EX.NO.02 DEPTH FIRST SEARCH

DATE:

AIM:

To implement Depth first search to traverse a graph and explore all vertices by visiting as far along each branch as possible before backtracking.

ALGORITHM:

step 1: Start

step 2: Initialize an empty stack & a list to keep track of visited nodes.

step 3: Push the starting node onto stack & mark visits.

step 4: While the stack is not empty, repeat step 5 to step 7.

step 5: Pop the top node from the stack.

step 6: Print/process the popped node.

step 7: For each adjacent unvisited neighbour of the popped node.

Step 8: Mark 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 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
```