FernUniversität in Hagen

# Lösungsvorschläge zur Nachklausur „1661 Datenstrukturen I"

## 20.9.2008

## Aufgabe 1

(a)

Es bezeichne $t_i$, $1 \le i \le n$, den Zugriff auf die $i$-te Komponente $t_i$ eines $n$-Tupels $t = (t_1, t_2, \ldots, t_i, \ldots, t_n)$.

**algebra** *landregister*

| | | | | |
|---|---|---|---|---|
| **sorts** | *lot, pair, coord, bool* | | | |
| **ops** | *createLot*: | $pair(coord) \times pair(coord)$ | $\rightarrow$ | *lot* |
| | *empty*: | | $\rightarrow$ | *lot* |
| | *splitX*: | $lot \times coord$ | $\rightarrow$ | *pair(lot)* |
| | *splitY*: | $lot \times coord$ | $\rightarrow$ | *pair(lot)* |
| | *neighboured*: | $lot \times lot$ | $\rightarrow$ | *bool* |
| | *merge*: | $lot \times lot$ | $\rightarrow$ | *lot* |
| | *area*: | *lot* | $\rightarrow$ | *coord* |
| | *perimeter*: | *lot* | $\rightarrow$ | *coord* |
| | *createPair*: | $T \times T$ | $\rightarrow$ | *pair(T)* |
| | *first*: | *pair(T)* | $\rightarrow$ | *T* |
| | *second*: | *pair(T)* | $\rightarrow$ | *T* |

**sets** *coord* $= \mathbb{N}_0$ (es kann auch $\mathbb{Q}$ oder $\mathbb{R}$ verwendet werden)

*bool* $= \{true, false\}$

*lot* $= \{ (x_{\min}, x_{\max}, y_{\min}, y_{\max}) \subset coord \times coord \times coord \times coord \mid x_{\min} < x_{\max}, y_{\min} < y_{\max} \} \cup \{\varnothing\}$

*pair(T)* $= \{ (x, y) \mid x \in T, y \in T \}$, $T$ beliebig

**functions**

$createPair(x, y) = (x, y)$

$first(p) = p_1$

$second(p) = p_2$

$$createLot(x, y) = \begin{cases} \varnothing, \text{ falls } x_1 \ge x_2 \text{ oder } y_1 \ge y_2 \\ (x_1, x_2, y_1, y_2), \text{ sonst} \end{cases}$$

$$area(r) = \begin{cases} 0, \text{ falls } r = \varnothing \\ (r_2 - r_1) \cdot (r_4 - r_3), \text{ sonst} \end{cases}$$

$$perimeter(r) = \begin{cases} 0, \text{ falls } r = \varnothing \\ 2 \cdot ((r_2 - r_1) + (r_4 - r_3)), \text{ sonst} \end{cases}$$

$$empty = \varnothing$$

$$splitX(r,x) = \begin{cases} (r,r), \text{ falls } r=\varnothing \\ (\varnothing,r), \text{ falls } x \le r_1 \\ (r,\varnothing), \text{ falls } x \ge r_2 \\ ((r_1,x,r_3,r_4),(x,r_2,r_3,r_4)), \text{ sonst} \end{cases}$$

$$splitY(r,y) = \begin{cases} (\varnothing,\varnothing), \text{ falls } r=\varnothing \\ (\varnothing,r), \text{ falls } y \le r_3 \\ (r,\varnothing), \text{ falls } y \ge r_4 \\ ((r_1,r_2,r_3,y),(r_1,r_2,y,r_4)), \text{ sonst} \end{cases}$$

$$neighboured(a,b) = \begin{cases} true, \text{ falls } a=\varnothing \text{ oder } b=\varnothing \\ true, \text{ falls } (a_1,a_2)=(b_1,b_2) \wedge ( \quad (a_3=b_3) \vee (a_4=b_4) \\ \qquad \vee (a_4=b_3) \vee (a_3=b_4)) \\ true, \text{ falls } (a_3,a_4)=(b_3,b_4) \wedge ( \quad (a_1=b_1) \vee (a_2=b_2) \\ \qquad \vee (a_2=b_1) \vee (a_1=b_2)) \\ false, \text{ sonst} \end{cases}$$

$$merge(a,b) = \begin{cases} \varnothing, \text{ falls } neighboured(a,b)=false \\ a, \text{ falls } b=\varnothing \\ b, \text{ falls } a=\varnothing \\ (\min(a_1,b_1),\max(a_2,b_2),\min(a_3,b_3),\max(a_4,b_4)), \text{ sonst} \end{cases}$$

(b)

Wir stellen hier einige Axiome exemplarisch vor:

*first(createPair(a,b)) = a*

*second(createPair(a,b)) = b*

*area(empty) = 0*

*area(createLot(a,b,c,d))* ≥ 0

*area(merge(a,b))* ≤ *area(a)* + *area(b)*

*area(first(splitX(a,x)))* + *area(second(splitX(a,x)))* = *area(a)*

*area(first(splitY(a,y)))* + *area(second(splitY(a,y)))* = *area(a)*

*perimeter(empty)* = 0

*perimeter(createLot(a,b,c,d))* ≥ 0

*perimeter(merge(a,b))* ≤ *perimeter(a)* + *perimeter(b)*

*neighboured(first(splitX(a,x)),second(splitX(a,x)))* = *true*

*neighboured(first(splitY(a,y)),second(splitY(a,y)))* = *true*

*neighboured(a,b)* = *neighboured(b,a)*

$\neg$ *neighboured*(a,b) $\Rightarrow$ *merge(a,b) = empty*

*merge(a,b)* $\neq$ *empty* $\Rightarrow$ *neighboured(a,b)*

*merge(empty,empty) = empty*

*merge(empty,a) = a*

*merge(a,b) = merge(b,a)*

*merge(first(splitX(r,x)), second(splitX(r,x))) = r*

*merge(first(splitY(r,y)), second(splitY(r,y))) = r*

$(x \leq x_1) \Rightarrow splitX(createLot(x_1,x_2,y_1,y_2),x) = createPair(\emptyset,createLot(x_1,x_2,y_1,y_2))$

$(x \geq x_2) \Rightarrow splitX(createLot(x_1,x_2,y_1,y_2),x) = createPair(createLot(x_1,x_2,y_1,y_2),\emptyset)$

$(y \leq y_1) \Rightarrow splitY(createLot(x_1,x_2,y_1,y_2),y) = createPair(\emptyset,createLot(x_1,x_2,y_1,y_2))$

$(y \geq y_2) \Rightarrow splitY(createLot(x_1,x_2,y_1,y_2),y) = createPair(createLot(x_1,x_2,y_1,y_2),\emptyset)$

## Aufgabe 2

Bei einer Tabellengröße von $m$=10 besteht der mittlere auszuschneidende Block nur aus einer einzelnen Ziffer. Als mittleres Element wurde hier dasjenige gewählt, welches links der „tatsächlichen Mitte" der Ziffernfolge von $k^2$ liegt. Die Auswahl des rechts davon liegenden Elements ist natürlich genauso richtig. Die Länder werden auf die folgenden Behälternummern abgebildet:

| Land | $k$ | $k^2$ | $h(k)$ | $h_1(k)$ | $h_2(k)$ | $h_3(k)$ | $h_4(k)$ |
|---|---|---|---|---|---|---|---|
| Deutschland | 30 | $9\underline{0}0$ | 0 | | | | |
| Frankreich | 25 | $6\underline{2}5$ | 2 | | | | |
| Griechenland | 34 | $1\underline{1}56$ | 1 | | | | |
| Italien | 30 | $9\underline{0}0$ | 0 | 1 | 4 | | |
| Niederlande | 28 | $7\underline{8}4$ | 8 | | | | |
| Polen | 43 | $1\underline{8}49$ | 8 | 9 | | | |
| Portugal | 49 | $2\underline{4}01$ | 4 | 5 | | | |
| Schweden | 30 | $9\underline{0}0$ | 0 | 1 | 4 | 9 | 6 |
| Spanien | 36 | $1\underline{2}96$ | 2 | 3 | | | |
| Tschechien | 42 | $1\underline{7}64$ | 7 | | | | |

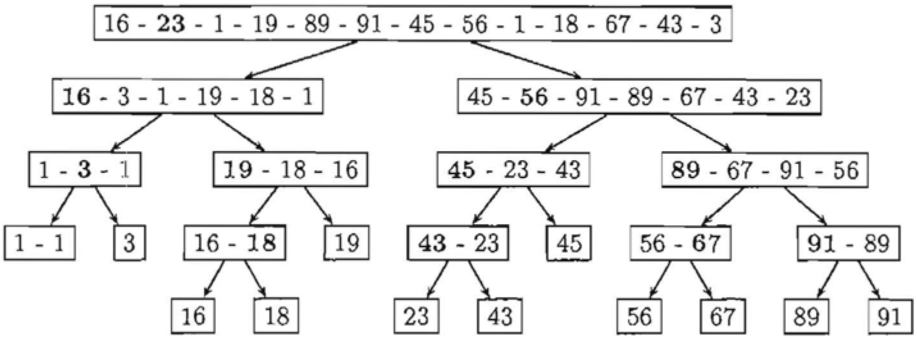
Der Inhalt der Tabelle nach Einfügen aller Länder ist

| | | | |
|---|---|---|---|
| 0 | Deutschland | 5 | Portugal |
| 1 | Griechenland | 6 | Schweden |
| 2 | Frankreich | 7 | Tschechien |
| 3 | Spanien | 8 | Niederlande |
| 4 | Italien | 9 | Polen |

## Aufgabe 3

(a)

Der Aufrufbaum für Quicksort ist der folgende:




(b)

Nach der ersten Phase erhalten wir die folgende Behälterkonstellation:

| $B_0$ | $B_1$ | $B_2$ | $B_3$ | $B_5$ | $B_6$ | $B_7$ | $B_9$ |
|---|---|---|---|---|---|---|---|
| 900 | 321 | 12 | 123 | 555 | 666 | 567 | 29 |
| 200 | 1 | | 63 | | | 187 | |
| | | | | | | 987 | |

Nach der zweiten Phase ergibt sich folgendes Bild:

| $B_0$ | $B_1$ | $B_2$ | $B_5$ | $B_6$ | $B_8$ |
|---|---|---|---|---|---|
| 900 | 12 | 321 | 555 | 63 | 187 |
| 200 | | 123 | | 666 | 987 |
| 1 | | 29 | | 567 | |

Nach der dritten Phase sind die Behälter wie folgt gefüllt:

| $B_0$ | $B_1$ | $B_2$ | $B_3$ | $B_5$ | $B_6$ | $B_9$ |
|---|---|---|---|---|---|---|
| 1 | 123 | 200 | 321 | 555 | 666 | 900 |
| 12 | 187 | | | 567 | | 987 |
| 29 | | | | | | |
| 63 | | | | | | |

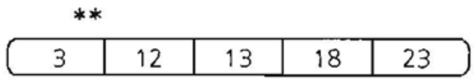
Daraus ergibt sich die sortierte Folge:

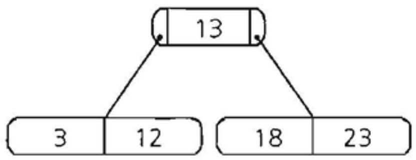
1 - 12 - 29 - 63 - 123 - 187 - 200 - 321 - 555 - 567 - 666 - 900 - 987

## Aufgabe 4

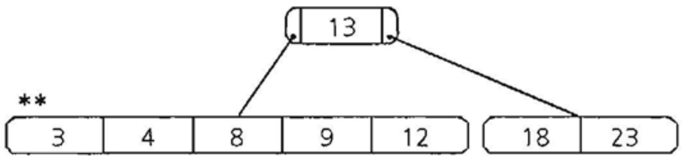
(a)

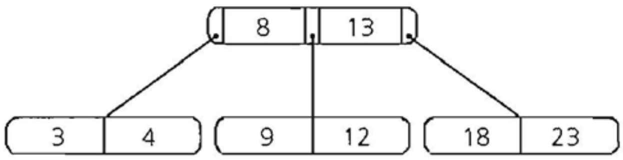
Nach dem Einfügen der Schlüssel 23, 13, 18, 12 und 3 ist die erste Overflow-Behandlung notwendig. Die betroffenen Knoten werden mit ** markiert.



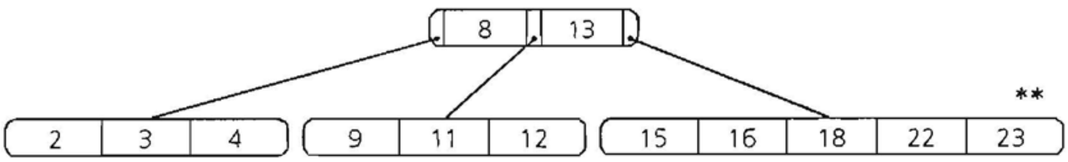



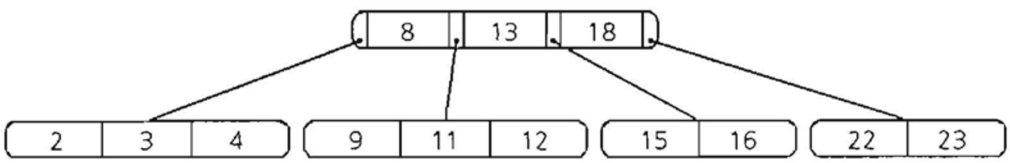

Der nächste Overflow tritt beim Einfügen des Schlüssels 9 auf.



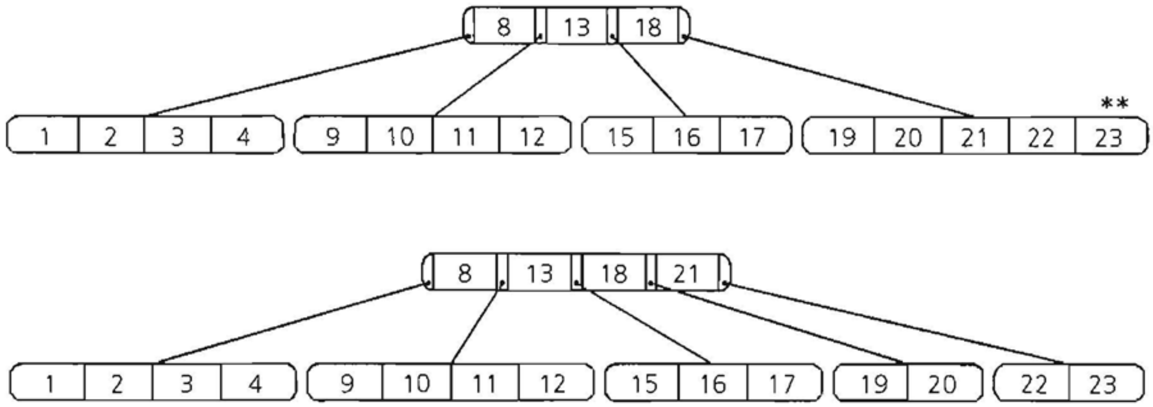




Die nächste Overflow-Behandlung wird mit dem Einfügen des Schlüssels 16 nötig.

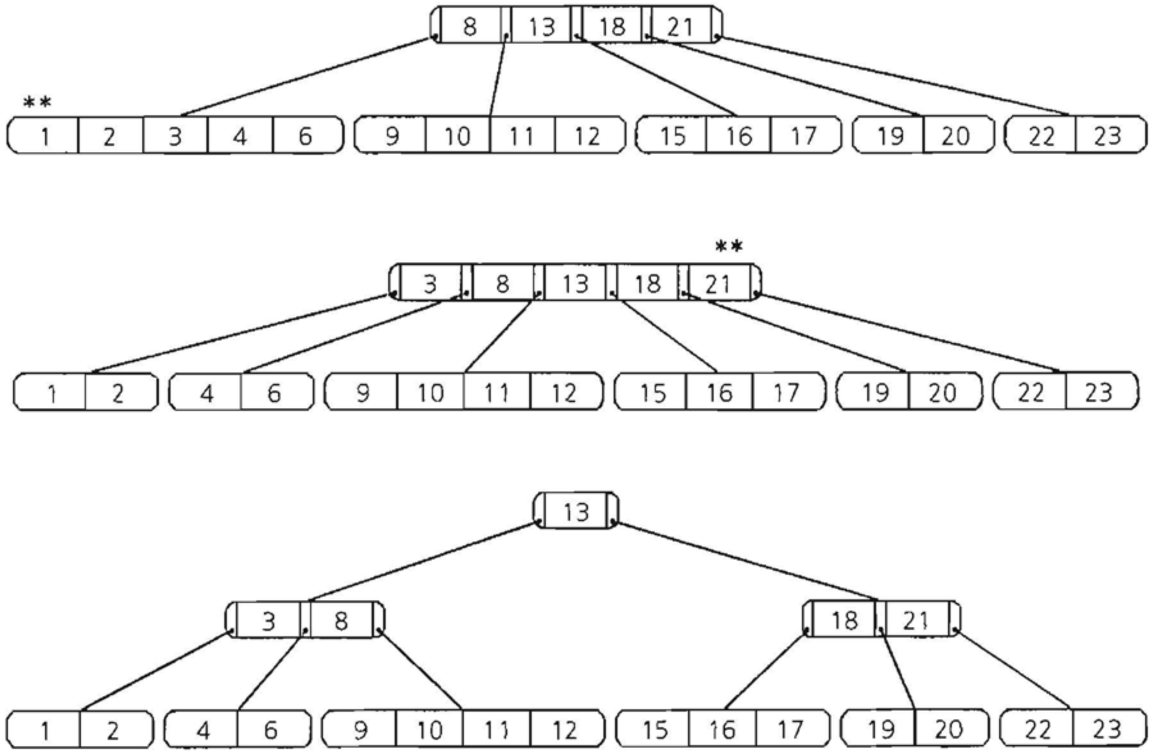


Eine weitere Overflow-Behandlung wird mit dem Einfügen des Schlüssels 20 nötig.

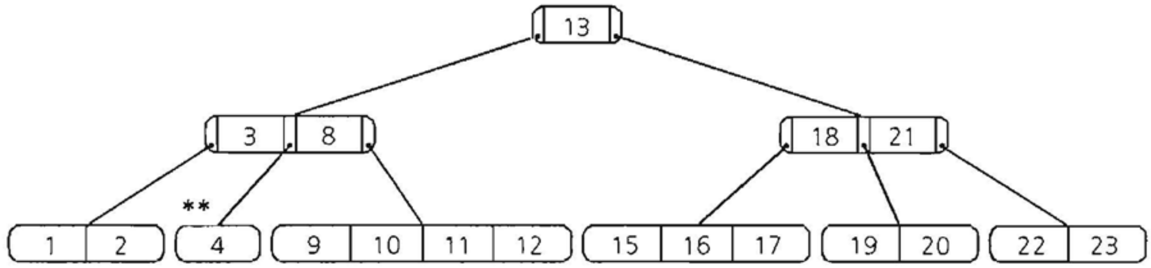


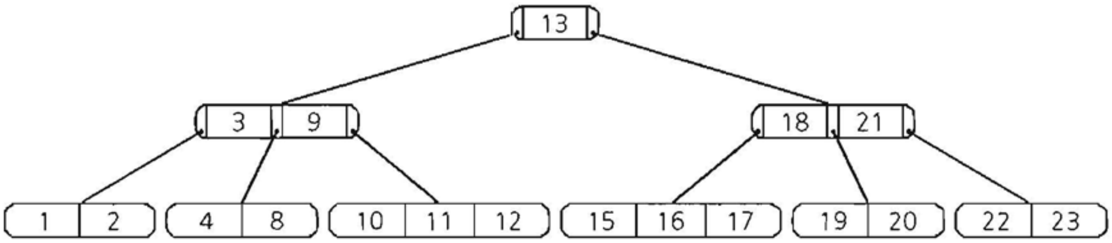

Das Einfügen des letzten Schlüssels 6 führt zu einer fortgesetzten Overflow-Behandlung:



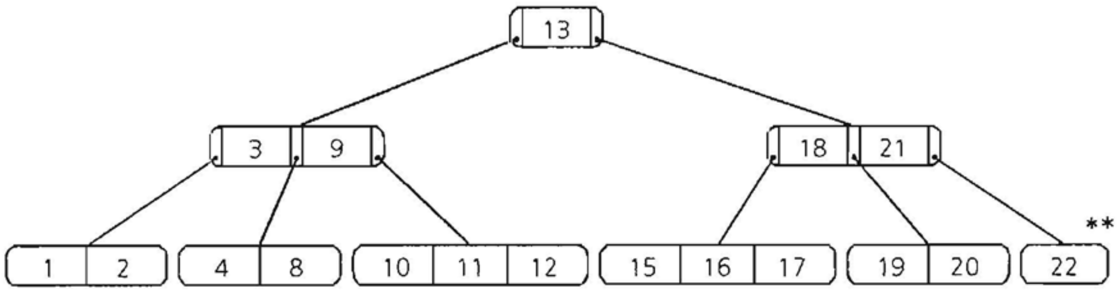
(b)

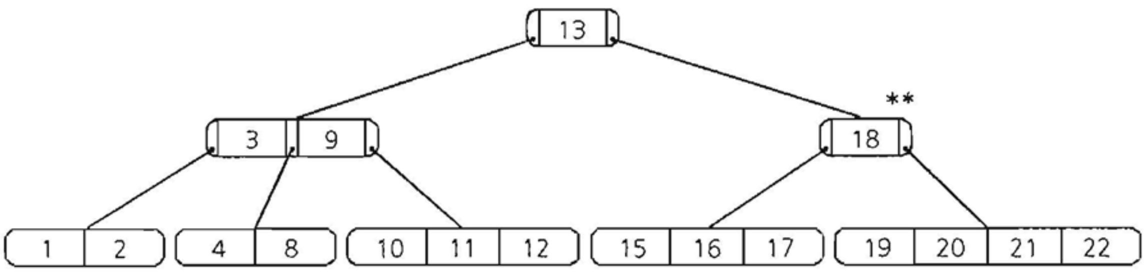
Das Löschen des Schlüssels 6 erzeugt einen Underflow, der durch Balancieren mit dem rechten Nachbarn ausgeglichen werden kann:

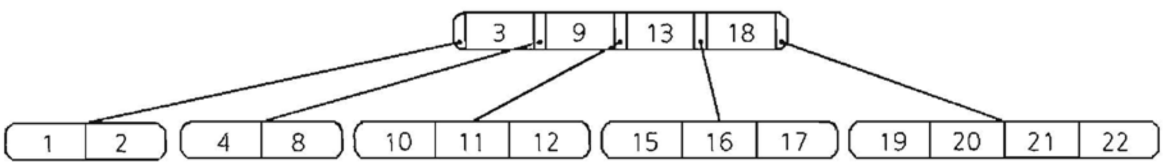



Nun wird der Schlüssel 23 gelöscht. Es entsteht ein Underflow, der nicht durch Balancieren ausgeglichen werden kann. Es erfolgt ein Merge mit dem benachbarten Knoten. In dessen Folge entsteht ein Underflow, der nur durch einen weiteren Merge ausgeglichen werden kann, in den auch die Wurzel einbezogen ist.



Zum Schluss erfolgt nach dem Löschen des Schlüssels 2 ein letzter Merge zur Balancierung des B-Baumes.