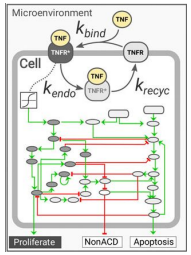
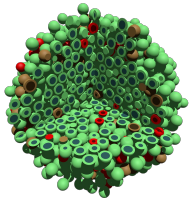
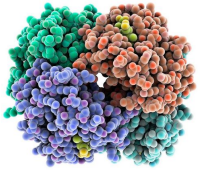
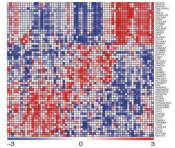




De las redes intracelulares a las simulaciones de sistemas multicelulares





Profesor: Dr. Miguel Ponce de León (miguel.ponce@bs.es) - BSC

Coordinador: Dr. Flavio Pazos (flavio.pazos@gmail.com) - IIBCE/IP

Apoyan:

PEDECIBA MEC-UDELAR

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Institut Pasteur de Montevideo

iibce
INSTITUTO DE INVESTIGACIONES BIOLÓGICAS CLEMENTE ESTABLE

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Tema 3

Fundamentos de teoría de grafos y redes complejas (Parte I)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Graph Theory I & II

- Why use network representation?
- A bit of history
- Basic definitions and graph types
- Graph representations
- Graph traversing and shortest path problems

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Levels of description of a system

- **Structure**
  - Components + interactions
- **Dynamics**
  - Steady states, Bi-stability
  - Oscillations, Chaos, etc
- **Control (regulation)**
  - Homeostasis (robustness)
  - Positive or Negative Feedback loops

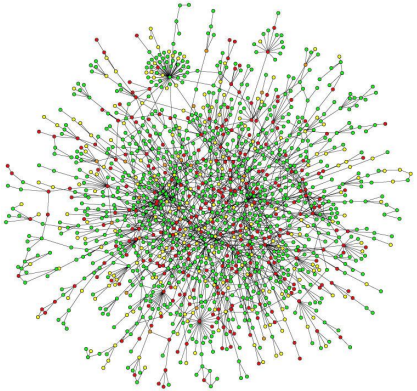
Complexity due to the large number of parameters and non-linear interactions!

"... With four parameters I can fit an elephant, and with five I can make him wiggle his trunk..."

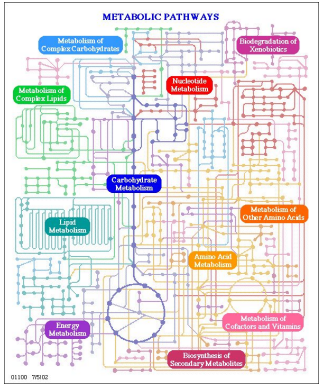
John von Neumann (~1953)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

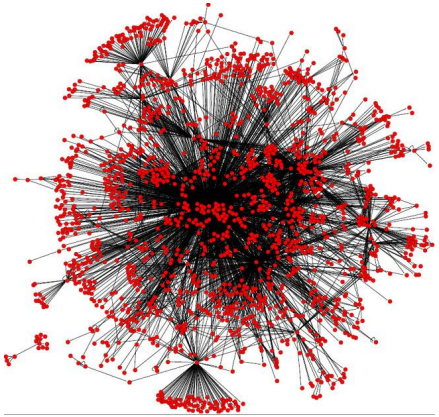
Networks are everywhere!



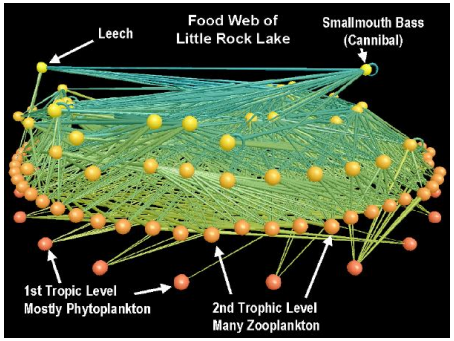
PPi networks



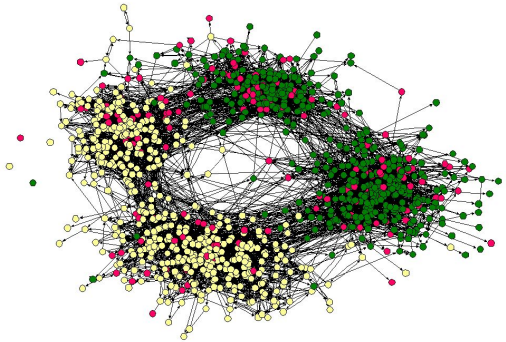
Metabolic networks



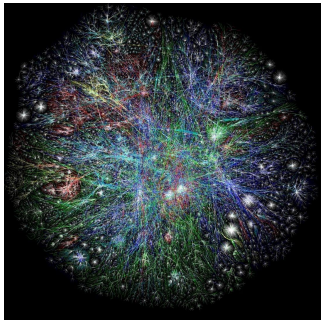
Regulatory networks



Ecological networks



Social networks



Technological networks

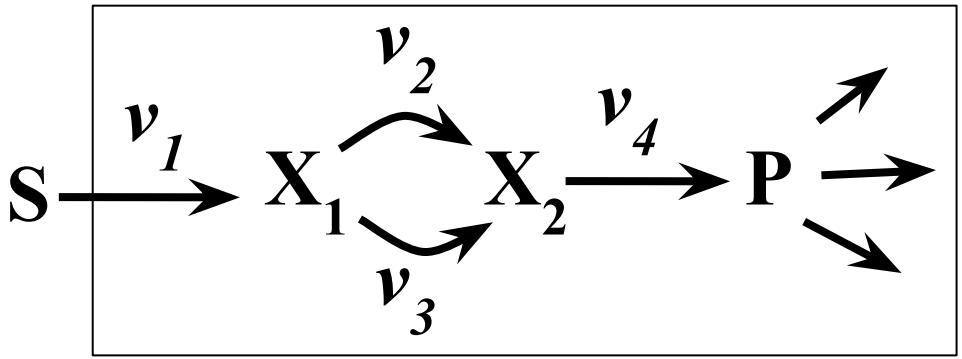
Properties of Models: behavior

Structure: relation between variables and parameters

Dynamic: time-evolution of systems variable

Causes:

- *External:* influences from the environment (input)
- *Internal:* processes within the system, noise, etc.

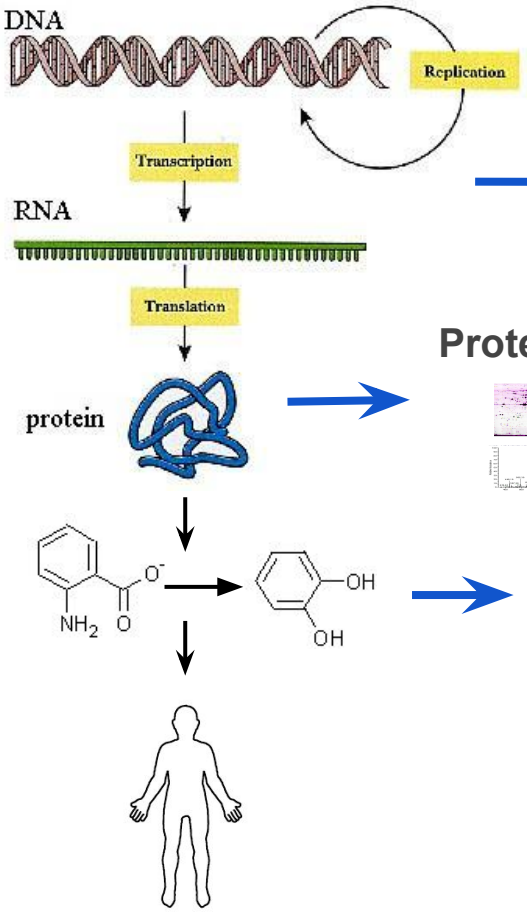


Structure

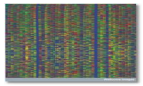
$$\begin{bmatrix} 1 & -1 & -1 & 0 \\ 0 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} \frac{d[A]}{dt} \\ \frac{d[B]}{dt} \end{bmatrix}$$

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

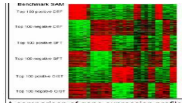
From omics to network reconstruction



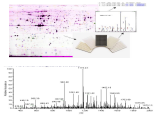
Genomics



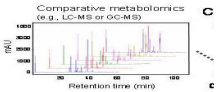
Transcriptomics



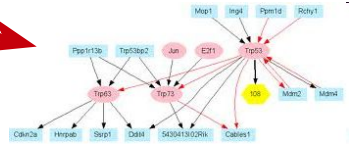
Proteomics



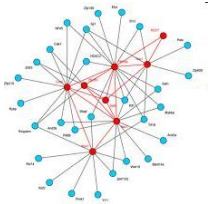
Metabolomics



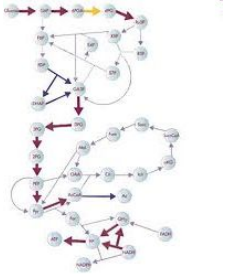
Gene regulation nets

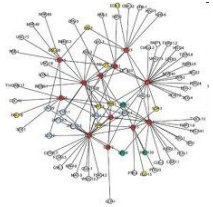


Protein nets (interaction, fosforilation, signaling)



Metabolic nets





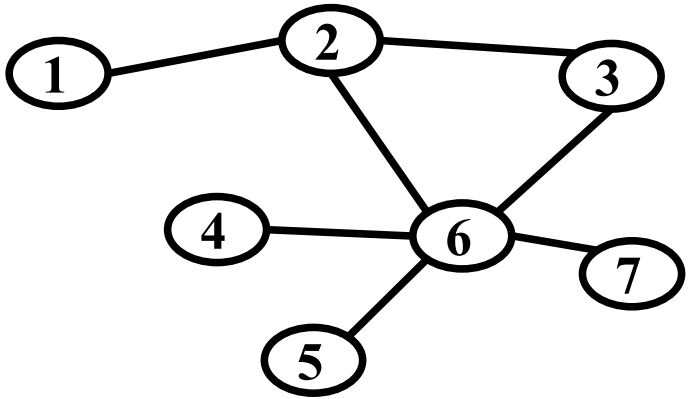
Genotype nets, phenotipic nets, disease-related nets, …

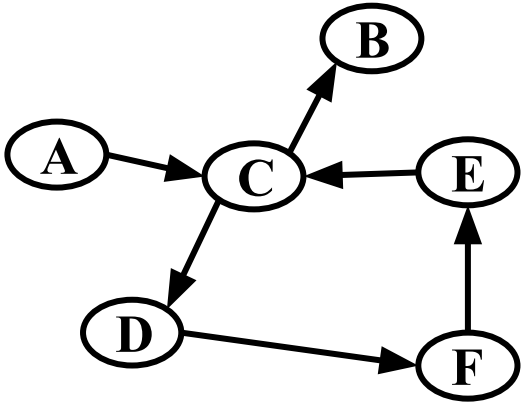
What is a Network?

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

What is a Network?

- **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 is network theory?

- ***Network theory*** provides a set of techniques for analyzing graphs
- ***Complex systems network theory*** provides techniques for represent and analyze any system of interacting agents, which can be represented as a network
- Applying network theory to a system means using a graph-theoretic representation

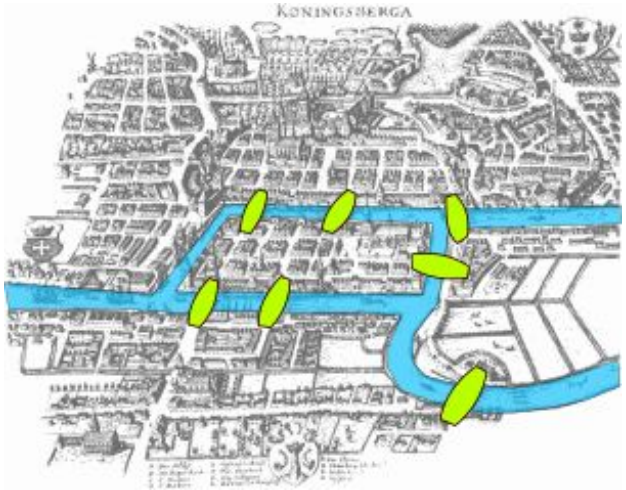
What makes a problem graph-like?

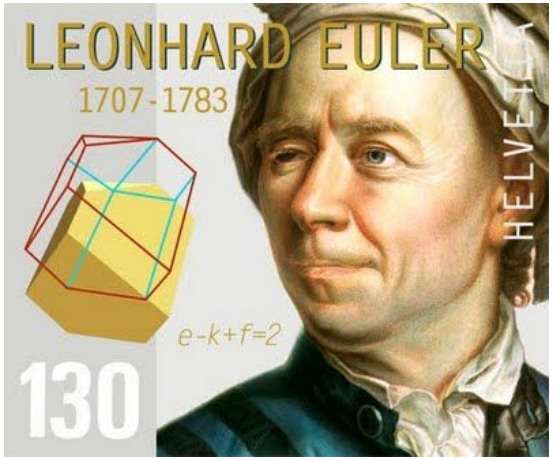
- There are two components to a graph
  - **Nodes** and **edges**
- In graph-like problems, these components have natural correspondences to problem elements
  - **Entities** are nodes and **interactions** between entities are edges
- Most complex systems are graph-like

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

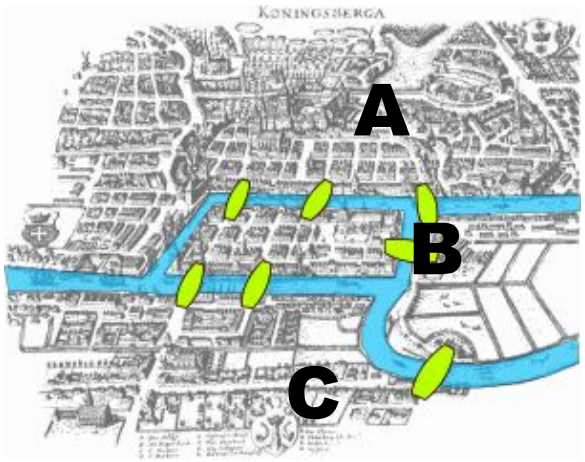
Introduction to graph theory: history

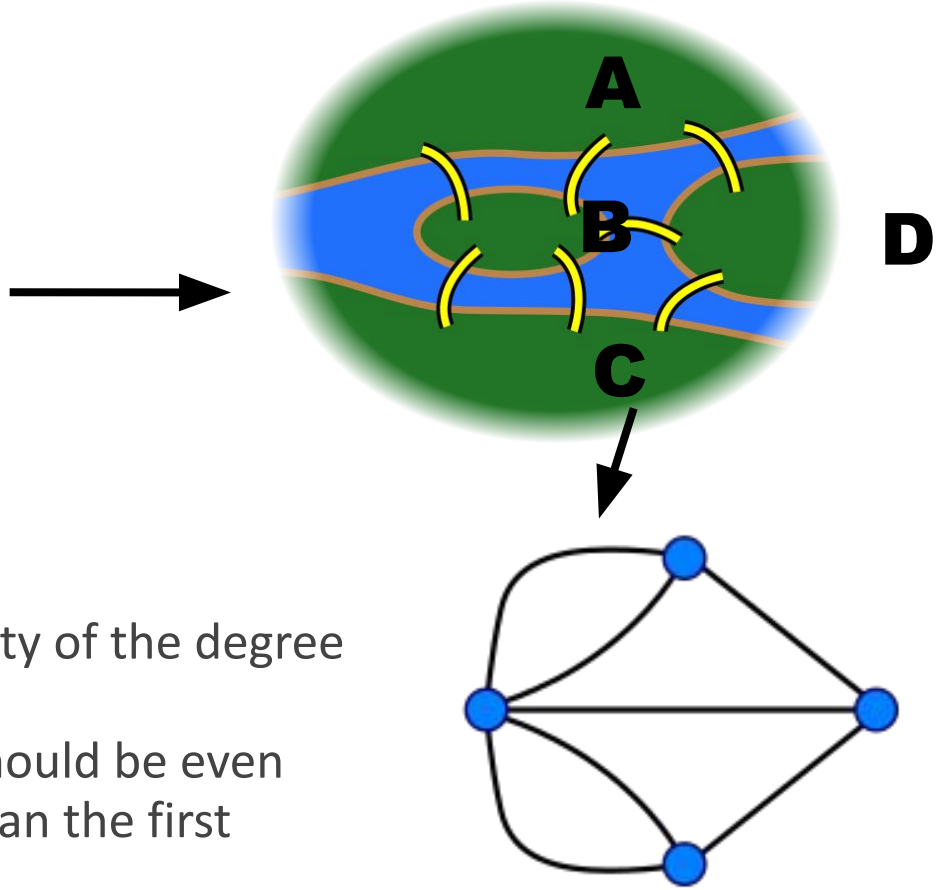
- **Graph** theory was born around ~1735
- Developed to solve Bridges of Konigsberg problem:
  - traverse through the 7 bridges of the city without crossing a bridge twice and go back to the origin





Introduction to graph theory: history



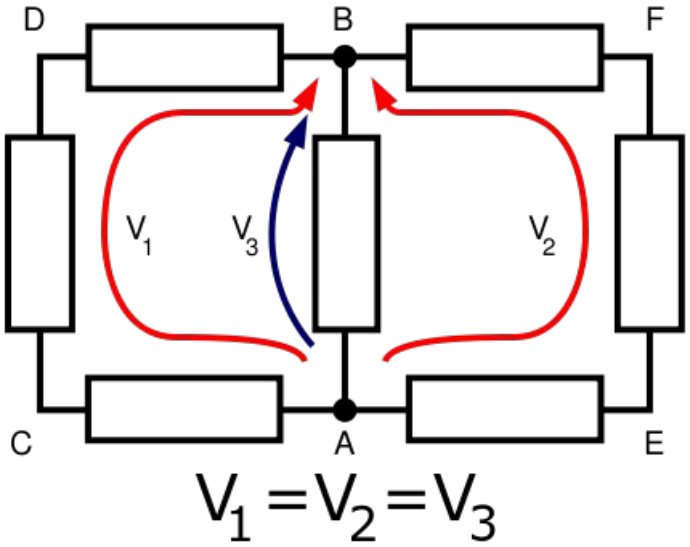


- The solution is based on the parity of the degree of each node
- Degree of intermediate nodes should be even
- The last node is the same one than the first

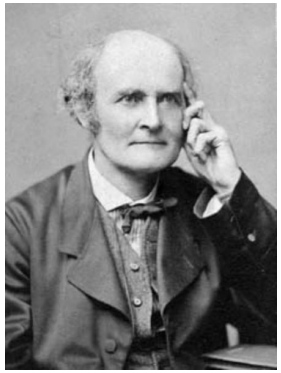
Introduction to graph theory: history

Trees in Electric Circuits

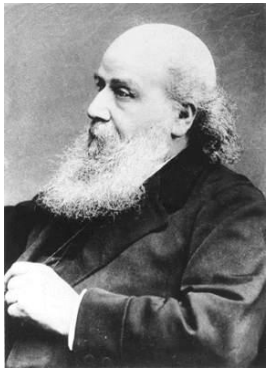
Gustav Kirchhoff



Introduction to graph theory: history

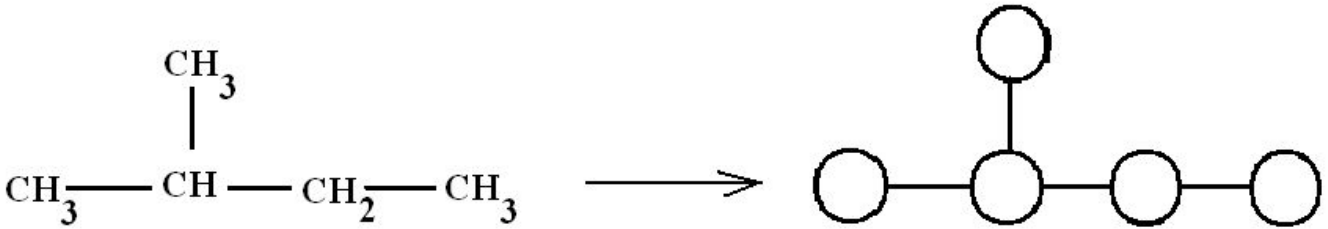


Arthur Cayley



James J. Sylvester

George Polya



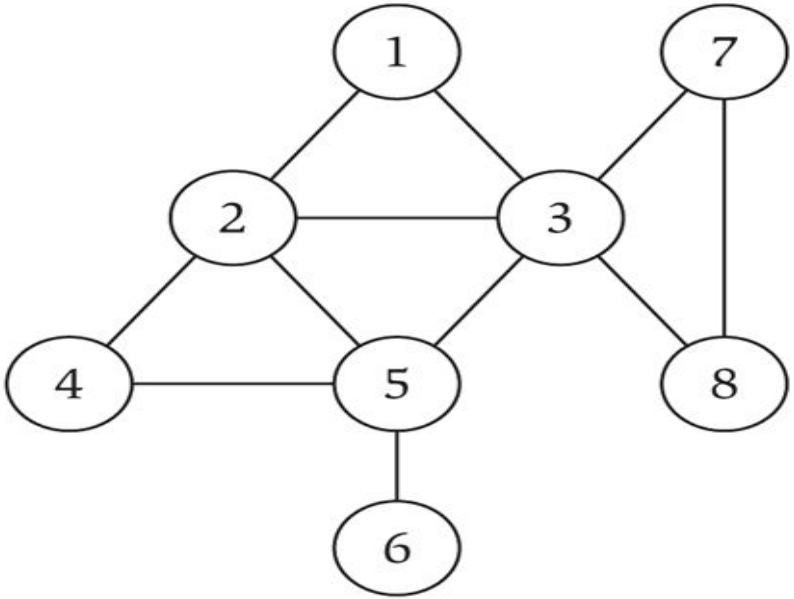
Graph theory: definitions

- **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: $n = |V|$, $m = |E|$.

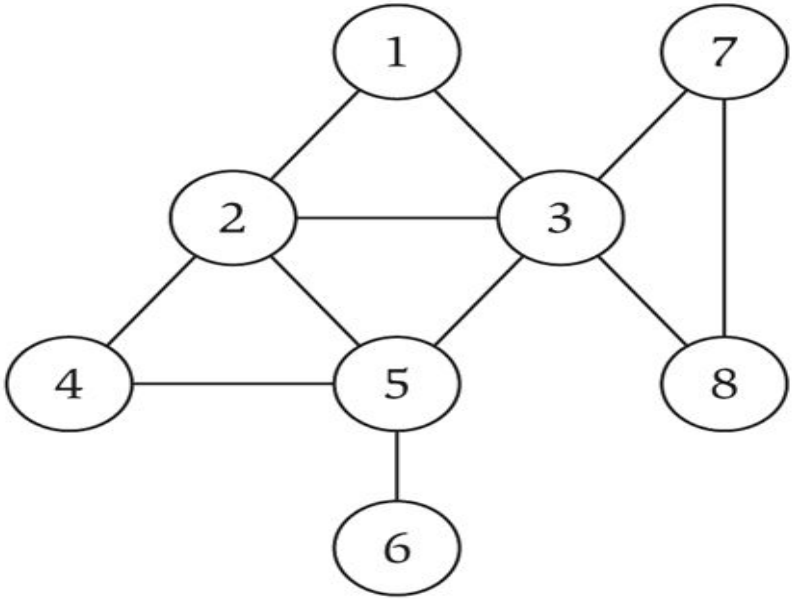
Graph theory: definitions

- **Vertex**
  - Basic Element
  - Drawn as a node or a dot.
  - Vertex set of G is usually denoted by **V(G)**, or $\mathbf{V_G}$
- **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 $\mathbf{E_G}$

Graph theory: example



Graph theory: example



- $V = \{ 1, 2, 3, 4, 5, 6, 7, 8 \}$
- $E = \{\{1,2\},\{1,3\},\{2,3\},\{2,4\},\{2,5\},\{3,5\},\{3,7\},\{3,8\},\{4,5\},\{5,6\}\}$
- $n = 8$
- $m = 11$

Graph theory: definitions

Set Notation

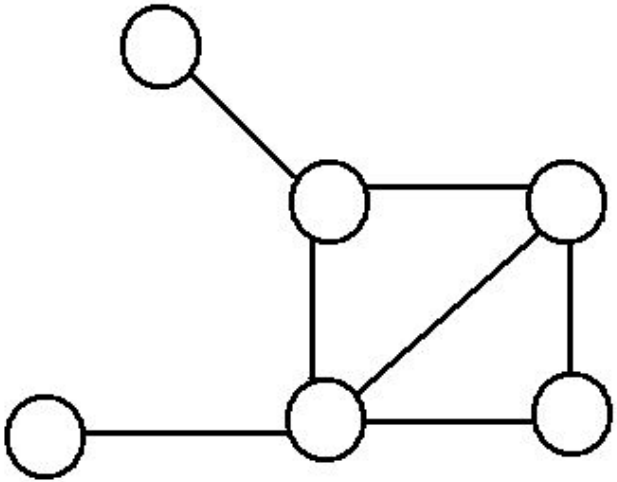
- $\mathbf{A = \{1, 2, 3\}}$ → Set definition (by extension)
- $\mathbf{A = \{x \,/\, x \in N, 0 < x < 4\}}$ → Set definition (comprehension)
- $\mathbf{A = B}$ → equal sets
- $\mathbf{A \cup B}$ → union of sets A and B
- $\mathbf{A \cap B}$ → intersection of sets A and B
- $\mathbf{A \subseteq B}$ → A is included in B
- $\boldsymbol{x} \in \mathbf{A}$ → x is an element of set A

Graph theory: definitions

- **For graph G(V,E):**
  - If edge e={u,v} ∈ E(G), we say that **u and v are adjacent** or **neigbors**
  - u and v are **incident** with e
  - u and v are end-vertices of e
  - An edge where the two end vertices are the same is called a **loop**, or a self-loop
- **Graph types:**
  - Simple or Undirected
  - Directed or Digraph
  - Mixed (some edges directed some undirected)
  - Weighted (weights on edges or nodes)

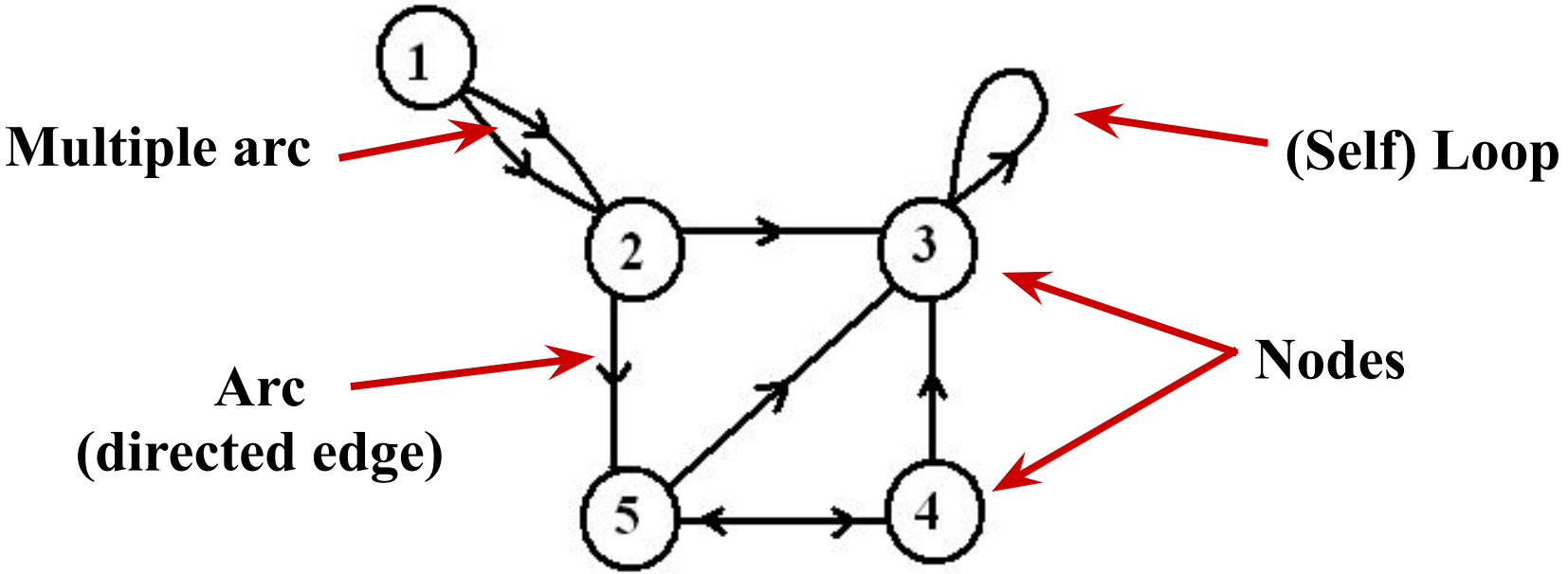
Simple Graphs

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



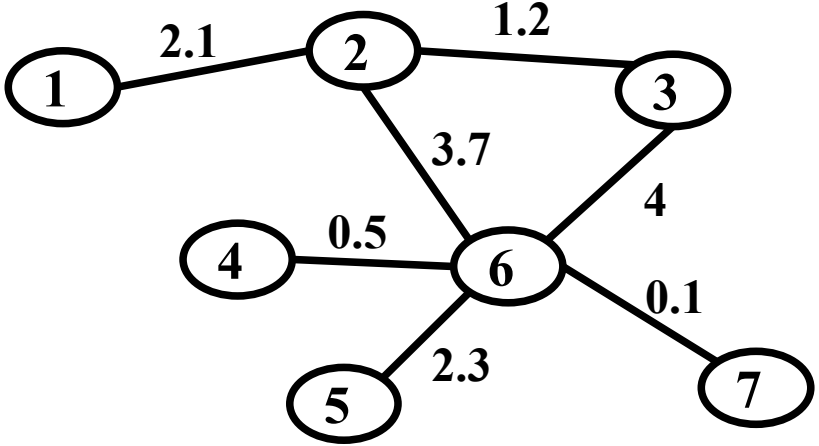
Directed Graph (digraph)

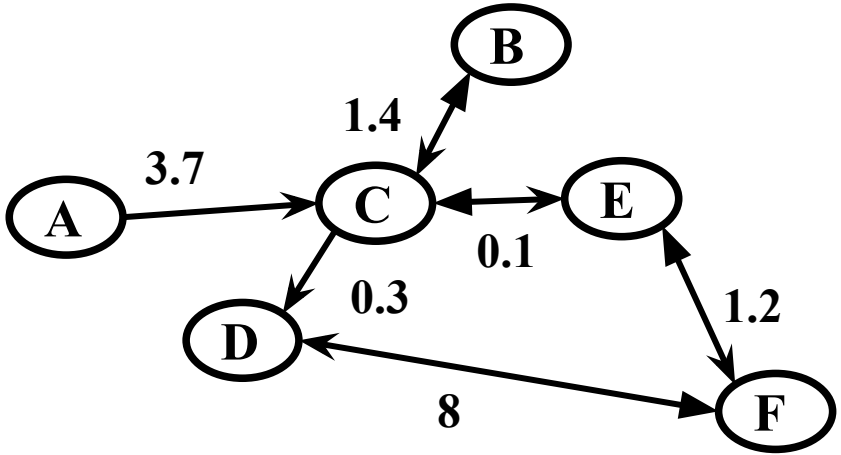
- Edges have **directions** (named arcs)
- An edge is an ordered pair of nodes



Weighted Graph: G = (V, E, w)

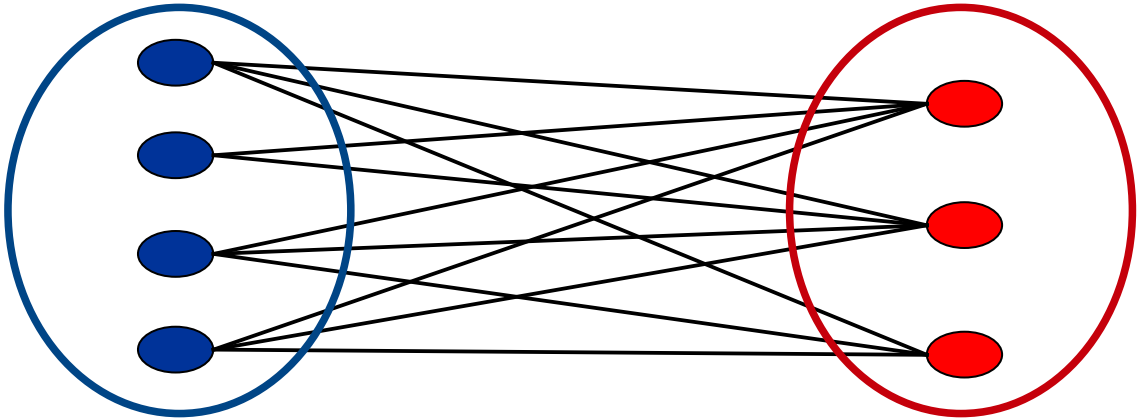
- Is a graph for which each edge has an associated ***weight***, usually given by a ***weight function*** *w:*
  - $E \rightarrow \mathbf{R}$ (*i.e.* each edge is assigned to a real values).





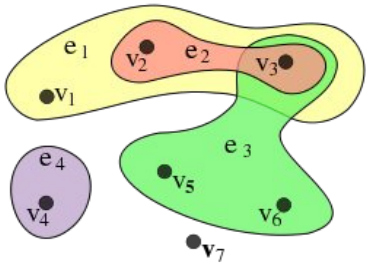
Bi-partite graphs

- **Def.** An undirected graph G = (V, E) is **bipartite** if the nodes can be colored red or blue such that every edge has one red and one blue end.
- **Applications:**
  - *Scheduling:* machines = red, jobs = blue.
  - *Metabolic networks:* metabolites = blue, enzymes = red.



Hypergrahs

- **Hypergraphs**: $E \subseteq 2^V$ (all subsets of elements of V)
  - Each edge (hyperedge) is a subset of vertices.
- **Example:**
  - $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$
  - $E = \{e_1, e_2, e_3, e_4\} = \{\{v_1, v_2, v_3\}, \{v_2, v_3\}, \{v_3, v_5, v_6\}, \{v_4\}\}$



- **Applications:**
  - Rarely used in network biology...
  - metabolic networks where several substrates react to build products.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Structures and structural metrics

- **Graph structures** are used to isolate interesting or important sections of a network

- **Structural metrics** provide **measurements** of structural properties of a graph or network:

  - **Local metrics:** refer to a single node in a graph (*e.g.* degree)

  - **Global metrics:** refer to a whole graph (*e.g.* diameter)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Structures and structural metrics

Graph structures are used to isolate interesting or important sections of a network

Structural metrics provide **measurements** of structural properties of a graph or network:

- **Local metrics:** refer to a single node in a graph (*e.g.* degree)
- **Global metrics:** refer to a whole graph (*e.g.* diameter)

Graph structures

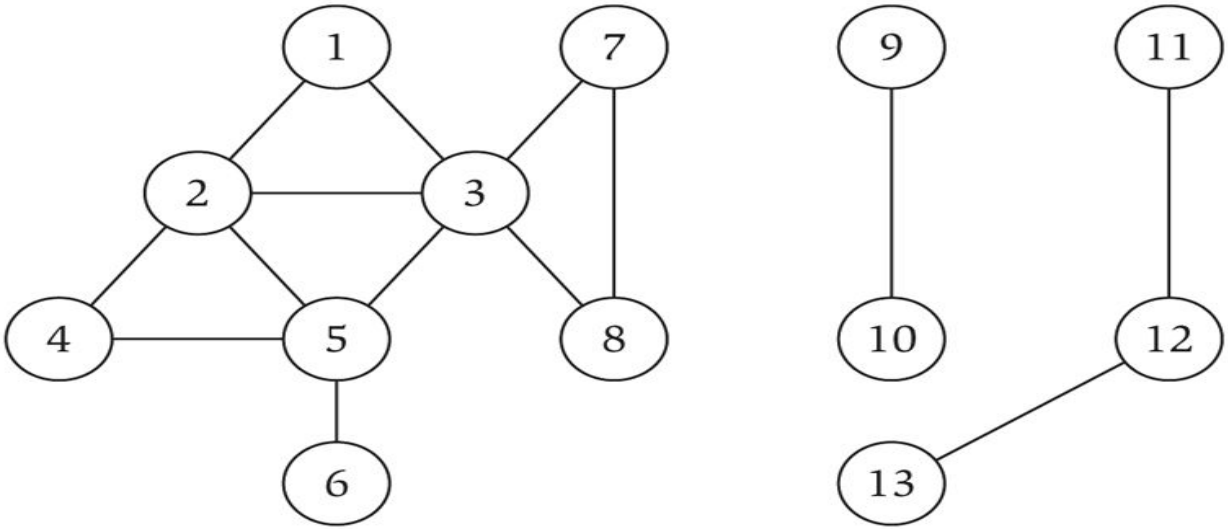
- Identify **interesting sections** of a graph
- Interesting because:
  - form a significant domain-specific structure
  - they significantly contribute to graph properties
- A subset of the nodes and edges in a graph that **possess certain characteristics**, or relate to each other in particular ways

Graph connectivity

- A graph is **connected** if
  - Any node can reach 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.

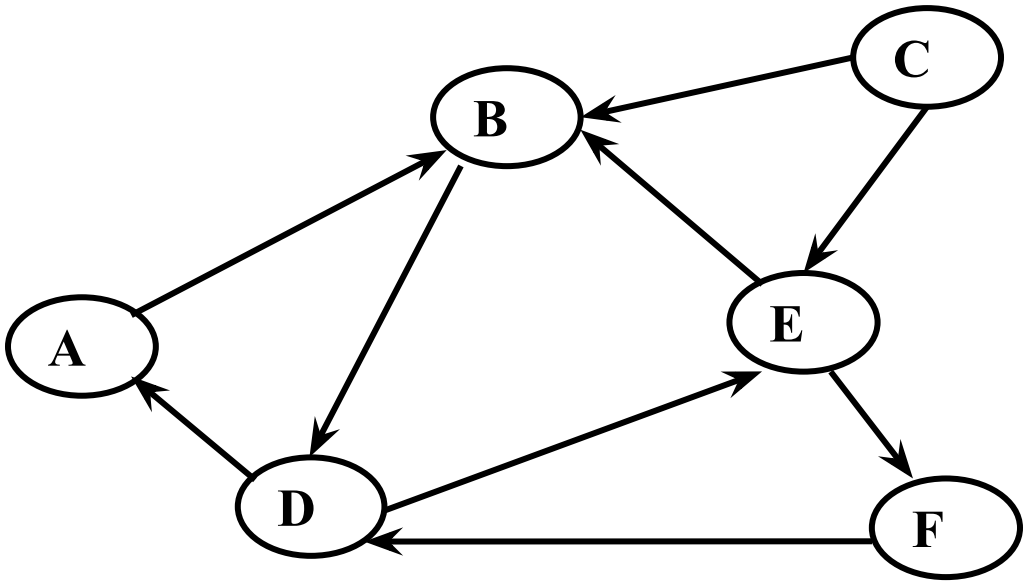
Connected Components

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



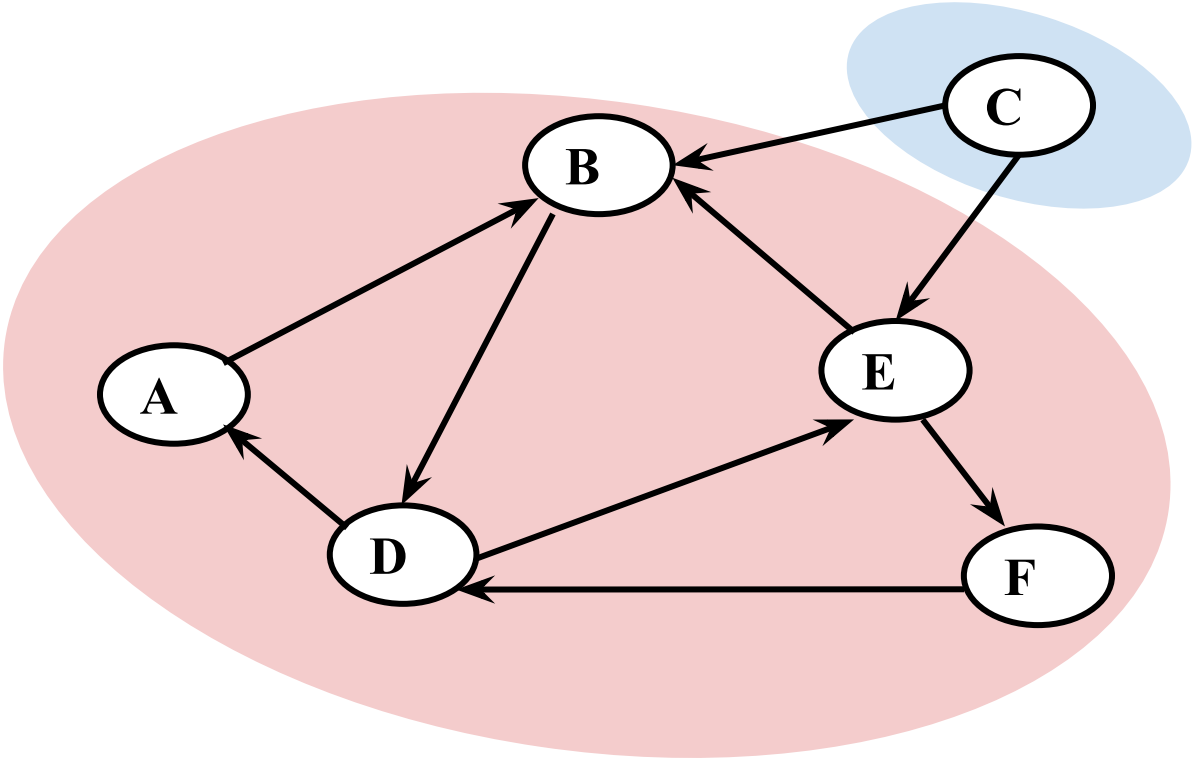
Strongly Connected Components

. How many **strongly connected components** are in the graph?



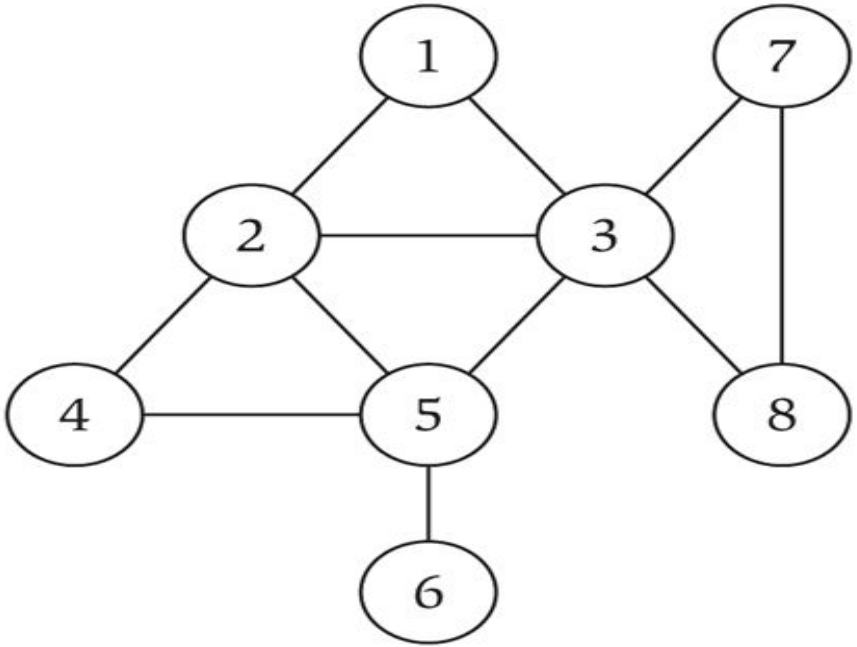
Strongly Connected Components

. How many **strongly connected components** are in the graph?



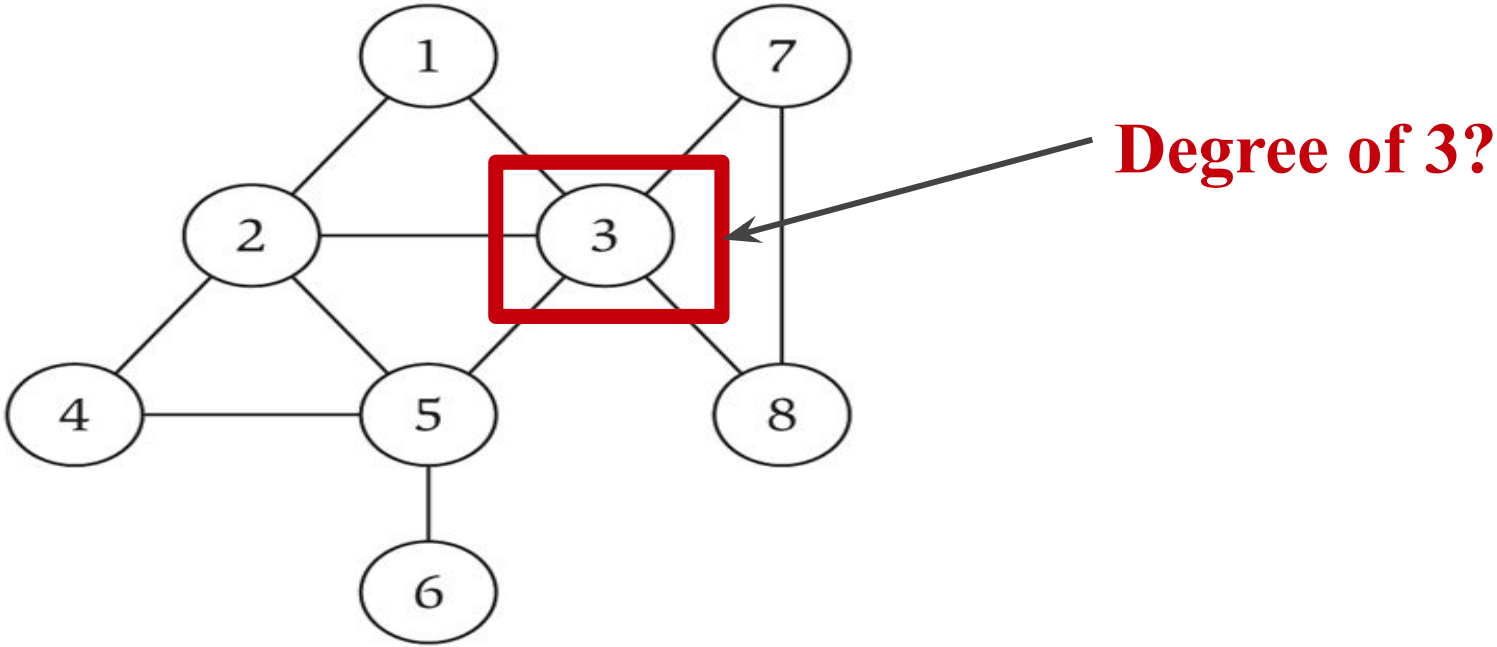
Degree of a node

- Number of edges incident on a node



Degree of a node

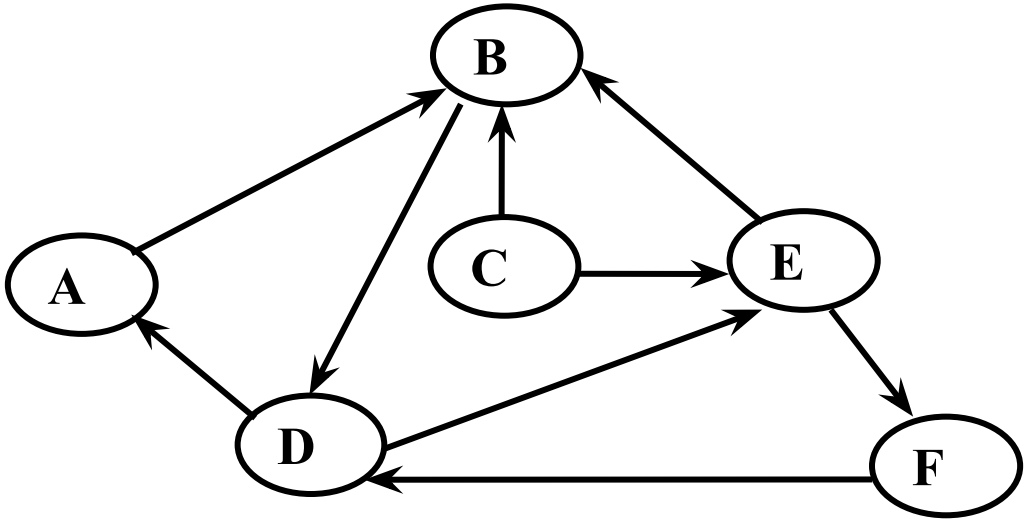
- Number of edges incident on a node



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Degree of a node (Di-graph)

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



Examples

- In-degree(**A**) = 1
- Out-degree(**A**) = 1

- In-degree(**B**) = 3
- Out-degree(**B**) = 1

- In-degree(**E**) = ?
- Out-degree(**E**) = ?

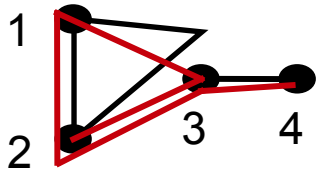
Degree: simple facts

- 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

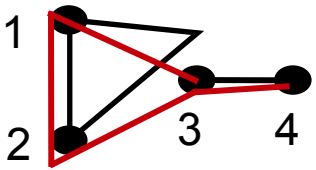
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Path and walks

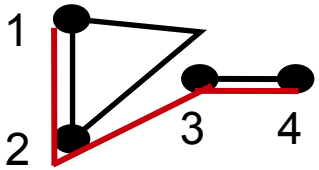
- **Def.** A **path** in an undirected graph G = (V, E) is a sequence P of nodes $v_1, v_2, \ldots, v_{L-1}, v_L$ where each
  - $(v_i, v_{i+1}) \in E$ for all $i = 1 \ldots L-1$
- (nodes can repeat, but edges do not.)
- **Def.** A **walk** is a path in which edges/nodes can be repeated.
- **Def.** A **path is simple** if all nodes are distinct.



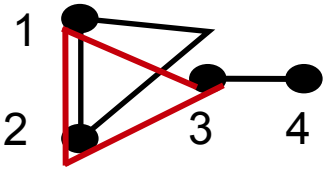
Walk:
{2,3}, {3,1}, {1,2}, {2,3}, {3,4}



Path:
{3,1}, {1,2}, {2,3}, {3,4}



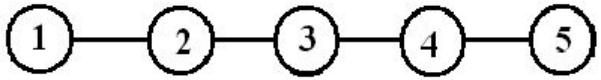
Simpl Path:
{1,2}, {2,3}, {3,4}



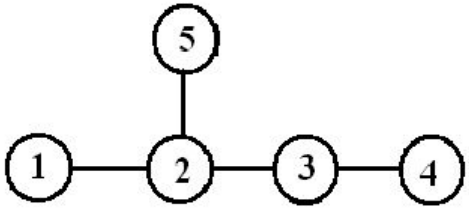
Cycle:
{3,1}, {1,2}, {2,3}

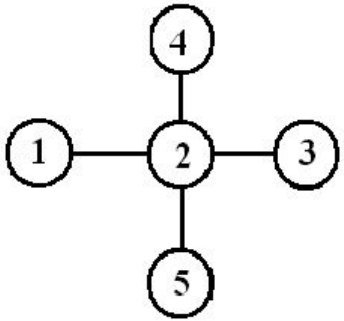
Trees

- Connected **Acyclic Graph**
- Two nodes have exactly one path between them
- **Examples:**



Path



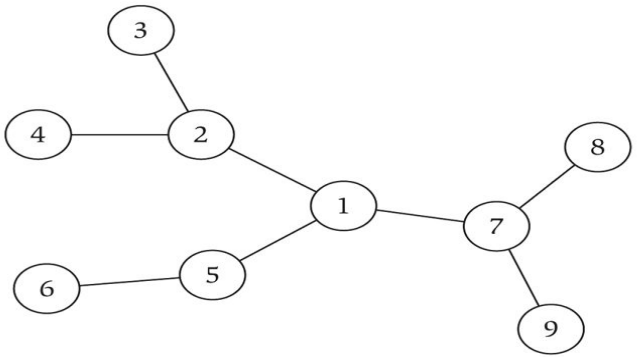


Star

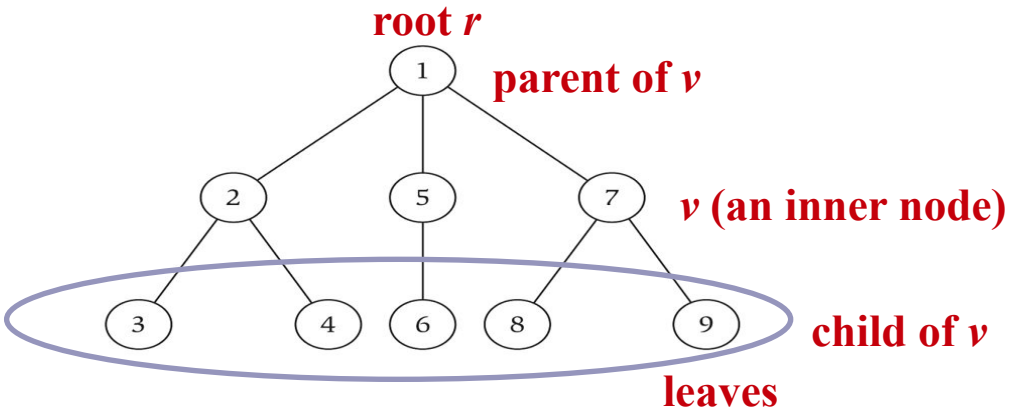
Trees and roots

- **Rooted tree.** Given a tree T, choose a root node r and orient each edge away from r.
- **Importance:** Models hierarchical structure.

a tree



the same tree, rooted at 1



Statistical properties of networks: degree distribution

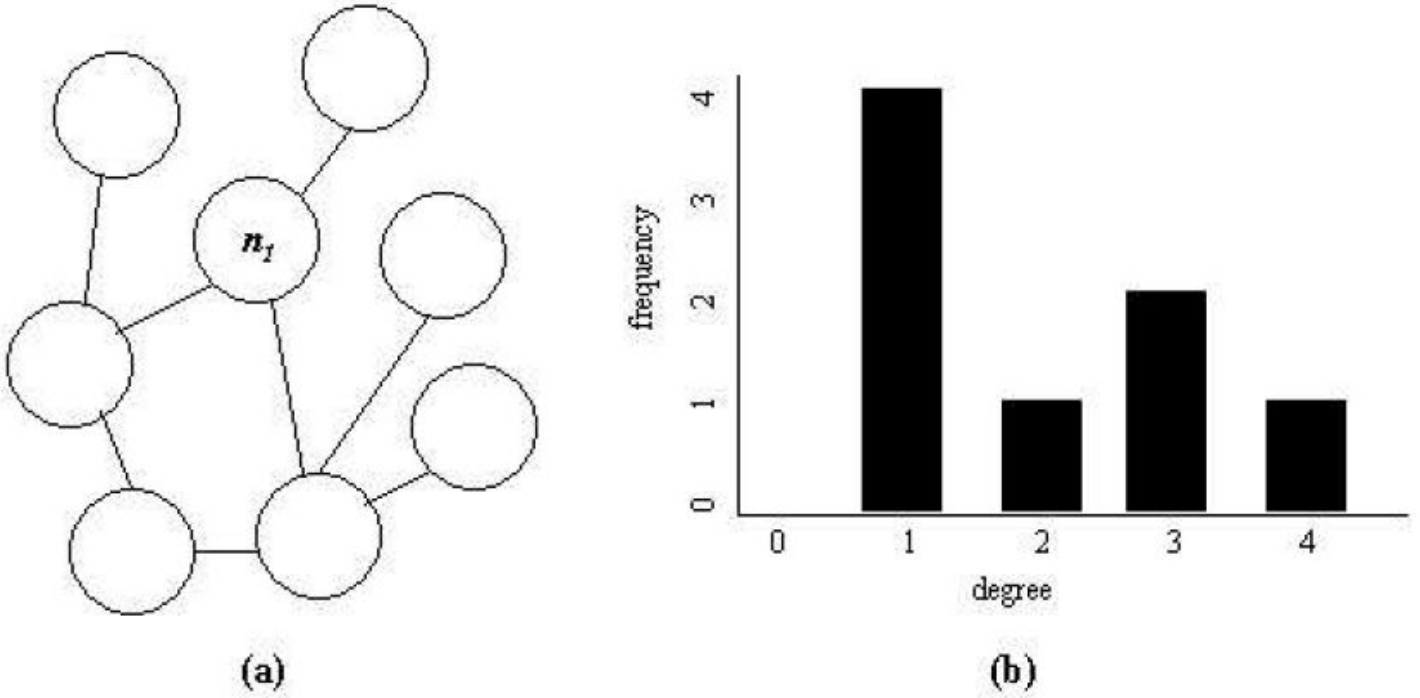


Fig. 1. (a) Example of a network. The node labeled n_1 has degree 3 (i.e., three links to other nodes). (b) Degree distribution of the network.

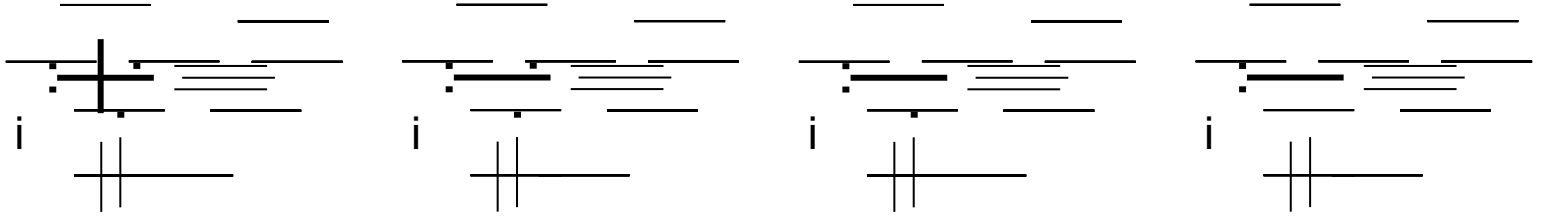
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Clusteing Coefficient (C)

The clustering coefficient of a node is the number of connections between its neighbors (*n*) over the maximum number of possible connections between them.

Ci = n / (ki*(ki-1)/2) (being *ki* the degree of the node = number of neighbors)

It gives an idea how clustered or sparse is the neighborhood of a node.



Ci=3/3 =1.0 Ci= 2/3 =0.67 Ci= 1/3 =0.33 Ci= 0/3 =0.0

The clustering coefficient of a graph is the average of the clustering coefficients of all its nodes. It gives an idea of how clustered/interconnected a graph is (vs. “star-like” graphs and trees)

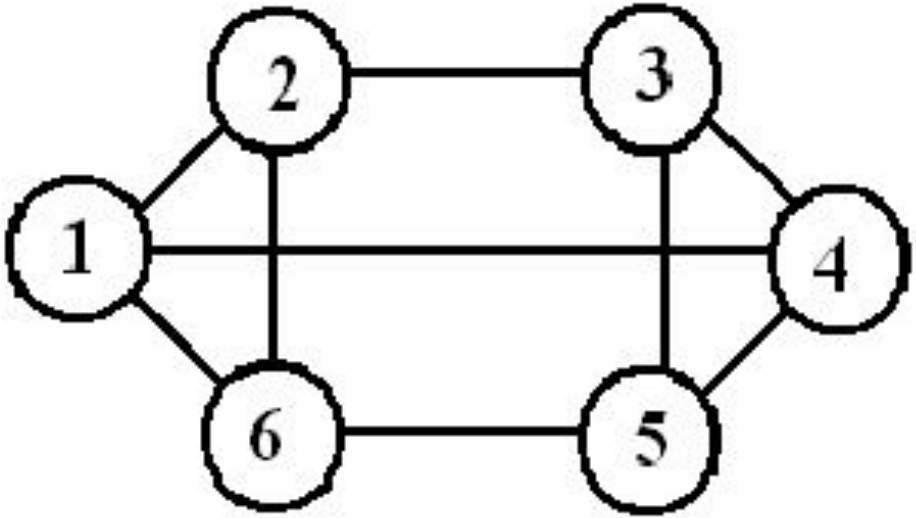
Characteristic path length and diameter

- The characteristic path length is the average of the lengths of the shortest paths (distances) between all nodes
  - $L = \langle d_{ij} \rangle$

- The diameter of a graph is the maximum path length.
  - $D = max(d_{ij})$

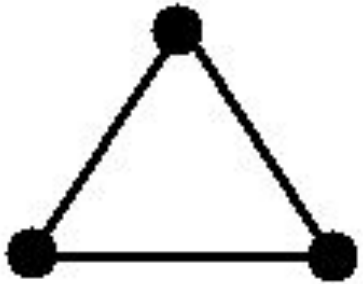
Regular graph

- Connected Graph
- All nodes have the same degree (a lattice or grid)

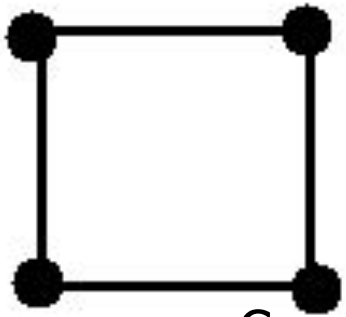


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

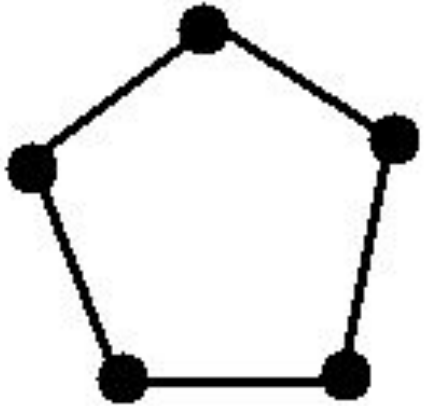
Special regular graph: Cycles



C_3



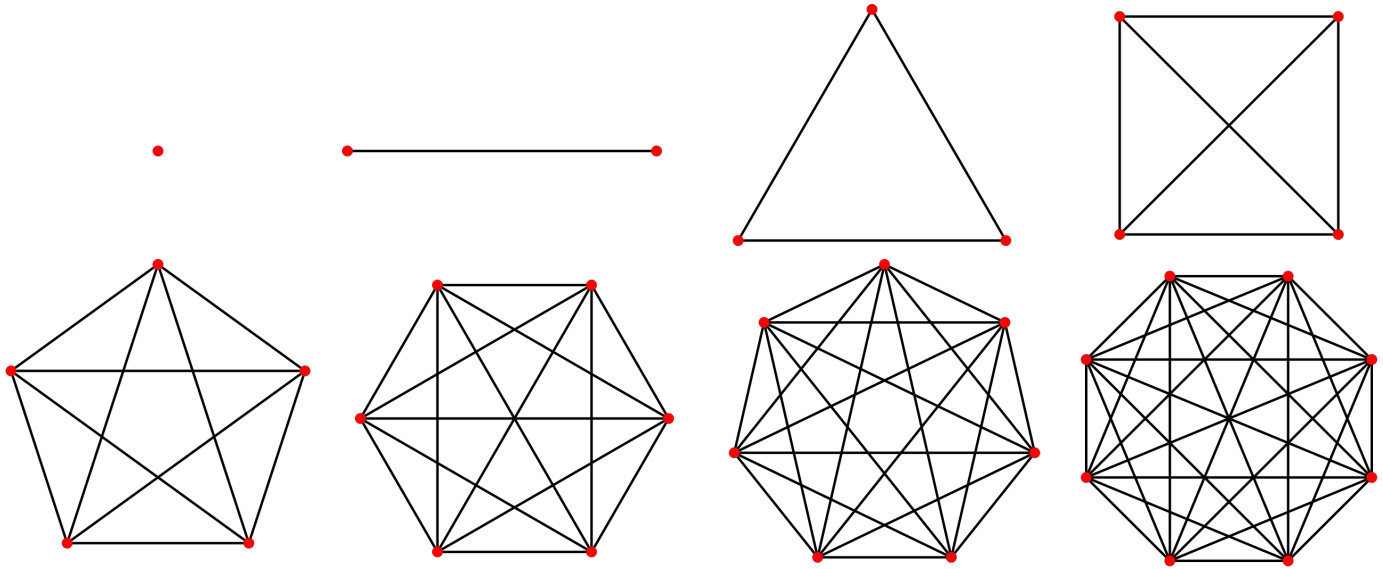
C_4



C_5

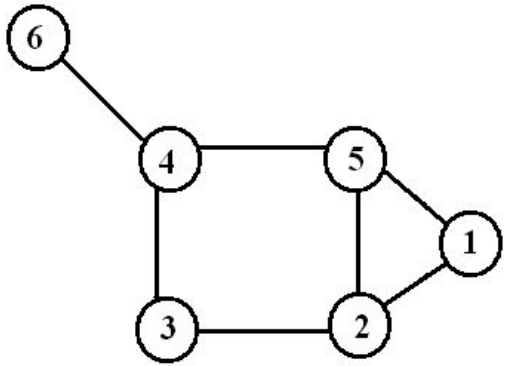
Special regular graph: Complete Graph

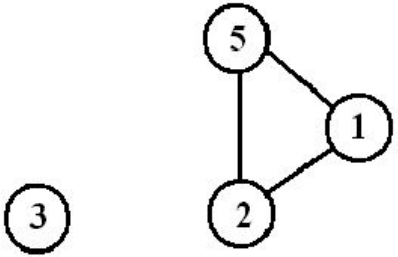
- Every pair of vertices are adjacent
- Has n(n-1)/2 edges



Subgraph

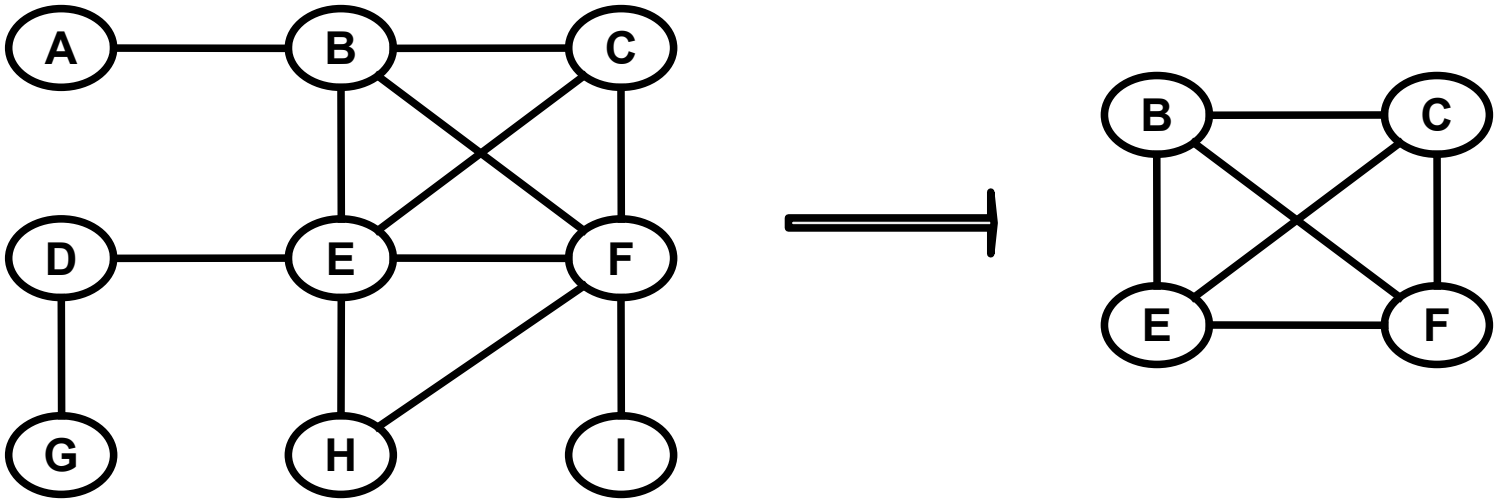
- Vertex and edge sets are subsets of those of G
  - H = (V',E') is a subgraph of G = (V,E) <=> V' $\subseteq$ V, E' $\subseteq$ E
- A **supergraph** of a graph G is a graph that contains G as a subgraph.





Special Subgraph: Cliques

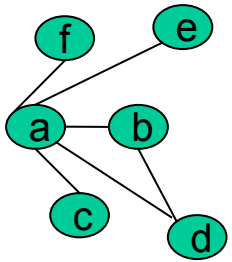
- A clique is a maximum complete connected subgraph.

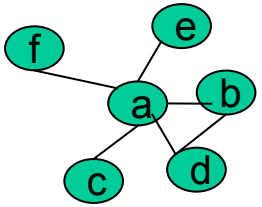


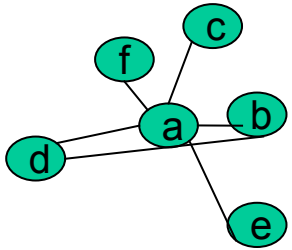
Graph representation: Visual layout

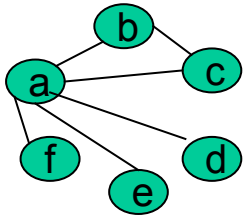
A **layout** is an arrangement of the graph nodes (and edges) in 2D or 3D which facilitates its visualization and/or makes more evident some properties of the graph.

G = ({a, b, c, d, e, f}, (a,b), (a,c), (a,d), (a,e), (a,f), (b,d))

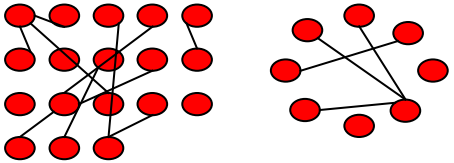




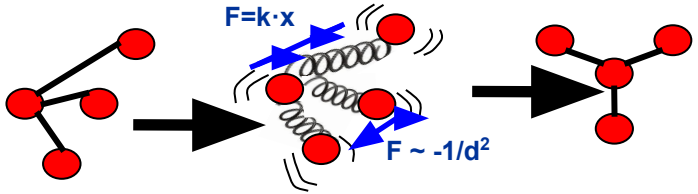




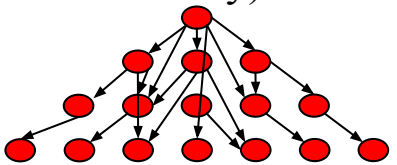
Regular arrangements



Based on "physics"



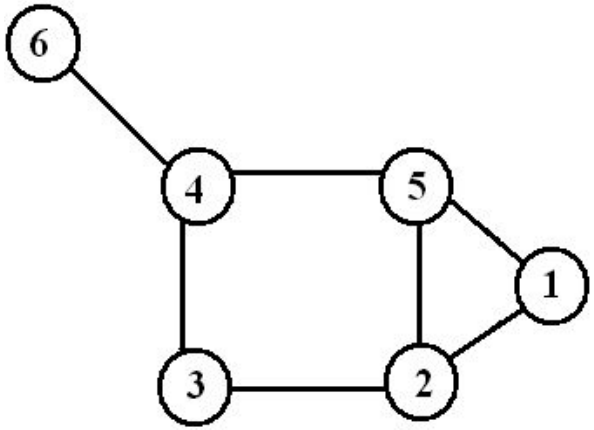
Based on topological properties (e.g. sort by k, to emphasize hierarchy)



Graph representation: Matrix

- **Adjacency Matrix**
  - Dimensions: |V| x |V|
  - Boolean values (adjacent or not)
  - Or Edge Weights
- **Incidence Matrix**
  - Dimensions: |V| x |E|
  - [vertex, edges] contains the edge's data

Graph representation: Matrix



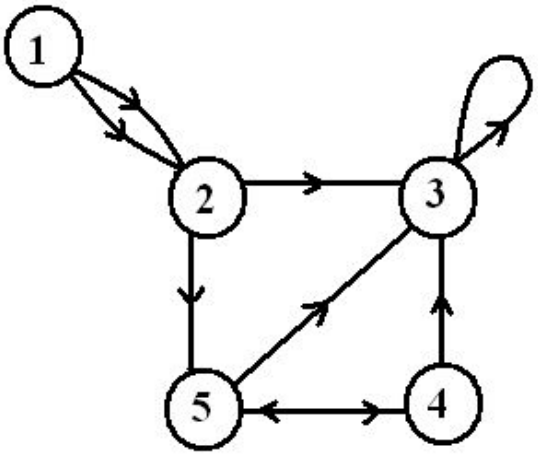
	1	2	3	4	5	6
1	0	1	0	0	1	0
2	1	0	1	0	1	0
3	0	1	0	1	0	0
4	0	0	1	0	1	1
5	1	1	0	1	0	0
6	0	0	0	1	0	0

	1,2	1,5	2,3	2,5	3,4	4,5	4,6
1	1	1	0	0	0	0	0
2	1	0	1	1	0	0	0
3	0	0	1	0	1	0	0
4	0	0	0	0	1	1	1
5	0	1	0	1	0	1	0
6	0	0	0	0	0	0	1

Graph 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 ∈ V , ADJ [u] points to all its adjacent vertices.

Graph representation: list



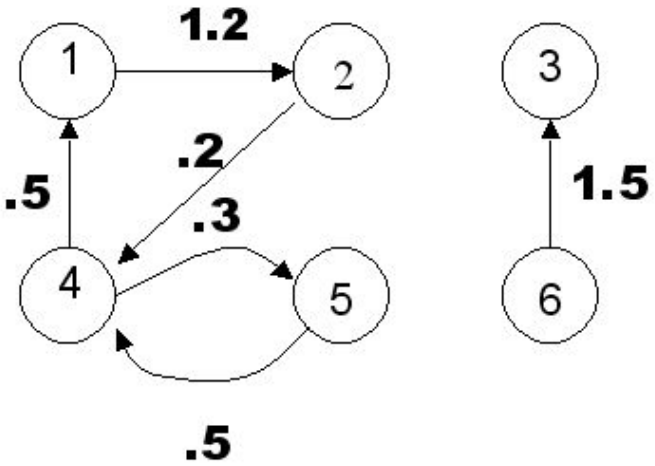
Edge List

- 1 2
- 1 2
- 2 3
- 2 5
- 3 3
- 4 3
- 4 5
- 5 3
- 5 4

Adjacency1 List

- 1 → 2 2
- 2 → 3 5
- 3 → 3
- 4 → 3 5
- 5 → 3 4

Edge Lists for Weighted Graphs

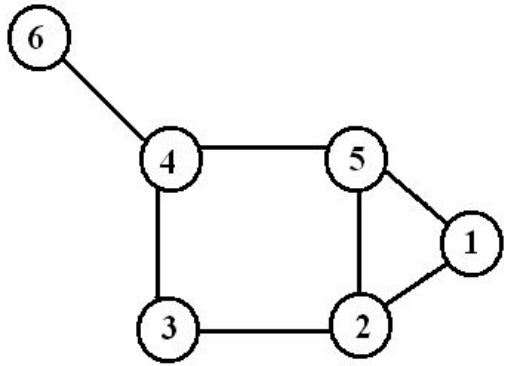


Edge List

. 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** between two vertices – a path of minimal length.
- **Length** – number of edges.
- **Distance** between u and v – the length of a shortest path between them (or ∞ if a path does not exist)
- **Distance matrix** D = (d_{ij}) $|V|$ x $|V|$ such that
  d_{ij} is the topological distance between i and j.

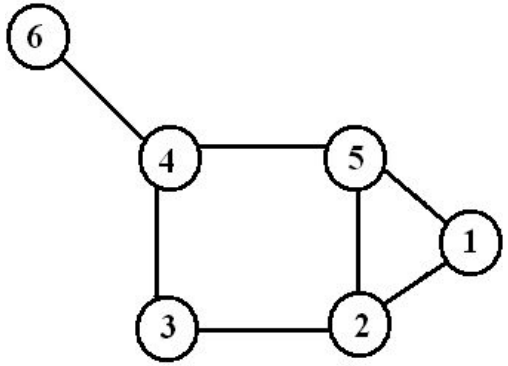


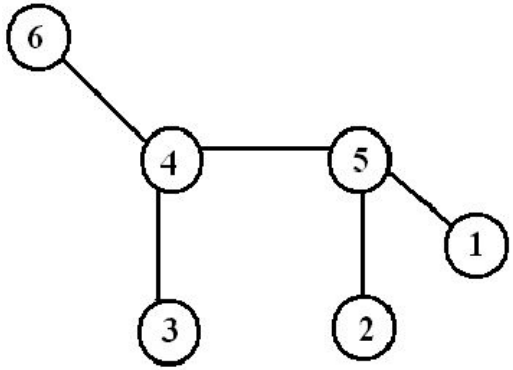
	1	2	3	4	5	6
1	0	1	2	2	1	3
2	1	0	1	2	1	3
3	2	1	0	1	2	2
4	2	2	1	0	1	1
5	1	1	2	1	0	2
6	3	3	2	1	2	0

Graph Algorithm

Spanning tree

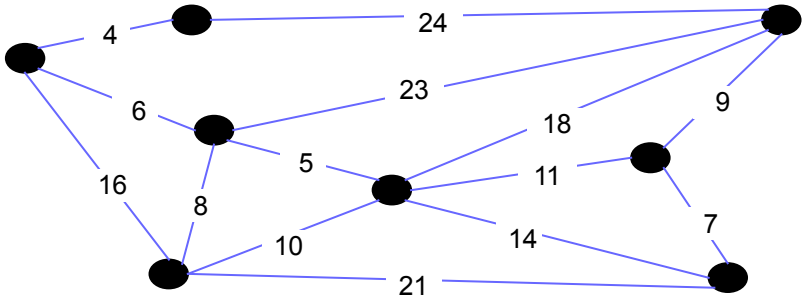
- Let G be a **connected** graph.
- A **spanning tree** in G is a **subgraph** of G that **includes every node** and **is also a tree**.



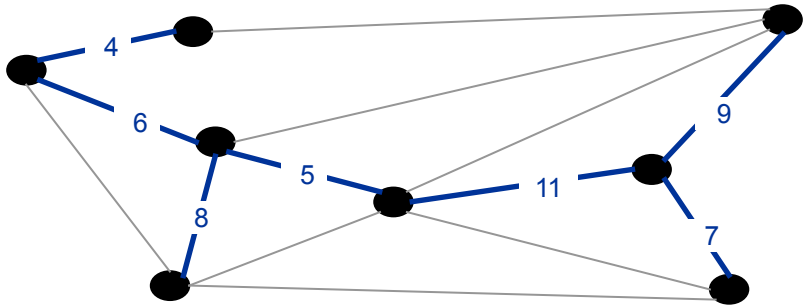


Algorithm: minimum spanning tree

- **Minimum spanning tree**.
- G = (V, E, w) connected weighted graph c_e, an **MST** is a subset of the edges T in E such that T is a spanning tree (contains all nodes of G) whose sum of edge weights is minimized.



G = (V, E)



T, $\Sigma_{e \in T} c_e = 50$

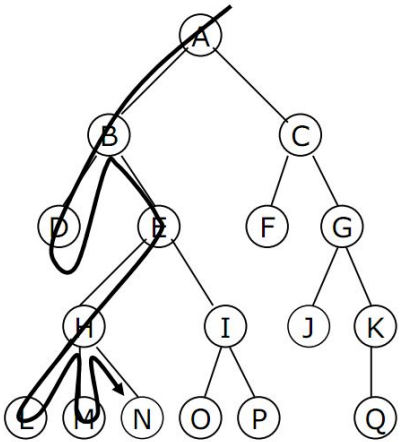
Cayley's Theorem: There are n^{n-2} spanning trees of K_n (can't use brute force).

MST applications

- **Network design:**
  - telephone, electrical, hydraulic, TV cable, computer, road
- **Indirect applications:**
  - identifying functional modules in weighted gene co-expression networks
  - predicting subgraph structure of protein complexess from affinity purification studies
  - Clustering
- Approximation algorithms for **NP-hard** problems:
  - traveling salesperson problem, Steiner tree
  - Kruskal's Algorithm

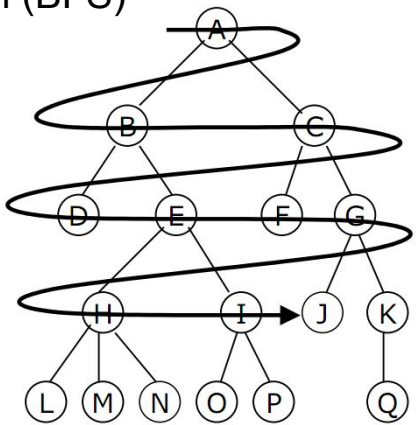
Graph Traversing

- Given a graph G(V,E), explore every vertex and every edge
- Using adjacency list is more efficient
- Example algorithms:
  - Depth-first search (DFS)



DFS

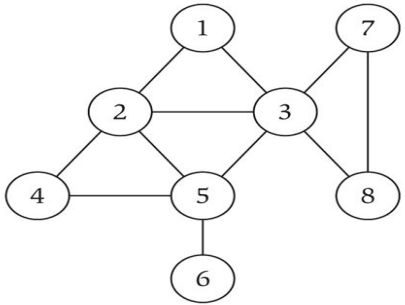
Breadth-first search (BFS)

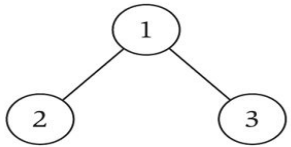


BFS

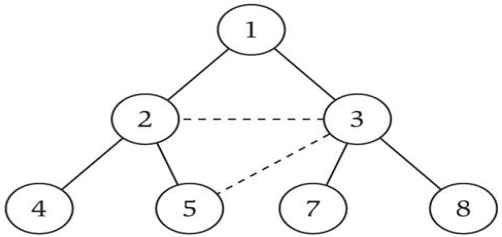
Graph Traversing

- BFS example:

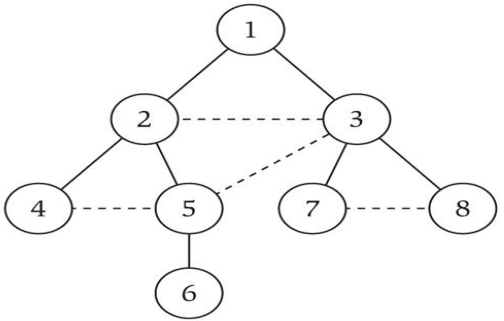




(a)



(b)



(c)

L_0

L_1

L_2

L_3

Graph Traversing

- Running time of BFS: linear, $O(|V|+|E|)$, using adjacency list
- BFS: Python code

```
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)
    for next in graph[start] - visited:
        dfs(graph, next, visited)
    return visited
```

Graph Traversing

- DFS applications:
  - Determines whether G is connected
  - Computes the connected components of G (strongly connected components of a digraph = directed graph)
  - Path / cycle finding
  - Topological sort (ordering of vertices of digraph G(V,E) such that for every edge (u,v) in E, u appears before v in the ordering)
  - Linear running time

- BFS applications:
  - Computes the distance from s to each reachable vertex in unewighted G
  - Finds shortest paths from s to all other nodes in unweighted G
  - Finds a simple cycle, if there is one
  - Computes the connected components of G

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Bibliography

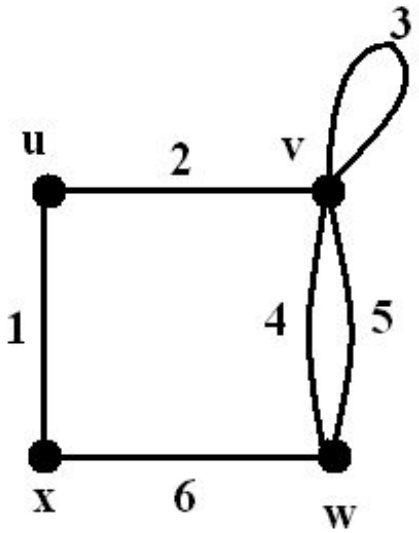
- *"Algorithm design"* - Chapter 3, Jon Kleinberg & Eva Tarods
- *"Graph Theory with Applications"*, J. A. Bondy and U. S. R. Murty
- *"Lecture Notes on Graph Theory"*, Tero Harju
- *"Statistical mechanics of complex networks"* Albert Lazlo Barabasi & Reka Albert
- The web….

Exercise

Exercise 1

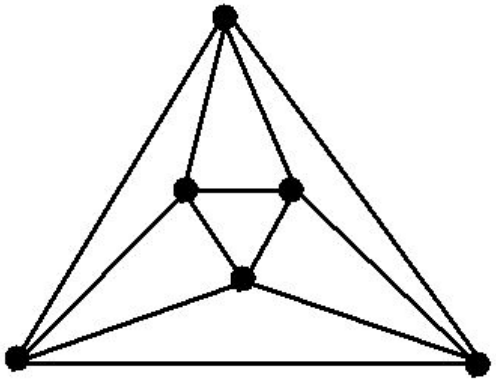
Which of the following statements hold for this graph?

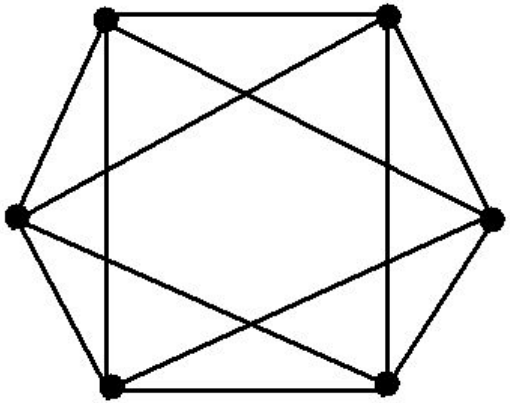
(a) nodes v and w are adjacent;
(b) nodes v and x are adjacent;
(c) node u is incident with edge 2;
(d) edge 5 is incident with node x.



Exercise 2

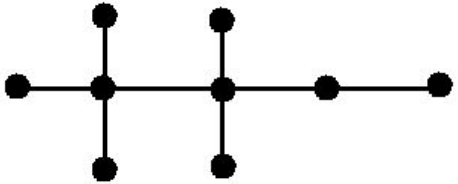
Are the following two graphs isomorphic?



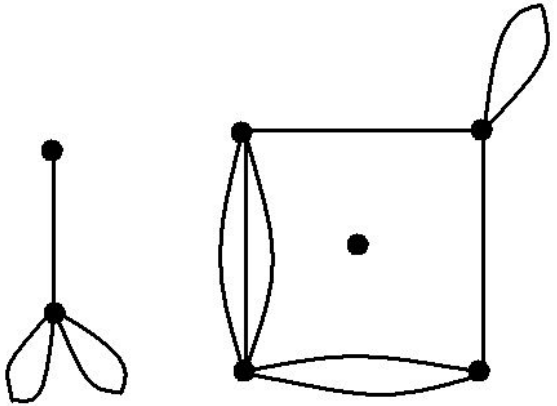


Exercise 3

Write down the degree sequence of each of the following graphs:



(a)



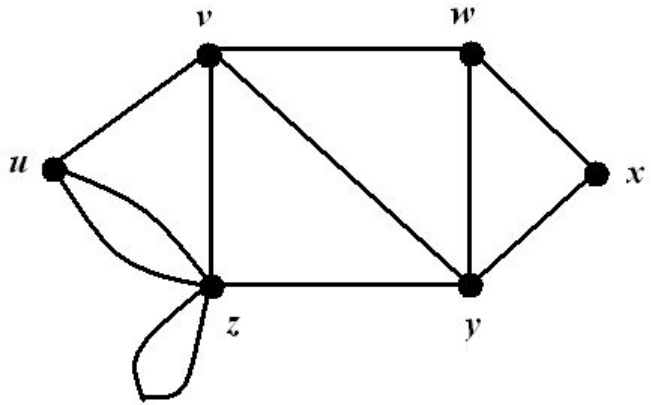
(b)

Exercise 4

Complete the following statements concerning the graph given below:

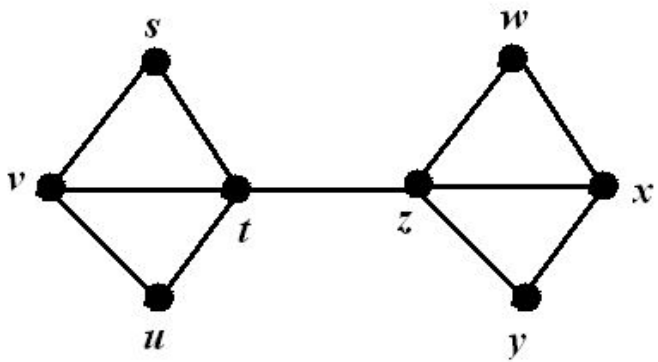
(a) *xyzzvy* is a _____of length____between___and__;

(b) *uvyz* is _____of length ____between___and__.



Exercise 5

Write down all the paths between s and y in the following graph. Build the distance matrix of the graph.



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Exercise 6

Draw the graphs given by the following representations:

Node list

1 2 3 4
2 4
3 4
4 1 2 3
5 6
6 5

Edge list

1 2
1 4
2 2
2 4
2 4
3 2
4 3

Adjacency matrix

	1	2	3	4	5
1	0	2	0	1	1
2	2	0	0	1	1
3	0	0	0	0	0
4	1	1	0	0	2
5	1	1	0	2	0