IDDFS

def dls(graph, node, target, limit, visited):

if node == target:

print(f"Found target {target} at depth {limit}")

return True

if limit <= 0:

return False

visited.add(node)

for neighbor in graph[node]:

if neighbor not in visited:

if dls(graph, neighbor, target, limit - 1, visited):

return True

return False

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

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

print(f"\nTrying depth limit: {depth}")

visited = set()

if dls(graph, start, target, depth, visited):

print("Target found!")

return True

print("Target not found within max depth.")

return False

# Get user input

graph = {}

n = int(input("Enter number of nodes: "))

for \_ in range(n):

node = input("Enter node: ")

neighbors = input(f"Enter neighbors of {node} (space separated): ").split()

graph[node] = neighbors

start\_node = input("Enter start node: ")

target\_node = input("Enter target node to search: ")

max\_depth = int(input("Enter max depth to search: "))

print("\nIterative Deepening DFS Trace:")

iddfs(graph, start\_node, target\_node, max\_depth)

# Example Input:

# Enter number of nodes: 4

# Enter node: A

# Enter neighbors of A (space separated): B C

# Enter node: B

# Enter neighbors of B (space separated): D

# Enter node: C

# Enter neighbors of C (space separated): D

# Enter node: D

# Enter neighbors of D (space separated):

# Enter start node: A

# Enter target node to search: D

# Enter max depth to search: 3

# Example Output:

# Iterative Deepening DFS Trace:

# Trying depth limit: 0

# Trying depth limit: 1

# Trying depth limit: 2

# Found target D at depth 0

# Target found!