CS 4820, Summer 2010 Out: July 9. **Due: Tuesday, July 13, 8:30am**

Homework 2

Reading: Chapter 4.1, 4.2, 4.5, 4.6.

Problem 1. As a president of a small law firm, you want to provide excellent service to your clients. Most of your clients want your legal advice as soon as possible (time is a big crunch for them), but you find it hard to multi-task and can provide service to one client at a time. The clients you serve earlier are happy about your service, but ones that are served late... not so much! You decide that everyday you would like to minimize the delay of a client's case, weighted by their importance (to you).

So, here is the problem. At the beginning of the day, you have the papers for the cases of n clients that are ready to be worked on. Each case i has an estimate of how long it is going to take to finish the work on that case, let us call this estimate t_i and assume that each case takes *exactly* that amount of time (after all, you haven't been in this business for years for no good reason!). Each case also has some value to you, say v_i for case i. Starting early after the morning cup of coffee, you want to finish all cases one by one in some order. Let us define the completion time of cases as follow: if you finish the cases in the order $i, j, k...$, then the finish time of i is $F_i := t_i$, that of j is $F_j := t_i + t_j$, that of k is $F_k = t_i + t_j + t_k$, and so on for all the n cases. (Note that the finish times depend on the order in which the cases are handled.) Naturally, you would like to finish more valuable cases earlier and less valuable cases later. So, you decide you want to minimize the sum $\sum_{i=1}^{n} v_i F_i$ (which is called the *objective function* for this problem). How would you decide the schedule such that it minimizes this objective of the sum of value-weighted finish times?

Example: If there are three cases with $t_1 = 1, v_1 = 1$, $t_2 = 1, v_2 = 3$ and $t_3 = 4, v_3 = 4$, the schedule $(1, 2, 3)$ would give value-weighted finish time equal to $1 \times 2 + 3 \times 4 + 4 \times 8 = 46$, while the schedule $(2, 1, 3)$ is better with value-weighted finish time equal to $3 \times 2 + 1 \times 4 + 4 \times 8 = 42$.

Problem 2. (Coin changing) One day, mysteriously, you end up in the land of coins. There are no bills in that town, and everybody carries a bagful of coins all the time (no credit cards either). As someone who is an optimization enthusiast, you want to carry as few coins everyday as possible (why carry weight when you don't need to). You decide to come up with an algorithm find the minimum number of coins to make up a certain sum of target money.

(2a) Describe a greedy algorithm to make change (of given target amount) consisting of quarters (25 cents), dimes (10 cents), nickles (5 cents), and pennies (1 cent). (And of course prove its correctness.)

(2b) Does your greedy algorithm works for all denomination of coins? Either prove that it does, or give a counter example (i.e., a set of coin denominations and a target sum) for which your algorithm does not yield an optimal solution.

Problem 3. (Hill-optimal paths) After coming back from your ordeal in the coin land, you have become an avid bicyclist. Over the course of just a few months, you have formed a new club of avid bikers called Hate-the-Hills. Now, the problem is that the members of the Hate-the-Hills club are from many different towns, and it is hard to get together easily for the weekend rides. As a determined club, the club members decide to lobby for building some bike trails among the cities, so that it is easy to get around. The cities are not so far away, and everybody likes biking so much (on flat surface at least) that they are not concerned about how far they bike, just that they really want to avoid the hills (hence the name). The hills are really what determine the difficulty of the ride.

The cities can be viewed as n nodes in a graph (call the node set V). The set of trails among cities can be viewed as undirected edges in this graph (let E be the set of edges). Each potential trail or edge (between two cities) is associated with the height h_e, which is the height of the highest point along the trail. We will assume that no two trails have the same heights. The height of a path (consisting of many edges) is just the height of the highest edge in that path. That is, for path $P = (e_1, e_2, ..., e_k)$, the height $h(P)$ of path P is just $\max_{i=1}^{k} h_i$. A path from i to j is *hill-optimal* if it achieves the minimum height among all paths from i to j (in graph (V, E)). And of course, the members Hate-the-Hills club prefer hill-optimal paths.

With enough lobbying for constructing trails, the state agrees to build the trails. But the bikers need to decide which trails to build (and they need to build as fewest possible). If E' is the subset of the edges selected, then everyone would like (V, E') to be a connected subgraph of (V, E), and more strongly, for every pair i and j, the height of the hill-optimal path in (V, E') should be no greater than the height of the hill-optimal path in the full graph (V, E). If the subgraph (V, E') has this property, we say that it is *hill-optimal connected subgraph.*

Given that they want a hill-optimal connected subgraph, they would like to build as few trails as possible (size of E' should be as small as possible). They conjecture the following (as proved in class, use the fact that there is only one MST when edge weights are all different):

Conjecture 1. The minimum spanning tree of G with respect to the edge weight h_e is a hill-optimal connected subgraph.

This seems a little counter-intuitive, since it seems unlikely that a set of just $n-1$ edges can give rise to a hill-optimal connected subgraph (in which every pair has hill-optimal path). In the lack of any evidence to the contrary, a subgroup of Hate-the-Hills club bikers suggest an even bolder conjecture:

Conjecture 2. A subgroup (V, E') is a hill-optimal connected subgraph if an only if it contains the edges of the minimum spanning tree.

Of course, the second conjecture implies the first, since the minimum spanning tree contains its own edges.

After some hard thinking, Hate-the-Hills club is running out of ideas. They are searching for an algorithms enthusiast to help them with these conjectures. Can you help them? Here are the questions:

(i) Is Conjecture 1 true, for all choices of G with distinct heights h_e for each edge? Give a proof or counterexample with explanation.

(ii) Is Conjecture 2 true, for all choices of G with distinct heights h_e for each edge? Give a proof or a counterexample with explanation.

Hint: Use two lemmas proved in lectures on Thursday, one about when an edge must belong to any MST, and second about when an edge must not belong to any MST.