CS123A Module 2 Week 6
Programming Assignment #1 (50pts)
(There are Two Parts. Each Part Is Worth 25pts. Complete Both Parts)

PROGRAMMING ASSIGNMENT #1:

This two part programming assignment provides practice with performing both local and global alignment. BLAST will be used for local alignment and the Needleman-Wunsch algorithm will be used for global alignment. There arfe many other local and global alignment algorithms.

Local Alignment Algorithms:

Smith-Waterman: Was first proposed by Temple F. Smith and Michael S. Waterman in 1981. Like the Needleman–Wunsch algorithm, of which it is a variation, Smith–Waterman is a dynamic programming algorithm. As such, it has the desirable property that it is guaranteed to find the optimal local alignment with respect to the scoring system being used (which includes the substitution matrix and the gap-scoring scheme).

Temple Smith at Yale in 2008

Michael Waterman in 2004

FASTA & FASTP:
FASTA is a DNA and FASTP is a protein sequence alignment software package first described by David J. Lipman and William R. Pearson in 1985. The current FASTA package contains programs for protein:protein, DNA:DNA, protein:translated DNA (with frameshifts), and ordered or unordered peptide searches. Recent versions of the FASTA package include special translated search algorithms that correctly handle frameshift errors (which six-frame-translated searches do not handle very well) when comparing nucleotide to protein sequence data.


David Lipman (right) with Paul Ginsparg in June 2013


William R. Pearson
Professor
wrp@virginia.edu
+1 (434) 924-2818

PSI-BLAST: Combination of Smith-Waterman and PSI-BLAST profile construction strategy.

SAM: A local and global alignment algorigthm with profile HMMMs.

and more ● ● ●

Global Alignment Algorithms:

Needleman-Wunsch:

The Needleman–Wunsch algorithm is an algorithm used in bioinformatics to align protein or nucleotide sequences. It was one of the first applications of dynamic programming to compare biological sequences. The algorithm was developed by Saul B. Needleman and Christian D. Wunsch and published in 1970.

Saul Ben Needleman (1928-2019)

Christian D. Wunsch (1939 -)

GGSEARCH & GLSEARCH: For proteins

CUDAlign: DNA sequence alignment of unrestricted size in single or multiple GPUs. Local, SemiGlobal, Global

and more ● ● ●

INSTRUCTIONS:

Part 1:

NON-CS MAJORS:

Instructions:

1. Use your favorite genome portal to locally align (using BLAST) the sequences in the files seq1.txt and seq2.txt located in Files -> Module 3 Alignment -> Week 6 -> Data folder. Report the score that you obtain. Assess the quality of the alignment, is it a good, OK, or poor score and why?

2. Although we have just started discussing and using the NCBI portal to perform BLAST-global alignment. You can do this at https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastSearch&PROG_DEF=blastn&BLAST_PROG_DEF=blastn&BLAST_SPEC=GlobalAln&LINK_LOC=BlastHomeLink

3. Globally align the sequences in the files seq1.txt and seq2.txt Report the score that you obtain. Assess the quality of the alignment, is it a good, OK, or poor score and why?

4. Which alignment, global or local, performed the best? Explain.

5. Use tools at the NCBI portal (you should now know or have an idea how to do this without being told which specific tool to use) to identify the type (i.e., name) of protein and organisms represented by the sequences in seq1.txt and seq2.txt

6. Develop a hands on exercise that provides a step-by-step set of instructions on how to perform the above local and global alignments of the provided sequences. Submit this work sheet with your answers along with your hands on exercise.

CS MAJORS:

This programming assignment involves introducing the Biopython package to perform pairwise sequence alignment to determine if two sequences from two different organisms are similar or related to each other.

About Biopython:

Biopython is a set of libraries to provide the ability to deal with ”things” of interest to biologists working on the computer. In general this means that you will need to have at least some programming experience (in python, of course!) or at least an interest in learning to program. Biopython's job is to make your job easier as a programmer by supplying reusable libraries so that you can focus on answering your specific question of interest.

One thing to note about Biopython is that it often provides multiple ways of doing the same thing. This can be frustrating since one often wants or needs to know just one correct and efficient way to do something. However, having multiple ways to accomplish the same task can be a real benefit because it gives you lots of flexibility and control over the libraries.

Instructions:

1. If Biopython is not already installed in Python 3.x on your computer, then in a shell window, execute the following command to install it.

   ```
   pip3 install biopython
   ```

2. Read instructions about performing pairwise local and global alignment using Biopython at
   http://biopython.org/DIST/docs/api/Bio.pairwise2-module.html

3. Create a Python module named <your initials>_pairwise_alignment.py that reads in the protein sequences in the files named seq1.txt and seq2.txt and both globally and locally aligns them using the BLOSUM62 matrix. Make sure to use the "`format_alignment`" function in the Biopython package to print out the alignment results.

4. Which alignment, global or local, performed the best?

5. Use tools at the NCBI portal (you should now know or have an idea how to do this without being told which specific tool to use) to identify the type (i.e., name) of protein and organisms represented by the sequences in seq1.txt and seq2.txt

6. Submit your code and this work sheet. NOTE: do not use absolute paths in your code. Your submission should work “relative” to any Python working directory that your code is placed and run.

Part 2:

NON-CS MAJORS:

Do the same as in Programming Assignment #1, except click on the "Algorithm parameters" link and find a Gap Cost that improves, if at all, the score you achieved in Part 1 of Programming Assignment #1. Report a gap cost that you found to give you the best score. Submit this work sheet and indicate the best alignment that you find.

CS MAJORS:

Do the same as in Part 1 of Programming Assignment #1, except experiment with setting the `penalize_extend_when_opening` and `penalize_end_gaps` alignment parameters to see if you can get a better global alignment score than you achieved in Part 1 of Programming Assignment #1.

Submit your code and this work sheet. NOTE: do not use absolute paths in your code. If you use absolute paths, your code will be deemed not working. Your submission should work "relative" to any Python working directory that your code is placed and run.