Homework 1

CS 1810, Fall 2025

Out: Sept. 15
Due: Sept. 26, 11:59 PM EST

Please upload your solutions on Gradescope. You can use LaTeX or a word document to write up your answers, but we prefer you use LaTeX. You may scan hand-written work or images for parts of solutions **only if** they are extremely clean and legible. Please ensure that your name does not appear anywhere in your handin.

We highly recommend that you do not use AI to answer these questions as they serve as good practice of the concepts that you will need to apply on your exams. However, if you use any sort of AI, we ask that you write a 500 word reflection on your AI usage for every part of every question for which you use AI. If we have reason to suspect AI usage and no reflection is written, you will receive 0 points on that question.

Problem 1: Global alignment table

It is a warm summer night at Mushroom Gorge, and the kart racers are hungry after their race! Mario is especially hungry and is excited to munch on mushrooms for dinner. Inspired by this tranquil scene, you (a computational biologist) decide to perform a global alignment on `MARIO` and `MUSHROOM`!

Create and fill out the global alignment dynamic programming table to align `MARIO` and `MUSHROOM`. For scoring, use the match bonus $+1$, mismatch penalty -1, and indel penalty -1. When there is a tie between a mismatch and an indel alignment, use a back pointer that indicates a mismatch. Make sure to include back pointers for all cells in the table. If there is a tie between indels, you can include both arrows. What is the score of the optimal alignment and to which alignment does this score correspond?

Problem 2: Backtracking

One way to obtain the optimal alignment between two sequences is to first fill out the dynamic programming table using the Needleman-Wunsch algorithm and then backtrack from the bottom right cell of the table, $S_{m,n}$, where m, n represent the lengths of the two sequences. To do so, you would start from the value of $S_{m,n}$, recalculate which cell this max value originated from, and then move to the corresponding cell. While keeping track of your path through the table, you would continue to recalculate and backtrace until you reach the top left cell, $S_{0,0}$. Since there can be ties between matches/mismatches and indels, depending on the scoring scheme, it is possible to have multiple optimal alignments for two sequences.

Note: However, the more computationally efficient solution is to place backpointers as you fill out the dynamic programming table and compute the value in each cell. Afterwards, you can follow the existing backpointers from $S_{m,n}$ to $S_{0,0}$ to determine the optimal alignment.

For this problem, we have already filled out a dynamic programming table for globally aligning `CGCGAGTCG` and `CGACG`. The scoring scheme for this alignment is a match bonus of $+1$, mismatch penalty of -1, and an indel penalty of -1.

	-	C	G	C	G	A	G	T	C	G
-	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
C	-1	1	0	-1	-2	-3	-4	-5	-6	-7
G	-2	0	2	1	0	-1	-2	-3	-4	-5
A	-3	-1	1	1	0	1	0	-1	-2	-3
C	-4	-2	0	2	1	0	0	-1	0	-1
G	-5	-3	-1	1	3	2	1	0	-1	1

a. Fill out the backpointers in the dynamic programming table above and include both backpointers when there is a tie between a match/mismatch and an indel. Give all possible optimal alignments from this dynamic programming table and their alignment score.

b. If we were to prioritize a match/mismatch over an indel when there is a tie, what would be the optimal alignment?

c. The time-complexity of the Needleman-Wunsch algorithm, which is used to fill out the global alignment dynamic programming table, is $\mathcal{O}(mn)$. What is the time-complexity of backtracking through the table to get the optimal alignment? (Assume that ties are resolved to produce a single optimal alignment.) Explain your answer.

Problem 3: Multiple alignment

In class, you have learned about the global alignment of two strings. A natural question to ask is how we might go about aligning three strings. Your task is to design an efficient algorithm that solves the following problem.

> GLOBAL ALIGNMENT OF THREE STRINGS
>
> **Input:** Strings u, v, and w and a scoring function δ that takes in three arguments.
>
> **Output:** An alignment of u, v, and w for which the score is maximal (as defined by δ) among all alignments of u, v, and w.

Whereas we had to work fairly hard to build up the original binary global alignment algorithm from scratch, it turns out that we can obtain an algorithm for the global alignment of three strings by making some careful modifications to the algorithm you've seen in class.

In your solution, you *must*

- describe how the structure of the dynamic programming table changes in the extension to three strings,
- give a new recurrence relation for finding the score of an optimal alignment of a given combination of prefixes of u, v, and w,
- specify the initialization and order in which the new dynamic programming table should be filled out.

If you wish, you may also include diagrams, pseudocode, mathematical expressions, and plain English text in your answer. However, please do not submit a block of code with no accompanying explanation.

Once we know how to extend our global alignment algorithm to three strings, we might consider making the general extension to k strings:

- Show that if we continue to insist on constructing dynamic programming tables, the runtime of our algorithm will be $O((2n)^k)$ for k sequences of length n. Since this approach is exponential in k, we will quickly hit a computational wall in attempting to align sequences of even modest length.

  Hint: How many cells are in the new dynamic programming table and how many operations are in the new recurrence relation?

Bonus: Sketch in a few sentences a reasonable heuristic we might use to skirt this problem if we would still like to align multiple sequences.

Problem 4: Counting global alignments

When implementing algorithms, there are different approaches to solving a problem. Brute force programming is a direct approach to solving a task that depends more on computer processing power and memory, while dynamic programming optimizes by breaking a task into smaller problems and drawing upon solutions to the smaller problems in order to solve the overall task.

A brute force approach to global alignment would iterate through every possible alignment of the two input strings, computing the score and storing the max along the way. To understand the performance of such an approach, we need to know the number of possible alignments between two strings, one of length m and the other of length n. Finding an exact expression for the number of possible alignments is actually fairly difficult. In this problem, we simply ask you to determine whether the number of alignments is

1. polynomial in m and n
2. logarithmic in m and n
3. exponential in m and n

Indicate your answer choice by number and justify your response.

Problem 5: A Fowl Virus

Tired of racing Moo Moo Meadows, you decide to create a new racing course called Cluck Cluck Farm. One day, while tending to the chicken coop, you have a disturbing realization: the chickens have started dying at alarming rates! Being the seasoned computational biologist you are, you realize that there is a viral epidemic ravaging the chicken coops. Luckily, you have a sequencing center next to the racing course. You manage to isolate some of the virus and sequence it. Now, with the sequence in your flash drive, it is up to you to save the chickens!

Here, you'll find the sequence for the terrible virus plaguing the fowl. You know that the virus is a retrovirus and is inserting genes into the poor hens' DNA. It's up to you to find out what gene or genes are causing this farmhouse mayhem. Fortunately, you know about BLAST (Basic Local Alignment Search Tool). Navigate to the BLAST website. You're only interested in your chicken's genome so type "chickens" into the species search box and click "search." Now, to find out what gene the virus is infecting your chickens with, copy and paste the viral genome into the box asking for a FASTA sequence and click "BLAST." Once NCBI returns your query, investigate the alignments it returns and try to get to the bottom of this plague.

a. Figure out what gene your chickens are being infected with and what disease is killing them as a result. Specifically, make sure you check out the alignment for chromosome 20! Be sure to provide justification for your answers. The virus itself has an interesting history; if you're interested, try to figure out what virus it is exactly.

Hint: Pay attention to the **features** section and look up specific words to figure out this mysterious disease.

Bonus: The general structure of a retrovirus genome is composed of coding regions for the gag-pol-env polypeptides (NOT proteins) and other proteins that assemble the polypeptides into proteins. Using this information, find a closely related virus for the virus above. Give a short description of how you found the related protein.