Algorithmische Grundlagen und Vermittlung der Informatik
Lehrstuhl 11 (Algorithm Engineering)
Prof. Dr. Johannes Fischer

technische universität dortmund

# Text Indexing and Information Retrieval

Übungsblatt 9

Besprechung: 15.12.2014

## Aufgabe 1 (Theorie)

Zeigen Sie die benötigten Datenstrukturen für das Document-Retrieval auf der Textkollektion {`yabbadabbadoo`, `abbabba`, `baaba`, `abba`, `dayaabba`}. Führen Sie auch einige Beispiel-Anfragen mit dem Algorithmus `report` aus dem Skript aus. Nutzen Sie evtl. `http://www.uni-ulm.de/~tschnatt/suffixarrayapplet/suffixarrayapplet.html`.

## Aufgabe 2 (Theorie)

Wir haben in der VL eine Möglichkeit gesehen, wie wir den Kartesischen Baum auf $s$ Knoten in $2s+1$ Bits darstellen können (da die letzten 2 Bits immer gleich waren, reichen eigentlich $2s-1$ Bits). Es gibt aber auch noch andere Möglichkeiten, Bäume mit einer Bitsequenz darzustellen:

a) Führe eine Tiefensuche durch den Baum durch. Wann immer ein Knoten das erste Mal gesehen wird, schreibe eine „(“ in die (anfangs leere) Bitsequenz. Wann immer ein Knoten das letzte Mal gesehen wird, schreibe eine „)“ in die Bitsequenz. Wird nun „(“ als „`1`“ und „“ als „`0`“ interpretiert, erhält man so eine Bitsequenz der Länge $2s$.

b) Führe eine Tiefensuche durch den Baum durch. Wann immer ein Knoten mit $k$ Kindern das erste Mal besucht wird, schreibe die Zahl $k$ in Unärcode (also die Bitfolge $1^k0$) in die Bitsequenz.

c) Führe eine Tiefensuche durch den Baum durch. Wann immer ein Blatt besucht wird, schreibe „`00`“. Wann immer ein Knoten mit genau einem linken Kind das erste Mal besucht wird, schreibe „`10`“. Wann immer ein Knoten mit genau einem rechten Kind das erste Mal besucht wird, schreibe „`01`“. Wann immer ein Knoten mit zwei Kindern das erste Mal besucht wird, schreibe „`11`“.

Eignen sich diese Repräsentationen für die Identifikation der Kartesischen Bäume als Binärzahl? Falls ja, warum? Falls nein, wo liegt das Problem?

## Aufgabe 3 (Theorie)

Entwerfen Sie einen Algorithmus, der für einen Text $T$ der Länge $n$ und ein Muster $P$ der Länge $m$ alle Positionen in $T$ findet, an der $P$ mit maximal $k$ Mismatches vorkommt. Die Laufzeit sollte in $O(nk+m)$ sein. Hinweis: bauen Sie das Suffix-Array für den Text $T\#P$ und nutzen Sie lcp-Anfragen.