Homework 2

CS 181, Fall 2025

Out: Sep. 27
Due: Oct. 6, 11:59 PM

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.

Problem 1: Free End Gap Alignment

In class you will soon be learning about motifs. Wikipedia defines a motif as "a nucleotide or amino-acid sequence pattern that is widespread and has, or is conjectured to have, biological significance". Combinatorial pattern matching algorithms are often used to find motifs, but in this homework we'll see how alignment algorithms can be used for the same task.

Let's design an alignment algorithm to find motifs for us. Global alignment won't work for the motif problem because it will align our short motif to a large region of the longer string with many gaps. Since we expect our motif to match on a small area, we might guess that local alignment is appropriate. But local alignment will also have trouble because we want to align the *entirety* of our motif, and local alignment may pick out a subsequence of the motif. To sidestep these problems, we introduce a new kind of alignment: *Free End Gap Alignment*. We want to align the entire motif to a subsequence of the longer sequence, with as few gaps in the aligned section as possible. To this end, we will discount any gaps that occur on the ends of a sequence. For example, the optimal **global** alignment of `GTAGGCTTAAGGTTA` and `TAGATA` is

```
GTAGGCTTAAGGTTA
-TAG----A---T-A
```

with a score of -3 (assuming our usual scoring scheme of +1 for match, -1 for gap and mismatch). An optimal Free End Gap Alignment, however, would be

```
GTAGGCTTAAGGTTA
-TAGA--TA------
```

with a higher score of 2. Indeed, this latter alignment better captures the essence of what it means to find a motif in a longer sequence. We formalize the Free End Gap Alignment problem as follows:

FREE END GAP ALIGNMENT

Input: Strings u, v, and a scoring function δ that takes in two characters as arguments.

Output: An alignment of u, v, for which the score is maximal (as defined by δ) and for which end gaps are free (i.e. not penalized) in the shorter sequence.

a. Design a dynamic programming solution to the Free End Gap Alignment problem. Your solution must include

- a new recurrence relation, if necessary
- a description of how your table is initialized
- how to backtrack through your table, including where to start and where to end

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.

Now to see the algorithm at work. We've implemented Free End Gap Alignment for you already. Enter your CS181 Projects directory and run `git pull`. This will create the `hw2` directory containing the FEGA scripts: `FEGAlignment.sh` and `fegalignment.pyc`. The pyc file is a compiled python file that has been compiled within the course Docker container.

The implementation takes in two arguments, a long sequence and a short sequence to align it against (the scoring scheme is the standard scheme mentioned prior). The code will then output *all* optimal alignments. Here's an example of how to run our implementation:

```
> sh FEGAlignment.sh GTAGGCTTAAGGTTA TAGATA
GTAGGCTTAAGGTTA
-TAGA-T-A------
GTAGGCTTAAGGTTA
-------TA-GAT-A
```

b. Use our implementation to count the number of DNA fragments that are created by exposing the following sequence (all three lines make up one contiguous sequence, written 5' to 3') to the restriction enzyme *Bam*HI, which has the recognition site *G/GATCC*. How would your answer change if the polarity of the following sequence was reversed? **Please enter the Docker container before running our implementation.**

```
TCCATTGATGCCACGGCGGATCCTGGAGAGCAGCAGCGACTTGCATACATCAGATCAGAGTAATACTAGC
ATGCGATAAGTCCCTAACTGACTATGGATCCTTCTAGAGTCAACTTCAGGACATATGGTCTCTGGATCCC
GTGGATCCTTCCTAGGAATCAGATTGGATCCTGGTTAACCATCAAACAGGTCTTGAGTCTAAAATTGTCG
```

Problem 2: Affine Gap Alignment

Affine gap penalties provide a more biologically realistic model for sequence alignment by charging differently for gap opening versus gap extension. We will define affine alignment in this problem as we have been in class, using the objective function

$$\alpha(\text{\# matches}) - \beta(\text{\# mismatches}) - \gamma(\text{\# gap clusters}) - \epsilon(\text{\# single-letter gaps})$$

Consider the sequences:

```
u = ACGTAG
v = AGTAG
```

With scoring parameters: $\alpha = +2, \beta = 1, \gamma = 3, \epsilon = 1$.

a. The affine gap alignment algorithm uses four matrices: V, E, F, and G. Define what each matrix represents and write the recurrence relations for all three matrices. Be sure to specify the initialization conditions.

b. Using the sequences and parameters above, fill in the four matrices V, E, F, and G for positions $(i, j) = (2, 2)$, $(2, 3)$, and $(3, 2)$. Show your calculations step by step.

c. Compare the optimal alignment score of the full length sequences using affine gaps versus linear gaps (with gap penalty = -2). Which alignment would be preferred biologically and why?

Problem 3: Homology

Finding conserved patterns across different species is important for evolutionary biology. Consider the following sequences:

A : 5′ AGCTTCGAAGTTATCTTGGACGGACTTG 3′
B : 5′ AGTTTCCCAGGATATCTTCGAACGACTG 3′
C : 5′ AGGCTTCCCATCCTCCTATAAAGGTAGG 3′

We wish to find whether B or C is a homologous protein to A, that is, a protein that originates from some ancestral species. The optimal alignment for A and B is

A: AGCTTCGAAGT-TATCTTGGA-CGGACTTG
B: AGTTTCCCAGGATATCTTCGAACG-ACT-G

with a score of 12. The optimal alignment between A and C is

A: AG-CTTCGAAGT--TATCT-TGGACGGACTTG-
C: AGGCTTCCCA-TCCTC-CTATAAA-GG--TAGG

with a score of 1. After splicing the DNA into cells, you find that the DNA actually transcribes the following protein sequences (using the one-letter amino acid code):

A: SSYLGR
B: SDIFER
C: SSYKGR

A gene is composed of segments of protein-coding DNA called *exons*, which are split by segments of noncoding DNA called *introns*. We can split genes A, B, and C into introns and exons as follows (exons are underlined and all other regions represent introns).

A :5′ AGCTTCGAAGTTATCTTGGACGGACTTG 3′
B :5′ AGTTTCCCAGGATATCTTCGAACGACTG 3′
C :5′ AGGCTTCCCATCCTCCTATAAAGGTAGG 3′

a. Notice how similar A and C are as proteins despite being much more dissimilar than A and B as DNA sequences. Give a biological reason for this observation (*hint*: look at the segmentation of the genes into introns and exons!). Why might it be a better idea to align by amino acid sequence rather than DNA?

b. There are 20 different amino acids and only 4 different nucleotides (used in DNA). Give a probabilistic reason why aligning by amino acid sequence might be better than by DNA sequence.

Problem 4: Statistical Foundations of Sequence Alignment

The most popular type of substitution matrices are known as BLOSUM matrices. Blocks, or fixed regions in a given set of aligned sequences, are used to create these substitution matrices. We are studying sequence similarities among some breeds of cattle on Sorin's farm and will analyze BLOSUM matrices derived from their genomes.

Given an alphabet, $\sum = \{\alpha, \beta, \gamma, \delta\}$, the following block contains the sequences of a single, 3-residue protein from each of nine cattle breeds: angus, belted galloway, brahman, charolais, dexter, gelbvieh, hereford, holstein, limousin.

Angus	α	γ	δ
Belted Galloway	β	γ	α
Brahman	γ	γ	γ
Charolais	β	β	γ
Dexter	β	γ	δ
Gelbvieh	γ	α	δ
Hereford	α	β	β
Holstein	γ	γ	δ
Limousin	β	γ	β

By determining the evolutionary information in the above block, we will develop an improved scoring scheme for sequence alignment.

a. What is the total number of residues we have in this block? How many ways can we pick a pair of letters in each column? How many possible ways of picking letters from the above table are there? Assume the paired letters must be in the same column.

b. Determine the observed frequencies for each alignment pair.

c. For each alignment pair, determine the expected frequency based on the above block, and calculate the log-likelihood score, f, that compares the observed and expected frequencies for each pair.

d. Explain the meaning of positive, negative, and zero log-likelihood scores. Given this interpretation, why would using log-likelihood scores as the scoring scheme be desirable for sequence alignment?

e. Interpret the final log-likelihood scores in this problem in terms of what they indicate about conservation of letters in this alphabet. Comment on how realistic this scoring scheme would be if we were to use it to align actual genome sequences.

f. **Bonus:** More formally, we can express the log-likelihood score of aligning two letters a and b as:

$$f_{ab} = 2\log_2 \frac{p_{ab}}{q_{ab}}$$

where:

i. $p_{ab} = \mathbb{P}(a, b|M)$ is the joint probability of observing aligned letters a and b under the alignment model M

ii. $q_{ab} = \mathbb{P}(a, b|R) = \mathbb{P}(a|R)\mathbb{P}(b|R)$ is the joint probability of observing aligned letters a and b under a random model where letters appear independently.

Recall that local alignment requires the expected score of aligning two letters to be non-positive to ensure that alignments remain local. Therefore, if we want to use BLOSUM matrices for local alignment, we need to ensure that the expected score f_{ab} is not positive.

Prove that the expected score $\mathbb{E}[f_{ab}]$ for a randomly aligned pair of letters is not positive for any alignment model M.

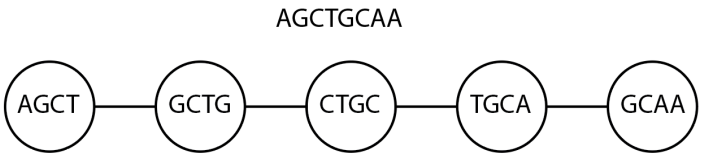
Hint: For any concave function g, $\mathbb{E}[g(X)] \leq g(\mathbb{E}[X])$.

Problem 5: Pseudoalignment

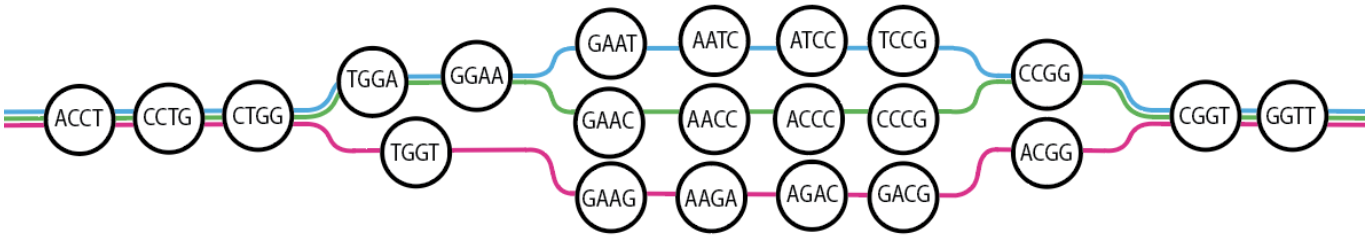
One important application of the sequence alignment algorithms we have been discussing in class is the alignment of reads from RNA sequencing (RNA-seq) experiments. In an RNA-seq experiment, mRNA transcripts from single cells (scRNA-seq) or a conglomeration of cells (bulk RNA-seq) are reverse-transcribed into DNA, which can be sequenced on a next generation sequencing machine. The sequence of each transcript, called a "read", can then be aligned to a reference genome. The number of reads aligning to each gene gives us an idea of the level at which each gene is being expressed.

RNA-seq experiments can generate millions of reads. As efficient as our dynamic programming alignment algorithms are, however, aligning all of these reads back to a reference genome usually requires a high-performance computing cluster (like OSCAR at Brown!). This is where "pseudoalignment" algorithms come in handy. Pseudoalignment algorithms don't actually align all reads to a reference genome, but rather identify which transcripts a read could have originated from. This approach results in a dramatic decrease in computational complexity, allowing researchers to perform RNA-seq quantification on a lowly laptop!

Please read the following blog post explaining how kallisto uses pseudoalignment. The post references a De Bruijn Graph, which is central to the kallisto algorithm. A De Bruijn Graph is a graph with k-mers as nodes. Nodes connected by an edge represent overlapping k-mers, offset by 1 base. A path through a De Bruijn Graph represents a sequence. For instance, the sequence "AGCTGCAA" can be represented by the full path through the following De Bruijn Graph, composed of 4-mers:



Now, consider the following transcriptome De Bruijn Graph (T-DBG):



a. Consider the read "CCTGGAACCCGGTT". Split it into 4-mers, and using the T-DBG above, list the k-compatibility class of the node corresponding to each 4-mer (ex. "ACCT" = {A,B,C}).

b. Which transcript could the read "CCTGGAACCCGGTT" have come from? How did your work in part (a) help you figure this out?

c. If the kallisto algorithm were run on the sequence "CCTGGAACCCGGTT" with the above T-DBG as the kallisto index, which 4-mers would have been hashed to a k-compatibility class?

d. In general, why doesn't the kallisto algorithm have to hash every k-mer of a given read to a k-compatibility class? Please use your own words in your answer.

If you are interested in learning more about pseudoalignment, check out the full kallisto paper!