CS 170 Homework 8

Due **Friday 10/25/2024, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

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

2 Knightmare

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.

Provide a 4-part solution. Your algorithm's runtime should be $O(2^{3Y}XY)$, and return your answer mod 1789.

Hint: consider an approach similar to Dis6 Q4 (Counting Strings).

3 Max Independent Set Again

You are given a connected 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. We will first tackle the problem in the special case that T is a binary tree, and then generalize our solution to a general tree T.

(a) Assume that T is a binary tree, i.e. every node has at most 2 children. Describe an $O(nk^2)$ algorithm that solves this special case, and analyze its runtime. Proof of correctness and space complexity analysis are not required.

Hint: your subproblem should be similar to the one used for the max independent set in a tree problem, but you need to also account for the "current" independent set size (why do we need to do this?).

(b) Now, 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.

(c) Describe an $O(nk^2)$ algorithm to solve the general case (i.e. when T is any arbitrary tree), 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) and (b), 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

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 to 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) Min Objective: $\min \sum_i c_i x_i$

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

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

(d) Equality Constraint: $x_2 = b_2$

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

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

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

(h) Min Max Objective: $\min \max(x_1, x_2, x_3, x_4)$

Hint: use a dummy variable!

5 Baker

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) 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 using Simplex.

(b) 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 Simply Simplex

Consider the following linear program.

$$\max\ 5x + 4y$$
$$\text{subject to } \begin{cases} x + 3y \leq 15 \\ 3x - 2y \leq 12 \\ 4x + 4y \geq 16 \\ x \geq 0, y \geq 0 \end{cases}$$

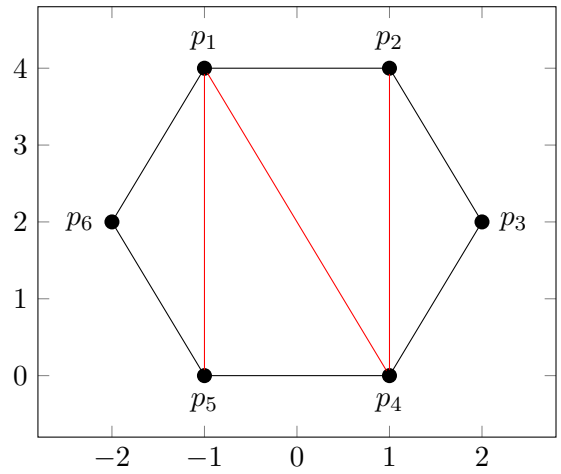
(a) Sketch the feasible region. Make sure to clearly label your axes and vertices.

(b) Run the Simplex algorithm on this LP starting at $(0, 4)$. What are the vertices visited? **Please show your work.**

7 (OPTIONAL) Triangulating a Polygon

You are given a convex polygon P that has n vertices $p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n)$.. We define a triangulation of P to be a collection T of $n-3$ diagonals from P such that no two diagonals intersect inside the polygon. A triangulation splits the polygon's interior into $n-2$ disjoint triangles.

For example, suppose we have a hexagon $P = [p_1 = (-1, 4), p_2 = (1, 4), p_3 = (2, 2), p_4 = (1, 0), p_5 = (-1, 0), p_6 = (-2, 2)]$. One possible triangulation T is $\{(p_1, p_4), (p_1, p_5), (p_2, p_4)\}$, denoted in red in the depiction below:



Now, we define the cost of a triangulation to be the sum of the square lengths of the diagonals forming the triangulation:

$$\text{cost}(T) \coloneqq \sum_{(p_i, p_j) \in T} d(p_i, p_j)^2$$

where the length of a diagonal connecting $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ is $d(p_i, p_j) = \sqrt{|x_i - x_j|^2 + |y_i - y_j|^2}$, the Euclidean distance between the two points.

Your task is to devise a dynamic programming algorithm to compute the minimum cost of any triangulation of a given polygon P, i.e. $\min_T\{\text{cost}(T)\}$. **Please provide a 3-part solution.**

Hint: A greedy approach will not work here. If you are stuck, reading about Chain Matrix Multiplication in DPV section 6.5 may help guide your thinking.