Data Structure and Algorithm

Laboratory Activity No. 10

Intro to Graphs

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Elpedes, Glen Jorge A. | 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.

A diagram of a triangle with green dots

AI-generated content may be incorrect.

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

* 1. Discuss the following terms related to graphs:
     1. Undirected graph
     2. Directed graph
     3. Nodes
     4. Vertex
     5. Degree
     6. Indegree
     7. Outdegree
     8. Path
     9. Cycle
     10. Simple Cycle

# Results

1. Undirected Graph - simple tree where the parent-child relationships aren't strictly directional (like a family tree), it's treated as a connected set of nodes with non-directional edges.

Examples:

V = {(A, C), (C, A), (B, C), (C, B), (B, E), (E, B), (C, D), (D, C), (C, E), (E, C)}

2. Directed Graph - A collection of nodes joined by edges (or arcs) with a particular

direction is called a directed graph (or digraph).

Examples:

V = {A → C}

V = {C → A}

V = {B → C}

V = {C → B}

V = {B → E}

V = {E → B}

V = {C → D}

V = {D → C}

V = {C → E}

V = {E → C}

3. Node - Is a basic building block of a data structure, like a linked list or tree, that includes

data and one or more links to other nodes, which are frequently implemented as pointers.

Examples:

V = {A, B, C, D, E, F}

4. Vertex - Is a basic unit used to represent a specific point in a graph.It is a crucial part of

data storage and has edges connecting it to other vertices.

Examples:

V = {A, B, C, D, E, F}

5. Degree - The total number of edges incident to a vertex is known as its degree. This is

just the number of edges that connect a vertex in an undirected graph. The degree is further separated into in-degree and out-degree in directed graphs. A vertex's in-degree is the number of edges that point in its direction, and its out-degree is the number of edges that point in its opposite direction.

Example:

A = 1

B = 2

C = 4

D = 1

E = 2

F = 0

6. Indegree - The count of incoming connections from other vertices is represented by the

indegree of a vertex, which is the number of edges that point to it.

Examples:

A = 1

B = 0

C = 2

D = 1

E = 2

F = 0

7. Outdegree - Is used to describe how many edges come from a specific vertex. This idea,

which measures the number of connections a particular vertex has pointing away from it, is essential for examining the connectivity and flow in directed graphs.

Examples:

A = 1

B = 2

C = 2

D = 0

E = 0

F = 0

8. Path - Is characterized as a series of edges that join a series of vertices, enabling traversal

between them. Formally speaking, it can be defined as a series of edges where each edge is incident to the next, or, conversely, as a series of vertices where every pair of vertices is adjacent. A path is deemed simple if it doesn't repeat any vertices or edges, with the possible exception of the first and last vertices in a closed path.

Examples:

A → D

V = {(A, C), (C, D)

D → E

V = {(D, C), (C, E)}

A → E

V = {(A. C), (C, E),

V = {(A, C), (C, B), (B, E)}

B → D

V = {(B, C), (C, D)}

E → A

V = {(E, B), (B, C), (C, E)}

9. Cycle - Is a path that forms a closed loop without going over any edges more than once,

beginning and ending at the same vertex.

10. Simple Cycle - a cycle where every vertex in the path is unique and only the beginning

and ending vertices are repeated. Accordingly, a simple cycle is a closed path that starts and stops at the same vertex without going back to any other vertex.

# Conclusion

We studied graphs and their use in illustrating the relationships between points in this exercise. We used Python to create a graph, which is composed of nodes and edges. We also attempted to traverse the graph using a technique known as DFS. This gave us a better understanding of how graphs function and how to use them in programming and real-world scenarios.

**References**

[1] GeeksforGeeks, “Graph terminology in data structure,” *GeeksforGeeks*, Jul. 23, 2025. <https://www.geeksforgeeks.org/dsa/graph-terminology-in-data-structure/>

[2] “Graph Data Structure.” <https://www.programiz.com/dsa/graph>