Homework 11

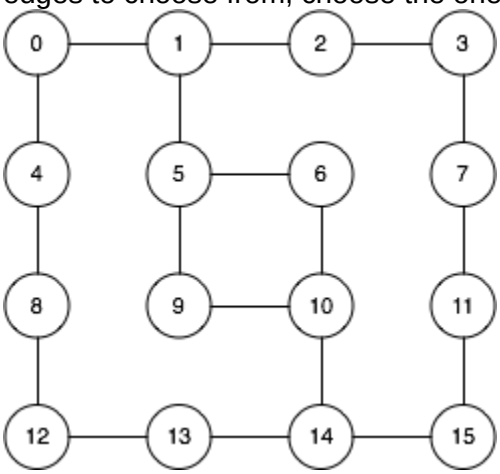
ECE 309 Fall 2019
Due: November 18, 2019
Upload an electronic copy of your answers to Moodle under HW11.

*This is a shared google document. This means (1) it may change to clarify content, and (2) other people can view your comments on this file. If you have questions, you are encouraged to comment directly on this document, but **do not add your answers here**. Make a copy into your private Google Drive and then edit the document.*

DO NOT ADD ANSWERS TO THE SHARED DOC! THAT'S CONSIDERED CHEATING!

1. Graph Traversal

For the following graph, show the order of the following traversals. If there are multiple adjacent edges to choose from, choose the one with the lowest number first.



a. [15 points] A breadth-first search starting at 10.
b. [15 points] A depth-first search starting at 10.
c. [15 points] A depth-first search starting at 15.

a. 10 - 6 - 9 - 14 - 5 - 13 - 15 - 1 - 11 - 12 - 0 - 2- 7 - 8 - 3 – 4
b. 10 - 6 - 5 - 1 - 0 - 4 - 8 - 12 - 13 - 14 - 15 - 11 - 7 - 3 - 2 - 9
c. 15 - 11 - 7 - 3 - 2 - 1 - 0 - 4 - 8 - 12 - 13 - 14 - 10 - 6 - 5 - 9

2. Finding a Path

(a) [20 points] Show pseudo-code for an algorithm that finds a path between two nodes. The algorithm must take a start node and an end node, and the algorithm returns a list showing the sequence of adjacent nodes to pass through from start to end -- in other words, return the path from start to end.

Hint: use a depth-first search, that begins at the start node and stops when the end node is reached.

```
bool DFS(Graph &G, List &Visited, Node * Start_node, Node * End_node)

{

        if ( !Visted(Start_node) ) {

                Vistited.add(Start_node)

                if(Start_node ==  End_note){

                        return true;

                }


                if(G.adjacencylist(Start_node)){

                        Visited.delete(Start_node)

                        return false;

                }


                for( Graph::iterator it = G.adjacencylist(Start_node); !it.end(); it++ ) {

                        DFS(G,Visited,it.node,End_node);

                }

        }

        return false;

}
```

(b) [10 points] Given an example of how your pseudo-code works on a graph. [Note, it should illustrate the changes you make to the search algorithm.]

Finding path from A to F

1. Visit A
2. Find adjacent B
3. Visit B
4. Find adjacent E
5. Visit E
6. No more adjacent, Delete E, Return to B
7. Find adjacent D
8. Visit D
9. Find adjacent F
10. Get to the end F
11. Return to D
12. Return to B
13. Return to A
14. Find adjacent C
15. No more adjacent, Delete C, Return to A
16. Done