

EX.NO:2

DATE:4/9/2024

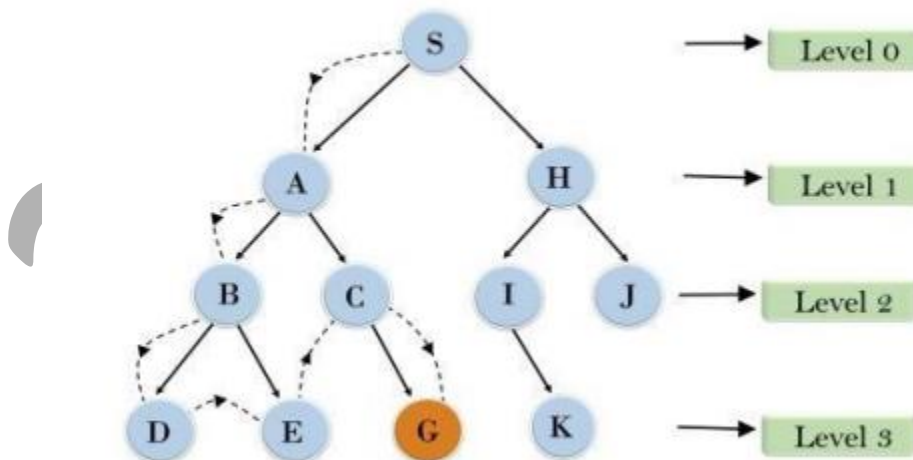
Reg.no:220701064

## 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

### Depth First Search



### CODE:

```
def dfs_recursive(graph, start, visited=None):
```

```

    if visited is None:
        visited = set()
    visited.add(start)
    print(start)

    for neighbor in graph[start]:
        if neighbor not in visited:
            dfs_recursive(graph, neighbor, visited)

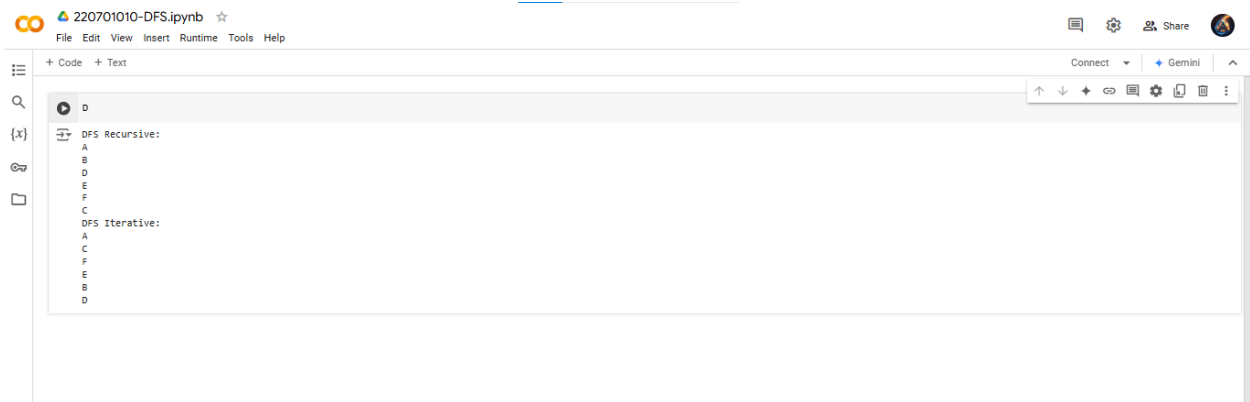
graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F'],
    'F': ['C', 'E']
}
print("DFS Recursive:")
dfs_recursive(graph, 'A')
def dfs_iterative(graph, start):
    visited = set()
    stack = [start]

    while stack:
        vertex = stack.pop()
        if vertex not in visited:
            print(vertex)
            visited.add(vertex)
            stack.extend(neighbor for neighbor in graph[vertex] if
neighbor not in visited)
graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F'],
    'F': ['C', 'E']
}

print("DFS Iterative:")
dfs_iterative(graph, 'A')

```

## OUTPUT:



The image shows a Jupyter Notebook interface with a single code cell. The notebook title is "220701010-DFS.ipynb". The code cell contains two blocks of text: "DFS Recursive:" followed by a list of nodes A, B, D, E, F, C, and "DFS Iterative:" followed by a list of nodes A, C, F, E, B, D. The interface includes a top menu bar with options like File, Edit, View, Insert, Runtime, Tools, and Help. On the left, there is a sidebar with icons for file explorer, search, and other notebook functions. On the right, there is a toolbar with icons for running code, saving, and other actions.

```
D
DFS Recursive:
A
B
D
E
F
C
DFS Iterative:
A
C
F
E
B
D
```