CS 170 Homework 5

Due **Monday 2/23/2026, at 10:00 pm (grace period until 11:59pm)**

Study Group

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

1 Motel Choosing (Solo Question; 5 points)

You are traveling a long distance by car, and wish to travel as close to T miles per day as possible. On a day you travel x miles, you receive a penalty of $|T - x|^2$. Your goal in this question is to minimize the total penalty in the following setting: there are several motels on your route, and each day, you start where you ended the previous day and you must stop at some motel at the end of the day (on the first day you start at position 0). The motels are at specified positions $0 < r_1 < \cdots < r_n$. Your destination is the final motel at position r_n.

Describe an efficient algorithm that outputs the minium penalty, given the locations r_i of the motels and the value of T. Explain why your algorithm is correct, and analyze the runtime.

2 Making Change (5 points)

You are using a currency with coins of n different positive integer values $v_1, \ldots, v_n$. Give an efficient algorithm that takes as input an integer m along with the values $v_1, \ldots, v_n$, and outputs the number of different combinations of coins that have value m. Explain why your algorithm is correct, and analyze the runtime.

For example, if $m = 5$ and there are $n = 3$ coins with values $v_1 = 1$, $v_2 = 3$, $v_3 = 4$, then then there are 3 valid combinations of coins: $1 + 1 + 1 + 1 + 1$, $1 + 1 + 3$, and $1 + 4$.

3 Minimizing Lateness (5 points)

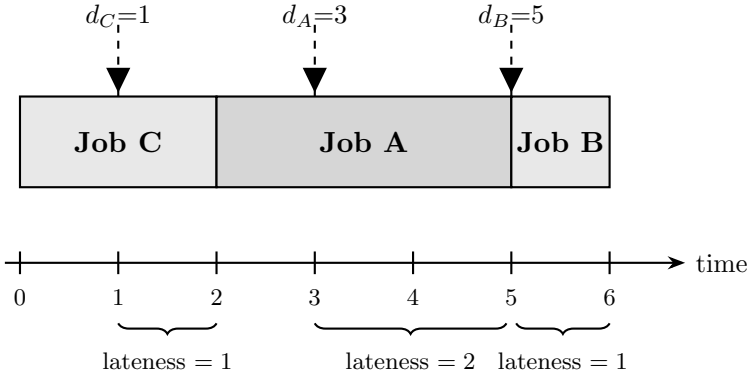
You have n different jobs that you need to perform one at a time. Each job i takes some number t_i of hours, and has some deadline d_i (measured in hours from when you start your first job). If you finish job i at some time e_i after the deadline d_i (i.e. $e_i > d_i$), then it is $e_i - d_i$ hours late, so we say it has "lateness" $e_i - d_i$. If you finish job i before the deadline d_i, it then is done on time, and has lateness 0. Let L be the maximum lateness of any job. Naturally, you want L to be as small as possible.

Give an efficient algorithm that finds an order of the jobs with the smallest possible value of L. Explain why your algorithm is correct.

Example Input

Job A: t_A=3, d_A=3 | Job B: t_B=1, d_B=5 | Job C: t_C=2, d_C=1

Best Schedule



$L^* = \max(1,\ 2,\ 1) = 2$

4 Egg Drop (7 points)

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). You may reuse an egg until it breaks, at which point you cannot use it any more. Each egg drop requires effort (climbing a lot of steps), so given n and k we want to compute the minimum number of egg drops needed to determine the value of ℓ.

Notice that one egg is sufficient to determine the value of ℓ, if we are willing to use up to n eggs drops — simply drop it from every floor, starting from floor 1 and going up one floor each time until it breaks. If we have more eggs to work with we might take more risks and use fewer drops. For example with 2 eggs, we can drop the first from floor $n/2$, and then depending on the outcome select which at most $n/2$ drops of the second egg would determine the answer.

Given k eggs and n floors, let $f(n, k)$ be the minimum number of egg drops that are needed to find ℓ.

(a) (2 points) Assume for simplicity that n is a power of 2. Show that with $k = \log_2(n) + 1$ eggs, you can determine the floor ℓ by dropping each egg exactly once. That is, $f(n, \log_2(n) + 1) \leq \log_2(n) + 1$.

Hint: Binary search.

(b) (5 points) Give a dynamic programming algorithm to compute $f(n, k)$ (no longer assuming n is a power of 2). To do so, you should write a recurrence relation for $f(n, k)$ in terms of smaller values of n and k.

5 DNA or Coding (7 points)

For full credit, you should do **one of the following two** questions. (You may solve the other for fun if you want!)

1. **DNA**

   This question is intentionally exploratory and open-ended; there is not a single correct answer.

   A DNA sequence can be expressed as a sequence of 'A', 'T', 'C', and 'G' characters, such as "AGGCTTACAT". Such sequences are inevitably modified in organisms over time, meaning that characters can be added, removed, or changed.

   (a) (3 points) Propose a way to measure how different two DNA sequences are. That is, for every pair of sequences s_1, s_2, your measure should specify a nonnegative integer $m(s_1, s_2)$, which is smaller for more "similar" sequences.

   In particular, a pair of identical sequences $s_1 = s_2$ should be given value $m(s_1, s_2) = 0$. Meanwhile, inserting or deleting a single character in a sequnce s_2 should only change $m(s_1, s_2)$ by at most 1.

   Argue why your measure is a meaningful notion of similarity, and give an efficient dynamic programming algorithm to compute your measure.

   (b) (4 points) A common problem called "read mapping" is that you are given a short (~ 100 character) sequence, and you want to find where it fits within a much longer (≥ 1 billion character) sequence. Will your approach from part (a) still work well in this setting? Explain why or why not, and if not, modify your approach to address the issues.

   (c) (3 points; you may do this part **in place of part (a)** for full credit) Look up the history of shotgun sequencing and paired-end sequencing in the context of the human genome project, and write a brief paragraph summarizing what you understood.

2. **Coding** (7 points)

   For this weeks coding questions, we'll implement the Edit Distance algorithm you saw in lecture. There are two ways that you can access the notebook and complete the problems:

   (a) **On Datahub**: click here and navigate to the `hw04` folder.

   (b) **On Local Machine**: `git clone` (or if you already cloned it, `git pull`) from the coding homework repo,

   https://github.com/Berkeley-CS170/cs170-sp26-coding

   and navigate to the `hw04` 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 "HW05 (Coding)".
- *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:

  (a) Describe the steps you've taken to debug the issue prior to posting on Ed.

  (b) Describe the specific error you're running into.

  (c) 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.

