Question:8

DEPTH FIRST SEARCH

AIM

To implement Depth First Search algorithm using Python

ALGORITHM

1. Import the `deque` class from the `collections` module to use a double-ended queue.
2. Define the `dfs` function that takes two parameters: the graph as a dictionary and the starting node.
3. Initialize an empty set called `visited` to keep track of visited nodes.
4. Initialize a stack (implemented as a list) with the starting node `start`, and mark it as visited.
5. While the stack is not empty:
   a. Pop a node `node` from the top of the stack.
   b. Print the popped node (or perform any desired operation on it).
   c. Iterate over the neighbors of the popped node:
      - If a neighbor `neighbor` has not been visited:
        - Mark it as visited.
        - Push the neighbor onto the stack.
6. If there are no more nodes in the stack, the DFS traversal is complete.
7. In the `if __name__ == "__main__":` block:
   a. Define a graph as a dictionary where the keys represent nodes and the values represent their neighbors.
   b. Print a message indicating the start of DFS traversal.
   c. Call the `dfs` function with the defined graph and the starting node `'A'`.

CODE

```
def dfs(graph, start):
    visited = set()
    stack = [start]  # Use a list as a stack
    visited.add(start)
    while stack:
        node = stack.pop()
        print(node, end=" ")
        for neighbor in reversed(graph[node]):
            if neighbor not in visited:
```

```
            visited.add(neighbor)
            stack.append(neighbor)

if __name__ == "__main__":
    graph = {
        'A': ['B', 'C'],
        'B': ['A', 'D', 'E'],
        'C': ['A', 'F'],
        'D': ['B'],
        'E': ['B', 'F'],
        'F': ['C', 'E']
    }
    print("DFS Traversal:")
    dfs(graph, 'A')
```

OUTPUT

```
DFS Traversal:
A B D E F C
```