NASARAWA STATE UNIVERSITY, KEFFI

FACULTY OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER SCIENCES

ASSIGNMENT BY GROUP 3 MEMBERS

KASHIM IDRIS

ACHI PERPETUA NKIRUKA

EMMANUEL BAKARE

COURSE CODE: CMP831 ADVANCED DATA STRUCTURE AND ALGORITHM OF SOFTWARE ENGINEERING

Group Assignment

Use the Depth First Search algorithm to Implement Graph traversal.

The goal of this project is to ultimately allow anyone to learn and understand the graph traversal DFS method of a given Graph

Scope:

GUI-based Graph traversal using Depth First Search algorithm.

Project requirements:

- ✓ Brief background of Graph traversal
- ✓ Full coding in Python / Visual basic/Java
- ✓ Full video/slide report on the algorithm of the algorithm
- ✓ The executable file

Graph Traversal: Depth First Search

For graph traversal, we normally use Breadth-First Search or Depth-First Search.

What is a Depth-First Search?

Depth-First Search or simply DFS is a graph traversal algorithm that uses the concept of backtracking or exhaustive search.

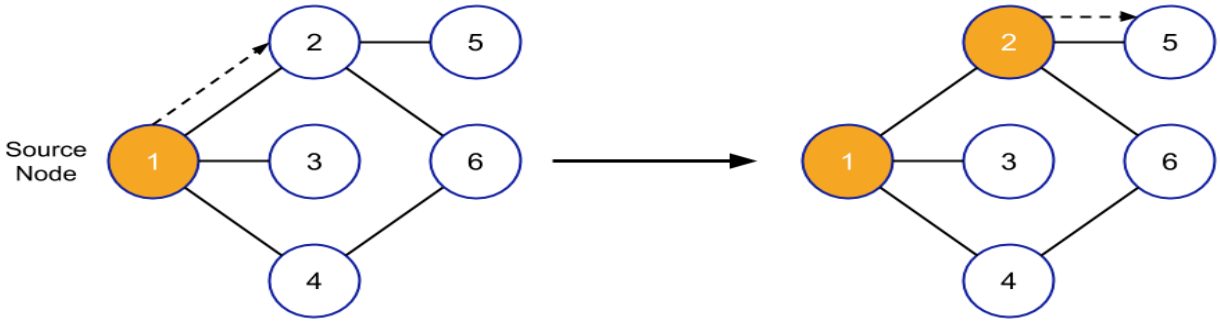
In DFS, we keep on visiting the nodes until there is some neighbour node and if there is no neighbour node then we backtrack to the previous node and again we will visit all the nodes ahead of that node. This can also be performed with the help of the exhaustive search.

All you need to do is just select one path and traverse to the end of this path and after that, come back to the previous node and traverse the nodes that are connected to those nodes and this process is going to continue again and again until there is some node that is not traversed.

If we reach the end of one path, then we come back to the recently visited node. So, while writing the code for the DFS algorithm, we are going to use the Stack data structure because Stack uses the Last in First Out order i.e., LIFO order.

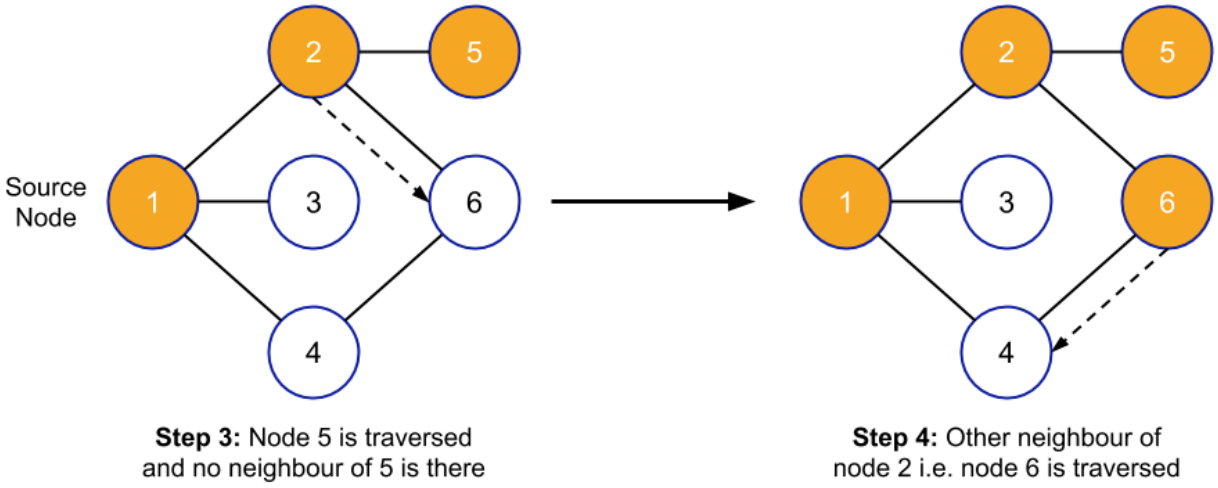
The following steps will be followed to implement the DFS:

- ✓ Select a starting/source node and add that node in the stack.
- ✓ Pop the top element of the stack and mark that node as visited. Push all the adjacent nodes of the top element in the stack.
- ✓ Repeat the above step until the stack is empty. Also, ensure that no visited nodes should be visited twice.



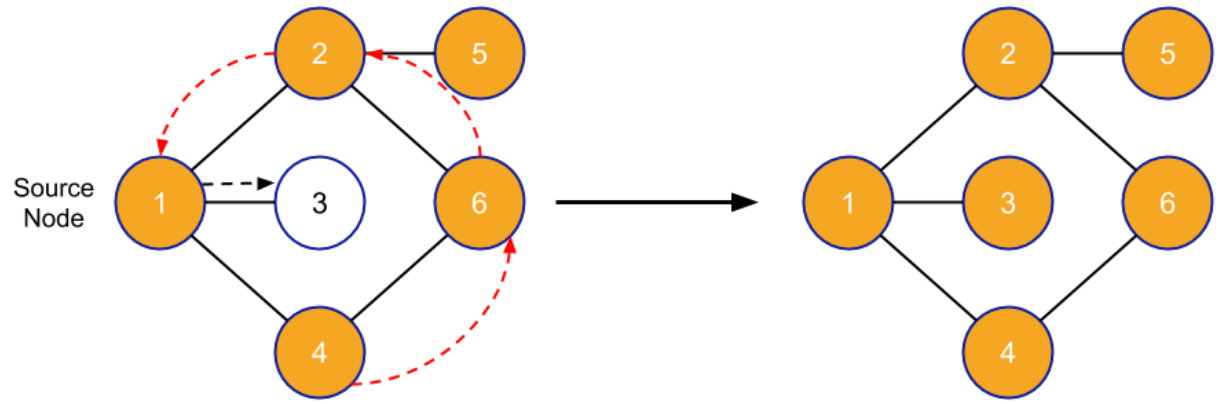
Step 1: We start from node 1 and traverse to it's neighbour

Step 2: Node 1 and 2 are traversed. Go for neighbour of Node 2



Step 3: Node 5 is traversed and no neighbour of 5 is there

Step 4: Other neighbour of node 2 i.e. node 6 is traversed



Step 5: Node 4 is traversed and next unvisited node is found by backtracking the nodes

Step 6: All nodes are traversed

Application of DFS

- ✓ **Minimum Spanning Tree:** DFS of unweighted graph results in Minimum Spanning Tree.
- ✓ **Bipartite Graph:** DFS can be used to find if a graph is bipartite or not.
- ✓ **Cycle detection:** Since we are maintaining the list of visited nodes, it can be used to detect if a cycle is present in the graph or not.
- ✓ **Topological Sorting:** Topological Sorting can be done with the help of DFS.
- ✓ **Strongly Connected Graph:** Using DFS, we can find if a graph is strongly connected or not i.e., is there any path from one node to other nodes of the graph.