Project 3: Constructing suffix trees from LCP and suffix arrays

Write a program, `gen-lcp`, that uses the construction algorithm from the previous algorithm to generate a suffix tree. Then use this suffix tree to output a file that contains the suffix indices in lexicographical order and the LCP array.

Finally, write a program, `search-st2`, where you have replaced the suffix tree construction algorithm that you wrote in the previous project with the new construction algorithm.

The "new" construction algo.? What exactly is that?

The program should support both generating and serialising a suffix tree and search for reads using it.

Give the tool an option, `-p`. If the program is called with this, is should take an additional argument that should be a FASTA file. In this case, create files that can be used to rebuild the suffix tree in linear time.

What is the idea of serializing a tree, if it can be constructed in linear time (McCreight), anyway ?

For searching, the `search-st2` program should take the same input and produce the same output as in all the previous projects.

So if I implement McCreight, I just need the string?

You choose what the preprocessing files should contain and how many there should be, but you should let the file names depend on the name of the FASTA file so the program does not need more parameters for searching than the programs in the previous projects.

Evaluation

You should upload your solution to Blackboard before the lecture **26/3**.

The solution you upload should contain code for building the `search-st2`program (with the appropriate build tools for the programming language you have chosen), and a one-page PDF file where you address the following questions:

- Insights you may have had while implementing and comparing the algorithms.
- Problems encountered, if any.
- An experiment that verifies the correctness of your implementations.
- An experiment that verifies that your implementation of `search-st2` uses no more time than O(n) for constructing the suffix tree and no more than O(m) for searching for a given read in it. Remember to explain your choice of test data. What are "best" and "worst" case inputs?

So, I can decide freely which algo. to impl.?

Handin of project 3