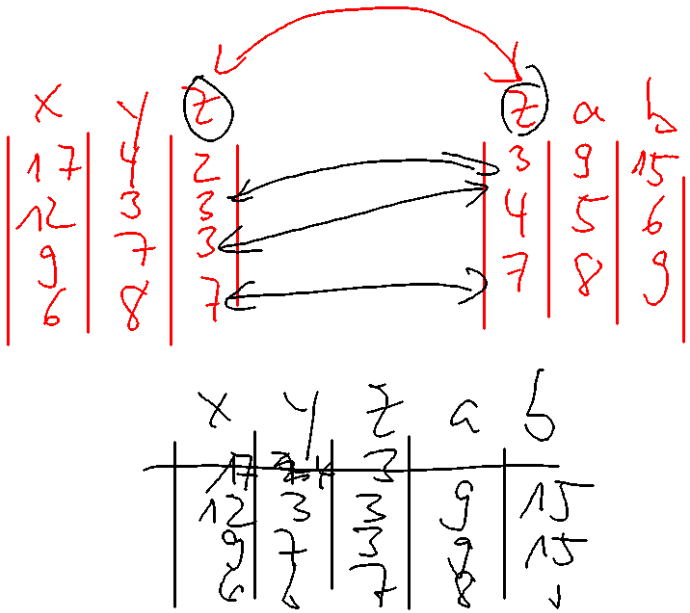
Script generated by TTT

Title: Täubig: GAD (28.06.2012)

Date: Thu Jun 28 12:32:21 CEST 2012

Duration: 45:27 min

Pages: 29



(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

Suchstrukturen (a, b)-Bäume

(a, b)-Baum / insert

Form-Invariante

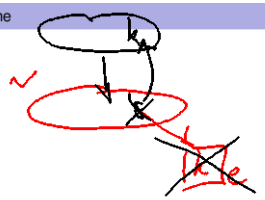
- alle Blätter haben dieselbe Tiefe, denn
  neues Blatt wird auf der Ebene der anderen eingefügt und
  im Fall einer neuen Wurzel erhöht sich die Tiefe aller Blätter um 1

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

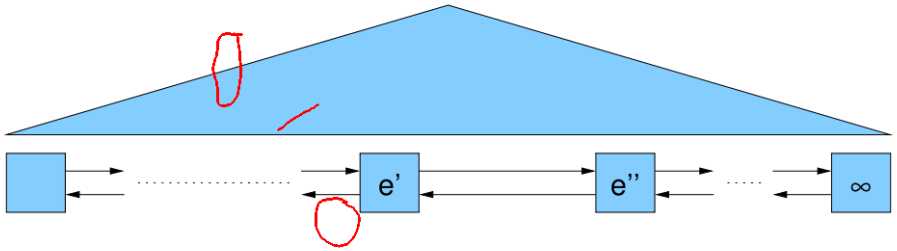
H. Täubig (TUM) GAD SS'12 352 / 637

(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)

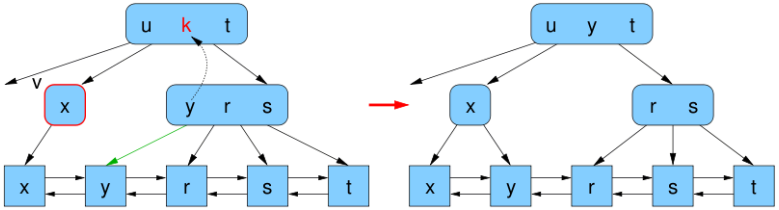


(a,b)-Baum

remove(k)

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

Beispiel: $(2,4)$-Baum

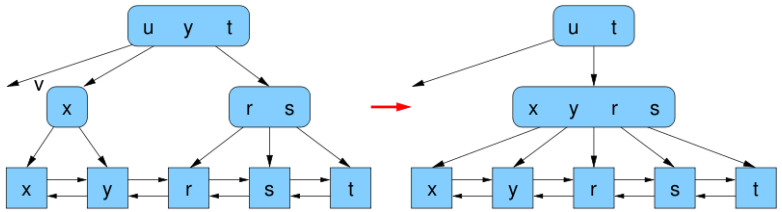


(a,b)-Baum

remove(k)

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

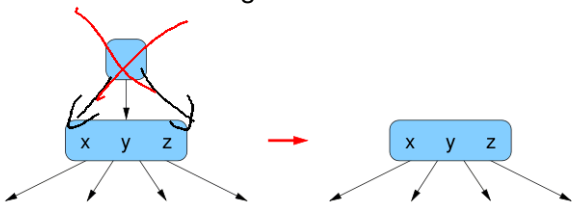
Beispiel: $(3,5)$-Baum



(a,b)-Baum

remove(k)

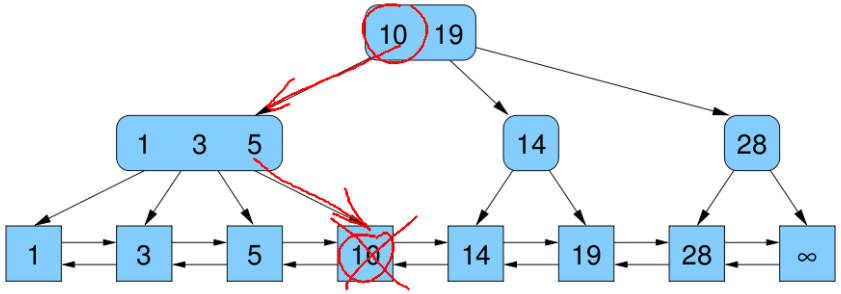
- Verschmelzungen können sich nach oben fortsetzen, ggf. bis zur Wurzel
- falls Grad der Wurzel < 2: entferne Wurzel
  neue Wurzel wird das einzige Kind der alten Wurzel



(a, b)-Baum / remove

$a = 2$, $b = 4$

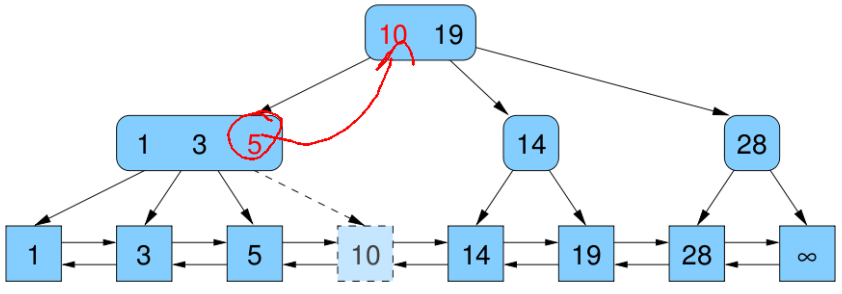
remove(10)



(a, b)-Baum / remove

$a = 2$, $b = 4$

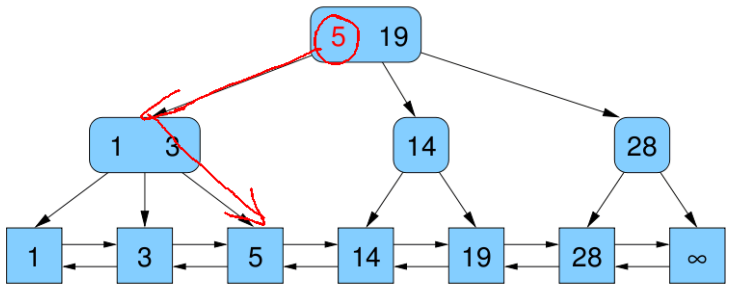
remove(10)



(a, b)-Baum / remove

$a = 2$, $b = 4$

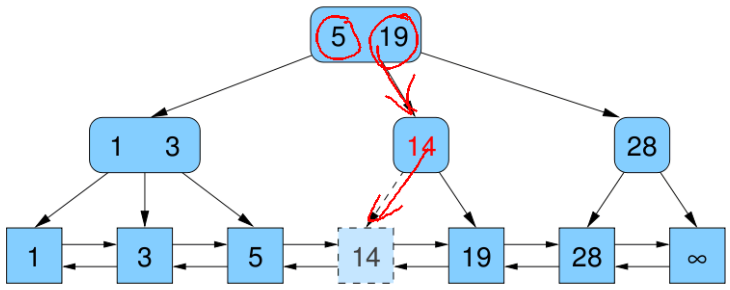
remove(10)



(a, b)-Baum / remove

$a = 2$, $b = 4$

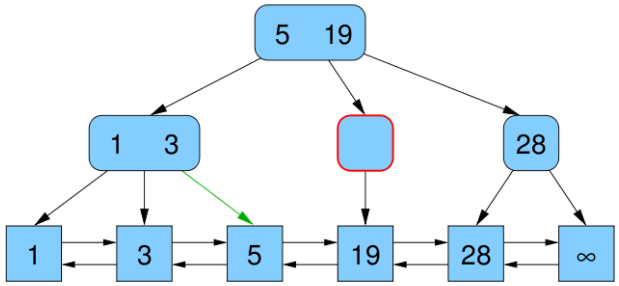
remove(14)



(a,b)-Baum / remove

$a = 2$, $b = 4$

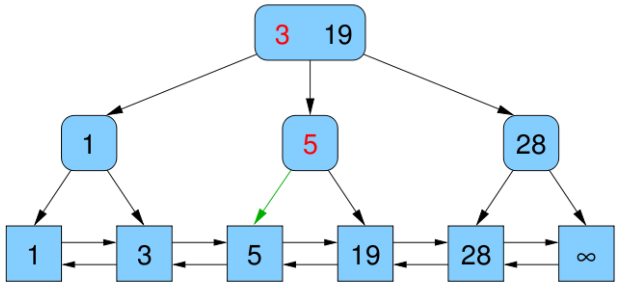
remove(14)



(a,b)-Baum / remove

$a = 2$, $b = 4$

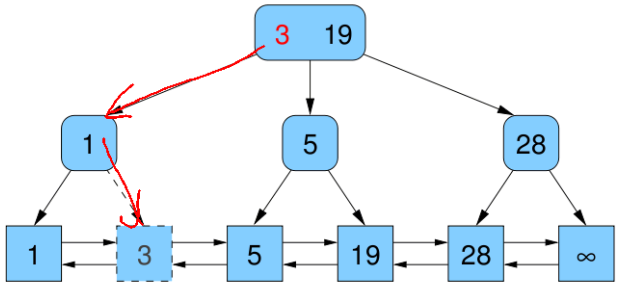
remove(14)



(a,b)-Baum / remove

$a = 2$, $b = 4$

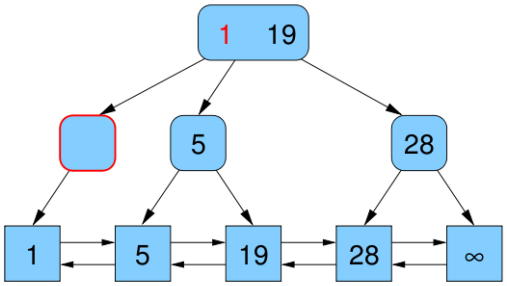
remove(3)



(a,b)-Baum / remove

$a = 2$, $b = 4$

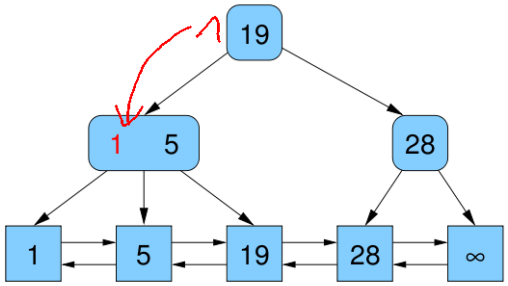
remove(3)



(a, b)-Baum / remove

$a = 2$, $b = 4$

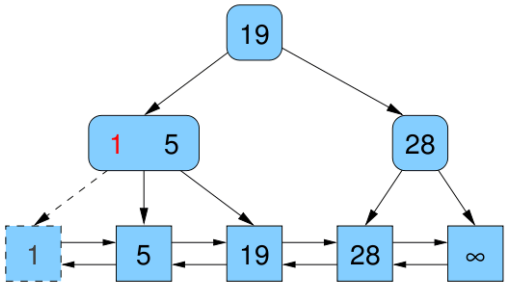
remove(3)



(a, b)-Baum / remove

$a = 2$, $b = 4$

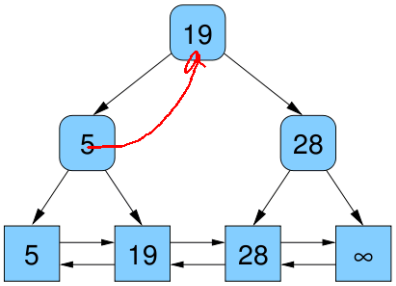
remove(1)



(a, b)-Baum / remove

$a = 2$, $b = 4$

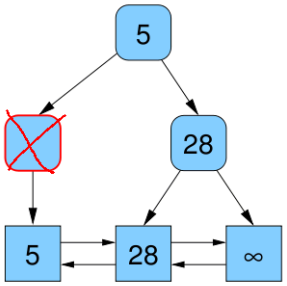
remove(1)



(a, b)-Baum / remove

$a = 2$, $b = 4$

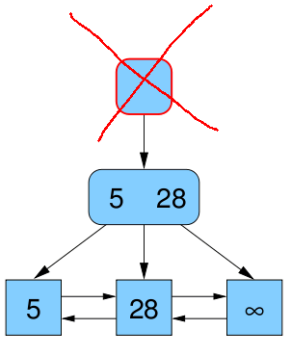
remove(19)



(a, b)-Baum / remove

$a = 2, b = 4$

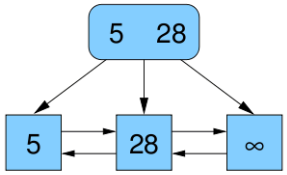
remove(19)



(a, b)-Baum / remove

$a = 2, b = 4$

remove(19)



(a, b)-Baum / remove

Form-Invariante

- alle Blätter behalten dieselbe Tiefe
- falls alte Wurzel entfernt wird, verringert sich die Tiefe aller Blätter

Grad-Invariante

$a - 1 + a = 2a - 1 \leq b$

- remove verschmilzt Knoten, die Grad $a - 1$ und a haben
- wenn $b \geq 2a - 1$, dann ist der resultierende Grad $\leq b$
- remove verschiebt eine Kante von Knoten mit Grad $> a$ zu Knoten mit Grad $a - 1$, danach sind beide Grade in $[a, b]$
- wenn Wurzel gelöscht, wurden vorher die Kinder verschmolzen, Grad vom letzten Kind ist also $\geq a$ (und $\leq b$)

Weitere Operationen im (a, b)-Baum

- min / max-Operation

  verwende first / last-Methode der Liste, um das kleinste bzw. größte Element auszugeben

  Zeit: $O(1)$

- Range queries (Bereichsanfragen)

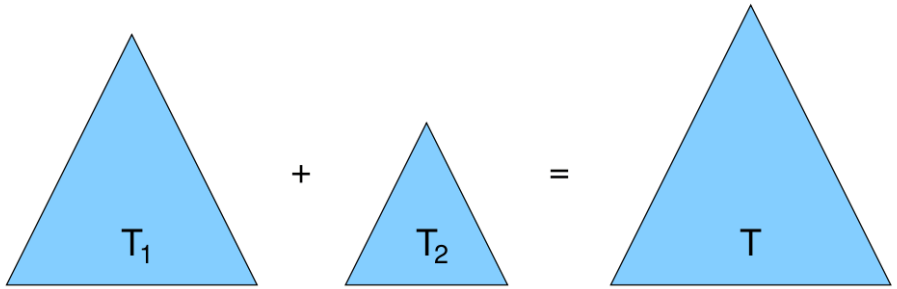
  suche alle Elemente im Bereich $[x, y]$:
  - führe locate(x) aus und
  - durchlaufe die Liste, bis Element $> y$ gefunden wird

  Zeit: $O(\log n + \text{Ausgabegröße})$

- Konkatenation / Splitting

Konkatenation von (a, b)-Bäumen

- verknüpfe zwei (a, b)-Bäume T_1 und T_2 mit s_1 bzw. s_2 Elementen und Höhe h_1 bzw. h_2 zu (a, b)-Baum T
- Bedingung: Schlüssel in $T_1 \leq$ Schlüssel in T_2



Konkatenation von (a, b)-Bäumen

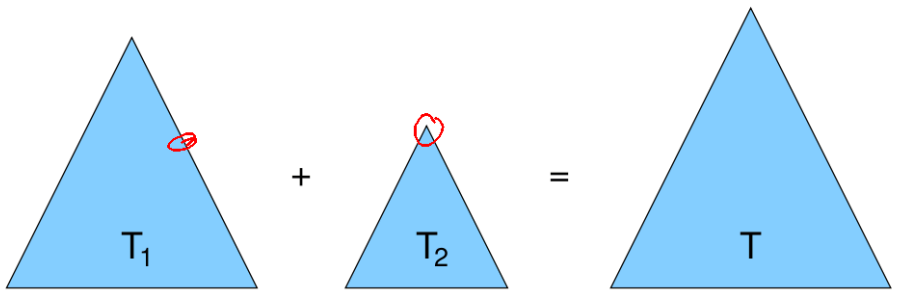
- lösche in T_1 das ∞-Dummy-Element
- wenn danach dessen Vater-Knoten $< a$ Kinder hat, dann behandle dies wie bei remove
- verschmelze die Wurzel des niedrigeren Baums mit dem entsprechenden äußersten Knoten des anderen Baums, der sich auf dem gleichen Level befindet
- wenn dieser Knoten danach $> b$ Kinder hat, dann behandle dies wie bei insert

$\Rightarrow$ falls Höhe der Bäume explizit gespeichert: Zeit $O(1 + |h_1 - h_2|)$

ansonsten (mit Höhenbestimmung): Zeit $O(1 + \max\{h_1, h_2\})$
$\subseteq O(1 + \log(\max\{s_1, s_2\}))$

Konkatenation von (a, b)-Bäumen

- verknüpfe zwei (a, b)-Bäume T_1 und T_2 mit s_1 bzw. s_2 Elementen und Höhe h_1 bzw. h_2 zu (a, b)-Baum T
- Bedingung: Schlüssel in $T_1 \leq$ Schlüssel in T_2



Konkatenation von (a, b)-Bäumen

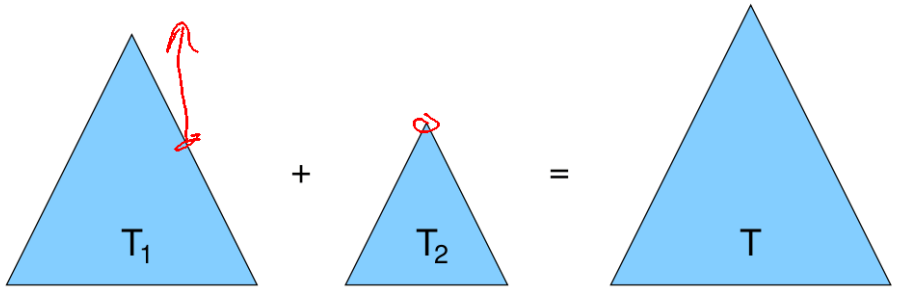
- lösche in T_1 das ∞-Dummy-Element
- wenn danach dessen Vater-Knoten $< a$ Kinder hat, dann behandle dies wie bei remove
- verschmelze die Wurzel des niedrigeren Baums mit dem entsprechenden äußersten Knoten des anderen Baums, der sich auf dem gleichen Level befindet
- wenn dieser Knoten danach $> b$ Kinder hat, dann behandle dies wie bei insert

$\Rightarrow$ falls Höhe der Bäume explizit gespeichert: Zeit $O(1 + |h_1 - h_2|)$

ansonsten (mit Höhenbestimmung): Zeit $O(1 + \max\{h_1, h_2\})$
$\subseteq O(1 + \log(\max\{s_1, s_2\}))$

Konkatenation von (a, b)-Bäumen

- verknüpfe zwei (a, b)-Bäume T_1 und T_2 mit s_1 bzw. s_2 Elementen und Höhe h_1 bzw. h_2 zu (a, b)-Baum T
- Bedingung: Schlüssel in $T_1 \leq$ Schlüssel in T_2



Konkatenation von (a, b)-Bäumen

- lösche in T_1 das ∞-Dummy-Element
- wenn danach dessen Vater-Knoten $< a$ Kinder hat, dann behandle dies wie bei remove
- verschmelze die Wurzel des niedrigeren Baums mit dem entsprechenden äußersten Knoten des anderen Baums, der sich auf dem gleichen Level befindet
- wenn dieser Knoten danach $> b$ Kinder hat, dann behandle dies wie bei insert

$\Rightarrow$ falls Höhe der Bäume explizit gespeichert: Zeit $O(1 + |h_1 - h_2|)$

ansonsten (mit Höhenbestimmung): Zeit $O(1 + \max\{h_1, h_2\})$
$\subseteq O(1 + \log(\max\{s_1, s_2\}))$