Algorithmen zur Sequenzanalyse

Wintersemester 2014/2015 — **Übungsblatt 5** — Prof. Dr. E. Ohlebusch, T. Beller

Besprechung am 16.12.2014 — Institut für Theoretische Informatik

---

Aufgabe 5.1.

Sei S ein String der Länge n ohne Abschlusszeichen \$ und L die letzte Spalte der Matrix, die alle n Rotationen von S in lexikografischer Reihenfolge enthält. Sei SA das Suffix Array von S und der String $\mathsf{BWT}[1..n]$ definiert durch:

$$\mathsf{BWT}[i] = \begin{cases} S[\mathsf{SA}[i]-1] & \text{falls } \mathsf{SA}[i] > 1 \\ S[n] & \text{falls } \mathsf{SA}[i] = 1 \end{cases}$$

Finden Sie einen String S für den gilt $L \neq \mathsf{BWT}$.

Aufgabe 5.2.

Erweitern Sie Algorithmus 15 des Skripts so, dass zusätzlich das Suffix Array des Strings berechnet wird. Ist es möglich, das LF-Array mit dem Suffix Array zu überschreiben um Speicherplatz zu sparen?

Aufgabe 5.3.

Implementieren Sie die Move-To-Front-Transformation. Wenden Sie diese auf die Strings $S = diesisteintestdiesisteintest\$$ und $\mathsf{BWT} = tt\$ttiittddeessiieeeeiissssnn$ an. Vergleichen Sie die mittlere Huffman-Codewortlänge der beiden Move-To-Front-Transformierten. Testen Sie, ob sich in der Praxis ein Array oder eine Liste besser zur Verwaltung der Buchstabenliste eignet.

Aufgabe 5.4.

Algorithmus 1 zeigt eine alternative Transformation. Welche Laufzeit hat diese? Implementieren Sie Algorithmus 1 und vergleichen Sie ihn mit der originalen Move-To-Front-Transformation: Welcher Algorithmus ist schneller? Welche Ausgabe kann der Huffman-Code besser komprimieren?

Algorithm 1 Alternative zur Move-To-Front-Transformation von L.

```
1: function MTF_ALTERNATIV(L)
2:     Initialize an array ord containing for each characters from Σ its order in Σ
3:     R[i] ← ord[L[1]]
4:     for i ← 2 to n do
5:         if ord[L[i − 1]] > ord[L[i]] then
6:             R[i] ← ord[L[i]] + σ − ord[L[i − 1]]
7:         else
8:             R[i] ← ord[L[i]] − ord[L[i − 1]]
```