CSE 2012- Design and Analysis of Algorithms Practice Problem Sheet (String Matching Problem)

Practice makes you Perfect

String Matching Problem

Given a text $T[1, ..., n]$ of length n and a Pattern $P[1, ..., m]$ of lenght $m \leq n$. Elements of P and T are characters drawn from a finite alphabet Σ. P occurs with with shift s in text T if $T[s+1, s+2, ..., s+m] = P[1, 2, ...m]$. If P occurs with shift s , we call s a valid shift. Otherwise, we call s an invalid shift. Task of 'String Matching Problem' is to find all valid shifts with which a given pattern P occurs in T.

1. Let the pattern contains the occurence of empty space character, denoted for the purpose of understanding as $\diamond$. The pattern may look like $ab \diamond ba \diamond c$, which is just *ab ba c*, in turn just *abbac*. Design an algorithm to compute the valid shifts of P . Analyse your algorithm with time-complexity.

2. Robin-Karp Algorithm discussed in the class, computes all the valid shifts of s of P in T. Here P and T are one-dimensional arrays. Extend the Robin-Karp algorithm where the P and T are of two dimensional arrays. In other words, given a $n \times n$ of characters and a pattern of size $m \times m$ $m \leq n$, design an algorithm to identify the occurence of P in T.

3. Given a text $T[1, ..., n]$ of length n and k Patterns $P_1[1, ..., m]$, $P_2[1, ..., m]$, ..., $P_k[1, ..., m]$, modify the Robin-Karp algorithm to compute the occurence of any one of the patterns in T.

4. Let y^i denote the concatenation of a string y with itself i times. For example $(ab)^3$ is *ababab*. A string $x \in \Sigma^*$ is said to have a repetitive factor r if $x = y^r$, for some string $y \in \Sigma^*$, $r > 0$. Let $\rho(x)$ denote the largest r such that x has a repetition factor r. Given $P[1, 2, ..., m]$, design an efficient algorithm to compute the $\rho(P_i)$, $i = 1, 2, ...m$. Here, P_i is the $i - th$ symbol in P. Analyse your running time with time-complexity.

5. String Matching Problem discussed in the class is of one dimension. Propose the 2-dimensional equivalent of the String Matching Problem, called as 2D-String matching Problem. Design an algorithm for the 2D-String Matching Problem. Analyse your running time with time-complexity.

6. Given j texts $T_1[1, ..., n]$, $T_2[1, ..., n]$,... $T_j[1, ..., n]$, of length n and k Patterns $P_1[1, ..., m]$, $P_2[1, ..., m]$, ..., $P_k[1, ..., m]$, modify the Robin-Karp algorithm to compute the occurrence of all the valid shifts of all the patterns in all the texts. Analyse your running time with time-complexity.

7. Given two texts T, T', design a linear-time algorithm to determine whether the T is a cyclic rotation of T'. For example, 'car' is the cyclic rotation of 'arc' since 'car' can be obtained by a cyclic rotation of the symbols in 'arc'. Analyse your running time with time-complexity.

8. Consider the pattern P and the text T. Given the 'Prefix-function Table' of the string PT (π table for the string PT), design an algorithm to compute the valid shifts of P in T. Analyse your running time with time-complexity.