RYERSON UNIVERSITY

Faculty of Engineering, Architecture and Science

Department of Electrical and Computer Engineering

Program: Biomedical Engineering

Course Number	**BME 808**
Course Title	**Computations in Genetic Engineering**
Semester/Year	Winter 2021
Instructor	P. Siddavaatam

Lab No.	**4**

Report Title	**Gene Annotation by sequence**

Section No.	
Group No.	
Submission Date	
Due Date	

Name	Student ID	Signature*

(Note: remove the first 4 digits from your student ID)

* By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your Academic record under the Student Code of Academic Conduct, which can be found online at: www.ryerson.ca/senate/current/pol60.pdf.

Objectives:

Recall that gene annotation, that is finding genes, is generally approached from four different angles: (1) sequence-based methods, which rely on consensus sequences such as ATG for the start codon and AGGAGG for the Shine-Dalgarno region; (2) alignment-based methods, which rely on a large database (like Genbank) of previously annotated genes; (3) content-based methods, such as codon bias and CpG islands, which rely on certain characteristics of coding sequences or promotor regions; and (4) probability-based methods, such as Hidden Markov Models (HMM) or Neural Nets (NN), which explicitly (HMM) or implicitly (NN) take advantage of a combination of probabilities of nucleotides being at certain positions, such as near the borders of introns and exons.

In the previous lab you implemented an algorithm to find CpG islands in eukaryotes, where sequence based methods are not so useful because of weak consensus sequences in the promoter region and at intron-exon borders. Here, you will implement and use the findPattern algorithm which is good for gene annotation in prokaryotes. The findPattern algorithm is given in Chapter 9 of the course text.

Materials:

1. Python

2. The following findPattern algorithm from Chapter 9 of Bioinformatics, by St.Clair and Visic:

```
findPattern(pattern, searchText, startLoc, stopLoc, increment, threshold)
  patternLen = length of pattern
  textLen = length of searchText
  for each i from startLoc to stopLoc by increment
    count = 0
    j = i
    # count number of matching characters at this location i
    for each k from 0 to patternLen
      if searchText[j] == pattern[k]
        count ++
      j ++
    # check if the number of matched characters is enough
    if count >= threshold
      return i
  return -1
```

3. The diagram of the prokaryote genome that you can find in Chapter 9 of the course text. That diagram indicates that (1) the separation between the Shine-Dalgarno sequence and the start codon is 3 to 7 bp, and (2) the separation between TATAAT (the -10 sequence) and AGGAGG is 50 to 500 bp, and (3) the separation between the -35 sequence TTGACA and the -10 sequence is 15 to 19 bp.

4. The seqeunce for a plasmid, AF516335, from Enteroccocus bacteria, which is a plasmid known to contain several antibiotic resistant genes (ARGs).

Report:

1. Write a program which implements the findPattern algorithm given above. In the program, use the findPattern algorithm to find the first ATG in AF516335. *Show your program and the output. (10 points)*

2. Expand the program so that after finding ATG, it uses that as a starting point to look for a stop codon. The Open Reading Frame (ORF) goes from ATG to the first stop codon, which is one of TAG, TGA or TAA. If the resulting ORF is not at least some minimum acceptable length called SHORTEST = 99 nucleotides, then reject it. If it is long enough (>= SHORTEST) then print the position of its start and stop codons. In any case, continue looking until you find all of the ORFs that are long enough. The difference in the two cases, however, is that when the ORF is acceptable in length, you continue the search after the stop codon, whereas if the ORF was rejected for being to short, you continue the search after the start codon. (That way you won't be finding overlapping ORFs). You will need a loop, of course, to find all of the ORFs. *Show your program and its output. (10 points)* The structure is something like this:

i = 0, countORFs = 0

while i < the length of the sequence

  atg = find ATG but looking from position i in the sequence

  if atg == -1 then break (since there are no more ATGs or ORFs in the sequence)

  stop = find STOP (which could be TAG, TGA or TAA, whichever is first after the current ATG and in the same frame)

  if there was no STOP, set stop to -1

  orfLength = stop - atg

  if orfLength >= SHORTEST :

    countORFs ++

    output 'ORF#', countORFs, from atg to stop

    i = stop + 3 (so that we look for the next ORF after this ORF)

else :

i = atg + 3 (so that we look for the next ORF after the current atg)

Show your program and output. (10 points)

My output started as follows:

ORF# 1 ATG = 37 STOP = 529

ORF# 2 ATG = 745 STOP = 1438

ORF# 3 ATG = 1477 STOP = 2569

3. Expand your program once more so that when it finds a candidate ORF as in Q2, then it then looks for the AGGAGG prior to the ATG with a **separation of 3 to 7 bp**, allowing a mismatch of **one** nucleotide in the AGGAGG matching sequence. *Show your program and its output. (10 points)* Format of the output should be like:

>Highly_resistant_Enterococcus_plasmid_sequence

Length of sequence: 17510

ORF# 1 SHINE = 3734 ATG = 3747 STOP = 4776 LENGTH = 1029

ORF# 2 SHINE = 5808 ATG = 5820 STOP = 6363 LENGTH = 543

...