دانشگاه کردستان
University of Kurdistan
زانکۆی کوردستان

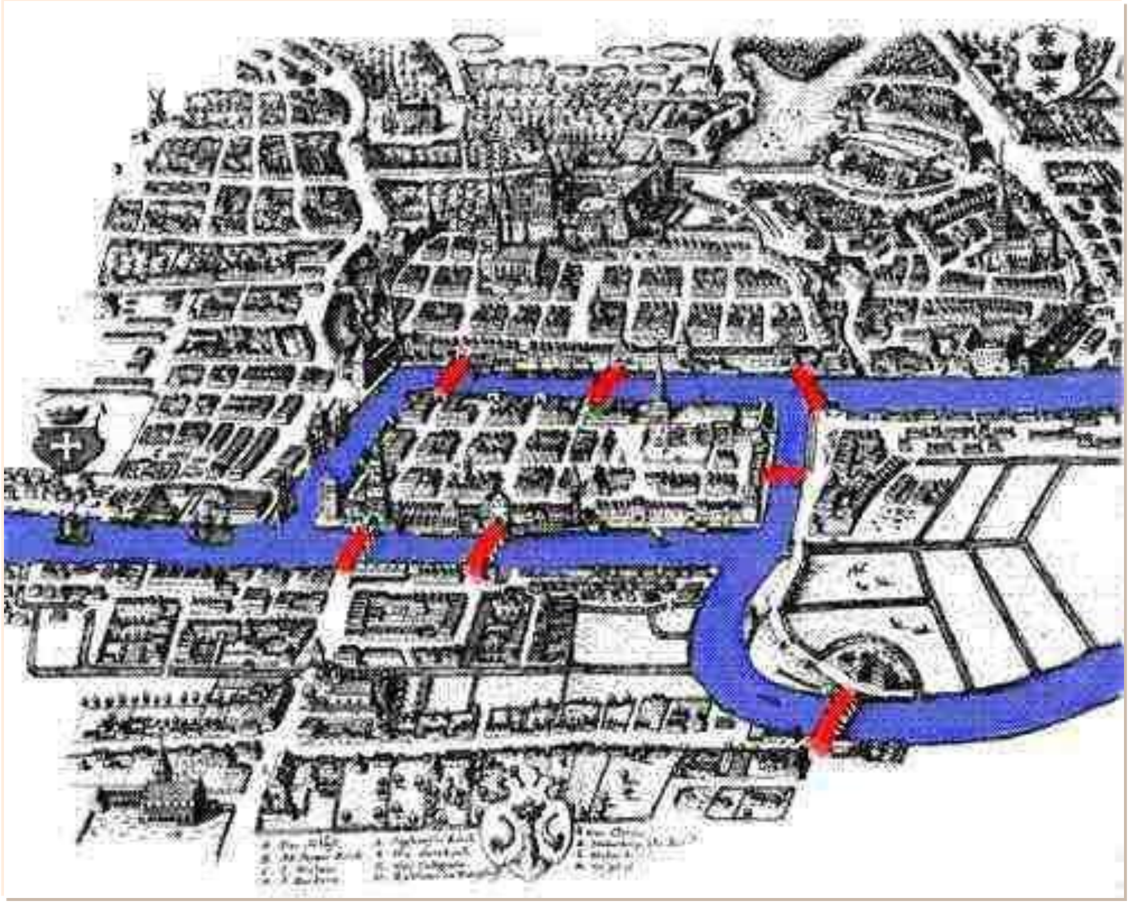
Department of Computer and IT Engineering
University of Kurdistan

Complex Networks

Graph Theory

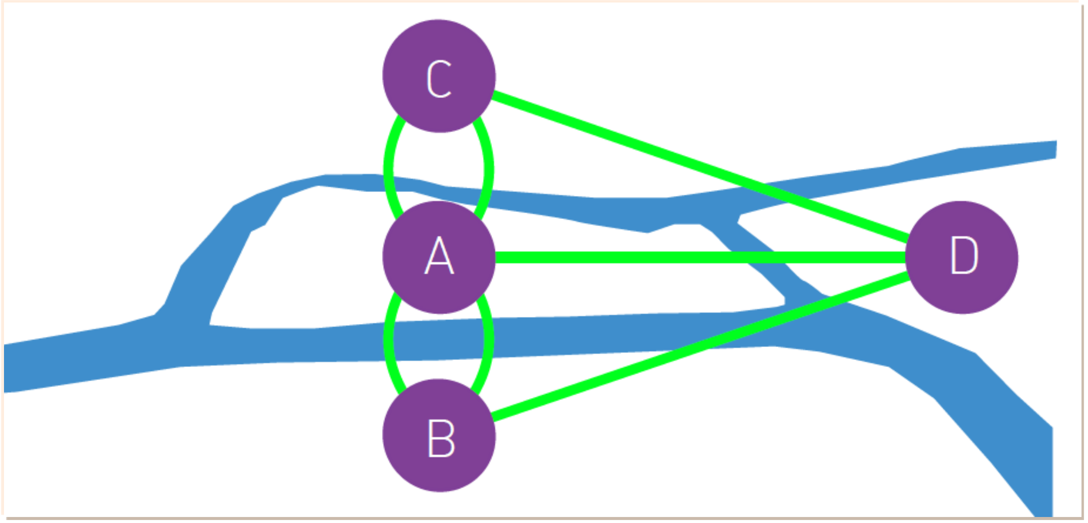
By: Dr. Alireza Abdollahpouri

The Bridges of Konigsberg



Can one walk across the seven bridges and never cross the same bridge twice?

The Bridges of Konigsberg



Can one walk across the seven bridges and never cross the same bridge twice?

1735: Euler's theorem:

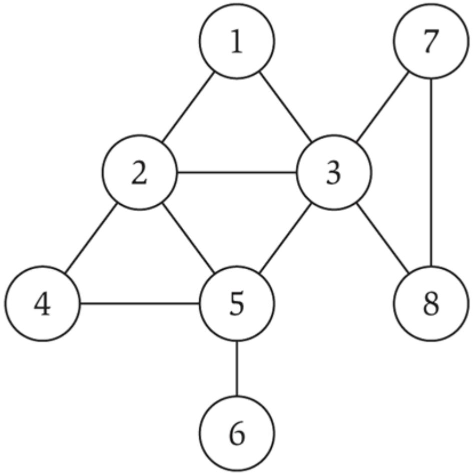
(a) If a graph has more than two nodes of odd degree, there is no path.

(b) If a graph is connected and has no odd degree nodes, it has at least one path.

No matter how smart we are, we will never find the desired path. Networks have properties encoded in their structure that limit or enhance their behavior.

Introduction to graph theory

- Graph – mathematical object consisting of a set of:
  - *V* = nodes (vertices, points).
  - *E* = edges (links, arcs) between pairs of nodes.
  - Denoted by *G = (V, E).*
  - Captures pairwise relationship between objects.
  - Graph size parameters: *n = |V|, m = |E|.*



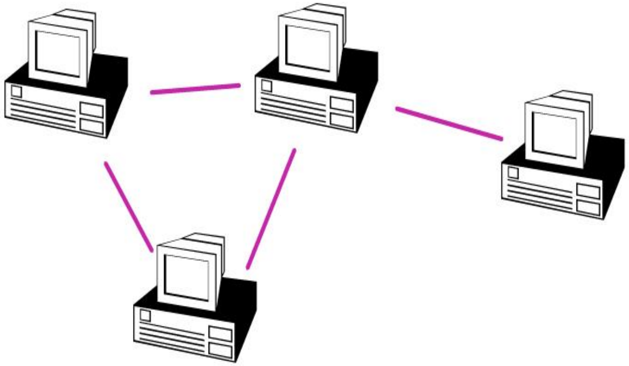
V = { 1, 2, 3, 4, 5, 6, 7, 8 }

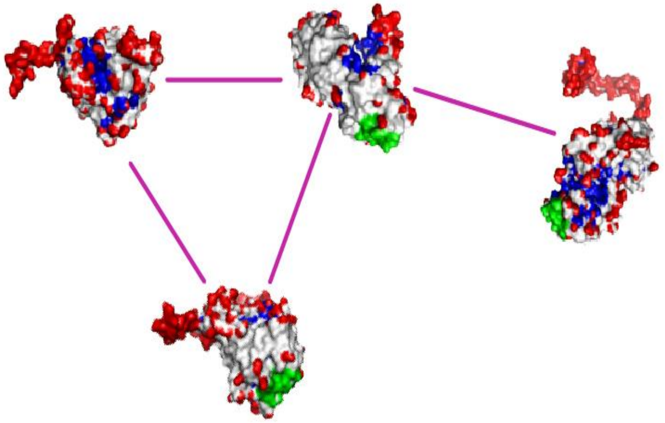
E = { {1,2}, {1,3}, {2,3}, {2,4}, {2,5}, {3,5}, {3,7}, {3,8}, {4,5}, {5,6} }

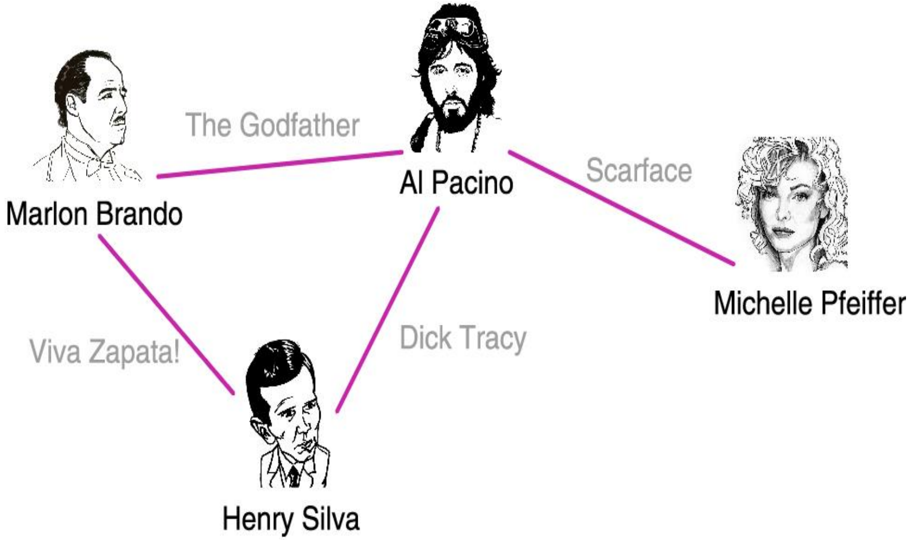
n = 8

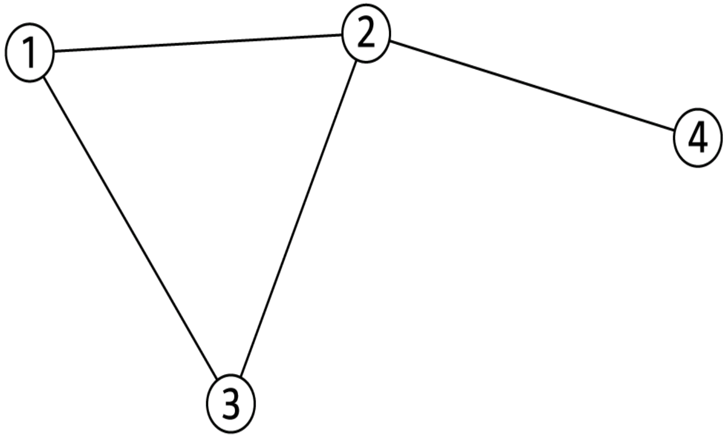
m = 11

A Common Language









N=4
M=4

Networks or Graphs

network **often refers to real systems**

- **www,**
- **Social network**
- **Metabolic network.**

Language: (Network, node, link)

graph**: mathematical representation of a network**

- **web graph,**
- **social graph (a Facebook term)**

Language: (Graph, vertex, edge)

We will try to make this distinction whenever it is appropriate, but in most cases we will use the two terms interchangeably.

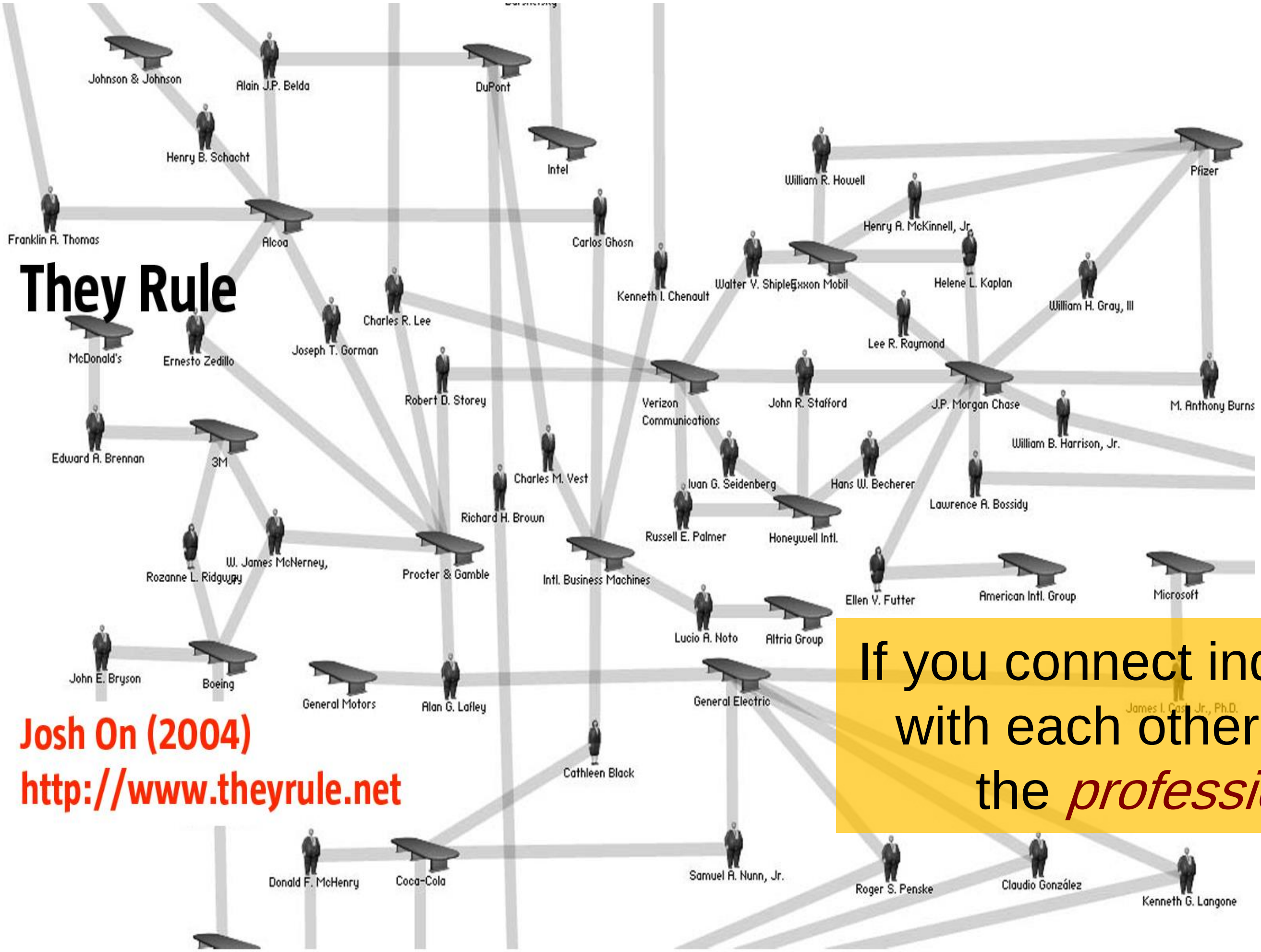
Choosing a Proper Representation

- The choice of the proper network representation determines our ability to use network theory successfully.

- In some cases there is a unique, unambiguous representation.
- In other cases, the representation is by no means unique.

- For example, the way we assign the links between a group of individuals will determine the nature of the question we can study.

Choosing a Proper Representation



Josh On (2004)
http://www.theyrule.net

If you connect individuals that work with each other, you will explore the *professional network.*

Choosing a Proper Representation

If you connect individuals based on their first name (*all Peters connected to each other*), you will be exploring what?

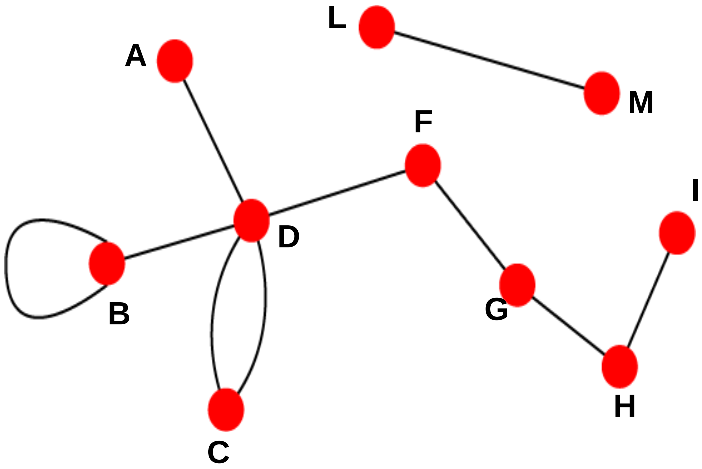
It is a network, nevertheless.

may have little practical utility

Undirected vs. Directed Networks

Undirected

Links: undirected (*symmetrical*)

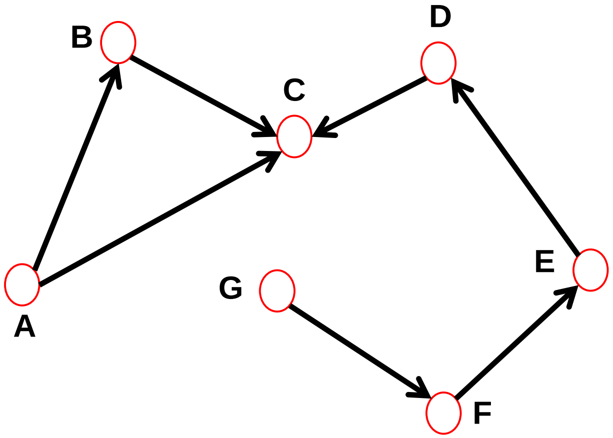


Undirected links :
coauthorship links
Actor network
protein interactions

Directed

Links: directed (*arcs*).

Digraph = directed graph:



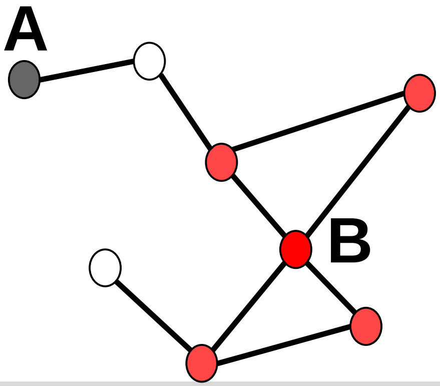
An undirected link is the superposition of two opposite directed links.

Directed links :
URLs on the www
phone calls
metabolic reactions

Degree, Average Degree and Degree Distribution

Node Degrees

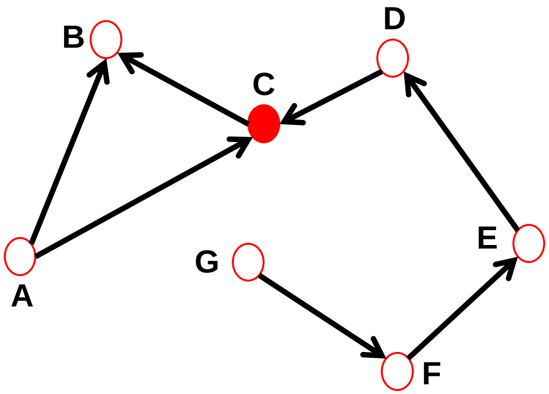
Undirected



Node degree: the number of links connected to the node.

$$k_A = 1 \qquad k_B = 4$$

Directed



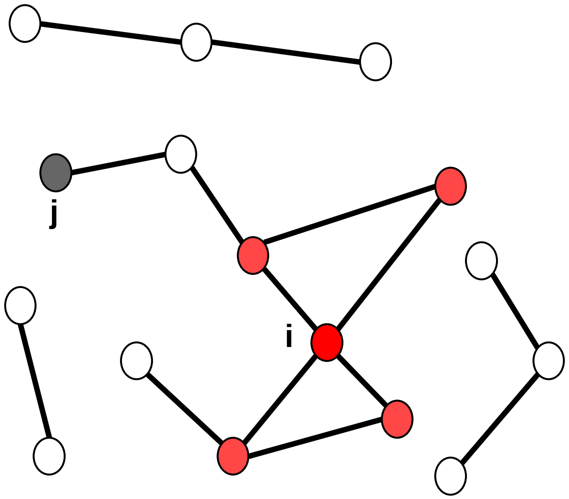
In *directed networks* we can define an in-degree and out-degree. The (total) degree is the sum of in- and out-degree.

$$k_C^{in} = 2 \qquad k_C^{out} = 1 \qquad k_C = 3$$

Source: a node with $k^{in}= 0$; Sink: a node with $k^{out}= 0$.

Average Degree

Undirected

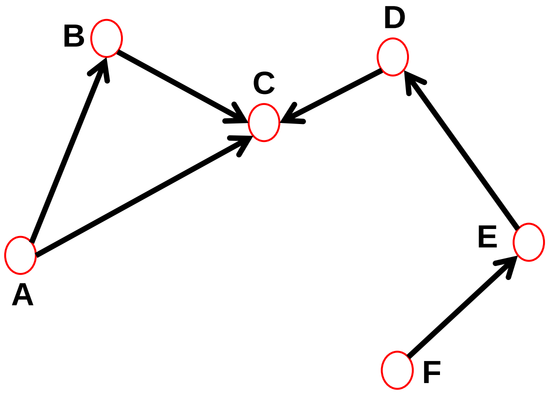


$$\langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2L}{N}$$

N – the number of nodes in the graph

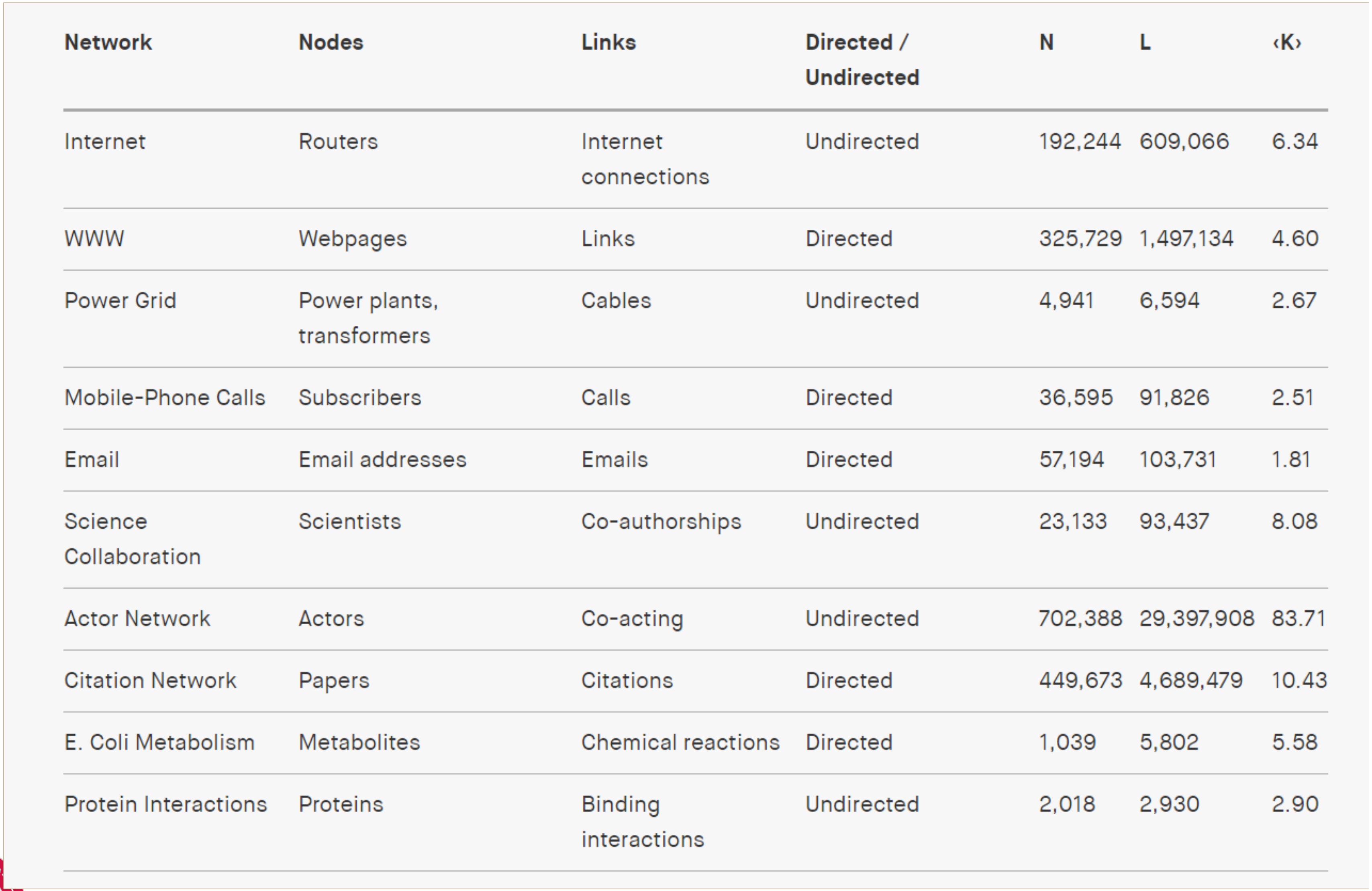
L: Number of links in the graph

Directed



$$\langle k^{\text{in}} \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i^{\text{in}} = \langle k^{\text{out}} \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i^{\text{out}} = \frac{L}{N}$$

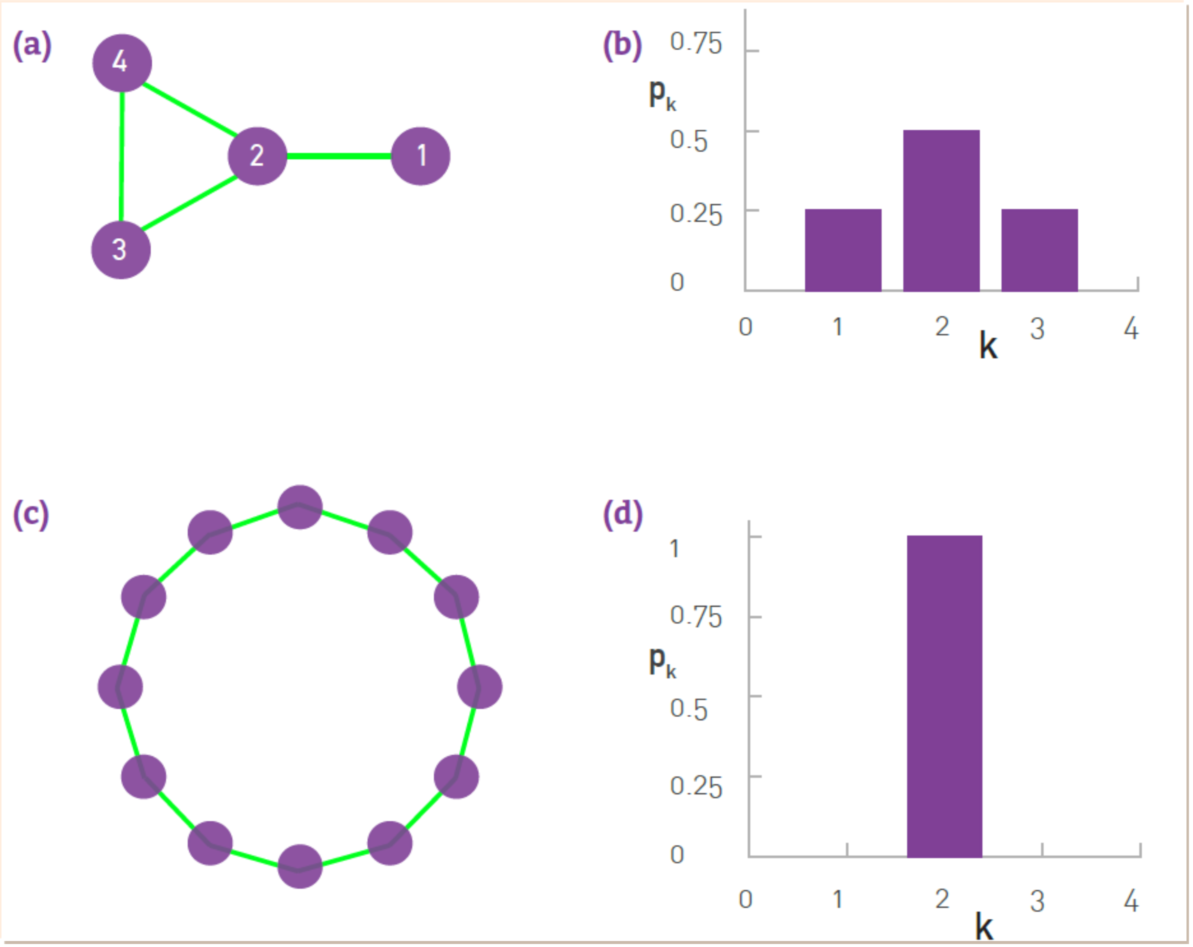
Average Degree (Real Networks)



Network	Nodes	Links	Directed / Undirected	N	L	‹K›
Internet	Routers	Internet connections	Undirected	192,244	609,066	6.34
WWW	Webpages	Links	Directed	325,729	1,497,134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594	2.67
Mobile-Phone Calls	Subscribers	Calls	Directed	36,595	91,826	2.51
Email	Email addresses	Emails	Directed	57,194	103,731	1.81
Science Collaboration	Scientists	Co-authorships	Undirected	23,133	93,437	8.08
Actor Network	Actors	Co-acting	Undirected	702,388	29,397,908	83.71
Citation Network	Papers	Citations	Directed	449,673	4,689,479	10.43
E. Coli Metabolism	Metabolites	Chemical reactions	Directed	1,039	5,802	5.58
Protein Interactions	Proteins	Binding interactions	Undirected	2,018	2,930	2.90

Degree Distribution

P(k): probability that a randomly chosen node has degree *k*



N_k = # nodes with degree k

$P(k) = N_k / N$ ➔ plot

Degree Distribution

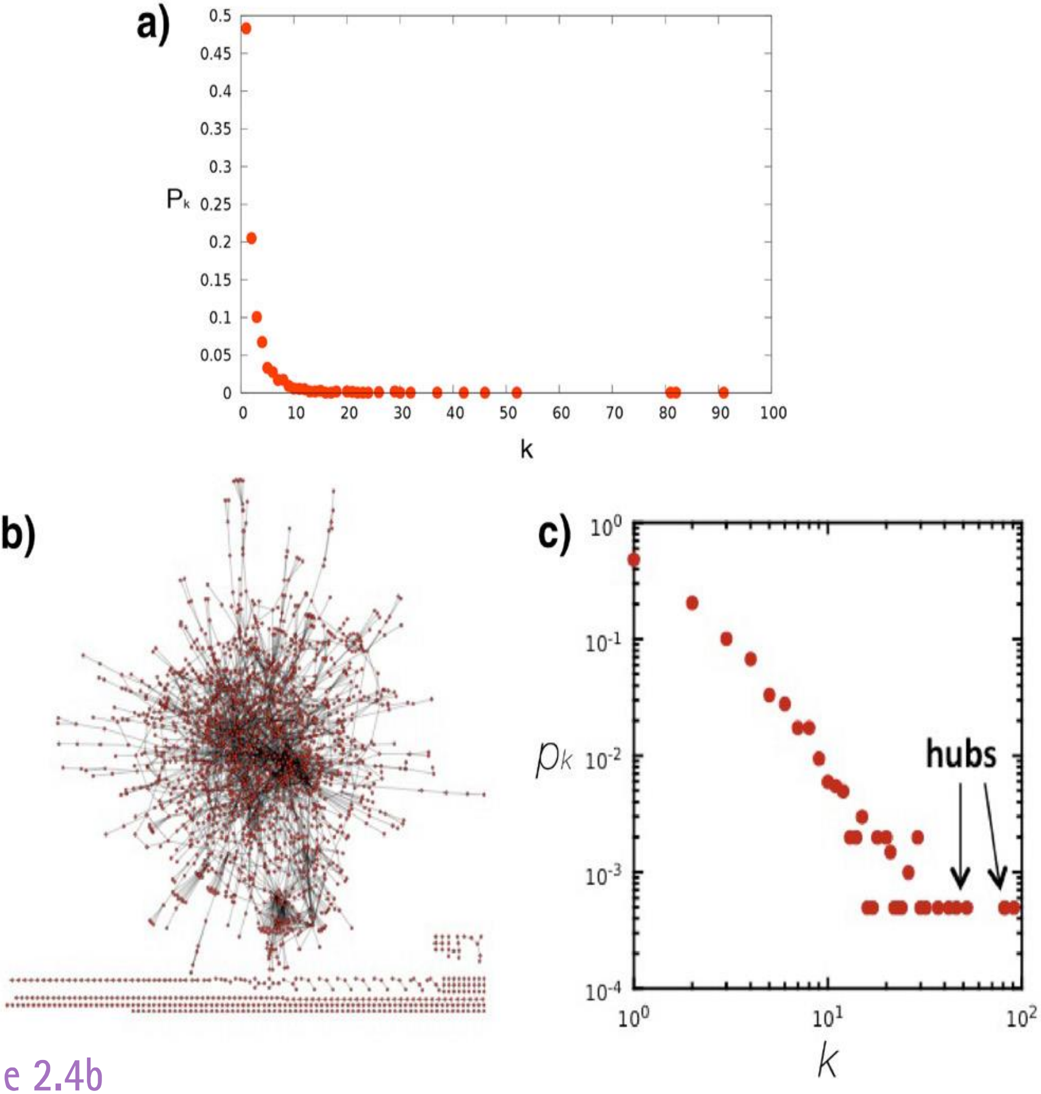
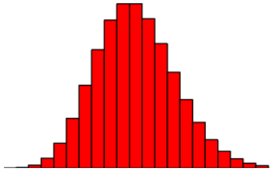


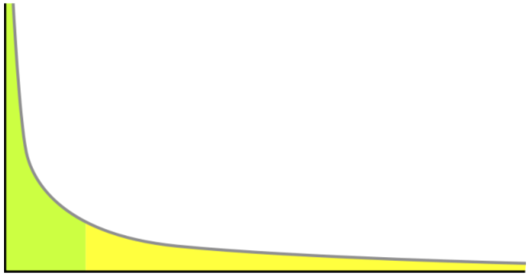
Image 2.4b

Degree Distribution

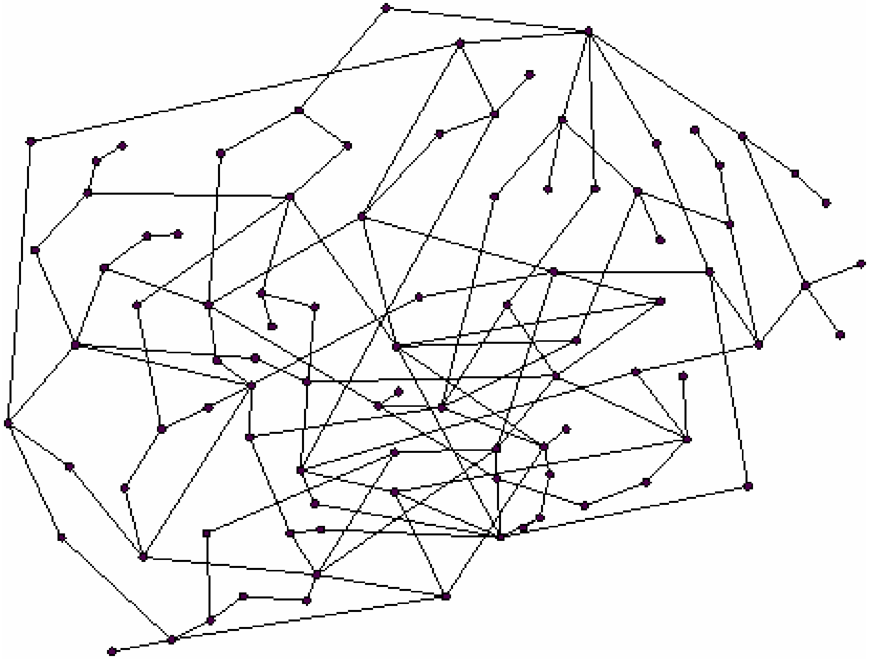
- Let p_k denote a fraction of nodes with degree k
- We can plot a histogram of p_k vs. k
- In a Erdos-Renyi random graph degree distribution follows Poisson distribution



- Degrees in real networks are heavily skewed to the right
- Distribution has a long tail of values that are far above the mean
- Heavy (long) tail:
  - Amazon sales
  - word length distribution, …

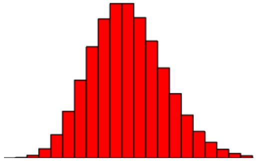


Poisson vs. Scale-free network

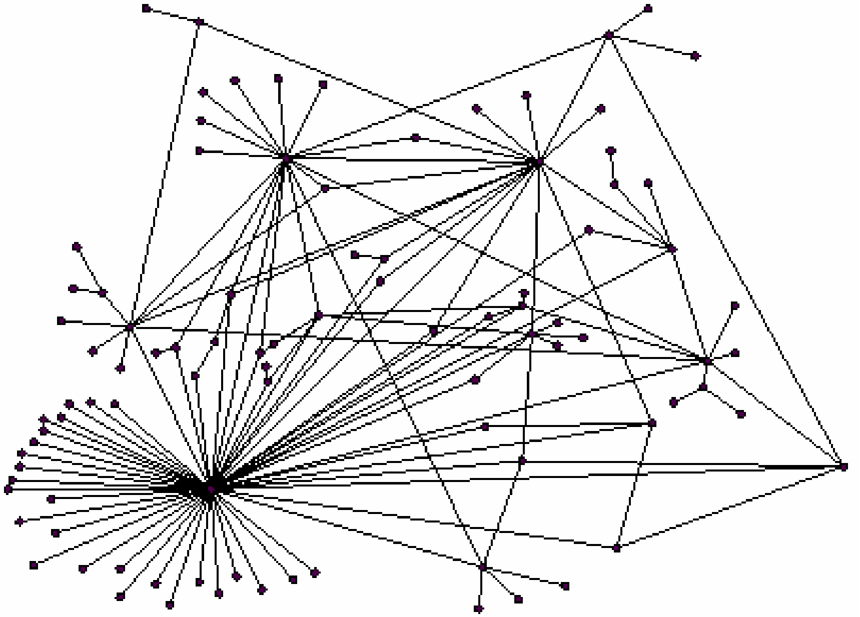


Poisson network

(Erdos-Renyi random graph)

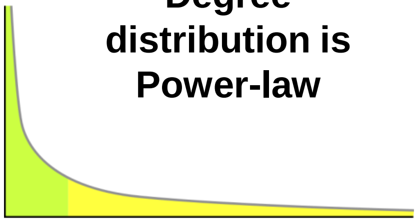


Degree distribution is Poisson

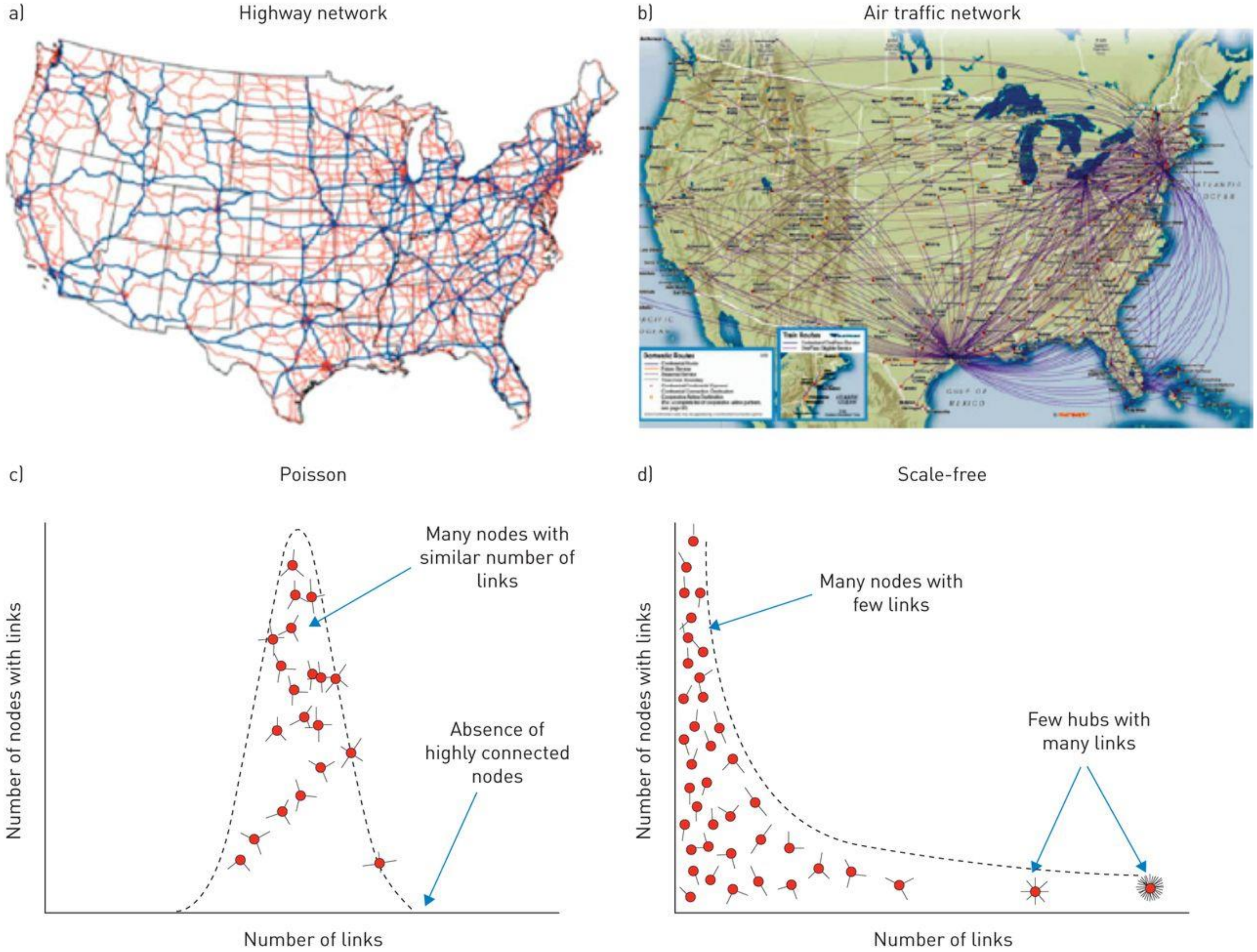


Scale-free (power-law) network

Degree distribution is Power-law



Poisson vs. Scale-free network


a)
Highway network
b)
Air traffic network
c)
Poisson
Many nodes with similar number of links
Absence of highly connected nodes
Number of nodes with links
Number of links
d)
Scale-free
Many nodes with few links
Few hubs with many links
Number of nodes with links
Number of links

Graph Representation

Graph Representation

- Adjacency matrix
- Adjacency list

Adjacency Matrix

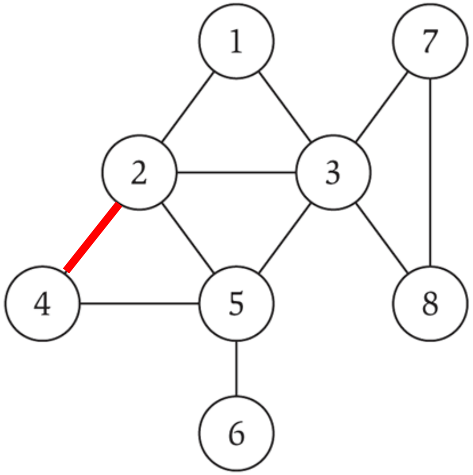
$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ A_{N1} & \dots & \dots & A_{NN} \end{pmatrix}$$

$A_{ij} = 1$ if there is a link pointing from node *j* to node *i*

$A_{ij} = 0$ if nodes *i* and *j* are not connected to each other

Graph Representation: Adjacency Matrix

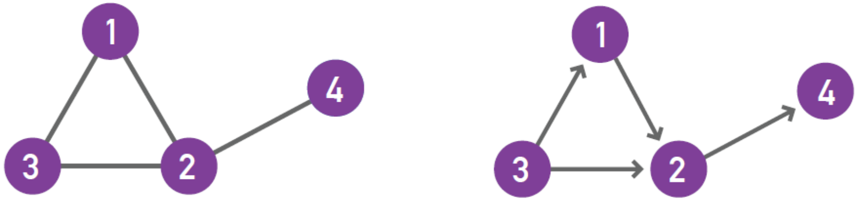
- Adjacency matrix. n-by-n matrix with $A_{uv} = 1$ if (u, v) is an edge.
  - ○ Two representations of each edge (symmetric matrix for undirected graphs; not for directed graphs).
  - ○ Space: proportional to n^2.
    - ○ Not efficient for *sparse graphs* (small number of edges compared to the maximum possible number of edges in the graph),
      - ➢ e.g., biological networks
    - ○ Algorithms might have longer running time if this representation used
  - ○ Checking if (u, v) is an edge takes Θ(1) time.
  - ○ Identifying all edges takes $\Theta(n^2)$ time.



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

← ∑ = degree of node 2

Adjacency Matrix



$$A_{ij} = \begin{matrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix} \qquad A_{ij} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

$$k_2 = \sum_{j=1}^{4} A_{2j} = \sum_{i=1}^{4} A_{i2} = 3 \qquad k_2^{\text{in}} = \sum_{j=1}^{4} A_{2j} = 2,\ k_2^{\text{out}} = \sum_{i=1}^{4} A_{i2} = 1$$

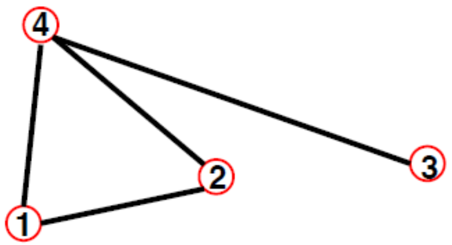
$$A_{ij} = A_{ji} \quad A_{ii} = 0 \qquad A_{ij} \neq A_{ji} \quad A_{ii} = 0$$

$$L = \frac{1}{2} \sum_{i,j=1}^{N} A_{ij} \qquad L = \sum_{i,j=1}^{N} A_{ij}$$

$$\langle k \rangle = \frac{2L}{N} \qquad \langle k^{\text{in}} \rangle = \langle k^{\text{out}} \rangle = \frac{L}{N}$$

Adjacency Matrix

Undirected



$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

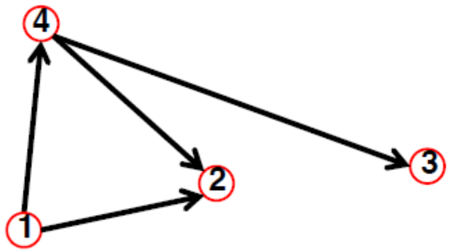
$$A_{ij} = A_{ji}$$
$$A_{ii} = 0$$

$$k_i = \sum_{j=1}^{N} A_{ij}$$

$$k_j = \sum_{i=1}^{N} A_{ij}$$

$$L = \frac{1}{2}\sum_{i=1}^{N} k_i = \frac{1}{2}\sum_{ij}^{N} A_{ij}$$

Directed



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

$$A_{ij} \neq A_{ji}$$
$$A_{ii} = 0$$

$$k_i^{out} = \sum_{j=1}^{N} A_{ij}$$

$$k_j^{in} = \sum_{i=1}^{N} A_{ij}$$

$$L = \sum_{i=1}^{N} k_i^{in} = \sum_{j=1}^{N} k_j^{out} = \sum_{i,j}^{N} A_{ij}$$

Adjacency Matrix

Advantages

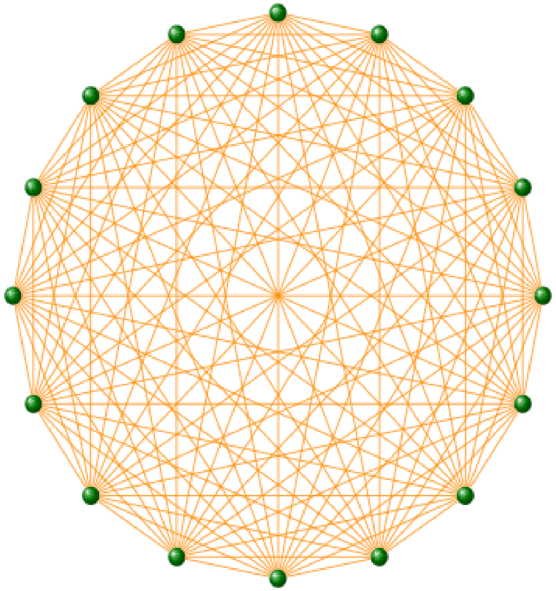
- Convenient for analytical calculations
- Easy to remove/add an edge (changing the value of an element in the matrix is O(1)

Disadvantages

- Needs a lot of memory - O(N^2) space
- Inconvenient for numerical calculations

Complete Graph

The maximum number of links a network of N nodes can have is:



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

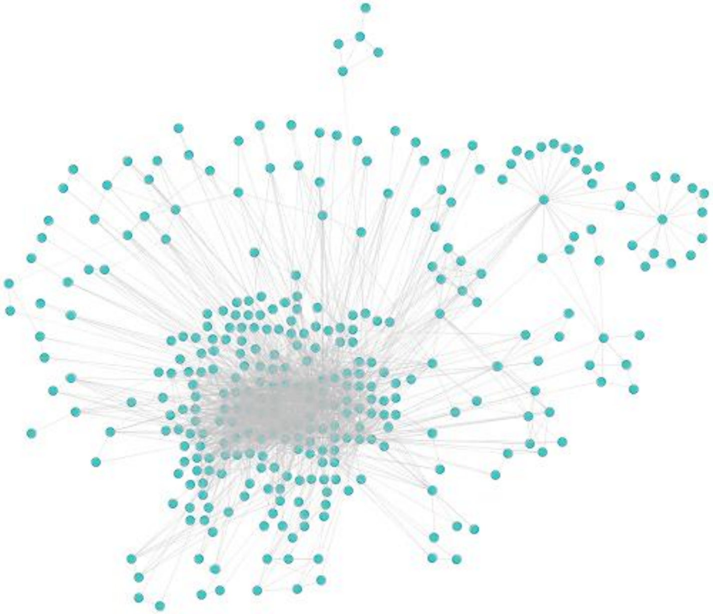
A graph with degree L=L$_{\mathbf{max}}$ is called a complete graph, and its average degree is **<k>=N-1**

Real Networks are Sparse

Most networks observed in real systems are sparse:

$L \ll L_{max}$ or $\langle k \rangle \ll N-1$.

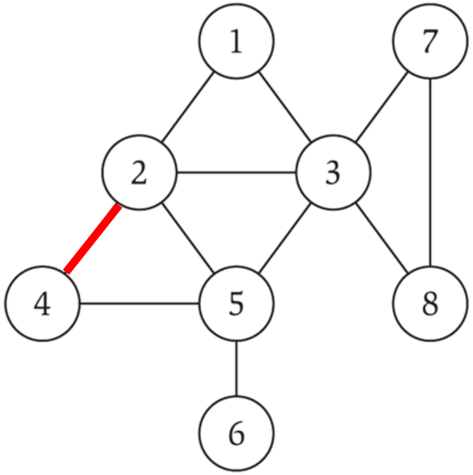
WWW (ND Sample):	N=325,729;	L=1.4 10^6	$L_{max}=10^{12}$	<k>=4.51
Protein (*S. Cerevisiae*):	N= 1,870;	L=4,470	$L_{max}=10^7$	<k>=2.39
Coauthorship (Math):	N= 70,975;	L=2 10^5	$L_{max}=3\ 10^{10}$	<k>=3.9
Movie Actors:	N=212,250;	L=6 10^6	$L_{max}=1.8\ 10^{13}$	<k>=28.78

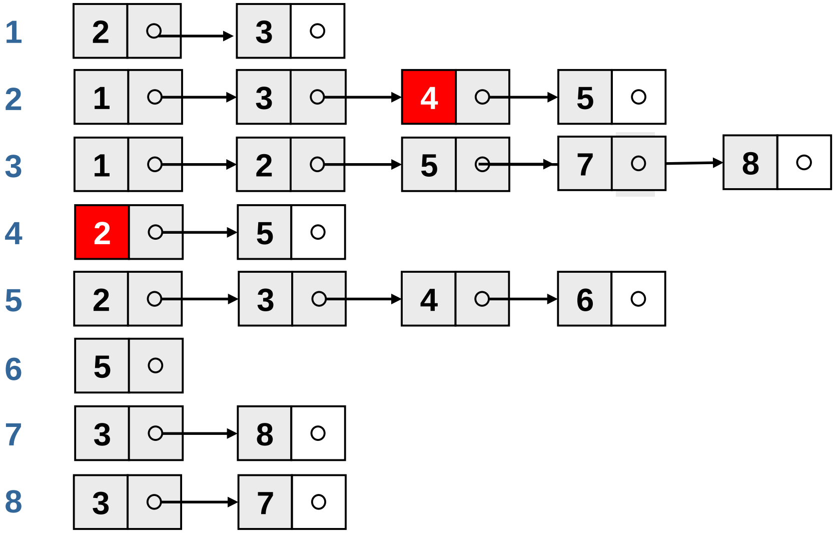


(Source: Albert, Barabasi, RMP2002)

Graph Representation: Adjacency List

- Adjacency list. Node indexed array of lists.
  - ○ Two representations of each edge.
  - ○ Space proportional to *m + n*.
  - ○ Checking if *(u, v)* is an edge takes *O(deg(u))* time.
  - ○ Identifying all edges takes *Θ(m+n)* time = linear time for *G(V,E)*.
  - ○ Requires *O(m+n)* space. Good for dealing with sparse graphs.





1: 2 → 3

2: 1 → 3 → 4 → 5

3: 1 → 2 → 5 → 7 → 8

4: 2 → 5

5: 2 → 3 → 4 → 6

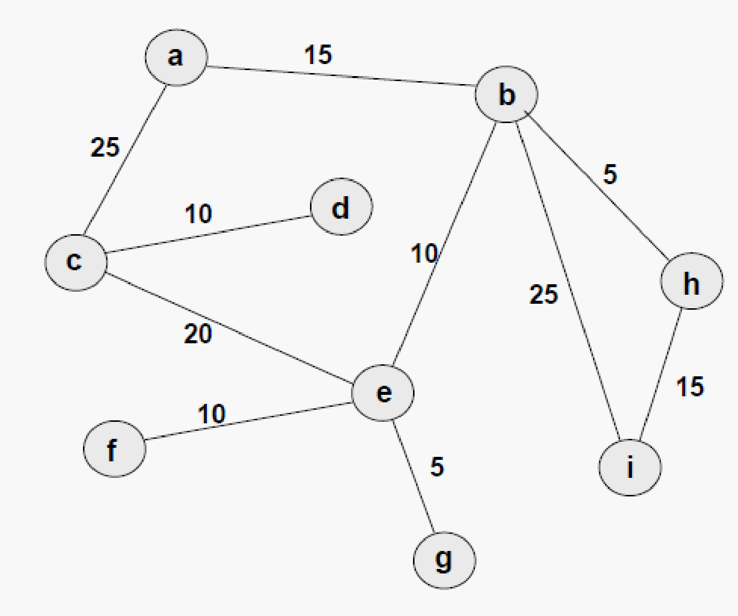
6: 5

7: 3 → 8

8: 3 → 7

Weighted Graphs

- In many applications, each edge of a graph has an associated numerical value, called a weight.
- Usually, the edge weights are nonnegative integers.
- Weighted graphs may be either directed or undirected.



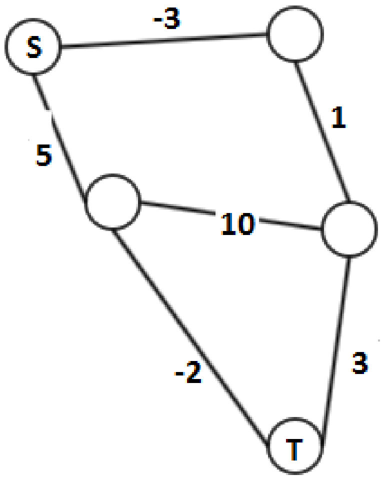
The weight of an edge is often referred to as the "cost" of the edge. In applications, the weight may be a measure of the length of a route, the capacity of a line, the energy required to move between locations along a route, etc.

Weighted Graphs

Weight of edges can represent everything in real world, e.g amount of money to be transferred from one account to an other account can be **positive** or **negative**:

- One gene activates/ inhibits another
- One person trusting/ distrusting another

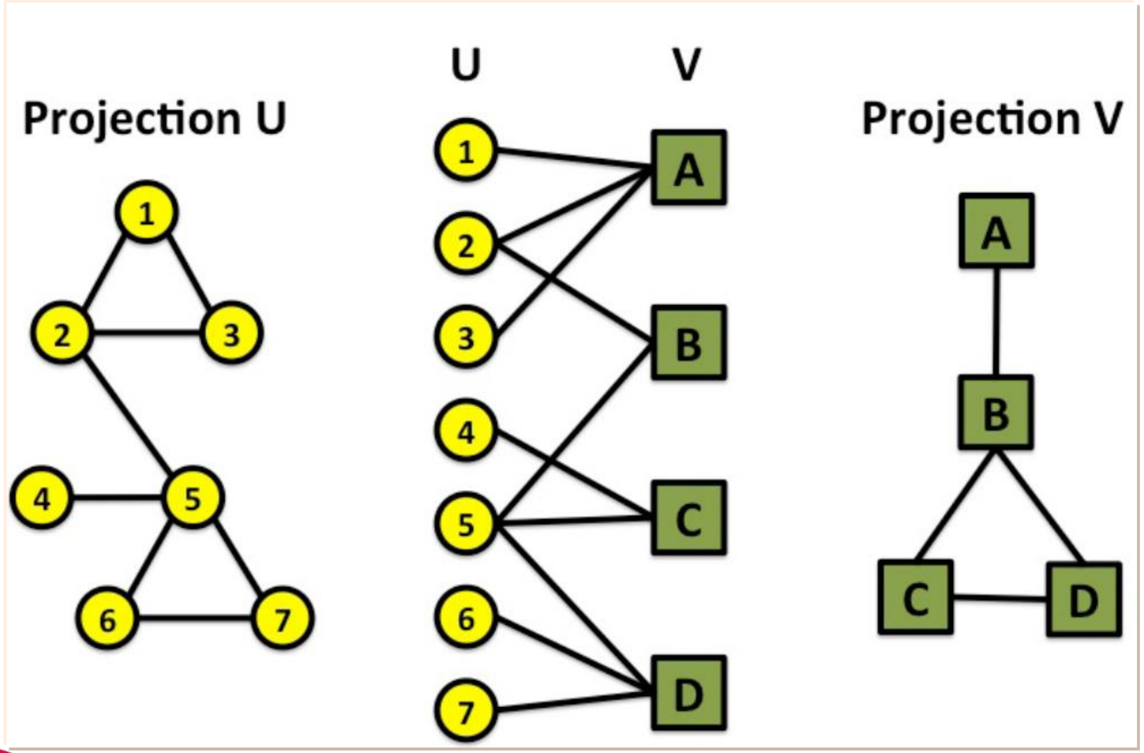
For *weighted networks* the elements of the adjacency matrix carry the weight of the link as: $A_{ij} = w_{ij}$



Think of a driver, who gets paid to drive his employer from **s** to **t** but he pay between **a** and **b** (say travelling between his home and his workplace).

Bipartite Graphs

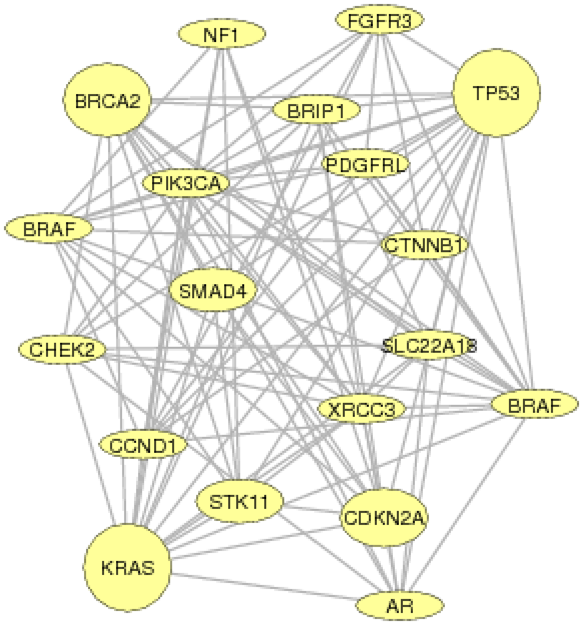
bipartite graph (or bigraph) is a graph whose nodes can be divided into two disjoint sets *U* and *V* such that every link connects a node in *U* to one in *V*; that is, *U* and *V* are independent sets.



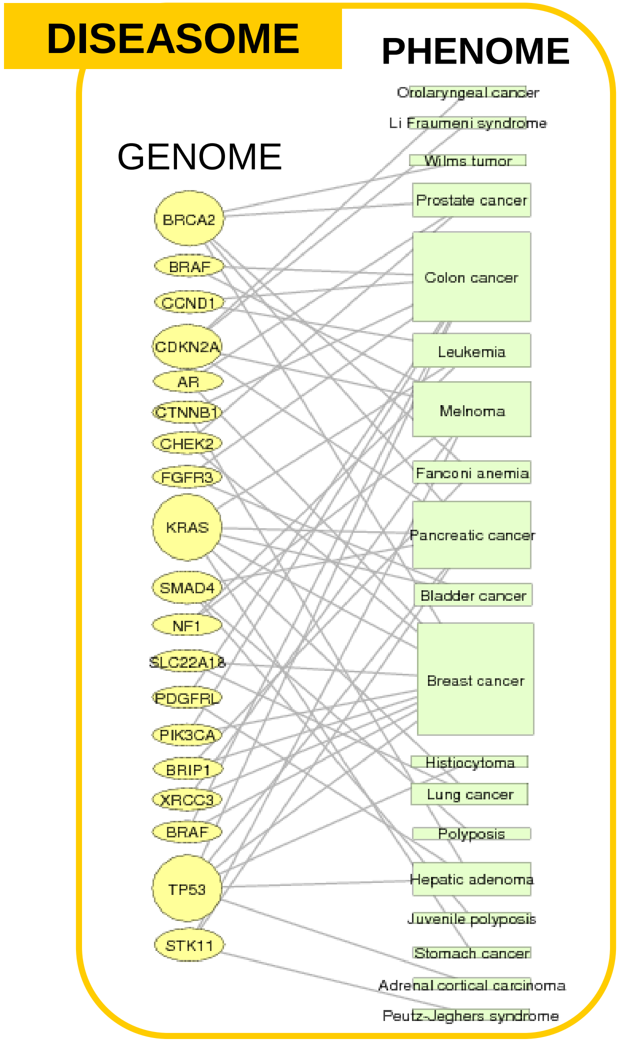
Examples:

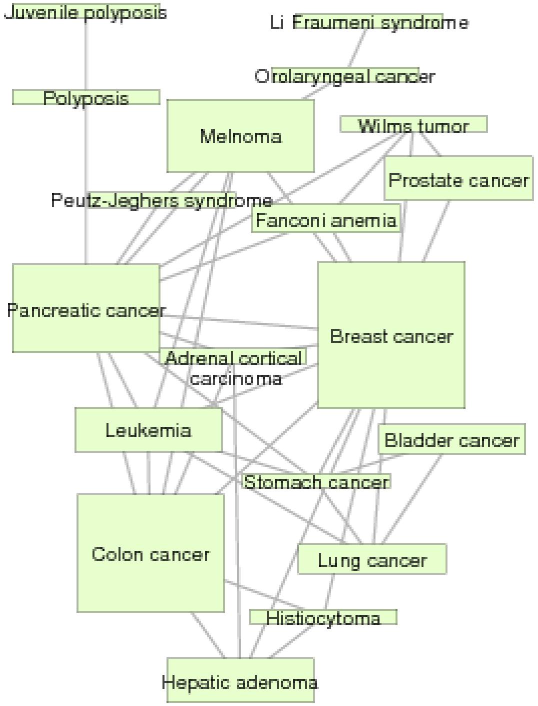
Hollywood actor network
Collaboration networks
Disease network (diseasome)

Bipartite Graphs(Gene Disease Network)



Gene network





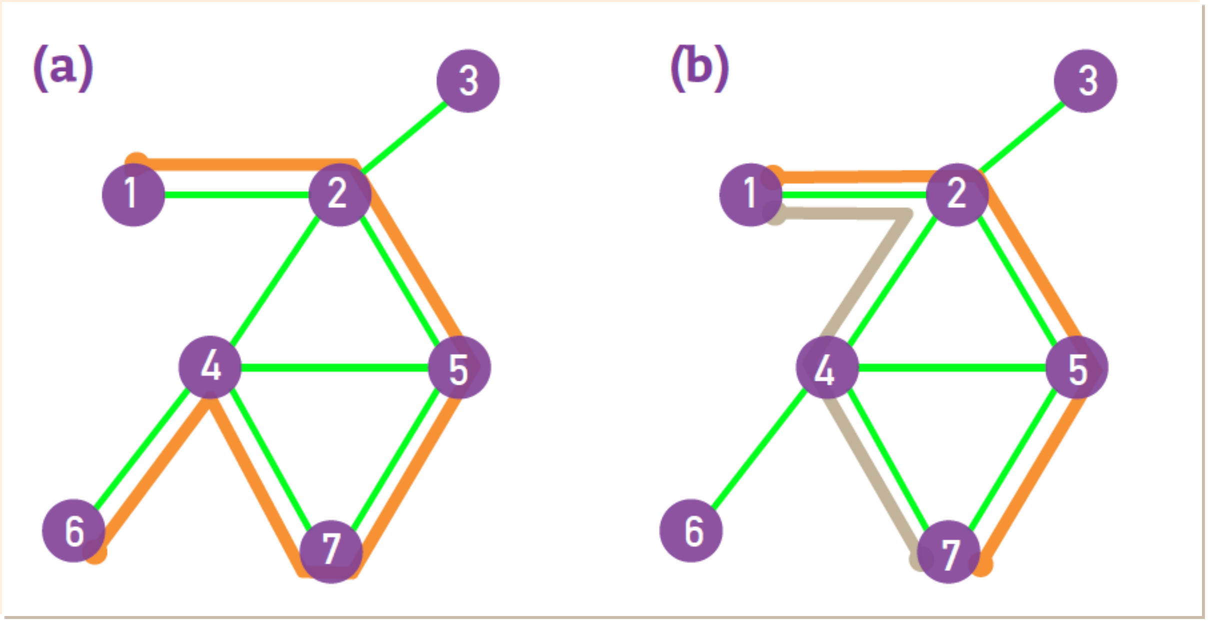
Disease network

Paths

A *path is* a sequence of nodes in which each node is adjacent to the next one

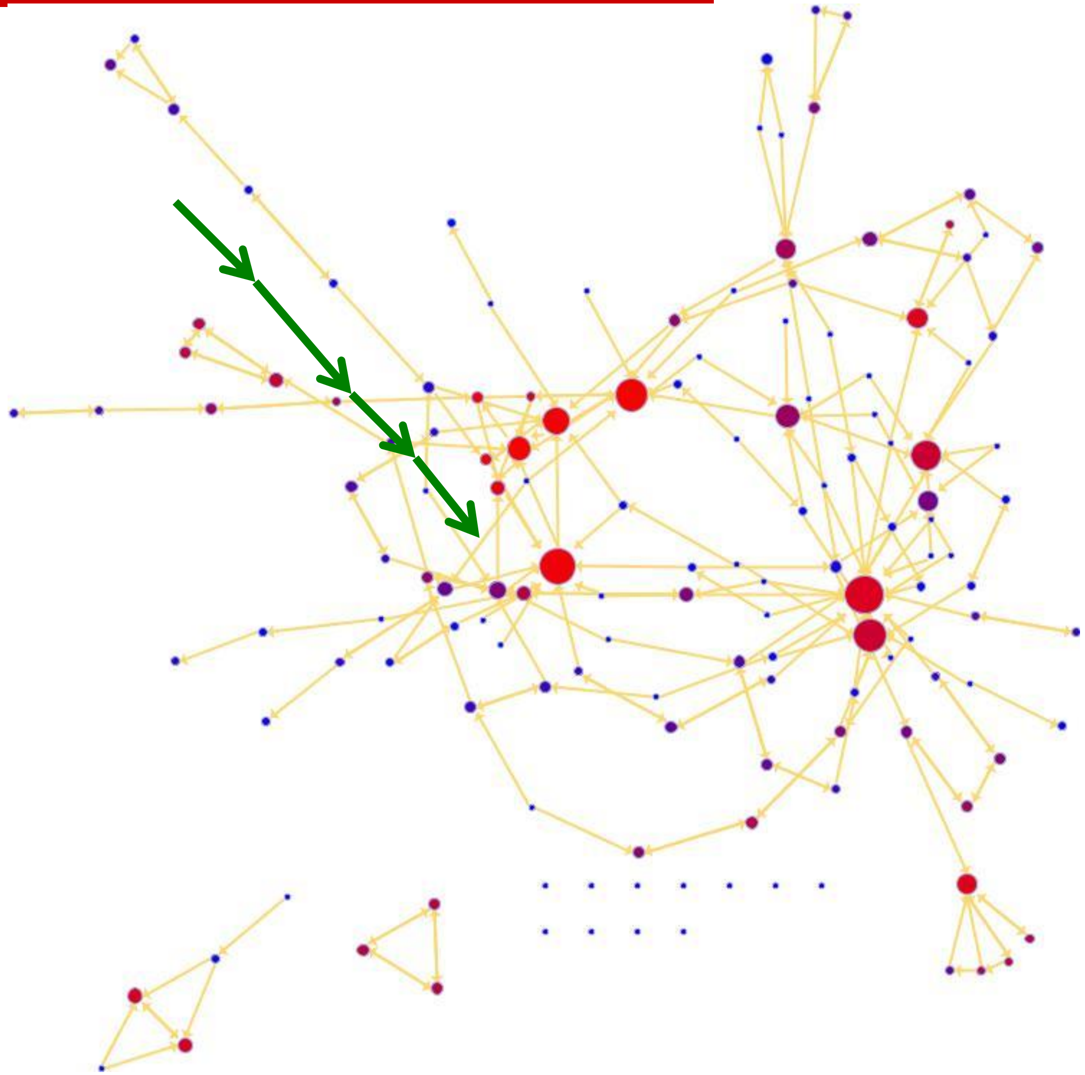
$P_{i0,in}$ of length *n* between nodes i_0 and i_n is an ordered collection of *n+1* nodes and *n* links

$$P_n = \{i_0, i_1, i_2, \ldots, i_n\} \qquad P_n = \{(i_0, i_1), (i_1, i_2), (i_2, i_3), \ldots, (i_{n-1}, i_n)\}$$



- In a directed network, the path can follow only the direction of an arrow.

Characterizing networks: How far apart are things?



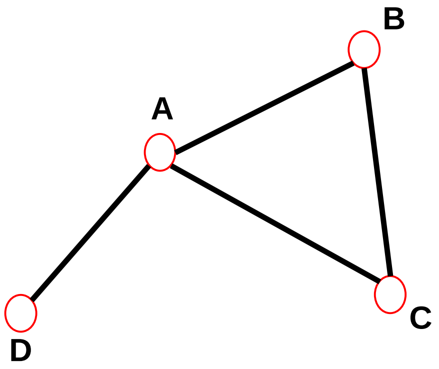
Network metrics: paths

- A ***path*** is any sequence of vertices such that every consecutive pair of vertices in the sequence is connected by an edge in the network.
  - For directed: traversed in the correct direction for the edges.

- path can visit itself (vertex or edge) more than once
  - ***Self-avoiding paths*** do not intersect themselves.

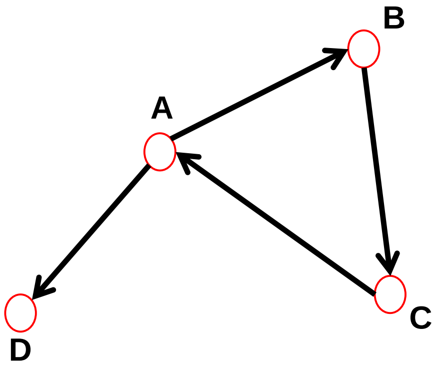
- Path length **r** is the number of edges on the path
  - Called hops

Distance in a Graph- Shortest Path, Geodesic Path



The *distance (shortest path, geodesic path)* between two nodes is defined as the number of edges along the shortest path connecting them.

***If the two nodes are disconnected, the distance is infinity.**

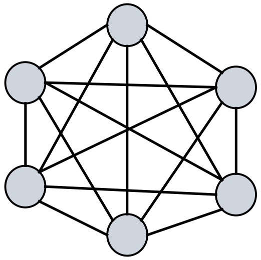


In directed graphs each path needs to follow the direction of the arrows.

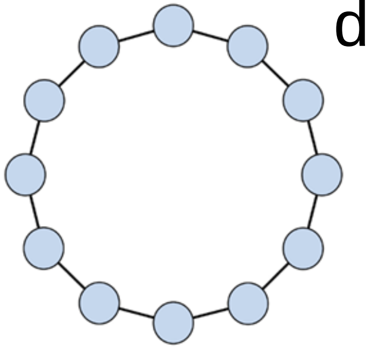
Thus in a digraph the distance from node A to B (on an AB path) is generally different from the distance from node B to A (on a BCA path).

Average distance in networks

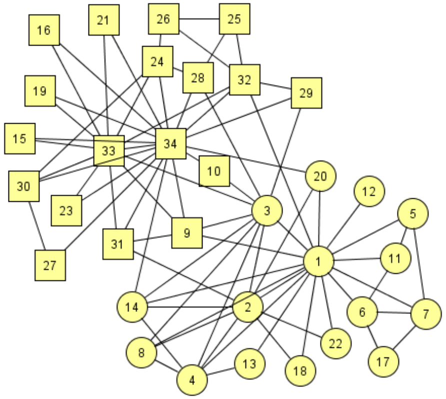
clique: d=1

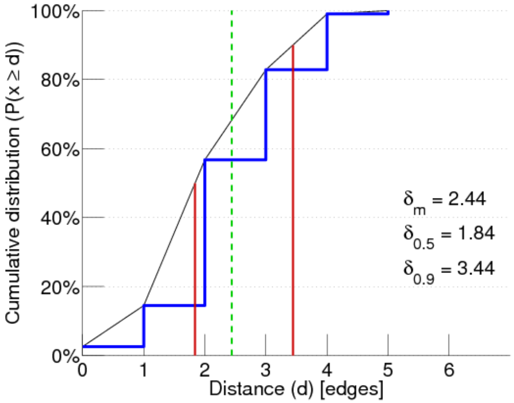


regular lattice (ring): d~N

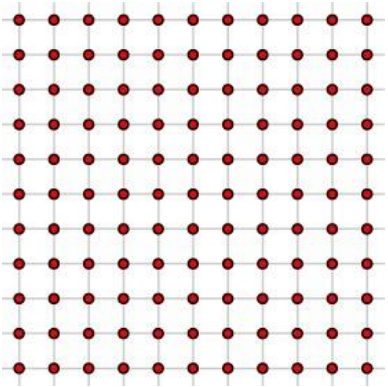


karate club friendship network: d=2.44

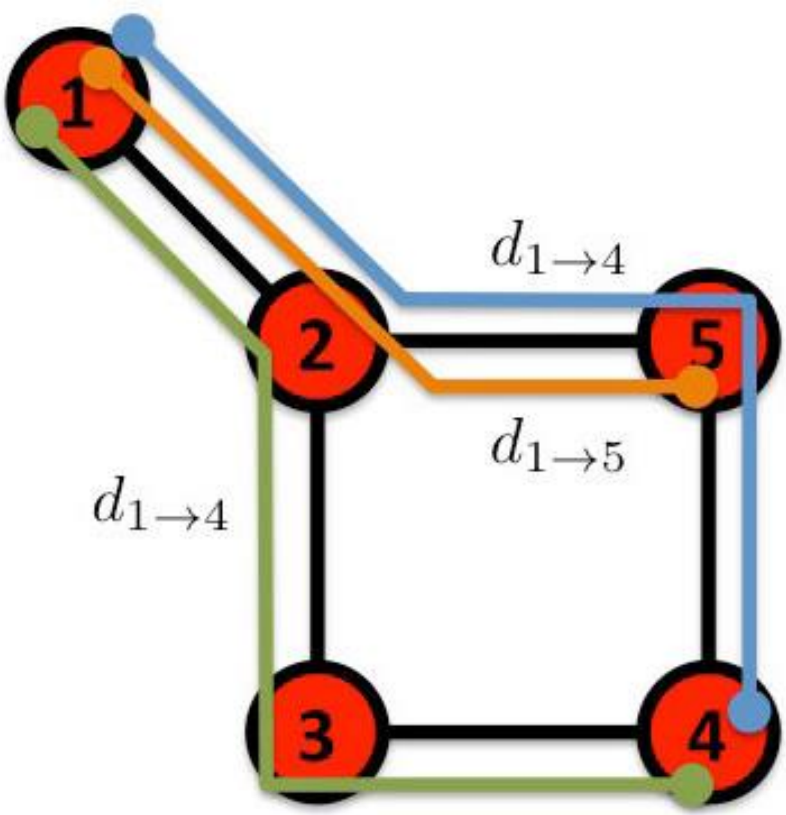




regular lattice (square): $d \sim N^{1/2}$



Shortest Path



Shortest Path

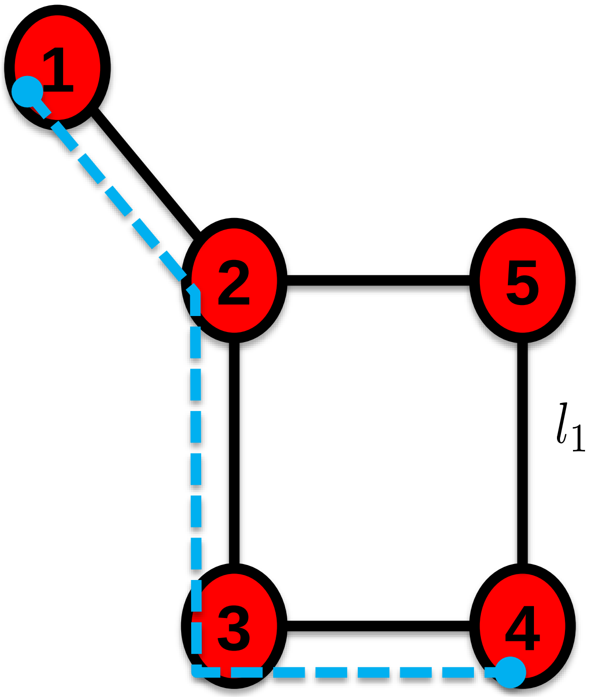
$d_{1\to4} = 3$

$d_{1\to5} = 2$

The path with the shortest length between two nodes (distance).

Diameter and Average Path Length

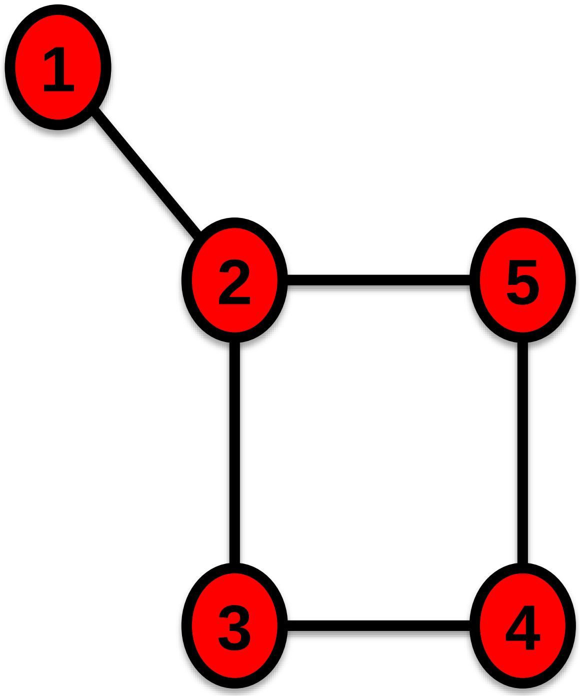
Diameter



$$l_{1\to 4} = 3$$

The longest shortest path in a graph

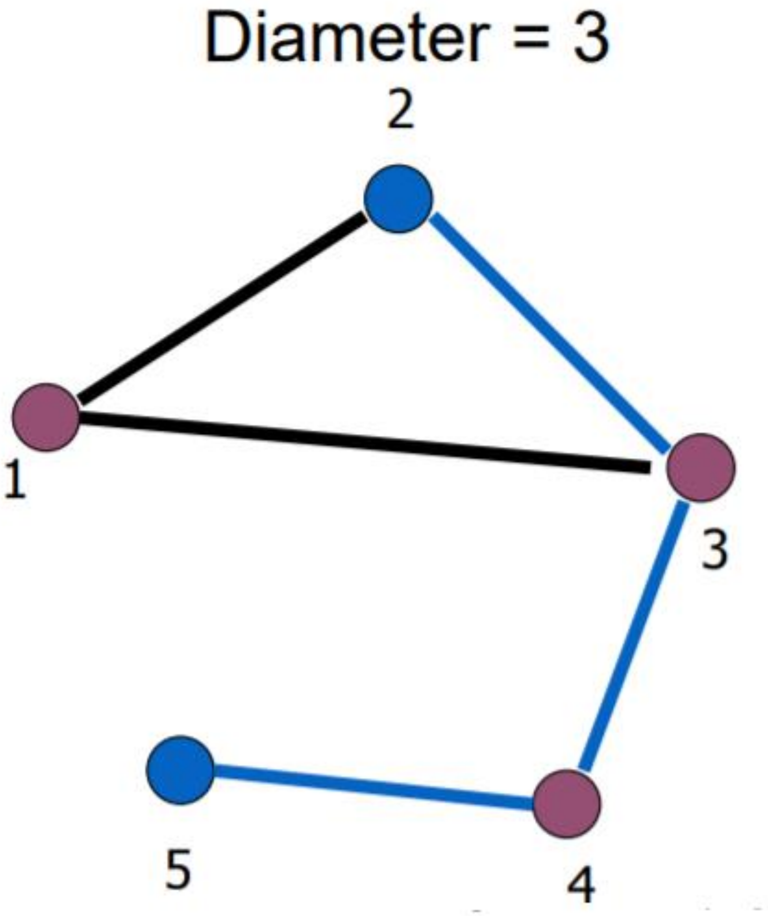
Average Path Length

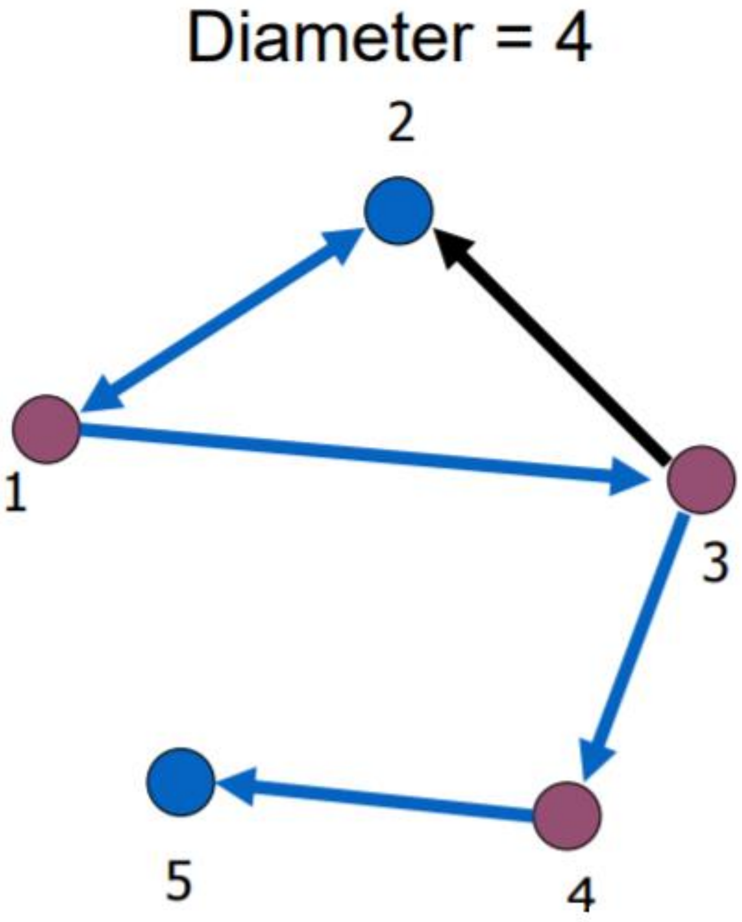


$$(l_{1\to 2} + l_{1\to 3} + l_{1\to 4} + \\ + l_{1\to 5} + l_{2\to 3} + l_{2\to 4} + \\ + l_{2\to 5} + l_{3\to 4} + l_{3\to 5} + \\ + l_{4\to 5})\,/10 = 1.6$$

The average of the shortest paths for all pairs of nodes.

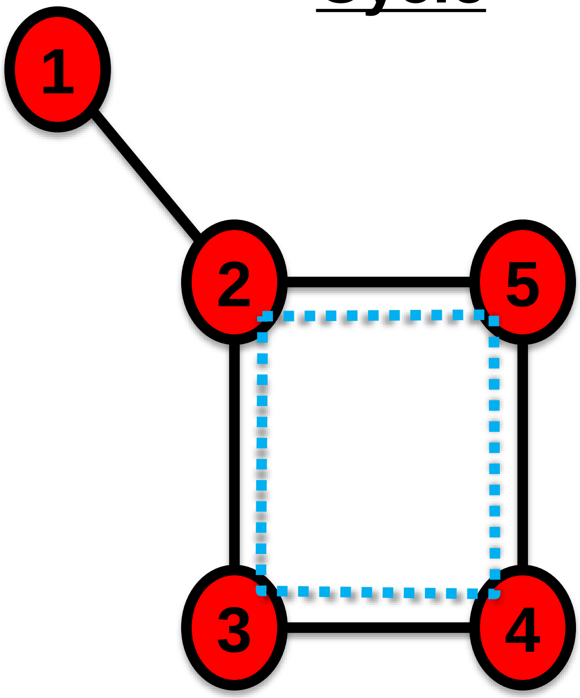
Diameter in directed and undirected Graphs





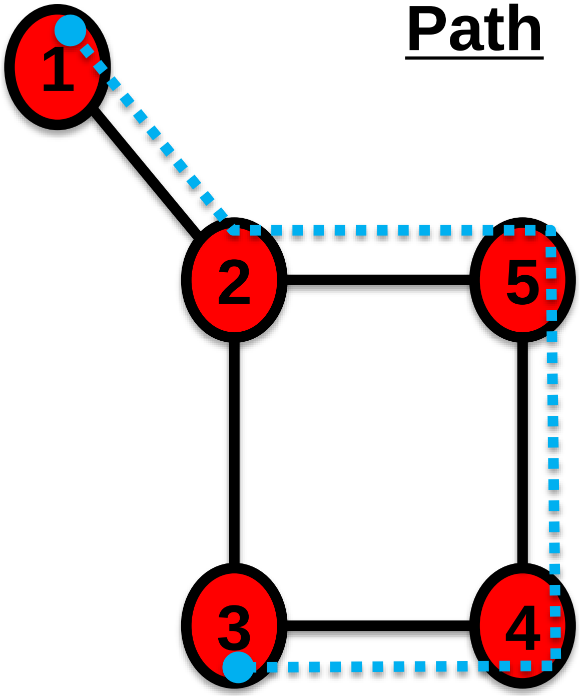
Cycle and Self-avoiding Path

Cycle



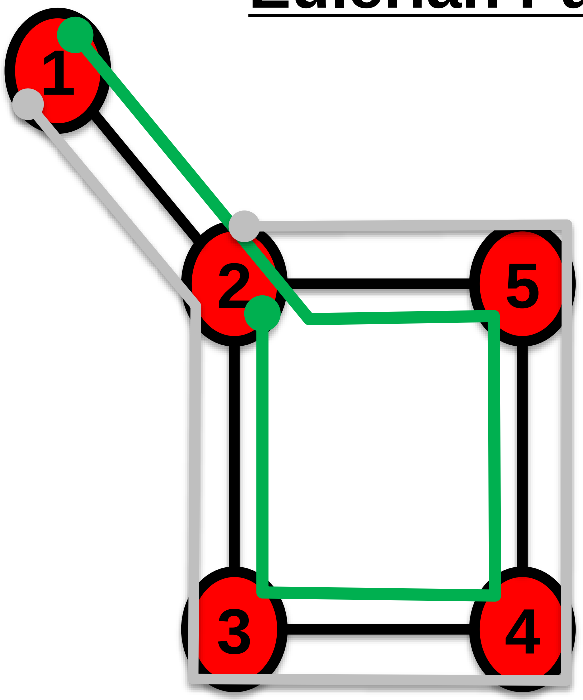
A path with the same start and end node.

Self-avoiding Path

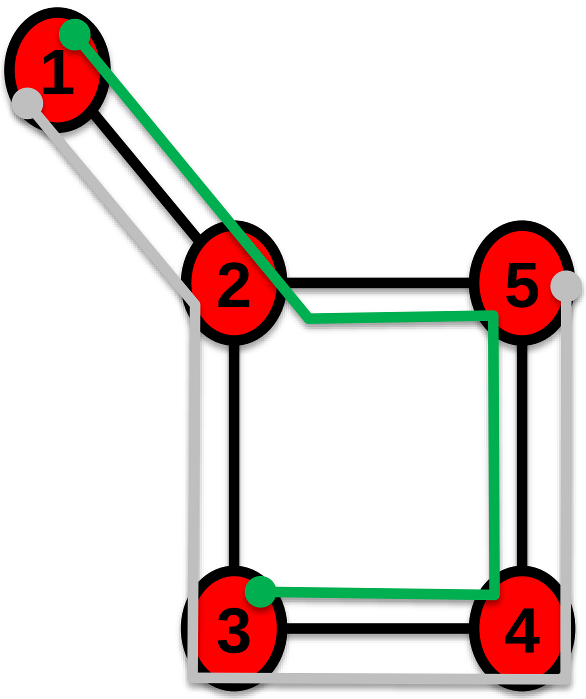


A path that does not intersect itself.

Eulerian and Hamiltonian Path



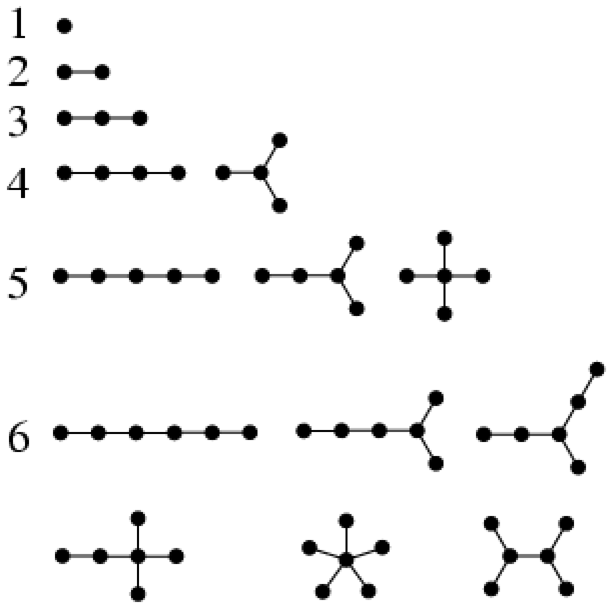
A path that traverses each link exactly once.



A path that visits each node exactly once.

Trees

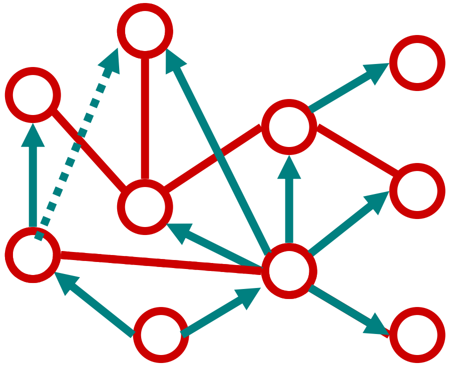
- Trees are undirected graphs that contain **no cycles**



- For n nodes, number of edges m = n-1
- Any node can be dedicated as the root

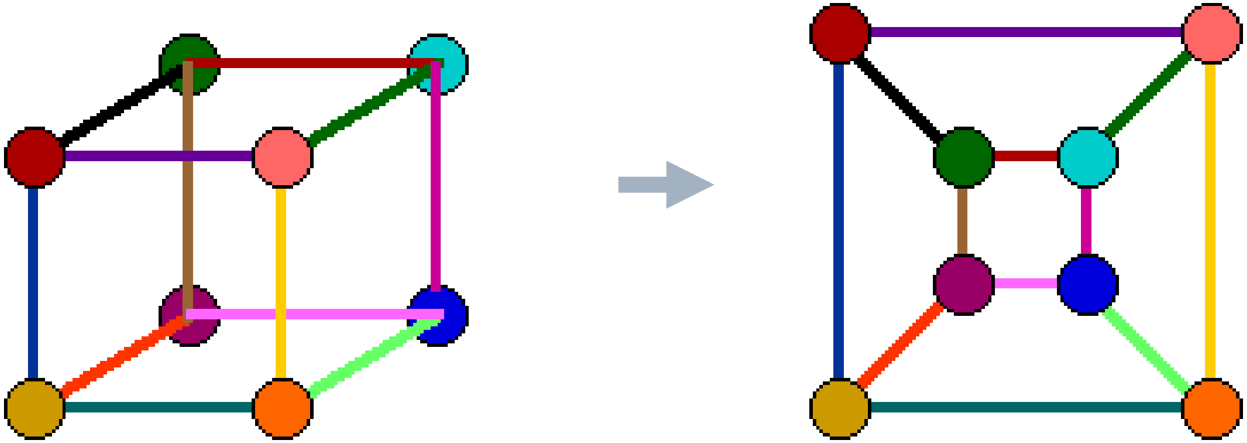
Examples of trees

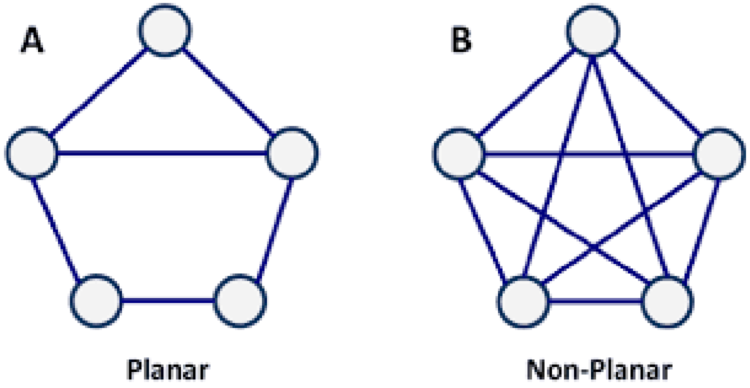
- In nature
  - trees
  - river networks
  - arteries (or veins, but not both)
- Man made
  - sewer system
- Computer science
  - binary search trees
  - decision trees (AI)
- Network analysis
  - minimum spanning trees
    - from one node – how to reach all other nodes most quickly
    - may not be unique, because shortest paths are not always unique
    - depends on weight of edges



Planar graphs

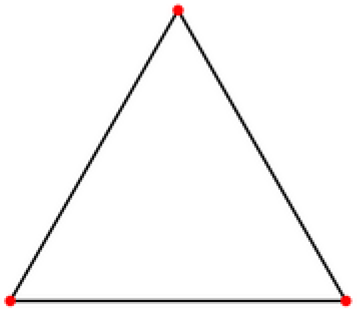
- A graph is planar if it can be drawn on a plane without any edges crossing



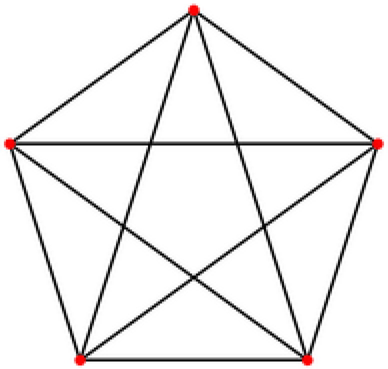


Cliques and complete graphs

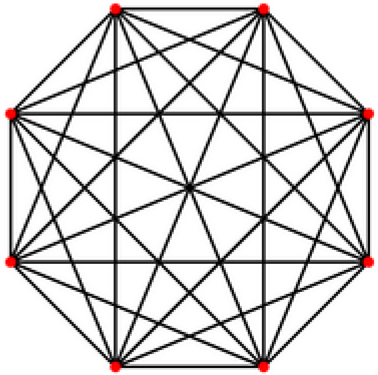
- K_n is the complete graph (clique) with K vertices
  - each vertex is connected to every other vertex
  - there are n*(n-1)/2 undirected edges



K_3



K_5



K_8

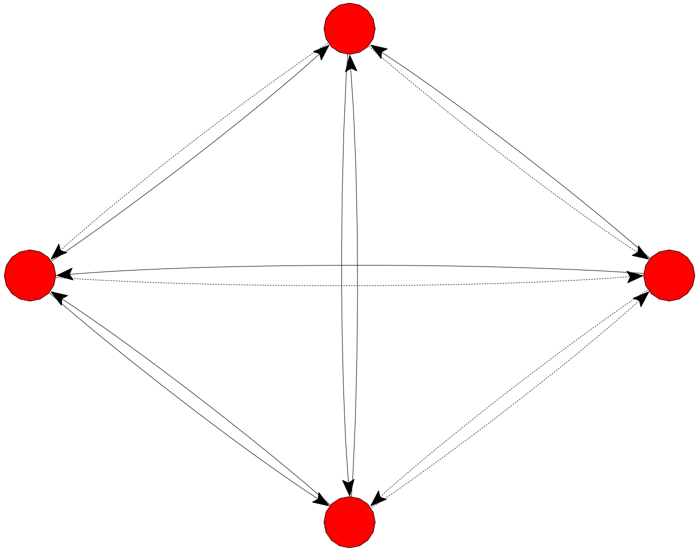
Network metrics: graph density

- Of the connections that may exist between n nodes
  - directed graph

    $e_{max} = n*(n-1)$

  - undirected graph

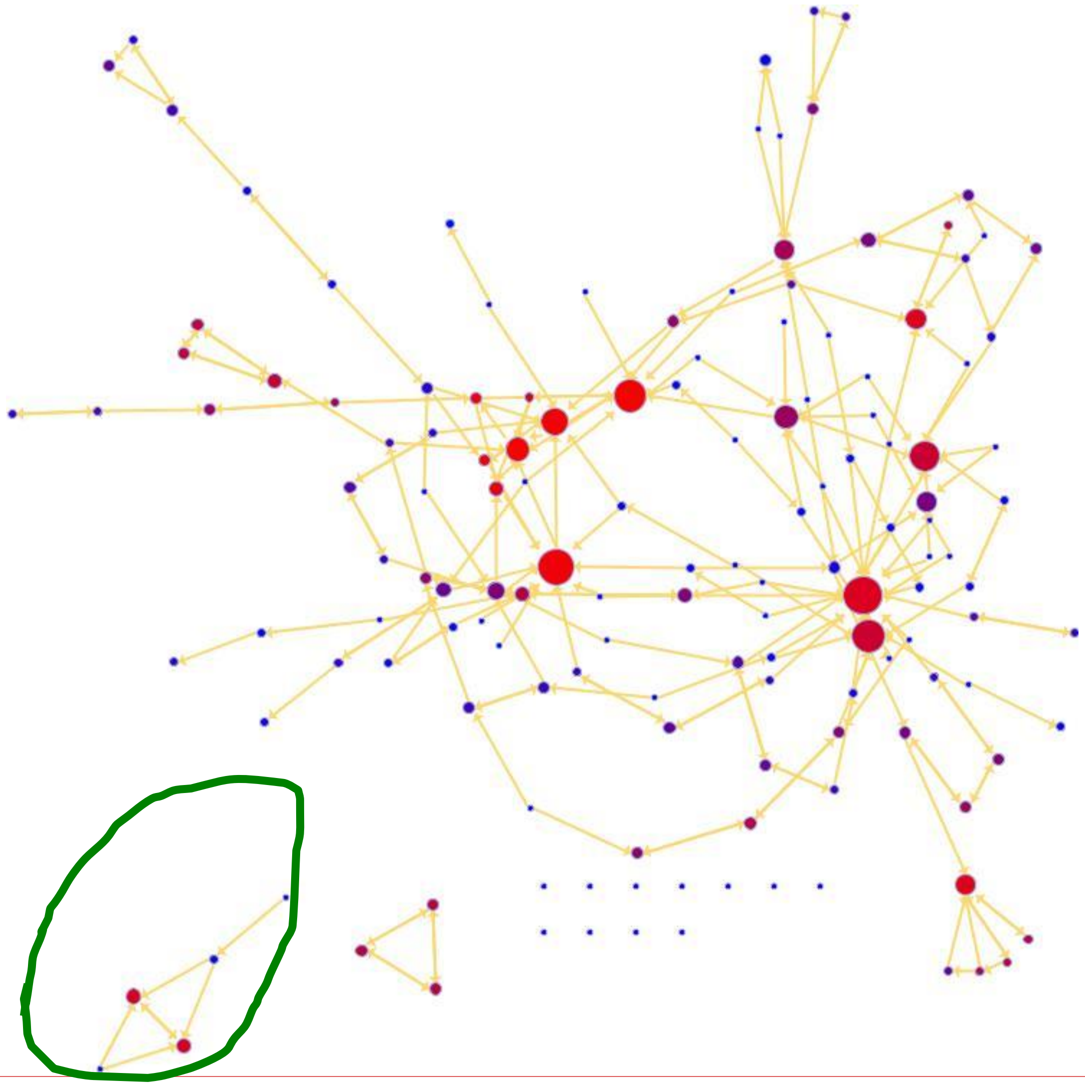
    $e_{max} = n*(n-1)/2$



- What fraction are present?
  - density = e / e_{max}
  - For example, out of 12 possible connections, this graph has 7, giving it a density of 7/12 = 0.583

CONNECTEDNESS

Characterizing networks:Is everything connected?



Network metrics: components

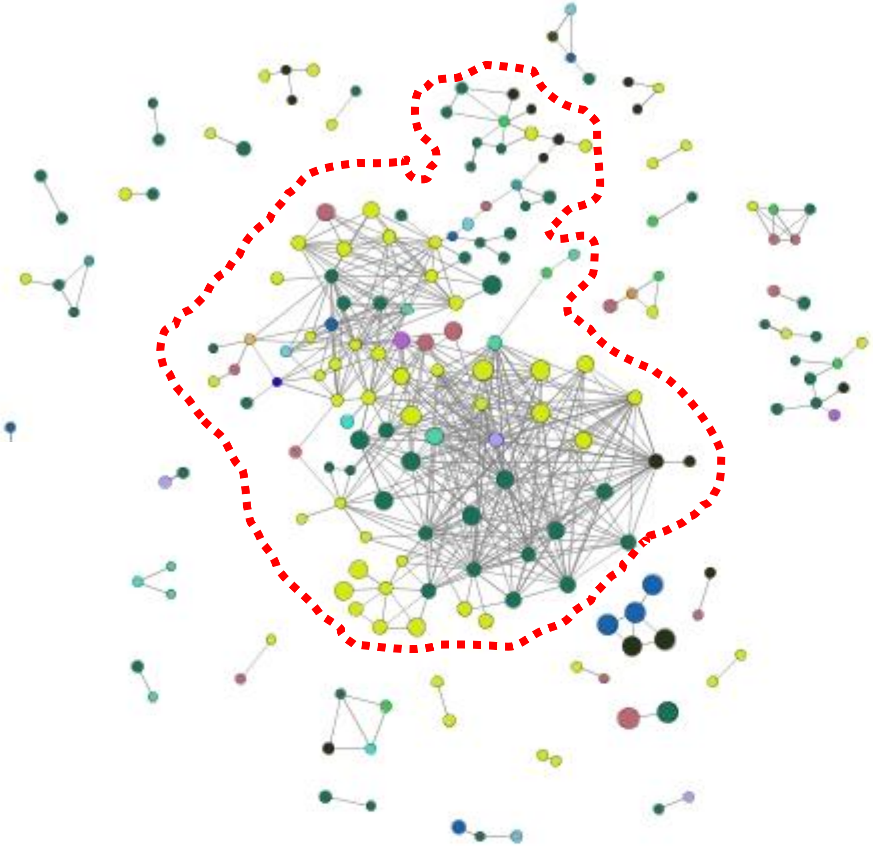
- If there is a path from every vertex in a network to every other, the network is ***connected***
  - otherwise, it is ***disconnected***

- **Component**: A subset of vertices such that there exist at least one path from each member of the subset to others and there does not exist another vertex in the network which is connected to any vertex in the subset
  - Maximal subset

- A singeleton vertex that is not connected to any other forms a size one component
- Every vertex belongs to exactly one component

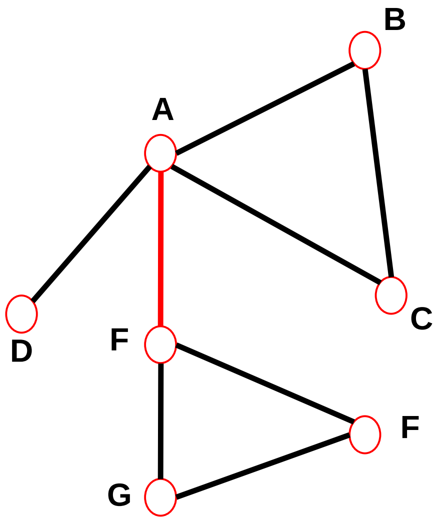
Network metrics: size of giant component

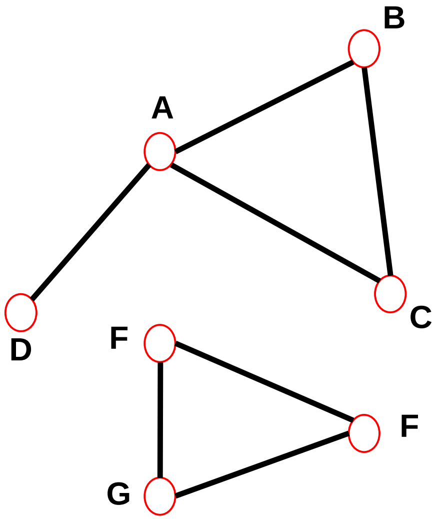
- if the largest component encompasses a significant fraction of the graph, it is called the **giant component**



Connectivity of Undirected Graphs

Connected (undirected) graph: any two vertices can be joined by a path.
A disconnected graph is made up by two or more connected components.





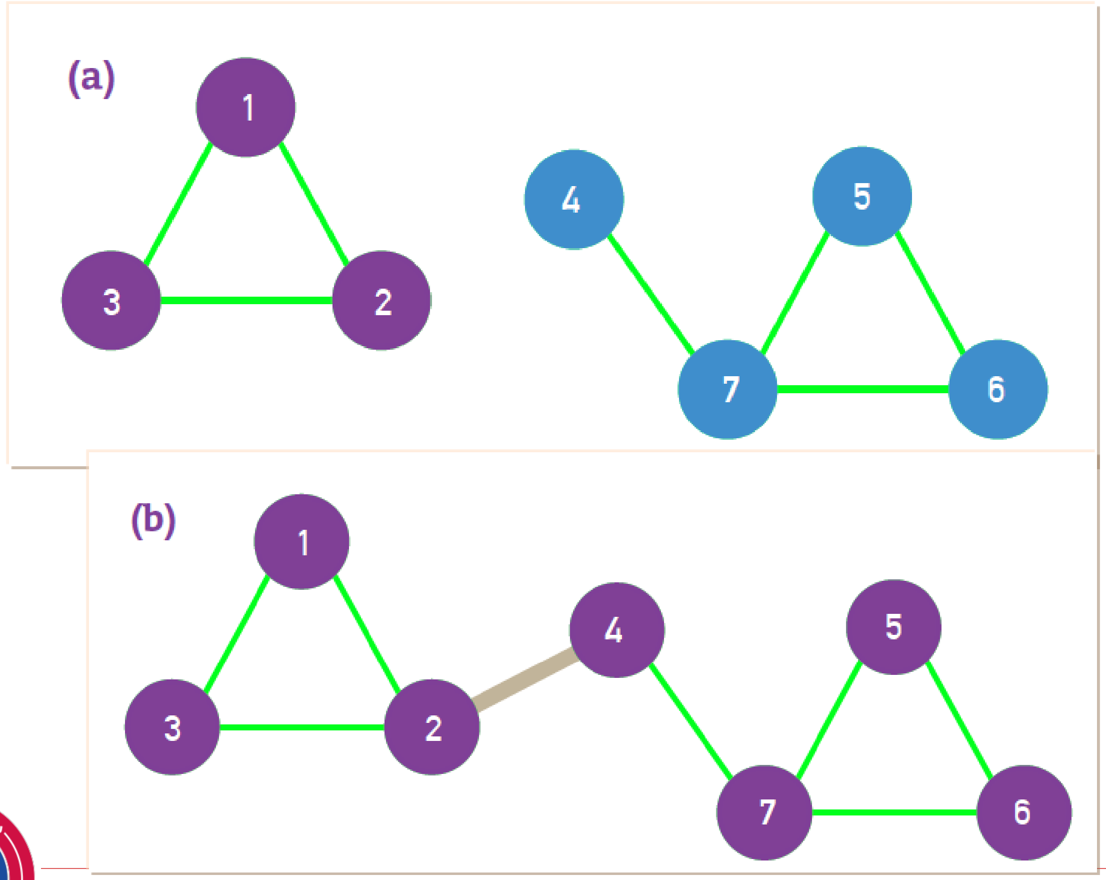
Largest Component:
Giant Component

The rest: Isolates

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

Connectivity of Undirected Graphs

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



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

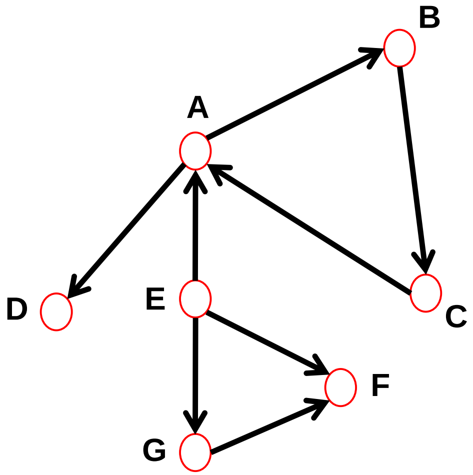
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

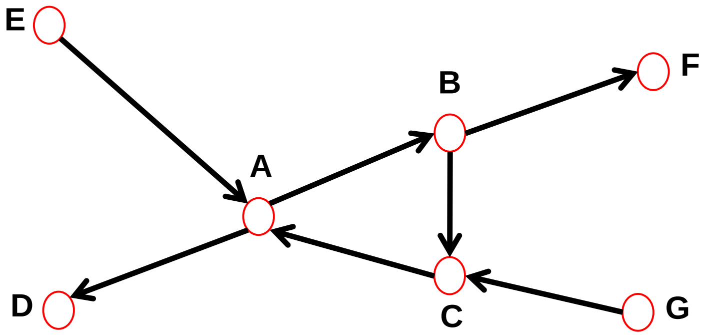
Connectivity of Directed Graphs

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

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

Strongly connected components can be identified, but not every node is part of a nontrivial strongly connected component.



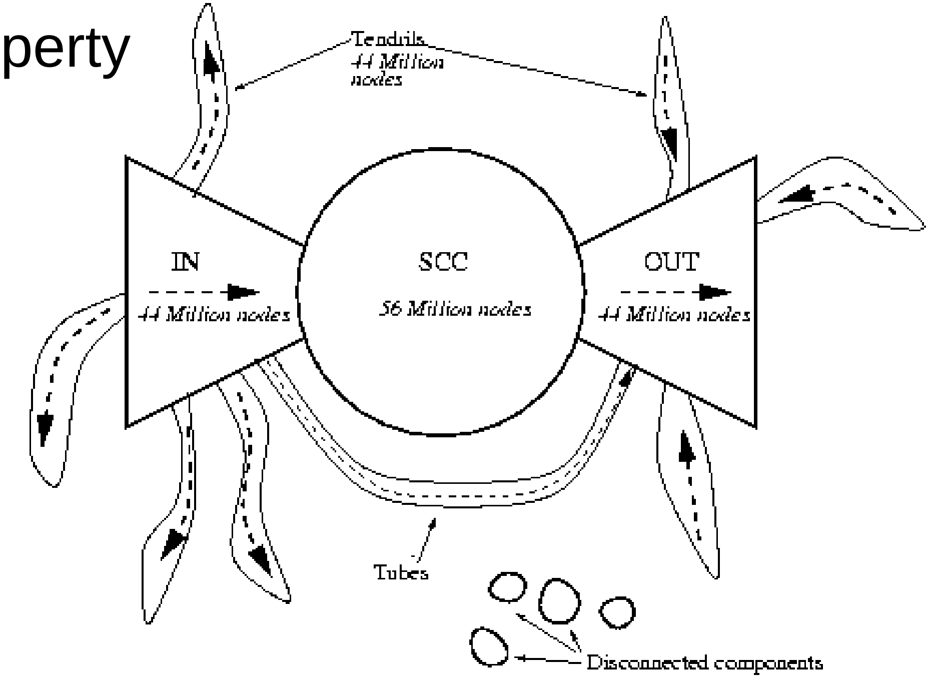


In-component: nodes that can reach the scc,

Out-component: nodes that can be reached from the scc.

Bow-Tie structure of the web

- Strongly Connected Component (SCC)
  - Core with small-world property
- Upstream (IN)
  - Core can’t reach IN
- Downstream (OUT)
  - OUT can’t reach core
- Disconnected components

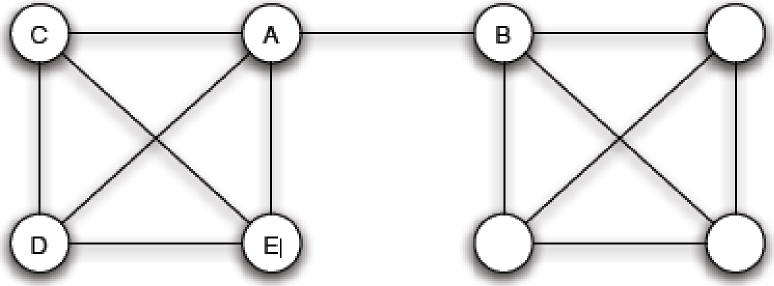


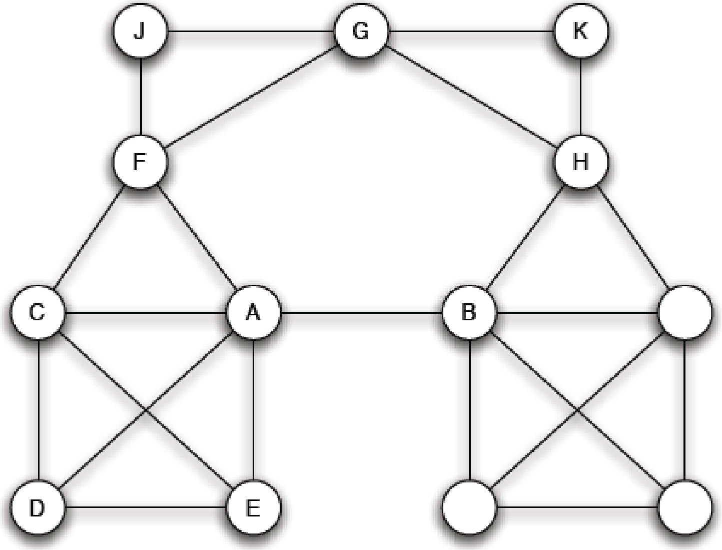
Broder et al. (Graph Structure of the Web, 2000)

Examined a large web graph (200M pages, 1.5B links)

Bridges and Local Bridges

- And edge that joins two nodes A and B in a graph is called a ***bridge*** if deleting the edge would cause A and B to lay in two different components
- ***local bridge*** - in real-world networks (with a giant component) - if deleting an edge between A and B would increase distance > 2



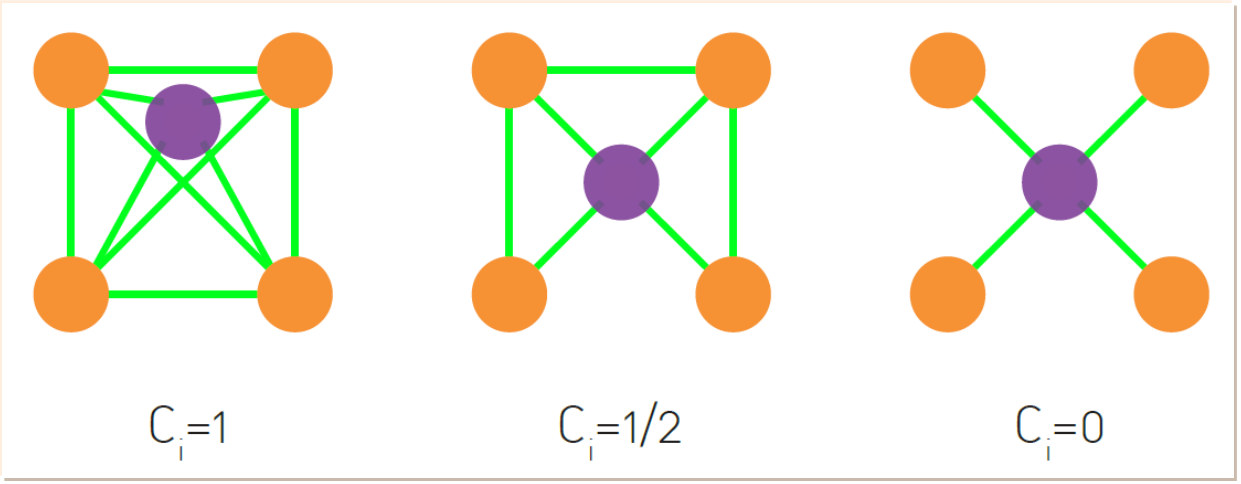


Clustering coefficient

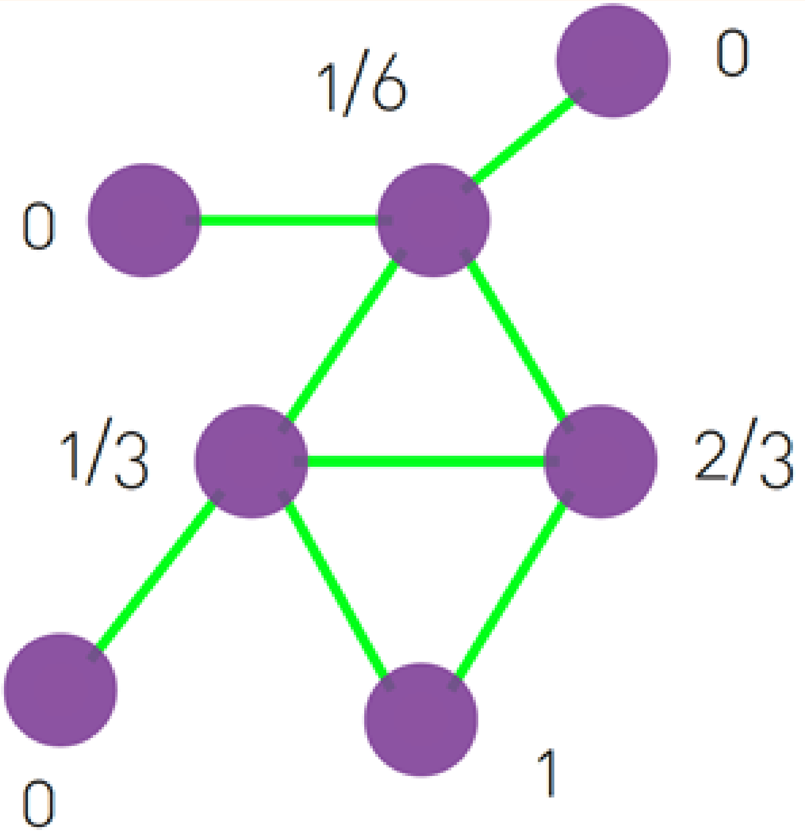
Clustering Coefficient

- what fraction of your neighbors are connected?
  - Node i with degree k_i
  - C_i in [0,1]

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



Clustering Coefficient

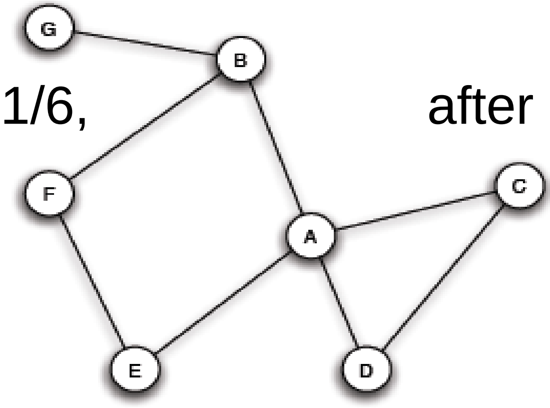


$$\langle C \rangle = \frac{13}{42} \approx 0.310$$

The Clustering Coefficiency

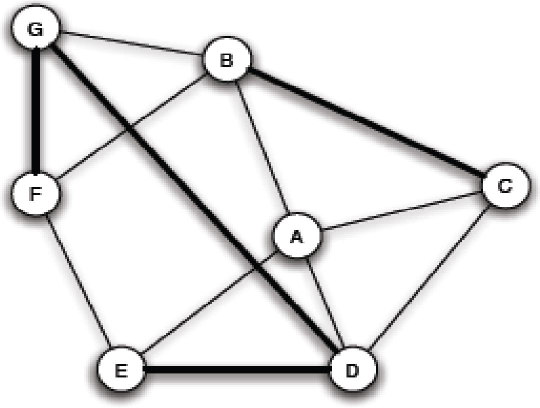
- *The clustering coefficiency* of a node A is a probability that two randomly selected friends of A are friends with each other
- Example: the clustering Coefficiency of ***A*** before and after the new edges?

before = 1/6,



(a) *Before new edges form.*

after = 1/2

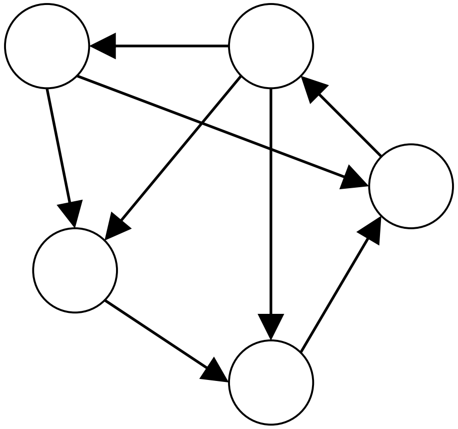


(b) *After new edges form.*

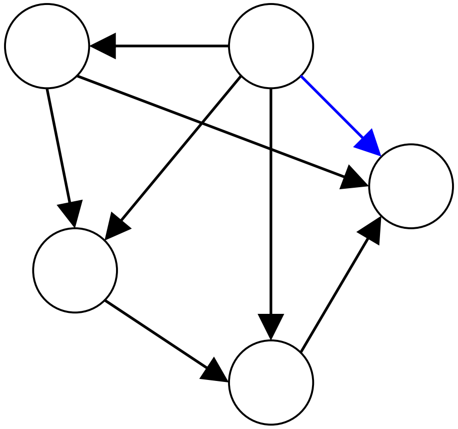
Strongly Connected Directed graphs

Every pair of vertices are reachable from each other

Strongly Connected

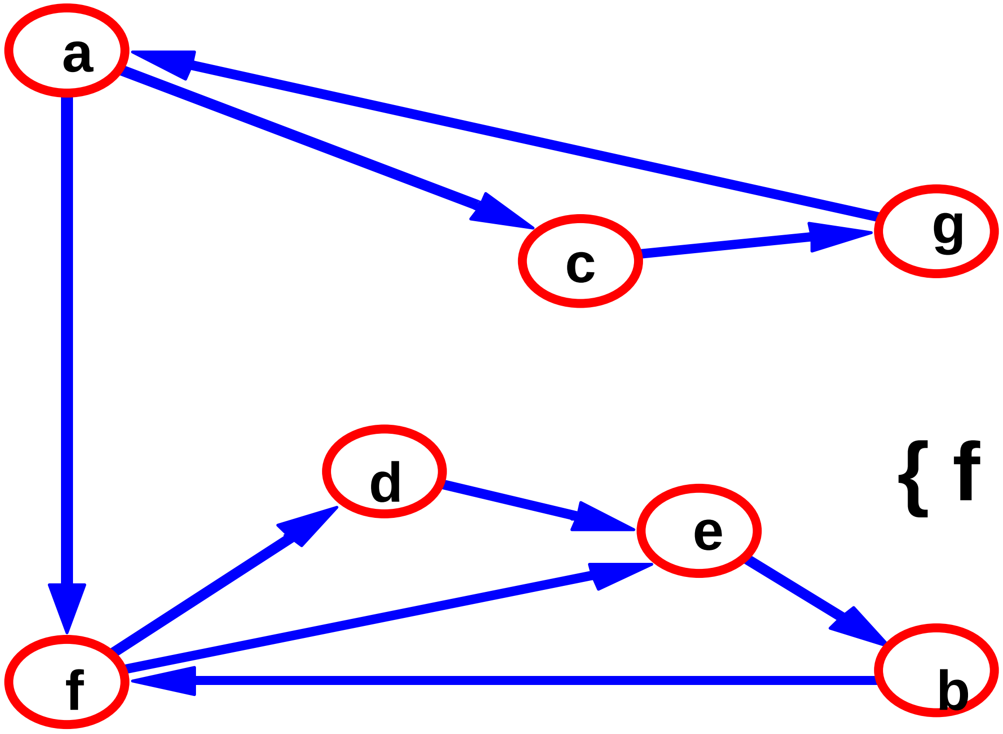


Not Strongly Connected



Strongly Connected Components

A strongly connected *component* of a graph is a maximal subset of nodes (along with their associated edges) that is strongly connected. Nodes share a strongly connected component if they are inter-reachable.



{ a , c , g }

{ f , d , e , b }

Graphology

Real networks can have multiple characteristics

WWW > directed multigraph with self-interactions

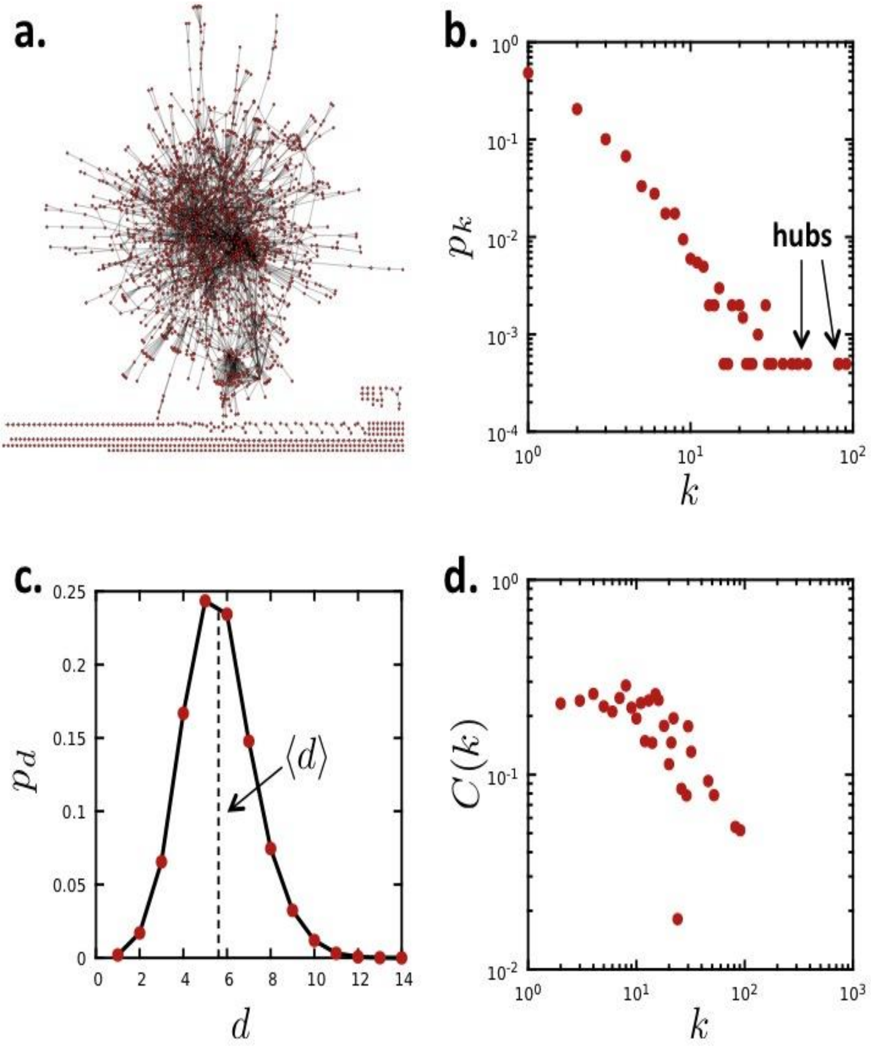
Protein Interactions > undirected unweighted with self-interactions

Collaboration network > undirected multigraph or weighted.

Mobile phone calls > directed, weighted.

Facebook Friendship links > undirected, unweighted.

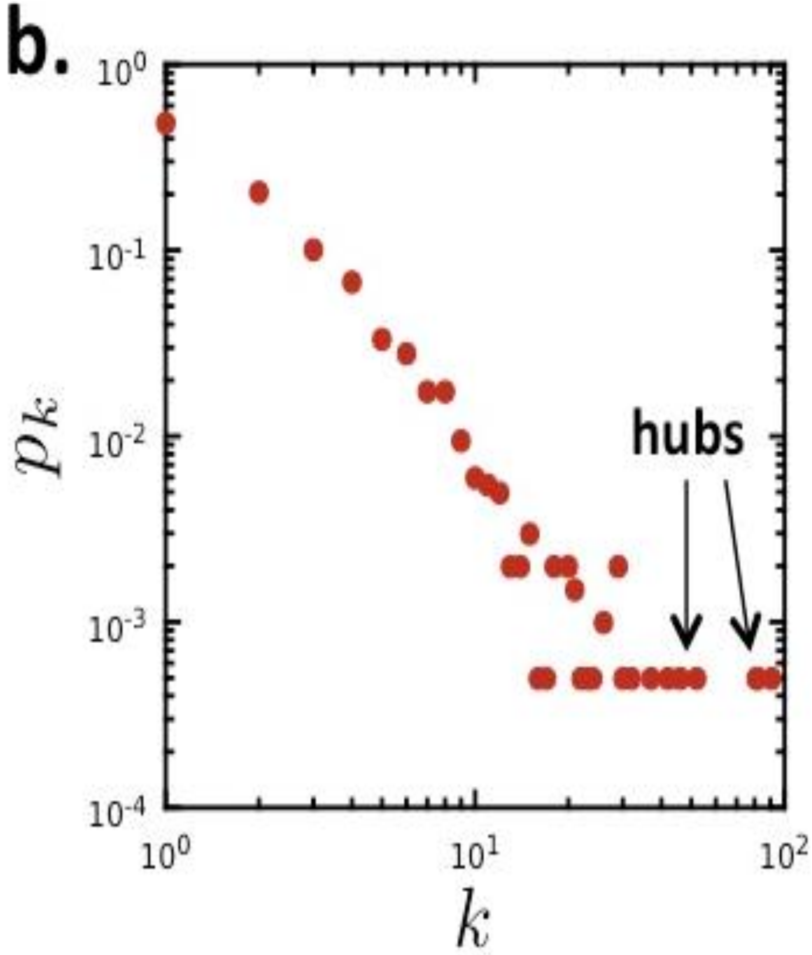
Case study: Protein-Protein Interactions



Undirected network
N=2,018 proteins as nodes
L=2,930 binding interactions as links.
Average degree <k>=2.90.

Not connected: 185 components
the largest (giant component) 1,647 nodes

Case study: Protein-Protein Interactions

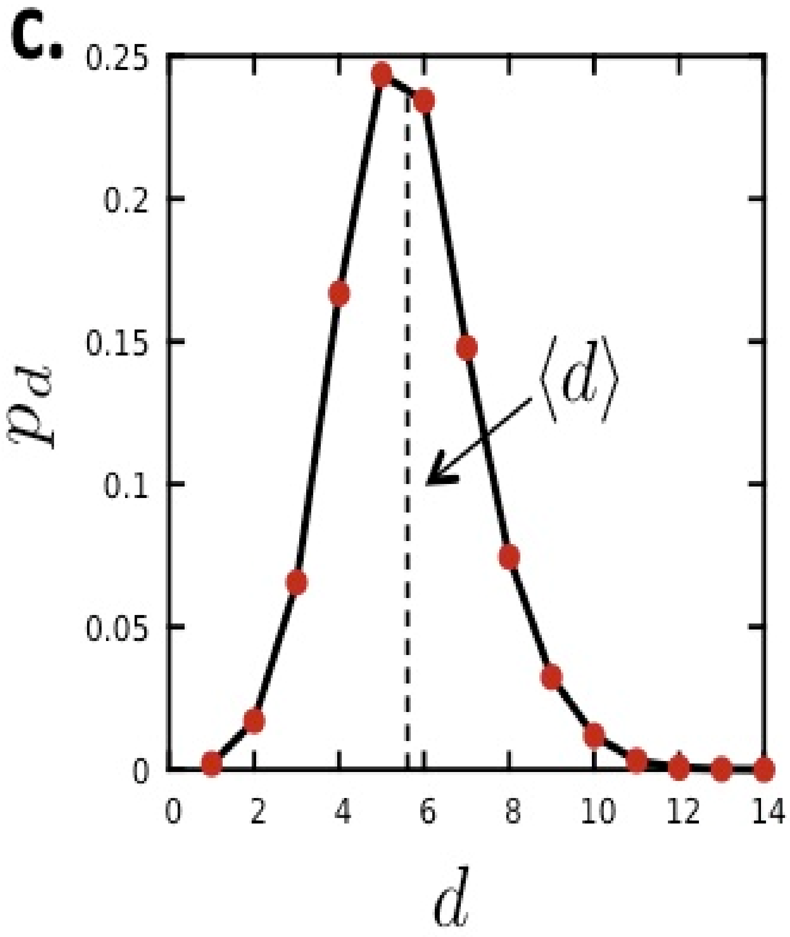


p_k is the probability that a node has degree k.

N_k = # nodes with degree k

$p_k = N_k / N$

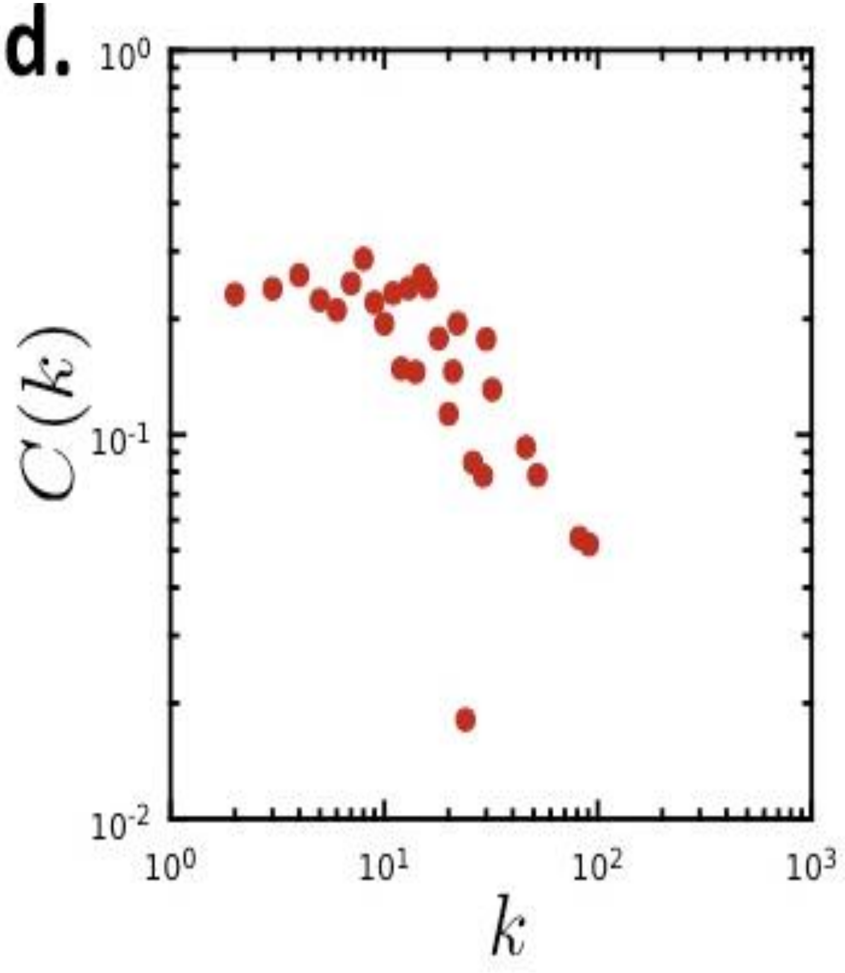
Case study: Protein-Protein Interactions



d_{max}=14

<d>=5.61

Case study: Protein-Protein Interactions



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

<C>=0.12

Real network properties

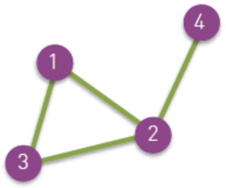
- Most nodes have only a small number of neighbors (degree), but there are some nodes with very high degree (power-law degree distribution)
  - scale-free networks
- If a node x is connected to y and z, then y and z are likely to be connected
  - high clustering coefficient
- Most nodes are just a few edges away on average.
  - small world networks
- Networks from very diverse areas (from internet to biological networks) have similar properties
  - Is it possible that there is a unifying underlying generative process?

Processes on networks

- Why is it important to understand the structure of networks?

- **Epidemiology**: Viruses propagate much faster in scale-free networks
- Vaccination of random nodes does not work, but targeted vaccination is very effective

(a) **Undirected**

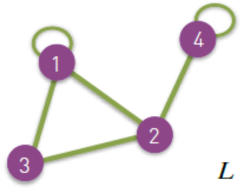


$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$A_{ii} = 0$ $\qquad$ $A_{ij} = A_{ji}$

$L = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij}$ $\qquad$ $< k >= \frac{2L}{N}$

(b) **Self-loops**

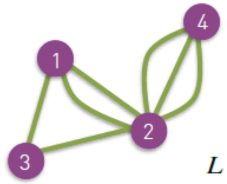


$$A_{ij} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

$\exists i, A_{ii} \neq 0$ $\qquad$ $A_{ij} = A_{ji}$

$L = \frac{1}{2}\sum_{i,j=1,i\neq j}^{N} A_{ij} + \sum_{i=1}^{N} A_{ii}$ $\qquad$?

(c) **Multigraph**
(undirected)

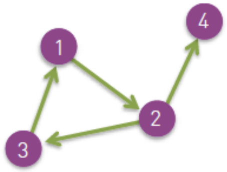


$$A_{ij} = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 3 \\ 1 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

$A_{ii} = 0$ $\qquad$ $A_{ij} = A_{ji}$

$L = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij}$ $\qquad$ $< k >= \frac{2L}{N}$

(d) **Directed**

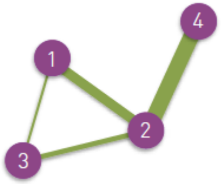


$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$A_{ij} \neq A_{ji}$

$L = \sum_{i,j=1}^{N} A_{ij}$ $\qquad$ $< k >= \frac{L}{N}$

(e) **Weighted**
(undirected)

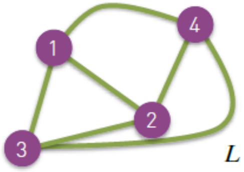


$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

$A_{ii} = 0$ $\qquad$ $A_{ij} = A_{ji}$

$< k >= \frac{2L}{N}$

(f) **Complete Graph**
(undirected)



$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$A_{ii} = 0$ $\qquad$ $A_{i\neq j} = 1$

$L = L_{\max} = \frac{N(N-1)}{2}$ $\qquad$ $< k >= N-1$

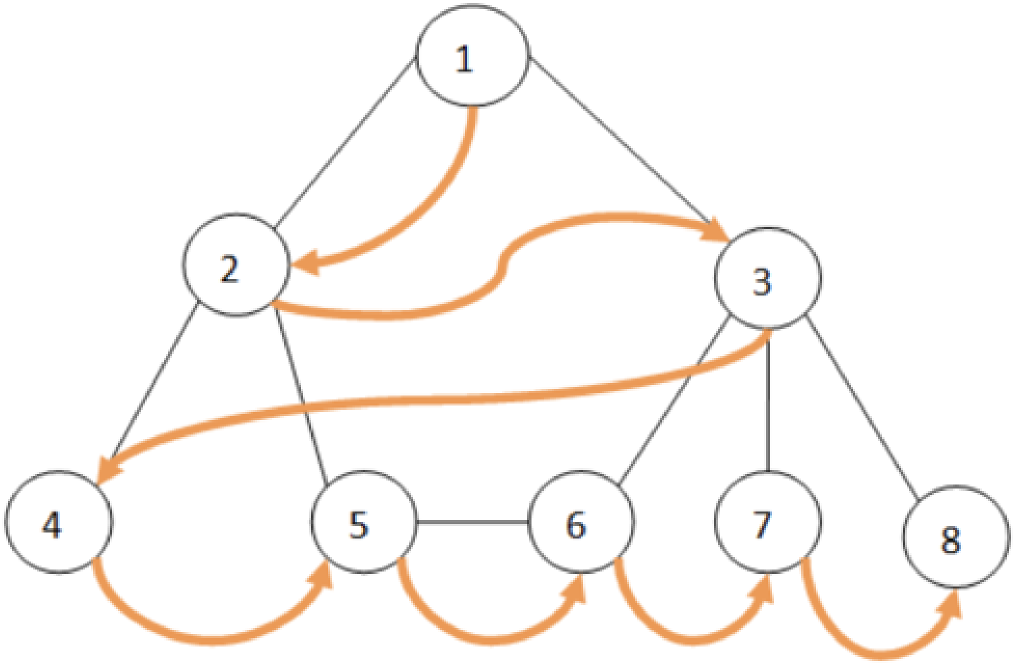
Graph Searching Algorithms

Graph-searching Algorithms

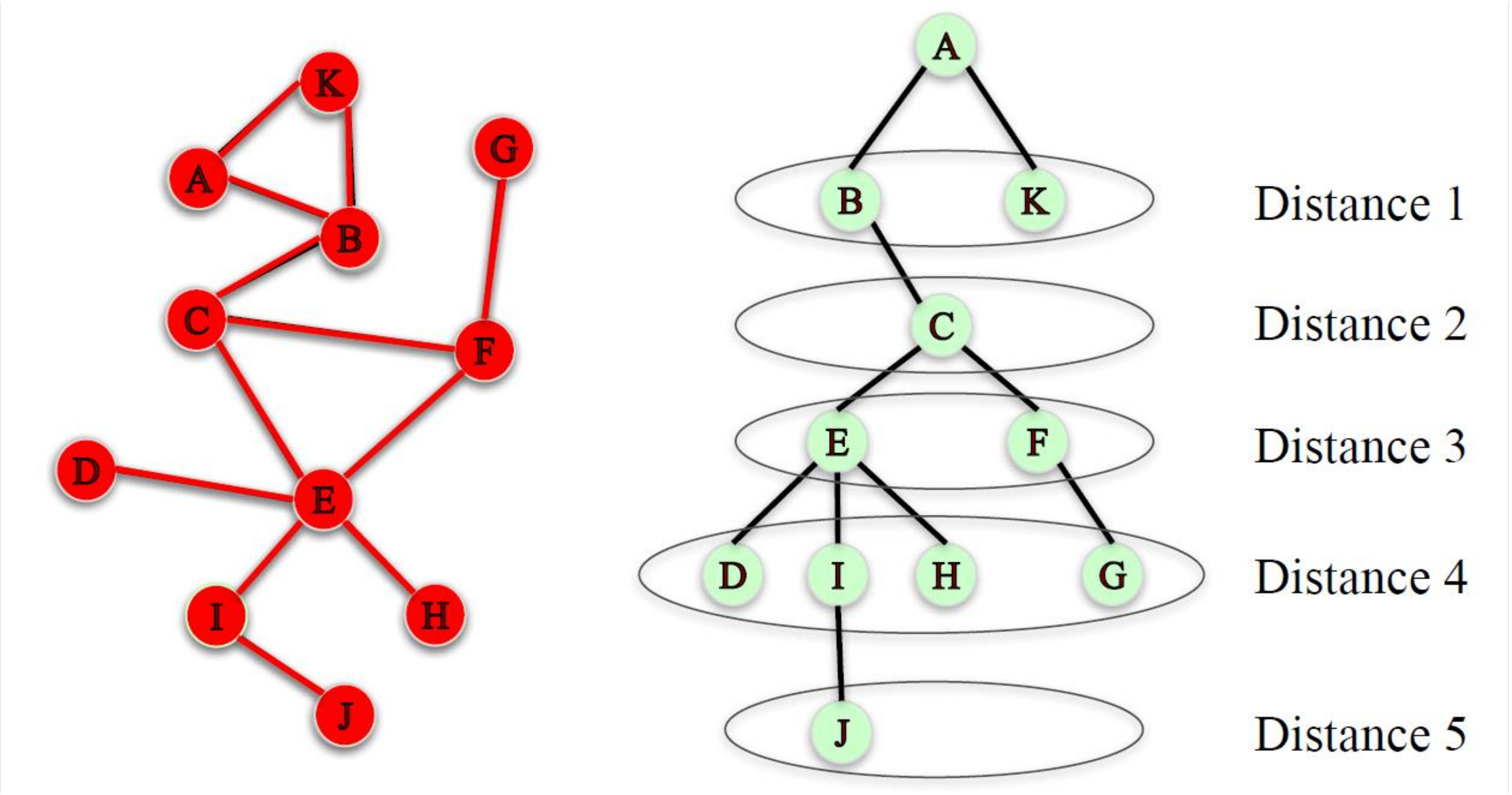
- Searching a graph:
  - Systematically follow the edges of a graph to visit the vertices of the graph.
- Used to discover the structure of a graph.
- Standard graph-searching algorithms.
  - Breadth-first Search (BFS).
  - Depth-first Search (DFS).

Breadth-first Search

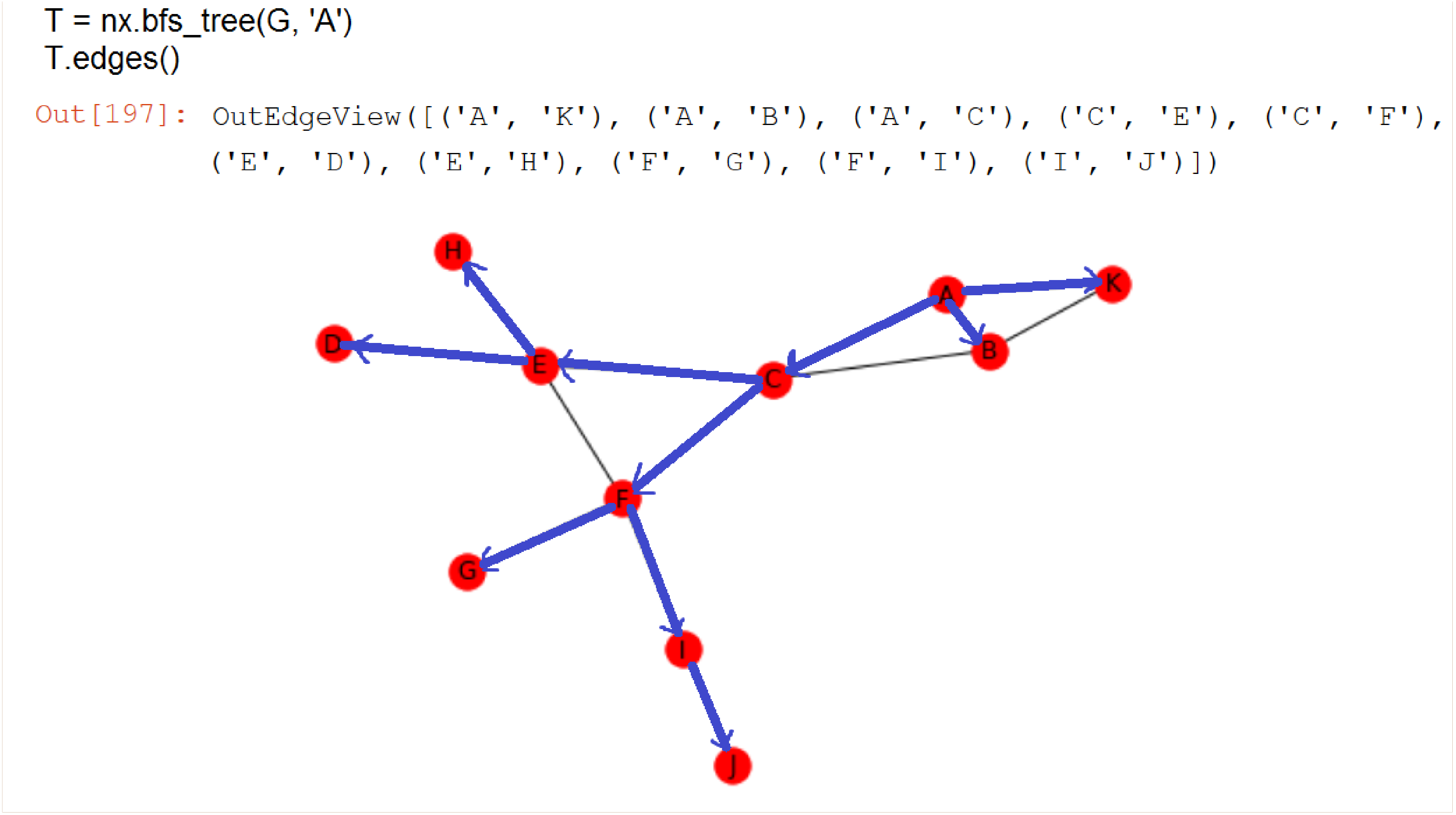
BFS tries to learn local neighbors first



Breadth-first Search



Breadth-first Search



```
T = nx.bfs_tree(G, 'A')
T.edges()
```

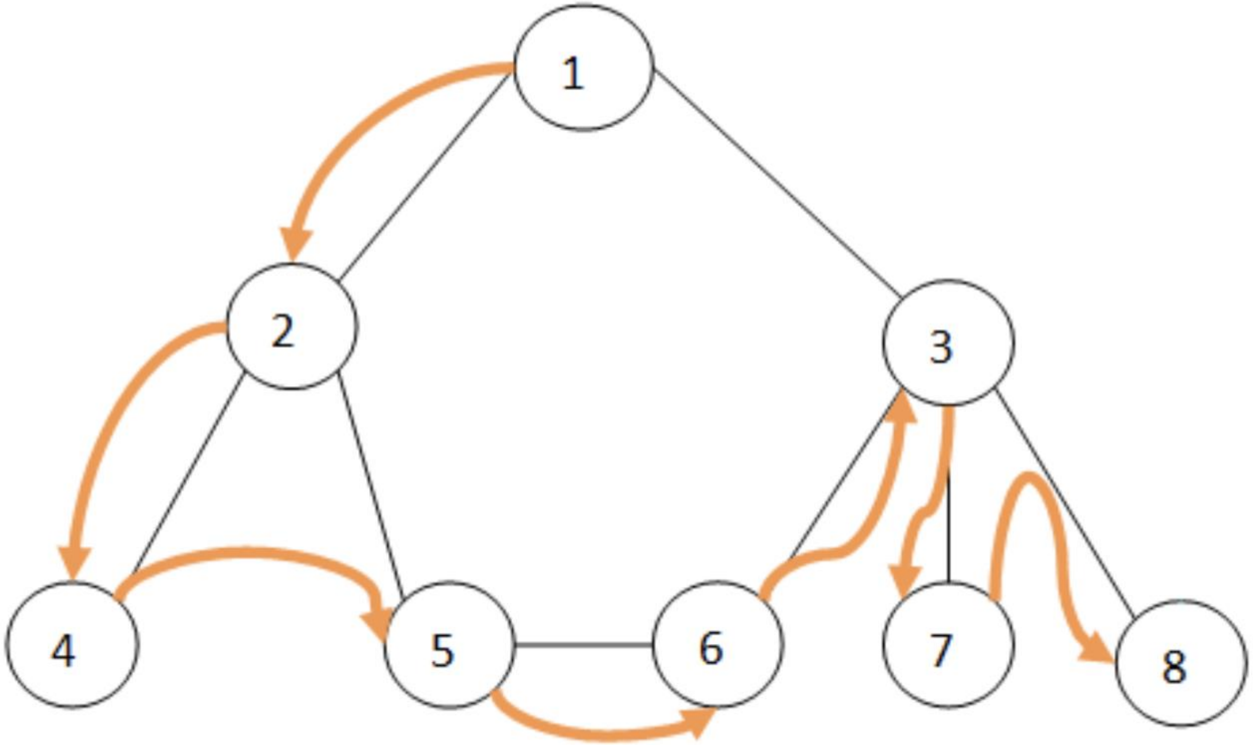
```
Out[197]: OutEdgeView([('A', 'K'), ('A', 'B'), ('A', 'C'), ('C', 'E'), ('C', 'F'),
          ('E', 'D'), ('E','H'), ('F', 'G'), ('F', 'I'), ('I', 'J')])
```

Depth-first Search (DFS)

- Explore edges out of the most recently discovered vertex *v*.
- When all edges of *v* have been explored, backtrack to explore other edges leaving the vertex from which *v* was discovered (its *predecessor*).
- “Search as deep as possible first.”
- Continue until all vertices reachable from the original source are discovered.
- If any undiscovered vertices remain, then one of them is chosen as a new source and search is repeated from that source.

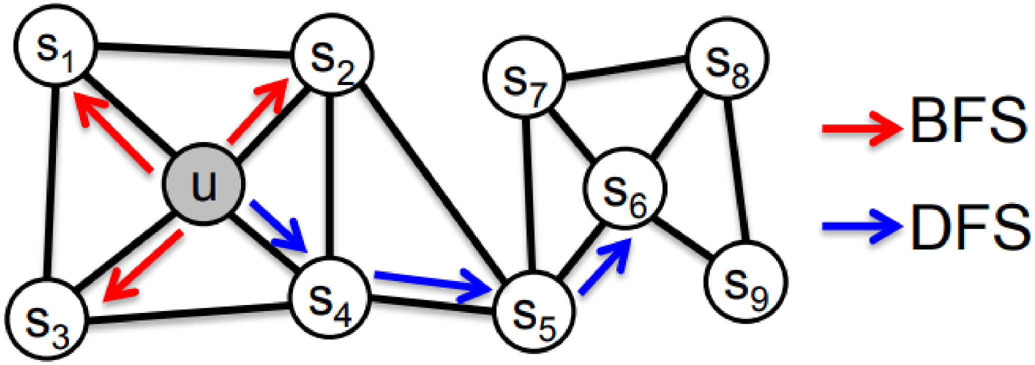
Depth-first Search

DFS is better for learning global variables



BFS and DFS for node embedding

Node2vec (node embedding using biased random walk)



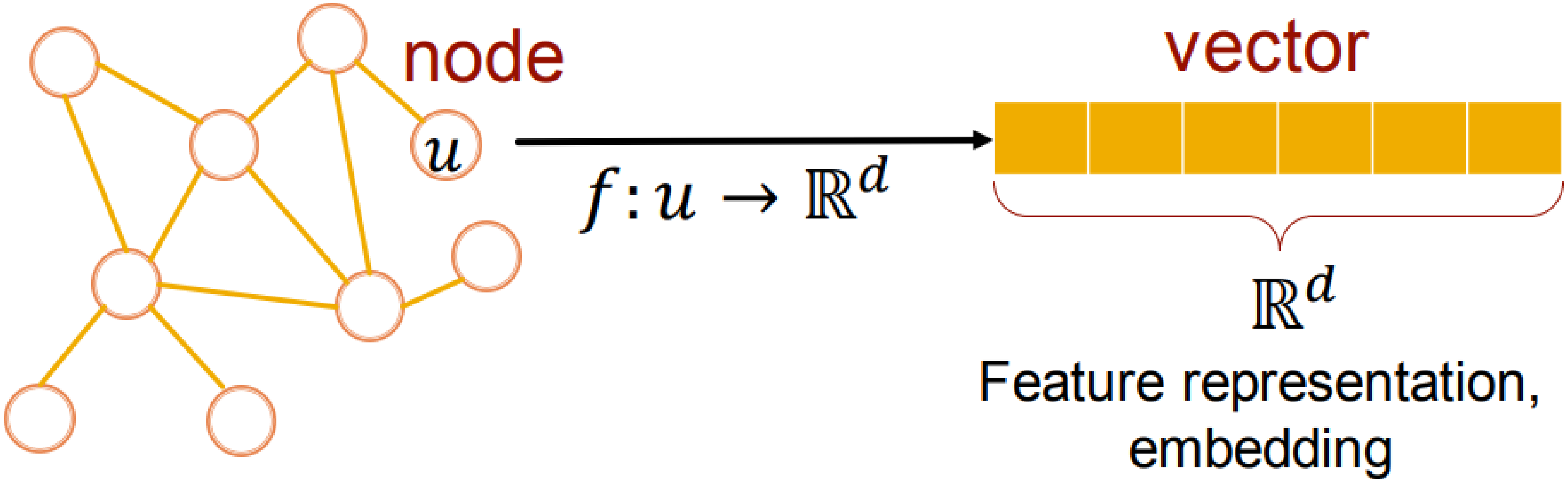
Walk of length 3 ($N_R(u)$ of size 3):

$N_{BFS}(u) = \{\, s_1, s_2, s_3\}$ Local microscopic view

$N_{DFS}(u) = \{\, s_4, s_5, s_6\}$ Global macroscopic view

Trade off between **local** and **global** views of the network

BFS and DFS for node embedding



$f : u \rightarrow \mathbb{R}^d$

$\mathbb{R}^d$

Feature representation, embedding

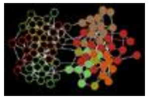
Applications:

- Node classification
- Link prediction
- Graph classification
- Anomalous node detection
- Clustering

Network Analysis

- What is a network?
  - a bunch of nodes and edges
- How do you characterize it?
  - with some basic network metrics
- How did network analysis get started?
  - it was the mathematicians
- How do you analyze networks today?
  - with network analysis tools (pajek, Gephi, …)

Some network analysis tools

	Pajek	network analysis and visualization, menu driven, suitable for large networks	platforms: Windows (on linux via Wine) download
	**Netlogo**	agent based modeling recently added network modeling capabilities	platforms: any (Java) download
	GUESS	network analysis and visualization, extensible, script-driven (jython)	platforms: any (Java) download

Other useful software tools:

visualization and analysis:

UCInet - user friendly social network visualization and analysis software (suitable smaller networks)

iGraph - if you are familiar with R, you can use iGraph as a module to analyze or create large networks, or you can directly use the C functions

Jung - comprehensive Java library of network analysis, creation and visualization routines

Graph package for Matlab (untested?) - if Matlab is the environment you are most comfortable in, here are some basic routines

SIENA - for p* models and longitudinal analysis

SNA package for R - all sorts of analysis + heavy duty stats to boot

NetworkX - python based free package for analysis of large graphs

InfoVis Cyberinfrastructure - large agglomeration of network analysis tools/routines, partly menu driven

visualization only:

GraphViz - open source network visualization software (can handle large/specialized networks)

TouchGraph - need to quickly create an interactive visualization for the web?

yEd - free, graph visualization and *editing* software

specialized:

fast community finding algorithm

motif profiles

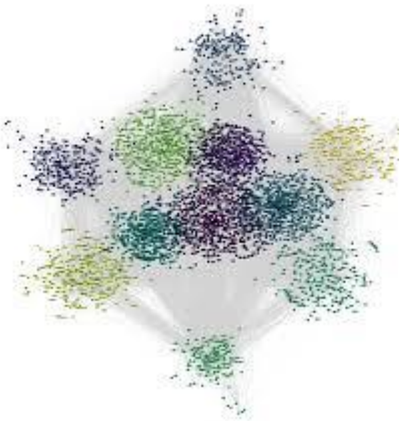
CLAIR library - NLP and IR library (Perl Based) includes network analysis routines

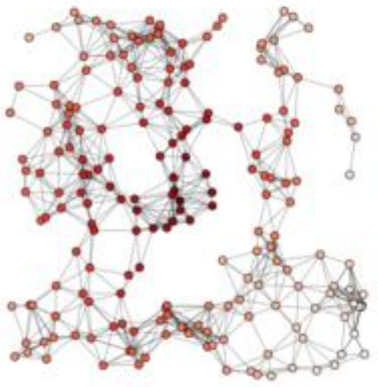
finally: INSNA long list of SNA packages

NetworkX

- NetworkX (http://networkx.lanl.gov/)
  - It is a **Python** language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

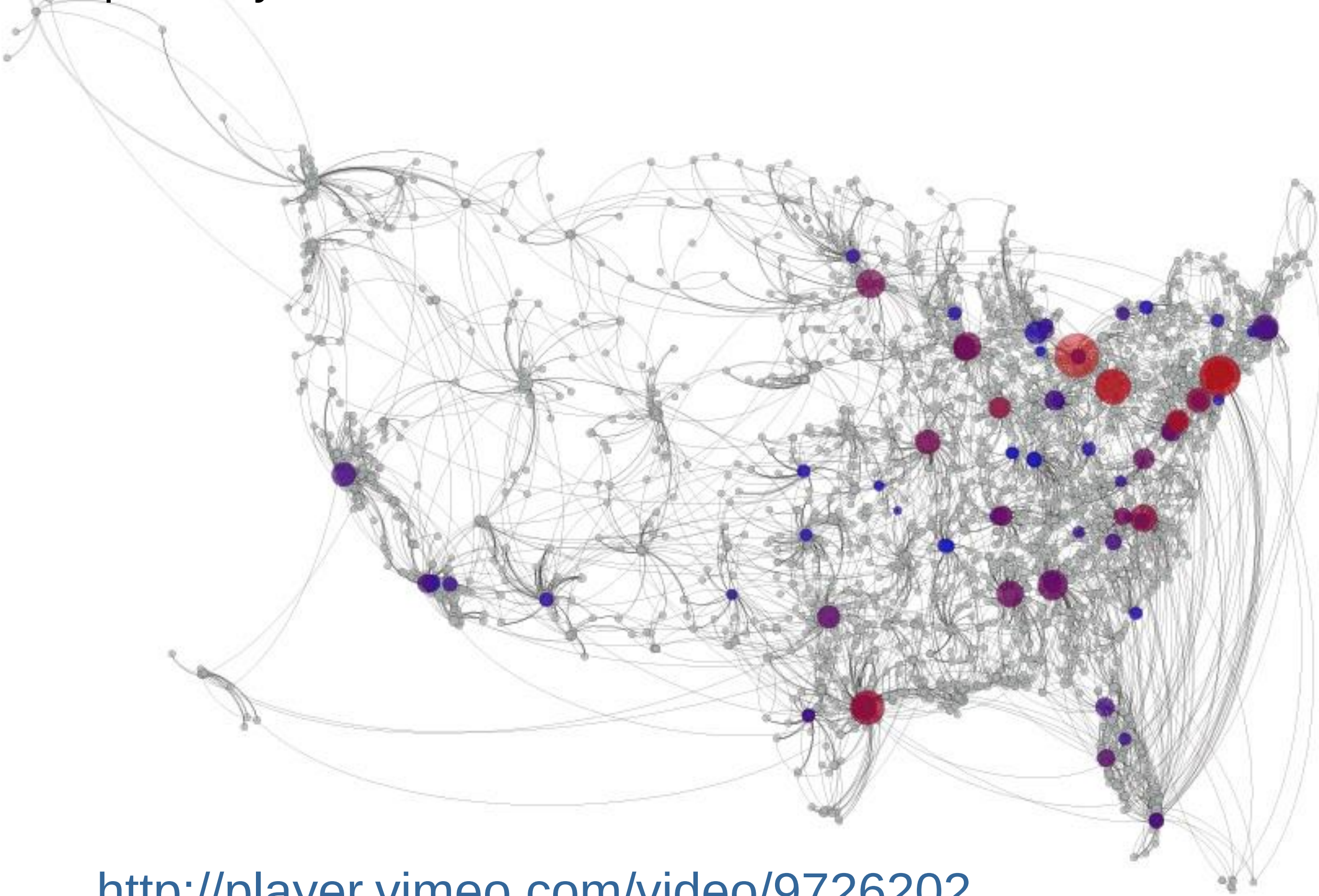
- Just run:
  - easy_install networkx





Other visualization tool: gephi

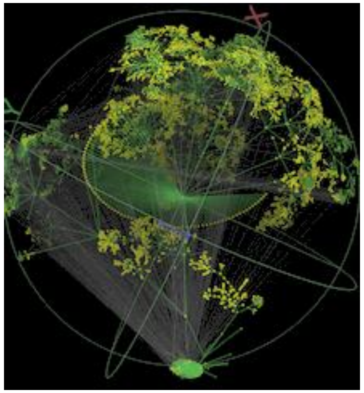
- http://gephi.org
  - primarily for visualization, has some nice touches



http://player.vimeo.com/video/9726202

Other visualization tool: Walrus

- developed at CAIDA available under the GNU GPL.
- "…best suited to visualizing moderately sized graphs that are nearly trees. A graph with a few hundred thousand nodes and only a slightly greater number of links is likely to be comfortable to work with."
- Java-based

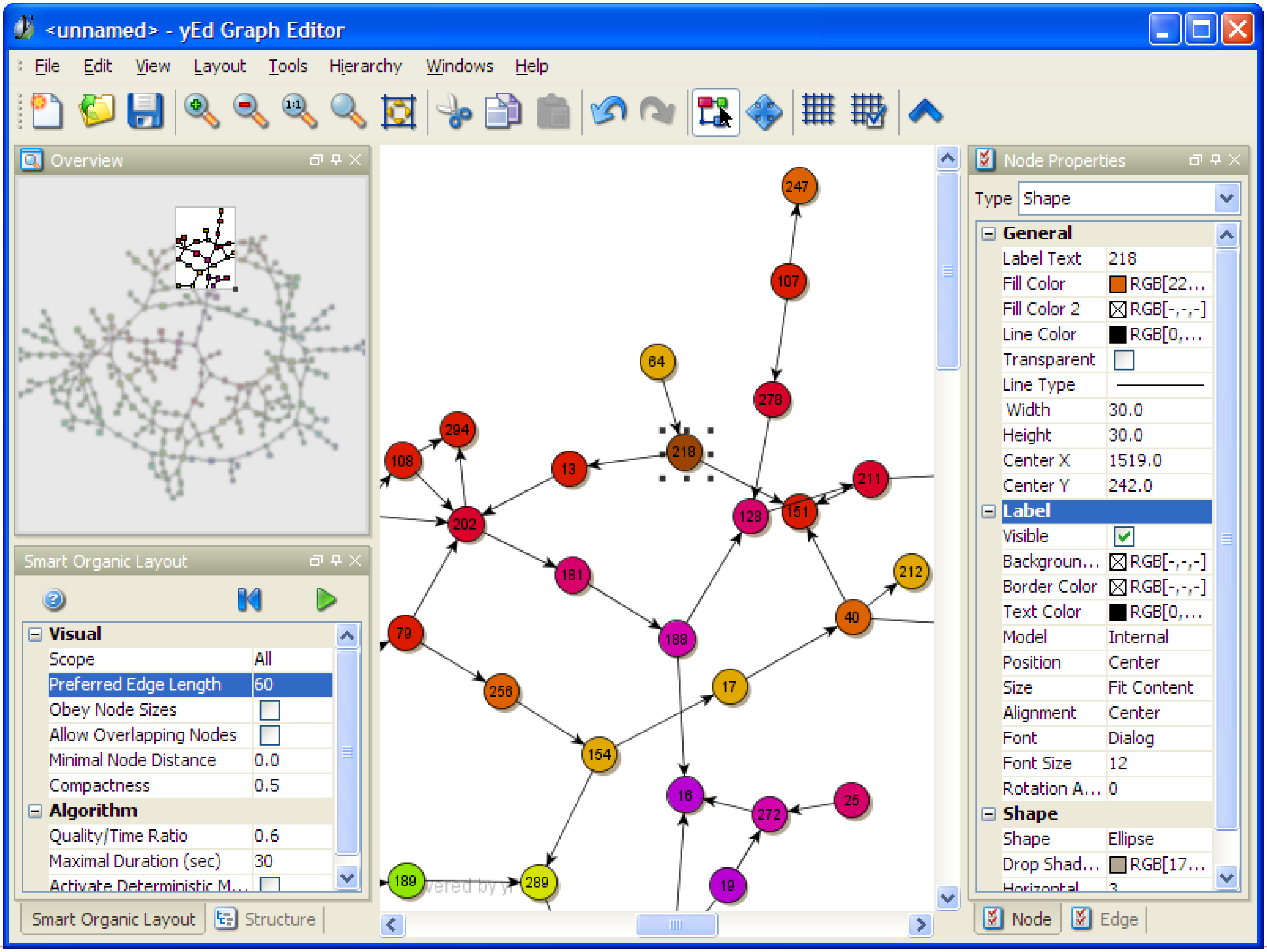


- Implemented Features
  - rendering at a guaranteed frame rate regardless of graph size
  - coloring nodes and links with a fixed color, or by RGB values stored in attributes
  - labeling nodes
  - picking nodes to examine attribute values
  - displaying a subset of nodes or links based on a user-supplied boolean attribute
  - interactive pruning of the graph to temporarily reduce clutter and occlusion
  - zooming in and out

Source: CAIDA, http://www.caida.org/tools/visualization/walrus/

Other visualization tool: YEd

http://www.yworks.com/en/products_yed_about.htm
(good primarily for layouts)



Smart Organic Layout

Visual	
Scope	All
Preferred Edge Length	60
Obey Node Sizes	□
Allow Overlapping Nodes	□
Minimal Node Distance	0.0
Compactness	0.5
Algorithm	
Quality/Time Ratio	0.6
Maximal Duration (sec)	30
Activate Deterministic M...	□

Node Properties

Type Shape

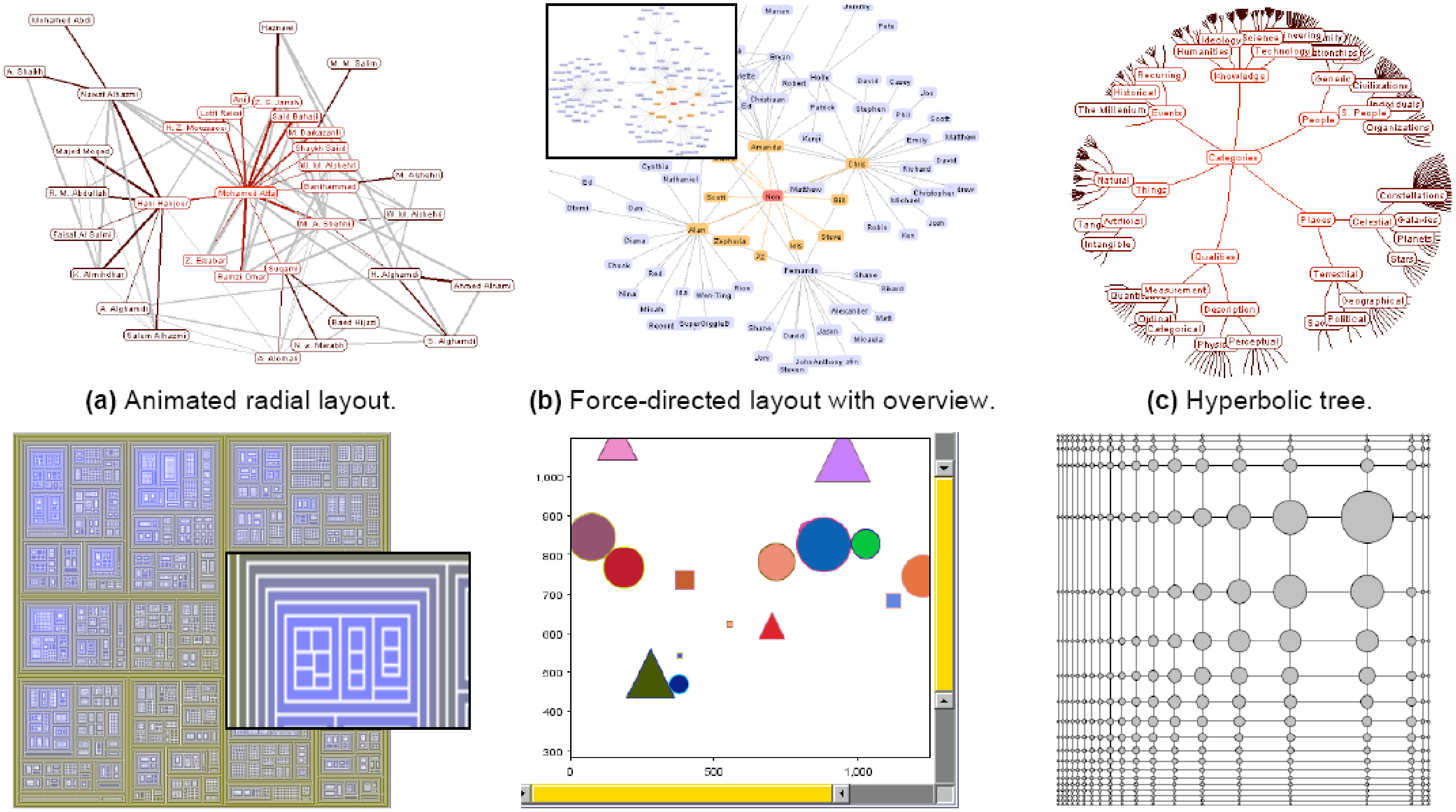
General	
Label Text	218
Fill Color	■ RGB[22...
Fill Color 2	☒ RGB[-,-,-]
Line Color	■ RGB[0,...
Transparent	□
Line Type	———
Width	30.0
Height	30.0
Center X	1519.0
Center Y	242.0
Label	
Visible	☑
Backgroun...	☒ RGB[-,-,-]
Border Color	☒ RGB[-,-,-]
Text Color	■ RGB[0,...
Model	Internal
Position	Center
Size	Fit Content
Alignment	Center
Font	Dialog
Font Size	12
Rotation A...	0
Shape	
Shape	Ellipse
Drop Shad...	■ RGB[17...
Horizontal...	3

Other Visualization tool: Prefuse

- user interface toolkit for interactive information visualization
  - built in Java using Java2D graphics library
  - data structures and algorithms
  - pipeline architecture featuring reusable, composable modules
  - animation and rendering support
  - architectural techniques for scalability

- requires knowledge of Java programming
- website: http://prefuse.sourceforge.net

Simple Prefuse visualizations



(a) Animated radial layout.

(b) Force-directed layout with overview.

(c) Hyperbolic tree.

(d) TreeMap.

(e) SpotPlot scatterplot.

(f) Fisheye graph.

(g) Fisheye menu.

Source: Prefuse, https://github.com/prefuse

Questions