Prof. Dr. E. Ábrahám

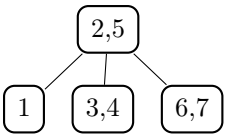
F. Corzilius, S. Schupp, T. Ströder

Allgemeine Hinweise:

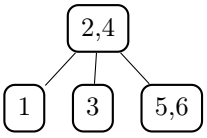
- Die **Hausaufgaben** sollen in Gruppen von je **2 bis 3 Studierenden** aus der **gleichen Kleingruppenübung (Tutorium)** bearbeitet werden. **Namen und Matrikelnummern** der Studierenden sind auf jedes Blatt der Abgabe zu schreiben. **Heften bzw. tackern Sie die Blätter!**
- Die **Nummer der Übungsgruppe** muss **links oben** auf das **erste Blatt** der Abgabe geschrieben werden. Notieren Sie die Gruppennummer gut sichtbar, damit wir besser sortieren können.
- Die Lösungen müssen **bis Montag, den 16.06.2014 um 9:00 Uhr** in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55). Alternativ können Sie die Lösungen auch in Ihrem Tutorium vor der Abgabefrist direkt bei Ihrer Tutorin/Ihrem Tutor abgeben.
- Wenn Sie in einer Aufgabe einen **Algorithmus** erstellen müssen, so soll dieser in **Pseudo-Code** (gerne an `Java` bzw. `C++` orientiert) oder in `Java` bzw. `C++` verfasst werden. Reichen Sie den Algorithmus **nur in Maschinenschrift** ein, entweder Ihrem Tutor vor dem Abgabedatum per Email oder ausgedruckt an die Lösung geheftet. Einfache Syntaxfehler (z. B. ein fehlendes Semikolon) geben keine Punktabzüge. Inwieweit im Pseudo-Code abstrahiert werden darf, wird in der Aufgabenstellung bekannt gegeben. Ansonsten gilt es, sich möglichst nahe an den Möglichkeiten, die `Java` bzw. `C++` bieten, zu orientieren.

Tutoraufgabe 1 (2–3–4–Bäume):

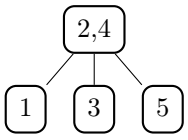
a) Löschen Sie den Wert 3 aus dem folgenden 2–3–4–Baum und geben Sie den dabei entstehenden Baum an:



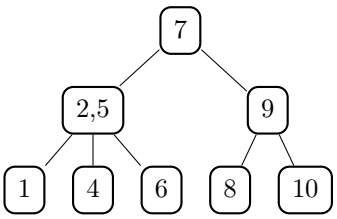
b) Löschen Sie den Wert 3 aus dem folgenden 2–3–4–Baum und geben Sie den dabei entstehenden Baum an:



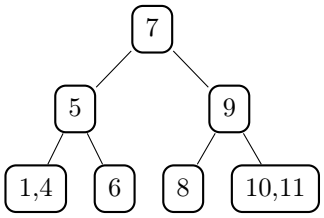
c) Löschen Sie den Wert 4 aus dem folgenden 2–3–4–Baum und geben Sie den dabei entstehenden Baum an:



d) Löschen Sie den Wert 7 aus dem folgenden 2–3–4–Baum und geben Sie den dabei entstehenden Baum an:

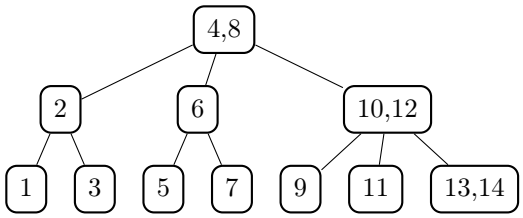


e) Löschen Sie den Wert 9 aus dem folgenden 2–3–4–Baum und geben Sie den dabei entstehenden Baum an:

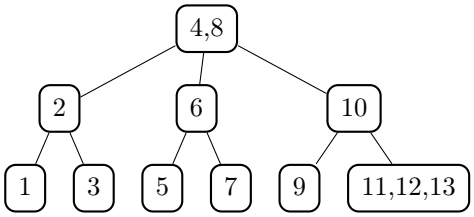


Aufgabe 2 (2–3–4–Bäume): (1 + 1 + 1 = 3 Punkte)

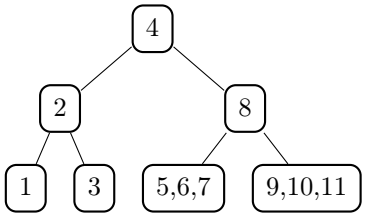
a) Löschen Sie den Wert 13 aus dem folgenden 2–3–4–Baum und geben Sie den dabei entstehenden Baum an:



b) Löschen Sie den Wert 9 aus dem folgenden 2–3–4–Baum und geben Sie den dabei entstehenden Baum an:



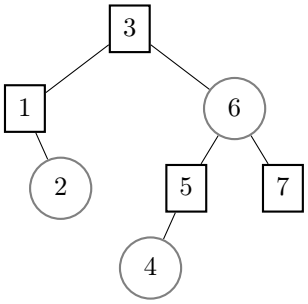
c) Löschen Sie den Wert 4 aus dem folgenden 2–3–4–Baum und geben Sie den dabei entstehenden Baum an:



Tutoraufgabe 3 (Rot–Schwarz–Bäume):

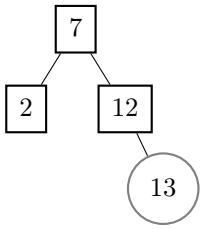
a) Fügen Sie die Schlüssel 1, 3, 5, 6, 6, 4 und 2 in dieser Reihenfolge in einen initial leeren Rot-Schwarz Baum ein. Geben Sie den Baum direkt nach dem Einfügen sowie nach jeder Rotation an.

b) Löschen Sie den Knoten, den die Suche nach dem Schlüssel 6 ausgibt, aus dem folgenden Rot-Schwarz Baum. Geben Sie den Baum direkt nach dem Löschen sowie nach jeder Rotation an.

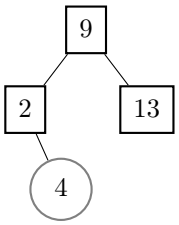


Aufgabe 4 (Rot–Schwarz–Bäume): (1,5 + 1,5 + 1,5 + 1,5 = 6 Punkte)

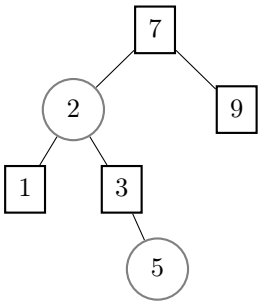
a) Fügen Sie den Schlüssel 13 in den folgenden Rot-Schwarz Baum ein. Geben Sie den Baum direkt nach dem Einfügen sowie nach jeder Rotation an.



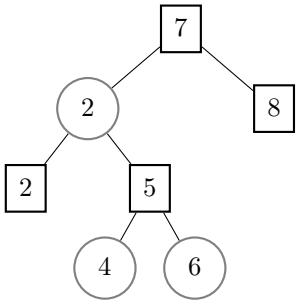
b) Fügen Sie den Schlüssel 3 in den folgenden Rot-Schwarz Baum ein. Geben Sie den Baum direkt nach dem Einfügen sowie nach jeder Rotation an.



c) Löschen Sie den Knoten, den die Suche nach dem Schlüssel 9 ausgibt, aus dem folgenden Rot-Schwarz Baum. Geben Sie den Baum direkt nach dem Löschen sowie nach jeder Rotation an.



d) Löschen Sie den Knoten, den die Suche nach dem Schlüssel 2 ausgibt, aus dem folgenden Rot-Schwarz Baum. Geben Sie den Baum direkt nach dem Löschen sowie nach jeder Rotation an.



Aufgabe 5 (Knobeleien): (10 Punkte)

Gegeben sei die Datenstruktur `234Tree`, welche den Datentyp 2–3–4–Baum implementieren soll.
Ein Objekt vom Typ `234Tree` hat die folgenden Attribute:

- `234Tree[] children`: Das Array, welches die Kinder des Knotens enthält.
- `int[] keys`: Das Array, welches die Schlüssel des Knotens enthält.
- `int filled`: Der aktuelle Füllgrad des Knotens.
- `boolean leaf`: Gibt an, ob der Knoten ein Blatt ist.

Die Folgenden Funktionen sind definiert:

- `234Tree()`: Konstruktor, welcher einen leeren Knoten erzeugt.
- `234Tree(234Tree[] children, int[] keys)`: Konstruktor, welcher einen Knoten mit den Kindern in `children` (in genau der Reihenfolge) und den Schlüsseln `keys` (in genau der Reihenfolge) erzeugt.
  Vorbedingung: children.size() = keys.size() + 1

Zusätzlich können Sie folgende Funktionen nutzen:

- Die Funktion `void insertAt(Element[] array, Element key, int pos)` fügt einen Schlüssel `key` vom Typ `Element` an der Position `pos` in das Array `array` ein. Dabei werden nachfolgende Elemente nach hinten verschoben.
- Die Funktion `int findKeyPos(int[] array, int key, int startpos, int endpos)` findet in dem Array `array` die Einfügeposition (Rückgabewert) für den Schlüssel `key` zwischen den Array–Indizes `startpos` und `endpos`.

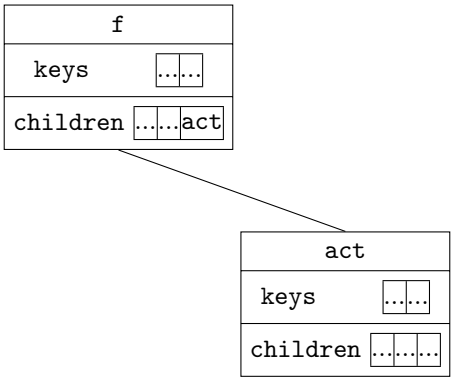
a) Implementieren Sie die Funktion `boolean split(234Tree father, int fatherIndex)`, welche **einen Split** eines Knotens wie aus der Vorlesung durchführt. Sie dürfen dabei davon ausgehen, dass der aktuelle Knoten den maximalen Füllgrad hat. Der Rückgabewert gibt dabei an, ob in der Wurzel (`true`) oder im Innern des Baumes (`false`) gesplittet wurde. Implementieren Sie die Funktion so, dass nach einem Split an der Wurzel die neue Wurzel der aktuelle Knoten ist, bei einem Split im innern des Baumes soll der aktuelle Knoten die Wurzel des neuen linken Teilbaums werden.

b) Implementieren Sie die Funktion `void insert(int key, 234Tree father, int fatherIndex)`, welche das Einfügen des Schlüssels `key` in einen 2–3–4–Baum–Knoten mit dem Vaterknoten `father` durchführen soll. `fatherIndex` ist dabei die Position des Verweises auf den aktuellen Knoten im `children`–Array des Vaterknotens.
Ein Einfügen in den gesamten 2–3–4–Baum findet also durch den Aufruf `insert(key, null, 0)` statt. Sie sollen dabei die Funktion `boolean split(234Tree father, int fatherIndex)` aus der vorherigen Aufgabe benutzen und wie in der Vorlesung sämtliche Knoten auf dem Weg zum Einfügepunkt splitten.

Hinweise:

- *Da bei einem Split innerhalb des Baumes nicht klar ist, welcher Knoten nach dem Split der aktuelle ist, nutzen Sie den Vaterknoten, um rekursiv weiter abzusteigen.*
- *Die Parameter* `father` *und* `fatherIndex` sind lediglich für die Methode `boolean split(234Tree father, int fatherIndex)` und den Fall eines Splits im innern des Baumes interessant.

Beispiel (`act` sei der aktuelle Knoten):



`father = f`, `fatherIndex = 2`