Algorithmics II (H)

Tutorial Exercises on String and Text Algorithms

1. Show that if *S* is a string of length *n* (whose last character appears nowhere else in *S*), then the suffix tree *T* for *S* contains $\leq n$-1 branch nodes.

2. For the string `abracadabra`, draw
   (a) a suffix trie;
   (b) a suffix tree with strings as edge labels;
   (c) a suffix tree with 'short' edge labels.

3. Show that an O(*n*) suffix tree construction algorithm is not possible without the use of short edge labels.

4. Describe an efficient algorithm to find a shortest substring of a given string *S* that appears only once in *S*.

5. A circular string of length *n* can be *linearised* in *n* different ways, namely by 'breaking' the circle between two characters, these two characters then becoming the first and last characters of the linearised string. The *circular string linearisation* problem involves breaking the string to produce the lexicographically smallest possible string (i.e. the one that would appear first if all of these strings were listed in 'dictionary' order). Explain how to carry out circular string linearisation in linear time using a suffix tree.

6. How would you use a regular expression to describe the Unix * symbol – i.e. a symbol that matches any number of arbitrary characters?

7. Draw a NDFA for the regular expression `R = a (b | c)* (d | e)`, where Σ={a,b,c,d,e}. Ensure that each state has at most one outgoing transition labelled by a character of Σ, and at most two outgoing transitions labelled by ε.

8. Simulate the effect of the NDFA in the previous question when executed on the string `abbacbabd`.

9. Suggest an efficient way of implementing the type `Set_Of_States` in the NDFA simulation function.

10. Use dynamic programming to find a longest common subsequence of the strings `abbadcadbc` and `dcbabcacda`.

11. A shortest common supersequence (SCS) of two strings S_1 and S_2 is a string T such that both S_1 and S_2 are subsequences of T, and T is as short as possible. Prove that

$$s = m + n - l$$

where m and n are the lengths of S_1 and S_2 respectively, l is the length of an LCS and s is the length of an SCS of the two strings. Given an LCS, show how to find an SCS, and do this for the strings in Question 10 above.

12. Write a recursive function with memoisation to calculate the n^{th} Fibonacci number a_n defined by

$$a_n = a_{n-1} + a_{n-2} \qquad \text{subject to} \qquad a_0 = a_1 = 1.$$

13. Write a recursive function to calculate the edit distance between two strings X and Y using the recurrence relation for d_{ij}, the distance between the i^{th} prefix of X and the j^{th} prefix of Y, together with memoisation.