Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computational Biomedicine I
Fall Semester 2020

Project 1: Efficient Search and Read Alignment

Assigned on: **21.9.2020** Due by: **11:59pm on 03.11.2020**

Overview

With this exercise sheet, we will present the first practical project of the class. The topic of this project is the alignment of high-throughput sequencing read data to a reference genome sequence. The goal is to build a simple alignment software tool that is capable of generating alignments for the given input data and outputing them in a given reference format.

In particular, in this exercise you will:

a) decide on and implement a genome indexing strategy

b) decide on and implement a read alignment method

c) define and implement routines to read input data and generate correctly formated output

d) define and apply evaluation metrics to judge the quality of your tool

We have split the work into several packages. There is no need for you to follow this division of work exactly, but it should serve as a guide to which steps are deemed to be important. However, it is strongly encouraged that each group member takes on one identifiable technical problem.

Work Package 1.1 – Input Data

We have generated two kinds of input data sets. The first kind (`output_tiny_30xCov*`) is a very small set of approx. 1100 reads generated from a short, artificial reference sequence of only 5000 bases. Along with the read data and the reference sequence, this set also contains the optimal alignment output, as was generated along with the sequence. The alignment output is provided in SAM format. You can use this data set as a quick-to-run test for implementation correctness.

The data sets of the second kind (`output_MxCov*`) consist of read sets simulated using different coverage values from chromosome 22 of the human reference genome. These sets are much more realistic, but still quite small compared to typical sets from whole genome sequencing. We provide three different data sets, for coverage values `M` of 5, 10 and 30, respectively. These data sets will be the input to your program. You can work with the smallest coverage for the project and use the higher coverages for benchmarking at the end.

For each data set, we provide the genome sequence the reads were sampled from and should be aligned to, as well as two read files in fastq format. The two fastq files contain read pairs, where one read originates from one end of a DNA fragment and the other read from the other end. The order of reads in the two fastq

files (`*[12].fq.gz`) is the same. For the small testing data set, we also provide the expected alignments in SAM format.

All input data is available for download from:

▷ https://public.bmi.inf.ethz.ch/eth_intern/teaching/cbm_2020/cbm_2020_project1/

Please download all data and familiarize yourself with the provided data formats. In case of any questions, please use the exercise sessions or e-mail the TAs. All data is text-based and should be human-readable.

The goal of this working package is to devise and implement a reader for the input data.

Work Package 1.2 – Output Data

We expect you to generate your output alignments in standard SAM format. The complete specification of the format can be found here:

▷ https://github.com/samtools/hts-specs/blob/master/SAMv1.pdf

Please familiarize yourself with the output format. To make things easier, we accept submissions that contain only a minimal set of information. For us to accept your submission, the following information needs to be present:

- minimal header including the `@HD` and `@SN` lines
- QNAME field
- FLAG with binary flag indicating whether the alignment is reversed
- FLAG binary flag indicating whether the alignment is secondary
- RNAME field
- POS
- CIGAR

All other fields can be omitted (if the specification allows) or be set to their default values (0 for integers, * for strings). For CIGAR strings, you should use = and `X` to denote pairwise sequence matches and mismatches, respectively.

The goal of this work package is to understand the output format specification and design and implement code that generates correctly formatted output for each of the alignments.

Work Package 1.3 – Genome Index

After learning about different indexing strategies for sets of strings, you should discuss and decide on a strategy you would like to pursue for the projects. Aspects you should include into your discussion are: time and effort to implement said indexing, space and time requirements to construct the index as well as space and time requirements for the alignment task.

The goal of this work package is to devise and implement a method that generates an index on the input data. Ideally, this index can be stored on disk and re-used at a later time. This way the up-front cost for index-creation has to be paid only once.

Work Package 1.4 – Sequence Alignment

Using the genome index generated in work package 1.3, you should design and implement a read alignment strategy.

Think about and discuss which cost functions are useful and appropriate from a biological point of view.

The goal of this work package is to develop an alignment software tool that can find the position(s) of one or many reads provided in fastq format in a fully indexed genome. The output should be written using the output module generated in work package 1.2.

Submission

For development and versioning of your code, we provide each group with a git repository using the department's GitLab instance. The same git repository will also be used for your project submission. Once the project is due, we will take the code on the `master` branch of your group's repository and execute it on the input data.

Code needs to be executable in the provided conda environment. If additional packages have been approved for use by the TA team, please provide a conda recipe. Any solution that is not using Python needs to be provided as a Docker container and a detailed README on how to execute the code. **Please understand that for practical purposes any code that a qualified TA is unable to get running within 10-15 minutes due to missing information, can not be graded**.

Each submission needs to be accompanied by a 5 minute presentation per group member. Each group member should give a brief summary of the group's solution and briefly outline the parts they worked on themselves. Everybody should be using their own slide deck (or other preferred media), but exchanging within and across groups is encouraged. A platform for discussion across groups is provided via moodle.

So one more time: If there is any additional knowledge required to run your code or necessary prerequisites to be made, please add a README file containing all relevant information to your repository. If any part of your code requires compilation, you are required to provide a Makefile.