May 11, 2021

Assignment 4
Algorithms for Sequence Analysis, Summer 2021

Algorithmic Bioinformatics · Prof. Dr. Sven Rahmann

Exercise 1: Shortest unique substring (4 Theory)
Find the unqiue shortest substring of $s =$ `abbaabbaa`. For this, first construct the suffix array and lcp array for the text $T =$ `$`. Explain in detail how long the shortest unique substring at each rank has to be, and which candidates have to be discarded because they are only unique with the sentinel.

Exercise 2: Maximal unique matches (4 Theory)
Find all maximal unique matches between the strings $s =$ `aaababb` and $t =$ `bbaaba`. For this, first construct and show the suffix array and lcp array for the text $T = s\$t\#$. Explain in detail how you find the MUMs from the lcp array of T by considering *all* local maxima and giving reasons for discarding some of them.

Exercise 3: Induced sorting by pen and paper (4 Theory)
For text $T =$ `ACATACATACATACCATACATACAT$`, do the following:

(a) Compute the type array, LMS-substrings, and the string's representation based on the reduced alphabet.

(b) Is the order of suffixes at LMS-positions determined by the LMS-substrings computed in the previous subtask? If not, illustrate how the induced sorting algorithm recurses to find the order of LMS suffixes and repeat (a) and (b) recursively until the process terminates.

Exercise 4: Maximal Unique Matches (4 Programming)
We want to compare two bacterial genomes:

- *Escherichia coli* strain K-12, substrain MG1655,
- *Escherichia coli* O157:H7 strain Sakai (EHEC).

Write a program that takes two FASTA files as input, parses the FASTA files, creates a concatenated text T, creates the *pos* and *lcp* arrays (using any algorithm, even naive!), and then runs the MUM algorithm to output all maximal unique matches between the genomes in the FASTA files.
Details: A small CLI and FASTA file parsing are provided for you. FASTA files may contain more than one sequence. If the first FASTA file consists of sequences $s_1, \dots, s_n$, the text $S = s_1\&s_2\&\dots s_n\&\$$ is created. If the second FASTA file consists of sequences $t_1, \dots, t_m$, the text $T = t_1\%t_2\%\dots t_m\#$ is created. Note that $\# < \$ < \% < \&$ in ASCII order (35, 36, 37, 38), so the sentinels are smaller than the separators, and the special characters of the second sequence are smaller than the corresponding characters of the first sequence. The algorithm is then applied to the concatenation ST. Fill in your MUM

search code in the function where you find the `TODO` marker. If you cannot get the function to compile with numba, remove the `@njit` decorator!
Two small examples are provided (`example1.fa`, `example2.fa`). The larger genomic FASTA files are compressed; you need to run `gunzip` first to use them.

Remarks

- 50% of points in each category theory and programming (over all exercises and not each assignment sheet separately) are necessary to take the exam.
- You are allowed to work in groups of two, and only one of the group members needs to submit.
- Submission is via GitHub Classroom (as discussed in the tutorials).
- Source code must be sufficiently commented and documented to be understandable.
- Compilation instructions and tools must be provided if required (e.g., a Makefile).
- In addition to source code, the output must be provided.
- Also, a file AUTHORS with your name(s) must be provided.
- Copying between groups will result in zero points for all involved groups.

Hand in date: Monday, May 17, before 20:00