CS302 - Problem Set 10

1. In this problem set, you will consider a final approach for mitigating harmful effects of algorithms: banning (through legislation) the use of certain types of algorithms, particularly in certain contexts. Read this webpage by Amnesty International advocating for the banning of facial recognition algorithms. Briefly research legislation on facial recognition algorithms here in Vermont.

   Do you think this is an good method of limiting the harm that algorithms can do? What are the advantages and disadvantages of this approach? Who gets to make the decisions regarding these bans, and is that good or bad?

2. Consider the following variations on problems we've previously solved using dynamic programming algorithms. If the algorithm we used no longer works, explain why, and then briefly describe a fix and the impact of the fix on the runtime. If the algorithm can stay the same, you don't need to explain.

   (a) In MWIS on a line, we only considered graphs where the vertex weights were positive integers. Would we have to change the algorithm if we used negative integers instead?

   (b) In MWIS on a line, we only considered graphs where the vertex weights were positive integers. Would we have to change the algorithm if we used positive real numbers instead?

   (c) In the Knapsack problem, we only considered items whose values were positive integers. Would we have to change the algorithm if the values of items were allowed to be negative integers instead? (Assume that each item's cost is still a positive integer.)

   (d) In the Knapsack problem, we only considered items whose values were positive integers. Would we have to change the algorithm if the values of items were allowed to be positive real numbers instead? (Assume that each item's cost is still a positive integer.)

   (e) In the Knapsack problem, we only considered items whose costs were positive integers and the total capacity was a positive integer. Would we have to change the algorithm if the costs of items were allowed to be positive or negative integers? (A negative cost would mean that adding that item to your knapsack would increase the size of your knapsack.) (Assume that each item's value is still a positive integer.)

   (f) **[Challenge]** In the Knapsack problem, we only considered items whose costs and values were positive integers and the total capacity was a positive integer. Would we have to change the algorithm if we used both negative and positive integers for item costs and values?

3. Let `SelfReference` be an algorithm that takes as input a sorted (in increasing order) array A of n distinct (no repeats) positive and negative integers, and returns an index i such that $A[i] = i$, or returns "None" otherwise. (Assume the indices of A start at 1 and go to n.)

   (a) Write psuedocode for `SelfReference` that is as fast as possible. (Think about which design paradigm might be best - try to do as much as you can without hints to simulate a more real-world algorithm design experience.)

   (b) What is the big-O runtime of your algorithm?

4. Given two strings x and y, the edit distance $D(x, y)$ is the minimum number of insertions or deletions or substitutions (a substitution involves replacing one character in the string with another) required to turn x into y. This is used for spell checkers: if someone types a word that is not in the dictionary, you often want to find the word that is closest to it in edit distance. For example, if someone typed "graffe" its edit distance from "graft" is 2 (delete "e", substitute "t" for the final "f"), while its edit distance to "giraffe" is 1 (insert "i").

   (a) Write pseudocode for an algorithm that takes in two strings (x and y), and calculates $D(x, y)$. (Think about which design paradigm might be best - try to do as much as you can without hints to simulate a more real-world algorithm design experience. However, this is also a hard problem, so if/when you do get stuck, please use the hints!)

   (b) What is the big-O runtime of your algorithm?