CS 681 Fall 2001 – Problem Set 1

Due in class Friday, September 21

Note: The problems will be graded on clarity and conciseness in addition to correctness. Question (0) is not to be handed in.

(0) *Don't hand this part in.* In the lectures on the minimum spanning tree problem, we've mainly assumed (to make the discussion simpler) that all edge costs are distinct. (I.e. there are no two different $e, f \in E$ so that $c_e = c_f$.)

(a) As one way to believe that this assumption is not really crucial for the correctness of algorithms, show that Kruskal's algorithm still computes a minimum spanning tree even when all the edge costs are not distinct.

(b) However, distinct edge costs do lead to some interesting consequences. Show that if all edge costs are distinct, then in fact there is a unique minimum spanning tree of G.

(1) Recall our *tree swapping* operation from lecture: If $G = (V, E)$ be a connected graph, T is a spanning tree of G, and e is an edge of G that is not in T, then we can obtain a new spanning tree T' of G as follows. We add e to T, obtain a unique cycle C, and remove an edge $f \neq e$ from this cycle. We now have a spanning tree $T' = T + e - f$ that is different from T.

Prove that given any two spanning trees T_1 and T_2 of a connected graph G, there is a sequence of (possibly several) tree swapping operations that transforms T_1 into T_2 — i.e., it starts with T_1, and ends up with T_2.

(2) Every September, somewhere in a far-away mountainous part of the world, the county highway crews get together and decide which roads to keep clear through the coming winter. There are n towns in this county, and the road system can be viewed as a (connected) graph $G = (V, E)$ on this set of towns, each edge representing a road joining two of them. In the winter, people are high enough up in the mountains that they stop worrying about the *length* of roads and start worrying about their *altitude* — this is really what determines how difficult the trip will be.

So each road — each edge e in the graph — is annotated with a number a_e that gives the altitude of the highest point on the road. We'll assume that no two edges have exactly the same altitude value a_e. The *height* of a path P in the graph is then the maximum of a_e over all edges e on P. Finally, a path between towns i and j is declared to be *winter-optimal* if it achieves the minimum possible height over all paths from i to j.

The highway crews are going to select a set $E' \subseteq E$ of the roads to keep clear through the winter; the rest will be left unmaintained and kept off limits to travelers. They all agree that whichever subset of roads E' they decide to keep clear, it should clearly have the property that (V, E') is a connected subgraph; and more strongly, for every pair of towns i and j, the height of the winter-optimal path in (V, E') should be no greater than it is in the full graph $G = (V, E)$. We'll say that (V, E') is a *minimum-altitude connected subgraph* if it has this property.

Given that they're going to maintain this key property, however, they otherwise want to keep as few roads clear as possible. One year, they hit upon the following conjecture:

The minimum spanning tree of G, with respect to the edge weights a_e, is a minimum-altitude connected subgraph.

(Note that, thanks to the result of Question (0b) and our assumption that all a_e are distinct, it is okay for us to speak of *the* minimum spanning tree.)

Initially, this conjecture is a somewhat counter-intuitive claim, since the minimum spanning tree is trying to minimize the *sum* of the values a_e, while the goal of minimizing altitude seems to be asking for a fairly different thing. But lacking an argument to the contrary, they begin considering an even bolder second conjecture:

A subgraph (V, E') is a minimum-altitude connected subgraph if and only if it contains the edges of the minimum spanning tree.

Note that this second conjecture would immediately imply the first one, since the minimum spanning tree contains its own edges.

So here's the question:

(a) Is the first conjecture true, for all choices of G and altitudes a_e? Give a proof or a counter-example with explanation.

(b) Is the second conjecture true, for all choices of G and altitudes a_e? Give a proof or a counter-example with explanation.

(3) One of the first things you learn in calculus is how to minimize a differentiable function like $y = ax^2 + bx + c$, where $a > 0$. The minimum spanning tree problem, on the other hand, is a minimization problem of a very different flavor: there are now just a finite number of possibilities for how the minimum might be achieved — rather than a continuum of possibilities — and we are interested in how to perform the computation without having to exhaust this (huge) finite number of possibilities.

One can ask what happens when these two minimization issues are brought together, and the following question is an example of this. Suppose we have a connected graph $G = (V, E)$. Each edge e now has a *time-varying edge cost* given by a function $f_e : \mathbf{R} \to \mathbf{R}$. Thus, at time t, it has cost $f_e(t)$. We'll assume that all these functions are positive over their entire range. Observe that the set of edges constituting the minimum spanning tree of G may change over time. Also, of course, the cost of the minimum spanning tree of G becomes a function of the time t; we'll denote this function $c_G(t)$. A natural problem then becomes: find a value of t at which $c_G(t)$ is minimized.

Suppose each function f_e is a polynomial of degree 2: $f_e(t) = a_e t^2 + b_e t + c_e$, where $a_e > 0$. Give an algorithm that takes the graph G and the values $\{(a_e, b_e, c_e) : e \in E\}$, and returns a value of the time t at which the minimum spanning tree has minimum cost. Your algorithm should run in time polynomial in the number of nodes and edges of the graph G. You may assume that arithmetic operations on the numbers $\{(a_e, b_e, c_e)\}$ can be done in constant time per operation.

(4) Suppose you are given a connected graph $G = (V, E)$ in which each edge has a cost of either 0 or 1. You are also given an integer k. Give a polynomial-time algorithm that determines whether G has a spanning tree of cost *exactly* k.