CS 170 Homework 6

Due **3/07/2022, 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, you must explicitly write "none".

2 Egg Drop Revisited

Recall the Egg Drop problem from Homework 5:

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 Knightmare

Give a dynamic programming 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. Return your answer mod 1337.

For this problem, write your answer in the following 4-part format:

(a) Define a function $f(\cdot)$ in words, including how many parameters are and what they mean, and tell us what inputs you feed into f to get the answer to your problem.

(b) Write the "base cases" along with a recurrence relation for f.

(c) Prove that the recurrence correctly solves the problem.

(d) Analyze the runtime and space complexity of your final DP algorithm. Can the bottom-up approach to DP improve the space complexity? (For example, we saw in class that Knapsack can be solved in O(nW) space, but one can reduce that to O(W) space via the optimization of only including the last two rows.)

4 Balloon Popping Problem.

You are given a sequence of n-balloons with each one of a different size. If a balloon is popped, then it produces noise equal to $s_{left} \cdot s_{popped} \cdot s_{right}$, where s_{popped} is the size of the popped balloon and s_{left} and s_{right} are the sizes of the balloons to its left and to its right. If there are no balloons to the left, then we set $s_{left} = 1$. Similarly, if there are no balloons to the right then we set $s_{right} = 1$, while calculating the noise produced.

After popping a balloon, the balloons to its left and right become neighbors. (Note that the total noise produced depends on the order in which the balloons are popped.)

Design a polynomial-time dynamic programming algorithm to compute the the maximum noise that can be generated by popping the balloons. *Hint: Read the section of the textbook on Matrix Chain Multiplication.*

Example:
Input (Sizes of the balloons in a sequence): ④ ⑤ ⑦
Output (Total noise produced by the optimal order of popping): 175

Walkthrough of the example:

- **Current State** ④ ⑤ ⑦
  Pop Balloon ⑤
  Noise Produced $= 4 \cdot 5 \cdot 7$

- **Current State** ④ ⑦
  Pop Balloon ④
  Noise Produced $= 1 \cdot 4 \cdot 7$

- **Current State** ⑦
  Pop Balloon ⑦
  Noise Produced $= 1 \cdot 7 \cdot 1$

- **Total Noise Produced** $= 4 \cdot 5 \cdot 7 + 1 \cdot 4 \cdot 7 + 1 \cdot 7 \cdot 1$.

(a) Define your subproblem.

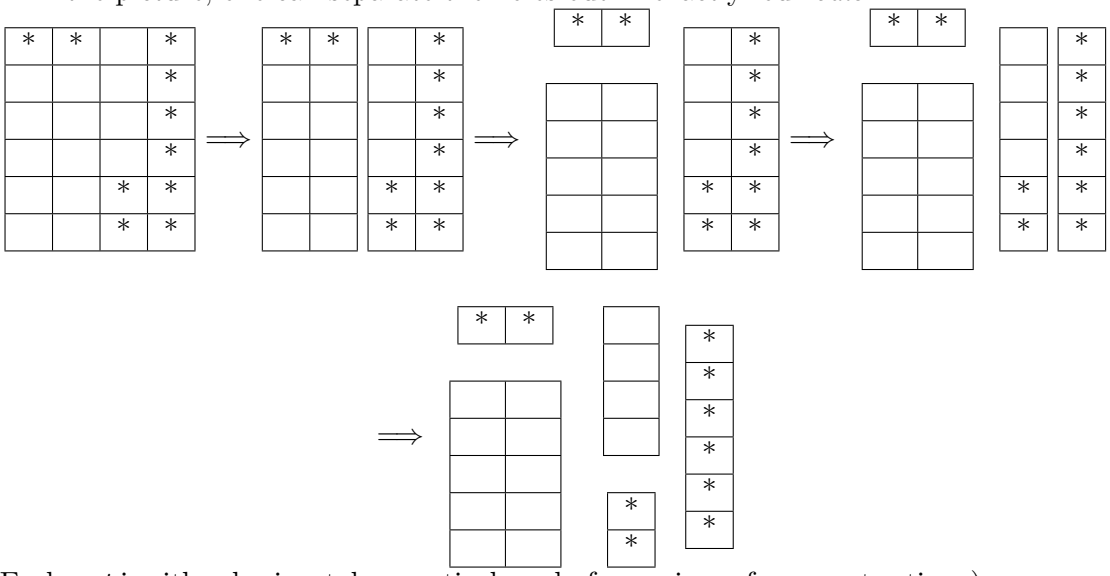
(b) What are the base cases?

(c) Write down the recurrence relation for your subproblems. What is the runtime of a dynamic programming algorithm using this subproblem?

5 Paper Cutting

There is a piece of paper consisting of an $m \times n$ rectangular grid of squares. Some of the squares have holes in them, and you cannot use paper with a hole in it. You would like to *cut* the paper into pieces so as to separate all the squares with holes from all the intact squares.

For example, shown below is a 6×4 piece of paper with holes in squares marked *. As shown in the picture, one can separate the holes out in exactly four *cuts*.



(Each *cut* is either horizontal or vertical, and of one piece of paper at a time.)

Design a DP based algorithm to find the smallest number of cuts needed to separate all the holes out. Formally, the problem is as follows:

Input: Dimensions of the paper $m \times n$ and an array $A[i, j]$ such that $A[i, j] = 1$ if and only if the ij^{th} square has a hole.

Goal: Find the minimum number of cuts needed to separate the holes out.

(a) Define your subproblem.

(b) Write down the recurrence relation for your subproblems.

(c) What is the time complexity of solving the above mentioned recurrence? Provide a justification.