Aufgabenblatt #4

In Ihrer Übungsumgebung unter Linux im Informatik-Labor finden Sie das Projekt Aufgabenblatt 4. Dieses Projekt besteht aus fünf Aufgaben, von denen Sie vier Aufgaben erweitern müssen. Die fünfte wird als Ad-hoc Aufgabe in der Übung gestellt und darf bis dahin nicht verändert werden.

Abgabe: Die Abgabe muss bis Montag, den 15.12. um 15:59 erfolgen. Alle Änderungen am Projekt in der Übungsumgebung, die bis zu diesem Zeitpunkt von Ihnen vorgenommen wurden, werden von uns als Abgabe betrachtet. Verspätete Abgaben werden nicht akzeptiert.

Bitte beachten Sie folgende Punkte:

- Dieses Aufgabenblatt wird benotet.
- Verändern Sie bitte nicht die Ordnerstruktur oder die Dateinamen!
- Ändern Sie nur die Inhalte der benötigten Dateien!
- Die Antworten zu den Zusatzfragen können bei den jeweiligen Aufgaben als Kommentare geschrieben werden. Achten Sie dabei auf korrekte Kommentare!

Aufgabe 1: Sortieren

Implementieren Sie in der gegebenen Klasse `SortedIntArray1` die Methode `sort` nach folgender Vorgehensweise:

- Man beginnt mit dem Sortiervorgang an der Position i=1 und vergleicht in jedem Schritt den Wert an dieser Position mit seinem Vorgänger (i und i-1)
- Sind die zwei Werte in der richtigen Reihenfolge aufsteigend sortiert, dann wird die Position um eins erhöht.
- Sind die zwei Werte nicht in der richtigen Reihenfolge, dann werden sie vertauscht. Die Position wird um eins erniedrigt, falls i>1, ansonsten wird die Position um eins erhöht.
- Der Algorithmus terminiert, wenn man an der letzten Position im Array ankommt und der Wert an dieser Position in der richtigen Reihenfolge ist.

Hinweise: Verändern Sie nicht die Signaturen der Methoden! Ergänzen Sie in diesem Fall nur den Code in der `sort`-Methode.

Zusatzfragen:

- Warum führt dieser Algorithmus zu einer sortierten Sequenz von Zahlen?

- Wie viele Schleifen brauchen Sie für die Implementierung? Lässt sich aus der Anzahl der Schleifen schon auf die Komplexität schließen?
- Mit welchem Sortieralgorithmus aus der Vorlesung lässt sich dieser Sortieralgorithmus vergleichen? Welche Schritte sind hier ähnlich?

Aufgabe 2: Effizientes Sortieren

Implementieren Sie in dieser Aufgabe in der gegebenen Klasse `SortedIntArray2` die `sort`-Methode so, dass sie auch größere Arrays in sinnvoller Zeit (< 1 Sekunde) aufsteigend sortieren kann. Sie müssen den Sortieralgorithmus selbst ausimplementieren und dürfen keinen entsprechenden Aufruf aus der Java-API verwenden. Implementieren Sie weiters die Methode `findSpecialElement` so, dass jenes Element im Array gefunden und zurückgegeben wird, das größer als die n kleinsten Elemente im Array ist. n wird dieser Methode als Parameter übergeben.

Hinweise:

- Verändern Sie nicht die Signaturen der Methoden! Ergänzen Sie in diesem Fall nur den Rumpf der `sort`-Methode und der `findSpecialElement`-Methode.
- Sie dürfen auch zusätzliche Methoden implementieren.

Zusatzfragen:

- Welche API-Aufrufe bietet Java für das Sortieren von Arrays an?
- Warum sind diese Methoden statisch implementiert?
- Welcher Sortieralgorithmus wird in der Java (1.7) für das Sortieren von Arrays verwendet?

Aufgabe 3: Suchen

Implementieren Sie eine rekursive Variante der binären Suche. Implementieren Sie dazu in der Klasse `SortedIntArray3` die Methode `binarySearch` neu. Die Methode liefert `true` zurück, wenn das Element vorhanden ist und `false`, wenn es nicht vorhanden ist.

Hinweis: Sie dürfen zusätzliche Methoden verwenden!

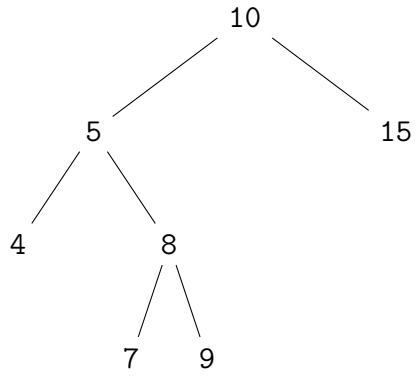
Zusatzfragen:

- Warum muss für die binäre Suche das Array sortiert sein?
- Wie kann man eine binäre Suche in einem absteigend sortierten Array implementieren?
- Was müssen Sie zusätzlich bei der Implementierung beachten, wenn Sie die binäre Suche auf Objekte einer Klasse ausführen?

Aufgabe 4: Baum

Implementieren Sie in dieser Aufgabe in der gegebenen Klasse `IntTree` die folgenden Methoden:

- `countNodes`: Liefert die Anzahl der Knoten im Baum zurück. Wird ohne Parameter auf dem Baum aufgerufen.
- `height`: Liefert die Höhe des Baumes zurück. Der leere Baum hat die Höhe 0. Hat der Baum nur einen Knoten (Wurzel), dann hat er die Höhe 1. Mit jeder zusätzlichen Stufe von Nachfolgern erhöht sich die Höhe um 1. Der nachfolgend abgebildete Baum hat die Höhe 4.



Hinweis: Sie dürfen auch zusätzliche Methoden in der Klasse `Node` implementieren.

Zusatzfragen:

Für die Zusatzfragen gehen Sie von folgenden zusätzlichen Methoden in `IntTree` aus:

```
public void printX() {
    if (this.root != null) {
        this.root.printX();
    }
}

public void printY() {
    if (this.root != null) {
        this.root.printY();
    }
}
```

Weiters gibt es in der Klasse `Node` noch folgende neue Methoden:

```
void printX() {
    if (this.left != null) {
        this.left.printX();
    }
    if (this.right != null) {
        this.right.printX();
    }
    System.out.println(this.elem);
}

void printY() {
    System.out.println(this.elem);
    if (this.left != null) {
        this.left.printY();
    }
    if (this.right != null) {
        this.right.printY();
    }
}
```

Beantworten Sie bitte folgende Fragen:

- Sie fügen in einen IntTree die folgenden Zahlen nacheinander ein: 10, 4, 5, 15, 9, 12, 3. In welcher Reihenfolge werden die Zahlen ausgegeben, wenn Sie `printX` aufrufen. Erklären Sie auch, wie diese Ausgabe entsteht.
- Sie fügen in einen IntTree die folgenden Zahlen nacheinander ein: 15, 5, 4, 10, 9, 12, 3. In welcher Reihenfolge werden die Zahlen ausgegeben, wenn Sie `printY` aufrufen. Erklären Sie auch, wie diese Ausgabe entsteht.

Aufgabe 5: Ad-hoc Aufgabe

Aufgabe 5 wird direkt in den Übungen vorgestellt und bearbeitet.