Informatik II: Algorithmen und Datenstrukturen SS 2015

Vorlesung 8b, Mittwoch, 17. Juni 2015
(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

- Inhalt
  - (a, b)-Bäume … Prinzip + viele Beispiele
  - (2, 4)-Bäume …. amortisierte Analyse
  - Ü8, Aufgabe 2: Beweis aus der Vorlesung für (4, 9)-Bäume anpassen

(a,b)-Bäume 1/11

UNI FREIBURG

- Motivation
  - Mit dem einfachen binären Suchbaum von gestern haben 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

    Das Problem hatten wir auch schon beim Hashing ... die Lösung da waren universelle Klassen von Hashfunktionen

(a,b)-Bäume 2/11

UNI FREIBURG

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

    AVL-Bäume

    AA-Bäume

    Rot-Schwarz-Bäume

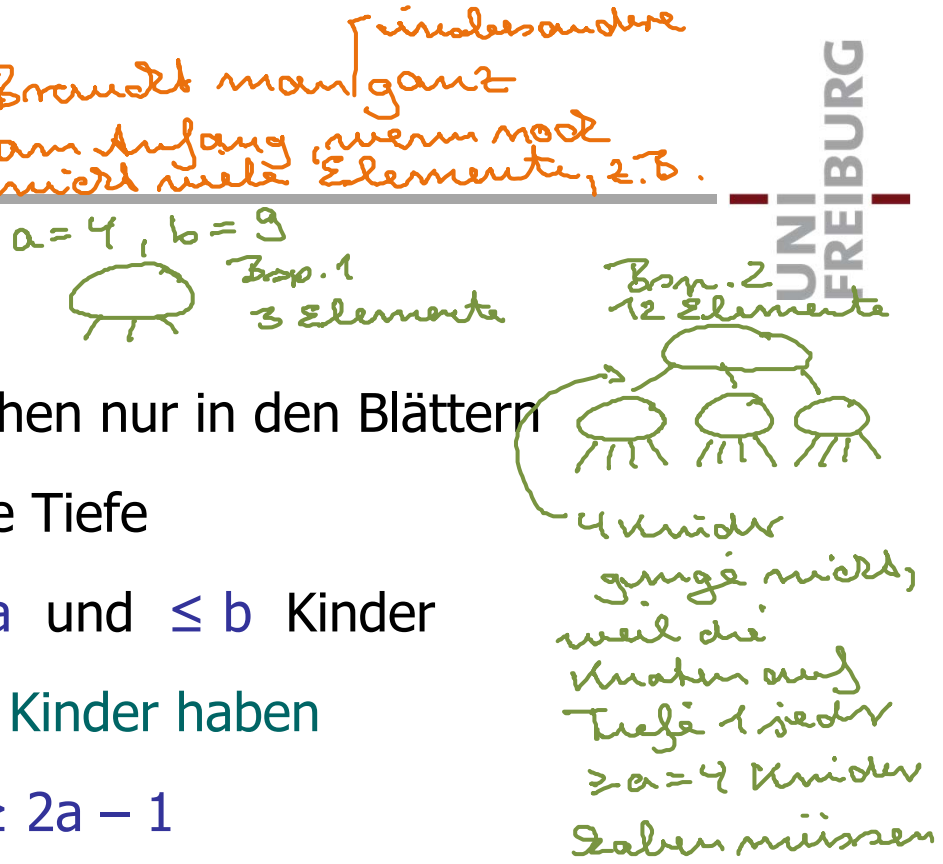
    Splay trees

    Treaps

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

    Die sind intuitiv, einfach, praktisch und schmecken gut

(a,b)-Bäume 3/11



- 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

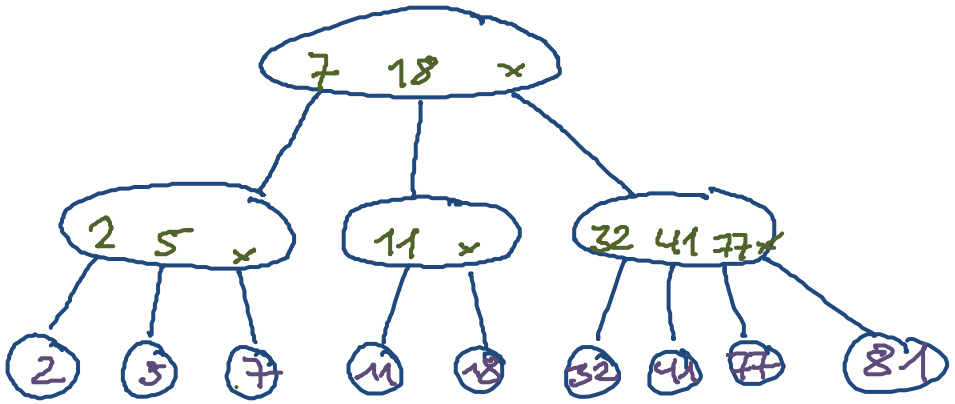
    Nur die Wurzel darf weniger Kinder haben
  - Wir verlangen $a \geq 2$ und $b \geq 2a - 1$

    Warum sehen wir gleich
  - An jedem inneren Knoten steht für jedes Kind außer dem rechtesten der größte Schlüssel in dessen Unterbaum

    Den größten Schlüssel aus dem rechtesten Unterbaum braucht man nicht + es gäbe auch Effizienzprobleme … warum siehe gleich

(a,b)-Bäume 4/11

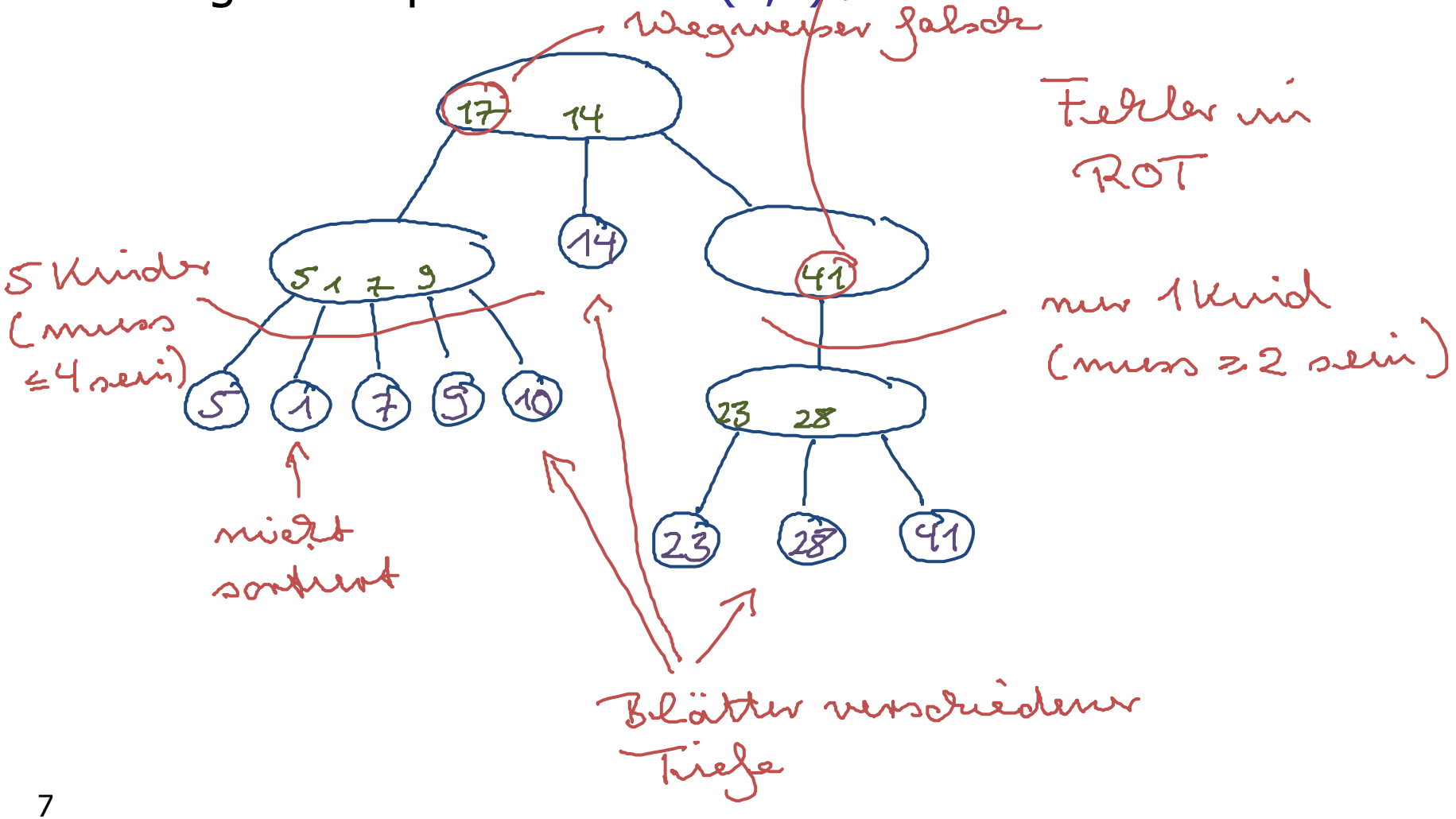
- Positivbeispiel für einen (2,4)-Baum



x = da steht NIX

(a,b)-Bäume 5/11

- Negativbeispiel für einen (2,4)-Baum



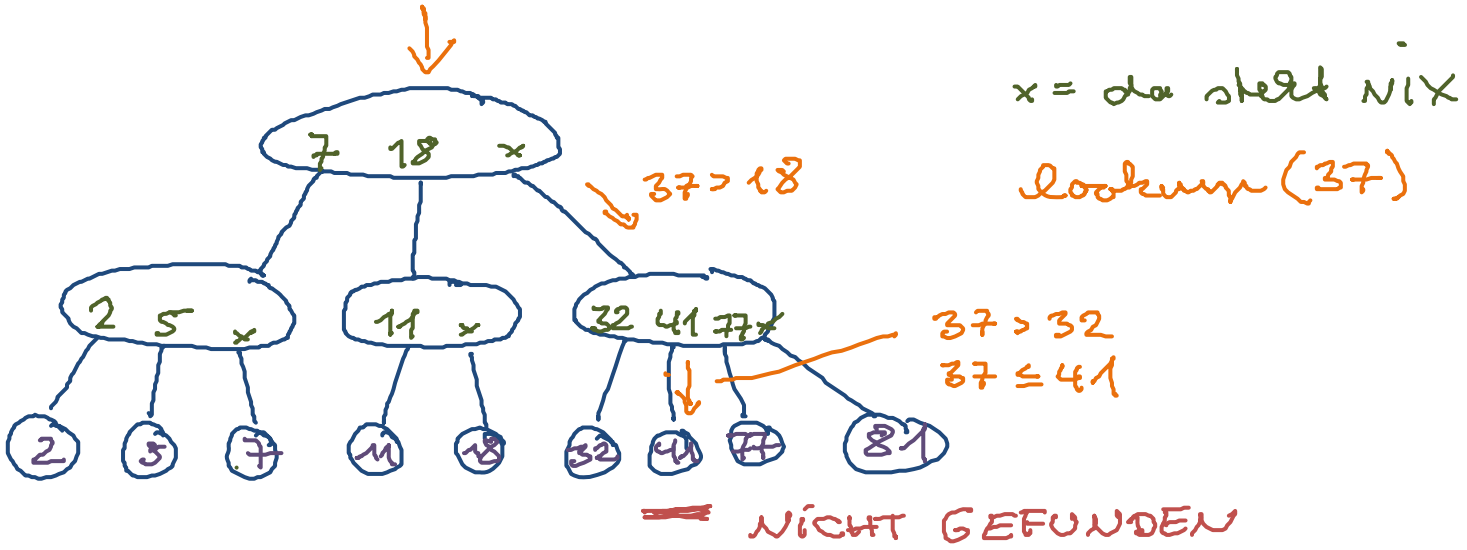
(a,b)-Bäume 6/11

UNI FREIBURG

- Die Operation **lookup**
  - Im Prinzip genau so wie beim binären Suchbaum

    Suche von der Wurzel abwärts, und die Schlüssel an den inneren Knoten weisen den Weg

    Bei einem Knoten mit k Kindern reichen k – 1 "Wegweiser"



(a,b)-Bäume 7/11

- Die Operation **insert**
  - Finde die Stelle, wo der neue Schlüssel einzufügen ist, und füge dort ein neues Blatt ein

    Der Elternknoten kann jetzt b + 1 Knoten haben

    Falls das der Fall ist, **Aufspalten** des Elternknotens in zwei Knoten mit $\lceil b/2 \rceil$ bzw. $\lfloor b/2 \rfloor + 1$ Kinder

    Für $b \geq 2a - 1$ ist $\lceil b/2 \rceil \geq a$ und $\lfloor b/2 \rfloor + 1 \geq a$

    Der Großelternknoten kann jetzt b + 1 Kinder haben

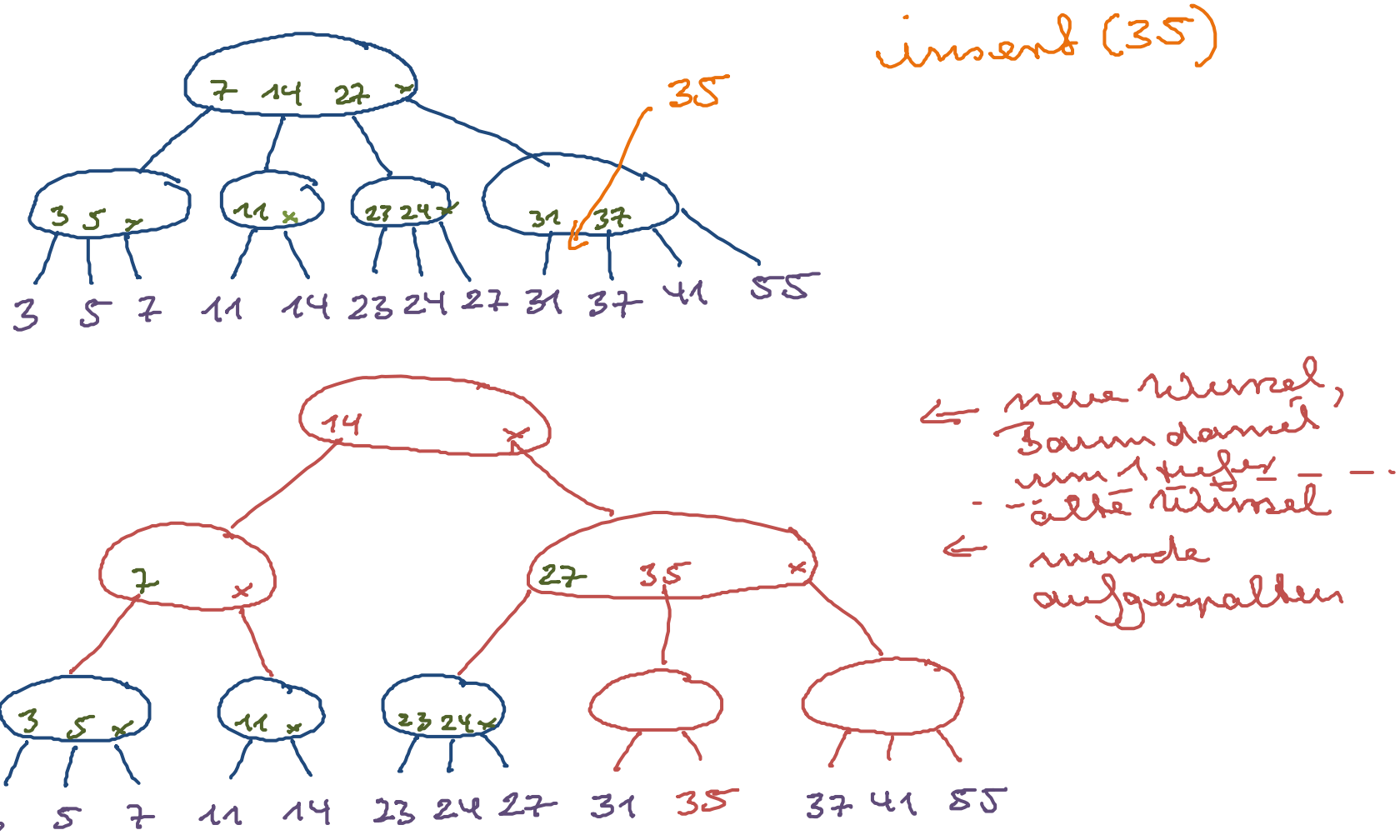
    Dann spalten wir den auf dieselbe Weise auf ... usw.

  - Wenn das bis zur Wurzel geht, spalten wir auch diese auf und erzeugen einen neuen Wurzelknoten

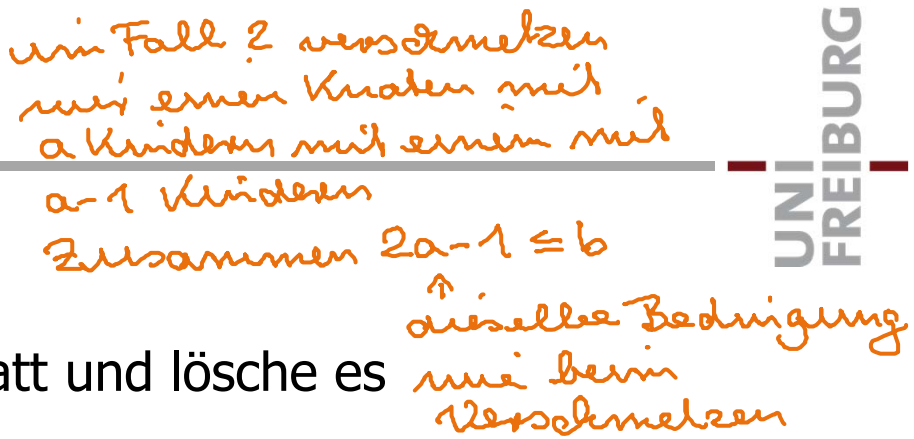
    Dann (und nur dann) wird der Baum um **1** tiefer

(a,b)-Bäume 8/11

- Die Operation **insert** ... Beispiel



(a,b)-Bäume 9/11



- Die Operation **remove**
  - Finde das zu entfernende Blatt und lösche es

    Der Elternknoten kann jetzt a – 1 Kinder haben

    Fall 1: Eines der anderen Kinder des Elternknoten hat > a Kinder → dann eins von da **klauen** und fertig

    Fall 2: Sonst **verschmelzen** wir den Elternknoten mit einem seiner Geschwister

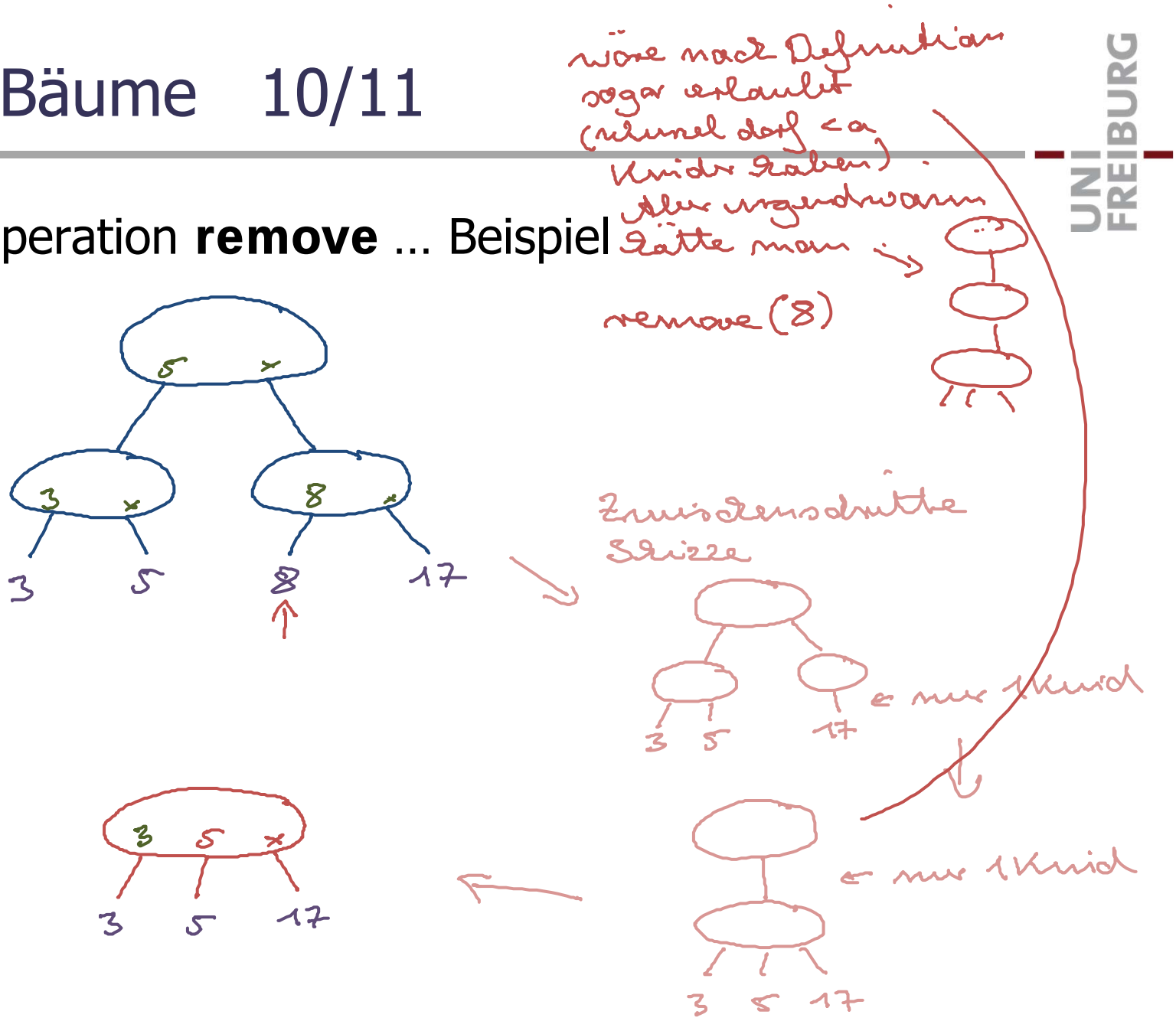
    Der Großelternknoten kann jetzt a – 1 Kinder haben

    Damit verfahren wir dann genauso … usw.
  - Wenn das bis zur Wurzel geht und die am Ende nur noch ein Kind hat, mache dieses Kind zur neuen Wurzel

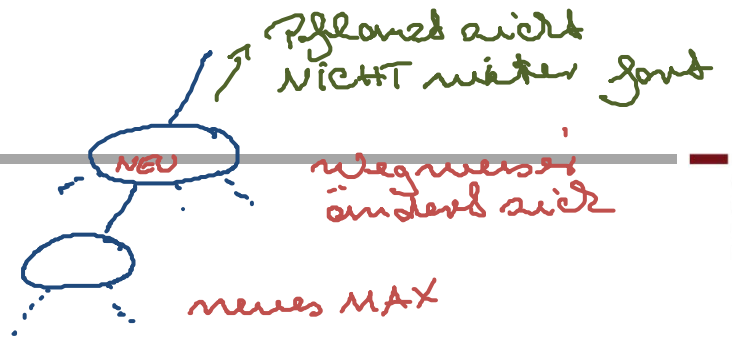
    Dann (und nur dann) wird der Baum um **1** weniger tief

(a,b)-Bäume 10/11

- Die Operation **remove** … Beispiel



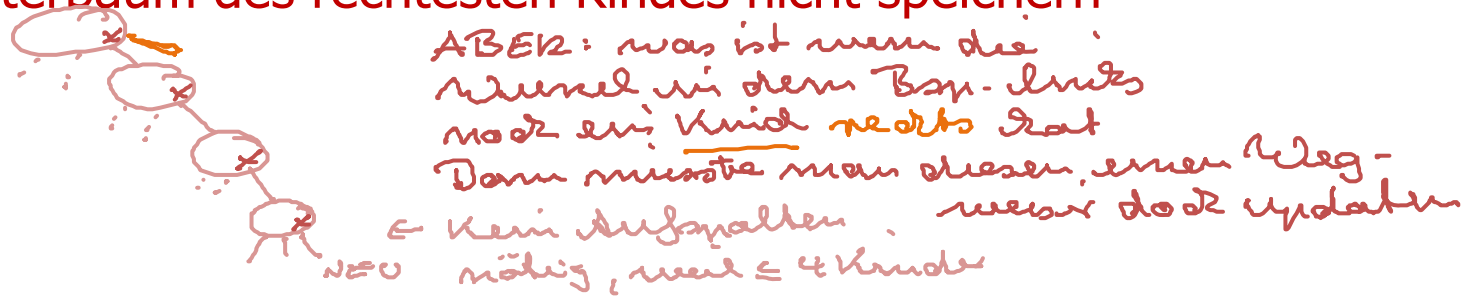
(a,b)-Bäume 11/11



- Update der "Wegweiser"
  - Sowohl bei einem insert wie bei einem remove ändern sich auch einige Wegweiser
  - Insbesondere an den Knoten, die Teil von der "Kette" von Aufspaltungen / Verschmelzungen bzw. vom Klauen sind

    Da haben wir eh schon konstante Kosten pro Knoten
  - Ansonsten ist höchstens noch der Elternknoten des Knotens am Ende einer solchen Kette betroffen

    Das gilt aber nur, weil wir den größten Wert im Unterbaum des rechtesten Kindes nicht speichern



Analyse (2,4)-Bäume 1/8

UNI FREIBURG

- Laufzeit 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 $d \leq 1 + \log_a n = O(\log n)$
  - Bei genauerem Hinsehen fällt auf

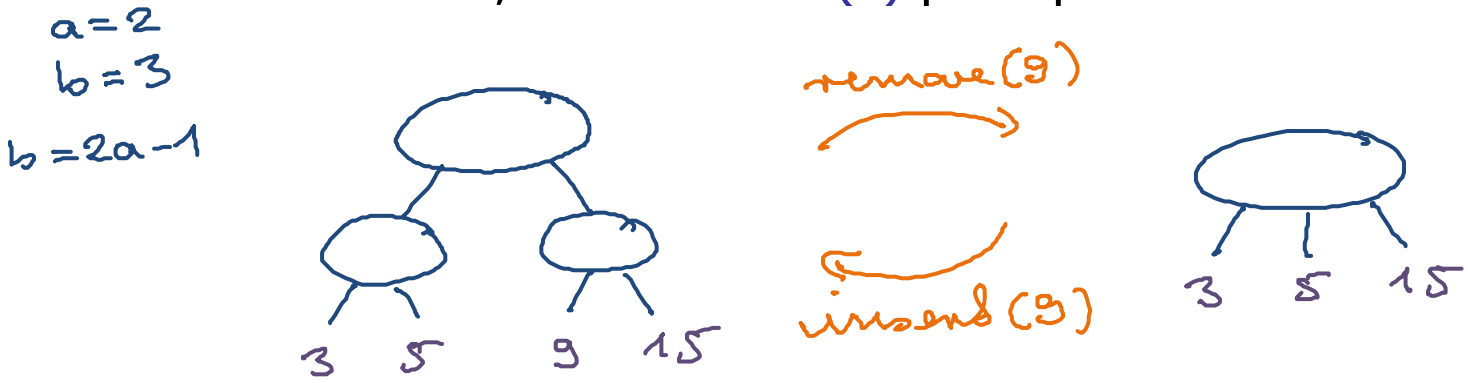
    Die Operation lookup braucht immer Zeit $\Theta(d)$

    Aber insert und remove gehen "oft" in Zeit $O(1)$

    Nur im worst case müssen alle Knoten auf dem Weg zur Wurzel geteilt / verschmolzen werden
  - Das wollen wir jetzt genauer analysieren ...

Analyse (2,4)-Bäume 2/8

- Wir brauchen jetzt $b \geq 2a$
  - Für $b = 2a - 1$ kann man eine Folge von Operationen konstruieren, mit Kosten $\Theta(d)$ pro Operation



Analyse (2,4)-Bäume 3/8

UNI FREIBURG

- Satz
  - Für $b \geq 2a$ ist die Laufzeit für eine beliebige Abfolge von n insert oder remove Operationen $O(n)$

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

    Übungsblatt 8, Aufgabe 2: Beweis für $a = 4$, $b = 9$

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

Analyse (2,4)-Bäume 4/8

UNI FREIBURG

- Beweis, Intuition
  - **Beobachtung:** wann ist ein insert oder remove teuer:

    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 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

Analyse (2,4)-Bäume 5/8

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

    $\Phi_0 := 0$ (Potenzial am Anfang, für den leeren Baum)
- Mastertheorem aus Vorlesung 6b, Folie 15
  - 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 6/8

- **Fall 1:** i-te Operation ist ein insert
  - Pro Aufspaltung erhöht sich das Potenzial um 1

    Vorher Grad 4, nachher Grad 2 und Grad 3
  - An dem Knoten, an dem die Kette von Aufspaltungen endet, kann sich das Potenzial um 1 verringern

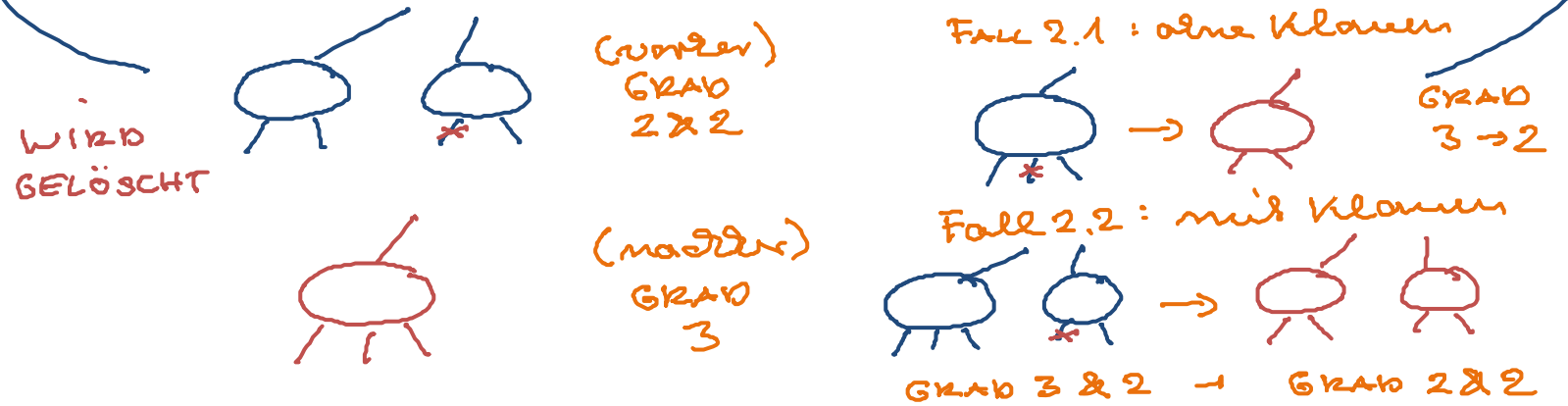
    Wenn vorher Grad 3 und anschließend Grad 4
  - Also $\Phi_i - \Phi_{i-1} \geq m - 1$, mit m = Anzahl Aufspaltungen

Analyse (2,4)-Bäume 7/8

- **Fall 2:** i-te Operation ist ein remove
  - Pro Verschmelzung erhöht sich das Potenzial um 1

    Vorher Grad 2 und 2, nachher Grad 3
  - An dem Knoten, an dem die Kette von Verschmelzungen endet, kann sich das Potenzial um 1 verringern

    Wenn vorher Grad 3 und anschließend Grad 2 (entweder der Knoten selber, oder der Nachbar von dem man klaut)
  - Also $\Phi_i - \Phi_{i-1} \geq m - 1$, mit m = Anzahl Verschmelzungen



Analyse (2,4)-Bäume 8/8

- Zusammenfassung Analyse
  - In beiden Fällen gilt also $\Phi_i - \Phi_{i-1} \geq m - 1$

    Wobei m = Anzahl Aufspaltungen bzw. Verschmelzungen
  - Daraus folgt $T_i \leq A \cdot (\Phi_i - \Phi_{i-1}) + B$

    Nochmal zur Intuition: das heißt, wenn es teuer wird, dann erhöht sich das Potenzial entsprechend
  - Aus dem Master-Theorem folgt dann $\sum_{i=1..n} T_i = O(n)$

    Also amortisiert / im Durchschnitt konstante Laufzeit

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

$T_i \leq A' \cdot m + B'$ Konstante Kosten pro Verschmelzung / Aufspaltung + konstante Grundkosten

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

$=:A$

Literatur / Links

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

    7 Sorted Sequences [Kapitel 7.2 und 7.4]
  - In Wikipedia

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

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

    Es gibt immer noch keine deutsche Version … also wenn sich da jemand berufen fühlt, dann gerne !