COEN-352 Tutorial #11

June 20th, 2023

GRAPH DS

Definition: a non-linear data structure consisting of vertices and edges.

- Vertices are set of objects connected via edges.
- E.G., One can think of the components of a Social Media are like nodes that have relationships to other nodes defined by edges.
- A graph is usually represented as **G = {V, E}**, where **G** is the graph space, **V** is the set of vertices and **E** is the set of edges

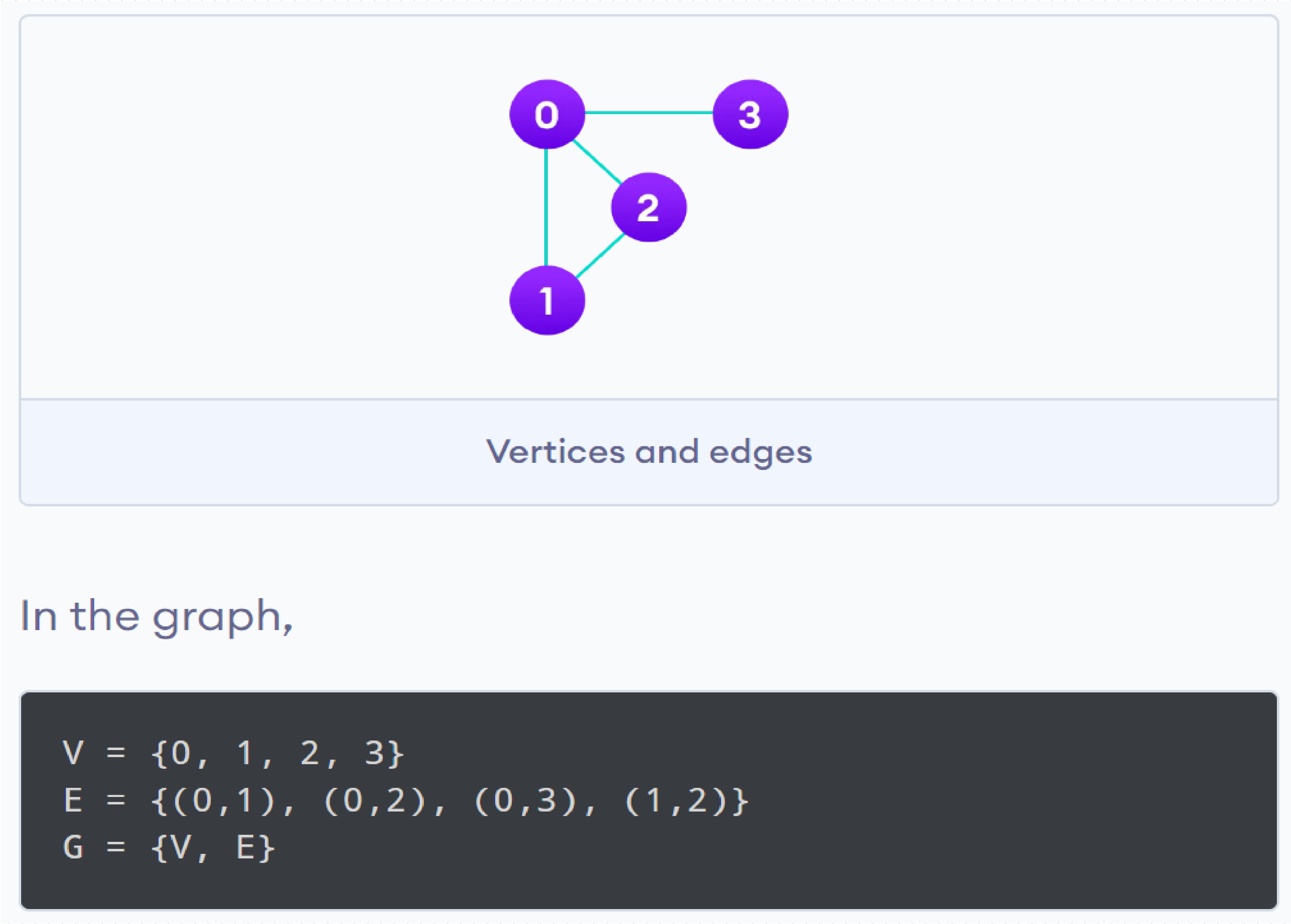
Some Definitions/Terms:

- **Vertex:** A Node in the Graph
- **Edge:** A link between two vertices in the Graph
- **Adjacency:** Two vertices are adjacent if there is an edge between them
- **Path:** Path represents a sequence of edges between the two vertices

Graph Representation:

- Adjacency Matrix
- Adjacency List

Example 1



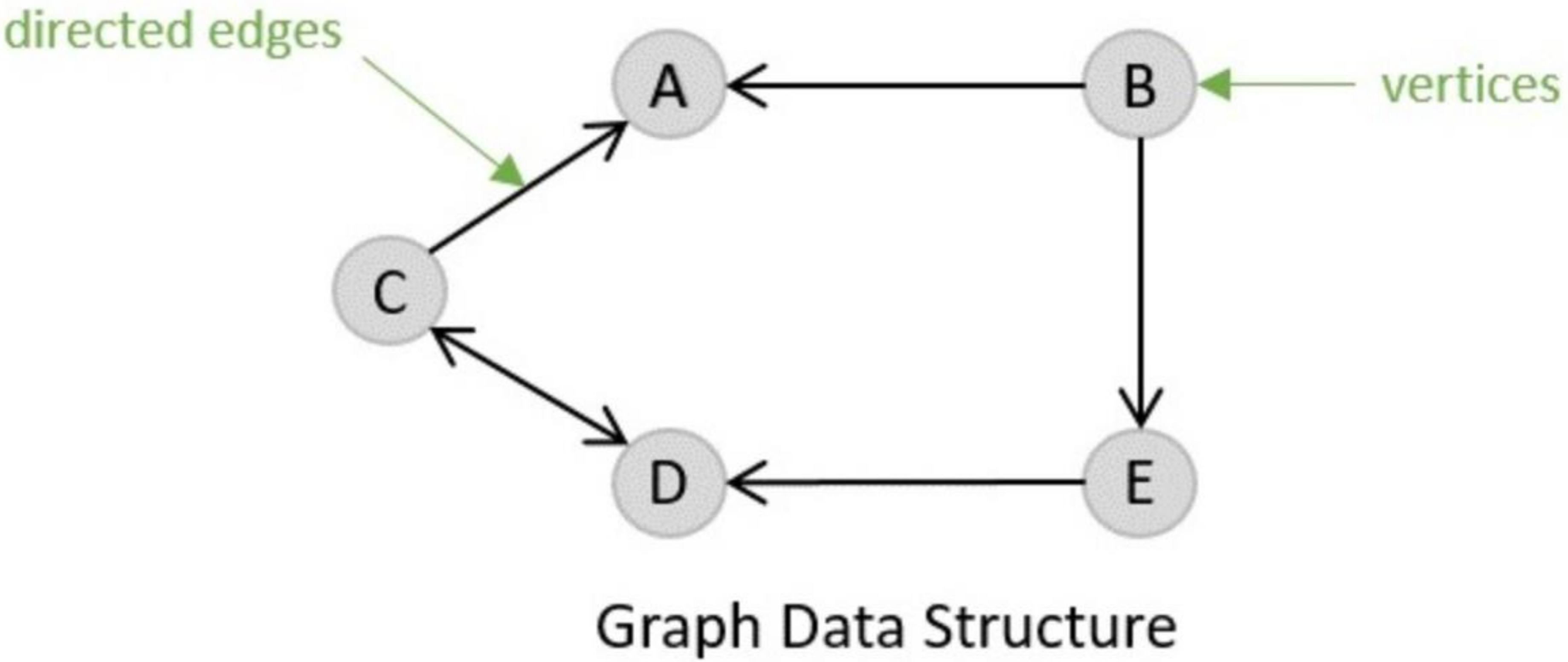
Vertices and edges

In the graph,

```
V = {0, 1, 2, 3}
E = {(0,1), (0,2), (0,3), (1,2)}
G = {V, E}
```

Source: https://www.programiz.com/dsa/graph

Example 2



Directed Graph

Source: https://www.tutorialspoint.com/data_structures_algorithms/graph_data_structure.htm#

Adjacency Matrix vs List

Adjacency List: An array of linked lists that contain the set vertices connected to the first element in the array.

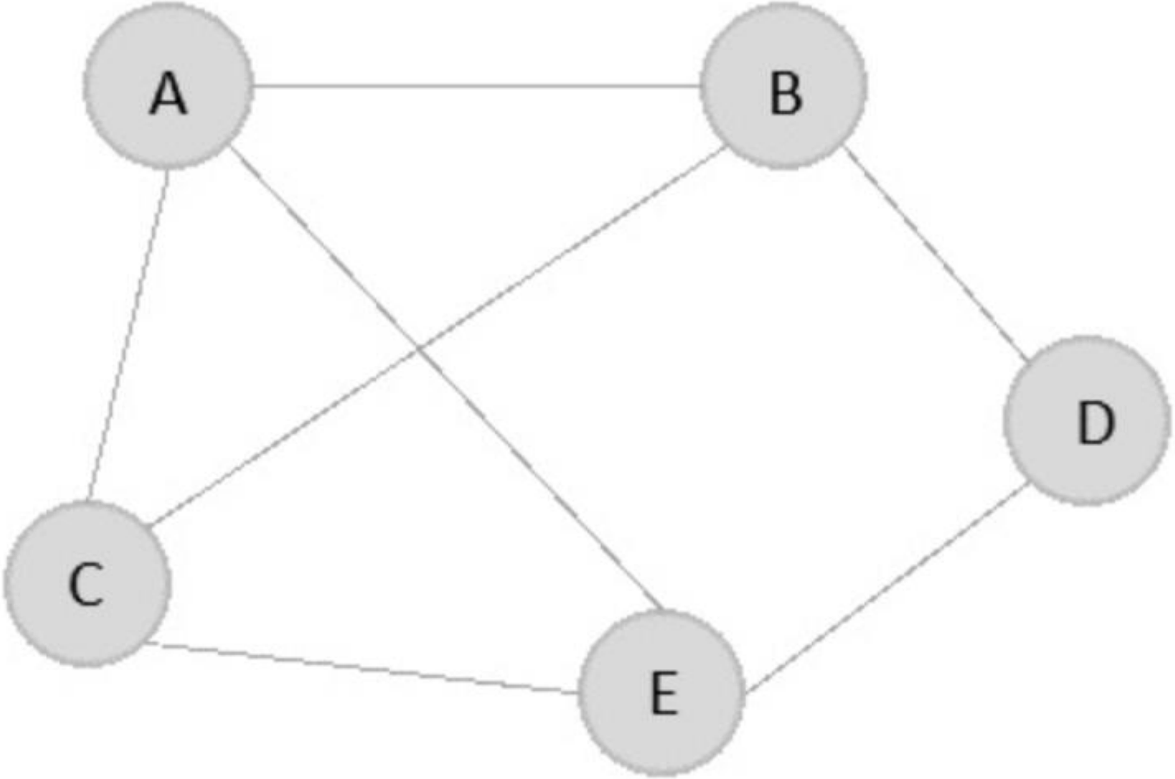
Adjacency Matrix: A 2D Array of size V x V where each element in the array can be 1 or 0.

- For a Graph **G[][]**, a slot **G[i][j] = 1** indicates that there is an edge from vertex **i** to vertex **j** and if **G[i][j] = 0**, it means otherwise.

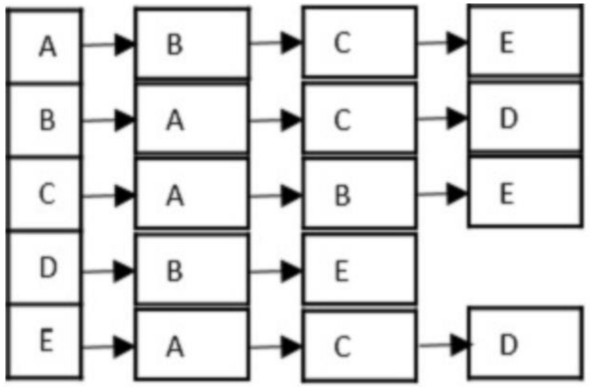
Adjacency List vs Matrix:

Operations	Adjacency Matrix	Adjacency List
Space Complexity	$O(\|V\|^2)$	$O(\|V\|+\|E\|)$
Inserting a Vertex	$O(\|V\|^2)$	O(1)
Inserting an Edge	O(1)	O(1)
Removing an Edge	O(1)	$O(\|E\|)$
Removing a Vertex	$O(\|V\|^2)$	$O(\|V\|+\|E\|)$

Adjacency Matrix vs List



Adjacency List Representation



A → B → C → E

B → A → C → D

C → A → B → E

D → B → E

E → A → C → D

Adjacency Matrix Representation

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 1 & 0 & 1 \\ B & 1 & 0 & 1 & 1 & 0 \\ C & 1 & 1 & 0 & 0 & 1 \\ D & 0 & 1 & 0 & 0 & 1 \\ E & 1 & 0 & 1 & 1 & 0 \end{array}$$

Types of Graphs

- **Weighted Graph:** The edges have weights, hence paths also.
- **Unweighted Graph:** Edges are not weighted.
- **Undirected Graph:** Edges have no sense of direction or you can assume that the edges are bi-directional.
- **Directed Graph:** Edges are directed from a vertex to its adjacent vertex.
    - Also called a **Digraph**
- **Acyclic Graph:** A graph with no cycles (paths from a node to itself).
- **Directed Acyclic Graph (DAG):** Directed graph with no cycles.

EXERCISES

Exercise 1: `toString()` method for an adjacency-list-based graph.

- The `toString()` method returns string representation of the graph.
- Use the graph implementation from the GitHub Repo.

Exercise 2: Create a method that returns the vertex with the most links.

- Use the graph implementation from the GitHub Repo.

THANK YOU