19/11/2024 17:53

Exno: 2 depth first search

Date:

AIM:

To implement the Depth-first search (DFS) algo in Python, which traverses or searches a graph or tree data structure by exploring as far as possible along each branch before backtracking.

Algorithm:

1. Initialize the Graph:
   - Represent the graph using an adjacency list matrix. Each node or vertex has a list of nodes (or edges) it is connected to.
2. Define the DFS function:
   - create a recursive function DFS (graph, visited that explores each node and its
   - Mark the current node as visited to avoi revisiting.
   - For each adjacent node (neighbor) that ha been visited, recursively apply the DFS f
   - Handle Unvisited Nodes:

If the graph is disconnected, ensure function is called for any unvisited noo iterating through all nodes in the graph.

19/11/2024 17:53

4. Implementation variants:

- Recursive DFS: simple and elegant, making use the call stack.
- Iterative DFS: using an explicit stack to avoid deep recursion issues.

5. User Input:

- Take input for the number of nodes and edges, as well as the connections between them.

- 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 edges: "))
    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)
art_node = int(input("Enter the starting node: "))
("DFS Traversal starting from node", start_node)
ited = 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 2
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 2 3 5 4 2

Result: