Graph Theory

Graphs

Sergio Peignier sergio.peignier@insa-lyon.fr

Associate Professor
INSA Lyon
Biosciences department

Table of contents

1. Motivation | Definition
2. (Some) Types of Graphs
3. Walking in the Graph
4. Subgraphs: the Graphs Inside
5. Trees and Forests
6. Representing Graphs
7. Exercises

Motivation | Definition

Graph | Motivation

- Need to **understand complex systems**.
- **Data availability**: Big data era.
- **Universality**: Systems share **similar features**.
- **Graphs: Common framework** to study many kinds of **complex systems**.

Complex systems

- **System** composed of **many interacting components**
- **Different** kinds of **interactions** between the parts
  e.g., competition, dependency, ...
- **Arising properties**
  e.g., emergence, adaptation, spontaneous order ...
- **Alternative paradigm to reductionism**

Graph | Informal definition

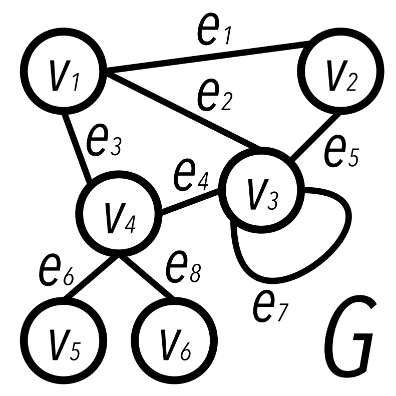
- **Mathematical structure**
- **Model** pairwise **relations** between **objects**.
- Objects $\rightarrow$ **vertices** or **nodes**
- Relation $\rightarrow$ **edges**, **arcs**, or **lines**.

Graph | Formal definition

- **Graph:** $G = \langle V, E \rangle$
- **Set of nodes:** $V = \{v_1, v_2, \dots\}$
- **Set of edges:** $E = \{e_1, e_2, \dots\}$ s.t.

  $e = \langle v_i, v_j \rangle \in E$ **if**:
  $v_i \in V$, $v_j \in V$ and $\exists$ **connection between** v_i and v_j
- v_i and v_j are the **endpoints** of the edge
- v_i and v_j are **adjacent** nodes
- e is **incident** to v_i and v_j

Graph | Definition



- Set of nodes: $V = \{v_1, v_2, \dots\}$
- Set of edges: $E = \{e_1, e_2, \dots\}$
- Nb. of nodes: $|V|$, Nb. of edges: $|E|$
- Graph: $G = \langle V, E \rangle$

node	edge
person	friendship
neuron	synapse
company	ownership
gene	regulation

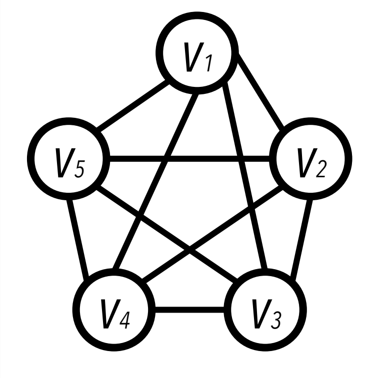
(Some) Types of Graphs

Empty Graph

$$G = \langle V, E \rangle, \quad E = \emptyset, V = \emptyset$$

Complete Graph

$$G = \langle V, E \rangle, \quad E = \{\langle v_i, v_j \rangle\} | \langle v_i, v_j \rangle \in V^2, \text{ and } v_i \neq v_j\}$$

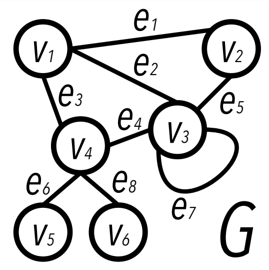


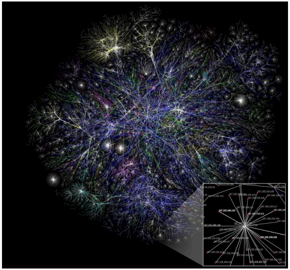
Question: $|E| = ?$

Undirected Graphs

$$G = \langle V, E \rangle, \quad \langle v_i, v_j \rangle \in E \equiv \langle v_j, v_i \rangle \in E, \quad E \subseteq V \times V$$

- The **directions** of edges do **not matter**.
- **Interactions** are possible in **both directions**.
- e.g., Internet backbone
  nodes: routers, edges: physical wires

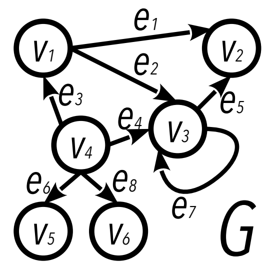


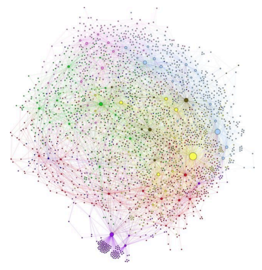


Directed Graphs

$$G = \langle V, E \rangle, \quad \langle v_i, v_j \rangle \in E \not\equiv \langle v_j, v_i \rangle \in E$$

- The **directions** of edges **matter**.
- **Interactions** are possible in **specified directions**.
- e.g., Honeybee transcriptional regulatory network [Sobotka et al.].
  nodes: genes, edges: regulation

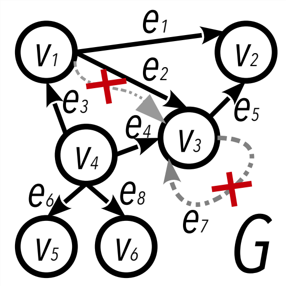




Simple Directed Graphs (digraph)

$$G = \langle V, E \rangle, \quad \langle v_i, v_j \rangle \in E \not\equiv \langle v_j, v_i \rangle \in E \text{ and } v_i \neq v_j$$

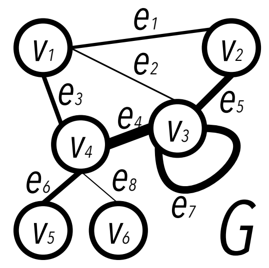
- **Directed graph**
- **No loops** (otherwise: loop-digraph)
- **No multiple arrows** (otherwise: directed multigraph)

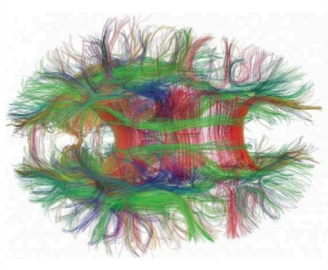


Weighted Graphs

$$G = \langle V, E, w \rangle, \quad w : \langle v_i, v_j \rangle \to \mathbb{R}$$

- **Interaction strength** assigned by the **edge weight**.
- e.g., Connectome [McCoss et al.].
  nodes: neurons, edges: synapses

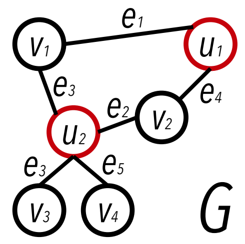


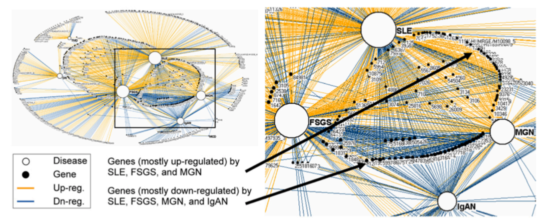


Bipartite Graphs

$$G = \langle U, V, E \rangle, \quad U \cap V = \emptyset, \quad \forall \langle u, v \rangle \in E, u \in U \text{ and } v \in V$$

- **Interactions** between **two different** kinds of **nodes**.
- e.g., Gene-disease network [Bhavnani et. al].
  nodes: genes/diseases, edges: relationship

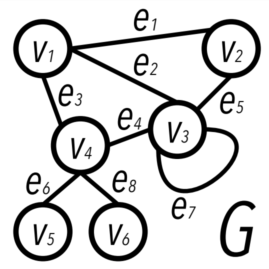




Planar Graphs

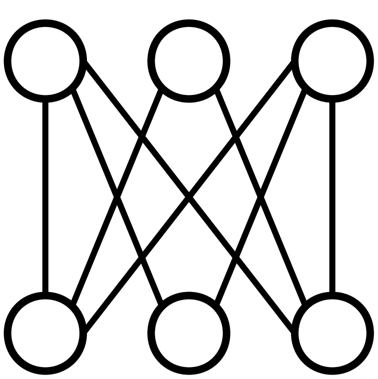
$$G = \langle V, E, loc \rangle, \quad loc : v \to (x, y) \in \mathbb{R}^2$$

- **Nodes located** in a **plane** (**Geo-localised**) s.t. **edges only intersect in endpoints.**
- e.g., Rail system [Viana et. al]. nodes: station, edges: railroad



Planar Graphs | Question

Is this graph planar?

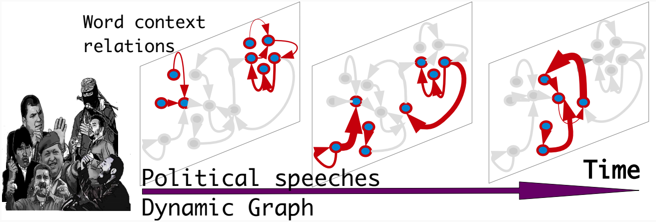


Temporal Graphs

$$G = \langle V_T, E_T \rangle, \quad \forall (t, d) \in \mathbb{R}^2_+ : v(t) \in V_T, \langle v_i, v_j, t, d \rangle \in E_T$$

- d: **duration**, t: **time stamp**
- Temporal edges encode **dynamic interactions**.
- Dynamic nodes encode the **graph evolution**.
- e.g., political speech semantic evolution

  nodes: words, edges: contextual relations

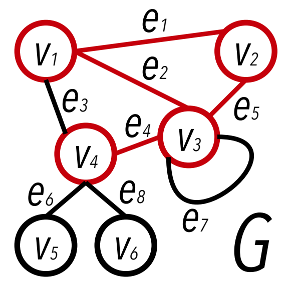


Walking in the Graph

Walk

Walk Sequence of alternating nodes and edges $(v_o, e_1, v_1, e_2, v_2, \ldots)$ s.t.:

- $\forall i \leq k,\ e_i = \langle v_{i-1}, v_i \rangle$
- k: **length** of the walk (nb. of visited edges)

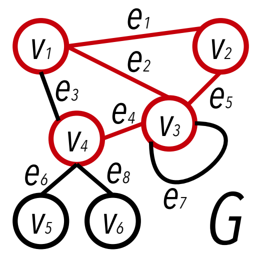


$(v_4, e_4, v_3, e_5, v_2, e_1, v_1, e_2, v_3, e_4, v_4)$

Open / Closed Walk

Open walk: first and last nodes are **not the same**

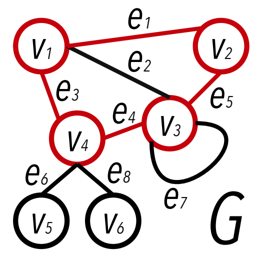
$v_0 \neq v_k$



$(v_4, e_4, v_3, e_5, v_2, e_1, v_1, e_2, v_3)$

Closed walk: first and last nodes are **the same**

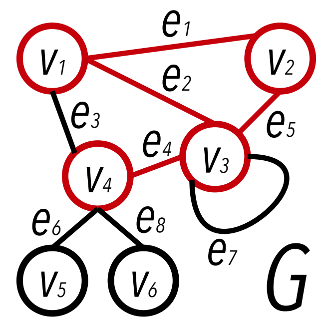
$v_0 = v_k$



$(v_4, e_4, v_3, e_5, v_2, e_1, v_1, e_3, v_4)$

Trail

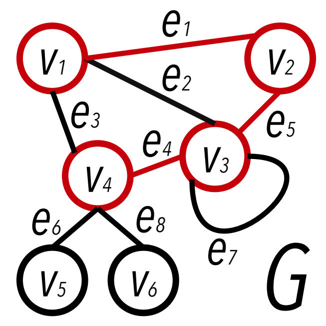
Walk without repeated edges



$(v_4, e_4, v_3, e_5, v_2, e_1, v_1, e_2, v_3)$

Path

Open trail with no repeated nodes.

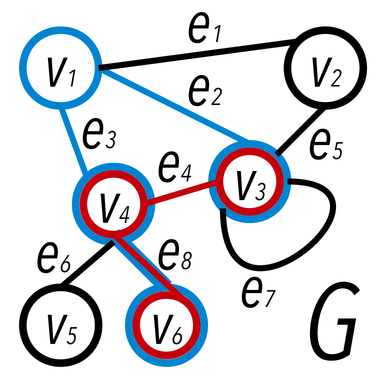


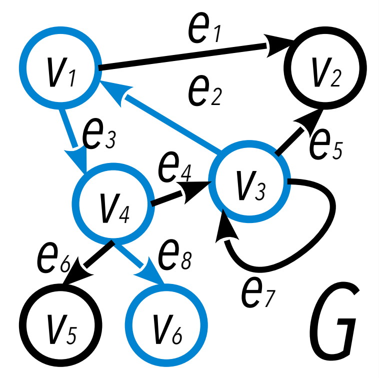
$$(v_4, e_4, v_3, e_5, v_2, e_1, v_1)$$

Distance

Length of the **Geodesic/Shortest Path** between 2 nodes

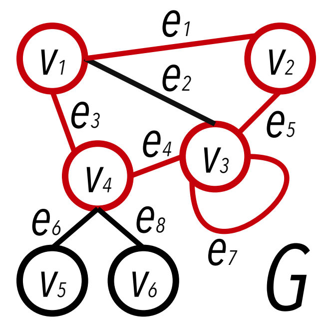
- Distance between disconnected nodes $\rightarrow \infty$
- Distances in undirect. and direct. graphs may differ.





Circuit

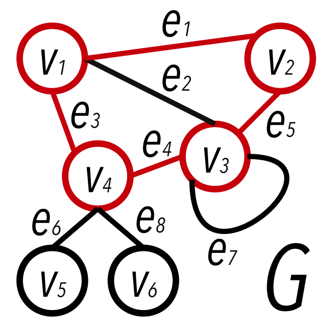
Closed trail possibly with **repeated nodes**



$(v_4, e_4, v_3, e_7, v_3, e_5, v_2, e_1, v_1, e_3, v_4)$

Cycle

Closed trail with **no repeated nodes besides** the **first** and the **last** ones.

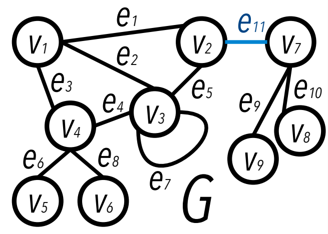


$$(v_4, e_4, v_3, e_5, v_2, e_1, v_1, e_3, v_4)$$

Connectivity

Let $G = \langle V, E \rangle$

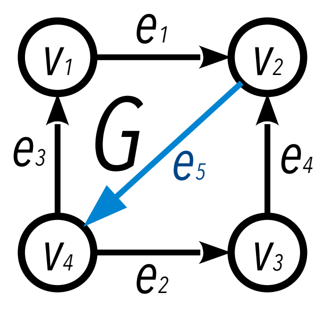
- $v_i \in V$ and $v_j \in V$ are **connected** if:
  $\exists$ a path $(v_i, \ldots, v_j)$ in G.
- G is **connected** if:
  $\forall v_i \in V$ and $\forall v_j \in V$, v_i and v_j are **connected.**
- The graph is in **one piece**.



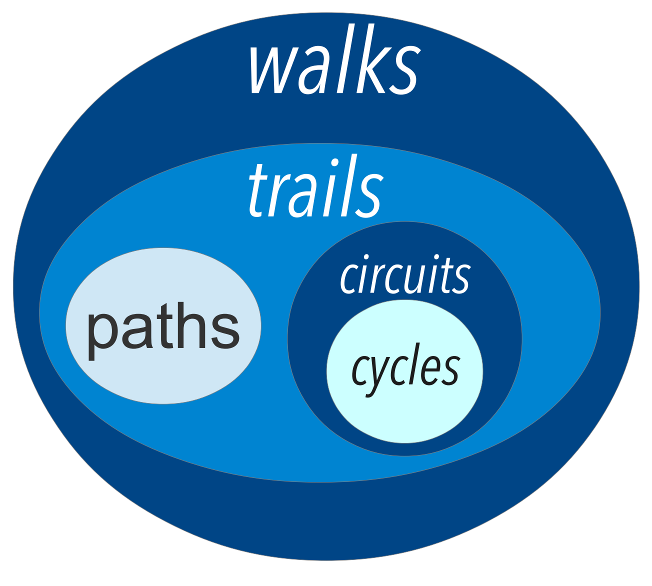
Connectivity

Let $G = \langle V, E \rangle$ a **directed graph**

- G is **strongly connected** if $\forall v_i \in V$ and $\forall v_j \in V$: $\exists$ path $(v_i, \dots, v_j)$ **and** $\exists$ path $(v_j, \dots, v_i)$.
- G is **weakly connected** if: $\forall v_i \in V$ and $\forall v_j \in V$ $\exists$ path $(v_i, \dots, v_j)$ **or** $\exists$ path $(v_j, \dots, v_i)$.



Summary

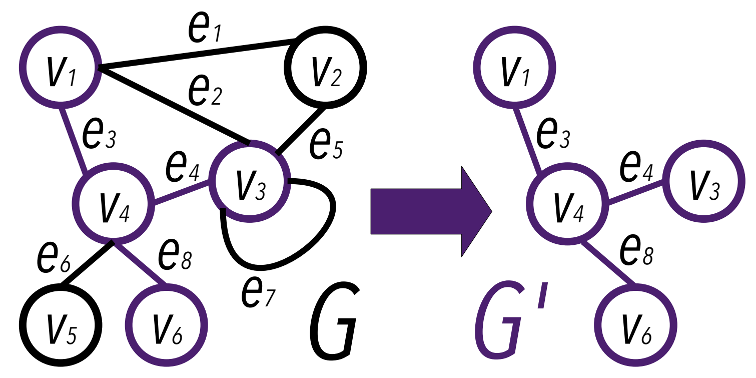


Subgraphs: the Graphs Inside

Subgraph

$G' = \langle V', E' \rangle$ is a **subgraph** of $G = \langle V, E \rangle$ if:

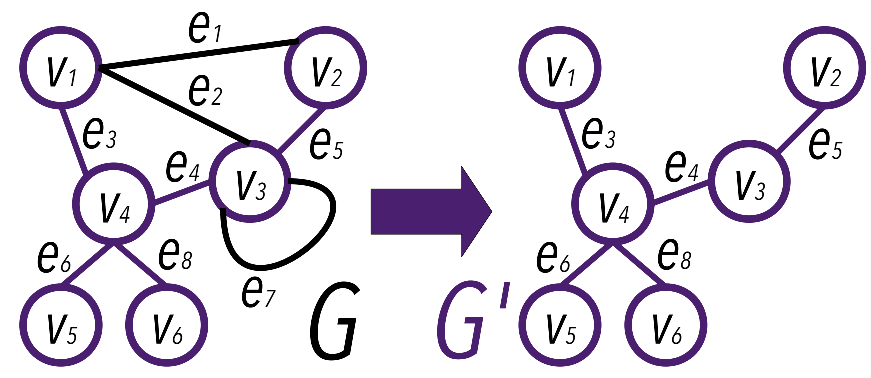
$V' \subseteq V$ and $E' \subseteq E$.



Spanning Subgraph

$G' = \langle V', E' \rangle$ is a **spanning subgraph** of $G = \langle V, E \rangle$ if:

G' is **connected** and $V' = V$.

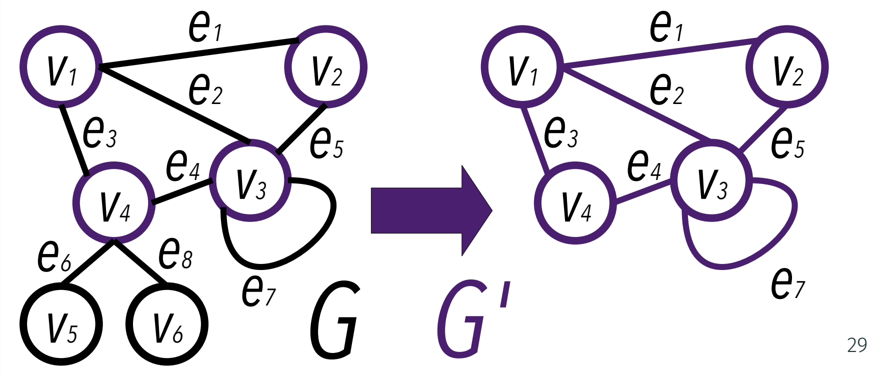


Induced Subgraph

$G' = \langle V', E' \rangle$ is a **induced subgraph** of $G = \langle V, E \rangle$ if:

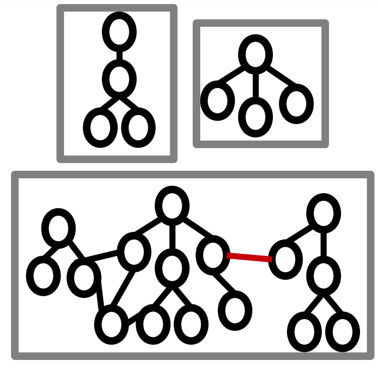
$V' \subseteq V$, and $E' = E|_{V'} = \{\langle v_i, v_j \rangle \in E | v_i \in V' \text{ and } v_j \in V'\}$

$E|_{V'}$: edges in E that have both endpoints in V'



Connected Components | Undirected graphs

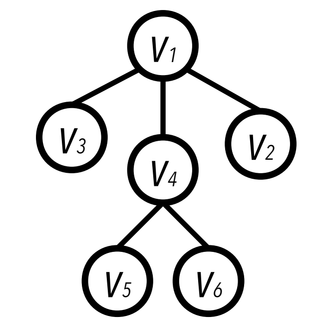
- **Connected subgraph** G' that is **not connected** to **any additional** node in the original graph $G = \langle V, E \rangle$.
- **Bridge**: edge e s.t. $G' = \langle V, E \setminus \{e\} \rangle$ is **disconnected**, while G was not.



Trees and Forests

Equivalent Definitions

- **Acyclic** and **connected graph** $T = \langle V, E \rangle$.
- Any **two nodes** are connected by exactly **one path**.
- T would be **disconnected** if a **single edge** is **removed**.
- **Cycle** formed if **any edge is added**.



Further Definitions

- **Internal/inner/branch** node:
  Node with **at least 2 neighbors**.
- **External/outer/leaf** node:
  Node with **1 neighbor**.

Claims

- **Claim 1:** A tree has **at least one leaf**
- **Claim 2:** A tree $T = \langle V, E \rangle$, if $|V| = n$ then $|E| = n - 1$
- **Proofs:** ?

Draft proof 1

- Pick a node v
- Follow one path starting at v
- v cannot be visited twice (acyclic)
- So the path eventually stops $\rightarrow$ leaf.

Draft proof 2

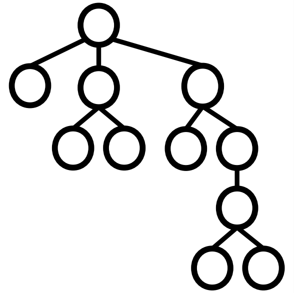
Let $T = \langle V, E \rangle$ be a tree with $|V| = n$

- Base case $n = 1$: True
- Let $n = k$ and assume that every tree with k nodes has $k - 1$ edges.
    - Let v_{k+1} denote a new node.
    - Pick a node $v \in V$ (connect).
    - Let $T' = \langle V \cup \{v_{k+1}\}, E \cup \{\langle v, v_{k+1} \rangle\} \rangle$.
    - T' is a tree with $k + 1$ nodes and k edges.
    - It is not possible to add more edges (otherwise cycles appear).

By induction on n, a tree with $|V| = n$ has $|E| = n - 1$

Rooted Tree

Tree with a distinguished node called ”root”.



Parents and Children

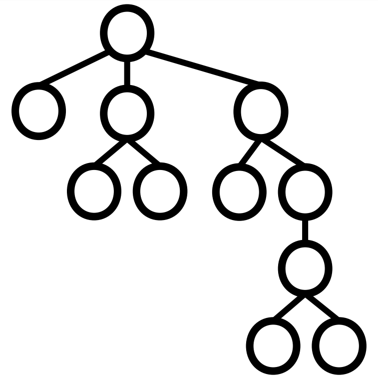
- **Parent** of v:
  - Node **connected to** v that is in the **path to the root**
  - v is its **child**.
- Each node has a **unique parent** (except the root)
- **Descendant** of v: **children** of v or **descendant** of any **children** of v (recursive).
- **Sibling** of v: Node **sharing the same parent** of v.
- **Root with 1 child**: **external node**
- k-ary tree:
  Rooted tree s.t. **each** node has **at most** k **children**.

Height and Depth

- **Height** of a **node** v:
  Length of the **longest path** from v to a **leaf**.
- **Height** of a **tree**: **Height** of the **root**.
- **Depth/level** of v: **Length of the path** to its **root**.
- $depth(root) = 0$, $height(leaf) = 0$
- **Empty tree** has **depth** and **height** -1

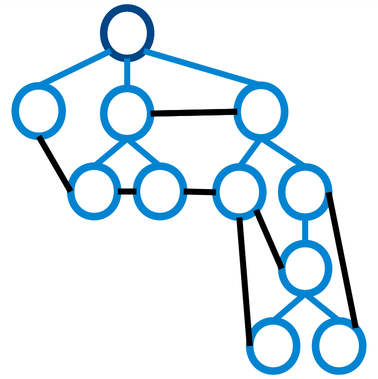
Example

Exemplify each one of the previous **concepts**.



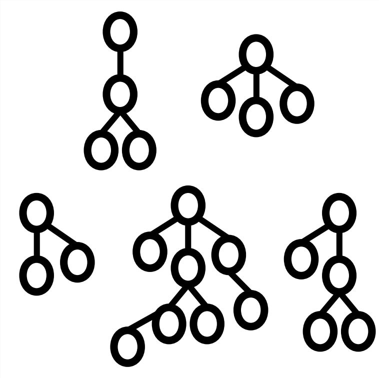
Spanning Tree

Spanning subgraph that is a **tree**.



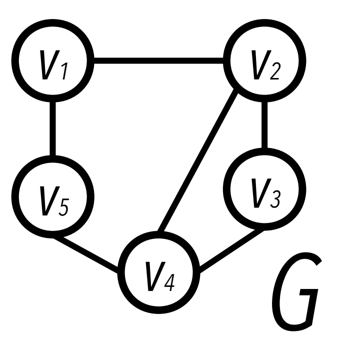
Forest

- Forest: Disjoint union of trees
- Forest: Acyclic graph not necessarily connected.



Representing Graphs

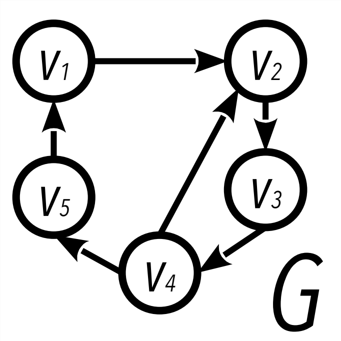
Adjacency Matrix



$$\begin{bmatrix} & v_1 & v_2 & v_3 & v_4 & v_5 \\ v_1 & 0 & 1 & 0 & 0 & 1 \\ v_2 & 1 & 0 & 1 & 1 & 0 \\ v_3 & 0 & 1 & 0 & 1 & 0 \\ v_4 & 0 & 1 & 1 & 0 & 1 \\ v_5 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

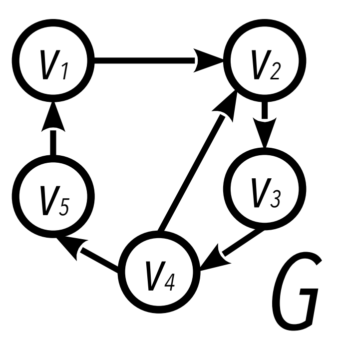
- Matrix A, $|V| \times |V|$: $A_{i,j} = 1$ if $\langle i, j \rangle \in E$, 0 otherwise.
- **Advantage**: Direct access,
- **Disadvantage**: Memory hungry.

Adjacency Matrix | Exercise



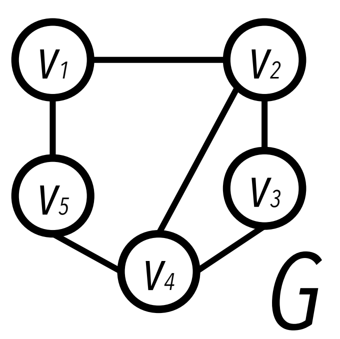
$$\begin{array}{cc} & \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} & \begin{bmatrix} ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \end{bmatrix} \end{array}$$

Adjacency Matrix | Exercise



$$
\begin{array}{c|ccccc}
 & v_1 & v_2 & v_3 & v_4 & v_5 \\
\hline
v_1 & 0 & 1 & 0 & 0 & 0 \\
v_2 & 0 & 0 & 1 & 0 & 0 \\
v_3 & 0 & 0 & 0 & 1 & 0 \\
v_4 & 0 & 1 & 0 & 0 & 1 \\
v_5 & 1 & 0 & 0 & 0 & 0
\end{array}
$$

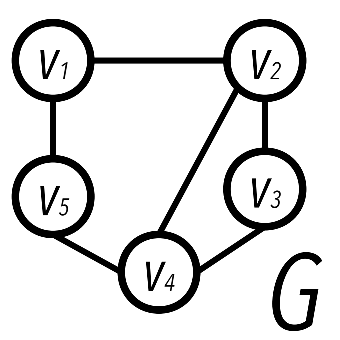
Edge list



$$\begin{bmatrix} v_1 & v_2 \\ v_2 & v_3 \\ v_3 & v_4 \\ v_4 & v_5 \\ v_5 & v_1 \\ v_2 & v_4 \end{bmatrix}$$

- **Advantage**: Memory friendly,
- **Disadvantage**: Slow access.

Adjacency/Neighbour list



$V_1, * \rightarrow V_2, * \rightarrow V_5, /$

$V_2, * \rightarrow V_1, * \rightarrow V_3, * \rightarrow V_4, /$

$V_3, * \rightarrow V_2, * \rightarrow V_4, /$

$V_4, * \rightarrow V_2, * \rightarrow V_5, * \rightarrow V_3, /$

$V_5, * \rightarrow V_1, * \rightarrow V_4, /$

- Memory optimized.
- Fast search.
- Out nodes for directed graphs.

Time/Space Complexity (a Detour)

- **Time Complexity**:
  Amount of time to run the algorithm
    - **Counting** nb. of **elementary operations**.
    - Hypothesis:
      elementary operations take a **constant time**.
    - **Function** of the **size** of the **input**.

- **Space Complexity**:
  Amount of memory needed to run the algorithm
    - **Counting** nb. of **elementary memory allocations**.
    - Hypothesis: **elementary allocations** are **constant**.
    - **Function** of the **size** of the **input**.

Time/Space Complexity Notation

Big O notation $\mathcal{O}(n)$:

Asymptotic behavior of the **complexity** when the **input size** n **increases**.

Examples:

Complexity	Big O
Constant	$\mathcal{O}(1)$
Logarithmic	$\mathcal{O}(log\ n)$
Linear	$\mathcal{O}(n)$
Log-linear	$\mathcal{O}(n\ log\ n)$
Quadratic	$\mathcal{O}(n^2)$
Polynomial	$\mathcal{O}(n^k)$
Exponential	$\mathcal{O}(k^n)$

Complexity relations

Let two $f(n)$ and $g(n)$ be two **functions**.

$f(n) = \mathcal{O}(g(n))$ if:

$\exists n_0,$ and $\exists c$ a constant $s.t. : \forall n \geq n_0 : f(n) \leq c \times g(n)$

Examples:

Let $f : n \rightarrow n$

$f(n) = \mathcal{O}(n), \quad f(n) = \mathcal{O}(n^2), \quad f(n) \neq \mathcal{O}(log\ n),$

$f(n) \neq \mathcal{O}(1), \quad f(n) = \mathcal{O}(k^n)$

Exercises

Exercise 1

Let $G = \langle V, E \rangle$ be a graph, with an adjacency matrix A.

Let $N^{(k)}$ be a $|V| \times |V|$ matrix, s.t., $N^{(k)}_{v_i,v_j}$ is the number of walks of length k between v_i and v_j. Show that:

$$N^{(k)}_{v_i,v_j} = [A^k]_{v_i,v_j}$$

Exercice 3

How does the adjacency matrix of a network with several components look like?

Exercise 4

Give $G = \langle V, E \rangle$ represented by and adjacency matrix A, write an algorithm that outputs the adjacency list representation, and viceversa.

Exercise 5

Show that for any connected graph $G = \langle V, E \rangle$, $\exists v \in V$ s.t. if we remove it, the graph remains connected.