CS 170 Homework 8

Due **Saturday 10/25/2025, at 10:00 pm (grace period until 11:59pm)**

1 Study Group (1 point)

List the names and SIDs of the members in your study group. If you have no collaborators, explicitly write "none".

2 Knightmare (10 points)

Give a dynamic programming algorithm to find the number of ways you can place knights on an X by Y ($X > Y$) chessboard such that no two knights can attack each other (there can be any number of knights on the board, including zero knights). Knights can move in a 2×1 shape pattern in any direction; a knight attacks any piece on a square it can move to in one move.

Describe your efficient algorithm below as follows.

(a) Define your subproblems.

(b) Write down your recurrence relation and base cases.

(c) Analyze the runtime of your algorithm.

(d) Analyze the space complexity of your algorithm.

Your algorithm's runtime should be $O(2^{3Y}XY)$. Return your answer mod 10007.

Hint: consider an approach similar to Discussion 6 Q2 (Counting Strings).

3 Max independent set: part two (10 points)

You are given a connected binary tree T with n nodes and a designated root r, where every vertex v has a weight $W[v]$. A set of nodes S is a k-independent set of T if $|S| = k$ and no two nodes in S have an edge between them in T. The weight of such a set is given by adding up the weights of all the nodes in S, i.e.

$$W(S) = \sum_{v \in S} W[v].$$

Given an integer $k \leq n$, your task is to find the maximum possible weight of any k-independent set of T.

(a) (10 points) Describe an $O(nk^2)$ algorithm and analyze its runtime. Proof of correctness and space complexity analysis are not required. Recall that in a binary tree, every node has at most 2 children.

Hint: your subproblem should be somewhat similar to the one used for the max independent set in a tree problem from lecture. However, you need to modify how your algorithm accounts for the "current" local state, or else you may not be able to achieve the desired runtime (why?).

(b) **Extra credit** (up to 5 points): Now let's try to generalize to arbitrary trees. Consider any arbitrary tree T, with no restrictions on the number of children per node. Describe how we can add up to $O(n)$ "dummy" nodes (i.e. nodes with weight 0) to T, as well as some edges, so that the resulting tree is a binary tree T_b. Then, describe an $O(nk^2)$ algorithm that works for general trees T, and analyze its runtime. Proof of correctness and space complexity analysis are not required.

Hint: there exists two ways (known to us) to solve this. One way is to combine parts (a) with the dummy nodes construction, and then modify the recurrence to account for the dummy nodes. The other way involves 3D dynamic programming, in which you directly extend your recurrence from part (a) to iterate across vertices' children. We recommend the first way as it may be easier to conceptualize, but in the end it is up to you!

4 Canonical form LP (11 points)

Recall that any linear program can be reduced to a more constrained *canonical form* where all variables are non–negative, the constraints are given by $\leq$ inequalities, and the objective is the maximization of a cost function.

More formally, our variables are x_i. Our objective is $\max c^\top x = \max \sum_i c_i x_i$ for some constants c_i. The jth constraint is $\sum_i a_{ij} x_i \leq b_j$ for some constants a_{ij}, b_j. Finally, we also have the constraints $x_i \geq 0$.

An example canonical form LP:

$$\text{maximize } 5x_1 + 3x_2$$

$$\text{subject to } \begin{cases} x_1 + x_2 - x_3 \leq 1 \\ -(x_1 + x_2 - x_3) \leq -1 \\ -x_1 + 2x_2 + x_4 \leq 0 \\ -(-x_1 + 2x_2 + x_4) \leq 5 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

For each of the subparts below, describe how we should modify it so that it satisfies canonical form. If it is impossible to do so, justify your reasoning.

Note that the subparts are independent of one another. Also, you may assume that variables are non-negative unless otherwise specified.

(a) (1 point) Min Objective: $\min \sum_i c_i x_i$

(b) (1 point) Lower Bound on Variable: $x_1 \geq b_1$

(c) (1 point) Bounded Variable: $b_1 \leq x_1 \leq b_2$

(d) (1 point) Equality Constraint: $x_2 = b_2$

(e) (1 point) More Equality Constraint: $x_1 + x_2 + x_3 = b_3$

(f) (1.5 points) Unconstrained Variable: $x \in \mathbb{R}$

(g) (1.5 points) Absolute Value Constraint: $|x_1 + x_2| \leq b_2$ where $x_1, x_2 \in \mathbb{R}$

(h) (1.5 points) Another Absolute Value Constraint: $|x_1 + x_2| \geq b_2$ where $x_1, x_2 \in \mathbb{R}$

(i) (1.5 points) Min Max Objective: $\min \max(x_1, x_2, x_3, x_4)$

Hint: use a dummy variable!

5 Baker (9 points)

You are a baker who sells batches of brownies and cookies (unfortunately no brookies... for now). Each brownie batch takes 4 kilograms of chocolate and 2 eggs to make; each cookie batch takes 1 kilogram of chocolate and 3 eggs to make. You have 80 kilograms of chocolate and 90 eggs. You make a profit of 60 dollars per brownie batch you sell and 30 dollars per cookie batch you sell, and want to figure out how many batches of brownies and cookies to produce to maximize your profits.

(a) (5 points) Formulate this problem as a linear programming problem; in other words, write a linear program (in canonical form) whose solution gives you the answer to this problem. Draw the feasible region, and find the solution via inspection. Briefly explain how you found your solution.

(b) (4 points) Suppose instead that the profit per brownie batch is P dollars and the profit per cookie batch remains at 30 dollars. For each vertex you listed in the previous part, give the range of P values for which that vertex is the optimal solution.

6 [Coding] More dynamic programming (6 points)

For this week's homework, you'll implement another dynamic programming algorithm: the TSP algorithm with backtracking. There are two ways that you can access the notebook and complete the problems:

1. **On Datahub**: click here and navigate to the `hw08` folder.
2. **On Local Machine**: `git clone` (or if you already cloned it, `git pull`) from the coding homework repo,

   https://github.com/Berkeley-CS170/cs170-fa25-coding

   and navigate to the `hw08` folder. Refer to the `README.md` for local setup instructions.

Notes:

- *Submission Instructions:* Please download your completed submission `.zip` file and submit it to the Gradescope assignment titled "HW08 (Coding)."
- *Getting Help:* Conceptual questions are always welcome on Edstem and office hours; *note that support for debugging help during OH will be limited.* If you need debugging help first try asking on the public Edstem threads. To ensure others can help you, make sure to:

  1. Describe the steps you've taken to debug the issue prior to posting on Ed.
  2. Describe the specific error you're running into.
  3. Include a few small but nontrivial test cases, alongside both the output you expected to receive and your function's actual output.

  If staff tells you to make a private Ed post, make sure to include *all of the above items* plus your full function implementation. If you don't provide them, we will ask you to provide them.
- *Academic Honesty Guideline:* We realize that code for some of the algorithms we ask you to implement may be readily available online, but we strongly encourage you to not directly copy code from these sources. Instead, try to refer to the resources mentioned in the notebook and come up with code yourself. That being said, we **do acknowledge** that there may not be many different ways to code up particular algorithms and that your solution may be similar to other solutions available online.

