Awesome 😃 Here’s your **DFS 30-Day LeetCode Practice List** with direct links.  
I matched each problem to the roadmap I gave you earlier so you can just open → solve → move on.

**🗓 30-Day DFS LeetCode List**

**Week 1: Tree DFS Basics**

1. [Binary Tree Preorder Traversal](https://leetcode.com/problems/binary-tree-preorder-traversal/)
2. [Maximum Depth of Binary Tree](https://leetcode.com/problems/maximum-depth-of-binary-tree/)
3. [Path Sum](https://leetcode.com/problems/path-sum/)
4. [Path Sum II](https://leetcode.com/problems/path-sum-ii/)
5. [Lowest Common Ancestor of a Binary Tree](https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/)
6. [Diameter of Binary Tree](https://leetcode.com/problems/diameter-of-binary-tree/)
7. **Review Day**

**Week 2: Grid DFS (Matrix Problems)**

1. [Number of Islands](https://leetcode.com/problems/number-of-islands/)
2. [Max Area of Island](https://leetcode.com/problems/max-area-of-island/)
3. [Flood Fill](https://leetcode.com/problems/flood-fill/)
4. [Surrounded Regions](https://leetcode.com/problems/surrounded-regions/)
5. [Word Search](https://leetcode.com/problems/word-search/)
6. [Number of Distinct Islands (Premium)](https://leetcode.ca/all/694.html) (free mirror: LC 694)
7. **Review Day**

**Week 3: Backtracking (DFS + Undo)**

1. [Subsets](https://leetcode.com/problems/subsets/)
2. [Subsets II](https://leetcode.com/problems/subsets-ii/)
3. [Combinations](https://leetcode.com/problems/combinations/)
4. [Permutations](https://leetcode.com/problems/permutations/)
5. [Combination Sum](https://leetcode.com/problems/combination-sum/)
6. [N-Queens](https://leetcode.com/problems/n-queens/)
7. [Sudoku Solver](https://leetcode.com/problems/sudoku-solver/) (Capstone Backtracking)

**Week 4: Graph DFS (Advanced Applications)**

1. [Clone Graph](https://leetcode.com/problems/clone-graph/)
2. [Cycle Detection (Undirected Graph)](https://leetcode.com/problems/graph-valid-tree/)
3. [Course Schedule II (Cycle in Directed Graph + Topo Sort)](https://leetcode.com/problems/course-schedule-ii/)
4. [Topological Sort → Course Schedule II again]
5. [Connected Components in Undirected Graph](https://leetcode.com/problems/number-of-connected-components-in-an-undirected-graph/)
6. [All Paths From Source to Target](https://leetcode.com/problems/all-paths-from-source-to-target/)
7. [Course Schedule (DFS Cycle Detection)](https://leetcode.com/problems/course-schedule/)
8. [Word Ladder II](https://leetcode.com/problems/word-ladder-ii/)
9. **Final Review:** Pick 5 hardest above & re-solve from scratch

✨ Pro Tip:  
For every problem, practice **both recursive DFS and iterative DFS (stack)** at least once. That will make you interview-ready.

Do you want me to also make you a **Python DFS template (trees, grids, graphs)** that you can reuse for all these problems?

Perfect 👍 I’ll give you a **Python DFS template pack** that you can reuse in **trees, grids, and graphs**.

**🐍 Python DFS Template Pack**

**1. DFS on Trees**

def dfs\_tree(node):

if not node:

return

# ✅ Pre-order (do work before children)

print("Pre:", node.val)

dfs\_tree(node.left)

dfs\_tree(node.right)

# ✅ Post-order (do work after children)

print("Post:", node.val)

Usage: Preorder, Postorder, Path Sum, Diameter, etc.

**2. DFS on Grids / Matrices**

def dfs\_grid(grid, r, c, visited):

rows, cols = len(grid), len(grid[0])

# Out of bounds / Already visited / Invalid cell

if r < 0 or r >= rows or c < 0 or c >= cols or grid[r][c] == 0 or (r, c) in visited:

return

visited.add((r, c)) # Mark visited

# ✅ Explore 4 directions

directions = [(1,0), (-1,0), (0,1), (0,-1)]

for dr, dc in directions:

dfs\_grid(grid, r+dr, c+dc, visited)

Usage: Number of Islands, Max Area of Island, Flood Fill, etc.

**3. DFS on Graphs (Adjacency List)**

def dfs\_graph(node, graph, visited):

if node in visited:

return

visited.add(node)

print("Visiting:", node)

for neighbor in graph[node]:

dfs\_graph(neighbor, graph, visited)

Usage: Clone Graph, Cycle Detection, Connected Components.

**4. DFS with Backtracking**

def dfs\_backtrack(path, options):

if not options:

print("Path found:", path)

return

for i, choice in enumerate(options):

path.append(choice) # ✅ Choose

dfs\_backtrack(path, options[:i]+options[i+1:]) # ✅ Explore

path.pop() # ✅ Undo (Backtrack)

Usage: Subsets, Permutations, N-Queens, Sudoku.

⚡ Key Reminder:

* **Tree DFS** → Rooted, simple recursive
* **Grid DFS** → Needs boundary + visited checks
* **Graph DFS** → Needs visited set to avoid infinite cycles
* **Backtracking** → DFS + undo step

👉 Do you want me to make you a **single “DFS Cheat Sheet PDF”** with these templates + common patterns so you can keep it open while solving LeetCode?