from collections import defaultdict, deque

def dfs(graph, start, goal, depth\_limit):

"""Depth-First Search with a depth limit."""

visited = set()

stack = [(start, 0)] # (node, depth)

while stack:

node, depth = stack.pop()

if node == goal:

return True, node # Found the goal

if node not in visited and depth <= depth\_limit:

visited.add(node)

# Push neighbors onto stack, prioritizing exploration

for neighbor in reversed(graph.get(node, [])):

stack.append((neighbor, depth + 1))

return False, None # Goal not found within depth limit

def iddfs(graph, start, goal, max\_depth):

"""Iterative Deepening Depth-First Search."""

for depth\_limit in range(max\_depth + 1):

print(f"Exploring with depth limit: {depth\_limit}")

found, node = dfs(graph, start, goal, depth\_limit)

if found:

print(f"Goal found at depth limit {depth\_limit}")

return True

print("Goal not found within the maximum depth.")

return False

# Example Usage:

graph = {

'A': ['B', 'C'],

'B': ['A', 'D', 'E'],

'C': ['A', 'F', 'G'],

'D': ['B'],

'E': ['B', 'H'],

'F': ['C'],

'G': ['C'],

'H': ['E']

}

start\_node = 'A'

goal\_node = 'H'

max\_depth\_limit = 5

iddfs(graph, start\_node, goal\_node, max\_depth\_limit)

start\_node = 'A'

goal\_node = 'Z' # A node that does not exist

max\_depth\_limit = 5

print("\nTrying to find a non-existent node:")

iddfs(graph, start\_node, goal\_node, max\_depth\_limit)

