Übungsblatt 6 for Effiziente Algorithmen (Winter 2025/26)

Hand In: *Until 2025-11-28 18:00, on ILIAS.*

Problem 1

30 points

Die Sequenz der Fibonacci Wörter $(w_i)_{i\in\mathbb{N}_0}$ ist wie folgt rekursiv definiert:

$$\begin{aligned} w_0 &= \texttt{a} \\ w_1 &= \texttt{b} \\ w_n &= w_{n-1} \cdot w_{n-2} \qquad (n \geq 2) \end{aligned}$$

Aufgelöst führt die Rekursion zu den Wörtern $w_2 = \texttt{ba}$, $w_3 = \texttt{bab}$, $w_4 = \texttt{babba}$, etc.

(Die Längen der Wörter $|w_0|, |w_1|, |w_2|, \ldots$ sind *Fibonacci-Zahlen*, sodass $|w_n| = F_{n+1}$, wobei die Fibonacci-Zahlen definiert sind als $F_0 = 0$, $F_1 = 1$, und $F_n = F_{n-1} + F_{n-2}$, für $n \geq 2$.)

- Erstellen Sie die Transitions-Funktion δ des String-Matching Automaten für w_6 und zeichnen Sie den zugehörigen String-Matching Automaten.
- Erstellen Sie das Failure Link Array *fail* und zeichnen Sie den KMP Automaten mit Failure Links für w_6.

Problem 2

30 points

Wenden Sie für das Muster $P = \texttt{dacbdc}$ und den Text $T = \texttt{eacbdecbdcdabbccacbdc}$ den Boyer-Moore Algorithmus an. Geben Sie als Rechenweg analog zur Vorlesung eine Tabelle an, in welcher in jeder Zeile das Muster zum zu vergleichenden Substring von T ausgerichtet ist. Begründen Sie in jedem Schritt, welche Regel angewandt wurde. Geben Sie in jedem Schritt an, wie viele Zeichen miteinander verglichen wurden.

Problem 3

20 + 20 points

Im Folgenden betrachten wir ein Muster P und einen Text T, wobei $|T| = n$, $|P| = m$, und $n \geq m \geq 1$.

a) Zeigen Sie, dass jeder Pattern Matching Algorithmus im Worst Case mindestens $\lfloor n/m \rfloor$ Zeichen betrachten muss.

b) Geben Sie für jedes $m \geq 1$ und jedes $n \geq m$ ein Muster P und einen Text T an, sodass der Boyer-Moore Algorithnus genau $\lfloor n/m \rfloor$ Zeichen betrachtet. Begründen Sie Ihre Lösung.

Problem 4

30 points

Angenommen in einem Muster P können Lückenzeichen τ spezifiziert werden. Ein Lückenzeichen τ kann auf beliebige Teilstrings (auch einen leeren Teilstring) matchen. Zum Beispiel würde das Muster `ab`τ`ba`τ`b` auf den String `cabcdbab` matchen, weil τ in diesem Fall zunächst auf `cd` und dann auf den leeren String matchen kann.

Beschreiben Sie textuell einen Algorithmus, der einen endlichen Automaten erstellt. Der Automat soll das Vorkommen eines Musters $P[0..m)$ mit Lückenzeichen in einem Text T (mit $|T| = n$) in $\mathcal{O}(n)$ finden. Begründen Sie die Korrektheit Ihrer Lösung.

Problem 5

20 + 20 + 10 points

In dieser Aufgabe sollen Sie den Knuth-Morris-Pratt Algorithmus in Java implementieren und dann erweitern. Lösen Sie dazu die folgenden Teilaufgaben.

a) Erstellen Sie eine Methode `boolean kmp(String text, String pattern)`. Implementieren Sie darin den Knuth-Morris-Pratt Algorithmus gemäß der Vorstellung in der Vorlesung. Testen Sie Ihre Implementierung in einer `main`-Methode mittels dem Text `ababbababa` an folgenden Mustern:

   1. `abab`
   2. `aba`
   3. `aab`

b) Vergleichen Sie die Laufzeit Ihrer Implementierung aus a) mit der Substring Suche in Java (`contains`-Methode eines Strings). Können Sie Eingaben konstruieren, bei welchen Sie Vorteile des jeweiligen Algorithmus aufzeigen können? Wenn ja, erstellen Sie ein entsprechendes Experiment. Begründen Sie Ihre Lösung.

c) Erstellen Sie eine Methode

`int[] kmpWithPositions(String text, String pattern)`.

Erweitern Sie darin Ihre Implementierung aus a), um alle Positionen im Text zurückzugeben, welche den Start des Musters kennzeichnen. Wiederholen Sie den Test Ihrer `main`-Methode, um alle Positionen auszugeben.