**Name: Usman Ul Haq**

**Roll# BSAIM-035**

**AI LAB TASKs**

**Documentation**

**Documentation of the Program DFS with Stack & Node**

This program implements Depth-First Search (DFS) using a stack for traversing a graph. It allows adding edges between nodes and then performs a DFS traversal starting from a given node.

Functioning of the Program

1. Defining the Node Class (Not Used in Graph)
   * A Node class is defined for binary tree structures, but it is not directly used in the graph implementation.
2. Graph Representation:
   * The Graph class is implemented using an adjacency list stored in a dictionary.
   * Each node maps to a list of its neighbors.
3. Adding Edges:
   * The add\_edge method ensures that nodes exist in the adjacency list and appends their neighbors.
4. Depth-First Search (DFS) Using a Stack:
   * The dfs\_stack method:
     + Starts from a given node.
     + Uses a stack to keep track of nodes to visit.
     + Uses a set to track visited nodes.
     + Reverses neighbors before adding them to the stack, ensuring proper traversal order.
     + Prints nodes as they are visited.
5. Building and Traversing the Graph:
   * The program adds edges between nodes to create the following graph structure:

mathematica

CopyEdit

A

/ \

B C

/ \ / \

D E F G

* + DFS traversal starting from 'A' will visit nodes in the order:

mathematica

CopyEdit

A B D E C F G

1. Output:

mathematica

CopyEdit

DFS Traversal using Stack:

A B D E C F G

Conclusion

This program demonstrates graph traversal using Depth-First Search (DFS) with a stack, a key algorithm used in pathfinding, AI search strategies, and network analysis.

**2. Research about "Inorder, Preorder, Postorder" and implement in DFS**

Documentation of the Program

This program implements a binary tree and performs three types of tree traversal techniques:

Inorder Traversal (Left → Root → Right)

Preorder Traversal (Root → Left → Right)

Postorder Traversal (Left → Right → Root)

Functioning of the Program

1. Defining the Binary Tree Structure

The TreeNode class is used to create nodes in the tree.

Each node has:

A value representing the data.

Two child pointers: left and right, initialized as None.

2. Tree Traversal Functions

Each traversal function is recursive, meaning it calls itself to visit nodes in a structured order.

Inorder Traversal (Left → Root → Right)

Recursively visits the left subtree.

Prints the root node.

Recursively visits the right subtree.

Preorder Traversal (Root → Left → Right)

Prints the root node.

Recursively visits the left subtree.

Recursively visits the right subtree.

Postorder Traversal (Left → Right → Root)

Recursively visits the left subtree.

Recursively visits the right subtree.

Prints the root node.

3. Constructing the Binary Tree

The program manually builds the following binary tree:

A

/ \

B C

/ \ / \

D E F G

4. Executing and Displaying the Traversals

The program calls each traversal function and prints the nodes in the respective order.

Expected Output

Inorder Traversal:

D B E A F C G

Preorder Traversal:

A B D E C F G

Postorder Traversal:

D E B F G C A

Conclusion

This program effectively demonstrates binary tree traversals, which are fundamental in data structures, searching algorithms, and hierarchical data processing.