May 4, 2021

Assignment 3
Algorithms for Sequence Analysis, Summer 2021

Algorithmic Bioinformatics · Prof. Dr. Sven Rahmann

Exercise 1: Examples (4 Theory)
For this task, we define the order $\texttt{\$} < \texttt{A} < \texttt{B} < \texttt{C} < \cdots < \texttt{Z}$.

(a) Construct the suffix tree for the string $s = \texttt{ATANANTAN\$}$. Order the children of each suffix tree node lexicographically. Also specify the corresponding suffix array *pos*.

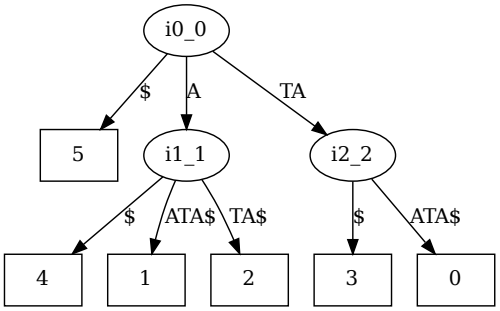
(b) Construct the *lcp* array in the naive way (left-to-right). How many character comparisons (successful + unsuccessful) do you need in total?

Exercise 2: Nodes and edges in suffix trees (4 Theory)
Prove the following lemma: A suffix tree of string $s\$$ with $|s\$| = n$ has exactly n leaves and there exist at most $n - 1$ inner nodes and at most $2(n - 1)$ edges.

Exercise 3: Suffix trees from suffix arrays (4 Theory)
Given the *pos* and *lcp* arrays, how can you reconstruct the suffix tree? Formally describe a left-to-right scanning algorithm to do this.
Hint: The *lcp* array gives the string depth of the deepest internal node of the tree between two lexicographically adjacent suffixes. Keep branching off from the currently rightmost path at the correct string depth.
Bonus: Analyze the running time of your algorithm!

Exercise 4: Drawing suffix trees from suffix arrays (4 Programming)
Implement your solution of the previous exercise to construct the suffix tree in DOT format (explained at https://en.wikipedia.org/wiki/DOT_(graph_description_language); see also the example below). Use circles for inner nodes and boxes for leaves. Annotate leaves with suffix starting positions, as usual. Annotate inner nodes with `i`, a running ID number, an underscore, and their string depth (as shown below; the root is `i0` with a depth of 0, so `i0_0`). Annotate edges with the substring that they represent.
There are many programs to visualize the corresponding graphs, for instance the graphviz package that contains the command `dot`. It can be used like this, resulting in the figure below: `dot -Tpdf input.dot > output.pdf`
The DOT file for the tree of `TAATA$` should look similar to the following:

```
digraph mytree {
  i0_0 -> 5 [label="$"]
  5 [shape=box]
  i0_0 -> i1_1 [label="A"]
  i1_1 -> 4 [label="$"]
  4 [shape=box]
  i1_1 -> 1 [label="ATA$"]
  1 [shape=box]
  i1_1 -> 2 [label="TA$"]
  2 [shape=box]
  i0_0 -> i2_2 [label="TA"]
  i2_2 -> 3 [label="$"]
  3 [shape=box]
  i2_2 -> 0 [label="ATA$"]
  0 [shape=box]
}
```



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 10, before 20:00, using GitHub Classroom