University of Warwick

International Foundation Programme (IFP)

Computer Science

(Lecture 18 – Week 19) Worksheet 1 – Graph Data Structures

Project Title: Social Network Graph

Project Description:

In this project, you will build a social network graph using graph data structures. You will create a program that allows users to add new members to the network and define relationships between members. The program should be able to answer questions about the social network, such as who is friends with whom and how many degrees of separation there are between two members.

Project Requirements:

- Implement a Graph class that will represent the social network. Each node in the graph should represent a social network member, and each edge should represent a relationship between two members.
- Implement a method to add a new member to the social network. The method should include the member's name and any additional information about the member you want to store.
- Implement a method to add a relationship between two members. The method should take in the names of the two members and add an edge between them.
- Implement a method to find all the friends of a particular member. The method should take in the name of the member and return a list of all the member's friends.
- Implement a method to find the path between two members. The method should take in the names of the two members and return the number of edges that connect them.

Example Usage:

```
# Create a new social network
network = Graph()

# Add some members to the network
network.add_member("Alice", age=25, location="New York")
network.add_member("Bob", age=30, location="Los Angeles")
network.add_member("Charlie", age=35, location="Chicago")
network.add_member("David", age=40, location="Seattle")

# Add some relationships between members
network.add_relationship("Alice", "Bob")
network.add_relationship("Bob", "Charlie")
network.add_relationship("Charlie", "David")

# Find all the friends of Alice
alice_friends = network.find_friends("Alice")
print(alice_friends) # Output: ["Bob"]
```

```
# Find the shortest path between Alice and David
shortest_path = network.shortest_path("Alice", "David")
print(shortest_path) # Output: 3
```

Project Title: Network Connectivity

Project Description:
You must implement a graph representing a list of connections between computers in a network, represented as lists [a, …, n] where *a* to *n* are the IDs of two or more connected computers. Your task is to determine if the entire network is connected.

Project Requirements:
Write a Graph class that allows you to add and remove a graph vertex (each vertex represents a computer in the network) and define the edges between vertices (an edge represents the connection between computers).
Write a function `is_network_connected` that returns a `bool` value. The function parameter can be the graph containing the vertices representing the computers in the network. It returns a `boolean` value indicating if the entire network is connected, i.e., it returns True if each computer has at least one connection.
Hint: You can use graph traversal algorithms (e.g., depth-first search or breadth-first search) to explore the connected components of the network. If you can traverse all vertices in the graph, that means all computers are connected.

Example Usage:
```
# Create a network graph
network = Graph()

# Add vertices and connect them
network.add_vertex("A")
network.add_vertex("B")
network.add_edge("A", "B")

# Verify is all computers are connected
is_network_connected(network)
Output: True

# Add more computers to the network and connect them
network.add_vertex("C")
network.add_vertex("C")
network.add_edge("C", "D")

# Verify is all computers are connected
is_network_connected(network)
Output: False
```