Laboratory for Bioinformatics Tools
Exercise 2: Pairwise sequence alignment and BLAST

Given: March 7, Due: March 21

General guidelines:

- Submit a single zip file with your code and a doc / pdf file with answers to the following questions. The name format of the submitted file should be "123456789_ex2.zip" – replace "123456789" with your ID number.
- Include only a single code file named "ex2.py" in the submission zip. This file should include all the code you wrote for this exercise.

Python guidelines:

- Test your code using python 3.8 or 3.9.
- Make sure to document and explain your code wherever needed.
- Code should be as efficient as possible.
- Code should be readable and well organized.
- It is advised to break code into functions wherever possible.

Utilizing alignments for discovering evolutionary events

In the next few sections we will implement and use the local pairwise sequence alignment algorithm for finding evidence for evolutionary events.

a. Write a function "local_pwalignment". Input: Two sequences. Output: the score and alignment of the optimal local alignment of the two sequences following the dynamic programming local alignment algorithm we learned in class. Use the following scoring regime:

$$\sigma(x,y) = \begin{cases} +3 & x = y \\ -2 & x = "-" \oplus y = "-" \\ -\infty & x = y = "-" \\ -3 & else \end{cases}$$

where $\oplus$ is the XOR function. If there is more than one optimal alignment then arbitrarily return one of them.

Execution example:
> local_pwalignment("GGTTGACTA","TGTTACGG")
(13.0, 'GTTGAC', 'GTT-AC')

b. Briefly describe the FOXP1 gene and its functions (100 words max). Make sure to add references to the sources you use.

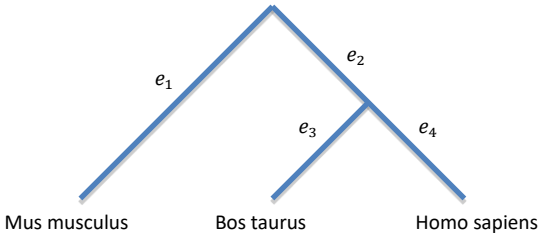
c. Download the RefSeq records (the sequences; in FASTA format) of the protein sequence of FOXP1 in the following three organisms, and attach the files to your submission:
   - Homo sapiens; Accession number: NP_001336267.1 ;name the file "Foxp1_Homo_sapiens.fasta"
   - Bos taurus; Accession number: NP_001077158.1; name the file "Foxp1_Bos_taurus.fasta"
   - Mus musculus; Accession number: NP_444432.1; name the file "Foxp1_Mus_musculus.fasta"

   If there are several isoforms of FOXP1 in some organism take the first one (e.g., if there are isoforms 1, 2 and 3 then take isoform 1, and if there are isoforms a, b and c then take isoform a). What are the RefSeq identifiers of the sequences you downloaded?

d. Run "local_pwalignment" on each of the three possible pairs of sequences. Report the scores of the alignments and save the alignments into three separate files as follows:
   - Homo sapiens vs. Bos Taurus ("Foxp1_Homo_sapiens_vs_Bos taurus.fasta")
   - Homo sapiens vs. Mus musculus ("Foxp1_Homo_sapiens_vs_Mus_musculus.fasta")
   - Mus musculus vs. Bos Taurus ("Foxp1_Mus_musculus_vs_Bos taurus.fasta")

   What is the largest insertion / deletion mutation you observe in each alignment?

   Consider the following evolutionary tree, which reflects the known evolutionary relations between the three organisms:



   For the purpose of this question, we define a substantial insertion / deletion event to be an insertion / deletion of at least five (consecutive) amino acids. Consider the substantial mutations you found in your alignments. Given that mutations are rare and irreversible, and based on the above tree, in which edges substantial mutations are most likely to have occurred in the evolution of FOXP1? Based on the topology of the tree, what is the most likely mutation type (insertion or deletion)? Why?

Investigating short tandem repeats in bacteria

In the following few sections we shall look at Short Tandem Repeats (STRs) in bacteria. STR is a repetitive group of DNA molecules (e.g., repetitions of AT, such as ATATATAT). Such repetitive

groups are relatively abundant in the human genome and have been shown to play a role in human variation. Particularly, STRs were found to be associated with several diseases such as cancer. Here, we are interested in looking for evidence that STRs are abundant in bacteria as well. Such a finding can suggest that STRs may play an important role in bacteria as well, and may serve as evidence that STRs are highly conserved across the evolution.

We are given a sequence that was found in bacterial specimen (in the file "query.txt"), however, we do not know what organism it came from. We will use BLAST to detect the organism and will investigate STRs in the genome of this organism.

e. Consider the case in which we are interested in using the BLAST algorithm for finding homologues of a long query. Assume that we are only interested in finding sequences with large homologous domains having high identity with the query, and further assume we need to run both the preprocessing step and the query step. Which parameter of the BLAST preprocessing step (that we mentioned in class) would you change in order to improve the query execution time of BLAST while not increasing the preprocessing execution time (and while still keeping the top results)? How would you change the value of this parameter and why? Elaborate.
f. Run NCBI's blastn (i.e. BLAST against nucleotides database) on the sequence provided in "query.txt". In our case, the default parameters of blastn are fine.

   Save a screenshot of the results page (the top ten results, as described under the "Descriptions" tab; name the file "blast.jpg" / "blast.png"). What is the top hit you got? Which organism it belongs to?
   Download the GeneBank full genome sequence of this organism in FASTA format and attach it to your submission. Name the file "genome.fasta".
g. Write a function "find_strs". Input: sequence S, STR s and a parameter r. Output: the number of non-overlapping STRs of type s that were found in the sequence S, such that each STR had at least r repetitions each time. Use Python's regular expression for extracting STRs in your implementation.

   Execution example:
   > find_strs("AAGAGAGTTAGAGTCAGC","AG",2)
   2 # we have AGAGAG and AGAG
   > find_strs("AAGAGAGTTAGAGTCAGC","AG",3)
   1 # we have AGAGAG
h. Write a function "find_strs3". Input: sequence S and a parameter r. Output: the total number of 3-length STRs that were found in the sequence S, such that each STR had at least r repetitions each time. Consider all 64 possible 3-length STRs, except for the 3-length homogeneous STRs (i.e., AAA, TTT, GGG and CCC). Use "find_strs" in your implementation.

Execution example:

> find_strs3 (“AAAGGAGGTGTTCGGTCGTCGTC”,2)

4 # we have AGGAGG, GTCGTCGTC, TCGTCG and CGTCGT

> find_strs3 (“AAAGGAGGTGTTCGGTCGTCGTC”,3)

1 # we have GTCGTCGTC

Run “find_strs3” on the sequence in “genome.fasta” with $r = 3$ and report the total number of STRs you got.

i. In this section we wish to define and implement a procedure for determining whether the number of STRs in “genome. fasta” is significant or not. In other words, we want to know if the large number of STRs we found in the genome is due to true biological phenomenon or is merely due to the composition of the sequence. Since we do not know the generative model of DNA sequences, we will devise a permutation test.

Denote our sequence by S and the total number of its 3-length STRs by $f_3^r(S)$ (which is the result of running “find_strs3” with S and r), we define the following permutation procedure:

1) Generate a random permutation (shuffle) π for S, such that $\pi(S)$ is S after permuting it according to π.
2) Find $f_3^r(\pi(S))$
3) Repeat steps (1) and (2) 100 times
4) Determine that $f_3^r(S)$ is statistically significant if

$$\left(\frac{\sum_{i=1}^{100} I\{f_3^r(S) < f_3^r(\pi^i(S))\}}{100}\right) < 0.05$$

where

$$I\{X > Y\} = \begin{cases} 1 & x > y \\ 0 & x \leq y \end{cases}$$

and π^i is the random permutation sampled in the i^{th} iteration of the procedure.

Write a function “permutation_test”. Input: sequence S and a parameter r. Output: True / False, indicating if the number of 3-length STRs in S with at least r repetitions each is statistically significant according to the above procedure. Run “permutation_test” on the sequence in “genome.fasta” with $r = 3$ and report whether you found the number of 3-length STRs to be statistically significant.