July 12, 2021

Assignment 13
Algorithms for Sequence Analysis, Summer 2021

Algorithmic Bioinformatics · Prof. Dr. Sven Rahmann

Hand in date: Monday, July 19, before 20:00
All of these tasks yield bonus points.

Exercise 1: Bloom filter (4 Theory)

(a) Assume that you have a bloom filter with h hash functions, m bits space and a load factor of ℓ (the load factor is the number of 1-bits divided by m). If a random non-present object is queried, what is the probability of a false positive answer? Compute this probability in terms of h, m and ℓ in general, and for $h = 4$, $\ell = 0.7$ and in the limit of large m.

(b) In the same setting as above, after n elements have been inserted, what is the expected load factor? Assume that each hash value is completely random. Give a good approximation of the load factor for $n = m/h$. (Hint: With each element, we are setting h bits to 1, but some of them may already have been set to 1 previously. Remember the Jukes-Cantor correction?)

Exercise 2: Minimizers (4 Theory)

Consider the standard order `A` < `C` < `G` < `T`.

(a) List all (4,4)-minimizers of the string $T =$ `CGATCCTGCACCTCATAG`.

(b) List all (5,3)-minimizers of T.

(c) For a fixed k, what values of w guarantee that there are no gaps between two consecutive minimizers; that is, all letters are covered by at least one minimizer except at most $w - 1$ at each end of the string?

Exercise 3: Genome assembly (6 Programming)

Download the assembly puzzle from https://www.rahmannlab.de/talks/puzzle.pdf. These are 17 double-stranded reads of length 20. The genome has a length of approximately 100. You may cut out the reads and attempt the puzzle by hand (no points).
Write a program to assemble these reads; choose one assembly paradigm:

- overlap-consensus approach: Compute optimal overlaps between all pairs of reads (note the two possible different orientations!) and greedly assemble the pair with the highest overlap score (suggested: match/mismatch/gap: $5, -4, -5$). Remove the two assembled reads and put the new assembled fragment into the pool; repeat until no good overlaps remain.

- DBG or k-mer approach. Extract the k-mer set of the reads (suggestion: $k = 6$ or $k = 7$), remove k-mers that occur rarely, assemble unitigs (branch-free linear chains of k-mers). Be careful with the two possible orientations, so it is best to use canonical k-mer encodings.

In the end, output the assembled genome or the set of assembled contigs. Your implementation should be accompanied by a report describing the implementation. This can be done within the code as well (e.g., detailed docstrings).