Exp No: 02

Date : 17/08/24

DFS (Depth First Search)

Aim:

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

Algorithm:

1) Start
2) Start from a given vertex and mark it as visited.
3) Visit the current vertex.
4) Recursively visit each unvisited neighbouring vertex of the current vertex.
5) After visiting all neighbour, backtrack to the previous vertex.
6) Stop when all vertices connected to the starting vertex are visited.

Program code:

```
from collections import defaultdict
class Graph
    def __init__(self):
        self.graph = defaultdict(list)
    def addEdge(self,u,v):
        self.graph[u].append(v)
```

```
def DFSUtil(self, v, visited):
    visited.add(v)
    print(v, end=' ')
    for neighbour in self.graph[v]:
        if neighbour not in visited:
            self.DFSUtil(neighbour, visited)

def DFS(self, v):
    visited = set()
    self.DFSUtil(neighbour, visited)

if __name__ == "__main__":
    g = Graph()
    n = int(input("Enter the no of edges:"))
    for _ in range(n):
        u,v = map(int, input("Enter edge (u,v) ").split())
        g.addEdge(u,v)
    start_vertex = int(input("Enter the starting vertex for DFS. "))
    print(f"following is DFS starting from vertex {start_vertex}): ")
    g.DFS(startvertex)
```

outputs:

```
Enter the no of edge: 6
Enter edge (u,v): 0 1
Enter edge (u,v): 0 2
Enter edge (u,v): 1 2
Enter edge (u,v): 2 0
Enter edge (u,v): 2 3
Enter edge (u,v): 3 3
```

Enter the starting vertex for DFS : 2

Following is Depth First Traversal (starting from vertex 2): 2 0 1 3

Result:

Thus, the program of depth first search was successfully executed and the output was verified.