Piotr Styczyński (ps386038)

Algorithms for genomic data analysis | AADG | MIM UW | Bioinformatyka

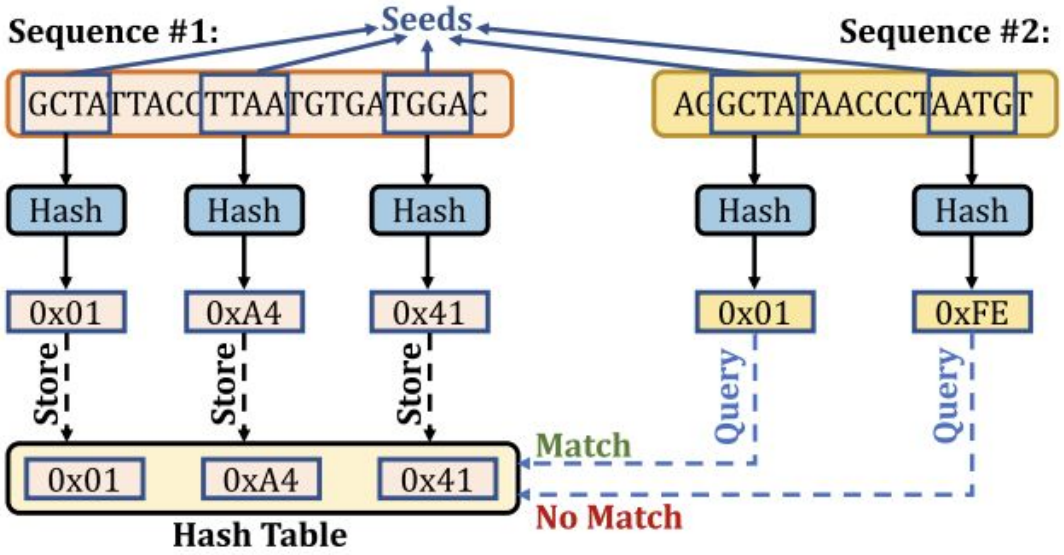

Figure 1. Finding seed matches with a single lookup of hash values.

Aligner implementation

The current implementation works (on the highest-level) somewhere like this:

1. Load all target and query sequences and convert them to numpy arrays
2. I build a minimizer index from the target sequence, which will store all positions and origins for each distinct minimizer found in the reference
3. Ignore too frequent minimizers should (controlled by parameter) in that target index
4. For each query
    i. Build a minimizer index for that sequence
    ii. All minimizers of a query are to be searched against the reference index to find matches. From the list of all matches for a pair of (query, reference), the longest linear chain should represent the best candidate for a good alignment between the pair. That region can be obtained in quasilinear time by solving the longest increasing subsequence problem on the list of minimizer matches.
    iii. I invoke aligner (Needleman-Wunsch) only on the found regions
    iv. I print matched regions with correct position paddings
5. Print summary reports

Usage

```
ps386038@students:~/Code/aadg-genomics-class$ python3 mapper.py  ./data/reference.fasta ./data/reads0.fasta
Poetry (version 1.7.1)
Installing dependencies from lock file

No dependencies to install or update

Installing the current project: aadg-genomics-class (0.1.0)
[info] [cli            ] pipeline.py:31       |  Invoked CLI with the following args: aadg_genomics_class/cli.py ./data/reference.fasta ./data/reads0.fasta
[info] [reporter       ] task_reporter.py:114 |  Starting task: Load target sequence
[info] [reporter       ] task_reporter.py:114 |  Starting task: Create minimizer target index
[info] [prefilter      ] prefilter.py:26      |  Reduced target kmer count by prefiltering: 323604 -> 323281 (Eliminated 0.099% top kmers with f=0.001)
[info] [reporter       ] task_reporter.py:114 |  Starting task: Load query 'read_0'
[info] [reporter       ] task_reporter.py:114 |  Starting task: Load query 'read_1'
[info] [reporter       ] task_reporter.py:114 |  Starting task: Load query 'read_2'
[info] [reporter       ] task_reporter.py:114 |  Starting task: Load query 'read_3'
[info] [reporter       ] task_reporter.py:114 |  Starting task: Load query 'read_4'
[info] [reporter       ] task_reporter.py:114 |  Starting task: Load query 'read_5'
[info] [reporter       ] task_reporter.py:114 |  Starting task: Load query 'read_6'
[info] [reporter       ] task_reporter.py:114 |  Starting task: Load query 'read_7'
[info] [reporter       ] task_reporter.py:114 |  Starting task: Load query 'read_8'
[info] [reporter       ] task_reporter.py:114 |  Starting task: Load query 'read_9'
[info] [cli            ] pipeline.py:89       |  Wrote records to output.txt
[info] [reporter       ] task_reporter.py:96  |  Printing execution summary
Sequence read alignnment (Total time: 5.8 s)
 +-- Load target sequence (176.54 ms)
 +-- Create minimizer target index (5.2 s)
 |   +-- Prefilter target index (67.72 ms)
 +-- Load query 'read_0' (254.27 ms)
 |   +-- Get minimizers (1.24 ms)
 |   +-- Extend (231.18 ms)
 |   +-- Align (21.73 ms)
 +-- Load query 'read_1' (17.83 ms)
 |   +-- Get minimizers (1.34 ms)
 |   +-- Extend (1.92 ms)
 |   +-- Align (14.48 ms)
 +-- Load query 'read_2' (16.79 ms)
 |   +-- Get minimizers (1.17 ms)
 |   +-- Extend (1.39 ms)
 |   +-- Align (14.12 ms)
 +-- Load query 'read_3' (19.8 ms)
 |   +-- Get minimizers (1.06 ms)
 |   +-- Extend (1.78 ms)
 |   +-- Align (16.88 ms)
 +-- Load query 'read_4' (16.95 ms)
 |   +-- Get minimizers (1.05 ms)
 |   +-- Extend (1.57 ms)
 |   +-- Align (14.24 ms)
 +-- Load query 'read_5' (19.65 ms)
 |   +-- Get minimizers (1.11 ms)
 |   +-- Extend (2.03 ms)
 |   +-- Align (16.42 ms)
 +-- Load query 'read_6' (20.31 ms)
 |   +-- Get minimizers (1.09 ms)
 |   +-- Extend (2.13 ms)
 |   +-- Align (16.99 ms)
 +-- Load query 'read_7' (19.82 ms)
 |   +-- Get minimizers (1.03 ms)
 |   +-- Extend (1.95 ms)
 |   +-- Align (16.74 ms)
 +-- Load query 'read_8' (20.89 ms)
 |   +-- Get minimizers (1.18 ms)
 |   +-- Extend (2.34 ms)
 |   +-- Align (17.29 ms)
 +-- Load query 'read_9' (19.05 ms)
     +-- Get minimizers (1.04 ms)
     +-- Extend (1.67 ms)
     +-- Align (16.26 ms)
[info] [reporter       ] task_reporter.py:102 |  Operation completed.
```

Other implementation details

1. Heavy Numpy trickery usage, like:

```
kmers_min_pos = np.add(np.argmin(sliding_window_view(kmers, window_shape=window_len), axis=1), np.arange(0, sequence_len - window_len + 1))
```

2. Nice time reporting

3. Timed each section of the aligner to come up with most efficient implementations (for example using bit-shifting for kmers representation)

- Nice win: reduction of time from 30s to 1.5s for alignments on test0.fasta

Thank you!