OBJECTIVES:

- Get basic idea of tree traversing using Depth First Search (DFS) algorithm.
- Implementing DFS algorithm using your preferred programming language.

BACKGROUND STUDY

- Should have prior knowledge on any programming language to implement the algorithm.
- Need knowledge on queue.
- Input a graph.
- DFS algorithm that would be covered in theory class and that knowledge will help you here to get the implementation idea.

RECOMMENDED READING

- http://en.wikipedia.org/wiki/Depth_first_search
- BOOK

DEPTH-FIRST SEARCH

Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. One starts at the root (selecting some node as the root in the graph case) and explores as far as possible along each branch before backtracking.

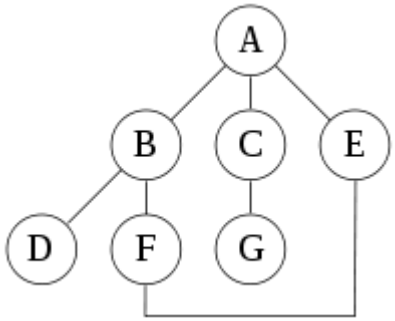

Figure 1: A graph for DFS

For the above graph, a depth-first search starting at A, assuming that the left edges in the shown graph are chosen before right edges, and assuming the search remembers previously visited nodes and will not repeat them (since this is a small graph), will visit the nodes in the following order: A, B, D, F, E, C, G.

When goal is stated in the problem definition, searching stops just after finding the goal node. In that case, all nodes might not be traversed. Search starts from start/root node. Whenever a new node is visited, it checks with goal node. If goal node found, it stops searching. If all nodes of the tree are traversed and goal does not match with any node, it indicates goal is not defined in the graph. In both cases, it ends by showing appropriate message.

Pseudocode that can be used to implement DFS algorithm is as following:

```
1  procedure DFS(G,v):
2      label v as discovered
3      for all edges e in G.adjacentEdges(v) do
4          if edge e is unexplored then
5              w ← G.adjacentVertex(v,e)
6              if vertex w is unexplored then
7                  label e as a discovered edge
8                  recursively call DFS(G,w)
9              else
10                 label e as a back edge
11     label v as explored
```

To implement this algorithm, you have to take a graph G, a start node v and a goal node as input from the user. Graph is stored in an adjacency matrix G. To trace all explored nodes you have to take an array explored. Index of that array will represent the nodes. True value in an index will indicate the node is explored and false is for explored. Initially value of all nodes will be unexplored. An array e is used to keep track of discovered and undiscovered edges.

DFS procedure is call for graph G with start node v.

LABROTORY EXERCISES

1. Implement DFS algorithm using the pseudocode described above in your preferred language.

   Input: A graph G, a start node v and a goal node D.

   Output: “Goal found” if goal can be found using DFS, otherwise “Goal not found”.