Introduction to Focus Areas in Bioinformatics – WS19/20

Prof. Dr. Tim Conrad

12 Project 12

- Deadline: 26.01.2020, 23:59
- All files need to be available through your GIT repository, in the directory "Project 12".
- You can work in teams up to 3 people. Please state in the report the group member names.
- If your code is in python, I must be able to run your code within a Google Colab notebook. If your code is not in Python or R, you must provide a manual how to compile and run it on a Linux machine.

We will perform a mapping of shotgun reads to a reference genome using „text" search in genomics sequences and compare the runtime (and memory consumption) of the C++ and Python implementations.

12.1 Implementation

(1) Implement the following four search algorithms

   (1) In Python: A simple search algorithm of your choice (pick one from the appendix) that does NOT use an index.
   (2) In C++: the same simple search algorithm as in the Python version.
   (3) In Python: An index-based search algorithm using the BWT and the FM Index.
   (4) In C++: the same index based search algorithm as in the Python version.

Hints:

- You can use existing implementations – but then you have to mention the author (and the URL where you found the code).
- The implementations of the same algorithm in Python and C++ can be very different. It is only important that you use the same algorithm.

For the index-based C++ implementation, you are supposed to follow the Nanocourse by Reinert et al., which can be found here: http://docs.seqan.de/seqan/learning-resources/fm_index.htm .

(2) Benchmark your solutions using the Nanocourse data.
Compare the wall-clock time to search for 100, 1000, 10.000, 100.000, 500.000, 1.000.000 reads within chr4 of the Drosophila genome. You can either select the respective number of reads randomly from the given list or just use the first n.

(3) Optional: try to estimate and compare the memory footprint of your program during each benchmark run (maximum memory consumption during a run).

12.2 SeqAn Implementation (Optional)

- Implement a search algorithm based on BWS and the FM index using the Seqan library. (Follow the Nanocourse.)
- Compare the wall-clock time (and memory footprint) of your C++ and the SeqAn implementation.

12.3 Deliverables

Your need to upload all source codes and a report to your GIT repository.

- The report should be about 300-600 words in length (this is roughly 1 pages, depending on your layout).
- The report must be delivered in PDF format using the BMC template (including the abstract as defined in project 5).
- The following sections must be present (you can add more if needed):
    - Implementation

      (1) Describe briefly your four implementations (or five, if you did the SeqAn part).
      (2) Demonstrate that your implementations work using a very simple example – e.g. some very short text in which you find a word. Show the results.

    - Benchmarks

      (1) Show the benchmark results comparing the wall-clock run-time of your tests using a table that looks similar to this:

Algorithm	**n=100**	**n=1000**	**…**
Python - no index			
C++ - no index			
Python – BWT & FM index			
C++ - BWT & FM index			
C++ - SeqAn BWT & FM index			

      Do not forget to report how long it takes to build the index, where applicable.

      If a run with a large number of reads takes too long (>10 minutes, excluding building the index) for a particular algorithm, you can leave that one out.

      (2) Optional: create a similar table that shows the memory footprints.

      (3) Briefly discuss your results. Also comment on how these algorithms are used in the read mapping task.