Westfälische Wilhelms-Universität Münster
Übungen zur Vorlesung „Datenstrukturen und Algorithmen“ im SoSe 2017

Prof. Dr. Klaus Hinrichs
Aaron Scherzinger

Blatt 10

Abgabe im Learnweb bis zum 13.07.2017 um 10 Uhr

Aufgabe 36: (6 Punkte) Betrachten Sie einen binären Suchbaum $\mathcal{B}$, der in jedem seiner insgesamt `n` Knoten einen `int`-Wert gespeichert hat. Sofern ein Knoten keinen rechten bzw. linken Nachfolger hat, wird als entsprechender Nachfolger `null` gespeichert. Betrachten Sie den im Learnweb zur Verfügung gestellten Programmrahmen und implementieren Sie die rekursive `JAVA`-Methode `binSort` in der Datei `Aufgabe36.java`, die folgenden Methodenkopf hat:

```
public int binSort(BinTreeNode nnode, int[] a, int pos)
```

Die Anzahl der Felder im Array `a` ist mindestens so groß wie die Anzahl der Knoten im Baum. Zu Beginn Ihres Algorithmus seien alle Einträge von `a` auf `0` gesetzt. Beachten Sie ferner folgende Anforderungen:

- Ihre Methode soll mit der Wurzel des binären Baumes $\mathcal{B}$, (einer Referenz auf) `a` und mit `pos=0` aufgerufen werden und als Ergebnis die sortierte Folge der Knotenwerte in den Arrayfeldern `a[0], ..., a[n-1]` ablegen.
- Bei jedem rekursiven Aufruf wird der Teilbaum mit Wurzel `node` betrachtet. Der Parameter `pos` gibt die Position im Array `a` an, von der an die in diesem Teilbaum befindlichen Werte im Array `a` abgelegt werden sollen.
- `binSort` soll die Anzahl der im Teilbaum mit Wurzel `node` befindlichen Werte zurückgeben.

Gehen Sie hierbei davon aus, dass Sie den Knotenwert einer Instanz der Klasse `BinTreeNode` mit Hilfe der Methode `public int getValue()` ermitteln können. Der linke bzw. rechte Nachfolger einer Instanz des Klasse `BinTreeNode` ist über `public BinTreeNode getLeftChild()` bzw. `public BinTreeNode getRightChild()` abfragbar. Ihre Methode darf keine globalen Variablen verwenden.
Welchen asymptotischen Zeitaufwand besitzt die von Ihnen realisierte Methode?

Aufgabe 37: ((7+7)+(3+5)=22 Punkte) Bearbeiten Sie die folgenen Aufgaben zu (a, b)-Bäumen.

(a) Es sei $\mathcal{B}$ ein $(3, 6)$-Baum.

(i) Fügen Sie die Elemente

$$20, 40, 60, 80, 99, 50, 70, 30, 10, 22, 23, 24, 25, 26, 27, 90, 88, 42, 44, 48, 79, 81, 84, 86$$

in dieser Reihenfolge nacheinander in $\mathcal{B}$ ein. Zeichnen Sie den Baum jeweils mindestens vor und nach jeder Aufteilung von Knoten im Rahmen einer Einfügeoperation.

(ii) Löschen Sie die Elemente

$$20, 40, 60, 80, 99, 50, 70, 30, 10, 22, 23, 24, 25, 26, 27, 90, 88, 42, 44, 48, 79, 81, 84, 86$$

in dieser Reihenfolge nacheinander aus $\mathcal{B}$. Zeichnen Sie den Baum jeweils mindestens vor und nach dem Zusammenfassen von Knoten im Rahmen einer Löschoperation.

(b) Betrachten Sie einen (a, b)-Baum $\mathcal{B}$, der n Objekte speichert.

(i) Geben Sie detailliert an, wie zu einem gegebenen Objekt aus $\mathcal{B}$ der Nachfolger bzw. der Vorgänger in $\mathcal{B}$ bestimmt werden kann. Ihr Verfahren soll eine Laufzeit von $\mathcal{O}(\log_a n)$ haben.

(ii) Modifizieren Sie die Datenstruktur so, dass die Bestimmung des Nachfolgers bzw. des Vorgängers in $\mathcal{O}(1)$ Zeit möglich ist. Geben Sie Ihre Modifikation an und beschreiben Sie detailliert, wie die Operationen `insert` und `delete` entsprechend abgeändert werden müssen. Auch die modifizierten Fassungen dieser Operationen sollen eine Laufzeit von $\mathcal{O}(\log_a n)$ haben.

Aufgabe 38: (13 Punkte) Betrachten Sie einen anfangs leeren AVL-Baum. Fügen Sie nacheinander die Elemente $4, 3, 2, 1, 8, 7, 6, 5, 0$ und 9 in den Baum ein. Zeichnen Sie nach jeder Einfügeoperation den resultierenden Baum und geben Sie zu jedem Knoten sein Gewicht an. Geben Sie weiterhin an, welche Rebalancierungsoperationen notwendig waren. Löschen Sie aus dem so erzeugten Baum die Elemente $9, 0, 5, 6, 7, 8, 1, 2, 3$ und 4, wobei Sie Ihr Vorgehen in der gleichen Weise dokumentieren.

Aufgabe 39: ((5+5+5+5)+5=25 Punkte)

Bearbeiten Sie die folgenden Aufgaben zu Hashing.

(a) Betrachten Sie eine Hashtabelle mit neun Einträgen, in die ganzzahlige positive Schlüssel n gemäß der Hashfunktion

$$h(n) := n \mod 9$$

eingefügt werden. Zeigen Sie, wie das sequentielle Einfügen der Elemente

$$5, 28, 19, 15, 20, 33, 12, 17$$

in die genannte Hashtabelle erfolgt, wobei Sie Kollisionen nach den im Folgenden genannte Verfahren auflösen. Zeichnen Sie den Zustand der Hashtabelle nach dem Einfügen jedes einzelnen Schlüssel jeweils komplett neu.

(i) Überlaufketten (separate chaining)

(ii) Verschmolzene Ketten (coalesced chaining)

(iii) Offene Adressierung (open addressing) und lineares Sondieren (linear probing)

(iv) Doppeltes Hashing (double hashing) mit zweiter Hashfunktion $g(n) = 1 + (n \mod 8)$

(b) Entwerfen Sie eine möglichst kleine, perfekte Hash-Tabelle und -Funktion zur Speicherung der Zahlen $23, 35, 56, 81, 86$ und 91. Benutzen Sie hierbei keine Fallunterscheidung!

Bonusaufgabe$^{(*)}$:

Aufgabe 40: (20+(2+2+2)=26 Punkte) Bearbeiten Sie die folgenden Aufgaben zum ADT `DICT`:

(a) Realisieren Sie den ADT `DICT` in der in der Vorlesung vorgestellten duplikatfreien Variante in `JAVA` unter Verwendung eines (von Ihnen implementierten) AVL-Baumes. Erweitern Sie hierzu entsprechend die in der Vorlesung besprochenen und im Learnweb bereitgestellten Klassen `BSTNode` und `Dictionary`. Denken Sie an eine aussagekräftige Dokumentation gemäß `JAVADOC` und testen Sie Ihre Implementierung.

(b) Betrachten Sie eine Menge von 2^{32} ganzen Zahlen, die in einer baumbasierten Realisierung des ADT `DICT` gespeichert sind. Nehmen Sie an, dass die Bäume jeweils so implementiert sind, dass jeder Zugriff auf einen Knoten einen Festplattenzugriff erfordert. Geben Sie an, wie viele Zugriffe bei der Suche nach einem bestimmten Element im schlechtesten Fall notwendig werden, wenn die Baumstruktur wie folgt gewählt ist:

(i) Binärer Suchbaum (ii) AVL-Baum (iii) B-Baum mit $b = 255$

$^{(*)}$**Hinweis**: Bei dieser Aufgabe handelt es sich um eine Bonusaufgabe, d.h. Sie können mit dieser Aufgabe zusätzliche Punkte in der Übung erwerben, es ist aber möglich, ohne diese Aufgabe 100% der Übungspunkte zu erreichen. Wir empfehlen allen Teilnehmerinnen und Teilnehmern die Bearbeitung der Aufgabe.