CSE 231, Spring 2008

Programming Project 10

Assignment Overview

This assignment will give you more experience on the use of classes. You will write your own class to do a simplified version of the protein transcription process

This assignment is worth 60 points (6.0% of the course grade) and must be **completed and turned in before 11:59 on Monday, April 7, 2008.**

Background

This project is a simple implementation of the protein transcription process. The basic process is nicely described on wikipedia at http://en.wikipedia.org/wiki/Transcription_%28genetics%29 but instead of doing all that, we will do a simplified version.

The short version (from the computational point of view) is this. A long biological process translates strings of DNA into the amino acids, and this sequence of amino acids is what makes proteins. The 'language' of DNA is a sequence of one of four bases: Adenine, Thymine, Guanine, Cytosine which are usually shortened to ATGC (again, see http://en.wikipedia.org/wiki/DNA for more details). The sequence of ATGC's encoded in DNA is genetic code discovered by Watson and Crick in 1953. The transcription process begins by reading the DNA sequence in groups of three, called codons. Each codon, each sequence of three DNA bases, codes for one of the 20 amino acids that are the basis for all proteins. There are also two control codons, a START codon, which starts transcription, and some STOP codons, which end transcription.

To make this happen, you need a table that maps amino acids to codons (and potentially, vice versa) that we provide. If you examine that table, the 64 combinations of the four bases as codons (4 * 4 * 4) are all represented, and some of the amino acids are coded by multiple codons.

In summary, the transcription process takes in a sequence of DNA bases and yields a sequence of amino acids. You are to write a Python program which does that transcription.

Basic Algorithm

Here is the basic algorithm ignoring important issues detailed below (like the START and STOP codons):

```
Loop through the DNA string:
    Read three characters -- a codon
    Look the codon up in the table to find the corresponding triName
    Print the triName
```

Requirements

Create a class, called `Transcriber`, which will do the transcription process. It is coded as follows:

1. The constructor of the class takes a single argument, a filename from which can be read the amino acid to codon information. The table provided, called codonTable.txt, has the following format:
    a. each line represents an amino acid or transcription command
    b. the line format has space separated fields, which are in order:
        i. proteinName triName oneLetterName listOfCodons
2. You will provide a method called `transcribe`.
    a. It takes a single argument, a string of DNA bases (a string of ATGC characters)
    b. It yields a string of triName amino acids, separated by ':'
    c. Transcribe works as follows:
        i. read the DNA string until the START codon is read.
        ii. for each codon, translate the codon into its triName amino acid.
        For example, if you read the codon `CAG`, you can see from the table entry:

```
Glutamine      Glu Q CAA   CAG
```
That the codon `CAG` translates to the triName `Glu`.

iii. continue reading until the DNA string ends or the STOP codon is read.

3. One odd thing. In the common biological translation (that is, using the provided table), the START codon is the same as the Methionine (met) codon.. When reading begins, that codon codes for START. After transcription begins, that codon codes for Methionine (met).
4. Your transcriber should print an error if:
    a. it does not read a codon from the string (that is, there are less than three DNA bases at the end of the string if transcription is still in process)
    b. if the DNA string ends and transcription never began
    c. if the DNA string ends and transcription never ended

Deliverables

You must use handin to turn in the file **proj10.py** – this is your source code solution; be sure to include your section, the date, the project number and comments describing your code. Please be sure to use the specified file name, and save a copy of your proj10.py file to your H drive as a backup.

Other good information

1. A demo program called proj10.pyc is provided that can be called from test.py.
2. test.py is an example of how the TAs will use your class program. Make sure your class works with the provided test.py
3. Your program should be able to work with any properly coded table.
4. The part of the DNA string before the START is irrelevant as is the part of the DNA string after the STOP
5. The START and STOP commands should be stored separately from the transcription table as they may be redundant. That is, in the common table ATG is the START codon as well as the met codon.

Extra Credit

If you get done early, there are some add-ons that can be put on this project. However, to qualify you must:

- get the regular project done first
- meet with me to discuss what extra credit you will do and what it is worth.