Name: SasiPrakash R

Reg no:231701049

**DFS TRAVESAL**

**Aim:**

To implement the DFS Travesal.

**Program:**

def dfs(graph, start, visited=None):

if visited is None:

visited = set()

visited.add(start)

print(start, end=" ")

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

if neighbor not in visited:

dfs(graph, neighbor, visited)

graph = {

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

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

'C': ['F'],

'D': [],

'E': ['F'],

'F': []

}

print("DFS Traversal:")

dfs(graph, 'A') #output: A B D E F C

print()

graph2 = {

0: [1, 2],

1: [0, 2],

2: [0, 1, 3],

3: [2],

4: [5],

5: [4]

}

print("DFS Traversal of disconnected graph:")

dfs(graph2, 0)

dfs(graph2, 4)

print()

def dfs\_adj\_matrix(adj\_matrix, start, visited=None):

if visited is None:

visited = [False] \* len(adj\_matrix) #initialize all to not visited.

visited[start] = True

print(start, end = " ")

for neighbor in range(len(adj\_matrix[start])):

if adj\_matrix[start][neighbor] == 1 and not visited[neighbor]:

dfs\_adj\_matrix(adj\_matrix, neighbor, visited)

adj\_matrix = [

[0, 1, 1, 0],

[1, 0, 1, 1],

[1, 1, 0, 0],

[0, 1, 0, 0]

]

print("DFS traversal using adjacency matrix:")

dfs\_adj\_matrix(adj\_matrix, 0)

print()

**Output:**

DFS Traversal:

A B D E F C

DFS Traversal of disconnected graph:

0 1 2 3 4 5

DFS traversal using adjacency matrix:

0 1 2 3

**Result:**

Thus the code is executed successfully.