Homework 6: Not that kind of programming

Due Date: The problems from this assignment are due at 11:59 PM on **Friday** February 23rd.
Problem resubmissions are due on **Monday** February 26th.

Collaboration: You are allowed (and encouraged!) to discuss these problems at a high level with others. But, please read the full collaboration policy to ensure you are keeping your discussions at the right level, and remember to cite any collaborators.

Problems to Submit: We will count your 1 best mechanical question and your 3 best long-form questions.
We strongly recommend you skim all the problems in a section before attempting them. While we try to make problems in a given section of approximately equal difficulty, you may find some to be easier than others.

Directions Unless otherwise noted, you are allowed to use algorithms described in class (or prerequisite courses) as though you had library implementations of them. For example, you can say "run the BFS-based 2-coloring algorithm from class on the graph G" or "run the bipartite checking algorithm from lecture 5 on G." We also have a list of data structures and algorithms you can use from 373 here.

Since this is a course about efficient algorithms, algorithms that are slower than necessary are unlikely to get "E" scores, but (unless otherwise noted) we do not care about constant factors. In general, an algorithm that is correct but (mildly) slower than optimal will get an equal or higher score to an algorithm that is fundamentally incorrect, but fast.

Mechanical Problems

1. Mechanical: Solve an LP

Our goal in this problem is to make sure you can use at least one LP solver. Java (in general) doesn't have *great* support for linear programming (there are libraries, but other languages have much better support). Therefore we won't restrict how you solve this problem. Some possibilities include:

- This online LP solver. (Robbie recommends this one)
- Microsoft Excel
- CVXPy (a python-based solver; requires downloading and installing the library)
- Any other method you feel comfortable with.

Use an LP solver to find an optimal point for the LP below. In your response on gradescope, include:

- A list of the settings of the variables in your optimal point.
- A screenshot of how you found your solution (e.g. of the webpage or of an excel spreadsheet – You do not have to include the excel document or code itself, if you choose those options. Just the screenshot)

$$\text{maximize} \quad 6x_1 + 10x_2 + 7x_3 + 12x_4 + 9x_5$$

$$\begin{aligned} \text{subject to:} \quad x_1 + 2x_3 &\leq 16 \\ x_2 + 2x_4 + x_5 &\leq 21 \\ x_1 &\leq 10 \\ x_3 + x_5 &\leq 13 \\ 2x_3 + x_4 &\leq 16 \\ x_1 + x_2 &\leq 11 \end{aligned}$$

Long-Form Problems

2. You are the CEO

You have a tree that represents the manager relationships in a company. Every vertex represents an employee and every parent-child relationship means the parent is the manager of the child node. Your graph might not be binary (i.e. a manager may have varying amounts of people reporting to them).

As the CEO of the company, you need all employees to submit a report so you can determine who gets promoted. Everyone will have their report due either this week, next week, or the week after.

Because writing reports detracts from doing other work, you need to ensure that no parent and child in the tree have their reports due in the same week. Additionally, your goal is to have as few people have their reports due this week to give more people time to prepare.

More formally, given a tree T, your goal is to assign exactly one of the labels $0, 1, 2$ to every vertex, so that no edge connects identical labels and the minimum number of '0' labels are used.

(a) Give pseudocode for an algorithm to run for this problem (Hint: You don't need DP here! We only needed one line). Give a 1-2 sentence explanation for why your algorithm is correct.

(b) Now, suppose that some of the leaves come pre-labeled. There may still be some unlabeled leaves, and all internal (i.e., non-leaf) nodes are unlabeled. We'll find a dynamic programming algorithm to solve the problem.

- Write a recurrence (or multiple recurrences) that describe the minimum number of 0 labels in a tree where you don't get to change any of the pre-labeled nodes.If there is no way to assign weeks (say one manager has children pre-labeled with all three possible labels), return ∞.
- Give an English description of what each of your recurrence(s) calculate and what each of the parameters means (1-2 sentences each should be enough).
- State what inputs to give to (one of your) recurrences that will be the minimum number of people labeled as 0.

(c) We said in class that a post-order traversal is usually sufficient to calculate DP on trees. Is that sufficient for your DP? Justify in 2-3 sentences. If your answer is "no" also include 1-2 sentences on what evaluation order you would use instead.

(d) What is the big-O running time to evaluate your recurrence? Let n be the number of vertices in your tree. No justification required.

3. Birthday Maximization

Robbie's sauerkraut restaurant has two birthday party recipes, the "Knuth-Morris-Party©" and the "Ford-Fulker-Fiesta©". Each recipe has a slightly different blend of our proprietary sauerkraut; the "Knuth-Morris-Party©" recipe has a ratio of 5 pounds of cooked cabbage to 3 tablespoons of refined salt, and the "Ford-Fulker-Fiesta©" has a ratio of 7 pounds of cooked cabbage to 1 tablespoon of refined salt.[1]).
In order to cook our recipes, we need to turn raw cabbage and unrefined salt into their cooked and refined forms. To do so, there are a few methods that we can use: baking, broiling, basting, or blending. Each of these methods takes in a different amount of raw material and fuel and produces a different amount of finished material. Due to issues in the supply chain, we only have a limited amount of raw cabbage, unrefined salt, and fuel. We, as proprietary shareholders and owners of Robbie's Sauerkraut Restaurant, want to know how much time we should run each process (if at all) to maximize the total amount of party sauerkrauts (the sum of the amount of "Knuth-Morris-Party©" and "Ford-Fulker-Fiesta©.")

[1] Slight modification of the recipe here: https://www.thekitchn.com/how-to-make-homemade-sauerkraut-in-a-mason-jar-193124

You may assume all quantities in this problem (hours we run the processes and amount of each recipe we produce) are continuous (i.e., fractions are fine).

	Requirement Per Hour			Output per Hour	
Process	Raw cabbage (in lbs)	Unrefined salt (in tbsps)	Fuel	Cooked cabbage	Refined salt
Baking	8	3	55	10	7
Broiling	7	5	70	12	3
Basting	7	9	85	5	13
Blending	2	3	40	3	1
Amount available	250	460	2050		

(a) What variables will be included in your LP? For each give a brief (few words) description of its purpose (e.g. "number of hours the baking process is used")

(b) What is your objective function? Briefly explain how you came up with it.

(c) What constraints do you need, and why?

4. Minecraft

You run a business in Minecraft selling different types of doors. Your business is split into 3 sectors. You know the number of diamonds you can get from the sale of what each sector produces, but since they have limited throughput, each sector can only produce at most a fixed total number of combined doors (oak, spruce, and birch) in a given week.

Also, since market supply is limited, for each of the three types of raw materials (oak, spruce, and wood planks), your business is only able to procure at most some fixed amount (in stacks[2]) of each.

Come up with an LP to maximize expected diamonds in a given week. For simplicity's sake, you may assume anything in the problem is allowed to be fractional.

		# of products produced per stack of plank type			Expected diamonds per door sold		
Company	Max. num. of doors producible	Oak	Spruce	Birch	Oak	Spruce	Birch
1	2200	20	4	12	10	35	18
2	1700	30	10	14	11	40	16
3	1800	25	8	15	12	50	17
Maximum amount of wood planks available (stacks):		160	100	120			

(a) What variables will be included in your LP? For each give a brief (few words) description of its purpose (e.g. "number of hours the baking process is used")

(b) What is your objective function? Briefly explain how you came up with it.

(c) What constraints do you need, and why?

[2] A stack in Minecraft represents a quantity of 64.

5. Real World: LPs

Algorithms have effects in the real-world. Linear Programs (and some of their relatives like integer programs) are widely used in businesses for automating decision making. But no application in the real world is perfect. In this problem you'll think about impacts of using linear programs in the real world.

The goal of this exercise is for you to consider the effects of running algorithms in the real-world. This problem is a mix of technical tasks and non-technical ones. The technical aspects can be "right" or "wrong", but the non-technical aspects are unlikely to be simply "right" or "wrong" – we won't have to **agree** with the non-technical aspects of your analysis to consider them a good analysis. Our evaluation will be based on how well they connect to the technical aspects, as well as the depth of reasoning demonstrated.

5.1. Reading

Read the excerpt of Cathy O'Neal's *Weapons of Math Destruction* linked on the resources tab on Ed. Specifically, read through the section break (the three dots) on page 130 of the pdf. You may read the rest if you wish, but it is not required. O'Neal's book focuses on what she calls "weapons of math destruction": uses of math and algorithms that harm the people affected by their outputs.

5.2. A Model

The scheduling systems O'Neal discusses in the chapter you read are often implemented as integer programs. One way to model scheduling as an integer program would be as follows:

Let $x_{i,j,k}$ be the variable representing assigning employee i to shift j on day k.

- Employees are numbered $1, ..., n$.
- Shifts are numbered 1 (the opening shift), 2 (late morning through afternoon), and 3 (the closing shift).
- Days are numbered $1, ..., 7$ (Monday through Sunday).

Our LP is:
min $\sum x_{i,j,k}$

Subject to:

(every worker works between 3 and 6 shifts)
$3 \leq \sum_{j,k} x_{1,j,k} \leq 6$
$3 \leq \sum_{j,k} x_{2,j,k} \leq 6$
...
$3 \leq \sum_{j,k} x_{n,j,k} \leq 6$

(every morning shift has at least 4 employees)
$\sum_{i} x_{i,1,1} \geq 4$
$\sum_{i} x_{i,1,2} \geq 4$
...
$\sum_{i} x_{i,1,7} \geq 4$

(and every other shift has at least 3 employees)
$\sum_{i} x_{i,2,1} \geq 3$
$\sum_{i} x_{i,2,2} \geq 3$
...
$\sum_{i} x_{i,3,7} \geq 3$

(integrality constraint – in an LP this would be $0 \leq x_{i,j,k} \leq 1$)
$x_{i,j,k} \in \{0, 1\}$ for all i, j, k.

(a) Suppose you wish to enforce that "clopening" is not allowed. I.e. no one can be on a close shift on one day and on an opening shift the next. Describe constraint(s) and/or variable(s) to add to the linear program to ensure that no one will be assigned such a shift.

(b) There are many other requirements that a company might need when making a schedule, for example:

- Employee 1 needs the full day off on Tuesday.
- Employee 2 is under 18, so cannot work night shifts on Monday through Thursday.
- Employee 3 is a trainee, who can only work a shift if Employee 1 or Employee 2 (or both) are also there to supervise.
- Employees 5,6, and 7 are managers. At least one must work every closing shift.
- Employees 8 and 9 don't get along well and cannot work the same shift.

Choose two of these requirements (or replace them with requirements of your own) and explain how to represent them by adding constraints and/or variables to the LP.

(c) Think of at least one policy that workers might want that **cannot** be efficiently represented[3] in the linear program. State the policy (in 1-2 sentences) and briefly (1-3 sentences) explain your intuition for why those policies would be difficult/impossible to efficiently represent.

5.3. Are constraints enough?

O'Neal argues at the end of the section we read that activities like the last section of this assignment are fundamentally insufficient – that as software becomes more powerful businesses will exploit it in a way that harms workers unless they are **regulated** in how they use it. She specifically mentions draft legislation that would require schedules be posted at least two weeks in advance to mitigate those harms.

Would you support legislation of this type? Describe why or why not. In your argument give at least one **technical** reason (i.e., a statement about algorithms, software, or linear programming — you might take inspiration from the things you can and can't represent in the last section, or make other statements), and mark it with an asterisk so we can find it. Your overall argument (which can include both technical and non-technical aspects) should be at least 5 sentences, but can be longer if you wish.

You can answer generally about hypothetical legislation of this type, or you (optionally) may read this specific draft legislation if you want to see a specific example of what it could look like.

6. Programming: The Dynamically Programmed Textile Factory

Caution: This is the hardest dynamic programming problem we've given you this quarter (at least Robbie thinks it is). The reason we're giving you this problem as a coding one is so you can instantly see whether your answer is right or not. You'll want to think carefully about the recurrence *first* before you jump into coding.

You are given a rectangular piece of cloth with dimensions $X \times Y$, where X and Y are positive integers. You also have a list of n dresses that can be made using the cloth. For the ith dress, you know that a rectangle of cloth of dimensions $a_i \times b_i$ is needed to make it, and that the selling price of the dress is p_i, all of which are positive integers.

You have a machine that can make a cut any rectangular piece of cloth into two pieces, either horizontally or vertically. Design an algorithm that determines the best return on the $X \times Y$ piece of cloth, that is, a strategy for repeatedly cutting the cloths so that the dresses made from the resulting pieces give the maximum sum of selling prices.

[3] by which we mean you would need only a constant number of constraints per person (or per shift) to represent the constraint.

You are free to make as many copies of a given dress as you wish, or none, if so desired. You may rotate the cloth during the manufacturing process (so a 5×3 cloth can make a 3×5 dress). Your solution should use an iterative approach, and run in polynomial time of X, Y and n.

Hint: You will need many recursive calls in a correct recurrence; this means in your iterative code, you'll look at many other locations in the table. Our approach relies heavily on the fact that every input is an integer; take advantage of that assumption!

Resubmission

You may resubmit two problems from any combination of past homeworks. When you do, remember to fill out the form on the assignments page so we know which problem you submitted.