CSE 417 25au Homework 5: Dynamic Programming

Released: Friday, October 31, 2025 @ 11:30am

First due by: Friday, November 7, 2025 @ 11:59pm

Last resubmissions by: Wednesday, November 19, 2025 @ 11:59pm

Instructions

For Problems 9, 9X.1, and 10, you have four options for submission:

- **Film a video in which you explain your solution.** See the Homework Guide for more details.
- **Use LaTeX to type your solutions.** A template is provided in the "Tasks" page of the course website, if you like.
- **Use Google Docs or Microsoft Word to type your solutions.** If doing so, please use the Equation Editor to ensure that any equations are legible and easy to read.
- **Handwrite your solutions on paper or digitally.** Please write neatly and if on paper, scan in black/white mode, not grayscale.

We prefer either video or LaTeX, but accept any of the 4 options.

A few more reminders:

- **Submit all problems on Canvas.** Each problem should have its own submission. *Do not* submit one large file containing answers to several problems.
- **Suggested word counts are rough guidelines.** We won't actually count, but if your writing is verbose to the point of obscuring your main argument, we may ask you resubmit more concisely.
- **Review the collaboration policy in the syllabus**. Collaboration is encouraged but strict rules apply, and remember to cite your collaborators.
- If you don't finish in time, we encourage you to be honest and just upload what you have so far. Resubmission won't cost you anything, and we can give you timely feedback on your partial progress by submitting on time.

Happy problem solving!

Problem 9: Better edit distance

The purpose of this problem is to reinforce and extend the algorithm for edit distance from class.

In class, we discussed how to compute the "distance" between two strings, given two parameters: a "gap penalty" δ for any insertion or deletion of a character between two strings (i.e. between "chat" and "cat") and a "mismatch penalty" $\alpha(x, y)$ for swapping one character out for another (i.e. the penalty is $\alpha(\text{c}, \text{t})$ between "chat" and "that"). For simplicity, assume that $\alpha(x, y) = \alpha(y, x)$ and $\alpha(x, x) = 0$. It is allowed to perform two gaps instead of a mismatch if it is cheaper, meaning that we would align

x:	c		h	a	t		x:	c	h	a	t
y:		t	h	a	t	instead of	y:	t	h	a	t
cost:	δ	δ	0	0	0		cost:	$\alpha(\text{c}, \text{t})$	0	0	0

if $\alpha(\text{c}, \text{t}) > 2\delta$. The *edit distance* is the minimum total penalty accrued among all possible ways to go from one string to the other via insertion, deleting, and swapping letters.

In real life, especially in biological situations involving DNA, a flat δ penalty for every gap is not very realistic. For example, the sequence "AAAA" has a much higher chance of being biologically miscopied as "AATTTTAA" than "ATATATAT", despite both having a gap penalty of 4δ in both cases by the above definition. Thus, in this problem, let δ **be the penalty for the first gap in a row**, and let ϵ **be the additional penalty for each additional consecutive gap** in the same string. For example, the gap penalty between "AAAA" and "AATTTTAA" is $\delta + 3\epsilon$, and the gap penalty between "AAAA" and "ATATATAT" is 4δ.

The *better edit distance* is the minimum total penalty to go between two strings, using this updated notion of gap penalty and the original notion of mismatch penalty.

Figure out how to use a dynamic programming algorithm to compute this better edit distance between any two input strings $x = x_1 x_2 \ldots x_m$ and $y = y_1 y_2 \ldots y_n$. Here are a few hints:

- Start with the recurrence from the original edit distance problem. What additional information can you keep track of in order to determine which penalty to apply?
- We've written a post, now on the course website (linked under Tasks) and pinned on Ed, that describes a technique that could be helpful to this problem. Read it!

Then answer the questions below (which are written in an appropriate order for explaining your solution, NOT necessarily the order in which you should brainstorm how to solve the problem).

1. Describe in English what your recurrence will compute, in terms of its parameters. If you have more than one recurrence, do this for each one. Be unambiguous—the reader should be able to compute exactly what you mean via brute force after reading your description (and the definitions in the problem statement). Then answer: In terms of your recurrences, what is your final output?

   (For example, for original edit distance, you could say, "dist(i, j) is the edit distance between strings $x_1 x_2 \ldots x_i$ and $y_1 y_2 \ldots y_j$. The final output should be dist(m, n).")

2. Write the recurrence(s) that compute the better edit distance described above. Be sure to write the base case(s) somewhere.

   (For example, for original edit distance, you could say that

$$\text{dist}(i,j) = \min \begin{pmatrix} \text{dist}(i-1, j-1) + \alpha(x_i, y_j) \\ \text{dist}(i-1, j) + \delta \\ \text{dist}(i, j-1) + \delta \end{pmatrix}$$

   with base cases $\text{dist}(k, 0) = \text{dist}(0, k) = k\delta$ for all k.)

3. Describe the size and shape of the data structure(s) that you need to implement this recurrence. In what order would you fill the entries of the data structure(s) for a bottom-up implementation? What is the total running time in big-O?

4. Follow your algorithm to compute by hand the better edit distance between "AA" and "ATT", given $\alpha(\text{A}, \text{T}) = 5$, $\delta = 3$, and $\epsilon = 1$. The answer should be 7. (If you do not get 7, double check your recurrence because it is probably wrong.) Provide table(s) similar to slide XX(TODO) from Lecture 4.3 (a.k.a. Lecture 16).

(There are no suggested word counts for this question because each part is just a short answer, not long explanation. Video submissions should take 4–6 minutes total.)

Problem 9X: Better edit distance (Extensions)

This is an extension problem that builds on the ideas of Problem 9. **Pick one** of the following to complete:

1. This question extends the edit distance problem in a different way. For this problem, we will think about the original edit distance problem from class for simplicity, though similar ideas would also work to extend Problem 9. Recall that the original edit distance algorithm from class used $O(mn)$ time, where the two strings have m and n letters, respectively. This is considered quadratic time, and is a significant barrier for applications when m and n are large. Computer scientists believe that $\Omega(mn)$ time is necessary for edit distance in general (though still an open problem), but this is not the case when the two strings are already guaranteed to be similar.

   In particular, line up the two strings, demonstrating where there are gaps and mismatches. For conciseness, we refer to both gaps and mismatches as *edits*. Count the *number* of edits (NOT the edit *distance* with α and δ). Suppose you are guaranteed that the optimal edit distance requires no more than k edits, for some constant k. Then you can actually find the exact edit distance using linear time!

   (a) Recall that dist(i, j) denotes the edit distance between substrings $x_1x_2 \ldots x_i$ and $y_1y_2 \ldots y_j$. Suppose that for the given input, the optimal edit distance requires $\leq k$ edits. Quickly prove that whenever $|i - j| > k$, the optimal alignment for the subproblem dist(i, j) is not used as part of the optimal alignment for the whole problem. (suggested 30–80 words)

   (b) Assume that you wanted to compute edit distance using the bottom-up method. Under the guarantee that the optimal edit distance requires $\leq k$ edits, describe new base cases and a new order to compute edit distance while filling only $O(n)$ entries of the $m \times n$ array dist, where dist$[i][j]$ holds the the edit distance between substrings $x_1x_2 \ldots x_i$ and $y_1y_2 \ldots y_j$. A picture may help. (suggested 30–80 words)

   (c) In most programming languages (as well as the standard mathematical model of computation for algorithms), initializing an $m \times n$ array still takes $O(mn)$ time, even if you don't fill it with anything. Describe briefly how you can avoid creating this large array, in order to compute the edit distance in $O(n)$ time. (As usual, you should not use hash maps, because their worst-case performance is very slow when there are collisions.) (suggested 25–50 words)

2. One important application of edit distance is to compare one's genome with a reference genome to identify the most likely gaps and mutations, which can be predictors of future disease and illnesses. Companies such as 23andMe provide these services to the public. However, there are significant privacy concerns with the proliferation of genetic data, as well as concern that genetic data can be used for harm.

   Read about the Genetic Information Nondiscrimination Act of 2008. Then, join the discussion on Canvas by responding to the following prompt, raising other questions of your own choosing, or replying directly to other students' responses. (suggested 150–300 words)

> What are some positive and negative ways that genetic data can be used? What size role should government play in regulating genetic data? Should algorithm designers for algorithms like edit distance (or newer algorithms used in bioinformatics) bear responsibility for downstream concerns of whether they are used for good or bad purposes?

You are *not* required to cite the article or any other sources, but be sure to support your arguments with clear reasons. If you discuss any technical information that is not in the lecture slides or this reading, please cite your sources. **Resubmissions will not be available for this part.**

Problem 10: Planning a date

You are planning a date for you and your partner, with a total budget of C dollars, and the date should last at most T minutes. You have a list of n possible destinations to visit, and each destination has a cost c_i, time t_i, and will give you and your partner h_i units of happiness (all integers). Additionally, assume that going to every destinations takes an additional 30 minutes of driving. Include the time it takes to drive to the first destination, but not the time it takes to drive back from the last destination.

For example, with \$110 and 300 minutes (5 hours), you and your partner could: Start at the theater, which costs \$40 and 120 minutes, and gain 5 happiness. Then spend 30 minutes driving to a restaurant, where you spend \$60 and 60 minutes, and gain 10 happiness. Finally spend 30 minutes driving to the park to take a walk, which is free and lasts another 30 minutes, and you gain another 7 happiness. This itinerary gives you 22 happiness and satisfies the constraints because:

- You spent $\$40 + \$60 + \$0 \leq \$110$.
- You spent $120 + 60 + 30 = 210$ minutes in activities, plus $30 \times 3 = 90$ minutes driving, which is $210 + 90 \leq 300$ minutes total.

Figure out how to use a dynamic programming algorithm to compute the maximum happiness you can get with the given inputs. Here is a hint to get you started:

- Follow the strategy described in class, to first consider whether to include or exclude the last destination. In each case, what is the remaining subproblem?

Then answer the questions below (which are written in an appropriate order for explaining your solution, NOT necessarily the order in which you should brainstorm how to solve the problem).

1. Describe in English what your recurrence will compute, in terms of its parameters. If you have more than one recurrence, do this for each one. Be unambiguous—the reader should be able to compute exactly what you mean via brute force after reading your description (and the definitions in the problem statement). Then answer: In terms of your recurrence(s), what is your final output?
2. Write the recurrence(s) that compute the better edit distance described above. Be sure to write the base case(s) somewhere.
3. Describe the size and shape of the data structure(s) that you need to implement this recurrence. In what order would you fill the entries of the data structure(s) for a bottom-up implementation? What is the total running time in big-O?
4. Explain how you could compute the actual list of destinations that achieves the maximum happiness, instead of just the value of the maximum happiness. (suggested 50–100 words, pseudocode optional)
5. In English, when you say a date lasted 5 hours, you usually don't count the time it takes to drive to the first destination—the date starts there. In one sentence, say how to modify your solution to only count driving time between destinations, so that if you visit k destinations, you only count driving $k - 1$ times during the date.

(Video submissions are encouraged to take 3–6 minutes total.)