20CYS304

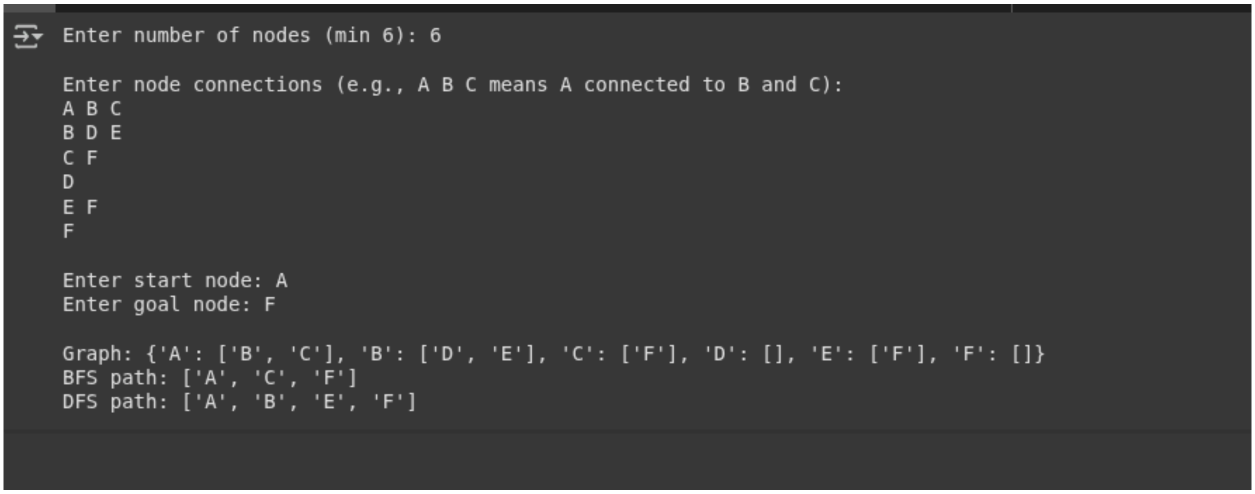
Artificial Intelligence & Neural Networks

Lab -1 BFS & DFS to traverse a graph or tree

Praneesh R V

CB.SC.U4CYS23036

```
from collections import deque

# Get graph input
graph = {}
num_nodes = int(input("Enter number of nodes (min 6): "))

print("\nEnter node connections (e.g., A B C means A connected to B and C):")
for _ in range(num_nodes):
    parts = input().split()
    node = parts[0]
    neighbors = parts[1:]
    graph[node] = neighbors

# Start and goal node
start = input("\nEnter start node: ")
goal = input("Enter goal node: ")

# BFS
def bfs(graph, start, goal):
    visited = set()
    queue = deque([[start]])

    while queue:
        path = queue.popleft()
        node = path[-1]
        if node == goal:
            return path
        if node not in visited:
            visited.add(node)
            for neighbor in graph.get(node, []):
                queue.append(path + [neighbor])
    return None

# DFS
def dfs(graph, start, goal):
    visited = set()
    stack = [[start]]

    while stack:
        path = stack.pop()
        node = path[-1]
        if node == goal:
            return path
        if node not in visited:
            visited.add(node)
            for neighbor in reversed(graph.get(node, [])):
                stack.append(path + [neighbor])
    return None

# Run and print
print("\nGraph:", graph)
print("BFS path:", bfs(graph, start, goal))
print("DFS path:", dfs(graph, start, goal))
```



```
Enter number of nodes (min 6): 6

Enter node connections (e.g., A B C means A connected to B and C):
A B C
B D E
C F
D
E F
F

Enter start node: A
Enter goal node: F

Graph: {'A': ['B', 'C'], 'B': ['D', 'E'], 'C': ['F'], 'D': [], 'E': ['F'], 'F': []}
BFS path: ['A', 'C', 'F']
DFS path: ['A', 'B', 'E', 'F']
```