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 nodes.

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.