Algorithms — CS 102 — Spring 2014
Instructor: Achlioptas
Homework 2 Greedy Algorithms

1. Your friend is working as a camp counselor, and he is in charge of organizing activities for a set of junior-high-school-age campers. One of his plans is the following mini-triathalon exercise: each contestant must swim 20 laps of a pool, then bike 10 miles, then run 3 miles. The plan is to send the contestants out in a staggered fashion, via the following rule: the contestants must use the pool one at a time. In other words, first one contestant swims the 20 laps, gets out, and starts biking. As soon as this first person is out of the pool, a second contestant begins swimming the 20 laps; as soon as he or she is out and starts biking, a third contestant begins swimming.., and so on. Each contestant has a projected swimming time (the expected time it will take him or her to complete the 20 laps), a projected biking time (the expected time it will take him or her to complete the 10 miles of bicycling), and a projected running time (the time it will take him or her to complete the 3 miles of running). Your friend wants to decide on a schedule for the triathlon: an order in which to sequence the starts of the contestants. The *completion time* of a schedule is the earliest time at which all contestants will be finished with all three legs of the triathalon, assuming they each spend *exactly* their projected swimming, biking, and running times on the three parts. (Again, note that different participants can bike and run simultaneously, but at most one person can be in the pool at any time.) What's the best order for sending people out, if one wants the whole competition to be over as early as possible? More precisely, give an efficient algorithm that produces a schedule whose completion time is minumum.

2. You are consulting for a trucking company that does a large amount of business shipping packages between New York and Boston. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed limit W on the maximum amount of weight they are allowed to carry. Boxes arrive at the New York station one by one, and each package i has weight w_i. The trucking station is quite small, so at most one truck can be at the station at any time. Company policy requires that boxes are shipped in the order they arrive; otherwise, a customer might get upset upon seeing a box that arrived after his make it to Boston faster. At the moment, the company is using a simple greedy algorithm for packing: they pack boxes in the order they arrive, and whenever the next box does not fit, they send the truck on its way.

   But they wonder if they might be using too many trucks, and they want your opinion on whether the situation can be improved. Here is their thinking: maybe sometimes sending off a truck that is not full allows some subsequent trucks to be better packed.

   Prove that, for any sequence of boxes with specified weights, the algorithm currently in use minimizes the number of trucks sent out. Your proof should follow the type of analysis used for the Interval Scheduling Problem: it should establish the optimality of this greedy packing algorithm by identifying a measure under which it "stays ahead" of all other solutions.

3. For each statement below, if true **give a short explanation**, if false, **give a counterexample**.

   (a) Suppose you are given a graph G, with edge costs that are all positive and distinct. Let T be a minimum spanning tree of G. Now suppose we replace each edge cost c_e by its square, c_e^2, thereby creating a new MST instance with the same graph but different costs.

   True or false? T is a minimum spanning tree for the new instance.

(b) Suppose you are given a graph G, with edge costs that are all positive and distinct. Let P be a minimum-cost s-t path in G. Now suppose we replace each edge cost c_e by its square, c_e^2, thereby creating a new instance of the problem with the same graph but different costs.

True or false? P must still be a minimum-cost s-t path for this new instance.

4. Let $G = G(V, E)$ be an arbitrary, connected, undirected graph with vertex set V and edge set E. Assume that each $e \in E$ has a distinct positive length ℓ_e, i.e., if $e \neq e'$ then $\ell_e \neq \ell_{e'}$. Let f be an arbitrary strictly increasing function from real numbers to real numbers. We want to compute a subset of edges $S \subseteq E$ such that:

   (a) $G(V, S)$ is connected.

   (b) $\sum_{e \in S} f(\ell_e)$ is as small as possible.

   Can we do this if evaluating f even once takes an infinite amount of time?

5. In trying to understand the combinatorial structure of spanning trees, we can consider the set of all possible spanning trees of a given graph and study its properties. Here is one way to do this.

   Let G be a connected graph, and T, T' be 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 not in T.

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

   Prove that for any connected graph G, the resulting graph H is connected.

Disclaimer

You have to justify everything. This means that you have to explain both why your algorithm is correct, i.e., why it produces the correct answer for every input, and why it runs in the time that you claim. Think of the person reading what you write as your boss who [having gotten you stuck with the problem] would like to earn praise for the group by showcasing your solution at the group's presentation to "the higher ups"[1]. This means that your boss *wants* your solution to be correct, but she is not willing to risk public humiliation in case you "missed" something. So, she will be reading what you write very carefully, but with good intent. Make her life easy by explaining things in a complete, lucid way that makes it clear that you've considered all the details. Explicitly stating all the details, describing data structures, and (God forbid!) presenting code-style-pseudocode is **not** the way to do this. Formulating principles, invariants and properties that are maintained by your algorithm **is** the way to do it[2]. And much harder.

[1] If you think that this is a negative way of representing professional reality, one that enforces cultural and class stereotypes, I strongly suggest you stay in college for as long as you can afford...

[2] The textbook itself, in my opinion, does a pretty good job at this. Don't be afraid to use words, but do so in complete, well-thought sentences. Take time. Don't "text" your answers.