**Documentation: DFS and Tree Traversals**

1. **Depth First Search (DFS) – Graph Traversal**

**Definition:**

**DFS is a graph traversal algorithm that starts at a given node and explores as far as possible along each branch before backtracking.**

**Steps:**

1. Start from the source node.

2. Visit the node and mark it as visited.

3. Recursively visit all unvisited neighbors.

4. Backtrack when no unvisited neighbors remain.

**Example Graph:**

0 connected with 1, 2

1 connected with 0, 3, 4

2 connected with 0

3 connected with 1

4 connected with 2, 3

**DFS Output:** 0 → 1 → 3 → 4 → 2

1. **Binary Tree Traversals Definition:**

**Tree traversal is the process of visiting each node of a tree in a specific order. The three main types are Inorder, Preorder, and Postorder.**

1. **Tree Structure Example:**

1 / \ 2 3 / \ 4 5

**Inorder Traversal (Left → Root → Right):**

**Steps:** Visit left subtree → Visit root → Visit right subtree.

**Output:** 4 2 5 1 3

**Preorder Traversal (Root → Left → Right):**

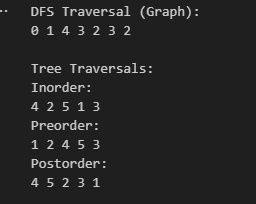
**Steps:** Visit root → Visit left subtree → Visit right subtree.

**Output:** 1 2 4 5 3

**Postorder Traversal (Left → Right → Root):**

**Steps:** Visit left subtree → Visit right subtree → Visit root.

***Output*:** 4 5 2 3 1

****

**Documentation: DFS with Stack (Graph & Tree)**

1. **DFS with Stack (Graph)**

**Definition:**

**Depth First Search (DFS) using a stack is an iterative method to traverse a graph. Instead of recursion, a stack is used to keep track of nodes to visit next.**

**Steps:**

1. Start by pushing the source node onto the stack.

2. Pop the top node from the stack.

3. If it has not been visited, mark it as visited and print it.

4. Push all its unvisited neighbors onto the stack.

5. Repeat until the stack is empty.

**Graph Example:**

0 connected with 1, 2

1 connected with 0, 3, 4

2 connected with 0

3 connected with 1

4 connected with 2, 3

**DFS Traversal (Graph Output): 0 → 2 → 1 → 4 → 3**

1. **DFS with Stack (Tree)**

**Definition:**

**DFS for a binary tree can also be performed iteratively using a stack. This approach avoids recursion by manually controlling the traversal order with a stack.**

**Steps:**

1. Push the root node onto the stack.

2. Pop the top node, print it, and mark it as visited.

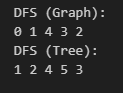
3. Push the right child (if exists), then the left child (if exists).

4. Repeat until the stack is empty.

**Tree Example:**

****

**DFS Traversal (Tree Output): 1 → 2 → 4 → 5 → 3**

****