Homework #5

You do not need to turn in these problems. The goal is to reinforce what we learned in class, as well as to cover material we didn't have time to cover in class. The in-class quiz that will cover the same or similar problems. Material on homework can also appear on exams.

Problem 1: Bottleneck Edges in Minimum Spanning Trees

One of the basic motivations behind the Minimum Spanning Tree Problem is the goal of designing a spanning network for a set of nodes with minimum *total* cost. Here we explore another type of objective: designing a spanning network for which the *most expensive* edge is as cheap as possible.

Specifically, let $G = (V, E)$ be a connected graph with n vertices, m edges, and positive edge costs that you may assume are all distinct. Let $T = (V, E')$ be a spanning tree of G; we define the *bottleneck edge* of T to be the edge of T with the greatest cost.

A spanning tree T of G is a *minimum-bottleneck spanning tree* if there is no spanning tree T' of G with a cheaper bottleneck edge.

(a) Is every minimum bottleneck tree of G a minimum spanning tree of G? Prove or give a counter example.

(b) Is every minimum spanning tree of G a minimum bottleneck tree of G? Prove or give a counter example.

Problem 2: Coin Changing

Consider the problem of making change for n cents using the *fewest* number of coins.

(a) When people give change, they usually use the following greedy strategy: Select the largest coin denomination that is less than n (say it is c), and use one of these coins. Repeat for $n - c$ cents. Prove that this greedy algorithm gives the optimal solution for US coins: quarters, dimes, nickels, and pennies.
Hint: Prove that an optimal solution for $n \geq 25$ must include at least 1 quarter, that an optimal solution for $10 \leq n < 25$ must include at least one dime, and that an optimal solution for $5 \leq n < 10$ must include at least one nickel.

(b) Although this greedy algorithm is always optimal for US coins, it may not be for other sets of coin denominations. Give a set of coin denominations for which your greedy algorithm does not yield an optimal solution. Your set should include a penny to ensure that you can always successfully make change. (If you are establishing a new currency, you probably want to avoid such sets of coin denominations!)

Problem 3: Huffman Coding

In this problem, we are going to work toward proving the optimality of the Huffman Algorithm. (This is covered in Section 4.8 of the textbook.)

(a) All the prefix trees we saw in class have the following property: a node is either a leaf or it has 2 children. Prove that a binary tree in which there is a node with only 1 child cannot correspond to an optimal prefix code.

(b) Prove that there is always an optimal prefix tree in which the two lowest frequency symbols are assigned to leaves with the same parent.

(c) Prove that the Huffman algorithm achieves the minimum average number of bits per letter of any prefix code.
In your proof you can use the following fact: Let T be a binary prefix tree. Suppose x and y are two leaves with the same parent. Let T' be the tree formed from T by replacing x, y and their parent with a leaf z, and making $f(z) = f(x) + f(y)$. Then $ABL(T) = ABL(T') + f(z)$.
Hint: Part (b) is also used in the proof.