Divyadharshini K

241501053

Experiment 2

IMPLEMENTATION OF DEPTH FIRST SEARCH

Aim:

To implement depth first search

Case Scenario:

A robotic delivery system is implemented in a smart warehouse. The warehouse is modeled as a graph, where each node represents a storage unit and each edge represents a possible path. The robot needs to pick up a package from a starting point and deliver it to the correct storage location.

The robot's movement strategy is to explore the storage units by going as deep as possible before backtracking if needed. The warehouse is not fully mapped, so the robot uses a Depth First Search (DFS) algorithm to explore the paths.

Procedure:

Step 1: Input the Graph

🎬 Represent the warehouse as a graph (Adjacency List).
🎬 Define the start node and goal node.

Step 2: Initialize DFS

🎬 Use a set (visited) to track visited nodes.
🎬 Use a list (path) to store the current traversal path.

Step 3: Recursive DFS Function

Step 3: Recursive DFS Function

1. Mark the current node as visited.
2. Add the current node to the path.
3. Check if the current node is the goal:
   - o If yes, return the path.
   - o If no, proceed with the next steps.
4. Explore all neighboring nodes:

   - If a neighbor is not visited, recursively call DFS on it.
   - If a valid path is found, return it.
5. If no path is found, return None.

Step 4: Call the DFS Function

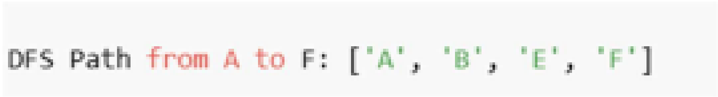
- Call DFS with the given start and goal nodes.
- Print the path found (if any).

Program:

```
# Depth First Search (DFS) implementation for a

warehouse graph  # Sample warehouse graph as

an adjacency list

warehouse_graph = {

 'A': ['B', 'C'],

 'B': ['D', 'E'],

 'C': ['F'],

 'D': [],

 'E': ['F'],

 'F': []

}

# Function to perform DFS

def dfs(graph, start, goal,

visited=None, path=None):   if visited is

None:

 visited = set()

 if path is None:

 path = []
```

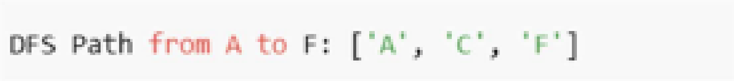
```
# Mark current node as visited and add to path
visited.add(start)
path.append(start)
# If goal is found, return the path
if start == goal:
return path

# Explore neighbors
for neighbor in graph[start]:
if neighbor not in visited:
result = dfs(graph, neighbor, goal, visited, path[:]) # Use path[:] to
copy path   if result: # Stop if a path is found
return result
return None # No path found
# Example usage
start_node = 'A'
goal_node = 'F'
path_found = dfs(warehouse_graph, start_node, goal_node)
print(f"DFS Path from {start_node} to {goal_node}: {path_found}")
```

Output:

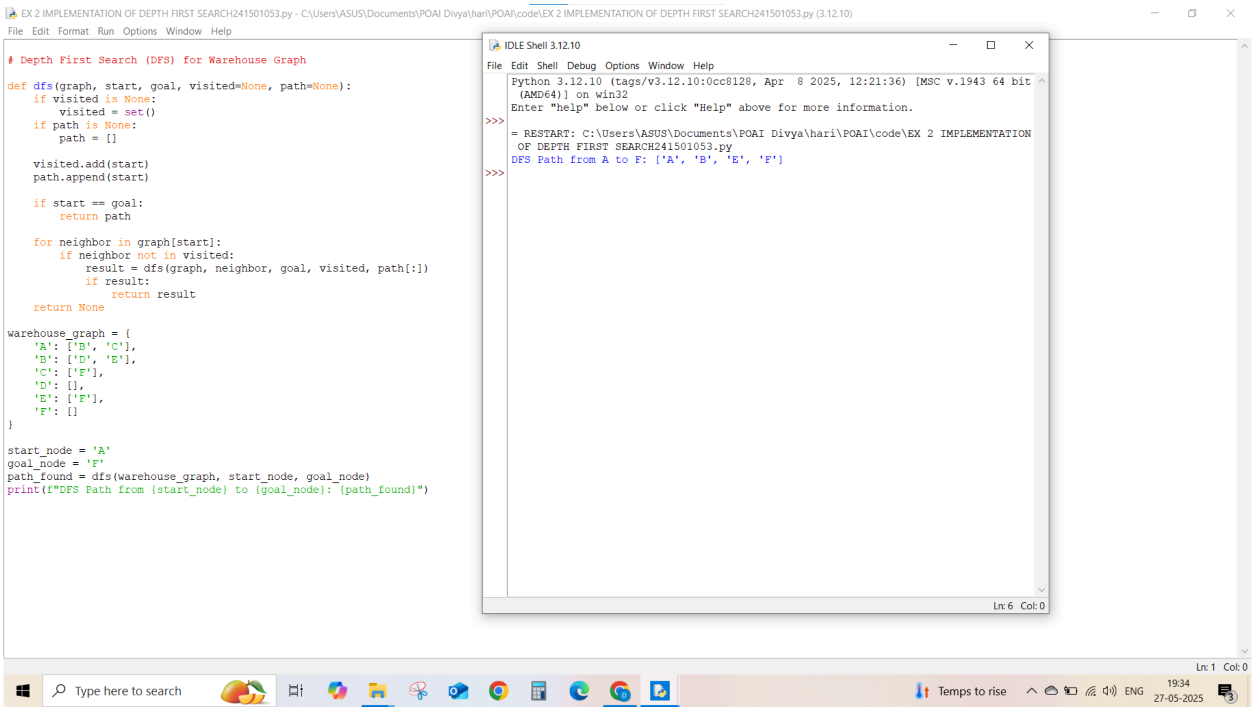


```
DFS Path from A to F: ['A', 'B', 'E', 'F']
```

Or



```
DFS Path from A to F: ['A', 'C', 'F']
```



EX 2 IMPLEMENTATION OF DEPTH FIRST SEARCH241501053.py - C:\Users\ASUS\Documents\POAI Divya\hari\POAI\code\EX 2 IMPLEMENTATION OF DEPTH FIRST SEARCH241501053.py (3.12.10)

```python
# Depth First Search (DFS) for Warehouse Graph

def dfs(graph, start, goal, visited=None, path=None):
    if visited is None:
        visited = set()
    if path is None:
        path = []

    visited.add(start)
    path.append(start)

    if start == goal:
        return path

    for neighbor in graph[start]:
        if neighbor not in visited:
            result = dfs(graph, neighbor, goal, visited, path[:])
            if result:
                return result
    return None

warehouse_graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}

start_node = 'A'
goal_node = 'F'
path_found = dfs(warehouse_graph, start_node, goal_node)
print(f"DFS Path from {start_node} to {goal_node}: {path_found}")
```

IDLE Shell 3.12.10

```
Python 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC v.1943 64 bit
 (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: C:\Users\ASUS\Documents\POAI Divya\hari\POAI\code\EX 2 IMPLEMENTATION
 OF DEPTH FIRST SEARCH241501053.py
DFS Path from A to F: ['A', 'B', 'E', 'F']
>>>
```