Algorithmen zur Sequenzanalyse

Wintersemester 2019/2020 — **Übungsblatt 4** — Prof. Dr. E. Ohlebusch,

Besprechung am 6.12.2019 — Institut für Theoretische Informatik

---

Aufgabe 4.1.

Wir betrachten die Menge M aller Teilstrings eines Strings S, die genau einmal in S vorkommen. Gesucht sind alle kürzesten Strings aus M. Geben Sie einen Algorithmus an, der dieses so genannte *shortest unique substring* Problem in linearer Zeit löst. Wählen Sie ein geeignetes Ausgabeformat.

Aufgabe 4.2.

Ein String ω ist ein *prefix tandem repeat* des Strings S wenn gilt:

- ω ist ein Präfix von S
- es gibt einen String u mit $uu = \omega$

Geben Sie einen linearen Algorithmus an, der alle *prefix tandem repeats* von S ausgibt. Wählen Sie ein geeignetes Ausgabeformat.

Aufgabe 4.3.

Sei ω ein längstes Repeat. Zeigen Sie, dass ω ein supermaximales Repeat (und damit auch ein maximales Repeat) sein muss.

Aufgabe 4.4.

1. Entwerfen Sie einen linearen Algorithmus, der den *lcp*-Intervallbaum einmal bottom-up durchläuft und dabei alle lcp-Intervalle aufzählt. Ein lcp-Intervall bestand bislang aus den Komponenten $\langle lcp, lb, rb, childList\rangle$. Statt der Kindliste soll als vierte Komponente hier die Liste der lcp-Indizes (in aufsteigender Reihenfolge) des lcp-Intervalls berechnet werden, d.h. ein lcp-Intervall besteht nun aus den Komponenten $\langle lcp, lb, rb, lcpIndices\rangle$.
2. Erweitern Sie den Algorithmus um ein Array $lastOcc$ der Größe σ, sodass $lastOcc[c]$ den Index speichert, an dem der Buchstabe $c \in \Sigma$ zuletzt in der BWT vorkam. Genauer gesagt soll die for-Schleife der bottom-up Traversierung folgende Invariante haben:
   - Vor jeder Ausführung des Rumpfes der for-Schleife für einen Wert k ($2 \leq k \leq n+1$) enthält $lastOcc[c]$ ($c \in \Sigma$) den Index des letzten Vorkommens von c in $\mathsf{BWT}[1..k-1]$ (wobei $lastOcc[c] = 0$ gelten soll, falls c nicht in $\mathsf{BWT}[1..k-1]$ vorkommt).
3. Implementieren Sie nun die Prozedur $process(lastInterval, lastOcc)$, sodass der Gesamtalgorithmus alle supermaximalen Repeats ausgibt.
4. Analysieren Sie die worst-case Laufzeit des Algorithmus.