CS302 - Problem Set 9

1. Continuing our investigation of frameworks for thinking about the ethical and societal consequences of algorithms, please read the Fairness, Accountability, and Transparency in Machine Learning groups' Principles for Accountable Algorithms and a Social Impact Statement for Algorithms.

   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 help curb problematic aspect of applying these algorithms? When do you think such an approach should be used? What are the challenges of implementing such an approach? Do you like this framework? What are strengths or weaknesses of this framework relative to the previous ("Phases") approach? 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 are searching an array A of length n for an element with value t. You may assume A has no repeated elements and that t is in A. The strategy *sampling with replacement* works as follows: Let $T = \{1, 2, \ldots, n\}$. Choose ("sample") an element $g \in T$ at random, and check if $A[g] = t$. If it is, return g. If it is not, guess a new element at random from $T = \{1, 2, \ldots, n\}$. Repeat this process, until g such that $A[g] = t$ is found. What is the average runtime of this algorithm?

   Challenge: suppose instead you don't know whether t is in A or not. If t is not in A, what is the average runtime before you randomly sample all indices *sampling with replacement* and therefore realize that t is not in A?

3. 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?

4. (a) Write pseudocode to output the shortest path using the array produced in the first part of the Bellman-Ford algorithm.

Algorithm 1: ShortestPath(s, t, w, A)

Input	: Vertices s, t in a graph $G = (V, E)$ with no negative weight cycles, $|V| = n$, where G is given via an array of weights w where $w[u, v]$ gives the weight of edge (u, v), $(w[u, v] = \infty$ if there is no edge from u to $v)$, and an array A where the first dimension of A is indexed by the elements of V, and the second dimension is labeled by indices from 0 to $|V| - 1$. Then $A[v, i]$ contains the length of the shortest path from s to v that uses at most i edges.

Output: The shortest path from s to t.

1 **if** $A[t, n - 1] = \infty$ **then** return "No Path" ;

```
// Your pseudocode here!
```

(b) What is the runtime of your pseudocode if you are given the array w as described above?

(c) If you are instead given G as a reverse adjacency list, what would be the runtime of this algorithm, and why?

5. (Optional: If you would like additional practice with proving problems are in NP.) The problem MWIS-Decision is as follows. Given a graph $G = (V, E)$ as an adjacency matrix, a weighting function $w : V \to \mathbb{R}$ on the vertices of G, and $D \in \mathbb{R}$, is there an independent set on G with weight at least D? Prove MWIS-Decision$\in$ NP.