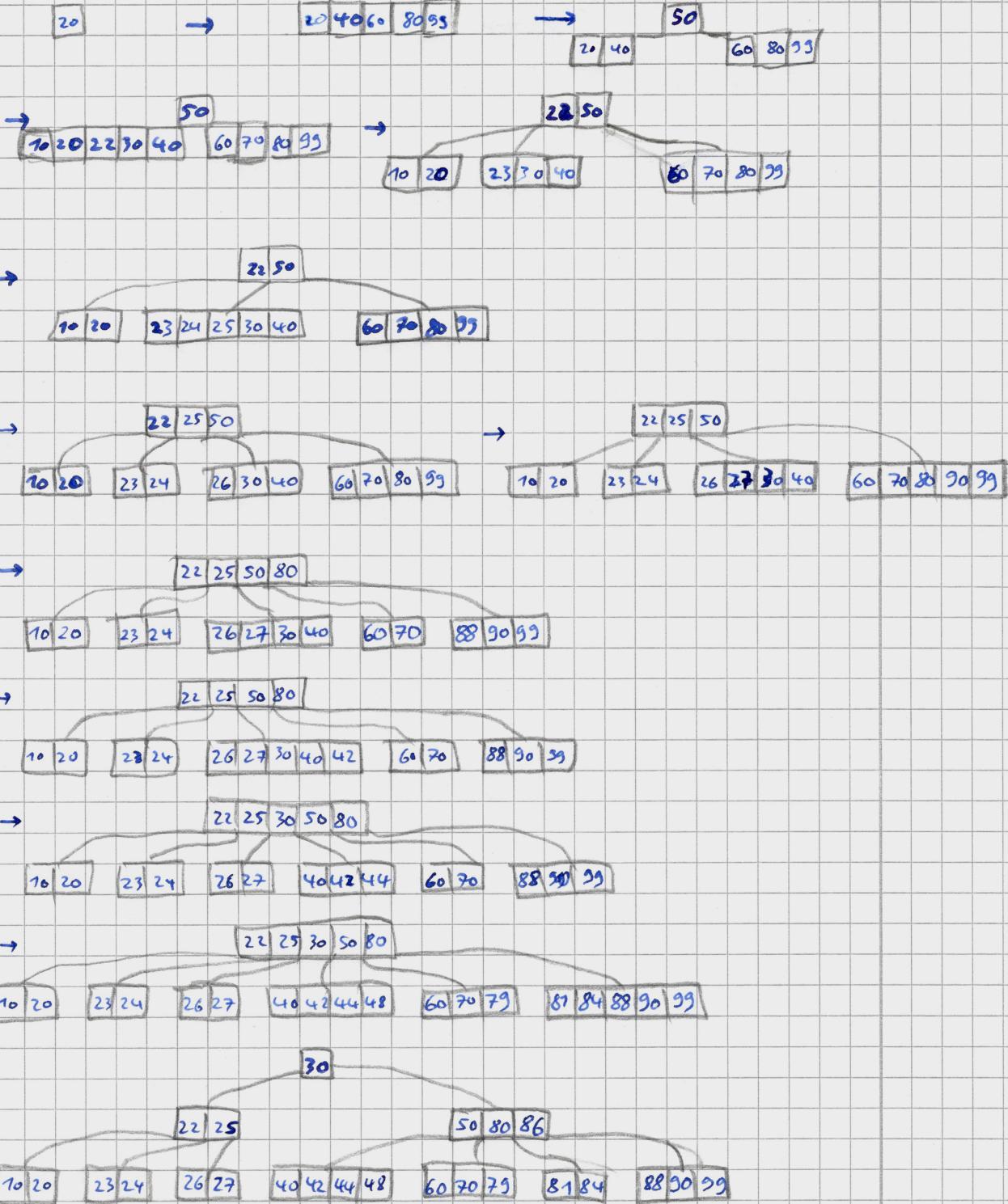


39) VOR: Es sei  $B$  ein  $(3,6)$ -Baum, also  $a=3$  und  $b=6$   
 und  $2 \leq n_B \leq 5$  (an der Wurzel)

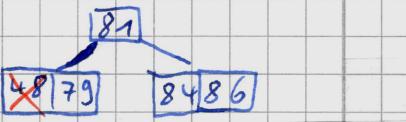
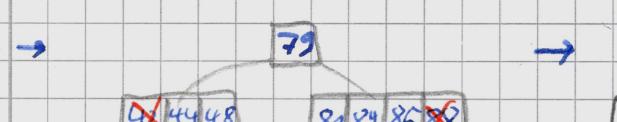
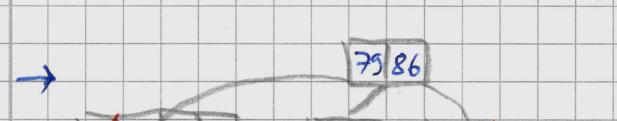
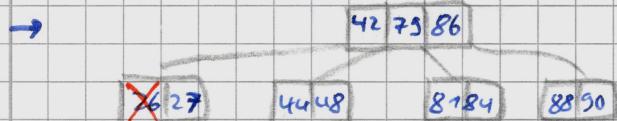
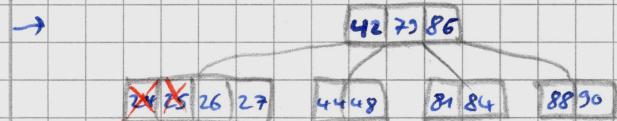
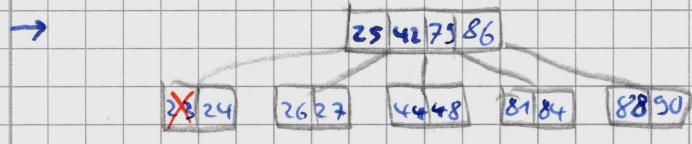
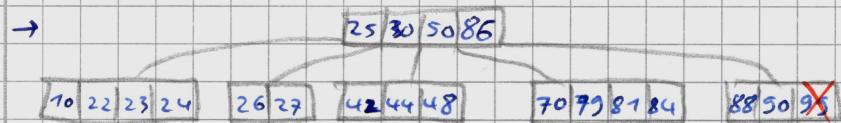
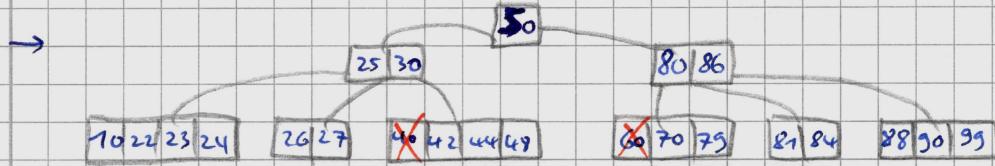
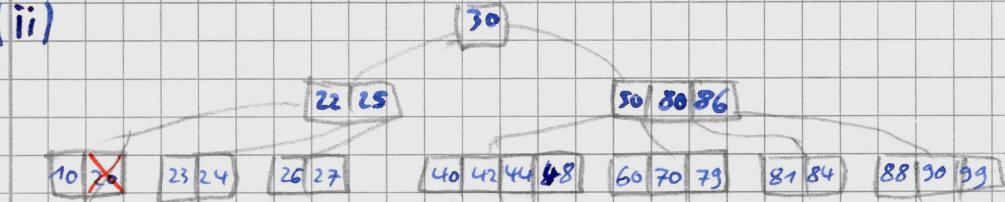
kann = 1

Median kann auch  
 anders gewählt werden

(i)



(ii)



### Aufgabe 39 b)

- (i) Gesucht ist ein Algorithmus zum Bestimmen des Vorgängers bzw. Nachfolgers einer gegebenen, in einem (a,b)-Baum  $\mathcal{B}$  enthaltenen Zahl. Dazu:

Sei  $e \in \mathcal{B}$  das Element, dessen (in diesem Fall) Vorgänger zu bestimmen ist. Dann gehe wie folgt vor:

1. Suche  $e$ , bzw. die Position von  $e$ .
2. Betrachte vom linken Kindknoten das reteste Element (falls vorhanden).
  - 2.1 Gehe solange den rechten Kindknoten, davon das reteste Element, runter, bis kein Kindknoten mehr vorhanden ist.
  - 2.2 Das nun betrachtete Element ist der Vorgänger.
  - 2.3 Terminiere.
3. Falls vorhanden, ist der linke benachbarte Separator der Vorgänger.
  - 3.1 Terminiere.
4. Gehe solange die Elternknoten hoch, bis links von der letzten gegangenen Kante ein Separator existiert oder man in der Wurzel angekommen ist.
  - 4.1 Falls dieser Separator existiert, so ist dieser Separator der Vorgänger.
  - 4.2 Falls man dieser Separator nicht existiert und man in der Wurzel angekommen ist, so existiert kein Vorgänger.
  - 4.3 Terminiere.

Analog (nur umgekehrt) für den Nachfolger.

Zur Laufzeit: Das Suchen des Elementes benötigt  $\mathcal{O}(\log_a n)$  Laufzeit, da die Höhe des Baumes nach oben durch  $\log_a \frac{n+1}{2}$  beschränkt ist. Jeder der drei Zweige (2), (3), (4) benötigt dementsprechend auch  $\mathcal{O}(\log_a n)$  Laufzeit (sieht man sehr schnell anhand der Anzahl der zu gehenden Kanten).

Also gilt insgesamt für diesen Algorithmus:  $T(n) \in \mathcal{O}(\log_a n)$ . □

- (ii) Gesucht ist nun eine modifizierte Struktur des (a,b)-Baums, die die Bestimmung des Vorgängers bzw. Nachfolgers in  $\mathcal{O}(1)$  Zeit erlaubt, Einfüge und Lösch-Operationen aber weiterhin in  $\mathcal{O}(\log n)$  Zeit durchführt.

Dafür erhält jedes Element im (a,b)-Baum einen Zeiger auf seinen Vorgänger und Nachfolger. Dadurch gilt schonmal, dass die Bestimmung von Vorgänger bzw. Nachfolger in  $\mathcal{O}(1)$  Zeit geschieht.

Bleibt zu zeigen, dass einfügen und löschen weiterhin in  $\mathcal{O}(\log n)$  Zeit durchgeführt wird:

- *insert*: Sei  $e \in ELEM$  das einzufügende Element und  $s \in \mathcal{B}$  der Vorgänger bzw.  $S \in \mathcal{B}$  der Nachfolger von  $e$ . Dann gehe wie folgt vor:
  1. Füge das Element, wie aus der Vorlesung bekannt, in  $\mathcal{B}$  ein.
  2. Ermittle nun, wie in (i), den Vorgänger  $s$  bzw. Nachfolger  $S$ .
  3. Setze die Zeiger für Vorgänger bzw. Nachfolger auf  $s$  bzw.  $S$ . Setze die Zeiger für den Nachfolger von  $s$  bzw. den Vorgänger von  $S$  auf  $e$ .

Die ersten beiden Operationen benötigen Laufzeit  $\mathcal{O}(\log n)$ . Also gilt auch insgesamt:  $T(n) \in \mathcal{O}(\log n)$ .

- *delete*: Sei  $e \in \mathcal{B}$  das zu löschen Element und  $s \in \mathcal{B}$  der Vorgänger bzw.  $S \in \mathcal{B}$  der Nachfolger von  $e$ . Dann gehe wie folgt vor:

1. Lösche das Element, wie aus der Vorlesung bekannt.
2. Setze die Zeiger für den Nachfolger von  $s$  auf  $S$ , für den Vorgänger von  $S$  auf  $s$ .

Da die zweite Operation konstanten Zeitaufwand  $\mathcal{O}(1)$  hat, gilt insgesamt:  $T(n) \in \mathcal{O}(\log n)$ .

Somit erfüllt diese Datenstruktur die geforderten Eigenschaften.  $\square$