Name: Michael Rogers

ID: 105667404

CSCI 3104 **Profs. Grochow & Layer**

Problem Set 6 **Spring 2019, CU-Boulder**

Quick links: 1a 1b 1c 2a 2b 2c 2d 3a 3b 3c

1. As a budding expert in algorithms, you decide that your semester service project will be to offer free technical interview prep sessions to your fellow students. Not surprisingly, you are immediately swamped with appointment requests at all different times from students applying many different companies, some with more rigorous interviews than others (i.e., some will need more help than others). Let A be the set of n appointment requests. Each appointment a_i in A is a pair $(start_i, end_i)$ of times and $end_i > start_i$. To manage all of these requests and to help the most student students that you can, you develop a greedy algorithm to help you manage which appointments you can keep and which ones you have to drop (you can only tutor one student at a time).

(a) (2 points) Draw an example with at least 5 appointments where a greedy algorithm that selects the shortest appointment will fail.

I don't know.

(b) (2 points) Draw an example with at least 5 appointments where a greedy algorithm that selects the longest appointment will fail.

I don't know.

(c) (6 points) Describe and prove correctness for a greedy algorithm that is guaranteed to choose the subset of appointments that will help the maximum number of students that you help.

I don't know.

2. While your algorithm is clearly efficient and can proveably help the most students, you begin to receive complaints from students that you didn't help (i.e., their appointment was not part of the optimal solution). One of the students even offers to pay extra, which gives you a great idea. You will now allow students to make a donation to your favorite charity to make it more likely that their job will be selected. Let each appointment in this new set of appointments A be a triple $(start_i, end_i, donation_i)$ of start and end times and donation amounts where $end_i > start_i$ and $donationu_i > 0$. You now need to update your algorithm to handle these donations along with the requested appointment times. In this new environment, you are trying to maximize the amount of money you raise for your charity.

(a) (2 points) Give a specific case were your greedy algorithm would fail.

I don't know.

(b) (5 points) Give a recursive algorithm that would solve this new case.

I don't know.

(c) (3 points) Add memoization to this algorithm.

I don't know.

(d) (10 points) Give a bottom-up dynamic programming algorithm.

I don't know.

3. (30 pts) The cashier's (greedy) algorithm for making change doesn't handle arbitrary denominations optimally. In this problem you'll develop a dynamic programming solution which does, but with a slight twist. Suppose we have at our disposal an arbitrary number of *cursed* coins of each denomination $d_1, d_2, \ldots, d_k$, with $d_1 > d_2 > \ldots > d_k$, and we need to provide n cents in change. We will always have $d_k = 1$, so that we are assured we can make change for any value of n. The curse on the coins is that in any one exchange between people, with the exception of $i = k-1$, if coins of denomination d_i are used, then coins of denomination d_{i+1} *cannot* be used. Our goal is to make change using the minimal number of these cursed coins (in a single exchange, i.e., the curse applies).

(a) (10 points) For $i \in \{1, \ldots, k\}$, $n \in \mathbb{N}$, and $b \in \{0, 1\}$, let $C(i, n, b)$ denote the number of cursed coins needed to make n cents in change using only the last i denominations $d_{k-i+1}, d_{k-i+2}, \ldots, d_k$, where d_{k-i+2} is allowed to be used if and only if $i \leq 2$ or $b = 0$. That is, b is a Boolean "flag" variable indicating whether we are excluding the next denomination d_{k-i+2} or not ($b = 1$ means exclude it).Write down a recurrence relation for C and prove it is correct. Be sure to include the base case.

I don't know.

(b) (10 points) Based on your recurrence relation, describe the order in which a dynamic programming table for $C(i, n, b)$ should be filled in.

I don't know.

(c) (10 points) Based on your description in part (b), write down pseudocode for a dynamic programming solution to this problem, and give a Θ bound on its running time (remember, this requires proving both an upper *and* a lower bound).

I don't know.