Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 LP Basics

Linear Program. A *linear program* is an optimization problem that seeks the optimal assignment for a linear objective over linear constraints. Let $x \in \mathbb{R}^d$ be the set of variables and $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$. The canonical form of a linear program is

$$\begin{aligned} &\text{minimize } c^\top x \\ &\text{subject to } Ax \geq b \\ &\qquad x \geq 0 \end{aligned}$$

Any linear program can be written in canonical form.

Let's check this is the case:

(i) What if the objective is maximization?

(ii) What if you have a constraint $Ax \leq b$?

(iii) What about $Ax = b$?

(iv) What if the constraint is $x \leq 0$?

(v) What about unconstrainted variables $x \in \mathbb{R}$?

Dual. The dual of the canonical LP is

$$\begin{aligned} &\text{maximize } b^\top y \\ &\text{subject to } A^T y \leq c \\ &\qquad y \geq 0 \end{aligned}$$

Weak duality: The objective value of any feasible dual $\leq$ objective value of any feasible primal
Strong duality: The *optimal* objective values of these two are equal.
Both are solvable in polynomial time by the Ellipsoid or Interior Point Method.

2 Job Assignment

There are I people available to work J jobs. The value of person i working 1 day at job j is a_{ij} for $i = 1, \ldots, I$ and $j = 1, \ldots, J$. Each job is completed after the sum of the time of all workers spend on it add up to be 1 day, though partial completion still has value (i.e. person i working c portion of a day on job j is worth $a_{ij}c$). The problem is to find an optimal assignment of jobs for each person for one day such that the total value created by everyone working is optimized. No additional value comes from working on a job after it has been completed.

(a) What variables should we optimize over? I.e. in the canonical linear programming definition, what is x?

(b) What are the constraints we need to consider? Hint: there are three major types.

(c) What is the maximization function we are seeking?

3 Linear regression

In this problem, we show that linear programming can handle linear regression. Let $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^d$ be given, where n, d are not assumed to be constant. However, assume all input numbers have constant bits.

(a) Recall that the ℓ_1 norm of a vector v is given by $\|v\|_1 = \sum_{i=1}^{d} |v_i|$. The L1 regression problem asks you to find $x \in \mathbb{R}^d$ that minimizes $\|Ax - b\|_1$.

(i) Provide a linear program that finds the opimtal x, given A, b.

(ii) Argue that it can be solved in polynomial time (in n, d).

(b) Recall that the ℓ_∞ norm of a vector v is given by $\|v\|_\infty = \max_i |v_i|$. The L_∞ regression problem asks you to find $x \in \mathbb{R}^d$ that minimizes $\|Ax - b\|_\infty$.

(i) Provide a linear program that finds the opimtal x, given A, b.

(ii) Argue that it can be solved in polynomial time (in n, d).

4 Power Outage

Suppose Alice is writing up her 170 homework when the power goes out. She continues writing her homework after the power goes out in attempts to finish on time. However, since the power is out, she continues to write out her homework while making some mistakes and finishes her homework. She doesn't remember when the power goes out and so now she has to check her work and change it to be correct. It takes her 1 second to insert a character into the homework, 2 seconds to erase a character, and 1.5 seconds to change one character to another. Compute how long it will take Alice to fix her homework if she optimally chooses her edits, given the string x with m characters is what she has written and the string y with n characters is what she intended to write.

5 Non-Prefix Code

As we have learned in lecture, the Huffman code satisfies the *Prefix Property*, which states that the bit string representing each symbol is not a prefix of the bit string representing any other symbol. One nice property of such codes is that, given a bit string, there is at most one way to decode it back to a sequence of symbols. However, this is not true anymore once we are working with codes that do not satisfy the Prefix Property. For example, consider the code that maps A to 1, B to 01 and C to 101. A bit string 101 can be interpreted in two ways: as C or as AB.

Your task is to, given a bit string s, determine how many ways one can interpret s. The mapping from symbols to bit strings of the code will be given to you as a dictionary d (e.g., in the example, $d = \{A : 1, B : 01, C : 101\}$); you may assume that you can access each symbol in the dictionary in constant time. Your algorithm should run in time at most $O(nm\ell)$ where n is the length of the input bit string s, m is the number of symbols, and ℓ is an upper bound on the length of the bit strings representing symbols.

Please give a 3-part solution.