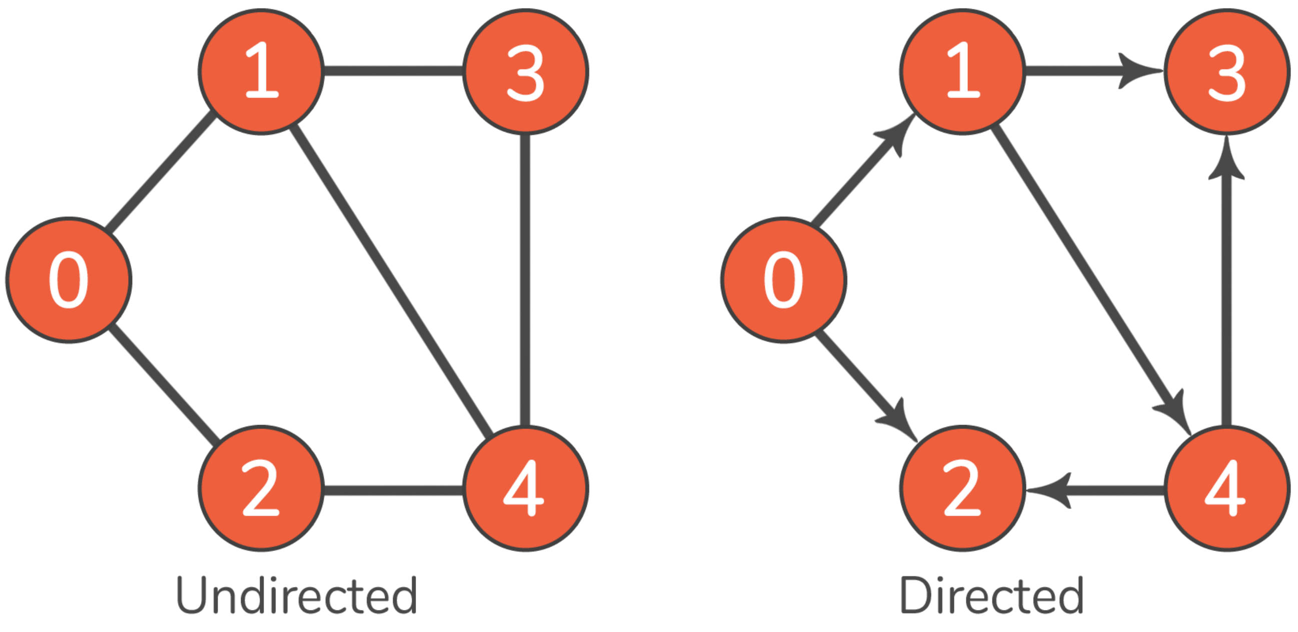
Graph Representation using Adjacency Matrix

CodeStudio

This program illustrates the representation of a graph using an adjacency matrix. The graph class encapsulates the logic for initializing, creating, and printing a graph. The adjacency matrix provides a compact way to store and visualize relationships between nodes.

Example:



Output:

```
Undirected Graph

0 : 0 1 1 0 0

1 : 1 0 0 1 1

2 : 1 0 0 0 1

3 : 0 1 0 0 1

4 : 0 0 1 1 0


Directed Graph

0 : 0 1 1 0 0
```

```
1 : 0 0 0 1 1

2 : 0 0 0 0 0

3 : 0 0 0 0 0

4 : 0 0 1 1 0
```

Graph Class

The **Graph** class is designed to efficiently represent graphs using an adjacency matrix. This matrix allows for a clear representation of connections between nodes.

Graph Constructor

Purpose:

- Initializes a graph with a specified number of nodes.

Time Complexity:

- **O(V^2): Nested loops for initializing the adjacency matrix.**

Space Complexity:

- **O(V^2): Space required to store the adjacency matrix.**

createGraph Function

Purpose:

- Creates the graph based on provided edges.

Time Complexity:

- **O(E), where E is the number of edges: Iterates over each edge once.**

Space Complexity:

- **O(1): Constant space for variables.**

printGraph Function

Purpose:

- Prints the adjacency matrix of the graph.

Time Complexity:

- **O(V^2): Nested loops for printing the adjacency matrix.**

Space Complexity:

- **O(1): Constant space for variables.**