UNIVERSITÄT DES SAARLANDES

ZBI ZENTRUM FÜR BIOINFORMATIK

Assignment 03

Algorithms for Sequence Analysis

Sven Rahmann and Jens Zentgraf

07.05.2024, due **13.05.2024**

03.1 Matching Statistics (6 Theory)

Let s, t be strings with $|s| \leq |t|$.
Assume we have a suffix tree of s (the shorter string) **with suffix links**.

Let $|t| = n$. Let $M = M[0:n]$ be an array of integers ("**matching statistics**"), such that $M[i]$ is the length of a longest substring that starts at position i in t, and that occurs also (somewhere) in s.

a. Describe how to compute M in $O(n)$ time, under the conditions stated above.
b. Describe how to obtain the longest common substring of s and t in $O(|s| + |t|)$ total time, given only s, t, with the help of M.

03.2: BNDM (4 Theory)

For this task, $\Sigma := \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, $P := \mathtt{AGATATAGATCAG}$ and $T := \mathtt{AAGTAAAGTAGAGATAGATATAGATTAGCGT}$.

Illustrate the position of search windows when BNDM is used to search for P in T. Show how far each window is read, and how far the window is shifted in each step.

03.3: From ESA to Suffix Tree (4 Theory + 4 Programming)

Theory Part (4 Theory)

[A] Given the `pos` and `lcp` arrays, how can you reconstruct the suffix tree?
Formally describe a left-to-right scanning algorithm to do this.
Analyze the running time of your algorithm.

[B] Run your algorithm by hand on the following input:

$$\texttt{pos} = [10, 9, 4, 7, 2, 5, 0, 8, 3, 6, 1], \quad \texttt{lcp} = [-1, 0, 1, 1, 3, 3, 5, 0, 2, 2, 4, -1]$$

Additionally, find a fitting text (over any alphabet).

Reminder

The `lcp` array gives the string depth of the deepest internal node of the tree between two lexicographically adjacent suffixes.

Programming Part (4 Programming)

Implement your algorithm to construct the suffix tree in DOT format
(explained **here**; see also the example on the **next slide**).
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 (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 (py)graphviz package that contains the command `dot`.
It can be used like this, resulting in the figure on the next slide:

```
dot -Tpdf input.dot > output.pdf
```

Hint: Do it in 2 passes: First build the tree, then output it in DOT format.

03.3: Programming Part Example

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]
}
```