Algorithmen und Datenstrukturen (ESE)
Entwurf, Analyse und Umsetzung von Algorithmen (IEMS)
WS 2012 / 2013

Vorlesung 11, Dienstag, 15. Januar 2013
(Balancierte Suchbäume)

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

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Organisatorisches
  - Ihre Erfahrungen mit dem Ü10 (Binäre Suchbäume)
- Balancierte Suchbäume
  - Vorlesung 10: bei binären Suchbäumen brauchen insert und lookup im worst case Zeit $\Theta(d)$, d = Tiefes des Baumes
  - Übungsblatt 10: Tiefe kann $O(\log n)$ sein (zufällige Schlüssel), kann aber auch $\Theta(n)$ sein (Schlüssel 1, 2, 3, ...)
  - **Heute:** Balancierte Suchbäume = immer Tiefe $O(\log n)$
  - Konkret (a,b)-Bäume, Korrektheitsbeweis für (2,4)-Bäume
  - **Übungsblatt 11:** Korrektheitsbeweis für (4,9)-Bäume

Erfahrungen mit dem Ü10 (binäre Suchbäume)

UNI FREIBURG

- Zusammenfassung / Auszüge Stand 15. Januar 15:36
  - Stoff in Vorlesung (Björn Buchhold) gut erklärt
  - Übungsblatt war gut gestellt und abwechslungsreich
  - Laufzeit-Experimente interessant + helfen beim Verständnis
  - Vernünftige `toString()` Methode war am schwierigsten
  - Vorprogrammieren lieber von Anfang an, nicht mit viel vorgeschriebenem Code anfangen
  - Anwendungsbeispiele wären gut
  - Leere Folie am Ende bitte weglassen … ok
  - Weltuntergang: leider erst nach dem nächsten Übungsblatt

Balancierte Bäume

UNI FREIBURG

- Motivation
  - Mit BinarySearchTree hatten wir lookup und insert in Zeit O(d), wobei d = Tiefe des Baumes
  - Wenn es gut läuft ist d = O(log n)
    - z.B. wenn die Schlüssel zufällig gewählt sind
  - Wenn es schlecht läuft ist d = Θ(n)
    - z.B. wenn der Reihe nach 1, 2, 3, ... eingefügt werden
  - Wir wollen uns aber nicht auf eine bestimmte Eigenschaft der Schlüsselmenge verlassen müssen
  - Und werden uns heute deswegen explizit darum kümmern, dass der Baum immer Tiefe O(log n) hat

(a,b)-Bäume 1/8

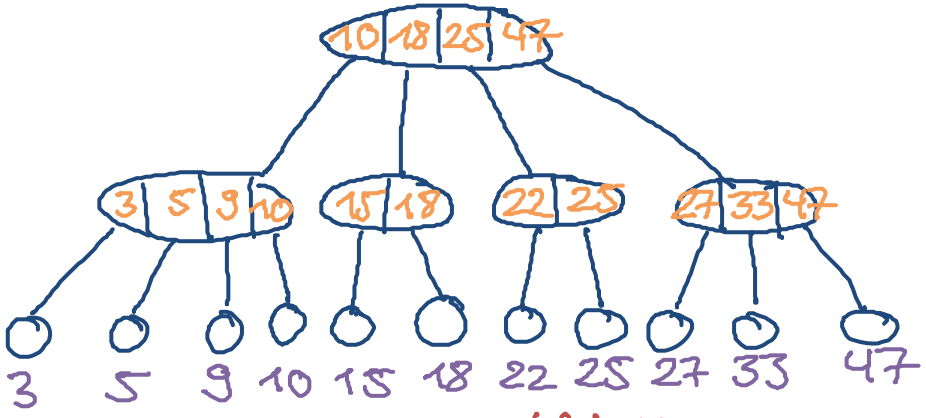
UNI FREIBURG

- Wie erreicht man immer Tiefe O(log n) ?
  - Es gibt Dutzende verschiedener Verfahren dafür
  - Heute (a,b)-Bäume ... die sind intuitiv, einfach und praktisch
- 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

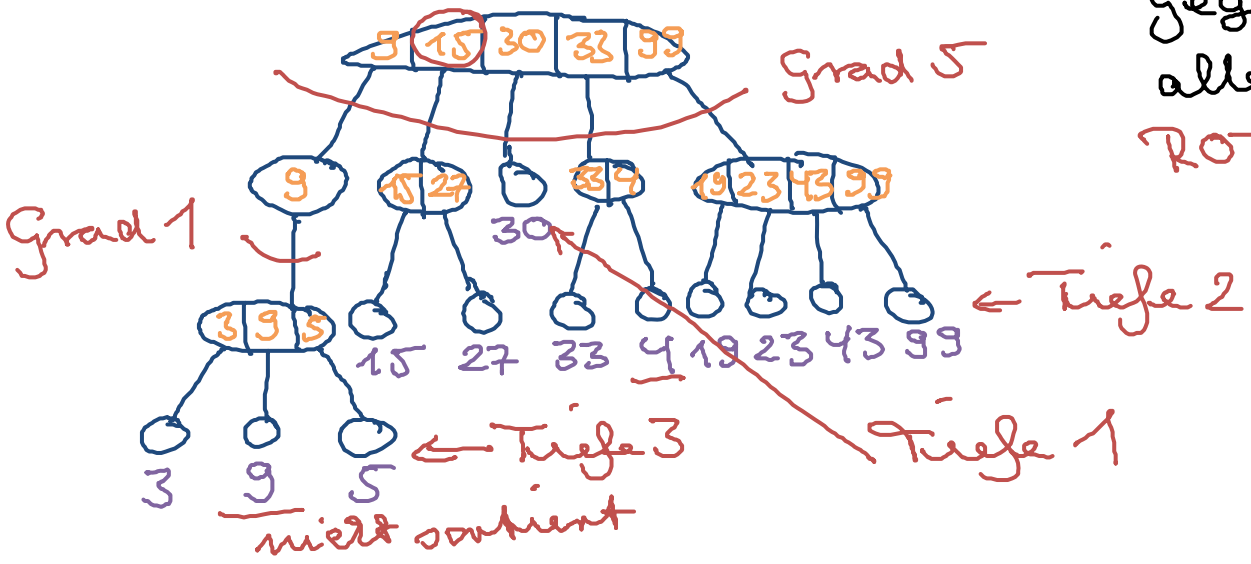
(a,b)-Bäume 2/8

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



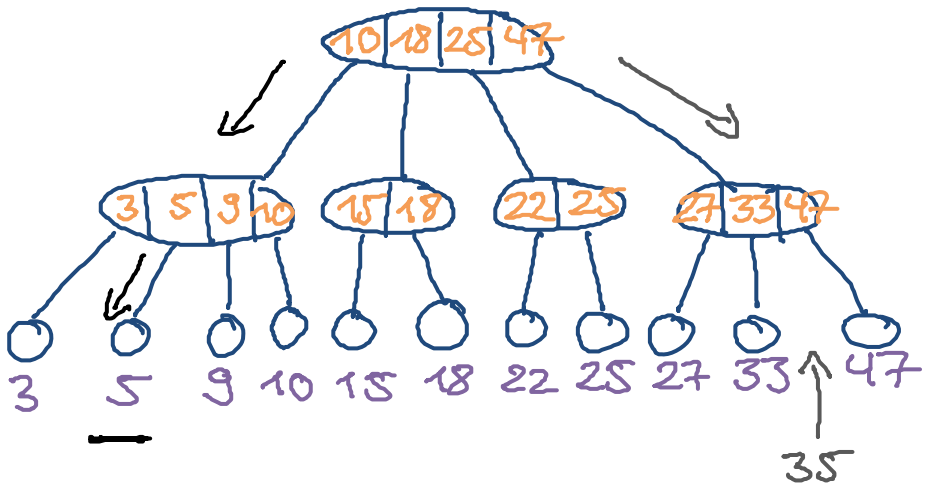
(2,4)-Baum der Tiefe 2 alle Knoten zwischen 2 und 4 Kindern

für einen (2,4)-Baum Gegenbeispiel, alle Fehler in ROT markiert



(a,b)-Bäume 3/8

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



Lookup (5)

Lookup (35)

(a,b)-Bäume 4/8

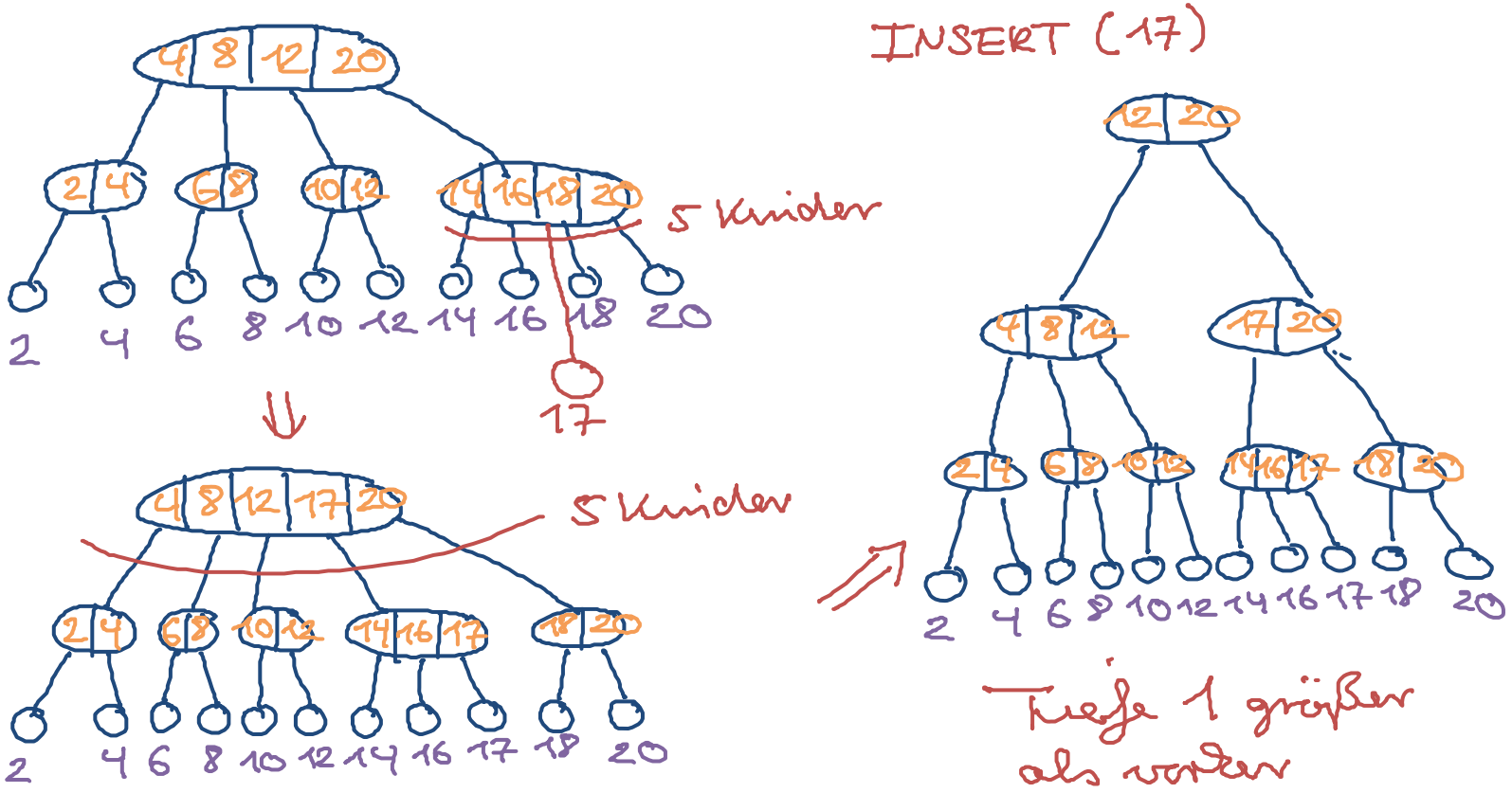
dafür brauchen wir gerade $b \geq 2a-1$

UNI FREIBURG

- 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 $\text{ceil}(b/2) \geq a$ und $\text{floor}(b/2) + 1 \geq a$
  - 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 5/8

- 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**

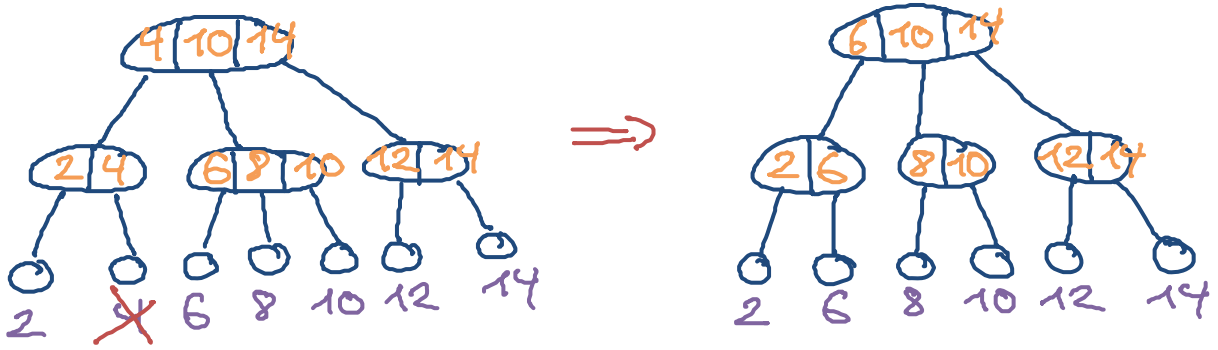
(a,b)-Bäume 8/8

Falls Knoten nach so was <a aber >1 Kinder hat, lässt man das einfach so (siehe Folie 5: Knoten darf <a Kinder haben)

- Entfernen eines Elementes (remove), Beispiele:

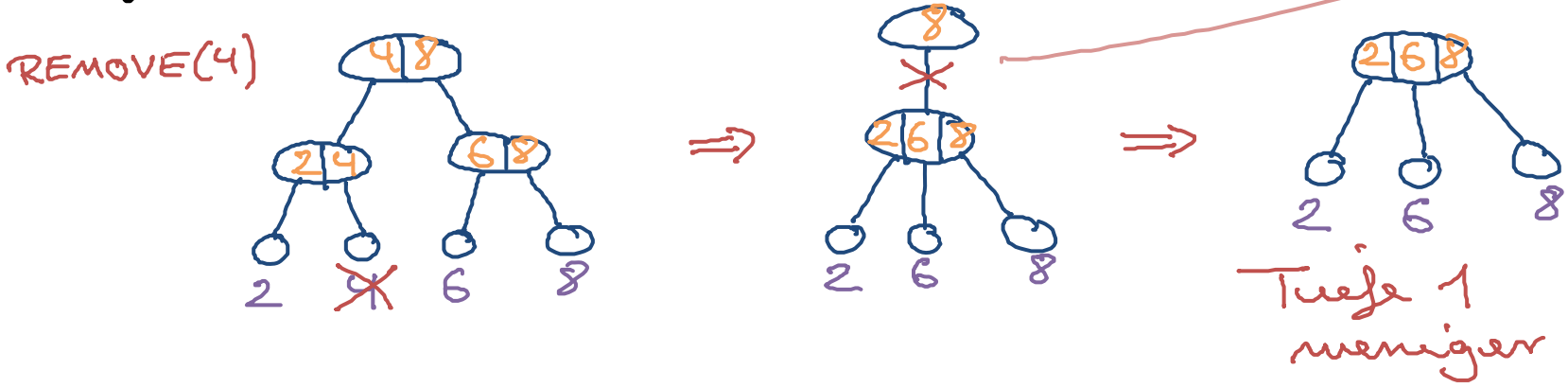
FALL 1 : KLAUEN

REMOVE(4)



FALL 2 : VERSCHMELZEN

REMOVE(4)



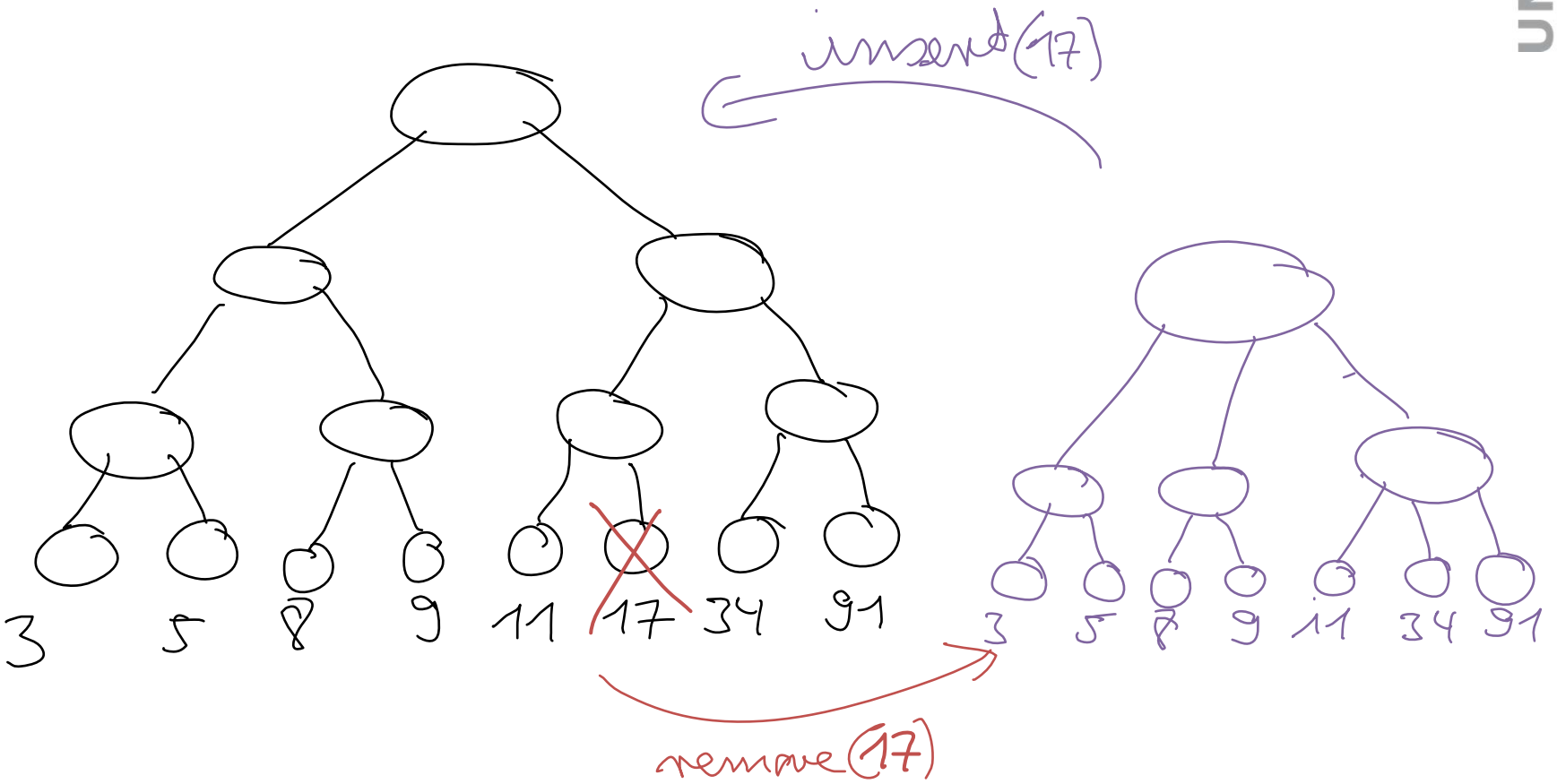
Analyse (a,b)-Bäume

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 **$b \geq 2a$**
  - Gegenbeispiel für (2,3)-Bäume, Analyse für (2,4)-Bäume

Gegenbeispiel für (2,3)-Bäume



Man kann zeigen:
Wenn $b = 2a - 1$, dann gibt es eine Folge von n Operationen mit Kosten $\Omega(n \cdot \log n)$.

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

- Terminologie
  - Wir betrachten eine Folge von n Operationen
  - Seien c_i die Kosten = Laufzeit der i-ten Operation
  - Sei Φ_i das Potential des Baumes nach der i-ten Operation
    = die Anzahl der Knoten mit Grad genau 3
    - Φ_0 = Potential am Anfang := 0
- Lemma
  - Es gilt $c_i \leq A \cdot (\Phi_i - \Phi_{i-1}) + B$ für irgendwelche $A, B > 0$
  - Daraus folgt dann $\sum_{i=1..n} c_i = O(n)$

$\Phi_1 - \Phi_0 + \Phi_2 - \Phi_1 + \Phi_3 - \Phi_2 + \dots$

= leerer Baum

für $A=1$ und $B=0$: $\Phi_i = \Phi_{i-1} + c_i$

$\Leftrightarrow \Phi_i - \Phi_{i-1} \geq \frac{c_i - B}{A} \Leftrightarrow \Phi_i = \Phi_{i-1} + \Omega(c_i)$

BEWEIS: $\sum_{i=1}^{n} c_i \leq \underbrace{A \cdot (\Phi_1 - \Phi_0) + B}_{c_1 \leq} + \underbrace{A \cdot (\Phi_2 - \Phi_1) + B}_{c_2 \leq} + \dots + \underbrace{A \cdot (\Phi_n - \Phi_{n-1}) + B}_{c_n \leq}$

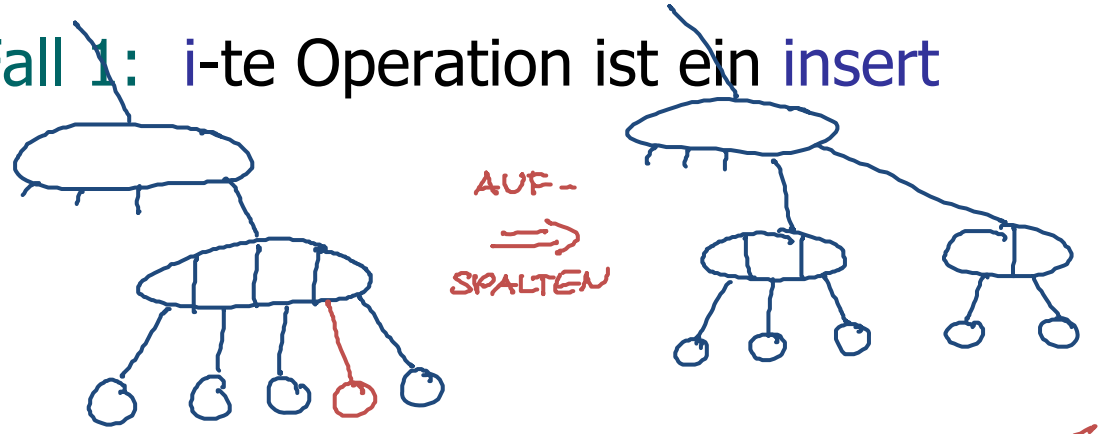
$= A \cdot (\Phi_n - \Phi_0) + B \cdot n$

$= A \cdot \Phi_n + B \cdot n = O(n)$

weil $\Phi_n \leq n$

Analyse (2,4)-Bäume 3/4

- Fall 1: i-te Operation ist ein insert



BEOBACHTUNG:
Für jedes Aufspalten hat man nachher einen Knoten vom Grad 3 mehr (vorher: 4, nachher: 3 und 2)

= vor dem Einfügen

$\Rightarrow$ m = # Aufspaltungen

Dann $c_i \leq A \cdot m + B$

und $\Phi_i = \Phi_{i-1} + m$ $\Rightarrow m = \Phi_i - \Phi_{i-1}$

$\Rightarrow$ Damit $c_i \leq A \cdot (\Phi_i - \Phi_{i-1}) + B$ ∎

Analyse (2,4)-Bäume 4/4

- Fall 2: i-te Operation ist ein remove

KLAUEN:

ein Knoten vom Grad 3 mehr oder weniger (je nach Grad von dem Knoten von dem man klaut)

VERSCHMELZEN:

ein Knoten vom Grad 3 mehr

INSGESAMT: $m \times$ VERSCHM. + höchstens 1 $\times$ KLAUEN

$\Rightarrow \quad c_i \leq A \cdot m + B'$

$\Rightarrow \quad c_i \leq A \cdot (\phi_i - \phi_{i-1}) + \underbrace{A + B'}_{=B}$

$\phi_i \geq \phi_{i-1} + m - 1$

$\Leftrightarrow \quad m = \phi_i - \phi_{i-1} + 1$

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)