DAYZERO DIAGNOSTICS

Software Coding Assessment

Introduction

First, thank you for taking the time to work through this coding assessment. Your efforts here help us get a better sense of how you approach real problems that we work on at DZD, and we hope this project gives you a view into the type of work you might encounter at our company.

This assessment is centered around parsing and organizing sequencing data, something we do frequently at DZD. The starting point is a genomics data file, `SP1.fastq` (file is attached). FASTQ is a text-based format for storing both a biological sequence (usually a sequence of DNA nucleotides, e.g., `GATTTGG`…) and its corresponding quality scores. A *k-mer* is a subsequence of k consecutive letters, e.g., `ATTATTTG` is an 8-mer. Counting the 3-mers within `ATTATTTG` (start at `ATT` and step right, one letter at a time), we have:

```
ATT: 2
TTA: 1
TAT: 1
TTT: 1
TTG: 1
```

We use relational databases like PostgreSQL to store metadata about DNA samples and analysis. In this assignment, you'll use a simpler database, SQLite (https://www.sqlite.org/index.html).

Your assignment is to write Python 3 software to solve **two** problems:

1. FASTQ parsing and database loading

Parse the FASTQ file SP1.fastq and extract the DNA sequences (write your own parser, don't use an off-the-shelf parser)

- Count all the 21-mers within the sequences
- Using the PySQLite Python API for SQLite:
  - Create a `data.db` SQLite database
  - Create a `kmer` table in `data.db` with two columns, (1) the k-mer string, (2) the count
  - Fill the table with the 21-mer counts

2. Match DNA sequences

When comparing DNA sequencing, often we are looking for similar but not necessarily identical sequences. Consider Hospital-Acquired Infection (HAI), scenarios in which a patient in a hospital gets infected by another patient, sometimes due to inadequate sanitation. To detect this, we compare DNA

samples from patient A and patient B to see how closely related they are. Bacterial DNA mutates frequently, so even in the case of a very recent infection, the two samples may not be identical.

Write a Python method `match` that takes two arguments

- `kseq` – a k-mer
- `seq` – a DNA sequence longer than k, possibly much longer

and returns a set (unique elements) containing all k-mers in `seq` that match `kseq` with at most two letters different. For example, given

```
kseq = 'ACGT'
seq = 'ACACACGT'
```

the k-mers in `seq` (in order) are `'ACAC', 'CACA', 'ACAC', 'CACG', and 'ACGT'`. The result is `['ACAC', 'ACGT']`.

How did you test the code for each problem?
How will the code perform with a very large input file?

Please send us your code along with instructions on how to run it.

You may find this tutorial helpful: SQLite Python (http://www.sqlitetutorial.net/sqlite-python/) The tutorial mentions both PySQLite and a more specialized API, APSW, be sure to use PySQLite. The SQLite command line shell is a handy way to inspect databases directly: https://www.sqlite.org/cli.html.

Find more info on fastqs here:
https://support.illumina.com/bulletins/2016/04/fastq-files-explained.html