Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH08-320101
Date: 2018-09-11
Due: 2018-09-20

ICS 2018 Problem Sheet #1

Problem 1.1: *boyer moore bad character rule* (2+2+2+2 = 8 points)

Let $\Sigma = \{A, B, C, D\}$ be an alphabet and $t \in \Sigma^*$ be a text of length n and $p \in \Sigma^*$ be a pattern of length m. We are looking for the first occurance of p in t.

Consider the text $t = ABAABDABBABABBCABABA$ and the pattern $p = BABA$.

You are asked to carry out string search. Write down how the search proceeds using the notation used on the slides. Show each alignment (via indentation) and indicate comparisons performed with a capital letter and comparisons not performed with lowercase letters.

a) Execute the naive string search algorithm. Show all alignments and indicate comparisons performed by writing uppercase characters and comparisons skipped by writing lowercase characters. How many alignments are used? How many character comparisons are done?

b) Execute the Boyer-Moore string search algorithm with the bad character rule only. How many alignments are used? How many character comparisons are done?

c) Determine the two-dimensional lookup table for the given pattern p, where each row represents a character of the alphabet and each column an index position for the pattern p (first index position 0).

d) The description of the bad character rule in the slides assumes that there is a two-dimensional lookup table indexed by the character not matching the current character of the pattern and the current position within the pattern. An alternative is to use a one-dimensional lookup table, which stores for every character the last occurance in the pattern. If the character does not exist in p, the lookup table contains -1. Since the lookup table only stores information about the last occurance of a character in p, it will not always produce optimal shifts.

   Write down the one-dimensional lookup table for p and execute the Boyer-Moore string search algorithm using only the bad character rule with this one-dimensional lookup table.

Problem 1.2: *leap year in the Gregorian calendar (haskell)* (1+1 = 2 points)

In the Gregorian calendar, a leap year occurs if (i) the year is a multiple of four and (ii) the year is not divisible by 100 or (iii) the year is divisible by 400. Note that (ii) and (iii) overlap but (iii) takes precedence. Write a Haskell function `isLeapYear` to determine whether a year is a leap year or not.

- Write a `isLeapYear` function using a Boolean expression involving the Boolean operators `&&` (and), `||` (or), and the Boolean function `not`.
- Write a `isLeapYear'` function using guards and without any usage of the Boolean operators `&&` (and), `||` (or), and the Boolean function `not`.

The Haskell function `div` returns how many times the first number can be divided by the second one and the function `mod` returns the remainder of an integer division.

Explain how you have tested your `isLeapYear` and `isLeapYear'` functions.