Module 4: Social Network Analysis

Upon completion of this module, you should be able to:

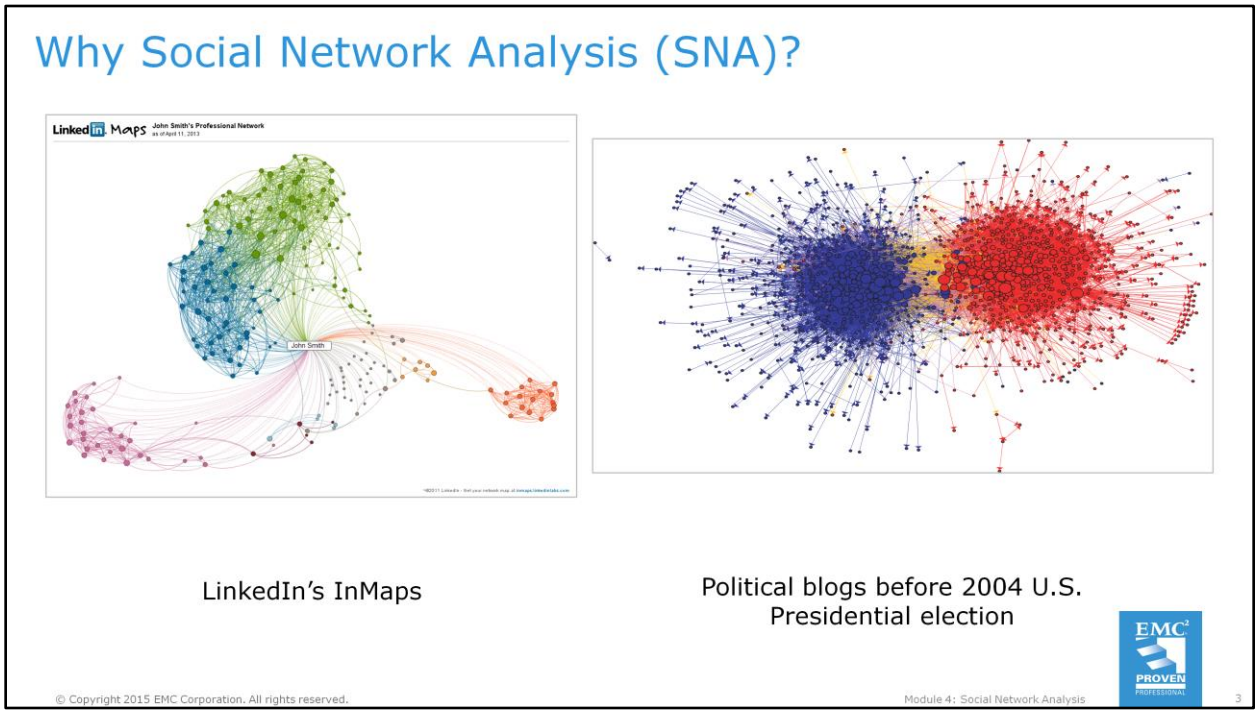
- Conduct Social Network Analysis using graph theory
- Describe characteristics of social network communities
- Identify tools available for Social Network Analysis

This module focuses on Social Network Analysis.

Lesson 1: Introduction to SNA and Graph Theory

This lesson covers the following topics:

- Basic notions of Social Network Analysis
- Graph Elements and Graph Theory
- Connectivity
- Giant Component
- Percolation Threshold



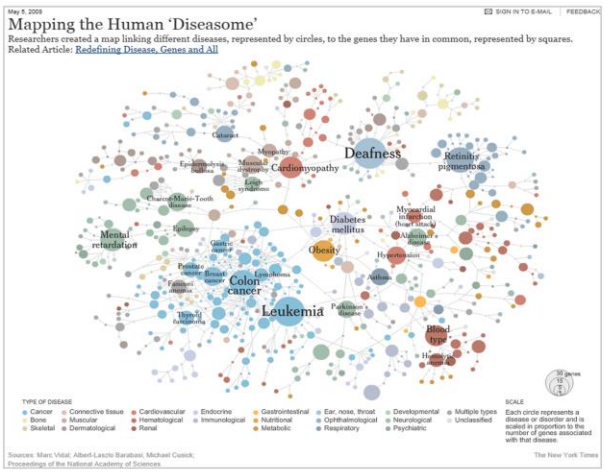
Here we show a few use cases of Social Network Analysis (SNA). The left image shows a user's LinkedIn social network, generated by LinkedIn's InMaps visualization tool. The graph displays this user as a node in the center of the graph. The user's contacts have been grouped into several categories and highlighted by different colors.

Links among Web pages can also reveal densely-knit communities and prominent sites. The right image shows the network structure of political blogs prior to the 2004 U.S. Presidential election that reveals two natural and well-separated clusters.

Reference:

Lada Adamic and Natalie Glance (205). The political blogosphere and the 2004 U.S. election: Divided they blog. *In Proceedings of the 3rd International Workshop on Link Discovery*, pp. 36-43.

Why Social Network Analysis (SNA)? (cont.)



NY Times Diseasome Map

Tracking and predicting future behaviors

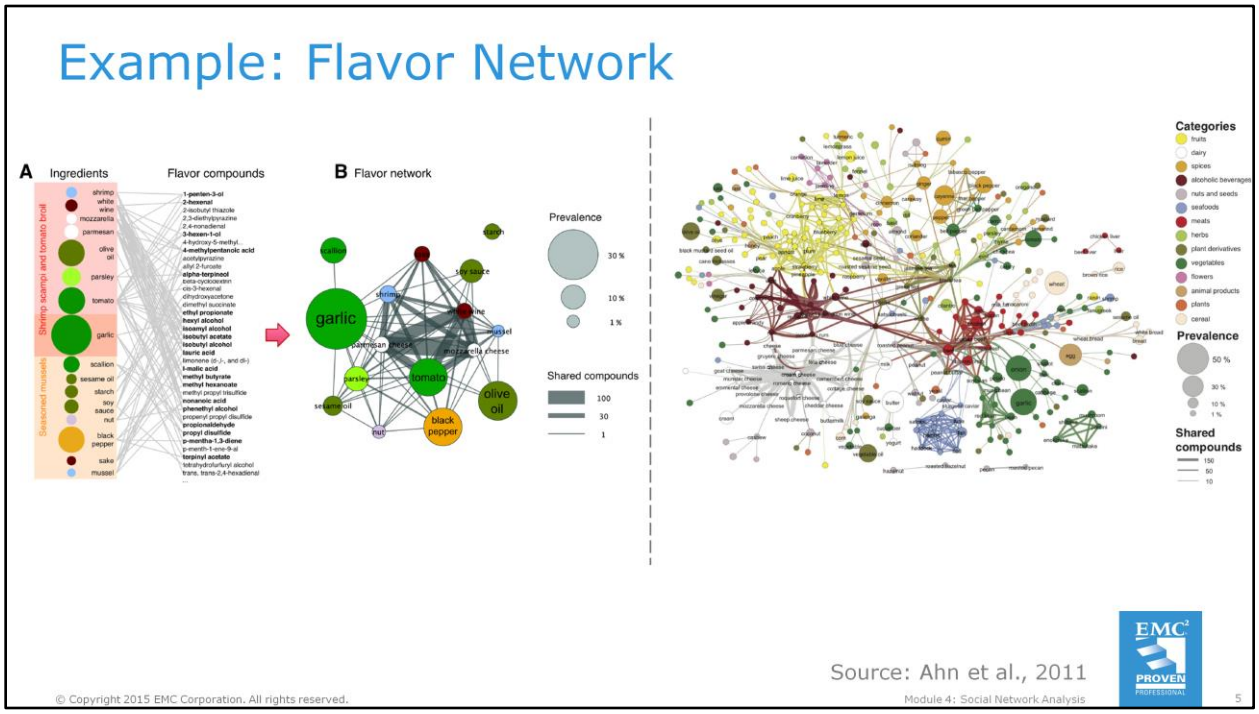
Here we show a few more use cases of Social Network Analysis (SNA). The left image shows a New York Times "diseasome" map linking different diseases, represented by circles, to the genes they have in common, represented by squares. The types of disease are distinguished by colors.

Raytheon, a multinational security firm, has developed software capable of tracking people's activities and predicting future behavior by mining data from social networking websites, as shown in the right image.

References:

New York Times Diseasome map: http://www.nytimes.com/interactive/2008/05/05/science/20080506_DISEASE.html

Software that tracks people on social media created by defense firm:
http://www.guardian.co.uk/world/2013/feb/10/software-tracks-social-media-defence



Here's a non-traditional use of social network analysis. In Ahn et al.'s work, the authors collected 56,498 recipes provided by two American repositories (epicurious.com and allrecipes.com) and a Korean repository (menupan.com). The recipes are grouped into geographically distinct cuisines (North American, Western European, Southern European, Latin American, and East Asian.) Their work found that western cuisines tend to use ingredient pairs that share many flavor compounds, while East Asian cuisines tend to avoid compound sharing ingredients.

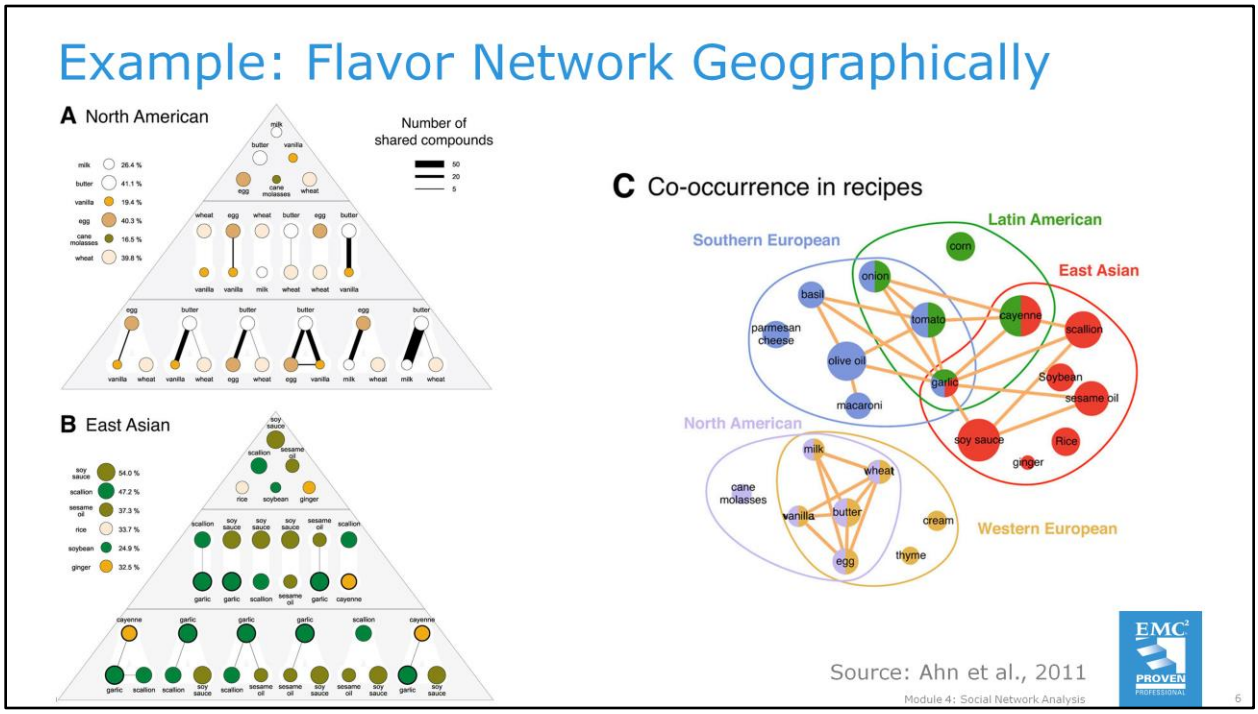
The left image shows (A) a bipartite network of ingredients from two recipes and the flavor compounds of such ingredients, and (B) the corresponding flavor network of the two recipes.

The right image shows the backbone of the flavor network.

References:

YY Ahn, SE Ahnert, JP Bagrow, AL Barabási (2011). Flavor network and the principles of food pairing. *Nature Scientific Reports*. http://www.nature.com/srep/2011/111215/srep00196/full/srep00196.html

A talk by YY Ahn: http://www.youtube.com/watch?v=tzOpookT8gU



Shown in figures (A) and (B) are the six most authentic single ingredients, ingredient pairs and triplets for North American and East Asian cuisines in a flavor pyramid. Colors indicate the different ingredient classes. From the two pyramids, we can see that the differences between the two cuisines are quite substantial. North American food heavily relies on dairy products, eggs and wheat. On the other hand, East Asian cuisine is dominated by plant derivatives like soy sauce, sesame oil, and rice and ginger. The two pyramids also illustrate the different affinities of the two regional cuisines towards food pairs with shared compounds. The most authentic ingredient pairs and triplets in the North American cuisine share multiple flavor compounds, indicated by black links, but such compound-sharing links are rare among the most authentic combinations in East Asian cuisine.

Figure (C) shows the six most authentic ingredients and ingredient pairs in different regional cuisines. A close relationship can be observed between North American and Western European cuisines. Garlic is a common ingredient shared among Southern European, Latin American, and East Asian cuisines, but rare in North American or Western European cuisines. When it comes to its signature ingredient combinations, Southern European cuisine is much closer to Latin American than Western European cuisine.

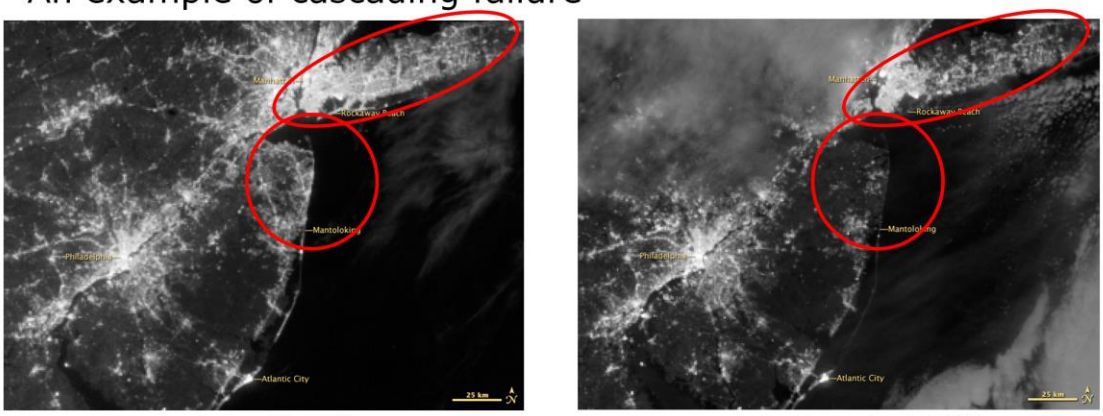
References:

YY Ahn, SE Ahnert, JP Bagrow, AL Barabási (2011). Flavor network and the principles of food pairing. *Nature Scientific Reports*. http://www.nature.com/srep/2011/111215/srep00196/full/srep00196.html

A talk by YY Ahn: http://www.youtube.com/watch?v=tzOpookT8gU

Why SNA? Cascading Failure in Power Grid Network

- Blackout in New Jersey and New York after hurricane Sandy
- An example of cascading failure



August 31, 2012 November 1, 2012

Source: http://earthobservatory.nasa.gov/IOTD/view.php?id=79589

Here are two satellite images from NASA showing the blackout in New Jersey and New York after hurricane Sandy struck the U.S. east coast in late October 2012, leaving a total damage of over $71 billion USD. The satellite images show that the lower third of Manhattan is dark on November 1, while Rockaway Beach, much of Long Island, and nearly all of central New Jersey are significantly dimmer. The barrier islands along the New Jersey coast, which are heavily developed with tourist businesses and year-round residents, are just barely visible in moonlight after the blackout.

This is a typical example of a cascading failure. When a network acts as a transportation system, a local failure shifts loads or responsibilities to other nodes. If the extra load is negligible, the rest of the system can seamlessly absorb it, and the failure remains effectively unnoticed. If the extra load is too much for the neighbor nodes to carry, the nodes will either tip or again redistribute the load to their neighbors. Either way, the network faces a cascading event. In the case of hurricane Sandy, the electricity could not be restored when a line went down, and the power had to be shifted to other lines. If the neighbor lines were not capable of carrying the extra load, they also tipped, or redistributed the increased load to their neighbors, which in turn tip those lines as well.

Cascading failures are common in most complex networks. For example, they often take place on the Internet, when traffic is rerouted to bypass malfunctioning routers, occasionally creating denial of service attacks on routers that do not have the capacity to handle extra traffic.

The images are available at a higher resolution on NASA's website:
http://earthobservatory.nasa.gov/IOTD/view.php?id=79589

Erdős Number

- Paul Erdős (1913—1996)
- 1500 papers, 500 collaborators
- Erdős number of all scholars
  - Average < 5
  - Most < 8

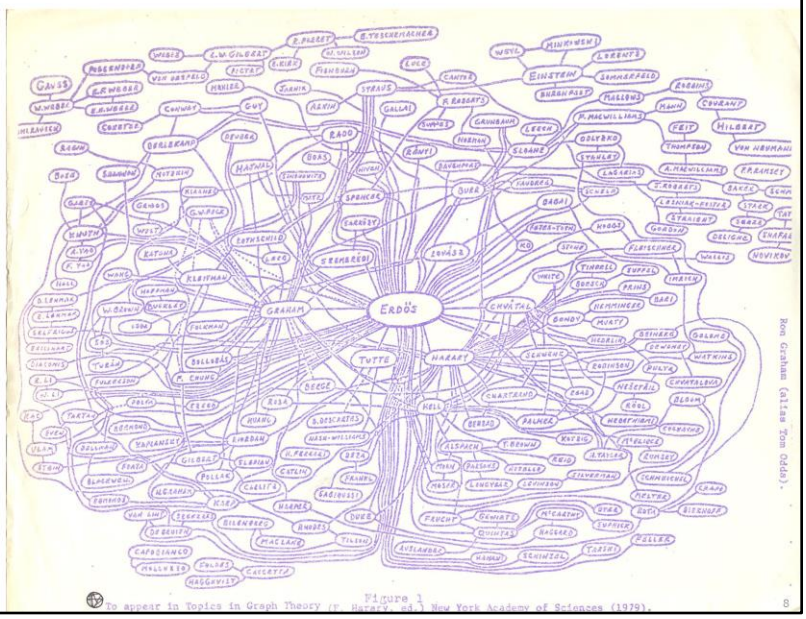


Figure 1
To appear in Topics in Graph Theory (F. Harary, ed.) New York Academy of Sciences (1979).

The **Erdős number** is named after the Hungarian mathematician Paul Erdős, who is known for his collaborative research with more than 500 researchers from various fields of study such as mathematics, physics, chemistry, medicine, economics and computer science.

The Erdős number describes the collaborative distance between a researcher and mathematician Paul Erdős, as measured by authorship of mathematical papers. Erdős wrote around 1,500 mathematical articles in his lifetime, mostly co-written. He had 511 direct collaborators and these are the people with Erdős number 1. The people who have collaborated with them (excluding Erdős himself) have an Erdős number of 2, and so on. Since the death of Paul Erdős in 1996, the lowest Erdős number that a researcher can obtain is 2.

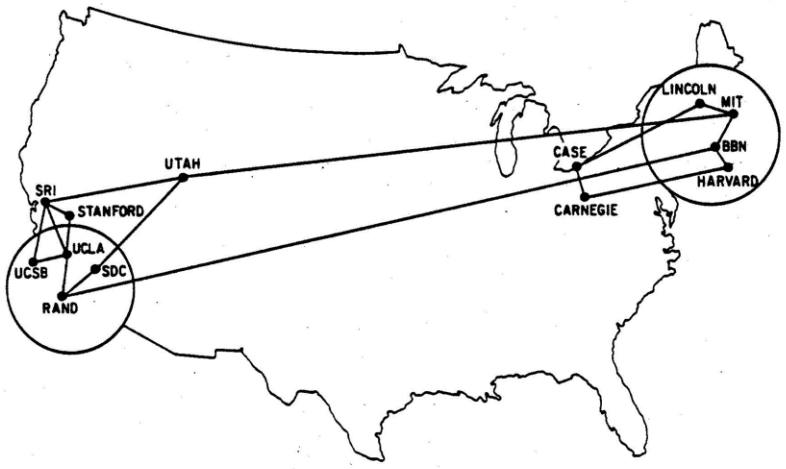
A smaller Erdős number implies a closer relationship with Erdős. Therefore, the Erdős number has become a criterion for scholars to find out the degree of collaboration. In fact, many Nobel laureates have Erdős numbers less than 5. Most scholars who have ever published papers have an Erdős number less than 8.

A number of variations on the concept have been proposed to apply to other fields. The best known is the **Bacon number**. The film actor Kevin Bacon has been in 56 movies so far. Anybody who has acted in a film with Bacon has a Bacon number of 1. Anybody who does not have a Bacon number 1 but has worked with somebody who does, has a Bacon number 2, and so on. It turns out most people in movies have a number 4 or less. Given that there are about 225,000 such people, this is remarkable.

Both the Erdős number and the Bacon number are examples of what's called the **small world phenomenon**, which will be discussed later in this module.

Graphs as Models of Networks

- ARPANET, December 1970

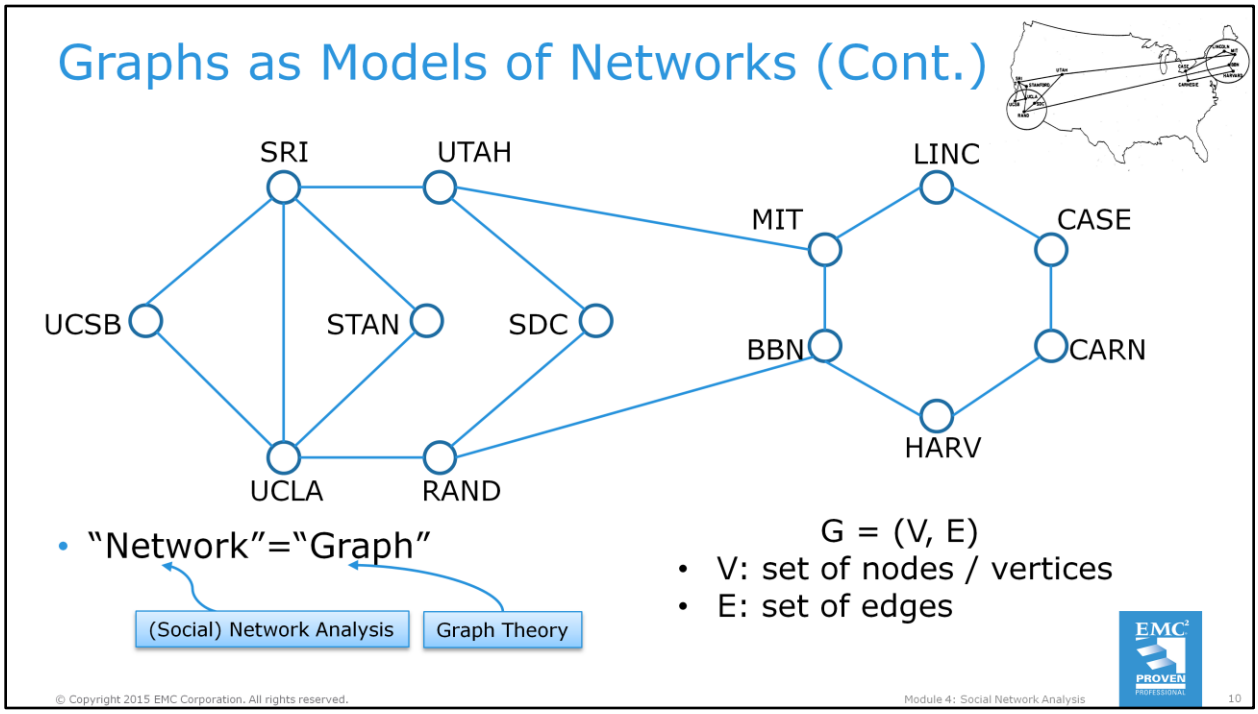


The slide shows a network in December 1970 depicting sites on the Internet, then known as the ARPANET. This network has 13 sites (or nodes), with 7 on the west coast and 6 on the east coast.

The image is from the following report:

Heart, F., McKenzie, A., McQuillian, J., and Walden, D. (January 4, 1978). *ARPANET Completion Report.* Bolt, Beranek and Newman, Burlington, MA.

More information can be found at http://som.csudh.edu/cis/lpress/history/arpamaps/.



Here is an alternate drawing of the ARPANET in December 1970.

Note that the term network from social network analysis is essentially the graph from graph theory. In graph theory, a graph is a way of specifying relationships among a collection of items. A graph consists of a set of circles, called nodes, with certain pairs of circles connected by links called edges. Formally, a graph can be defined as G=(V, E) where V represents the nodes (or vertices) and E represents the edges of the graph.

Graph Elements: Edges

- Directed: A → B (also called arcs or links)
  - A and B are two webpages and page A contains a hyperlink that points to B
  - A is B's manager
  - B knows A but A doesn't know B
- Undirected: A ↔ B, A – B, or [A, B]
  - A and B are two words with significant co-occurrence in texts
  - A and B are co-authors
  - A and B know each other
- Possible edge attributes
  - Weight
  - Ranking
  - Type
  - Properties depending on the structure of the rest of the graph

Edges are a fundamental unit of a graph. In the diagram of a graph, an edge is represented by a line or an arrow that connects two nodes. In the text form, an edge connecting nodes A and B can be denoted as A → B (or B → A depending on the direction of the arrow) for a directed graph and A ↔ B (or A – B, [A, B]) for an undirected graph.

In a directed graph, edge A → B could represent relationships such as webpage A provides hyperlinks that points to B, A is B's manager, A's post is retweeted by B, A follows B on Twitter, or B knows A but A doesn't know B.

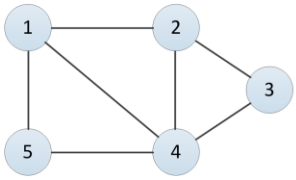
In an undirected graph, edge A ↔ B could represent relationships such as word A and word B have significant co-occurrence in texts, A and B are co-authors of a paper, A and B follow each other on Twitter, or A and B know each other.

Attributes of an edge may include:

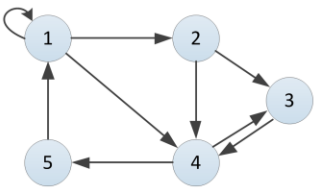
- Weight (such as the frequency of communication)
- Ranking (best friend, second best friend, etc.)
- Type (friend, family, co-worker)
- Properties depending on the structure of the rest of the graph. Later slides will cover some of these properties, such as betweenness.

Directed and Undirected Graphs

- Undirected graph: A graph with only two-way edges
  - Co-authorship network
  - Actor network
  - Protein-protein interactions



- Directed graph: A graph with one-way edges
  - URLs on the WWW
  - Phone calls
  - Metabolic reactions



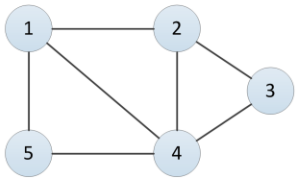
An undirected graph is a graph with only two-way edges. In an undirected graph G = (V, E), the edge set E consists of unordered pairs of nodes (or vertices). In contrast, a directed graph is a graph with one-way edges. In a directed graph G=(V, E), the edge set E consists of ordered pairs of nodes.

The slide shows some examples of directed and undirected graphs. Note that real networks can have multiple characteristics:

- WWW: directed graph with self-interactions
- Protein interactions: undirected and unweighted with self-interactions
- Collaboration network: undirected and weighted graph
- Mobile phone calls: directed and weighted
- Facebook friendship links: undirected and unweighted

We will talk about weighted and unweighted graphs in a moment.

Data Representation of Undirected Graphs



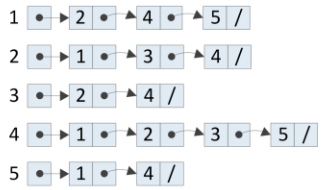
(i) Edge List

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

(ii) Adjacency Matrix

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

(iii) Adjacency List



A graph G=(V, E) can be represented in three different ways: edge list, adjacency matrix, and adjacency list.

The edge list simply lists the start node and the end node of each edge. For example, (1,2) represents the edge from node 1 to node 2.

The adjacency matrix consists of a |V|×|V| matrix such that for each $a_{ij} \in A$:

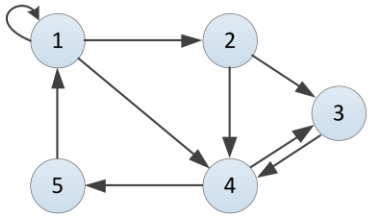
$$a_{ij} = \begin{cases} 1, & if\,(i,j) \in E \\ 0, & otherwise \end{cases}, \qquad 1 \le i,j \le |V|$$

Note that for undirected graphs, $a_{ii} = 0$, $a_{ij} = a_{ji}$.

The adjacency list consists of an array of |V| lists, one for each node in V. For each node u in V, the adjacent list stores all the nodes v such as there is an edge (u, v) ∈ E.

When the network is sparse, i.e., far less number of edges than the number of nodes, the edge list maybe a good choice. Both the edge list and the adjacency matrix allow fast read and write when the network is small. However, when the network is large, the edge list becomes inefficient due to the overhead of looping and sorting through the list. The space complexity of an adjacency matrix exponentially grows as the number of nodes increases, so it may not be a good choice for a large network or a sparse network. The adjacency list works well if the network is large or sparse because it allows fast read and write and it also saves storage. Readers are recommended to considers both time and space complexities and choose the most suitable representation according to their specific problems.

Data Representation of Directed Graphs



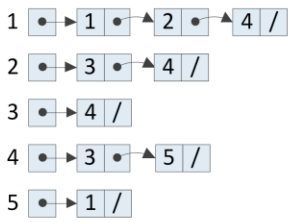
(i) Edge List

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

(ii) Adjacency Matrix

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

(iii) Adjacency List

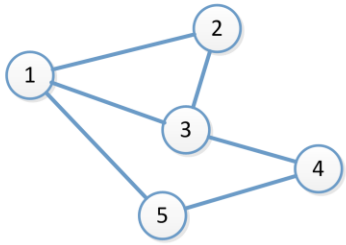


Similarly, the three representations can be applied to directed graphs.

Note that for directed graphs, the diagonal entries in the adjacency matrix $a_{ii} = 1$ if and only if node i has a self-loop, such as node # 1 shown in the slide.

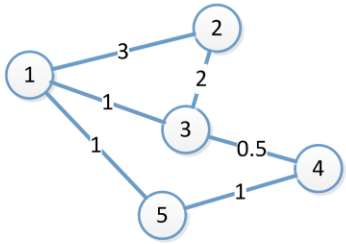
Unweighted and Weighted Graphs

- Unweighted graph
- $A = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$



Protein-protein interactions, WWW

- Weighted graph
- $A = \begin{pmatrix} 0 & 3 & 1 & 0 & 1 \\ 3 & 0 & 2 & 0 & 0 \\ 1 & 2 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$



Call graph, Metabolic networks

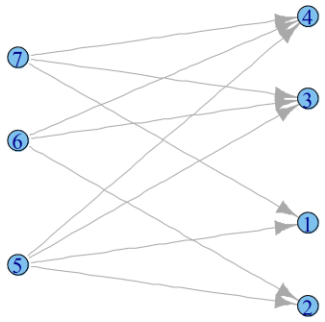
An unweighted graph doesn't associate weight to its edges. In the adjacency matrix representation, each edge by default has a weight 1.

A weighted graph labels every edge in the graph with a weight. Each weight is usually a real number. Weights are commonly seen to be positive, although in some cases, they may be negative, e.g., spreading negative feelings across a social network.

By default, $A_{ii} = 0$, $A_{ij} = A_{ji}$ for both unweighted and weighted graphs. One exception of $A_{ii} \neq 0$ is when node i in a directed graph has a self-loop, i.e., an edge pointing from node i to itself.

Bipartite Graph (or Bigraph)

- A graph whose nodes can be divided into two disjoint sets X and Y such that
  - every link connects a node in X to one in Y
  - that is, X and Y are independent sets
- Examples
  - Flavor/ingredient network
  - Disease network
  - Hollywood actor network
  - Author collaboration network

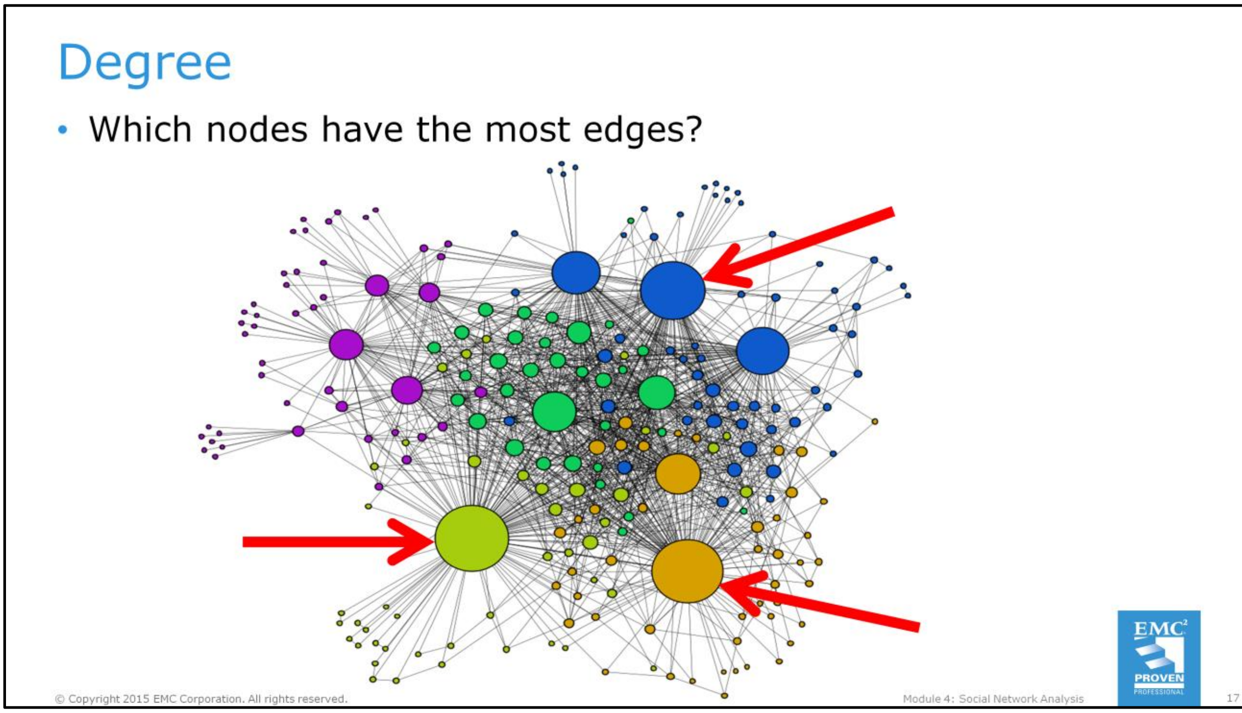


Bipartite graph, or bigraph, is a graph whose nodes can be divided into two disjoint sets X and Y such that every link connects a node in X to one in Y; that is, X and Y are independent sets.

At the beginning of this module, we saw the visualization of an ingredient / flavor compound network. That is an example of bipartite graph, where each edge maps an ingredient on the left (e.g., shrimp) to a flavor compound on the right (e.g., 1-penten-3-ol). Some other examples include a human disease network that maps a disease to a human gene, a Hollywood actor network that maps actors to their movies, and an author collaboration network that maps authors to their papers.

The following R codes show how to generate a random directed bipartite graph. It requires the `igraph` package which can be obtained by executing `install.packages("igraph")` in the R prompt.

```
library(igraph)
g <- bipartite.random.game(4, 3, p=0.8, mode="in", directed=TRUE)
lay <- layout.bipartite(g)
# Two column view
plot(g_bi, layout=lay[,2:1])
```



Next, we will introduce the metric, or degree, which is one of several properties of nodes.

Shown on the slide is a network of 1,297 flights among 235 airports, plotted using Gephi (http://gephi.github.io/). Each node represents an airport and each edge represents a flight between two airports. The size of a node corresponds to the number of flights that leaves or arrives at this airport. The data set can be obtained from: https://gephi.org/datasets/airlines.graphml.zip.

Graph Elements: Nodes and Node Properties

- Immediate connections (neighbors)
  - In-degree (number of inlinks)

Indegree: 2

  - Out-degree (number of outlinks)

Outdegree: 3

  - Degree (in or out)

Degree: 5

- Entire graph
  - Centrality (betweenness, closeness)

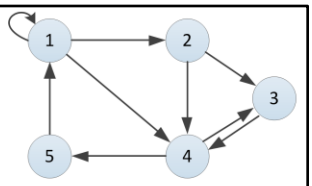
We can use degree to measure the immediate connections, i.e., neighbors, of a node. The degree of a node in an undirected graph is the number of edges incident on it.

For directed graphs, the degree can be further divided into in-degree and out-degree. The in-degree of a node is the number of directed edges that point to the node, that is, the number of inlinks. The out-degree of a node is the number of edges originating at it, that is, the number of outlinks.

The slide shows three examples. The first node has an in-degree of 2. The second node has an out-degree of 3 and the third node has a degree of 5.

At the level of the entire graph, centrality can help us measure the importance of each node. We will discuss a few centrality measures (such as betweenness and closeness) later in the lesson.

Identifying Degree from Adjacency Matrix



- Out-degree: $\sum_{j=1}^{n} a_{ij}$
  - Example:
    - Out-degree for node 4 is 2, obtained by summing the number of non-zero entries in the 4th row

$$A = \begin{Bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{Bmatrix} \qquad \sum_{j=1}^{5} a_{4j} = 2$$

- In-degree: $\sum_{i=1}^{n} a_{ij}$
  - Example:
    - The in-degree for node 3 is 2, obtained by summing the number of non-zero entries in the 3rd column

$$A = \begin{Bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{Bmatrix} \qquad \sum_{i=1}^{5} a_{i3} = 2$$

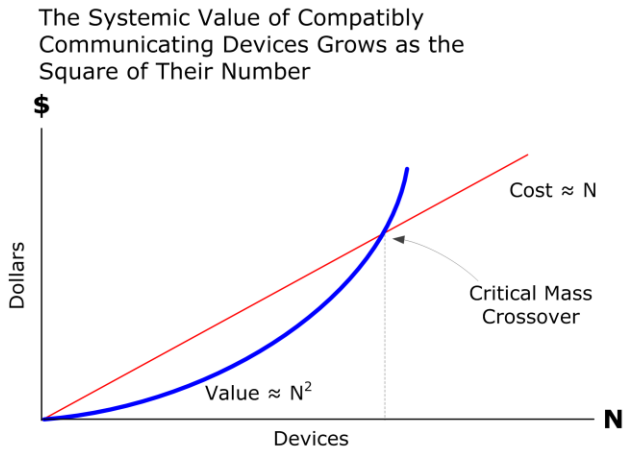
The slide shows a directed graph and its adjacency matrix. You can easily calculate the out-degree and in-degree of a node from the adjacency matrix.

To compute the out-degree of node *i*, you simply sum up all the numbers from the *i*-th row in the matrix. For example, the out-degree of node 4 is 2.

To compute the in-degree of node *i*, you sum up all the numbers from the *i*-th column in the matrix. For example, the in-degree of node 3 is 2.

Metcalfe's Law States the Value of a Network

- The value of a network is proportional to the square of the number of its nodes, N^2



Metcalfe's law, frequently quoted during the internet boom of 2000, states that the value of a network is proportional to the square of the number of its nodes, i.e. N^2. First formulated in this form by George Gilder in 1993, and attributed to Robert M. Metcalfe, the inventor of Ethernet, the idea behind Metcalfe's law is that the more individuals use a network, the more valuable it becomes. A fax machine is useless to you if there is no one to send a fax to. The more your acquaintances have a fax machine, the more valuable it is to you as well.

Reference:

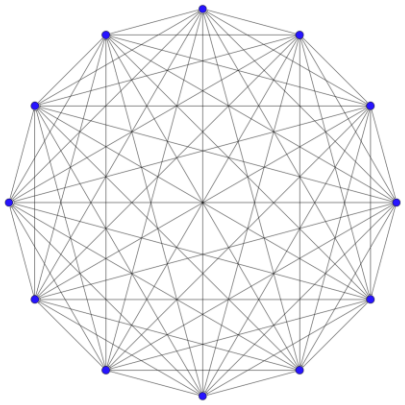
Gilder, G. (1993). Metcalf's law and legacy. *Forbes ASAP*, pp. *27*.

Complete Graph

- The maximum number of edges in a network of N nodes is

$$L_{max} = \binom{N}{2} = \frac{N(N-1)}{2}$$

- A graph containing L_{max} edges is called a **complete graph**
- Most networks in real systems are sparse



A notion related to the Metcalfe's law is the complete graph. For a network with N nodes, the maximum number of links of the network is $N \times (N-1)/2$. Such a network is called a **complete graph**. If a network has N=50 members, there are $L_{max} = 1{,}225$ different possible connections that these members can make to each other. If the network doubles in size to N=100, the maximum number of connections roughly quadruples and becomes $L_{max} = 4{,}950$, an effect often called **network effect** in economics.

Metcalfe's law defines that, if the nodes in a complete graph are equally valuable, then the total value of the network is proportional to $N \times (N-1)/2$, that is, roughly, N^2.

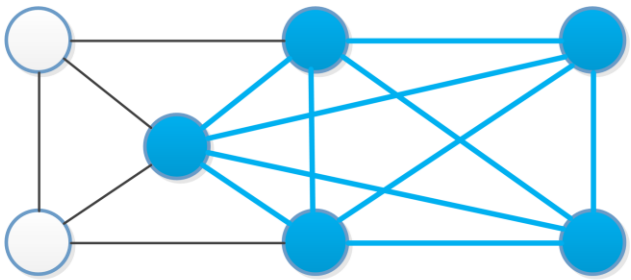
Note that, while all links are possible, in real networks not all links are present. In fact, most networks observed in real systems are sparse. Also, not all links are of equal value. Some links are used heavily while the vast majority of links are rarely utilized.

Reference:

Gilder, G. (1993). Metcalf's law and legacy. *Forbes ASAP*, pp. *27*.

Clique

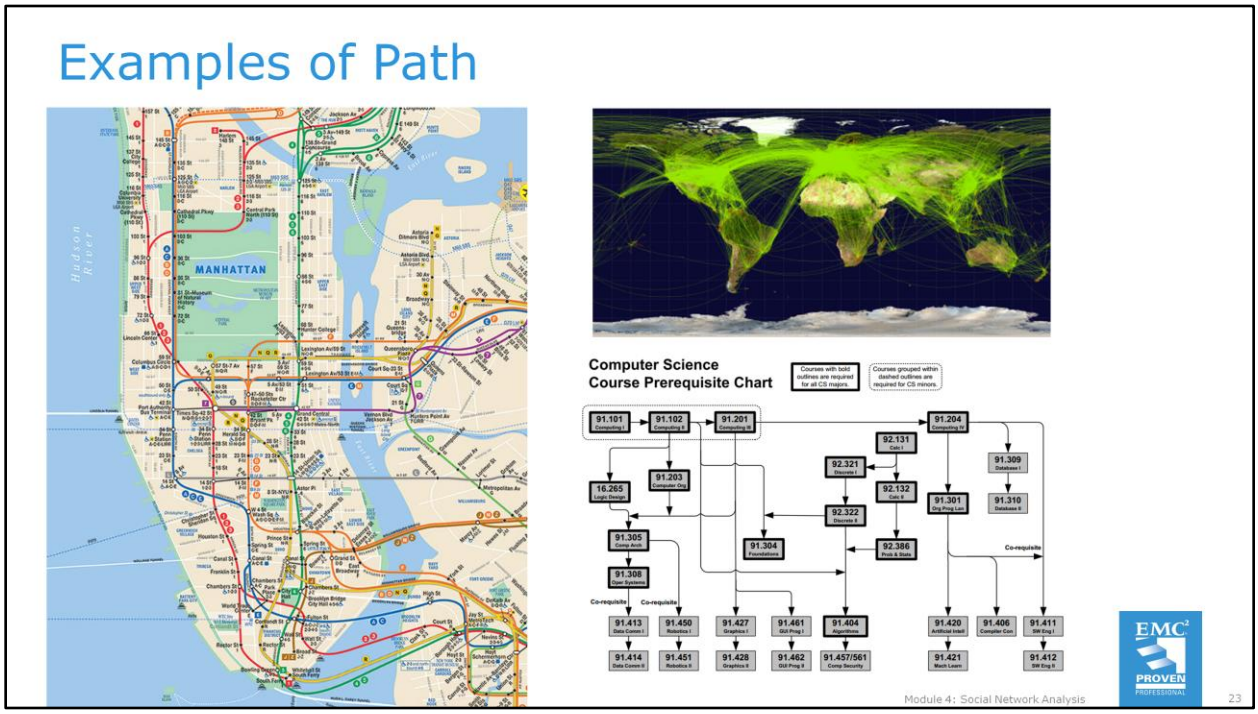
- The complete subgraph is called a **clique**
- k-clique: a clique with k nodes



A 5-clique in a undirected graph

Related to the complete graph is the notion of clique. A clique in an undirected graph G = (V, E) is a subset V' ⊆ V of vertices, each pair of which is connected by an edge in E. Therefore, a clique is a complete subgraph of G. A clique with k nodes is called a k-clique.

The example shows a 5-clique in a 7-node undirected graph.

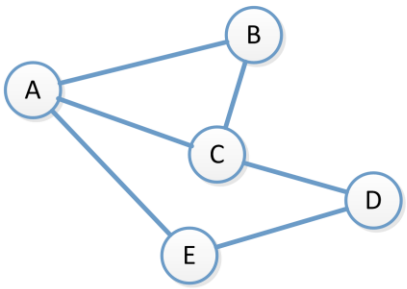


Path is an important notion in both graph theory and social network analysis. Here are a few real-world examples of network paths:

- New York City subway map: The route between any two connected stations is a path
- Airline routes: Each trip a passenger take from the origin airport to the destination airport (including connections) is considered as a path
- Flowchart of college course prerequisites: A course may have one or more prerequisite courses. A path connects two courses with one or more consecutive arrows.

Definition of Path

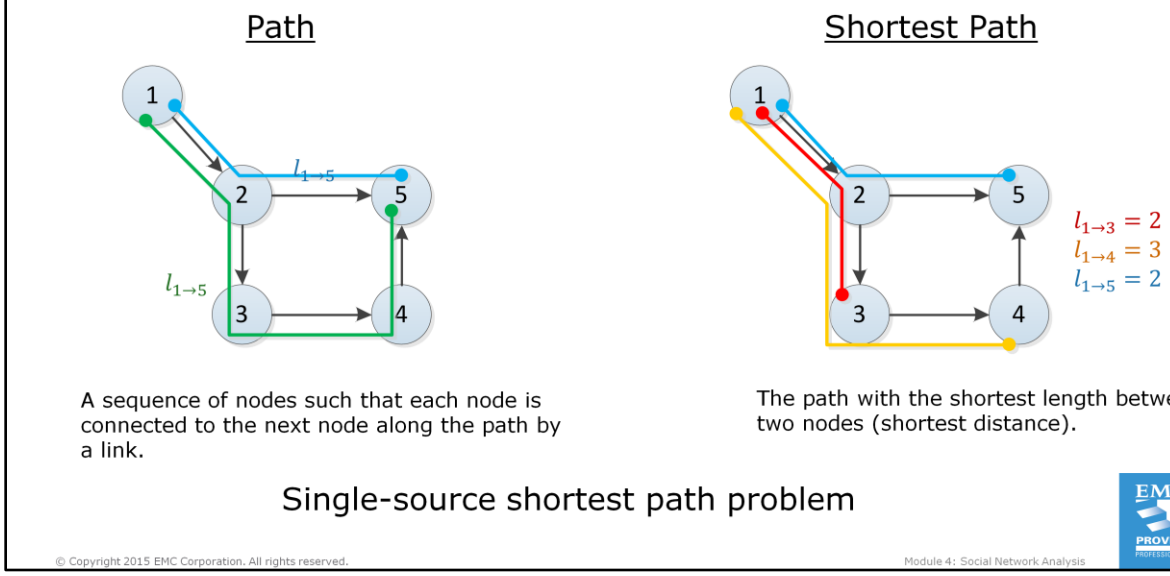
- Path is a sequence of nodes with the property that each consecutive pair in the sequence is connected by an edge
- Path can be one way or reciprocal
- Example: There are three paths between nodes A and C
  - A ↔ C
  - A ↔ B ↔ C
  - A ↔ E ↔ D ↔ C



Path is defined as a sequence of nodes where each consecutive pair in the sequence is connected by an edge in the graph. Path can be one way (for directed graphs) or reciprocal (for directed and undirected graphs).

For example, there are three paths between nodes A and C. These paths are reciprocal since this is a undirected graph.

Path and Shortest Path



In a network, **path** is defined as a sequence of nodes such that each node is connected to the next node along the path by a link. The **shortest path** is defined as the path with the shortest length between two nodes, i.e. the shortest distance.

In the example graph shown on the slide, there are two paths from node 1 to node 5: 1-2-5 (a distance of 2) and 1-2-3-4-5 (a distance of 4). The shortest path from node 1 to node 5 is 1-2-5 (a distance of 2). Similarly, the shortest path from node 1 to 3 is 1-2-3 (a distance of 2) and the shortest path from node 1 to node 4 is 1-2-3-4 (a distance of 3).

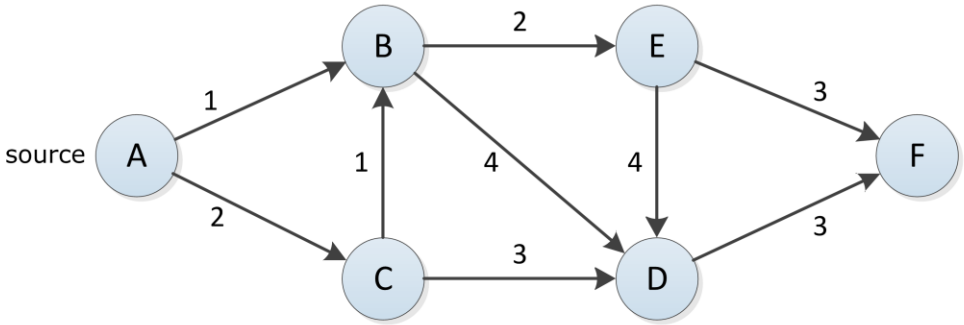
Shortest path is an important metric in network analysis. It can help answer questions such as:

- How many intermediate friends are connecting Bob and Alice?
- What is the shortest route to move cargo from station A to station B?
- What is the fastest route driving from the hotel to the airport?
- Which routers should be used to direct the traffic from server X to client Y?

In graph theory, the **single-source shortest path problem** is the problem of finding the shortest paths from a single source vertex to all other vertices in the graph.

Dijkstra's Algorithm: Solves the single-source shortest path problem

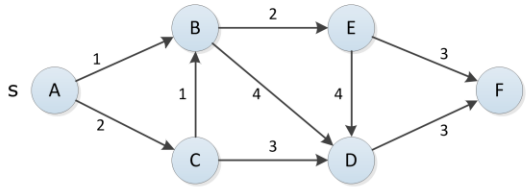
- Is a greedy algorithm that works for both directed and undirected graphs
- Requires nonnegative edge weights and a connected graph



Dijkstra's algorithm solves the single-source shortest-path problem on a weighted, directed or undirected graph $G = (V, E)$ for the case in which all edge weights are nonnegative. Dijkstra's algorithm is a greedy algorithm in that it makes a locally optimal choice at each iteration with the hope of finding a global optimum. The graph is required to be connected.

Weight of any two nodes u and v is assumed to be at least 0. That is, $\text{weight}(u, v) \geq 0$ where $u, v \in V$.

Dijkstra's Algorithm




```
1  dist[s] ← 0                                (distance to source vertex is zero)
2  path[s] ← [s]                              (path to source vertex is the source vertex itself)
3
4  for all v ∈ V - {s} do
5     dist[v] ← ∞                             (set all other distances to infinity)
6     path[v] ← ∞                             (set all other paths to infinity)
7  S ← ∅                                      (S, the set of visited vertices is initially empty)
8  Q ← V                                      (Q, the queue initially contains all vertices)
9  while Q ≠ ∅ do                             (while the queue is not empty)
10    u ← minDistance(Q, dist)                (select the element of Q w/ the min. value of dist)
11    S ← S U {u}                             (add u to list of visited vertices)
12    for all v ∈ neighbors[u] do
13        if dist[v] > dist[u] + weight(u, v) then     (if new shortest path found)
14            dist[v] ← dist[u] + weight(u, v)         (set new value of minimum distance)
15            path[v] ← path[u] + {v}                  (store the new shortest path)
16    Q ← Q - {u}                                (remove u from Q)
17 return dist, path
```


This slide shows the full Dijkstra's algorithm. The *dist* stores the shortest distance from the source node *s* to every node in the graph. The *path* stores the nodes along a shortest path from the source node *s* to every node in the graph. Below is how the Dijkstra's algorithm works at each iteration, given the example graph where node A is the source *s*:

Initialization:

S = {}, Q = {A, B, C, D, E, F}, dist = {A: 0, B: ∞, C: ∞, D: ∞, E: ∞, F: ∞}, path = {A: [A], B: ∞, C: ∞, D: ∞, E: ∞, F: ∞}

After the 1st iteration:

u = A, S = {A}, dist = {A: 0, B: 1, C: 2, D: ∞, E: ∞, F: ∞},

path= {A: [A], B: [A, B], C: [A,C], D: ∞, E: ∞, F: ∞}, Q = {B, C, D, E, F}

After the 2nd iteration:

u = B, S = {A, B}, dist = {A: 0, B: 1, C: 2, D: 5, E: 3, F: ∞}, path= {A: [A], B: [A, B], C: [A,C], D: [A,B,D], E: [A,B,E], F: ∞}, Q={C, D, E, F}

After the 3rd iteration:

u = C, S = {A, B, C}, dist = {A: 0, B: 1, C: 2, D: 5, E: 3, F: ∞}, path= {A: [A], B: [A, B], C: [A,C], D: [A,B,D], E: [A,B,E], F: ∞}, Q={D, E, F}

After the 4th iteration:

u = E, S = {A, B, C, E}, dist = {A: 0, B: 1, C: 2, D: 5, E: 3, F: 6}, path= {A: [A], B: [A, B], C: [A,C], D: [A,B,D], E: [A,B,E], F: [A,B,E,F]}, Q={D, F}

After the 5th iteration:

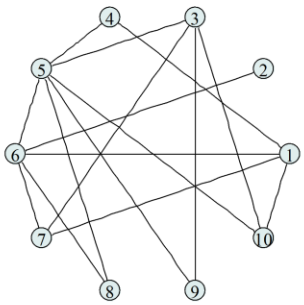
u = D, S = {A, B, C, E, D}, dist = {A: 0, B: 1, C: 2, D: 5, E: 3, F: 6}, path= {A: [A], B: [A, B], C: [A,C], D: [A,B,D], E: [A,B,E], F: [A,B,E,F]}, Q={F}

After the 6th iteration:

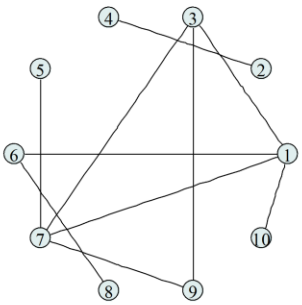
u = F, S = {A, B, C, E, D, F}, dist = {A: 0, B: 1, C: 2, D: 5, E: 3, F: 6}, path= {A: [A], B: [A, B], C: [A,C], D: [A,B,D], E: [A,B,E], F: [A,B,E,F]}, Q={}

Network Diameter

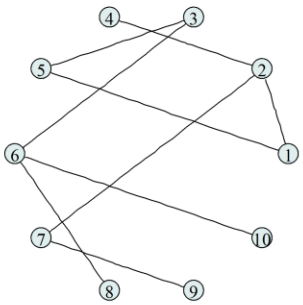
- The *longest shortest path* between any two nodes



Diameter: 3



Diameter: 5



Diameter: 7

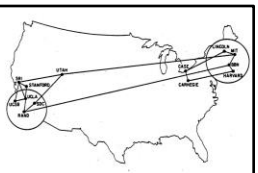
The network diameter is the longest shortest path between any two nodes.

The graphs shown in the slide have diameters of 3, 5, and 7, respectively. The rightmost graph has a relatively large diameter, because it takes at most 7 edges to travel between one node to another. For example, a path from node 8 to 9 is: 8-6-3-5-1-2-7-9.

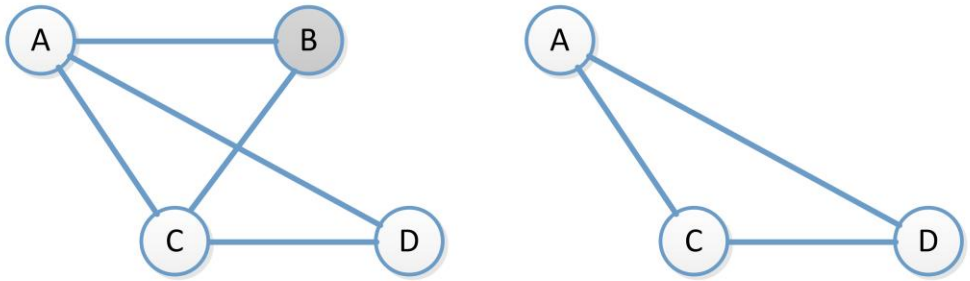
The following R code plots the rightmost graph using the `igraph` package which can be obtained by executing `install.packages("igraph")` in the R prompt. The other two graphs can be plotted similarly.

```
library (igraph)
g3 <- graph( c(1,2,1,5, 2,7,2,4, 3,6,3,5, 6,10,6,8, 7,9), directed=FALSE )
plot(g3, layout=layout.circle, edge.color="black", vertex.color="azure2",
     vertex.label.color="black", vertex.label.cex=1.2)
```

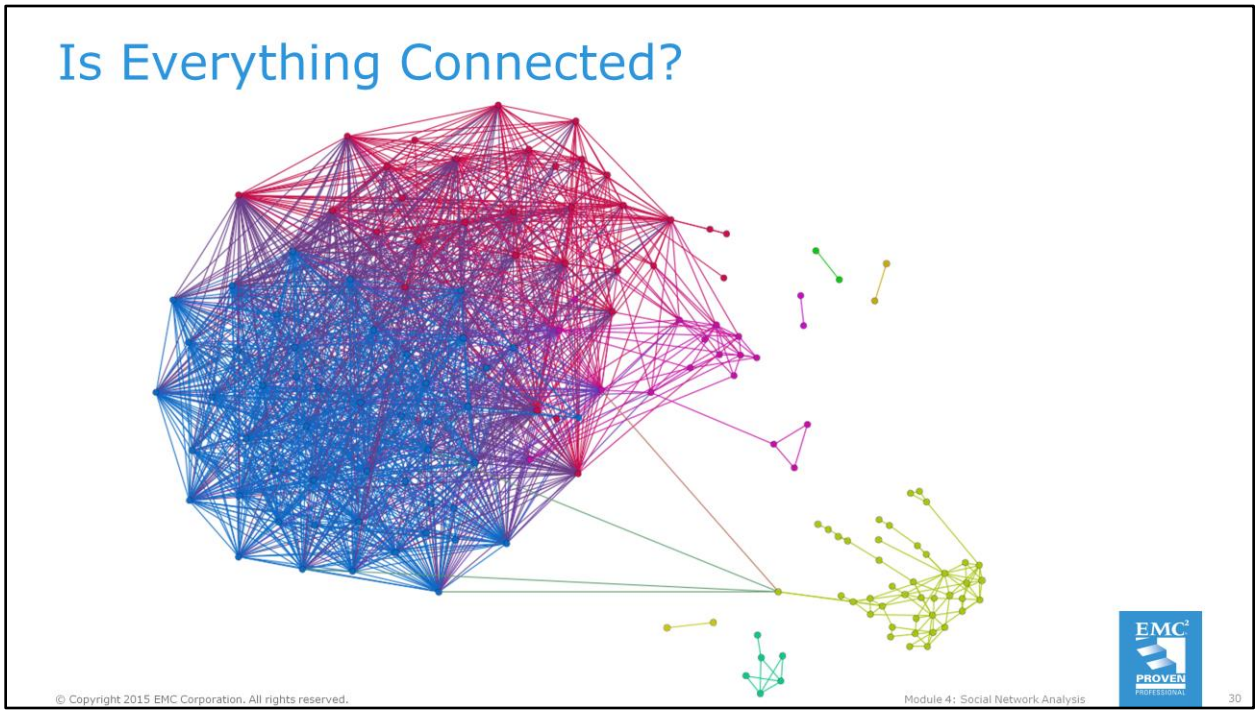
Cycles



- A ring structure with at least three edges
- If any edge fails (e.g. a cable is broken), there is an alternative to route the traffic
  - If node B is down, node A, C and D can still communicate



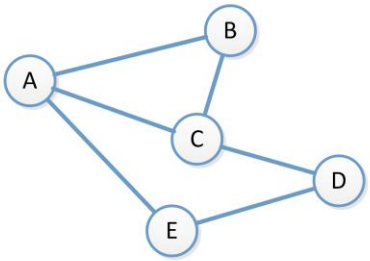
A cycle is like a ring that contains at least three edges. If an edge in the cycle fails, an alternative path can be used to route the traffic.

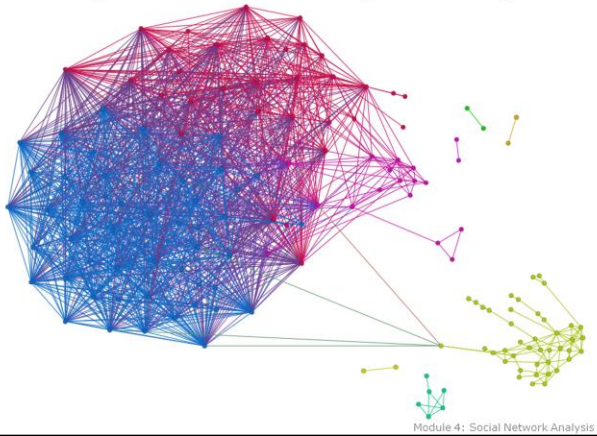


This graph shows a user's friends on Facebook and their connections. The user is not shown on the graph. We can see that this user's friend network is broken down into several connected pieces. Next, let's use this graph to introduce the notion of connectivity.

Connectivity

- Can every node in the graph reach every other node by a path?
  - If yes: we say the graph is **connected**
  - If not: the graph can be broken down into connected pieces (also known as **connected components** or **components**)

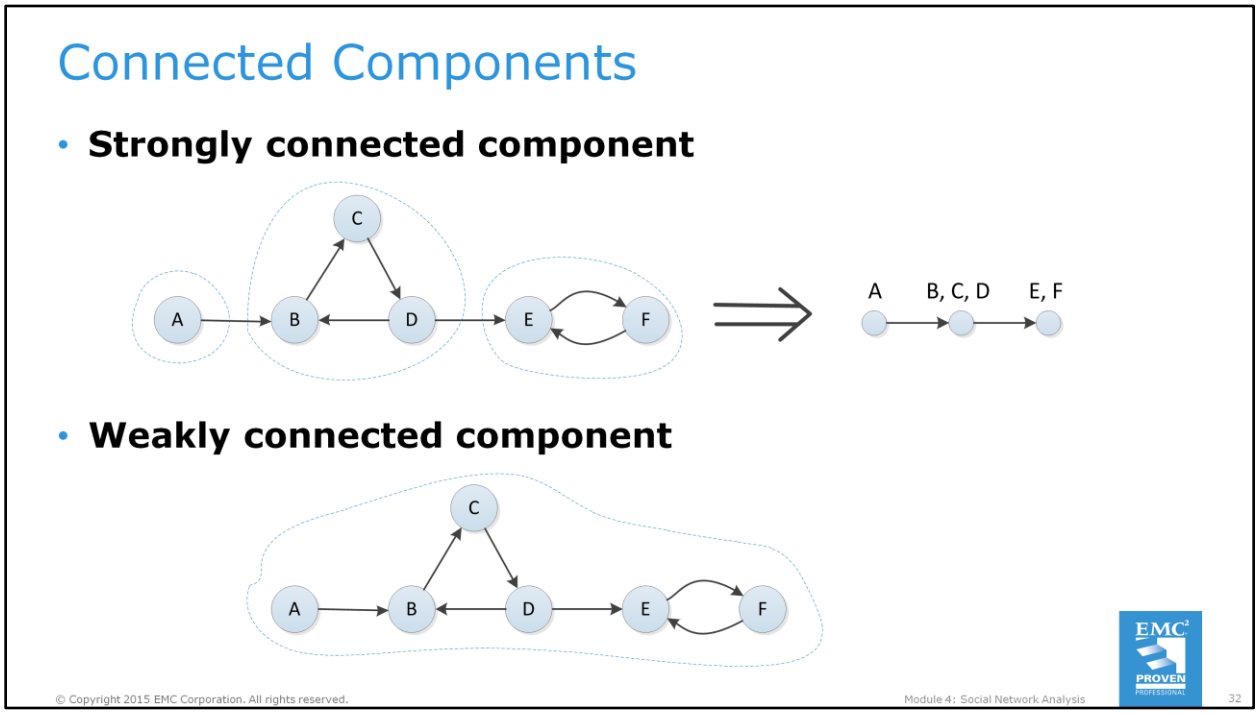




A graph G = (V, E) is connected if every node in the graph can be reached from every other node by following a set of edges, otherwise the graph is not connected and it can be broken down into connected pieces called connected components.

The slide shows two example graphs. The graph on the left is connected and the graph on the right is not connected.

Undirected graphs generally talk about connected components. For directed graphs, connected components can be further categorized into strongly connected components and weakly connected components, as shown on the next slide.



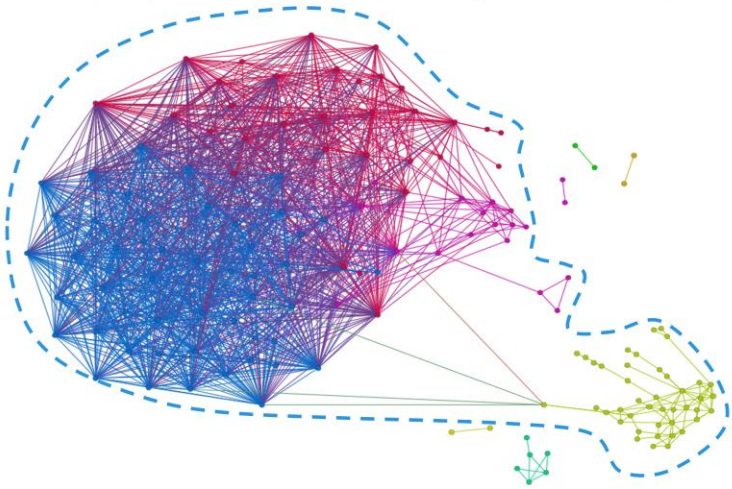
In a directed graph G = (V,E), two nodes u and v are **connected** if there is a path from u to v, and one from v to u. This relation between nodes is reflexive, symmetric, and transitive, so it is an equivalence relation on the nodes. Therefore V can be partitioned into disjoint sets, called the **strongly connected components** of the graph. Each node in a strongly connected component can be reached from every other node in the component by following the directed links. For example, the graph in the slide has three strongly connected components: (1) A, (2) B C D, and (3) E F.

We can shrink each of these strongly connected components down to a single node, and draw an edge between two of them if there is an edge from a node in the first component to a node in the second component. The resulting directed graph has to be a **directed acyclic graph** (DAG), that is, a graph that doesn't have any cycles. This is because, a cycle containing several strongly connected components would merge them all into a single node. In other words, every directed graph is a DAG of its strongly connected components. The example graph therefore can be simplified to a three node graph.

A directed graph G = (V,E) is weakly connected if replacing all of its directed edges with undirected edges produces a connected graph. A **weakly connected component** is a *maximal* subgraph of a directed graph such that each node in the subgraph can be reached from every other node by following links in either direction. The example graph is weakly connected because if you replace the directed edges with undirected edges, every node can be reached from every other node. Therefore this graph only has one weakly connected component: A B C D E F.

Giant Component

- If the largest connected component encompasses *a significant fraction* of the graph, it is called the **giant component**



Definition of the giant component is given here. Basically, a giant component dominates a significant portion of a graph. The next slide will introduce random graphs as a way of studying the giant component.

Erdős-Rényi Random Graph

- Useful for studying the giant component
- Assumptions
  - Undirected graph
  - Nodes are connected at a certain probability
- Represents a random graph as G(n,p) or G(n,m)
  - n: the number of nodes
  - p or m (choose one)
    - p: probability for drawing an edge between two arbitrary nodes
    - m: the number of edges

G(n=100, p=0.01)

The Erdös-Rényi random graph is named after Paul Erdős and Alfréd Rényi, who first introduced the model in 1959. The Erdös-Rényi random graph is useful for studying the giant component, especially the percolation threshold presented in the next slide.

Given a positive integer n and a probability value $0 \leq p \leq 1$, the Erdős-Rényi model defines a graph $G(n,p)$ to be the undirected graph on n vertices and for all pairs of vertices u,v there is an edge (u,v) with probability p.

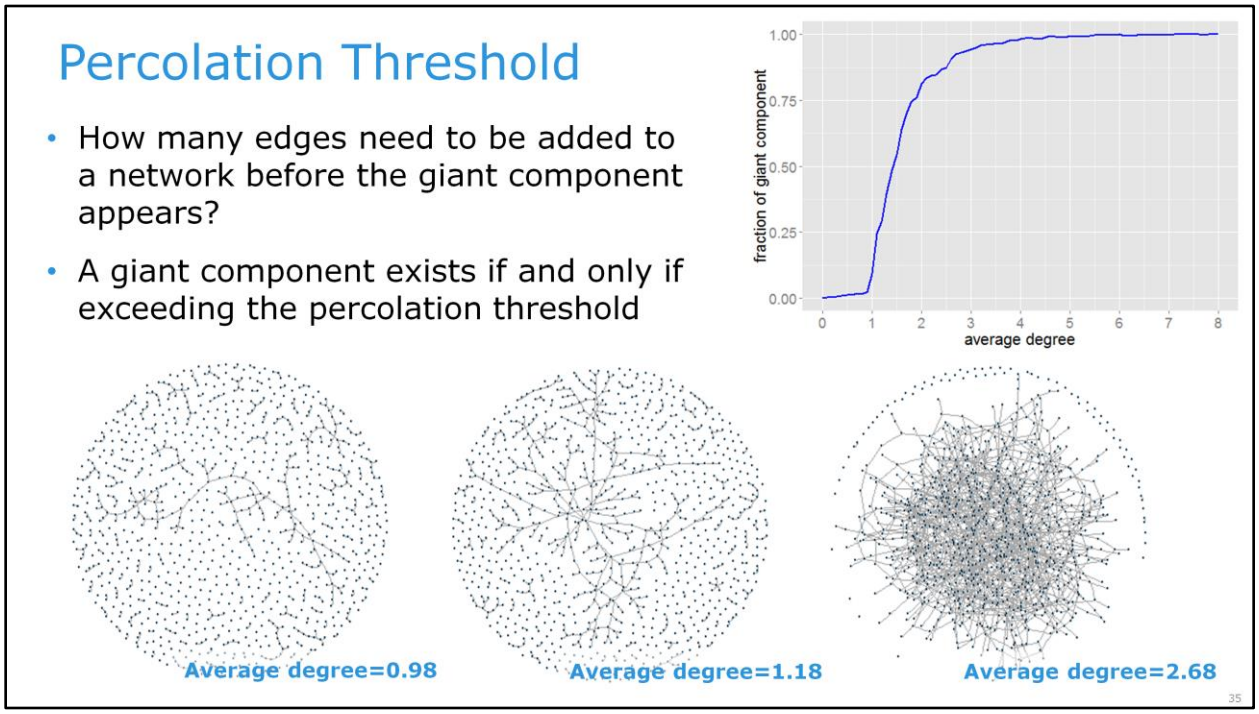
Alternatively, the Erdős-Rényi model can define parameter m in place of p for the number of edges of the graph.

Shown in the slide is an Erdős-Rényi random graph of 100 vertices and $p = 0.01$. This graph can be generated using the following R code. The `igraph` package is required and you can obtain it by executing `install.packages("igraph")` in the R prompt.

```
library (igraph)
g <- erdos.renyi.game(n=100, p=0.01, type="gnp")
plot(g,layout=layout.fruchterman.reingold, vertex.size=4, vertex.label=NA)
```

Reference:

Erdős, P. and Rényi, A. (1959). On random graphs. *Publicationes Mathematicae,* 6, pp. 290–297.



Percolation threshold focuses on the formation of long-range connectivity in random systems. **A giant component exists if and only if the percolation threshold is exceeded.** For social network analysis, percolation threshold tells how many edges (or the average degree per node) a network should have before a giant component appears.

The plot in the slide shows the fraction of the giant component in a random graph in relation to the average degree per node. As the average degree increases to 1, a giant component suddenly appears.

The graphs on the bottom of the slide shows three random graphs each with 1,000 nodes. The average degree per node is at 0.98, 1.18, and 2.68, respectively. It confirms that the giant component appears when the average degree increases to 1.

Percolation threshold is a useful notion in social network analysis. For example, think about this question: how many other friends besides you does each of your friends have? When the average degree of your friends is 1, each of your friends is expected to have another friend, who in turn has another friend, so on and so forth. Therefore the giant component emerges.

Erdős-Rényi Random Graph: The G(*n*, *p*) Model

- If $np > 1$, then graph G(*n*, *p*) will almost surely have a unique giant component
  - Percolation threshold
- The threshold of the connectivity of G(*n*, *p*) depends on $\frac{\ln n}{n}$
  - If $p \ll \frac{\ln n}{n}$, then the graph will have isolated vertices, and thus be disconnected
  - If $p \gg \frac{\ln n}{n}$, then the graph will be connected

As discussed before, the Erdős-Rényi random graph model represents a random graph as either $G(n,p)$ or $G(n,m)$. Let's look at the $G(n,p)$ representation, and more particularly, how the random graph changes in terms of p, the probability for drawing an edge between two arbitrary nodes in the graph.

In Erdős and Rényi's work in 1960, they describes how changing *p* affects the random graph:

- If $np \rightarrow c$ where c is a constant and c>1, then a unique giant component will emerge in the random graph. Since np corresponds to the average degree of the graph, this matches the percolation threshold in the previous slide.
- The threshold of the connectivity of $G(n,p)$ depends on $\frac{\ln n}{n}$ and
  - If $p \ll \frac{\ln n}{n}$, then the graph will have isolated vertices, and thus be disconnected
  - If $p \gg \frac{\ln n}{n}$, then the graph will be connected

Reference:

Erdős, P., and Rényi, A. (1960). On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5, pp. 17-61.

Check Your Knowledge

- How is social network analysis used in the real world?
- What are the three data representations of a directed graph?
- How does the weakly connected component differ from the strongly connected component?
- What is a giant component?
- What is the percolation threshold of a random graph?

Write your answers here.

Lesson 1: Summary

During this lesson the following topics were covered:

- Basic notions of Social Network Analysis
- Graph Elements and Graph Theory
- Connectivity
- Giant Component
- Percolation Threshold

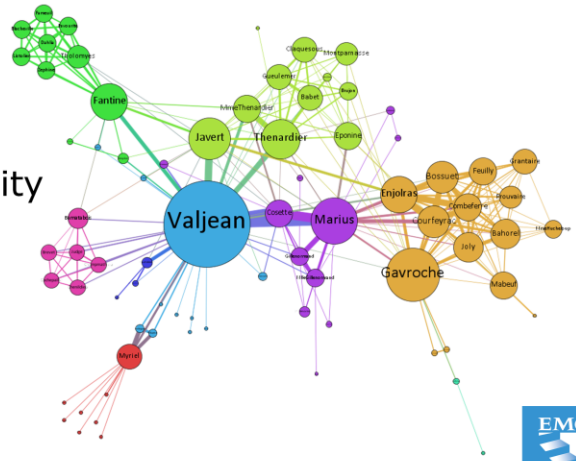
Lesson 2: Most Important Nodes

This lesson covers the following topics:

- Degree Centrality
- Closeness Centrality
- Betweenness Centrality
- Eigenvector Centrality and PageRank

Discover the Most Important Nodes with Centrality

- Why does it matter to discover the most important nodes?
  - Crucial for studying contagious outbreaks in a network
    - Pathogen
    - Information
    - Norms
    - Behaviors
- Different measures of centrality
  - Degree
  - Closeness (shortest path)
  - Betweenness
  - Eigenvector centrality
    - PageRank



Centrality refers to indicators which identify the most important nodes within a graph. Studying the important nodes is crucial for researching the contagious outbreaks in a network, such as pathogen, information, norms and behaviors.

There are many existing measures for centrality, such as degree centrality, closeness centrality, betweenness centrality and eigenvector centrality (PageRank). We will discuss these different measures in a moment.

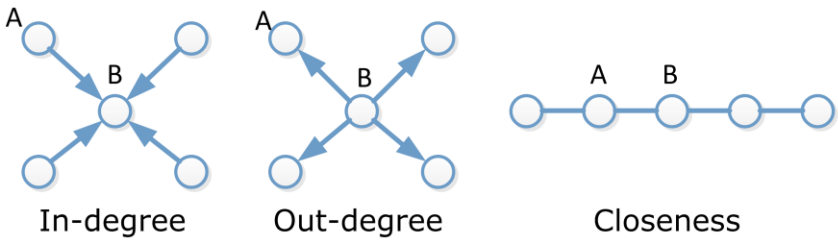
Note that a centrality which is optimal for one application is often sub-optimal for a different application. That's why we need so many different centralities. The definition of "central" also varies by context/purpose. Generally different centrality measures are positively correlated. When they are not correlated, it often suggests something interesting about the network. When you are conducting social network analysis, it's recommended to try several centrality measures to help discover interesting aspects of the network.

Centrality can be used for many applications, such as WWW, food webs, influence, hereditary, paper citations, diseases, and neural networks.

The graph in the slide visualizes the characters and their co-appearances from the book Les Misérables. The size of a node represent its degree. We will look at the degree centrality in the next slide. The color represents the modularity, a measure we will discuss later in the module.

Measure Centrality with Degree and Closeness

- Degree and closeness can measure centrality of each node
  - In-degree: which node has the highest in-degree
  - Out-degree: which node has the highest out-degree
  - Closeness: which node is the closest to all the other nodes
- In each of the following networks, B always has a higher centrality than A with a particular measure



In-degree Out-degree Closeness

Degree and closeness can measure centrality of each node in a graph.

As seen earlier in the lesson, in-degree and out-degree tell us which nodes have the highest inlinks or outlinks. In-degree and out-degree can be used to measure the centrality of a node. The node that receives the highest value is considered the center of the graph.

Closeness is based on the length of the average shortest path between a node and all other nodes in the network. It measures which node is the closest to all the other nodes. A node with the smallest closeness score is considered the center of the graph.

In the example graphs shown on the slide, node B always has a higher centrality than A for each measure of centrality.

In-degree(A)=0, In-degree(B)=4, therefore B has a higher in-degree centrality.

Out-degree(A)=0, Out-degree(B)=4, therefore B has a higher out-degree centrality.

Closeness(A)=1+1+2+3=7, Closeness(B)=2+1+1+2=6 , therefore B has a higher closeness centrality.

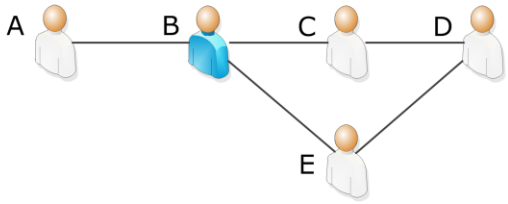
Betweenness Centrality

- Measures the importance of a node by counting the number of shortest paths through the graph that include that node:

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

- Often expressed in a normalized form by dividing g(v) by the total number of possible paths not using v as an endpoint.

- For example,
  - Betweenness (C) ≈ 0.17
  - Betweenness (B) ≈ 0.58



The betweenness centrality of a node (v) is the sum of the ratios of $\sigma_{st}(v)$ (the number of shortest paths between nodes s and t that go through v) to σ_{st} (the total number of shortest paths between s and t) for all the paths between two different nodes in the graph where neither end-point is v. The notion of betweenness is based on communication flow. A person who lies on the communication paths can control communication flow, and is thus important. For each pair of nodes s and t in the graph that are connected by a path, we imagine having one unit of fluid "flow" along the edges from s to t. If s and t belong to different connected components, then no fluid flows between them. The flow between s and t divides itself evenly along all the possible shortest paths from s and t; so if there are m shortest paths from s and t, then 1/m units of flow pass along each one.

The example social network shown on the slide contains 5 nodes. Consider calculating the betweenness for node C. Since we aren't examining paths with C as an endpoint, there are (5-1)*(5-2)/2 =6 possible paths to be examined. These paths include: [A,B], [A,D], [A,E], [B,D], [B,E], and [D,E].
Calculating g(C) –
there is one shortest path for [A,B], it does not include C – this adds 0 to g(C)
there are two shortest path for [A,D], one includes C – this adds 1/2 to g(C)
there is one shortest path for [A,E], it does not include C – this adds 0 to g(C)
there are two shortest path for [B,D], one includes C – this adds 1/2 to g(C)
there is one shortest path for [B,E], it does not include C – this adds 0 to g(C)
there is one shortest path for [D,E], it does not include C – this adds 0 to g(C)
The g(C) is 1 (0+1/2+0+1/2+0+0) which can be normalized by dividing by 6 or ≈ .17

To calculating betweenness for Node B, the paths to examine are: [A,C], [A,D], [A,E], [C,D], [C,E], and [D,E].
Calculating g(B) –
there is one shortest path for [A,C], it includes B – this adds 1 to g(B)
there are two shortest path for [A,D], both include B – this adds 1 to g(B)
there is one shortest path for [A,E], it includes B – this adds 1 to g(B)
there is one shortest path for [C,D], it does not include B – this adds 0 to g(B)
there are two shortest path for [C,E], one includes B – this adds 1/2 to g(B)
there is one shortest path for [D,E], it does not include B – this adds 0 to g(B)
The g(B) is 3.5 (1+1+1+0+1/2+0) which can be normalized by dividing by 6 or ≈ .58

We can repeat this process for node A and D as well. It turns out node B has the highest betweenness centrality. Therefore B is considered the center of the network. If node C is removed from the network, A, B, D and E can still communicate with each other; however, if node B is removed, all paths to A are broken.

The maximum normalized betweenness for a Node is 1 and can be achieved in a network with a central hub and all other nodes are spokes from that hub. There is one shortest path between any two nodes (not including the hub) and the hub is in the middle of every path.

NOTE: All of the above is for undirected graphs. The concept can be expanded for directed graphs.

Eigenvector Centrality

- The measure of the influence of a node in a network
- Basic idea: **an important node is connected to important neighbors**
  - Connections to high-scoring nodes contribute more
  - Each vertex is assigned a score proportional to the sum of scores of its neighbors
- Google's PageRank is a variant of eigenvector centrality

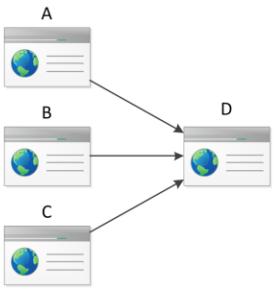
Eigenvector centrality is a metric that measures the influence of a node in a network. Nodes are assigned relative scores based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes.

Eigenvectors were first developed by Bonacich, who was looking for a way to improve on degree closeness.

Next, let's take a look at Google's PageRank, a variant of the eigenvector centrality.

Overview of PageRank

- PageRank is a "vote" by all the other pages on the Web, about how important a page is
  - A link to a page counts as a vote of support
  - PageRank of a page depends on the PageRank of the pages pointing to it
- Two factors
  - **Inlinks**: D has inlinks from A, B and C
  - **Outlinks**: A, B and C have outlinks to D
- Inlinks from important pages are more significant than inlinks from average pages



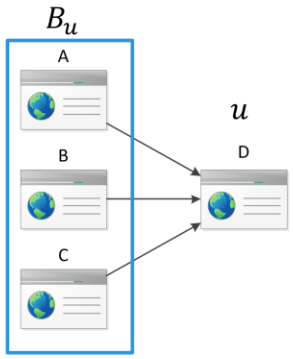
PageRank is a method for rating the importance of web pages using the link structure of the web.

It was proposed by Sergey Brin and Lawrence Page at Stanford in 1998, before they co-founded Google.

Reference:

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, *30*(1), pp. 107-117. http://infolab.stanford.edu/~backrub/google.html

Common Definition of PageRank



- The PageRank (PR) of a webpage u is defined as
  - $PR(u) = \frac{1-d}{N} + d \cdot \sum_{v \in B_u} \frac{PR(v)}{L_v}$
    - B_u: the set of webpage u's inlinks
    - L_v: the number of outlinks of page v
    - d: a damping factor $d \in (0,1]$ (e.g., $d = 0.85$)
    - N: total number of webpages
- Question: Webpage D has three inlinks A, B and C. Assuming PR(A)=0.2, PR(B)=0.4, PR(C)=0.9. The number of forward links of A, B and C are 100, 20, and 2, respectively. Let d=0.85 and N=15,000. What is the PageRank of D?

The PageRank of a webpage u, PR(u) is defined in the slide. Its parameters include: B_u which is the set of u's backlinks, v which references each webpage that points to u, L_v which is the number of forward links of page v, and finally d which is a damping factor set between 0 and 1. Sergey Brin and Lawrence Page used *d*=0.85 in their experiments.

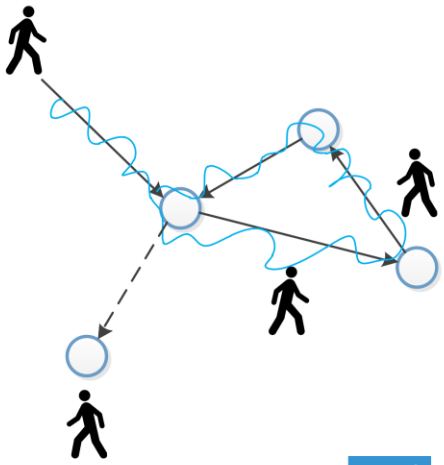
PageRanks form a probability distribution over all the web pages. The PageRank of each webpage is less than 1. PageRanks is typically normalized so the PageRanks of all webpages sum up to 1.

Given the question in the slide, the PageRank of webpage D is:

$$PR(D) = \frac{0.15}{15000} + 0.85 * \left(\frac{0.2}{100} + \frac{0.4}{20} + \frac{0.9}{2}\right) \approx 0.4$$

Iteratively Compute PageRank with Random Walk

- Let a random walker explore a network of webpages for a long time
- At any time t, he is on some page v
- At time t+1, he follows an outlink from v uniformly at random and ends up on some page u linked from v
- Allow him to *teleport* to a random node if he is stuck in a loop or a dead end
- Process repeats indefinitely



Here we present the basic idea behind the **random walk** algorithm, a popular algorithm widely used in SNA for tasks such as graph exploration. This algorithm is used by PageRank.[1]

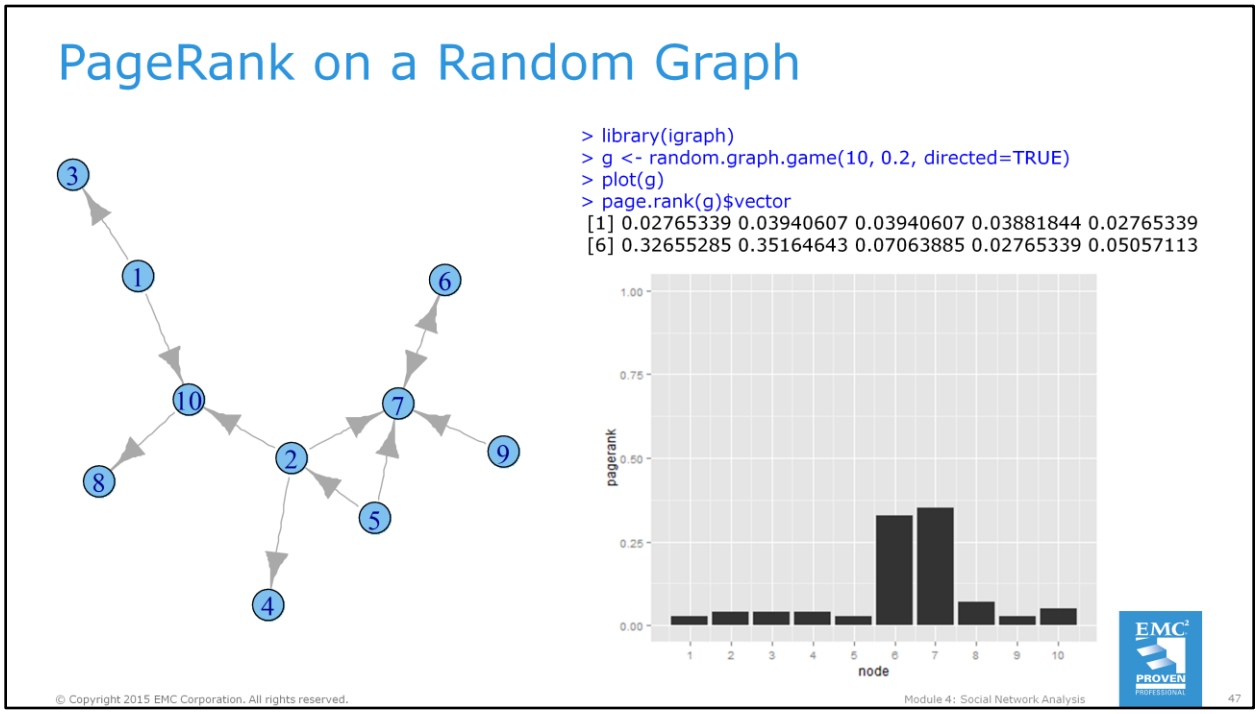
Imagine we ask a random walker, to explore a network for a very long time. He follows the directions of the edges, but at any node, he can choose an outlink at random. If he is stuck in a cycle or trapped in a dead end for a while, we allow him to teleport to a random node in the graph. The probability he visits a node can be used to measure how important that node is in the network.

The term random walk was first introduced by Karl Pearson in 1905, and was later applied to various fields such as ecology, economics, psychology, computer science, physics, chemistry, and biology. In Brin and Page's work, they explained how PageRank uses random walk to rank each webpage. In PageRank, we can consider a random surfer who is given a webpage at random and keeps clicking on links, never hitting the back link, but eventually gets bored and starts on another random webpage. The probability that he visits a certain webpage is its PageRank and the *d* damping factor, discussed in the previous slide, corresponds to the probability that the random surfer gets bored and requests another random page, just like the random walker is teleported to a new node.[2]

Reference:

[1] Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, *30*(1), pp. 107-117. http://infolab.stanford.edu/~backrub/google.html

[2] Pearson, K. (1905). The problem of the random walk. *Nature*, 72/1865, pp. 294.



The slide shows how to compute PageRank over a random network in R using the igraph library. The second line of the code generates a directed Erdős-Rényi random graph following the G(n, p) model where n=10 and p=0.2. This means the random graph will contain 10 nodes and an edge is drawn between two arbitrary nodes with a 0.2 probability.

Function page.rank(g) computes the PageRank of each node in the graph. We can see that node 7 has the highest PageRank therefore is considered the most important node. This makes sense because node 7 has the highest in-degrees and it also has links from other importance nodes, such as node 6.

The following code generates the bar chart shown on the slide:

```
library(ggplot2)
node <- seq(length(V(g)))
pagerank <- page.rank(g)$vector
data <- data.frame(node, pagerank)
ggplot(data, aes(node, pagerank)) +
  geom_bar(stat = "identity") +
  scale_x_continuous(breaks=node) +
  scale_y_continuous(limits=c(0, 1))
```

Check Your Knowledge

- How is centrality measured with in-degree and out-degree?
- How is the closeness centrality defined?
- What does it mean to have a high degree but low betweenness?
- What is the basic idea of eigenvector centrality and PageRank?
- How is Random Walk used in PageRank?

Write your answers here.

Lesson 2: Summary

During this lesson the following topics were covered:

- Degree Centrality
- Closeness Centrality
- Betweenness Centrality
- Eigenvector Centrality and PageRank

Lesson 3: Communities and Small World

This lesson covers the following topics:

- Modularity
- Power Law
- Strong and Weak Ties
- Density
- Clustering Coefficient
- Small World Phenomenon

Communities

- We have seen how nodes are connected
- Random graphs are useful
- However, in real world
  - Networks often display a certain structure which deviate themselves from random connections
  - Example: social network of friendships between 34 members of a karate club at a university
- We need ways of examining the structure of the whole population

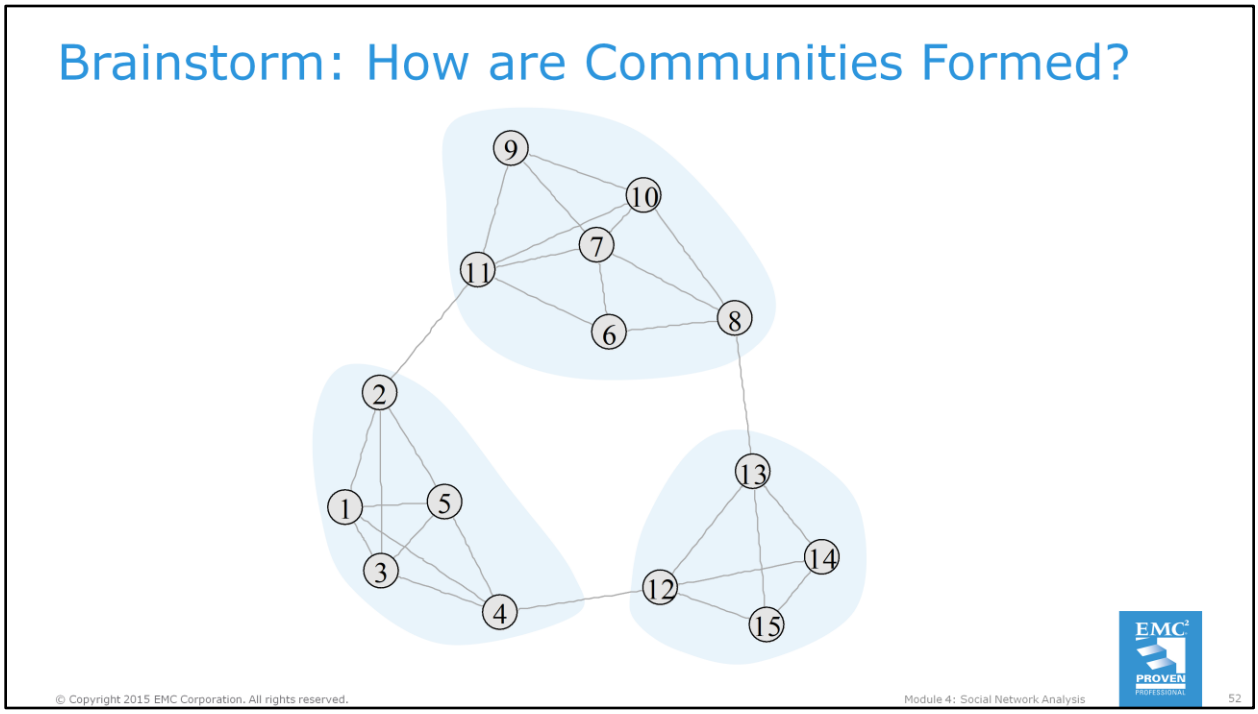
So far we have only focused on the ways that individuals are connected in a social network. Random networks, such as the Erdős-Rényi random graph model, are helpful for studying the connections among the nodes and understanding notions such as the percolation threshold.

In the real world, however, networks often display a certain structure which deviate themselves from random connections. For example, the graph on the slide shows the social network of friendships among 34 people in a university karate club studied by the anthropologist Wayne Zachary in the 1970s. A bigger node corresponds to a larger degree. The two largest nodes are central to two separate groups (red and blue) and these two nodes are not friends with each other. It turns out that the two nodes correspond to the instructor and the student founder of the club. Due to their conflicts, they have separated the karate club into two rival clubs.[1]

Therefore, we need additional ways of examining the structuring of the population rather than individuals. Discovering communities can help us measure the isolation of groups and understand opinion dynamics or idea adoption. The following slides will adopt a more macro perspective that focuses on the social structures within which individual actors are embedded.

Reference:

[1] Wayne Zachary (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4), pp. 452—473.

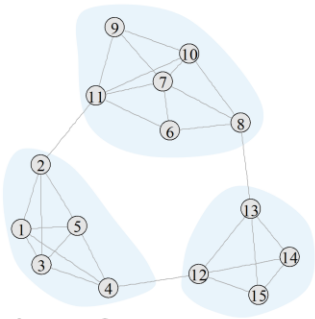


The graph shown on the slide has three communities:

- 1,2,3,4,5
- 6,7,8,9,10,11
- 12,13,14,15

Let's take a moment and think of the possible ways that a community can form.

How are Communities Formed?



- Mutuality of ties
  - Everybody in the group knows everybody else
  - They form a complete graph
- Frequency of ties among members
  - Everybody in the group has links to a decent number of others in the group
- Closeness or reachability of subgroup members
  - Individuals are separated by at most *n* hops
- Relative frequency of ties among subgroup members compared to nonmembers

There are several possible ways that a community may form. First, a community can form when people share mutual ties. That means, the members form a complete graph in which everyone knows everyone else within the community.

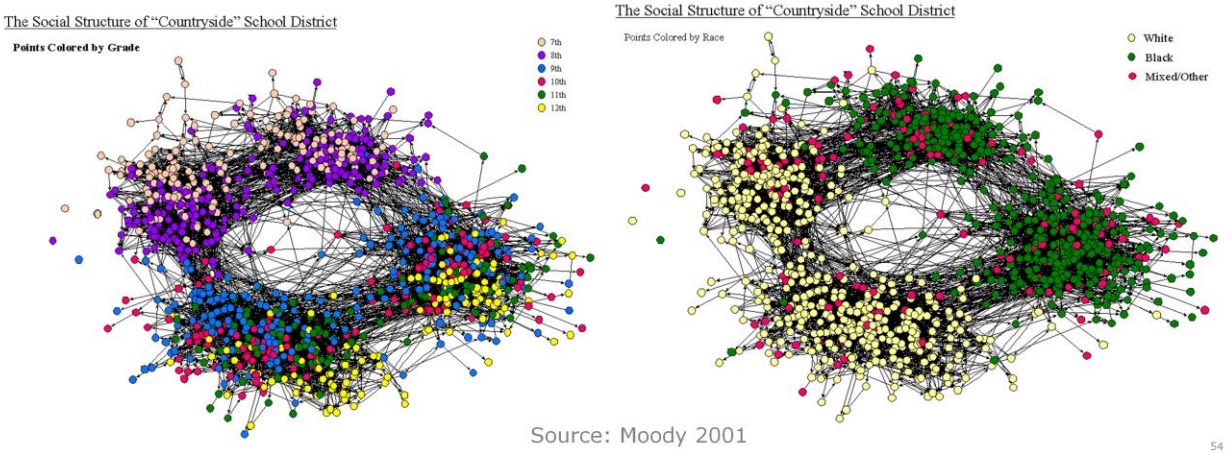
Second, the members may not form a complete graph, but each of these members talk to a decent number of other members within the community.

Third, some members may not talk to a decent number of other members, but it won't take many edges to travel between any two members within the community. That is, individuals are separated by at most n hops, where n is relatively a small number.

Fourth, the members of a community communicate to each other more often compared to nonmembers.

Homophily: The principle that we are similar to our friends

- A basic notion governing the structure of social networks
  - Your friends are more similar to you than a random collection of individuals



Homophily is one of the most basic notions governing the structure of social networks. It is the principle that we tend to be similar to our friends in terms of age, race, interests, opinions, etc. than a random collection of individuals.

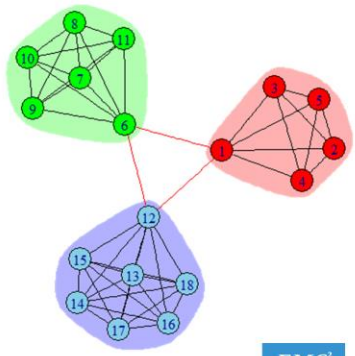
Therefore, homophily can divide a social network into densely connected, homogeneous communities that are weakly connected to each other. The two graphs by Moody show how the social network of a town's middle school and high school is separated into different communities based on grade, on the left, and race, on the right.

Reference:

James Moody (November 2001). Race, school pp. 679—716.

Modularity

- Modularity is a property that tells if a division of a network into communities is optimal
- An edge either falls within a community or is between a community and the rest of the network
- Definition: $Q = \frac{1}{2m}\sum_{vw}\left[A_{vw} - \frac{k_v k_w}{2m}\right]\delta(c_v, c_w)$
  - A_{vw}: element from the adjacency matrix
  - k_v, k_w: degree of node v and w
  - m: total number of edges in the graph
  - c_v, c_w: community that v or w belong to
  - $\delta(c_v, c_w)$: 1 if $c_v = c_w$ and 0 otherwise



Source: Clauset et al., 2004

By using a set of heuristics and strong and weak ties as seen earlier, a network can be divided into communities; however, those methods are somewhat arbitrary. How can we tell if a division is optimal?

Modularity is a property of a network that divides itself into communities. It measures when the division is a good one, in the sense that there are many edges within each community and only a few edges between any two communities.

The definition of modularity is given in the slide. It includes the following variables:

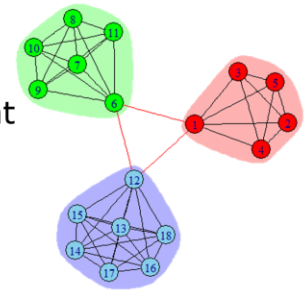
- v and w: two nodes in the network
- A_{vw}: element from the adjacency matrix that $A_{vw} = 1$ if v and w are connected and 0 if v and w are not connected.
- k_v, k_w: degrees of v and w
- m: total number of edges in the network
- c_v, c_w: community that v or w belong to
- $\delta(c_v, c_w) = 1$ when $c_v = c_w$, otherwise $\delta(c_v, c_w) = 0$.

Reference:

Clauset, A., Newman, M. E., & Moore, C. (2004). Finding community structure in very large networks. *Physical review E*, 70(6), 066111. Available: http://arxiv.org/pdf/cond-mat/0408187.pdf

How to Use Modularity to Divide Network into Communities

- Algorithm:
  - Start with all nodes as isolates
  - Follow a greedy strategy:
    - Successively join clusters with the greatest increase in modularity
    - Stop when the modularity does not increase anymore from joining any two clusters
- Scalable to very large graphs
  - Amazon's people who bought this also bought that
  - 400,000+ nodes and 2+ million edges
- Implemented in most SNA tools



The paper by Clauset et al. explains how modularity can be used to divide a network into communities. First, start by treating all the nodes in the network as isolates. Then, follow a greedy strategy by successively joining clusters with the greatest increase in modularity. The iteration stops when the modularity does not continue to increase from joining any two clusters.

Modularity is scalable to very large graphs. In their paper, Clauset et al. applied the algorithm to a recommender network of books from the online retailer Amazon.com, which is a very large graph with over 400,000 nodes and over 2 million edges.

Modularity is implemented in most SNA tools, such as Gephi, iGraph and NetworkX. The following code uses iGraph to plot a network and divides it into three communities. The resulting graph is shown on the slide.

```
library (igraph)
g <- graph.full(5) %du% graph.full(6) %du% graph.full(7)
g <- add.edges(g, c(1,6, 1,12, 6, 12))
wc <- walktrap.community(g)
modularity(wc)
membership(wc)
plot(wc, g, colbar=c("red", "green", "skyblue"), mark.border=NA)
```

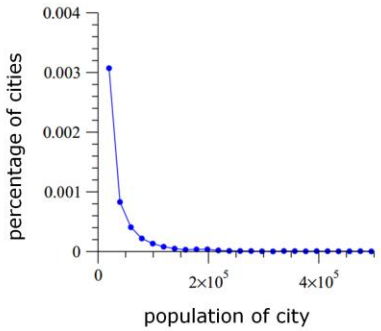
Note that the `walktrap.community()` function uses the random walk algorithm to find densely connected subgraphs, because short random walks tend to stay within the same community.

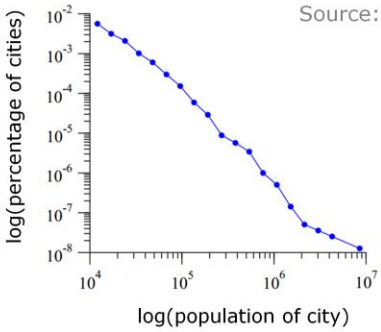
Reference:

Clauset, A., Newman, M. E., & Moore, C. (2004). Finding community structure in very large networks. *Physical review E*, 70(6), 066111. Available: http://arxiv.org/pdf/cond-mat/0408187.pdf

Power Law Distribution

- Small occurrences are extremely common, whereas large instances are extremely rare
  - Distribution follows $P(x) = cx^{-\alpha}$ for some small constant α





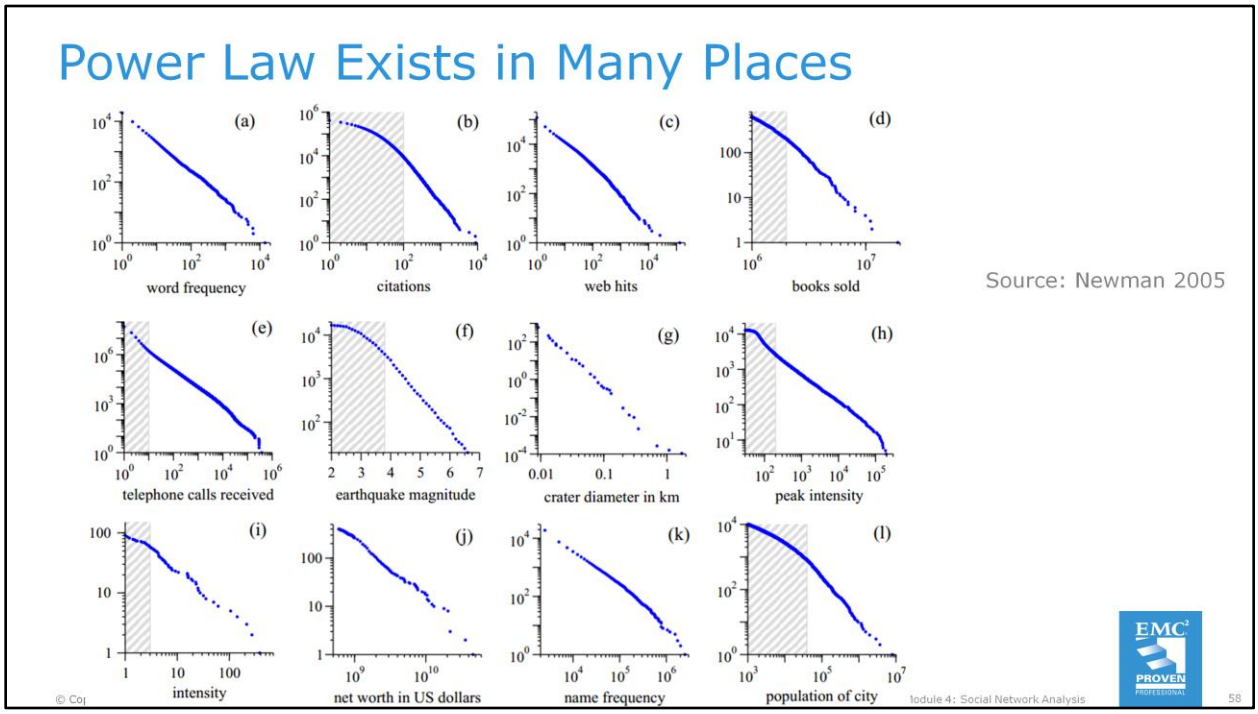
Source: Newman 2005

- Zipf's law is a special case of the Power Law

In the NLP module, we discussed Zipf's law, which models the distribution of terms in a text collection. Zipf's law is actually a special case of the Power law. When the probability of measuring a value of some quantity varies inversely as a power of that value, the quantity is said to follow a power law. A power law distribution has a long tail -- small occurrences are extremely common, whereas large instances are extremely rare.

Reference:

Newman, M. E. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary physics*, 46(5), 323-351. Available: http://arxiv.org/pdf/cond-mat/0412004v3.pdf



In Newman's work, he has shown that the power law appears widely in various fields such as physics, biology, earth and planetary sciences, economics and finance, computer science, demography and social sciences. Shown on the slide are the cumulative distributions or rank/frequency plots of twelve quantities which appear to follow the power law.

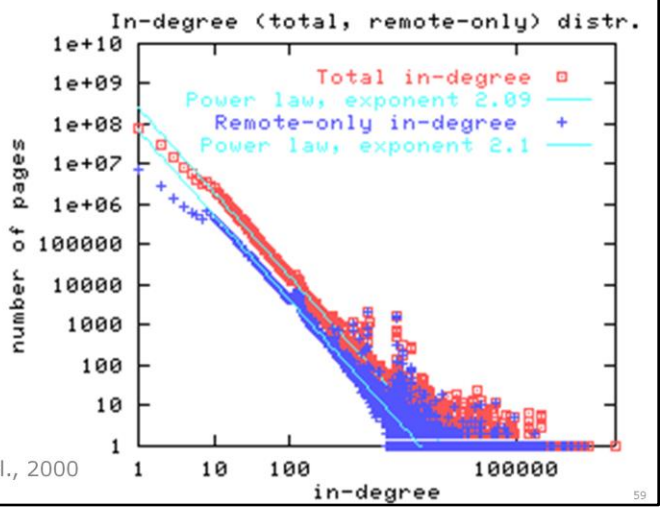
These twelve quantities are: (a) numbers of occurrences of words in the novel *Moby Dick* by Hermann Melville, (b) numbers of citations to scientific papers published 1981—1997, (c) numbers of hits on web sites by 60,000 users of the America Online Internet service on December 1st, 1997, (d) numbers of copies of bestselling books sold in the US between 1895 and 1965, (e) number of calls received by AT&T telephone customers in the US for a single day, (f) magnitude of earthquakes in California between January 1910 and May 1992, (g) diameter of craters on the moon, (h) peak gamma-ray intensity of solar flares in counts per second, measured from Earth orbit between February 1980 and November 1989, (i) intensity of wars from 1816 to 1980, measured as battle deaths per 10,000 of the population of the participating countries, (j) aggregate net worth in dollars of the richest individuals in the US in October 2003, (k) frequency of occurrence of family names in the US in the year 1990, and (l) populations of US cities in the year 2000.

Reference:

Newman, M. E. (2005). Power laws, Pareto distributions and Zipf's law. Contemporary physics, 46(5), pp. 323-351. Available: http://arxiv.org/pdf/cond-mat/0412004v3.pdf

Scale-Free Network

- Network whose degree distribution follows a power law
  - Fraction of nodes in the network having *k*-degree: $P(k) \sim k^{-\alpha}$
- What does it mean to be scale-free?
  - The power law looks the same no matter what scale we look at it on (1 to 100 or 1000 to 100,000)
  - Shape of the distribution does not change



Source: Broder et al., 2000

59

A scale-free network is a network whose degree distribution follows a power law. That means, the fraction P(k) of nodes in the network having k-degree is $P(k) \sim k^{-\alpha}$. It's called scale-free because both the power law and the shape of the distribution look the same regardless of the scale of the network.

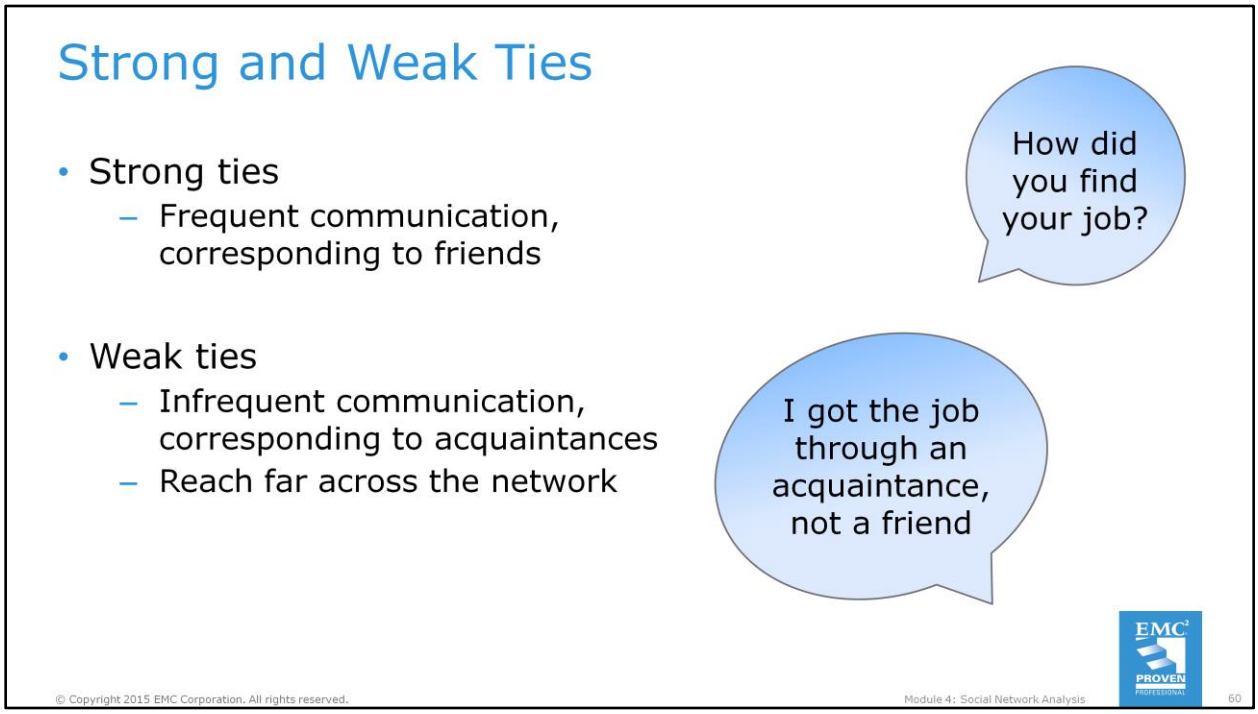
It turns out, many real-world networks are scale-free:

- The internet, if a node is a computer/router and an edge is a connection between them
- The WWW, if a node is a web page and an edge is a hyperlink to (or from) that page
- Wikipedia, if a node is an article and an edge is when that article references another
- Peer-to-peer network, where a node is a peer and an edge is an established connection between two peers
- Protein-protein interactions, if a node is a protein and an edge represents the interaction between two proteins
- Metabolic network, where nodes are Enzymes and metabolites and edges are their interactions

The graph shows that the number of webpage in-links in a log-log plot follows a power law distribution.

Reference:

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A. & Wiener, J. (2000). Graph structure in the web. *Computer networks*, 33(1), pp. 309-320.
http://www.cis.upenn.edu/~mkearns/teaching/NetworkedLife/broder.pdf



Mark S. Granovetter proposed the notion of strong and weak ties in his paper, "The strength of weak ties," written in 1973. It is by far considered as one of the most influential sociology papers ever written and also one of the most cited.

As part of his Ph.D. research, Mark S. Granovetter interviewed people who had recently changed employers to learn how they discovered their new jobs. He found that many people learned information leading to their current jobs through acquaintances rather than close friends.

In a social network where the nodes are the people and the edges specify the social relationship among the people, the weight of the edge can be defined as the strength of the connection, quantified to either weak or strong.

Strength is vaguely defined as a combination of the amount of time, the emotional intensity, the intimacy or mutual confiding, and the reciprocal services which characterize the tie.

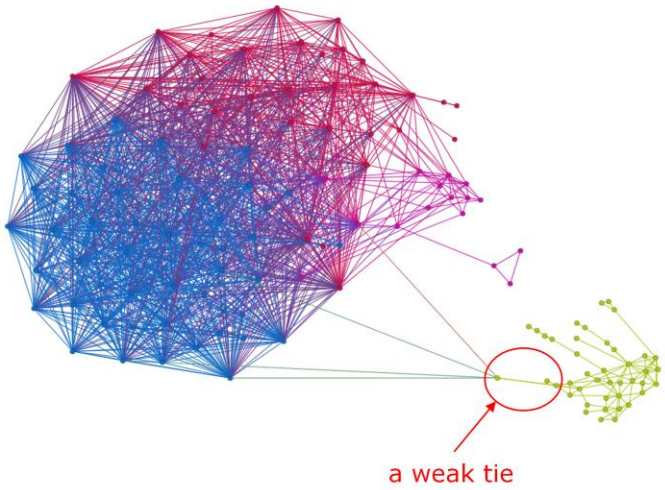
The strong links are called strong ties and they correspond to friends. Nodes that form strong ties tend to communicate frequently but the ties are redundant due to the high clustering. The weak links are called weak ties and they correspond to acquaintances. Nodes that form weak ties do not communicate frequently but these weak ties can reach far across the network.

Reference:

Granovetter, M. S. (1973). The strength of weak ties. *American journal of sociology*, pp. 1360-1380. http://www.jstor.org/stable/2776392

An Example of Weak Tie

- Weak ties are critical to spread a new idea across the network

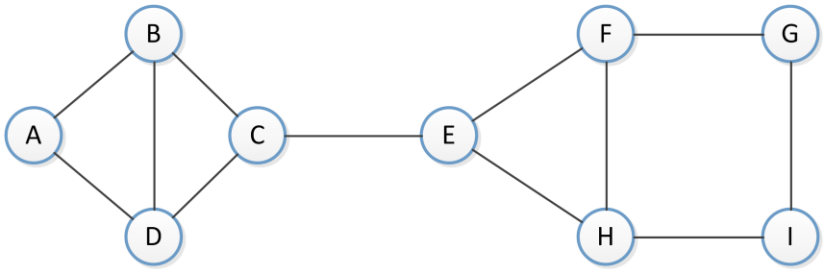


In a social network, strong ties tend to form small communities. These communities are connected through weak links. Removing these weak links will greatly increase the cost of cross-community communications. In fact, removal of weak ties increases path lengths more than the removal of strong ties. In real systems, **people with many weak ties are critical to spread a new idea across the network**.

A weak tie is annotated in the giant component we have seen previously. Removal of this weak tie will immediately break the connection between group red-blue and group green.

Measuring the Density of a Cluster

- A cluster of density p is a set of nodes such that each node in the set has at least a p fraction of its network neighbors in the set
- For example:
  - Density (A, B, C, D) = 2/3
  - Density (E, F, G, H, I) = 2/3



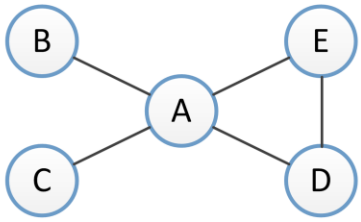
Next, let's explore the notion of density. A graph can be broken into one or more clusters. Each cluster of density p is a set of nodes such that each node in the set has at least a p fraction of its network neighbors in the set.

In the example graph, cluster (A B C D) contains four nodes. Node A, B and D have all their neighbors in the set. But node C only has 2 out of its 3 neighbors in this set. (Node E is a neighbor of C that is outside the cluster.) Therefore this cluster has a density 2/3.

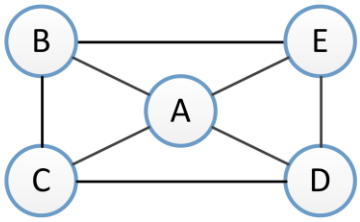
Similarly, cluster (E F G H I) also has a density 2/3 because node E only has 2 out of its 3 neighbors in the set.

Clustering Coefficient (or Transitivity)

- What portion of your neighbors are connected?
- The clustering coefficient of node A is defined as the probability that two randomly selected friends of A are friends with each other
- Every node in a complete graph has a clustering coefficient of 1
- Most SNA tools can report the clustering coefficient



(i) Before new edges form, the clustering coefficient of A is 1/6



(ii) After new edges form, the clustering coefficient of A becomes 2/3

The clustering coefficient, or transitivity, of a node measures the portion of its neighbors that are connected. It is defined as the probability that two randomly selected friends of a node are friends with each other. Therefore in a complete graph where any two nodes are connected, the clustering coefficient of every node is 1. In other words, **a more connected network will have nodes of higher clustering coefficients**.

Let's take a look at graph (i), where node A has four neighbors: B, C, D, and E. The four neighbors can form (4*3)/2=6 pairs. Before new edges form, only D and E are friends of all the neighbors. Therefore the clustering coefficient of node A is 1/6.

After new edges have formed as shown in graph (ii), 4 of the 6 pairs of A's neighbors are connected: BC, CD, DE, and BE. Therefore the clustering coefficient of node A is 4/6=2/3.

Most SNA tools support the clustering coefficient. For example, the `transitivity()` function in iGraph and the `clustering()` function in NetworkX can report the clustering coefficient of a node. Gephi reports the average clustering coefficient of the entire graph and it can visualize the clustering coefficient of each node.

The following code shows how to use NetworkX in Python to compute the clustering coefficient of a complete graph:

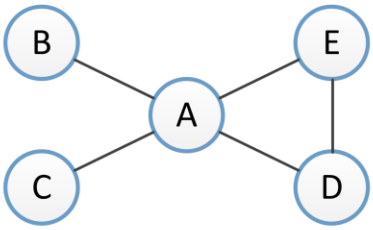
```
>>> import networkx as nx
>>> G = nx.complete_graph(5)
>>> print(nx.clustering(G))
{0: 1.0, 1: 1.0, 2: 1.0, 3: 1.0, 4: 1.0}
```

References:

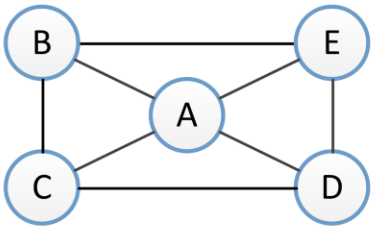
- iGraph (http://igraph.org/)
- NetworkX (http://networkx.github.io/)
- Gephi (http://gephi.github.io/)

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
- Basic idea behind friend and product recommendations



(i) Before new edges form



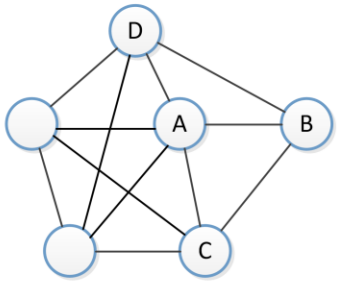
(ii) After new edges form

It is often useful to consider how a social network evolves over the time. In particular, how do nodes arrive and depart and how do edges form and vanish? The triadic closure principle specifies that, 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 A is a mutual friend of B and C, then the formation of an edge between B and C produces a scenario in which all three nodes A, B, and C have edges connecting each other – a triangle in the network. If we take snapshots of a social network over time, it is common to see the formation of such triangles, as illustrated in the slide. Triadic closure forms the basic idea behind the "People You May Know" of online social networks such as LinkedIn and Facebook, and product recommendation of online retailers such as Amazon.

.

Edge Embeddedness and Neighborhood Overlap

- Embeddedness: number of common neighbors the two endpoints have
- For example, embeddedness(A—B)=2
  - A and B have the two common neighbors C and D



- Neighborhood overlap =

$$\frac{\text{number of nodes who are neighbors of both A and B (embeddedness)}}{\text{number of nodes who are neighbors of at least one of A or B}}$$

- A and B has a neighborhood overlap of ½
- If an edge has an embeddedness of 0, hence a neighborhood overlap of 0, it's very likely to be a weak tie

We define the **embeddedness** of an edge in a network to be the number of common neighbors the two endpoints have. For example, the A—B edge has an embeddedness of 2, since A and B have the two common neighbors C and D.

Embeddedness is the numerator of neighborhood overlap. The **neighborhood overlap** of A and B is defined to be the number of nodes who are neighbors of both A and B divided by the number of nodes who are neighbors of at least one of A and B. In the given graph, A and B have two common neighbors C and D, and A has two additional neighbors. Therefore A and B have a neighborhood overlap of ½.

If both the embeddedness and neighborhood overlap of edge A—B are zero, that is, A and B don't have any common neighbors, then A—B is very likely to be a weak tie.

Now think about these questions: What does it mean for an edge to have a high embeddedness? What could be the advantages? How about disadvantages?

Small-World: An experiment by Milgram

- Conducted by Yale University psychologist Stanley Milgram in 1960s

In the 1960s, Yale University psychologist Stanley Milgram conducted several experiments, which leads to the notion of Small World – a principle that most people in a society are linked by short chains of acquaintances. Recall that we've seen the Erdős number and the Bacon number at the beginning of this module. They are both examples of the small world phenomenon.

In Milgram's experiment, he put an advertisement in the newspaper to recruit people in Nebraska. The volunteers were given a letter and a contact in Boston, Massachusetts. Instead of mailing the letter to the contact in Massachusetts directly, these individuals were asked to forward the letter to someone they knew on the first-name basis in order to transmit the letter to the destination as fast as possible.

Reference:

Milgram, S. (1967). The small world problem. *Psychology today*, 2(1), pp. 60-67.

Small-World: An experiment by Milgram (cont.)

- A person P in Nebraska was given a letter to deliver to another person Q in Massachusetts. P was told about Q's address and occupation and instructed to send the letter to someone she knew on a first-name basis, in order to transmit the letter to the destination *as fast as possible*.
- Outcome:
  - 22% of the letters reached target
  - Average path length: 5.5 or 6

Six degrees of separation

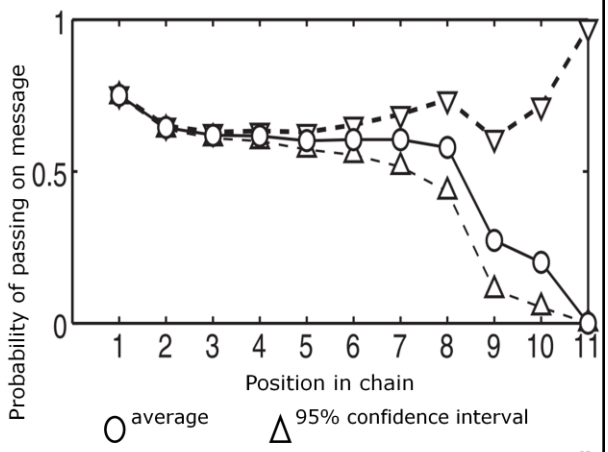
Shortly after the start of the experiment, the target contact in Massachusetts started to receive letters. Some letters would arrive to the target in as few as one or two hops, while some chains were composed of as many as nine or ten hops. Although many people didn't pass the letter forward, 64 of the 296 letters eventually reached the target contact, about a 22% success rate. Among these chains, the average path length was around 5.5 or 6. These findings have led to the later widespread acceptance of the term "six degrees of separation."

Reference:

Milgram, S. (1967). The small world problem. *Psychology today,* 2(1), pp. 60-67.

A Recent Experiment Similar to Milgram's

- First large-scale replication of Milgram's experiment
  - 18 targets from 13 countries
  - 60,000+ email users
  - 24,163 e-mail chains
  - 384 chains reached the targets
  - Average path length: 4
- Are longer or shorter email chains more likely to complete?

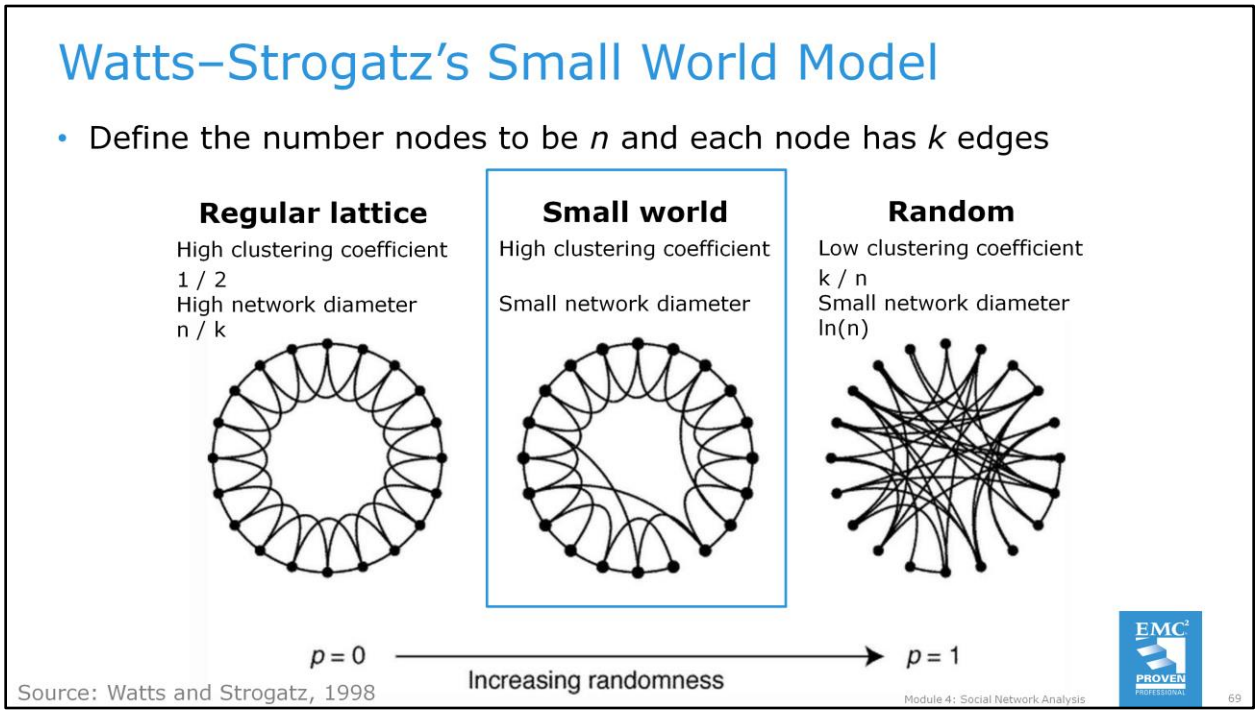


In 2003, Peter Dodds, Roby Muhamad, and Duncan Watts conducted the first large-scale replication of Milgram's experiment. Their experiment included 61,168 e-mail users whom attempted to reach one of 18 target persons in 13 countries by forwarding messages to acquaintances. These users constituted 24,163 distinct e-mail chains, 384 of which eventually reached the targets, with an average path length of 4.

Are longer or shorter email chains more likely to reach the targets? In Dodds et al.'s work, they reported that the attrition rate was pretty steady regardless of the position in the email chains.

Reference:

Dodds, P. S., Muhamad, R., & Watts, D. J. (2003). An experimental study of search in global social networks. science, 301/5634, pp. 827-829.



Let's look at how to model small world networks. As presented in the work by Watts and Strogatz, small world networks can be seen as a middle ground between the regular graph and the random graph.

The regular graphs refer to lattices that are characterized by high clustering coefficients and high network diameters. The left graph shows a ring lattice with 20 nodes. Each node has 4 neighbors. At most (4*3)/2=6 edges can exist between these 4 neighbors but there are only 3 edges. Therefore this ring lattice has a clustering coefficients of ½. Its network diameter is 20/4=5.

Random graph on the other hand is characterized by low clustering coefficients and small network diameters. Such a graph has a clustering coefficient of k/n and a diameter of $\ln(n)$.

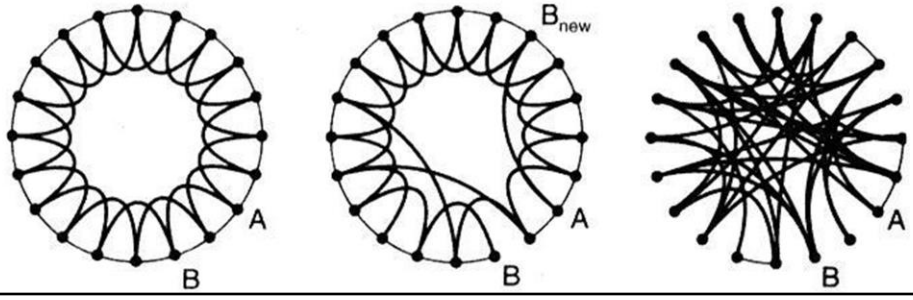
Watts and Strogatz define small world networks as graphs with high clustering coefficients (like regular graphs) but low network diameters (like random graphs). Based on this definition, they proposed a method to generate a random small world network by reducing its diameter while maintaining the high clustering coefficient.

Reference:

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393/6684, pp. 440-442.

Watts–Strogatz's Small World Model (Cont.)

- Start with a regular lattice
- Progressively rewire the edges to become a random graph
- The network diameter gets reduced more if the edge is rewired to a node that is farther away
- Require $n \gg k \gg \ln(n) \gg 1$ where $k \gg \ln n$ ensures a connected graph



Watts and Strogatz's small world model works as follows: We start with a regular lattice like the one on the left. Next, we progressively rewire the graph, one edge after another, until we obtain a random graph, like the one on the right. To rewire an edge, we disconnect it from its original end node and rewire it to a random node in the graph. During this process, the network diameter gets reduced more if the edge is rewired to a node that is farther away. After only a few steps, the diameter can be reduced dramatically while the clustering coefficient remains high.

Recall that in the $G(n,p)$ representation of the Erdős-Rényi random graph model, we discussed that the threshold of the connectivity of $G(n, p)$ depends on $\frac{\ln(n)}{n}$ and the graph is guaranteed to be connected if $p \gg \frac{\ln(n)}{n}$. Therefore, the degree k of a node in a random graph should yield $k \gg \ln n$ to ensure the graph to be connected. Watts and Strogatz's model further requires $n \gg k \gg \ln(n) \gg 1$ for the small world network.

Most SNA tools provide functions to generate Watts-Strogatz small world networks. In `igraph` for example, you can use the following R code to generate a 20-node, 4-neighbor small world network with a 0.1 rewiring probability. The rewiring probability can be set to any value between 0 and 1 where 0 produces a regular lattice and 1 produces a random graph.

```
library(igraph)
g <- watts.strogatz.game(dim=1, size=20, nei=4, p=0.1)
plot(g, layout=layout.circle)
```

Next, you can use the following code to compute the clustering coefficient (transitivity) and the average path length of the network.

```
transitivity(g, type="average")
average.path.length(g)
```

Try changing the rewiring probability *p* and see how that affects the clustering coefficient and the average path length.

Check Your Knowledge

- What is modularity?
- What are a few examples of Power Law?
- Why are weak ties useful?
- How is clustering coefficient defined?
- What is triadic closure?
- What is edge embeddedness?
- What does six degrees of separation mean?

Write your answers here.

Lesson 3: Summary

During this lesson the following topics were covered:

- Modularity
- Power Law
- Strong and Weak Ties
- Density
- Clustering Coefficient
- Small World Phenomenon

Lab Exercise 10: Visualizing Social Network Data

- This lab introduces you to social network analysis using Gephi.

- After completing the tasks in this lab you should be able to:
  - Import datasets into Gephi
  - Apply a layout to the network, identify communities and compute metrics
  - Filter, preview and export the graph

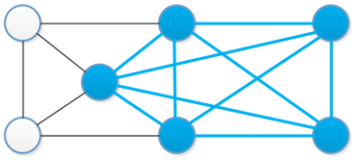
Lesson 4: Network Problems and SNA Tools

This lesson covers the following topics:

- The Clique Problem
- The Traveling Salesman Problem
- SNA tools

The Clique Problem

- Recall that a k-clique is a complete subgraph with k nodes
- The clique problem is to find:
  - CLIQUE = $\{\langle G, k\rangle$: G is a graph with a clique of size $k\}$
- Naïve method
  - List all the k subsets of nodes V in G
  - Check each one to see if it forms a clique
- Time complexity $> ck^2\binom{n}{k}$
  - As $n \to \infty$, the number of cliques can become substantially large
  - Cannot be solved in polynomial time



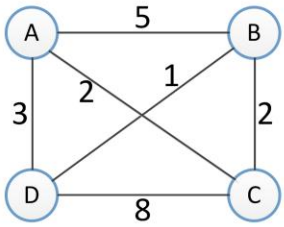
A 5-clique in graph G=(V, E) where n=|V|=7

Recall that earlier in the module, we discussed that a k-clique is a complete subgraph with k nodes. The clique problem is the optimization problem of finding all cliques of a given size k that exists in a graph.

A naïve way of tackling the problem is to list all the k subsets of nodes V in the graph G=(V, E), and check each one to see if it forms a clique. The time complexity of such an algorithm is at least $ck^2\binom{|V|}{k}$ for some constant c. Since k can become substantially large as the number of nodes $|V| \to \infty$, the algorithm runs in super-polynomial time. It's quite unlikely to find an efficient algorithm for the clique problem.

Traveling Salesman Problem

- Given a weighted graph $G = (V, E)$ where $|V| = n$ and each node is a city
- A salesman wants to visit n cities
  - Visit each city exactly once
  - Finish at the city where he starts from
  - Also minimize the tour cost
- For example, a minimum-cost tour is
  - A → C → B → D → A
  - Cost: 8
- Number of possible tours: n!
- Appear in many disciplines
  - Networks
  - Manufacturing
  - Plane routing, etc.



The traveling salesman problem is another famous NP-complete problem. Assume we have a weighted graph $G = (V, E)$ to represent a map. The graph contains *n* nodes and each node is a city. Each edge in E represents a route between two cities. A salesman wants to visit all the *n* cities. But he can only visit each city exactly once and must finish at the city where he started from. The trip between any two cities has a cost (the weight of the edge) and the salesman wants to minimize the cost of his entire tour.

Take the 4-node graph in the slide for example, a minimum-cost tour could be A → D → B → C → A and the cost is 8.

There is no polynomial time known solution for this problem. If we use an exhaustive search, or brute-force strategy, we will have *n*! number of possible trips. The number of tours grows incredibly quick as we add more cities to the map.

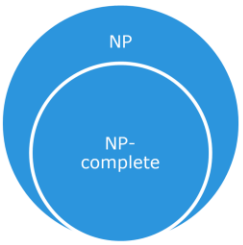
The traveling salesman problem can appear in many disciplines beyond networks and traveling salespeople, such as manufacturing, plane routing, telephone routing, and structure of crystals. For example, consider in manufacturing, a robot arm is assigned to solder all the connections on a printed circuit board. The shortest trip that visits each solder point exactly once defines the most efficient path for the robot. If you run into a similar problem, you should know that it's impossible to solve such a problem in polynomial time. Therefore you need to reframe your problem and find alternative solutions.

References:

- Traveling salesman problem: http://en.wikipedia.org/wiki/Travelling_salesman_problem
- The traveling salesman problem: http://www.math.uwaterloo.ca/tsp/
- Comic on the traveling salesman problem (XKCD): http://xkcd.com/399/

Both Problems are Known as NP-Complete Problems

- The clique problem and the traveling salesman problem are two well-known NP-complete problems
- NP-complete is a subset of NP
  - Solution of an NP problem can be **verified** in polynomial time
  - But an NP problem **cannot be solved** in polynomial time

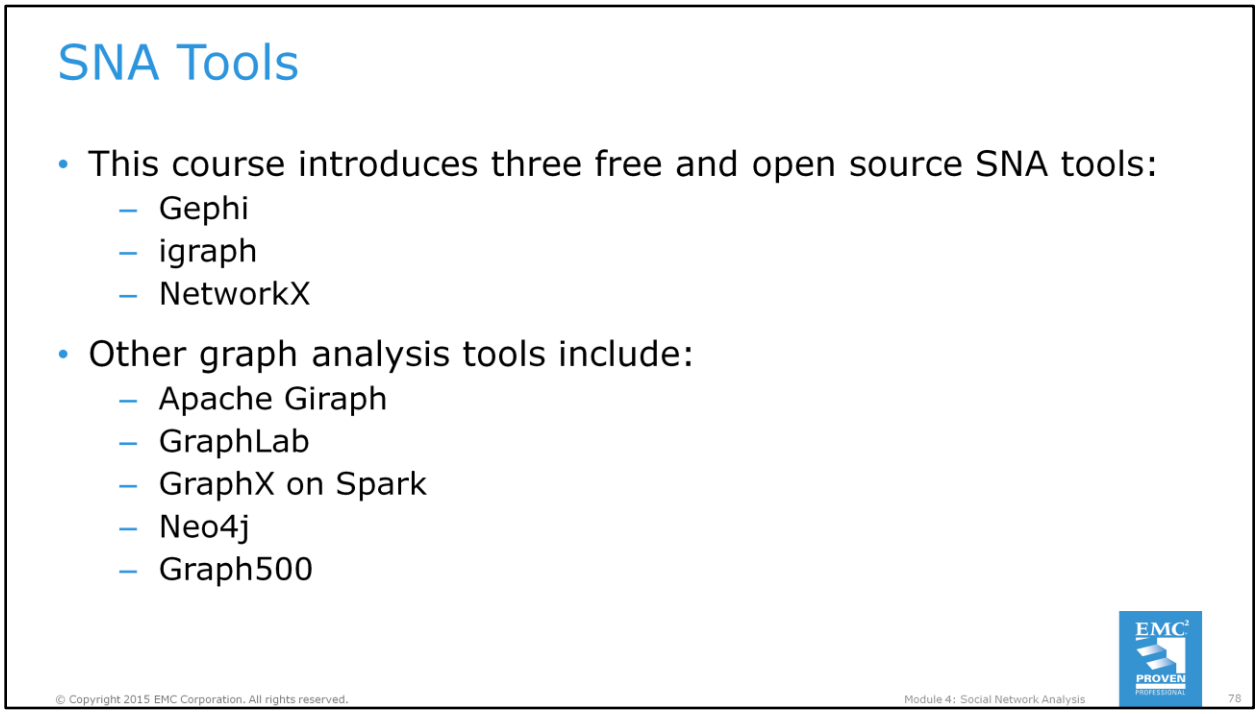


- Network related NP-complete problems
  - Clique problem
  - Traveling salesman problem
- Useful to know these problems
  - Understand what SNA problems are not practical to solve

The clique problem and the traveling salesman problem from the previous slides are two well-known NP-complete problems.

NP-complete problems are in NP, the set of problems whose solutions can be verified in polynomial time but the problems are believed to be unsolvable in polynomial time. NP stands for **nondeterministic polynomial time**. A problem x in NP is NP-complete if every problem in NP can be reducible into x in polynomial time. Despite decades of study, nobody has yet been able to determine conclusively whether NP-complete problems are in fact solvable in polynomial time.

Both the clique problem and the traveling salesman problem are related to networks or graphs. It's important to understand these NP-complete problems. So the next time you run into a similar problem when you conduct a social network analysis, you can quickly reach the conclusion that it may not be practical to solve such a problem and instead find ways around it.



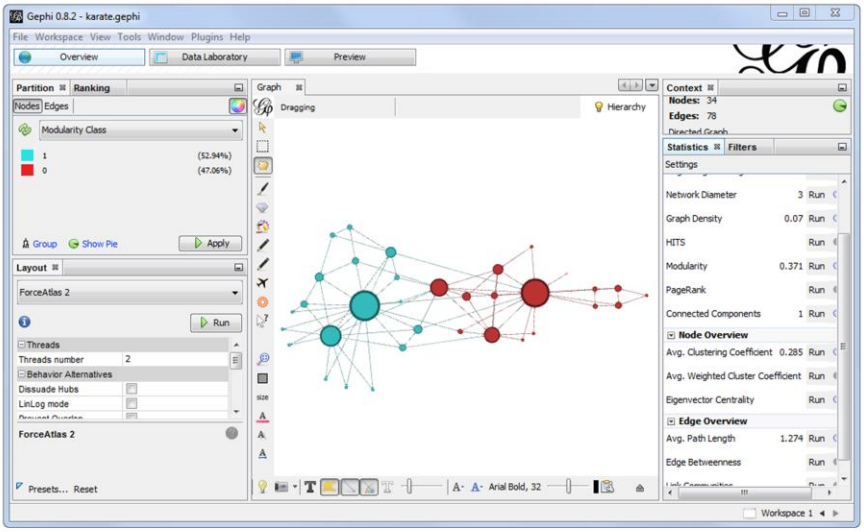
This module introduced three tools for social network analysis: Gephi, igraph and NetworkX. All of these three tools are free and open source and they all provide extensive features for network analysis.

Gephi is a GUI platform that enables point-and-click while igraph and NetworkX are two libraries to programmatically conduct network analysis.

Note that there are other graph analysis tools that also support social network analysis and graph analysis. A few of these tools include Apache Giraph for Hadoop (http://giraph.apache.org/), GraphLab (http://dato.com), GraphX on Spark (http://spark.apache.org/graphx/), Neo4j graph database (http://www.neo4j.org/), and Graph500 (http://www.graph500.org/). When you conduct social network analysis, you should evaluate the problem and choose the tool that is the most suitable.

Gephi

- An open-source GUI platform for graph visualization: http://gephi.github.io/



Gephi is an open-source graph visualization platform. It provides interactive visualization and exploration for various kinds of networks and complex systems, and dynamic and hierarchical graphs. Gephi support Windows, Linux and Mac OS. You can obtain a copy from: http://gephi.github.io/.

You can import your own network data into Gephi for network analysis, or you can download and try out the datasets from Gephi's wiki page: https://wiki.gephi.org/index.php?title=Datasets.

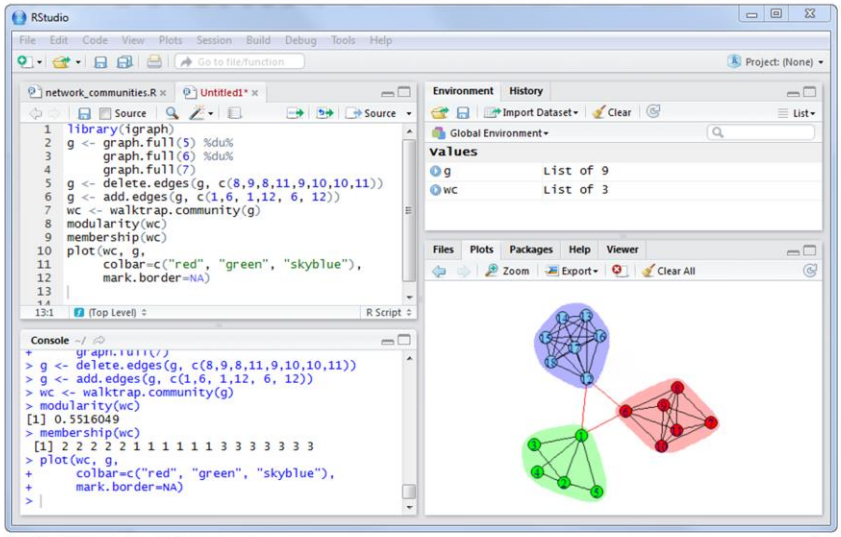
Gephi allows the easy manipulation and visualization of a graph. You can change the graph layout by choosing one of the built-in layout algorithms. The resulting visualization can be exported to images or PDF files.

Gephi includes many of the metrics discussed in this module, such as degree, path length, network diameter, density, modularity, PageRank, and clustering coefficient.

The screenshot shows an example of using Gephi to analyze Zachary's karate club dataset. The dataset can be obtained from Gephi's wiki page.

igraph

- Available in R, Python and C/C++ (http://igraph.org/)



igraph in RStudio

The next tool we will discuss is the igraph, a network analysis package available in R, Python and C/C++ (http://igraph.org/). For R, igraph is available on CRAN and therefore can be easily installed in R and loaded into the workspace by executing the following commands:

```
# Download and install the package from CRAN
install.packages("igraph")
# Load package
library(igraph)
```

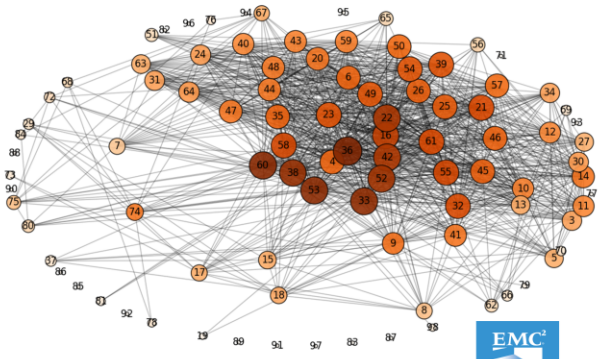
The screenshot shows an example of using igraph inside Rstudio to create a network, compute the modularity, and highlight the communities in the graph.

NetworkX

- A Python package for analysis of complex networks
- http://networkx.lanl.gov/
- Efficient, scalable, and portable

Shown is a visualization of 552,073 Hubway bicycle trips among 95 stations

- Node: A Hubway bicycle rental station
- Edge: One or more trips between two stations
- Node size: Degree
- Node color: PageRank



NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.[1]

NetworkX is suitable for operation on large real-world graphs, e.g., graphs in excess of 10 million nodes and 100 million edges. Since NetworkX is written in Python and uses Python's dictionary structure, it's efficient, scalable, and portable.

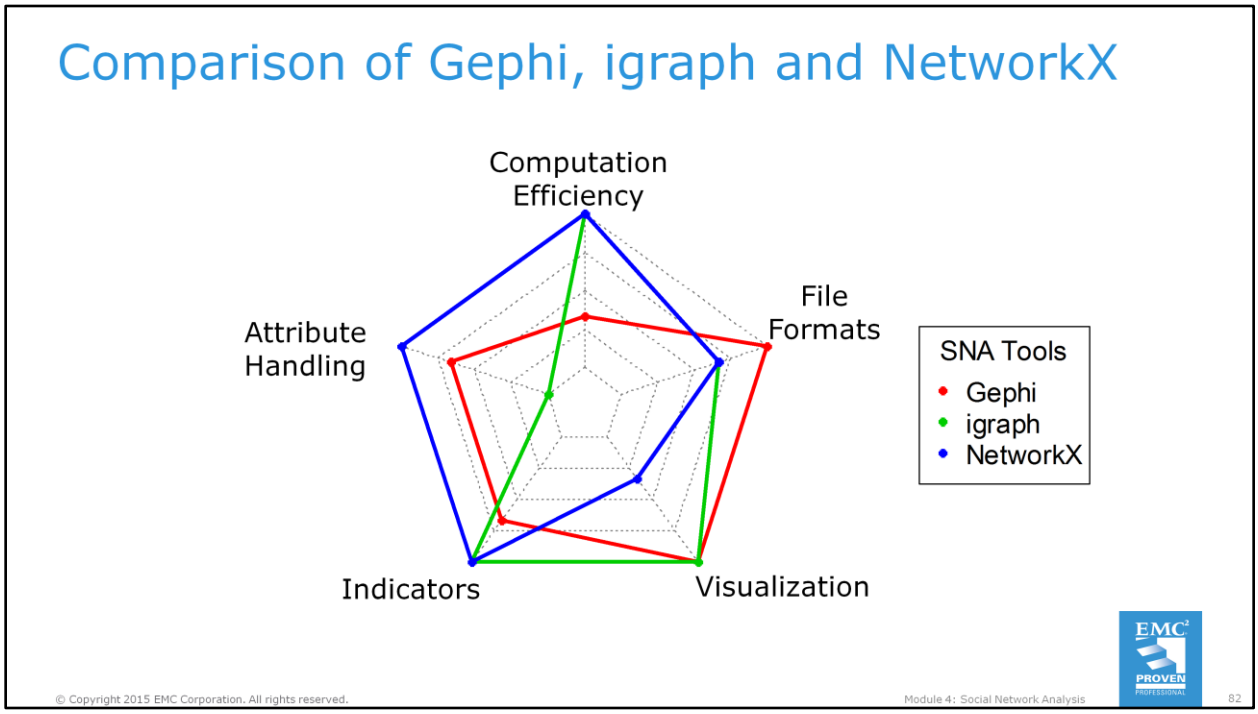
The screenshot shows an example that uses NetworkX to visualize the Hubway data. Hubway is a bicycle sharing system in the city of Boston, Massachusetts.[2] The screenshot visualizes the 95 Hubway stations and the 552,073 trips Hubway users took between July 28, 2011 and October 1st, 2012. Each node represents a Hubway station that is labelled with a unique ID. Each edge corresponds to one or more trips between the two stations. The size of a node corresponds to the number of trips originated or terminated at this station. A bigger node indicates a more popular station. A node with a darker color corresponds to a station with a higher PageRank in the network. The data can be obtained from the Hubway Data Challenge website .[3]

References:

[1] NetworkX: http://networkx.lanl.gov/

[2] Hubway: http://www.thehubway.com/

[3] Hubway Data Challenge: http://hubwaydatachallenge.org/



This lesson introduced three SNA tools: Gephi, igraph and NetworkX. Readers are advised to choose the most suitable tool according to their needs. A comparison of these popular SNA tools is visualized in the following 5 axes:

- Computation time: How fast each tool computes a graph
- Attribute handling: Attributes of nodes and edges supported by each tool
- Indicators: Properties supported by each tool, such as types of centrality and PageRank
- Visualization: Types of network layouts supported
- File formats: Input/output file formats supported

The igraph is available in both R and Python. If you used igraph under R, you may have to pre-filter your data separately in advance before you load the data into R. R holds all data in your active workspace in RAM. If you are running R on a 32-bit system, for example, you only have a 4 GB limit to the RAM R can access. Gephi experiences a similar problem. However, by using NetworkX in Python, or igraph in Python or C/C++, you can easily streamline the data ETL, analysis, and visualization.

Check Your Knowledge

- What is the Clique Problem?
- What is the Traveling Salesman Problem?
- Can we solve Traveling Salesman Problem in polynomial time?

Further reading:

Networks, crowds, and markets, by David Easley and Jon Kleinberg
http://www.cs.cornell.edu/home/kleinber/networks-book/

Network Science, by Albert-László Barabási http://barabasilab.neu.edu/networksciencebook/

Introduction to social network methods, by Robert A. Hanneman and Mark Riddle
http://faculty.ucr.edu/~hanneman/nettext/

Lesson 4: Summary

During this lesson the following topics were covered:

- The Clique Problem
- The Traveling Salesman Problem
- SNA tools

Lab Exercise 11: Analyzing the Hubway Data

- This lab introduces you to applying social network analysis to a real-world dataset, with tools such as Python, Gephi and NetworkX.

- After completing the tasks in this lab you should be able to:
  - Learn how to preprocess a dataset
  - Visualize the dataset in Gephi
  - Write Python codes to compute various metrics
  - Use NetworkX to visualize the dataset

Module 4: Summary

Key points covered in this module:

- Social Network Analysis using graph theory
- Characteristics of social network communities
- Tools available for Social Network Analysis