58093 String Processing Algorithms

Separate Exam, 5 April 2011 at 16-20 Examiner: Juha Kärkkäinen

Please write on each sheet: your name, student number or identity number, signature, course name, exam date and sheet number. You can answer in English, Finnish or Swedish.

1. [*3+3+3+3 points*] Each of the following pairs of concepts are somehow connected. Describe the main connecting factors or commonalities as well as the main separating factors or differences.

   (a) (Knuth–)Morris–Pratt algorithm and Shift-and algorithm.

   (b) String quicksort and ternary tree.

   (c) String binary search and backward search.

   (d) Compact trie and suffix tree.

   A few lines for each part is sufficient.

2. [*6+6 points*] Let $S[0..n)$ be a string over an alphabet Σ of size σ. A q-gram is a string of length q. Let $occ[0..\sigma^q)$ be an array that for each q-gram Q, in lexicographic order, tells how many times Q occurs in S. For example, if $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, $q = 2$ and $S = \mathtt{acbbbbac}$, then $occ = (0, 0, 2, 1, 3, 0, 0, 1, 0)$. Describe algorithms for computing occ given q and S:

   (a) $\mathcal{O}(n + \sigma^q)$ time algorithm for integer alphabet $[0..\sigma)$.

   (b) $\mathcal{O}(n \log \sigma + \sigma^q)$ time algorithm for an ordered alphabet.

   In both cases, prove the time complexity.

3. [*6+6 points*]

   (a) Compute the edit distance between strings `tukholma` and `stockholm` using the dynamic programming algorithm described on the course.

   (b) Give *all* optimal alignments between `tukholma` and `stockholm`, i.e., alignments with the same cost as the edit distance.

4. [*6+6 points*] Let A, B, B' and C be strings such that $A \le B \le C$ and $A \le B' \le C$.

   (a) Prove that $lcp(B, B') \ge lcp(A, C)$. You may assume only basic definitions from the course to be known, i.e., do not use any lemmas or theorems from the course.

   (b) Describe how the above result can be used to speed up string binary searching.

5. [*12 points*] The reverse of the string $A = a_1 a_2 \ldots a_m$ is the string $A^R = a_m \ldots a_1$. Describe an algorithm that, given two strings S and T, finds the shortest string X such that X occurs in S but neither X nor X^R occurs in T. The time complexity should be linear on a constant size alphabet.