CSCI 3100: Algorithms
Homework 6

Required Problems

1. Let $G = (V, E)$ be an undirected graph with a weight $w(e)$ for each edge $e \in E$. Assume you are given a minimum spanning tree T for G. Now assume that a single new edge e is added to G, connecting two nodes u and v and with a cost c.

   (a) Give an efficient algorithm to test if T is still a minimum spanning tree for this new graph. Make your algorithm run in $O(m+n)$ time. Can you do it in $O(n)$ time? Please be sure to note any assumptions you make about what data structure is used to represent both T and G.

   (b) Suppose T is no longer the minimum spanning tree for G. Give an algorithm that is as fast as possible to update T to the new minimum spanning tree.

2. Problem 10 from Chapter 11 of the texbook

3. Problem 14 from Chapter 11 of the textbook

4. Extra credit problem (please submit on paper during your oral grading time slot):

   You've been called in to help some network administrators diagnose the extent of failure in their network. The network is designed to carry traffic from a source s to a target t, so we'll model it as a directed graph where every edge has capacity 1 and where each node lies in at least one path from s to t.

   Now, when everything is running smoothly, the maximum flow has value k. However, the current situation (and the reason you are here) is that an attacker has destroyed some of the edges in the network, so there is no s to t path. The administrators know that the attacker only destroyed k edges, the minimum number needed to separate s and t.

   Now the administrators are running a tool on s with the following behavior. If you issue the command *ping(v)*, for a given node v, it will tell you if there is currently a path from s to v. They'd like to determine the extent of failure using this tool, but it's not practical to ping every node in the graph. Therefore, they are coming to you to design an algorithm that reports the full set of nodes not currently reachable from s. You could do this by pinging everyone, but you'd like to do it using many fewer ping commands.

   Give an algorithm that accomplishes this task using only $O(k \log n)$ ping commands.