Algorithmen & Datenstrukturen 1

Sommersemester 2022

Prof. Dr. Martin Hoefer
Tim Koglin, Marco Schmalhofer

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

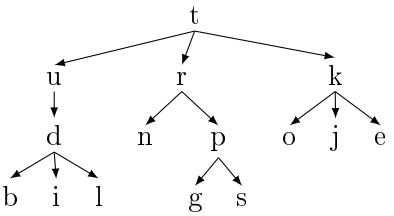
Institut für Informatik
Algorithmen und Komplexität

Übungsblatt 3

Ausgabe: 03.05.2022
Abgabe: 10.05.2022, **08:00**

Aufgabe 3.1 *Baum-Traversierungen* (2 + 2 + 2 + 3 Punkte)

Betrachten Sie den rechts dargestellten geordneten Baum B. Geben Sie an, in welcher Reihenfolge die Knoten in B in einem



a) Präorder-Durchlauf

b) Inorder-Durchlauf

c) Postorder-Durchlauf

besucht werden. Eine Begründung ist jeweils nicht notwendig.

d) Ein *anderer* Baum B' wurde mit Postorder traversiert. Die resultierende Reihenfolge der Besuche ist A, L, G, O, 1. Geben Sie einen möglichen Baum B' an, der diese Reihenfolge produziert. Begründen Sie außerdem, ob B' eindeutig ist oder nicht.

Aufgabe 3.2 *Heaps* (1 + 4 + 4 Punkte)

Wir betrachten das folgende Array von Strings

```
wenn, zwei, sich, streiten, freut, der, dritte, sich
```

a) Hat das Array Heap-Order bezüglich lexikographischer Ordnung?

b) Fügen Sie die Strings der Reihe nach in einen zu Beginn leeren Min-Heap (lexikographisch) ein. Geben Sie den Heap nach jeder `repair_up()` Operation als Array an.

c) Führen Sie auf dem Heap aus Aufgabenteil b) vier mal `delete_min()` aus. Geben Sie den Heap nach jeder `repair_down()` Operation als Array an.

Aufgabe 3.3 *k-näre Heaps* (3 + 4 + 4 + 3 Punkte)

Im Folgenden betrachten wir eine Verallgemeinerung der in der Vorlesung diskutierten binären Heaps. In sogenannten *k-nären* Heaps, wobei $k \geq 3$, gilt weiterhin die Heap-Ordnung. Jedoch ändert sich die Heap-Struktur: In einem k-nären Heap der Tiefe t hat jeder Knoten in Tiefe höchstens $t-2$ genau k Kinder (statt genau zwei Kindern bei binären Heaps). In Tiefe $t-1$ gibt es einen Knoten v mit höchstens k Kindern. Alle Knoten in Tiefe $t-1$ links des Knotens v haben genau k Kinder; alle Knoten in Tiefe $t-1$ rechts von v sind Blätter.

a) Zeigen Sie, dass die Tiefe des zugehörigen Baums mit n Knoten $\Theta(\log_k n)$ beträgt.

b) Geben Sie an,

   i) an welcher Position im Array sich die Kinder von Knoten i für $1 \leq i \leq n$ befinden (sofern sie existieren),

   ii) an welcher Position im Array sich der Elternknoten von Knoten j für $2 \leq j \leq n$ befindet.

c) Passen Sie die Prozedur `Repair_down()` so an, dass sie für k-näre Heaps funktioniert. Beschreiben Sie die geänderte Prozedur in Pseudocode und analysieren Sie die Laufzeit Ihres Verfahrens asymptotisch in k und n.

d) Passen Sie die Prozedur `Repair_up()` so an, dass sie für k-näre Heaps funktioniert. Beschreiben Sie die geänderte Prozedur in Pseudocode und analysieren Sie die Laufzeit Ihres Verfahrens asymptotisch in k und n.

Aufgabe 3.4 *Längste Wege in Bäumen* (8 Punkte)

Gegeben sei ein ungerichteter gewurzelter Baum $B = (V, E)$ in Kind-Geschwister-Darstellung. Die Wurzel sei r.

Entwerfen Sie einen *möglichst effizienten* Algorithmus, welcher die Länge eines längsten einfachen Weges in B berechnet. Geben Sie Ihren Algorithmus in Pseudocode an, analysieren Sie seine Laufzeit und begründen Sie seine Korrektheit.