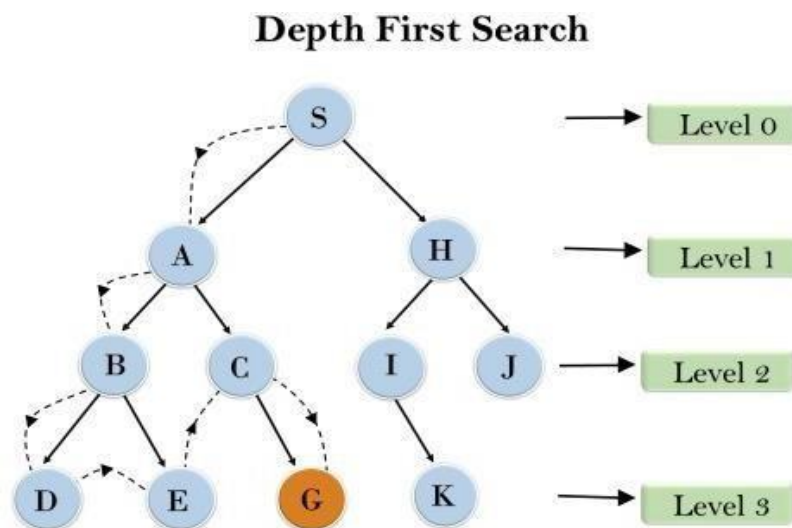


DEPTH-FIRST SEARCH

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:

```
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')

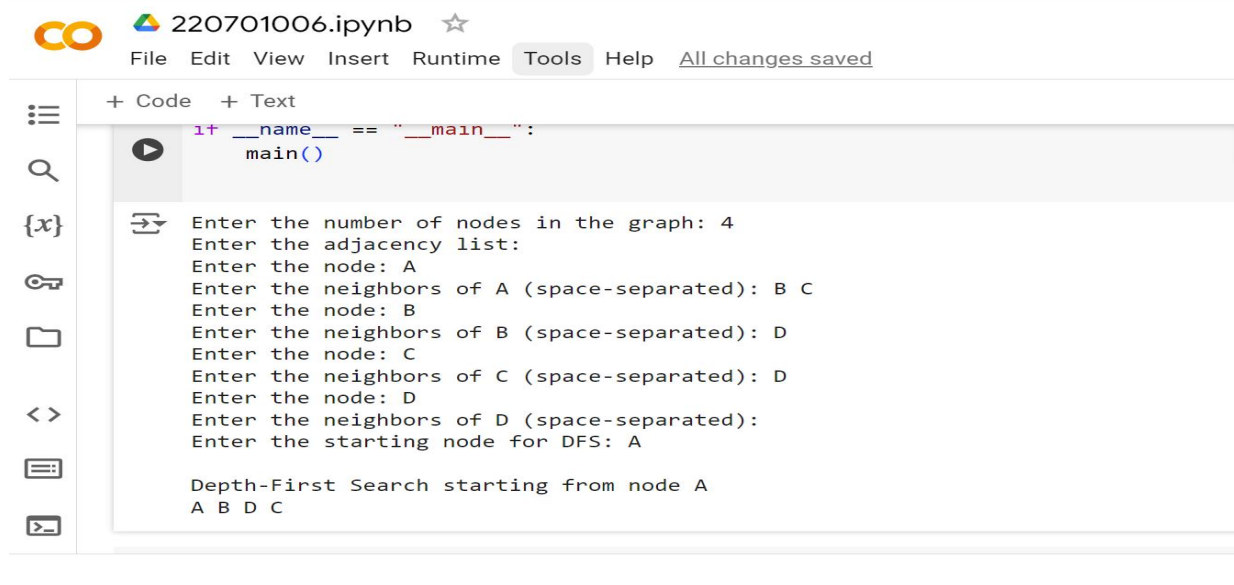
def dfs_iterative(graph, start):
    visited = set()
    stack = [start]

    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 screenshot shows a Jupyter Notebook titled "220701006.ipynb". The interface includes a top bar with the Jupyter logo, the file name, a star icon, and a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help, and a link "All changes saved". Below the menu bar, there are tabs for "+ Code" and "+ Text". The left sidebar contains icons for file management (three horizontal lines, search, {x}, key, folder, <>, list, and a terminal icon). The main area displays a code cell with the following Python code:

```
if __name__ == "__main__":  
    main()
```

Below the code cell, the output is shown, consisting of a series of prompts and user inputs for a Depth-First Search (DFS) algorithm:

```
Enter the number of nodes in the graph: 4  
Enter the adjacency list:  
Enter the node: A  
Enter the neighbors of A (space-separated): B C  
Enter the node: B  
Enter the neighbors of B (space-separated): D  
Enter the node: C  
Enter the neighbors of C (space-separated): D  
Enter the node: D  
Enter the neighbors of D (space-separated):  
Enter the starting node for DFS: A  
  
Depth-First Search starting from node A  
A B D C
```

RESULT:

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