Module 4 - Graphs: BFS and DFS

1. Describe an algorithm that, given a directed graph G represented as an *adjacency matrix*, returns whether or not the graph contains vertex with in-degree $|V| - 1$ and out-degree 0. In other words, does the graph have a node such that every other node points to it, but it does not point to any other node. Your algorithm must be $O(V)$. Note that there are $\Theta(V^2)$ cells in your adjacency matrix so you'll need to be clever here.

2. Write clear pseudo-code to solve the following:

   given a graph G, a start vertex s, and a vertex node t, use *DFS* to find any path from s to t and return the list of vertices in that path. Your algorithm should stop the search as soon as it finds any path. If t is not reachable from s, return an empty path (i.e., an empty list). The vertices in the list that is returned should be in order from s to t. G could be directed or undirected. For this problem, please use an implementation of the search algorithm taught in class and modify it.

3. This question is about the *depth-first search tree* and *breadth-first search tree* generated from a given **connected** graph G. Recall that these trees are formed by including the subset edges from E that are traversed to first discover each node in the respective search. With this in mind, prove the following claim:

   If T_d is the depth-first search tree generated by running DFS on G rooted at some node u, and T_b is the breadth-first search tree generated by running BFS on G rooted at that same node u, then $T_d = T_b \rightarrow G = T_d = T_b$. In other words, if BFS and DFS produce the same tree, then the entire graph G was already a tree.

   Update! Assume that graph G is undirected!

4. For a given undirected graph G, prove that the depth of a DFS tree cannot be smaller than the depth of the BFS tree. (Clearly state your proof strategy or technique.)