Experiment 8- Depth First Search

Learning Objective: Student should be able to traverse the graph using depth first search technique.

Tools: C/C++/Java/Python under Windows or Linux environment.

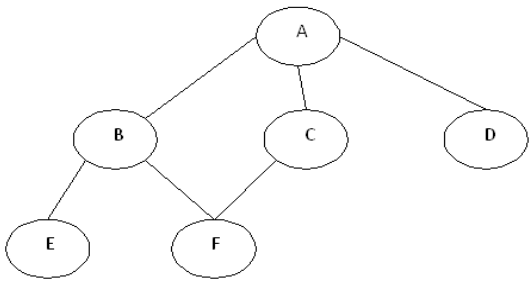
Theory: The breadth first search (BFS) and the depth first search (DFS) are the two algorithms used for traversing and searching a node in a graph. They can also be used to find out whether a node is reachable from a given node or not.

Depth First Search (DFS)

The aim of DFS algorithm is to traverse the graph in such a way that it tries to go far from the root node. Stack is used in the implementation of the depth first search. Following are the rules to be followed for DFS:

- **Rule 1** − Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack.
- **Rule 2** − If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.)
- **Rule 3** − Repeat Rule 1 and Rule 2 until the stack is empty.

Let's see how depth first search works with respect to the following graph:



As stated before, in DFS, nodes are visited by going through the depth of the tree from the starting node. If we do the depth first traversal of the above graph and print the visited node, it will be "A B E F C D". DFS visits the root node and then its children nodes until it reaches the end node, i.e. E and F nodes, then moves up to the parent nodes.

Algorithm:

Step 1: Push the root node in the Stack.
Step 2: Loop until stack is empty.
Step 3: Peek the node of the stack.

Step 4: If the node has unvisited child nodes, get the unvisited child node, mark it as traversed and push it on stack.
Step 5: If the node does not have any unvisited child nodes, pop the node from the stack.

Advantage:

1. Memory requirement is Linear WRT Nodes.
2. Less time and space complexity rather than BFS.
3. Solution can be found out by without much more search.

Disadvantage:

1. Not Guaranteed that it will give you solution.
2. Cut-off depth is smaller so time complexity is more.
3. Determination of depth until the search has proceeds.

Applications:

1. Finding Connected components.
2. Topological sorting.
3. Finding Bridges of graph.

Learning Outcomes: The student should have the ability to implement DFS traversing technique.

Course Outcomes: Upon completion of the course students will be able to traverse the given graph in DFS technique.

Conclusion:

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				