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 Egg Drop Revisited

Recall the Egg Drop problem from Homework 7:

You are given m identical eggs and an n story building. You need to figure out the highest floor $h \in \{0, 1, 2, \dots n\}$ that you can drop an egg from without breaking it. Each egg will never break when dropped from floor h or lower, and always breaks if dropped from floor $h+1$ or higher. ($h = 0$ means the egg always breaks). Once an egg breaks, you cannot use it any more. However, if an egg does not break, you can reuse it.

Let $f(n, m)$ be the minimum number of egg drops that are needed to find h (regardless of the value of h).

Instead of solving for $f(n, m)$ directly, we define a new subproblem $M(x, m)$ to be the maximum number of floors for which we can always find h in at most x drops using m eggs.

For example, $M(2, 2) = 3$ because a 3-story building is the tallest building such that we can always find h in at most 2 egg drops using 2 eggs.

(a) Find a recurrence relation for $M(x, m)$ that can be computed in constant time given the previous subproblems. Briefly justify your recurrence.

Hint: As a starting point, what is the highest floor that we can drop the first egg from and still be guaranteed to solve the problem with the remaining $x - 1$ drops and $m - 1$ eggs if the egg breaks?

(b) Give an algorithm to compute $M(x, m)$ given x and m and analyze its runtime.

(c) Modify your algorithm from (b) to compute $f(n, m)$ given n and m.

Hint: If we can find h when there are more than n floors, we can also find h when there are n floors.

(d) Show that the runtime of the algorithm from part (c) is $O(nm)$. Compare this to the runtime you found in last week's homework.

(e) Show that we can implement the algorithm from part (c) to use only $O(m)$ space.

2 LP Basics

Linear Program. A *linear program* is an optimization problem that seeks the optimal assignment for a linear objective over linear constraints. Let $x \in \mathbb{R}^n$ 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

$$\begin{aligned} \text{maximize } & c^\top x \\ \text{subject to } & Ax \le b \\ & x \ge 0 \end{aligned}$$

Any linear program can be written in canonical form.

Let's check this is the case:

(i) What if the objective is minimizaiton?

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

(iii) What about $Ax = b$?

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

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

3 Huffman and LP

Consider the following Huffman code for characters a, b, c, d: $a = 0, b = 10, c = 110, d = 111$.

Let f_a, f_b, f_c, f_d denote the fraction of characters in a file (only containing these characters) that are a, b, c, d respectively. Write a linear program with variables f_a, f_b, f_c, f_d to solve the following problem: What values of f_a, f_b, f_c, f_d (that can generate this Huffman code) result in the Huffman code using the most bits per character?

4 Simplex Practice

(a) Consider the following linear program:

$$\begin{aligned} \text{Maximize } & x+y \\ \text{subject to: } & -\frac{1}{2}x+y \leq 3 \\ & x+2y \leq 12 \\ & y \leq 4 \\ & 3x+y \leq 21 \\ & x,y \geq 0 \end{aligned}$$

Draw the feasible region. Write out the sequence of vertices traversed by the simplex algorithm starting at $(0,3)$.

(b) Consider the following linear program on three variables:

$$\begin{aligned} \text{Maximize } & f(x,y,z) \\ \text{subject to: } & x+2y \leq 10 \\ & z \leq 3 \\ & x,y,z \geq 0 \end{aligned}$$

Is it possible for $p = \{(0,0,0),(0,5,0),(10,0,0),(10,0,3)\}$ to be a valid path that the simplex algorithm takes for some linear function f? If so, is there a linear function, f, such that p is the *only* possible valid path that the simplex algorithm may take?

(c) Over all linear programs in two variables and n constraints (including $x_1, x_2 \geq 0$), determine the minimum and the maximum number of vertices that the simplex algorithm may need to traverse over including its starting and ending points. You may assume that we start already at some vertex on the feasible region, $\vec{x}$.

(d) In the simplex algorithm, we can choose *any* better vertex upon each iteration. What if we make the following modification: choose the *best* adjacent vertex. How does this change the answer to part (c)?