Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year:2023), BSc. in CSE (Day)

LAB REPORT - 03

Course Title: Artificial Intelligence Lab

Course Code: CSE-316 **Section:** PC-201 DC

Student Details

Name		Students Id
1.	Md. Romzan Alom	201902144

Lab Date: 06-05-2023

Submission Date: 13-05-2023

Course Teacher's Name: Rusmita Halim Chaity

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status	
Marks: ……	**Signature: ……**
Comments: ……	**Date: ……**

1. TITLE OF THE LAB EXPERIMENT

Implement a program to perform topological search using IDDFS in python.

2. OBJECTIVES/AIM

- To learn how Iterative Deepening depth-first search (IDDFS) works..
- To acquire knowledge about python.
- To implement IDDFS on python.
- To learn about mesh topological search.
- To learn python operators.
- To learn conditional statements in python.
- To learn loops in python.
- To learn recursive functions in python.

3. PROCEDURE / ANALYSIS / DESIGN

Problem: Implementing a program to carry out mesh topological search using IDDFS was the goal of this lab. The software ought to ask the user to enter the mesh network's nodes, edges, starting and ending nodes, among other information. The mesh topological search should then be carried out by the software using IDDFS, and if a path between the start node and the destination node is discovered, it should be printed. Two classes, MeshNode and Mesh, should be used in the program's Python implementation to represent the nodes and the mesh network, respectively.

4. IMPLEMENTATION

Code of this problem:

```
class MeshNode:
def __init__(self, name):
self.name = name
self.neighbors = []

def add_neighbor(self, neighbor):
self.neighbors.append(neighbor)

class Mesh:
def __init__(self):
self.nodes = {}
def add_node(self, name):
node = MeshNode(name)
self.nodes[name] = node
```

```
def add_edge(self, node1, node2):
if node1 in self.nodes and node2 in self.nodes:
self.nodes[node1].add_neighbor(self.nodes[node2])
self.nodes[node2].add_neighbor(self.nodes[node1])

def iddfs_search(self, start_node_name, goal_node_name):
depth = 0
while True:
visited = set()
result = self._depth_limited_search(start_node_name, goal_node_name, depth, visited)
if result is not None:
return result
depth += 1

def _depth_limited_search(self, current_node_name, goal_node_name, depth, visited):
if depth == 0 and current_node_name == goal_node_name:
return [current_node_name]
elif depth > 0:
visited.add(current_node_name)
for neighbor in self.nodes[current_node_name].neighbors:
if neighbor.name not in visited:
result = self._depth_limited_search(neighbor.name, goal_node_name, depth - 1, visited)
if result is not None:
return [current_node_name] + result
else:
return None

mesh = Mesh()
while True:
node_input = input("Enter a node name (or 'done' to finish): ")
if node_input == "done":
break
mesh.add_node(node_input)

while True:
edge_input = input("Enter an edge in the format 'node1 node2' (or 'done' to finish): ")
if edge_input == "done":
break
node1, node2 = edge_input.split()
mesh.add_edge(node1, node2)

start_node = input("Enter the name of the start node: ")
goal_node = input("Enter the name of the goal node: ")

result = mesh.iddfs_search(start_node, goal_node)

if result is None:
print("No path found.")
else:
print("Path found:")
print(" -> ".join(result))
```

5. TEST RESULT / OUTPUT

```
Enter a node name (or 'done' to finish): 1
Enter a node name (or 'done' to finish): 2
Enter a node name (or 'done' to finish): 3
Enter a node name (or 'done' to finish): 4
Enter a node name (or 'done' to finish): 5
Enter a node name (or 'done' to finish): 6
Enter a node name (or 'done' to finish): 7
Enter a node name (or 'done' to finish): done
Enter an edge in the format 'node1 node2' (or 'done' to finish): 1 3
Enter an edge in the format 'node1 node2' (or 'done' to finish): 2 5
Enter an edge in the format 'node1 node2' (or 'done' to finish): 1 4
Enter an edge in the format 'node1 node2' (or 'done' to finish): 2 7
Enter an edge in the format 'node1 node2' (or 'done' to finish): 3 5
Enter an edge in the format 'node1 node2' (or 'done' to finish): 6 7
Enter an edge in the format 'node1 node2' (or 'done' to finish): done
Enter the name of the start node: 1
Enter the name of the goal node: 6
Path found:
1 -> 3 -> 5 -> 2 -> 7 -> 6
```

Figure_01: Output of this problem

The mesh network and the user-provided start and target nodes affect the program's results while performing a mesh topological search using IDDFS. The program will print "Path found:" followed by the names of the nodes in the path from the start node to the target node separated by "->" if it is successful in finding a path from the start node to the goal node. When a path from the start node to the destination node cannot be found, the program will report "No path found."

In this output, the user takes input nodes 1 to 7 then linked them to create graph. Applying mesh topological search using IDDFS on this graph to find the shortest path from the starting point (1) to the destination point (6). The resulting path is printed as a list that is 1->3->5->2->7->6.

6. ANALYSIS AND DISCUSSION

This experiment mainly based on software. So, it may have so software error. Based on the focused objective(s) to learn the step-by-step working system of that problem. This problem is similar like DFS that’s why I can easily understand. The task will help us to create new new solution from different problems. The main hard part of this experiment is successfully completed traversal. IDDFS-based mesh topological search was carried out by the program

effectively. Program modularity and readability were improved by the use of the MeshNode and Mesh classes. The application was made interactive by the user input prompts, which also let the user enter any mesh network of their choosing. To make it simple for the user to grasp the outcome, the program also reported the path, if one was located, in a legible format. We face so many problem for understanding mesh topological of python and their working system. Now, we get so many knowledge to execute those function. Those are very important for our future lab task.

7. CONCLUSION:

The Python program for mesh topological search utilizing IDDFS was successfully developed. The software offered a user-interactive interface where users could enter their own mesh networks and showed how to apply the IDDFS algorithm for mesh topological search. There are several applications for the program that involve looking for nodes in mesh networks.