PROGRAMMING ASSIGNMENT #1

Cpt S 471/571, Spring 2019

Due: February 12, 2019, by 11:59pm Pacific Time @Blackboard (WSU) Assignments dropbox

---

General Guidelines:

- This is an *individual* programming assignment. Reproduction of source codes from online resources or other's people's assignments is *not* allowed. All source code should be entirely yours, written from scratch.
- You can use C or C++ or Java. For reasons of memory efficiency, C or C++ will be highly preferred over Java (or Python). C# should be okay too.
- Grading will be based on a combination of factors such as correctness, coding style, implementation efficiency, exception handling, source code documentation, and code reusability.
- When coding, please remember that you will be reusing the code for this assignment in future projects of this course. So make sure you provide a nice interface and modularized code structure during design.
- **Submission:** The assignment should be zipped into an archive folder (named after your name - e.g., Program1-XYZ.zip), and uploaded into the WSU Blackboard (learn.wsu.edu) dropbox for Project #1. Submissions are due by 11:59pm on the due date. A 24-hour grace period is allowed although it will incur a late penalty of 10%. Submissions that are more than 24 hours late will be returned without grading.

Assignment:

Implement two algorithms:
i) Needleman-Wunsch algorithm for computing OPTIMAL GLOBAL ALIGNMENT, and
ii) Smith-Waterman algorithm for computing OPTIMAL LOCAL ALIGNMENT,

both using ***affine gap penalty function*** between two input DNA sequences, s_1 and s_2, of lengths m and n respectively.

Each cell of your Dynamic Programming table ("DP table") should have the following structure:

```
struct DP_cell {
   int score;
   ... // add any other field(s) that you may need for the implementation
}
```

At the start of the program, you should read the alignment score parameters from a user-specified input file (optional). The default name of the file, if the user does not specify one, should be "***parameters.config***" in the present working directory. The parameters.config file should allow the user to specify one scoring parameter in each line (space or tab delimited). For example:

```
match    1
mismatch    -2
h  -5
g -1
```

The command prompt usage for your program should look as follows:

$ <executable name> <input file containing both s1 and s2> <0: global, 1: local> <optional: path to parameters config file>

Input File Formats:

The two input sequences should be given as input in one text file. The text file should be in what is called the "multi-sequence FASTA format", which is as follows:

- The format allows the file to contain any number of sequences, although in this program project you will have only two sequences as input.
- Each sequence will first start with a HEADER line, which has the sequence name in it. This header line will always starts with the ">" symbol and is immediately followed (without any whitespace character) by a word that will serve as the unique identifier (or name) for that sequence. Whatever follows the first whitespace character after the identifier is a don't care and can be ignored in your program.
- The header line is followed by the actual DNA sequence which is a string over the alphabet {a,c,g,t}. The sequence can span multiple lines and each line can variable number of characters (but no whitespaces or any other special characters).

For example:

An input file called "input.fasta" could look like:

>s1 sequence
acatgctacacgtactccgataccccgtaaccgataacgatacacagacct
cgtacgcttgctacaacgtactctataaccgagaacgattgaca
tgcctcgtacacatgctacacgtactccgatgaccccgt

>s2 sequence
acattctacgaacctctcgataaccccataaccgataacgattgacacctcgt
acgctttctacaacttactctctcgataaccccataaccgataacgattgacacctc
gtacacatggtacatacgtactctcgataccccgt

At completion, the program should output/print the following information:

- Parameter values from the parameters.config file: match score, mismatch penalty, gap penalties (h, g)
- Display the "names" or identifiers of the two sequences aligned
- The final optimal score
- For global alignment: display *any one* optimal alignment (using optimal path retrace) s.t. the alignment is shown wrapped up with each line containing at most 60 aligning positions.
- Along with each alignment display, report the corresponding numbers for matches, mismatches and gaps (insertions + deletions) and opening gaps.
- For local alignment: again, display any one optimal scoring local alignment.

All output should be to the standard output.

- Example Input/Output

As shown above, the alignment is wrapped after every 60 alignment positions, and on both sides the starting and ending indices of the aligning positions in the respective strings should be displayed. Then, a pipe symbol "|" is shown in columns wherever there is a match. (Other columns contain a whitespace character).

After implementing and testing, run your program on the following two input files, redirect the standard output into a global alignment and a local alignment output text files and attach them as part of your submission:

Opsin1_colorblindness_gene.fasta: Contains two sequences - Opsin1 gene in human vs. Opsin1 gene in mouse (this is one of the genes responsible for colorblindness)

Human-Mouse-BRCA2-cds.fasta: Contains two sequences - BRCA2 gene in human vs. BRCA2 gene in mouse (this is one of the breast cancer genes)

You are welcome to create your own additional test cases using small genes from the NCBI GenBank data (e.g., genes related to color blindness from human vs. mouse).

Additional references: NCBI GenBank

-

CHECKLIST FOR SUBMISSION:

___ source code

___ output files for the above two real world inputs

___ All the above zipped into an archive

-