**Dynamic Programming - Affine Gap Alignment**

There is a nice tutorial on dynamic programming at <http://www.sbc.su.se/~per/molbioinfo2001/seqali-dyn.html> with pointers to an example at <http://www.sbc.su.se/~per/molbioinfo2001/dynprog/dynamic.html>.

If you want a textbook presentation, one of the best is in Chapter 2 of Durbin, Eddy, Krogh, and Mitchison's book *Biological Sequence Analysis*. Chapter 5 in Compeau and Pevzner provides a nice handling of the material as well..

Use the recurrence relationships presented in class. The are provided in the lecture notes.

Be careful of the boundary conditions. For global alignment, you should have "start"-"start" aligned with 0 cost in the MATCH state and neg. infinity in Ix and Iy, so that a gap opening penalty is incurred if you start with an insertion or deletion. Before start, all rows and columns are negative infinity, and the rest of the start row and column can be inferred from the recurrence relation (see lecture notes). Make sure that your initial conditions are consistent with the recurrence relations people sometimes get the indices of their matrices reversed, or confuse the Ix and Iy matrices.

For local alignment, the recurrence relations and the boundary conditions are different.  Again, use the recurrences from Durbin and Eddy, or those presented in class.

Name your program "testAlign", so that we can test everyone's program with a script that executes:

* testAlign.py --align=local < test.fasta
* testAlign.py --align=global < test.fasta .

Turn in both your program and the output from running the program on the sequence pairs given in the test files (eCommons::resources) . You'll need a substitution matrix for this assignment, BLOSUM50 is provided in eCommons::Resources.

**Q1 (hand computation):**

Compute the score for the following global alignment using the BLOSUM50 scoring matrix with a **gap-opening cost of 12 and a gap-extension cost of 1** (all are in 1/2 bit units to be compatible with the BLOSUM50 matrix). Show what you added up to get the score, since the final number alone could easily be incorrect due to a transcription or addition error.

---CTNIDDSADKN-R--  
VLDAME-EIGEGDDIKMV

Note: this is neither the global nor the local optimal alignment.

**Q2 (align module):**

Create a module (align.py) that contains two classes: LocalAligner and GlobalAligner. Both classes should have the same methods:

aligner= LocalAligner(subst, gaps, xseq, yseq)  
aligner= GlobalAligner(subst, gaps, xseq, yseq)

subst and gaps are parameters for the substitution matrix and gap costs for the alignments to use; xseq and yseq are both sequences in fasta format, that may or may not have been previously aligned

**aligner.score ()**

This method assumes that xseq and yseq are to be scored as is ( in their current alignment).

Use this method to check that the alignments you output score what the dynamic programming algorithm provides.

**aligner.align()**

The implementation of this method is private. It should remove any non sequence characters ( gap symbols, spaces, hyphens, underscores or dots). The method should return the best alignment score.

It must be able to subsequently provide a traceback using that method. Consider using a list of lists, a mesh ( 2D or 3D linked structure), or a vector field.

**aligner.traceback()**

Returns a list object that contains the aligned xseq and yseq sequences.

**testAlign.py**

Write a program testAlign.py where you use your Aligner modules to do a master-slave alignment program.

**Required Options:**

* --substMatrix BLOSUM50  (Specifies the file to read the substitution matrix from (integer values in BLOSUM format))
* --align local or --align global (Specify either local or global alignment.)
* --open 10  (Gap opening penalty)
* --extend 2  (Gap extension penalty.)

The program should read a fasta file from stdin. The fasta files may have alignment characters in them, like spaces, periods, and hyphens. The alphabet for the fasta input is taken to be the characters used as indices in the substitution matrix.

The first sequence of the fasta file is taken to be the "master" sequence. All subsequent sequences are aligned to that first sequence and then both are output to stdout as a pair. The score for each alignment should be output to stderr.

The gap penalties should (by default) be open=12, extend=1.

* [python5-1.fasta](http://classes.soe.ucsc.edu/bme205/Fall10/python5-1.fasta) This example is taken from Durbin and Eddy, and there are 2 local alignments with the same optimal score.
* [python5-2.fasta](http://classes.soe.ucsc.edu/bme205/Fall10/python5-2.fasta)
* [python5-3.fasta](http://classes.soe.ucsc.edu/bme205/Fall10/python5-3.fasta)
* [python5-4.fasta](http://classes.soe.ucsc.edu/bme205/Fall10/python5-4.fasta)
* [python5-5.fasta](http://classes.soe.ucsc.edu/bme205/Fall10/python5-5.fasta)
* [python5-6.fasta](http://classes.soe.ucsc.edu/bme205/Fall10/python5-6.fasta)

To help you with your debugging, I provide possible output for local alignment of python5-3, with g=12, e=1:

>fat

GARFIELD-------ISAFATCAT

>cat

GARFIELDWASTHELASTFATCAT

>fat

GARFIELDISAFATCAT

>cat2

GARFIELDISAFATCAT

>fat

GARFIELDISA--FATCAT

>cat3

GARFIELDISTHEFATCAT

>fat

GARFIELDISAFA-TCAT

>cat4

GARFIELDISAFASTCAT

>fat

GARFIELDISA--FA-TCAT

>cat4

GARFIELDISTHEFASTCAT

>fat

GARFIELDISAFAT

>cat5

GARFIELDIS-FAT

>fat

SAYGA

>say

SAYFA

with scores 79, 108, 90, 96, 78, 68 and 19.

Possible global alignments are

>fat

ISAYGARFIELD-------ISAFATCAT

>cat

----GARFIELDWASTHELASTFATCAT

>fat

ISAYGARFIELDISAFATCAT

>cat2

----GARFIELDISAFATCAT

>fat

ISAYGARFIELDISA--FATCAT

>cat3

----GARFIELDISTHEFATCAT

>fat

ISAYGARFIELDISAFA-TCAT

>cat4

----GARFIELDISAFASTCAT

>fat

ISAYGARFIELDISA--FA-TCAT

>cat4

----GARFIELDISTHEFASTCAT

>fat

ISAYGARFIELDISAFATCAT

>cat5

----GARFIELDIS----FAT

>fat

ISAYGARFIELDISAFATCAT

>say

-SAY--------------FAT

with scores 64, 93, 75, 81, 63, 40, and -11.

**Bonus Points**

* linear gap costs

Implement linear gap costs and compare running times for affine and linear algorithms. You may need to use longer test sequences to get repeatable timing results, as the expected 3-fold difference in the inner loop may be buried in overhead.​