Fundamental Algorithms
CSCI-GA.1170-001/Summer 2016

Homework 10

Problem 1 (CLRS 32.1-2). (1 point) Suppose that all characters in the pattern P are different. Show how to accelerate Naive-String-Matcher to run in time $O(n)$ on an n-character text T.

Problem 2 (CLRS 32.1-4). (2 points) Suppose we allow the pattern P to contain occurrences of a *gap character* $\diamondsuit$ that can match an arbitrary string of characters (even one of zero length). For example, the pattern `ab◊ba◊c` occurs in the text `cabccbacbacab` as $\texttt{c}\underline{\texttt{ab}}\texttt{cc}\underline{\texttt{ba}}\texttt{cba}\underline{\texttt{c}}\texttt{ab}$ and as $\texttt{c}\underline{\texttt{ab}}\texttt{ccbac}\underline{\texttt{bac}}\texttt{ab}$.

Note that the gap character may occur an arbitrary number of times in the pattern but not at all in the text. Give a polynomial-time algorithm to determine whether such a pattern P occurs in a given text T, and analyze the running time of your algorithm.

Problem 3 (CLRS 32.2-1). (1 point) Working modulo $q = 11$, how many spurious hits does the Rabin-Karp matcher encounter in the text $T = 3141592653589793$ when looking for the pattern $P = 26$?

Problem 4 (CLRS 32.3-1). (1 point) Construct the string-matching automaton for the pattern $P = aabab$ and illustrate its operation on the text string $T = aaababaabaababaab$.

Problem 5 (CLRS 32.3-3). (1 point) We call a pattern P *nonoverlappable* if $P_k \sqsupset P_q$ implies $k = 0$ or $k = q$. Describe the state-transition diagram of the string-matching automaton for a nonoverlappable pattern.

Problem 6 (CLRS 32.4-1). (1 point) Compute the prefix function π for the pattern *ababbabbabbababbabb*.

Problem 7 (CLRS 32.4-7). (1 point) Give a linear-time algorithm to determine whether a text T is a cyclic rotation of another string T'. For example, `arc` and `car` are cyclic rotations of each other.

Problem 8. (3 points) The *longest palindromic substring* is a maximum-length contiguous substring of a given string that is a palindrome. For example, the longest palindromic substring of `ultramarine` is `ramar`.

Give an efficient algorithm to determine the longest palindromic substring of a given string. Explain the algorithm and illustrate its operation on the string `evenness`.