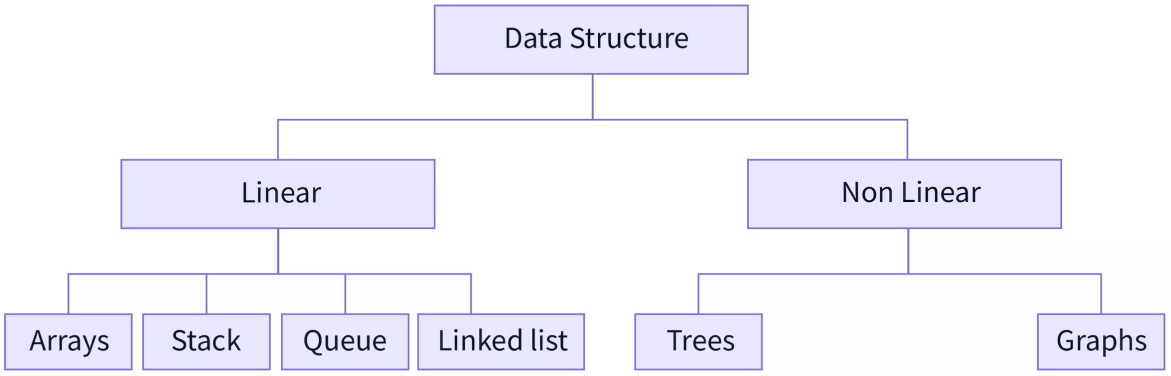
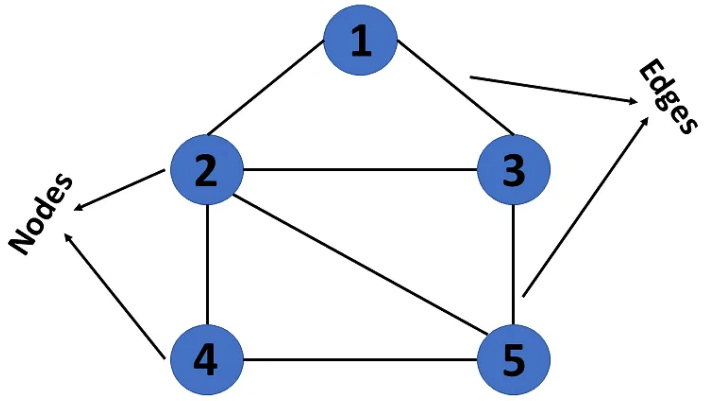
Graph



A graph (abstract data type) is a non-linear kind of data structure made of nodes or vertices and edges. The edges connect any two nodes in the graph, and the nodes are also known as vertices.



This graph has a set of vertices V= { 1,2,3,4,5} and a set of edges E= { (1,2),(1,3),(2,3),(2,4),(2,5),(3,5),(4,5} and Graph = {V, E}.

→ **Adjacency**: A vertex is said to be adjacent to another vertex if there is an edge connecting them. Vertices 2 and 3 are not adjacent because there is no edge between them.

→ **Path**: A sequence of edges that allows you to go from vertex A to vertex B is called a path.

→ **Cycle:** A cycle is a path that starts and ends at the same vertex. A simple cycle does not contain any repeated vertices or edges.

→ **Weight:** A weight can be assigned to an edge, representing the cost or distance between two vertices.

When to use Graphs

→ When you need to represent and analyze the relationships between different objects or entities.

→ When you need to perform network analysis.

→ When you need to identify key players, influencers or bottlenecks in a system.

→ When you need to make predictions or recommendations.

Representations of Graph

There are 2 representations of graph which are **Adjacency Matrix** and **Adjacency List**.

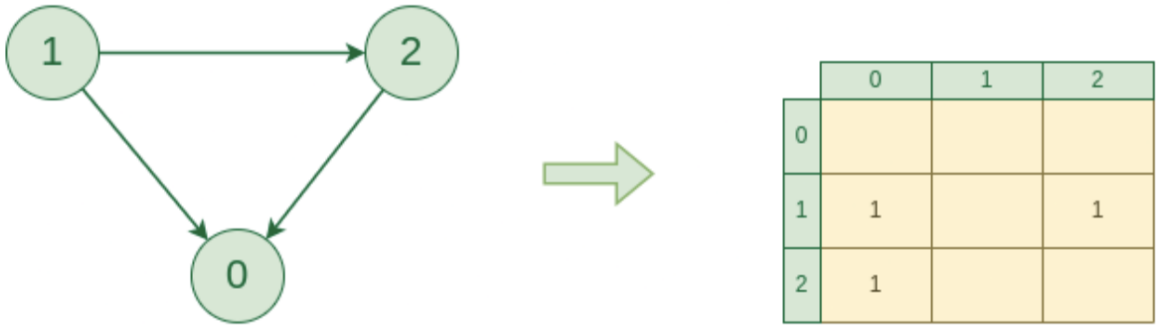
1) Adjacency Matrix

→ An adjacency matrix is a way of representing a graph as a matrix of boolean (0's and 1's).

→ Let's assume there are **n** vertices in the graph, 2D matrix **adjMat[n][n]** having dimension n x n will be created.

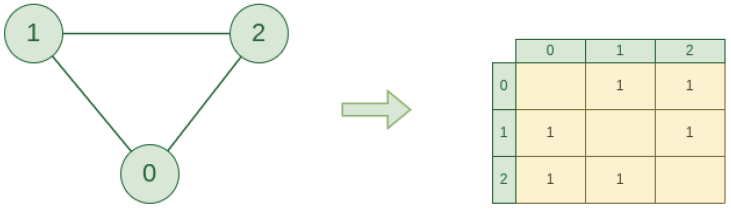
→ If there is an edge from vertex **i** to **j**, mark **adjMat[i][j]** as **1**. If there is no edge from vertex **i** to **j**, mark **adjMat[i][j]** as **0**.

→ When we are creating matrix for directed graph, the direction arrowhead shows is important.



Directed Graph

Adjacency Matrix



Undirected Graph

Adjacency Matrix

2) Adjacency List

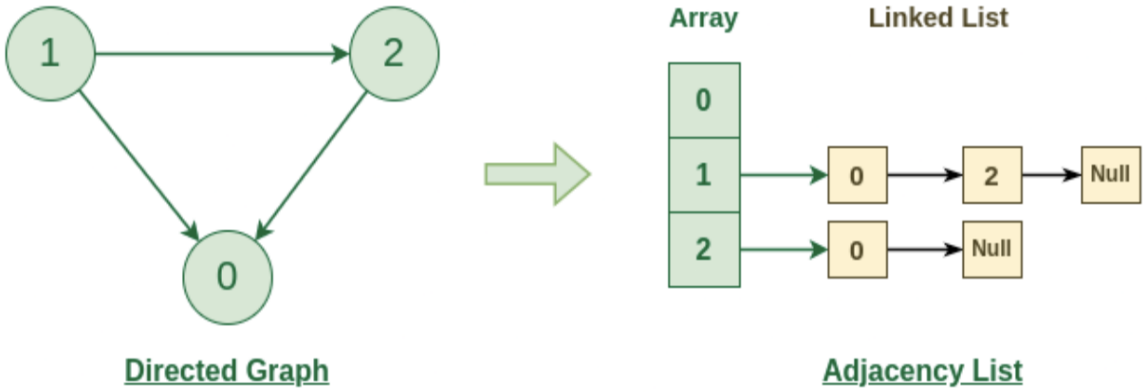
→ An array of Lists is used to store edges between two vertices.

→ The size of array is equal to the number of **vertices (i.e, n)**. Each index in this array represents a specific vertex in the graph. The entry at the index i of the array contains a linked list containing the vertices that are adjacent to vertex **i**.

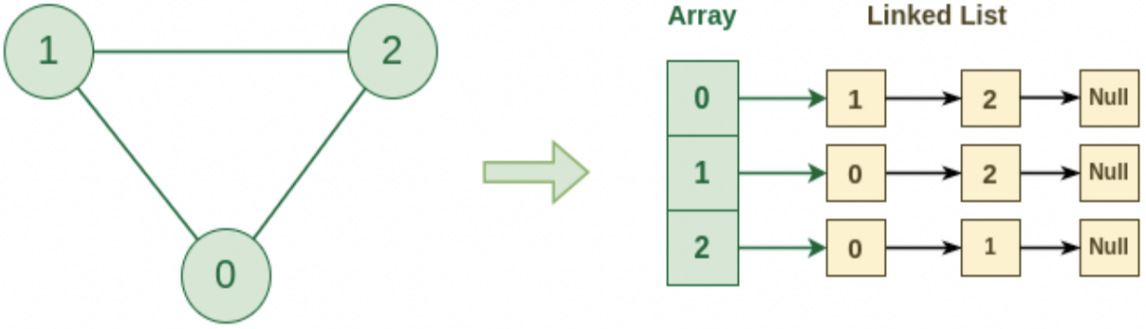
→ Let’s assume there are **n** vertices in the graph So, create an **array of list** of size **n** as **adjList[n].**

→ When we are creating list for directed graph, the direction arrowhead shows is important.

→ An adjacency list is efficient in terms of storage because we only need to store the values for the edges. For a graph with millions of vertices, this can mean a lot of saved space.



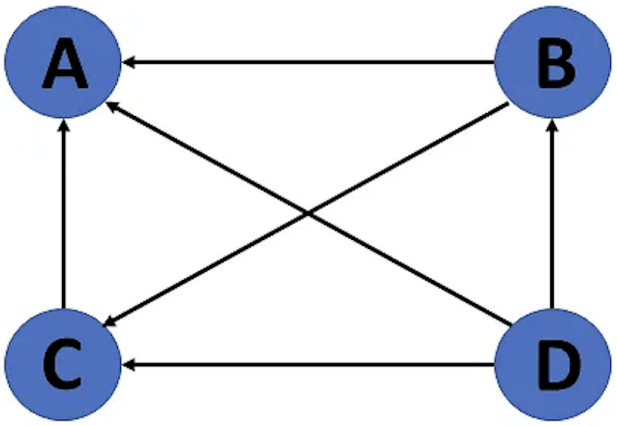
Directed Graph Adjacency List



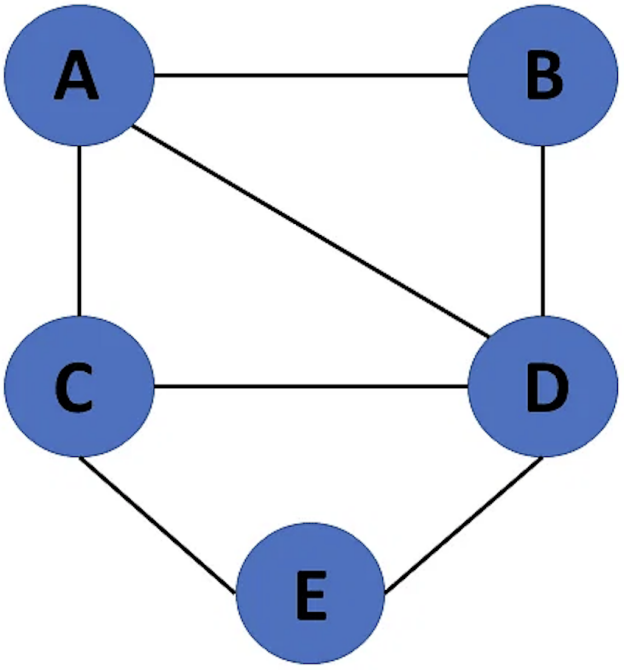
Undirected Graph Adjacency List

Types of Graph

→ **Directed Graph:** A directed graph also referred to as a digraph, is a set of nodes connected by edges, each with a direction.



→ **Undirected Graph:** An undirected graph comprises a set of nodes and links connecting them. The order of the two connected vertices is irrelevant and has no direction. You can form an undirected graph with a finite number of vertices and edges.

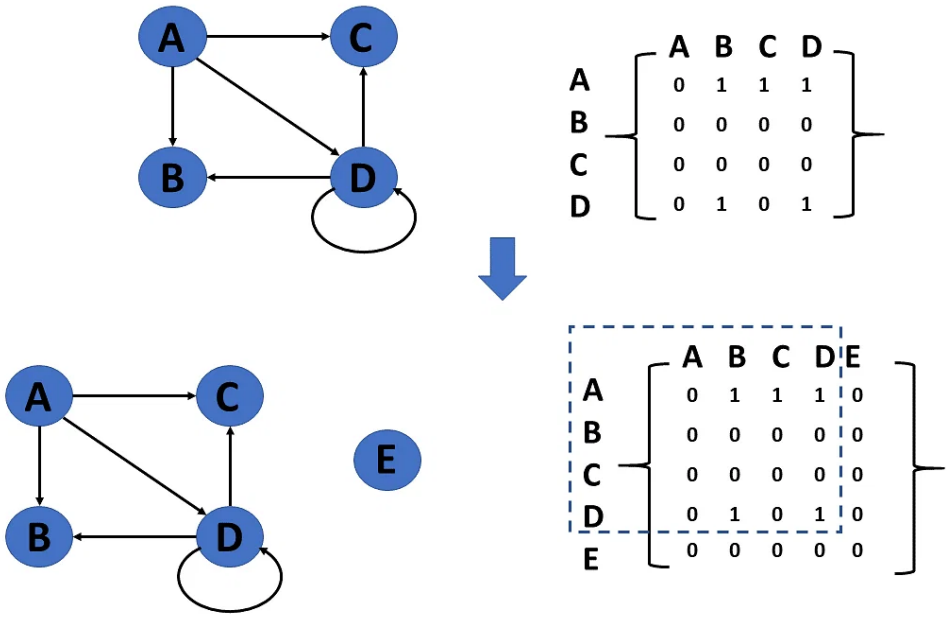


📌 `/Other Types of Graph :` There are also other types of graph that you may want to learn, you can find them at the link below.

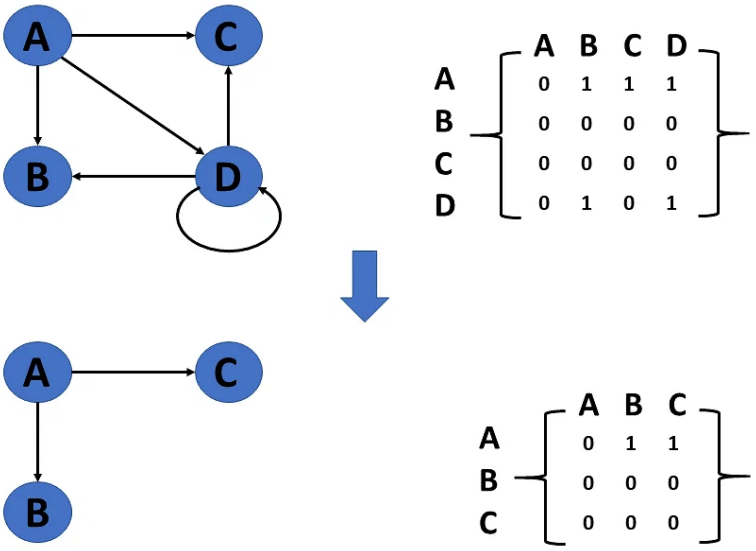
> What is Graph in Data Structure & Types of Graph?
>
> Explore what is graph in data structure, its types, terminologies, representation and operations. Read on to know how to implement code in graph data structure.
>
> https://www.simplilearn.com/tutorials/data-structure-tutorial/graphs-in-data-structure

Operations of Graph

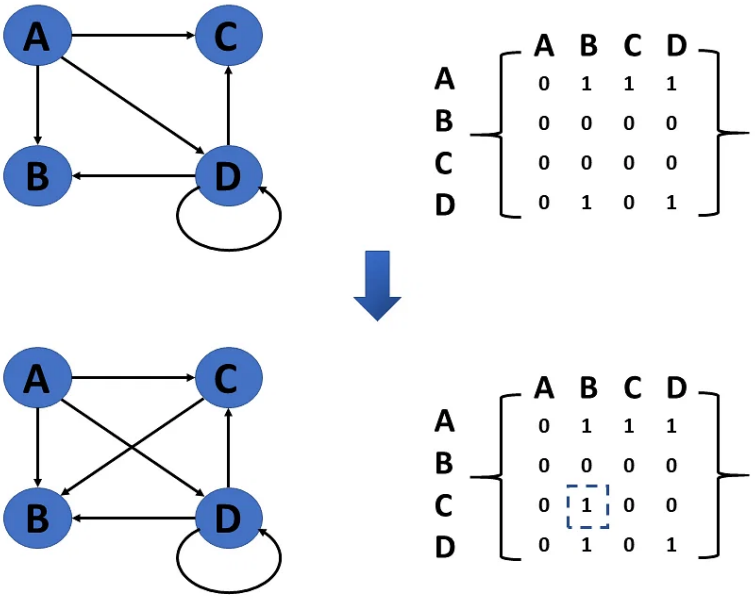
1) Add Vertex : We add a vertex(node) in a graph. When you add a vertex that after introducing one or more vertices or nodes, the graph's size grows by one, increasing the matrix's size by one at the row and column levels.



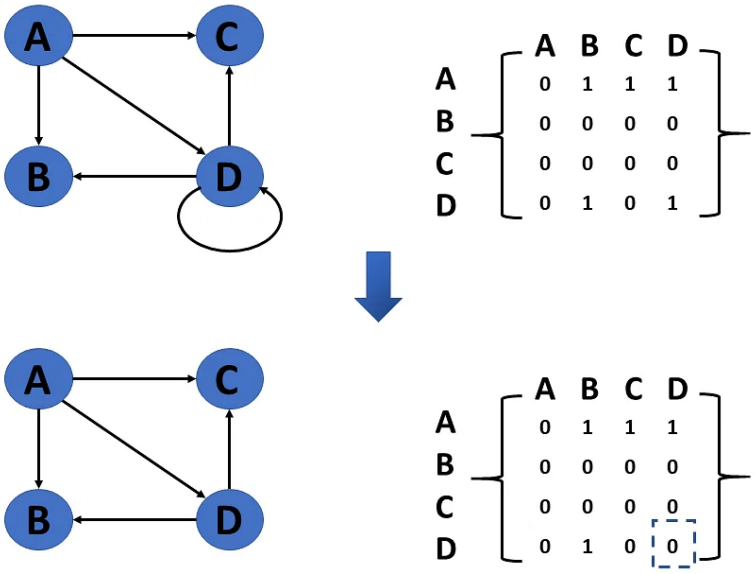
2) Remove Vertex : We remove a vertex from the graph. But, before removing the vertex, we also remove all the edges associated with that vertex. It means, while removing a vertex, we remove any edge that is outgoing or ingoing to that vertex.



3) Insert Edge : This operation is responsible for adding an edge between a pair of vertices.

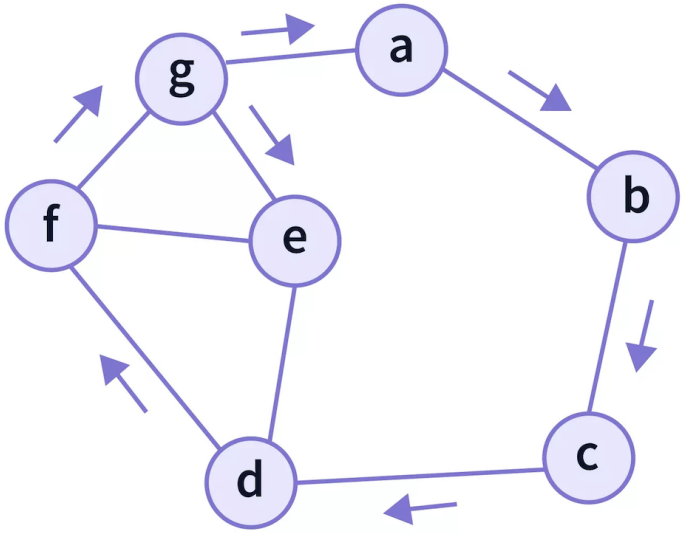


4) Remove Edge : It involves removing an edge from a graph. If all the edges from a particular vertex(node) is removed, then that vertex(node) becomes an isolated vertex.



5) Has Edge : This operation checks for an edge in the graph. It checks, whether an edge is present or not in the graph data structure. (By giving the nodes)

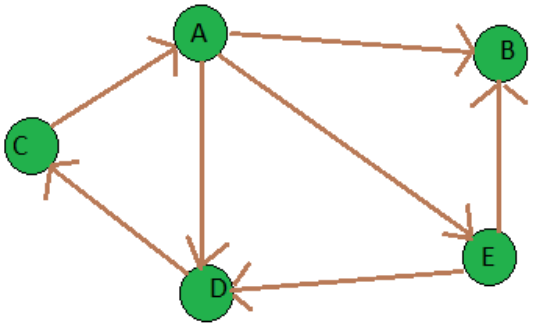
6) Graph Traversal : It includes traversing all the edges of a graph.



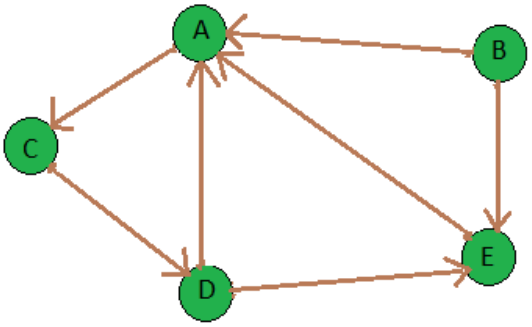
Graph Traversal

7) Display Vertex : It displays one or more vertices of the graph.

8) Get Transpose : The same set of vertices with all of the edges reversed compared itself.



(i)



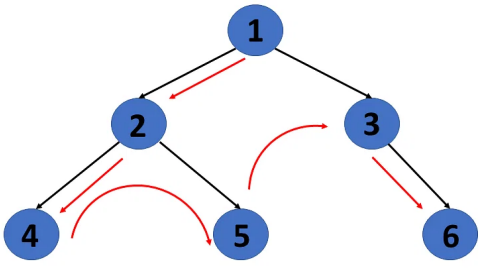
(ii)

```
Input : figure (i) is the input graph.
Output : figure (ii) is the transpose graph of the given graph.
```

Traversal Algorithms for Graph

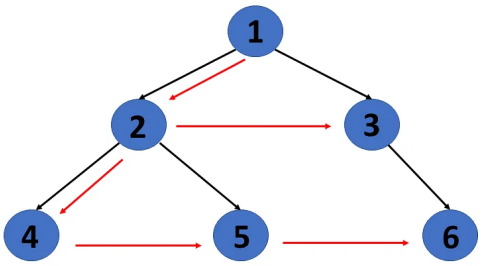
→ Both of the algorithms are recursive algorithm for searching all the vertices of a graph or tree data structure.

1) Deep First Search : It begins at the root node and examines each branch as far as feasible before backtracking.



DFS	1	2	4	5	3	6

2) Breadth First Search : It begins at the root of the graph and investigates all nodes at the current depth level before moving on to nodes at the next depth level.



BFS	1	2	3	4	5	6

Application Areas

→ **Social media analysis**: Social media platforms generate vast amounts of data in real-time, which can be analyzed using graphs to identify trends, sentiment, and key influencers. This can be useful for marketing, customer service, and reputation management.

→ **Network monitoring:** Graphs can be used to monitor network traffic in real-time, allowing network administrators to identify potential bottlenecks, security threats, and other issues. This is critical for ensuring the smooth operation of complex networks.

→ **Financial trading:** Graphs can be used to analyze real-time financial data, such as stock prices and market trends, to identify patterns and make trading decisions. This is particularly important for high-frequency trading, where even small delays can have a significant impact on profits.

→ **Internet of Things (IoT) management:** IoT devices generate vast amounts of data in real-time, which can be analyzed using graphs to identify patterns, optimize performance, and detect anomalies. This is important for managing large-scale IoT deployments.

→ **Autonomous vehicles:** Graphs can be used to model the real-time environment around autonomous vehicles, allowing them to navigate safely and efficiently. This requires real-time data from sensors and other sources, which can be processed using graph algorithms.

→ **Disease surveillance**: Graphs can be used to model the spread of infectious diseases in real-time, allowing health officials to identify outbreaks and implement effective containment strategies. This is particularly important during pandemics or other public health emergencies.

→ **GPS systems** and **Google Maps** use graphs to find the shortest path from one destination to another.

→ The **Google Search** algorithm uses graphs to determine the relevance of search results.

→ **World Wide Web** is the biggest graph. All the links and hyperlinks are the nodes and their interconnection is the edges. This is why we can open one webpage from the other.

→ The best example of graphs in the real world is Facebook. Each person on Facebook is a node and is connected through edges. Thus, A is a friend of B. B is a friend of C, and so on.

The basis of Comparison	Graph	Tree
Definition	Graph is a non-linear data structure.	Tree is a non-linear data structure.
Structure	It is a collection of vertices/nodes and edges.	It is a collection of nodes and edges.
Structure cycle	A graph can be connected or disconnected, can have cycles or loops, and does not necessarily have a root node.	A tree is a type of graph that is connected, acyclic (meaning it has no cycles or loops), and has a single root node.
Edges	Each node can have any number of edges.	If there is n nodes then there would be n-1 number of edges
Types of Edges	They can be directed or undirected	They are always directed
Root node	There is no unique node called root in graph.	There is a unique node called root(parent) node in trees.
Loop Formation	A cycle can be formed.	There will not be any cycle.
Traversal	For graph traversal, we use Breadth-First Search (BFS), and Depth-First Search (DFS).	We traverse a tree using in-order, pre-order, or post-order traversal methods.
Applications	For finding shortest path in networking graph is used.	For game trees, decision trees, the tree is used.
Node relationships	In a graph, nodes can have any number of connections to other nodes, and there is no strict parent-child relationship.	In a tree, each node (except the root node) has a parent node and zero or more child nodes.

📚 References

GeeksforGeeks | A computer science portal for geeks

A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles, quizzes and practice/competitive programming/company

https://www.geeksforgeeks.org/

Programiz: Learn to Code for Free

Learn to code in Python, C/C++, Java, and other popular programming languages with our easy to follow tutorials, examples, online compiler and references.

https://www.programiz.com

Simplilearn | Online Courses - Bootcamp & Certification Platform

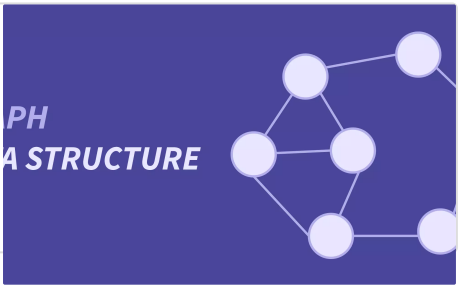
Simplilearn is the popular online Bootcamp & online courses learning platform that offers the industry's best PGPs, Master's, and Live Training. Start upskilling!

https://www.simplilearn.com/

Graph Data Structure - Scaler Topics

Learn about Graph in Data Structure by Scaler Topics. Explore graph data structure, their visual representation, terminologies, operations and types.

https://www.scaler.com/topics/data-structures/graph-in-data-structure/



✍🏻 Author → Serhat Kumas

https://www.linkedin.com/in/serhatkumas/

SerhatKumas - Overview

Computer engineering student who loves coding in different fields instead of focusing on a one spesific area. - SerhatKumas

https://github.com/SerhatKumas