Seminar "Implementierung von Stringalgorithmen"

Leiß/Goller, CIS

25.4.2004

2 Implementierung eines Suffix-Tries

Aufgabe 2.1 Wahlweise in C++ oder in SML soll, aufbauend auf der Trie-Implementierung aus Aufabe 1.2, der Algorithmus zur Konstruktion eines Suffix-Tries aus der Vorlesung (vom 27.04., Folie 14 aus der Foliensammlung auf

`http://www.cis.uni-muenchen.de/~leiss/stringverarbeitung-04/trie.ps1`

implementiert werden.

Zusätzlich zu den an den Knoten vorhandenen Kanten werden nun also Suffix-Zeiger benötigt, die dazu dienen, in $O(1)$ Zeit von einem Knoten v_k (der das k-te Suffix von $a_1 \dots a_i$, d.h. den String $a_{i-k+1} \dots a_i$ repräsentiert), zum Knoten v_{k+1} (der das als nächstes einzufügende Suffix $a_{a-k+2} \dots a_i$ repräsentiert) zu gelangen.

Die Suffix-Zeiger werden natürlich erst nach und nach, im Zuge der Konstruktion des Baumes, eingefügt. Sie werden nur als Hilfsstruktur während der Konstruktion benötigt, könnten also am Ende der Konstruktion wieder entfernt werden – sofern die konkrete Implementierung das ohne weiteres zuläßt. Sind beispielsweise die Suffix-Zeiger als echte Pointer in die Datenstruktur des Knoten eingebunden, so wird ihre Entfernung nicht ohne beträchtlichen Aufwand zu realisieren sein (trotzdem ist dies, zumindest für C++, die für den Anfang empfohlene Strategie). Zur weiteren Diskussion dieser Frage wird herzlich ins Forum

`http://www.die-informatiker.net/viewtopic.php?t=1877`

eingeladen.

Aufgabe 2.2 Den vorhandenen Trie-Implementierungen sollen Funktionen hinzugefügt werden, die den Trie traversieren (also z.B. rekursiv alle seine Knoten besuchen) und ihn "ausmessen", indem sie etwa die durchschnittliche Anzahl von Verzweigungen aller Knoten (oder pro Ebene, d.h. jeweils einen eigenen Durchschnittswert für aller Knoten derselben Tiefe), bestimmen – oder andere Daten erheben (z.B. die Verteilung der Buchstaben des Alphabets auf die Knoten oder Ebenen).

Aufgabe 2.3 Eine Funktion soll implementiert werden, die den Trie traversiert und einen Ausdruck auf STDOUT erzeugt, der als Input für das Programm Dot geeignet ist. Auf diese Weise können wir sehr einfach zu Visualisierungen unserer Baumstruktur kommen. Installationsanweisungen und Angaben zum Inputformat finden sich in Kürze auf der Kursseite.