1 Ukkonen’s algorithm

Read the posted chapter from Gusfield’s book on Ukkonen’s algorithm. Then answer in your own words about several aspects of this algorithm. Please be precise and right to the point.

1. I mentioned in the class: in the implicit suffix tree, “once a leaf, always a leaf”. That is, when a leaf is created for one prefix of $S$, this leaf will remain a leaf in later phases. Why is this?

2. In class, I said suffix links always exist for each internal node in the implicit suffix tree. Why?

3. In class, I said when finding the location in the implicit suffix tree to refine to allow a new suffix, we don’t need to explicitly compare the entire string label of an edge. Instead, we just need to match the first letter of the edge label; if it matches, we move to the node below; if it does not match, we stop. Now why will this work?

2 Rotations

Given a string $S$, and consider all $|S|$ rotations of $S$. That is, let $S = PQ$ (P is a prefix and Q is a suffix of S), $S' = QP$ is a rotation of S. Now design an algorithm that outputs a list of lexicographically sorted rotations in time linear to the length of $S$.

3 Longest common extensions

Read carefully the posted tandem repeat writeup by Gusfield. Find a simple way to compute longest common extensions (LCE) as needed in it. Note: you must show that your way of computing LCE as needed by the divide and conquer algorithm still allows overall $O(n \log n + z)$ time. Recall LCE asks for the length of longest common prefix of two given suffixes. You can not use complex tool like constant time lowest common ancestor in suffix trees. In fact, you do not even need suffix tree. This looks not easy, but some simple algorithm we went over in class should help...