BME 335 – BME Computer Applications

MATLAB Project

Background:

The source for the information used in project is an article that appeared in a magazine IEEE Spectrum in July 2013 (http://spectrum.ieee.org/biomedical/devices/the-dna-data-deluge) and from the Human Genome Project (http://www.genome.gov/10001772)

The human genome project, a project to map all of the human genes (about 24,000) started in October 1990 and was completed in April 2003. There are four different types of DNA nucleotides: adenine (A), thymine (T), cytosine (C), and guanine (G) and the human DNA has roughly 3 billion nucleotides. These are grouped into triplets called codons which is a sequence of three nucleotides. These codons allow the body to code for the 20 nucleic acids that are used to build all of the proteins in the body. One specific codon, TAC, indicates the start of a gene. Any of the three codons, ACT, ATT, and ATC, can indicate the end of a gene.

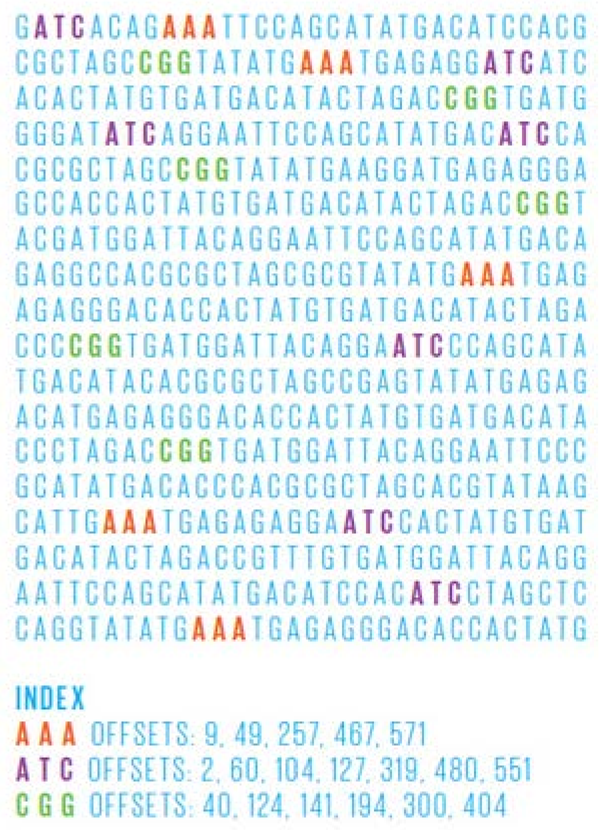
Sequencing a human genome is a computationally intensive process. The first steps begin in a test tube: the DNA strand is split down the middle, it is copied many times, then split into much shorter segments. A machine called a sequencer then identifies the string of nucleotides in each fragment. A computer or a set of computers then takes all of the strands and tries to re-order them using the existing human genome as a reference. The difficulty is that there are hundreds of thousands of strands completely out of order with many overlapping and much repetition. Developing algorithms to take the sequencer data and sort it accurately and efficiently while keeping the cost down is an active area of research

Section A: Genome Indexing:

In this Section, you will take a look at one of the approaches for sorting the sequencer data based on indexing the genome. In Genome Indexing, we look for key sequences of nucleotides and record their locations in the genome. For example, the sequence 'GATTACA' occurs roughly 697,000 times in the human genome.

The picture on the following page (taken from the IEEE Spectrum article), illustrates the idea. As illustrated in the picture, we scan through the genome looking for some key triplets, in this case: 'AAA', 'ATC', and 'CGG'. When a codon is found, the location in the genome is recorded.

Genome Indexing



```
GATCACAGAAATTCCAGCATATGACATCCACG
CGCTAGCCGGTATATGAAATGAGAGGATCATC
ACACTATGTGATGACATACTAGACCGGTGATG
GGGATATCAGGAATTCCAGCATATGACATCCA
CGCGCTAGCCGGTATATGAAGGATGAGAGGGA
GCCACCACTATGTGATGACATACTAGACCGGT
ACGATGGATTACAGGAATTCCAGCATATGACA
GAGGCCACGCGCTAGCGCGTATATGAAATGAG
AGAGGGACACCACTATGTGATGACATACTAGA
CCCCGGTGATGGATTACAGGAATCCCAGCATA
TGACATACACGCGCTAGCCGAGTATATGAGAG
ACATGAGAGGGACACCACTATGTGATGACATA
CCCTAGACCGGTGATGGATTACAGGAATTCCC
GCATATGACACCCACGCGCTAGCACGTATAAG
CATTGAAATGAGAGAGGAATCCACTATGTGAT
GACATACTAGACCGTTTGTGATGGATTACAGG
AATTCCAGCATATGACATCCACATCCTAGCTC
CAGGTATATGAAATGAGAGGGACACCACTATG

INDEX
AAA OFFSETS: 9, 49, 257, 467, 571
ATC OFFSETS: 2, 60, 104, 127, 319, 480, 551
CGG OFFSETS: 40, 124, 141, 194, 300, 404
```

Image Copied from IEEE Spectrum July 2013

Write a script file called count_script that will do the following:

1) Import the text file sequence_long.txt into MATLAB
2) Prompt the user to enter three triplets.
3) Loop through the DNA array and record all of the locations of the given triplets (codons).
4) Create a report file called ***report_count.txt*** and add to this file:
    a. The total number of first codons found and first 10 locations (offsets) for the first codon.
    b. The total number of second codons found and first 10 locations (offsets) for the second codon.
    c. The total number of third codons found and first 10 locations (offsets) for the third codon.
5) In particular for this project report you will need to test your script by using the following three codons: 'AAA', 'ATC' and 'CGG'. Caution must be taken not to include the same nucleotide to count adjacent codons. For example:

    'A' 'T' 'C' 'G' 'G' 'A' 'G'

    ATC is counted but not CGG because the C is already part of ATC

Make sure the file output is clearly labeled and easy to read for a user when they view or print the text file. The file should follow the following format:

```
Name: Your Names Here
Group: Your Team Number
Date: The date here
Section A: Codon Counting

For XXX Codon:
Total Number:   XY    First 10 Positions:   1,2,3,4,5,6,7,8,9,10

For YYY Codon:
Total Number:   XY    First 10 Positions:   1,2,3,4,5,6,7,8,9,10

For ZZZ Codon:
Total Number:   XY    First 10 Positions:   1,2,3,4,5,6,7,8,9,10
```

Section B. Finding Number of Genes and Start/End Positions of Genes in Genome File

Write a script called short_codon that will identify the length (number of nucleotides) in the string, the start position and end position of the gene in the genome file ***sequence_short.txt***. Recall that the codon TAC indicates the start of a gene and any of the three codons, ACT, ATT, and ATC, can indicate the end of a gene. Your function should create a report called ***report_short.txt*** somewhat similar to the one below.

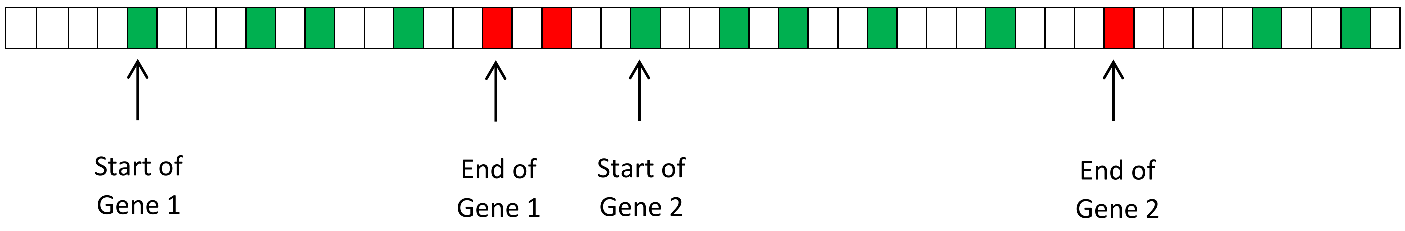
```
Name: Your Names Here
Group: Your Team Number
Date: The Date Here
Section B: DNA Pattern Matching – Short Sequence
Sequence length:   YZ

Start Position:     7     End Position:     18
```

Write a script called long_codon that will identify the start position and end position of all of the genes in the genome file ***sequence_long.txt*** relative to the start of the file. The program should also count how many genes are found. Your function should create a report called ***report_long.txt*** that includes your name, the date, project title, ***the start and end position for every gene*** in the sequence and the ***total number of genes*** in the sequence.

<u>Note</u>:

1. *The long sequence has several thousand start and end codons but only a few of the starts and ends actually indicate the presence of a gene. The beginning of a gene sequence is the first start codon after any end codon and the end of the sequence is when any one of the end codons is identified. Here is an illustration (green = start codon and red = end codon):*



2. *Your program should begin searching for a start codon. Once a start codon is found, the program should do the following:*
   - *Record the start position in the report_long.txt file*
   - *Switch tasks to start looking for an end codon (stop looking for a start)*
   - *Continue to move through the codon array until an end codon is found*

3. *Once an end codon is found, the program should do the following:*
   - *Record the end position in the report_long.txt file*
   - *Switch tasks to start looking for a start codon (stop looking for an end)*
   - *Continue to move through the codon array until a start codon is found*

4. *This pattern should continue until the codon array has been completely scanned.*

5. *Defining a variable to help keep track of whether you are searching for a start codon or for an end codon would be useful. This variable is often referred to as a flag. It would take on one value while you are in "search for a start" mode and another value while you are in "search for an end" mode.*

6. *Each start-end combination indicates another gene. Remember to count the genes and record the start and end position of each gene in the genome in your report_long.txt file.*

Extra Credit: For all your programs, you should have checks in place for user input and file operations that will give clear feedback and exit the program if any errors occurs.

<u>Your Report:</u>
Your Group is required to submit to Blackboard a Word document report that contains the following:

1)	Title Page, indicating person responsible for each part	-
2)	Flow chart for Section A (count_script program)	10
3)	Report_short.txt for the three codons: ‘AAA’, ‘ATC’ and ‘CGG’	10
4)	Flow chart for short_codon	5
5)	Report_short.txt	5
6)	Flow chart for long_codon	15
7)	Report_long.txt	15
8)	A brief description of your programs and explanation how you tested and sections that may not be working or correct.	10

In addition submit to Blackboard the following script files:

1)	Count_script m-file	10
2)	Short_codon m-file	5
3)	Long_codon m-file	15

All scripts should make use of suitable variable names and comments for easy debugging (and grading!)