Experiment: 1.2

Student Name: Nishant Kumar Mehta **UID:** 21BCS-*3402*
Branch: CSE **Section/Group:** IOT-602/B
Semester: 5^{th} **Date:** 21/08/2023
Subject Name: AIML Lab **Subject Code:** 21CSH-316

1. **AIM:** Implement the DFS algorithm and analyze its performance and characteristics

2. **Tools/Resource Used:**
   - Graph: The DFS algorithm requires a graph as input, which can be represented using an adjacency list or adjacency matrix..

3. **Algorithm:**
   - We will start by putting any one of the graph's vertex on top of the stack.
   - After that take the top item of the stack and add it to the visited list of the vertex.
   - Next, create a list of that adjacent node of the vertex. Add the ones which aren't in the visited list of vertexes to the top of the stack.
   - Lastly, keep repeating steps 2 and 3 until the stack is empty.

4. **Program Code:**

```
def dfs(graph, start, visited):
    visited.add(start)
    print(start, end = ' ')

    for neighbor in graph[start]:
        if neighbor not in visited:
            dfs(graph, neighbor, visited)

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

visited = set()

dfs(graph, 'A', visited)
```

5. **Output/Result:**

```
A B D E F C
PS C:\Users\NI$HANT\OneDrive\Documents\
GitHub\DSA-ALPHA>
```

6. Learning Outcomes:

- The time complexity of DFS is O(|V| + |E|), where |V| is the number of vertices and |E| is the number of edges in the graph. In the worst case, DFS visits all vertices and edges once.
- DFS guarantees to visit all vertices in a connected graph. However, if the graph is disconnected, multiple DFS calls are required to visit all vertices.
- The space complexity of DFS is O(|V|) in the worst case, as it requires a stack or recursive calls to keep track of the vertices to visit.