Übungen aus Algorithmen und Datenstrukturen

Übung 4

Binäre Bäume

Aufgabe 1 – Einfügen in einen binären Baum

Implementieren Sie eine Möglichkeit, einen neuen Datensatz nach dem Breadth-first-Prinzip in einen einfachen binären Baum einfügt!

- Implementieren Sie dazu den Algorithmus
  `protected void breadthFirstAppend(Node newNode)` in einer von `BTree` abgeleiteten Klasse, welche ein neues Node nach dem Breath-first-Prinzip in den Baum einfügt!
- Implementieren Sie weiters in ihrer Klasse die öffentliche Methode
  `public void insert(Object data)`, welche den obigen Algorithmus verwendet, um den neuen Datensatz `data` einzufügen!

Sechs Zusatzpunkte

Wenn ihr es schafft, den Algorithmus `void breadthFirstAppend(Node newNode)` ohne Hilfsdatenstruktur (`Queue`) zu implementieren, gibts sechs Zusatzpunkte!

Richtwert der Implementierung: 28 Zeilen

Aufgabe 2 – Löschen aus einem einfachen binären Baum

Implementieren Sie zwei Algorithmen `protected void remove(Node node)` und `public void remove(Object data)`, welche einen Datensatz aus einem binären Baum löschen.

- Entwickeln Sie eine Lösungsidee und beschreiben Sie diese in Prosa.
- Skizzieren Sie, welche Referenzen an Ihrem Verfahren beteiligt sind und wie diese „verbogen“ werden müssen, um das gewünschte Ergebnis zu erhalten.
- Testen Sie Ihre Algorithmen mit ganzen Zahlen als Datensätze.

Hinweis:

- Die Reihenfolge der Datensätze im Baum muss nicht erhalten bleiben.
- D.h. Sie können die Datensätze beliebig „umreihen“.

Richtwert der Implementierung: 16 Zeilen

Aufgabe 3 – Analyse des `removeLeaf(Node node)`-Algorithmus

Analysieren Sie den Algorithmus `removeLeaf(Node node)` auf Seite 109 des 3. Kapitels. Dieser Algorithmus löscht ein Blatt oder ein Halbblatt aus einem binären Baum.

- Beschreiben Sie die Arbeitsweise des Algorithmus in Prosa und stilisierter Prosa.
- Finden Sie die Schwachstelle in diesem Algorithmus und beschreiben Sie, wie Sie diesen überarbeiten würden um diese Schwachstelle zu beseitigen.