Invitae Bioinformatics Exercise B.2

Instructions

The following exercise should take ~4-6 hours. We are looking for a solution that you feel exemplifies your skills as a software engineer and bioinformatician. We prefer that you submit your **completed** solution within a week, but let us know if you need additional time so that we can accomodate you. Please follow the same guidelines you would for submitting a pull request to your peers, and let us know if you have any questions!

Please clearly state any assumptions you have made, explicitly document what you believe are key strengths and/or weaknesses of your implementation (in particular, how efficient is your solution?). Include information about how you tested your solution.

Problem Statement

The objective is to write software that translates transcript coordinates to genomic coordinates. For example, consider the simple transcript TR1, which aligns to the genome as follows:

```
COORD           0    5    10   15   20     25   30   35   40   45   50
GENOME:CHR1     ACTGTCATGTACGTTTAGCTAGCC--TAGCTAGGGACCTAGATAATTTAGCTAG
TR1                GTCATGTA-------CTAGCCGGTA-----------AGATAAT
                   |    |           |    |               |   |
                   0    5           10   15              20  24
```

We can compactly express this alignment in the same way that we compactly represent a read alignment in the SAM/BAM format: using a position and CIGAR string. In this case, the (0-based) position is CHR1:3, and the CIGAR string is `8M7D6M2I2M11D7M`. For this exercise, you may assume that the transcript is always mapped from genomic 5' to 3'.

The objective is then to translate a (0-based) transcript coordinate to a (0 based) genome coordinate. For example the fifth base in TR1 (i.e. TR1:4) maps to genome coordinate CHR1:7. Similarly, TR1:13 maps to CHR1:23 and TR1:14 maps to an insertion immediately before CHR1:24.

Problem Specification

The software should be implemented in a language of your choice (with some preference towards python), and should conform to community-preferred style guidelines (e.g., python should be "pythonic" and conform to PEP-8). Code will be evaluated both on correctness and overall quality. Your solution should take the following inputs:

1. A four column (tab-separated) file containing the transcripts. The first column is the transcript name, and the remaining three columns indicate it's genomic mapping: chromosome name, 0-based starting position on the chromosome, and CIGAR string indicating the mapping.
2. A two column (tab-separated) file indicating a set of queries. The first column is a transcript name, and the second column is a 0-based transcript coordinate.

Your solution should handle errors appropriately. The output is a four column tab separated file with one row for each of the input queries. The first two columns are exactly the two columns from the second input file, and the remaining two columns are the chromosome name and chromosome coordinate, respectively. Example input/output is provided below.

Third-party libraries can be used as much as desired except for solving the primary bioinformatics problem of translating coordinates. Correctness is naturally the most important feature of any solution. Following that, the implementation should demonstrate good software engineering practices with an eye towards efficiency and generality.

Bells and Whistles

This exercise can be extended in a number of ways, if desired (this is absolutely not necessary, but it might be interesting to think about):

1. Handle transcripts mapping with reverse orientation, i.e. remove the limitation that transcripts map genomic 5' to 3' by allowing the transcript definition to include a binary variable for direction.
2. Consider mapping genomic coordinates onto transcript coordinates.
3. Consider mapping a transcript *range* onto a genomic range (or the reverse). A transcript CIGAR onto a genomic CIGAR (or the reverse)?
4. A real-world implementation of this code would need transcripts from external sources. Discuss where and how you might obtain these. Additionally, you may want to: search extremely long and/or millions CIGAR strings, as well as perform billions of queries. Briefly describe how you would modify your existing solution to accommodate these scenarios.

Sample Input/Output

Input file 1:

```
TR1    CHR1   3      8M7D6M2I2M11D7M
TR2    CHR2   10     20M
```

Input file 2:

```
TR1    4
TR2    0
TR1    13
TR2    10
```

Output file:

```
TR1    4      CHR1   7
TR2    0      CHR2   10
TR1    13     CHR1   23
TR2    10     CHR2   20
```