Experiment - 2

Depth First Search

AIM

To implement the Depth-First Search (DFS) algorithm in Python.

ALGORITHM

1. Represent the graph using an adjacency list or matrix. Each node (or vertex) has a list of adjacent nodes (or edges) it is connected to.
2. Create a recursive function DFS (graph, node, visited) that explores each node and its neighbors.
3. Mark the current node as visited to avoid revisiting.
4. For each adjacent node (neighbor) that hasn't been visited, recursively apply the DFS function.
5. If the graph is disconnected, ensure the DFS function is called for any unvisited node. by iterating through all nodes in the graph.
6. Take input for the number of nodes and edges and connections.

Program

```
def DFS (graph, node, visited):
  if node not in visited:
    print (node, end = ' ')
    visited . add (node)
    for neighbor in graph [node]:
      DFS (graph, neighbor, visited)
```

```
if __name__ == "__main__":
    nodes = int(input("Enter the number of nodes:"))
    edges = int(input("Enter the number of nodes:"))
    graph = {i: [] for i in range(nodes)}
    print("Enter the edges (node1 node2):")
    for _ in range(edges):
        u, v = map(int, input().split())
        graph[u].append(v)
        graph[v].append(u)
    start_node = int(input("Enter the starting
                                          node: "))
    print("DFS Traversal starting from node ",
                              start_node, ":")
    visited = set()
    DFS(graph, start_node, visited)
```

Output

```
Enter the number of nodes:6
Enter the number of edges :7
Enter the edges (node1 node2):
0 1
0 2
1 3
1 4
2 4
3 5
4 5
Enter the starting node:0
DFS Traversal starting from node 0 :
0 1 3 5 4 2
```

Result

The Depth-First search algorithm was successfully implemented and the output is verified.