
1: Suffix Trees and Tries

Highly recommend using this simulator when trying to visualize suffix trees:

<https://brenden.github.io/ukkonen-animation/>

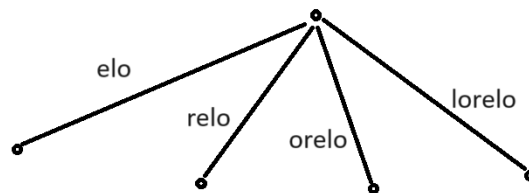
1. Explain briefly how we can use a suffix tree to find the longest repeated substring within a single string? What about the longest common substring of a set of strings?
2. Why is a \$ symbol necessary at the end of a string when setting up a suffix tree? What would happen if the \$ was not there?
3. Why do we use suffix tries and not prefix tries? What would a prefix trie of the string "apple" look like?
4. Show the following statement is false: in any suffix tree (with or without termination symbols), every possible suffix of the string will be exactly represented in a path from the root node to a leaf node.
5. **This problem is not actually covered until after Fall break.** Create a (generalized) suffix trie for the set of words $S = \{bats, cats\}$ over the alphabet $\Sigma = \{a, b, c, s, t\}$. (note in lecture we spent most of the time on the compressed generalized suffix tree, but this is the same without the edge compression)

2: Solutions

1. To obtain the longest repeated substring within a single string, we need to find the longest path from the root node to an internal node (not a leaf node) within the generalized suffix tree. This can also be thought of as traversing the suffix tree from the root to the deepest internal node. The longest common substring of a set of strings can be found by assigning each string a label (for instance, giving unique terminating symbols such as #, \$, % etc.). Then, the longest common substring would involve traversing to the deepest node in the tree containing symbols of all the strings in the set.
2. The \$ symbol is the unique identifier that ensures every suffix in the tree is unique. With suffix trees that contain a termination character, it would be possible to traverse backwards from each leaf node to obtain every unique suffix within a string. If the \$ symbol was not there, it would be possible to have overlapping suffixes that would make interpretation of the tree more difficult.
3. Any prefix trie of a string would simply look like the letters corresponding to the string connected in a straight line. For instance, the prefix tree of the string "apple\$" would look like:



4. In a suffix tree without termination symbols (\$), strings for which a suffix matches the prefix of another suffix will not have that suffix represented in the tree. For instance, for a suffix tree of the string "lorel",



would look like this:

In this case, since "lo" is both a suffix and the prefix of the suffix "lorel", it will not be represented as a path from the root to a leaf node.

5. The trie should look similar to the following:

