technische universität dortmund

Fakultät für Informatik
Lehrstuhl 11 / Algorithm Engineering

Petra Mutzel
Nicola Beume, Christian Bockermann, Christian Horoba,
Ingo Schulz, Dirk Sudholt, Christine Zarges

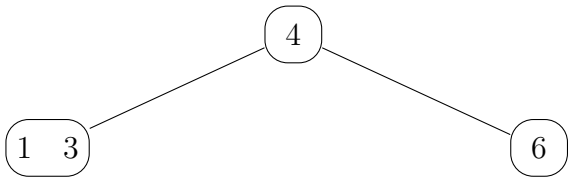
Sommersemester 2009

DAP2 Übung – Blatt 8

Ausgabe: 4. Juni, **Abgabe:** **10. Juni, 16:00 Uhr**, **Block:** C

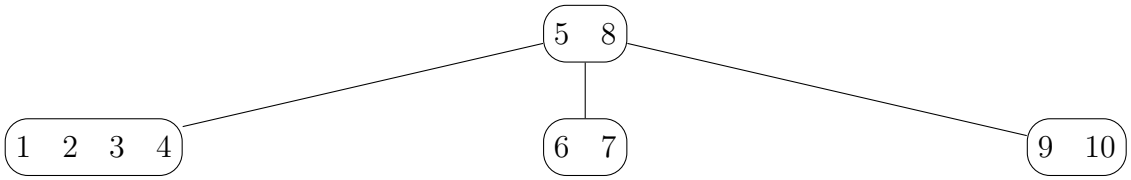
Aufgabe 8.1 (4 Punkte)

a) Betrachte den folgenden B-Baum der Ordnung 3 (2-3-Baum).



Füge nacheinander die Schlüssel 5, 2, 7 und 8 ein und zeichne die entstehenden Bäume.

b) Betrachte den folgenden B-Baum der Ordnung 5.



Lösche nacheinander die Schlüssel 5, 7, 9 und 1 und zeichne die entstehenden Bäume.

Aufgabe 8.2 (4 Punkte)

Betrachte die Schlüsselmenge $\{1, 3, 5, 6, 7, 12, 15\}$.

a) Gib einen 2-3-Baum mit minimaler Tiefe an, der diese Schlüssel speichert. Baue den Baum schrittweise auf, indem du die Schlüssel in geeigneter Reihenfolge in einen leeren Baum einfügst. Zeichne den Baum nach jeder Einfügeoperation.

b) Gib einen 2-3-Baum mit maximaler Tiefe an, der diese Schlüssel speichert. Baue den Baum schrittweise auf, indem du die Schlüssel in geeigneter Reihenfolge in einen leeren Baum einfügst. Zeichne den Baum nach jeder Einfügeoperation.

Aufgabe 8.3 (4 Punkte)

Betrachte die Prozedur

BTreeToList(**ref** BTreeNode *node*, **ref** SListElement $first$, **ref** SListElement *last*).

Sie erwartet einen Zeiger *node* auf einen B-Baum und einen Zeiger $first$ auf das erste Element und einen Zeiger *last* auf das letzte Element einer einfach verketteten Liste. Sie soll die Schlüssel, die in dem B-Baum enthalten sind, in aufsteigend sortierter Reihenfolge an die übergebene Liste anhängen.

Die Prozedur greift auf die folgenden Datenstrukturen zurück.

```
1: struct BTreeNode
2:     var int k                                   ▷ Die Anzahl der Schlüssel
3:     var KeyType key[1,...,k]                    ▷ Die Schlüssel im Knoten (sortiert)
4:     ref DataType data[1,...,k]                  ▷ Die mit den Schlüsseln verknüpften Daten
5:     ref BTreeNode child[0,...,k]                ▷ Die Kinder des Knotens
6: end struct
```

```
1: struct SListElement
2:     var ValueType value                         ▷ Der Wert im Listenelement
3:     ref SListElement next                       ▷ Der Nachfolger des Listenelementes
4: end struct
```

Implementiere die Prozedur (Pseudocode), wobei *nicht* auf zusätzliche Prozeduren und Funktionen zurückgegriffen werden soll. Wie muss die Prozedur aufgerufen werden, um alle Schlüssel aus einem gegebenen B-Baum in einer Liste zu speichern?

Hinweis: Beachte, dass die lokalen Variablen *node*, $first$ und *last* die Argumente, die der Prozedur übergeben werden, referenzieren (call-by-reference).

Präsenzaufgabe 8.4

Die Prozedur

Split(**ref** TreeNode T, **ref** KeyType a, **ref** TreeNode T_1, **ref** TreeNode T_2)

soll aus einem binären Suchbaum T und einem Schlüssel a zwei binäre Suchbäume T_1 und T_2 konstruieren, wobei T_1 alle Schlüssel x aus T enthält, für die $x \leq a$ gilt, und T_2 alle Schlüssel x aus T enthält, für die $x > a$ gilt. Die Prozedur Split darf den ursprünglichen Baum T verändern.

Beschreibe einen effizienten Algorithmus für Split und bestimme seine Laufzeit.

Die folgende Datenstruktur soll dabei verwendet werden.

```
1: struct TreeNode
2:     var KeyType a                               ▷ Der Schlüssel im Knoten
3:     ref TreeNode left                           ▷ Das linke Kind des Knotens
4:     ref TreeNode right                          ▷ Das rechte Kind des Knotens
5: end struct
```