BMS 321

Assignment 1

1- Write a program that outputs a list of indices of ALL occurrences of a substring in a sequence.

2- Here's a short DNA sequence:

ACTGATCGAATTCGTATAGTAGAATTCTATCATACAGAATTCTATATATCGATGCGGAATTCTTCAT

The sequence contains a recognition site for the EcoRI restriction enzyme, which cuts at the motif G\*AATTC (the position of the cut is indicated by an asterisk). Write a program which will output and calculate the sizes of all the fragments that will be produced when the DNA sequence is digested with EcoRI.

3- Write a program to ask for a DNA sequence. Translate the DNA into protein. When the codon doesn’t code for anything (eg, stop codon), use “\*”. Ignore the extra bases if the sequence length is not a multiple of 3. Use the file “table.txt” for the translation.

Hint: use a dictionary for the translation table

table = {"UUU":"F", "UUC":"F", "UUA":"L", "UUG":"L", "UCU":"S",

"UCC":"s", "UCA":"S", "UCG":"S", "UAU":"Y", "UAC":"Y", "UAA":"STOP",

"UAG":"STOP", "UGU":"C", "UGC":"C", "UGA":"STOP", "UGG":"W",

"CUU":"L", "CUC":"L", "CUA":"L", "CUG":"L", "CCU":"P", "CCC":"P",

"CCA":"P", "CCG":"P","CAU":"H", "CAC":"H", "CAA":"Q", "CAG":"Q",

"CGU":"R", "CGC":"R", "CGA":"R", "CGG":"R", "AUU":"I", "AUC":"I",

"AUA":"I", "AUG":"M", "ACU":"T", "ACC":"T", "ACA":"T", "ACG":"T",

"AAU":"N", "AAC":"N", "AAA":"K", "AAG":"K", "AGU":"S", "AGC":"S",

"AGA":"R", "AGG":"R", "GUU":"V", "GUC":"V", "GUA":"V", "GUG":"V",

"GCU":"A", "GCC":"A", "GCA":"A", "GCG":"A", "GAU":"D", "GAC":"D",

"GAA":"E", "GAG":"E", "GGU":"G", "GGC":"G", "GGA":"G", "GGG":"G"} #

Extra data in case you want it.

stop\_codons = [ 'TAA', 'TAG', 'TGA']

start\_codons = [ 'TTG', 'CTG', 'ATG']

You can test your script with this sequence:

"CCGGAACCGACCATTGATGAG"

4- Write a program to find the longest common subsequence (not a substring) in a list of sequences.

5- Use dictionaries to get the reverse complement of a string

Enter a sequence: CCTGTATT.

The reverse complement sequence is AATACAGG.

6- Given two Sequences S and T of equal length, the Hamming distance between S and T, denoted by H(S,T), is the number of corresponding symbols that differ in S and T. Write a program that takes two sequences and calculate the hamming distance between them.

Enter the first sequence: GAGCCTACTAACGGGAT.

Enter the second sequence: CATCGTAATGACGGCCT.

Output: The hamming distance is 7

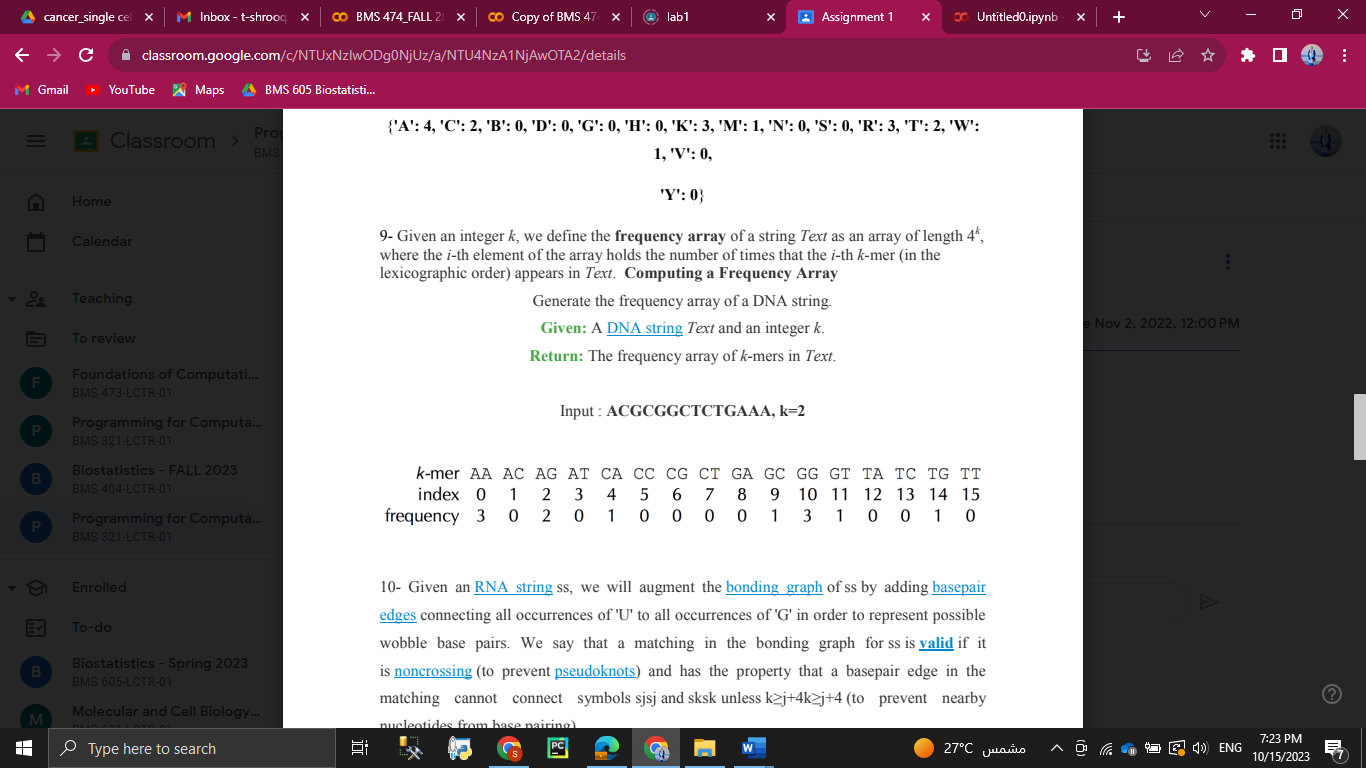
7- Given an integer k, we define the frequency array of a string Text as an array of length 4k, where the i-th element of the array holds the number of times that the i-th k-mer (in the lexicographic order) appears in Text. Computing a Frequency Array

Generate the frequency array of a DNA string.

Given: A DNA string Text and an integer k.

Return: The frequency array of k-mers in Text.

Input: ACGCGGCTCTGAAA, k=2



8-

A:

I. Write a program that build a dictionary contains all possible trinucleotides and their occurrences from a DNA sequence.

Input:

sequence = "AATGATCGATCGTACGCTGA"

II. From the dictionary, retrieve the occurrence of these examples: ('CGA','AAT', 'TCG')

Hint: Assume that there is one reading frame that starts from the first nucleotide

Output:

dict ={'AAT': 1, 'GAT': 2, 'CGA': 1, 'TCG': 2, 'TAC': 1, 'GCT': 1}

CGA 1

AAT 1

TCG 2

B:

Modify “A” to include all reading frames.

Input:

sequence = "AATGATCGATCGTACGCTGA"

Output:

s is AATGATCGATCGTACGCTGA

{'AAT': 1, 'GAT': 2, 'CGA': 1, 'TCG': 2, 'TAC': 1, 'GCT': 1}

CGA 1

AAT 1

TCG 2

s is ATGATCGATCGTACGCTGA

{'ATG': 1, 'ATC': 2, 'GAT': 2, 'CGT': 1, 'ACG': 1, 'CTG': 1}

s is TGATCGATCGTACGCTGA

{'TGA': 2, 'TCG': 2, 'ATC': 2, 'GTA': 1, 'CGC': 1}

TCG 2

9- Given an RNA string ss, we will augment the bonding graph of ss by adding base pair edges connecting all occurrences of 'U' to all occurrences of 'G' in order to represent possible wobble base pairs. We say that a matching in the bonding graph for ss is valid if it is noncrossing (to prevent pseudoknots) and has the property that a base pair edge in the matching cannot connect symbols sjsj and sksk unless k≥j+4k≥j+4 (to prevent nearby nucleotides from base pairing).

See Figure 1 for an example of a valid matching if we allow wobble base pairs. In this problem, we will wish to count all possible valid matchings in a given bonding graph.

see Figure 2 for all possible valid matchings in a small bonding graph, assuming that we allow wobble base pairing.

A screenshot of a computer

Description automatically generated

Given: RNA string ss (of length at most 200 bp).

Return: The total number of distinct valid matchings of base pair edges in the bonding graph of ss. Assume that wobble base pairing is allowed.

Input:

AUGCUAGUACGGAGCGAGUCUAGCGAGCGAUGUCGUGAGUACUAUAUAUGCGCAUAAGCCACGU

Output:

284850219977421