CS612 – Algorithms in Bioinformatics

Homework Assignment 1
Due Fri. Feb. 21 before midnight, on Gradescope

Objectives

- Learn how to perform sequence alignment and analyze the results
- Understand and apply the theory of sequence alignment
- Practice programming BLAST with Biopython.

The FASTA format

FASTA (pronounced "fast-ay") is a text-based format for representing either nucleotide sequences or peptide sequences, in a single letter code. It is very simple. The top line starts with a $>$ followed by free text. The rest are the nucleotides or amino acid letter codes.

Part 1 – Practice

1. **Sequence alignment hands-on exercise.** You are given the following protein sequence:
   $>$protein
   `TCPFADPAALYSRQDTTSGQSPLAAYEVDDSTGYLTSDVGGPIQDQTSLKAGIRGPTLLEDFMFRQKIQHFDHERVPERAV`

   The sequence is also available at the course webpage: http://www.cs.umb.edu/ nurith/cs612/protein.fasta .

   (a) Go to the Blast website at http://blast.ncbi.nlm.nih.gov. Select "Protein Blast" to get to the BlastP website. Paste the above sequence to the top window. Use the default parameters – nr database (non-redundant protein sequence database). Hit "BLAST" – the search may take up to a minute.

   Answer the following questions about the best hit:

   i. What is the name of the sequence?
   uncharacterized protein VTJ04DRAFT_724 [Mycothermus thermophilus]

   ii. what is the identifier (Accession) on the far right?
   XP_069223333.1 (this is an NCBI accession number)

   iii. what is the alignment score ("max score")?
   171. The raw score is 433.

   iv. what is the percent identity and query coverage?
   both 100%.

   v. what is the E-value? 2e-47

   vi. are there any gaps in the alignment? No. This is basically the same sequence.

(b) Repeat the search above with the UniprotKB/SwissProt database and answer the same questions as above.

i. What is the name of the sequence? RecName: Full=Catalase-3; Flags: Precursor [Neurospora crassa OR74A]

ii. what is the identifier (Accession) on the far right?
Q9C169.1 (this is a Uniprot accession number)

iii. what is the alignment score ("max score")?
100. The raw score is 250.

iv. what is the percent identity and query coverage?
both 100% identity and %100 query coverage.

v. what is the E-value? 2e-25

vi. are there any gaps in the alignment? There are two gaps.

(c) Repeat search with nr as the database and PAM30 as a substitution matrix. This can be done in the blastP homepage by opening "algorithm parameters" at the bottom of the page. Observe the changes between the results of a and c due to the change in the substitution matrix: Look at the first entry that differs between a and c. What is its rank in a and c? What is the name of the protein sequence in this entry?

The two searches are similar, but not identical. The protein from (a) is in 8th place here, but it's the same basically, only with a structure.

i. What is the name of the sequence? Chain A, Catalase [Mycothermus thermophilus]

ii. what is the identifier (Accession) on the far right?
7VN0_A (this is a PDB accession number)

iii. what is the alignment score ("max score")?
175. The raw score is 407.

iv. what is the percent identity and query coverage?
both 100% identity and %99 query coverage.

v. what is the E-value? 6e-45

vi. are there any gaps in the alignment? There are no gaps.

2. **DNA sequence alignment:** The following sequence was constructed by NCBI scientist Mark Boguski for Michael Chrichton's "The Lost World" of the Jurassic Park series:
>DinoDNA from THE LOST WORLD p. 135
GAATTCCGGAAGCGAGCAAGAGATAAGTCCTGGCATCAGATACAGTTGGAGATAAGGACG
GACGTGTGGCAGCTCCCGCAGAGGATTCACTGGAAGTGCATTACCTATCCCATGGGAGCC
ATGGAGTTCGTGGCGCTGGGGGGGCCGGATGCGGGCTCCCCCACTCCGTTCCCTGATGAA
GCCGGAGCCTTCCTGGGGCTGGGGGGGGGCGAGAGGACGGAGGCGGGGGGGCTGCTGGCC
TCCTACCCCCCCTCAGGCCGCGTGTCCCTGGTGCCGTGGGCAGACACGGGTACTTTGGGG
ACCCCCCAGTGGGTGCCGCCCGCCACCCAAATGGAGCCCCCCCACTACCTGGAGCTGCTG
CAACCCCCCCGGGGCAGCCCCCCCCATCCCTCCTCCGGGCCCCTACTGCCACTCAGCAGC
GGGCCCCCACCCTGCGAGGCCCGTGAGTGCGTCATGGCCAGGAAGAACTGCGGAGCGACG
GCAACGCCGCTGTGGCGCCGGGACGGCACCGGGCATTACCTGTGCAACTGGGCCTCAGCC
TGCGGGCTCTACCACCGCCTCAACGGCCAGAACCGCCCGCTCATCCGCCCCAAAAAGCGC
CTGCTGGTGAGTAAGCGCGCAGGCACAGTGTGCAGCCACGAGCGTGAAAACTGCCAGACA
TCCACCACCACTCTGTGGCGTCGCAGCCCCATGGGGGACCCCGTCTGCAACAACATTCAC
GCCTGCGGCCTCTACTACAAACTGCACCAAGTGAACCGCCCCCTCACGATGCGCAAAGAC
GGAATCCAAACCCGAAACCGCAAAGTTTCCTCCAAGGGTAAAAAGCGGCGCCCCCCGGGG
GGGGGAAACCCCTCCGCCACCGCGGGAGGGGGCGCTCCTATGGGGGGAGGGGGGGACCCC

```
TCTATGCCCCCCCCGCCGCCCCCCCCGGCCGCCGCCCCCCCTCAAAGCGACGCTCTGTAC
GCTCTCGGCCCCGTGGTCCTTTCGGGCCATTTTCTGCCCTTTGGAAACTCCGGAGGGTTT
TTTGGGGGGGGGGCGGGGGGTTACACGGCCCCCCCGGGGCTGAGCCCGCAGATTTAAATA
ATAACTCTGACGTGGGCAAGTGGGCCTTGCTGAGAAGACAGTGTAACATAATAATTTGCA
CCTCGGCAATTGCAGAGGGTCGATCTCCACTTTGGACACAACAGGGCTACTCGGTAGGAC
CAGATAAGCACTTTGCTCCCTGGACTGAAAAAGAAAGGATTTATCTGTTTGCTTCTTGCT
GACAAATCCCTGTGAAAGGTAAAAGTCGGACACAGCAATCGATTATTTCTCGCCTGTGTG
AAATTACTGTGAATATTGTAAATATATATATATATATATATATATCTGTATAGAACAGCC
TCGGAGGCGGCATGGACCCAGCGTAGATCATGCTGGATTTGTACTGCCGGAATTC
```

The sequence is also available at the course webpage: http://www.cs.umb.edu/ nurith/cs612/dino.fasta

Perform a Blast search using blastn (nucleotide search) and the default non-redundant (nt) nucleotide database.

(a) What are the two main species used to construct the dinosaur DNA sequence?

Frog and chicken

(b) Repeat the search with blastx (DNA vs. protein sequence) using the default non-redundant protein sequence database. Look at the top sequence alignment and retrieve the hidden message there (hint: look at the gaps...).

MARK WAS HERE NIH
(Mark Boguski was obviously playing a little practical joke on the way).

Part 2 – Theory

1. Lesk book question 5.3: The edit distance between the strings `agtcc` and `cgctca` is 3, consistent with the following alignment:

```
ag-tcc
cgctca
```

Find the sequence of three edit operations that convert `agtcc` to `cgctca`.

- Substitution $a \rightarrow c$ at first position
- Insertion (c after g)
- Substitution $c \rightarrow a$ at last position

2. **Dynamic programming:**

(a) Use the Needleman Wunsch global alignment Dynamic programming formula in slide set no. 2 to find the sequence alignment score of the two DNA sequences `ATCGAACTGCC` and `TACGCACTCCA`. Show the filled dynamic programming matrix using +1 for a match, -1 for a mismatch and -1 for a gap penalty in a way similar to the slide sets. Additionally, show the alignment itself. Show the filled dynamic programming matrix using +1 for a match, -1 for a mismatch and -1 for a gap penalty in a way similar to the slide sets. Additionally, show the alignment itself.

	y_j	A	T	C	G	A	A	C	T	G	C	C
x_i	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
T	-1	↖ -1	↖ 0	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8	← -9
A	-2	↖ 0	← -1	↖ -1	← -2	↖ -1	← -2	← -3	← -4	← -5	← -6	← -7
C	-3	↑ -1	↖ -1	↖ 0	← -1	← -2	↖ -2	↖ -1	← -2	← -3	↖ -4	↖ -5
G	-4	↑ -2	↖ -2	↑ -1	↖ 1	← 0	← -1	← -2	↖ -2	↖ -1	← -2	← -3
C	-5	↑ -3	↑ -3	↖ -1	↑ 0	↖ 0	↖ -1	↖ 0	← -1	← -2	↖ 0	↖ -1
A	-6	↖ -4	↖ -4	↑ -2	↑ -1	↖ 1	↖ 1	← 0	↖ -1	↖ -2	↑ -1	↖ -1
C	-7	↑ -5	↑ -5	↑ -3	↑ -2	↑ 0	↖ 0	↖ 2	← 1	← 0	↖ -1	↖ 0
T	-8	↑ -6	↑ -6	↑ -4	↑ -3	↑ -1	↑ -1	↑ 1	↖ 3	← 2	← 1	← 0
C	-9	↑ -7	↑ -7	↑ -5	↑ -4	↑ -2	↑ -2	↑ 0	↑ 2	↖ 2	↖ 3	↖ 2
C	-10	↑ -8	↑ -8	↖ -6	↑ -5	↑ -3	↑ -3	↖ -1	↑ 1	↑ 1	↖ 3	↖ 4
A	-11	↑ -9	↑ -9	↑ -7	↑ -6	↑ -4	↖ -2	↑ -2	↑ 0	↖ 0	↑ 2	↑ 3

```
-ATCGAACTGCC-
-|-||.|||-||-
TA-CGCACT-CCA
```

(b) Repeat with the Smith-Waterman local alignment algorithm and the same scoring scheme.

See below. Notice that there can be several solutions.

	y_j	A	T	C	G	A	A	C	T	G	C	C
x_i	0	0	0	0	0	0	0	0	0	0	0	0
T	0	↖ 0	↖ 1	← 0	← 0	← 0	← 0	← 0	↖ 1	← 0	← 0	← 0
A	0	↖ 1	← 0	← 0	← 0	↖ 1	↖ 1	← 0	← 0	← 0	← 0	← 0
C	0	↑ 0	↖ 0	↖ 1	← 0	← 0	↖ 0	↖ 2	← 1	← 0	↖ 1	↖ 1
G	0	← 0	← 0	↑ 0	↖ 2	← 1	← 0	↑ 1	↖ 1	↖ 2	← 1	← 0
C	0	← 0	← 0	↖ 1	↑ 1	↖ 1	← 0	↖ 1	← 0	↖ 1	↖ 3	↖ 2
A	0	↖ 1	← 0	↑ 0	↑ 0	↖ 2	↖ 2	← 1	← 0	↑ 0	↑ 2	↖ 2
C	0	↑ 0	← 0	↖ 1	← 0	↑ 1	↑ 1	↖ 3	← 2	← 1	↖ 1	↖ 3
T	0	← 0	↖ 1	← 0	↖ 0	← 0	← 0	↑ 2	↖ 4	← 3	← 2	↑ 2
C	0	← 0	↑ 0	↖ 2	← 1	← 0	← 0	↖ 1	↑ 3	↖ 3	↖ 4	↖ 3
C	0	← 0	← 0	↖ 1	↖ 1	← 0	← 0	↖ 1	↑ 2	↑ 2	↖ 4	↖ 5
A	0	↖ 1	← 0	↑ 0	↑ 0	↖ 2	↖ 1	← 0	↑ 1	↑ 1	↑ 3	↑ 4

```
CGAACTGCC
||.|||-||
CGCACT-CC
```

(c) The semi-global alignment is a variant of global alignment that does not penalize for gaps at the beginning and/or the end of one of the sequences but penalizes in the middle. In other words, the

conditions are the same as in global alignment:

$$S_{i,j} = \max \begin{cases} S_{i-1,j-1} + s(x_i, y_j) \\ S_{i-1,j} + g \\ S_{i,j-1} + g \end{cases}$$

but the boundary conditions are different: $S_{i,0} = S_{0,j} = 0$. Calculate the semi-global alignment of the two sequences.

	y_j	A	T	C	G	A	A	C	T	G	C	C
x_i	0	0	0	0	0	0	0	0	0	0	0	0
T	**0**	↑ -1	↖ 1	← 0	← -1	↑ -1	↑ -1	↑ -1	↖ 1	← 0	↑ -1	↑ -1
A	0	↖ **1**	← 0	↖ 0	↖ -1	↖ 0	↖ 0	← -1	↑ 0	↖ 0	← -1	← -2
C	0	↑ 0	↖ 0	↖ **1**	← 0	↑ -1	↑ -1	↖ 1	← 0	↑ -1	↖ 1	↖ 0
G	0	↑ -1	↑ -1	↑ 0	↖ **2**	← 1	← 0	↑ 0	↖ 0	↖ 1	← 0	↖ 0
C	0	↖ -1	← -2	↖ 0	↑ 1	↖ **1**	← 0	↖ 1	← 0	↑ 0	↖ 2	↖ 1
A	0	↖ 1	← 0	← -1	↑ 0	↖ 2	↖ **2**	← 1	← 0	↑ -1	↑ 1	↖ 1
C	0	↑ 0	↖ 0	↖ 1	← 0	↑ 1	↑ 1	↖ **3**	← 2	← 1	↖ 0	↖ 2
T	0	↑ -1	↖ 1	← 0	↖ 0	↑ 0	↑ 0	↑ 2	↖ **4**	← **3**	← 2	← 1
C	0	↖ -1	↑ 0	↖ 2	← 1	← 0	← -1	↖ 1	↑ 3	↖ 3	↖ **4**	↖ 3
C	0	↖ -1	↑ -1	↖ 1	↖ 1	← 0	← -1	↖ 0	↑ 2	↑ 2	↖ 4	↖ **5**
A	0	↖ 1	← 0	↑ 0	↑ 0	↖ 2	↖ 1	← 0	↑ 1	↑ 1	↑ 3	↑ **4**

```
-ATCGAACTGCC-
-|-|||.|||-||-
TA-CGCACT-CCA
```

3. The longest common subsequence (LCS) is the ungapped result of the global alignment. In other words – given two sequences, X and Y, Z it is the longest subsequence of X and Y if Z is a subsequence of X and Y and it is the longest of all subsequences. For example - the LCS of ABLE and BLUE is BLE (these letters appear in both sequences, in the same relative order). The Shortest common supersequence (SCS) of two sequences X and Y is the shortest sequence which has X and Y as subsequences. For example – The shortest common supersequence of ABLE and BLUE is ABLUE.

   (a) Describe how to obtain the SCS of two sequences, X and Y, given their LCS.

   First of all, find the LCS. Then fill every gap with the letter of the non-gapped sequence and add every mismatching character on both sides.

   (b) What is the SCS of `TACGGGTAT` and `GGACGTACG`?

   Here is the dynamic programming matrix of the two sequences:

	y_j	G	G	A	C	G	T	A	C	G
x_i	0	←-1	-2	-3	-4	-5	-6	-7	-8	-9
T	-1	↖-1	↖-2	↖-3	↖-4	↖-5	↖-4	←-5	←-6	←-7
A	-2	↖-2	↖-2	↖-1	←-2	←-3	←-4	↖-3	←-4	←-5
C	-3	↖-3	↖-3	↑-2	↖0	←-1	←-2	←-3	↖-2	←-3
G	-4	↖-2	↖-2	←-3	↑-1	↖1	←0	←-1	←-2	↖-1
G	-5	↖-3	↖-1	←-2	↑-2	↖0	↖0	←-1	←-2	↖-1
G	-6	↖-4	↖-2	↖-2	↑-3	↖-1	↖-1	↖-1	↖-2	↖-1
T	-7	↑-5	↑-3	↑-3	↖-3	↑-2	↖0	←-1	←-2	↑-2
A	-8	↑-6	↑-4	↖-2	←-3	↑-3	↑-1	↖1	←0	←-1
T	-9	↑-7	↑-5	↑-3	↖-3	↑-4	↖-2	↑0	↖0	↖-1

The LCS is (mismatches and gaps are in bold)
−TAC**GG**GTA**−T**
GGAC**−−**GTA**CG**

Hence, the SSC is GGTACGGGTATCG.

4. **Substitution matrices:**

   (a) Given the BLOSUM-62 matrix (see sequence class notes or http://www.ncbi.nlm.nih.gov/IEB/ToolBox/C_DOC/lxr/source/data/BLOSUM62), find the score of the following alignment (assume this is the optimal alignment):
   THISSEQ
   THATSEQ
   5+8-1+1+4+5+5=27

   (b) Repeat with the PAM-250 matrix. It can be found in Lesk, page 257, or here: http://www.ncbi.nlm.nih.gov/IEB/ToolBox/C_DOC/lxr/source/data/PAM250
   3+6-1+1+2+4+4=19

5. **Multiple Sequence Alignment:** Extend the dynamic programming formula to 3 dimensions. What is the run time in this case? How many cases do we have to compare this time?

   Hint: this time the matrix is cubic since instead of a 2-dimensional matrix we need to run on a cube of $m \times n \times k$ where m,n, and k are the lengths of the three sequences. Every path goes from one vertex of the cube and traveling inside the cube to another vertex. Try to count how many such paths there can be.

   The run time is cubic, $O(m * n * k)$ because you have to fill out a 3-D matrix instead of a 2-D, going over all the possible positions. However, what makes the problem difficult is not only the increased number of sequences, but the number of neighbors we have to compare for every step. This time we have 7 instead of 3 (think of the neighbors that precede a given position). In other words, the number of neighbors increases exponentially with the dimension, which hints to why the problem becomes NP-hard in higher dimensions.

Part 3 – Programming

For this part you need the biopython module installed (https://biopython.org/docs/1.75/api/index.html) and in particular - BioBlast. In particular – look at the NCBIWWW module.

Here is an extensive tutorial: https://biopython.org/docs/dev/Tutorial/index.html Write a python program that implements questions 1 and 2 using BioBlast. Using NCBIWWW it should not be difficult. Use the web run and do not try to install and run standalone BLAST! It will take a lot of space to install the databases locally. Try to have your program obtain the results in a human-readable format. By default the results are in xml, but you can change the format or use the parse tools from the NCBIXML library. **Notice:** The BLAST run may take a while (30–60 seconds or more). Attach the code as part of your answer.