8. Übung Algorithmen und Datenstrukturen WS 2016/17

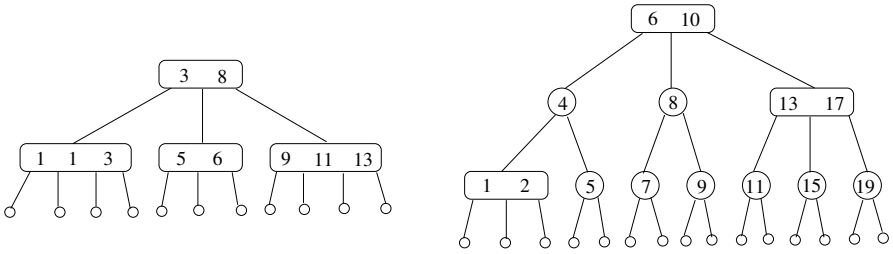
Klaus Kriegel **Abgabe: 03.01.2016, 12:00 Uhr**

Endgültige Version.

Aufgabe 1: (2,4)-Bäume I (3 + 4 Punkte)

a) Fügen Sie in den linken $(2, 4)$-Baum aus der Abbildung nacheinander Einträge mit den Schlüsseln 2 und 10 ein.

b) Löschen Sie aus dem rechten $(2, 4)$-Baum aus der Abbildung nacheinander Einträge mit den Schlüsseln 6, 17 und 9.
Verwenden Sie dabei immer die in der Vorlesung besprochenen Methoden und machen Sie die einzelnen Teilschritte (Position finden, Einfüng, Löschen und eventuelle Splits, Transfers ober Fusionen) deutlich.



Aufgabe 2: (2,4)-Bäume II (3 + 3 + 4 + 2 Punkte + 4 Zusatzpunkte)

In dieser Aufgabe sollen einige Methoden für (2,4)-Bäume implementiert werden. Zur Vereinfachung werden die Inhalte der Einträge vernachlässigt und der primitive Datentyp `int` für die Schüssel vereinbart. Wir verzichten auch auf das Geheimhaltungsprinzip, d.h. alle Attribute und Methoden sind `public`, so dass keine get- und set-Methoden implementiert werden müssen.

a) Mit der Klasse `MWayNode` werden die Knoten von (2,4)-Bäume beschrieben. Zentrale Bestandteile sind ein Array `A` vom Typ `int[]` und Attribute `MWayNode parent` sowie `LinkedLIst<MWayNode> children` für den Elternknoten und die Liste der Kinder. Wir vereinbaren, dass das Array `A` die Länge 5 hat, wobei `A[0]` die Anzahl der Kinder beinhaltet und `A[i]` für alle $i \geq 1$ den i-ten Schlüssel des Knotens hält (`A[4]` wird nur temporär beim Einfügen mit Überlauf verwendet). Implementieren Sie einen Default-Konstruktor für Blätter und drei Konstruktoren, die für ein, zwei oder drei `int`-Parameter Knoten mit diesen Schlüsseln und Blättern als Kinder erzeugen. Jedes Objekt vom Typ `MWayNode` kann zusammen mit allen seinen Nachfahren als ein Baum gesehen werden.

b) Konstruieren Sie im Testprogramm einen (2,4)-Baum der Höhe 2 (Wurzel und deren Kinder sind innere Knoten) mit mindestens 6 Blättern. Implementieren Sie zur Ausgabe eine `toString()`-Methode welche für einen Knoten v folgende Repräsentation erzeugt:

$rep(v) := ()$ wenn v Blatt ist.
$rep(v) := (k_1, \ldots, k_{d-1})[rep(v_1), \ldots, rep(v_d)]$ wenn v ein d-Knoten mit den Schlüsseln $k_1, \ldots, k_{d-1}$ und den Kindern $v_1, \ldots, v_d$ ist.

c) Implementieren Sie eine Methode `boolean check_2_4_Baum()`, die für einen Knoten testet, ob er mit seinen Nachfahren wirklich einen (2,4)-Bäum bildet. Varieren Sie dazu Ihr Testbeispiel so, dass verschiedene Fehler auftreten.

d) Implementieren Sie die Suche nach einem Blatt b als Einfügeposition für einen Schlüssel k sowie das Einfügen von k in den Elternknoten von b (hier darf zunächst ein Überlauf auftreten).

e) **Freiwillige Zusatzaufgabe:** Implementieren Sie die Reparatur eines Überlaufs durch Split-Operationen.