

Script generated by TTT

Title: Seidl: GAD (01.06.2016)

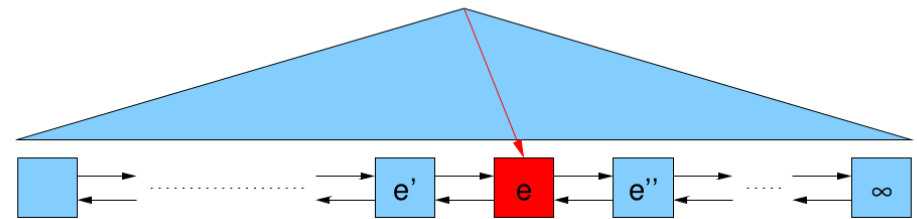
Date: Wed Jun 01 13:32:58 CEST 2016

Duration: 33:34 min

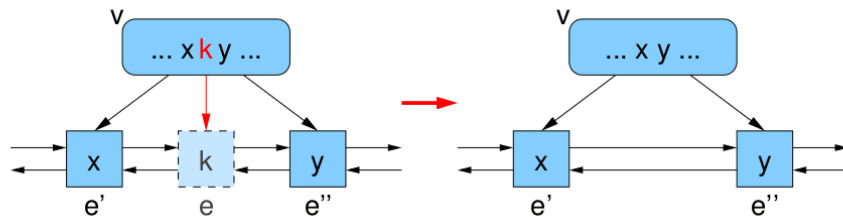
Pages: 22

 (a, b) -Baumremove(k)

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

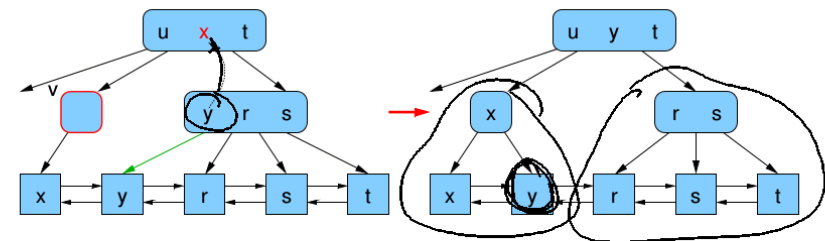
 (a, b) -Baumremove(k)

- 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) -Baumremove(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

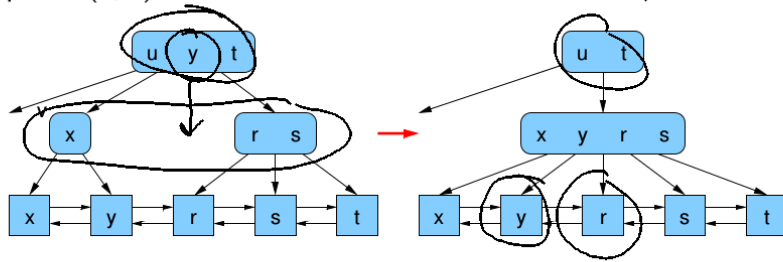
remove(k)

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

$$Q = 1 + Q = 2Q - 1$$

$$2Q - 2$$

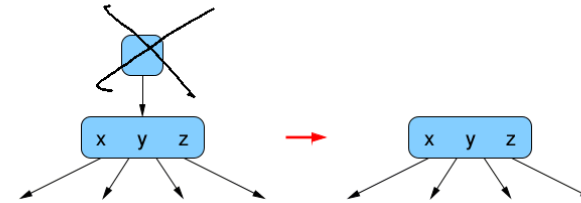
Beispiel: (3,5)-Baum



(a,b)-Baum

remove(k)

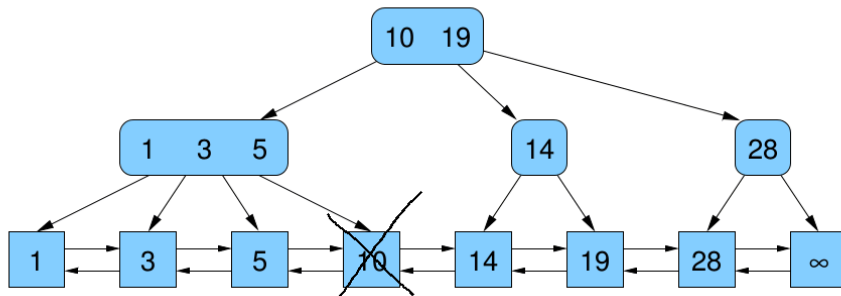
- Verschmelzungen können sich nach oben fortsetzen, ggf. bis zur Wurzel
- falls Grad der Wurzel < 2 : entferne Wurzel neue Wurzel wird das einzige Kind der alten Wurzel



(a,b)-Baum / remove

$a = 2, b = 4$

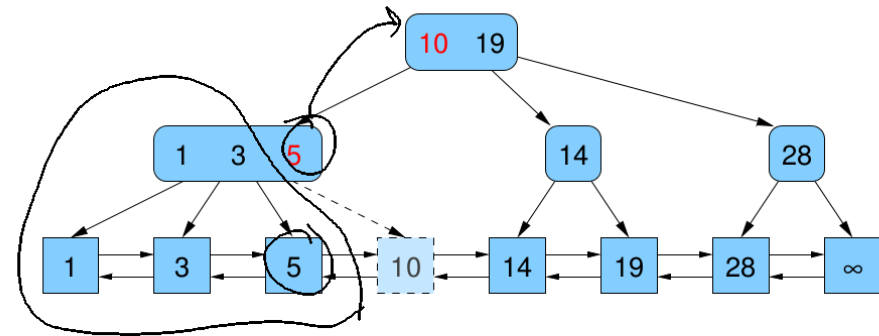
remove(10)



(a,b)-Baum / remove

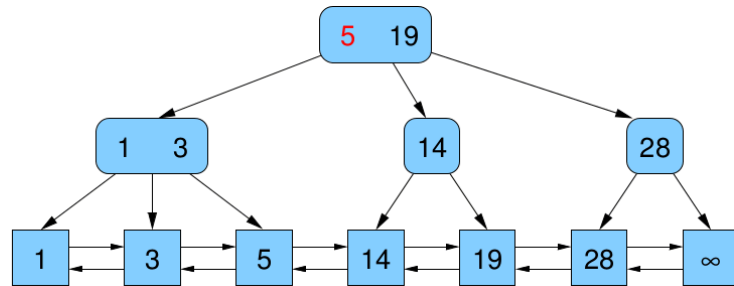
$a = 2, b = 4$

remove(10)

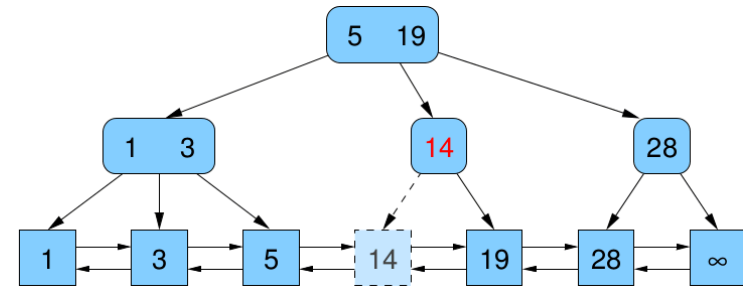


(a, b) -Baum / remove $a = 2, b = 4$

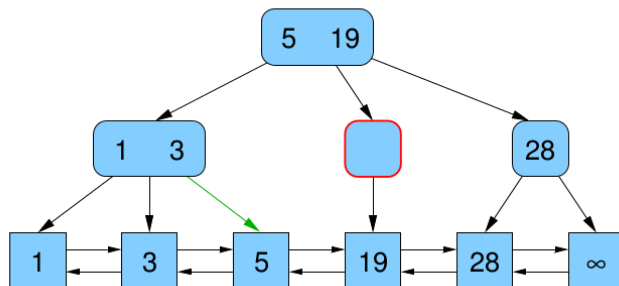
remove(10)

 (a, b) -Baum / remove $a = 2, b = 4$

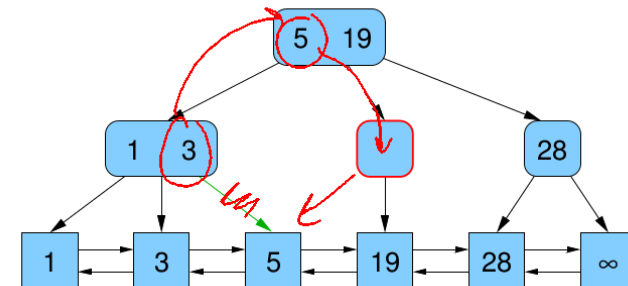
remove(14)

 (a, b) -Baum / remove $a = 2, b = 4$

remove(14)

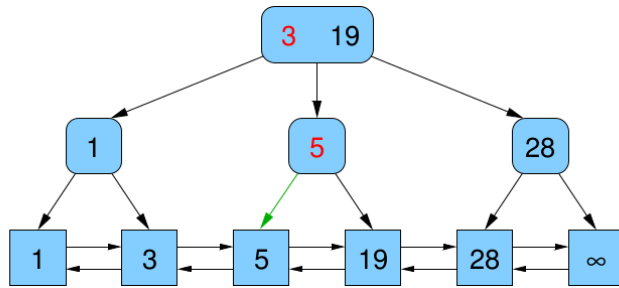
 (a, b) -Baum / remove $a = 2, b = 4$

remove(14)

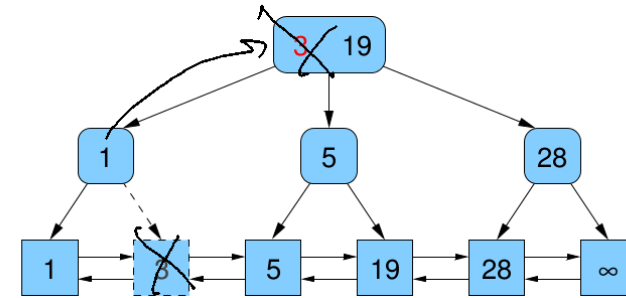


(a, b) -Baum / remove $a = 2, b = 4$

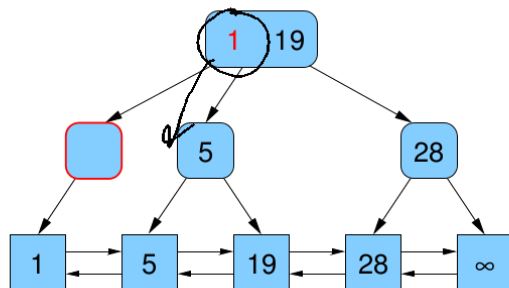
remove(14)

 (a, b) -Baum / remove $a = 2, b = 4$

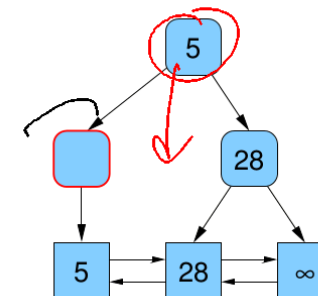
remove(3)

 (a, b) -Baum / remove $a = 2, b = 4$

remove(3)

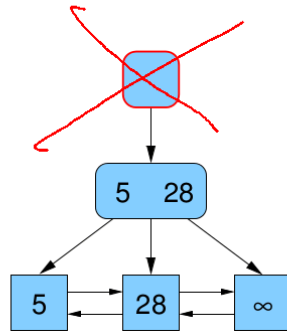
 (a, b) -Baum / remove $a = 2, b = 4$

remove(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

- 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 $[x, y]$:

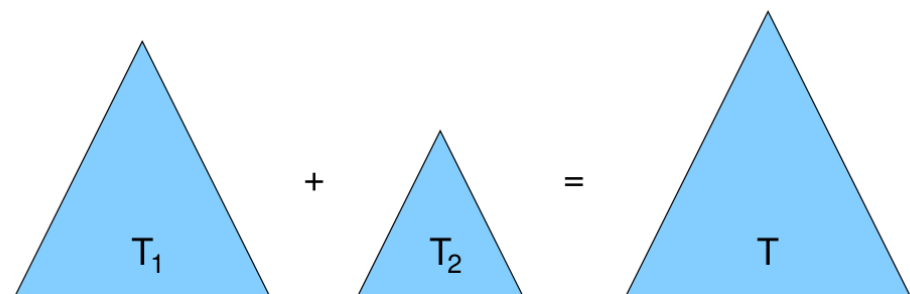
- ▶ führe locate(x) aus und
- ▶ durchlaufe die Liste, bis Element $> y$ gefunden wird

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

• Konkatenation / Splitting

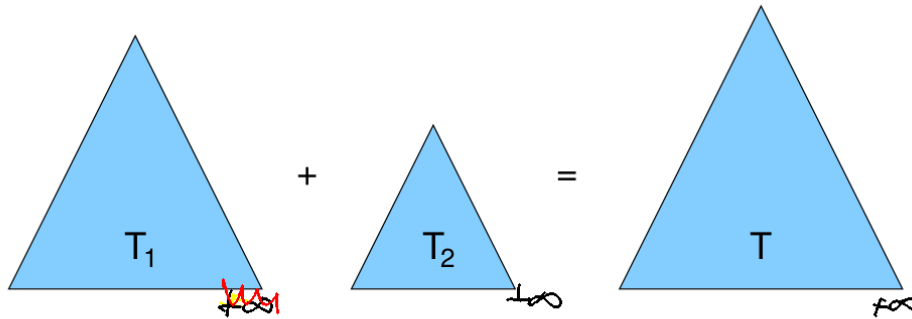
Konkatenation von (a, b) -Bäumen

- 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

- 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

⇒ 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

- 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

⇒ falls Höhe der Bäume explizit gespeichert: Zeit $O(1 + |h_1 - h_2|)$
 ansonsten (mit Höhenbestimmung): Zeit $O(1 + \max\{h_1, h_2\})$
 $\subseteq O(1 + \log(\max\{s_1, s_2\}))$