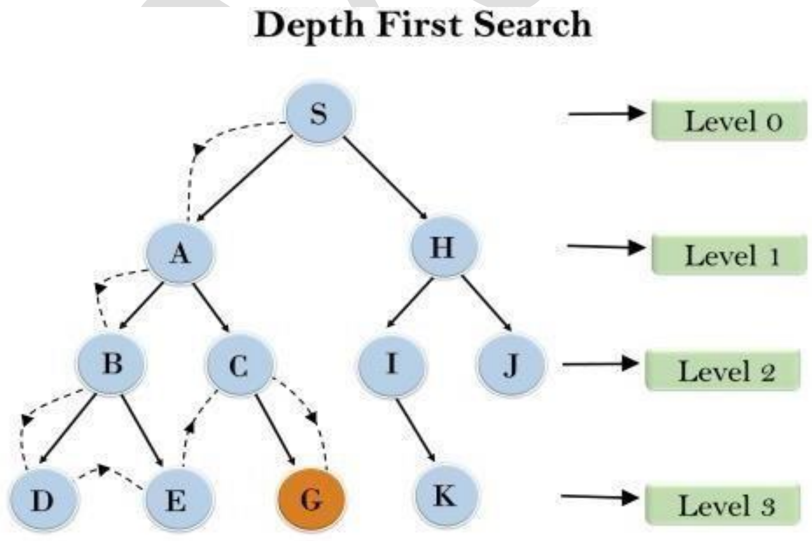
EX.NO:2 **DATE:4/9/2024**
Reg.no:220701030

DEPTH-FIRST SEARCH

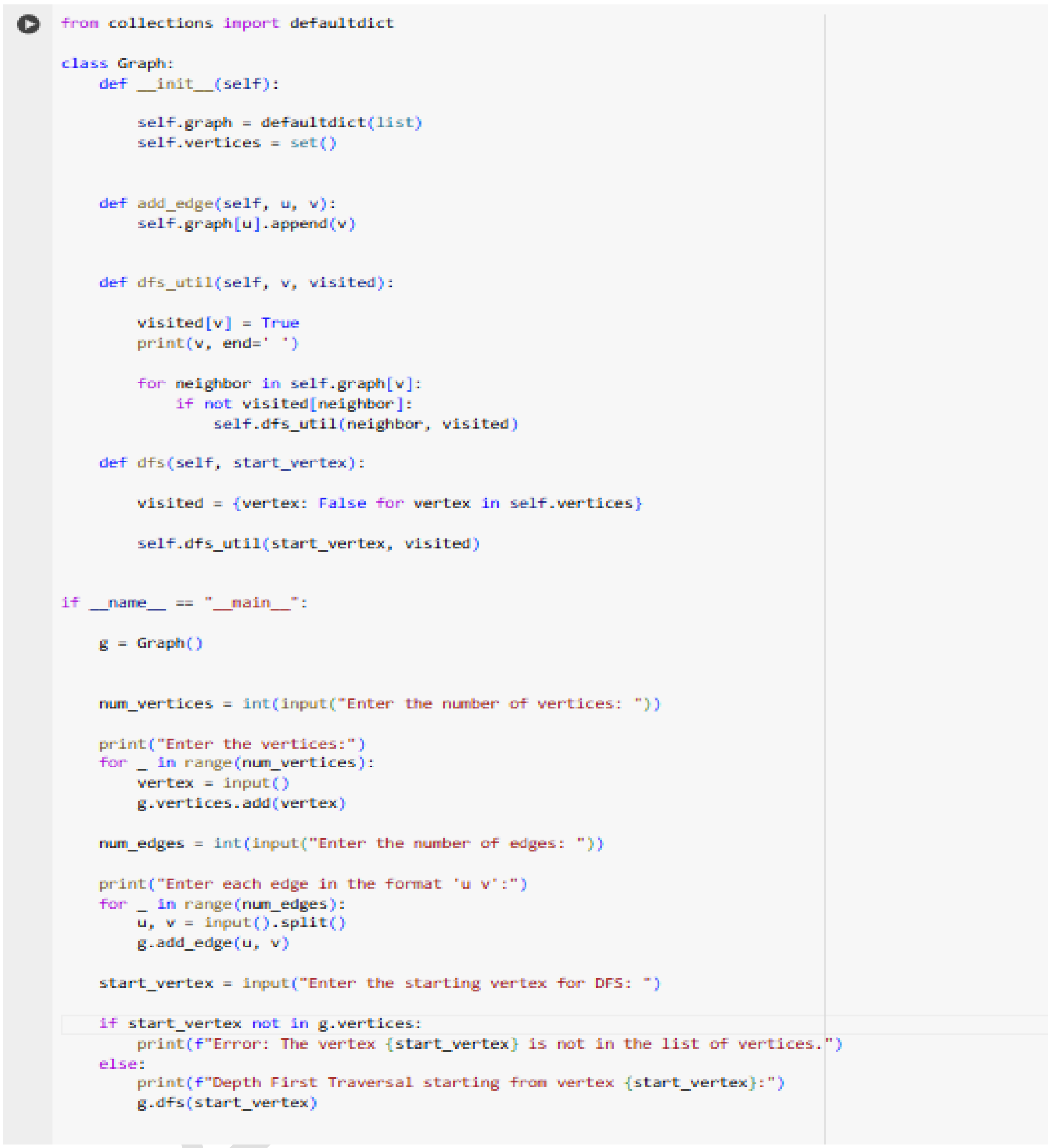
AIM:
To implement a depth-first search problem using Python

- 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.



Reg.no:220701030

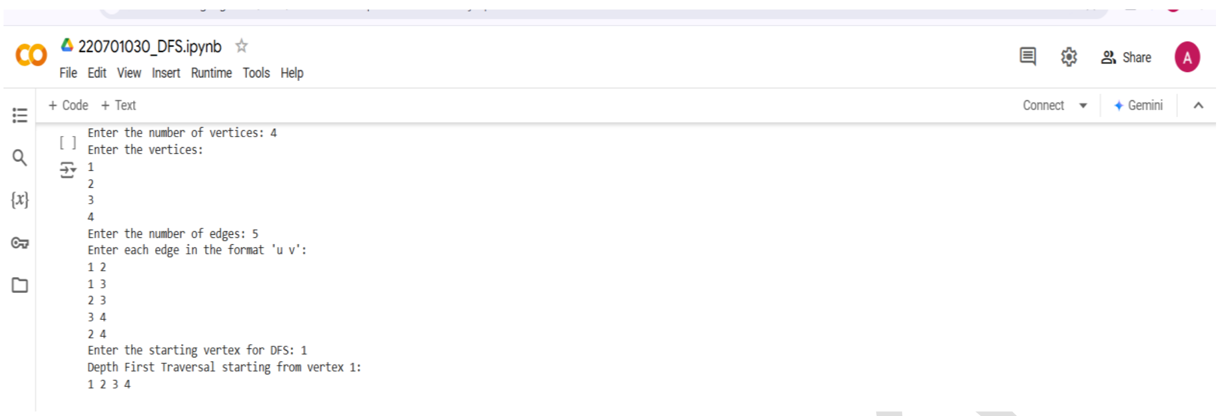
CODE:



```
from collections import defaultdict

class Graph:
    def __init__(self):

        self.graph = defaultdict(list)
        self.vertices = set()


    def add_edge(self, u, v):
        self.graph[u].append(v)


    def dfs_util(self, v, visited):

        visited[v] = True
        print(v, end=' ')

        for neighbor in self.graph[v]:
            if not visited[neighbor]:
                self.dfs_util(neighbor, visited)

    def dfs(self, start_vertex):

        visited = {vertex: False for vertex in self.vertices}

        self.dfs_util(start_vertex, visited)


if __name__ == "__main__":

    g = Graph()


    num_vertices = int(input("Enter the number of vertices: "))

    print("Enter the vertices:")
    for _ in range(num_vertices):
        vertex = input()
        g.vertices.add(vertex)

    num_edges = int(input("Enter the number of edges: "))

    print("Enter each edge in the format 'u v':")
    for _ in range(num_edges):
        u, v = input().split()
        g.add_edge(u, v)

    start_vertex = input("Enter the starting vertex for DFS: ")

    if start_vertex not in g.vertices:
        print(f"Error: The vertex {start_vertex} is not in the list of vertices.")
    else:
        print(f"Depth First Traversal starting from vertex {start_vertex}:")
        g.dfs(start_vertex)
```

Reg.no:220701030

OUTPUT:



```
Enter the number of vertices: 4
Enter the vertices:
1
2
3
4
Enter the number of edges: 5
Enter each edge in the format 'u v':
1 2
1 3
2 3
3 4
2 4
Enter the starting vertex for DFS: 1
Depth First Traversal starting from vertex 1:
1 2 3 4
```

RESULT: Thus, the Depth First Search program has been implemented successfully.