



POLITECNICO  
DI TORINO



e-Lite



# Introduction to Graphs

Tecniche di Programmazione – A.A. 2019/2020



# Summary

---

- ▶ Definition: Graph
- ▶ Related Definitions
- ▶ Applications

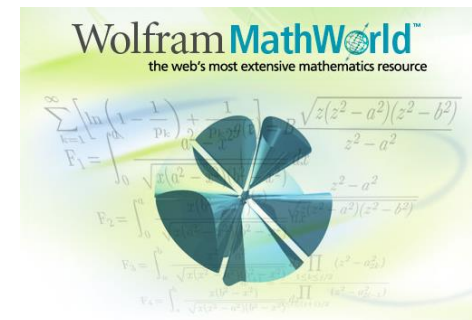


# Definition: **Graph**

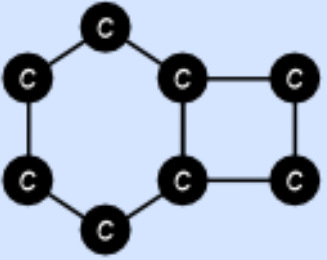
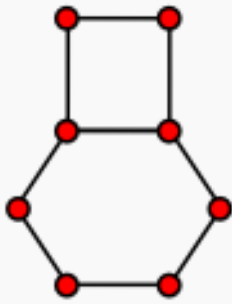
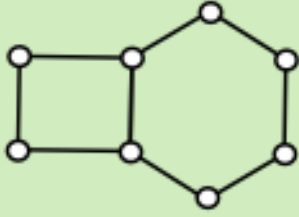

---

- ▶ A **graph** is a collection of **points** and **lines** connecting some (possibly empty) subset of them.
- ▶ The points of a graph are most commonly known as **graph vertices**, but may also be called “nodes” or simply “points.”
- ▶ The lines connecting the vertices of a graph are most commonly known as **graph edges**, but may also be called “arcs” or “lines.”

<http://mathworld.wolfram.com/>



# What's in a name?

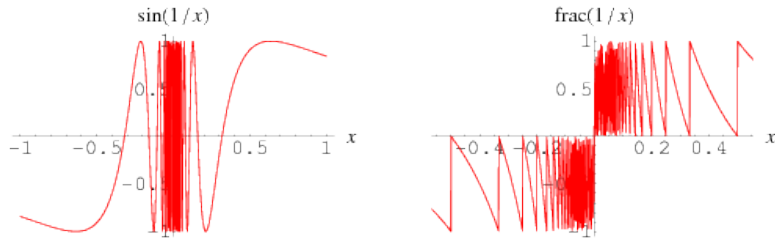
CHEMISTRY	SOCIAL NETWORKS	BIOLOGY	MATH
			<p>THEY LOOK THE SAME TO ME.</p> <p>LET'S CALL IT A GRAPH.</p> 
<p>BENZOCYCLOBUTADIENE</p> <p><b>C</b> CARBON ATOMS</p> <p>— <math>\sigma</math>-ELECTRON BONDS</p>	<p>spikedmath.com © 2011</p> <p><b>●</b> INDIVIDUALS</p> <p>— FRIENDSHIPS</p>	<p>PPI (SUB)NETWORK OF A SIMPLE ORGANISM</p> <p><b>○</b> PROTEINS</p> <p>— INTERACTIONS</p>	

**"MATHEMATICS IS THE ART OF GIVING THE SAME NAME TO DIFFERENT THINGS."**  
JULES HENRI POINCARÉ (1854-1912)

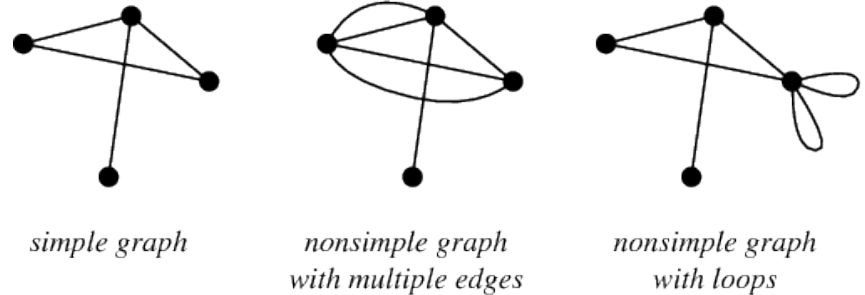
<http://spikedmath.com/382.html>

# Big warning: Graph $\neq$ Graph $\neq$ Graph

**Graph (plot)**  
(italiano: grafico)

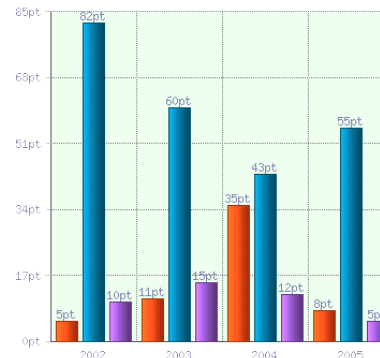


**Graph (maths)**  
(italiano: grafo)



$\neq$

**Graph (chart)**  
(italiano: grafico)



# History

---

- ▶ The study of graphs is known as **graph theory**, and was first systematically investigated by D. König in the 1930s
- ▶ Euler's proof about the *walk across all seven bridges of Königsberg* (1736), now known as the *Königsberg bridge problem*, is a famous precursor to graph theory.
- ▶ In fact, the study of various sorts of paths in graphs has many applications in real-world problems.

# Königsberg Bridge Problem

- ▶ Can the 7 bridges of the city of Königsberg over the river Pregel all be traversed in a single trip without doubling back, with the additional requirement that the trip ends in the same place it began?

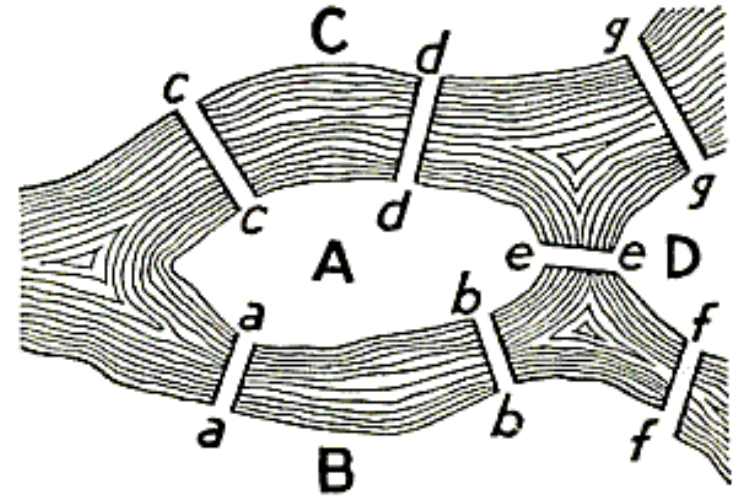
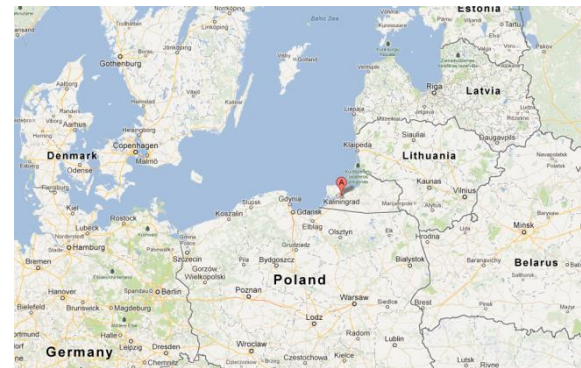


FIGURE 98. *Geographic Map:  
The Königsberg Bridges.*



Today: Kaliningrad, Russia



# Königsberg Bridge Problem

- ▶ Can the 7 bridges of the city of Königsberg over the river Pregel all be traversed in a single trip without repeating any bridge? If not, how many trips are required to traverse all bridges at least once?

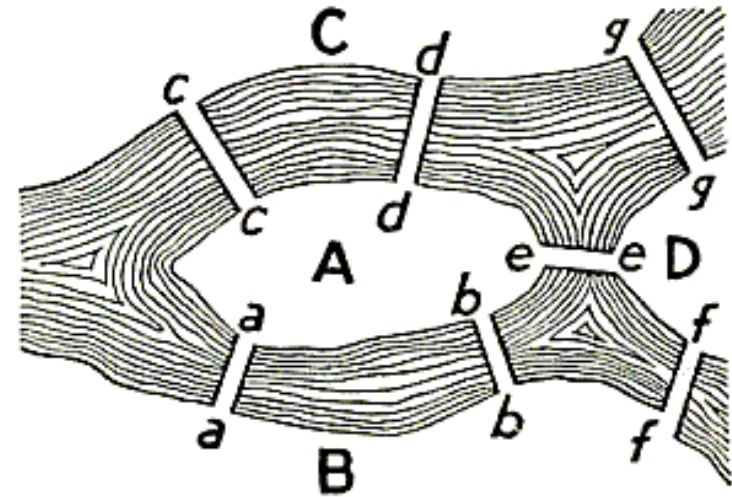
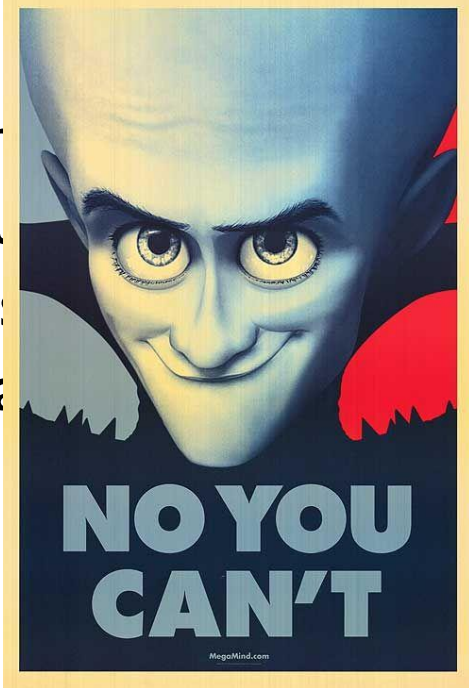
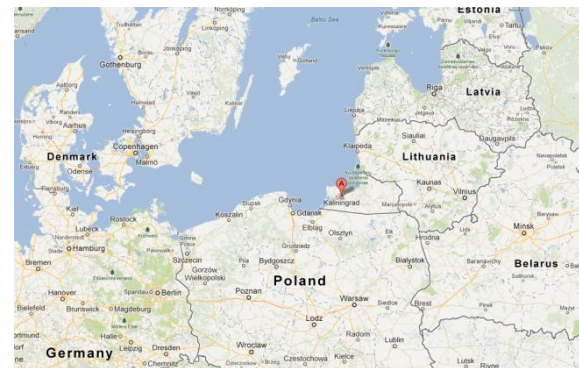
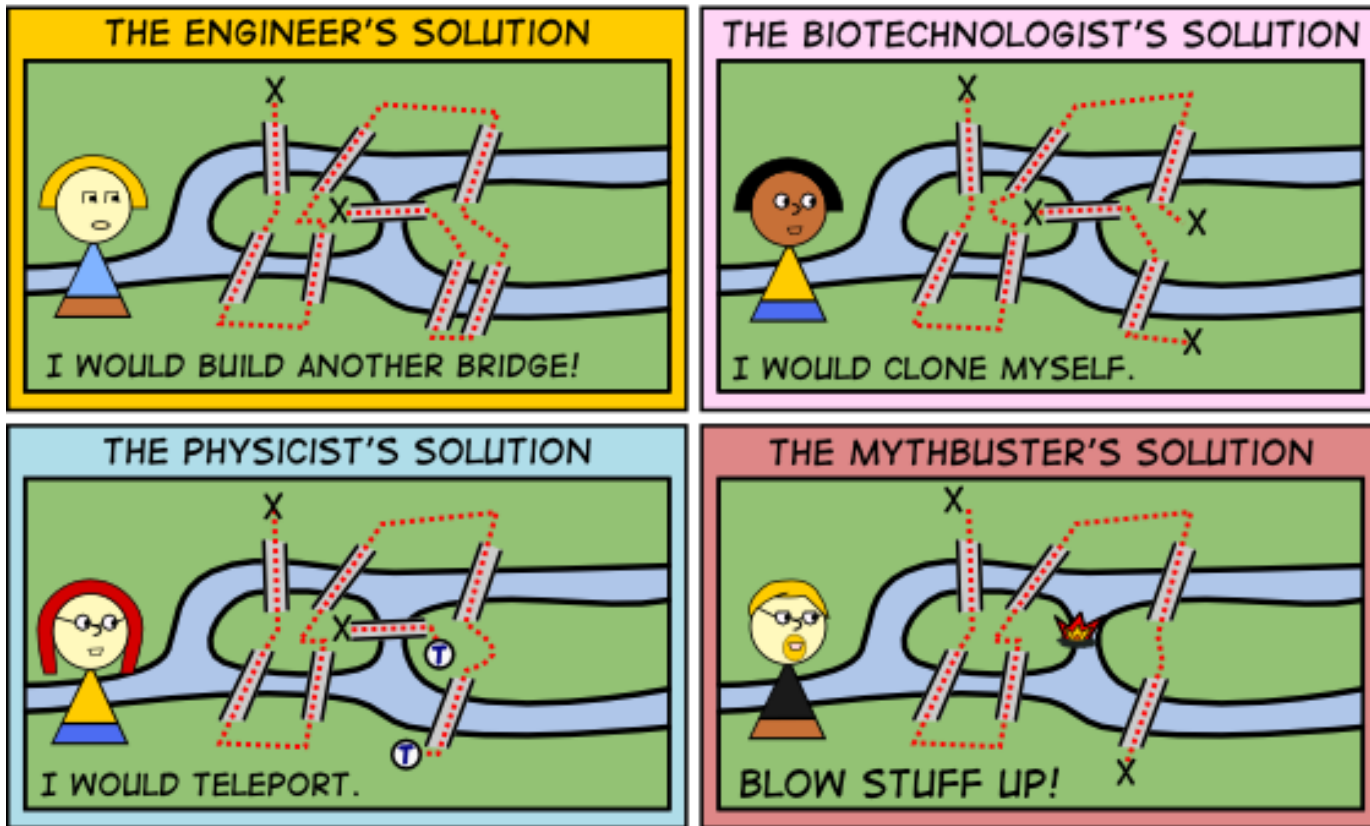


FIGURE 98. *Geographic Map: The Königsberg Bridges.*



Today: Kaliningrad, Russia

# Unless...

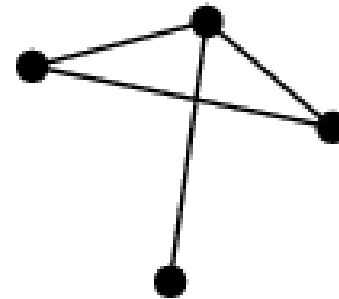


<http://spikedmath.com/541.html>

# Types of graphs: edge cardinality

- ▶ **Simple graph:**

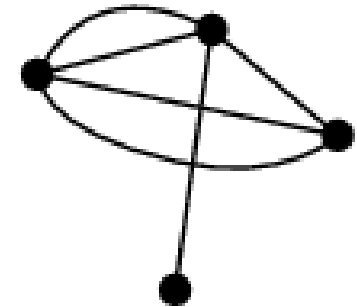
- ▶ At most one edge (i.e., either one edge or no edges) may connect any two vertices



*simple graph*

- ▶ **Multigraph:**

- ▶ Multiple edges are allowed between vertices



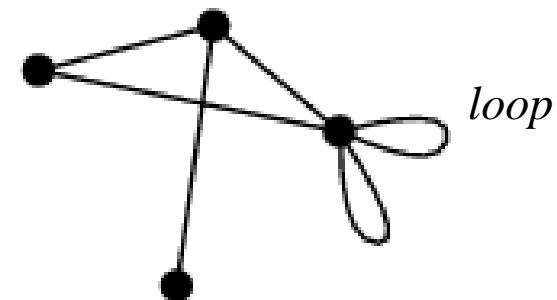
*multigraph*

- ▶ **Loops:**

- ▶ Edge between a vertex and itself

- ▶ **Pseudograph:**

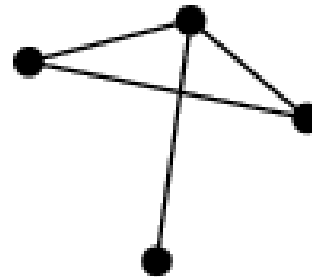
- ▶ Multigraph with loops



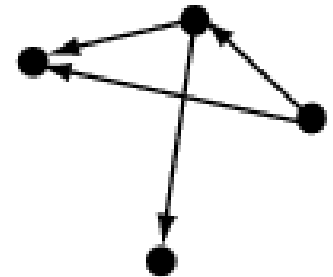
*pseudograph*

# Types of graphs: edge direction

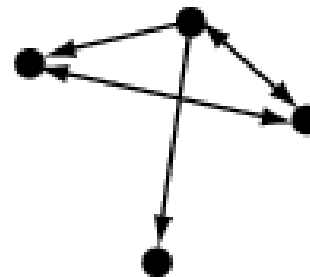
- ▶ Undirected
- ▶ Oriented
  - ▶ Edges have **one** direction (indicated by arrow)
- ▶ Directed
  - ▶ Edges may have **one or two** directions
- ▶ Network
  - ▶ Oriented graph with weighted edges



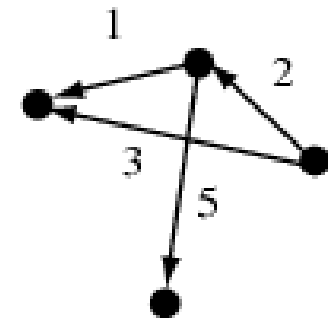
*undirected graph*



*oriented graph*



*directed graph*



*network*

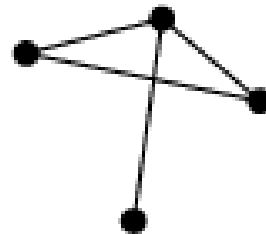
# Types of graphs: labeling

## ▶ Labels

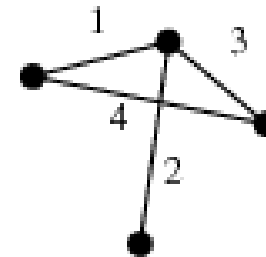
- ▶ None
- ▶ On Vertices
- ▶ On Edges

## ▶ Groups (=colors)

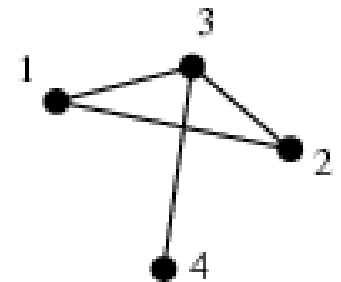
- ▶ Of Vertices
  - ▶ no edge connects two identically colored vertices
- ▶ Of Edges
  - ▶ adjacent edges must receive different colors
- ▶ Of both



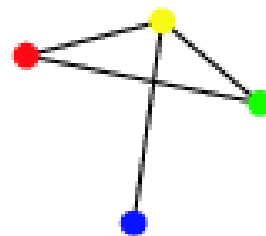
*unlabeled graph*



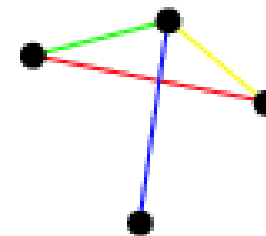
*edge-labeled graph*



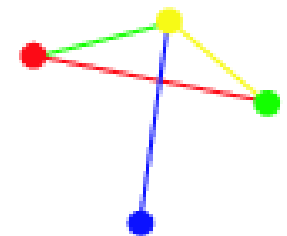
*vertex-labeled graph*



*vertex-colored graph*



*edge-colored graph*

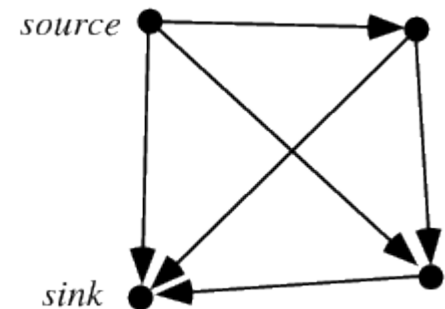


*vertex- and edge-colored graph*

# Directed and Oriented graphs

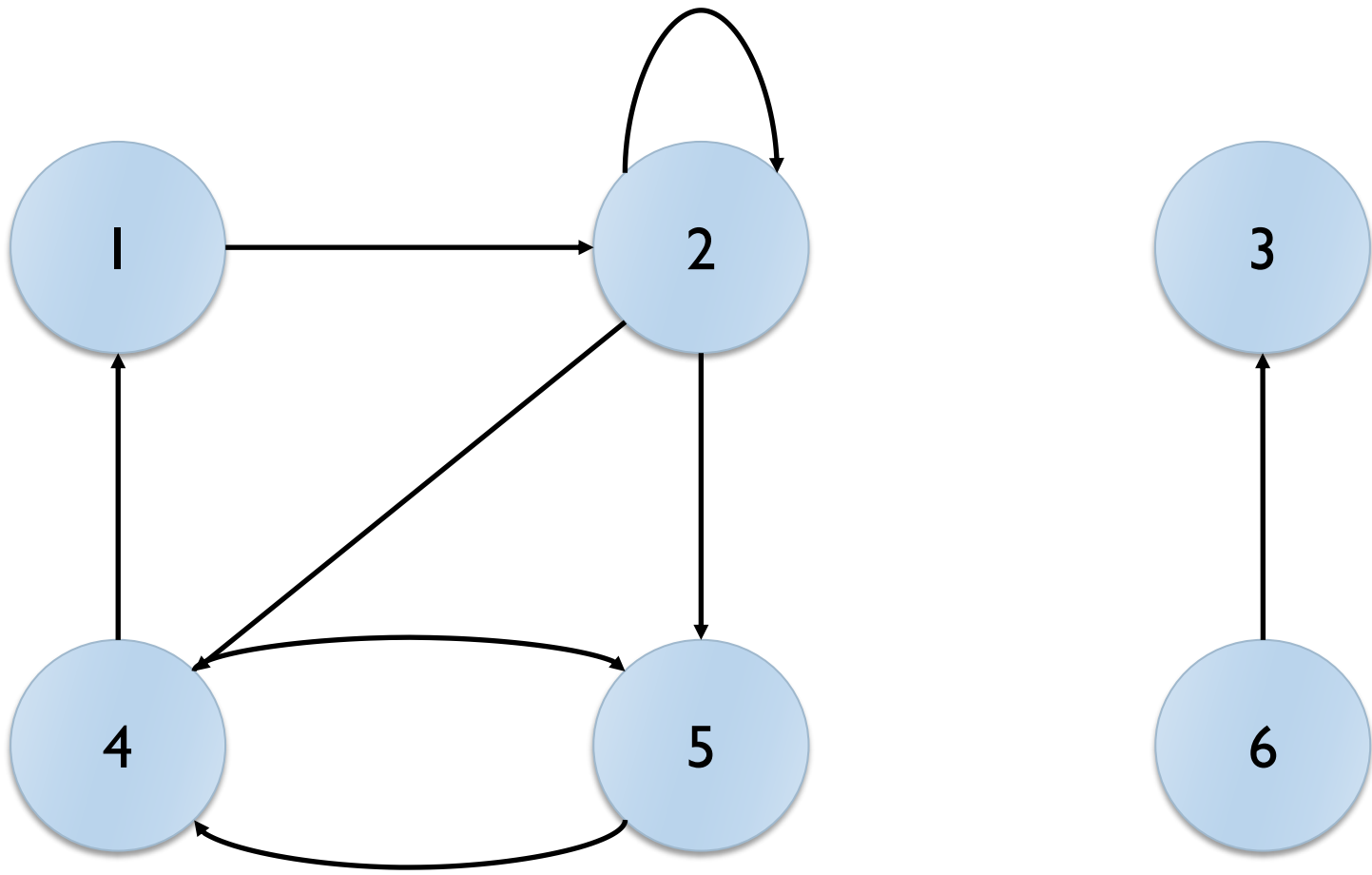
---

- ▶ A Directed Graph (*di-graph*)  $G$  is a pair  $(V,E)$ , where
  - ▶  $V$  is a (finite) set of *vertices*
  - ▶  $E$  is a (finite) set of *edges*, that identify a binary relationship over  $V$ 
    - ▶  $E \subseteq V \times V$



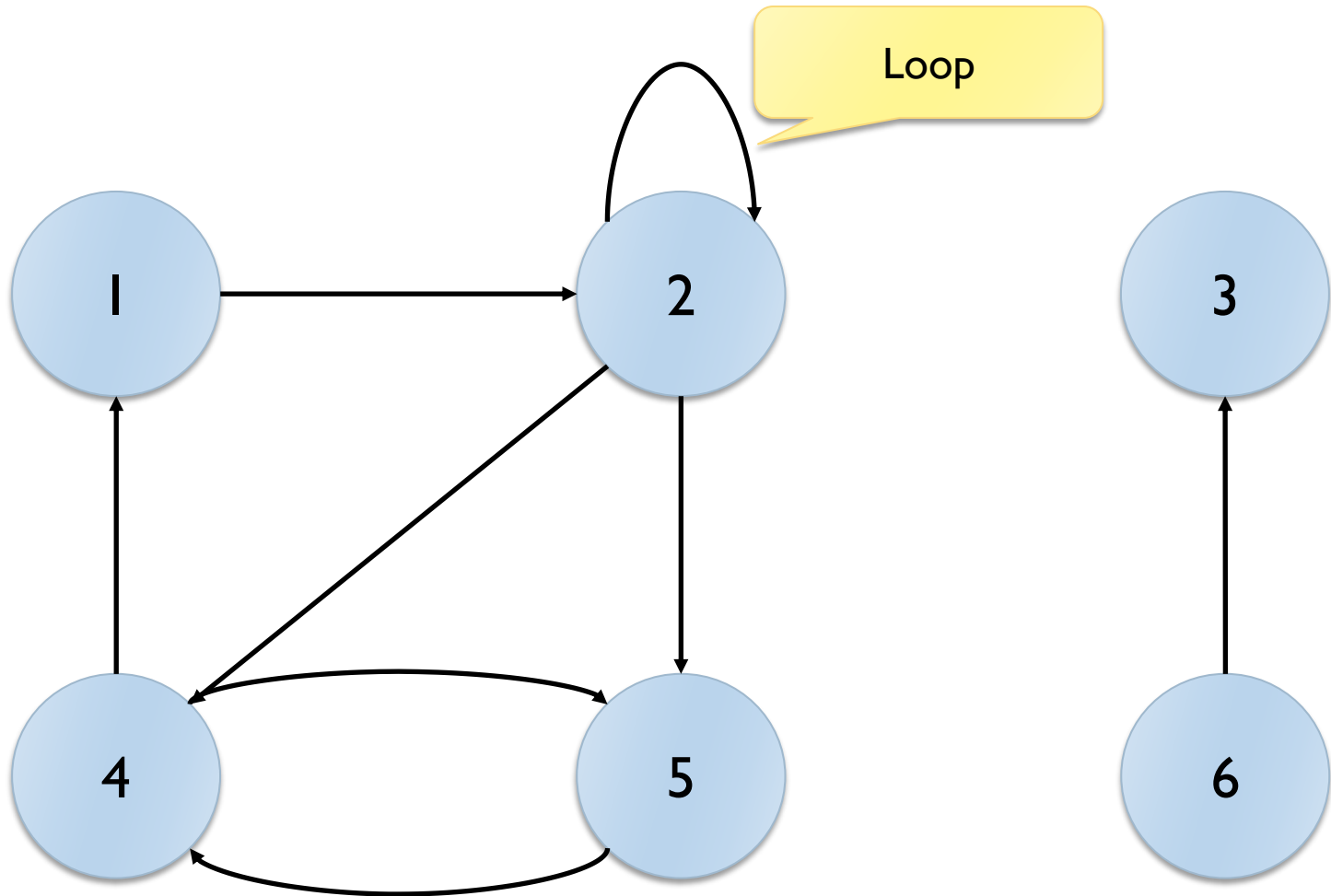
# Example

---



# Example

---

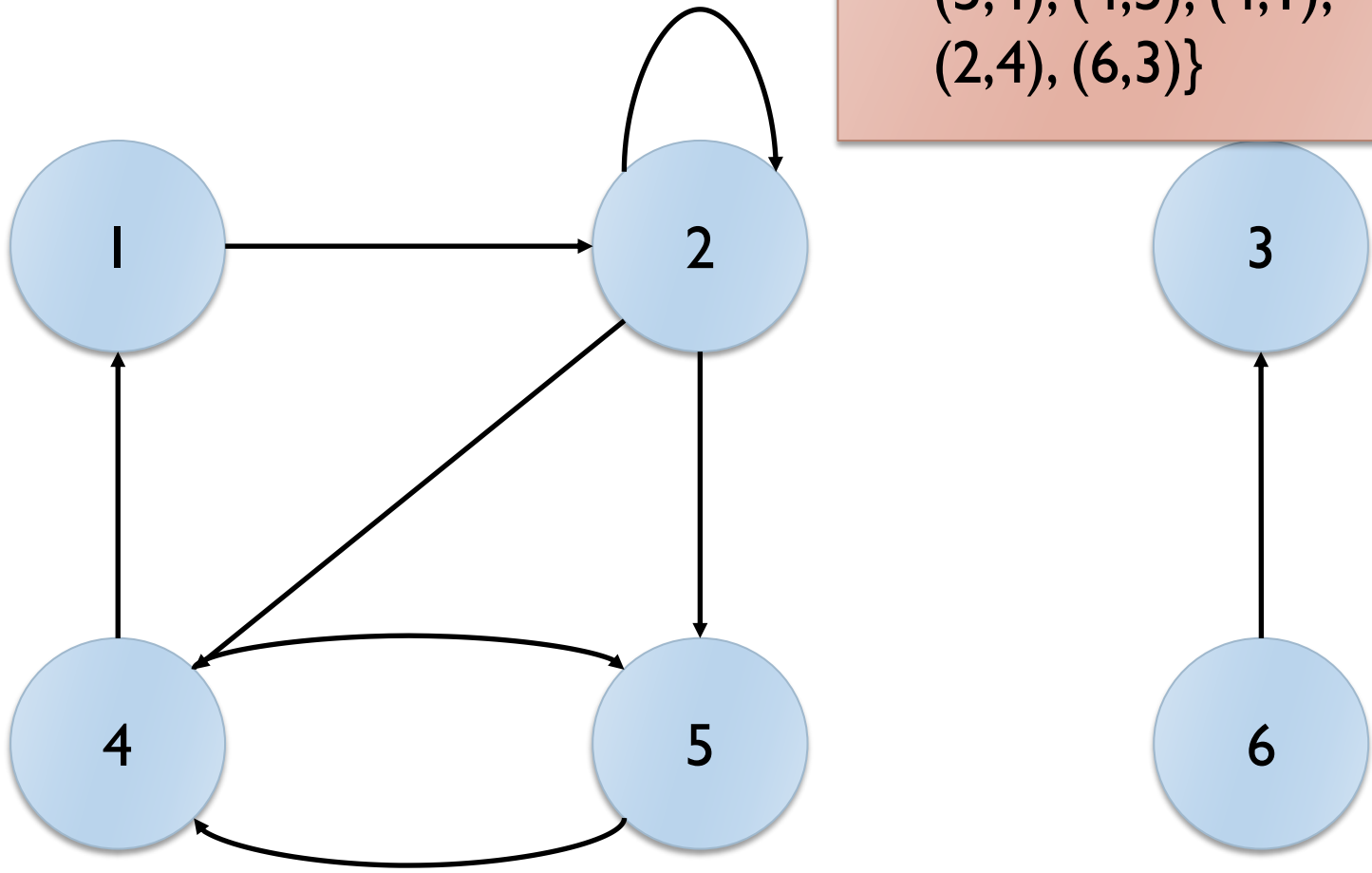




# Example

$$V = \{1, 2, 3, 4, 5, 6\}$$

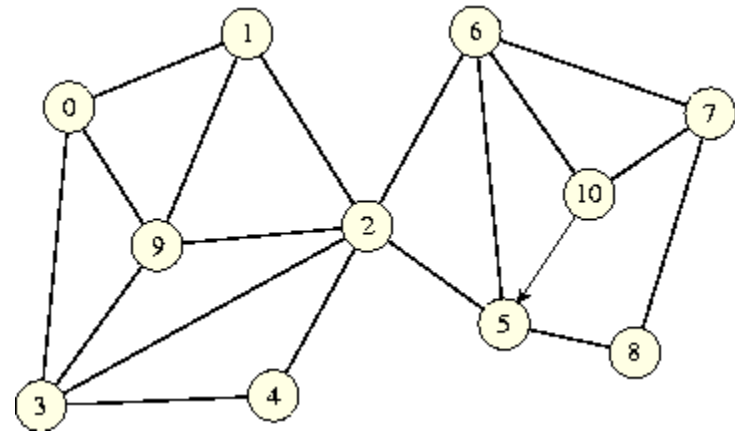
$$E = \{(1, 2), (2, 2), (2, 5), (5, 4), (4, 5), (4, 1), (2, 4), (6, 3)\}$$



# Undirected graph

---

- ▶ Ad **Undirected** Graph is still represented as a couple  $G=(V,E)$ , but the set  $E$  is made of **non-ordered pairs** of vertices

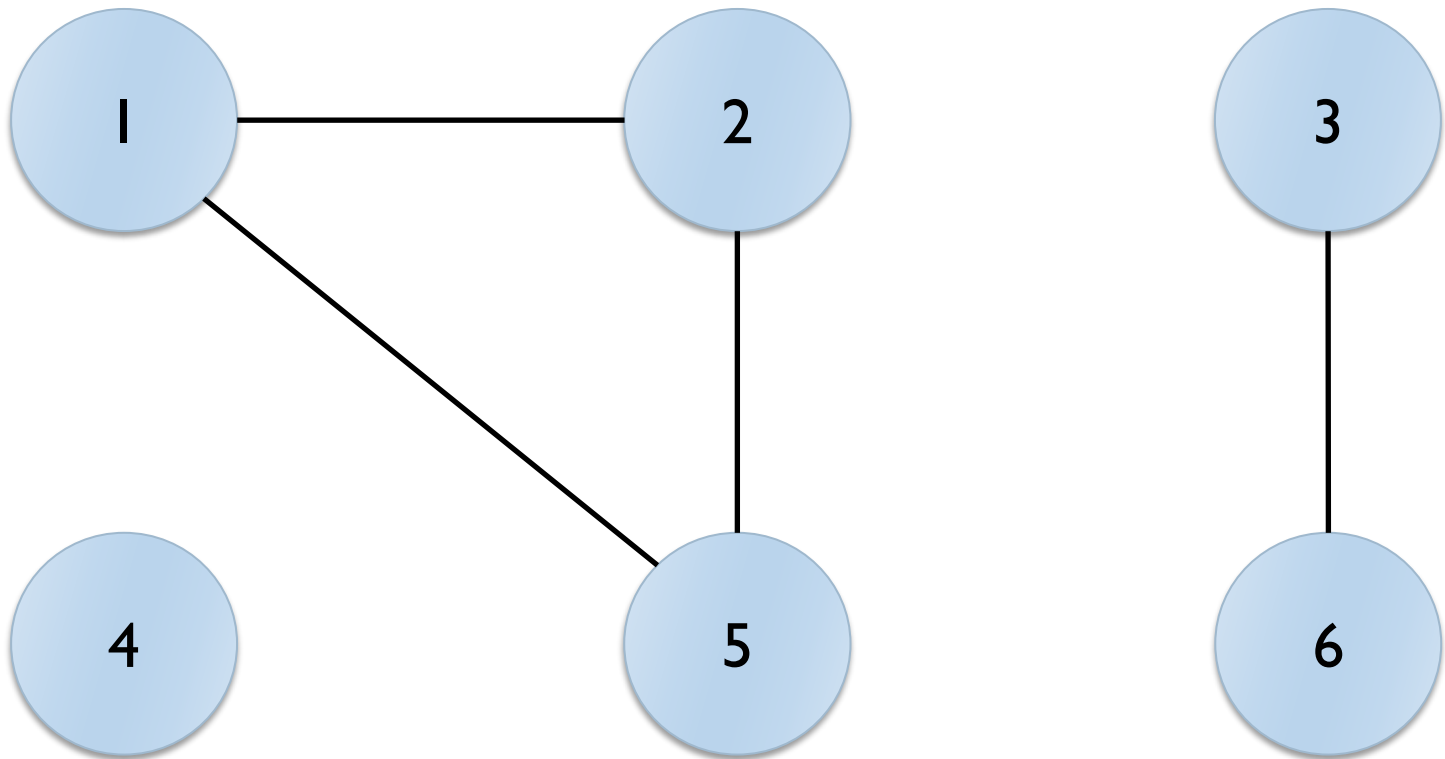


# Example

---

$V = \{1, 2, 3, 4, 5, 6\}$

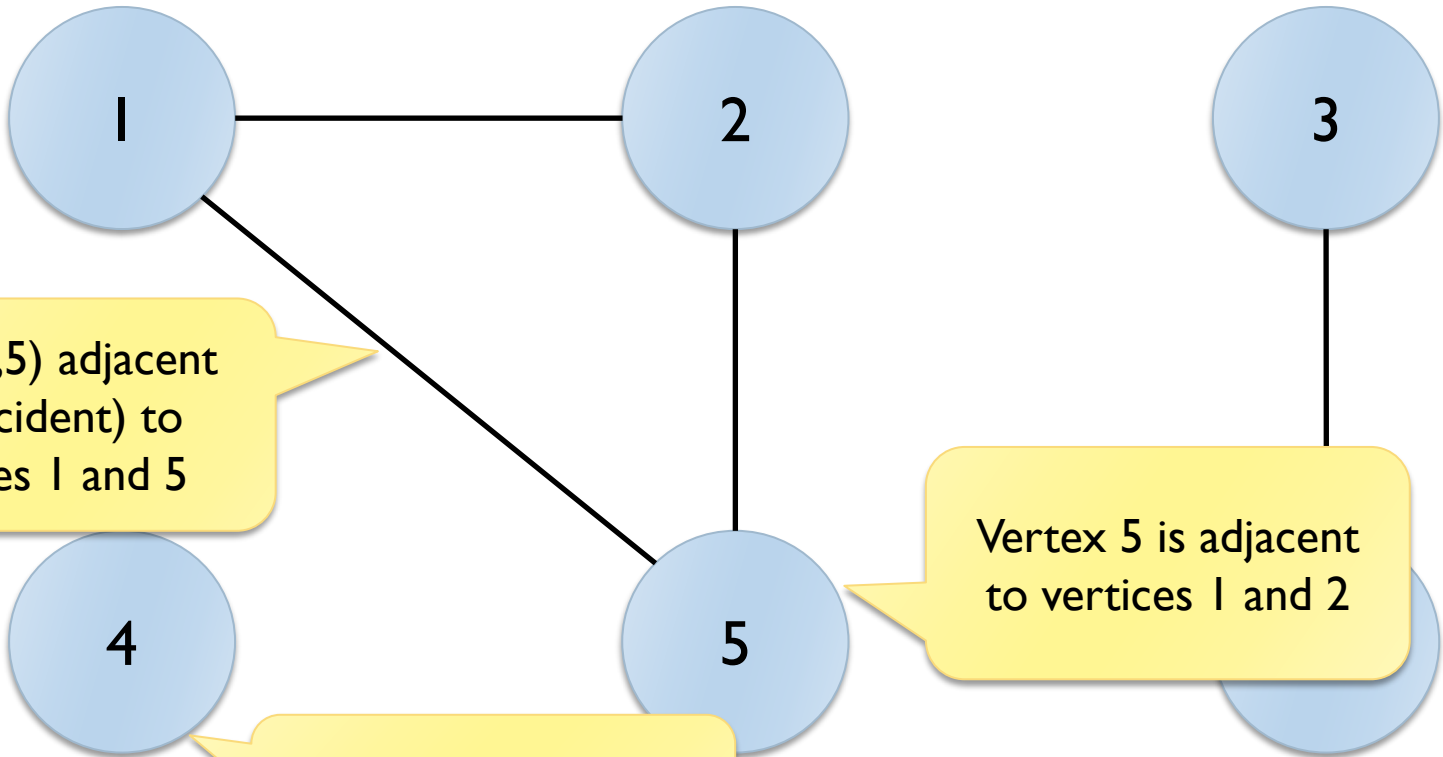
$E = \{\{1, 2\}, \{2, 5\}, \{5, 1\}, \{6, 3\}\}$



# Example

$V = \{1, 2, 3, 4, 5, 6\}$

$E = \{\{1, 2\}, \{2, 5\}, \{5, 1\}, \{6, 3\}\}$



Edge (1,5) adjacent  
(or incident) to  
vertices 1 and 5

Vertex 5 is adjacent  
to vertices 1 and 2

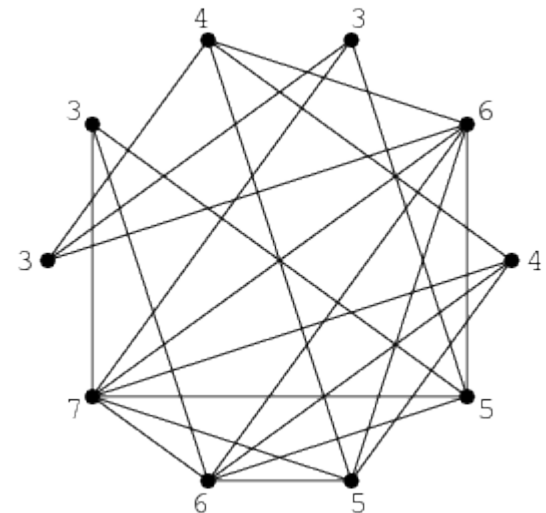
Vertex 4 is isolated



# Degree

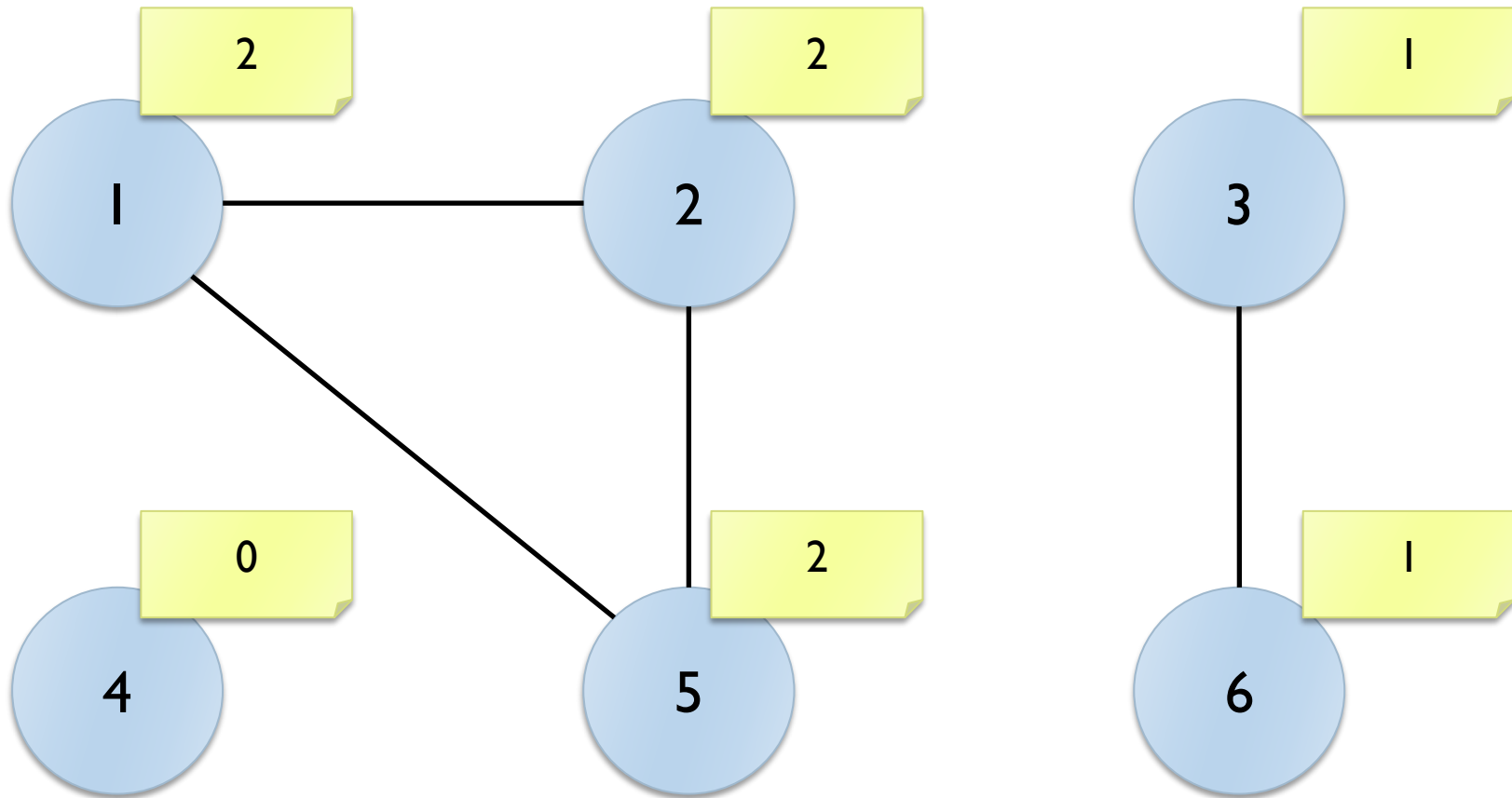
---

- ▶ In an *undirected* graph,
  - ▶ the **degree** of a vertex is the number of incident edges
- ▶ In a *directed* graph
  - ▶ The **in-degree** is the number of incoming edges
  - ▶ The **out-degree** is the number of departing edges
  - ▶ The **degree** is the sum of in-degree and out-degree
- ▶ A vertex with degree 0 is **isolated**

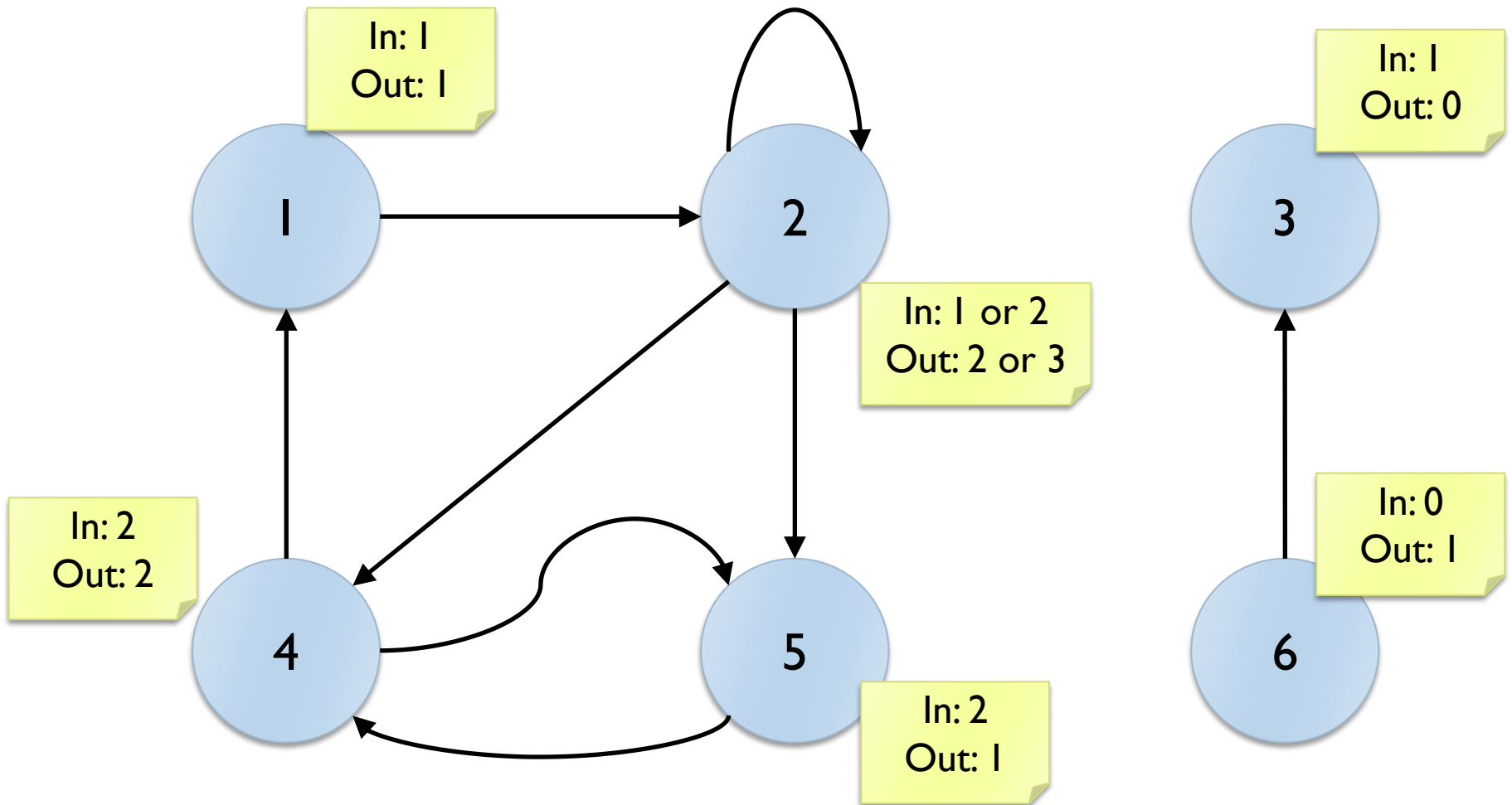


# Degree

---



# Degree





# Paths

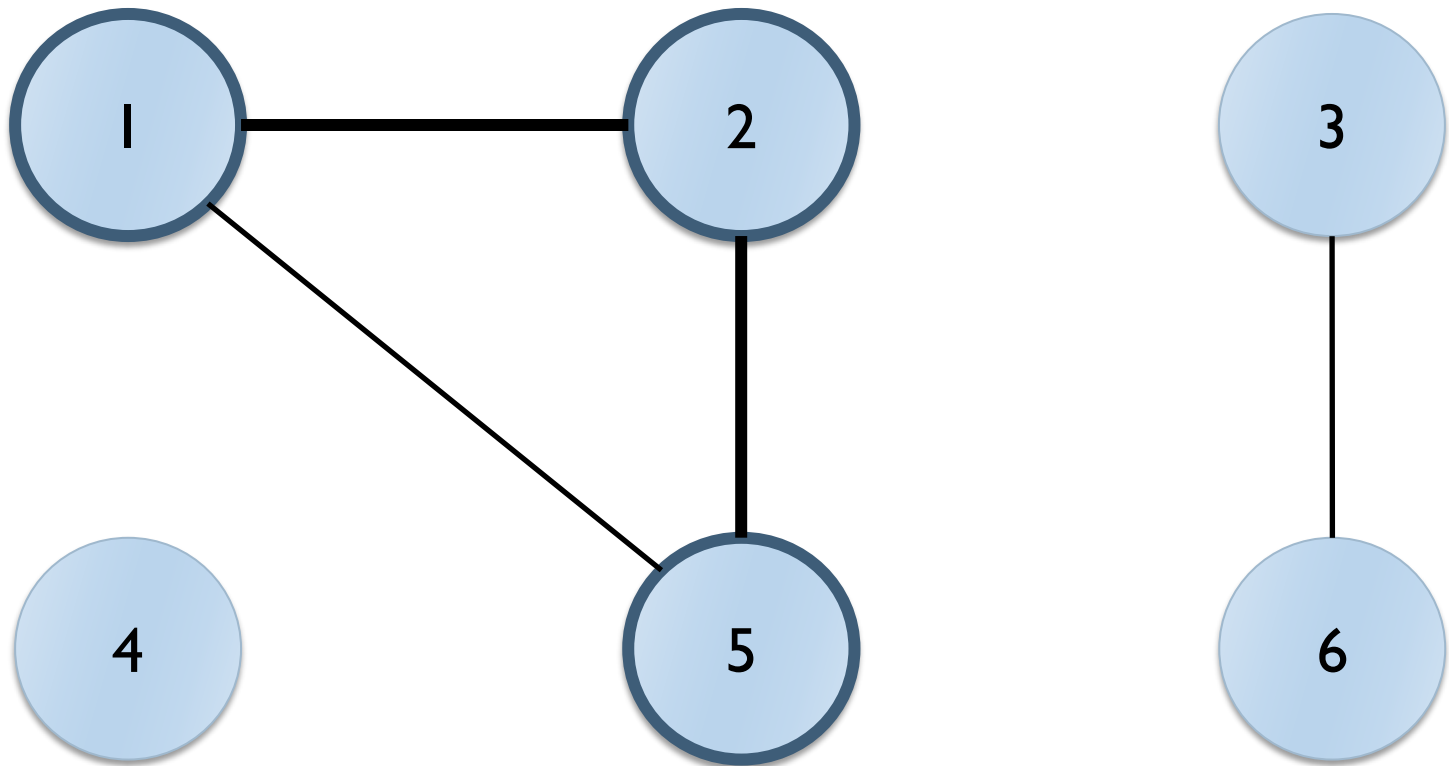
---

- ▶ A **path** on a graph  $G=(V,E)$  also called a trail, is a sequence  $\{v_1, v_2, \dots, v_n\}$  such that:
  - ▶  $v_1, \dots, v_n$  are vertices:  $v_i \in V$
  - ▶  $(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n)$  are graph edges:  $(v_{i-1}, v_i) \in E$
  - ▶  $v_i$  are distinct (for “simple” paths).
- ▶ The **length** of a path is the number of edges  $(n-1)$
- ▶ If there exist a path between  $v_A$  and  $v_B$  we say that  $v_B$  is **reachable** from  $v_A$

# Example

---

Path = { 1, 2, 5 }  
Length = 2



# Cycles

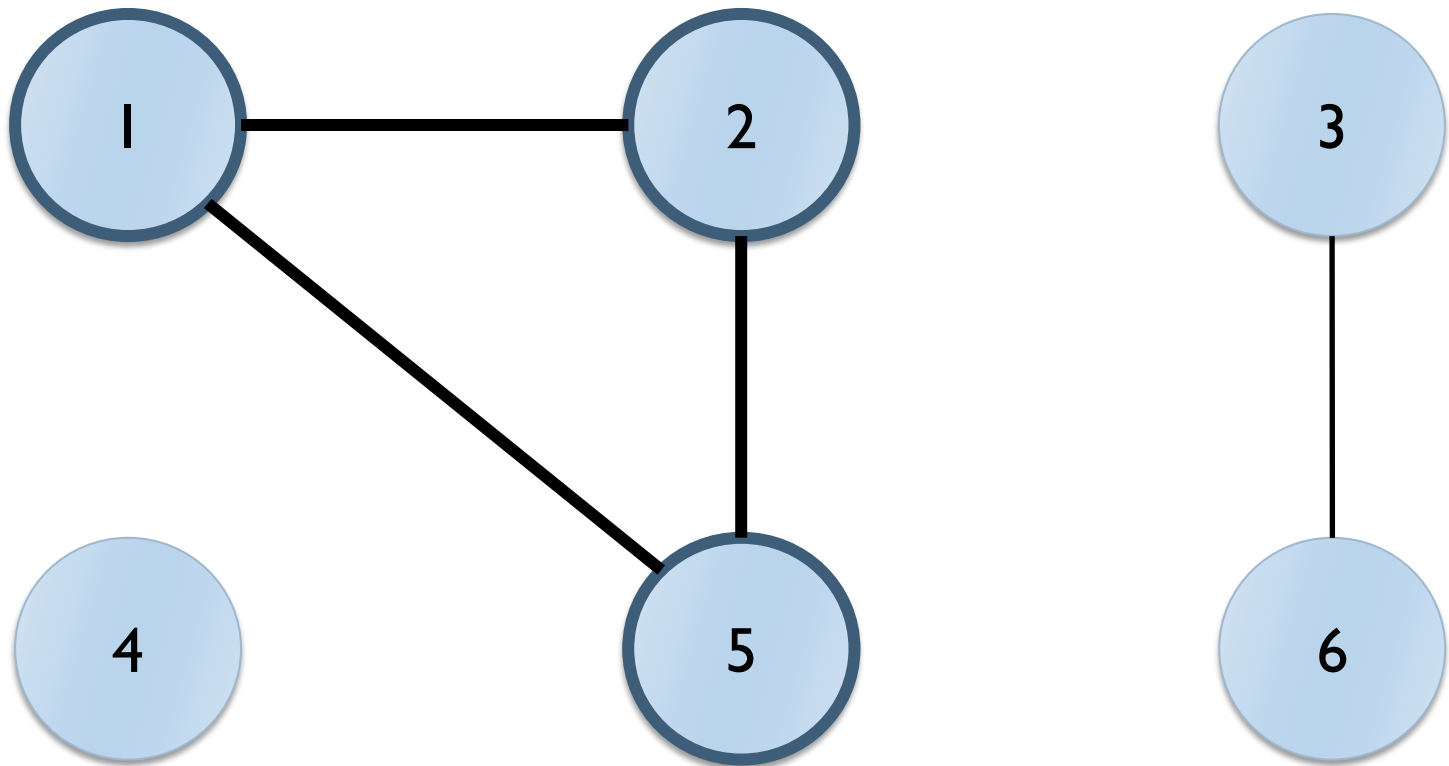
---

- ▶ A cycle is a path where  $v_1 = v_n$
- ▶ A graph with no cycles is said acyclic

# Example

---

Path = { 1, 2, 5, 1 }  
Length = 3



# Reachability (Undirected)

---

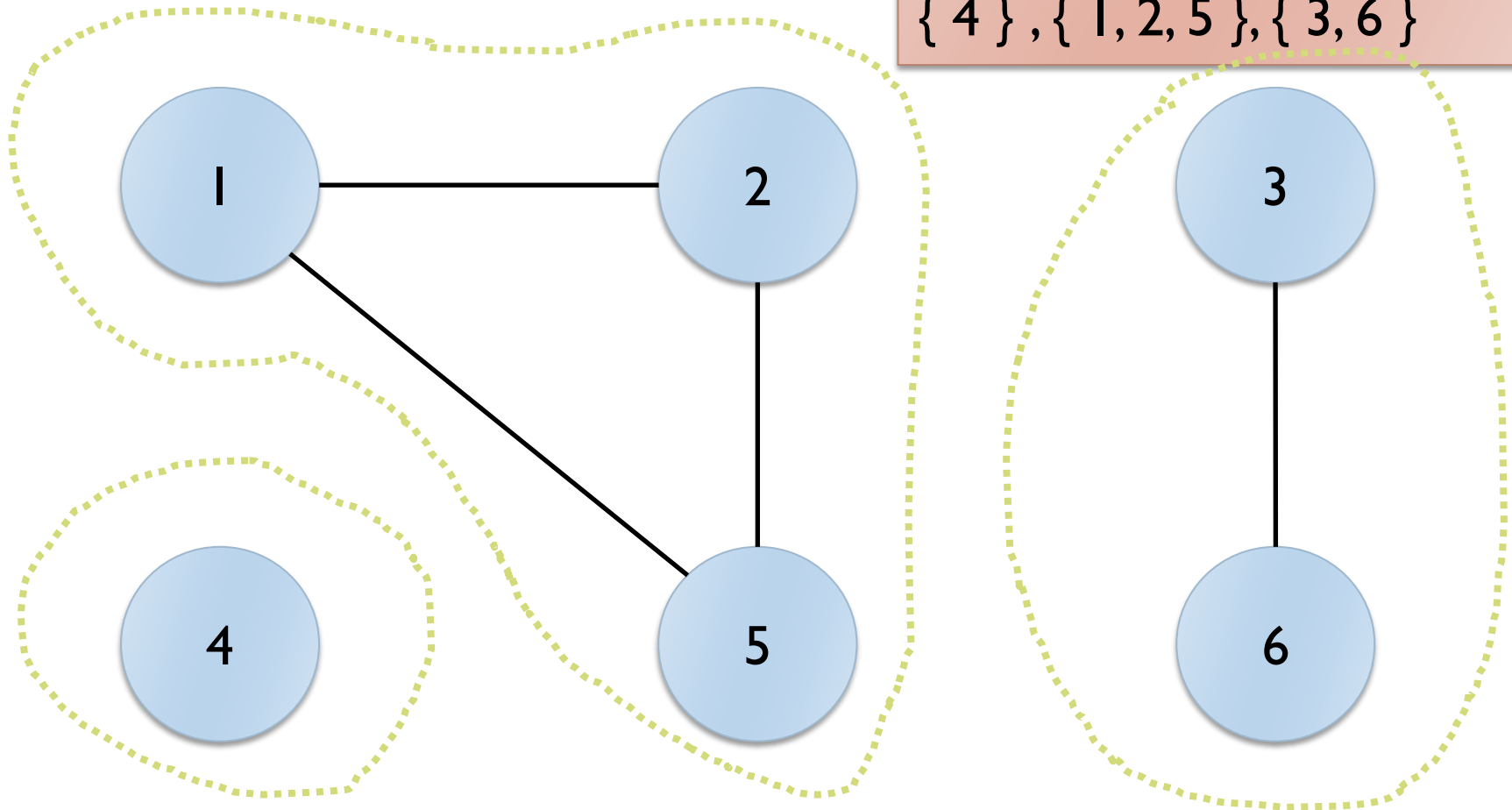
- ▶ An undirected graph is **connected** if, for every couple of vertices, there is a path connecting them
- ▶ The connected sub-graph of maximum size are called **connected components**
- ▶ A connected graph has exactly one connected component

# Connected components

The graph is **not** connected.

Connected components =  
3

$\{ 4 \}, \{ 1, 2, 5 \}, \{ 3, 6 \}$



# Reachability (Directed)

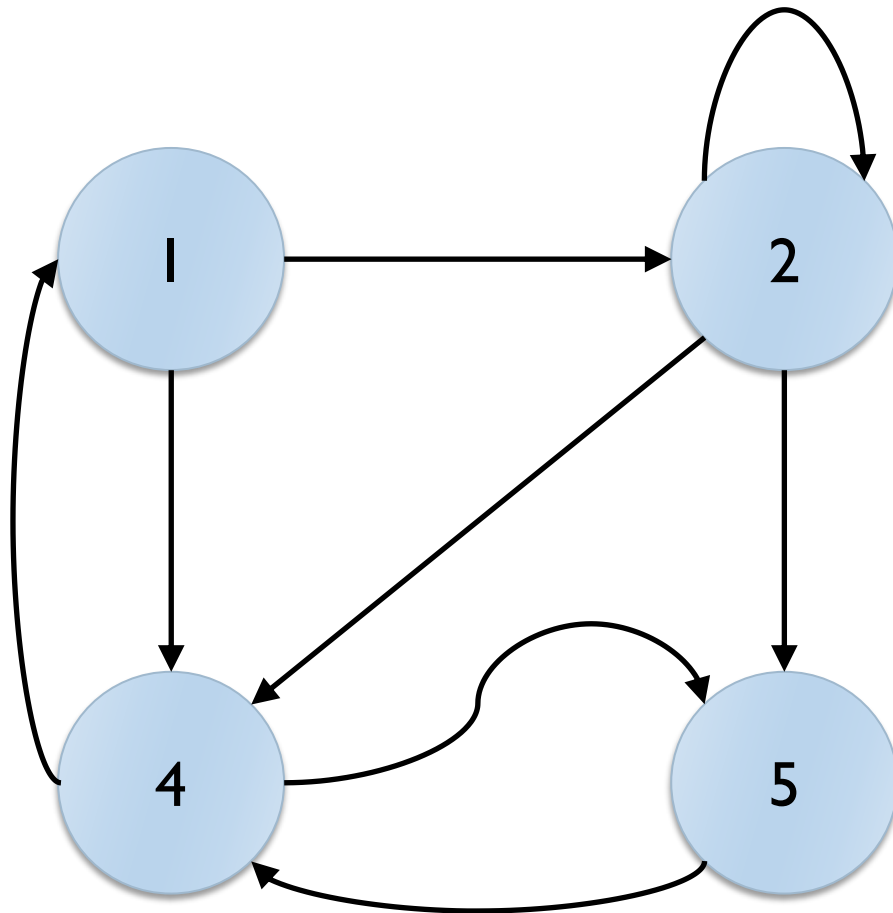
---

- ▶ A directed graph is **strongly connected** if, for every ordered pair of vertices  $(v, v')$ , there exists at least one path connecting  $v$  to  $v'$

# Example

---

The graph is **strongly connected**

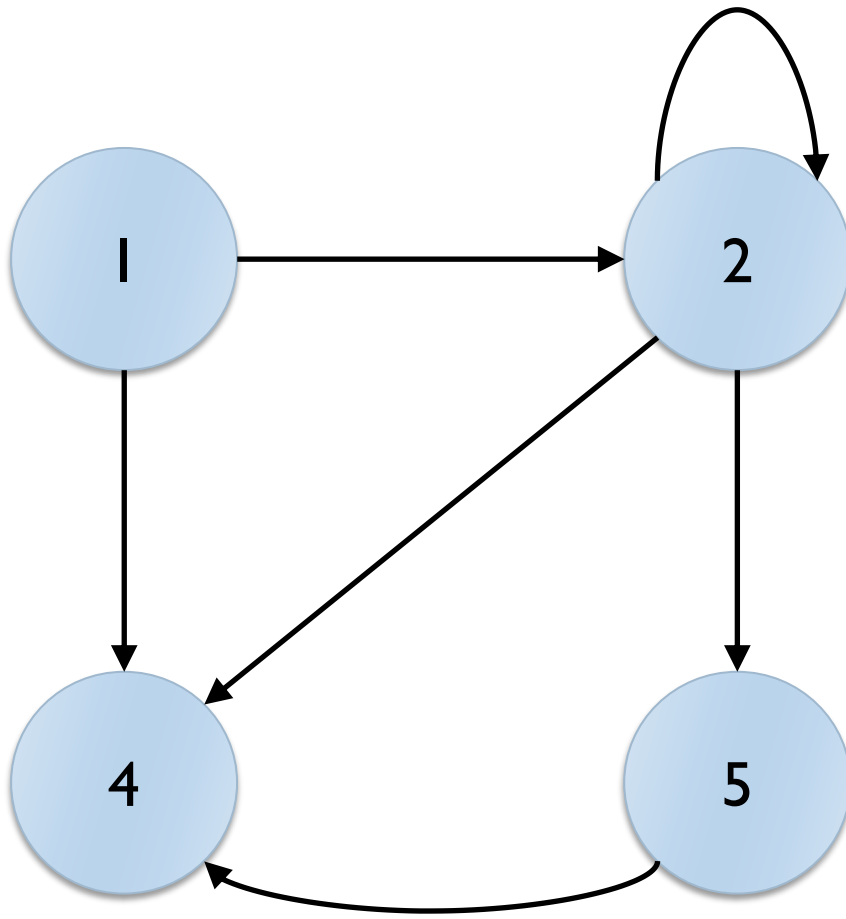




# Example

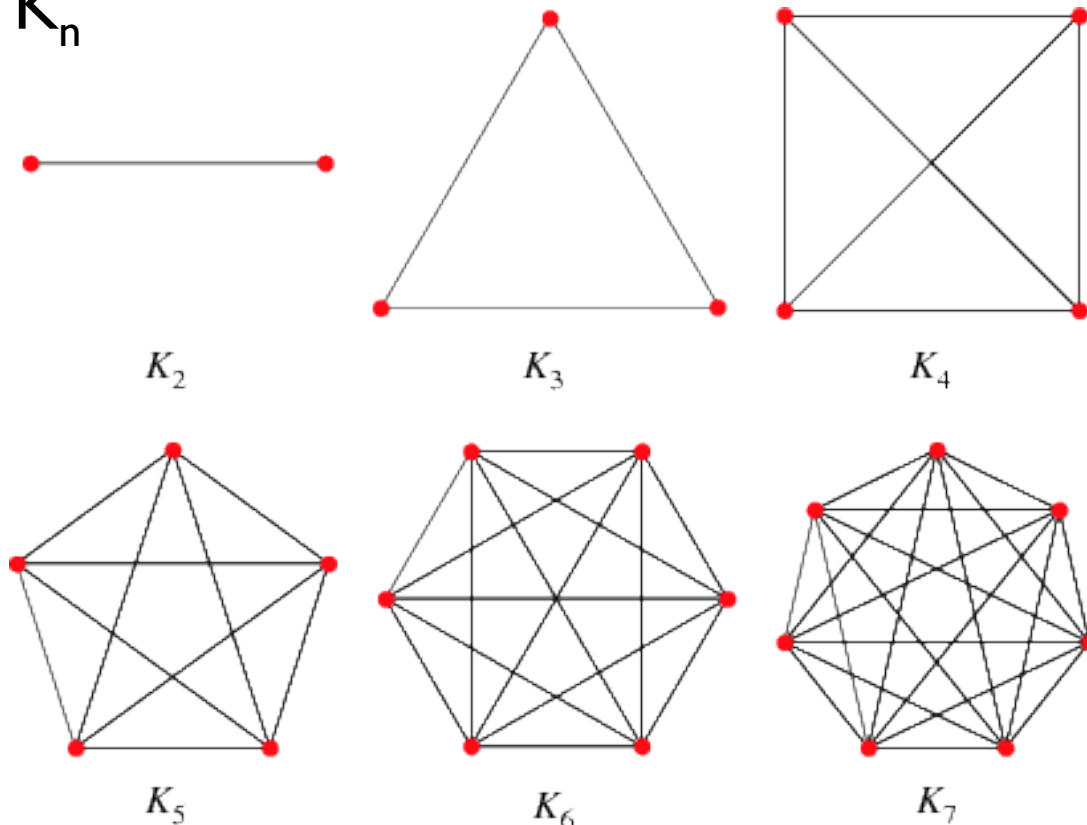
---

The graph is **not** strongly connected



# Complete graph

- ▶ A graph is complete if, for every pair of vertices, there is an edge connecting them (they are adjacent)
- ▶ Symbol:  $K_n$



# Complete graph: edges

---

- ▶ In a **complete** graph with  $n$  vertices, the number of **edges** is
  - ▶  $n(n-1)$ , if the graph is directed
  - ▶  $n(n-1)/2$ , if the graph is undirected
  - ▶ If self-loops are allowed, then
    - ▶  $n^2$  for directed graphs
    - ▶  $n(n-1)$  for undirected graphs

# Density

---

- ▶ The density of a graph  $G=(V,E)$  is the ratio of the number of edges to the total number of possible edges

$$d = \frac{|E(G)|}{|E(K_{|V(G)|})|}$$

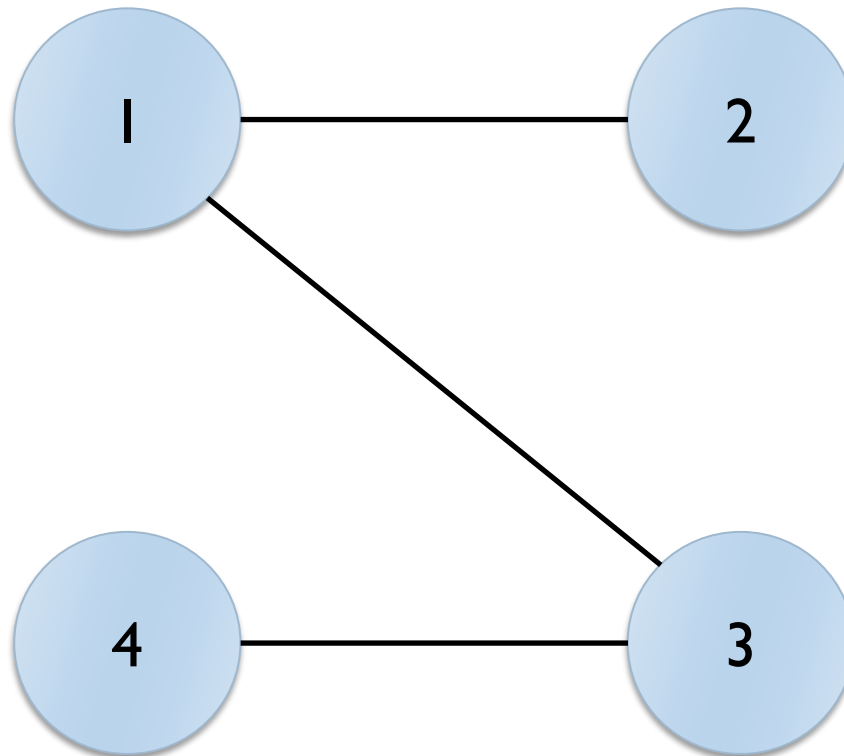
# Esempio

---

Density = 0.5

Existing: 3 edges

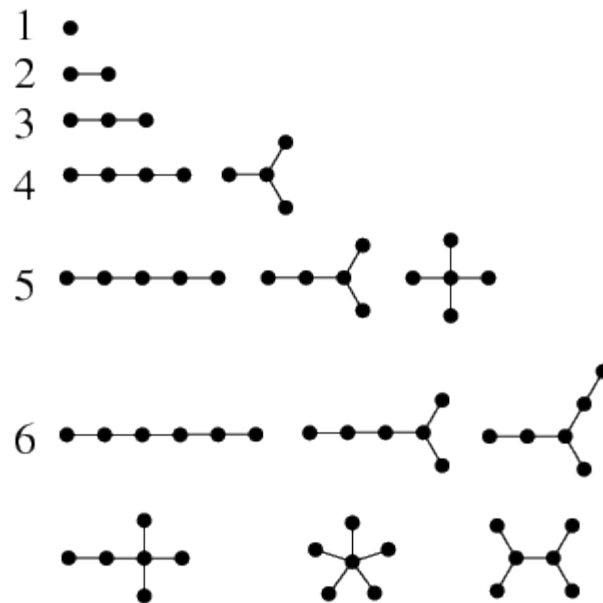
Total: 6 possible edges



# Trees and Forests

---

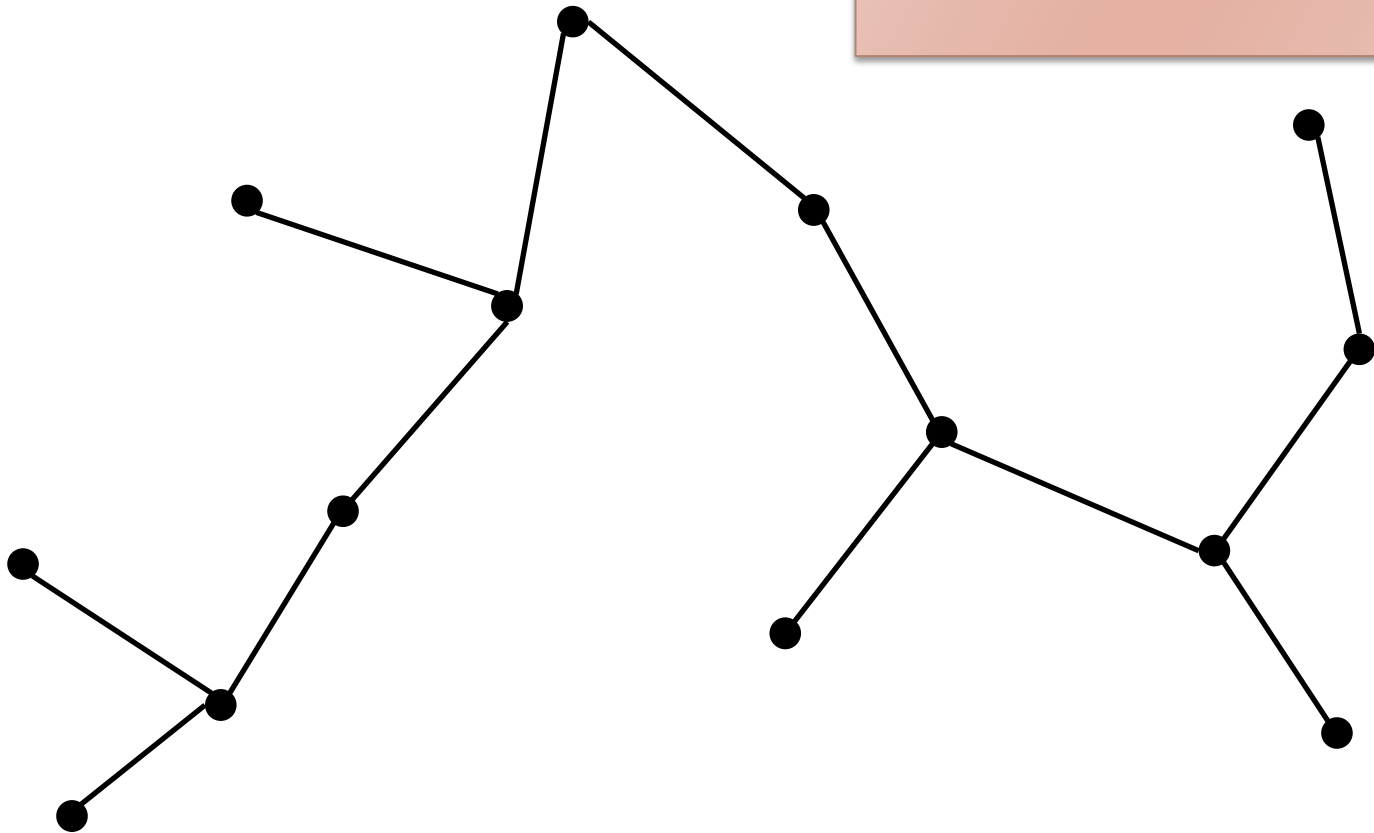
- ▶ An undirected acyclic graph is called **forest**
- ▶ An undirected acyclic connected graph is called **tree**



# Example

---

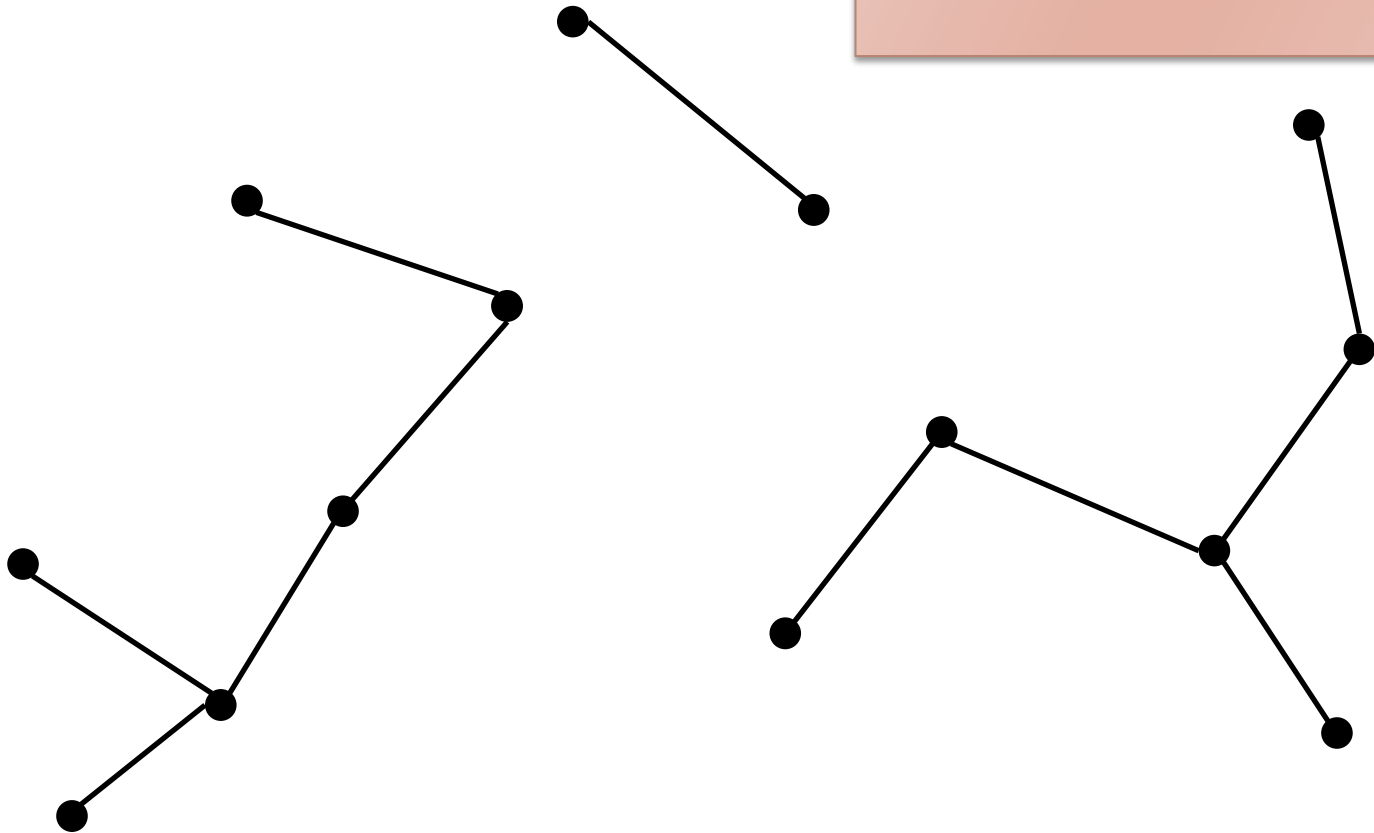
Tree



# Example

---

Forest

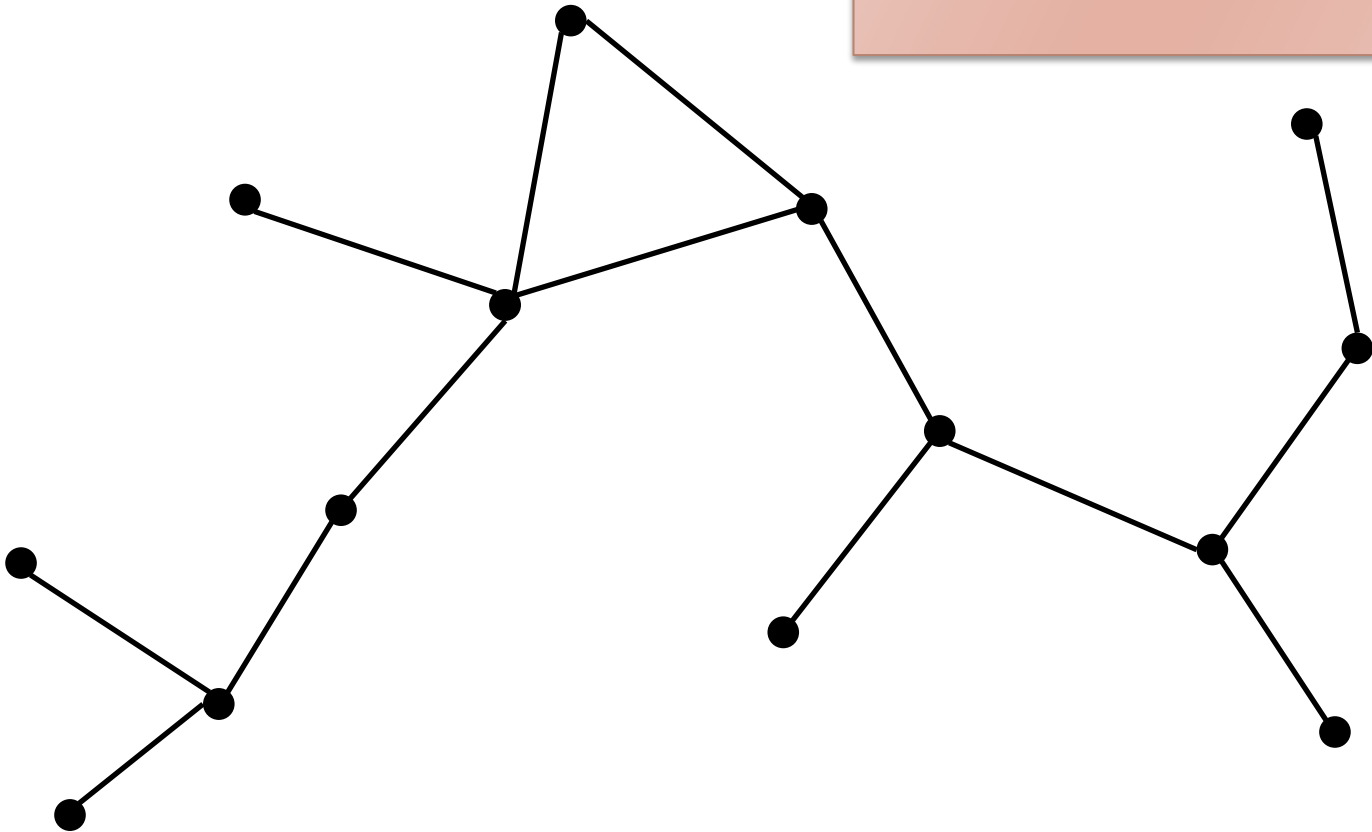




# Example

---

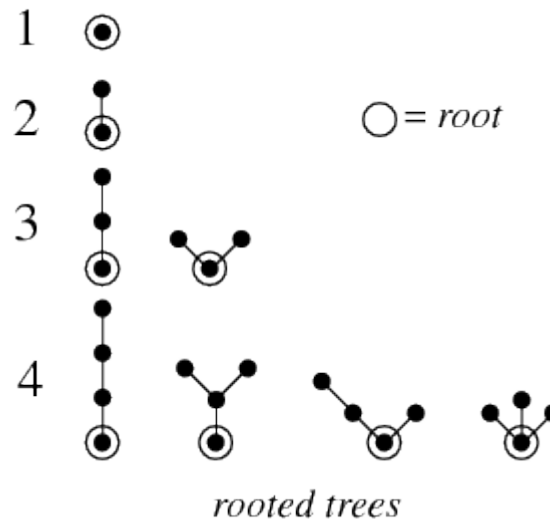
This is not a tree nor a forest  
(it contains a cycle)



# Rooted trees

---

- ▶ In a tree, a special node may be singled out
- ▶ This node is called the “**root**” of the tree
- ▶ Any node of a tree can be the root



# Tree (implicit) ordering

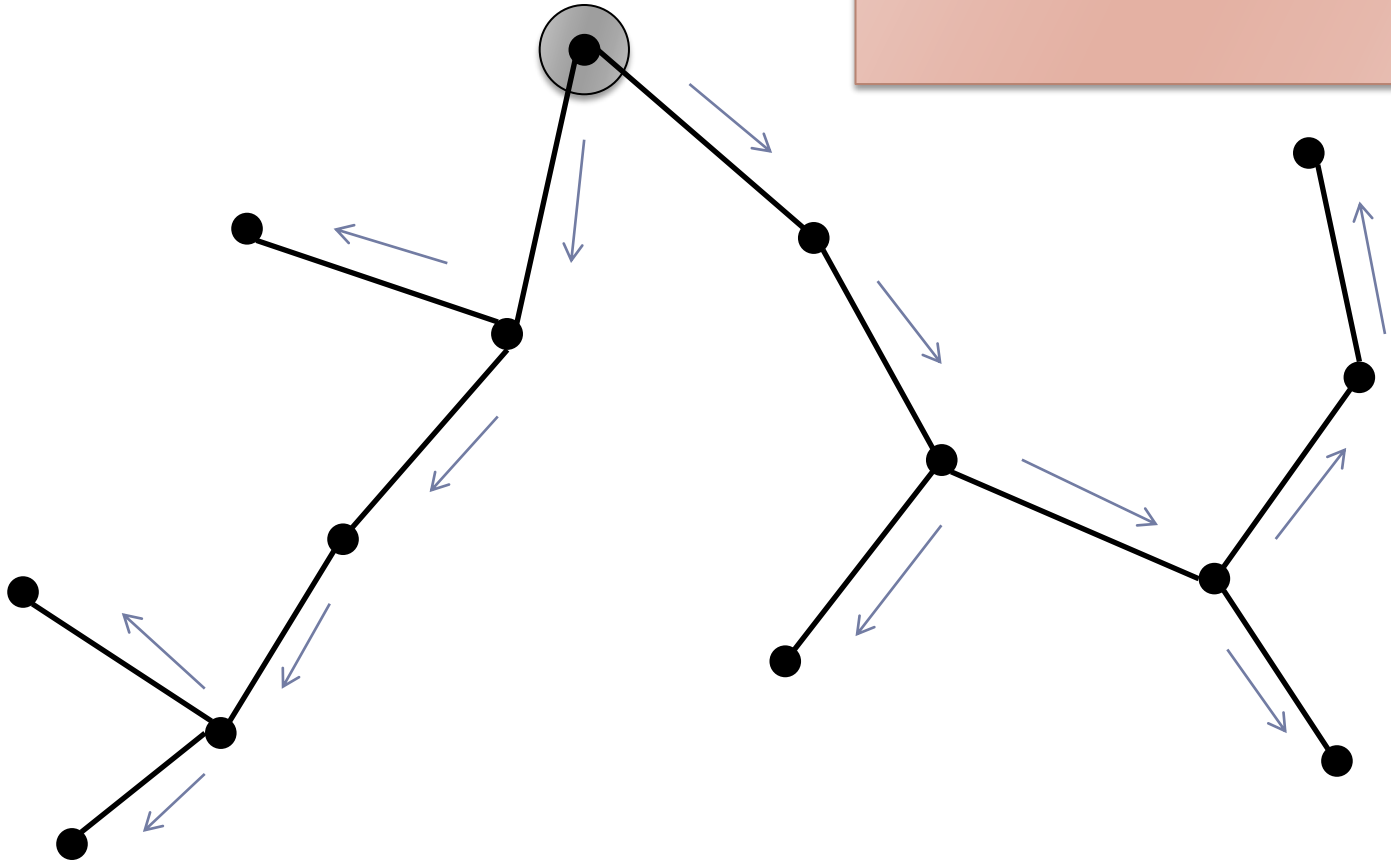
---

- ▶ The root node of a tree **induces an ordering** of the nodes
- ▶ The root is the “ancestor” of all other nodes/vertices
  - ▶ “children” are “away from the root”
  - ▶ “parents” are “towards the root”
- ▶ The root is the **only** node without parents
- ▶ All other nodes have exactly one parent
- ▶ The furthestmost (children-of-children-of-children...) nodes are “leaves”

# Example

---

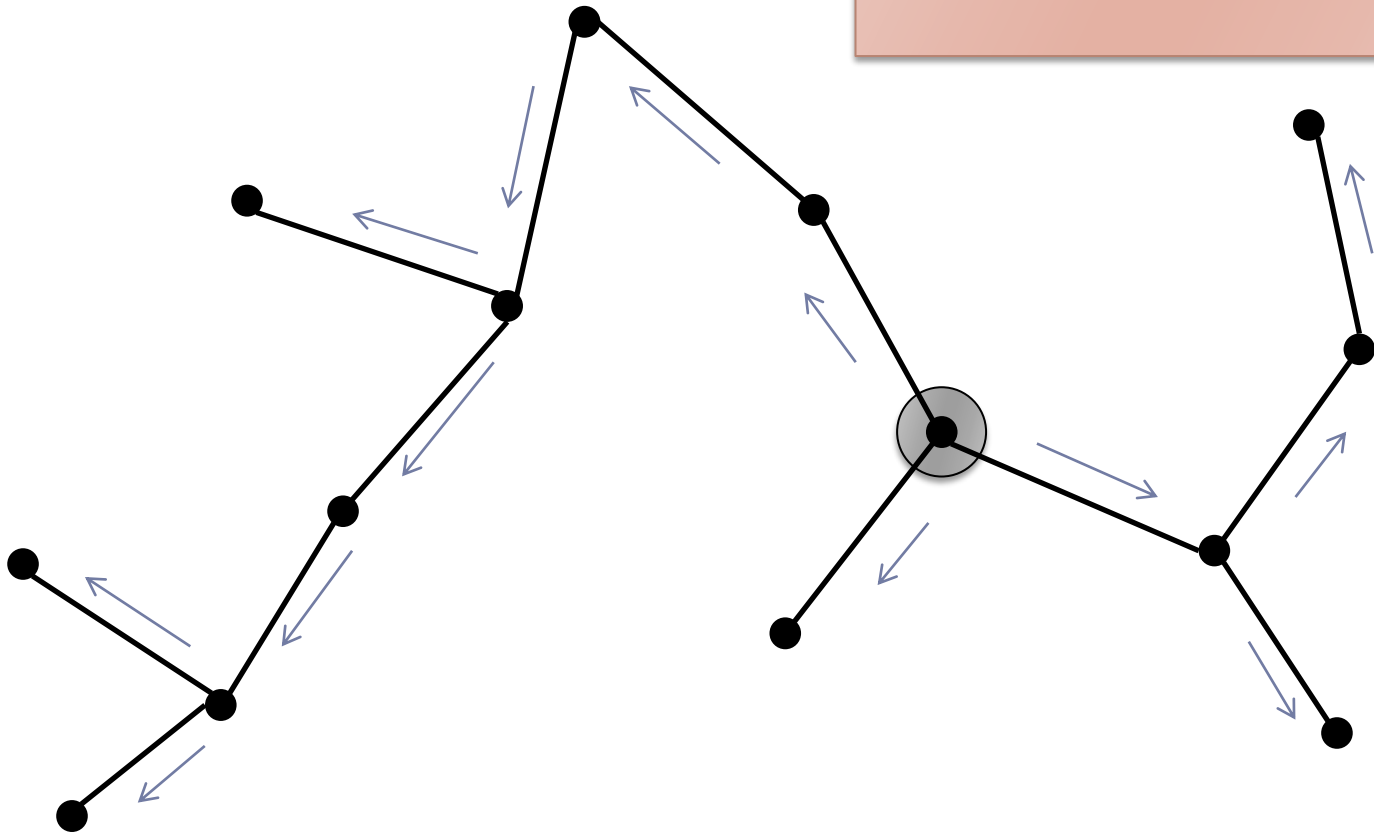
Rooted Tree



# Example

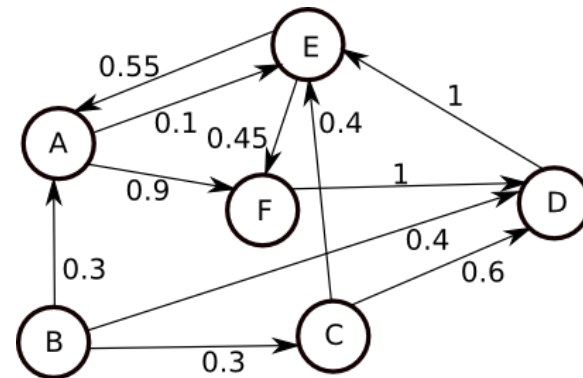
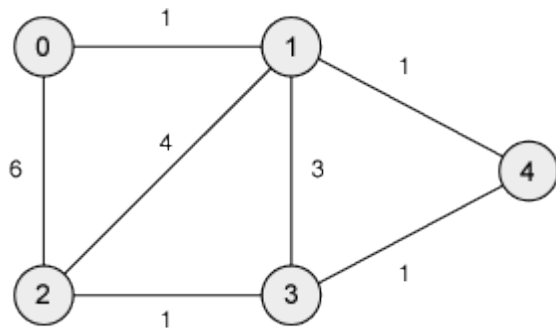
---

Rooted Tree



# Weighted graphs

- ▶ A weighted graph is a graph in which each branch (edge) is given a numerical weight.
- ▶ A weighted graph is therefore a special type of labeled graph in which the labels are numbers (which are usually taken to be positive).





# Graph applications

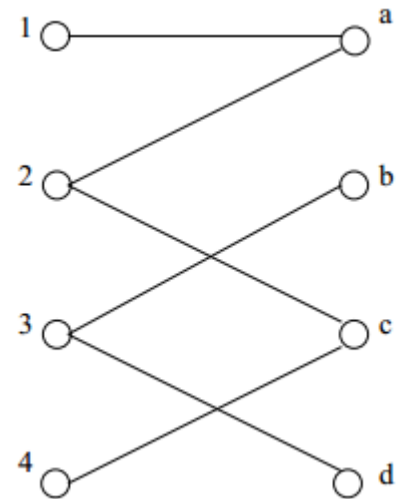
---

- ▶ **Graphs are everywhere**
  - ▶ Facebook friends (and posts, and 'likes')
  - ▶ Football tournaments (complete subgraphs + binary tree)
  - ▶ Google search index ( $V$ =page,  $E$ =link,  $w$ =pagerank)
  - ▶ Web analytics (site structure, visitor paths)
  - ▶ Car navigation (GPS)
  - ▶ Market Matching



# Market matching

- ▶  $H = \text{Houses } (1, 2, 3, 4)$
- ▶  $B = \text{Buyers } (a, b, c, d)$
- ▶  $V = H \cup B$
- ▶ Edges:  $(h, b) \in E$  if  $b$  would like to buy  $h$
- ▶ Problem: can all houses be sold and all buyers be satisfied?
- ▶ Variant: if the graph is weighted with a purchase offer, what is the most convenient solution?
- ▶ Variant: consider a 'penalty' for unsold items

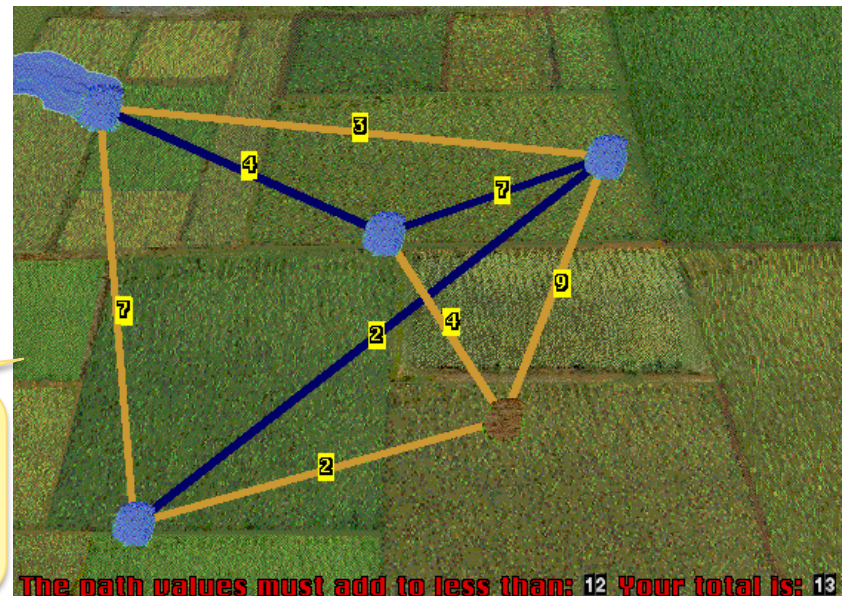


This graph is called  
“bipartite”:  
 $H \cap B = \emptyset$

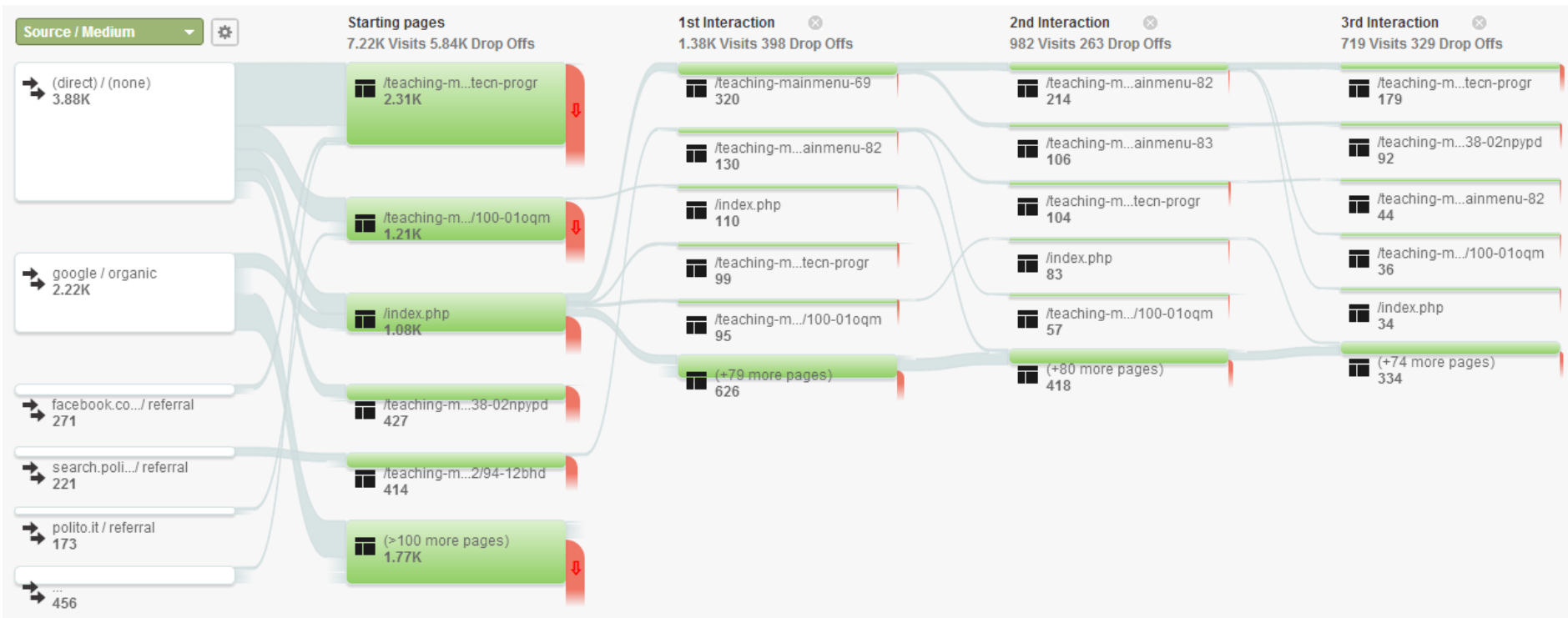
# Connecting cities

- ▶ We have a water reservoir
- ▶ We need to serve many cities
  - ▶ Directly or indirectly
- ▶ What is the most efficient set of inter-city water connections?
- ▶ Also for telephony, gas, electricity, ...

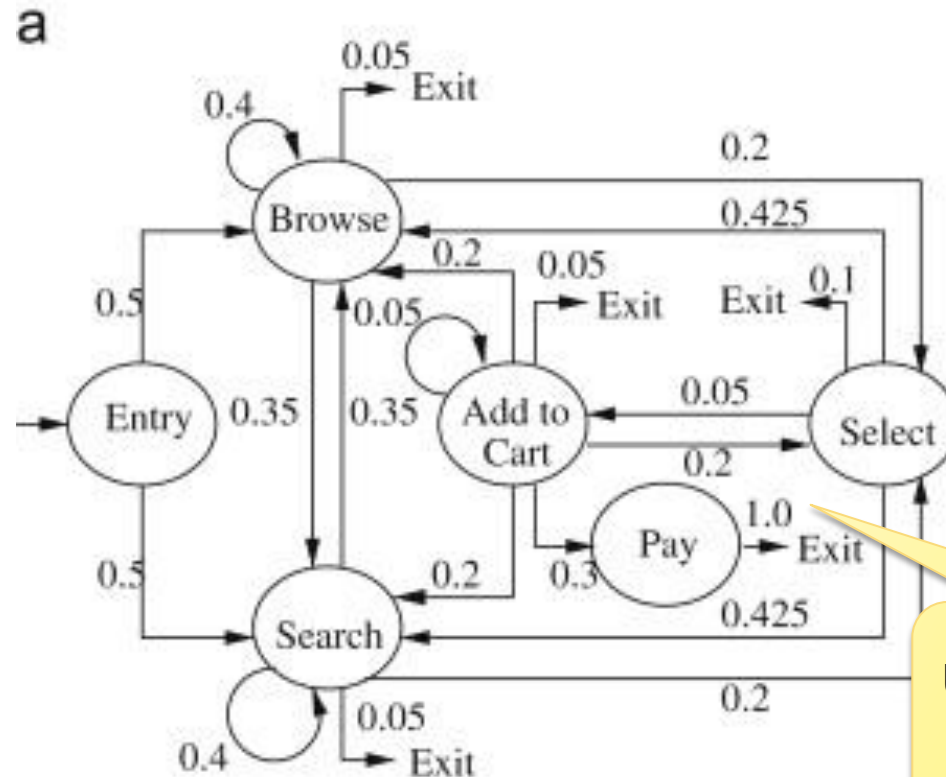
We are searching for the “minimum spanning tree”



# Google Analytics (Visitors Flow)



# Customer behavior



User actions encoded as frequencies

# Street navigation

---



TSP: The traveling salesman problem

We must find a “Hamiltonian cycle”

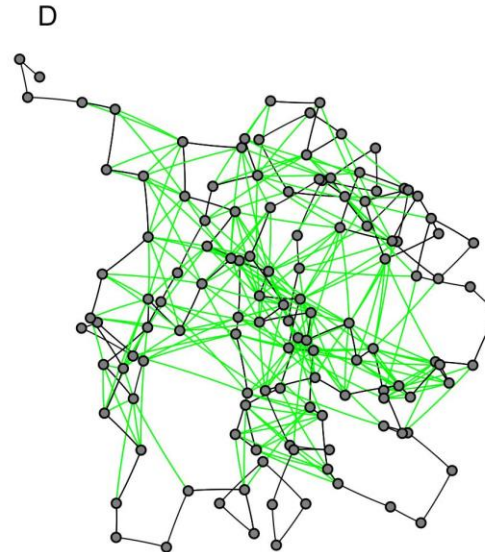
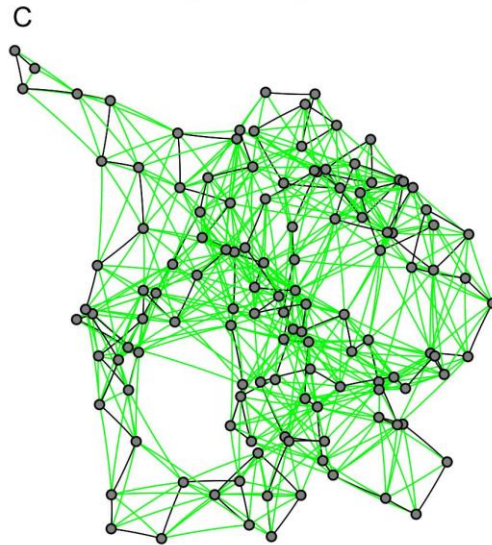
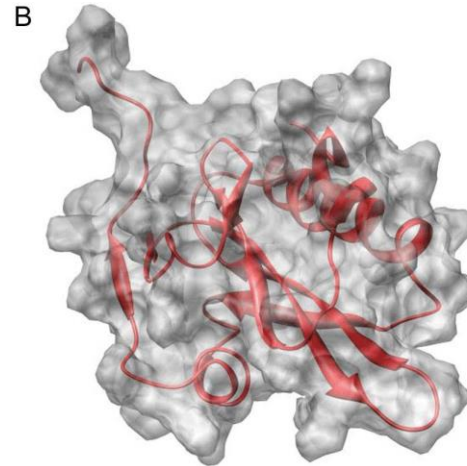
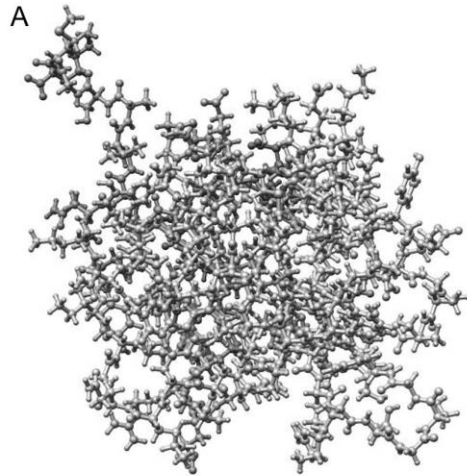


# Train maps



# Chemistry (Protein folding)

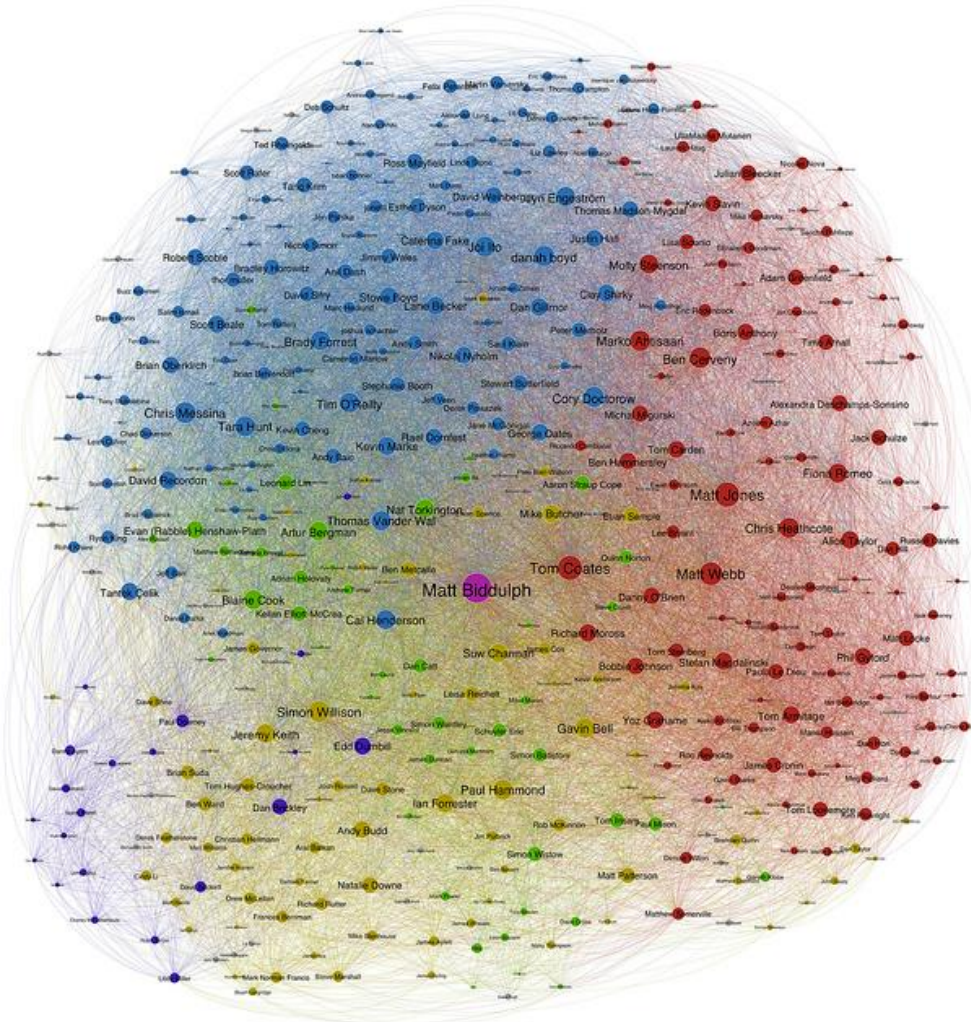
---





# Facebook friends

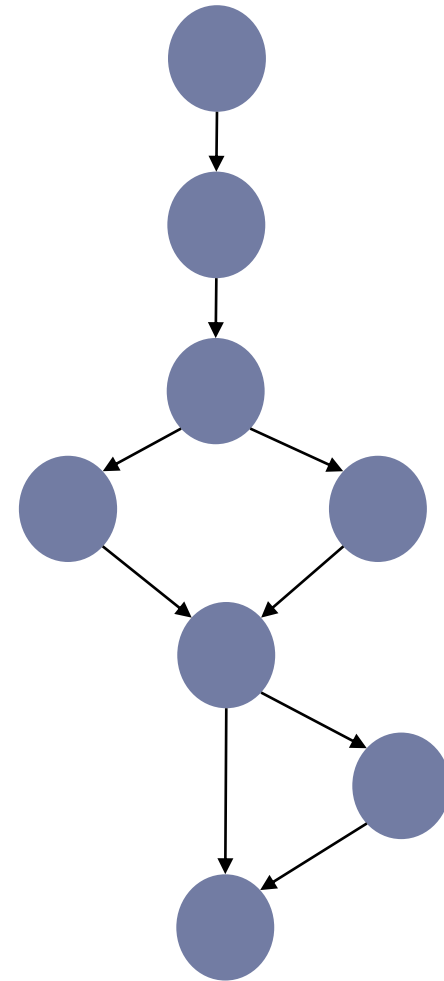
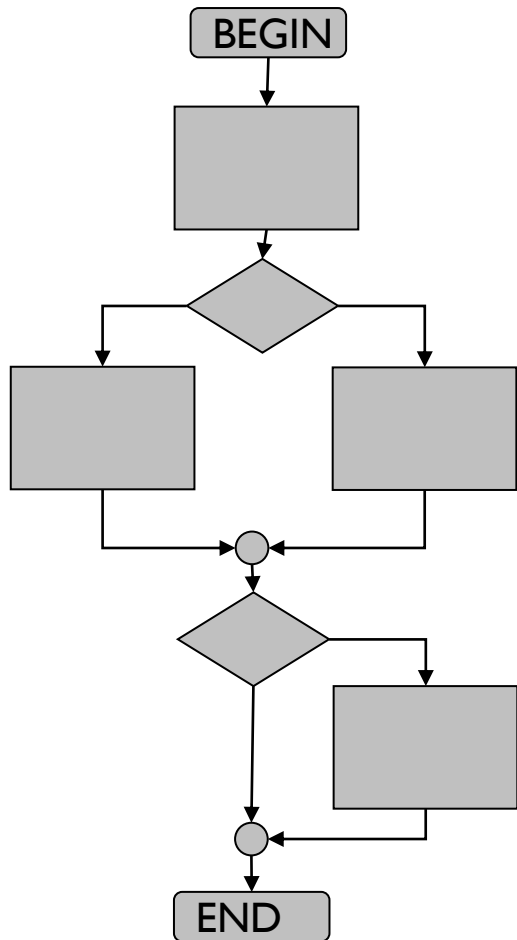
---



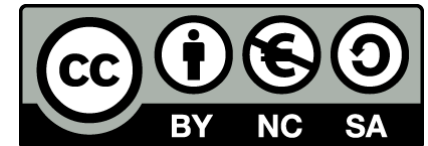







# Flow chart

---



# Licenza d'uso



- ▶ Queste diapositive sono distribuite con licenza Creative Commons “Attribuzione - Non commerciale - Condividi allo stesso modo (CC BY-NC-SA)”
- ▶ Sei libero:
  - ▶ di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera 
  - ▶ di modificare quest'opera 
- ▶ Alle seguenti condizioni:
  - ▶ **Attribuzione** — Devi attribuire la paternità dell'opera agli autori originali e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera. 
  - ▶ **Non commerciale** — Non puoi usare quest'opera per fini commerciali. 
  - ▶ **Condividi allo stesso modo** — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa. 
- ▶ <http://creativecommons.org/licenses/by-nc-sa/3.0/>