COMP 462 / 561 – Computational Biology Methods
Assignment #1
Due date: 23:59, October 2, 2019, on MyCourses

Notes: You can either type your assignment (using Word, Latex, etc), or handwrite it and scan it. In either case, you must submit a PDF document. For question 3, also submit your code.

Question 1. (10 points) Needleman-Wunsch algorithm
What is the optimal global alignment for APPLE and HAPE? Show all optimal solutions and the corresponding paths under the match score of +1, mismatch score of -1, and indel penalty cost(L) = -1*L (linear gap penalty).

Question 2. (5 points) Affine gap penalty
Give an example of a pair of short sequences for which the optimal pairwise alignment under the linear gap penalty scheme described in question 1 is different from the optimal pairwise alignment under an affine gap penalty with cost(L) = -2 - 0.5*L. Try to find the shortest sequences possible, and show the optimal alignment under the two scoring schemes.

Question 3. (45 points) Sequence alignment with arbitrary gap penalty
When aligning two sequences, it is important to choose an alignment scoring scheme (substitution matrix cost and gap penalty function) that best captures the types of evolutionary events that are common in the type of sequences being aligned. For example, if the two sequences being aligned are protein-coding genes, then insertions or deletions of length 3, 6, 9, etc. nucleotides are common, whereas those whose length is not a multiple of three are rare, because they would result in a frameshift, i.e in the complete change of the amino acid encoded by the portion of the gene that follows the indel point. The Needleman-Wunsch dynamic programming algorithm can be modified to identify the optimal alignment under that kind of scoring scheme… but it's a bit too complicated for this assignment. Instead we will consider the following (somewhat artificial) alignment problem, that we call the *multi-gap-free* alignment problem.

We say that an alignment is *multi-gap-free* if it does not contain consecutive gaps in the *same* sequence. For example, the following alignments are multi-gap-free:

```
ACAG
A-T-

AC-A-G
A-T-G-
```

But the following alignments are not multi-gap-free:

```
ACAG
A--T

ACA--
A-ATG
```

a) (5 points) *Multi-gap-free* only exist when the lengths *m* and *n* of the two sequences satisfy a certain constraint. What is that constraint?

b) (20 points) Give the pseudocode of an exact algorithm for optimal pairwise global multi-gap-free alignment. Assume that a substitution cost matrix *M* and a gap penalty cost *b* are given as input, along with the two sequences *S* and *T* to be aligned. Your algorithm should run in time *O(m n)*, where *m* and *n* are the lengths of the two sequences.

Hint: Study the algorithm for optimal pairwise alignment that works under the affine gap penalty (see the scan of Durbin et al.'s textbook on MyCourses). This should provide ideas for how to solve this question.

c) (20 points) Implement your algorithm, including the trace-back procedure, in the programming language of your choice. Do not use any external alignment package/library; write your program from scratch. Your algorithm should take four arguments: (1) File containing the two sequences to be aligned (FASTA format). (2) The score for matches; (3) The score for mismatches (we will assume that all mismatches are score identically); (4) The value of gap penalty *b*. Your program should print out the optimal alignment score, and the optimal alignment itself.

Use your program to compute the optimal multi-gap-free alignment for the following pairs of sequences, assuming matchScore=1, mismatchScore=-1, b=-1:
http://www.cs.mcgill.ca/~blanchem/561/hw1_short.fa
http://www.cs.mcgill.ca/~blanchem/561/hw1_medium.fa
http://www.cs.mcgill.ca/~blanchem/561/hw1_long.fa

Report the alignments found, together with their score, and submit your code on MyCourses. Your code will not be marked, but submitting an answer (alignment score) that would not have been produced by your own code would be considered cheating.

Question 4. (5 points) Longest common subsequence problem
Suppose that you have a very fast machine to execute the Smith-Waterman algorithm with any user-specific substitution cost matrix and linear gap penalty (such special-purpose machines exist and were fashionable in the 90's). How would you use it to solve the longest common subsequence: Given two DNA sequences S and T, find the longest sequence X that is a subsequence of both S and T.

Note: Subsequences and substrings are two different things. A subsequence is made of characters that occur in the right order in the input spring, but not necessarily consecutively. A substring is a set of consecutive characters of the input string. So every substring is a subsequence, but not vice-versa. For example, BOICS is a subsequence of BIOINFORMATICS, but it is not a substring. IRO is neither a subsequence nor a substring of BIOINFORMATICS.

Question 5. (15 points) NoDeletion alignment problem
Consider the NoDeletion global alignment problem, which consists of optimally aligning sequences S and T under a linear gap penalty for insertions but where deletions from S to T are not allowed. Let m and n be the lengths of S and T respectively, and let k = n – m (of course, the problem only makes sense if m <= n). Give an algorithm to solve the problem in time O(m*k).

Question 6. (10 points) Progressive multiple sequence alignment
The progressive alignment algorithm we have seen in class to solve the multiple sequence alignment is not guaranteed to produce an optimal solution. Assume a sum-of-pairs scoring scheme with a linear gap penalty of b = -1 and the cost of mismatches is -1.

Give a simple example of specific set of short DNA strings where the algorithm fails to produce an optimal result. Give the alignment and score produced by the algorithm, as well as the optimal alignment and its score.

Question 7. (10 points) Blast algorithm
Give an example of the two most similar DNA sequences of length 20 that Blast using word length w=5 will fail to align.

BONUS QUESTION. (20 points) Linear space pairwise alignment
The standard definition of the Needleman-Wunsch algorithm requires O(m*n) space to align sequences of length m and n respectively. In many cases, this is prohibitive. Of course, one can compute the score of the optimal alignment in linear space trivially by only keeping in memory the last two rows of the dynamic programming table. Let's call this the ForgetThePast-NW algorithm. However, this prevents us from recovering the alignment that achieves that score, since trace-back is impossible.

Describe a O(m+n) space algorithm to compute both the score and the alignment itself. Hint: Think Divide-and-Conquer! Find out where the path corresponding to the optimal alignment will cross the row m/2, using two calls to a modified ForgetThePast-NW. Then, recurse…