CS 170 Homework 7

Due **Monday 10/21/2024, 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, write "none".

DP Solution Writing Guidelines:

When writing your solutions to DP problems only, please follow the **4-part solution** template below (unless specified otherwise):

1. Define a function $f(\cdot)$ in words. Make sure to include how many parameters there are and what they mean, and tell us what inputs you feed into f to get the answer to your problem.
2. Write the recurrence relation for f, as well as the "base cases".
3. Prove that the recurrence correctly solves the problem. In almost all cases, you will want to use induction to prove the correctness of a DP algorithm.
4. Analyze the runtime and space complexity of your final DP algorithm. Note that the top-down and bottom-up approaches to DP have the same runtime complexity; however, bottom-up can potentially yield a better space complexity.

This content is protected and may not be shared, uploaded, or distributed.

2 Egg Drop

You are given m identical eggs and an n story building. You need to figure out the highest floor $b \in \{0, 1, 2, \ldots n\}$ that you can drop an egg from without breaking it. Each egg will never break when dropped from floor b or lower, and always breaks if dropped from floor $b+1$ or higher. ($b = 0$ 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 b (regardless of the value of b).

(a) Find $f(1, m)$, $f(0, m)$, $f(n, 1)$, and $f(n, 0)$. Briefly explain your answers.

Hint: use ∞ to denote that it is impossible to find b.

(b) Consider dropping an egg at floor h when there are n floors and m eggs left. Then, it either breaks, or doesn't break. In either scenario, determine the minimum remaining number of egg drops that are needed to find b in terms of $f(\cdot, \cdot)$, n, m, and/or h.

(c) Find a recurrence relation for $f(n, m)$.

Hint: whenever you drop an egg, call whichever of the egg breaking/not breaking leads to more drops the "worst-case event". Since we need to find b regardless of its value, you should assume the worst-case event always happens.

(d) If we want to use dynamic programming to compute $f(n, m)$ given n and m, in what order do we solve the subproblems?

(e) Based on your responses to previous parts, analyze the runtime complexity of your DP algorithm.

(f) Analyze the space complexity of your DP algorithm.

(g) Is it possible to modify your algorithm above to use less space? If so, describe your modification and re-analyze the space complexity. If not, briefly justify.

This content is protected and may not be shared, uploaded, or distributed.

3 Counting Targets

We call a sequence of n integers $x_1, \ldots, x_n$ *valid* if each x_i is in $\{1, \ldots, m\}$.

(a) Give a dynamic programming-based algorithm that takes in n, m and "target" T as input and outputs the number of distinct valid sequences such that $x_1 + \cdots + x_n = T$. Your algorithm should run in time $O(m^2n^2)$. Note that you can assume $T \leq mn$ since no valid sequences sum to more than mn.

Please provide a 4-part solution.

(b) **(Extra Credit)** Give an algorithm for the problem in part (a) that runs in time $O(mn^2)$.

Please provide the subproblem definition, recurrence relation (including base cases), and the runtime/space complexity analyses. You do not need to provide a proof of correctness.

This content is protected and may not be shared, uploaded, or distributed.

4 String Shuffling

Let x, y, and z be strings. We want to know if z can be obtained only from x and y by interleaving the characters from x and y such that the characters in x appear in order and the characters in y appear in order.

For example, if $x =$ **prasad** and $y =$ **SANJAM**, then it is true for $z =$ **praSANsadJAM**, but false for $z =$ **prasadSANJAMpog** (extra characters), $z =$ **prasSANJAad** (missing the final **M**), and $z =$ **prasadASNJAM** (out of order). How can we answer this query efficiently? Your answer must be able to efficiently deal with strings with lots of overlap, such as $x =$ **aaaaaaaaaab** and $y =$ **aaaaaaaac**.

(a) Design an efficient algorithm to solve the above problem, briefly justify its correctness, and analyze its runtime. You do *not* need to provide a space complexity analysis (you'll do this in the next part!).

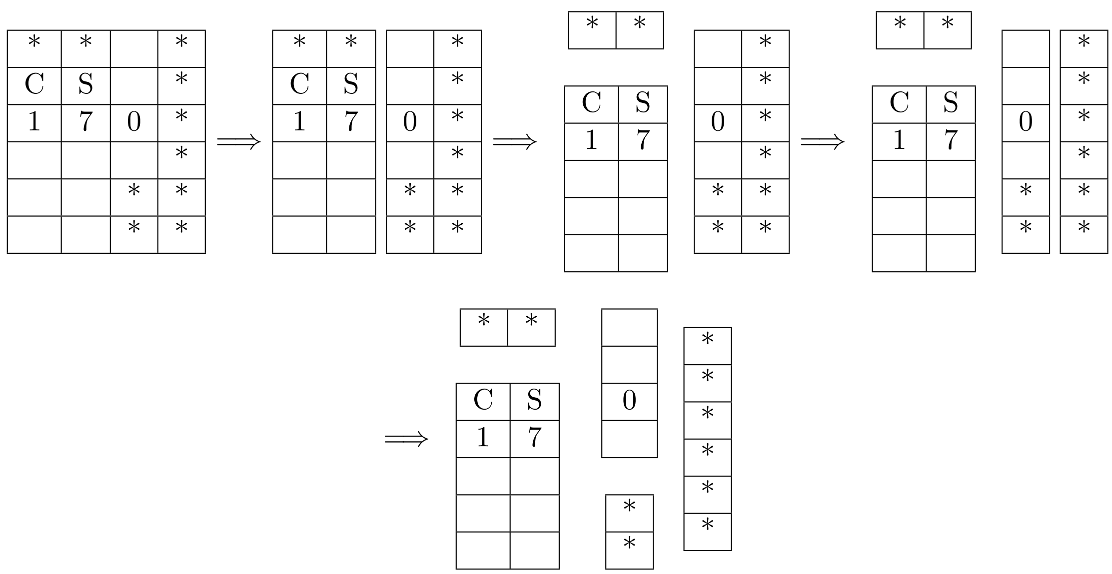
(b) Consider a bottom-up approach to our DP algorithm in part (a). Naively if we want to keep track of every solved sub-problem, this requires $O(|x||y|)$ space (double check to see if you understand why this is the case). How can we reduce the amount of space our algorithm uses?

This content is protected and may not be shared, uploaded, or distributed.

5 My Dog Ate My Homework

One morning, you wake up to realize that your dog ate some of your CS 170 homework paper, which is an $x \times y$ rectangular grid of squares. Some of the squares have holes chewed through them, and you cannot use paper that has a hole in it. You would like to cut the paper into pieces so as to separate all the tattered squares from all the clean, un-bitten squares. You want to do this so that you can save as much as your work as possible.

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



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

Formally, the problem is as follows:

> **Input:** Dimensions of the paper $x \times y$ and an array $P[i, j]$ such that $P[i, j] = 1$ if and only if the ij^{th} square has holes bitten into it.
>
> **Goal:** Find the minimum number of cuts needed so that the $P[i, j]$ values of each piece are either all 0 or all 1.

Design a DP-based algorithm to find the smallest number of cuts needed to separate all the bitten parts out in $O(x^3y^3)$ time. For **extra credit**, try to optimize your algorithm to achieve a runtime of $O((x + y)x^2y^2)$.

(a) Define your subproblem.

Hint: try making any arbitrary cut. What two subproblems do you now have? What parameters do you need to properly handle recursing on these two resulting subproblems?

(b) Write down the recurrence relation for your subproblems. A fully correct recurrence relation will always have the base cases specified.

This content is protected and may not be shared, uploaded, or distributed.

(c) Describe the order in which we should solve the subproblems in your DP algorithm.

(d) What is the runtime complexity of your DP algorithm? Provide a justification.

(e) What is the space complexity of your algorithm? Provide a justification.

This content is protected and may not be shared, uploaded, or distributed.

6 [Coding] More Advanced Dynamic Programming

For this week's homework, you'll implement implement some more dynamic programming algorithms. There are two ways that you can access the notebook and complete the problems:

1. **On Datahub**: click here and navigate to the `hw07` 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-fa24-coding`

   and navigate to the `hw07` 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 "Homework 7 Coding Portion".
- *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.

This content is protected and may not be shared, uploaded, or distributed.