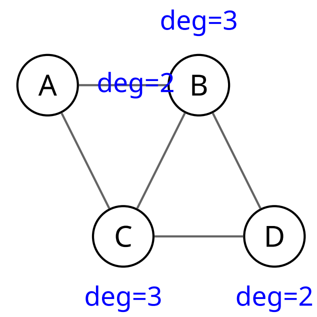
Graph Theory Fundamentals

Contents

1 Basic Concepts

1.1 Degree

The **degree** of a node is the number of edges connected to it. Node i's degree is written as $\deg(i)$.



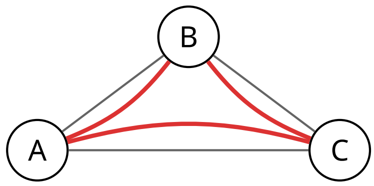
1.2 Walk, Trail, Path, and Cycle

Walk A list of edges sequentially connected to form a continuous route.

Trail A walk that doesn't go through any edge more than once.

Path A walk that doesn't go through any node (and therefore any edge) more than once.

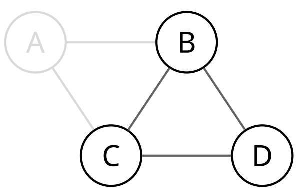
Cycle A walk that starts and ends at the same node without going through any node more than once on its way.



A **Cycle**: $A \to B \to C \to A$. Also a Path (until it closes) and a Trail.

1.3 Subgraph

A **subgraph** is simply a part of the original graph (a subset of nodes and edges).

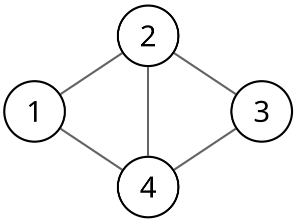


The solid black part (Nodes B, C, D and their connecting edges) is a **subgraph** of the larger graph containing A.

2 Connectivity & Components

2.1 Connected Graph

A graph is **connected** if a path exists between any pair of nodes. You can travel from any node to any other node.



Connected: No isolated parts.

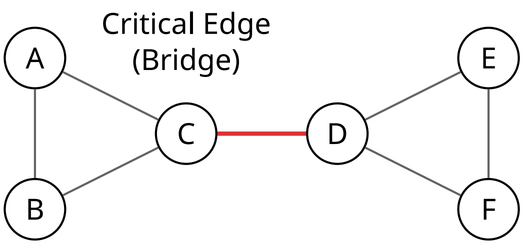
2.2 Connected Components: A Step-by-Step Breakdown

A **connected component** is a maximal subgraph that is connected within itself. A graph that is not connected is made up of multiple connected components.

To visualize this, let's see how a single connected graph breaks into components.

Step 1: The Initial Connected Graph

Here is a single, large graph. It is currently **one connected component** because Nodes A, B, and C can reach Nodes D, E, and F via the central "bridge" (Edge C-D).

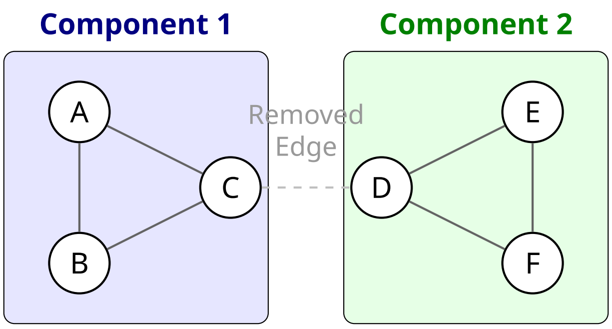


Step 2: Removing the Edge

We now remove the edge connecting **Node C** and **Node D**. Once this bridge is removed, there is no longer a path from the left side to the right side.

Step 3: The Resulting Components

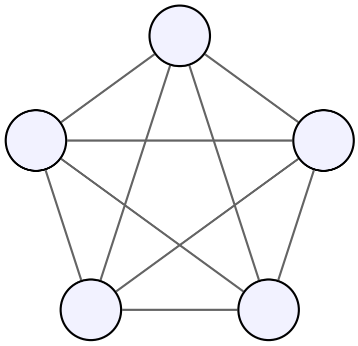
The graph is now **disconnected**. It has split into two separate connected components.



3 Structural Graph Types

3.1 Complete Graph

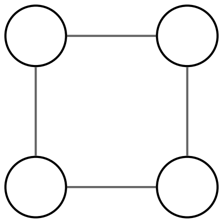
A graph in which any pair of nodes are connected. In a complete graph with n nodes (K_n), every node is connected to every other node.



Complete Graph (K_5)

3.2 Regular Graph

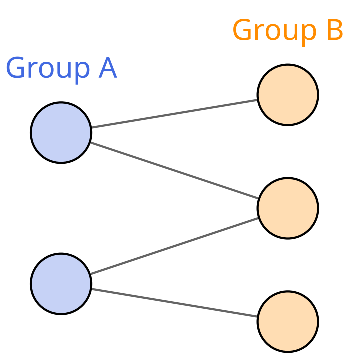
A graph in which all nodes have the same degree. Every complete graph is regular, but not every regular graph is complete.



2-Regular Graph (Cycle)
Every node has exactly degree 2.

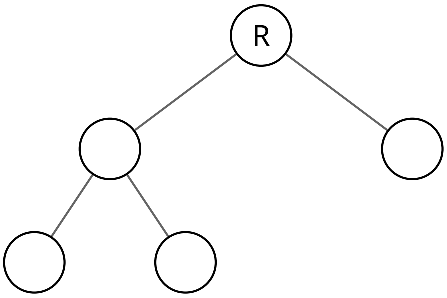
3.3 Bipartite Graph

A graph whose nodes can be divided into two disjoint sets (groups) such that every edge connects a node in one set to a node in the other. No edges exist within the same set.



3.4 Tree Graph

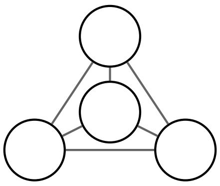
A connected graph with no cycles. A **Forest** is a collection of unconnected trees. Every tree is bipartite and planar.



Tree structure: No loops (cycles).

3.5 Planar Graph

A graph that can be drawn on a 2D plane such that no two edges cross each other.



K_4 drawn planar (no crossing lines).

4 Edge Properties

4.1 Undirected vs. Directed

- **Undirected Edge:** Symmetric. If A connects to B, B connects to A. The adjacency matrix is symmetric.
- **Directed Edge:** Asymmetric. A connection from A to B does not imply B to A.

Undirected Directed ($i \to j$)

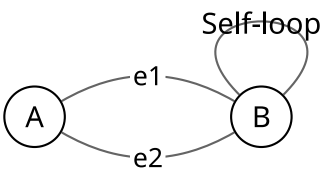
4.2 Unweighted vs. Weighted

- **Unweighted:** Edges represent binary existence (0 or 1).
- **Weighted:** Edges carry a value (weight, distance, cost). Node strength is the sum of weights.

Unweighted Weighted

4.3 Special Connections

- **Multiple Edges:** Edges sharing the same origin and destination.
- **Self-loop:** An edge that starts and ends at the same node.



Multiple Edges between A and B

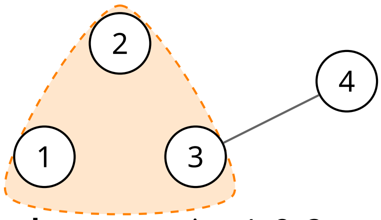
5 Graph Classifications

5.1 Definitions

Simple Graph A graph that contains NO directed, weighted, or multiple edges, and NO self-loops.

Multigraph A graph that allows multiple edges (and often self-loops).

Hypergraph A graph made of **Hyperedges**, which can connect any number of nodes (not just two) at once.



Hyperedge connecting 1, 2, 3

5.2 Standard Coding Definitions (NetworkX Style)

These terms are often used in computational graph libraries:

Graph Undirected simple graphs. (Self-loops are allowed).

DiGraph Directed simple graphs. (Self-loops are allowed).

MultiGraph Undirected multigraphs. (Self-loops and multiple edges allowed).

MultiDiGraph Directed multigraphs. (Self-loops and multiple edges allowed).