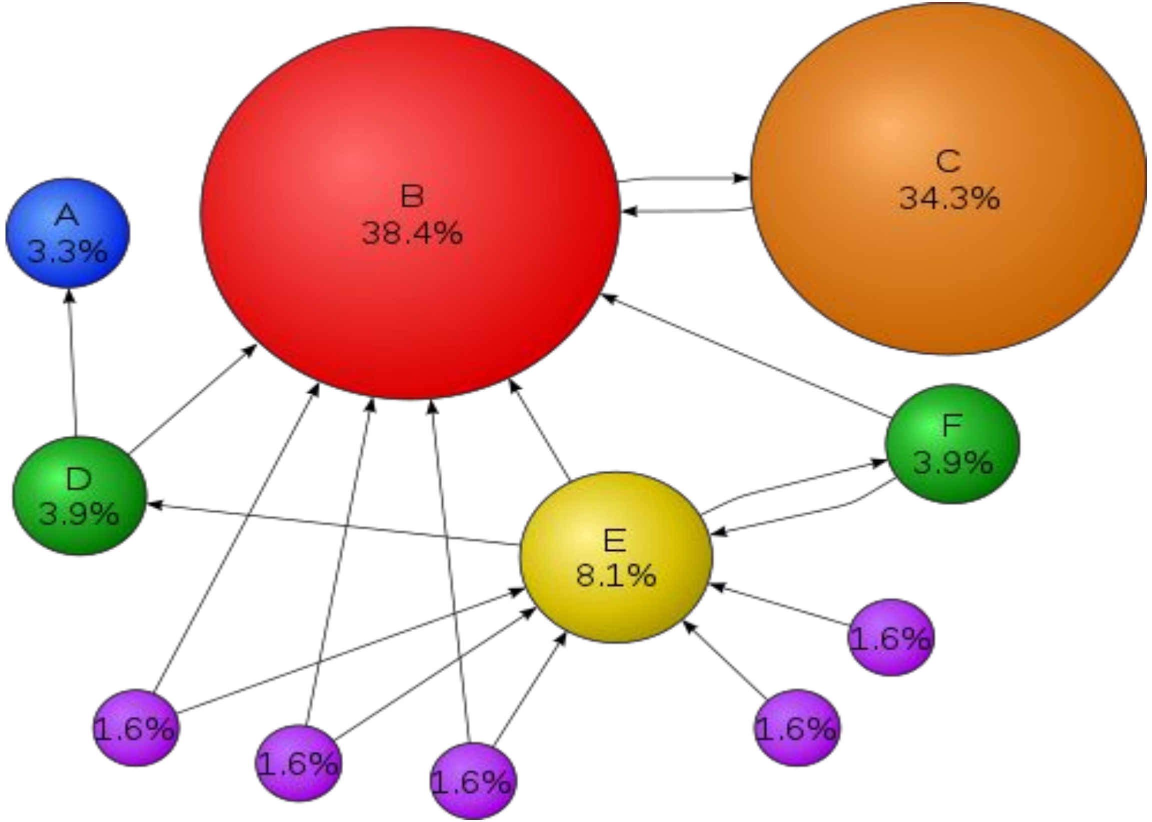
Page Rank

- Calculates importance of a node based on the importance of its incoming links
- Google’s pagerank is based on the normalized Eigenvector centrality combined with random jumps.
- Random jumps could be things like:
  - Entering an address in URL bar
  - Visiting a "Favorite" link
  - Visiting a home page (or any one of the links on it)
  - Visiting a link from a content aggregator / social media
- Hence, open to manipulation. Spamdexing and giving no votes to such links and follows is important.

(Original paper: http://infolab.stanford.edu/pub/papers/google.pdf)

Page Rank





Fraction of times a node would be visited according to network of probabilities

Image source: https://en.wikipedia.org/wiki/PageRank

Algorithms - Common Centrality Metrics in Depth

Closeness Centrality Algorithm

```
def closeness_centrality(G, u=None, distance=None, normalized=True):
	if distance is not None:
		# use Dijkstra's algorithm with specified attribute as edge weight
		path_length = functools.partial (nx.single_source_dijkstra_path_length, weight=distance)
	else:
		path_length = nx.single_source_shortest_path_length

	if u is None:
		nodes = G.nodes()
	else:
		nodes = [u]
	closeness_centrality = {}
	for n in nodes:
		sp = path_length(G,n)
		totsp = sum(sp.values())
		if totsp > 0.0 and len(G) > 1:
			closeness_centrality[n] = (len(sp)-1.0) / totsp
			# normalize to number of nodes-1 in connected part
			if normalized:
				s = (len(sp)-1.0) / ( len(G) - 1 )
				closeness_centrality[n] *= s
		else:
			closeness_centrality[n] = 0.0
	if u is not None:
		return closeness_centrality[u]
	else:
		return closeness_centrality
```

Parameters

G : graph (A NetworkX graph)

u : node, optional
Return only the value for node u

distance : edge attribute key, optional (default=None)
Use the specified edge attribute as the edge distance in shortest path calculations

normalized : bool, optional
If True (default) normalize by the number of nodes in the connected part of the graph.

https://www.geeksforgeeks.org/closeness-centrality-centrality-measure/

Betweenness Centrality Algorithm

```
def betweenness_centrality(G, k=None, normalized=True, weight=None,
                                              endpoints=False, seed=None):
    betweenness = dict.fromkeys(G, 0.0) # b[v]=0 for v in G
    if k is None:
        nodes = G
    else:
        random.seed(seed)
        nodes = random.sample(G.nodes(), k)
    for s in nodes:

        # single source shortest paths
        if weight is None: # use BFS
            S, P, sigma = _single_source_shortest_path_basic(G, s)
        else: # use Dijkstra's algorithm
            S, P, sigma = _single_source_dijkstra_path_basic(G, s, weight)

        # accumulation
        if endpoints:
            betweenness = _accumulate_endpoints(betweenness, S, P, sigma, s)
        else:
            betweenness = _accumulate_basic(betweenness, S, P, sigma, s)

    # rescaling
    betweenness = _rescale(betweenness, len(G), normalized=normalized,
                                     directed=G.is_directed(), k=k)
    return betweenness
```

Parameters

G : graph (A NetworkX graph.)

k : int, optional (default=None)
If k is not None use k node samples to estimate betweenness. The value of k <= n where n is the number of nodes in the graph. Higher values give better approximation.

normalized : bool, optional
If True the betweenness values are normalized by `2/((n-1)(n-2))`for graphs, and `1/((n-1)(n-2))` for directed graphs where `n`is the number of nodes in G.

weight : None or string, optional (default=None) If None, all edge weights are considered equal.Otherwise holds the name of the edge attribute used as weight.

endpoints : bool, optional. If True include the endpoints in the shortest path counts.

https://www.geeksforgeeks.org/betweenness-centrality-centrality-measure/

Eigenvector Centrality Algorithm

```
def eigenvector_centrality(G, max_iter=100, tol=1.0e-6, nstart=None, weight='weight'):
    from math import sqrt
    if nstart is None:
        # choose starting vector with entries of 1/len(G)
        x = dict([(n,1.0/len(G)) for n in G])
    else:
        x = nstart
    # normalize starting vector
    s = 1.0/sum(x.values())
    for k in x:
        x[k] *= s
    nnodes = G.number_of_nodes()

    for i in range(max_iter):   # make up to max_iter iterations
        xlast = x
        x = dict.fromkeys(xlast, 0)

        # do the multiplication y^T = x^T A
        for n in x:
            for nbr in G[n]:
                x[nbr] += xlast[n] * G[n][nbr].get(weight, 1)

        # normalize vector
        try:
            s = 1.0/sqrt(sum(v**2 for v in x.values()))

        # this should never be zero?
        except ZeroDivisionError:
            s = 1.0
        for n in x:
            x[n] *= s

        # check convergence
        err = sum([abs(x[n] - xlast[n]) for n in x])
        if err < nnodes*tol:
            return x

    raise nx.NetworkXError("""eigenvector_centrality():
power iteration failed to converge in %d iterations."%(i+1))""")
```

Parameters

G : graph (A NetworkX graph.)

max_iter : integer, optional.Maximum number of iterations in power method.

tol : float, optional. Error tolerance used to check convergence in power method iteration.

nstart : dictionary, optional. Starting value of eigenvector iteration for each node.

weight : None or string, optional. If None, all edge weights are considered equal. Otherwise holds the name of the edge attribute used as weight.

https://www.geeksforgeeks.org/eigenvector-centrality-centrality-measure/

Next Week:

Generated Network Topologies