

Practical – 1**1)DFS****Code:**

```
def depthfirstsearch(graph, start_node, goal_node):
    visited = set()
    stack = [(start_node, [start_node])]

    while stack:
        (current_node, path) = stack.pop()
        if current_node == goal_node:
            return path
        visited.add(current_node)
        for neighbor in graph[current_node]:
            if neighbor not in visited:
                stack.append((neighbor, path + [neighbor]))

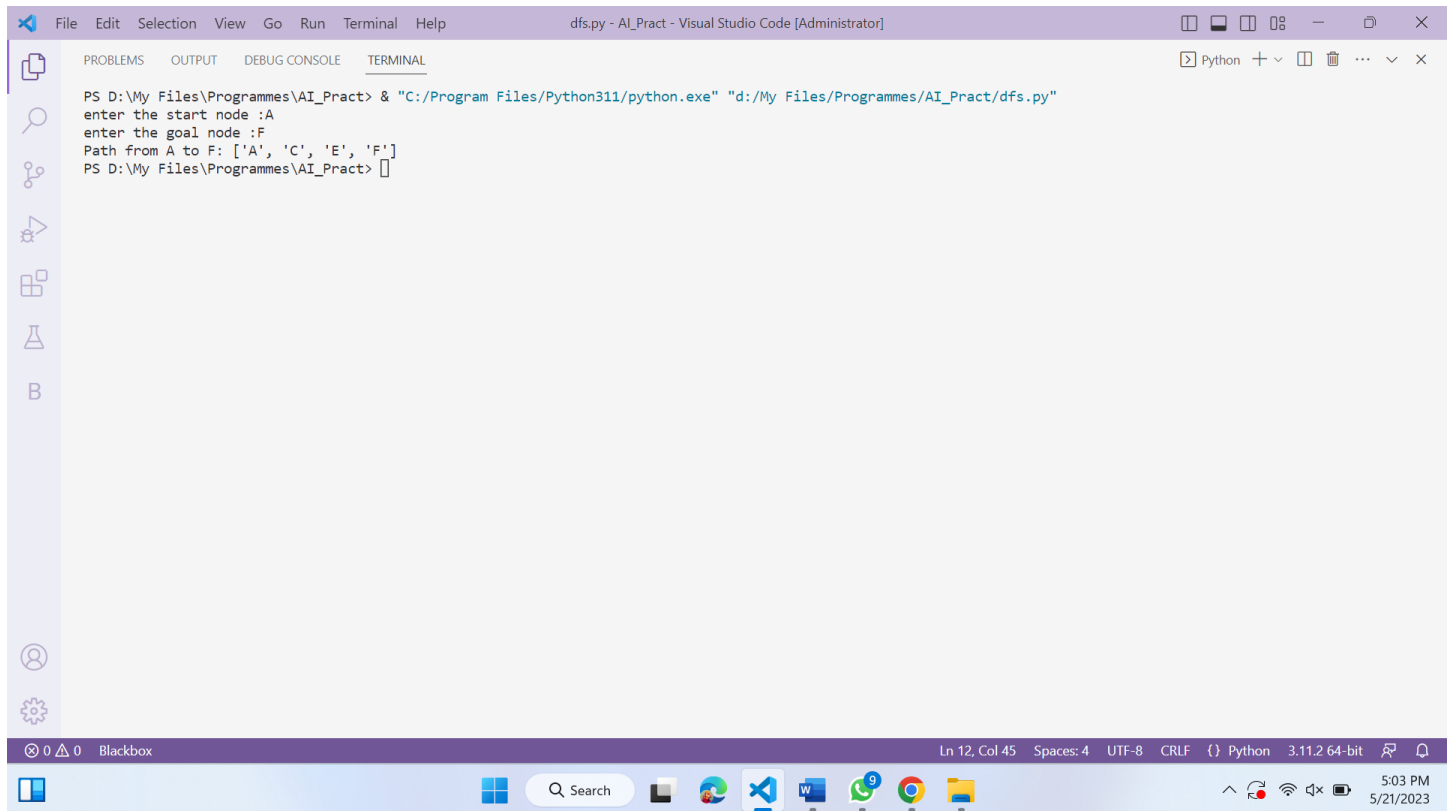
    return None

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

start_node = input("enter the start node :")
goal_node = input("enter the goal node :")

path = depthfirstsearch(graph, start_node, goal_node)
if path is not None:
    print(f"Path from {start_node} to {goal_node}: {path}")
else:
    print(f"No path found from {start_node} to {goal_node}")
```

Output:



The screenshot shows a Visual Studio Code window with a terminal open. The terminal title is "dfs.py - AI_Pract - Visual Studio Code [Administrator]". The terminal output shows the execution of a DFS algorithm. The user enters the start node 'A' and the goal node 'F'. The output shows the path from A to F as ['A', 'C', 'E', 'F']. The terminal prompt is PS D:\My Files\Programmes\AI_Pract>.

```
PS D:\My Files\Programmes\AI_Pract> & "C:/Program Files/Python311/python.exe" "d:/My Files/Programmes/AI_Pract/dfs.py"
enter the start node :A
enter the goal node :F
Path from A to F: ['A', 'C', 'E', 'F']
PS D:\My Files\Programmes\AI_Pract>
```

The status bar at the bottom of the terminal shows "Ln 12, Col 45", "Spaces: 4", "UTF-8", "CRLF", "Python", "3.11.2 64-bit", and the time "5:03 PM 5/21/2023".

2)BFS

Code:

```
from collections import deque
import graphlib

def breadthfirstsearch(graph, start_node, goal_node):
    queue = deque([(start_node, [start_node])])

    while queue:
        current_node, path = queue.popleft()
        print(f"Exploring node {current_node}: Explored nodes so far: {path}")
        if current_node == goal_node:
            return path
        for neighbor in graph[current_node]:
            if neighbor not in path:
                queue.append((neighbor, path + [neighbor]))
    return None

print(graphlib)
start_node = input("Enter Start Node: ")
goal_node = input("Enter Goal Node: ")

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

path = breadthfirstsearch(graph, start_node, goal_node)
if path is not None:
    print(f"Path from {start_node} to {goal_node}: {path}")
else:
    print(f"No path found from {start_node} to {goal_node}")
```

Output:

The image shows a Visual Studio Code editor window with a Python script for a Breadth-First Search (BFS) algorithm. The script is named 'bfs (1).py' and is located in the directory 'D:\My Files\Programmes\AI_Pract'. The script defines a graph with nodes A, B, C, D, E, and F, and their connections. It then implements a BFS function to find the shortest path from node A to node F. The output of the script is displayed in the terminal window, showing the explored nodes for each step and the final path from A to F: ['A', 'B', 'D', 'F']. The Visual Studio Code interface includes a sidebar with icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The bottom status bar shows the file name 'bfs (1).py', the editor language 'Python', and the version '3.11.2 64-bit'.