EXP NO : 2 DEPTH FIRST SEARCH.

DATE

AIM: To implement Depth first Search (DFS) to traverse a graph and explore all vertices by visiting as far along each branch as possible before backtracking.

ALGORITHM:

(1) Start

(2) Initialize an empty stack and a list to keep track of visted nodes.

(3) Push the starting node onto stack & mark visited.

(4) Pop the top node from the stack.

(5) Print or process the popped node.

(6) for each adjacent unvisited neighbour of the poped node.

(7) Mark the neighbour.

(8) Repeat until all reachable node.

(9) 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.