Informatik II: Algorithmen und Datenstrukturen SS 2013

Vorlesung 8b, Dienstag, 12. Juni 2013
(Balancierte Suchbäume)

Prof. Dr. Hannah Bast
Lehrstuhl für Algorithmen und Datenstrukturen
Institut für Informatik
Universität Freiburg

UNI FREIBURG

Balancierte Bäume

UNI FREIBURG

- Motivation
  - Mit BinarySearchTree hatten wir lookup und insert in Zeit $\Theta(d)$, wobei d = Tiefe des Baumes
  - Wenn es gut läuft ist $d = O(\log n)$

    zum Beispiel wenn die Schlüssel zufällig gewählt sind
  - Wenn es schlecht läuft ist $d = \Theta(n)$

    zum Beispiel wenn der Reihe nach 1, 2, 3, ... eingefügt wird
  - Wir wollen uns aber nicht auf eine bestimmte Eigenschaft der Schlüsselmenge verlassen müssen ... wie beim Hashing !
  - Und werden uns heute deswegen explizit darum kümmern, dass der Baum **immer** Tiefe $d = O(\log n)$ hat

(a,b)-Bäume 1/8

UNI FREIBURG

- Wie erreicht man immer Tiefe O(log n) ?
  - Es gibt Dutzende verschiedener Verfahren dafür:

    AVL-Bäume

    AA-Bäume

    Rot-Schwarz-Bäume

    Splay trees

    Treaps

    …
  - Wir machen heute (a,b)-Bäume

    Die sind intuitiv, einfach und praktisch

(a,b)-Bäume 2/8

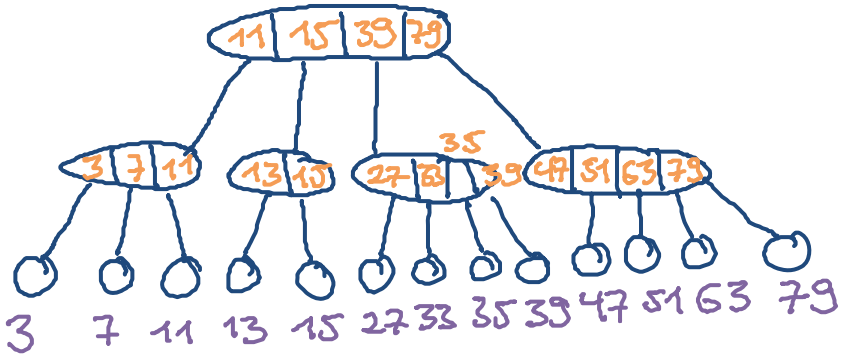
UNI FREIBURG

- Definition (a,b)-Baum
  - Die Elemente / Schlüssel stehen nur in den Blättern
  - Alle Blätter haben die gleiche Tiefe
  - Jeder innere Knoten hat $\geq a$ und $\leq b$ Kinder

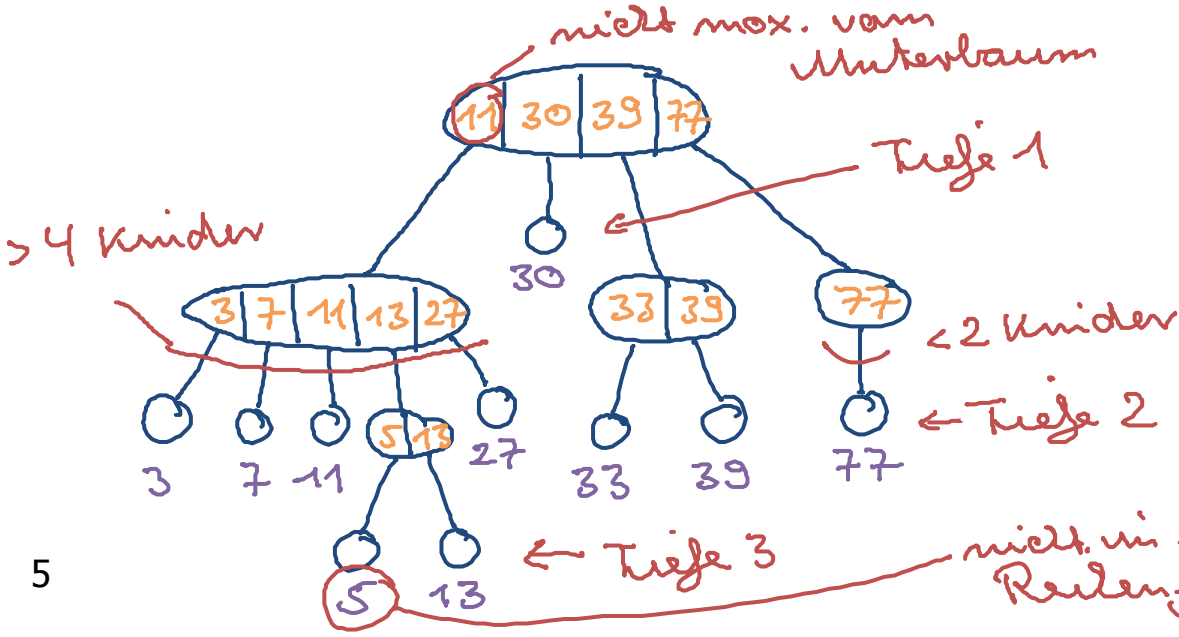
    (nur die Wurzel darf weniger Kinder haben)
  - Wir verlangen $a \geq 2$ und $b \geq 2a - 1$ ... warum sehen wir gleich
  - An den inneren Knoten steht für jedes Kind der größte Schlüssel in dem Unterbaum dieses Kindes

(a,b)-Bäume 3/8

- Beispiel und Gegenbeispiel für einen (2,4)-Baum



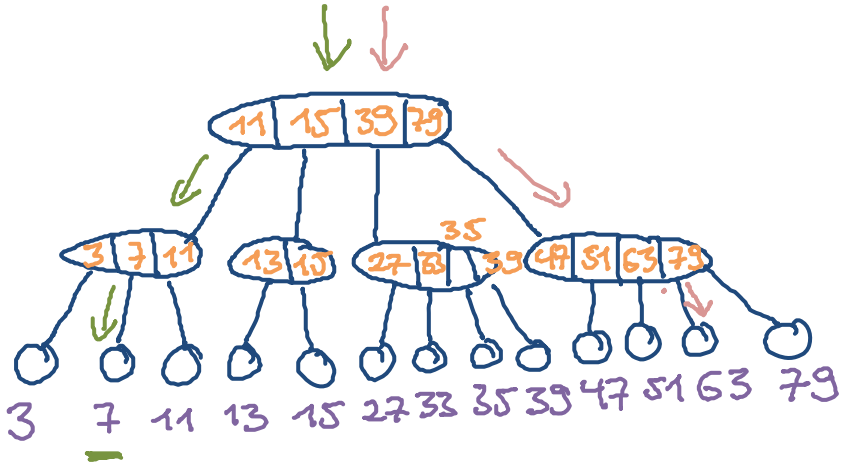
Positivbeispiel für einen (2,4) Baum



Negativbeispiel für einen (2,4) Baum alle problematischen Stellen in ROT markiert

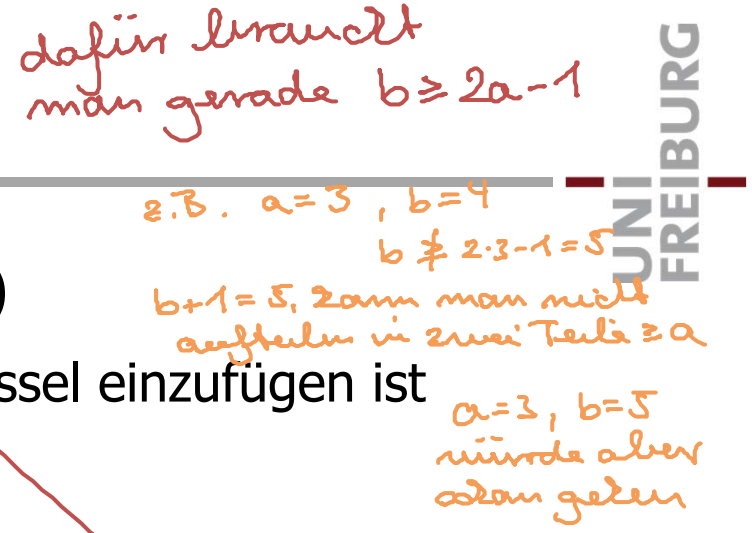
(a,b)-Bäume 4/8

- Schlüsselsuche (Lookup)
  - Im Prinzip genau so wie beim BinarySearchTree
  - Suche von der Wurzel abwärts
  - Die Schlüssel an den inneren Knoten weisen den Weg



lookup(7)
⇝ 7
lookup(53)
⇝ 63
(das nächst-größere)

(a,b)-Bäume 5/8



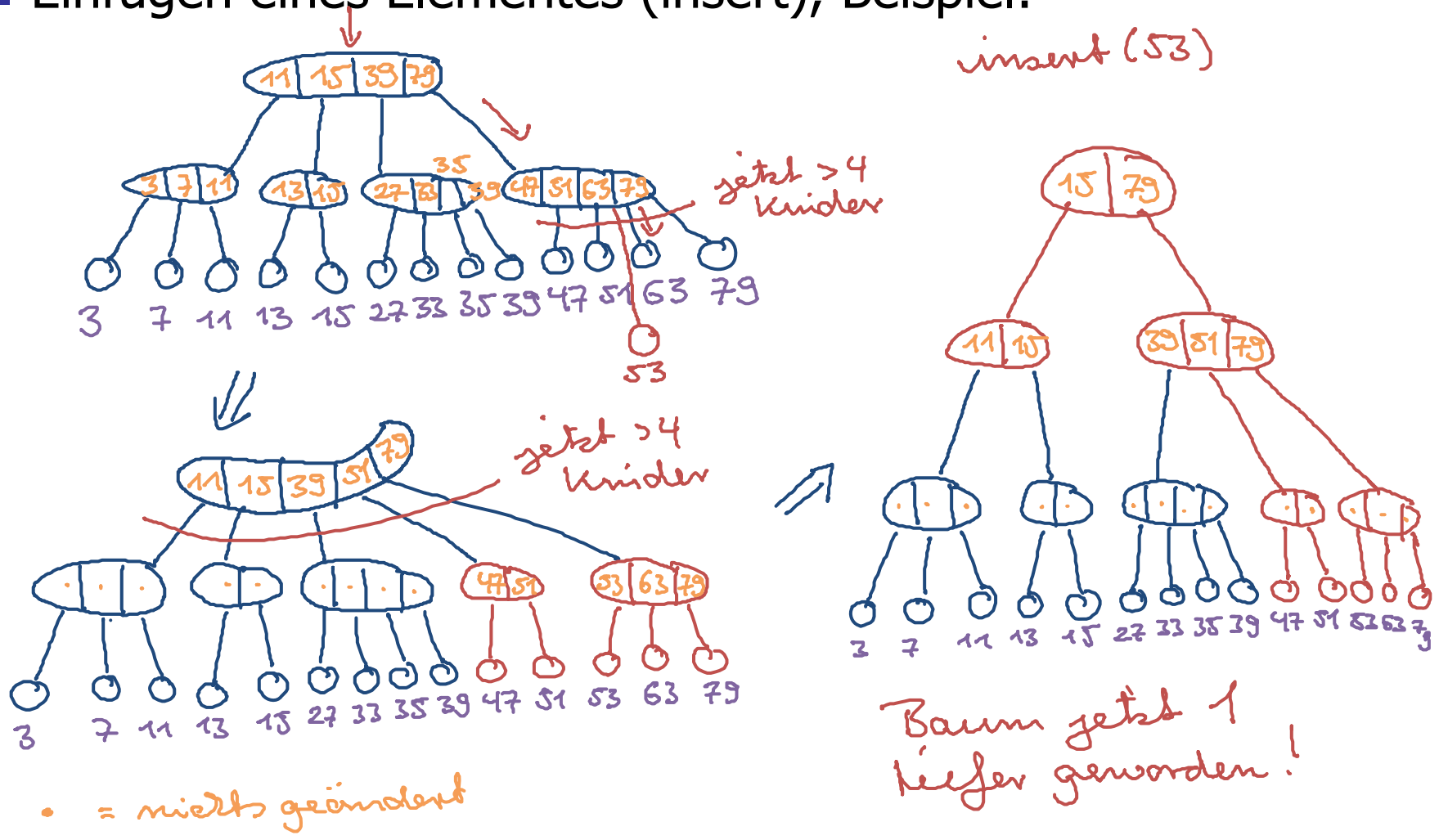
- Einfügen eines Elementes (insert)
  - Finde die Stelle, wo der neue Schlüssel einzufügen ist
  - Und füge dort ein neues Blatt ein
  - Achtung: der Elternknoten kann jetzt b+1 Knoten haben!
  - Dann **spalten** wir den Elternknoten einfach auf in zwei Knoten mit ceil(b/2) und floor(b/2) + 1 Kinder
    - für $b \geq 2a-1$ ist $ceil(b/2) \geq a$ und $floor(b/2) + 1 \geq a$
  - Der Großelternknoten kann jetzt b+1 Kinder haben
  - Dann spalten wir den auf dieselbe Weise auf ... usw.
  - Wenn das bis zur Wurzel geht, spalten wir auch die auf und erzeugen einen neuen Wurzelknoten → Baum dann **1 tiefer**

(a,b)-Bäume 6/8

für einen (2,4)-Baum

UNI FREIBURG

- Einfügen eines Elementes (insert), Beispiel:



(a,b)-Bäume 7/8

UNI FREIBURG

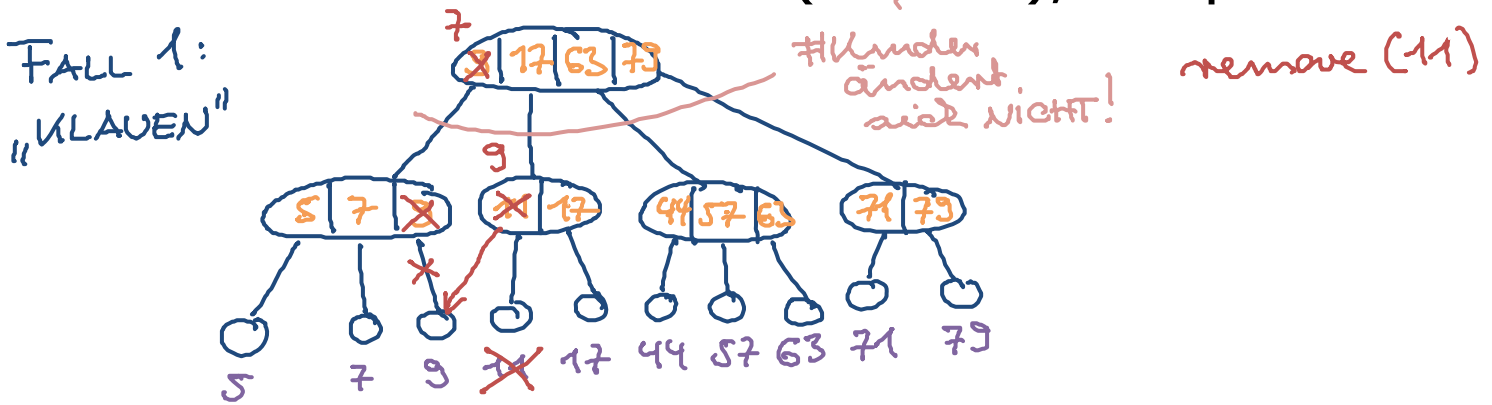
- Entfernen eines Elementes (remove)
  - Finde das zu entfernende Element
  - Und lösche das entsprechende Blatt
  - Achtung: der Elternknoten kann jetzt a-1 Kinder haben!
  - **Fall 1:** Falls eines der anderen Kinder des Elternknoten > a Kinder hat, **klauen** eins von da weg und sind fertig
  - **Fall 2:** Sonst **verschmelzen** wir den Elternknoten mit einem seiner Geschwister
  - Der Großelternknoten hat jetzt ein Kind weniger und kann jetzt a-1 Kinder haben **...** und so weiter evtl. bis zur Wurzel
  - Wenn die Wurzel am Ende nur noch ein Kind hat, mache dieses Kind zur neuen Wurzel → Baum dann **1 weniger tief**

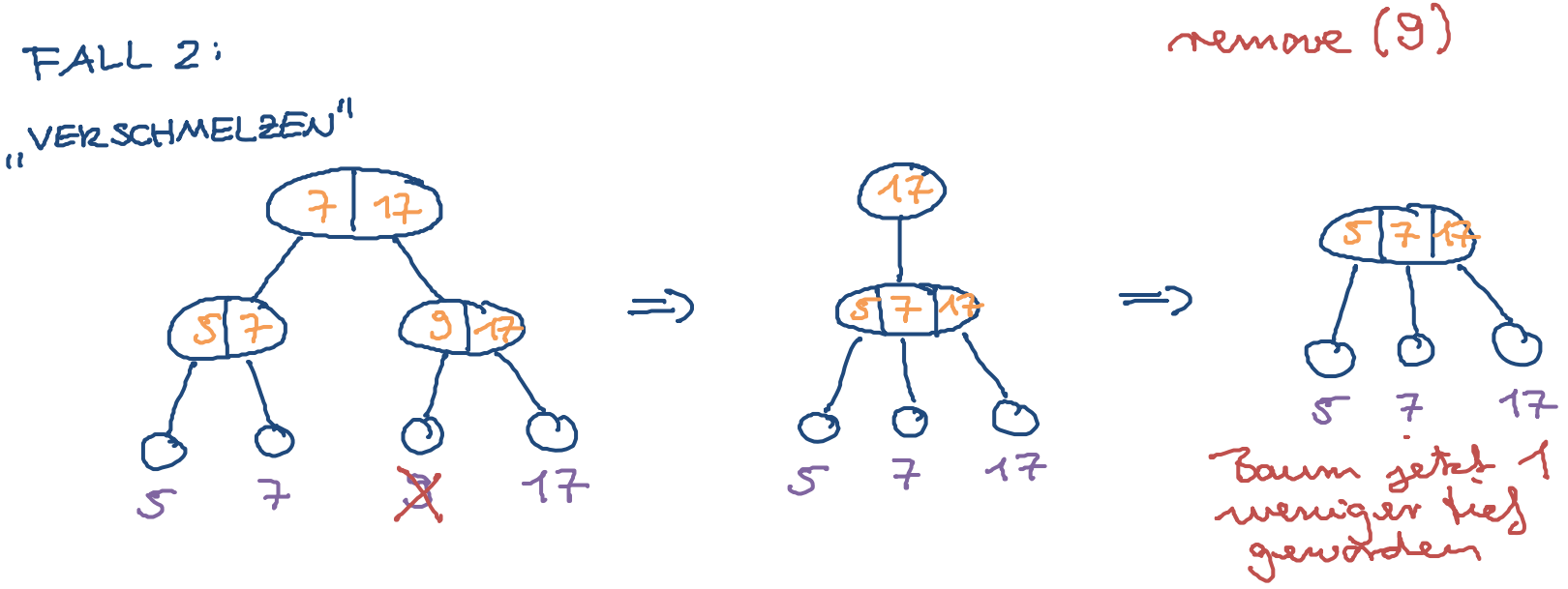
(a,b)-Bäume 8/8

wieder für (2,4)-Bäume

UNI FREIBURG

- Entfernen eines Elementes (remove), Beispiele:





Analyse (a,b)-Bäume 1/3

a und b gelten dabei als Konstanten

UNI FREIBURG

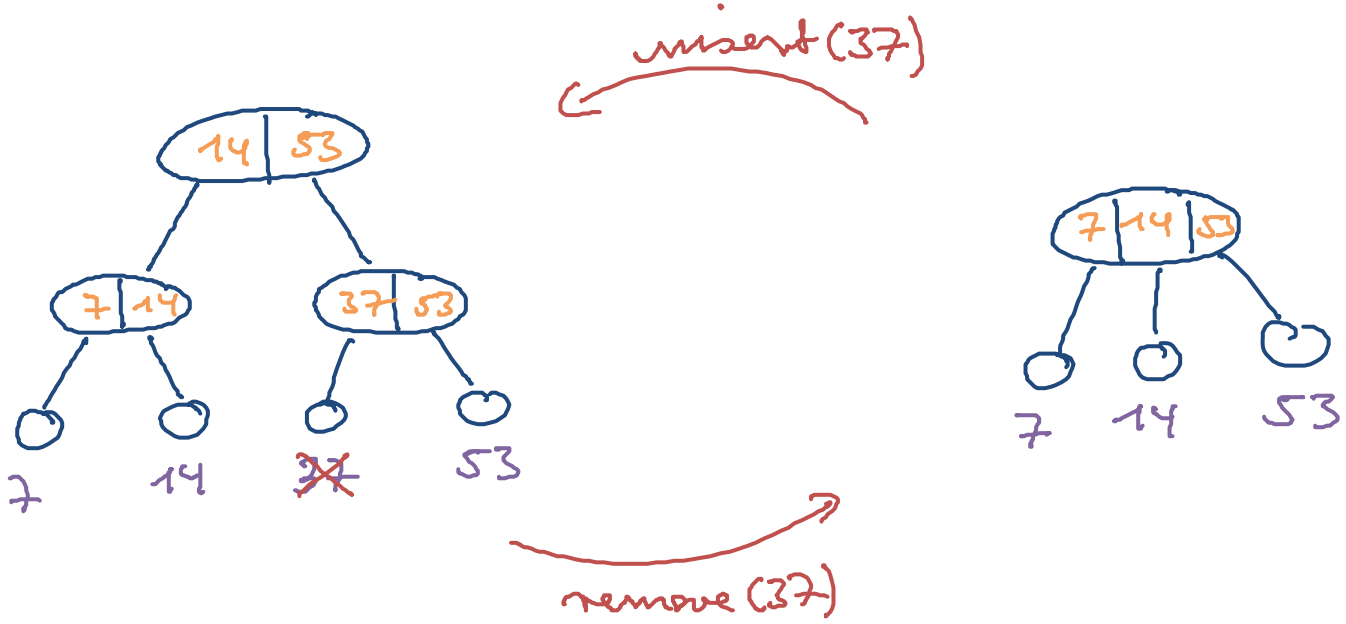
- Komplexität für lookup, insert, remove
  - Gehen alle in Zeit $O(d)$, wobei d = Tiefe des Baumes
  - Jeder Knoten, außer evtl. der Wurzel, hat $\geq a$ Kinder
    deshalb $n \geq a^{d-1}$ und deshalb $d \leq 1 + \log_a n = O(\log_a n)$
  - Bei genauerem Hinsehen fällt auf
    - Die Operation lookup braucht immer Zeit $\Theta(d)$
    - Aber insert und remove scheinen oft in $O(1)$ zu gehen

      (nur im schlechtesten Fall müssen alle Knoten auf dem Weg zur Wurzel geteilt / verschmolzen werden)
  - Das wollen wir jetzt genauer analysieren
  - Dafür reicht $b \geq 2a - 1$ allerdings nicht, wir brauchen **$b \geq 2a$**

Analyse (a,b)-Bäume 2/3

- Gegenbeispiel für $b = 2a - 1$

$a = 2,\ b = 3$
also (2,3)-Baum



auf diese Weise kann man für $b = 2a - 1$ eine Folge von n Operationen konstruieren, so dass jede $\Theta(\log n)$ kostet

Analyse (a,b)-Bäume 3/3

UNI FREIBURG

- Für $b \geq 2a$ gilt aber immer
  - … dass die Kosten für eine beliebige Folge von n Operationen (jede davon insert oder remove) $O(n)$ sind

    Also amortisiert / im Durchschnitt $O(1)$ pro Operation
  - Im Folgenden wollen wir das für $a = 2$, $b = 4$ beweisen
  - Übungsblatt 8, Aufgabe 2: $a = 3$, $b = 7$

    Wenn man es einmal verstanden hat, ist es aber auch leicht, das für allgemeine a und b mit $b \geq 2a$ zu beweisen

Analyse (2,4)-Bäume 1/4

UNI FREIBURG

- Intuition
  - Wenn alle Knoten im Baum **2** Kinder haben, müssen wir nach einem remove alle Knoten bis zur Wurzel verschmelzen
  - Wenn alle Knoten im Baum **4** Kinder haben, müssen wir nach einem insert alle Knoten bis zur Wurzel aufspalten
  - Wenn alle Knoten im Baum **3** Kinder haben, dauert es lange bis wir in eine dieser beiden Situationen kommen
  - Idee für die Analyse: nach einer teuren Operation ist der Baum in einem Zustand, dass es dauert, bis es wieder teuer wird
  - Ähnlich wie bei dynamischen Feldern: Reallokation ist teuer, aber danach dauert es, bis wieder realloziert werden muss

    Bei geeigneter "Überallokation" Kosten im Durchschnitt O(1)

Analyse (2,4)-Bäume 2/4

UNI FREIBURG

- Terminologie
  - Wir betrachten eine Folge von n Operationen
  - Seien T_i die Kosten = Laufzeit der i-ten Operation
  - Sei Φ_i das Potenzial des Baumes nach der i-ten Operation

    Φ_i := die Anzahl der Knoten mit Grad genau 3

    $\Phi_0 := 0$ (das Potenzial am Anfang, für den leeren Baum)
- Mastertheorem aus Vorlesung 6b, Folie 12
  - Falls gilt $T_i \leq A \cdot (\Phi_i - \Phi_{i-1}) + B$ für irgendwelche $A, B > 0$

    Dann $\Sigma_{i=1..n} T_i = O(n)$

Analyse (2,4)-Bäume 3/4

Grad von einem Knoten = #Kinder

- Fall 1: i-te Operation ist ein insert …

Bei Aufspalten, erhöht sich das Potenzial auf der Tiefe des Baumes um +1
Da wo das Aufspalten aufhört, kann sich das Potenzial um 1 erniedrigen (vorher Grad 3, danach Grad 4)

$m = \#\text{Aufspaltungen}$

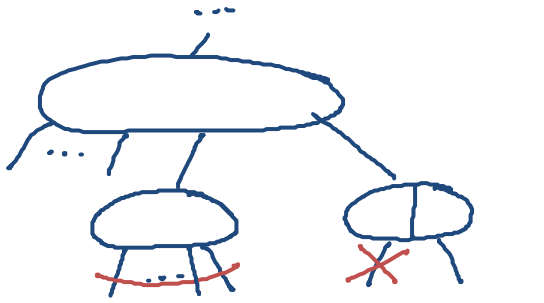
$$\Rightarrow \Phi_i \geq \Phi_{i-1} + m - 1 \iff m \leq \Phi_i - \Phi_{i-1} + 1$$

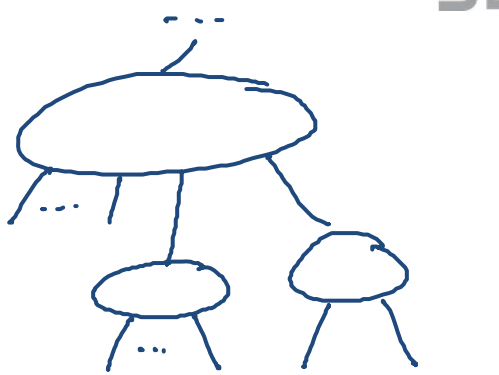
$$T_i \leq A \cdot m + B \leq A \cdot (\Phi_i - \Phi_{i-1}) + \underbrace{A + B}_{=:B'}$$

Analyse (2,4)-Bäume 4/4

- Fall 2: i-te Operation ist ein remove

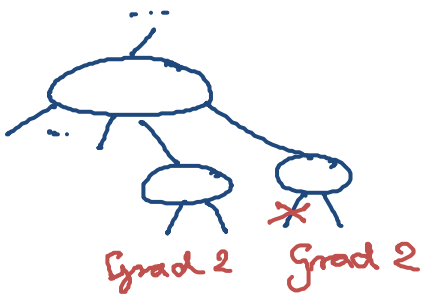
FALL 2.1
„KLAUEN"
(passiert nicht
nicht ganz =
höchstens
einmal)





ALSO: Potenzial entweder +1 oder −1

FALL 2.2.
„VERSCHMELZEN"



$m = \#\text{Verschmelzungen} \Rightarrow \Phi_i \geq \Phi_{i-1} + m - 1$

$T_i \leq A \cdot m + B \Rightarrow T_i \leq A \cdot (\Phi_i - \Phi_{i-1}) + \underbrace{A+B}_{=:B'}$

Literatur / Links

UNI FREIBURG

- (a,b)-Bäume
  - In Mehlhorn/Sanders:

    7 Sorted Sequences [Kapitel 7.2 und 7.4]
  - In Cormen/Leiserson/Rivest

    14 Red-Black Trees [die sind ähnlich, aber anders]
  - In Wikipedia

    http://en.wikipedia.org/wiki/(a,b)-tree (Englisch)

    http://cs.wikipedia.org/wiki/(a,b)-strom (Tschechisch)