Problem Set 5—Due Friday, Dec. 8

All must be handed in by Monday, Dec. 11, 10AM (solutions out then).

(10) Problem 1. The longest common extension queries is: given a string S and two positions i, j in S, find the length of the longest matching substrings that start at positions i and j respectively.

The setup is that a string S of length n will be specified, and preprocessed in $O(n)$ time. Thereafter, a series of longest common extension queries on S will be specified, i.e. pairs (i, j) will be specified. Each such longest common extension query must be answered in constant time, i.e. independent of n. Explain how this is all done. You may assume an $O(n)$ algorithm to construct a suffix tree.

(20) Problem 2. Problem 36.1 parts $a - c$.

(10) Problem 3. Suppose we are given a TSP problem where cities are points in the plane and distances are actual Euclidean distances (Call this the Euclidean TSP).

Show that the *proof* of theorem 36.15 in the text does **not** prove that the Euclidean TSP is NP-hard (this problem is in fact still NP-hard, but you are not asked to prove this).

(20) Problem 4. A string is a palindrome if it reads the same backwards as forwards (exactly). For example, "abba" and "abbba" are both palindromes. An occurrence of a substring S' of a string S is called a maximal palindrome if S' occurs in S; S' is a palindrome; and at one occurrence of S' in S, the letter before S' is different from the after S'. For example, in xxxabbaxxabbayxx, abba is a maximal palindrome, as is xxabbaxx. Given a string S of length n, give an O(n) time algorithm that finds all the starting positions, and lengths, of all the maximal palindromes in S.