Data Structure and Algorithm

Laboratory Activity No. 10

Intro to Graphs

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Monoy, Justin Rhey 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 - this is a type of graph that does not show the direction which must be taken between nodes. Instead, travel between nodes is allowed along an edge in either direction. There are no loops or multiple edges in undirected graphs.

Example from figure 1:

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

* + 1. Directed graph - this is a type of graph where the edges direct the path that must be taken to travel between connected nodes. The edges are typically represented as arrows.

Example from figure 1:

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}

* + 1. Nodes - is the fundamental component of a data structure, such as a linked list or tree, which consists of data and one or more links to other nodes, often implemented as pointers.

Example from figure 1:

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

* + 1. Vertex - is a basic type of graph representation that includes edges connecting it to other vertices. It is an essential component of data storage.

Example from figure 1:

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

* + 1. Degree - A vertex's degree is the total number of edges that incident to it. In an undirected graph, this is just the number of edges connecting a vertex. In directed graphs, the degree is further divided into in-degree and out-degree. The number of edges pointing in a vertex's direction is known as its in-degree, and the number of edges pointing in the opposite direction is known as its out-degree.

Example from figure 1:

A = 1

B = 2

C = 4

D = 1

E = 2

F = 0

* + 1. Indegree - is the number of edges coming to the vertex.

Example from figure 1:

A = 1

B = 0

C = 2

D = 1

E = 2

F = 0

* + 1. Outdegree - is the number edges which come out from the vertex.

Example from figure 1:

A = 1

B = 0

C = 2

D = 1

E = 2

F = 0

* + 1. Path - is defined as a set of edges that connect a set of vertices so that they can be traversed. It can be formally described as a set of edges where each edge is adjacent to the next or, on the other hand, as a set of vertices where each pair of vertices is adjacent. Unless the start and end vertices of a closed path are repeated, a path is considered simple if it contains no repeating vertices or edges.

Example from figure 1:

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)

* + 1. Cycle - is a path that starts and ends at the same vertex and creates a closed loop without crossing any edges more than once.
    2. Simple cycle - is a graph cycle that has only the starting and ending vertices and no other repeating vertices. In general, a cycle is considered simple if it cannot be divided into two or more cycles.

# Conclusion

This laboratory taught me a lot about data structures, especially in non-linear data structures which is graph where I can identify the type of graph, if the graph is directed or undirected, the nodes, vertex, degree, degree, degree, degree, degree, indegree, outdegree, path, cycle, and simple cycle of the graph. Graph is important in programming because it helps show data clearly and easily. It makes numbers or results easier to understand by turning them into pictures, like lines, bars, or circles, instead of just text or numbers.

**References**

[1] GeeksforGeeks, “What is Undirected Graph? | Undirected Graph meaning,” *GeeksforGeeks*, Jul. 23, 2025. <https://www.geeksforgeeks.org/dsa/what-is-unidrected-graph-undirected-graph-meaning/>

[2] GeeksforGeeks, “What is Directed Graph? | Directed Graph meaning,” *GeeksforGeeks*, Jul. 23, 2025. <https://www.geeksforgeeks.org/dsa/what-is-directed-graph-directed-graph-meaning/>

[3] “W3Schools.com.” <https://www.w3schools.com/dsa/dsa_theory_graphs.php>

[4] R. A. S, “Graph in Data Structure | Types & Explanation,” *Simplilearn.com*, Sep. 01, 2025. <https://www.simplilearn.com/tutorials/data-structure-tutorial/graphs-in-data-structure>

[5] “Degree of each vertex in the graph | Indegree and Outdegree of vertex.” <https://www.log2base2.com/data-structures/graph/degree-of-each-vertex-in-the-graph.html>