PROGRAM-8

DFS PROBLEM

AIM:-

To write and execute the python program for the DFS program.

PROCEDURE:-

- **Class Definition:**
  - Define the Graph class, which represents an undirected graph.
  - The constructor initializes an empty dictionary graph to store the graph data.
- **Add Edge Function**:
  - Define the add_edge method to add edges between nodes in the graph.
  - If the node is not already present in the graph, add it with an empty list of neighbors.
  - Append the neighbor to the list of neighbors for the given node.
- **Depth-First Search (DFS):**
  - Define the dfs_util method to perform the DFS traversal recursively.
  - Mark the current node as visited and print its value.
  - Recursively visit each unvisited neighbor of the current node.
- **DFS Traversal Function**:
  - Define the dfs method to initiate the DFS traversal from a specified start node.
  - Create an empty set visited to keep track of visited nodes.
  - Call the dfs_util method to perform the DFS traversal from the start node.
- **Example Usage**:
  - Create an instance of the Graph class.
  - Add edges between nodes using the add_edge method.
  - Call the dfs method to perform the DFS traversal starting from a specified node

CODING:-

```
class Graph:

    def __init__(self):
```

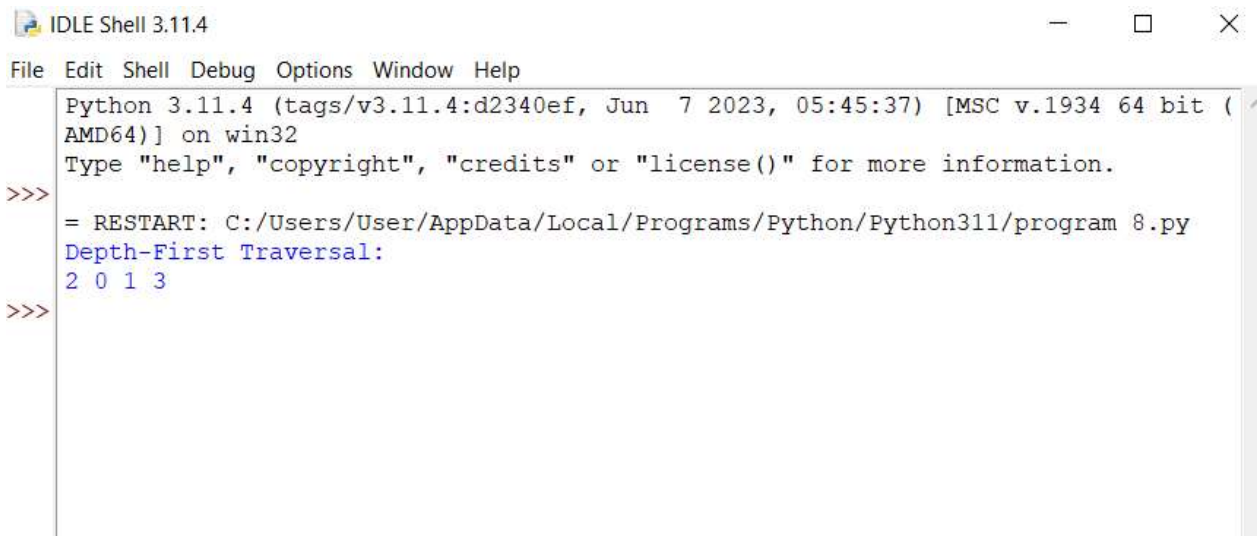
```
        self.graph = {}
    def add_edge(self, node, neighbor):
        if node not in self.graph:
            self.graph[node] = []
        self.graph[node].append(neighbor)
    def dfs_util(self, node, visited):
        visited.add(node)
        print(node, end=" ")
        for neighbor in self.graph.get(node, []):
            if neighbor not in visited:
                self.dfs_util(neighbor, visited)
    def dfs(self, start_node):
        visited = set()
        self.dfs_util(start_node, visited)
# Example usage:
graph = Graph()
graph.add_edge(0, 1)
graph.add_edge(0, 2)
graph.add_edge(1, 2)
graph.add_edge(2, 0)
graph.add_edge(2, 3)
```

```
graph.add_edge(3, 3)

print("Depth-First Traversal:")

graph.dfs(2)  # Starting DFS traversal from node 2
```

OUTPUT:-



RESULT:-

Hence the program has been successfully executed and verified.