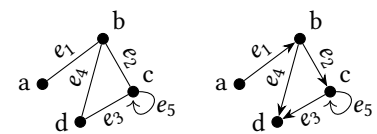
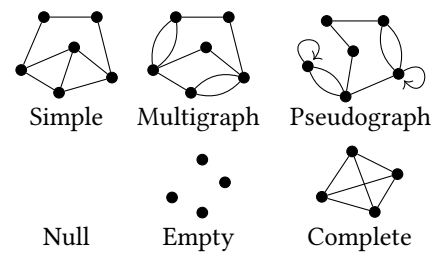


## 5 Graph Theory Cheatsheet

[Glossary](#)

- \* **Graph** is an ordered pair  $G = \langle V, E \rangle$ , where  $V = \{v_1, \dots, v_n\}$  is a set of vertices, and  $E = \{e_1, \dots, e_m\}$  is a set of edges.
  - Given a graph  $G$ , the notation  $V(G)$  denotes the vertices of  $G$ .
  - Given a graph  $G$ , the notation  $E(G)$  denotes the edges of  $G$ .
  - In fact,  $V(\cdot)$  and  $E(\cdot)$  functions allow to access “vertices” and “edges” of any object possessing them (e.g., paths).
- \* **Order** of a graph  $G$  is the number of vertices in it:  $|V(G)|$ .
- \* **Size** of a graph  $G$  is the number of edges in it:  $|E(G)|$ .
- \* Simple **undirected** graphs have  $E \subseteq V^{(2)}$ , i.e. each edge  $e_i \in E$  between vertices  $u$  and  $v$  is denoted by  $\{u, v\} \in V^{(2)}$ . Such *undirected edges* are also called *links* or *lines*.
  - $A^{(k)} = \{\{x_1, \dots, x_k\} \mid x_1 \neq \dots \neq x_k \in A\} = \{S \mid S \subseteq A, |S| = k\}$  is the set of  $k$ -sized subsets of  $A$ .
- \* Simple **directed** graphs have  $E \subseteq V^2$ , i.e. each edge  $e_i \in E$  from vertex  $u$  to  $v$  is denoted by an ordered pair  $\langle u, v \rangle \in V^2$ . Such *directed edges* are also called *arcs* or *arrows*.
  - $A^k = A \times \dots \times A = \{(x_1, \dots, x_k) \mid x_1, \dots, x_k \in A\}$  is the set of  $k$ -tuples (Cartesian  $k$ -power of  $A$ ).
- \* **Multi-edges** are edges that have the same end nodes.
- \* **Loop** is an edge that connects a vertex to itself.
- \* **Simple graph** is a graph without multi-edges and loops.
- \* **Multigraph** is a graph with multi-edges.
- \* **Pseudograph** is a multigraph with loops.
- \* **Null graph** is a “graph” without vertices.
- \* **Trivial (singleton) graph** is a graph consisting of a single vertex.
- \* **Empty (edgeless) graph** is a graph without edges.
- \* **Complete graph**  $K_n$  is a simple graph in which every pair of distinct vertices is connected by an edge.
- \* **Weighted graph**  $G = (V, E, w)$  is a graph in which each edge has an associated numerical value (the *weight*) represented by the **weight function**  $w : E \rightarrow \text{Num}$ .
- \* **Subgraph** of a graph  $G = \langle V, E \rangle$  is another graph  $G' = \langle V', E' \rangle$  such that  $V' \subseteq V, E' \subseteq E$ . Designated as  $G' \subseteq G$ .
- \* **Spanning (partial) subgraph** is a subgraph that includes all vertices of a graph.
- \* **Induces subgraph** of a graph  $G = \langle V, E \rangle$  is another graph  $G'$  formed from a subset  $S$  of the vertices of the graph and *all* the edges (from the original graph) connecting pairs of vertices in that subset. Formally,  $G' = G[S] = \langle V', E' \rangle$ , where  $S \subseteq V, V' = V \cap S, E' = \{e \in E \mid \exists v \in S : e \text{ I } v\}$ .
- \* **Adjacency** is the relation between two vertices connected with an edge.
- \* **Adjacency matrix** is a square matrix  $A_{V \times V}$  of an adjacency relation.
  - For simple graphs, adjacency matrix is binary, i.e.  $A_{ij} \in \{0, 1\}$ .
  - For directed graphs,  $A_{ij} \in \{0, 1, -1\}$ .
  - For multigraphs, adjacency matrix contains edge multiplicities, i.e.  $A_{ij} \in \mathbb{N}_0$ .
- \* **Incidence** is a relation between an edge and its endpoints.
- \* **Incidence matrix** is a Boolean matrix  $B_{V \times E}$  of an incidence relation.
- \* **Degree**  $\deg(v)$  the number of edges incident to  $v$  (loops are counted twice).
  - $\delta(G) = \min_{v \in V} \deg(v)$  is the **minimum degree**.
  - $\Delta(G) = \max_{v \in V} \deg(v)$  is the **maximum degree**.
  - HANDSHAKING LEMMA.  $\sum_{v \in V} \deg(v) = 2|E|$ .
- \* A graph is called  **$r$ -regular** if all its vertices have the same degree:  $\forall v \in V : \deg(v) = r$ .
- \* **Complement graph** of a graph  $G$  is a graph  $H$  on the same vertices such that two distinct vertices of  $H$  are adjacent iff they are non-adjacent in  $G$ .
- \* **Intersection graph** of a family of sets  $F = \{S_i\}$  is a graph  $G = \Omega(F) = \langle V, E \rangle$  such that each vertex  $v_i \in V$  denotes the set  $S_i$ , i.e.  $V = F$ , and the two vertices  $v_i$  and  $v_j$  are adjacent whenever the corresponding sets  $S_i$  and  $S_j$  have a non-empty intersection, i.e.  $E = \{\langle v_i, v_j \rangle \mid i \neq j, S_i \cap S_j \neq \emptyset\}$ .
- \* **Line graph** of a graph  $G = \langle V, E \rangle$  is another graph  $L(G) = \Omega(E)$  that represents the adjacencies between edges of  $G$ . Each vertex of  $L(G)$  represents an edge of  $G$ , and two vertices of  $L(G)$  are adjacent iff the corresponding edges share a common endpoint in  $G$  (i.e. edges are “adjacent”/“incident”).



Adjacency matrix:

$$A = \begin{bmatrix} a & b & c & d \\ a & 0 & 1 & 0 & 0 \\ b & 1 & 0 & 1 & 1 \\ c & 0 & 1 & 0 & 1 \\ d & 0 & 1 & 1 & 0 \end{bmatrix}$$

Incidence matrix:

$$B = \begin{bmatrix} e_1 & e_2 & e_3 & e_4 & e_5 \\ a & 0 & 0 & 0 & 0 \\ b & 1 & 1 & 0 & 1 \\ c & 0 & 1 & 1 & 0 \\ d & 0 & 0 & 1 & 1 \end{bmatrix}$$

- \* **Walk**<sup>1</sup> is an alternating sequence of vertices and edges:  $l = v_1 e_1 v_2 \dots e_{n-1} v_n$ .
  - **Trail** is a walk with distinct edges.
  - **Path** is a walk with distinct vertices (and therefore distinct edges).
  - A walk is **closed** if it starts and ends at the same vertex. Otherwise, it is **open**.
  - **Circuit** is a closed trail.
  - **Cycle** is a closed path.
- \* **Length** of a path (walk, trail)  $l = u \rightsquigarrow v$  is the number of edges in it:  $|l| = |E(l)|$ .
- \* **Girth**<sup>2</sup> is the length of the shortest cycle in the graph.
- \* **Distance**<sup>2</sup>  $\text{dist}(u, v)$  between two vertices is the length of the shortest path  $u \rightsquigarrow v$ .
  - $\varepsilon(v) = \max_{u \in V} \text{dist}(v, u)$  is the **eccentricity** of the vertex  $v$ .
  - $\text{rad}(G) = \min_{v \in V} \varepsilon(v)$  is the **radius** of the graph  $G$ .
  - $\text{diam}(G) = \max_{v \in V} \varepsilon(v)$  is the **diameter** of the graph  $G$ .
  - $\text{center}(G) = \{v \mid \varepsilon(v) = \text{rad}(G)\}$  is the **center** of the graph  $G$ .
- \* **Clique**<sup>2</sup>  $Q \subseteq V$  is a set of vertices inducing a complete subgraph.
- \* **Stable set**<sup>2</sup>  $S \subseteq V$  is a set of independent (pairwise non-adjacent) vertices.

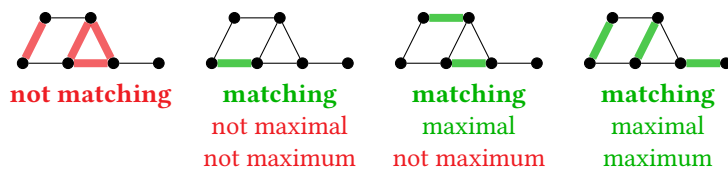
| Term  | V <sup>1</sup> | E <sup>2</sup> | "Closed" term |
|-------|----------------|----------------|---------------|
| Walk  | +              | +              | Closed walk   |
| Trail | +              | -              | Circuit       |
| Path  | -              | -              | Cycle         |
|       | -              | +              | (impossible)  |

<sup>1</sup>Can vertices be repeated?

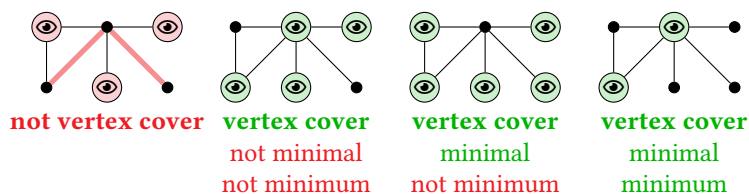
<sup>2</sup>Can edges be repeated?



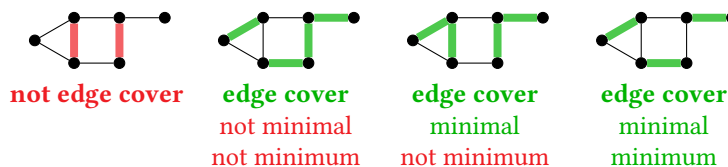
- \* **Matching**<sup>2</sup>  $M \subseteq E$  is a set of independent (pairwise non-adjacent) edges.



- \* **Perfect matching**<sup>2</sup> is a matching that covers all vertices in the graph.
  - A perfect matching (if it exists) is always a minimum edge cover (*but not vice-versa!*).
- \* **Vertex cover**<sup>2</sup>  $R \subseteq V$  is a set of vertices "covering" all edges.



- \* **Edge cover**<sup>2</sup>  $F \subseteq E$  is a set of edges "covering" all vertices.



- \* **Cut vertex (articulation point)** is a vertex whose removal increases the number of connected components.
- \* **Bridge** is an edge whose removal increases the number of connected components.
- \* **Biconnected graph** is a connected “nonseparable” graph, which means that the removal of any vertex does not make the graph disconnected. Alternatively, this is a graph without *cut vertices*.
- \* **Biconnectivity** can be defined as a relation on edges  $R \subseteq E^2$ :
  - Two edges are called *biconnected* if there exist two *vertex-disjoint* paths between the ends of these edges.
  - Trivially, this relation is an equivalence relation.
  - Equivalence classes of this relation are called **biconnected components**, also known as **blocks**.
- \* **Edge biconnectivity** can be defined as a relation on vertices  $R \subseteq V^2$ :
  - Two vertices are called *edge-biconnected* if there exist two *edge-disjoint* paths between them.
  - Trivially, this relation is an equivalence relation.
  - Equivalence classes of this relation are called **edge-biconnected components** (or *2-edge-connected components*).
- \* **Vertex connectivity**  $\kappa(G)$  is the minimum number of vertices that has to be removed in order to make the graph disconnected or trivial (singleton). Equivalently, it is the largest  $k$  for which the graph  $G$  is  $k$ -vertex-connected.
- \*  **$k$ -vertex-connected graph** is a graph that remains connected after less than  $k$  vertices are removed, i.e.  $\kappa(G) \geq k$ .
  - Corollary of Menger’s theorem: graph  $G = \langle V, E \rangle$  is  $k$ -vertex-connected if, for every pair of vertices  $u, v \in V$ , it is possible to find  $k$  *vertex-independent* (*internally vertex-disjoint*) paths between  $u$  and  $v$ .
  - $k$ -vertex-connected graphs are also called simply  *$k$ -connected*.
  - 1-connected graphs are called *connected*, 2-connected are *biconnected*, 3-connected are *triconnected*, etc.
  - Note the “exceptions”:
    - Singleton graph  $K_1$  has  $\kappa(K_1) = 0$ , so it is **not** *1-connected*, but still considered *connected*.
    - Graph  $K_2$  has  $\kappa(K_2) = 1$ , so it is **not** *2-connected*, but considered *biconnected*, so it can be a block.
- \* **Edge connectivity**  $\lambda(G)$  is the minimum number of edges that has to be removed in order to make the graph disconnected or trivial (singleton). Equivalently, it is the largest  $k$  for which the graph  $G$  is  $k$ -edge-connected.
- \*  **$k$ -edge-connected graph** is a graph that remains connected after less than  $k$  edges are removed, i.e.  $\lambda(G) \geq k$ .
  - Corollary of Menger’s theorem: graph  $G = \langle V, E \rangle$  is  $k$ -edge-connected if, for every pair of vertices  $u, v \in V$ , it is possible to find  $k$  *edge-disjoint* paths between  $u$  and  $v$ .
  - 2-edge-connected are called *edge-biconnected*, 3-edge-connected are *edge-triconnected*, etc.
  - Note the “exception”:
    - Singleton graph  $K_1$  has  $\lambda(K_1) = 0$ , so it is **not** *2-edge-connected*, but considered *edge-biconnected*, so it can be a *2-edge-connected component*.
- \* WHITNEY’S THEOREM. For any graph  $G$ ,  $\kappa(G) \leq \lambda(G) \leq \delta(G)$ .



$$\kappa(G) = 2, \lambda(G) = 3, \delta(G) = 3, \Delta(G) = 6$$

- \* **Tree** is a connected undirected acyclic graph.
- \* **Forest** is an undirected acyclic graph, *i.e.* a disjoint union of trees.
- \* An **unrooted tree (free tree)** is a tree without any designated *root*.
- \* A **rooted tree** is a tree in which one vertex has been designated the *root*.
  - In a rooted tree, the **parent** of a vertex  $v$  is the vertex connected to  $v$  on the path to the root.
  - A **child** of a vertex  $v$  is a vertex of which  $v$  is the parent.
  - A **sibling** to a vertex  $v$  is any other vertex on the tree which has the same parent as  $v$ .
  - A **leaf** is a vertex with no children. Equivalently, **leaf** is a *pendant vertex*, *i.e.*  $\deg(v) = 1$ .
  - An **internal vertex** is a vertex that is not a leaf.
  - A  **$k$ -ary tree** is a rooted tree in which each vertex has at most  $k$  children. *2-ary trees* are called **binary trees**.
- \* A **labeled tree** is a tree in which each vertex is given a unique *label*, *e.g.*,  $1, 2, \dots, n$ .
- \* **CAYLEY'S FORMULA**. Number of labeled trees on  $n$  vertices is  $n^{n-2}$ .
- \* **Prüfer code** is a unique sequence of labels  $\{1, \dots, n\}$  of length  $(n - 2)$  associated with the labeled tree on  $n$  vertices.
  - **ENCODING** (iterative algorithm for converting tree  $T$  labeled with  $\{1, \dots, n\}$  into a Prüfer sequence  $K$ ):
    - On each iteration, remove the leaf with *the smallest label*, and extend  $K$  with *a single neighbour* of this leaf.
    - After  $(n - 2)$  iterations, the tree will be left with *two adjacent* vertices — there is no need to encode them, because there is only one unique tree on 2 vertices, which requires 0 bits of information to encode.



- **DECODING** (iterative algorithm for converting a Prüfer sequence  $K$  into a tree  $T$ ):
  - Given a Prüfer code  $K$  of length  $(n - 2)$ , construct a set of “leaves”  $W = \{1, \dots, n\} \setminus K$ .
  - On each iteration:
    - (1) Pop the *first* element of  $K$  (denote it as  $k$ ) and the *minimum* label in  $W$  (denote it as  $w$ ).
    - (2) Connect  $k$  and  $w$  with an edge  $\langle k, w \rangle$  in the tree  $T$ .
    - (3) If  $k \notin K$ , then extend the set of “leaves”  $W := W \cup \{k\}$ .
  - After  $(n - 2)$  iterations, the sequence  $K$  will be empty, and the set  $W$  will contain exactly two vertices — connect them with an edge.