Day 15 DFS

ITSRUNTYM

1. Definition of DFS (Depth First Search)
Depth First Search is a **graph/tree traversal algorithm** that explores as far as possible along each branch **before backtracking**.

- It starts at a **root node (or source node)** and explores **deep** into each path.
- Think of it like going down one path all the way until you're stuck, then **backtrack** and try another path.

2. Types of DFS

- **Recursive DFS**: Uses the call stack.
- **Iterative DFS**: Uses an explicit stack (like Stack<Node>).

3. Where It Applies (Use Cases)
DFS is applied when:

- You want to **explore all paths** deeply before trying others.
- Problems involving **backtracking** (e.g. Sudoku, combinations, permutations).
- Tree or graph traversal (like in **diameter of tree**, **checking cycles** in graphs).
- Finding **connected components** in a graph.
- Topological Sorting.
- Maze or puzzle solving.

No.	Problem	Description	DFS Use	Link
1	**Binary Tree Inorder Traversal**	Traverse tree in Inorder (Left-Root-Right)	Recursive DFS	🔗 Link
2	**Binary Tree Preorder Traversal**	Traverse tree in Preorder (Root-Left-Right)	Recursive DFS	🔗 Link

3	**Binary Tree Postorder Traversal**	Traverse tree in Postorder (Left-Right-Root)	Recursive DFS	🔗 Link
4	**Maximum Depth of Binary Tree**	Return depth of the deepest leaf	DFS on children	🔗 Link
5	**Diameter of Binary Tree**	Longest path between any 2 nodes	DFS for height & diameter	🔗 Link
6	**Path Sum**	Check if root-to-leaf path sums to target	DFS with sum tracking	🔗 Link
7	**Path Sum II**	Find all root-to-leaf paths summing to target	DFS + backtracking	🔗 Link
8	**Sum of Left Leaves**	Add up all left leaf nodes	DFS with node direction check	🔗 Link
9	**Invert Binary Tree**	Flip left and right children recursively	DFS	🔗 Link
10	**Balanced Binary Tree**	Check if tree is height-balanced	DFS with height check	🔗 Link
11	**Lowest Common Ancestor of a Binary Tree**	Find LCA of two nodes	DFS from root and backtrack	🔗 Link
12	**Binary Tree Paths**	Return all root-to-leaf paths	DFS with path building	🔗 Link

13	**Subtree of Another Tree**	Check if one tree is a subtree of another	DFS + tree comparison	🔗 Link
14	**Same Tree**	Check if two trees are structurally and value-wise same	DFS comparison	🔗 Link
15	**Symmetric Tree**	Check if tree is mirror of itself	DFS comparing mirrored subtrees	🔗 Link