Algorithmen und Datenstrukturen (für ESE) WS 2011 / 2012

Vorlesung 10, Dienstag, 17. Januar 2012
(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 Ü9 (Binärer Suchbaum)
  - Sie hatten recht: Teile der Aufgaben zum Caching waren zu unpräzise gestellt (Wie genau? Annahmen an B und M?)
- Balancierte Suchbäume
  - In der letzten VL haben wir gesehen, dass bei einem binären Suchbaum die Operationen insert und lookup im worst case Zeit proportional zur **Tiefe** des Baumes brauchen
  - Im Ü9 haben Sie gesehen, dass Ihr BinarySearchTree Tiefe $O(\log n)$ haben kann, aber im worst case auch $\Theta(n)$
  - Also schauen wir heute, dass er immer Tiefe $O(\log n)$ hat

Ihre Erfahrungen mit dem Ü9 (BST)

UNI FREIBURG

- Zusammenfassung von Ihrem Feedback **Stand 17.1 15:12**
  - Lief bei den meisten gut + wieder deutlich weniger Arbeit
  - Übungsblatt hat den meisten Spaß gemacht
  - Vorlesung: alles rausgeholt was der Stift zuließ
  - Blöde Fehler haben die meiste Zeit gekostet
  - Meiste Zeit für Entknoten des Gehirns drauf gegangen
  - Lösung am nächsten Morgen unter der Dusche
  - Ehrgeiz eine schöne toString Methode zu schreiben
  - Die war aufwändig, warum keine Punkte dafür?
  - Man hätte Rekursion nochmal erklären sollen
  - Kompliment für die Qualität der Aufzeichnungen + schnell online

Balancierte Bäume

UNI FREIBURG

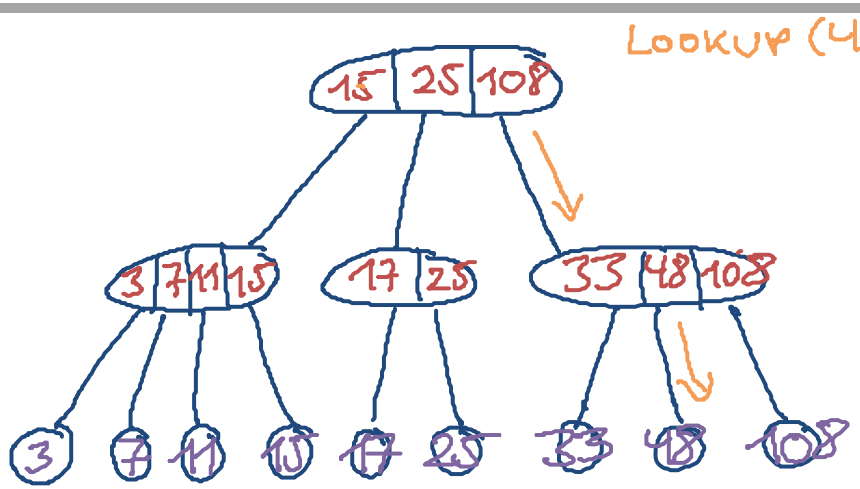
- Motivation
  - Mit BinarySearchTree hatten wir lookup und insert in Zeit O(depth), wobei depth = Tiefe des Baumes
  - Wenn es gut läuft ist depth = O(log n)
    - z.B. wenn die Schlüssel zufällig gewählt sind
  - Wenn es schlecht läuft ist depth = Θ(n)
    - z.B. wenn der Reihe nach 1, 2, 3, ... eingefügt 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

UNI FREIBURG

- Wie erreicht man immer Tiefe $O(\log n)$?
  - Es gibt Dutzende verschiedener Verfahren dafür
  - Wie schauen uns heute (a,b)-Bäume an
  - 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 $b \geq 2a - 1$ Begründung folgt
  - An den inneren Knoten steht für jedes Kind der größte Schlüssel in dem Unterbaum dieses Kindes

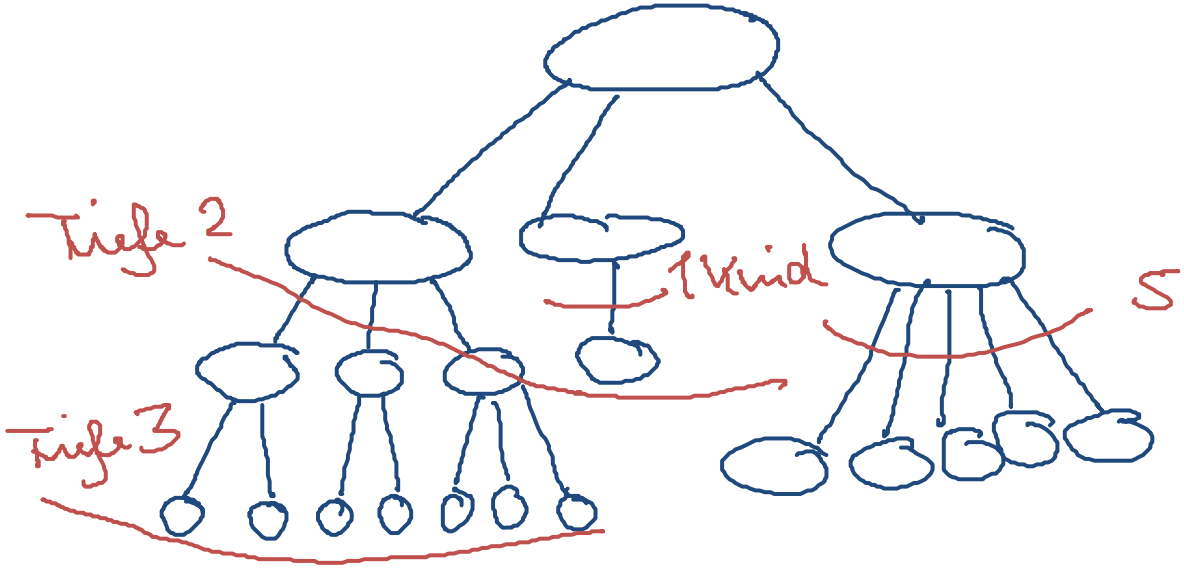
(a,b)-Bäume — Beispiele



(2,4)-Baum

Schlüssel stehen nur an den Blättern!

Die Schlüssel an den inneren Knoten dienen nur zum Suchen / Finden



KEIN (2,4)-Baum

(ohne Schlüssel, die spielen für das Bsp. keine Rolle)

(a,b)-Bäume — Lookup

UNI FREIBURG

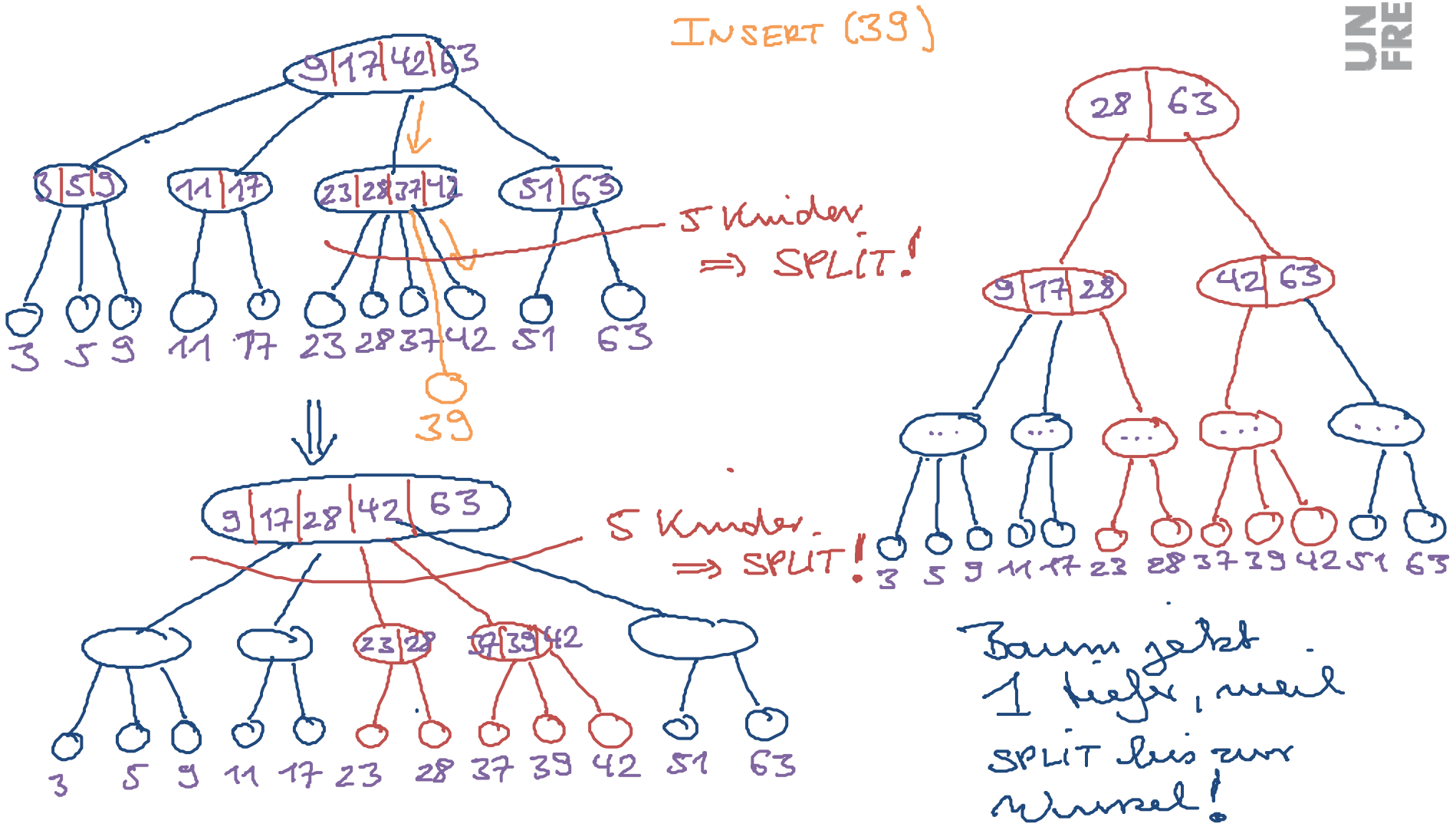
- Im Prinzip genauso wie beim BinarySearchTree
  - Suche von der Wurzel abwärts
  - Die Schlüssel an den inneren Knoten weisen den Weg

(a,b)-Bäume — Insert

UNI FREIBURG

- Einfügen eines Elementes / Schlüssels
  - 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 wird 1 tiefer

(a,b)-Bäume — Insert



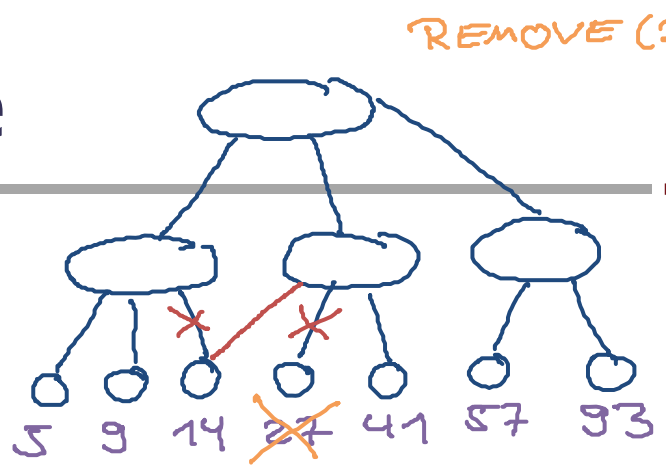
(a,b)-Bäume — Remove

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

das ist gerade die Bedingung von der Definition auf Folie 5

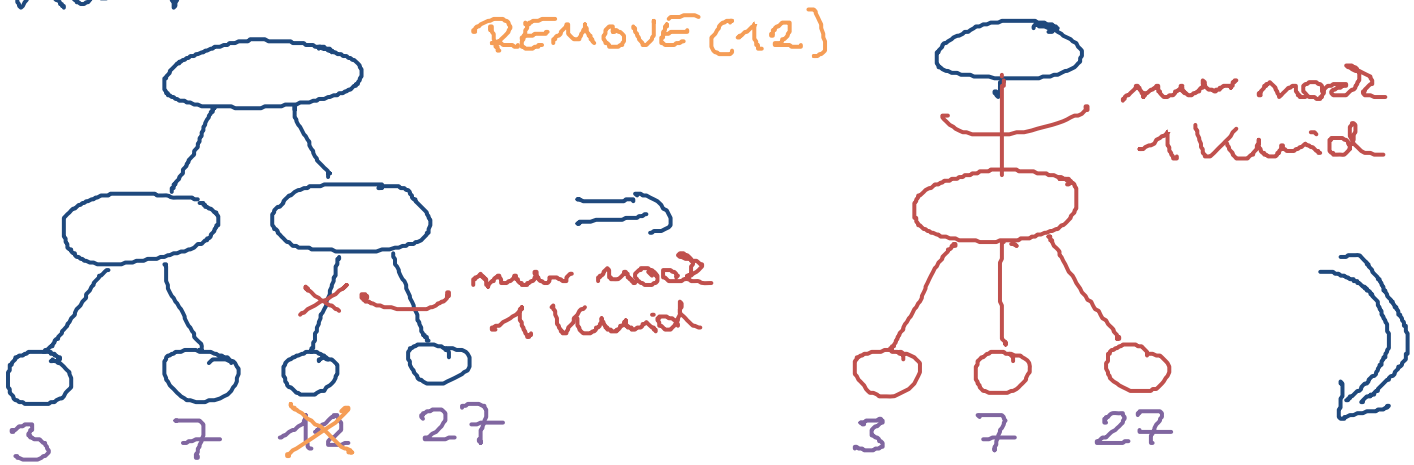
$a + a - 1 = 2a - 1 \leq b$

(a,b)-Bäume — Remove

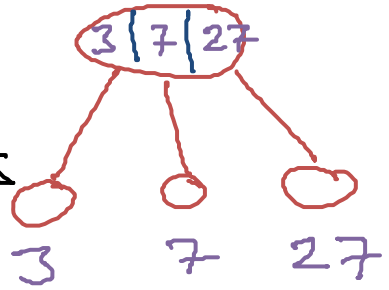


Einfacher Fall: KLAUEN

Komplizierterer Fall: VERSCHMELZEN

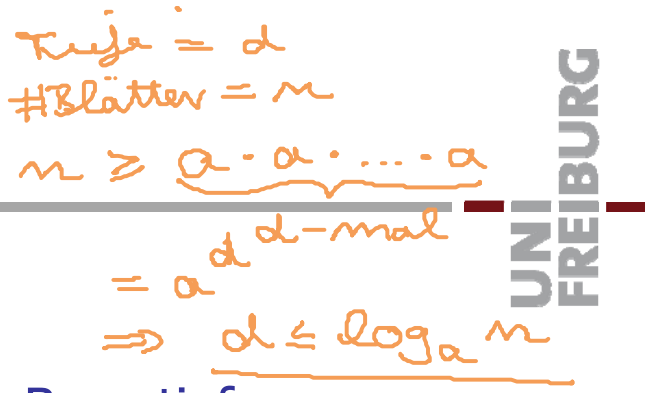


ACHTUNG: Klauen kann auch weiter oben im Baum noch passieren (als letzter Schritt nach einer Reihe von VERSCHMELZUNGS Operationen)



Tiefe 1 weniger!

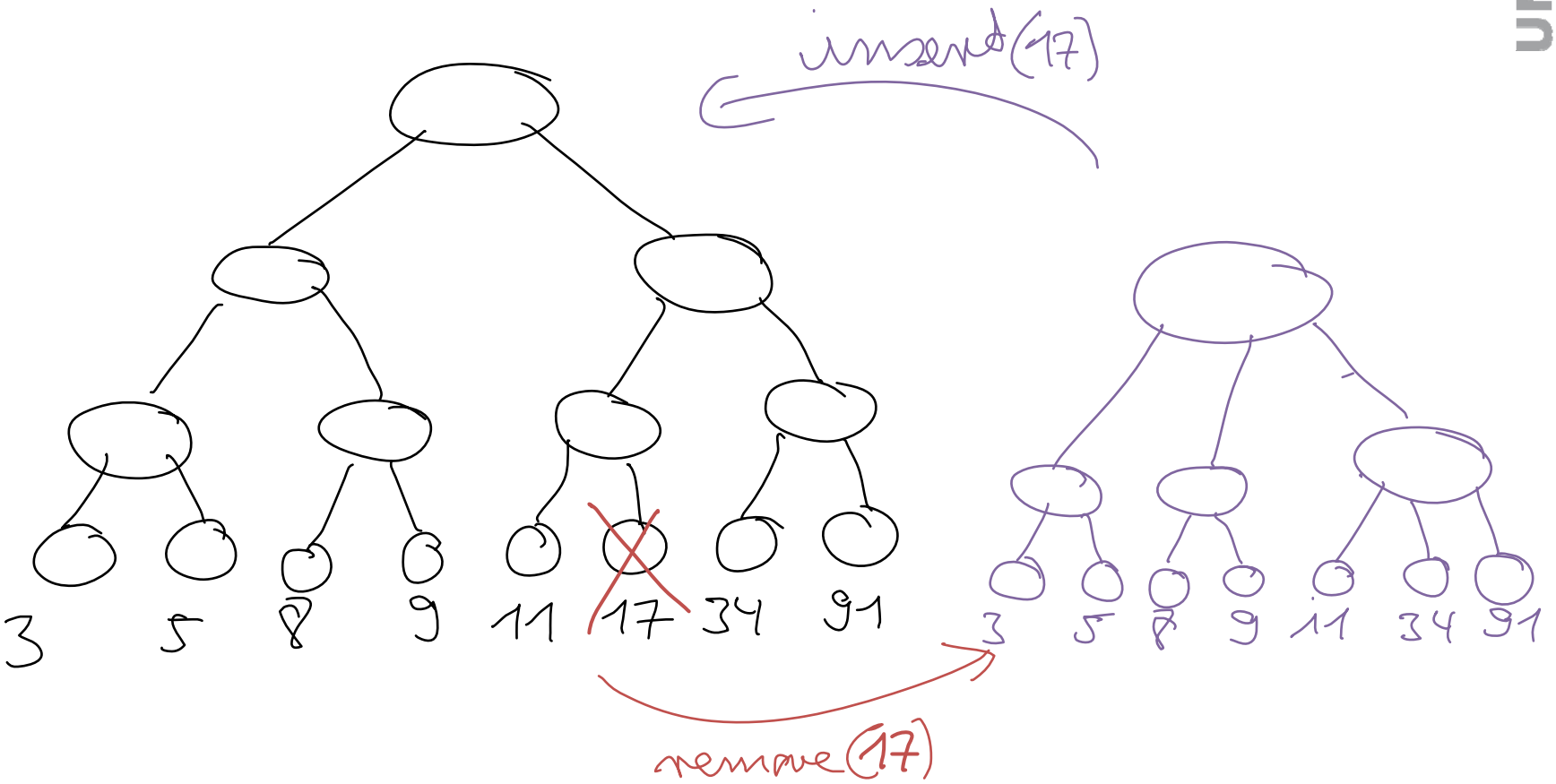
(a,b)-Bäume — Komplexität



- Kosten von lookup, insert, remove ...
  - ... sind alle $O(b \cdot depth)$, wobei depth = Baumtiefe
  - Weil jeder Knoten, außer evtl. der Wurzel, mindestens a Kinder hat, ist $depth = O(\log_a n)$
  - Bei genauerem Hinsehen fällt auf
    - Die Operation lookup braucht immer Zeit ~ depth
    - 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 $b \geq 2a$
  - Auf der nächsten Folie ein Gegenbeispiel für $a = 2$ und $b = 3$
  - Dann die Analyse für $a = 2$ und $b = 4$

(2,3)-Bäume — Gegenbeispiel



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

(2,4)-Bäume — Analyse 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: wir müssen formalisieren, dass der Baum nach einer teuren Operation in einem Zustand ist, so dass es lange dauert, bis wieder eine Operation teuer wird
    - das ist ähnlich wie bei den dynamischen Feldern: Reallokation ist teuer, aber danach dauert es, bis wieder realloziert werden muss, und wenn man es richtig macht, sind die Kosten im Durchschnitt konstant

(2,4)-Bäume — Analyse 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

für $A=1$, $B=0$
$\Phi_i \geq \Phi_{i-1} + c_i$ ✓

↗ die kann man so wählen, dass die Analyse gut klappt.

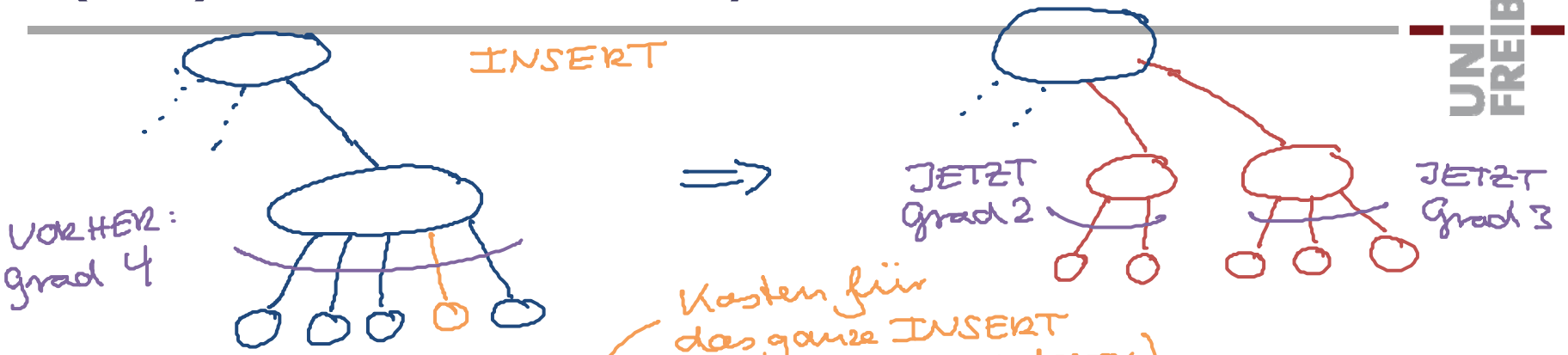
- Lemma
  - Es gilt $c_i \leq A \cdot (\Phi_i - \Phi_{i-1}) + B$ für irgendwelche $A, B > 0$
  - Daraus folgt dann $\sum_{i=1..n} c_i = O(n)$

$\Phi_n \leq n$; $\Phi_0 = 0$

$$\sum_{i=1}^{n} c_i \leq A \cdot (\Phi_1 - \Phi_0) + B + A \cdot (\Phi_2 - \Phi_1) + B + \dots + A \cdot (\Phi_n - \Phi_{n-1}) + B$$

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

(2,4)-Bäume — Analyse 3/4



Kosten für das ganze INSERT (ohne das LOOKUP davor)

m = Anzahl der SPLIT Operationen

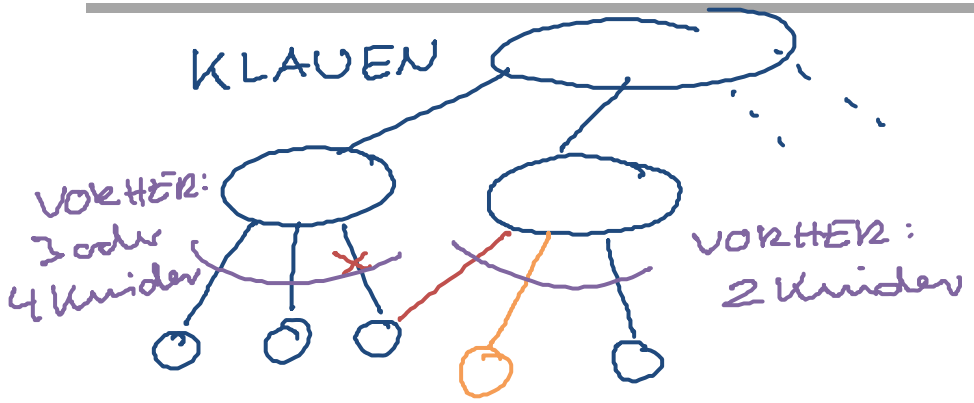
Dann 1. $c_i \leq A \cdot m + B$ für geeignete $A, B > 0$

2. $\Phi_i = \Phi_{i-1} + m$

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

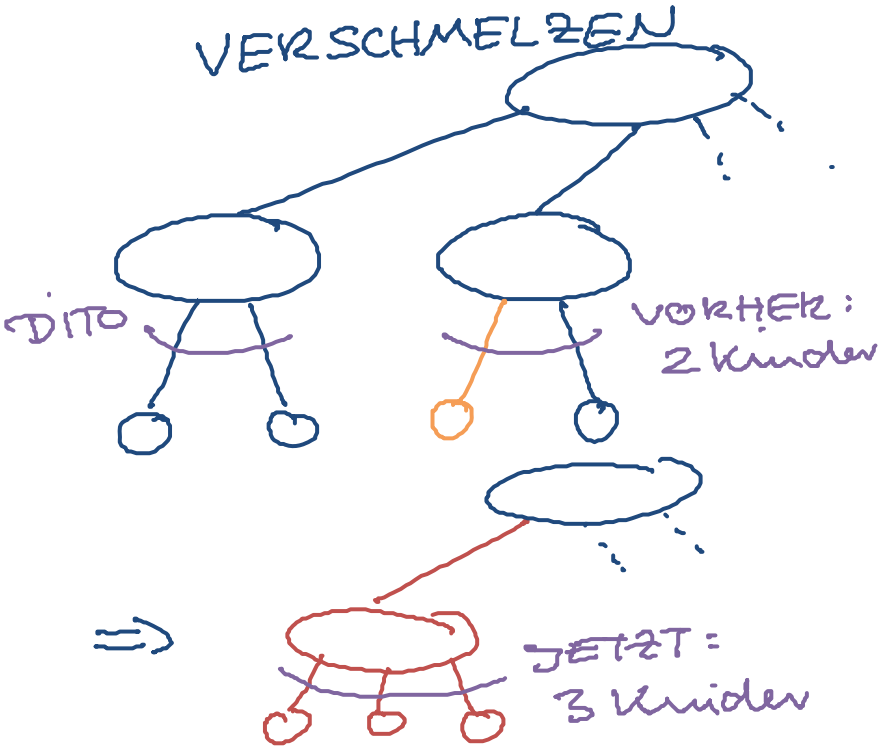
$\Rightarrow c_i \leq A \cdot (\Phi_i - \Phi_{i-1}) + B$

(2,4)-Bäume — Analyse 4/4



REMOVE

Potential wird
ein Knoten oder ein
niedriger



m = Anzahl der VERSCHMELZUNGS Operationen

$c_i \leq A' \cdot m + B'$ mit A', B' so dass es passt

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

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

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

$= A \cdot (\Phi_i - \Phi_{i-1}) + B$

Literatur / Links

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

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