CMPT 307 — Data Structures and Algorithms

Exercises on Greedy Algorithms. Due: Friday, November 1st

Reminder: the work you submit must be your own. Any collaboration and consulting outside resources must be explicitly mentioned on your submission.

1. Consider the problem of making change for n cents using the fewest number of coins. Assume each coin's value is an integer.

   (a) Describe a greedy algorithm to make change consisting of quaters, dimes, nickels, and pennies. Prove that your algorithm is optimal.

   (b) Give a set of coins for which the greedy algorithm does not yield an optimal solution. Your set should include a penny so that there is a solution for every value of n.

2. 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 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 counterexample.

   (b) Is every minimum spanning tree of G a minimum-bottleneck tree of G? Prove or give a counterexample.

3. 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 a 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 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.

   What is your opinion? If you think the algorithm is optimal, prove it. If you think it is not, give a set of boxes with specified weights, that are packed in a non-optimal way.

4. Let $G = (V, E)$ be an undirected graph with costs $c_e \geq 0$ on the edges $e \in E$. Assume you are given a minimum cost spanning tree T of G. Now assume that a new edge is added to G, connecting two nodes $v, w \in V$ with cost c.

   (a) Give an efficient algorithm to test if T remains the minimum cost spanning tree with the new edge added to G (but not to the tree T). Make your algorithm run in time $O(|E|)$. Can you do it in $O(|V|)$ time.

   (b) Suppose T is no longer a minimum cost spanning tree. Give a linear time algorithm (time $O(|E|)$ to update the tree T to a new minimum cost spanning tree.

5. Let us say that a graph $G = (V, E)$ is a *near-tree* if it is connected and has at most $n + 8$ edges, where $n = |V|$. Give an algorithm with running time $O(n)$ that takes a near-tree G with weights on its edges, and returns a minimum spanning tree of G. You may assume that all the edge weights are distinct.

6. Recall the problem of finding the number of inversions. As in the class, we are given a sequence of n numbers $a_1, \ldots, a_n$, which we assume are all distinct, and we define an inversion to be a pair $i < j$ such that $a_i > a_j$.

   We motivated the problem of counting inversions as a good measure of how different two orderings are. However, one might feel that this measure is too sensitive. Let us call a pair a *significant inversion* if $i < j$ and $a_i > 2a_j$. Give an $O(n \log n)$ algorithm to count the number of significant inversions between two orderings.

7. Suppose you are consulting for a bank that is concerned about fraud detection, and they come to you with the following problem. They have a collection of n bank cards that they confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object, containing a magnetic strip with some encrypted data, and it corresponds to a unique account in the bank. Each account can have many bank cards corresponding to it, and we will say that two bank cards are *equivalent* if they correspond to the same account.

   It is very difficult to read the account number off a bank card directly, but the bank has a high-tech "equivalence test" that takes two bank cards and, after performing some computations, determines whether they are equivalent.

   Their question is the following: among the collection of n cards, is there a set of more than $\frac{n}{2}$ of them that are equivalent to one another? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them in to the equivalence tester. Show how to decide the answer to their question with only $O(n \log n)$ invocations of the equivalence tester.

8. Consider an n-node complete binary tree T, where $n = 2^d - 1$ for some d. Each node v of T is labeled with a real number x_v. You may assume that the real numbers labeling the nodes are all distinct. A node v of T is a *local minimum* if the label x_v is less than the label s_w for all nodes w that connected to v by an edge.

   You are given such a complete binary tree T, but the labeling is only specified in the following implicit way: for each node v, you can determine the value x_v by probing the node v. Show how to find a local minimum of T using only $O(\log n)$ probes to the nodes of T.