Virus & Finding patterns in Sequences

In this class, we will revise some concepts of the biology of viruses in particular the case of SARS-Cov2. We will introduce the concept of object oriented programming to organize bioinformatics programs. We will then focus on the task of finding sequence patterns. First, by discussing basic algorithms for pattern finding. Then, introduce the concept of heuristics in algorithms and discuss the computational complexity. Finally, we will talk about the application of regular expressions and develop several exercises.

Learning Objectives

- Revise how viruses work and their biology
- Introduce basic algorithms for pattern finding.
- Discuss heuristics in algorithms and computational complexity.
- Describe the regular expressions syntax and its applications in pattern search.

1. Review the slides “Virus”.
2. Review the “Python Object-Oriented Programming”.
3. Solve the exercises on how to implement a class to describe a gene.

Finding patterns in sequences. Develop python programs for the different exercises.

1. Write a function that finds all the occurrences *search_all_occurrences(S, P)*.
   The input is pattern P, and a sequence S. The function should return a list of the indices where P occurs in S. Complete in the *sequence_pattern_finding.py* script.

2. Complete the *hamming_distance(s, p)* function.

3. Adapt the previous function to receive as additional input a number of mismatches *m (m < len(p))* and returns all the occurrences of *p* within a distance of m and respective indices.

4. Create a test function that reads from the input a DNA sequence and a pattern to search and finds all its occurrences. Use the function in *read_fasta.py* and test with one of the sequences in HBA1.DNA.fasta

5. Write a function that, given a DNA sequence, allows to detect if there are repeated sequences of size *k*. The result should be a dictionary with sub-sequences as keys, and their frequency as values.

   def repeated_subsequences_frequency(dna_seq, k = 10):

   …

   Complete in the *sequence_pattern_finding.py* script.

Finding patterns in sequences. Develop python programs for the different exercises.

6. In genomic sequences the introns boundaries are defined by sequence signals that can be recognised by their consensus sequences. Consider the splice sites signals as: **GTG** and **CAG**.

For the given signals find the sequences of all possible introns. The intron sequence will be defined by the contagious stretch between GTG and CAG.

Notes:

- extract the list of indices for the first and sequence signal; operate on the list of indices to find two consecutive indices in the sequence.
- for the analysis ignore the case of letters in the provided file, but use the lower-case sequences in file to compare with your results.

Test with *HBA1.DNA.fasta* and the coronavirus genome sequence.

7. Write a function that given two biological sequences A and B, a pattern size between low and high (integers) where low <= high, finds the frequency of all patterns of size between low and high of A present in B.

def finds_patterns_frequency(seqA, seqB, low, high):

For example consider seqA = “AAATG”, seqB =“CAAATC” and low =2 and high = 3

From seqA will tested:

AA (in B)
AA (in B)
AT (in B)
TG (not in B)
AAA (in B)
AAT (in B)
ATG (not in B)

Frequencies will be:

AA - 2
AT - 1
TG - 0
AAA - 1
AAT - 1
ATG - 0

Report only occurring patterns and report them by decreasing order of their frequency.

Exercise 2. A Class for Gene entity

Consider that a gene is an entity defined by the following attributes:

- name (string)
- chromosome (string)
- start (integer)
- end (integer)
- sequence (string) [assume this is the mRNA sequence]
- numberExons (integer)

Define in Python a Class that allows to specify a Gene. The class should implement the following methods:

- initializer/constructor with gene name
- set and get methods for all the variables
- calculate the length of the gene (by subtracting the end and start position)
- calculate GC content of the sequence
- convert to upper case the sequence

Using the NCBI search in the last class for the gene HBB:

- create an instance for this gene with the respective information
- apply and test all the methods