Due: In tutorial on January 27 by 12:10pm

1. [15 marks]

   (a) Suppose that the edge costs are all distinct in an input graph G to the MST problem. Prove that G has a unique minimum cost spanning tree. Do this by considering the *proof* that any algorithm that follows the blue-red colouring paradigm always finds a minimum cost spanning tree.

   (b) Now suppose that all edge costs are distinct, except $c_1 = c_2$, where $c_1 = c(e_1)$ and $c_2 = c(e_2)$ are the two minimum edge costs. Prove the following statement or give a counter-example: G has a unique minimum cost spanning tree.

   (c) Finally suppose that all edge costs are distinct, except $c_1 = c_2 = c_3$, where $c_1 = c(e_1)$, $c_2 = c(e_2)$, $c_3 = c(e_3)$ are the three minimum edge costs. Prove the following statement or give a counter-example: G has a unique minimum cost spanning tree.

2. [15 marks]

   Let $G(V, E)$ be a connected graph in which a cost is associated with each edge. Suppose you are given a *strict* subset S of V (i.e., S is a set of vertices that does not contain every vertex). We want to find a spanning tree T of G such that:
   (i) Every vertex in S is a leaf in T.
   (ii) T has minimum cost among all spanning trees satisfying (i).

   (a) Given $S \subset V$ (a strict subset of V), is there always a spanning tree such that the vertices in S are leaves in T? If so, give a short proof; if not, give a counter-example.

   (b) Briefly describe an algorithm to find a minimum weight tree satisfying (i), when such a tree exists. Note that your algorithm should *not* try to verify that such a tree exists: it should just work correctly when such a tree exists, and it can work incorrectly otherwise.

   (c) Give a short convincing proof that your algorithm does in fact find a minimum weight tree satisfying (i).

3. [15 marks]

   You are consulting for a trucking company that does a large amount of business shipping packages from Toronto to Montréal. 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 Toronto loading platform one by one, and each package i has a weight w_i. The loading platform is quite small, so at most one truck can be at the platform at any time. Company policy (and the limited space available at the platform) requires that boxes are shipped in the order they arrive—there would be complaints if boxes made it to Montréal out of order. 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 cannot be placed on the truck without exceeding the weight limit W, 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 how they are thinking: maybe one could decrease the number of trucks needed by sometimes sending off a truck that was less full, and in this way allow the next few trucks to be better packed?

Prove that, for a given set of boxes with specified weights, the greedy algorithm currently in use actually minimizes the number of trucks that are needed.

[This question is based on exercise number 3 on pages 189–190 of the textbook.]

4. [15 marks]

   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 km, then run 3 km. 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 km of bicycling), and a projected *running time* (the expected time it will take him or her to complete the 3 km of running). Your friend wants to decide on a *schedule* for the triathalon: an order in which to sequence the starts of the contestants. Let's say that 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 participants can bike and run simultaneously, but at most one person can be in the pool at any time; also, contestants must complete the events in the specified order—swimming, biking, running—to avoid giving anyone an unfair advantage.) 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 as small as possible.

   [This is based on exercise number 6 on page 191 of the textbook.]