CHRISTIAN-ALBRECHTS-UNIVERSITÄT ZU KIEL
Institut für Informatik, Arbeitsgruppe Algorithmen und Komplexität
Prof. Dr. K. Jansen, K.-M. Klein, F. Land M. Rau

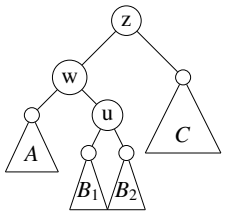
02. Juni 2016

Hausaufgaben zur Vorlesung »Algorithmen und Datenstrukturen«

Blatt 8

Hausaufgabe 8.1 (AVL-Bäume (2 Punkte))
Fügen Sie die Elemente $6,4,2,5,3,1$ in dieser Reihenfolge in einen initial leeren AVL-Baum ein. Entfernen Sie dann die Elemente $6,2,5$ (in dieser Reihenfolge). Geben Sie Zwischenschritte, insbesondere Rotationen, an.

Hausaufgabe 8.2 (AVL-Bäume (4 Punkte))
Zeigen Sie: Entsteht nach dem Anhängen eines neuen Blattes in B_1 oder B_2 eine Höhendifferenz bei z von $+2$, so müssen die Knoten w und u eine Höhendifferenz von 0 gehabt haben. Dabei sei z der erste Knoten auf dem Pfad vom Blatt zur Wurzel mit Höhendifferenz ungleich $+1,-1,0$.



Hausaufgabe 8.3 (HB-Bäume (4 Punkte))
Ein Knoten v_1 in einem Binärbaum T, der den gleichen Vater wie ein zweiter Knoten v_2 hat, wird auch *Bruder* von v_2 genannt. Ein Binärbaum T heißt HB-Baum genau dann, wenn er folgende Eigenschaften besitzt:

(i) Jeder Knoten mit einem Sohn hat einen Bruder mit zwei Söhnen.

(ii) Alle Blätter von T sind in der gleichen Ebene.

Beweisen Sie, dass ein HB-Baum der Höhe h mit n Blättern die Ungleichungskette

$$F_{h+2} \leq n \leq 2^h$$

erfüllt (wobei F_k die k-te Fibonacci-Zahl bezeichnet für ein $k \in \mathbb{N}$).

Programmieraufgabe 8.4 (Verkettete Listen und Bucketsort)
Teil a:
Verkettete Listen sind besoders geeignet, wenn die Elemente der Liste in ihrer Reihenfolge betrachtet und manipuliert werden. Um dieses Vorgehen zu ermöglichen, verwendet man das Konzept von Positionen. Eine Position ist ein Ort, an dem in der Liste ein Element abgespeichert ist, und hat jeweils Vorgänger- und Nachfolger-Positionen. In Java wird dieses Konzept durch das Interface `ListIterator` abgebildet. Implementieren Sie mit Hilfe von doppelter Verkettung eine Datenstruktur, die die Methoden `listIterator(int index)` und `size()` aus dem Interface `List` implementiert.

Teil b:
In der Vorlesung wurde das Verfahren Bucketsort vorgestellt, das n Zahlen im Bereich $0,\ldots,k-1$ in Zeit $O(n+k)$ sortiert. Mithilfe von Hashing kann man das Verfahren so erweitern, dass eine Liste der Länge n, die maximal k verschiedene Elemente enthält, in Zeit $O(n+k\log k)$ sortiert wird. Dazu geht man wie folgt vor:

1. Verwalte die Buckets (Listen) in einem assoziativen Array (auch Hash-Map oder Dictionary genannt), anstatt in einem Array der Länge k.
2. Füge jedes Element dem passenden Bucket in Zeit $O(n)$ hinzu. Falls ein Element zum ersten Mal auftritt, muss dafür ein neuer, leerer Bucket erstellt werden.
3. Sortiere die Buckets in Zeit $O(k\log k)$.
4. Erzeuge die sortierte Ausgabe in Zeit $O(n+k)$, indem die Inhalte der Buckets aneinandergehängt werden.

Für dieses Verfahren muss vorher nicht bekannt sein, wieviele verschiedene oder welche Elemente in der Eingabe vorkommen. Implementieren Sie dieses Verfahren. Die Eingabe enthält Schlüssel-Wert-Paare, die aufsteigend nach dem Schlüssel (das Attribut `key`) sortiert werden müssen.

Abgabe der theoretischen und praktischen Aufgaben Donnerstag, den 09. Juni, bis spätestens 14 Uhr. Die Abgabe der theoretischen Aufgaben erfolgt im Schrein. Die Abgabe der praktischen Aufgaben erfolgt im iLearn.