Experiment No. 1.2

Student Name: **Vikash Kumar** UID: **21BCS9899**
Branch: **CSE** Section/Group: **IoT 640 – 'B'**
Semester: **5th** Date of Performance: **/08/2023**
Subject Name: **AIML Lab** Subject Code: **21CSP-316**

1. Aim:

Implement DFS algorithm and analyse its performance and characteristics.

2. Objective:

The objective of this experiment is to implement the Depth-First Search (DFS) algorithm and analyze its performance and characteristics.

3. Input/Tools Used:

PC, Python Programming Language, A* implementation, Problem Scenario for testing the algorithm

4. Theory:

DFS algorithm in the data structure. It is a recursive algorithm to search all the vertices of a tree data structure or a graph. The depth-first search (DFS) algorithm starts with the initial node of graph G and goes deeper until we find the goal node or the node with no children.

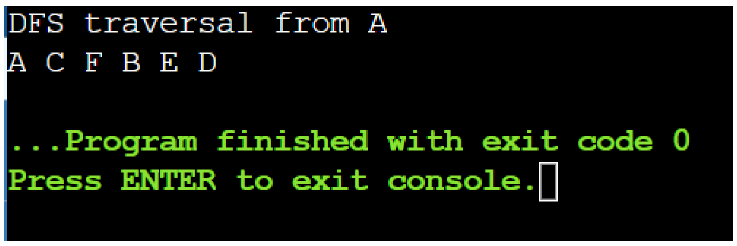
5. Procedure/Algorithm:

1. Start the DFS traversal from a specified start node in the graph.
2. Create an empty set called visited to keep track of visited nodes.
3. Create a stack and push the start node onto it.
4. While the stack is not empty, a. Pop a node from the stack. Assign it to the variable node. b. If the node is not in the visited set, do the following steps:
   Print the node to show that it has been visited. Add the node to the visited set.
5. Check if the node is present in the graph dictionary.
   If the neighbor is not already visited, push it onto the stack.
6. The DFS traversal is complete when the stack becomes empty.

6. Source Code:

```
def dfs(graph, start):
    visited = set()
    stack = [start]

    while stack:
        node = stack.pop()

        if node not in visited:
            print(node, end=' ')
            visited.add(node)
            neighbors = graph[node] if node in graph else []
            for neighbor in neighbors:
                if neighbor not in visited:
                    stack.append(neighbor)


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

start_node = 'A'
print("DFS traversal from", start_node)
dfs(graph, start_node)
```

7. Output:



```
DFS traversal from A
A C F B E D

...Program finished with exit code 0
Press ENTER to exit console.
```