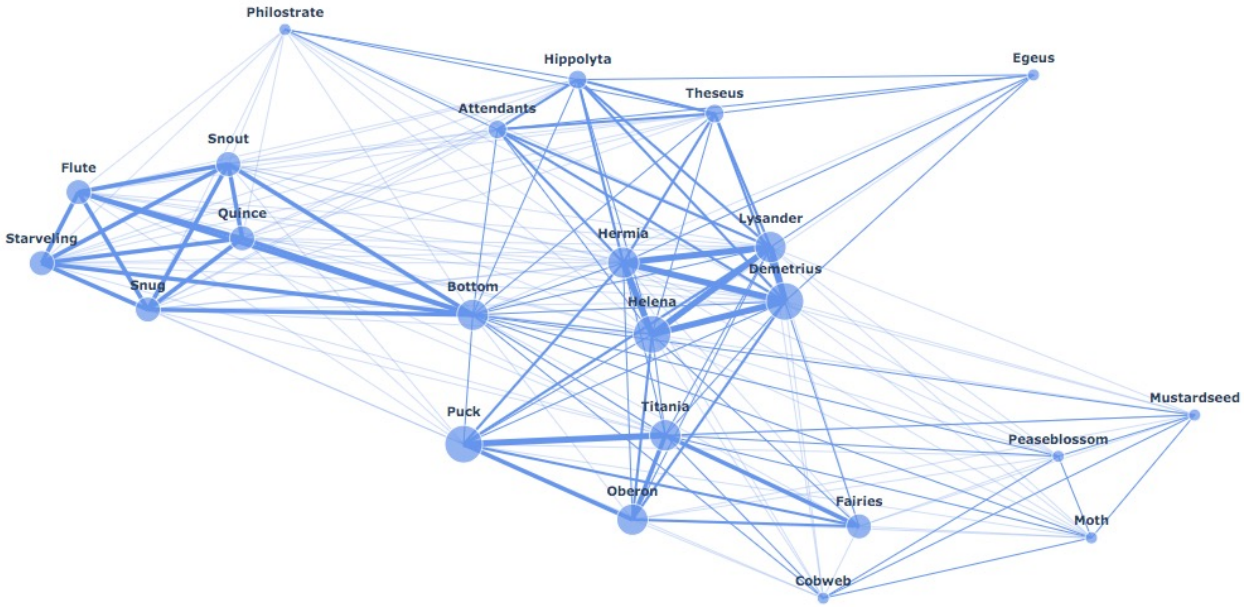
2024

TdP

Intro to Graphs

Theoretical foundations

Fulvio Corno
Giuseppe Averta
Carlo Masone
Francesca Pistilli



Introduction to Graphs

DEFINITION: GRAPH

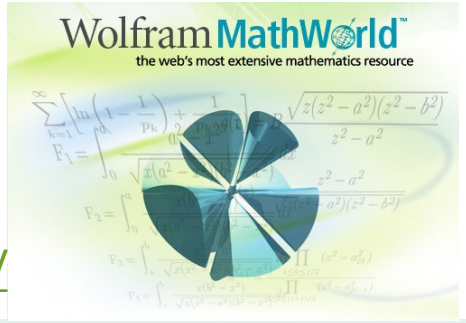
Definition: **Graph**

Un grafo è un oggetto matematico costituito da punti e linee: i punti sono tipicamente degli oggetti (nodi), mentre le linee possono rappresentare delle relazioni tra questi punti.

Punti → Vertici, nodi
Linee → Archi, spigoli

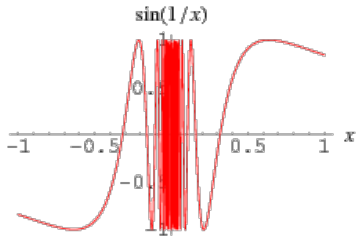
- A **graph** is a collection of **points** and **lines** connecting some (eventually empty) subset of them.
- The points of a graph are tipically known as **graph vertices**, but may also be called “nodes” or simply “points.”
- The lines connecting the vertices of a graph are called **graph edges**,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

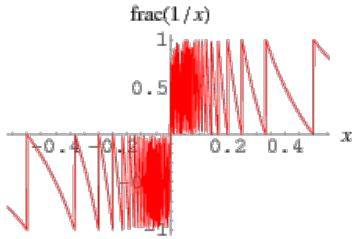
http://mathworld.wolfram.com/



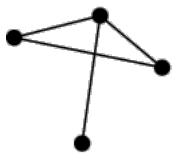
Big warning: Graph ≠ Graph ≠ Graph

- Graph (plot)
- (italiano: grafico)

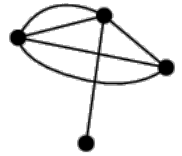




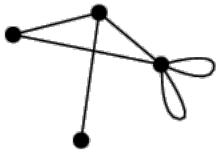
- Graph (maths)
- *(italiano: grafo)*



simple graph



nonsimple graph with multiple edges

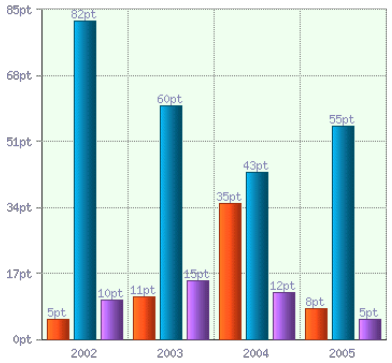


nonsimple graph with loops

≠

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.
- Indeed, the study of various sorts of paths in graphs has many applications in real-world problems.

Königsberg Bridge Problem

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

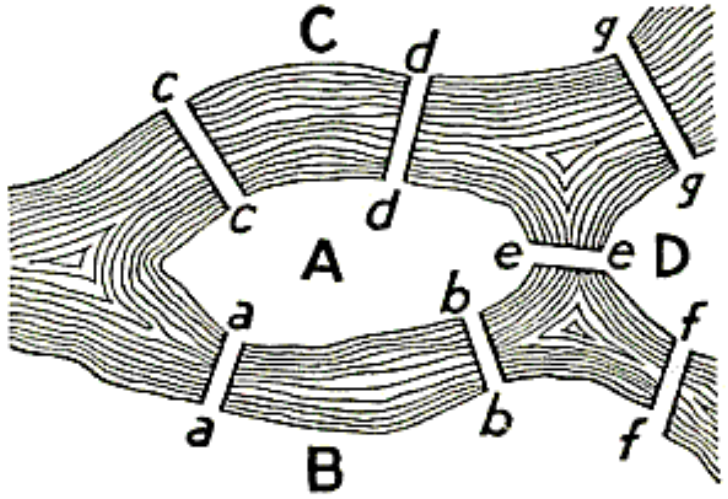


FIGURE 98. *Geographic Map: The Königsberg Bridges.*

• Problema: fare il percorso a Königberg attraversando tutti i ponti un'unica volta ritornando al punto di partenza.

IMPOSSIBILE!

Today: Kaliningrad, Russia

Königsberg Bridge Problem

- Can the 7 bridges the of the city of Königsberg over the river Preger all be traversed in a single trip without doubling back, with th quirement that the trip ends ace it began?

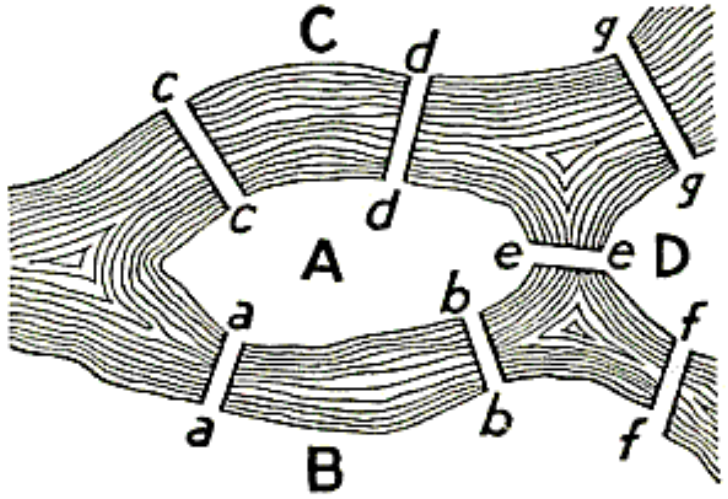
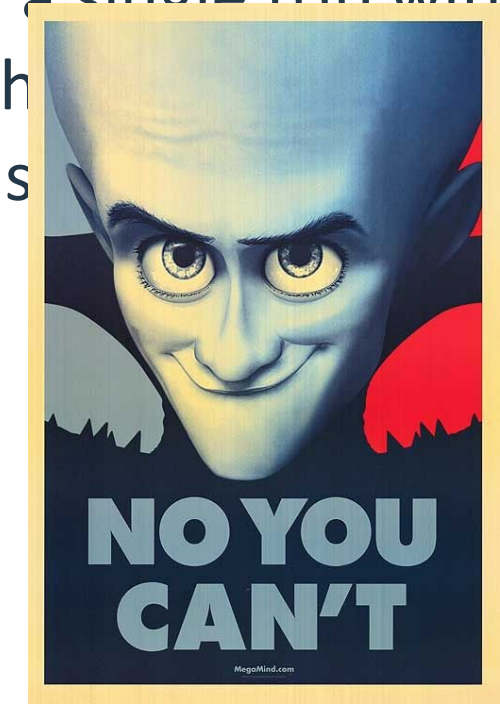


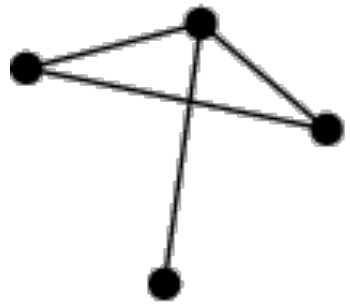
FIGURE 98. *Geographic Map: The Königsberg Bridges.*



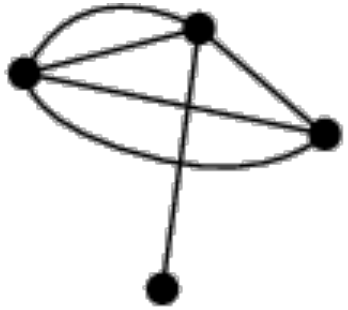
Today: Kaliningrad, Russia

Types of graphs: edge cardinality

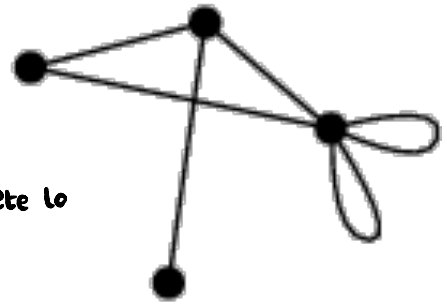
- Simple graph: Abbiamo una serie di nodi e due nodi possono essere collegati al max da un arco!
  - At most one edge (i.e., either one edge or no edges) may connect any two vertices
- Multigraph: Se i nodi possono essere collegati da più di un arco.
  - Multiple edges are allowed between vertices
- Loops: Grafi in cui ci sono degli archi che partono e arrivano allo stesso nodo.
  - Edge between a vertex and itself
- Pseudograph: Multigrafi con loop.
  - Multigraph with loops



simple graph



multigraph

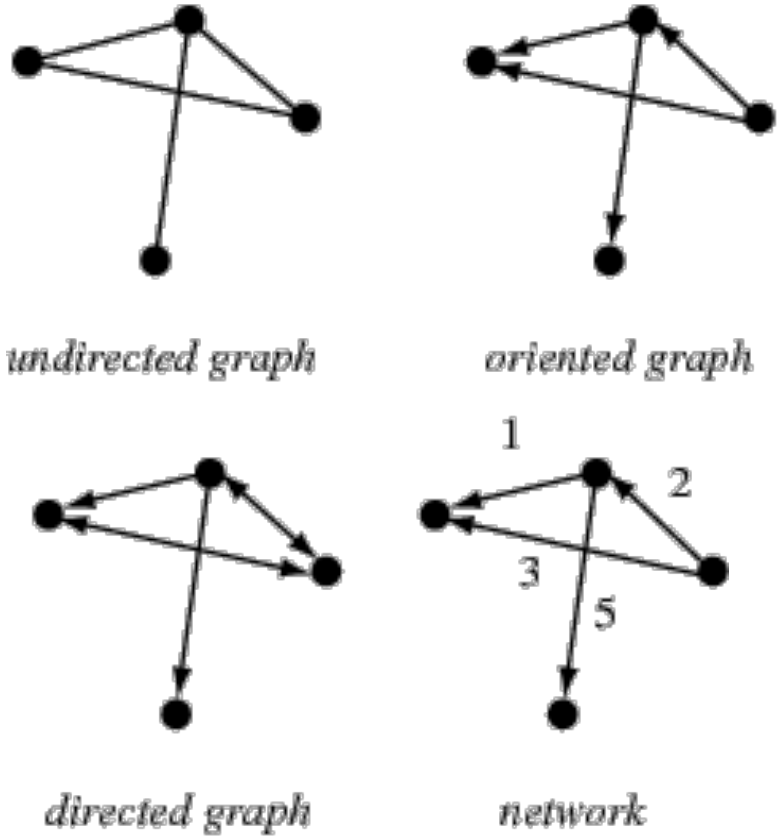


pseudograph

N.B: I nodi sono UNICI! Non posso aggiungere due volte lo stesso nodo a un grafo.

Types of graphs: edge direction

- Undirected ~> Archi NON direzionali
- Oriented ~> Archi direzionali con un solo verso!
  - Edges have **one** direction (indicated by arrow)
- Directed ~> Archi direzionali con uno o due versi.
  - Edges may have **one or two** directions
- Network ~> Grafi con un solo verso e pesati.
  - Oriented graph with weighted edges



undirected graph *oriented graph*

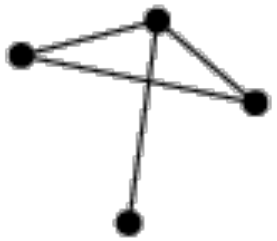
directed graph *network*

Types of graphs: labeling

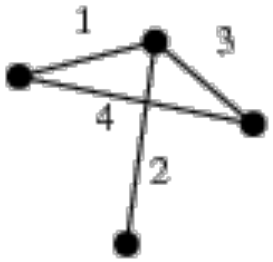
- Labels
  - None
  - On Vertices
  - On Edges
- Groups (=colors)
  - Of Vertices
    - no edge connects two identically colored vertices
  - Of Edges
    - adjacent edges must receive different colors
  - Of both

• Le Label possono essere aggiunte sia sugli archi che sui nodi e servono a associarle determinate informazioni.

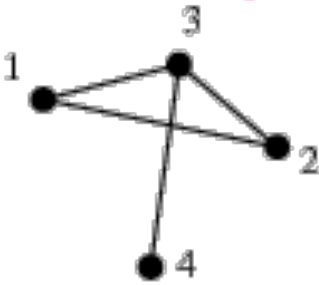
Vertici e archi possono essere raggruppati mediante colori.



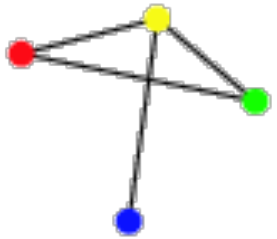
unlabeled graph



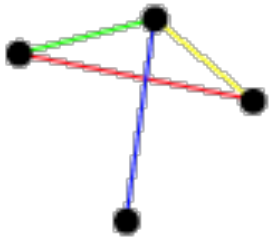
edge-labeled graph



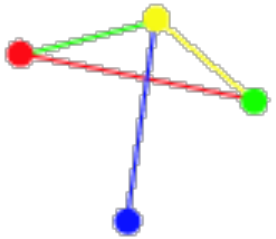
vertex-labeled graph



vertex-colored graph



edge-colored graph



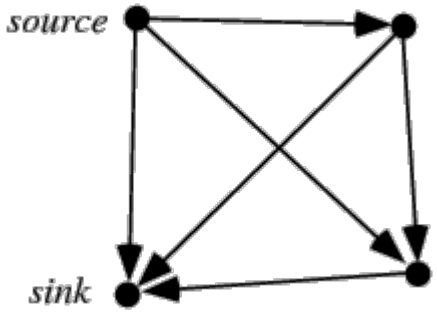
vertex- and edge-colored graph

Directed and Oriented graphs

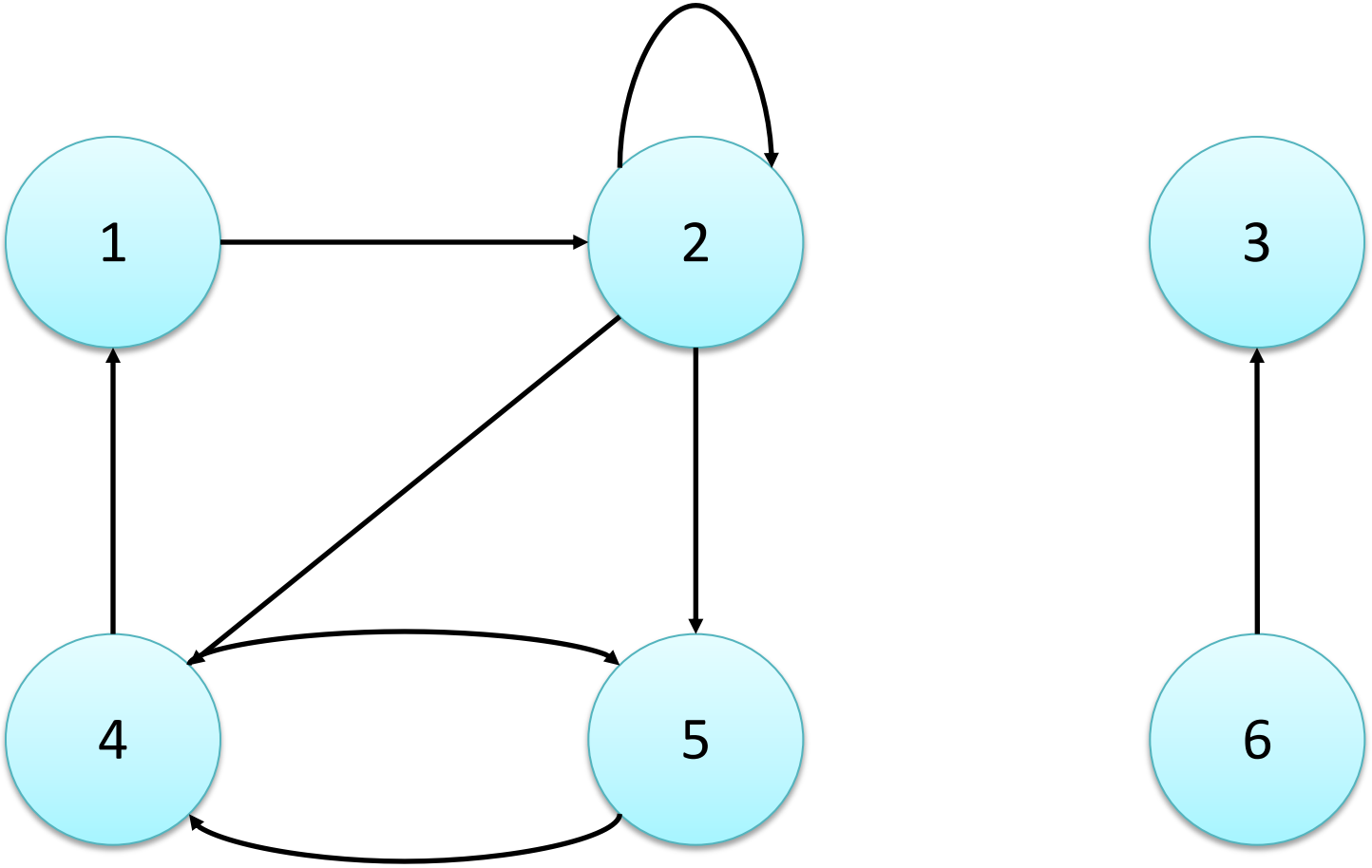
Grafo orientato!

- 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
    - $E \subseteq V \times V$

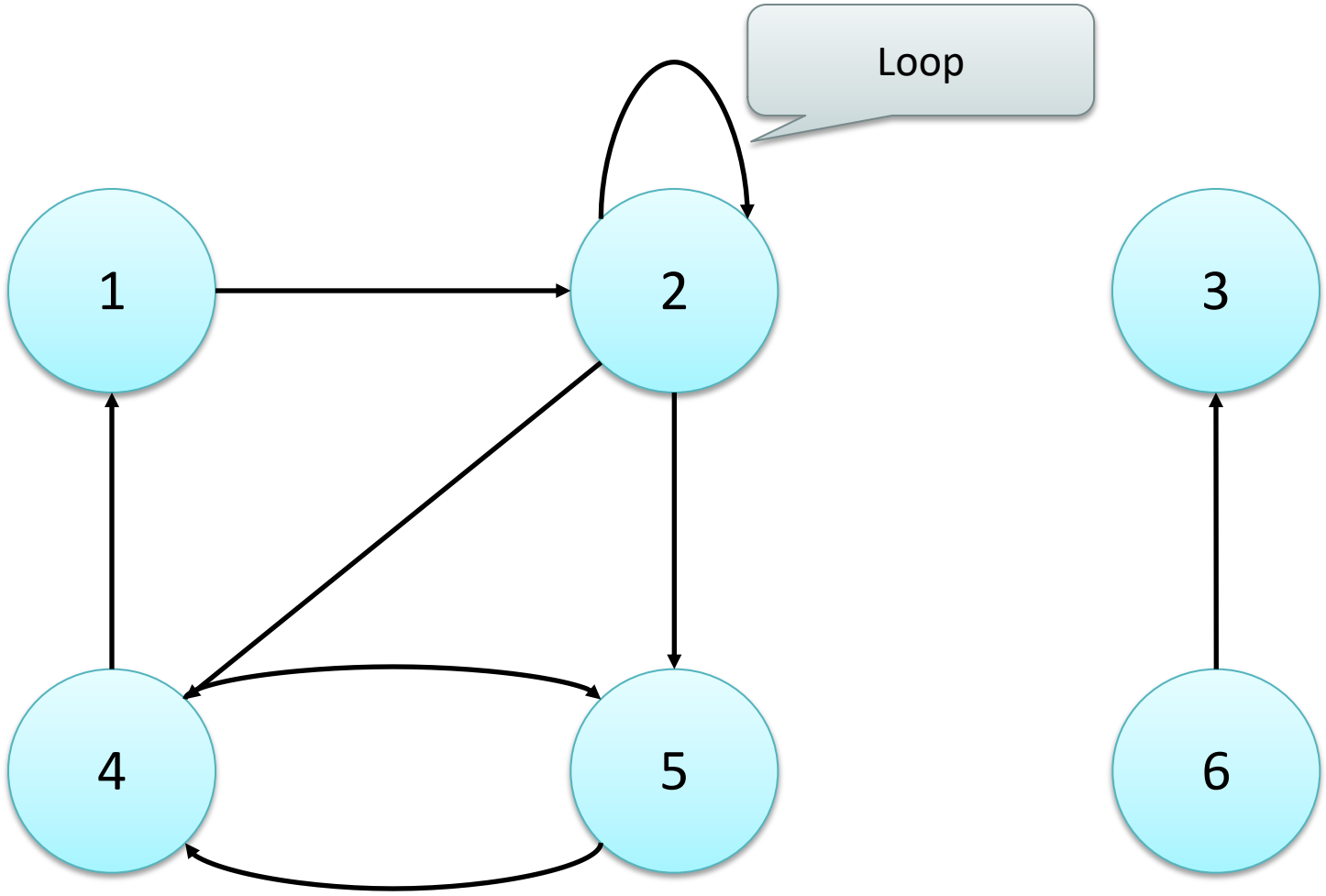
Si parte da un nodo e si finisce sempre a un altro nodo.



Example



Example



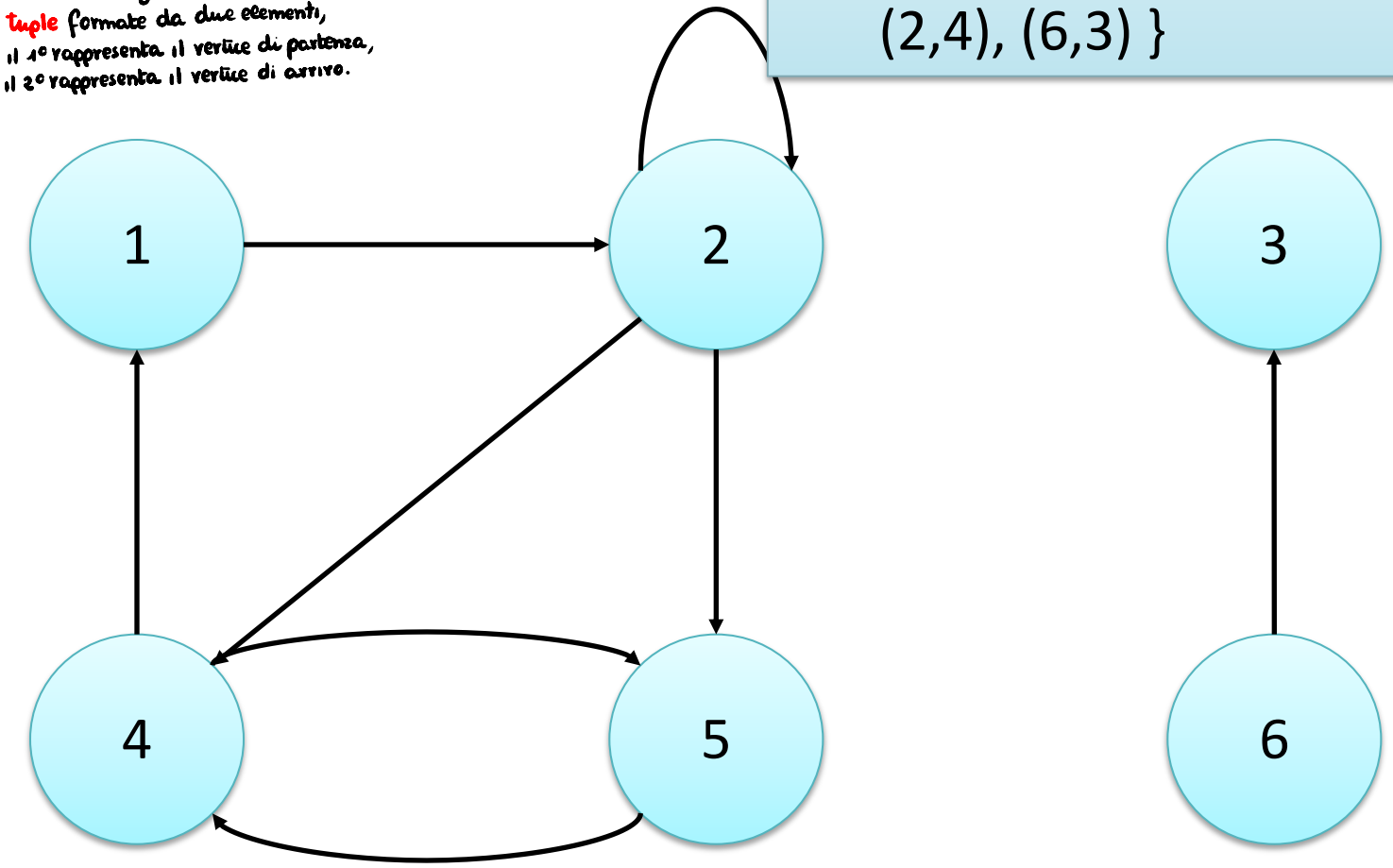
Example

- I vertici sono indicati in modo molto semplice;
- Gli archi vengono indicati come tuple formate da due elementi, il 1° rappresenta il vertice di partenza, il 2° rappresenta il vertice di arrivo.

Grafo rappresentato in maniera compatta!

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

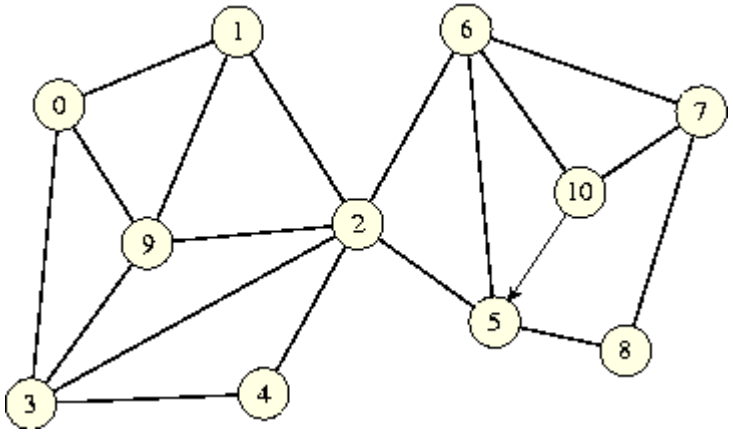
E = { (1,2), (2,2), (2,5), (5,4), (4,5), (4,1), (2,4), (6,3) }



Undirected graph

Grafo NON orientato

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



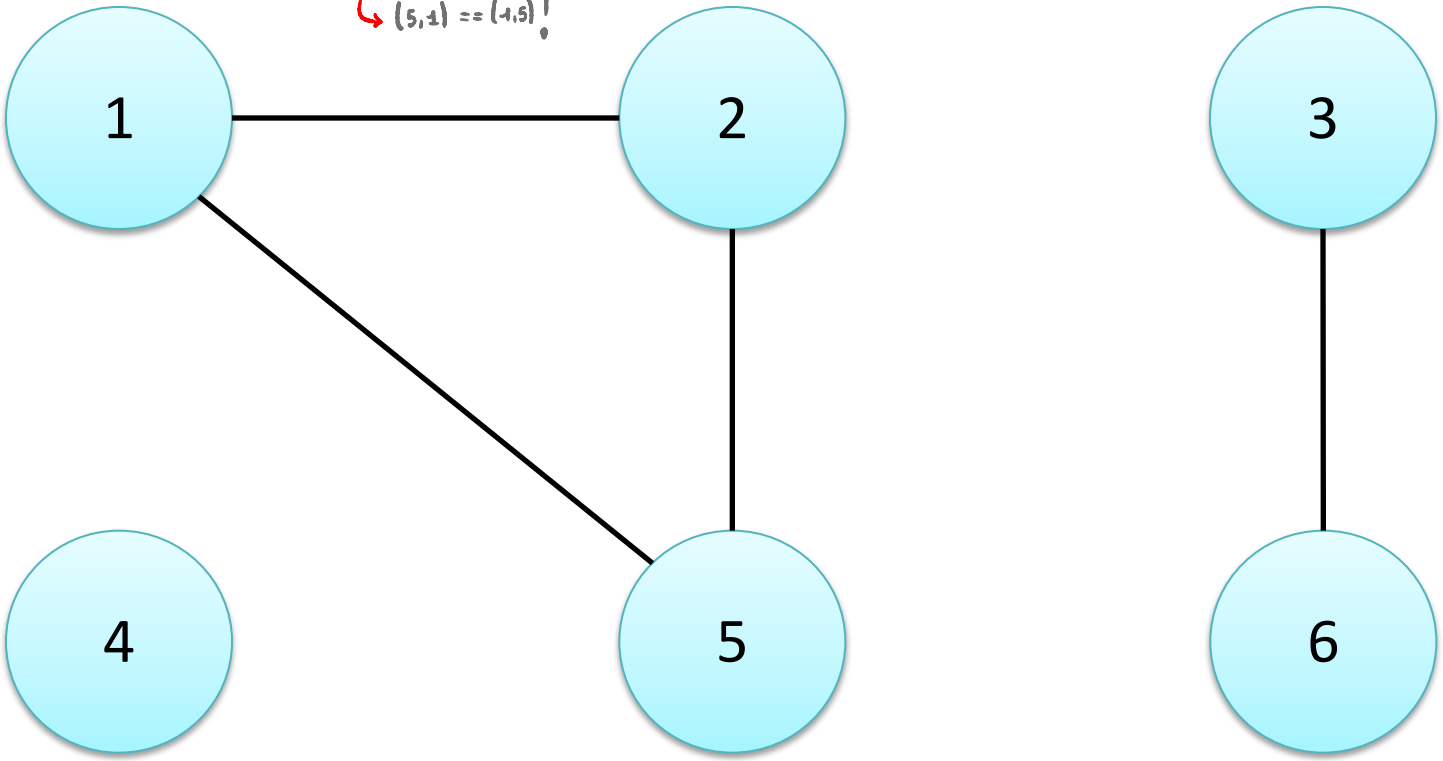
Example

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

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

In questo caso è indifferente il modo in cui indico gli archi, non avendo un verso!

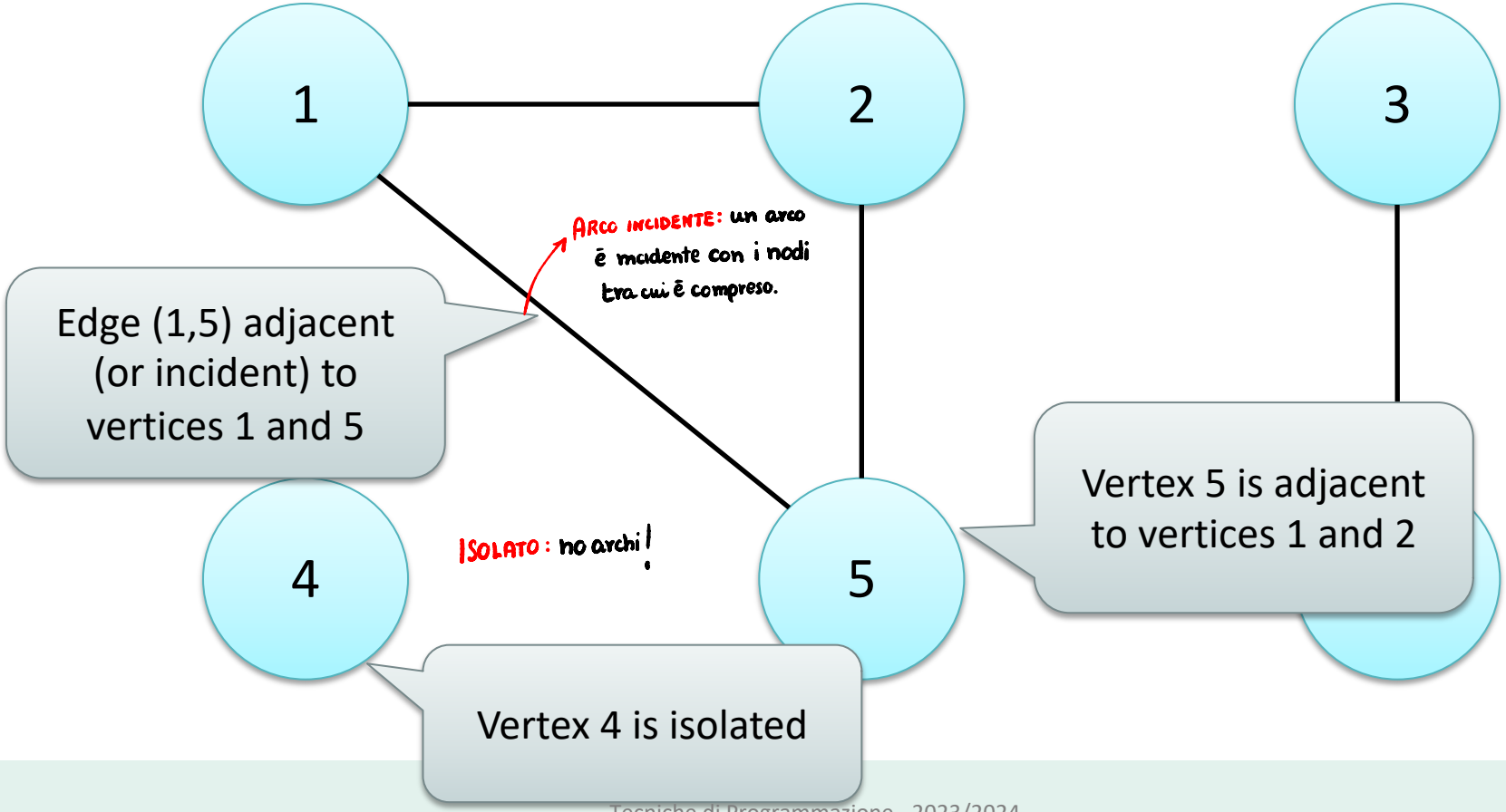
(5,1) == (1,5)!



Example

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

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



RELATED DEFINITIONS

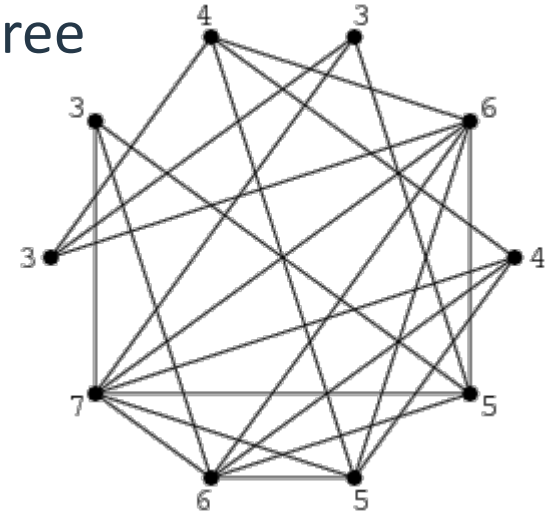
Degree

- 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**

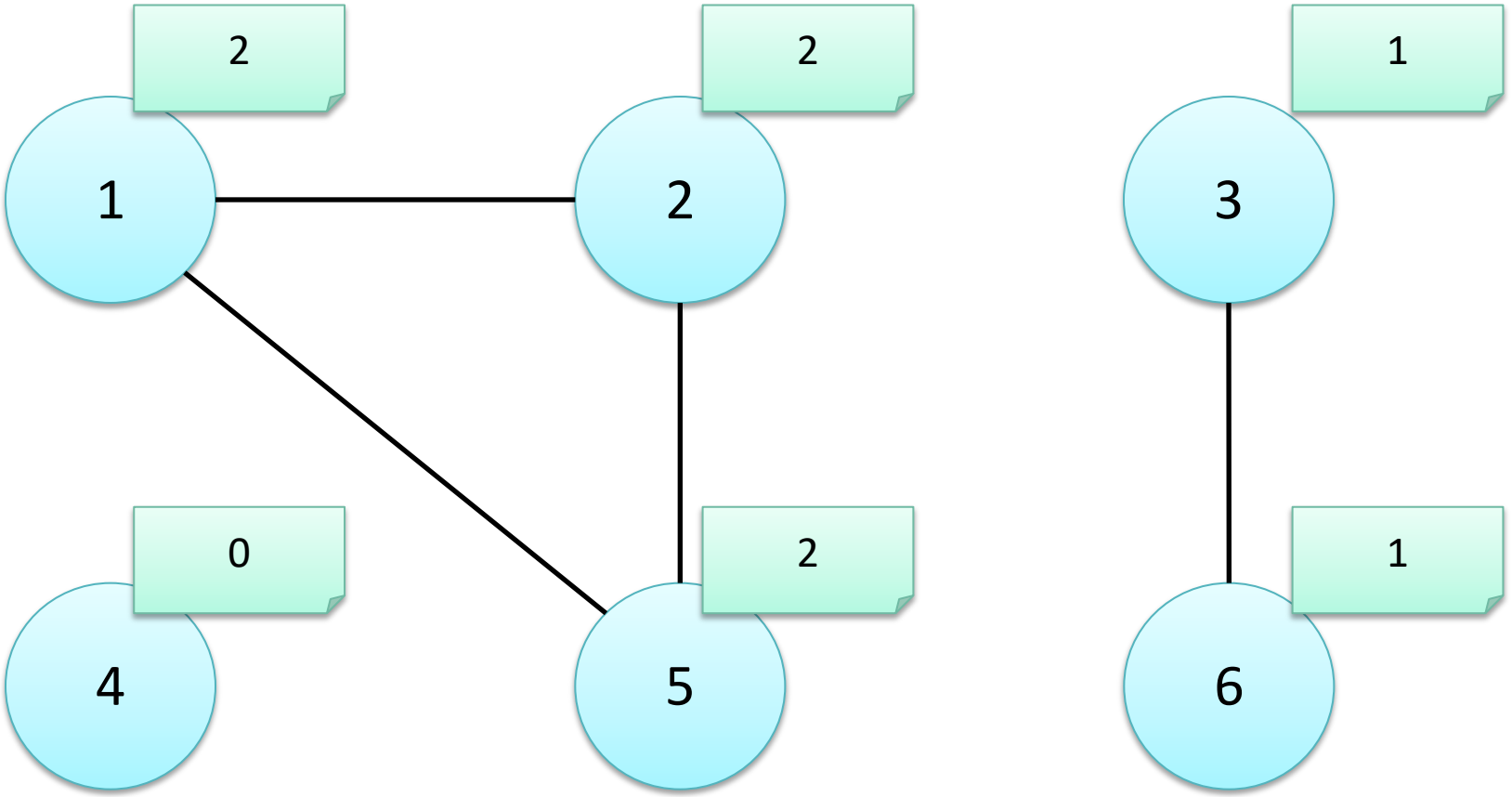
DEGREE: dimensionalità del nodo: numero di archi incidenti al nodo!

in-degree: numero di archi entranti nel nodo
out-degree: numero di archi uscenti dal nodo

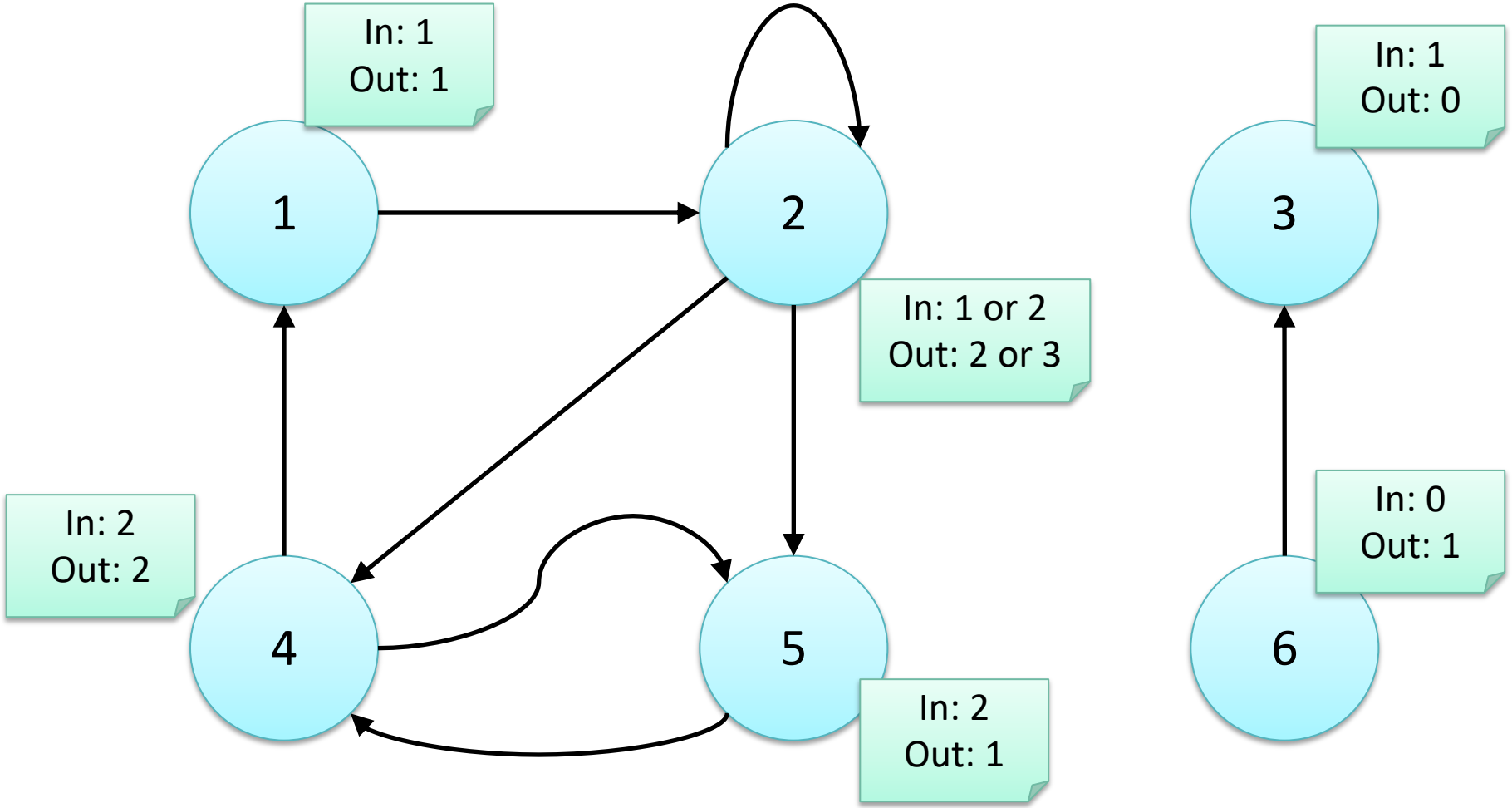
Un nodo con degree 0 è isolato!



Degree



Degree


In: 1
Out: 1
1
2
In: 1 or 2
Out: 2 or 3
In: 2
Out: 2
4
5
In: 2
Out: 1
In: 1
Out: 0
3
In: 0
Out: 1
6

Paths

Un percorso in un grafo è una sequenza di nodi tale x cui si può andare da nodo source a uno target distanti a piacere passando tra un nodo e uno successivo, che dovranno essere adiacenti! ↳ Path: lista ordinata di vertici!

N.B: Non è detto che ∃ un percorso fra due vertici!

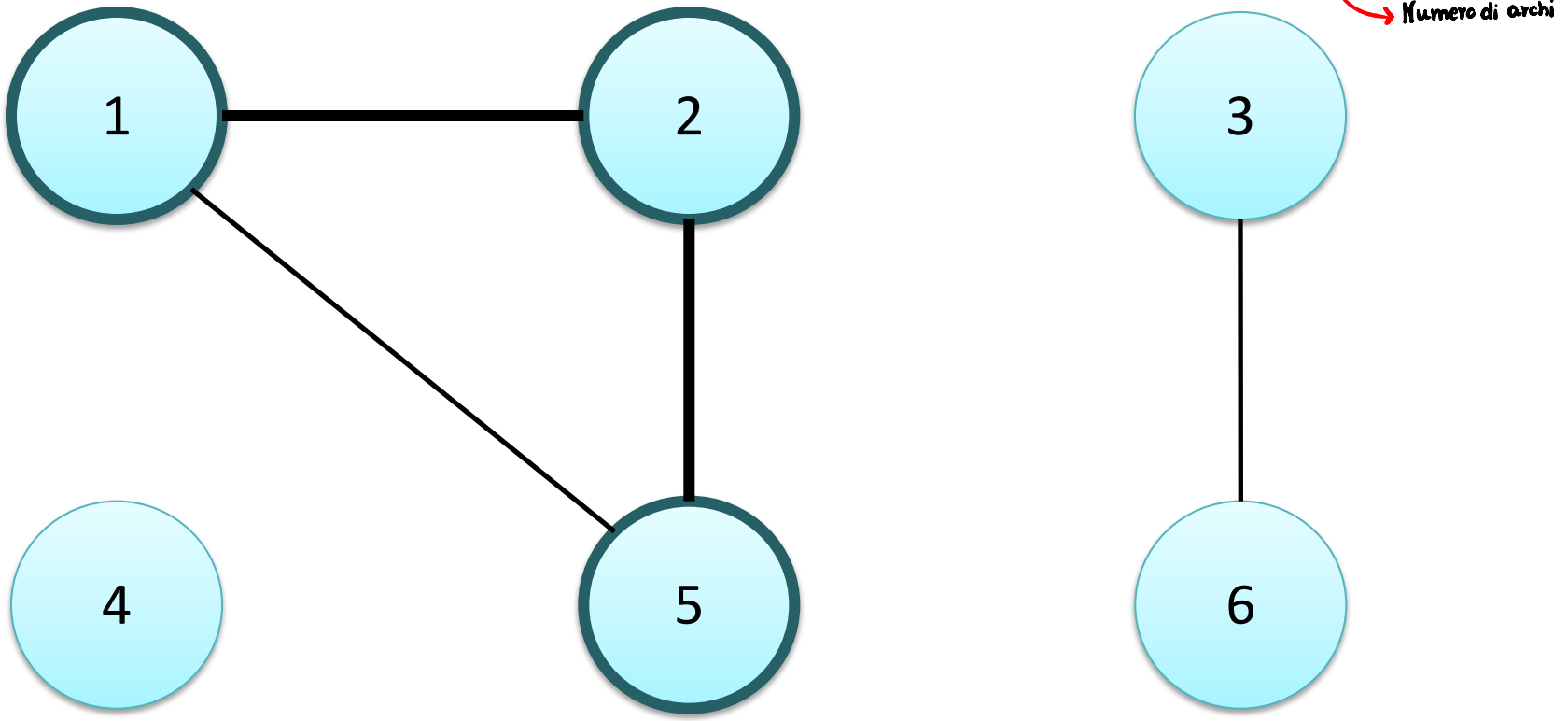
- A **path** on a graph G=(V,E), also called a trail, is a sequence $\{v_1, v_2, \ldots, v_n\}$ such that:
  - $v_1, \ldots, v_n$ are vertices: $v_i \in V$
  - $(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)$ are graph edges: $(v_{i-1}, v_i) \in E$
  - v_i are distinct (for "simple" paths).
- The **length** 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

Lunghezza percorso: numero di archi da attraversare!

Example

Path = (1, 2, 5)
Length = 2

Numero di archi



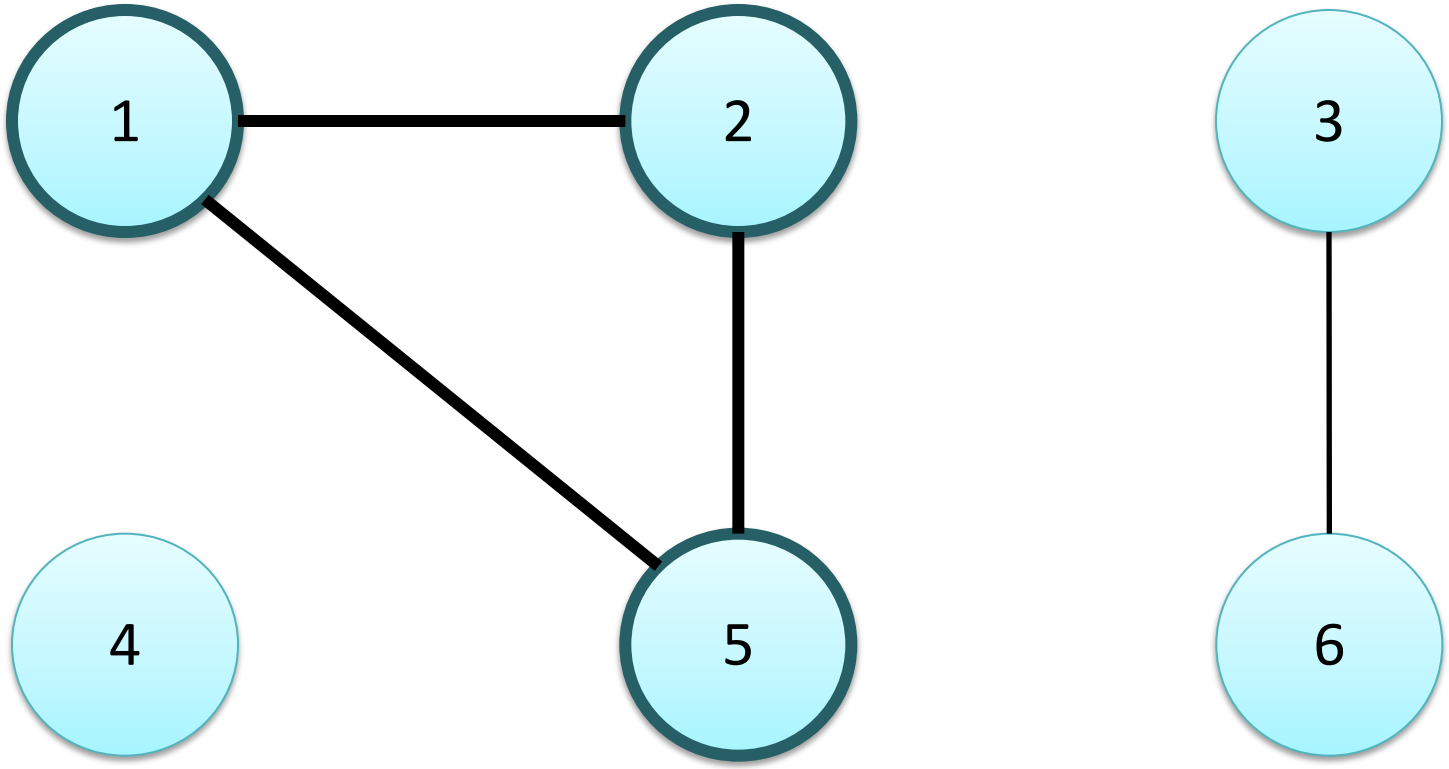
Cycles

Ciclo: percorso che parte da un nodo e arriva allo stesso nodo di partenza.

- 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)

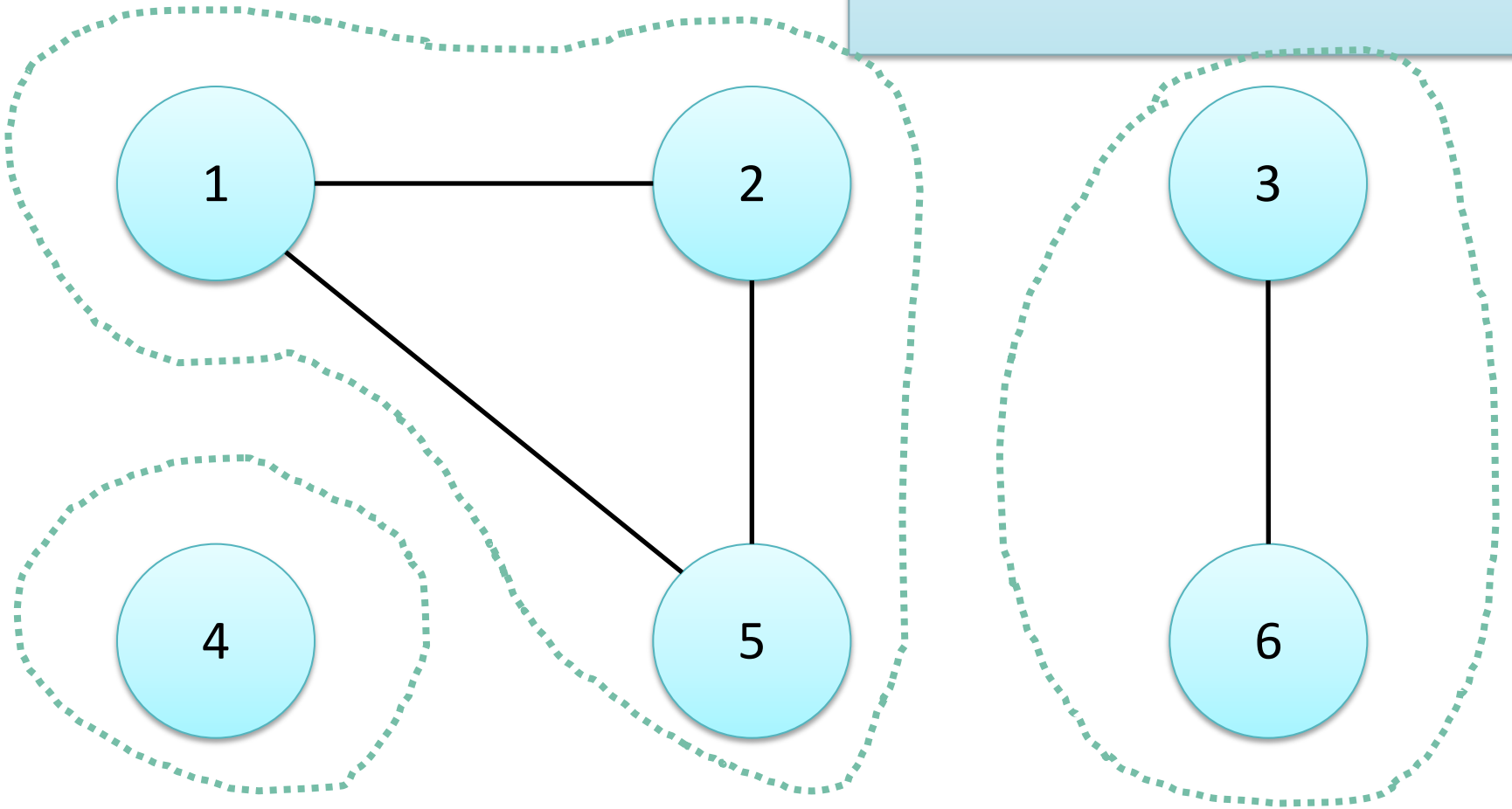
- An undirected graph is **connected** if, for every couple of vertices, there is a path connecting them
- The connected sub-graphs of maximum size are called **connected components**
- A connected graph has exactly one connected component

Nel caso di grafi non diretti, si dice che il grafo è connesso se dati due vertici, posso SEMPRE trovare un percorso tra questi due vertici.
In generale, posso sempre dividere il mio grafo non connesso in grafi più piccoli che sono connessi. I sotto-grafi connesso prendono il nome di componenti connesse. Può essere utile trovare la componente connessa più grande di un grafo non connesso (la dimensione del grafo si riferisce al numero di nodi del grafo!)
Un grafo connesso ha ESATTAMENTE una componente connessa.

Connected components

- tre ≠ componenti connesse in un grafo non connesso!

The graph is **not** connected.
Connected components = 3
{ 4 } , { 1, 2, 5 }, { 3, 6 }

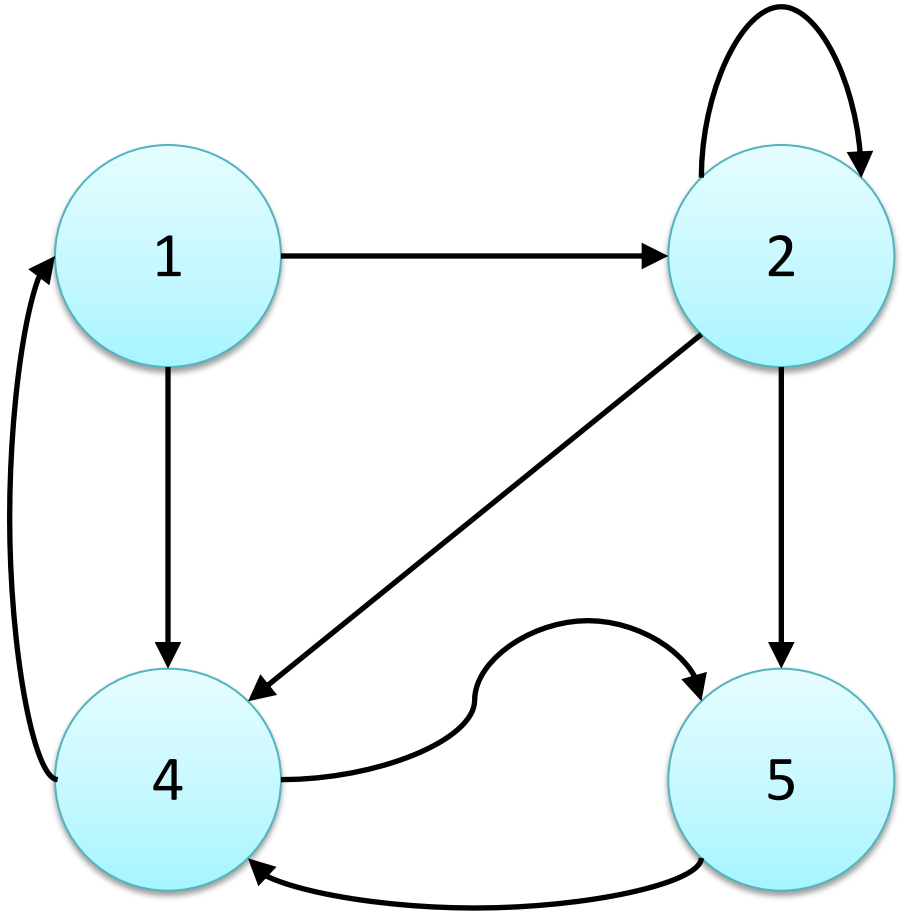


Reachability (Directed)

- 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'

Grafo fortemente connesso: se ∀coppia di nodi, ∃ almeno un percorso che collega i due nodi.

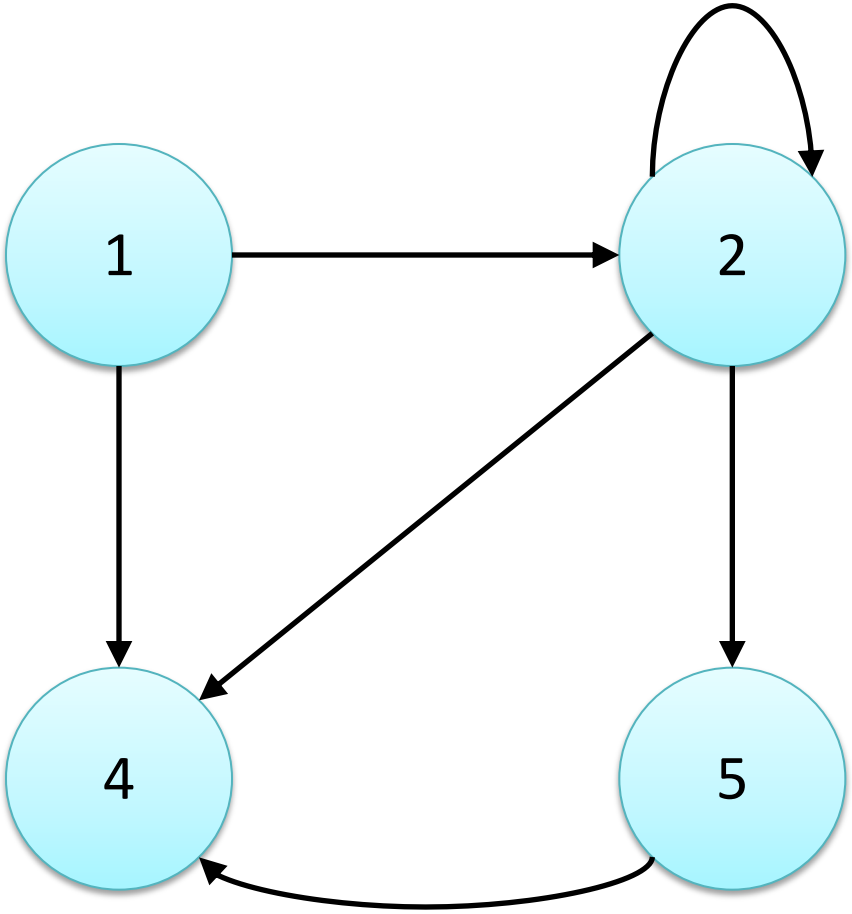
Example



The graph is **strongly connected**

Example

The graph is **not** strongly connected



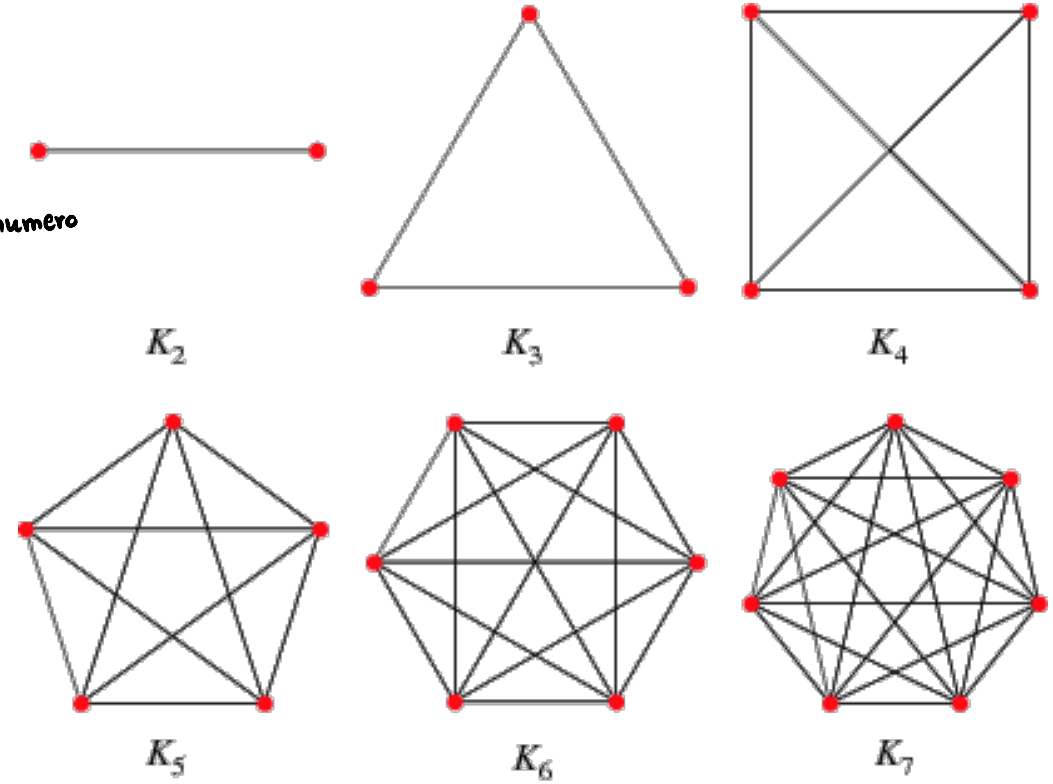
Perché non posso andare dal nodo 4 a qualsiasi altro nodo!

Complete graph

Grafo completo: ∀ coppia di nodi, ∃ un arco (singolare) che li collega!

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

K_n: Grafo completo di dimensione n; dove n è il numero di nodi.



Complete graph: edges

- In a **complete** graph with n vertices, the number of **edges** is

	Directed	Undirected
No self loops	$n(n-1)$	$\frac{n(n-1)}{2}$
With self loops	n^2	$\frac{n(n+1)}{2}$

Numero di archi in un grafo completo di dimensione n!

Density

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

Densità grafo: Numero archi del mio grafo / Numero archi di un grafo completo di dimensione pari al numero di vertici del grafo

↓

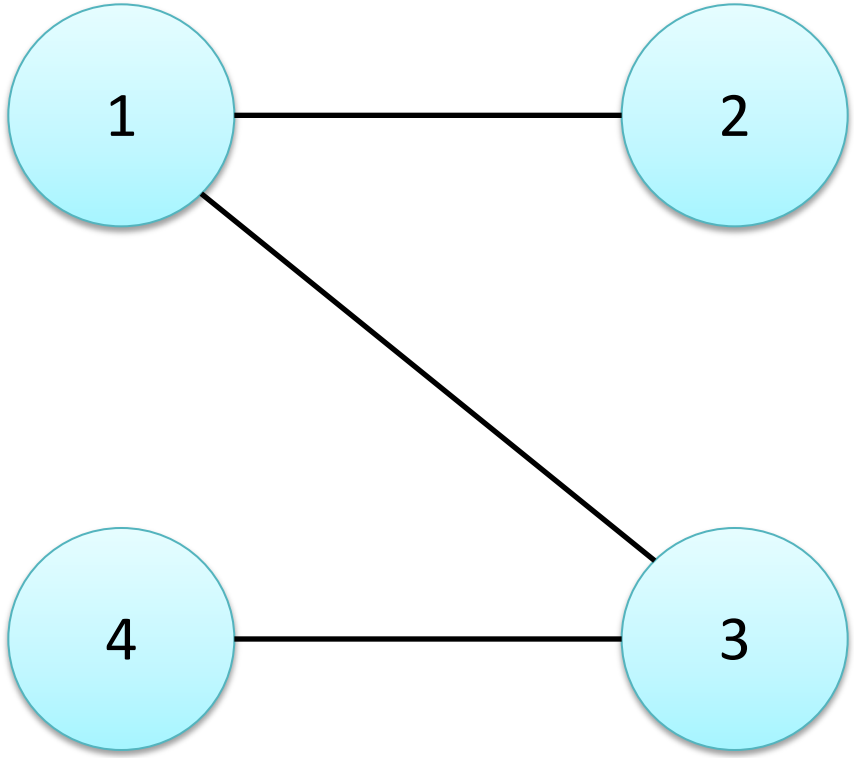
Rapporta il numero di archi del grafo in funzione del numero max di archi che potrei avere!

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

Example

Density = 0.5

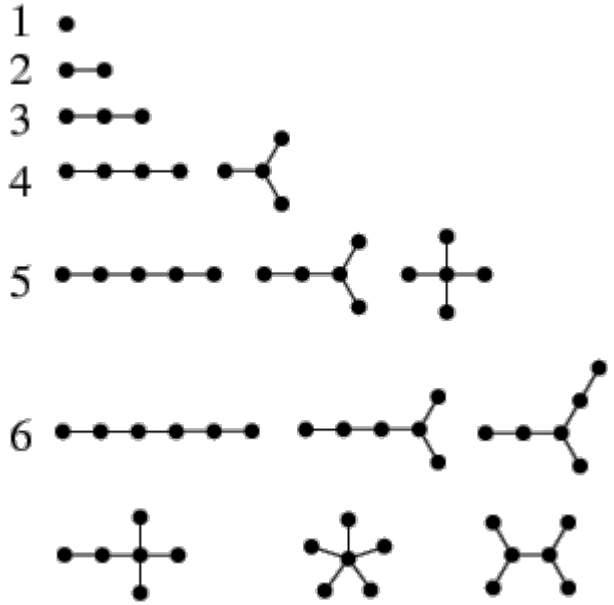
Existing: 3 edges
Total: 6 possible edges



Trees and Forests

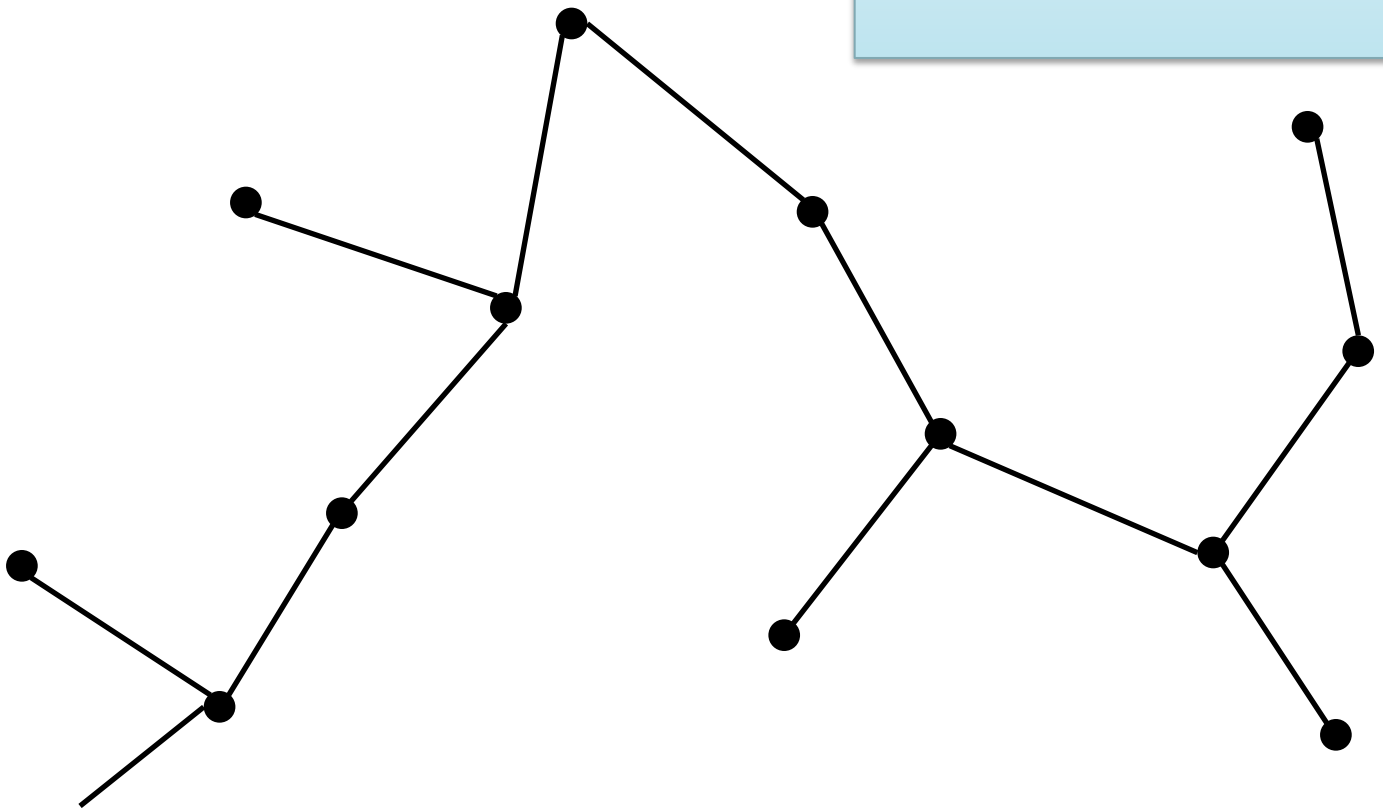
- Foresta: grafo NON diretto aciclico. ⇝ Insieme di alberi!
- Albero: grafo NON diretto, aciclico, ma CONNESSO.

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



Example

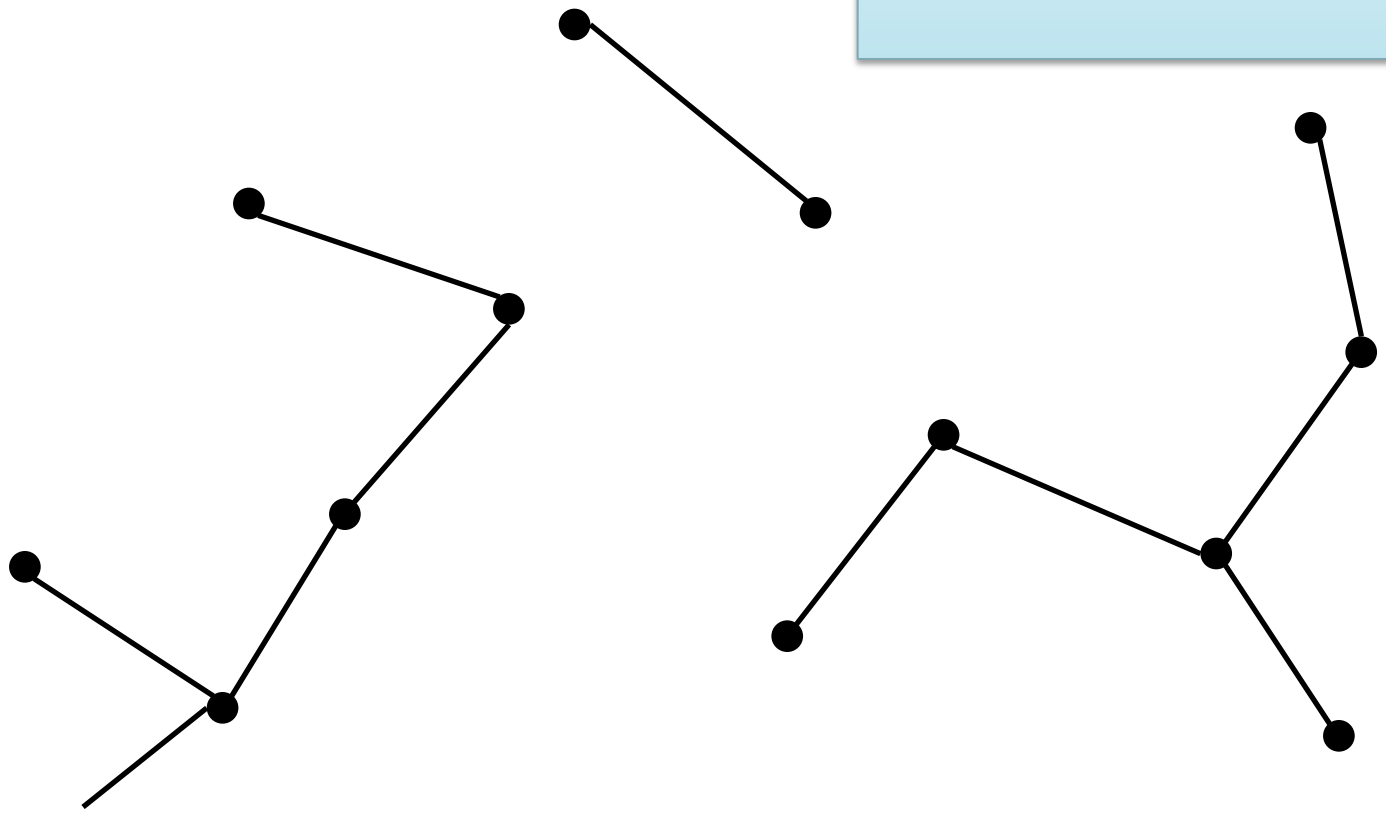
Tree



Example

Forest

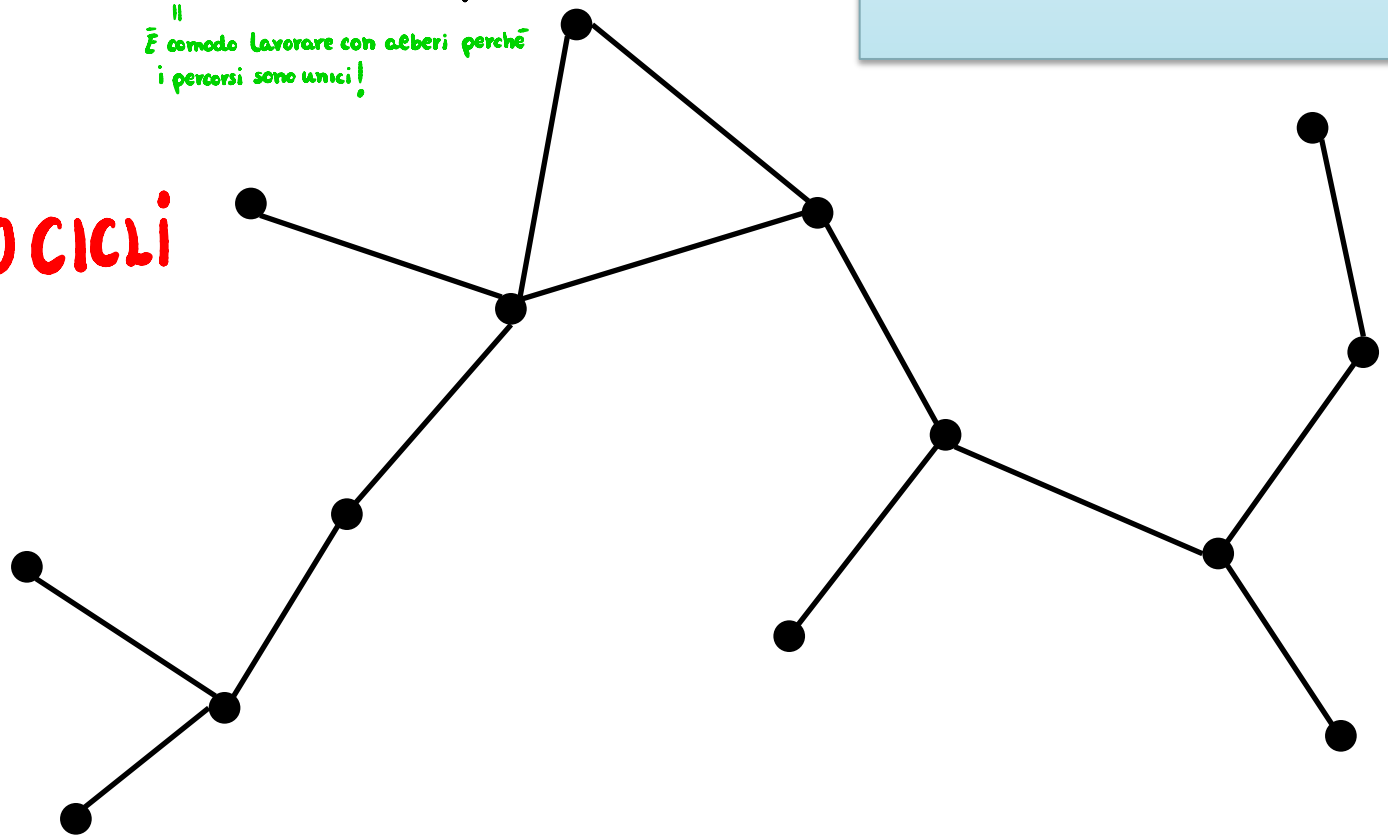
Insieme di tre alberi!



Example

- Se in un grafo è presente un ciclo, allora il grafo NON sarà un albero!
  - ↳ In un albero, partendo da un nodo non potrò + raggiungere lo stesso nodo in nessun modo!
    - ||
    - È comodo lavorare con alberi perché i percorsi sono unici!

ALBERO ⇝ NO CICLI

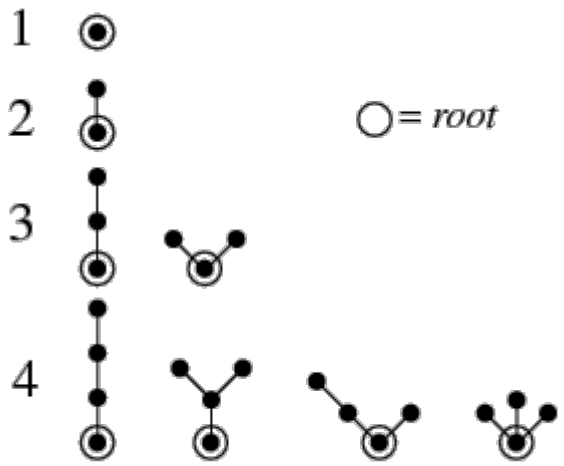


This is not a tree nor a forest
(it contains a cycle)

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

↪ Sugli alberi possiamo identificare un nodo che è favorito rispetto agli altri che si chiama radice. È il punto da cui parto a definire i ≠ percorsi.



rooted trees

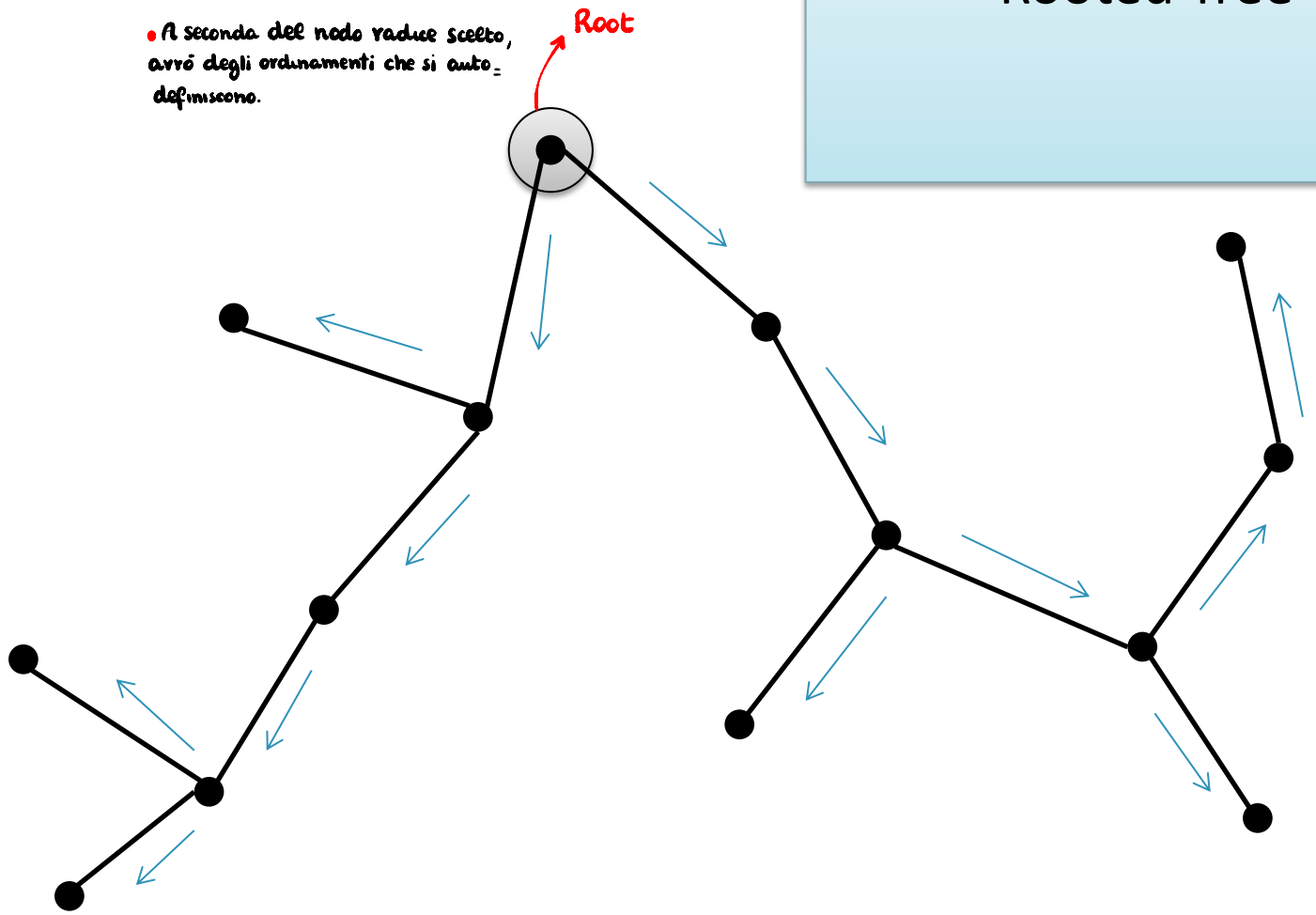
Tree (implicit) ordering

L'albero, implicitamente, definisce già un ordinamento. Perchè se definisco un nodo radice, implicitamente sto dicendo da un nodo a un altro avrò una direzione, perchè se il percorso è unico, anche la direzione sarà unica e ci sarà un unico ordinamento implicitamente determinato. Questo ordinamento definisce anche chi sono le radici e chi sono le foglie dell'albero. Le direzioni vanno sempre dai nodi più "prossimali" alla radice verso i nodi più distanti dalla radice.

- 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 furthermost (children-of-children-of-children…) nodes are "leaves"

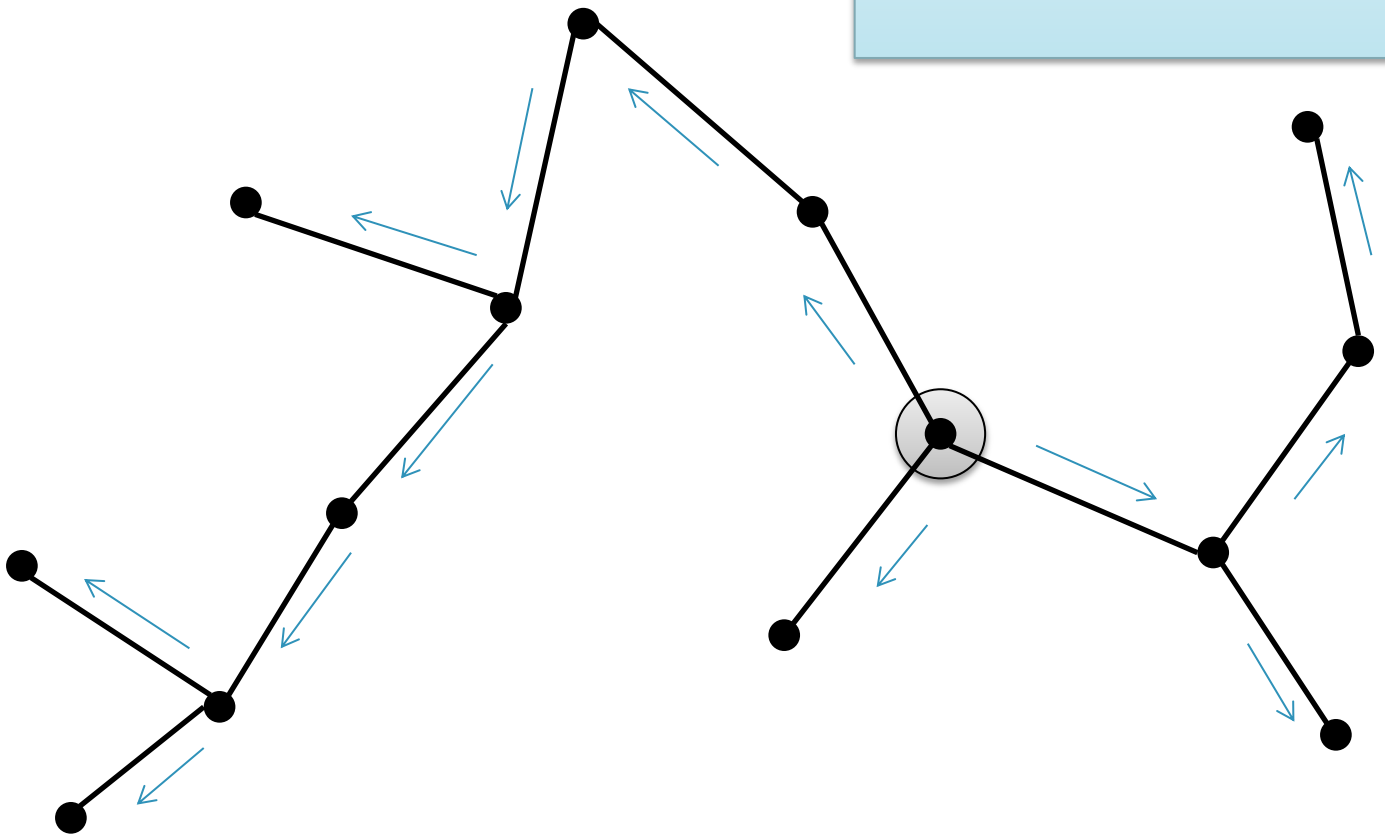
Example

Rooted Tree



Example

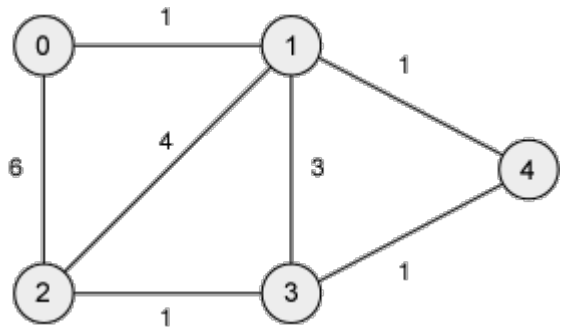
Rooted Tree

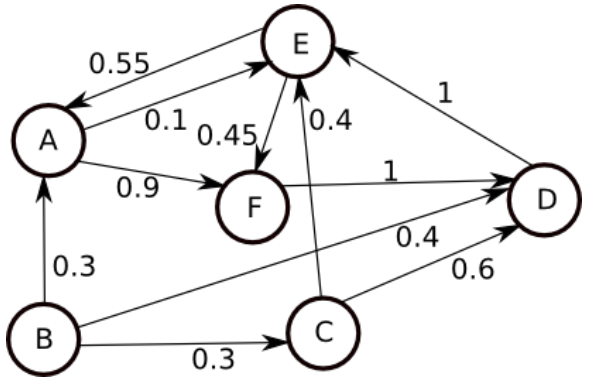


Weighted graphs

• Grafi pesati: ciascun arco che collega due nodi è caratterizzato da un costo (peso)!

- 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).





License

- These slides are distributed under a Creative Commons license “**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**”
- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- https://creativecommons.org/licenses/by-nc-sa/4.0/