**Introduction To Graph Theory**

**Iheb Gafsi**\*

INSAT Student

**Iheb.engineer@gmail.com**

**Definition:**

Graph Theory is a branch of mathematics and computer science that deals with the study of graphs/networks. Graphs are mathematical structures that represent relationships between pairwise objects that are referred to as vertices or nodes. The connections between these nodes can also be referred to as edges.

**Use Cases:**

Graph Theory can help us solve several problems, in mathematics, computer science, data science, social sciences, operations research, network analysis, and many more. Graph theory helps in understanding the connections and interactions between different elements in a system, leading to insights, algorithms, and optimizations for various real-world applications.

**Types of Graphs:**

* **Undirected Graphs:**

Undirected Graphs are the most generalized form. A graph is an undirected graph when the edges connecting it are undirected so the edge connecting the nodes and . **Figure 1** shows a little example of undirected graphs:

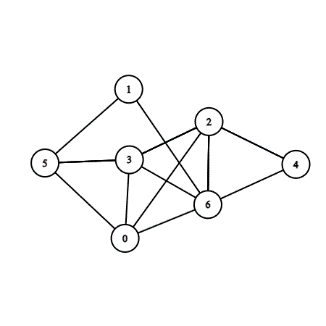
****

Figure 1: Undirected Graph

* **Directed Graphs:**

Unlike undirected graphs, directed graphs have directions so the edge connecting the nodes and . **Figure 2** illustrates an example of directed graphs:

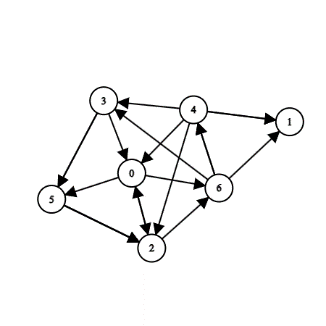
****

Figure 2: Directed Graph

* **Weighted Graphs:**

Unlike typical graphs, weighted graphs have weights associated with them so the edge connecting the nodes and is represented as where w is the corresponding weight. **Figure 3** represents an undirected weighted graph:

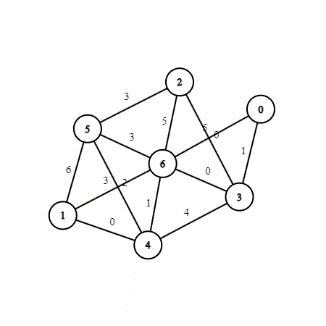
****

Figure 3: Directed Graph

* **Directed Acyclic Graphs DAGs:**

Directed Acyclic Graphs (DAGs) are directed graphs without cycles, making them suitable for modeling processes with clear cause-and-effect relationships. On the other hand, directed graphs can have cycles, allowing paths to loop back to the starting node. **Figure 4** shows an example of a DAG:

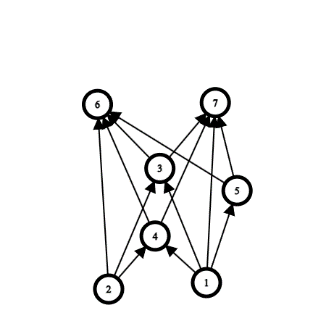
****

Figure 4: Directed Acyclic Graph

* **Trees:**

Trees are acyclic, connected graphs with no cycles and a unique starting point called the root. They are widely used in computer science, data structures, and represent hierarchical relationships efficiently. Each node has one parent, except for the root. **Figure 5** shows an instance of a tree:

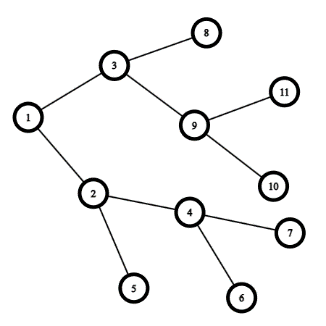
****

Figure 5: Tree

* **Bipartite Graphs:**

Bipartite graphs are graphs with two distinct sets of vertices, where edges only connect vertices from different sets. They are used to model relationships between two types of objects or entities and have applications in matching problems, network flow, and social networks. **Figure 6** illustrates what a bipartite graph looks like:

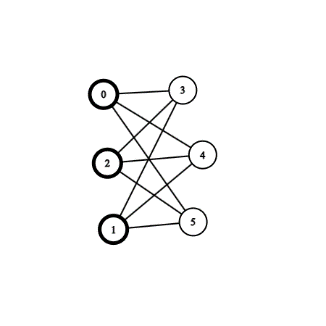
****

Figure 6: Bipartite Graph

* **Complete Graphs:**

A complete graph is simply a completely connected graph. Figure 7 below shows an example of a complete graph:

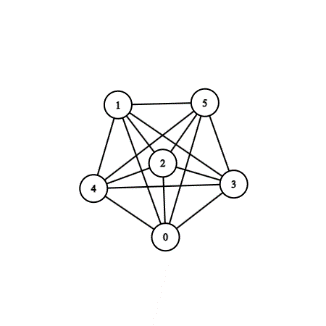
****

Figure 7: Complete Graph

**Representing a Graph:**

* **Adjacency Matrix:**

An adjacency matrix is a tool that allows us to best represent a directed weighted graph. Every element represents the weight of the edge connecting the i’th node to the j’th one. Here’s an example illustrated in **Figure 8**:

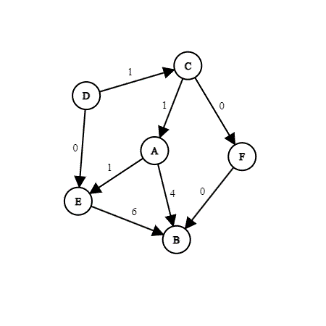
****

Figure 8: Graph for adjacency matrix

* **Adjacency List:**

An Adjacency List is a way to represent a graph as a map from nodes to lists of edges. Here’s the example for the case of **Figure 8**:

A [(B, 4), (E, 1)]

B []

C [(A, 1), (F, 0)]

D [(C, 1), (E, 0)]

E [(B, 6)]

F [(B, 0)]

* **Edges List:**

Unlike other representations we can simply represent our graph with its unordered list of connections. We will use the example of the case of **Figure 8** again:  
Graph = [(A, B, 4), (A, E, 1), (C, A, 1), (C, F, 0), (D, C, 1), (D, E, 0), (E, B, 6), (F, B, 0)]