

Aufgabe 8 Programmieren I


Hinweise

- Die Abgabe dieser Übungsaufgaben muss bis spätestens Sonntag, den 2. Januar 2022 um 23:59 Uhr im ISIS-Kurs erfolgt sein. Es gelten die Ihnen bekannten Übungsbedingungen.
- Lösungen zu diesen Aufgaben sind als gezippter Projektordner abzugeben. Eine Anleitung zum Zippen von Projekten finden Sie auf der Seite des ISIS-Kurses. *Bitte benutzen Sie einen Dateinamen der Form VornameNachname.zip.*
- Bitte beachten Sie, dass Abgaben im Rahmen der Übungsleistung für die Zulassung zur Klausur relevant sind. Durch Plagieren verirken Sie sich die Möglichkeit zur Zulassung zur Klausur in diesem Semester.

Erstellen Sie für diese Hausaufgabe ein neues Projekt mit Namen **Aufgabe8**.

Aufgabe 8.1 Generische binäre Bäume (1 Punkt)

Adaptieren Sie die Implementierung für Bäume, sodass alle `int`-Werte nun Objekte einer beliebigen Sub-Klasse von `Number` sein können.

Hinweis Benutzen Sie die `BTree`- und `BTreeNode`-Klassen von  GitHub. Testen Sie Ihre Lösung mit der `main`-Methode aus Abbildung 1.

```
public static void main(String[] args) {  
    BTree<Integer> t = new BTree<>();  
    BTree<Double> td = new BTree<>();  
    BTree<Float> tf = new BTree<>();  
    for (int i = 0; i < 20; i++) {  
        t.insert((int) Math.round(100 * Math.random()));  
        td.insert(100 * Math.random());  
        tf.insert((float) (100 * Math.random()));  
    }  
}
```

Abbildung 1: Test für generische Bäume

Neben allgemeinen (unsortierten) Bäumen und sortierten Bäumen gibt es insbesondere Bäume, in denen alle Pfade geordnet sind; d.h., wenn wir von der Wurzel in Richtung Blätter im Baum absteigen, werden die Werte immer größer. Insbesondere, für jeden Knoten mit zwei Kindern sind die Werte beider Kindknoten größer als der Wert des Knoten selbst.

Bitte wenden!

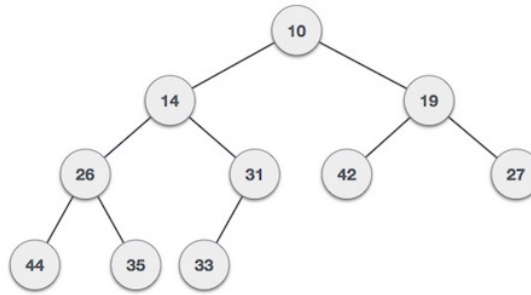


Abbildung 2: Ein Baum, in dem alle Kindknoten größere Werte haben als alle ihre Vorfahren.

Aufgabe 8.2 *Bäume in denen alle Kindknoten größere Werte haben als Ihre Eltern (4 Punkte)*

Implementieren Sie eine Klasse `MinTree`, basierend auf der `BTreeNode`-Klasse. Die `insert`-Funktionalität ist anders als bei Sortierbäumen. Sie muss lediglich sicherstellen, dass der neue Wert “tief genug” im Baum eingeordnet wird und dann größere Elemente weiter nach unten “geschoben” werden.

Implementieren Sie auch eine `takeMin`-Methode, die den Wert des Wurzel-Knotens zurück gibt (und die Wurzel aus dem Baum entfernt). Um das Minimum zu entfernen müssen Sie im wesentlichen einen Pfad im Baum finden, auf dem Sie alle Knoten eine Position nach oben schieben. Neben dem (Standard)-Konstruktor müssen Sie lediglich die `insert`- und `takeMin`-Methode implementieren. Ein Test für die `MinTree`-Klasse ist in Abbildung 3.

Hinweis Importieren Sie keine Pakete, Bibliotheksfunktionen, oder ähnliches (mit Ausnahme von `java.Math.random()`).

Anmerkung Beim “Absteigen” im Baum bzw. dem “Runterschieben” von Elementen ist es zweckmäßig, links oder rechts zufällig zu wählen (um dem Baum ungefähr balanciert zu halten).

```

public static void main(String[] args) {
    MinTree<Integer> t = new MinTree<>();
    final int MAX = 15;

    System.out.print("Die Liste der Zahlen ist: ");
    for (int i = 0; i < MAX; i++) {
        int tmp = (int) Math.round(2*MAX * Math.random());
        System.out.print(tmp + (i < MAX-1 ? ", " : ".\n"));
        t.insert(tmp);
    }

    System.out.print("Die Sortierung ergibt: ");
    for(int j = 0; j < MAX; j++){
        System.out.print(t.takeMin() + (j < MAX-1 ? ", " : ".\n"));
    }

}

```

Die Liste der Zahlen ist: 21, 20, 23, 16, 8, 9, 26, 23, 5, 22, 0, 5, 17, 10, 5.
 Die Sortierung ergibt: 0, 5, 5, 5, 8, 9, 10, 16, 17, 20, 21, 22, 23, 23, 26.

Abbildung 3: Test für die `MinTree`-Klasse

Aufgabe 8.3 *Alle Pfade in einem Baum (2 Punkte)*

Schreiben Sie eine Methode, die für einen Baum eine Menge aller Pfade zurück gibt. Dabei ist ein Pfad eine Liste von Knoten, dessen erstes Element die Wurzel ist und am Ende ein Blatt steht.

Hinweis Benutzen Sie für diese Aufgabe das *Java-Collections Framework*.

```
public static void main(String[] args) {
    BTree<Integer> t = new BTree<>();
    BTree<Double> td = new BTree<>();
    BTree<Float> tf = new BTree<>();
    for (int i = 0; i < 10; i++) {
        t.insert((int) Math.round(100 * Math.random()));
        td.insert(100 * Math.random());
        tf.insert((float) (100 * Math.random()));
    }

    Set<List<BTreeNode<Integer>>> test = t.allPaths();
    System.out.print("Pfade"+test);
}
```

Abbildung 4: Test für die Menge aller Pfade in einem Baum

Aufgabe 8.4 *Balancieren von min-Bäumen (1 Zusatzpunkt)*

Fügen Sie jedem Knoten in einem `MinTree`-Baum eine Ganzzahl hinzu, welche die Differenz der Anzahl der Knoten im linken und rechten Unterbaum angibt. Nutzen Sie diesen Indikator der Differenz, um beim Einfügen eines neuen Elements darauf zu achten, dass die Balance so gut wie möglich gehalten wird, und zwar soll die vorletzte Ebene des Baumes immer voll besetzt sein. Das bedeutet, für alle Bäume der Tiefe $n > 0$, dass die Anzahl der Blätter auf der Tiefe $n - 1$ eine Zweierpotenz ist.¹

¹Wenn der Baum nur die Wurzel ist oder leer, dann haben wir automatisch Balance.