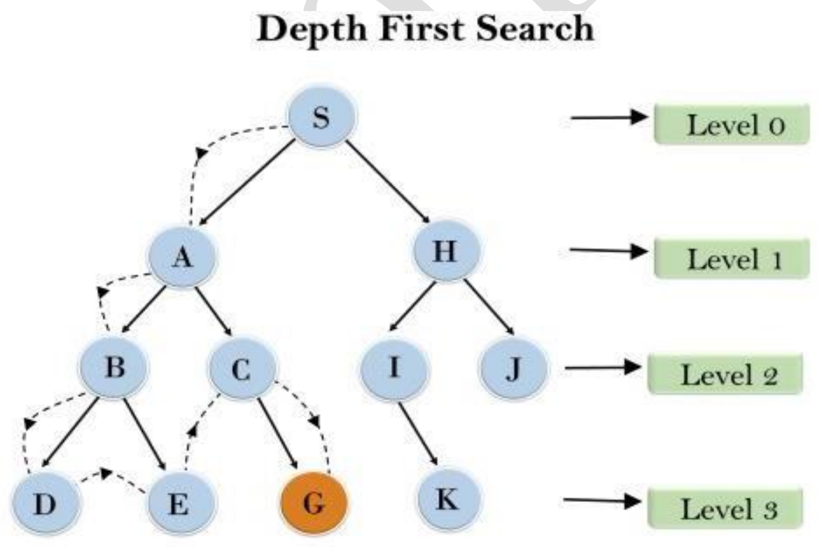
EX.NO: 02 **DATE:1 6 . 0 8 . 2 0 2 4**

R e g . n o : 2 2 0 7 0 1 0 5 4

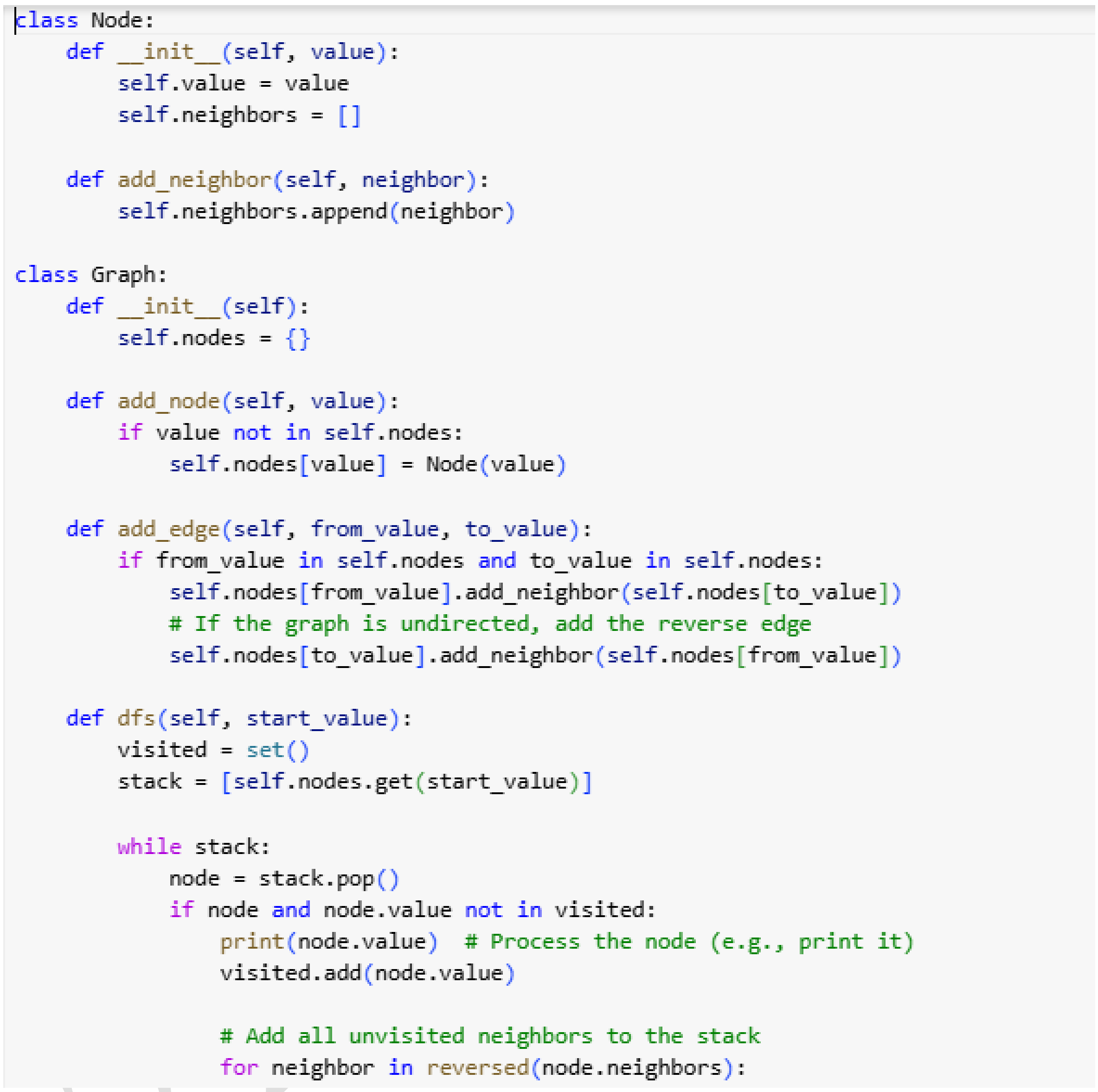
DEPTH-FIRST SEARCH

AIM:
To implement a depth-first search problem using Python.

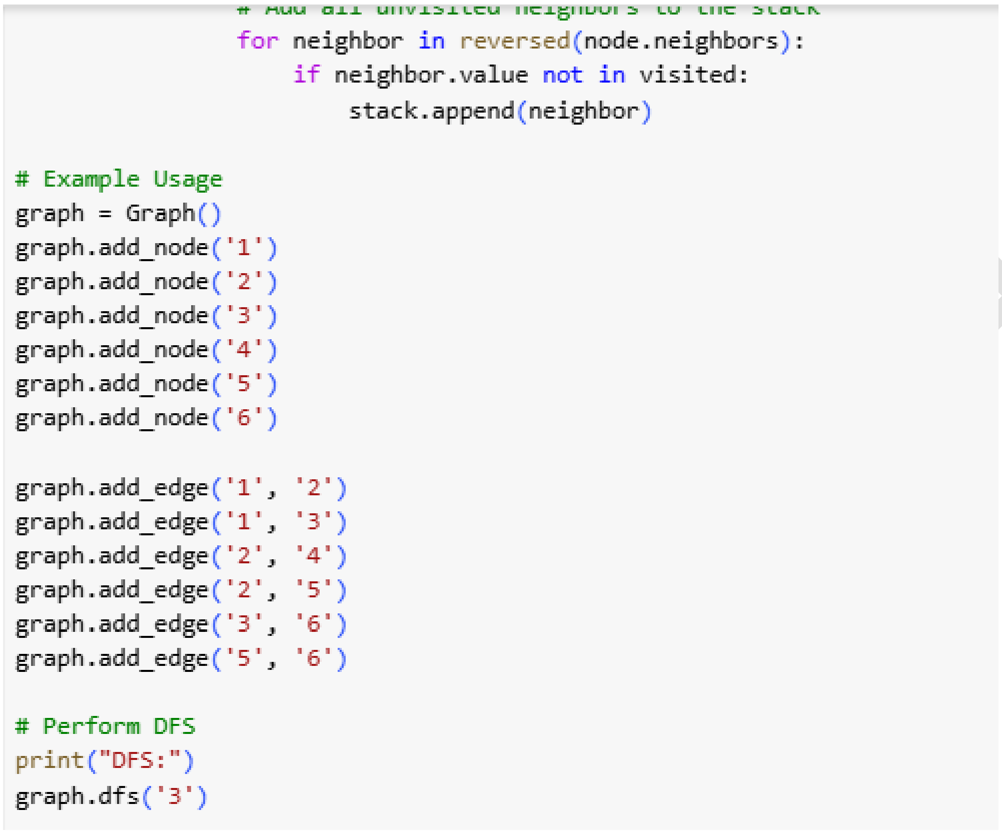
- Depth-first search (DFS) algorithm or searching technique starts with the root node of graph G, and then travel deeper and deeper until we find the goal node or the node which has no children by visiting different node of the tree.
- The algorithm, then backtracks or returns back from the dead end or last node towards the most recent node that is yet to be completely unexplored.
- The data structure (DS) which is being used in DFS Depth-first search is stack. The process is quite similar to the BFS algorithm.
- In DFS, the edges that go to an unvisited node are called discovery edges while the edges that go to an already visited node are called block edges.



Code:

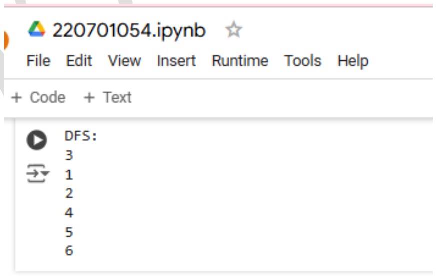


```
class Node:
    def __init__(self, value):
        self.value = value
        self.neighbors = []

    def add_neighbor(self, neighbor):
        self.neighbors.append(neighbor)

class Graph:
    def __init__(self):
        self.nodes = {}

    def add_node(self, value):
        if value not in self.nodes:
            self.nodes[value] = Node(value)

    def add_edge(self, from_value, to_value):
        if from_value in self.nodes and to_value in self.nodes:
            self.nodes[from_value].add_neighbor(self.nodes[to_value])
            # If the graph is undirected, add the reverse edge
            self.nodes[to_value].add_neighbor(self.nodes[from_value])

    def dfs(self, start_value):
        visited = set()
        stack = [self.nodes.get(start_value)]

        while stack:
            node = stack.pop()
            if node and node.value not in visited:
                print(node.value)  # Process the node (e.g., print it)
                visited.add(node.value)

                # Add all unvisited neighbors to the stack
                for neighbor in reversed(node.neighbors):
```



```
                # Add all unvisited neighbors to the stack
                for neighbor in reversed(node.neighbors):
                    if neighbor.value not in visited:
                        stack.append(neighbor)

# Example Usage
graph = Graph()
graph.add_node('1')
graph.add_node('2')
graph.add_node('3')
graph.add_node('4')
graph.add_node('5')
graph.add_node('6')

graph.add_edge('1', '2')
graph.add_edge('1', '3')
graph.add_edge('2', '4')
graph.add_edge('2', '5')
graph.add_edge('3', '6')
graph.add_edge('5', '6')

# Perform DFS
print("DFS:")
graph.dfs('3')
```

Output:



```
DFS:
3
1
2
4
5
6
```

Result:

Thus the python code for depth first search is executed successfully.