Algorithmen und
Berechnungskomplexität I
Wintersemester 2017/18

Abgabe: 28.11.2017, 12.00 Uhr
Besprechung: KW 49

UNIVERSITÄT BONN

PD Dr. Elmar Langetepe
David Kübel
Institut für Informatik

Übungsblatt 7

Aufgabe 7.1: (3+3+4 Punkte)

Wir betrachten ein Feld A mit n paarweise verschiedenen Einträgen.

(a) Geben Sie einen Algorithmus an, der ein Feld A mit n Elementen und eine beliebige Zahl $k \in \{1, \ldots, n\}$ als Eingabe erhält und das k-t kleinste Element von A in Linearzeit ermittelt.

Hinweis: Verwenden Sie eine Divide-and-Conquer-Strategie. Sie können davon ausgehen, dass der *Median* eines Feldes A, also das M-t kleinste Element von A mit $M = \lceil n/2 \rceil$, in Linearzeit bestimmt werden kann.

(b) Beweisen Sie die Korrektheit Ihres Algorithmus.

Hinweis: Verwenden Sie Induktion über die Größe des Feldes.

(c) Zeigen Sie, dass die Laufzeit Ihres Algorithmus linear ist. Sie können sich dabei auf den Fall einschränken, dass n eine Zweierpotenz ist.

Aufgabe 7.2: (4 Punkte)

Fügen Sie in einen anfangs leeren AVL-Suchbaum die Schlüssel 4, 5, 7, 2, 1, 3, 6, 0 in der angegebenen Reihenfolge ein. Löschen Sie anschließend die Knoten mit den Schlüsseln 4 und 7 in der gegebenen Reihenfolge. Zeichnen Sie den Baum nach jeder Änderung (einfügen, löschen, rotieren) und geben Sie stets die Balancefaktoren an.

Aufgabe 7.3: (4 Punkte)

Gibt es einen AVL-Baum der Höhe 9 mit 100 bzw. 101 Knoten? Beweisen Sie Ihre Aussage!

Aufgabe 7.4: (6 Punkte)

Eine Bereichsanfrage an einen *AVL*-Baum T besteht aus zwei Zahlen x und y, $x < y$. Es sollen in T alle Zahlen z berichtet werden mit $x \leq z \leq y$. Wir wollen in dieser Aufgabe zwei Möglichkeiten vergleichen, mit einem Blattsuchbaum Bereichsanfragen zu beantworten.

a) In jedem Blatt wird ein Zeiger auf das Blatt mit dem nächstgrößten Schlüssel und ein Zeiger auf das Blatt mit dem nächtkleinsten Schlüssel gespeichert.

- Wie müssen die Einfüge- und Löschoperation modifiziert werden, so dass nach Einfügen oder Entfernen eines Schlüssels die Zeiger der Blätter auf die richtigen Elemente zeigen?
- Wie kann mit diesem modifizierten *AVL*-Baum eine Bereichsanfrage beantwortet werden und in welcher Laufzeit?
- Was ist der Speicherplatzbedarf für den modifizierten *AVL*-Baum?

b) Für einen *AVL*-Baum ohne Modifikation betrachten wir folgenden Algorithmus. Berichte die Schlüssel s_x, s_y der Blätter v_x, v_y, in denen eine Suchanfrage in T nach x bzw. y endet genau dann, wenn $x \leq s_x \leq y$ bzw. $x \leq s_y \leq y$ gilt. Weiterhin, falls $v_x \neq v_y$: Laufe von der Wurzel von T aus die Suchpfade π_x und π_y nach x und y ab, bis diese sich in einem Knoten v aufteilen. Folglich gehört der linke Sohn v_l von v zu π_x und der rechte Sohn v_r zu π_y. Für alle Nachfahren w von v_l auf π_x, deren linker Sohn auf π_x liegt: Traversiere den rechten Teilbaum von w in Inorder-Reihenfolge und berichte die Schlüssel aller Blätter. Für alle Nachfahren w' von v_r auf π_y, deren rechter Sohn zu π_y gehört, berichte in Inorder-Reihenfolge die Schlüssel des linken Teilbaums von w'.

– Begründen Sie, warum obiges Verfahren eine Bereichsanfrage korrekt beantwortet.
– Wie ist die Laufzeit zum Beantworten einer Bereichsanfrage?

Laufzeiten und Speicherplatz können in der O-Notation angegeben werden. Für die Laufzeit- und Speicherplatzbetrachtungen bezeichnet n die Anzahl Schlüssel in T und k die Anzahl berichteter Zahlen.

Aufgabe 7.5: (1+1+2 Zusatzpunkte)

Wir betrachten eine Datenstruktur, die Mengen von Elementen mit geordneten Schlüsseln verwaltet und die folgenden zwei Operationen anbietet:

- INSERT(x): Fügt das Element x unter dem Schlüssel x.key in die Menge ein.
- FINDSMALLEST(k): Gibt das Element x mit dem k-t kleinsten Schlüssel zurück.

Der Einfachheit halber nehmen wir an, dass beim Aufruf von INSERT(x) kein Element mit dem Schlüssel x.key in der Datenstruktur vorhanden ist und dass die Datenstruktur beim Aufruf von FINDSMALLEST(k) mindestens k Elemente enthält.

Geben Sie jeweils eine Datenstruktur an, bei der die Ausführung von INSERT(x) und FINDSMALLEST(k) im Worst Case

(a) in Zeit $O(1)$ und $O(n)$,

(b) in Zeit $O(n)$ und $O(1)$ bzw.

(c) in Zeit $O(\log n)$ und $O(\log n)$

durchgeführt wird, wenn die Datenstruktur zum Zeitpunkt des Aufrufs n Elemente besitzt.