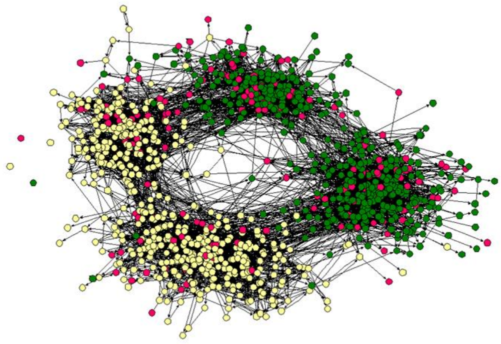
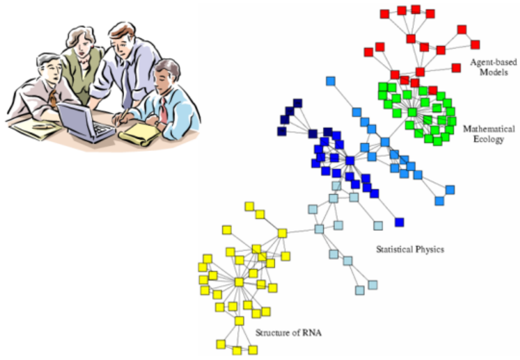
Introduction to Graph Theory

In real world applications, graph is the base for all kinds of *network.*

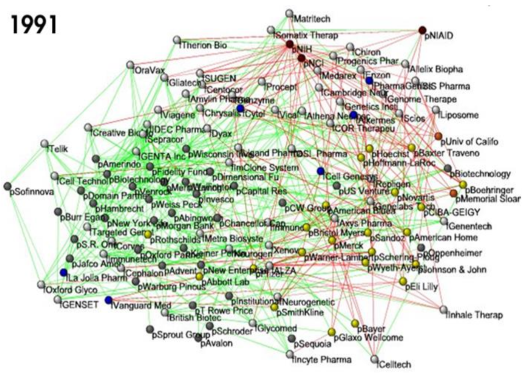
Friendship Network



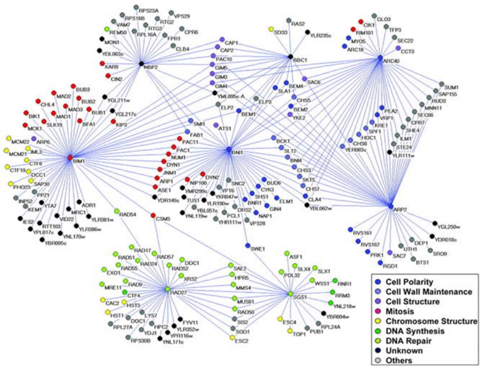
Scientific collaboration network



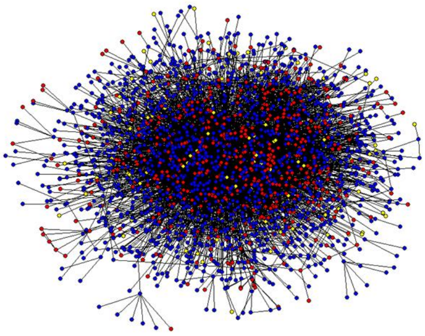
Business ties in US biotech-industry



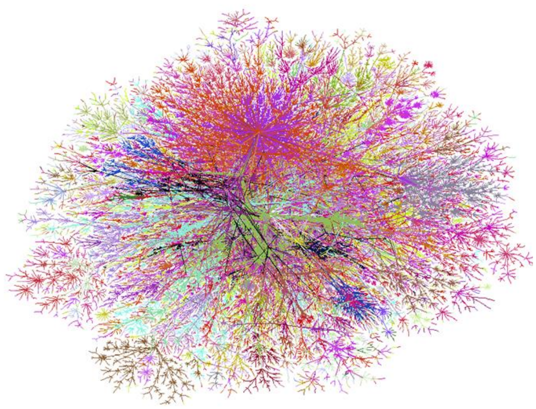
Genetic interaction network



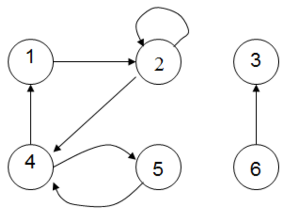
Protein-Protein Interaction Networks

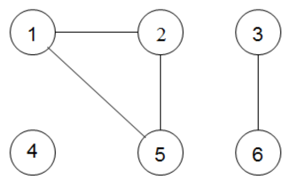


Internet



- Network = graph
- Informally a *graph* is a set of nodes joined by a set of lines or arrows.





Graph-based representations

Representing a problem as a graph can provide a different point of view
Representing a problem as a graph can make a problem much simpler

- More accurately, it can provide the appropriate tools for solving the problem

What makes a problem graph-like?

There are two components to a graph

- Vertex/Nodes and Edges

In graph-like problems, these components have natural correspondences to problem elements

- Entities are nodes and interactions between entities are edges
- Most complex systems are graph-like

Definition

G is an ordered triple G:=(V, E, f)

- V is a set of nodes, points, or vertices.
- E is a set, whose elements are known as edges or lines.
- f is a function
  - maps each element of E
  - to an unordered pair of vertices in V.

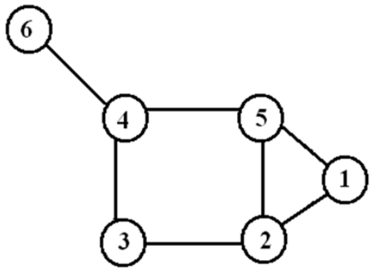
Vertex

- Basic Element
- Drawn as a *node* or a *dot*.
- **Vertex set** of *G* is usually denoted by *V*(*G*), or *V*

Edge

- A set of two elements
- Drawn as a line connecting two vertices, called end vertices, or endpoints.
- The edge set of G is usually denoted by E(G), or E.

Example – how to represent V and E

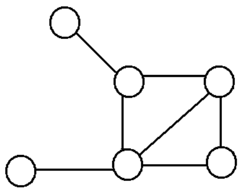


- V:={1,2,3,4,5,6}
- E:={{1,2},{1,5},{2,3},{2,5},{3,4},{4,5},{4,6}}

Types of graphs

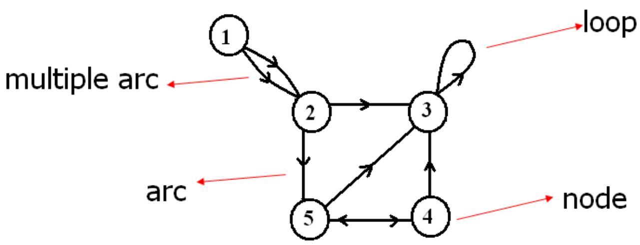
Simple Graphs

Simple graphs are graphs without multiple edges or self-loops.



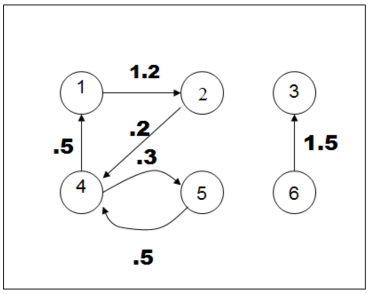
Directed Graph (digraph)

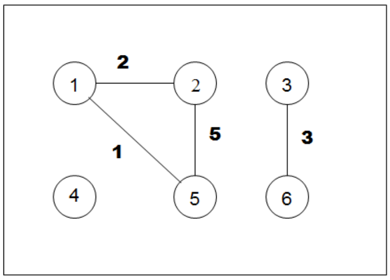
- Edges have directions
  - An edge is an *ordered* pair of nodes



Weighted graphs

- is a graph for which each edge has an associated ***weight***, usually given by a ***weight function*** $w\colon E \to \mathbf{R}$.





Connectivity

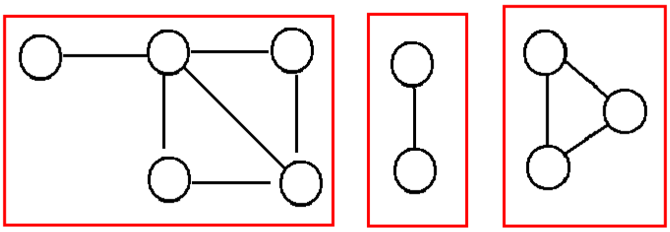
A graph is ***connected*** if

- you can get from any node to any other by following a sequence of edges OR
- any two nodes are connected by a path.

A directed graph is ***strongly connected*** if there is a directed path from any node to any other node.

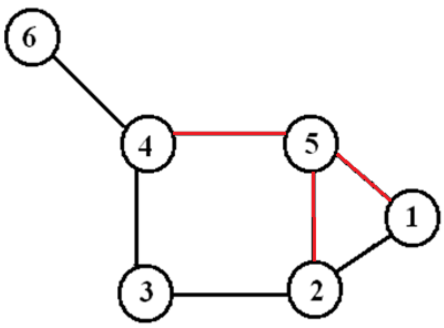
Component

Every disconnected graph can be split up into a number of connected ***components***.



Degree

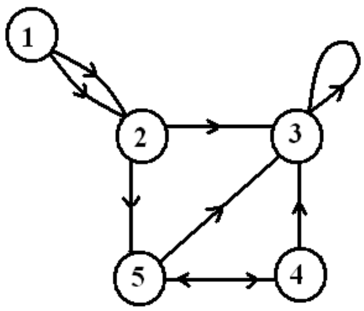
Number of edges *incident* on a node



The degree of 5 is 3

Degree (for Directed Graphs)

In-degree: Number of edges entering
Out-degree: Number of edges leaving
Degree = indeg + outdeg



outdeg(1)=2
indeg(1)=0

outdeg(2)=2
indeg(2)=2

outdeg(3)=1
indeg(3)=4

If G is a graph with m edges, then

$$\Sigma \deg(v) = 2m = 2\ |E\ |$$

If G is a digraph then

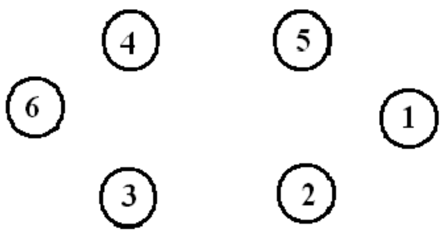
$$\Sigma\ \text{indeg}(v)=\Sigma\ \text{outdeg}(v) = |E\ |$$

Number of Odd degree Nodes is even

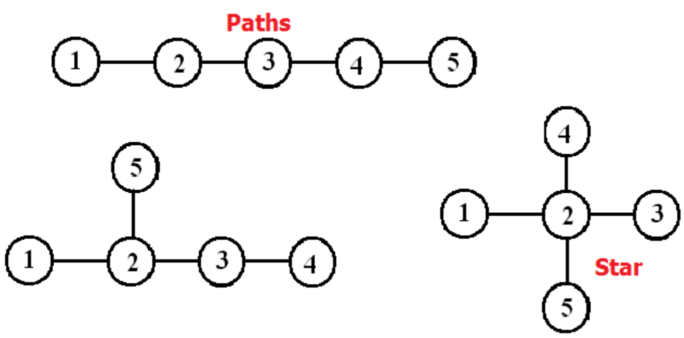
Special types of Graphs

Empty Graph / Edgeless graph
- No edge

Null graph
- No nodes
- Obviously no edge

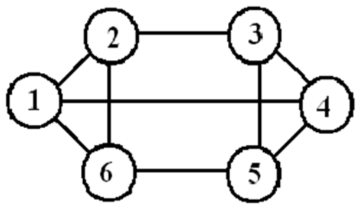


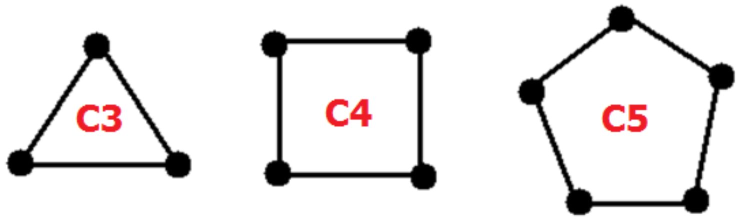
Trees
- Connected Acyclic Graph
- Two nodes have *exactly* one path between them
- Special ones are called 'paths' and 'star'



Regular

- Connected Graph
- All nodes have the same degree
- Special ones are called ‘cycles’ – see c3, c4, c5





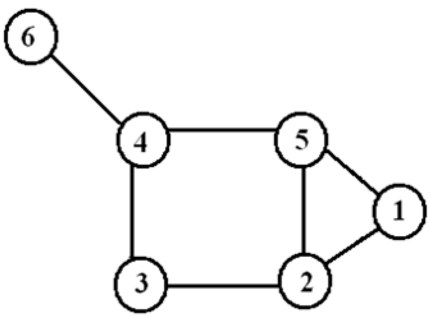
Representation (Matrix)

Incidence Matrix

- V x E
- [vertex, edges] contains the edge's data

Adjacency Matrix

- V x V
- Boolean values (adjacent or not)
- Or Edge Weights



$$
\begin{array}{c}
\begin{array}{ccccccc} 1,2 & 1,5 & 2,3 & 2,5 & 3,4 & 4,5 & 4,6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\end{array}
$$

Incidence Matrix

$$
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}
\begin{pmatrix}
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}
\end{array}
$$

Adjacency Matrix

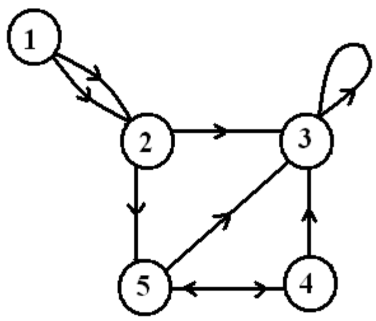
Representation (List)

Edge List

- pairs (ordered if directed) of vertices
- Optionally weight and other data

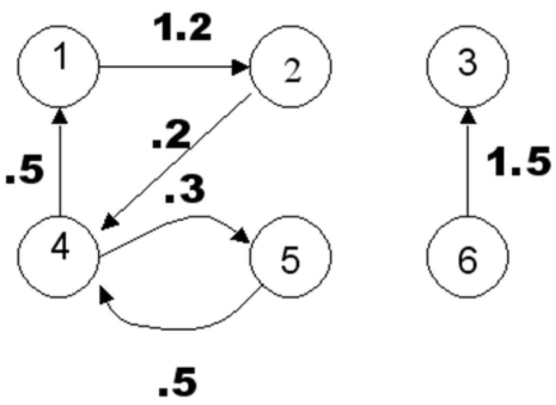
Adjacency List (node list)

- an array of | *V* | lists, one for each vertex in *V*.
- For each $u \in V$, *ADJ* [*u*] points to all its adjacent vertices.



Edge List	Node List
1 2	1 2 2
1 2	2 3 5
2 3	3 3
2 5	4 3 5
3 3	5 3 4
4 3	
4 5	
5 3	
5 4	

Edge List for Weighted Graphs



Edge List
1 2 1.2
2 4 0.2
4 5 0.3
4 1 0.5
5 4 0.5
6 3 1.5

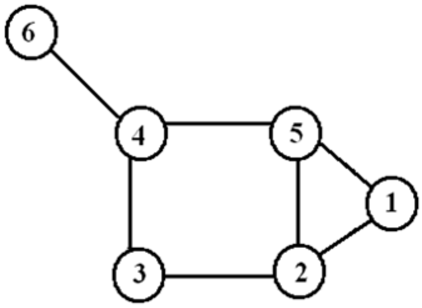
Topological Distance

A shortest path is the minimum path connecting two nodes.

The number of edges in the shortest path connecting *p* and *q* is the ***topological distance*** between these two nodes, $d_{p,q}$

Distance Matrix

$|V| \times |V|$ matrix $D = (d_{ij})$ such that d_{ij} is the topological distance between *i* and *j*.



$$\begin{array}{c c} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} 0 & 1 & 2 & 2 & 1 & 3 \\ 1 & 0 & 1 & 2 & 1 & 3 \\ 2 & 1 & 0 & 1 & 2 & 2 \\ 2 & 2 & 1 & 0 & 1 & 1 \\ 1 & 1 & 2 & 1 & 0 & 2 \\ 3 & 3 & 2 & 1 & 2 & 0 \end{pmatrix} \end{array}$$