Informatik II: Algorithmen und Datenstrukturen SS 2017

Vorlesung 8b, Mittwoch, **längster Tag** 2017

(Balancierte Suchbäume)

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

UNI FREIBURG

Blick über die Vorlesung heute

- Drumherum
  - Unser Gehirn — Warum so groß / nicht größer?
- Inhalt
  - Eindeutige Schlüssel — Was tun, wenn nicht der Fall?
  - (a, b)-Bäume — Prinzip + viele Beispiele
  - (2, 4)-Bäume — amortisierte Analyse
  - ÜB8, Bonusaufgabe: Potenzialfunktion für (3, 7)-Bäume

    Dafür gibt es 5 Bonuspunkte (= man braucht sie nicht für die volle Punktzahl, sie können einem aber dabei helfen, die volle Punktzahl zu erreichen)

    reine Verständnisaufgabe, wenig Schreiben erforderlich

Nicht eindeutige Schlüssel

UNI FREIBURG

- Umgang damit
  - In der VL8a hatten wir angenommen, dass alle Schlüssel (Keys) verschieden sind
  - Es ist nicht trivial, den Suchbaum so abzuändern, dass insert und remove nach wie vor in Zeit **O(d)** laufen

    Z.B. wenn alle Schlüssel gleich: unklar, ob man bei einem inneren Knoten nach links oder nach rechts gehen soll
  - In der Praxis zwei typische Lösungsansätze:

    1. Schlüssel **eindeutig machen** (zum Beispiel einen Zähler anhängen: Berlin.1, Berlin.2, Berlin.3, …)

    2. Pro Schlüssel **eine Menge von Values** speichern (z.B. in einer Liste oder in einem Feld) und nicht nur ein Value

Gehirn 1/5

- Warum so groß / nicht größer, Zitate von Ihnen
  - "Gehirn verbraucht sehr viel Energie (vor allem bei Mathe)"
  - "Zu schwer … würde beim Nachdenken zur Seite kippen"
  - "Würde nicht mehr dem Schönheitsideal entsprechen"
  - "Es wächst ja noch (meinen zumindest die Evolutionsforscher)"
  - "Noch mehr Dickschädel verträgt die Welt nicht"
  - "Allein die Tatsache, dass man diese Frage stellt, zeigt doch, dass wir selbst diese Größe nicht mal verdient haben"
  - "Es kommt nicht auf die Mächtigkeit der Teile eines Systems an, sondern auf die Mächtigkeit ihrer Interaktionen"

Gehirn 2/5

- Entwicklung der Größe (Durchschnittswerte)
  - Dinosaurier 100 cm^3
  - Homo habilis 600 cm^3
  - Homo erectus 1.000 cm^3
  - Homo neanderthalensis 1.500 cm^3
  - Homo sapiens 1.300 cm^3
  - Blauwal 7.000 cm^3

Gehirn 3/5

- Sinn und Zweck des Gehirns
  - Aus evolutionärer Sicht **alles andere als klar**
  - Unser Gehirn scheint **viel** mehr zu können als zum (guten) Überleben und Fortpflanzen notwendig ist
  - Es gibt dazu verschiedene Hypothesen, zum Beispiel

    Social Brain Hypothesis

    Management von sozialen Beziehungen (Menschen können bis zu 150), deutlich mehr als z.B. bei Schimpansen (ca. 50)

    Sexuelle Attraktivität

    Beeindrucken des anderen Geschlechts bzw. die Fähigkeit zu beurteilen, ob das auch beindruckend ist

Gehirn 4/5

- Der Pfau
  - Darwin: "The sight of a feather in a peacock's tail, whenever I gaze at it, makes me sick"

    Die Pfauenfedern sind hinderlich beim Fliegen und machen den Träger zu einer leichteren Beute für natürliche Feinde

    Aus evolutionärer Sicht also scheinbar eine schlechte Idee
  - Allerdings: herausragende Rolle bei der **Partnerwahl** … Studien zeigen starke Korrelation zwischen:

    1. Prächtigkeit des Gefieders
    2. Wahrnehmung sexueller Attraktivität beim Gegenüber
    3. Überlebensfähigkeit des zugehörigen Nachwuchses

Gehirn 5/5

- Fisherian Runaway … Ronald Fisher, 1930
  - Laut Fisher reichen schon folgende Voraussetzungen
    1. Das andere Geschlecht findet Merkmal X attraktiv
    2. Bei Merkmal X sind die Nachkommen überlebensfähiger
  - Das führt dann zu folgender positiver Feedback-Schleife
    1. Präferenz für Partner mit mehr X
    2. Bei den Nachkommen ist X übermäßig ausgeprägt
    3. Man braucht jetzt besonders viel X um attraktiv zu sein
    4. Es werden Partner mit noch mehr X ausgewählt, usw.
  - Ähnliches Prinzip wie bei sozialen Systemen mit bestimmten "Erfolgsmaßen" und der dann eintretenden "Inflation"

(a,b)-Bäume 1/11

UNI FREIBURG

- Motivation
  - Bei dem einfachen binären Suchbaum von gestern: lookup und insert in Zeit **Θ(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 = Θ(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

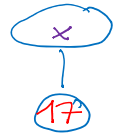
    …
  - Wir machen heute **(a,b)-Bäume**

    Die sind intuitiv, einfach, praktisch und biologisch abbaubar

(a,b)-Bäume 3/11

braucht man z.B. wenn der Baum sehr wenige Elemente hat

z.B. nur eins:



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

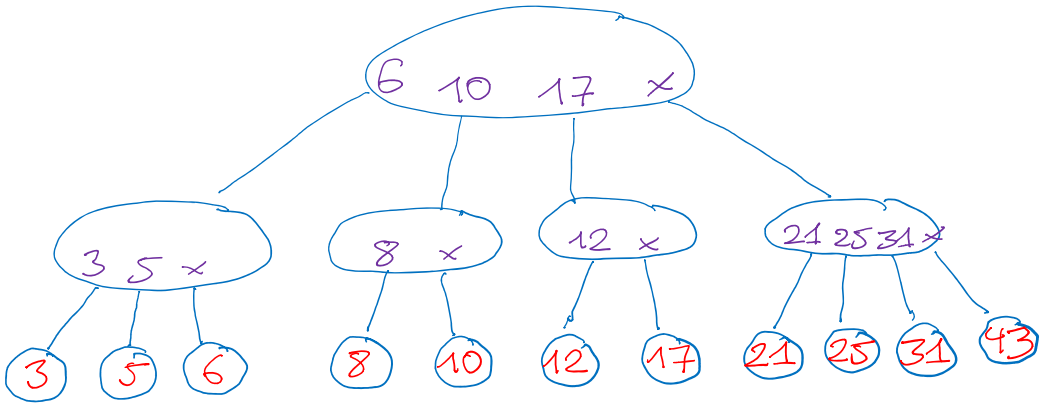
    Wurzel darf weniger Kinder haben, warum sehen wir gleich
  - Wir verlangen **a ≥ 2** und **b ≥ 2a – 1**

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

    Jeder "Wegweiser" gehört zu genau einem Blatt
  - Jedes Blatt weiß, wo sein zugehöriger Wegweiser steht

(a,b)-Bäume 4/11

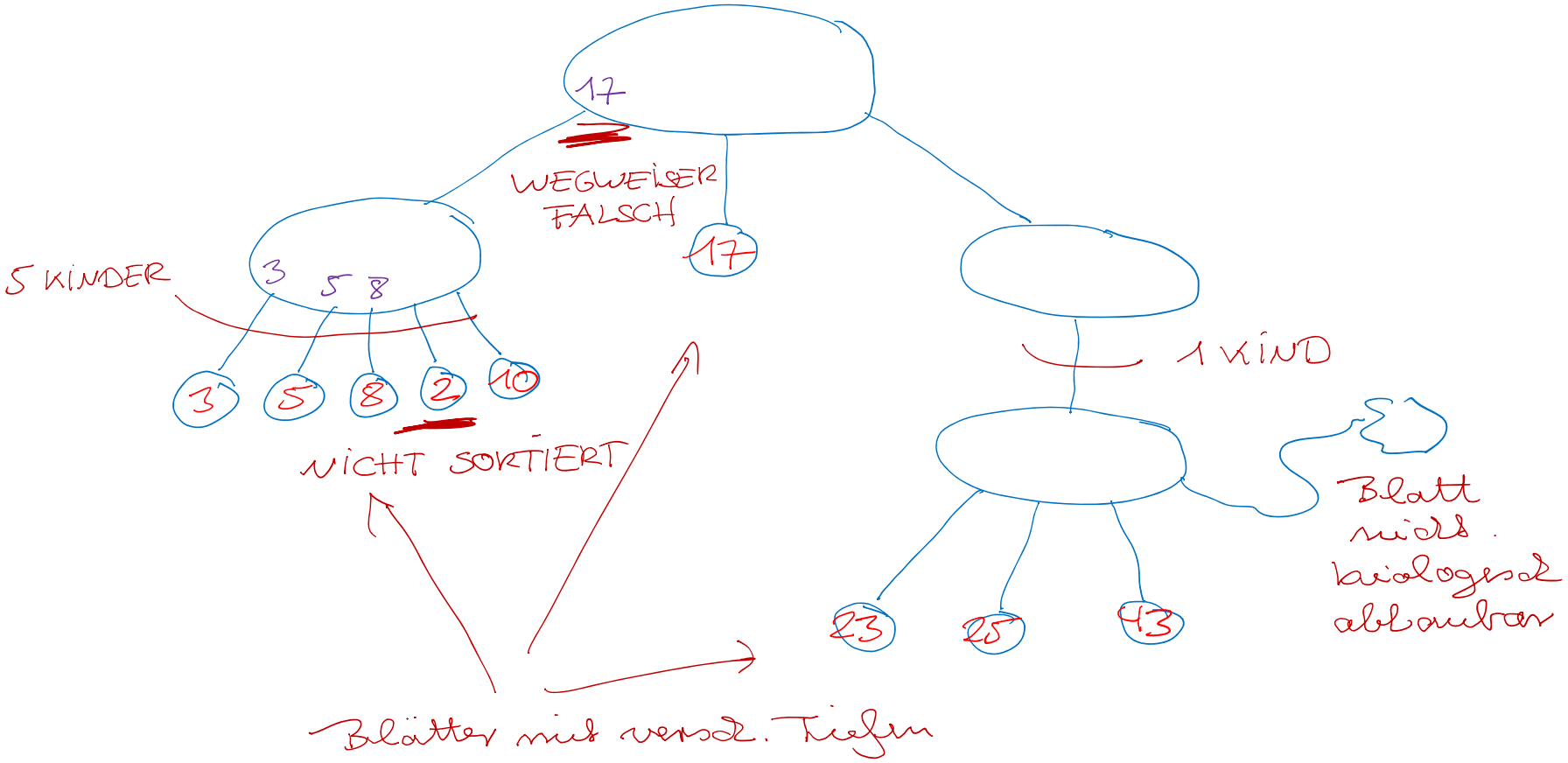
- Positivbeispiel für einen (2,4)-Baum



(a,b)-Bäume 5/11

alles was man nicht darf in ROT

- Negativbeispiel für einen (2,4)-Baum

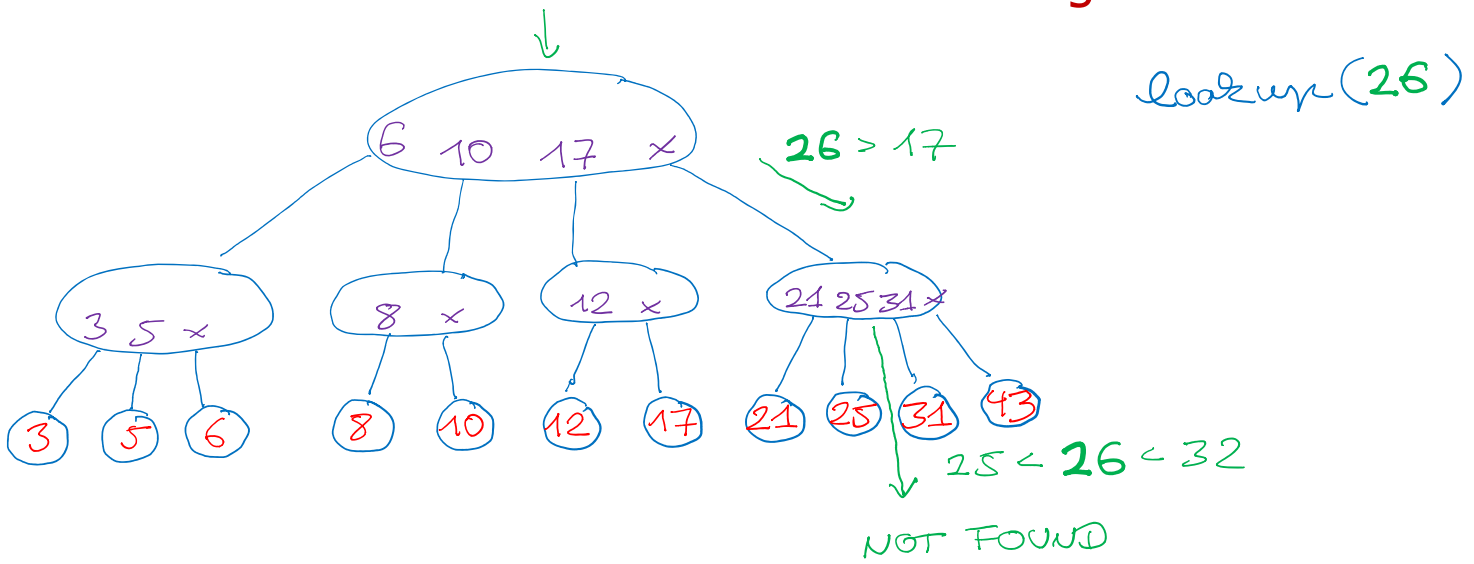


(a,b)-Bäume 6/11

- 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 Knoten mit k Kindern reichen k – 1 "Wegweiser"



(a,b)-Bäume 7/11

Beispiel $a = 2, b = 4$

- 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 (= 5) Knoten haben

    Falls das der Fall ist, **Aufspalten** des Elternknotens in zwei Knoten, einer mit $\lceil \mathbf{b/2} \rceil$ (= 2) und einer mit $\lfloor \mathbf{b/2} \rfloor + \mathbf{1}$ (= 3) Kindern

    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

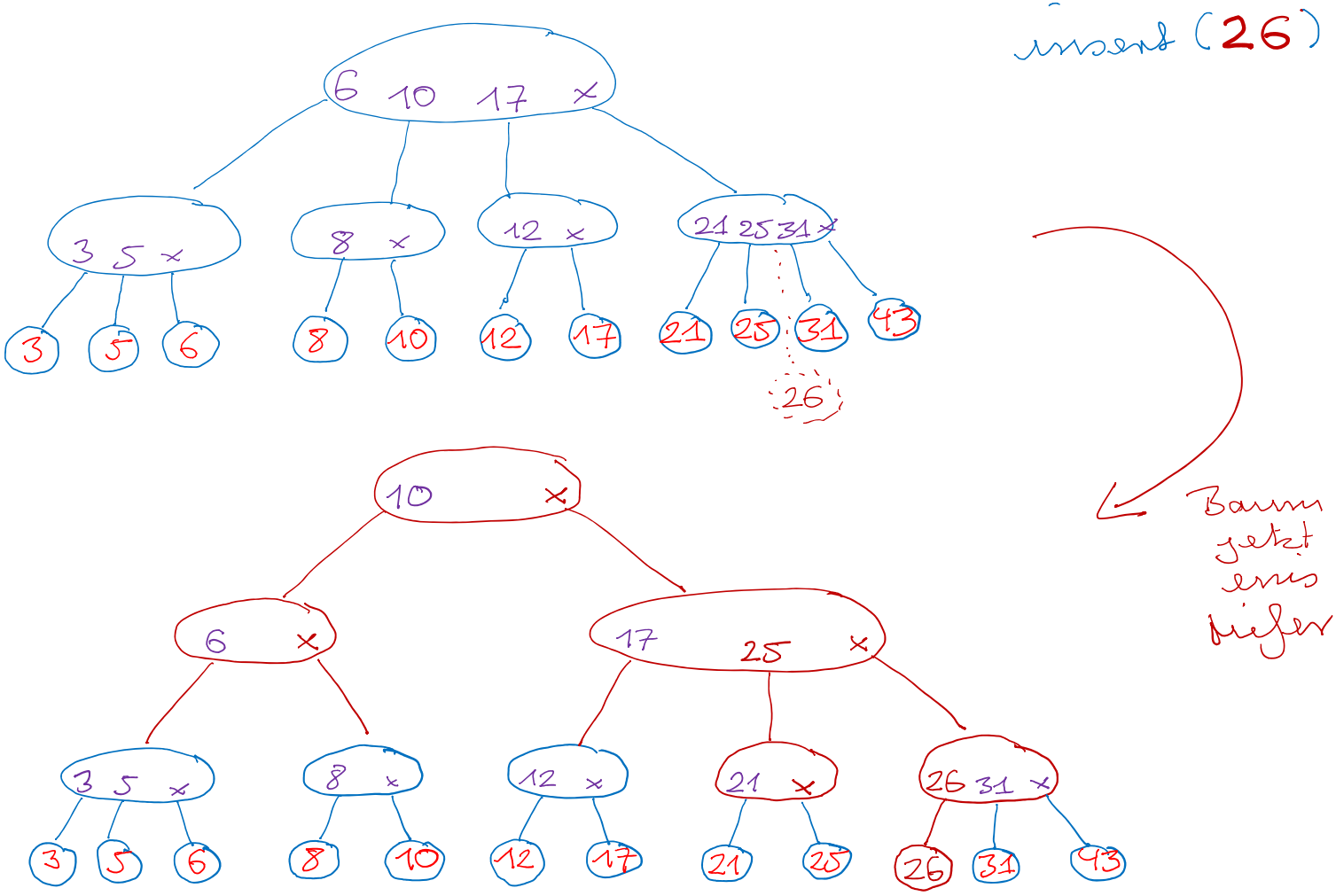
    Dann spalten wir den auf dieselbe Weise auf ... usw.
  - Wenn das bis zur Wurzel geht, spalten wir auch diese auf und erzeugen einen neuen Wurzelknoten

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

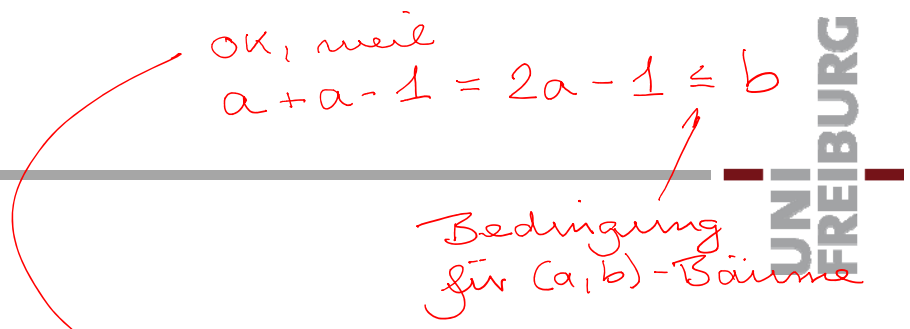
(a,b)-Bäume 8/11

alle Änderungen in ROT

- 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 (Klauen):** Eines der benachbarten Geschwister vom Elternknoten hat **> a** Kinder → ein Kind von da klauen
  - **Fall 2 (Verschmelzen):** wir verschmelzen den Elternknoten mit einem der benachbarten Geschwister mit nur **a** Kindern

    Der Großelternknoten kann jetzt a – 1 Kinder haben

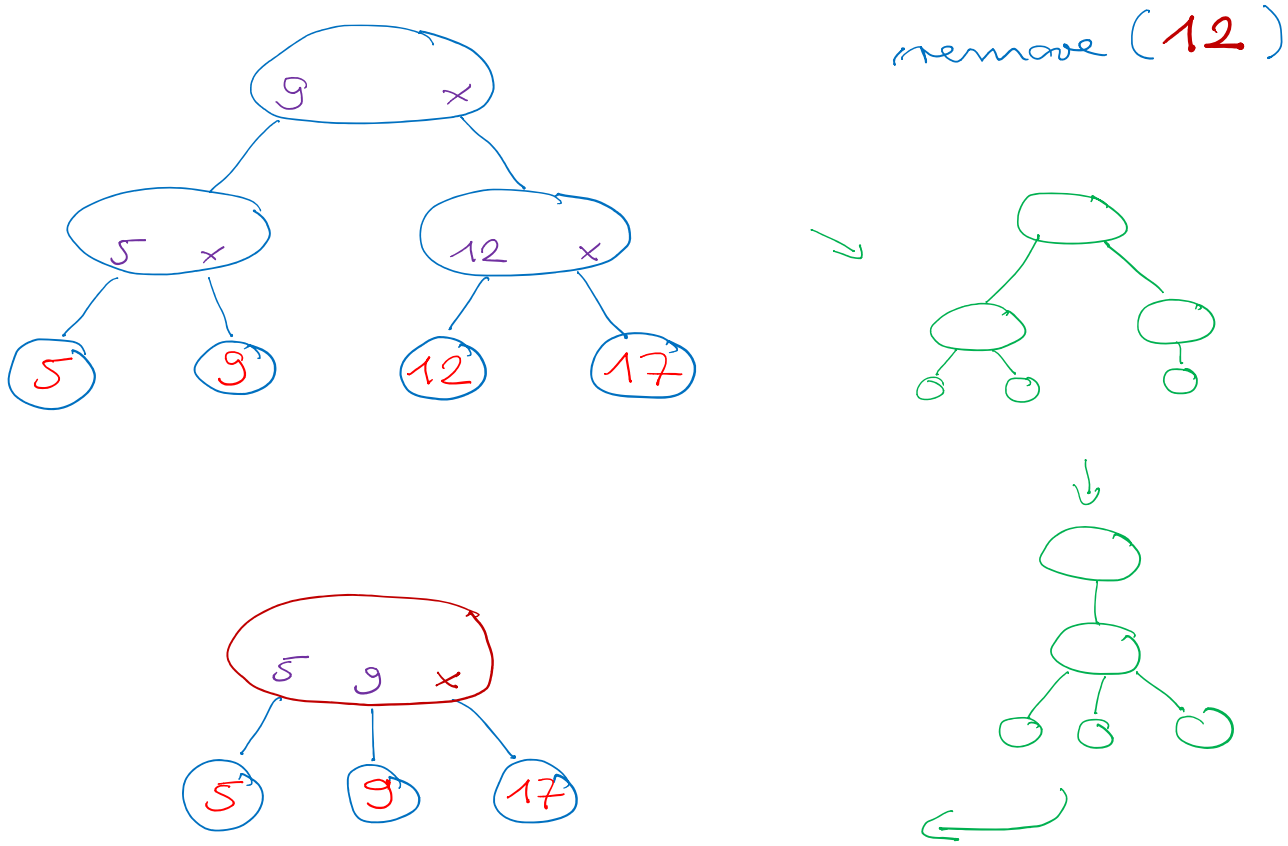
    Damit verfahren wir dann genauso … usw.
  - Wenn das bis zur Wurzel geht und die am Ende nur noch ein Kind hat, mache dieses Kind zur neuen Wurzel

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

(a,b)-Bäume 10/11

zwischen-dritte in grün

- 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
  - Das ist aber kein Problem, weil (siehe Folie 11):

    … jedem Wegweiser genau ein Blatt entspricht

    … jedes Blatt weiß, wo sein zugehöriger Wegweiser steht
  - Wir können also bei jeder Änderung an den Blättern einfach die zugehörigen Wegweiser mit ändern
  - An den inneren Knoten können wir die Wegweiser leicht beim Aufspalten / Verschmelzen anpassen

    Asymptotisch dadurch also **keine** zusätzlichen Kosten

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 $n \geq a^d$ und deshalb $d \leq \log_a n = O(\log n)$
  - Bei genauerem Hinsehen fällt auf

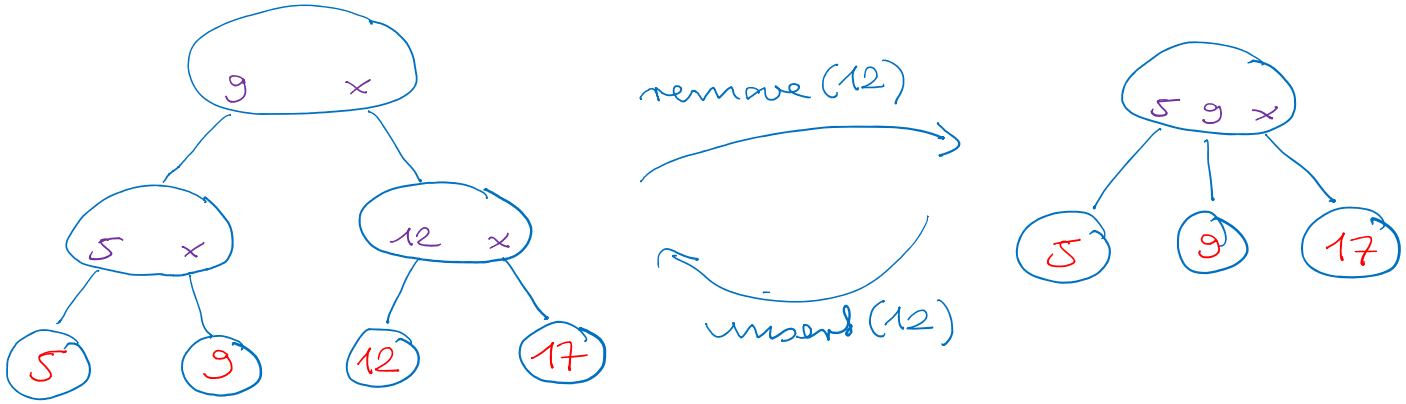
    Die Operation **lookup** braucht **immer** Zeit **$\Theta(d)$**

    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
  - Beispiel für einen (2,3)-Baum:



Analyse (2,4)-Bäume 3/8

- 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

    ÜB8, Bonusaufgabe: Potenzialfunktion für $a = 3$, $b = 7$

    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

- 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

- 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 $A, B > 0$

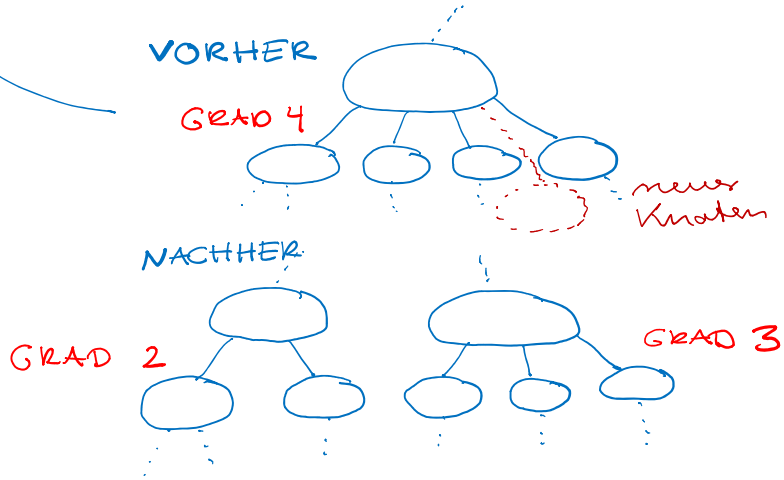
    Dann $\Sigma_{i=1..n} T_i = O(n)$ ... wenn $\Phi_n = O(n)$ für der Fall, weil #Knoten $\leq n$

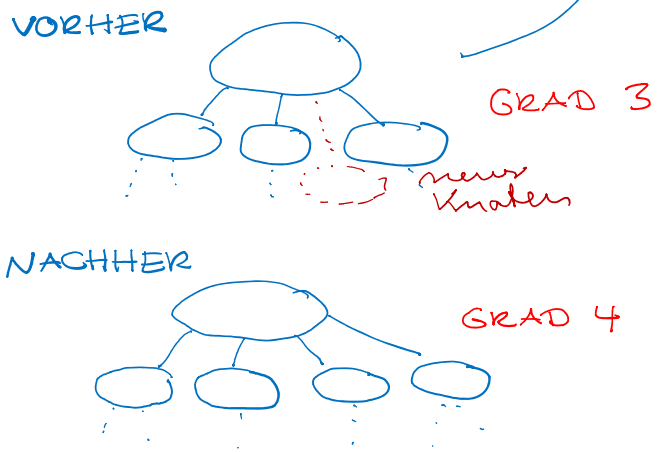
Analyse (2,4)-Bäume 6/8

- **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



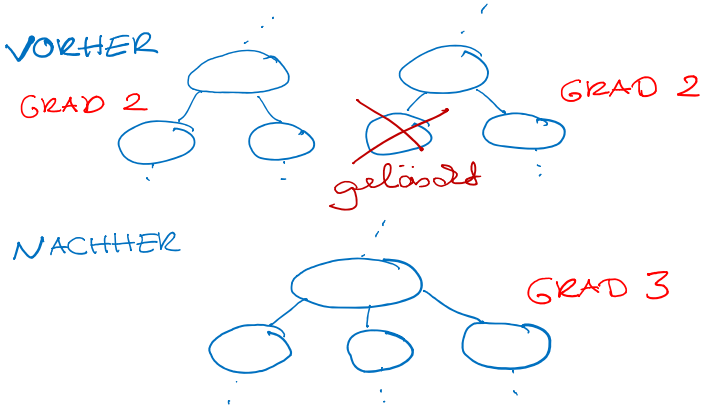


Analyse (2,4)-Bäume 7/8

- **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



selber malen
macht
schlau

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

    Also amortisiert / im Durchschnitt konstante Laufzeit

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

$T_i \leq A' \cdot m + B'$ weil: Kosten Aufspaltung oder Verschmelzung + $O(1)$

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

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

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

    https://de.wikipedia.org/wiki/(a,b)-Baum