Algorithms	**Problem Set 1**
CS 6820 Fall 2009	**Due September 18, 2009**

Some general instructions for the homework:

- If possible, please typeset the homework (i.e. format your solutions as an electronic file using latex or Word with mathematical notation).
- Homework solutions that are done as an electronic file can be handed in by directly uploading them to CMS. From the link to CMS on the course home page, you should be able to use the upload option associated with Problem Set 1. You can mail Ashwin (`ashwin85@cs.cornell.edu`) if you have any trouble with this.
- When a question asks for you to give an efficient algorithm for a problem, this means that you should describe an algorithm that runs in polynomial time, together with a proof that it is correct and a brief analysis of the running time. You should always try to make your algorithm run as efficiently as possible (though for the homework, unless a specific target running time is indicated, you should mainly make sure that it runs in polynomial time).

(1) *(KT Exercise 4.8)* Suppose you are given a connected graph G, with edge costs that are all distinct. Prove that G has a unique minimum spanning tree. *(Note: In class, we discussed the fact that this result is true, but without a proof. This question asks you to fill in the proof.)*

(2) *(KT Exercise 4.9)* 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.

(3) You're helping a group of physicists who are studying an unusual type of high-energy particle. In a typical experiment, they track a collection of n of these particular in a closed environment. Each of the n particles follows a linear trajectory through the environment, with the i^{th} particle following the line segment given by the parametric equation $\{(a_i + b_i t, c_i + d_i t) : -B \leq t \leq B\}$. In other words, the i^{th} particle follows the indicated line segment as t ranges from $-B$ to B.

Now, the *potential energy* of this collection of n particles at any given point in time is determined as follows. First, you build a complete graph on the n particles, treating each particle as a node, and joining each pair by an edge. Then you assign a cost to the (i, j) edge equal to the square of the distance between particles i and j. (Note that this cost is a function of time, since it based on the time-changing positions of i and j). Finally, the potential energy of the collection of particles at a time t is equal to the total cost of the edges in the MST of this graph at time t.

Now, as the particles move, the edge costs, the MST, and hence the potential energy all change over time. The physicists studying this system would like to find the value of t in the interval $[-B, B]$ at which the collection's potential energy is minimum. They're currently using an ad hoc heuristic search method designed to try finding this t, but they suspect that you know how to do better.

Give an efficient algorithm that takes the trajectories of the n particles as input, and determines the value of t in the interval $[-B, B]$ at which the potential energy of the collection of particles in minimized.

(4) *(KT Exercise 4.27)* In trying to understand the combinatorial structure of spanning trees, we can consider the space of *all* possible spanning trees of a given graph, and study the properties of this space.

Here is one way to do this. Let G be a connected graph, and T and T' two different spanning trees of G. We say that T and T' are *neighbors* if T contains exactly one edge that is not in T', and T' contains exactly one edge that is not in T.

Now, from any graph G, we can build a (large) graph $\mathcal{H}$ as follows. The nodes of $\mathcal{H}$ are the spanning trees of G, and there is an edge between two nodes of $\mathcal{H}$ if the corresponding spanning trees are neighbors.

Is it true that for any connected graph G, the resulting graph $\mathcal{H}$ is connected? Give a proof that $\mathcal{H}$ is always connected, or provide an example (with explanation) of a connected graph G for which $\mathcal{H}$ is not connected.

(5) *(KT Exercise 4.28)* Suppose you're a consultant for the networking company CluNet, and they have the following problem. The network that they're currently working on is modeled by a connected graph $G = (V, E)$ with n nodes. Each edge e is a fiber-optic cable that is owned by one of two companies — creatively named X and Y — and leased to CluNet.

Their plan is to choose a spanning tree T of G, and upgrade the links corresponding to the edges of T. Their business relations people have reached the following agreement with companies X and Y: in the tree T that is chosen, k of the edges will be owned by X and $n - k - 1$ of the edges will be owned by Y. (The number k is specified as part of the agreement.)

CluNet management now faces the following problem: It is not at all clear to them whether there even *exists* a spanning tree T meeting these conditions, and how to find one if it exists. So this is the problem they put to you: give a polynomial-time algorithm that takes G, with each edge labeled X or Y, and either (i) returns a spanning tree with exactly k edges labeled X, or (ii) reports correctly that no such tree exists.