Scientific Programming 14/01/2020

Before you start

Please write one single python script for each one of the lab parts and one text file with the answers to the theoretical questions.

IMPORTANT: Add your name and ID (matricola) on top of the .py and text files!

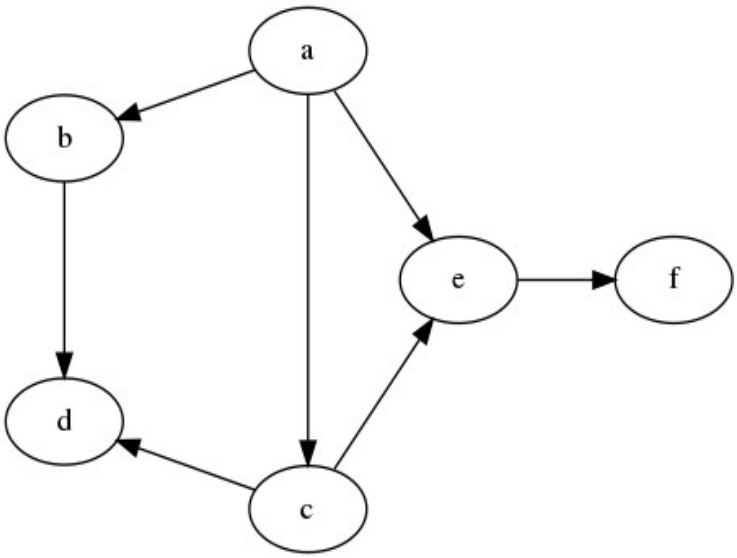
Theory

Please write the solution in a text file.

1. Given a list L of n elements, please compute the asymptotic computational complexity of the following function, explaining your reasoning.

```
def my_fun(L):
      n = len(L)
      if n <= 1:
            return 1
      else:
            L1 = L[0:n//2]
            L2 = L[n//2:]
            a = my_fun(L1) + min(L1)
            b = my_fun(L2) + min(L2)
            return max(a, b)
```

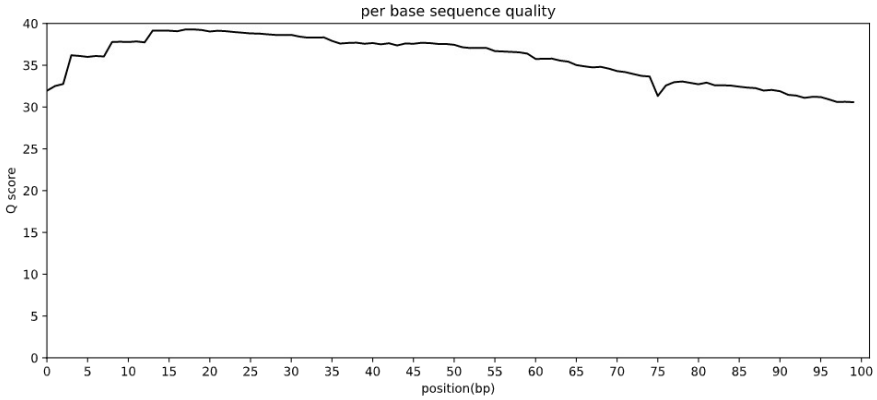
2. Briefly describe what a BFS visit of a graph is. Please provide the order of visited nodes of a possible BFS visit on the following graph (starting from node 'a'):



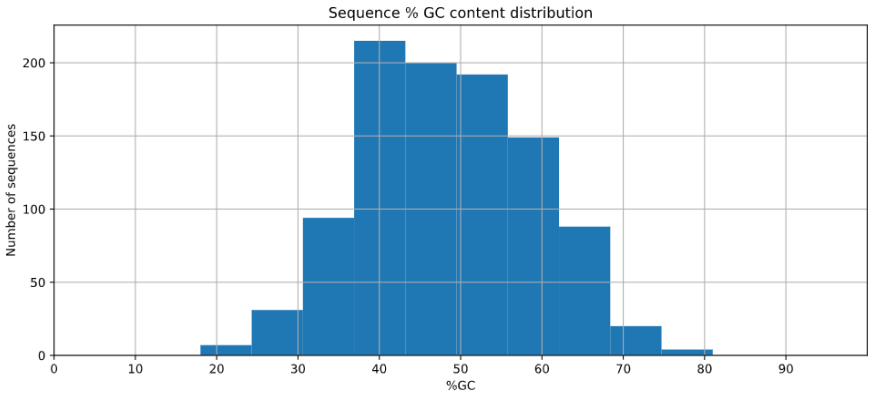
Practical part (A)

Implement the **FASTQAnalyzer** class that provides a small toolkit for sequence properties and quality analysis. The class should provide methods to:

a. **Load** a FASTQ file from a user-provided path and store the sequences within the class.
b. Compute the **per-base mean quality** (i.e. what is the average base quality at each position in the sequences?) and plot them like shown below:



c. Compute the **per-sequence GC content** (i.e. which is the percentage of G and C nucleotides in each sequence) and plot the obtained distribution like shown below:

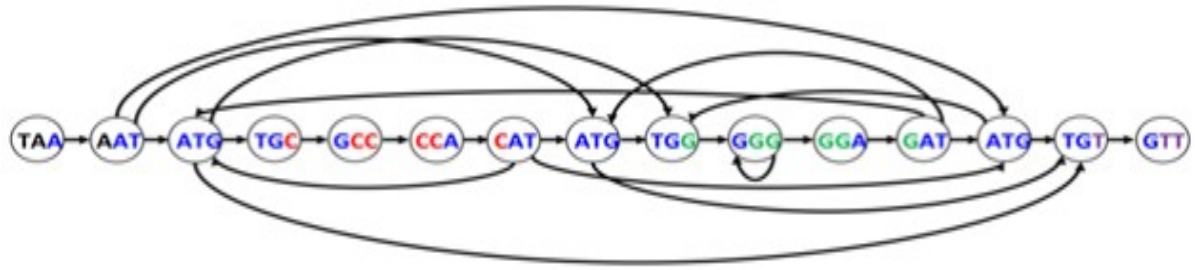


Test the class with the provided FASTQ file (**sample.fastq**).

Hint: Remember that you can use Biopyhton.

Practical part (B)

A De Bruijn graph is a directed graph representing overlaps between sequences of nucleotides. An example is shown below:



To build such graph, we first choose a **k-mer** size, and split the original "reference" sequence (e.g. the genome) into its k-mer components. Then a directed graph is constructed by connecting pairs of k-mers with overlaps between the first k-1 nucleotides and the last k-1 nucleotides. Given the linked-list implementation of a graph, **DiGraphLL**, implement:

1. **createDBG(reference, k)**: this method returns a De Bruijn graph of the reference sequence provided
2. **findSequence(DBG, sequence)**: this method traverses the DBG graph looking if the provided sequence can be expressed with the DBG (i.e. is in the reference sequence)

Test the methods with the following sequences and a k-mer size of 3:
Reference: TATGGGGTGACCGGGATTTACC
Seq1: GGTGAC
Seq2: TGGGCTGGA