Graph Theory

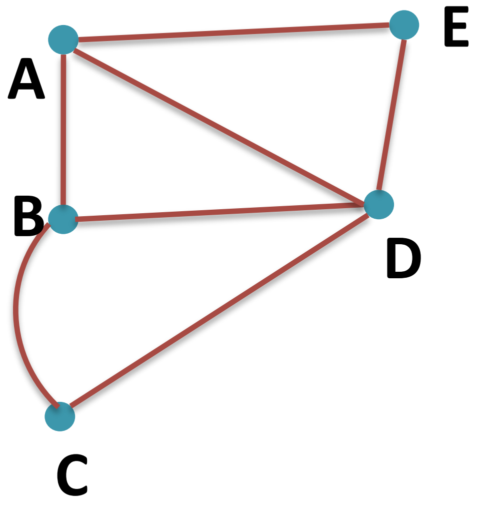
Lecture 2

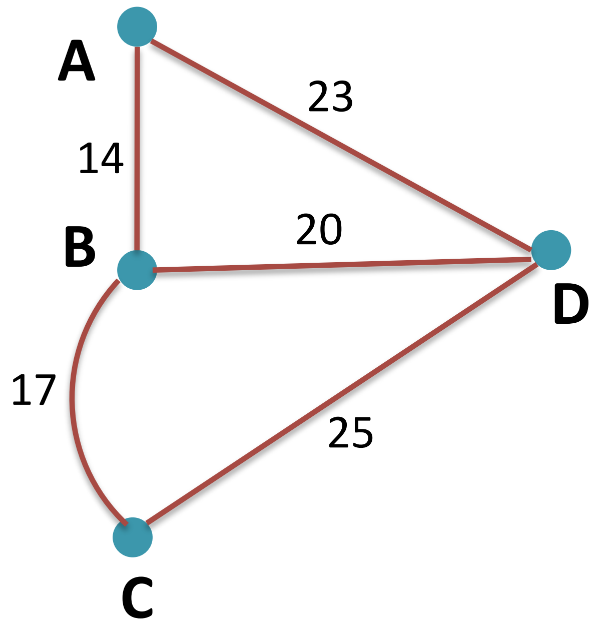
Graph Theory

- Some basic concepts of Graph theory
- Some examples of Special graphs
- Graph operations
- Graph paths and cycles
- Graph connectivity
- Tree and Bipartite graph
- Community structure in Networks
- Network model
- Network motifs

Shortest path

Between two nodes, there are multiple paths. The path having the shortest length is called the shortest path.



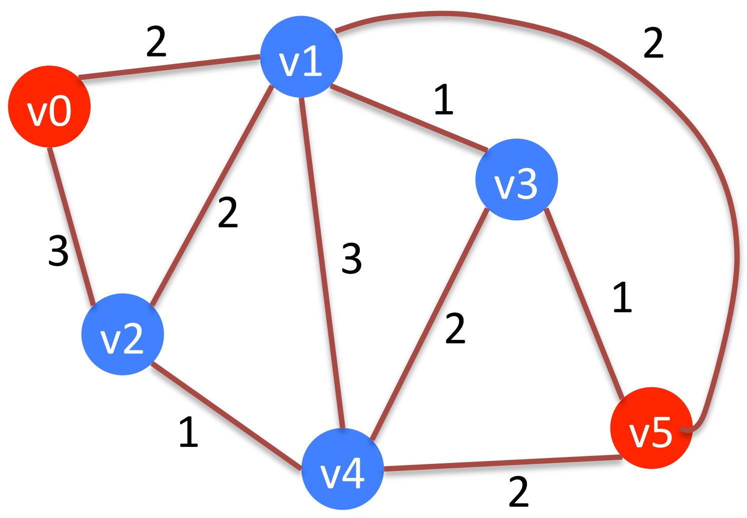


A – B – C
A – E – D – C

A – B – C : 14+7
A – D – C : 23+25

Distances between nodes

- The distance between two nodes is defined as the length of the shortest path.
- If the two nodes are disconnected, the distance is infinity.



Distances between nodes

- In digraphs, 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 BA path).

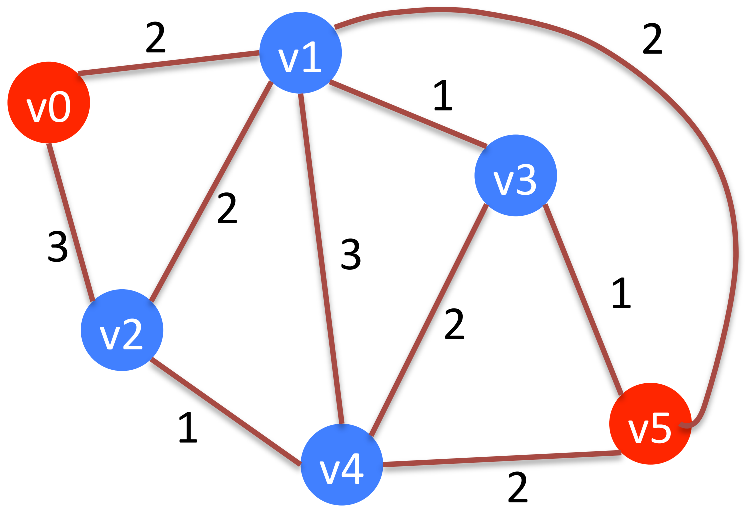
Distance matrix

- Q: How many entries will you need for an N-node graph?
- A: N(N-1) in a digraph, N(N-1)/2 in a symmetrical graph.

$$N_{pairs} = \binom{N}{2} = \frac{N(N-1)}{2}$$

Diameter

- Graph diameter: the maximum distance between any pair of nodes in the graph.
- Note: not the longest path.



Average distance

- Average distance for a connected graph (component).

$$\langle l \rangle \equiv \frac{1}{2N_{pairs}} \sum_{i,j\neq i} l_{ij}$$

, where l_{ij} is the distance from node i to node j, $N_{pairs} = \binom{N}{2} = \frac{N(N-1)}{2}$ and N is the number of nodes in the graph or component.

Since in a (symmetrical) graph $l_{ij} = l_{ji,}$ we only need to count them once $\langle l \rangle \equiv \frac{1}{N_{pairs}} \sum_{i,j>i} l_{ij}$

Betweenness centrality (load)

- For all node pairs (*i, j):*
  - Find all the shortest paths between nodes *i* and *j* - C(*i*,*j*)
  - Determine how many of these pass through node k - $C_k(i,j)$
- The betweenness centrality of node k is

$$g_k = \sum_{i \neq j} \frac{C_k(i,j)}{C(i,j)}$$

Graph efficiency

- To avoid infinity distance in graphs that are not connected and digraphs that are not strongly connected, one can define a graph efficiency (= average inverse distance)

$$\langle l \rangle = \frac{1}{2N_{\mathbf{paris}}} \sum_{i,j \neq i} \frac{1}{l_{ij}}$$

N_pairs is the number of node pairs

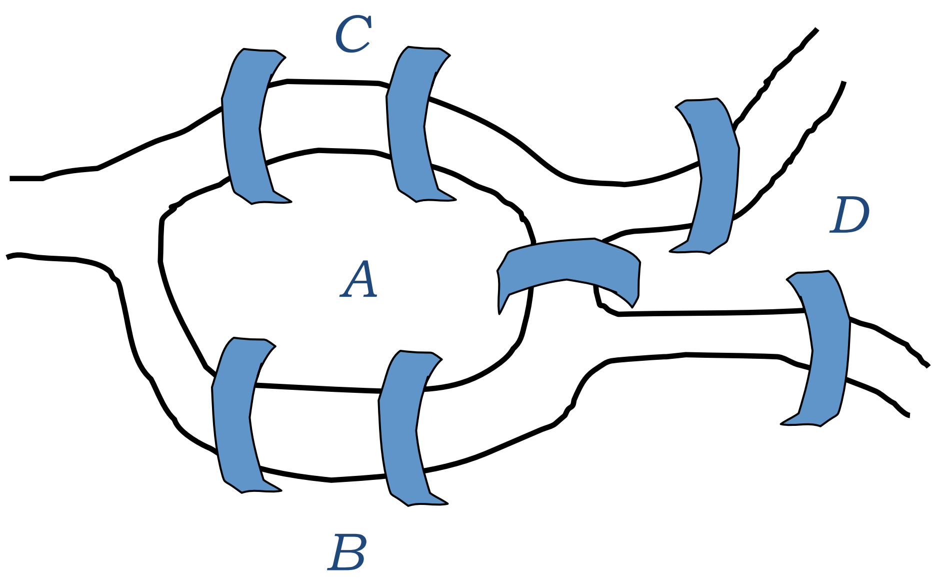
A graph has a small average distance, it has a large graph efficiency.

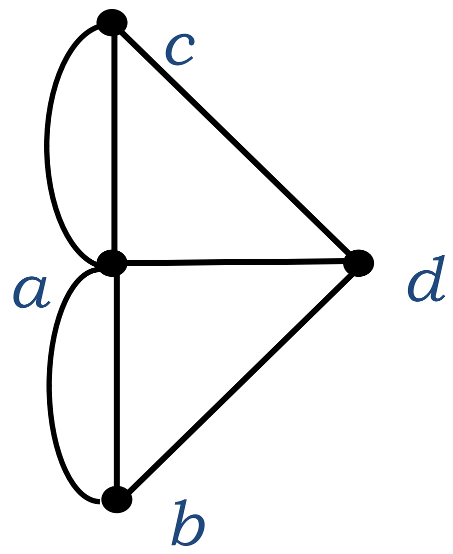
Graph Theory

- Some basic concepts of Graph theory
- Some examples of Special graphs
- Graph operations
- Graph paths and cycles
  - Euler Paths and Circuits
  - Hamilton Paths and Circuits
- Graph connectivity
- Tree and Bipartite graph
- Community structure in Networks
- Network model
- Network motifs

Euler Paths and Circuits

- The Seven bridges of Königsberg, Prussia (now called Kaliningrad and part of the Russian republic)





The townspeople wondered whether it was possible to start at some location in the town, travel across all the bridges once without crossing any bridge twice, and return to the starting point .

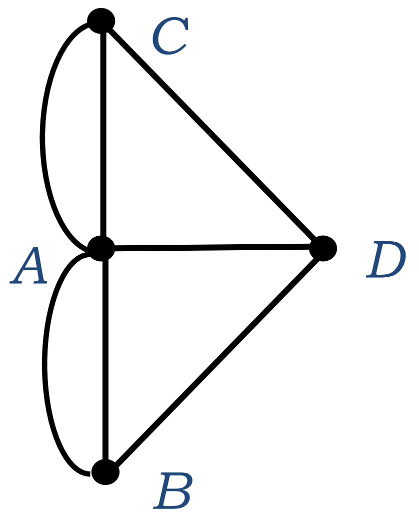
The Swiss mathematician **Leonhard Euler** solved this problem. His solution, published in 1736, may be the first use of graph theory.

Euler Paths and Circuits

- An *Euler path* is a path using every edge of the graph *G* exactly once.
- An *Euler circuit* is an Euler path that returns to its start.

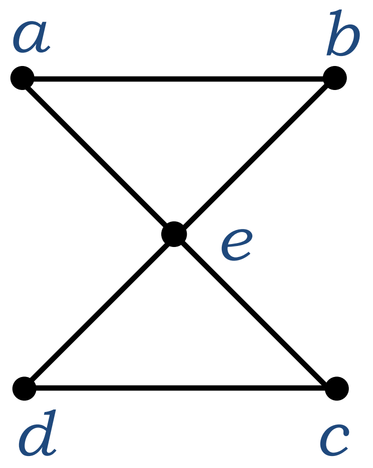
Does this graph have an Euler circuit?

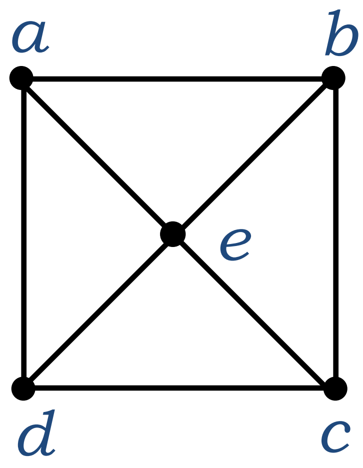
NO!

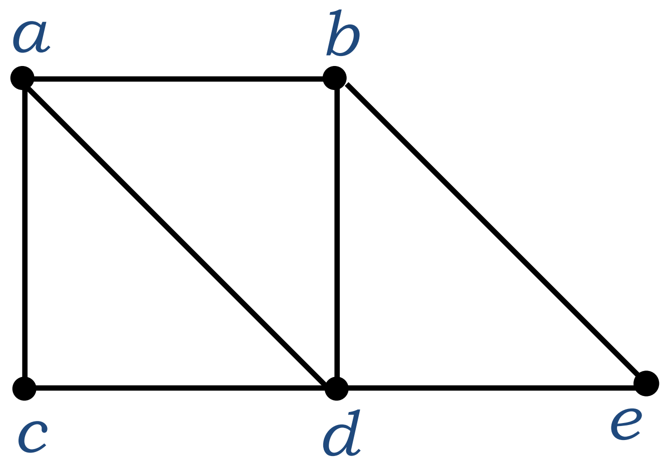


Example

- Which of the following graphs has an Euler *path*?







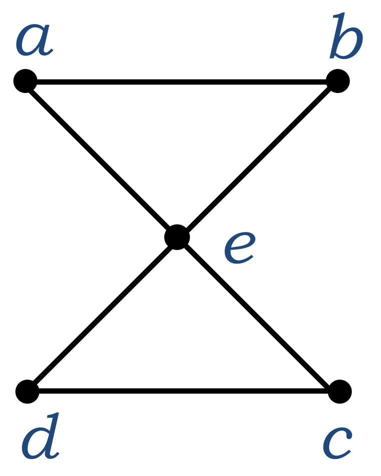
yes
(a-e-c-d-e-b-a)

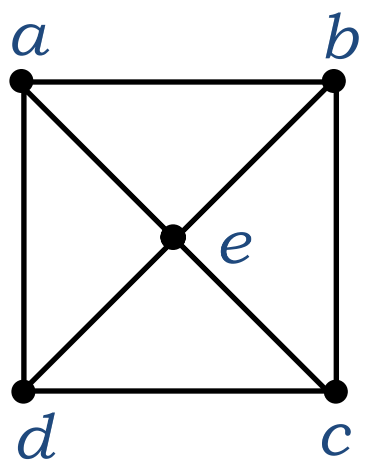
no

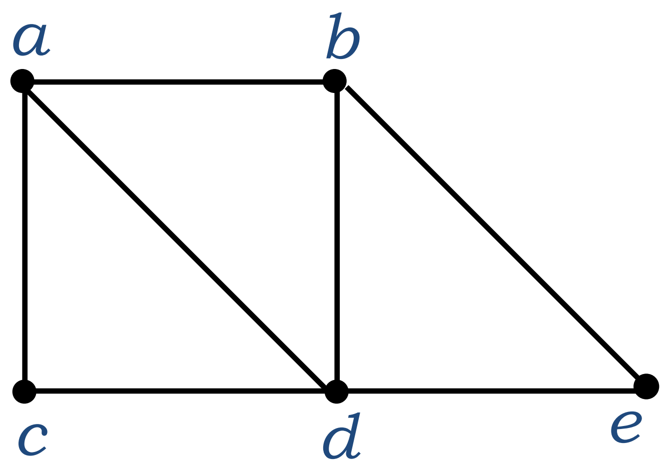
yes
(a-c-d-e-b-d-a-b)

Example

- Which of the following graphs has an Euler *circuit*?







yes
(a-e-c-d-e-b-a)

no

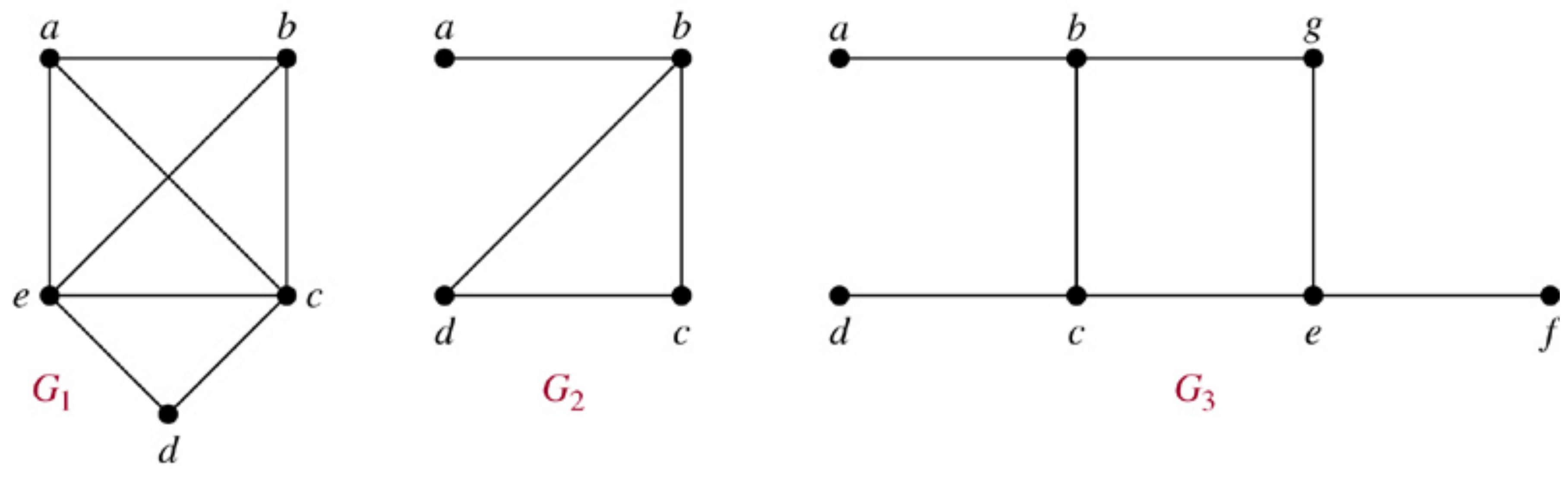
no

Hamilton Paths and Circuits

- A *Hamilton path* in a graph *G* is a path which visits every vertex in *G* exactly once.
- A *Hamilton circuit* is a Hamilton path that returns to its start.

Examples

- Which of these three figures has a Hamilton circuit? Or, if no Hamilton circuit, a Hamilton path?



Hamilton circuit Hamilton path

yes **yes** **neither**

(a-b-c-d-e-a) (a-b-c-d)

Finding Hamilton Circuits

- Unlike the Euler circuit problem, finding Hamilton circuits is hard.
- There is no simple set of necessary and sufficient conditions, and no simple algorithm.
- The best algorithms known for finding a Hamilton circuit in a graph or determining that no such circuit exists have exponential time complexity (in the number of vertices of the graph).
- Finding an algorithm that solves this problem with polynomial time complexity would be a major accomplishment because it has been shown that **this problem is NP-complete**.

Travelling Salesman Problem

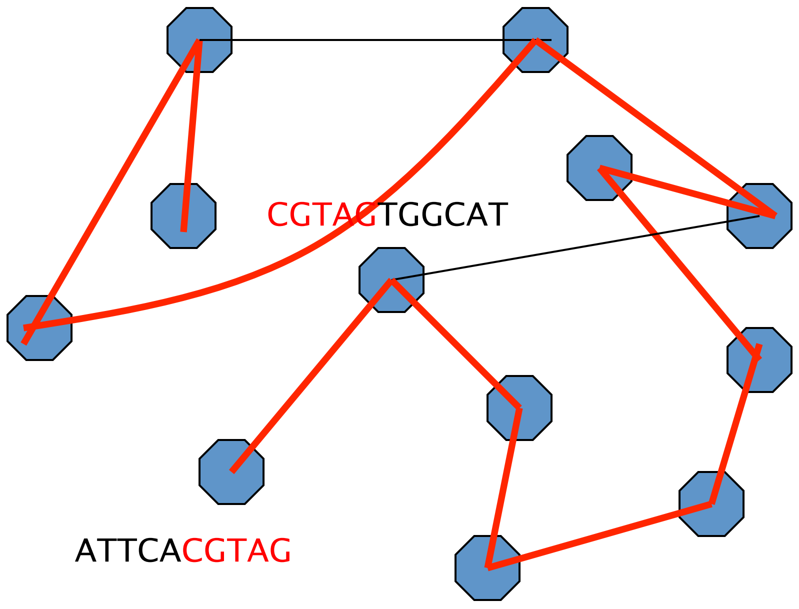
- A Hamilton circuit or path may be used to solve practical problems that require visiting “vertices”, such as:
  - planning, logistics, and the manufacture of microchips.
- A classic example is the Travelling Salesman Problem (TSP) – finding a Hamilton circuit in a complete graph such that the total weight of its edges is minimal. A direct application of TSP is for logistics.

Overlap-layout-consensus

Step 3: Consensus stage, Find Contigs

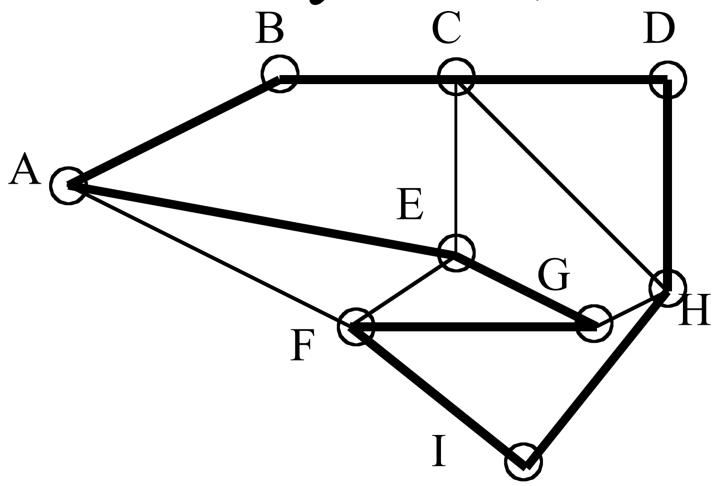
Try to find the Hamilton path:

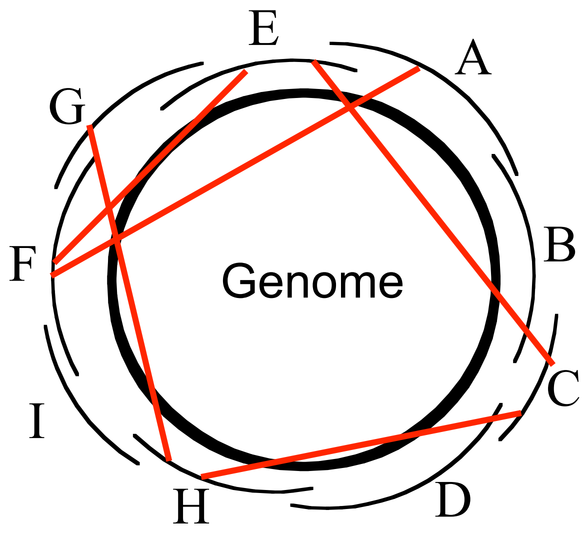
- A path in the graph contains each node exactly once.
- Following the Hamiltonian path, combine the overlapping sequences in the nodes into the sequence of the genome
- Computationally expensive (NP-hard problem)



Circular genome

- Hamilton circuit: visit each node (city) exactly once, returning to the start





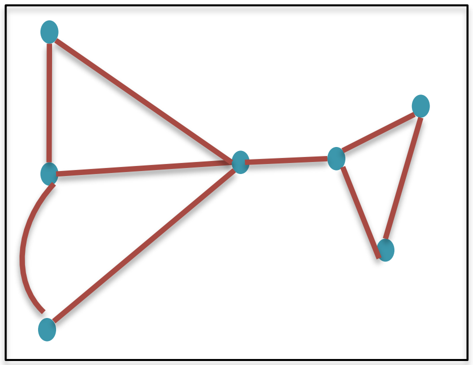
Graph Theory

- Some basic concepts of Graph theory
- Some examples of Special graphs
- Graph operations
- Graph paths and cycles
- Graph connectivity
- Tree and Bipartite graph
- Community structure in Networks
- Network model
- Network motifs

Connectivity

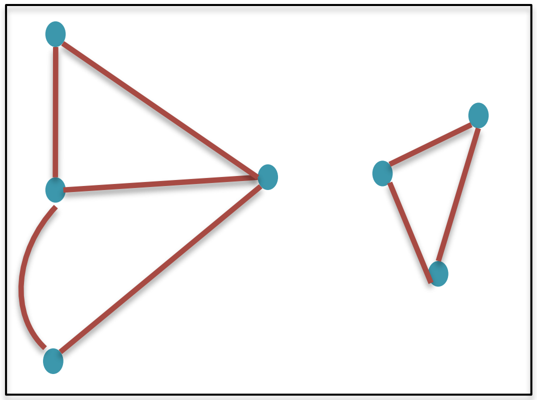
- Connected (undirected) graph: any two vertices can be joined by a path.
- Component: A maximal connected subgraph of *G* is called a component of *G*.
- A disconnected graph is made up by two or more connected components.

G1



C(G1) = 1

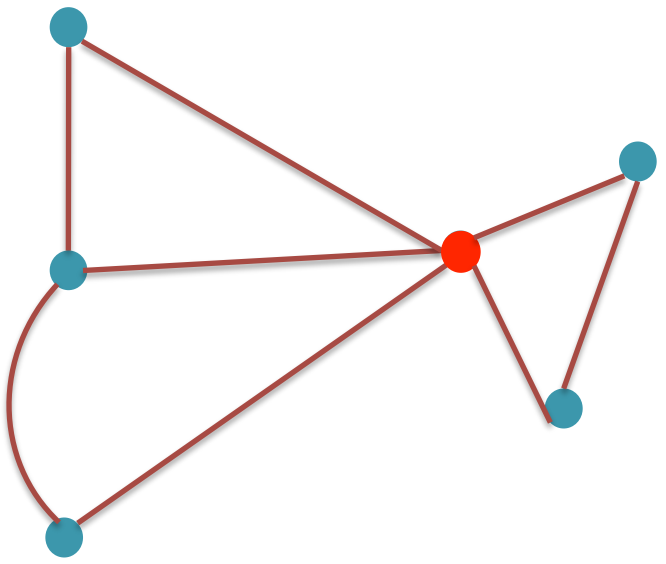
G2



C(G2) = 2

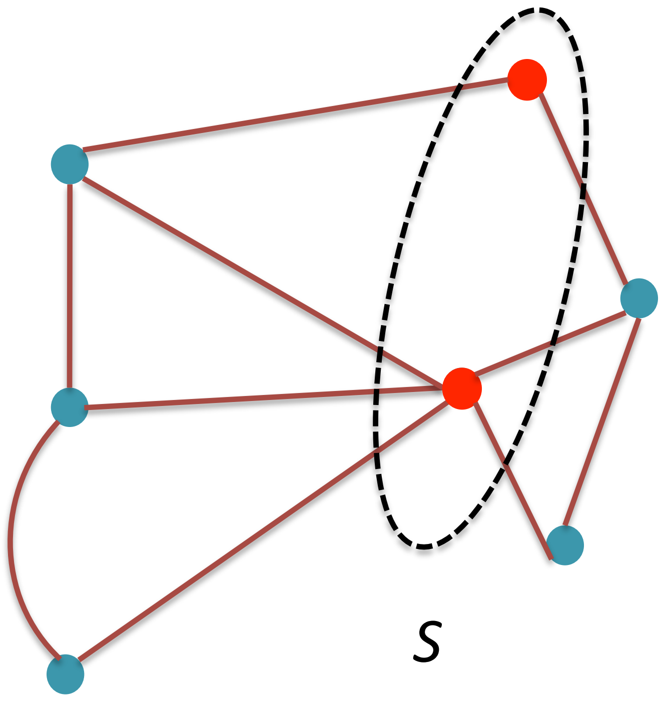
Cut vertex

- Definition: a vertex *v* is a cut vertex, if we erase it, the graph becomes disconnected.
- $C(G-v) > C(G)$



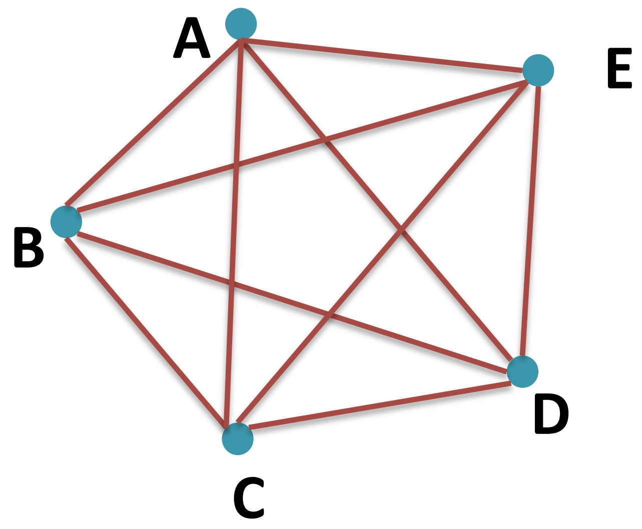
Separating sets

- Definition: A set of vertices, *S*, is separating sets if we erase it, the graph becomes disconnected.
- C(*G*-*S*) > C(*G*)



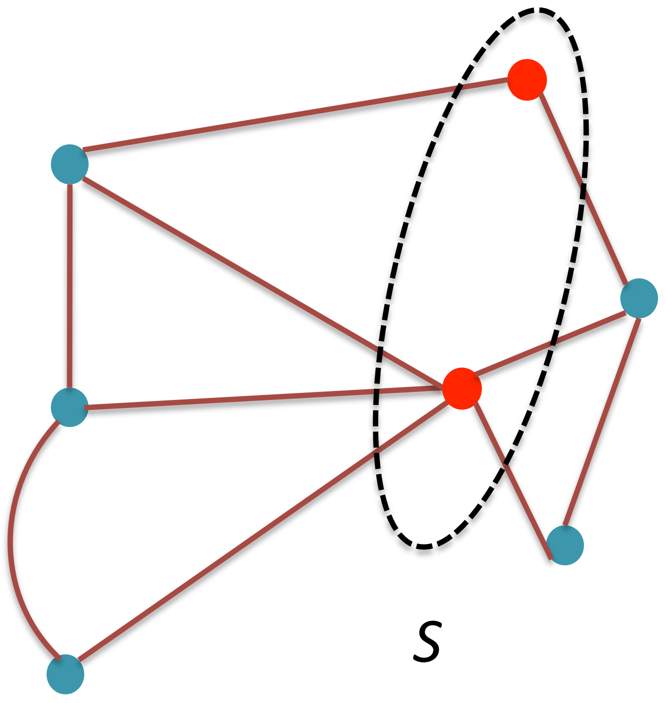
Separating sets

- Does a complete graph have a separating set?



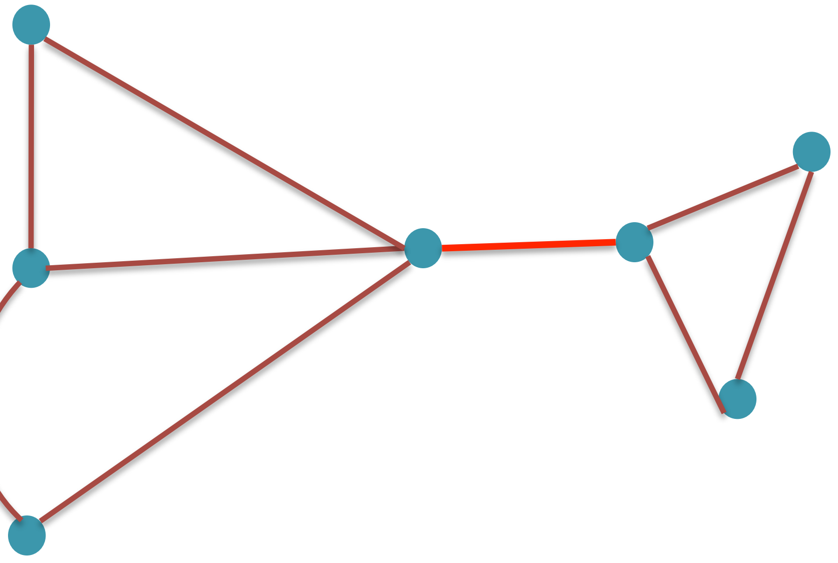
Connectivity Number

- Definition κ(G): the number of vertices of the minimum separating sets for a graph G.
- A graph is k-connected, if κ(G) ≥ k.



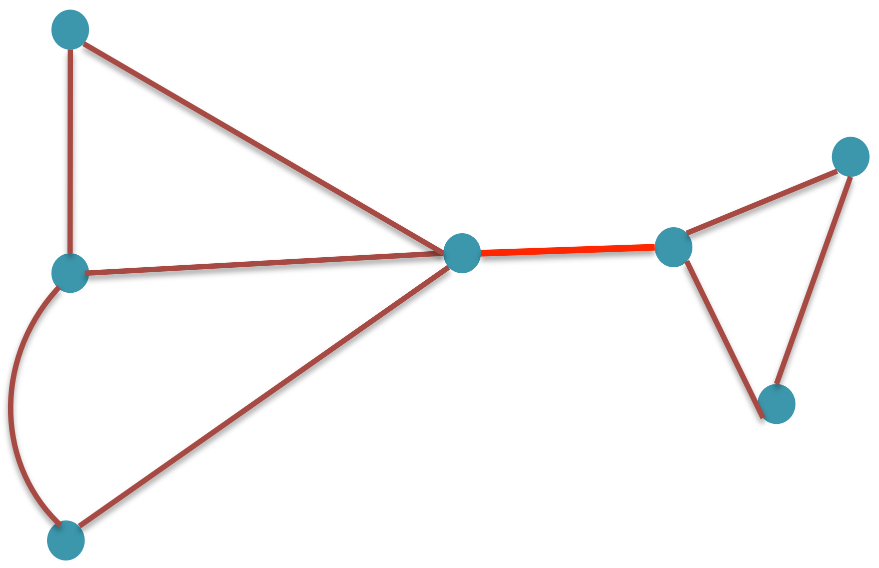
Bridge

- Definition: An edge *e* is a bridge if we erase it, the graph becomes disconnected.
- C(*G*-*e*) > C(*G*)



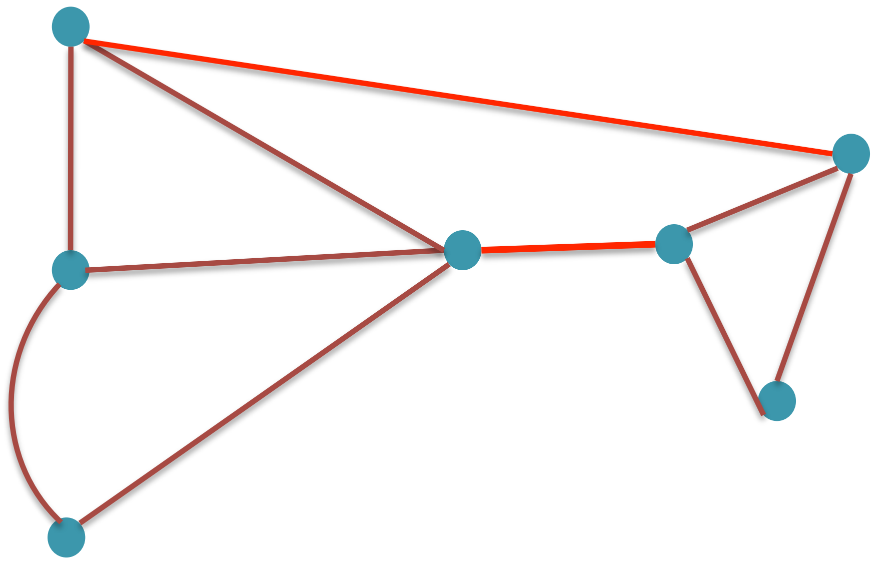
Bridge: Theorem

- An edge *e* is a bridge, if and only if *e* is not in any cycle of G.



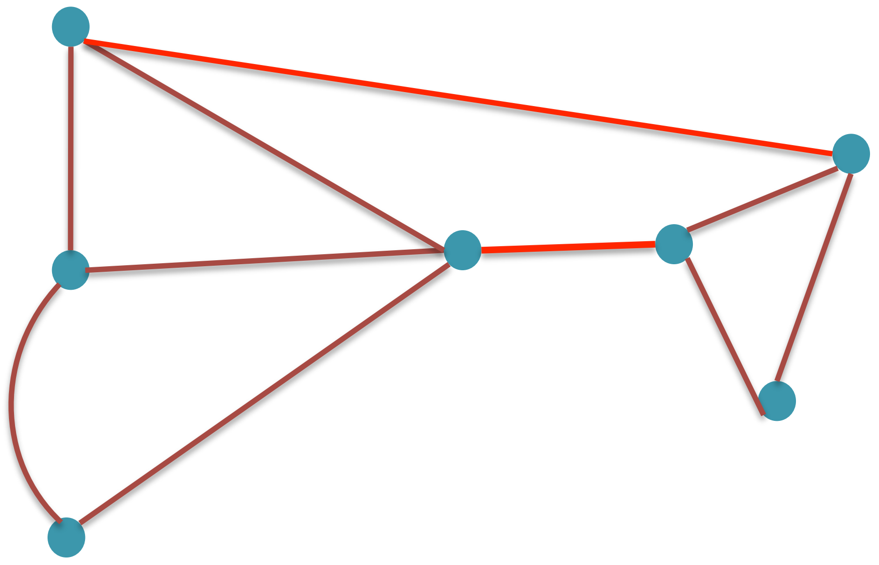
Edge cut

- Definition: A set of edges, *F*, is an edge cut if we erase it, the graph becomes disconnected.
- C(*G*-*F*) > C(*G*)



Edge Connectivity Number

- Definition $\kappa'(G)$: the number of edges of the minimum edge cut for a graph G.
- A graph is k-edge connected, if $\kappa'(G) \geq k$.

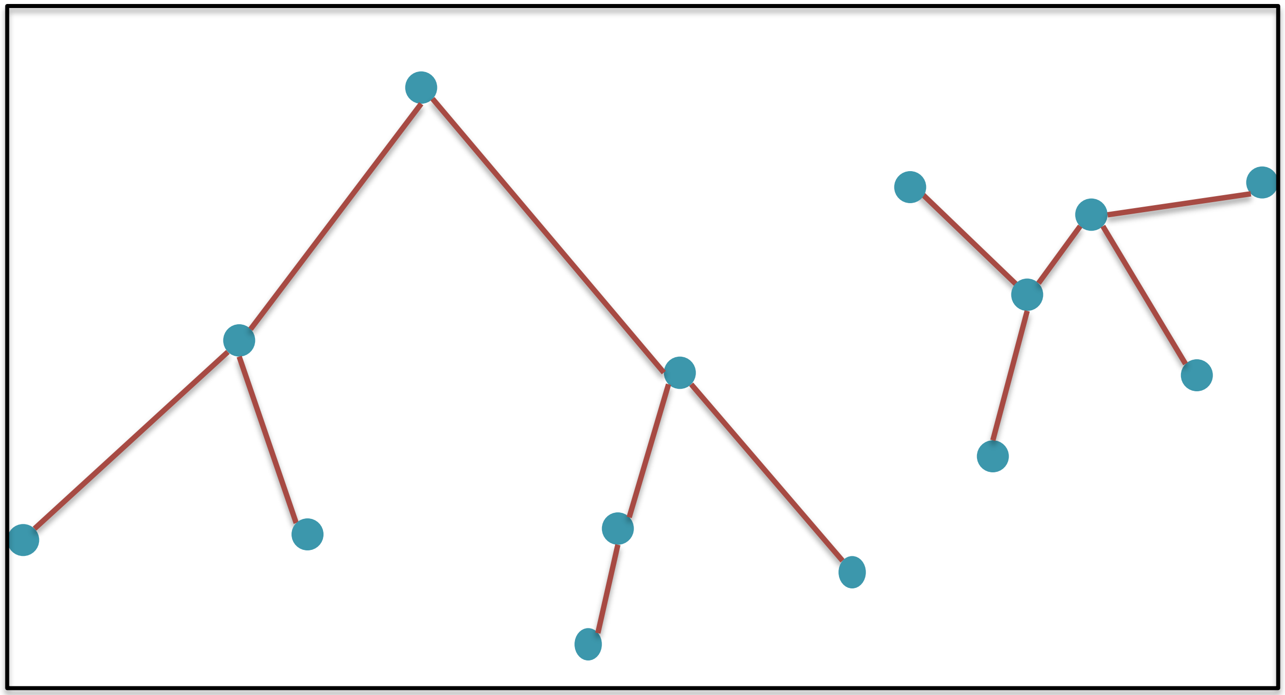


Graph Theory

- Some basic concepts of Graph theory
- Some examples of Special graphs
- Graph operations
- Graph paths and cycles
- Graph connectivity
- Tree and Bipartite graph
- Community structure in Networks
- Network model
- Network motifs

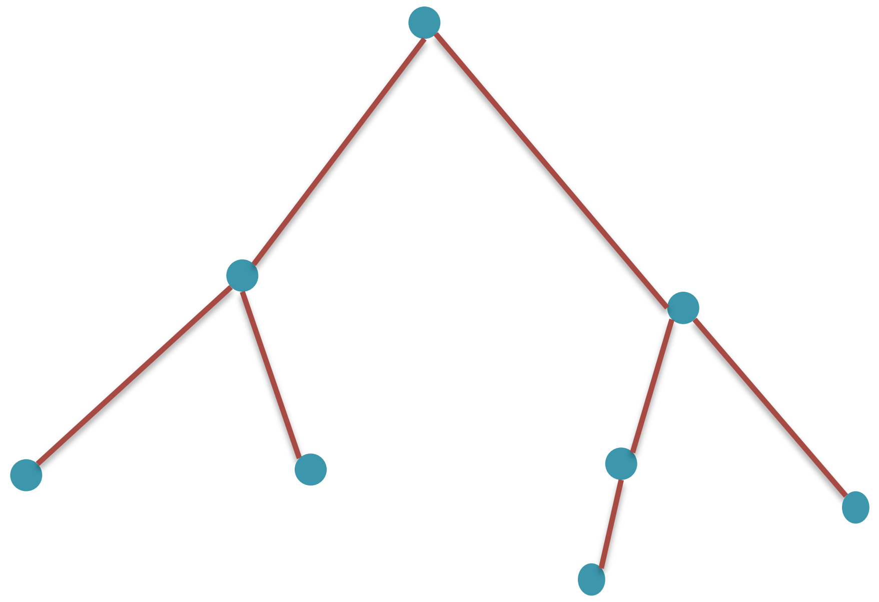
Forest and Trees

- A tree is an acyclic graph, but an acyclic graph is not necessary to be a tree.



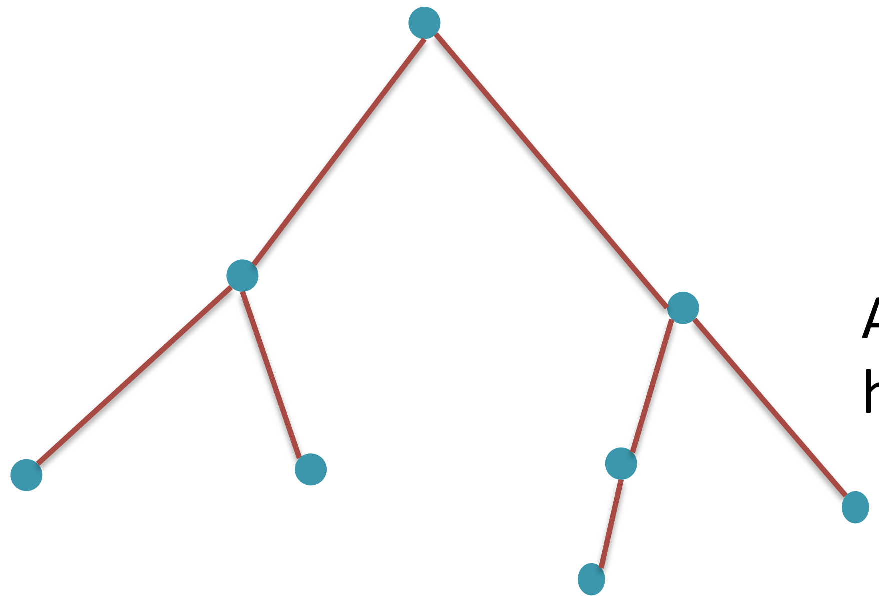
Edegs in a Tree

- Corollary: A graph is a tree if and only if its edges are bridges.
- Corollary: A graph with *n* vertices is a tree if and only if it has *n*-1 edges.



Features of Trees

- Corollary: A tree has more than 2 leaves.
- Equivalent statements:
  - Graph T is a tree
  - Any two vertices are connected in T by a unique path
  - T is acyclic and |E|=|V|-1



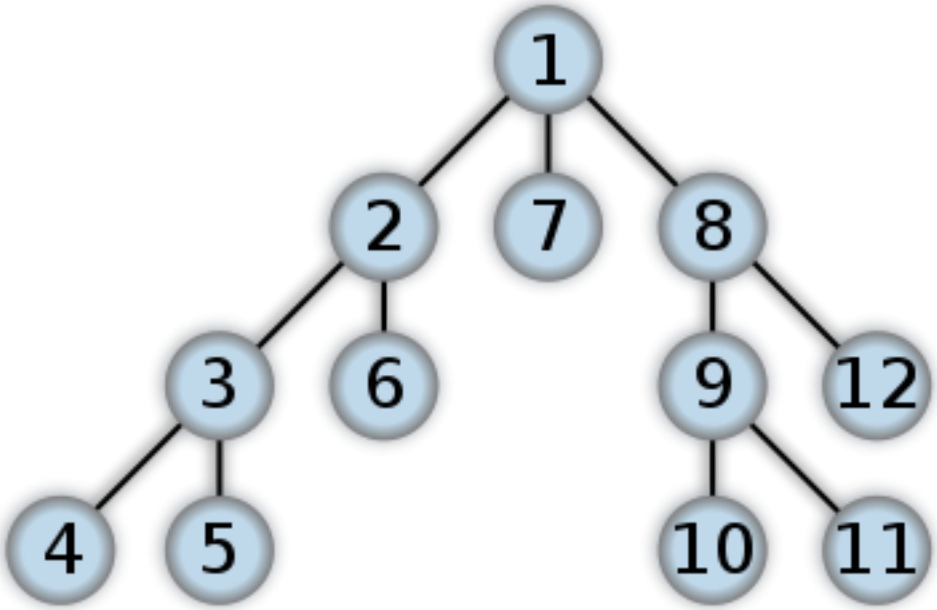
A leaf is a node that has one neighbor.

Searching

- Algorithms for traversing or searching all nodes of a tree.
    1. Depth First Search (DFS)
    2. Breadth First Search (BFS)

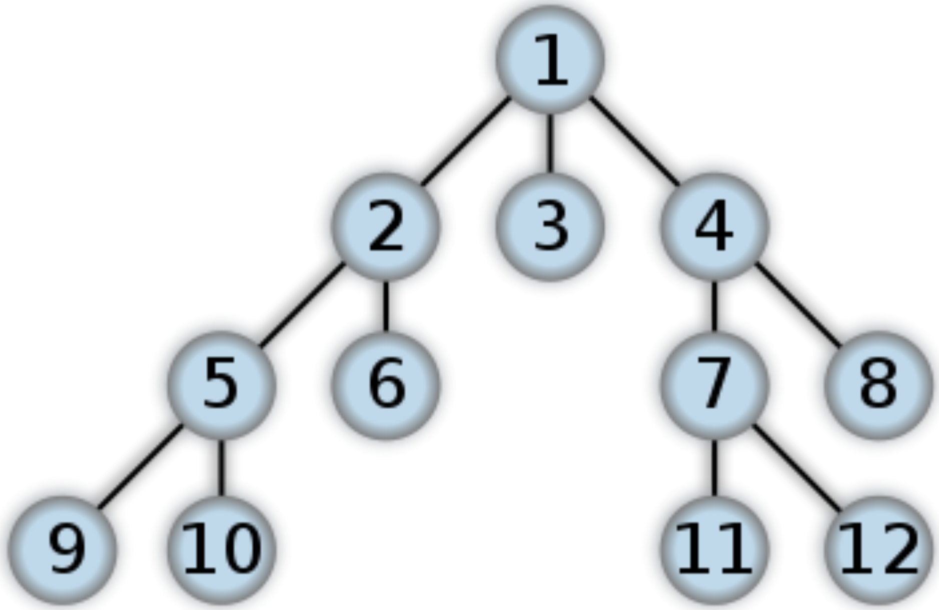
Depth First Search (DFS)

- One starts at the root (selecting some node as the root in the graph case) and explores as far as possible along each branch before backtracking. Once needs to backtrack, go back one edge and explores as deep as possible from another branch.



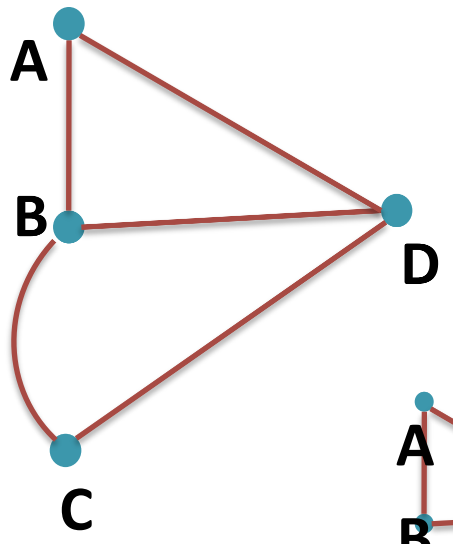
Breadth First Search (BFS)

- Begins at the root node and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal

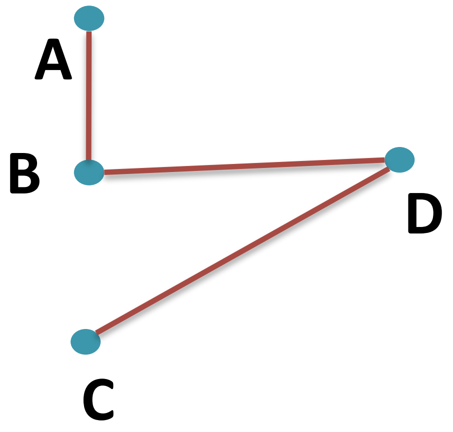


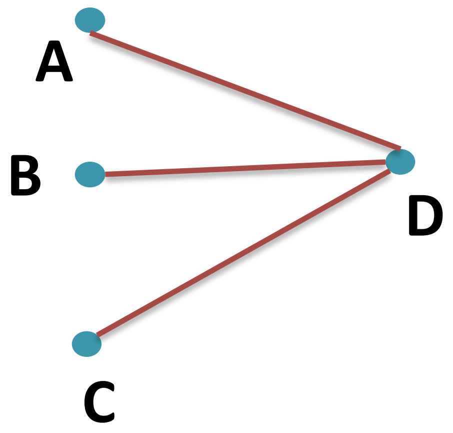
Spanning tree

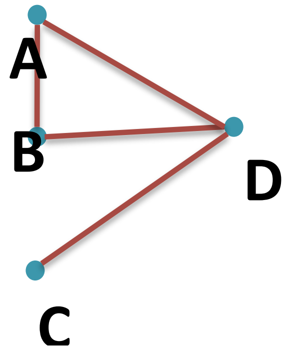
- *H* is a spanning subgraph of *G*, if V(H)=V(G) and E(H) $\subseteq$ E(G).
- If this subgraph is a tree, we call it a spanning tree.



G(V,E)







Spanning tree

- If *G*(*E*,*V*) is a K_n, how many spanning trees does it have.
- Cayley's formula:

$$t(K_n) = n^{n-2}$$

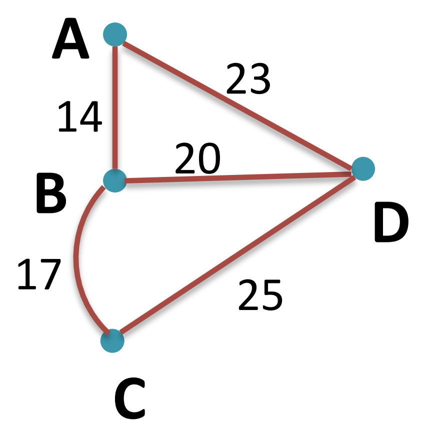
e.g. $t(K_5)=5^3=125$

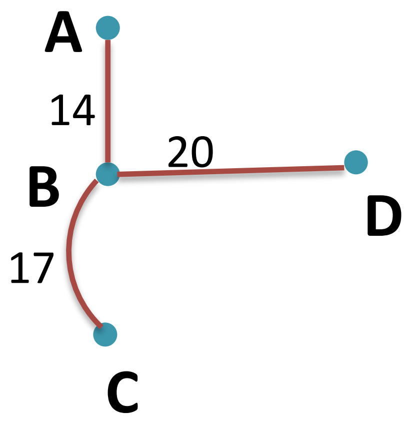
Optimization problems of spanning trees

- Maximum (minimum) spanning tree of a weighted graph.
- Degree-constrained spanning tree.
- The spanning tree with largest number of leaves.
- The minimum diameter spanning.

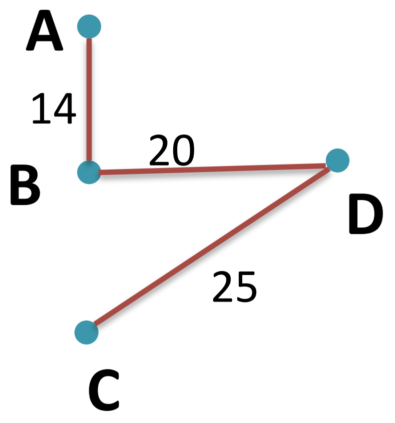
Minimum Spanning tree

- Assign a weight to a spanning tree by computing the sum of the weights of the edges in that spanning tree.
- The spanning tree having the smallest weight is the minimum spanning tree.





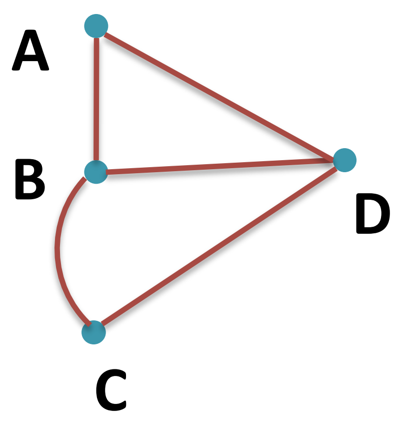
51

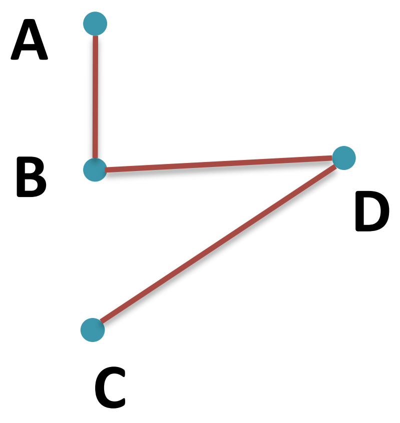


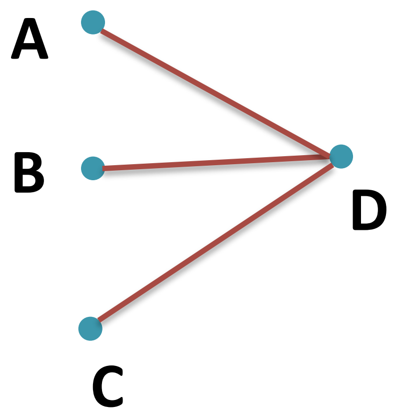
59

Degree-constrained spanning tree

- A spanning tree where the maximum vertex degree is limited to a certain constant *k*.



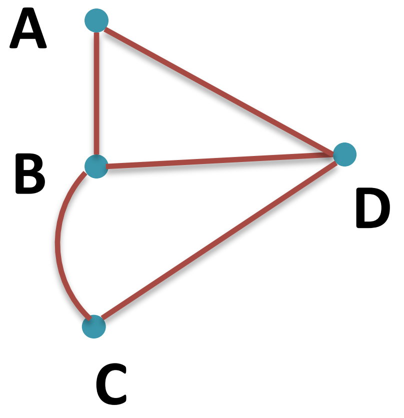


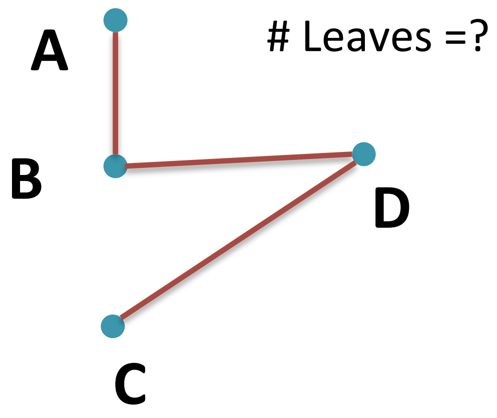


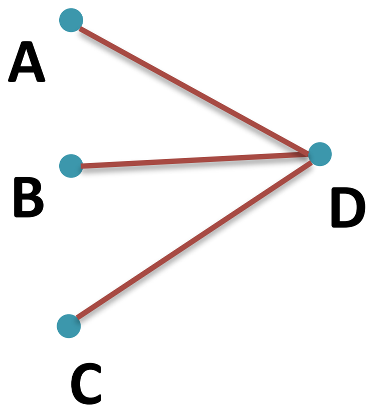
If *k*=2 yes no

The spanning tree with largest number of leaves

- A leaf is a node with degree=1 in a tree.
- A spanning tree with largest number of leaves has small diameter, small average distance, and large graph efficiency.

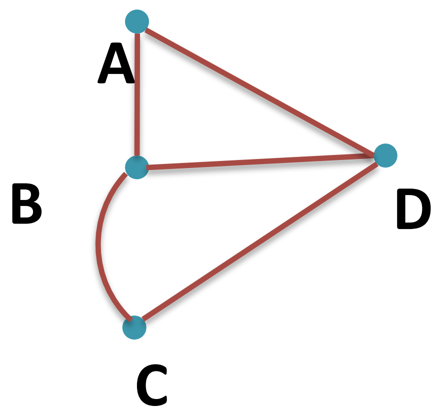


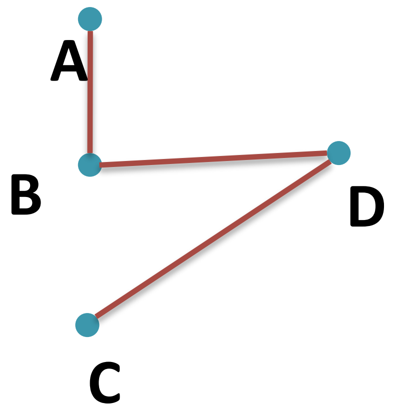


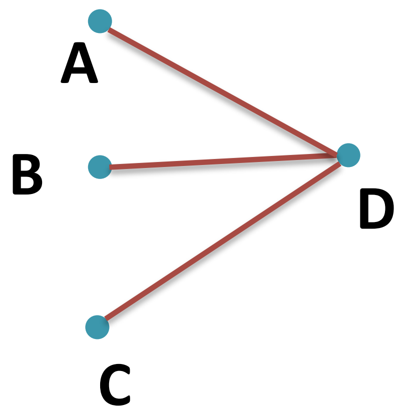


The minimum diameter spanning

- The minimum diameter: Small average distance and Large graph efficiency



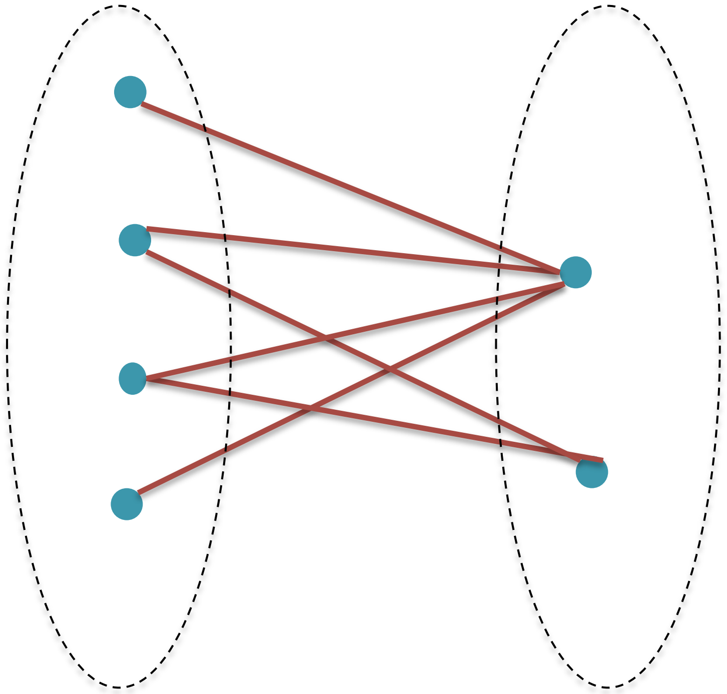




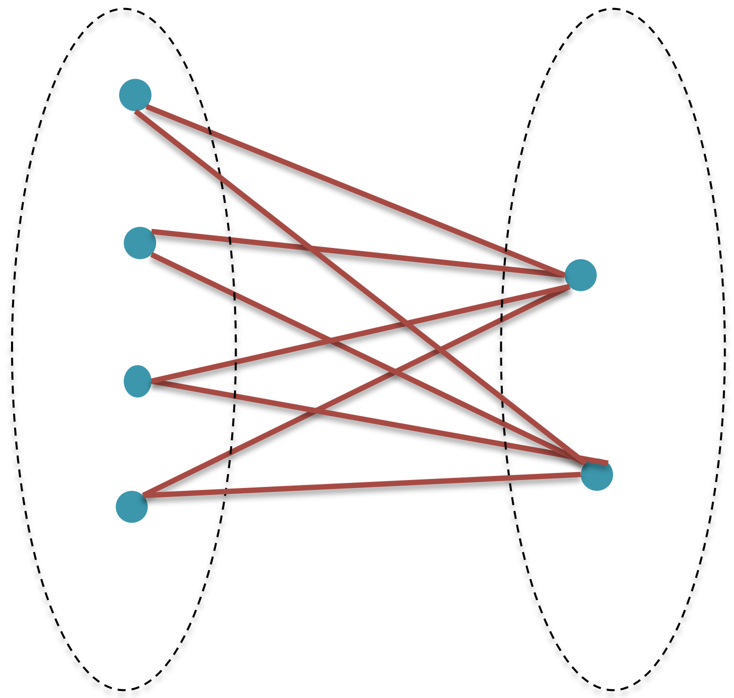
3

2

Bipartite graph



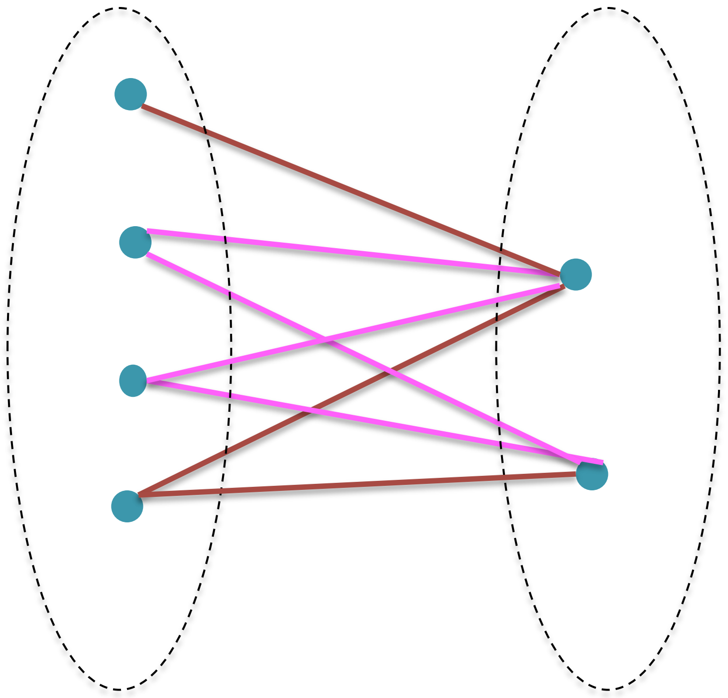
Complete Bipartite graph



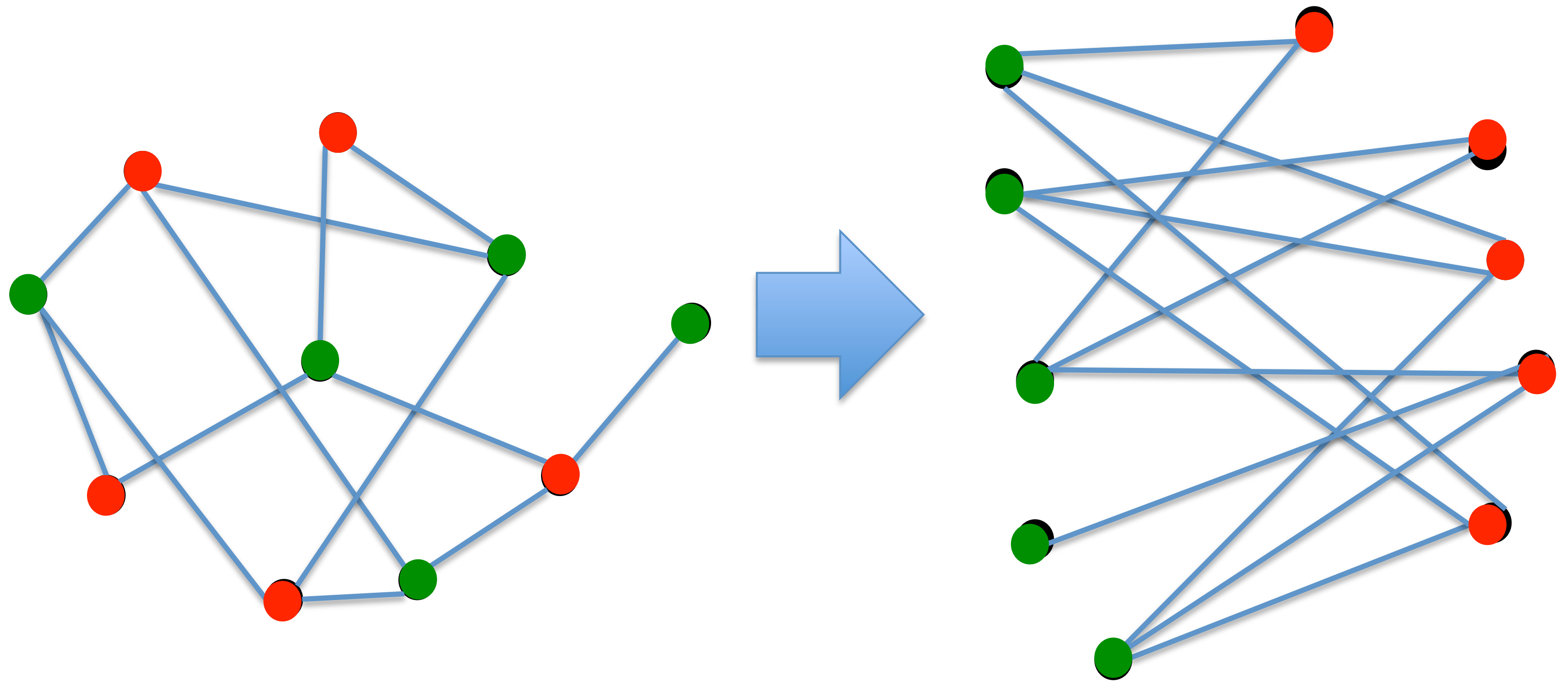
$K_{2,4}$

Theorem: Bipartite graph

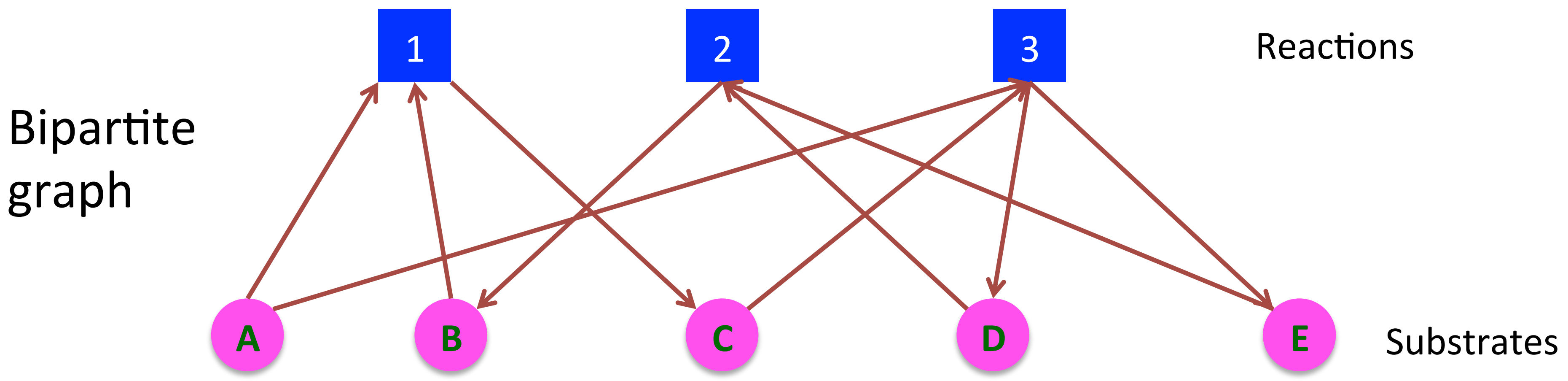
Theorem: A Graph G is a Bipartite if and only if graph has no odd cycles



Check if a graph is a Bipartite graph



Metabolic interaction networks



1 A + B → C

2 D + E → B

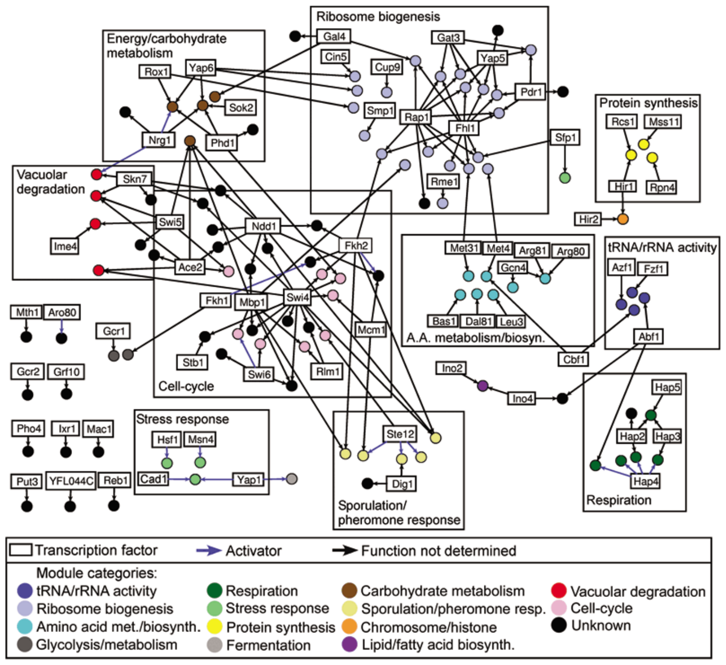
3 A + C → D + E

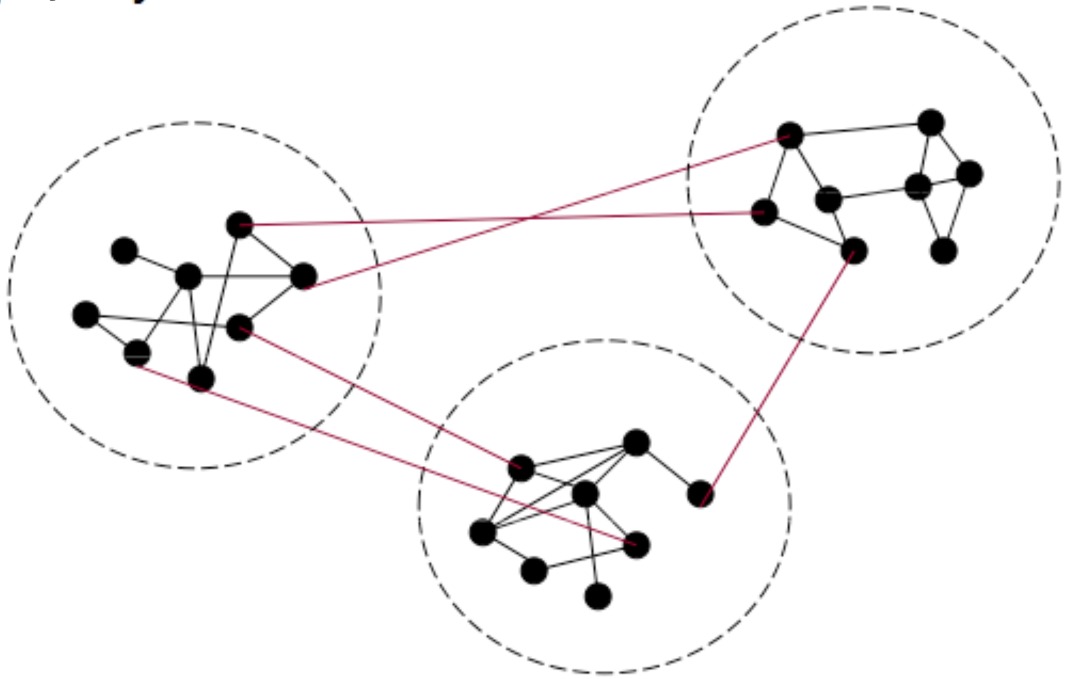
Graph Theory

- Some basic concepts of Graph theory
- Some examples of Special graphs
- Graph operations
- Graph paths and cycles
- Graph connectivity
- Tree and Bipartite graph
- Community structure in Networks
- Network model
- Network motifs

Community structure in networks

- Many real-world networks, especially social ones, exhibit community structure (also called modularity)
- Intuitively community structure can be defined as the existence of subgraphs that are densely connected but sparsely inter-connected.



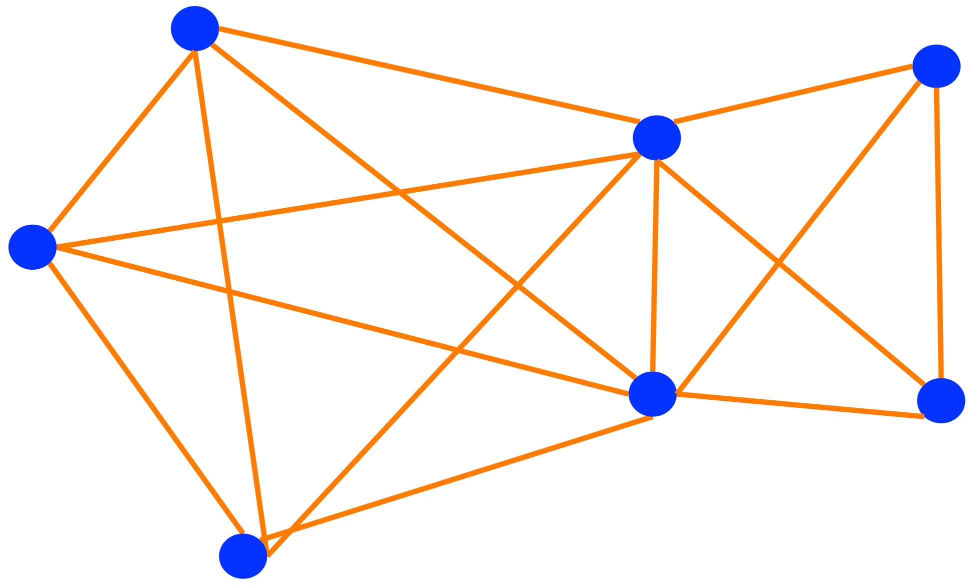


Examples of network communities

- Protein-Protein Interaction Networks
  - Communities :Protein complexes
- Metabolic Networks
  - Communities : Metabolic pathways
- Gene regulatory Networks
  - Communities : genes are involved in the same biological process

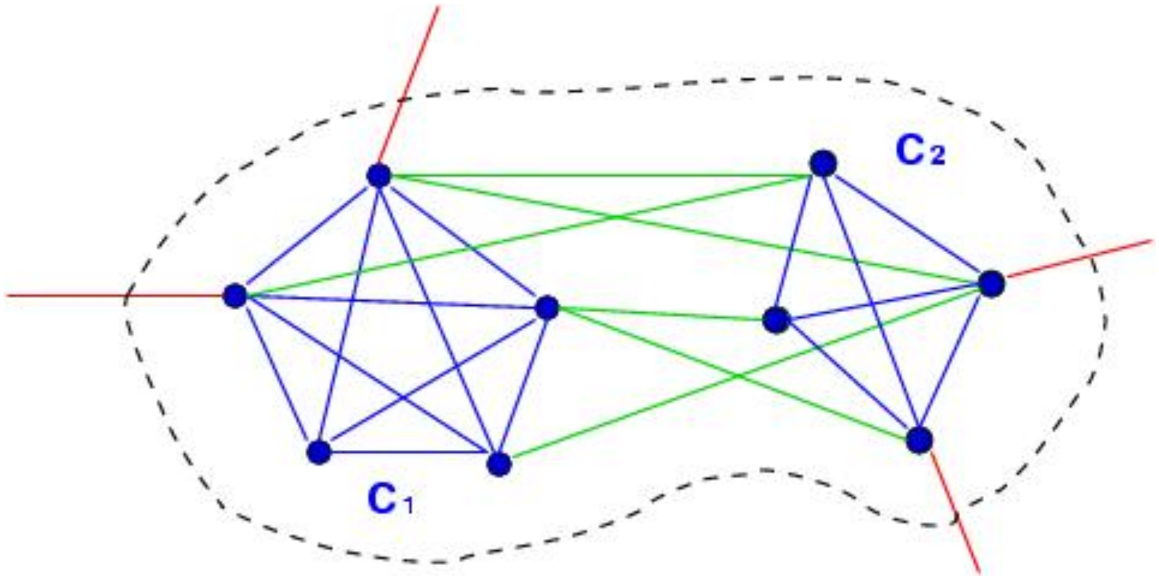
Concepts related to communities

- Clique (a complete subgraph)
- Chain of cliques –adjacent cliques share some nodes.



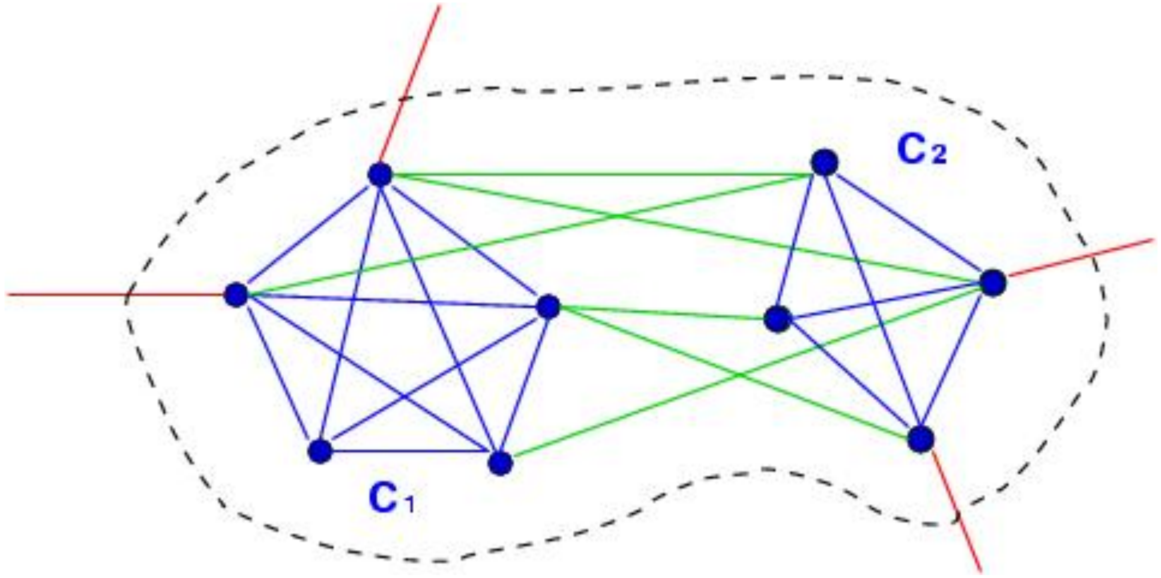
Concepts related to communities

- Definitions using the edges inside and outside a presumed community
  - k_{in}–edges of node *i* that stay inside the community
  - k_{out} –edges of node *i* that go outside of the community



Strong or Weak communities

- Strong community:
  $k_{in} \geq k_{out}$ for every node i in the community
- Weak community:
  $\sum k_{in} \geq \sum k_{out}$, where the sum is over nodes in the community

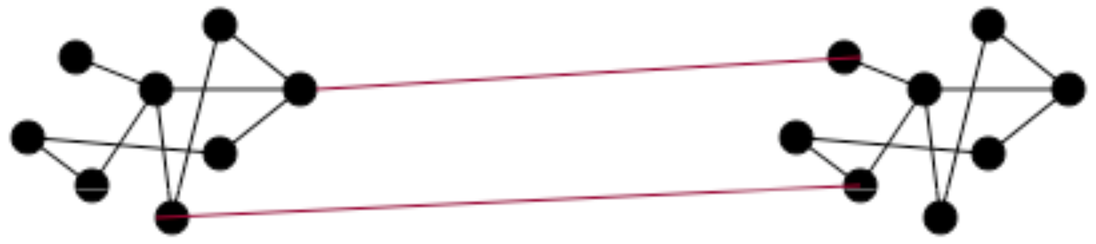


Algorithms to discover communities

- Input: A network G(v,e)
- Output:
  - The number of communities
  - Classification of nodes into these communities
- Algorithm:
  - Calculate the betweenness for all edges in the network.
  - Remove the edge with highest betweenness.
  - Recalculate the betweenness for all edges affected by the removal.
  - Repeat

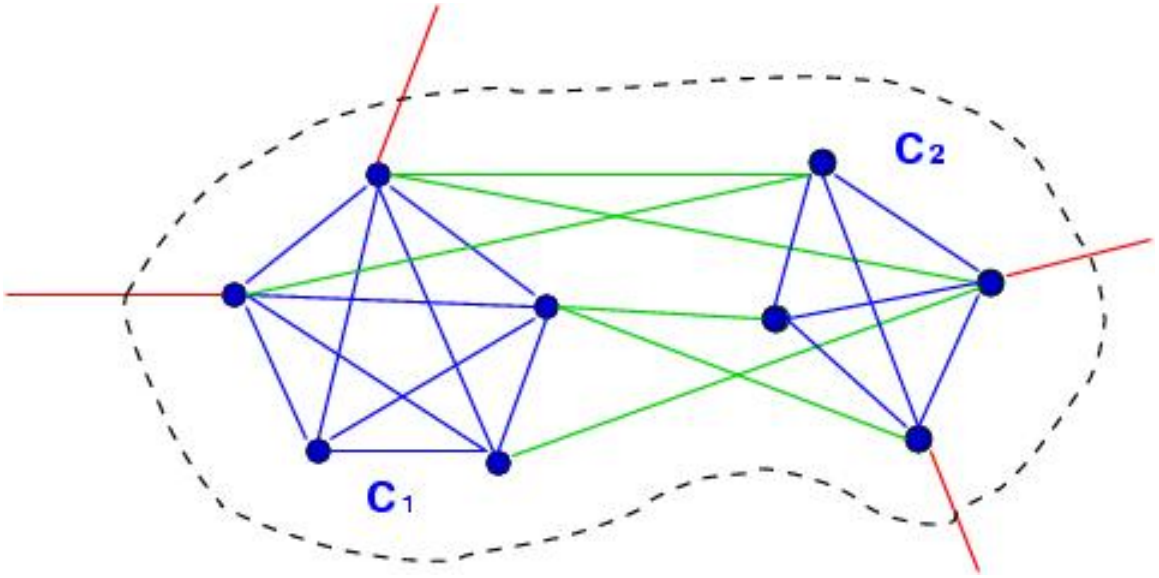
Algorithms to discover communities

- Betweenness centrality



Significant communities

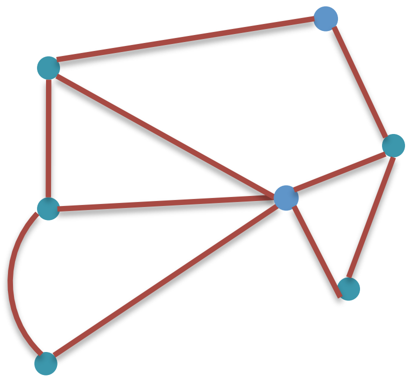
- To check if a particular division is significant, we can determine
  (1) The fraction of edges within the community divided to the fraction of edges from the community to outside.

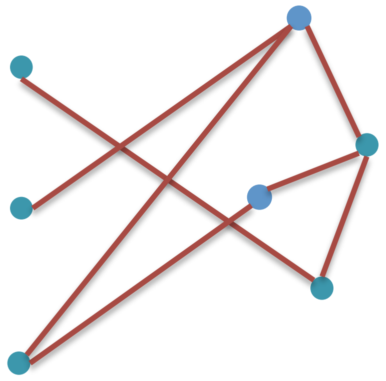


Significant communities

- To check if a particular division is significant, we can determine

  (2) The observed number of edges within the community divided by the expected number of edges. The expected number of edges within a community is the number of edges that are distributed randomly.

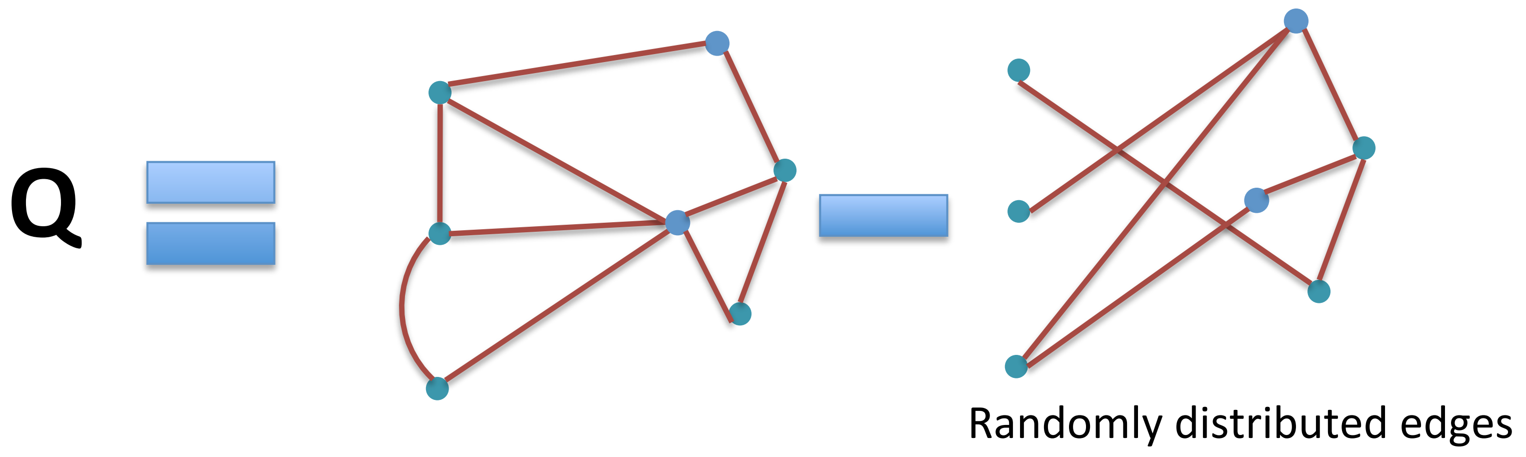




Randomly distributed edges

Significant communities

- To check if a particular division is significant, we can determine
  - (3) Modularity measure Q, defined as the fraction of edges that fall within the community, minus the expected value. The expected value is the number of edges that fall in the community at random distribution without regard for the community structure.



Graph Theory

- Some basic concepts of Graph theory
- Some examples of Special graphs
- Graph operations
- Graph paths and cycles
- Graph connectivity
- Tree and Bipartite graph
- Community structure in Networks
- Network model
- Network motifs

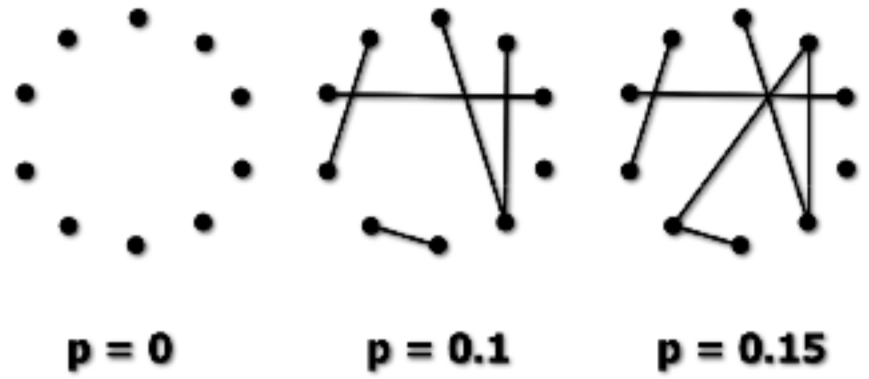
Network Models

- Properties common to many large-scale networks, independently of their origin and function.
- Network models:
  - Random graphs
  - Scale free graphs

Random Networks

- Uniformly random network:
  - distributes the edges uniformly among nodes.
- Probabilistic interpretation:
  - assigns equal probability to all graphs with exactly *M* edges.

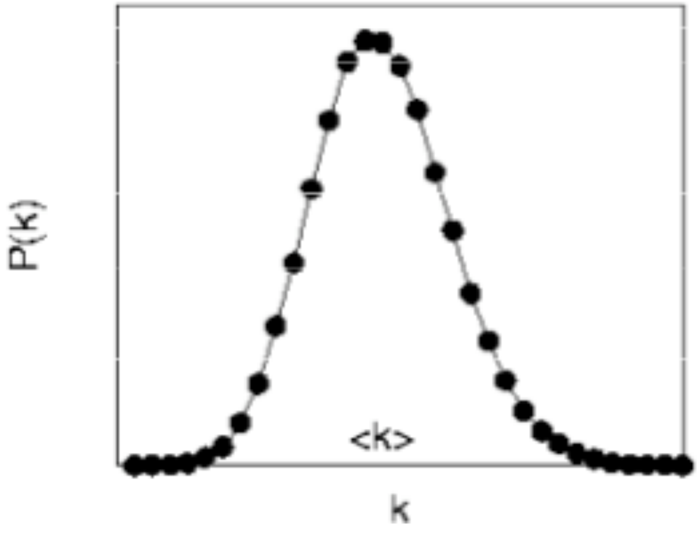
Random Networks



- fixed node number *N*
- connecting pairs of nodes with probability *p*

Expected number of edges: $$E = p\frac{N(N-1)}{2}$$

Node degrees in random graphs



Average degree:

$$\langle k \rangle \approx p|V|$$

Degree distribution:

$$P(k) \approx \binom{N-1}{k} p^k (1-p)^{N-1-k}$$

Most of the nodes have approximately the same degree. The probability of very highly connected nodes is exponentially small.

Some properties for random networks

- The average distance scale logarithmically with the network size.

$$l = \frac{\log(\mathbf{N})}{\log\langle \mathbf{dist} \rangle}$$

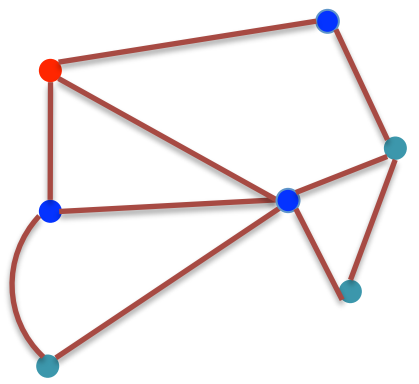
- The clustering coefficient depend on the network size.

$$C \propto |V|$$

Clustering coefficient

- Local clustering coefficient C_i for a vertex v_i is given by the proportion of links between the vertices within its neighborhood divided by the number of links that could possibly exist between them.

$$C_i = \frac{|e_{ij}|}{\mathbf{V}(\mathbf{V}-1)/2}$$



Clustering coefficient

- global clustering coefficient

$$C = \sum_{i} C_i$$

A scale free network

- Power-law degree distributions are found in diverse networks, especially for biological networks.

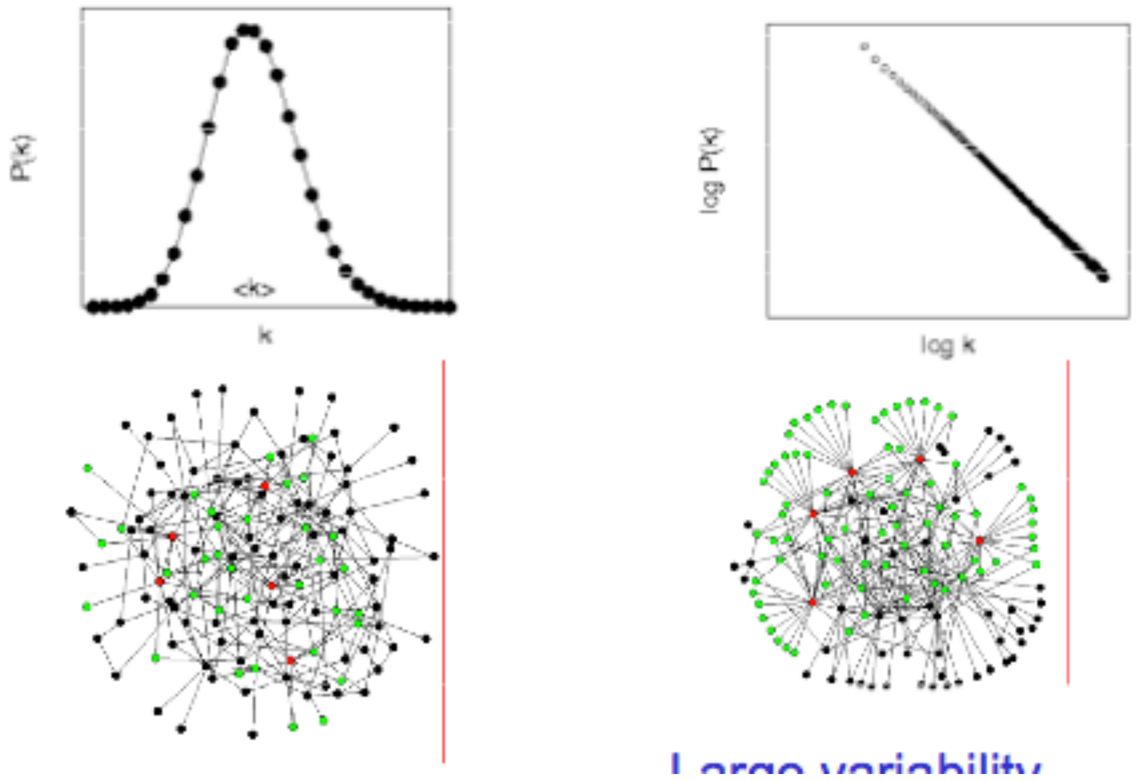
$$\log\big(\mathbf{P}(\boldsymbol{k})\big) \approx \ \boldsymbol{-}\gamma\ \log(\boldsymbol{k})$$

$$\mathbf{P}(\boldsymbol{k}) \approx \ \boldsymbol{c}\ \boldsymbol{k}^{-\gamma}$$

Power-law degree distributions

A scale free network

- Power-law degree distributions were found in diverse networks



Properties for Scale-free networks

- A scale-free network has vertices with a degree that greatly exceeds the average. The nodes with high degrees are often called "hubs”.
- The scale-free property strongly correlates with the network's robustness to failure. This hierarchy allows for a fault tolerant behavior.
- The clustering coefficient decreases as the node degree increases. This implies that the low-degree nodes belong to very dense sub-graphs and those sub-graphs are connected to each other through hubs.
- The average distance is shorter than other type of network.

Homework 8

- Due by April 10, 11:59PM
- Learn to use igraph package of R.
- Construct a graph
- Display a graph
- Calculate degrees, shortest paths, and betweenness etc.