Representation of Graphs

Two approaches to represent a graph will be covered in this lesson.

We'll cover the following

Ways to Represent a Graph

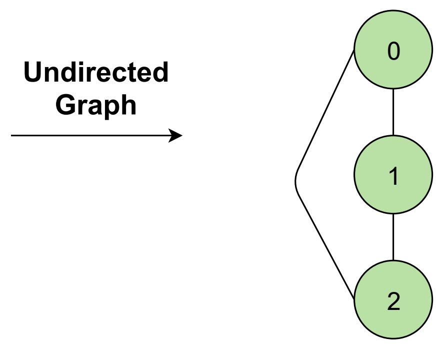
The two most common ways to represent a graph are:

1. Adjacency Matrix
2. Adjacency List

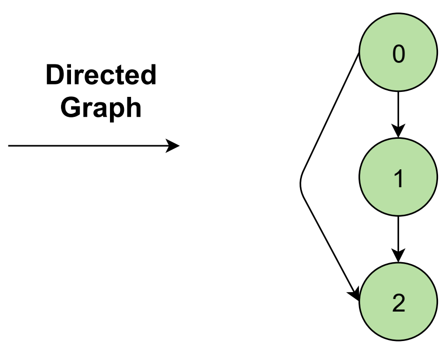
Adjacency Matrix

The **adjacency matrix** is a two-dimensional matrix where each cell can contain a **0** or **1**. If a cell contains **1**, there exists an edge between the corresponding vertices e.g., $Matrix[0][1] = 1$ shows that an edge exists between vertex **0** and **1**. The row and column headings represent the vertices.

	0	**1**	**2**
0	0	1	1
1	1	0	1
2	1	1	0



	0	**1**	**2**
0	0	1	1
1	0	0	1
2	0	0	0

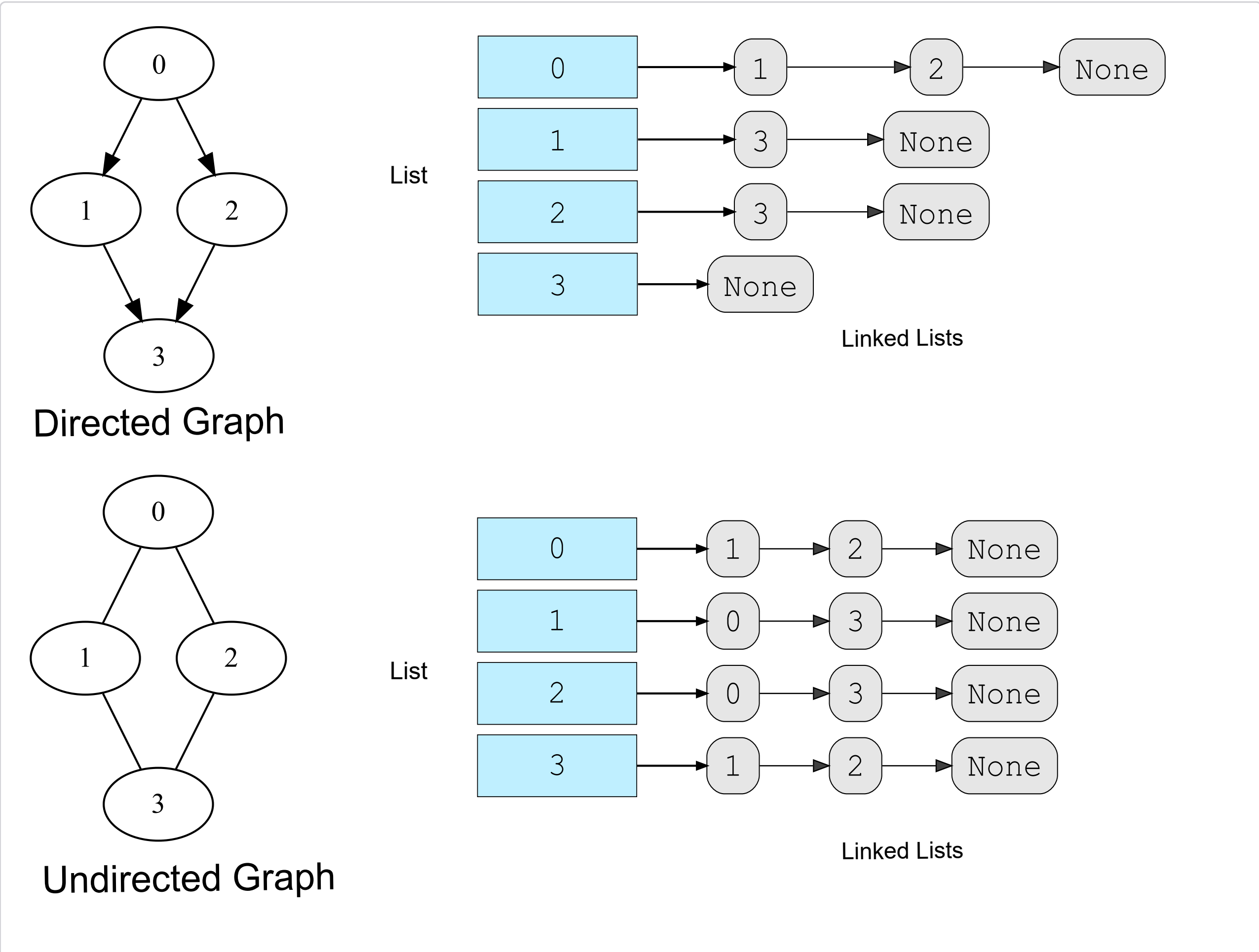


In the image above, there is a directed graph that has an edge going from vertex 0 to vertex 1, so there is a **1** at $Matrix[0][1]$ in the adjacency matrix. In the case of the undirected graph, we would have $Matrix[1][0] = 1$ as well since the edge is bidirectional.

For a directed graph, the usual convention is to think of the rows as sources and the columns as destinations.

Adjacency List

An array of linked lists is used to store all the edges in the graph. The size of the array is equal to the number of vertices in the graph. Each index of the array contains a vertex. This vertex points to a linked list that contains all the vertices connected to this one.



As you can see in the diagram above, all the vertices that connect directly to a vertex are appended in the corresponding linked list.

If a new vertex is added to the graph, it is simply added to the array as well.

> **Note:** In an undirected graph an edge is represented with both adjacent nodes having their linked list populated with the corresponding nodes.

In the next lesson, we are going to build a graph using the adjacency list approach.

Back

Types of Graphs

Next

Graph Implementation

Mark as Completed