

Exp NO: 2

## 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 & mark

Step 4 - While the stack is not empty repeat step 5 to 7.

Step 5 - Pop the top node from the stack.

Step 6 - Print 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

Step 10 - Repeat until all reachable nodes are visited

Step 11 - Stop



code

```
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':

A C F E D

Result:

Thus the DFS program is executed and output is verified successfully.