

Exp. No: 2 Depth first search

Date:

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

Step 3: Push the starting node onto stack & mark,

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 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 output is
verified successfully.