Exp. No : 2

Date :

Depth First Search

Aim :

To implement depth first search (DFS) to traverse a graph & explore all vertices by visiting as far along each branch as possible before backtracking

Algorithm :

Step-1: Start

Step-2: Initialize an empty stack and a list to keep track of visited nodes.

Step-3: Push the starting node onto stack & mask.

Step-4: while the stack is not empty, repeat 5 to 7

Step-5: Pop the top node from the stack

Step-6: Print or process the popped node.

Step-7: For each adjacent unvisited neighbour of the popped node

Step-8: Mark the neighbour as visited

Step-9: Push the unvisited neighbour onto the stack

Step-10: Repeat until all reachable node 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 neighbor in graph [node]:
            if neighbor not in visited:
```

```
        stack.append(neighbor)

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 output is verified successfully.