Lab\_7:

**Task 1**

You are given the following tree whose starting node is **A** and Goal node is **G.**

1. Depth Limited Search:

def depth\_limited\_search(node, goal, limit, path=[]):

# Add the current node to the path

path = path + [node]

if node == goal:

return path

if limit == 0:

return None

for child in get\_children(node):

result = depth\_limited\_search(child, goal, limit - 1, path)

if result is not None:

return result

# If no children lead to the goal, return None

return None

def get\_children(node):

children = {

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

'B': ['K', 'J'],

'F': [],

'D': ['G', 'C'],

'E': ['H', 'I'],

'K': ['N', 'M'],

'J': [],

'G': [],

'C': [],

'H': [],

'I': ['L'],

'N': [],

'M': [],

'L': [],

}

return children.get(node, [])

result\_path = depth\_limited\_search('A', 'G', 2)

print(result\_path)

b) Iterative Deepening Search.

def iterative\_deepening\_search(start\_node, goal\_node, max\_depth):

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

print(f"Searching at depth limit: {depth}")

result = depth\_limited\_search(start\_node, goal\_node, depth)

if result is not None:

return result

return None

def depth\_limited\_search(node, goal, limit, path=[]):

path = path + [node]

if node == goal:

return path

if limit == 0:

return None

for child in get\_children(node):

result = depth\_limited\_search(child, goal, limit - 1, path)

if result is not None:

return result

return None

def get\_children(node):

children = {

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

'B': ['K', 'J'],

'F': [],

'D': ['G', 'C'],

'E': ['H', 'I'],

'K': ['N', 'M'],

'J': [],

'G': [],

'C': [],

'H': [],

'I': ['L'],

'N': [],

'M': [],

'L': [],

}

return children.get(node, [])

start\_node = 'A'

goal\_node = 'G'

max\_depth = 5 # Se

print("Starting Iterative Deepening Search...")

path\_found = iterative\_deepening\_search(start\_node, goal\_node, max\_depth)

if path\_found:

print(f"\nGoal '{goal\_node}' found!")

print(f"Path to goal: {' -> '.join(path\_found)}")

else:

print(f"\nGoal '{goal\_node}' not found within max depth.")

**Comparison and Conclusion**

Depth Limited Search (DLS) is efficient in terms of memory and can avoid infinite loops, but its main drawback is that the solution may be missed if it lies beyond the chosen depth limit. On the other hand, Iterative Deepening Search (IDS) combines the advantages of Depth First Search and Breadth First Search. It guarantees completeness like BFS and uses less memory, while still being systematic.

Therefore, **Iterative Deepening Search provides the best path** because it ensures that the shallowest (optimal) solution is found without missing any possible path due to depth restrictions.