OBJECTIVES:

- Get the basic idea of tree traversing using Breadth First Search (BFS) algorithm.
- Implementing BFS 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.
- BFS 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/Breadth_first_search
- BOOK

BREADTH FIRST SEARCH

In graph theory, breadth-first search is a strategy for searching in a graph when search is limited to essentially two operations: (a) visit and inspect a node of a graph; (b) gain access to visit the nodes that neighbor the currently visited node. The BFS begins at a root node and inspects all the neighboring nodes. Then for each of those neighbor nodes in turn, it inspects their neighbor nodes which were unvisited, and so on.

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

```
1  procedure BFS(G,v) is
2      create a queue Q
3      create a set V
4      enqueue v onto Q
5      add v to V
6      while Q is not empty loop
7          t ← Q.dequeue()
8          if t is what we are looking for then
9              return t
10         end if
11         for all edges e in G.adjacentEdges(t) loop
12             u ← G.adjacentVertex(t,e)
13             if u is not in V then
14                 add u to V
15                 enqueue u onto Q
16             end if
```

```
17        end loop
18    end loop
19    return none
20 end BFS
```

Here Q-is a queue where all the nodes that are currently available to explore are stored. V-is a set of nodes where all explored nodes are stored. Both Q and V can be implemented using array.

User gives graph, start and goal node as input. G and v are input to the function BFS. G is a graph and v is the source node of that graph and both of them are user input. In step 2 and 3, just declare 2 arrays Q and V. But you need to implement necessary function to make Q works like a standard queue. In step 3 and 5, add source node v to Q and V. Following steps 6 to 18, write a loop which runs until Q is empty or goal node is found. If goal node is found, we return that node. Otherwise, we return nothing (NULL). At step 7, extract node from Q and stored it in t. If this is the goal node then return that node and these tasks are done in steps 8 to 10. Loop from step 11 to 17 is used to find all neighbors of node t. It's because if t is not the goal node then we need to find its neighbors and if they are not explored, we have explore them. Steps 13 to 16 first checks if the neighbor of t is already explored or not. If not then add this neighbor to V and Q.

LABROTORY EXERCISES

1. Implement BFS algorithm using your preferred language.