BIOI 3500, *Advanced Bioinformatics Programming*
Final Project

Final Project :: 150 points (total) :: **Due: 11:59 p.m. Friday, May 7th 2021**

Objectives

The final project builds on the skills and techniques developed during the semester. You are encouraged to reuse some of the code you wrote for other assignments (especially Assignment 11).

Description

Write a procedural Python (version 3) program, **`netID_FP.py`** where **`netID`** is your UNO NetID, that reads in a FASTA file containing one or more sequences (either DNA or protein sequences), performs a BLAST search (either locally or remotely using the NCBI web server via Biopython) for each sequence, and prints out the results of the search.

Command-line Options

Use the Python **`getopt`** module to process command-line arguments. Use the following option letters:

Option	Function
`i`	Specifies the name of the file containing the input sequence(s).
`o`	Specifies the name of the output file. If this option is not used, output should be sent to standard output (i.e., the screen).
`d`	Specifies the name of the database to search against.
`e`	Specifies the e-value cutoff for the BLAST search.
`l`	Specifies whether BLAST should be run locally (0 option) or through the NCBI web server (1 option).
`n`	Specifies the maximum number of results to show for each query sequence.

Using a function (**`usage()`**) your program should display usage information to the screen if an option at the command-line is not recognized.

Approach

1. The program should automatically guess whether the input file contains nucleotide sequences or protein sequences, and invoke either `blastn` or `blastp` accordingly.

2. For running BLAST locally, you can assume the user will have the database files in the same folder as your Python script.

3. The result file should be structured as follows (fields should be tab separated):

```
>HEADER_OF_QUERY_SEQUENCE_1
1. DEFINITION_OF_HIT_1 PERCENTAGE_IDENTITY COVERAGE
2. DEFINITION_OF_HIT_2 PERCENTAGE_IDENTITY COVERAGE
...
>HEADER_OF_QUERY_SEQUENCE_2
1. DEFINITION_OF_HIT_1 PERCENTAGE_IDENTITY COVERAGE
2. DEFINITION_OF_HIT_2 PERCENTAGE_IDENTITY COVERAGE
…
```

The definition field comes from the `hit_def` field in the alignment, and it should be limited to the first 50 characters of the string, followed by "..." if the string is longer than 50 characters. For example, the output for P02144.fasta (see Assignment 11) as input would look like this (the numbers for identity and coverage are arbitrary):

```
>sp|P02144|MYG_HUMAN Myoglobin OS=Homo sapiens OX=9606 GN=MB PE=1 SV=2
1. 3RGK_A Crystal Structure of Human Myoglobin Mutant...    0.856 0.995
2. 1MWC_A Wild Type Myoglobin With Co [Sus scrofa] 1M...    0.554  0.844
…
```

The hits should be sorted by significance (BLAST should already do that), and for each hit you should print identity and coverage of the highest scoring HSP (as you did for Assignment 11) if more than one HSP is present.

4. If you create intermediate output files (for example, XML BLAST output files), **make sure to have your program remove them** once parsing is done. *Hint: You can use a function in the* **os** *Python module to take care of that.*

Testing your program

1. Pick a few protein sequences and a few DNA sequences from your favorite genes to use as test (one input test file for DNA sequences, another input test file for protein sequences).
2. If you want, you can use the BLAST NCBI web tool to get a sense of what the results should look like as far as the hits are concerned.
3. Test your program in a modular way (does the local BLAST function as expected? Does the remote BLAST function work as expected? Does parsing of the BLAST output work correctly? Etc.)

Notes and specific requirements:

1. The program should be modular. This implies that your **`main()`** function will be relatively short and most of the work will be done by specific functions.
2. Excluding the initial command, your program must run without human intervention. This means that you will have to run BLAST from within your program (either remotely or locally).
3. Your code must be thoroughly documented and follow the rules of style specified the *Python Coding Style Guide*.

Submitting your program:

Please upload to Canvas a tar or zip archive with:

1. Your code.
2. The input files you used as test.
3. The output files corresponding to each input file.
4. A brief README file documenting how to use the program.