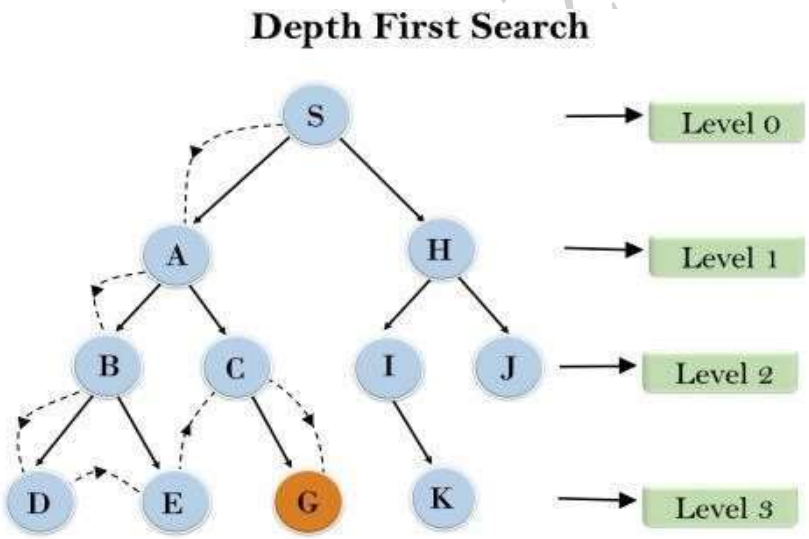
EX.NO: **DATE:**

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.

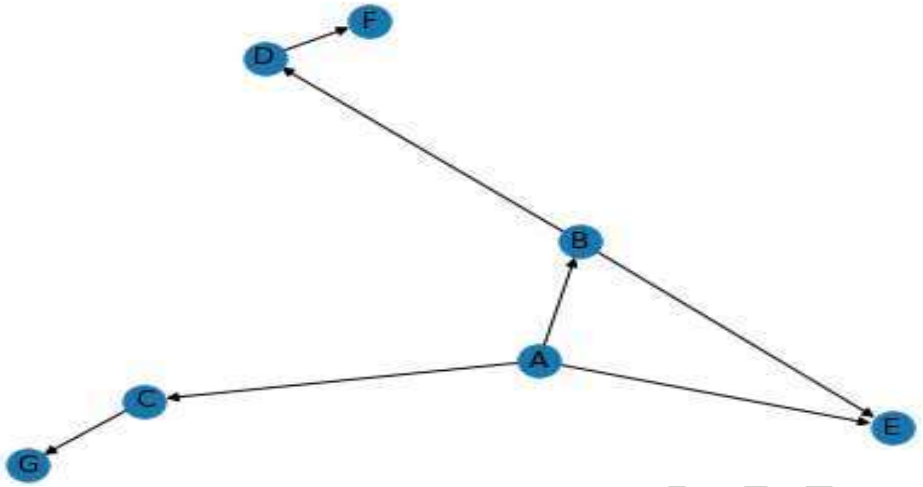


CODE:

```
class Graph:
    def __init__(self, vertices):
        self.vertices = vertices
        self.graph = {i: [] for i in range(vertices)}

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

    def dfs(self, start):
        visited = [False] * self.vertices
        stack = []
        stack.append(start)

        while stack:
            node = stack.pop()

            if not visited[node]:
                print(node, end=" ")
                visited[node] = True

            for neighbor in reversed(self.graph[node]):
                if not visited[neighbor]:
                    stack.append(neighbor)

if __name__ == "__main__":
    g = Graph(5)
    g.add_edge(0, 1)
    g.add_edge(0, 2)
    g.add_edge(1, 3)
    g.add_edge(1, 4)

    print("DFS Traversal starting from vertex 0:")
    g.dfs(0)
```

OUTPUT:

```
Following is DFS from (starting from vertex A)
A B D F E C G
```



RESULT:

Thus, the code has been successfully executed, and the output has been verified successfully.