Data Structure and Algorithm

Laboratory Activity No. 10

Intro to Graphs

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Caasi, Karl Benedict D. | 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 that is viewed as a connected collection of nodes with non-directional edges and in which the parent-child relationships are not necessarily directional, as in a family tree.

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 group of nodes connected by arcs or edges that have a specific   
A directed graph (sometimes known as a digraph) is one that has direction.

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 fundamental component of a data structure, such as a tree or linked list, that comprises information and one or more connections to other nodes, often used as pointers

.

Examples:

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

4. Vertex - is a fundamental unit that is used to show a particular graph point. It is an essential component of has edges that link it to other vertices and stores data.

Examples:

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

5. Degree - A vertex's degree is the total number of edges that incident to it. This is   
just the quantity of edges in an undirected graph that link 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:

A = 1

B = 2

C = 4

D = 1

E = 2

F = 0

6. Indegree - The number of connections flowing in from different vertices is shown by the   
the number of edges pointing to a vertex is known as its indegree.

Examples:

A = 1

B = 0

C = 2

D = 1

E = 2

F = 0

7. Outdegree - Is used to indicate the number of edges that originate from a certain vertex. Examining the connectedness and flow in directed graphs requires an understanding of this concept, which counts the amount of connections a given vertex has pointing away from it.

Examples:

A = 1

B = 2

C = 2

D = 0

E = 0

F = 0

8. Path - Is defined as a set of edges that connect a set of vertices to allow for traversal. in between them. In formal terms, it can be described as a set of vertices where every pair of vertices is adjacent, or as a set of edges where each edge is incident to the next. Except for the start and last vertices in a closed path, a path is considered simple if it has no repetitions of vertices or edges.

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 creates a closed loop without repeatedly crossing any edges,   
starting at the same vertex and finishing there.

10. Simple Cycle - a cycle in which each path vertex is distinct and only the start   
as well as repeated terminating vertices. Thus, a closed path that begins and ends at the same vertex without returning to any other vertex is called a simple cycle.

# Conclusion

In this assignment, we looked at graphs and how they are used to show the relationships between points. We created a graph, which is made up of nodes and edges, using Python. We also tried utilizing a method called DFS to navigate the graph. This helped us better understand how graphs work and how to apply them in real-world situations and programming.

**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>