02-713 Homework #8: Suffix Trees
Due: Apr. 9 by 9:30am

You may discuss these problems with your current classmates, but you must write up your solutions independently, without using common notes or worksheets. You must indicate at the top of your homework who you worked with. Your write up should be clear and concise. You are trying to convince a skeptical reader that your answers are correct. Your homework should be submitted via Autolab (`https://autolab.cs.cmu.edu/02713-s14/`) as a typeset PDF. A LaTeX tutorial and template are available on the class website if you choose to use that system to typeset. For problems asking for an algorithm: describe the algorithm, an argument why it is correct, and an estimation of how fast it will run. Use O notation for running times.

1. A *k-mismatch palindrome* is a string xy where, $|x| = |y|$ and reverse(y) and x are the same in all but at most k positions. Give an $O(kn)$-time algorithm to find all the k-mismatch palindromes in a string S of length n.

2. Suppose you are given a string s of length n. Describe an $O(n)$-time algorithm to find the longest string t that occurs both forwards and backwards in s. Your algorithm must use suffix trees or generalized suffix trees.

   For example: If $s =$ `yabcxqcbaz`, your algorithm should return $t =$ `abc` or $t =$ `cba` because both `abc` and its reverse `cba` occur in s and no longer such string exists.

3. **DNA contamination.** When sequencing DNA, often contaminant DNA (e.g., bacteria in the lab equipment, the operator's DNA, etc.) is accidentally sequenced in addition to the target DNA. Suppose you are given a set of strings $C_1, \ldots, C_K$ representing the DNA of known contaminants, where $\sum_i |C_i| = n$.

   A string S of length m is sequenced. Give an algorithm with runtime $O(n+m)$ that finds the locations of **all** the substrings of S that occur in some C_i and that are longer than a given parameter t. (We don't care *which* C_i any contaminant comes from, but we do want to know where every contaminant instance occurs in S.)

4. Let S be a string of length n. Give an $O(n)$-time algorithm to find the longest repeated substring of S such that at least two copies of the substring do not overlap in S.

5. **02-713 only.** Let T be a suffix tree for string S. Let $\mathbf{str}(u)$ be the string that is spelled out when walking from the root of T to a node u. A node in T is called *left diverse* if the occurrences of $\mathbf{str}(u)$ in S are not always preceded by the same character. For example, if $S = ababacb$ then the node representing ba is **not** left diverse since ba is always preceded by a. But the node with $\mathbf{str}(u) = b$ *is* left diverse because sometimes b is preceded by a and sometimes by c.

   Give an $O(|S|)$ algorithm to identify all the left-diverse nodes in T.