CS224 HW2 Due October 15, 2009

Problem 1. For a string A of length n, there are n cyclic rotations of A. For example, if the string is A = abba, the 4 cyclic rotations are: abba, bbaa, baab, aabb. If we lexicographically order those four strings, they are ordered aabb, abba, baab, bbaa. The last character of each string, in the lexicographic order of the strings, concatenated together creates the string A' = baba. In general, given string A, create the string A' of length n by forming the n cyclic rotations of A, sorting them in lexicographic order, extracting and concatenating the last character of each of the strings, in sorted order. Index I points to the position in A' occupied by the last character of the original string. In the example, I = 2. Let (A',I) denote the encoded string and the pointer to the character contributed by the original string. So the encoding of abba is (A',I) = (baba, 2).

Problem 1a. Devise an efficient algorithm to reconstruct the original string A from (A',I).

Problem 1b. Devise an efficient algorithm to construct A' from A using a suffix tree.

Problem 2. **Suffix-prefix matching**. Give an algorithm that takes in two strings α and β, of lengths n and m, and finds every suffix of α which exactly matches a prefix of β. The algorithm should run in $O(n+m)$ time. Show how to solve this using the Z-algorithm, and also how to solve it using a suffix tree. Can you see a solution using a suffix array along with the LCP array?

Problem 3. For a string S of length n, show how to compute the $L'(i)$ values (used in the Boyer-Moore algorithm) in O(n) time directly from a suffix tree for S. Can you see a solution using a suffix array along with the LCP array?

4. In our discussion of the linear-time algorithm to build a suffix array, we claimed that the suffix array of the string s' determines the suffix array of the set of suffices which start at positions 1 and 2 mod 3 in the original string s. Write a clear and complete explanation of that claim.

5. Give a short sketch of how to construct a suffix tree for a string S from a suffix array for S and the LCP array for S. The algorithm should run in O(n) time.

6. Can you see how to solve the longest common substring problem using a suffix array and the LCP array. I have not thought about this yet, so don't know if this is easy or not doable.