CS 4349.003 Homework 7

Due Wednesday October 23rd on eLearning

October 16, 2019

Please answer the following **2** questions.

1. Suppose we are given an directed graph $G = (V, E)$ and a vertex $s \in V$. We want to determine if there is any vertex $t \in V$ for which there are two distinct paths from s to t. (Recall, a path is a walk that has no repeated vertices.)

   Suppose we compute preorder and postorder times via a depth-first search *from s only* by calling $\mathrm{DFS}(s, 0)$ from Erickson Figure 6.3.

   (a) Suppose we find a forward edge $u \to v$ during our depth-first search from s. Prove G contains two or more distinct paths from s to v.

   (b) Suppose we find a cross edge $u \to v$ during our depth-first search from s. Prove G contains two or more distinct paths from s to v.

   (c) Suppose there are two or more distinct paths from s to some other vertex t. Prove our depth-first search from s will yield at least one forward or cross edge.

   (d) Describe and analyze an algorithm to determine if there is any vertex t for which there are two or more distinct paths from s to t. You may want to use the claims from the previous parts.

2. Most classical minimum spanning tree algorithms use the notions of "safe" and "useless" edges described in Erickson, but there is an alternative formulation. Let G be a weighted undirected graph, where the edge weights are distinct. We say that an edge e is ***dangerous*** if it is the longest edge in some cycle in G, and ***useful*** if it does not lie in any cycle in G.

   (a) Prove that the minimum spanning tree of G contains every useful edge. *[Hint: Every spanning tree of G is a connected subgraph that contains every vertex of G.]*

   (b) Prove that the minimum spanning tree of G does not contain any dangerous edge. *[Hint: Suppose a spanning tree T contains a dangerous edge e. How can you modify T so it no longer contains e and has lower total weight?]*

   (c) Describe and analyze an efficient implementation of the following algorithm, first described by Joseph Kruskal in the same 1956 paper where he proposed "Kruskal's algorithm". Examine the edges of G in *decreasing* order of edge weight; if an edge is dangerous, remove it from G. *[Hint: You can solve this problem before we discuss Kruskal's usual algorithm. Also, it won't be as fast as Kruskal's usual algorithm.]*