Worksheet 1.2

Student Name: Rydem
UID: 25MCI10275
Branch: MCA (AI & ML)
Section/Group: MAM-1 (A)
Semester: I
Date of Performance: 25/08/25
Subject Name: Artificial Intelligence Tools - I
Subject Code: 25CAP-614

1. Aim/Overview of the practical:

Experiment-2:
Implementation of Depth First Search

2(A). Coding:

```
import networkx as nx
import matplotlib.pyplot as plt

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

G = nx.DiGraph(graph)
pos = nx.spring_layout(G)
```

```
nx.draw(G,pos,with_labels=True, node_color='skyblue', node_size=2000, font_size=16, font_weight='bold', arrowsize=20)

plt.show()

def depth_first_search(graph, start_node):
    visited = set()

    def dfs(node):
        visited.add(node)
        print(node, end=" ")

        for neighbour in graph[node]:
            if neighbour not in visited:
                dfs(neighbour)

    dfs(start_node)

start_node = 'A'
depth_first_search(graph, start_node)
```

OUTPUT:-

```
PS E:\CLASS CODING> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe "e:/CLASS CODING/AI/dfs1.py"
A B D E F C
PS E:\CLASS CODING>
```

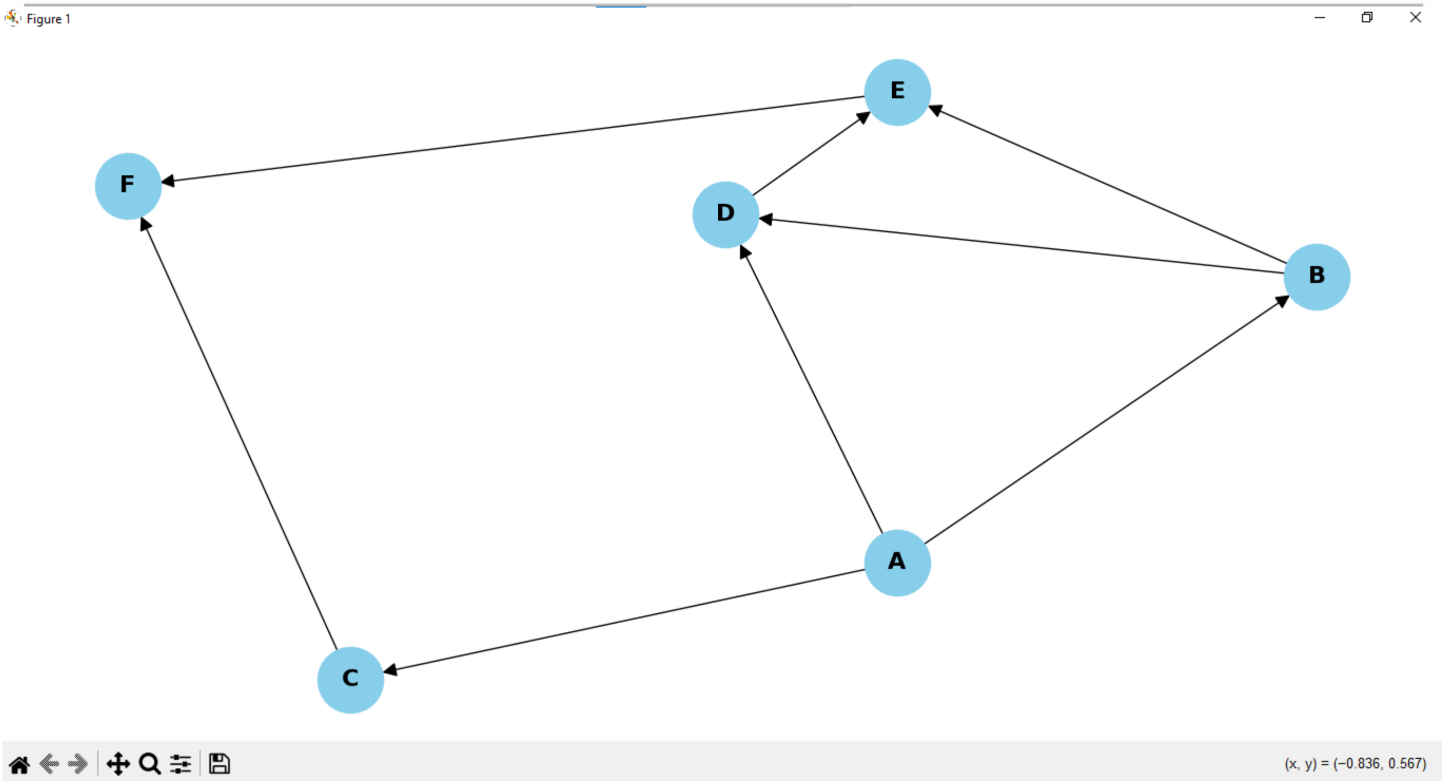


Figure 1
E
F
D
B
A
C
(x, y) = (−0.836, 0.567)

(B). CODING:

```
import networkx as nx
import matplotlib.pyplot as plt

graph ={
   'A': ['B','G'],
   'B': ['A','C','E'],
   'C': ['B','D'],
   'D': ['C','J','I'],
   'E': ['B','F','I'],
   'F': ['G','E'],
   'G': ['A','F','H','L'],
   'H': ['G','I'],
   'I':['D','E','H','K'],
   'J':['D','M','N'],
   'K':['I','L','M'],
   'L':['G','K','M','O'],
   'M':['L','K','J'],
   'N':['J','O'],
   'O':['N','L']
}

G = nx.Graph(graph)
pos = nx.spring_layout(G)

nx.draw(G,pos,with_labels=True, node_color='skyblue', node_size=2000, font_size=16, font_weight='bold',
arrowsize=20)

plt.show()

def depth_first_search(graph, start_node):
   visited = set()

   def dfs(node):
      visited.add(node)
      print(node, end=" ")
```

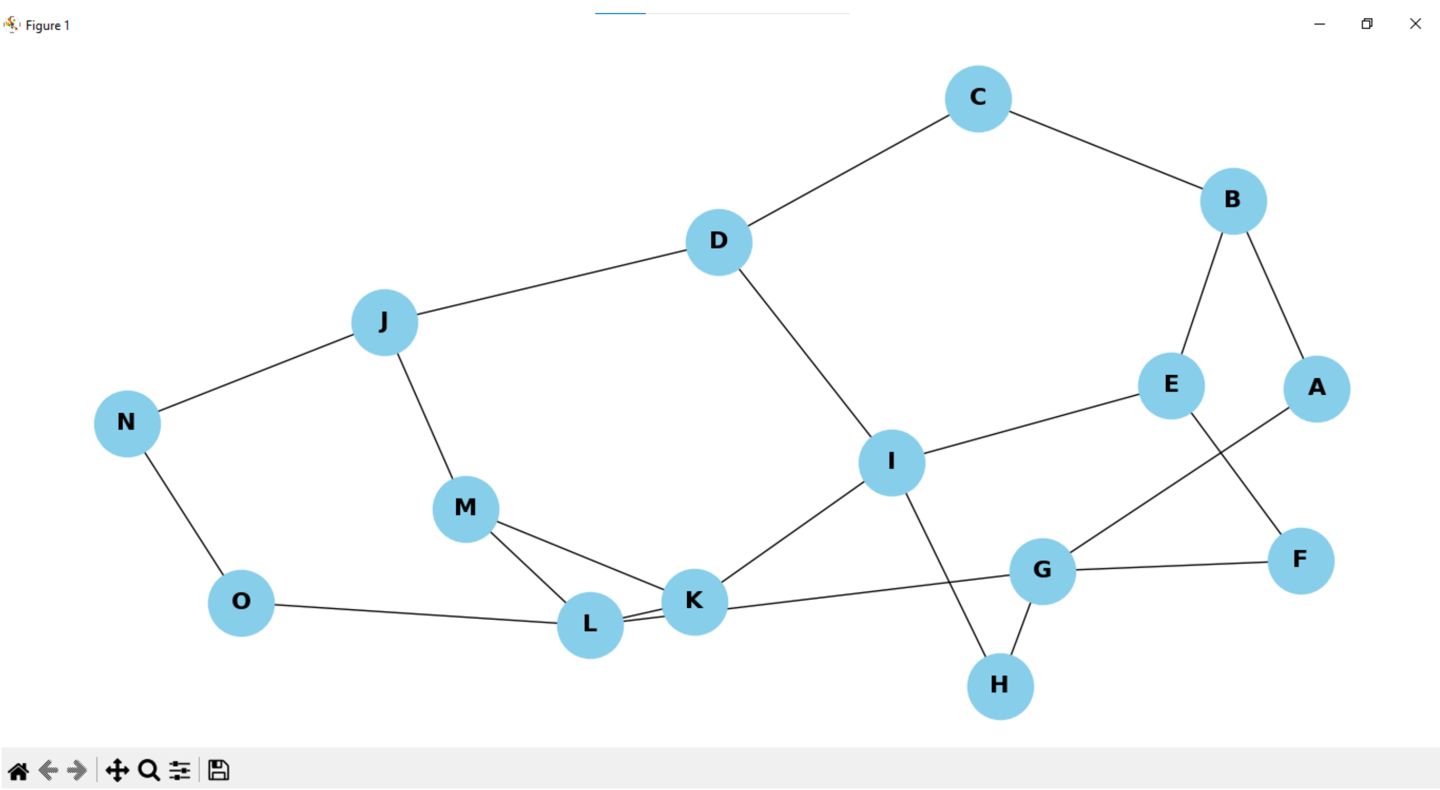
```
        for neighbour in graph[node]:
            if neighbour not in visited:
                dfs(neighbour)

    dfs(start_node)

start_node = 'O'
depth_first_search(graph, start_node)
```

(B). OUTPUT:

```
PS E:\CLASS CODING> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe "e:/CLASS CODING/AI/dfs2.py"
O N J D C B A G F E I H K L M
PS E:\CLASS CODING>
```



3. Learning outcomes (What I have learnt):

- Understand the working principle of the Depth First Search (DFS) algorithm.
- Learn how to implement DFS using recursion and stack-based approaches.
- Analyze the time and space complexity of DFS.
- Compare DFS with other graph traversal algorithms such as Breadth First Search (BFS).
- Develop problem-solving skills in applications like pathfinding, cycle detection, and topological sorting.