Assignment 6

Timothy Yong, Eric Bronner, Aedan Dispenza, Jason Davis

11/11/14

PROBLEM 1

1. Build a prefix tree out of the set of strings, D. $O(n)$
2. Query prefix tree to find S_1 and S_2. $O(m)$ (Essentially, the tree runs in $< O(n), O(m) >$)
3. Compute $LCA(S_1, S_2)$. $< O(n), O(1) >$

   This algorithm runs in $< O(n), O(m) >$, but can be reduced by hashing all leaf labels to $< O(n), O(1) >$.

PROBLEM 2

a. Construct a tree out of the string such that palindromes are recursively defined, and all other substrings are just one node. The number of edges in a tree is $O(n)$ in respect to the length of the string, so there cannot be more than linear palindromes.

b. Build a non-compressed suffix tree out of the string, and another of the reversed string. Take the intersection of these trees. Trim all branches that are not palindromes. $< O(nm), O(m) >$

PROBLEM 3

Build a non-compressed suffix tree out of the string, and another of the reversed string. Take the intersection of these trees; whatever intersects is a palindrome. Use DFS to preprocess the depths of the substrings. $O(n)$

PROBLEM 4

Construct a compressed suffix tree T out of S. Remove all leaf nodes and traverse the tree. Get the intersection between the tree returned from the algorithm from problem 2 and T, and store it in U. Subtract U from T. The longest remaining substring is the longest repeated substring.