**Experiment 2**

**Aim - Implement DFS/DLS/DFID search algorithm in Python.**

*Code:*

Tree = {  
 "A": ['C', 'B'],  
 "B": ['E', 'D'],  
 "C": ['G', 'F'],  
 "D": ['H'],  
 "E": ['J', 'I'],  
 "F": ['L', 'K'],  
 "G": ['M'],  
 "H": [],  
 "I": [],  
 "J": [],  
 "K": [],  
 "L": [],  
 "M": []  
}  
  
  
def DFS(root, target, stack):  
 while len(stack) != 0:  
 current\_node = stack.pop()  
 if current\_node == target:  
 print("Found goal")  
 break  
 else:  
 children = Tree[current\_node]  
 stack.extend(children)  
 print(stack)  
  
  
total\_depth = 3  
  
  
def DLS(current, limit, current\_depth, goal, stack):  
 stack.pop()  
 if (current\_depth > limit):  
 return False  
 if current == goal:  
 return True  
 else:  
 children = Tree[current]  
 stack.extend(children[::-1])  
 print(stack)  
 for child in children:  
 if DLS(child, limit, (current\_depth + 1), goal, stack):  
 return True  
  
  
def IDDFS(goal):  
 limit = 0  
 found = False  
 stack = ['A']  
 while not found:  
 print(f"At depth limit {limit}:")  
 found = DLS('A', limit, 0, goal, stack)  
 stack = ['A']  
 limit += 1  
 if limit > total\_depth:  
 print("NOT Exist")  
 break  
 if found:  
 print(f"Found at depth {limit - 1}")  
  
print("Niyati’s Code for DFS DLS & IDDFS")  
  
print("The Tree structure is:{Parent:children}")  
print(Tree)  
want\_to\_continue = 1  
while want\_to\_continue == 1:  
 root\_node = input("Enter Root Node: ")  
 goal\_node = input("Enter Goal Node: ")  
 user\_inp = input("What algorithm to use? Press 1 for DFS, 2 for DLS and 3 for IDDFS: ")  
 stack = ['A']  
 print(stack)  
 if user\_inp == '1':  
 DFS(root\_node, goal\_node, stack)  
 stack = ['A']  
 elif user\_inp == '2':  
 limit = int(input("Enter depth limit: "))  
 if DLS(root\_node, limit, 0, goal\_node, stack):  
 print("Found within given depth")  
 else:  
 print("Not Found within given depth")  
 stack = ['A']  
 elif user\_inp == '3':  
 IDDFS(goal\_node)  
 else:  
 print("Enter a valid number")  
 stack = ['A']  
  
 want\_to\_continue = int(input("Press 1 to continue and anything else to exit: "))

*Output:*

Niyati’s Code for DFS DLS & IDDFS

The Tree structure is:{Parent : children}

{'A': ['C', 'B'], 'B': ['E', 'D'], 'C': ['G', 'F'], 'D': ['H'], 'E': ['J', 'I'], 'F': ['L', 'K'], 'G': ['M'], 'H': [], 'I': [], 'J': [], 'K': [], 'L': [], 'M': []}

Enter Root Node: A

Enter Goal Node: G

What algorithm to use? Press 1 for DFS, 2 for DLS and 3 for IDDFS: 1

['A']

['C', 'B']

['C', 'E', 'D']

['C', 'E', 'H']

['C', 'E']

['C', 'J', 'I']

['C', 'J']

['C']

['G', 'F']

['G', 'L', 'K']

['G', 'L']

['G']

Found goal

Press 1 to continue and anything else to exit: 1

Enter Root Node: A

Enter Goal Node: G

What algorithm to use? Press 1 for DFS, 2 for DLS and 3 for IDDFS: 2

Enter depth limit: 1

['A']

['B', 'C']

['B', 'F', 'G']

['D', 'E']

Not Found within given depth

Press 1 to continue and anything else to exit: 1

Enter Root Node: A

Enter Goal Node: G

What algorithm to use? Press 1 for DFS, 2 for DLS and 3 for IDDFS: 3

At depth limit 0:

['A']

['B', 'C']

At depth limit 1:

['A']

['B', 'C']

['B', 'F', 'G']

['D', 'E']

At depth limit 2:

['A']

['B', 'C']

['B', 'F', 'G']

Found at depth 2

Press 1 to continue and anything else to exit: 0