15-750 Graduate Algorithms

Homework # 3 **due: February 27, 2001**

Groundrules:

- Homeworks will generally consist of easier *exercises* and harder *problems*. You should do the exercises by yourself, but you may work with a friend or two on the problems if you want — just write down who you worked with. One exception: no fair working with someone who has already figured out (or already knows) the answer. Of course, your writeup should be in your own words.
- If you've seen a problem before (sometimes we will give problems that are "famous"), then say that in your solution (it won't affect your score, we just want to know). Also, if you use any sources other than the book or those on the web page, write that down too. It's fine to look up a complicated sum or inequality, but don't look up an entire solution.

Exercises:

1. Let G be a connected, weighted graph. Prove or disprove:
   (a) If every vertex in G has a unique shortest incident edge (i.e., for each vertex, the shortest incident edge is well-defined), then there is a unique MST.
   (b) If there is a unique MST, then every vertex in G has a unique shortest incident edge.
   (c) If every cut in G has a unique shortest edge across the cut, then there is a unique MST.
   (d) If there is a unique MST, then every cut in G has a unique shortest edge.
2. Suppose you have a random number generator such that each time you ask for a bit, the probability it is a 1 is p (and the probability it is a 0 is $1 - p$). The bits are independent (this is a good thing) but p is unknown and may not be equal to $1/2$. Show how you can use this source to generate unbiased random bits, such that the expected number of calls to the generator per bit is $1/[p(1 - p)]$.

Problems:

3. Consider the following problem: Given a 3-colorable graph G, we want to color the nodes with two colors so that there are no monochromatic triangles. (I.e., each triangle in the graph has at least one blue node and at least one red node). Here is a simple randomized algorithm. Start with some arbitrary 2-coloring. Then repeat the following until done: pick an arbitrary monochromatic triangle, pick a *random* vertex in that triangle, and flip the color of that vertex. Your job is to show that with probability at least $1/2$, this algorithm will halt in at most $O(n^2)$ iterations.
   (a) Find a potential function T such that at each step of the algorithm, T goes up by 1 with probability $1/3$, stays the same with probability $1/3$, and goes down by 1 with probability $1/3$.

(b) Prove that in a $(1/3, 1/3, 1/3)$ random walk starting somewhere in the range $[0, n]$, after cn^2 steps (for some constant c) there is at least at $1/2$ chance you will no longer be in the range $[0, n]$.

To make this simpler you may want to first look at a $(1/2, 1/2)$ random walk. You can then use the fact that it is very unlikely you will stay put for more than half of the steps in a $(1/3, 1/3, 1/3)$ random walk. (Specifically, Hoeffding bounds tell us that if we flip a coin of bias p for m times, the chance that we get more than a $p + \epsilon$ fraction of heads is at most $e^{-2m\epsilon^2}$.)

(c) Put (a) and (b) together to get the desired guarantee for the algorithm.

Note: if you could find an efficient algorithm that also had no monochromatic 5-cycles, this would lead to better approximation algorithms for graph coloring than are currently known.

4. Prove that the minimum spanning tree for n points on the plane (using Euclidean distance) is a subgraph of their Delunay triangulation.