A potpourri of pleasing problems

Question 1. Let $f : \mathbb{N} \to \text{Bool}$ be a function that takes natural numbers as inputs and yields Boolean values as output. We say f is *monotone* if there is some $k \in \mathbb{N}$ such that $f(n) = \text{False}$ for all $n < k$ and $f(n) = \text{True}$ for all $n \geq k$. In other words, f starts out returning False, but as n gets bigger there comes some point when f switches to returning True (and never switches back). For example, the function $g(n) = n > 100$, which reports whether its input is greater than 100, is monotone: it outputs False for all $n < 101$ and then switches to True for all $n \geq 101$. As another example, consider the function $h(n)$ which reports whether or not $n \log_2 n > 3.6 \times 10^{13}$. Clearly h is monotone, since if some n satisfies this inequality then everything larger than n will also satisfy it; but unlike g, the critical value of n where h will switch from False to True is not *a priori* obvious.

Using pseudocode, describe an efficient algorithm which, given a monotone function f, returns the critical value $k \in \mathbb{N}$ at which f switches from False to True. What is the running time of your algorithm in terms of k?

1

Question 2. Consider the problem of making change for C cents using the fewest possible number of coins. Assume that each coin's value is an integer.

(a) Describe a greedy algorithm to make change for C cents using US quarters, dimes, nickels, and pennies.

It turns out that the greedy algorithm is actually optimal for US coins, though coming up with a proof of this fact is nontrivial.

(b) Give a set of coin denominations 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 C.

(c) Design and analyze an algorithm to make change using the fewest number of coins that works for any set of coins. That is, as input your algorithm should take

- n, the number of different coin types;
- a list $c_1, c_2, \ldots, c_n$ giving the values of the different coins (if you like, you may assume they are already sorted from smallest to largest); and
- the number of cents C we would like to make change for.

As output your algorithm should either report that it is not possible to make the required amount C using the given coins, or give a set of coins which add up to C such that the number of coins in the set is as small as possible. For example, if given as input $c_1 = 1$, $c_2 = 5$, $c_3 = 20$ and the target value $C = 47$, your algorithm should output the set $\{20, 20, 5, 1, 1\}$. Note that we assume there is an unlimited supply of coins of each type. Be sure to justify your algorithm's correctness and analyze its time complexity.

Question 3 (K&T 7.6)**.** Consider a set of mobile computing clients in a certain town who each need to be connected to one of several possible *base stations*. We'll suppose there are n clients, with the position of each client specified by its (x, y) coordinates in the plane. There are also k base stations; the position of each of these is specified by (x, y) coordinates as well.

We wish to connect each client to exactly one of the base stations, but our choice of connections is constrained in the following ways. First, there is a *range parameter*, denoted by r: a client can only be connected to a base station that is within distance r. There is also a *load parameter* L: no more than L clients can be connected to any single base station.

Your goal is to design a polynomial-time algorithm for the following problem. Given the positions of a set of clients and a set of base stations, as well as the range and load parameters r and L, decide whether every client can be connected simultaneously to a base station, subject to the constraints r and L. Be sure to analyze the running time of your algorithm.

3

Question 4. Let $G = (V, E)$ be a directed, weighted graph with positive edge weights, and let $s, t \in V$. Design and analyze an efficient algorithm to either find the length of the **longest** path from s to t in G, or report that paths from s to t can be arbitrarily long (because there is a cycle somewhere in the middle).

4