CS 170 HW 7

Due **2020-10-19, at 10:00 pm**

1 Study Group

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

In addition, we would like to share correct student solutions that are well-written with the class after each homework. Are you okay with your correct solutions being used for this purpose? Answer "Yes", "Yes but anonymously", or "No"

2 Egg Drop Revisited

Recall the Egg Drop problem from Homework 6:

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

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

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

(a) Find a recurrence relation for $M(d, k)$ 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 $d-1$ drops and $k-1$ eggs if the egg breaks?*)

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

(c) Modify your algorithm from (b) to compute $f(n, k)$ given n and k. (Hint: *If we can find ℓ when there are more than n floors, we can also find ℓ when there are n floors.*)

(d) Show that the runtime of the algorithm of part (c) is $O(nk)$.

(e) How can we implement the algorithm using $O(k)$ space?

3 Maximum Non-Crossing Bipartite Matching

Recall that given a graph G, a matching in G is a set of edges in G such that no two edges share an endpoint.

We are given a bipartite graph $G(L \cup R, E)$, where L and R are ordered. That is, the vertices of G can be partitioned into two ordered sets $L = \{\ell_1, \ell_2, \cdots \ell_m\}$ and $R = \{r_1, r_2 \ldots r_n\}$, such that every edge is of the form (ℓ_i, r_j).

We say a matching in G is *non-crossing* if it does not contain two edges $(\ell_i, r_j), (\ell_{i'}, r_{j'})$ such that $i < i'$ but $j > j'$. Pictorially, think of ℓ_i as being located at the point $(0, i)$ and r_j as being located at the point $(1, j)$. Then a matching is non-crossing if we can draw the line segment between the endpoints of every edge in the matching, without any of these line segments intersecting.

(a) Show that in any non-crossing matching in G, at least one of the following is true: (i) the edge (ℓ_m, r_n) exists and is in the matching, (ii) ℓ_m is not included in the matching, or (iii) r_n is not included in the matching.

(b) Give an $O(mn)$-time dynamic programming algorithm to find the size of largest non-crossing matching in G. Give a three-part solution. (Hint: Part (a) identifies cases that might be useful in writing the recurrence relation).

4 Jeweler

This is a solo question.

You are a jeweler who sells necklaces and rings. Each necklace takes 4 ounces of gold and 2 diamonds to produce, each ring takes 1 ounce of gold and 3 diamonds to produce. You have 80 ounces of gold and 90 diamonds. You make a profit of 60 dollars per necklace you sell and 30 dollars per ring you sell, and want to figure out how many necklaces and rings to produce to maximize your profits.

(a) Formulate this problem as a linear programming problem. Draw the feasible region, and find the solution (state the cost function, linear constraints, and all vertices except for the origin).

(b) Suppose instead that the profit per necklace is C dollars and the profit per ring remains at 30 dollars. For each vertex you listed in the previous part, give the range of C values for which that vertex is the optimal solution.

5 Modeling: Tricks of the Trade

One of the most important problems in the field of *statistics* is the *linear regression problem.* Roughly speaking, this problem involves fitting a straight line to statistical data represented by points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ on a graph. Denoting the line by $y = a + bx$, the objective is to choose the constants a and b to provide the "best" fit according to some criterion. The criterion usually used is the *method of least squares*, but there are other interesting criteria where linear programming can be used to solve for the optimal values of a and b.

Suppose instead we wish to minimize the sum of the absolute deviations of the data from the line, that is,

$$\min \sum_{i=1}^{n} |y_i - (a + bx_i)|$$

Write a linear program with variables a, b to solve this problem.
Hint: Create a new variable z_i that will equal $|y_i - (a + bx_i)|$ in the optimal solution.

6 Max Flow with Multiple Sinks/Sources

This is a solo question.

Consider the following variation of max flow: You are given a directed graph G, with capacities on all the edges. There are 2 source vertices s_1, s_2 and 2 sink vertices t_1, t_2, and now flow is allowed to originate from either source and arrive at either sink.

Call any flow that satisfies the usual capacity constraints on all edges and balance constraints for all non-source/sink vertices a 2-flow. Informally, a 2-flow is just like a flow, except it can originate at either source and move to either sink. The size of a 2-flow is the total amount of flow leaving s_1 and s_2 (equivalently, the total amount arriving at t_1 and t_2).

Show how to construct a directed graph G' (with edge capacities and one designated source and sink vertex) such that:

(1) If F is a 2-flow in G, there is a flow F' in G' of the same size,

(2) If F' is a flow in G', then there is a 2-flow F in G of the same size.

In addition to describing how to construct G', describe how to find F' given F and vice-versa.

7 Routing Data

The internet is modelled as a directed network $G = (V, E)$, where the vertices are data centers, and edges represent connections between data centers. There are k types of data, and for the ith type of data, there is a source data center s_i, a destination data center t_i, and we want to transfer at least r_i units of this type of data through the network from s_i to t_i (we are allowed to transfer more). Using edge $e = (u, v)$, we can transfer at most c_e total units of data from u to v. For example, if $c_e = 2$, we could use e to transfer 2 units of type 1 data, or 1 unit of each of type 1 and type 2 data. Our goal is to come up with a plan for routing each type of data, so that the total amount of data transferred is maximized.

Let $f_{i,e} \geq 0$ denote the number of units of the ith type of data we transfer using edge e. Describe how to write each of the following aspects of this problem as linear programming constraints or objectives in terms of these variables.

(a) For any vertex besides s_i, t_i, the amount of data type i arriving at that vertex is the same as the amount leaving.

(b) The total amount of any kind of data being transfered through edge e is at most c_e units.

(c) At least r_i units of data type i leave s_i (and arrive at t_i).

(d) The total amount of data being transferred (i.e., the sum over the source data centers of the amount of data leaving them) is as large as possible.

8 (Extra Credit) Knightmare

Give an efficient algorithm to find the number of ways you can place knights on an N by M ($M < N$) chessboard such that no two knights can attack each other (there can be any number of knights on the board, including zero knights). Clearly describe your algorithm and prove its correctness. Your algorithm's runtime can be exponential in M but should be polynomial in N.

(Please provide a 3-part solution)