Data Structure and Algorithm

Laboratory Activity No. 10

Intro to Graphs

Submitted by:	Instructor:
Sumel, Hendrix Nathan L.	Engr. Maria Rizette H. Sayo

October 11, 2025

- ## Objectives

Introduction

A graph is a visual representation of a collection of things where some object pairs are linked together. Vertices are the points used to depict the interconnected items, while edges are the connections between them. In this course, we go into great detail on the many words and functions related to graphs.
An undirected graph, or simply a graph, is a set of points with lines connecting some of the points. The points are called nodes or vertices, and the lines are called edges.

A graph can be easily presented using the python dictionary data types. We represent the vertices as the keys of the dictionary and the connection between the vertices also called edges as the values in the dictionary.

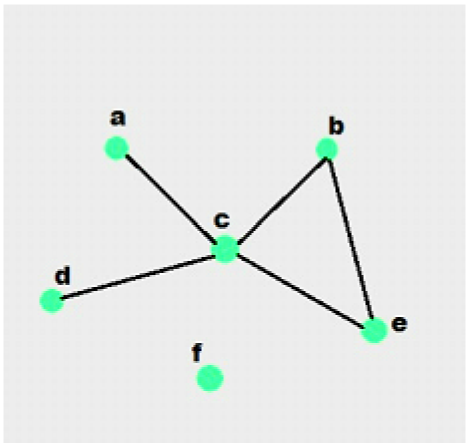


Figure 1. Sample graph with vertices and edges

This laboratory activity aims to implement the principles and techniques in:

- To introduce the Non-linear data structure – Graphs
- To discuss the importance of Graphs in programming

- ## Methods

  Discuss the following terms related to graphs:
  - Undirected graph
  - Directed graph
  - Nodes
  - Vertex
  - Degree
  - Indegree
  - Outdegree
  - Path
  - Cycle
  - Simple Cycle

Key Graph Terminology

1. **Undirected Graph**
   A graph where the edges have no direction. If node A is connected to node B, then node B is also connected to node A.

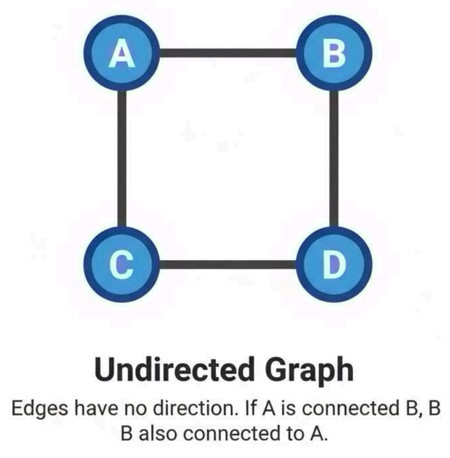


Figure 2. Undirected Graph

2. **Directed Graph**
   A graph where edges have a direction, indicating a one-way relationship from one node to another.

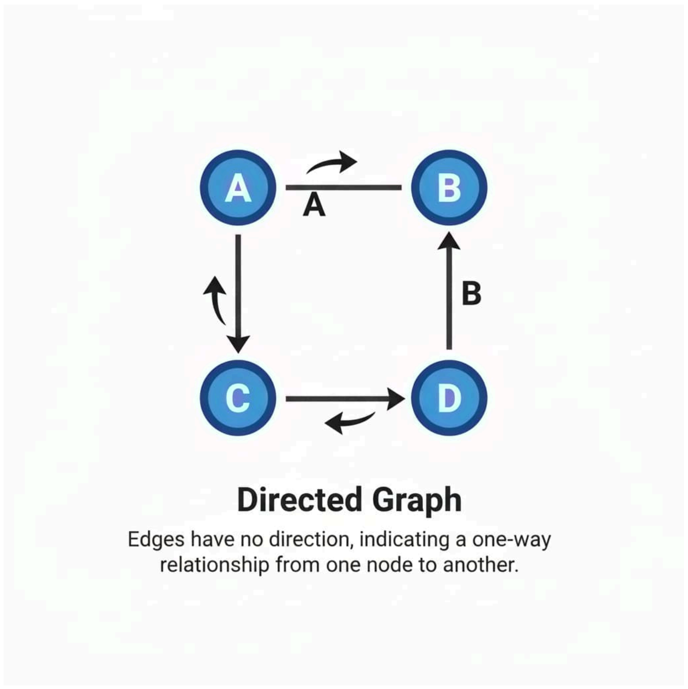

A
B
A
B
C
D
Directed Graph
Edges have no direction, indicating a one-way relationship from one node to another.

Figure 3. Directed Graph

3. **Node / Vertex**
   An individual element or point in a graph.

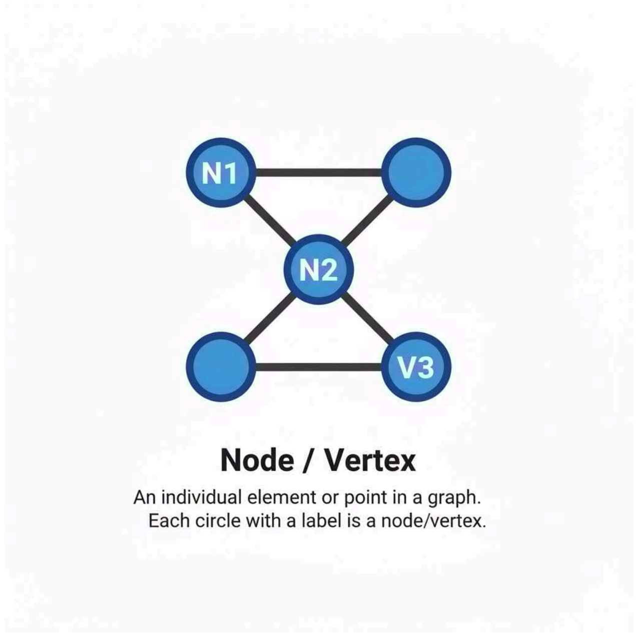

N1
N2
V3
Node / Vertex
An individual element or point in a graph.
Each circle with a label is a node/vertex.

Figure 4. Node/Vertex

4. **Edge**
 A connection or link between two nodes.

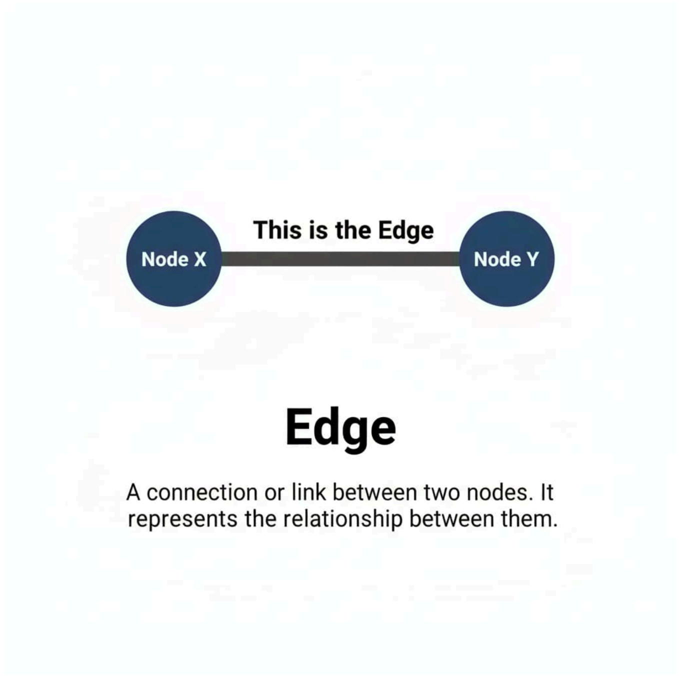

This is the Edge
Node X
Node Y
Edge
A connection or link between two nodes. It represents the relationship between them.

Figure 6. Edge

5. **Degree**
 The number of edges incident to a vertex. In undirected graphs, it's the total number of connections a node has.

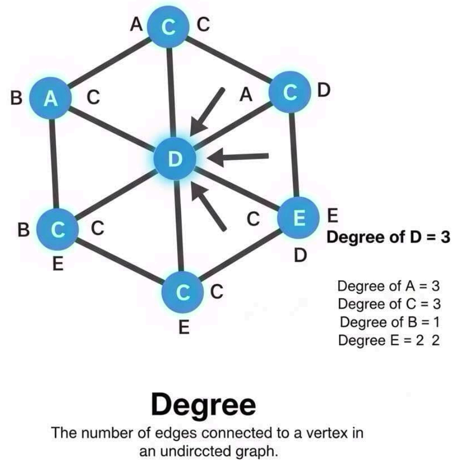


Figure 7. Degree

6. **Indegree**
   Number of edges directed *into* a vertex (used in directed graphs only).

7. **Outdegree**
   Number of edges directed *out* from a vertex (used in directed graphs only).

8. **Path**
   A sequence of vertices where each adjacent pair is connected by an edge.

9. **Cycle**
   A path that starts and ends at the same vertex with no repeated edges or vertices (except the start/end).

10. **Simple Cycle**
    A cycle that doesn't repeat any vertex or edge (except the starting and ending vertex).

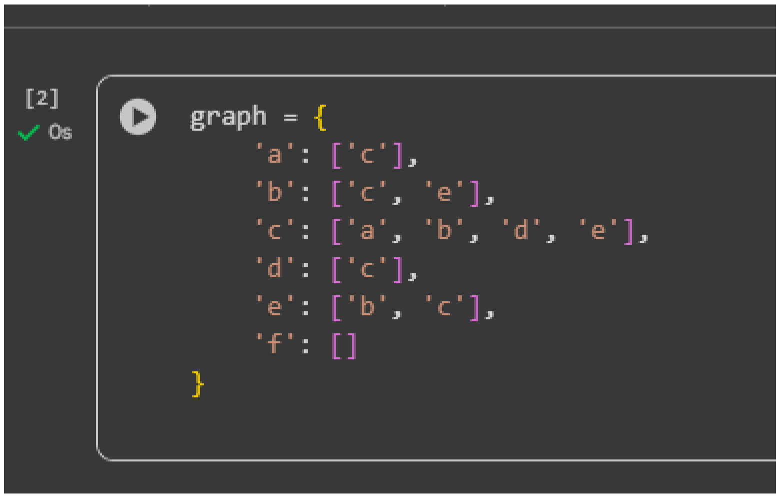
- ## Results

Figure 1. Sample undirected graph with vertices a-f and edges

B. Observations and Analysis

From **Figure 1**, we can observe the following:

- The graph contains **6 vertices**: {a, b, c, d, e, f}
- **Edges** present:
  (a, c), (b, c), (c, d), (c, e), (b, e)
- Node **c** has the **highest degree** (4), meaning it is the most connected node.
- Node **f** is **isolated**, having **degree 0**, thus not part of any edge.
- There exists a **cycle**: b → c → e → b
  - This is also a **simple cycle**, as no node (except the start/end) is repeated.



```
[2]
graph = {
    'a': ['c'],
    'b': ['c', 'e'],
    'c': ['a', 'b', 'd', 'e'],
    'd': ['c'],
    'e': ['b', 'c'],
    'f': []
}
```

If an image is taken from another literature or intellectual property, please cite them accordingly in the caption. Always keep in mind the Honor Code [1] of our course to prevent failure due to academic dishonesty.

- ## Conclusion

Graphs are essential tools in programming and computer science, enabling the modeling of complex relationships and networks. Understanding key concepts like degree, cycles, and paths is foundational for more advanced applications in data science, artificial intelligence, and optimization problems.

In this activity, we visualized an undirected graph, identified its structure, and explored the properties of its nodes and edges. We also practiced graph representation using Python dictionaries, preparing us for future implementations involving algorithms such as Depth-First Search (DFS), Breadth-First Search (BFS), and shortest path computations.

References

[1] J. Erickson, *Algorithms*, 2019. [Online]. Available: https://jeffe.cs.illinois.edu/teaching/algorithms/.
[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. MIT Press, 2009.
[3] "Graphs in Python – Real Python," *Real Python*. [Online]. Available: https://realpython.com/python-graphs-networkx/.