Ex.No: 3
Date.

Depth First Search

Aim:

To implement depth first search (DFS) to traverse a graph & explore all vertices using backtracking.

Algorithm:

Step 01: Start

Step 02: Initialise an empty stack to keep track of all the visited nodes.

Step 03: Push the starting node onto stack and mark the stack

Step 04: While the stack is not empty, repeat steps 5 to 7.

Step 05: Pop the top node from the stack

Step 06: Print or process the popped node.

Step 07: For each adjacent unvisited neighbour popped node

Step 08: Mark the neighbour as visited

Step 09: Push the unvisited neighbour onto the stack

Step 10. Repeat until all reachable nodes are visited

Step 11: Stop.

Program :

```
def dfs (graph, start):
    stack = [start]
    visited = set()
    while stack:
        node = stack.pop()
        if node not in visited:
            print (node, end = " ")
            visited.add (node)
            for neighbour in graph [node]:
                if neighbour not in visited:
                    stack . append (neighbour)

graph = {
    'A' : ['B', 'C'],
    'B' : ['D', 'E'],
    'C' : ['F'],
    'D' : [],
    'E' : ['F'],
    'F' : []
}
print ("DFS Traversal starting from node 'A' :")
dfs (graph, 'A')
```

Output:

DFS Traversal starting from node 'A':

ACFBED

Result:

Thus the DFS program is executed and the output is verified successfully.