CS302 - Problem Set 7

1. When discussing the ethical and societal consequences of algorithms, we have been doing it in a fairly unsystematic way. Most of us are not experts in philosophy or ethics or even know that much about the application domains for these algorithms (e.g. air traffic control). (Although one thing I've been pleasantly surprised with is that, despite our non-expertness, simply having a large group of people brainstorm seems to bring a lot of issues to light!)

   People who are experts in philosophy and ethics are considering these types of questions, and have come up with various frameworks for helping algorithm developers assess their algorithms. We will learn about and consider a few of these frameworks.

   Consider the following framework based on the (thanks to COVID now unfortunately relatively familiar) pharmaceutical drug testing phases:

Table 1. Accepted Phased Evaluation Structure for Pharmaceuticals, With a Proposed Parallel Structure for Evaluation of Algorithms

	Pharmaceuticals	Algorithms
Phase 1	Safety: Initial testing on human subjects	Digital testing: Performance on test cases
Phase 2	Proof-of-concept: Estimating efficacy and optimal use on selected subjects	Laboratory testing: Comparison with humans, user testing
Phase 3	Randomized Controlled Trials: Comparison against existing treatment in clinical setting	Field testing: Controlled trials of impact
Phase 4	Post-marketing surveillance: For long-term side-effects	Routine use: Monitoring for problems

   This was taken from the article Should We Trust Algorithms? by David Spiegelhalter. (You are not required to read the rest of the article, but I encourage you to if you are interested!)

   Think about some of the algorithms that we have considered in this class: scheduling algorithms for optimizing grades, Google, robot optimization, flight path optimization, cell tower optimization, Huffman's coding. Do you think this approach would

detect problematic behavior in these algorithms? What are the challenges of implementing such an approach? Who should regulate this process if implemented? Do you like this framework? When do you think such an approach should be used? Write a brief response to at least a few of these questions - we will discuss in class after seeing a couple of ideas for ethical frameworks.

2. Suppose you run a company with two offices, one in Washington and the other in San Francisco, California. You always spend the whole week in one location, but each weekend, you decide whether to fly to the other office. In week i, you can make W_i dollars if you are in Washington, and C_i dollars if you are in California. Each flight from one office to the other costs \$1000. Suppose you are initially in Washington, and you are given the lists of potential profits for each location: $W_1, W_2, \ldots, W_n$ and $C_1, C_2, \ldots, C_n$. Additionally, you are given $L_f \in \{C, W\}$, your final location, which is the location that you must be in after the nth week. This means if you spent the nth week in California, and $L_f = W$ you must spend a final \$1000 to fly back at to Washington the end of the nth week. In this problem, you will come up with an algorithm to maximize your profit.

   (a) Comment on societal/ethical impacts of this algorithm.

   (b) A greedy algorithm would look at how much money you would make in each week, and work in the place that will earn the most that week (after travel costs). Give a counter example to show why this greedy algorithm is not always optimal. (Your example should have $n \leq 3$, otherwise a later part of this problem will be more challenging.)

   (c) Create a dynamic programming algorithm (follow our usual steps, as described below, or for more challenge try to do as much on your own without my guidance/hints.)

      i. Identify the likely subproblems and recurrence objects. (Pre-Hint Hint: you need two types of subproblems, depending on whether you have to end up in Washington or California)

      ii. Determine the relevant final options for the recurrence object and write a recurrence relation for your recurrence object for each option. Also determine base case(s). Briefly explain. (Partial solution in hints if stuck.)

      iii. Turn your recurrence relation for your recurrence object into a recurrence relation for the objective function value.

      iv. Fill in pseudocode for an algorithm to determine the optimal schedule:

Algorithm 1: `CommuteSchedule`(M, L_f)

Input : Array M of size $n \times 2$, where $M[i, 0]$ is the money to be earned in W in week i, $M[i, 1]$ is the money to be earned in C in week i, and $L_f \in \{0, 1\}$ telling you where to be at the end of week n, where 0 corresponds to W and 1 corresponds to C.

Output: Array D such that $D[i] = 0$ if you should work week i in W and $D[i] = 1$ if you should work week i in C. (Your starting location is W.)

(d) What is the runtime of your algorithm?

(e) Apply your algorithm to your example from part (b) and show that it outputs the correct schedule.

3. Let `SelfReference` be an algorithm that takes as input a sorted (in increasing order) array A of n distinct (non repeating) positive and negative integers, and returns an index i such that $A[i] = i$, or returns 0 otherwise. (Assume the indices of A start at 1 and go to n.) Some hints are on the last page.

   (a) Write psuedocode for a recursive version of `SelfReference` that is as fast as possible.

   (b) Prove your algorithm is correct.

   (c) Analyze the runtime of your algorithm