1 Longest common substring

In class, I showed how to compute two flags, $f_1(v)$ and $f_2(v)$ for each node $v$ in the suffix tree for $T = T_1\$T_2$. Here, $f_i(v) = 1$ iff there exists a suffix inside the subtree rooted at $v$ that is from $T_i$ (i.e. its starting position is within $T_i$ of $T$).

Now write a simple program to set the two flags for all nodes in the suffix tree. Use the following notations: the root of suffix tree is $v_r$; the length of $T_1$ is $n_1$ and the length of $T_2$ is $n_2$; the suffix of a leaf node $v$ is $\text{suffix}(v)$; the number of children of node $v$ is $\text{nchild}(v)$; the children of node $v$ are $\text{child}(v, 1), \text{child}(v, 2), \ldots \text{child}(v, k)$ where $k = \text{nchild}(v)$. Please properly define your notations if you need more to work with. Do not worry too much syntax. Making the logic of the program work is what I am looking for.

2 Construct suffix tree from suffix array

2.1 Practice

Let $T = xabzac$. First tell me what is the suffix array for $T\$$. Also write down its LCP array. Then, use the suffix array and the LCP array to build the suffix tree for $T\$$. Show at least one intermediate step to illustrate how the construction works.

2.2 Why does the approach work?

In class, I said when creating suffix tree from suffix array (with LCP array), you can just check the LCP array to determine where you need to create a new branch within the path from the root to the last added leaf: the LCP value tells you where you should break. Now explain why we can safely ignore the previously added suffixes other than the last one. In particular, is it possible that the newly created branch will follow some existing branches leading to some other previous suffixes?

3 Application of suffix tree

We are given a string $T$ of length $n$. We want to compute values $N_i(T)$ for $1 \leq i \leq n$. Here, $N_i(T)$ is equal to the length of the longest suffix of the substring $T[1..i]$ that is also a suffix of the full string $T$. For example, for $T = cabdabadab$, $N_3(T) = 2$ and $N_6(T) = 5$ (note array index starts from 1). Remember that you need to compute $N_i(T)$ for each $i$ and your algorithm should run in time proportional to $n$ (i.e. $O(n)$). Computing a single $N_i(T)$ can be easily done by direct comparison. The goal is to compute each $N_i(T)$ with constant steps on average.

4 BWT

We are given a string $T = \text{“tartar”}$. Show how to perform forward Burrows-Wheeler transformation, backward Burrows-Wheeler transformation, and how to search for pattern $P = \text{“tar”}$ in the BWT (i.e. the L array). Note you need to show the major steps and briefly explain the critical parts.