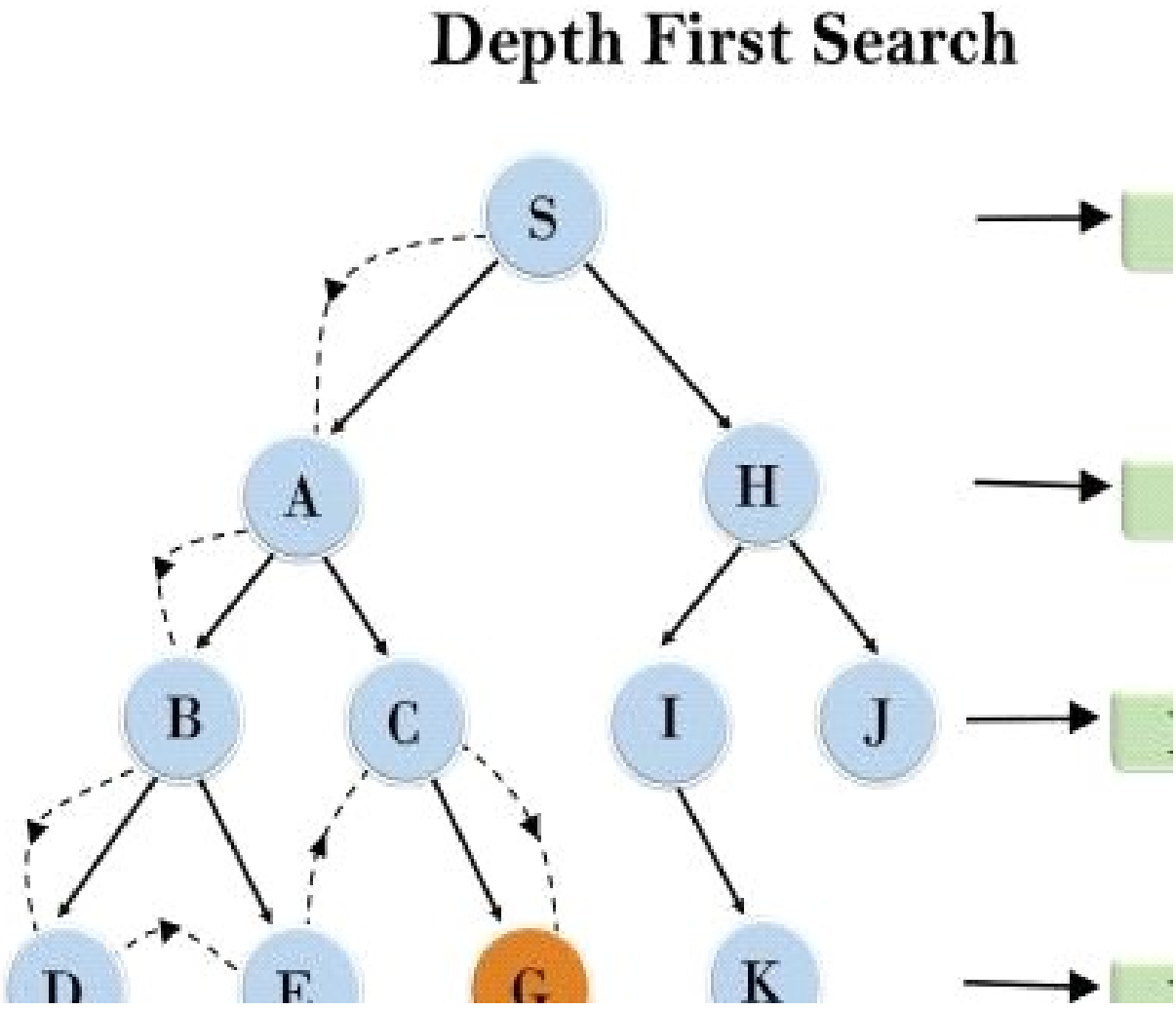
EX.NO: 2

DATE:

DEPTH FIRST SEARCH

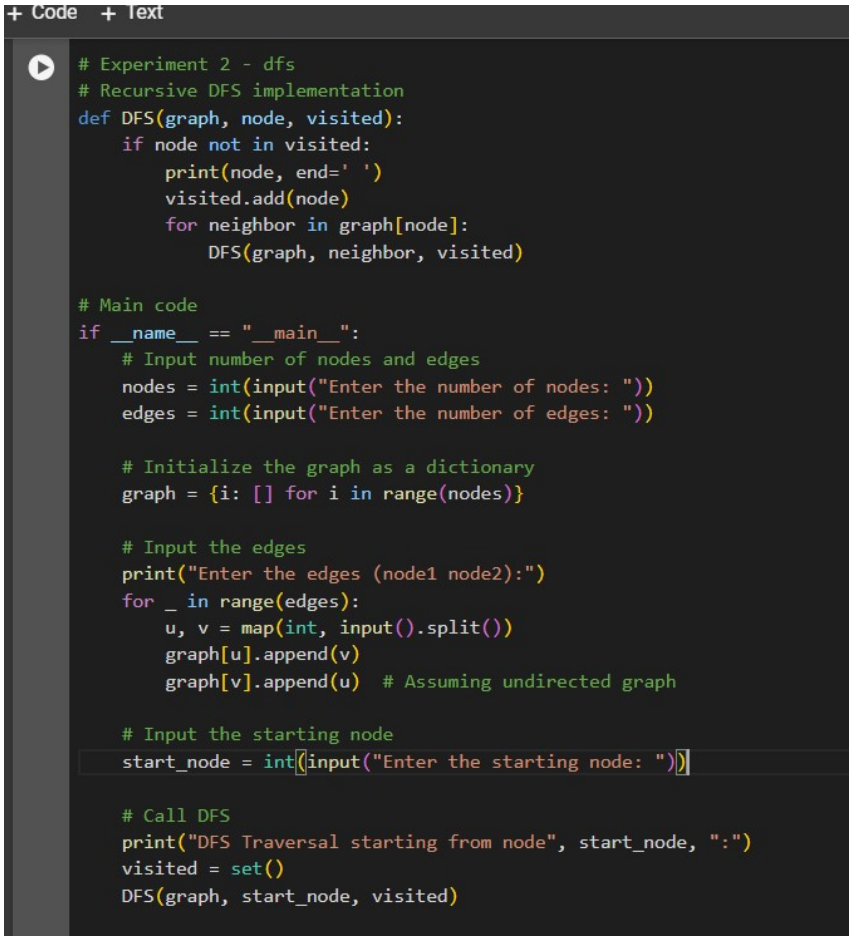
- Depth first search (DFS) algorithm or searching technique starts with the root node of the graph G, and then travel to deeper and deeper until we find the goal node or the node which has no children by visiting different node of the tree.
- The algorithm, then backtracks or return back from the dead 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 BFS algorithm.
- In DFS, the edges that goes to an unvisited node are called discovery edges while the edges that goes to an already visited node are called block edges.



AIM :

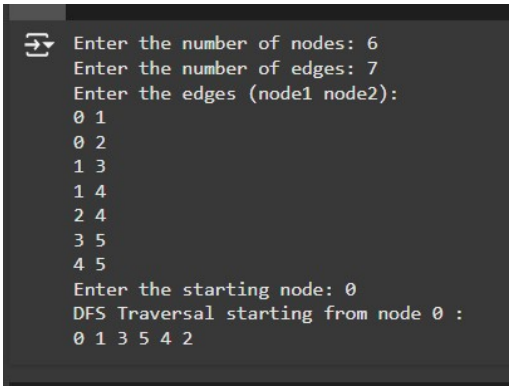
To implement a depth first search problem using python.

CODE :



```
+ Code + Text

# Experiment 2 - dfs
# Recursive DFS implementation
def DFS(graph, node, visited):
    if node not in visited:
        print(node, end=' ')
        visited.add(node)
        for neighbor in graph[node]:
            DFS(graph, neighbor, visited)

# Main code
if __name__ == "__main__":
    # Input number of nodes and edges
    nodes = int(input("Enter the number of nodes: "))
    edges = int(input("Enter the number of edges: "))

    # Initialize the graph as a dictionary
    graph = {i: [] for i in range(nodes)}

    # Input the edges
    print("Enter the edges (node1 node2):")
    for _ in range(edges):
        u, v = map(int, input().split())
        graph[u].append(v)
        graph[v].append(u)  # Assuming undirected graph

    # Input the starting node
    start_node = int(input("Enter the starting node: "))

    # Call DFS
    print("DFS Traversal starting from node", start_node, ":")
    visited = set()
    DFS(graph, start_node, visited)
```

OUTPUT :



RESULT :

Thus the program is suceesfully executed and output is verified