ASSIGNMENT 5

DUE: Tuesday November 9, 11:59 PM. DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.
Please read http://www.student.cs.uwaterloo.ca/~cs341 for general instructions and policies.

1. [10 marks]
   Let $G = (V, E)$ be a directed graph with $n$ vertices and $m$ edges. Suppose the vertices are labeled $\{1, 2, ..., n\}$, and edges are provided as an array $A$ of $(s, t)$ pairs, where the edge $(s, t)$ goes from vertex $s$ to vertex $t$.

   In each of the following parts, unless otherwise stated, you should give (i) a high level description, (ii) pseudocode, (iii) a correctness argument, and (iv) runtime analysis.

   (a) [1.5 marks] Provide pseudocode for a procedure $GetAdjRepr$ that takes $n$, $m$ and the array of edges as input, and returns an adjacency list representation $A$ of the graph. In this part, you do not need to give (i) a high level description, or (iii) a correctness argument.

   (b) [1.5 marks] Provide pseudocode for a procedure $ReverseAdj$ that takes $n$, $m$ and the output of (a) as input, and returns an adjacency list representation $R$ of the reverse graph in which the direction of every edge is reversed (so if $(s, t) \in A$ then $(t, s) \in R$). In this part, you do not need to give (i) or (iii). Note: arguments are NOT the same as in (a).

   (c) [4 marks] Suppose you are in a maze and you want to escape. The vertices of $G$ are the locations of the maze, and each edge $(s, t)$ describes a movement you can make: specifically from location $s$ to location $t$. Your goal is to move from where you are, at location 1, to the exit at location $n$. You have access to a powerful device that can reverse the direction of all edges, allowing you to move backwards along edges. You can use this device whenever you want, but you can only use it once, and its effects are permanent. Once you use it, all edges remain “reversed” forever, and you can only travel along them in their “reversed” direction.

   Give an algorithm that determines whether you can escape the maze, returning true if so, and false otherwise.

   (d) [3 marks] Suppose we want to reduce the problem in part (c) to a single call to DFS. Describe how to construct a graph $G'$ so that a single call to DFS($G'$) will solve this problem. Be sure to explain in detail which nodes and edges will be contained in $G'$. Your algorithm should run in $O(n + m)$ time. For this part, it is not necessary to give (ii), (iii) or (iv).
2. [10 marks] Let $G = (V, E)$ be an undirected graph with $n$ vertices and $m$ edges, and let $T \subseteq E$ be a spanning tree of $G$. A swap operation on $T$ replaces one edge $e$ of $T$ by an edge $f$ of $E$ such that the result, $T - \{e\} \cup \{f\}$, is again a spanning tree.

Given two spanning trees $T$ and $S$ of $G$, the goal of this question is to find a minimum length sequence of swaps that changes $T$ into $S$.

To be precise, we want a sequence of pairs of edges $(e_1, f_1), (e_2, f_2), \ldots, (e_k, f_k)$ such that

- $T_0 = T$
- for $i = 1, \ldots, k$, let $T_i$ be $T_{i-1} - \{e_i\} \cup \{f_i\}$. Each $T_i$ MUST be a spanning tree.
- $T_k = S$

and such that $k$ is minimized.

(a) [0 marks] [Warmup, don’t hand this in.] Show that $|T \setminus S| = |S \setminus T|$.

(b) [4 marks] Prove that the minimum number of swaps required to change $T$ to $S$ is $|T \setminus S|$.

(c) [4 marks] Give an $O(n^2)$ time algorithm to find a minimum length sequence of swaps to change $T$ into $S$.

Note: Part (d) asks for a faster more sophisticated algorithm, but you MUST answer this part with a straight-forward $O(n^2)$ algorithm.

(d) [2 marks] Give an $O(n \log n)$ time algorithm to find a minimum length sequence of swaps to change $T$ into $S$.

Note: This is very a challenging question and we expect that most of you will not be able to do it in a reasonable length of time, which is why it’s worth only 2 marks. We will not give part marks, so if you find this question too challenging, please just put it aside.