Protein Translation

April 28, 2021

0.1 CS 101 – Algorithmic Problem Solving I

0.1.1 Spring 2021

Project 3

0.2 Introduction

How do living organisms produce proteins which are structures responsible from all the major functions of a cell? In this project, you will be learning about the entire process of protein translation and creating different protein structures of a strand of E.coli bacteria. For extensive information, the following article is suggested as a good reading:

How does the cell convert DNA into working proteins? by Clancy et al.

The protein translation process starts with DNA to mRNA transcription: >"A DNA sequence has a double helix structure that looks like a staircase consisting of base pairs. There are four types of bases (nucleotides) in a DNA molecule: Adenine (A), Thymine (T), Guanine (G) and Cytosine (C). The bases are paired on a sugar-phosphate backbone structure. Adenine base is always paired with Thymine and Guanine base is always paired with Cytosine." > > From notes of E. Yildirim *Designing Computational Biology Workflows with Perl - Part 2*

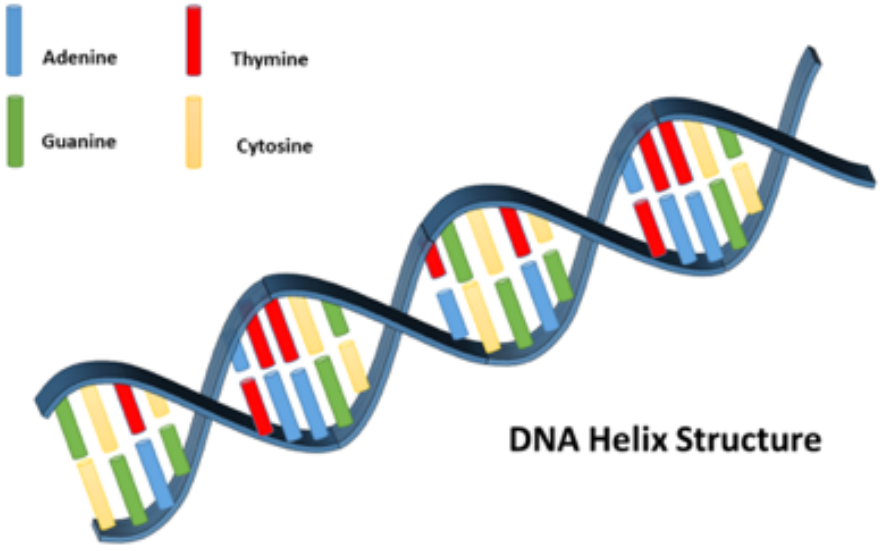

Figure 1. DNA double helix structure (From notes of E. Yildirim *Designing Computational Biology Workflows with Perl - Part 2*)

mRNA molecules are created from a single strand of DNA, through base pair matching with one exception. Instead of Tyhmine, mRNA has a base called Urasil (U).

For example: If the DNA strand has the following bases ATGCCCGTTA, its corresponding mRNA is UACGGGCAAU. A is paired with U, T is paired with A, C is paired with G and G is paired with C.

After mRNA is transcribed, it is translated into codons which are triplets of bases. Each codon has a special meaning and corresponds to a specific aminoacid. Then, aminoacids are sequenced to create a protein.

How do we indicate the start and end of an aminoacid sequence? As it turns out, some codons are reserved to indicate this. For example, `AUG` codon indicates the start of the protein synthesis, while three other codons indicate the end: `UAG`, `UGA`, `UAA`.

0.3 Reading the files into proper structures

In the project folder, there are two files that need to be used for the protein translation of E.coli bacteria. The first file `ecoli.fa` is a FASTA file which contains the DNA sequence data. Here is an excerpt from the file:

```
>Chromosome dna_rm:chromosome chromosome:ASM584v2:Chromosome:1:4641652:1 REF
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTC
TGATAGCAGCTTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGG
TCACTAAATACTTTAACCAATATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTAC
ACAACATCCATGAAACGCATTAGCACCACCATTACCACCACCATCACCATTACCACAGGT
AACGGTGCGGGCTGACGCGTACAGGAAACACAGAAAAAAGCCCGCACCTGACAGTGCGGG
CTTTTTTTTTCGACCAAAGGTAACGAGGTAACAACCATGCGAGTGTTGAAGTTCGGCGGT
ACATCAGTGGCAAATGCAGAACGTTTTCTGCGTGTTGCCGATATTCTGGAAAGCAATGCC
AGGCAGGGGCAGGTGGCCACCGTCCTCTCTGCCCCCGCCAAAATCACCAACCACCTGGTG
GCGATGATTGAAAAAACCATTAGCGGCCAGGATGCTTTACCCAATATCAGCGATGCCGAA
CGTATTTTTGCCGAACTTTTGACGGGACTCGCCGCCGCCCAGCCGGGGTTCCCGCTGGCG
CAATTGAAAACTTTCGTCGATCAGGAATTTGCCCAAATAAAACATGTCCTGCATGGCATT
AGTTTGTTGGGGCAGTGCCCGGATAGCATCAACGCTGCGCTGATTTGCCGTGGCGAGAAA
ATGTCGATCGCCATTATGGCCGGCGTATTAGAAGCGCGCGGTCACAACGTTACTGTTATC
GATCCGGTCGAAAAACTGCTGGCAGTGGGGCATTACCTCGAATCTACCGTCGATATTGCT
GAGTCCACCCGCCGTATTGCGGCAAGCCGCATTCCGGCTGATCACATGGTGCTGATGGCA
GGTTTCACCGCCGGTAATGAAAAAGGCGAACTGGTGGTGCTTGGACGCAACGGTTCCGAC
TACTCTGCTGCGGTGCTGGCTGCCTGTTTACGCGCCGATTGTTGCGAGATTTGGACGGAC
GTTGACGGGGTCTATACCTGCGACCCGCGTCAGGTGCCCGATGCGAGGTTGTTGAAGTCG
ATGTCCTACCAGGAAGCGATGGAGCTTTCCTACTTCGGCGCTAAAGTTCTTCACCCCCGC
```

The second file in the project folder is a CSV file named `codon_table.csv` which contains the `codon` list. Here is an excerpt from the file:

Codon

```
    <td>AA.Abv</td>
    <td>AA.Code</td>
    <td>AA.Name</td></tr>
<tr><td>UUU</td>
```

```
    <td>Phe</td>
    <td>F</td>
    <td>Phenylalanine</td></tr>
<tr><td>UUC</td>
    <td>Phe</td>
    <td>F</td>
    <td>Phenylalanine</td></tr>
<tr><td>UUA</td>
    <td>Leu</td>
    <td>L</td>
    <td>Leucine</td></tr>
<tr><td>UUG</td>
    <td>Leu</td>
    <td>L</td>
    <td>Leucine</td></tr>
<tr><td>CUU</td>
    <td>Leu</td>
    <td>L</td>
    <td>Leucine</td></tr>
```

In the above table, `AA.Abv` represents the abbreviation of the aminoacid, `AA.Code` represents the code for the aminoacid and `AA.Name` represents the actual name of the aminoacid. There are 64 codons in the file. One aminoacid can be represented with multiple codons, they all create the same aminoacid. For example, both `UUU` and `UUC` codons are translated as `phenylalanine`.

Use the following code to read the files into proper objects and structures. DNA string is of `string` type, while codon list is a two-dimensional vector consisting of a list of codons each of which also consists of `codon string` and `aminoacid code`.

[ ]:
```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <cctype>
#include <algorithm>
using namespace std;
string readFastaFile(string path);
void readCsvFile(string path);
string transcribe(string dnaString);
vector<string> translate(string mrnaString);
//global vector of vectors (2Dimensional vector) list
vector<vector<string>> mycodons;

string readFastaFile(string path)
{
    string dnaString;
    fstream newfile;
```

```
    newfile.open(path,ios::in);  // open a file to perform write operation using␣
 ↪file object
    if(newfile.is_open()){   //checking whether the file is open
        string line;
        getline(newfile, line);
        while(getline(newfile, line)){ //read data from file object and put it␣
 ↪into string.
            dnaString += line;
        }
        newfile.close(); //close the file object.
    }
    //cout << dnaString.substr(0,100)<< endl;
    return dnaString;
}

void readCsvFile(string path)
{
    ifstream inFile;
    string codon,Abv, AAcode, name;

    inFile.open(path);
    //read the first entry as column headers
    getline ( inFile, codon, ',' );
    getline ( inFile, Abv, ',' );
    getline ( inFile, AAcode, ',' );
    inFile >> name;
    while (!inFile.eof()) {//while we have not reached the end of file
        //read next entry
        getline ( inFile, codon, ',' );
        getline ( inFile, Abv, ',' );
        getline ( inFile, AAcode, ',' );
        inFile >> name;
        vector<string> temp;
        codon.erase(std::remove_if(codon.begin(), codon.end(), ::isspace), codon.
 ↪end());//clear from white space characters

        temp.push_back(codon);
        AAcode.erase(std::remove_if(AAcode.begin(), AAcode.end(), ::isspace),␣
 ↪AAcode.end());//clear from white space characters
        temp.push_back(AAcode); // add a vector entry consisting of codon and␣
 ↪AAcode
        mycodons.push_back(temp);//add the vector to mycodons 2D vector
    }
}
```

Examples:

mycodons[0][0] => UUU

mycodons[0][1] => F

mycodons[1][0] => UUC

mycodons[1][1] => F

...

0.4 Step 1 - DNA to mRNA transcription

Write a function `transcribe(dna_string)` that creates the mRNA string from the DNA string. Each base in `dna_string` must be matched to its corresponding mRNA base. There might be strange characters in the DNA string other than A, T, C, G. They should be ignored: A $\rightarrow$ U, T $\rightarrow$ A, G $\rightarrow$ C, C $\rightarrow$ G matchings are the only valid ones.

[ ]:
```
string transcribe(string dna_string){
    //this function must take the DNA string and construct a new mRNA string
    //then return the mRNA string
}
```

0.5 Step 2 - mRNA to Protein Translation

Write a function `translate` which accepts the mRNA string as a parameter and creates a string vector of proteins. Each item in the vector is a string that consist of the aminoacid codes of the protein. The function must return the protein vector as a result.

Each protein's aminoacid sequence starts with M (Methionine) which is the starting aminoacid and ends with a Stop aminoacid. So the function should:

- look for mRNA sequences that starts with AUG codon;
- detect the end (UAG, UGA, or UAA codon);
- in between, identify the corresponding aminoacids for the codons to construct the protein;
- save the protein string in the vector(Use push_back function).

[ ]:
```
vector<string> translate(string mrna_string)
{
    //create a protein vector and return it.
}
```

0.6 Use the following `print` and `main` function to connect the processes and print the resulting protein vector

[ ]:
```
void print_protein_list(vector<string> list)
{
    for(string line : list)
    {
        cout << line << endl;
    }
```

```
    cout << list.size() << " proteins listed" << endl;
}

int main()
{
    string dnastring = readFastaFile("ecoli.fa");
    readCsvFile("codon_table.csv");
    string mrnastring = transcribe(dnastring);
    vector<string> protein_list = translate(mrnastring);
    print_protein_list(protein_list);

    return 0;
}
```

The first few lines of the output should look like this:

```
0 -> MDGTHLILKStop
1 -> MKLVISVSRVCLFLMSHVLStop
2 -> MVVVVVMVSIATPDCACPLCLFSGVDCHARKKKLVSIAPLLVRSQLQAAMStop
3 -> MGYSRYGLHKNGLKTALSGGGSAPRATALTFESSStop
4 -> MTIARPAFStop
5 -> MELRWQLStop
6 -> MRRRHDRRTNARLTTLStop
7 -> MDAGRSPRATLQQLQLQDGPSLPRKDEAAISRSGGVVMGVAGQGLGNGLIFMAFRSSWSMRVTTVGTTSAStop
8 -> MRStop
9 -> MSStop
10 -> MDLDFLPNDLGDRHCLADRStop
11 -> MSSLRQVMASPIASQTLTAATATATRRPRStop
12 -> MHRRHNGStop
....
```