Script generated by TTT

Title: TÃ¤ubig: GAD (20.06.2013)

Date: Thu Jun 20 12:01:59 CEST 2013

Duration: 45:16 min

Pages: 28

Übersicht

(a,b)-Baum

Problem: Baumstruktur kann zur Liste entarten

Lösung: (a,b)-Baum

Idee:

- $d(v)$: Ausgangsgrad (Anzahl Kinder) von Knoten v
- $t(v)$: Tiefe (in Kanten) von Knoten v
- Form-Invariante:
  alle Blätter in derselben Tiefe: $t(v) = t(w)$ für Blätter v, w
- Grad-Invariante:
  Für alle internen Knoten v (außer Wurzel) gilt:

$$a \le d(v) \le b \qquad \text{(wobei } a \ge 2 \text{ und } b \ge 2a-1\text{)}$$

Für Wurzel r: $2 \le d(r) \le b$ (außer wenn nur 1 Blatt im Baum)

(a,b)-Baum

Lemma

Ein (a,b)-Baum für n Elemente hat Tiefe $\le 1 + \left\lfloor \log_a \frac{n+1}{2} \right\rfloor$.

Beweis.

- Baum hat $n+1$ Blätter ($+1$ wegen ∞-Dummy)
- Für $n = 0$ gilt die Ungleichung (1 Dummy-Blatt, Höhe 1)
- sonst hat Wurzel Grad ≥ 2,
  die anderen inneren Knoten haben Grad $\ge a$

$\Rightarrow$ Bei Tiefe t gibt es $\ge 2a^{t-1}$ Blätter

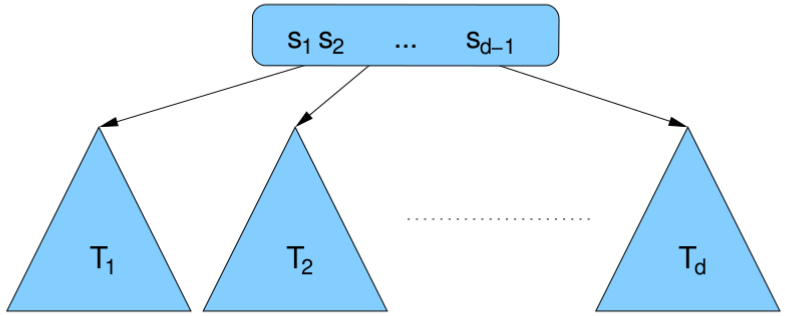
- $n+1 \ge 2a^{t-1} \quad \Leftrightarrow \quad t \le 1 + \left\lfloor \log_a \frac{n+1}{2} \right\rfloor$

$\log_a \frac{n+1}{2} \ge t-1$

□

(a, b)-Baum: Split-Schlüssel

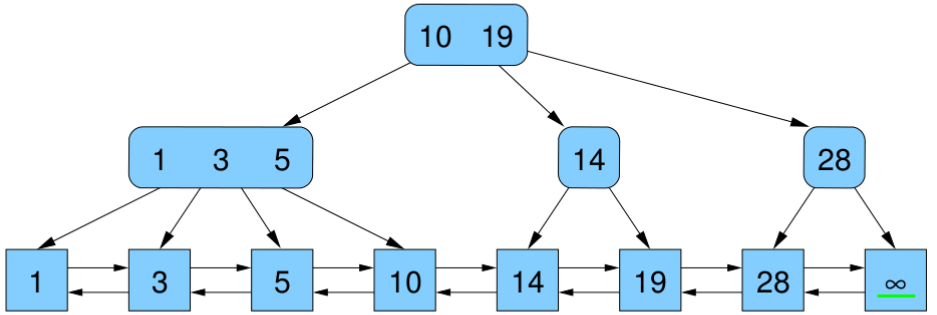
- Jeder Knoten v enthält ein sortiertes Array von $d(v) - 1$ Split-Schlüsseln $s_1, \ldots, s_{d(v)-1}$



- (a, b)-Suchbaum-Regel:
  Für alle Schlüssel k in T_i und k' in T_{i+1} gilt:
  $k \leq s_i < k'$ bzw. $s_{i-1} < k \leq s_i$ $\quad (s_0 = -\infty, s_d = \infty)$

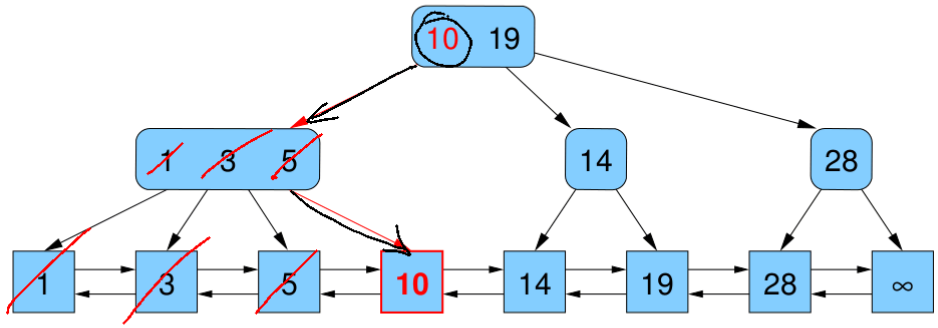
(a, b)-Baum

Beispiel:



(a, b)-Baum

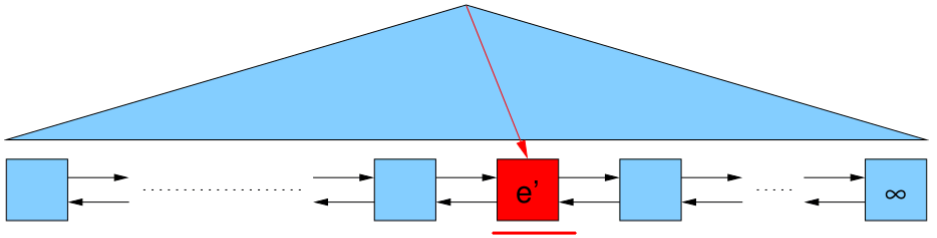
locate(9)



(a, b)-Baum

insert(e)

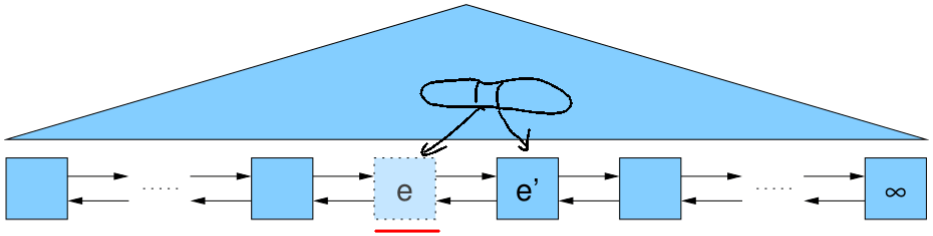
- Abstieg wie bei locate(key(e)) bis Element e' in Liste erreicht
- falls key(e')>key(e), füge e vor e' ein



(a, b)-Baum

insert(e)

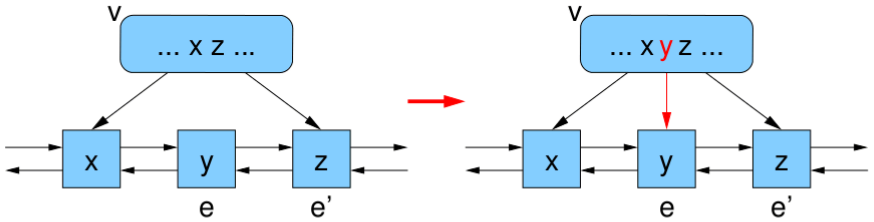
- Abstieg wie bei locate(key(e)) bis Element e′ in Liste erreicht
- falls key(e′)>key(e), füge e vor e′ ein



(a, b)-Baum

insert(e)

- füge key(e) und Handle auf e in Baumknoten v über e ein
- falls $d(v) \leq b$, dann fertig

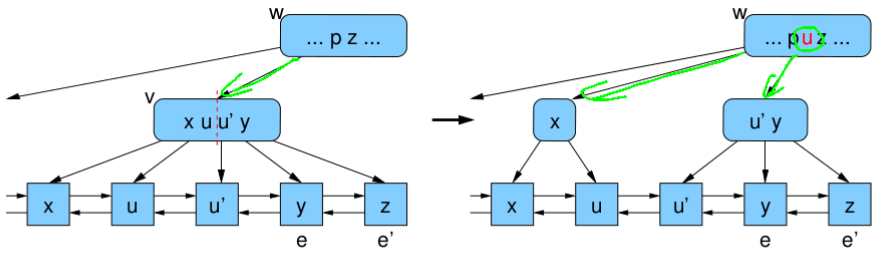


(a, b)-Baum

insert(e)

- füge key(e) und Handle auf e in Baumknoten v über e ein
- falls $d(v) > b$, dann teile v in zwei Knoten auf und
- verschiebe den Splitter (größter Key im linken Teil) in den Vaterknoten

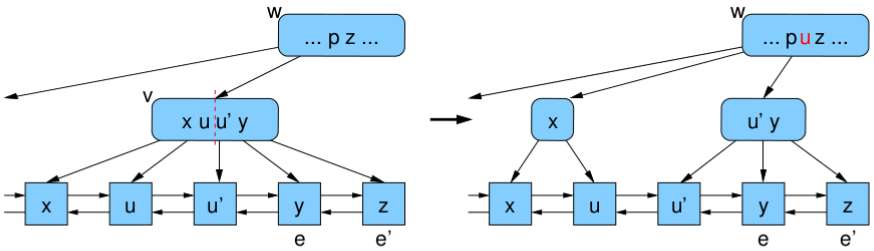
Beispiel: (2, 4)-Baum



(a, b)-Baum

insert(e)

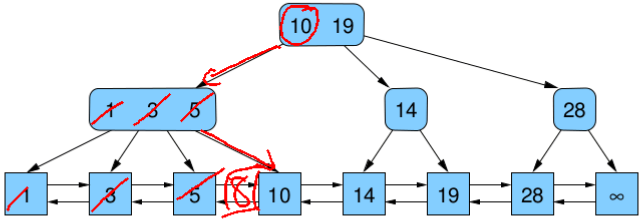
- falls $d(w) > b$, dann teile w in zwei Knoten auf usw.
  bis Grad $\leq b$
  oder Wurzel aufgeteilt wurde



(a, b)-Baum / insert

$a = 2, b = 4$

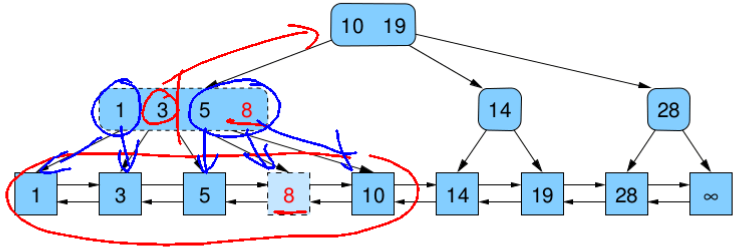
insert(8)



(a, b)-Baum / insert

$a = 2, b = 4$

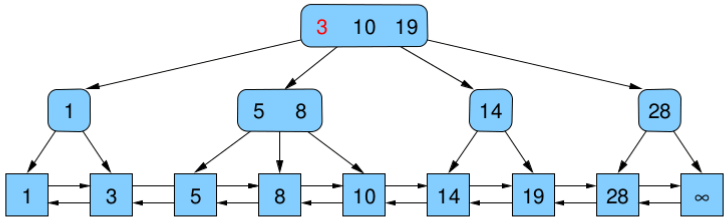
insert(8)



(a, b)-Baum / insert

$a = 2, b = 4$

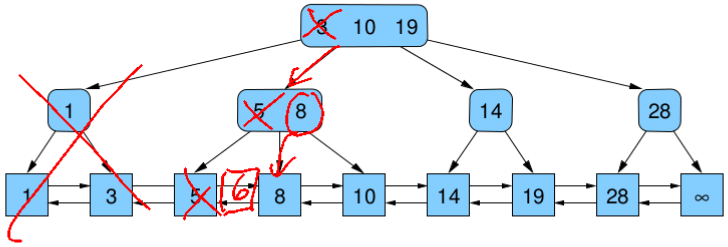
insert(8)



(a, b)-Baum / insert

$a = 2, b = 4$

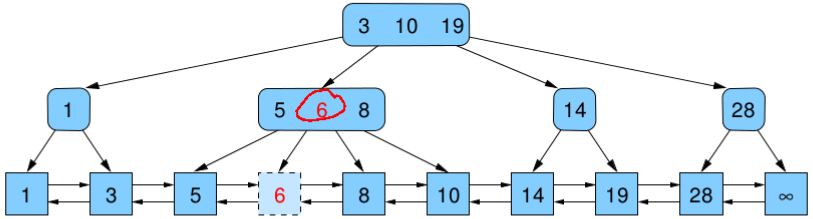
insert(6)



(a, b)-Baum / insert

$a = 2, b = 4$

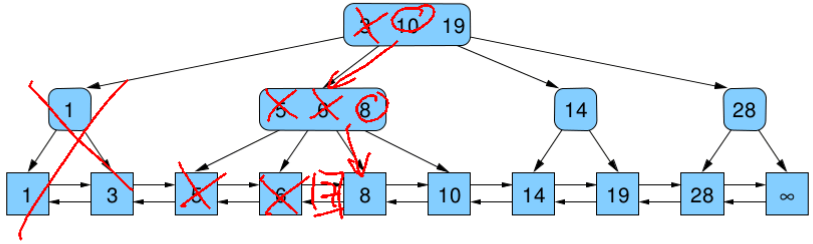
insert(6)



(a, b)-Baum / insert

$a = 2, b = 4$

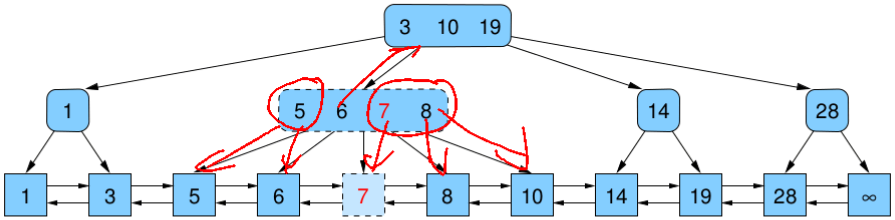
insert(7)



(a, b)-Baum / insert

$a = 2, b = 4$

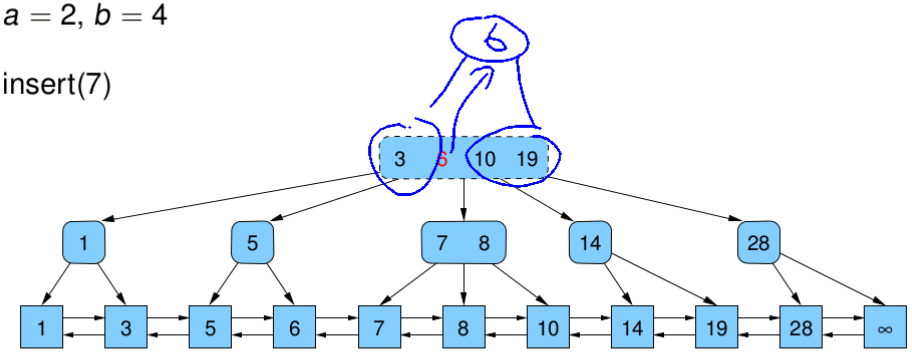
insert(7)



(a, b)-Baum / insert

$a = 2, b = 4$

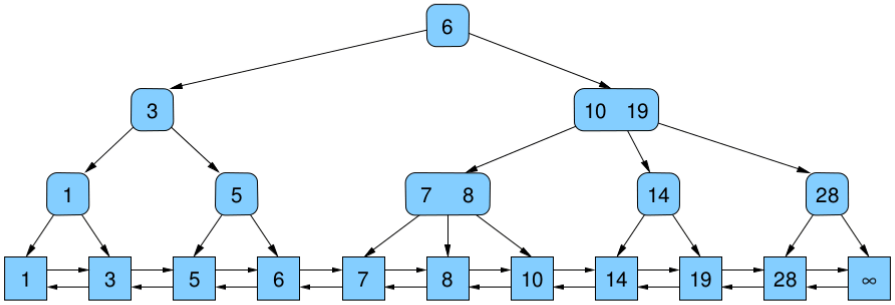
insert(7)



(a, b)-Baum / insert

$a = 2$, $b = 4$

insert(7)



(a, b)-Baum / insert

Form-Invariante

- alle Blätter haben dieselbe Tiefe, denn
  neues Blatt wird auf der Ebene der anderen eingefügt und
  im Fall einer neuen Wurzel erhöht sich die Tiefe aller Blätter um 1

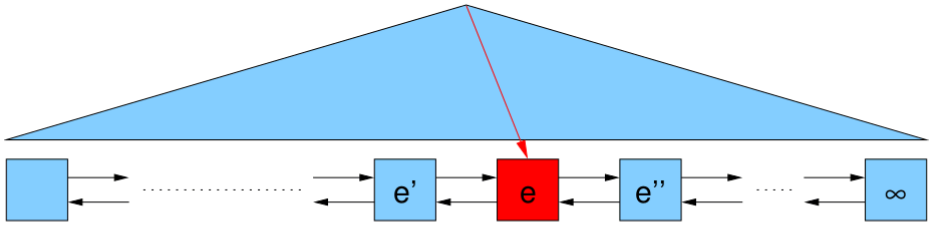
Grad-Invariante

- insert splittet Knoten mit Grad $b + 1$ in zwei Knoten
  mit Grad $\lfloor (b + 1)/2 \rfloor$ und $\lceil (b + 1)/2 \rceil$
- wenn $b \geq 2a - 1$, dann sind beide Werte $\geq a$
- wenn Wurzel Grad $b + 1$ erreicht und gespalten wird,
  wird neue Wurzel mit Grad 2 erzeugt

(a, b)-Baum

remove(k)

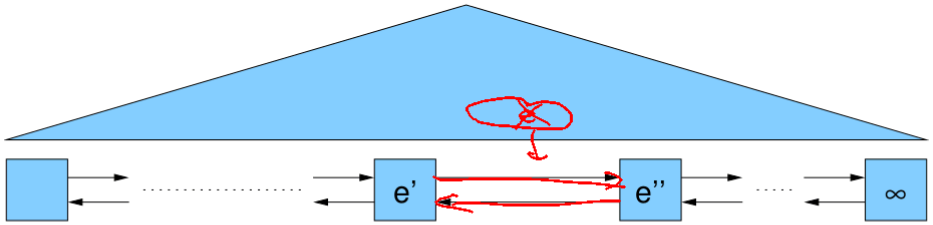
- Abstieg wie bei locate(k) bis Element e in Liste erreicht
- falls key(e) $= k$, entferne e aus Liste (sonst return)



(a, b)-Baum

remove(k)

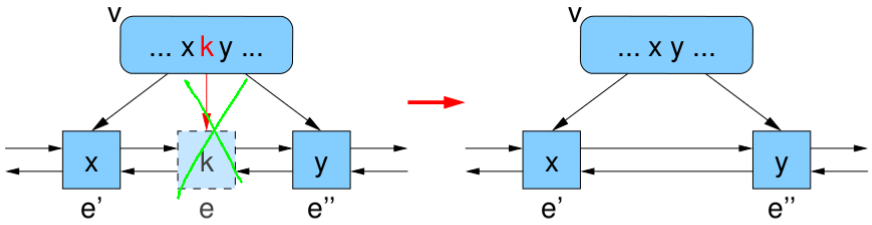
- Abstieg wie bei locate(k) bis Element e in Liste erreicht
- falls key(e) $= k$, entferne e aus Liste (sonst return)



(a, b)-Baum

remove(k)

- entferne Handle auf e und Schlüssel k vom Baumknoten v über e (wenn e rechtestes Kind: Schlüsselvertauschung wie bei binärem Suchbaum)
- falls $d(v) \geq a$, dann fertig

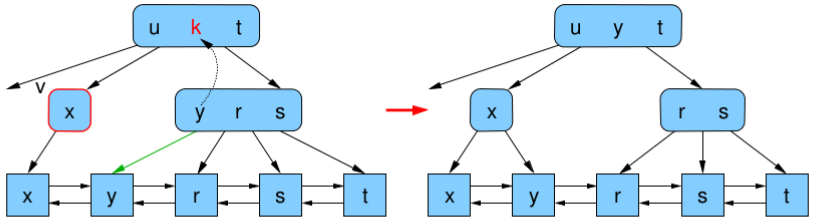


(a, b)-Baum

remove(k)

- falls $d(v) < a$ und ein direkter Nachbar v' von v hat Grad $> a$, nimm Kante von v'

Beispiel: (2, 4)-Baum

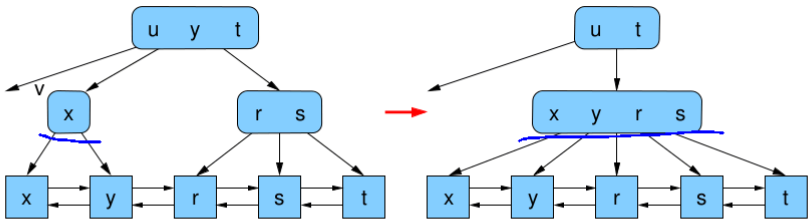


(a, b)-Baum

remove(k)

- falls $d(v) < a$ und kein direkter Nachbar von v hat Grad $> a$, merge v mit Nachbarn

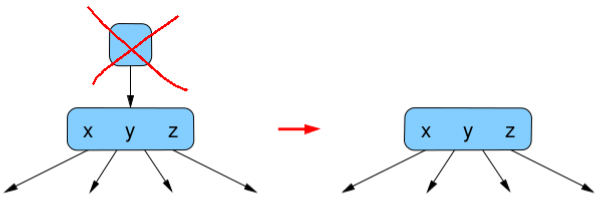
Beispiel: (3, 5)-Baum



(a, b)-Baum

remove(k)

- Verschmelzungen können sich nach oben fortsetzen, ggf. bis zur Wurzel
- falls Grad der Wurzel < 2: entferne Wurzel
  neue Wurzel wird das einzige Kind der alten Wurzel



(a, b)-Baum / remove

$a = 2, b = 4$

remove(10)