GS 373 Homework 6

Due May 17th before 1:30 PM on Canvas

- (100 points): 4 bioinformatics questions (80 points), 1 programming assignment (20 points).
- Submit answers to the bioinformatics questions in a Microsoft Word document or PDF via Canvas. Your answers do not need to contain the text of the questions, but they need to be clearly labeled (e.g., 1a., 3b., etc.)
- Submit the programming assignment as a separate .py file onto Canvas. The script should be able to be directly run by Python.

Bioinformatics Questions (80 points)

1. Gene structure (20 points)
   Go to the **UCSC Genome Browser** (https://genome.ucsc.edu/cgi-bin/hgGateway use the **hg38** build) and look up the human gene ***CD3E.***
   a) (5 points) **How many transcript variants does this gene have**, according to GENCODE v24? **How many exons does each one have**?
   b) (5 points) Find the "Human ESTs" option below the plot, set the drop-down menu to full and refresh: **What are ESTs and what does this EST track represent**?
   c) (5 points) Based on this track, **which of the transcripts from part a) above has more EST evidence**? (can be answered by eye)
   d) List **5 specific sequence structures** that might be used during *ab initio* gene prediction.

2. Markov models (20 points)
   a. (10 points) **Draw a diagram of a Markov model of DNA sequence** that can generate at least one sequence that **contains all four nucleotides**. Include a valid set of **transition probabilities** (each state's outgoing transitions must sum to 1). Then, **calculate the probability of the sequence CATG** given your model, assuming a 25% probability of starting at any particular nucleotide.
   b. (10 points) Draw a diagram of **another Markov model of DNA sequence in which every possible sequence of the same length is equally probable** (has the same probability according to the model).

3. Hidden Markov models (20 points)
   Imagine some researchers studying the spread of flu infection. In one study, these researchers identified and tracked flu-infected individuals, and measured how long they were highly contagious over the course of their infection. They calculated the share of infected individuals that displayed different symptoms - a fever and/or a cough – while they were contagious, and also during the stages at the beginning and end of their infections when they were not.

   In a second study, the researchers again collected daily records on whether a different group of flu-infected individuals displayed a fever and/or a cough. However, for this study they were unable to measure whether each participant was contagious on each day of their infection. They decide to use a Hidden Markov Model to infer how long and when each participant in the new study was likely contagious.

a. (10 points) **Draw a diagram of this HMM.** Include all arrows, but no need for probabilities.
b. (5 points) What would the **hidden states** of this HMM be? **What data could you use to assign their transition probabilities**?
c. (5 points) What would the **emissions** be? **What data could you use to assign the emission probabilities for each hidden state**?

4. (20 points)
Read the short paper "What is a hidden Markov model?" by Sean Eddy expanding on ideas from lecture. Answer the following questions about the splice site-finding HMM proposed in the paper:
   a. (9 points) What **prior knowledge** about exons, splice sites, and introns is used?
   b. (5 points) **State what calculation should be performed with this model to find the most likely state assignment for a particular site in the nucleotide sequence**.
   c. (6 points) **What is another example of a genomics problem where HMMs would not be appropriate** (besides the one given in the paper) and **why**?

Programming (20 points):

Write a program that calculates the G-C content of several DNA sequences listed in an input data file.

- Your program should take 2 arguments using `sys.argv`, both file names: an input file and an output file. The input file name should correspond with a FASTA file containing information on a list of DNA sequences. An example input file that you can use to test your code is included on Canvas.
- In a FASTA file, every DNA sequence is preceded by a header starting with the character ">". The DNA sequence then follows on the next line. (Sequences in a FASTA file can in reality be split over multiple lines, but for this problem you can assume each DNA sequence is found in a single line following each header line.)

Your program should perform the following steps:

1) Define a function called "`count_nucleotides`" that takes a single DNA sequence string as an argument, and returns a dictionary, where the keys are the 4 bases and the values are the number of occurrences of each base in the provided sequence. Here is more information on dictionaries.
2) Define a second function called "`gc_content`" that takes as an argument a dictionary returned from your `count_nucleotides` function and returns the corresponding fractional GC content.
3) Read in the sequences and headers from the input file and store in an appropriate data structure or structures. See the file-reading examples from Quiz Section.
4) Count the number of total occurrences of each nucleotide in the sequence using your function from step 1, and then use the resulting dictionary to calculate the fractional G-C content for the sequence.
5) In the specified output file, print on each line a sequence header, followed by a space, followed by your calculated G-C content for the corresponding sequence.

Example usage of your program:

```
python homework6_skh.py test_sequence.fasta test_output.txt
```

For the example, test_sequence.fasta contains text formatted as follows:

>sequence1_human
ACTGACTAGGGGGTTTCATAGCA
>sequence2_human
CAGTTTGACTGAGTGCGGAAGTCTAT

Contents of test_output.txt after running the example usage:

>sequence1_human 0.4783
>sequence2_human 0.4615

Outline of your program:

```
import sys

def count_nucleotides(sequence):
     ##First define function to count nucleotides

def gc_content(nuc_counts):
     ##Then define function to calculate G-C % based on nucleotide counts

input_file = sys.argv[1]
output_file = sys.argv[2]

##Read in file

##Count nucleotides, calculate G-C content

##Write result to output file
```