Problem 1: Isomorphism in suffix trees (25 points)

Prove the following statement:
In a suffix tree \( T \), the subtree rooted at a node \( u \) is isomorphic to the subtree rooted at a node \( v \) if and only if there is a directed path of suffix links from node \( u \) to node \( v \) and the number of leaves in the two subtrees is equal.

Problem 2: Suffix trees and MUMs (25 points)

Given strings \( x \) and \( y \), a maximal unique match (MUM) is a substring \( w \) that occurs precisely once in both \( x \) and \( y \) and is not contained in any longer word with this property.

- Given a threshold \( k \), describe an algorithm that produces all the MUMs in \( x,y \) longer than \( k \) using the suffix tree \( T \) built on the concatenation \( x\$_1y\$_2 \) where \( \$_1, \$_2 \) are unique separator symbols.
- Given a threshold \( k \), describe an algorithm that produces all the MUMs in \( x,y \) longer than \( k \) using only the suffix tree \( T \) built on the string \( x\$ \). You are not allowed to build a suffix tree for \( y \), or add the suffixes of \( y \) to the suffix tree of \( x \).

Your algorithms have to be time-efficient.

Problem 3: LCP in suffix arrays (25 points)

An essential component in the procedure to search for a pattern \( y \) in the suffix array of \( x \) is the availability of longest common prefix information between any two suffixes of \( x \). In class, we have stated that if one has LCP for all adjacent (sorted) suffixes, he can get the LCP for any other pair of suffixes. Prove the following lemma.

**Lemma:** Let \( \text{LCP}(i,j) \) be the length of the longest common prefix of the suffixes specified in position \( i \) and \( j \) in the suffix array of \( x \). Then, when \( j > i + 1 \) we have \( \text{LCP}(i,j) = \min_{k=i,...,j-1} \text{LCP}(k,k+1) \)

**Hint:** Show that the right-hand side of the equation above is both a lower- and an upper-bound on the left-hand side.
Problem 4: Finding maximal repeats using suffix arrays (25 points)

A maximal repeat in a string $x$ is a triple $(i, j, l)$ such that $x$ contains a repeat of length $l$ starting at positions $i$ and $j$, and this repeat cannot be extended further to the left or right. Formally, $x[i : i + l - 1] = x[j : j + l - 1]$, but $x[i - 1] \neq x[j - 1]$ and $x[i + l] \neq x[j + l]$. Given a string $x$, design an algorithm that finds the longest maximal repeat in $x$ in time $O(|x|)$, using a suffix array.

Note: You are expected to work on this assignment on your own. Anything that you submit that comes from “external sources” (a friend, a web page, a book, etc.) must be acknowledged and will be graded accordingly.