CS325: Analysis of Algorithms, Fall 2017

Practice Assignment 3*

Due: Tue, 11/21/17

Homework Policy:

1. Students should work on practice assignments individually. Each student submits to TEACH one set of *typeset* solutions, and hands in a printed hard copy in class or slides it under my door before the midnight of the due day. The hard copy will be graded.
2. Practice assignments will be graded on effort alone and will not be returned. Solutions will be posted.
3. The goal of the assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
4. You are allowed to discuss the problems with others, and you are allowed to use other resources, but you *must* cite them. Also, you *must* write everything in your own words, copying verbatim is plagiarism.
5. More items might be added to this list. ☺

Problem 1.

(a) Find a graph that has multiple minimum spanning trees.

(b) Prove that any graph with distinct edge weights has a unique minimum spanning tree.

(c) Find a graph with non-distinct edge weights that has a unique minimum spanning tree (can you generalize (b)?).

Problem 2. Recall the job scheduling problem. The input is composed of the starting and finishing times of n jobs. We would like to find the maximum set of pairwise disjoint jobs. Consider the following alternative greedy algorithms for the job scheduling problem. For each algorithm, either prove or disprove (by presenting a counter example) that it always constructs an optimal schedule.

(a) Choose the job that ends last, discard all conflicting jobs, and recurse.

(b) Choose the job that starts first, discard all conflicting jobs, and recurse.

(c) Choose the job that starts last, discard all conflicting jobs, and recurse.

(d) Choose the job with shortest duration, discard all conflicting jobs, and recurse.

*Problems are from Jeff Erickson's lecture notes, and the book "Introduction to Algorithms", by Corman, Leiserson, Rivest and Stein.

Problem 3. What is an optimal Huffman code for n characters whose frequencies are the first n Fibonacci numbers?