CS 490/590 Bioinformatics Project II: Pairwise Optimal DNA Sequence Alignment

Description: You are to implement the dynamic programming based sequence alignment algorithm, with the following three variations: global alignment, local alignment, and global alignment with affine gap penalties. For all three variations, your algorithm must take $O(n^2)$ time to construct the matrix (or the three-level matrix in the case of alignment with affine gap penalties) and linear time to extract the optimal alignment from the matrix constructed. The alignment problems considered for this project do not involve substitution matrices. And you only need to output one optimal alignment in case of ties.

I/O and Parameter settings: Your sequences should be in FASTA format, namely with a discarded header line followed by lines of sequence. Assume that your first input sequence is in such a formatted file named **sequence1.txt** and your second input sequence is similarly in **sequence2.txt**. Upon starting the program, you should prompt the user to enter what kind of alignment he/she wants to perform: **global, local,** or **affine.** (Affine is WLOG assumed to be global-affine.) Then, in the cases of global or local choices you need to prompt the user to enter the **match score**, **mismatch score,** and **gap score**. In the case of affine alignment, you need to prompt the user to enter the **match score**, **mismatch score, gap start score,** and **gap extension score**. For the gap scores, whatever value X the user enters (check that X is an integer), you will use -|X| in the computation. You may also check that the mismatch score is less than the match score.

Regarding the output and its format, you should output: the ratio of matched letters to the alignment length, the value of OPT, and **one solution path** (not all of them). It is most useful to the user if you output the solution alignment with nice formatting that places one string above the other, respectively aligning the gaps, matches, and substitutions, continuing with both strings to the next pair of lines when reaching a set number of characters per line (e.g. 60). This formatting is not critical, however, and at a bare minimum is also expressible by giving the **H,V,D** sequence involved in the path. You do not need to output the matrix, although it is actually quite useful for your own error checking.

Algorithmically: Whenever there are multiple optimal solutions (corresponding to multiple winning paths from the origin to the upper-right corner of the matrix-graph), you only need to keep track of one. As stated previously, you do need to construct your matrix/matrices in **quadratic time**. This is especially important in the case of affine gap penalties, where you should use compute three matrices instead of doing a cubic amount of computation. Refer to you class notes and slides for local and affine cases.

What to turn in: You must turn in a single zipped file containing your source code, a Makefile if needed for compilation, and a README file indicating how to execute your program.

Your program must be written in C/C++ or Java and compile using an open source compiler such as g++, gcc, or javac.

I will demo a Perl implementation of this project in class/lab so you will understand I/O, etc..

This assignment is due by MIDNIGHT of Thursday, September 19. Late submissions carry a -33% per day penalty. The assignment is worth 12% of your total grade.