6.851 Advanced Data Structures (Spring'12)

Prof. Erik Demaine TAs: Tom Morgan, Justin Zhang

Problem 9 *Sample solution*

String matching. Construct a compressed suffix trie over $S = S_1\$_1 S_2\$_2$, and then trim below $\$_1$ so that all leaves are either $\$_1$ or $\$_2$. The construction takes $O(|S_1| + |S_2| + \text{sort}(\Sigma))$, and the trimming can be done with a traversal of the trie in $O(|S_1| + |S_2|)$ time.

Traverse the trie and record for each node whether it has a $\$_2$ in its subtree as well as the number of $\$_1$s in its subtree. A given node corresponds to a substring appearing in S_2 iff it has an $\$_2$ in its subtree and it corresponds to a number of substrings in S_1 equal to the number of $\$_1$s in its subtree. To compute the total number of substrings of S_1 of length at least k that occur in S_2, traverse the tree and sum the number of $\$_1$ leaves for each node which has an $\$_2$ leaf and has letter depth at least k.

This procedure would work correctly on an uncompressed suffix trie, but it does not account for those substrings corresponding to nodes compressed into edges. To correct for this, when adding the number of $\$_1$ leaves for a node, multiply it by the length of the edge to its parent (or if the parent has letter depth less than k, multiply it by the node's letter depth minus $k - 1$).