Algorithmen und Datenstrukturen (ESE)
Entwurf, Analyse und Umsetzung von Algorithmen (IEMS)
WS 2014 / 2015

Vorlesung 11, Donnerstag, 15. Januar 2015
(Balancierte Suchbäume)

Junior-Prof. Dr. Olaf Ronneberger
Image Analysis Lab
Institut für Informatik
Universität Freiburg

[basiert auf der letztjährigen Veranstaltung von Prof. Dr. Hannah Bast, Lehrstuhl für Algorithmen und Datenstrukturen, mit Folien aus „Informatik 2" von Prof. Dr. Matthias Teschner]

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 $O(\log n)$
  - Kurz: AVL Bäume
  - Ausführlich: (a,b)-Bäume, Korrektheitsbeweis für (2,4)-Bäume
  - **Übungsblatt 11:** Korrektheitsbeweis für (4,9)-Bäume

Erfahrungen mit dem Ü10 (binäre Suchbäume)

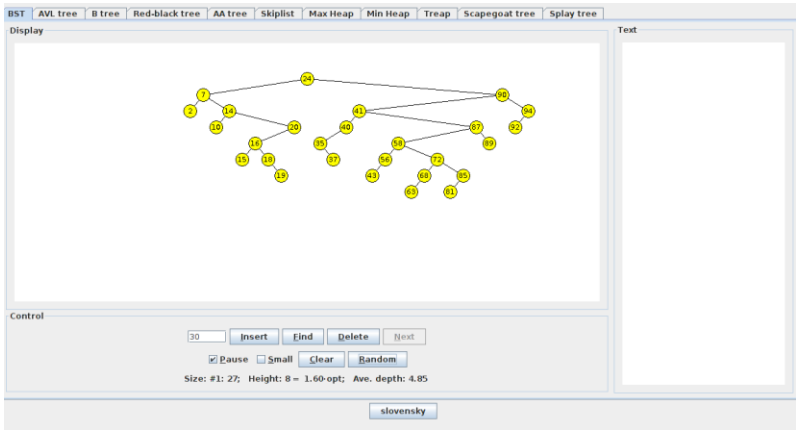
UNI FREIBURG

- Zusammenfassung / Auszüge (Stand 15.1. 8:30 Uhr)
  - Zeitaufwand im Mittel etwas mehr als vier Stunden
  - „segmentation fault" → valgrind für C++ Programmierer
  - toString() Methode war nicht ganz einfach
  - ein paar Rückmeldungen zur Vorlesung → Vielen Dank! Übernehme ich für nächstes Semester

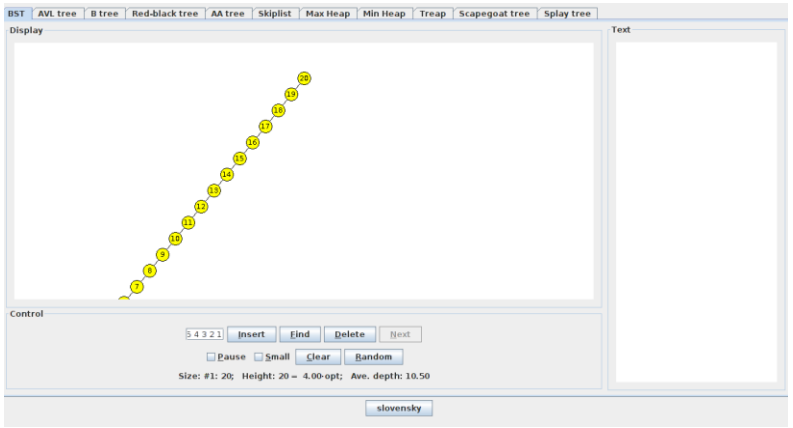
Balancierte Bäume: Motivation

- 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 20, 19, 18, ... eingefügt werden

  http://people.ksp.sk/~kuko/bak/



(30 mal **Insert** klicken bei leerem Eingabefeld, oder 30 eintippen und auf **Random** klicken)



(„20 19 18 17 16 15 usw." mit Space getrennt ins Eingabefeld, dann **Insert** klicken)

Balancierte Bäume

UNI FREIBURG

- 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

Balancierte Bäume: Verfahren

UNI FREIBURG

- Wie erreicht man immer Tiefe O(log n) ?
- Es gibt Dutzende verschiedener Verfahren dafür:
  - **AVL-Baum**:
    - Binärbaum (2 Kinder pro Knoten)
    - Ausbalancieren durch „**Rotation**“
  - **(a,b)-Baum**, oder **B-Baum**:
    - Knoten haben zwischen a und b Kindern
    - Ausbalancieren durch **Spalten** und **Verschmelzen** der Knoten
    - Einsatz in Datenbanken und Dateisystemen
  - **Rot-schwarz-Baum**
    - Binärbaum (2 Kinder pro Knoten) mit „schwarzen“ und „roten“ Knoten
    - Ausbalancieren durch **„Rotation“** und **„Umfärben“**
    - Einsatz in C++ std::map, und Java SortedMap und SortedSet
    - Kann direkt als (2,4)-Baum interpretiert werden.

AVL Baum: Definition

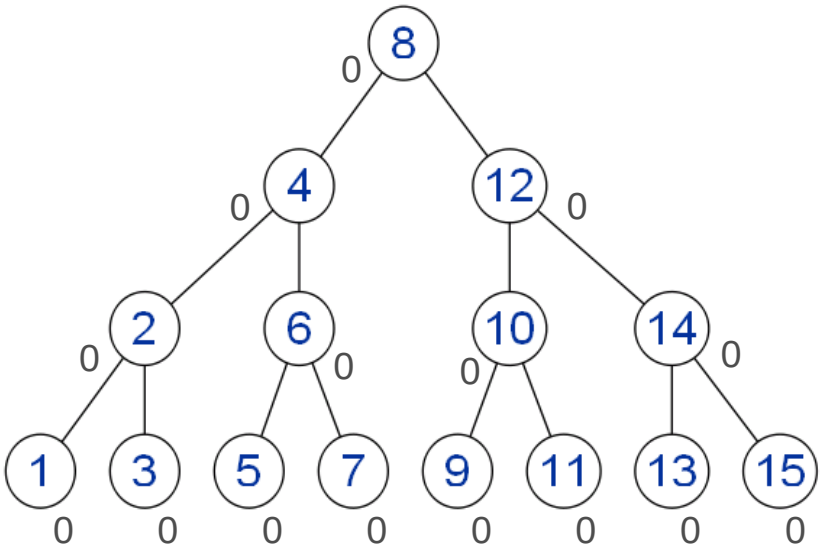
UNI FREIBURG

- Suchbaum mit modifizierten Einfüge- und Entferne-Operationen zur Einhaltung einer Höhenbedingung
- verhindert Degenerieren des Suchbaums
- Höhenunterschied von linkem und rechtem Teilbaum aller Knoten des Suchbaums ist maximal eins
- dadurch ist die Höhe des Suchbaums O (log n), womit alle weiteren Grundoperationen in O (log n) ausführbar sind.

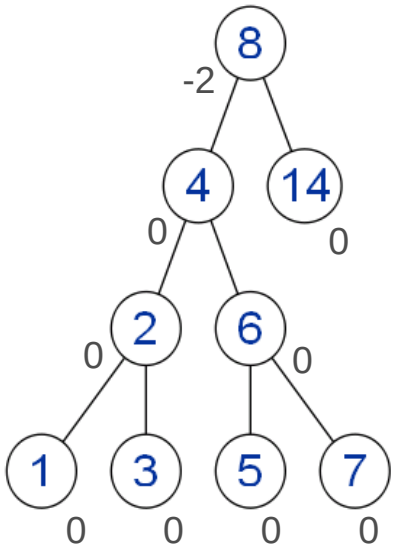
Georgy Maximovich **A**delson-**V**elskii, Yevgeniy Mikhailovich **L**andis,
An algorithm for the organization of information,
Doklady Akademia Nauk SSSR 1962.

AVL-Baum: Beispiele

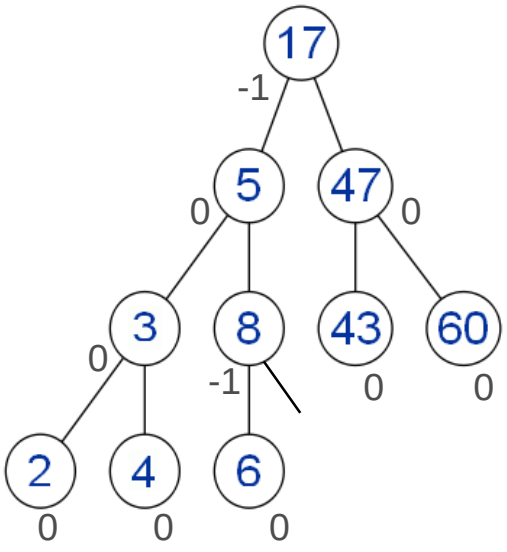
- Suchbäume mit Balance-Grad



AVL-Baum

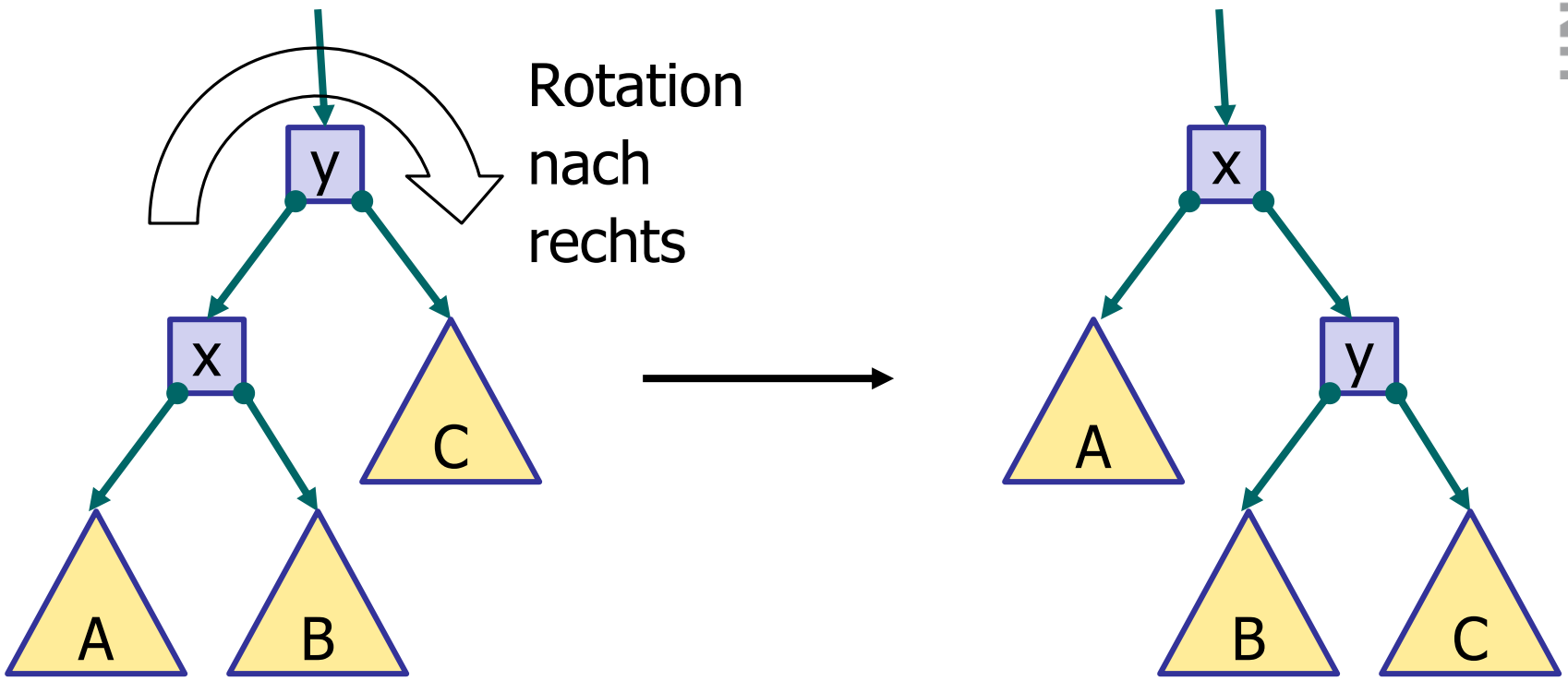


kein AVL-Baum



AVL-Baum

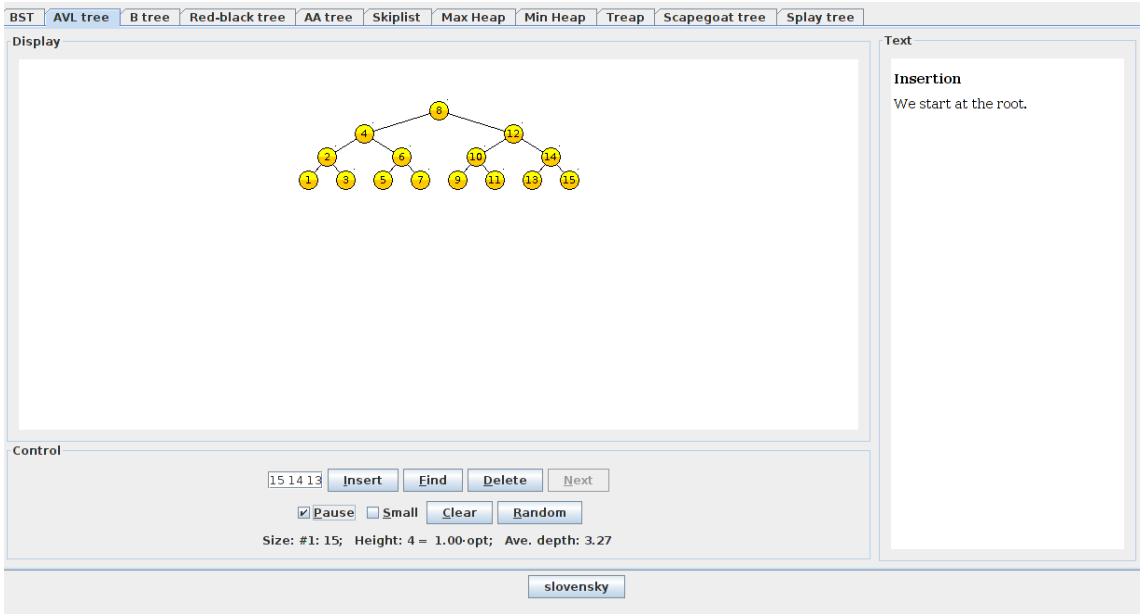
AVL Baum: Rebalancieren



- Zentrale Operation zum Rebalancieren: Rotation
- Beispiel: Nach der Rechtsrotation:
  - Teilbaum A ist eine Ebene höher und Teilbaum C eine Ebene tiefer
  - x und y haben ihr Eltern-Kind-Verhältnis geändert

AVL Baum Beispiel

- Sobald beim Einfügen oder Löschen eine Höhendifferenz von 2 entsteht, wird rebalanciert
- Viele Fallunterscheidungen, die wir uns nicht im Detail ansehen.
- Beispiel: Einfügen von 1,2,3,4,… http://people.ksp.sk/~kuko/bak/



(„1 2 3 4 5 6 7 usw." mit Space getrennt ins Eingabefeld, dann **Insert** klicken)

AVL Baum Zusammenfassung

UNI FREIBURG

- Historisch gesehen die erste Suchbaumstruktur, die insert, remove und lookup in O(log n) ermöglicht
- Aber nicht amortisierte Update-Kosten von O(1)
- Zusätzliche Speicher-Kosten: In jedem Knoten die Höhendifferenz vom linken und rechten Teilbaum
- Besser (und einfacher zu implementieren) sind (a,b)-Bäume

(a,b)-Bäume Überblick

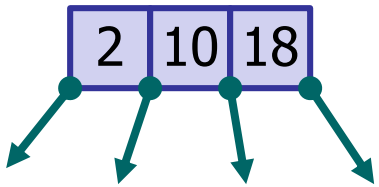
UNI FREIBURG

- Auch bekannt als B-Baum (B für „balanced“)
- Einsatz in Datenbanken und Dateisystemen
- Idee: Variable Anzahl von Elementen in den Knoten speichern
- Dadurch ist beim Einfügen/Ausbalancieren fast immer Platz für das neue Element.

(a,b)-Bäume Definition

UNI FREIBURG

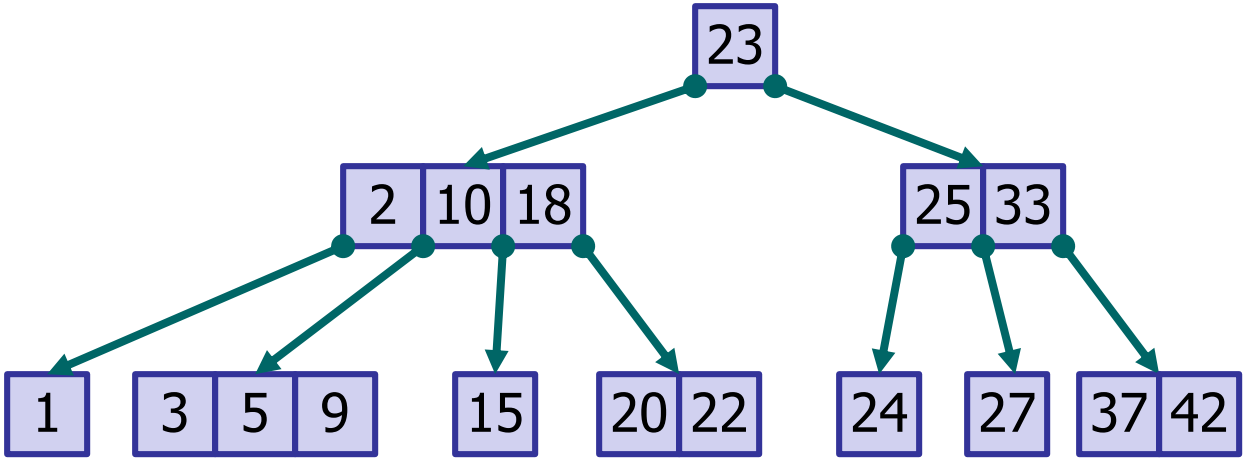
- Alle Blätter haben die gleiche Tiefe
- Jeder innere Knoten hat $\geq a$ und $\leq b$ Kinder (nur die Wurzel darf weniger Kinder haben)



- Ein Knoten mit n Kindern (bzw. n Pointern) heißt **Knoten vom Grad n** und speichert $n-1$ Elemente (sortiert)
- Unterbäume hängen „zwischen" den Elementen
- Wir verlangen $a \geq 2$ und $b \geq 2a - 1$ … warum sehen wir gleich

(a,b)-Bäume Beispiel

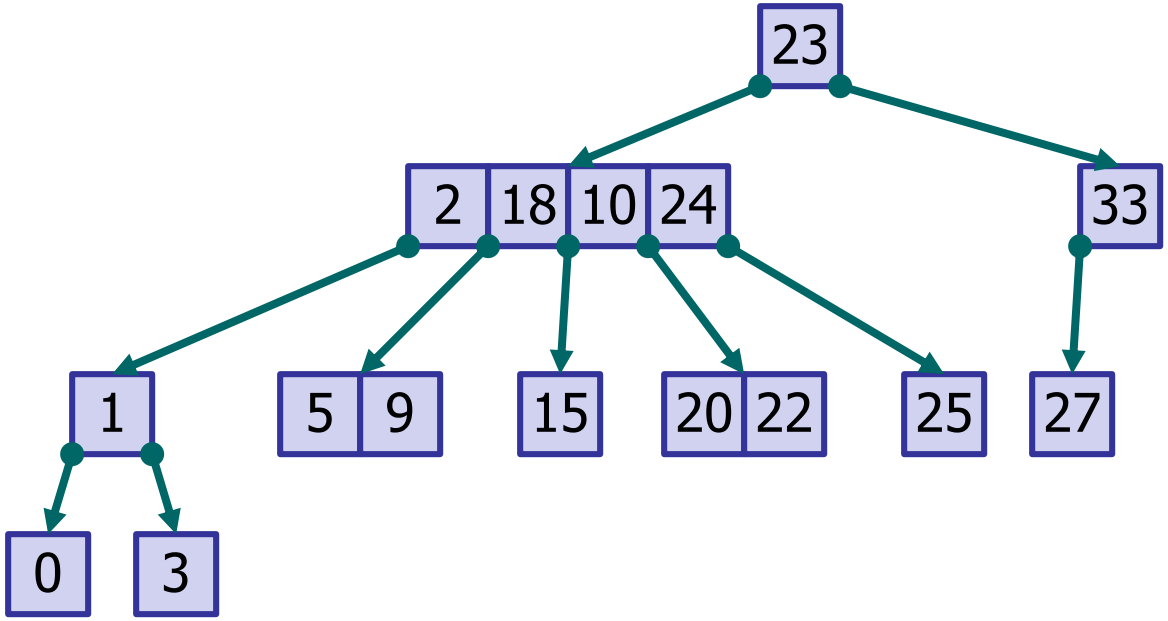
- Beispiel für einen (2,4)-Baum



(2,4)-Baum der Tiefe 3. Alle Knoten haben zwischen 2 und 4 Kinder, bzw. zwischen 1 und 3 Elemente

(a,b)-Bäume Gegenbeispiel

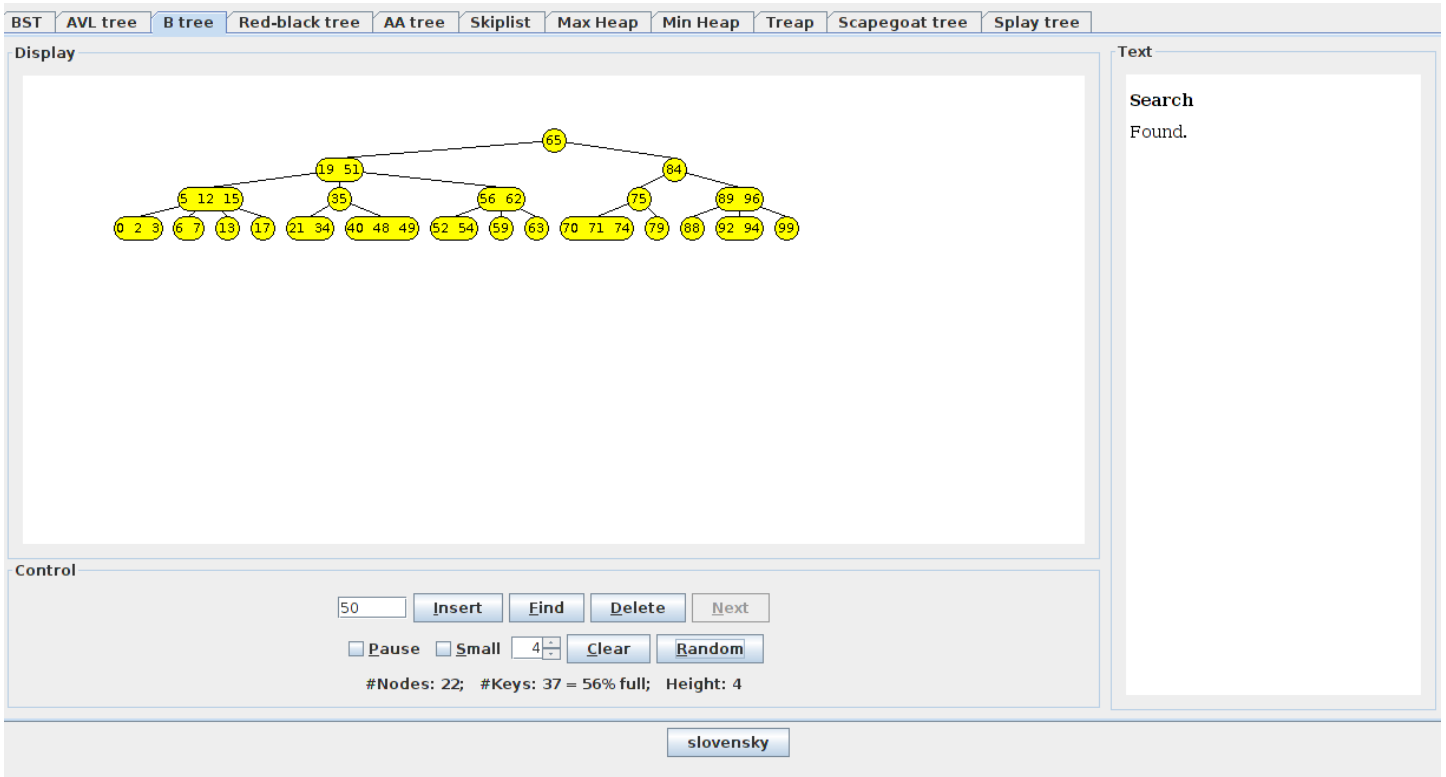
- Gegenbeispiel für einen (2,4)-Baum



  - Nicht korrekt sortiert (innerhalb der Knoten und Eltern-Kind)
  - Grad zu klein / zu groß
  - Blätter auf unterschiedlichen Ebenen

(a,b)-Bäume: lookup

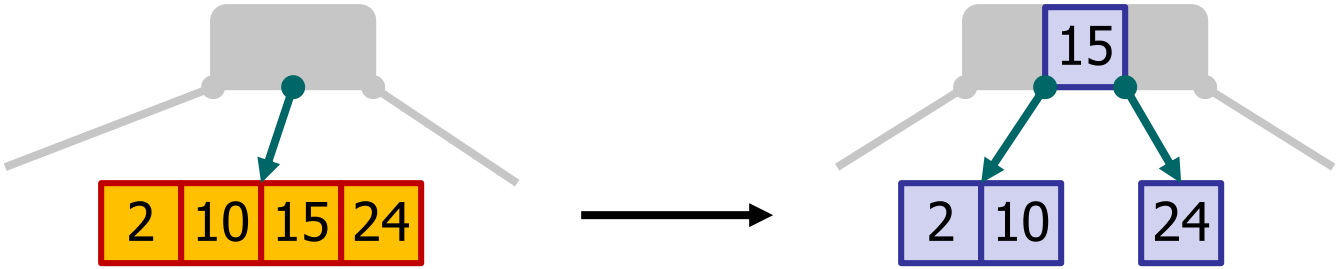
- Schlüsselsuche (Lookup)
  - Im Prinzip genau so wie beim BinarySearchTree
  - Suche von der Wurzel abwärts
  - Die Schlüssel an den inneren Knoten weisen den Weg



Size „4" einstellen
„50" ins Eingabefeld
Random klicken
Suchzahl eingeben
Find klicken

(a,b)-Bäume: insert

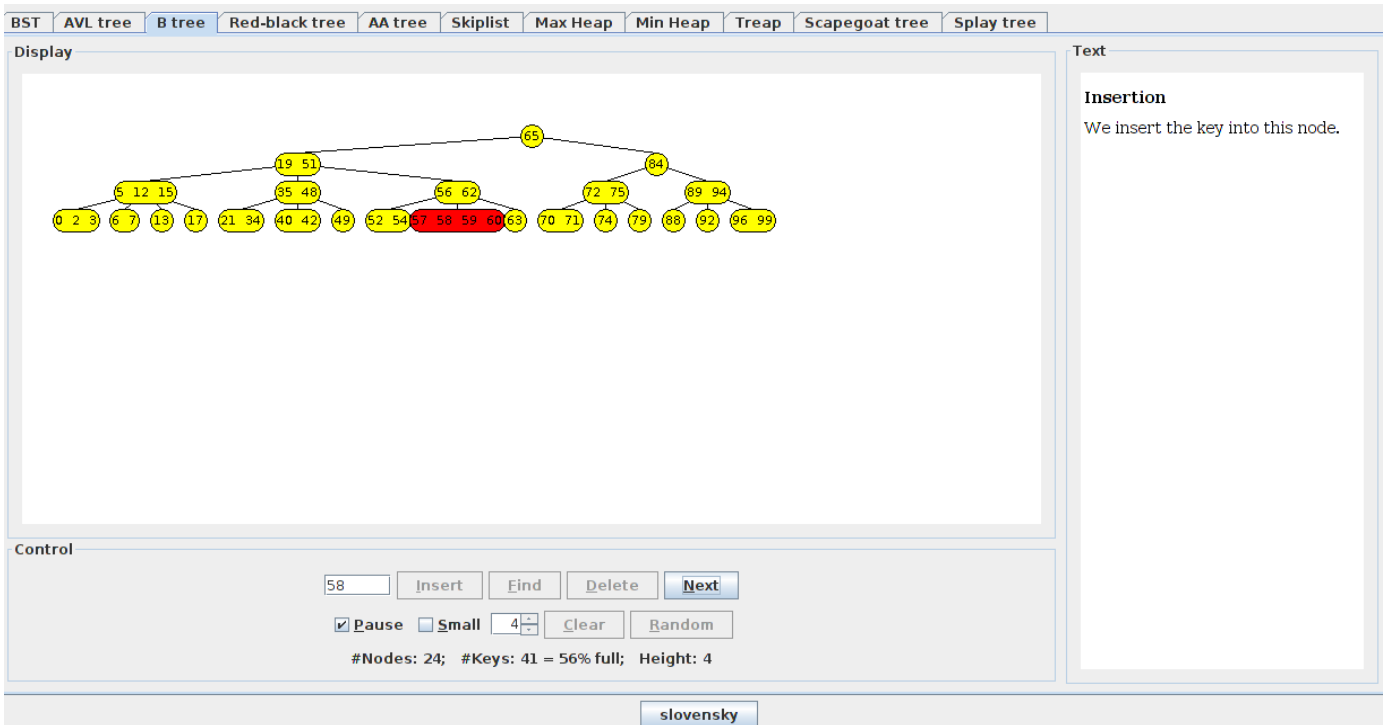
- Einfügen eines Elementes (insert)
  - Finde die Stelle, wo der neue Schlüssel einzufügen ist
  - Wir landen immer in einem Blatt
  - Füge das Element in diesen Knoten ein
  - Achtung: Knoten kann jetzt ein Element zu viel haben (Grad b+1)
  - Dann **spalten** wir den Knoten einfach auf



  - in einen Knoten mit ceil((b-1)/2) Elementen, ein Element für den Elternknoten, und einen Knoten mit floor((b-1)/2) Elementen
  - Deswegen $b \geq 2a-1$

(a,b)-Bäume: insert

- Fortsetzung: Einfügen eines Elementes (insert)
  - Der Elternknoten kann jetzt Grad b+1 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**



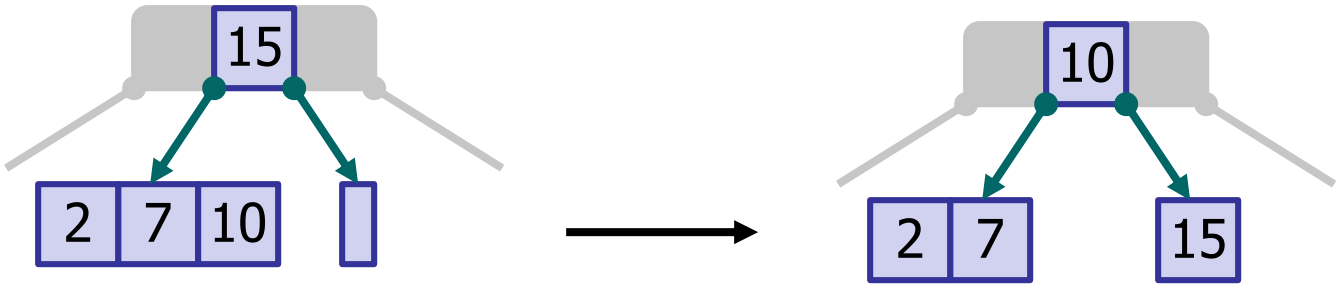
(a,b)-Bäume: remove

UNI FREIBURG

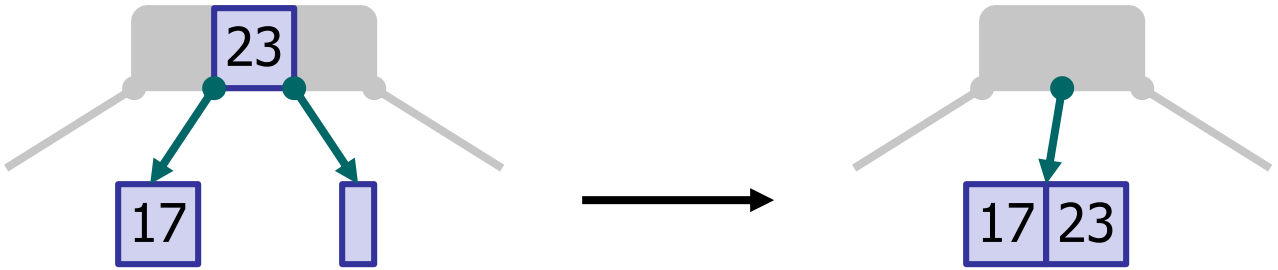
- Entfernen eines Elementes (remove)
  - Finde das zu entfernende Element
  - **Fall I:** Element im Blatt, entferne es direkt
  - **Fall II:** Element ist im inneren Knoten,
    - Suche seinen Nachfolger im rechten Teilbaum des Elements
    - Nachfolger steht immer in einem Blatt!
    - Ersetze Element durch den Nachfolger und entferne ihn aus dem Blatt
  - Achtung: das Blatt kann jetzt zu klein sein (Grad a-1)

(a,b)-Bäume: remove

- Fortsetzung: Entfernen eines Elementes (remove)
  - **Fall 1:** Falls der linke oder rechte Bruder Grad > a hat, **klauen** eins von da weg und sind fertig

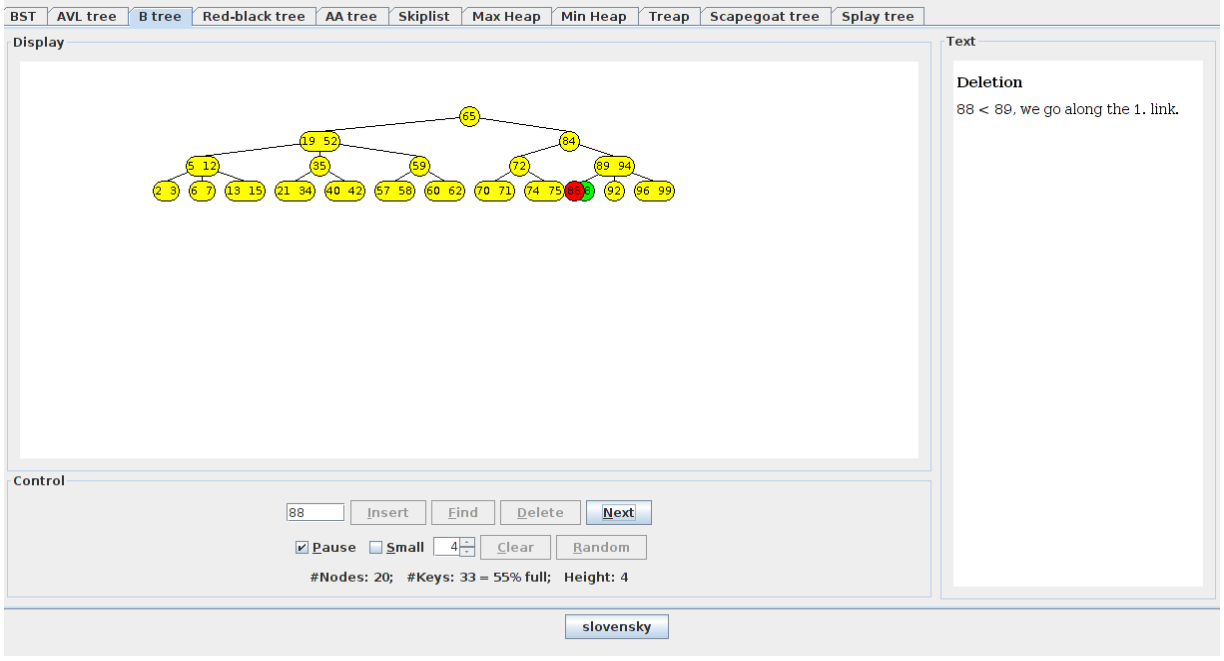


  - **Fall 2:** Sonst **verschmelzen** wir den Knoten mit seinem linken oder rechten Bruder.



(a,b)-Bäume: remove

- Fortsetzung: Entfernen eines Elementes (remove)
  - Nach einem Verschmelzen hat der Elternknoten jetzt ein Kind weniger und kann Grad a-1 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**



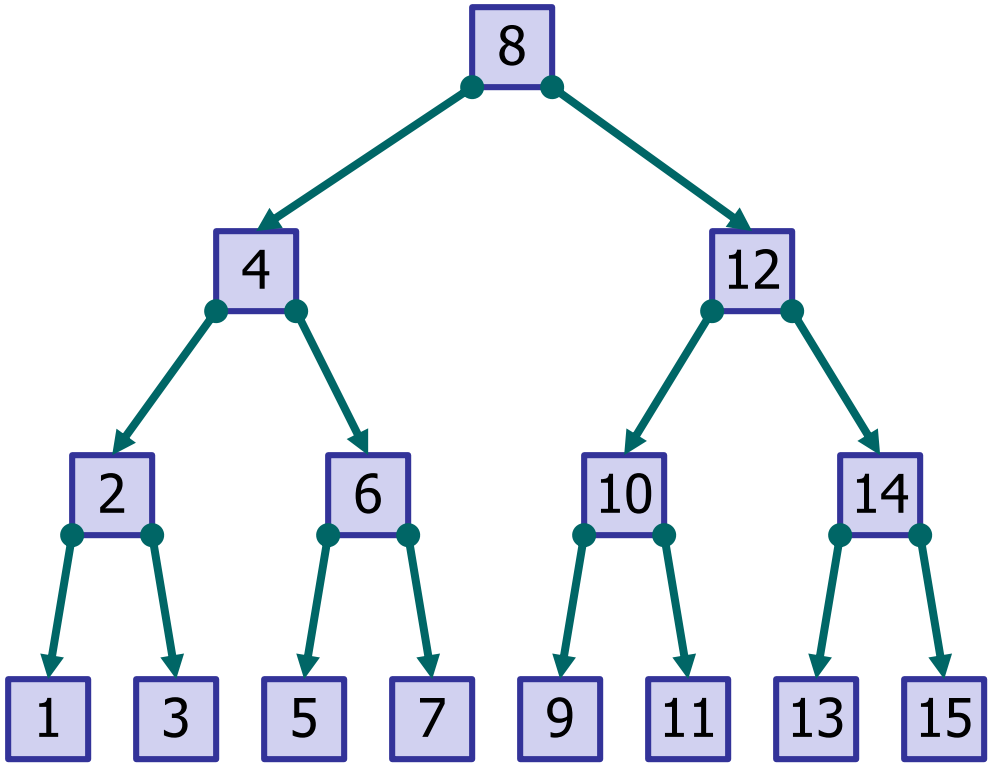
Analyse (a,b)-Bäume

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

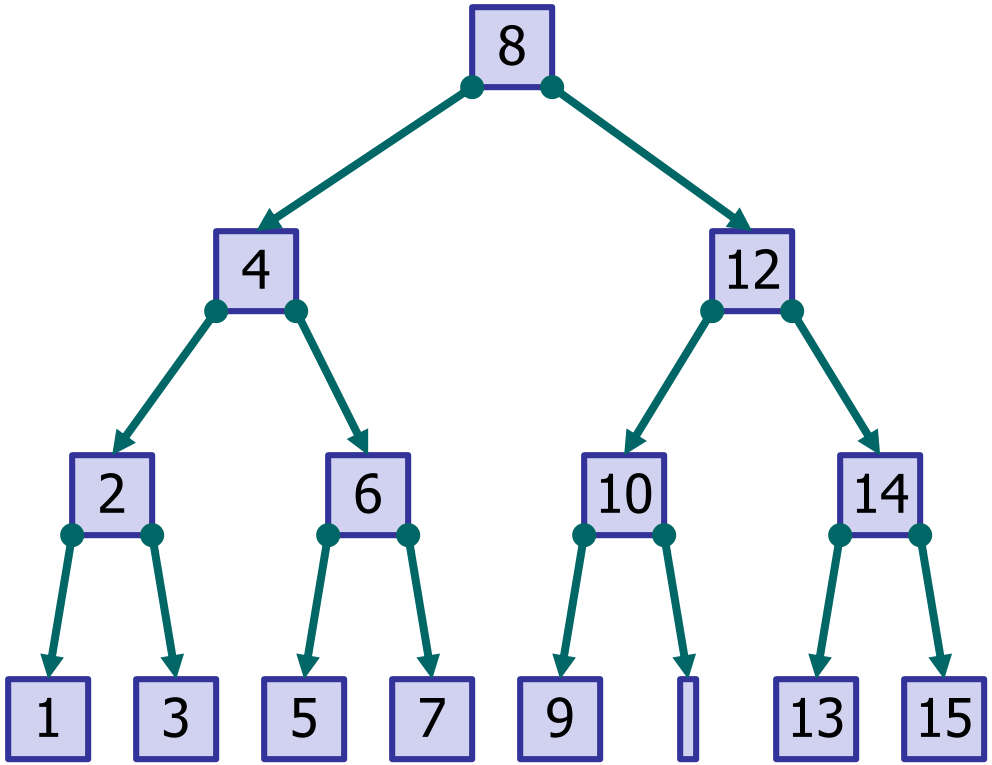
      (nur im schlechtesten Fall müssen alle Knoten auf dem Weg zur Wurzel gespalten / verschmolzen werden)
  - Das wollen wir jetzt genauer analysieren
  - Dafür reicht $b \geq 2a - 1$ allerdings nicht, wir brauchen **$b \geq 2a$**
  - Gegenbeispiel für (2,3)-Bäume, Analyse für (2,4)-Bäume

Gegenbeispiel für (2,3)-Bäume

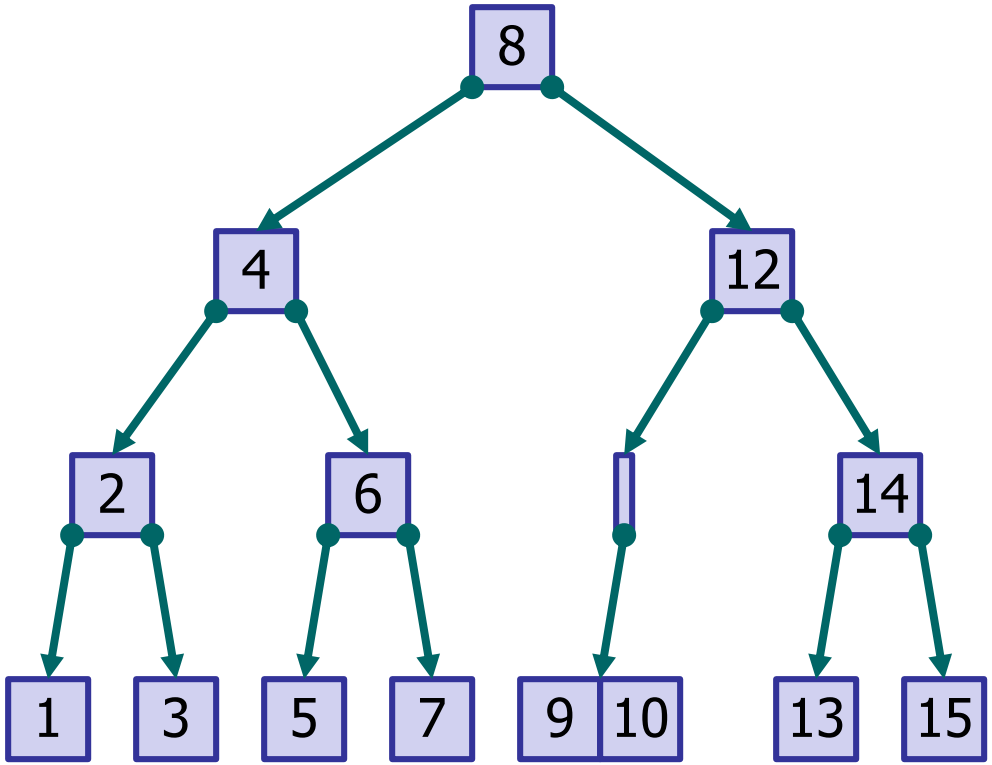


delete(11)

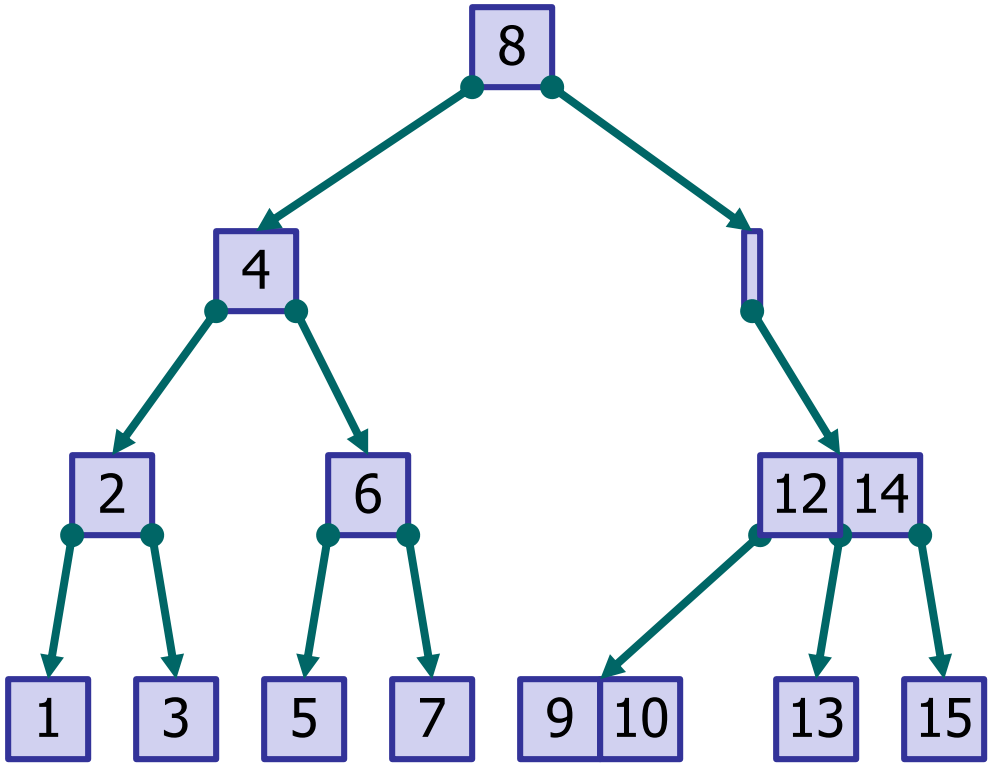
Gegenbeispiel für (2,3)-Bäume



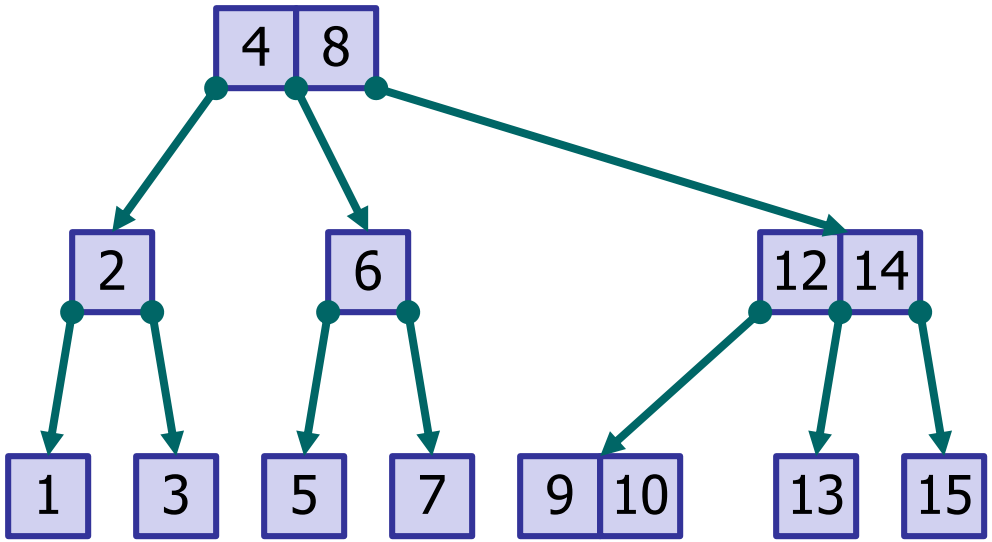
Gegenbeispiel für (2,3)-Bäume



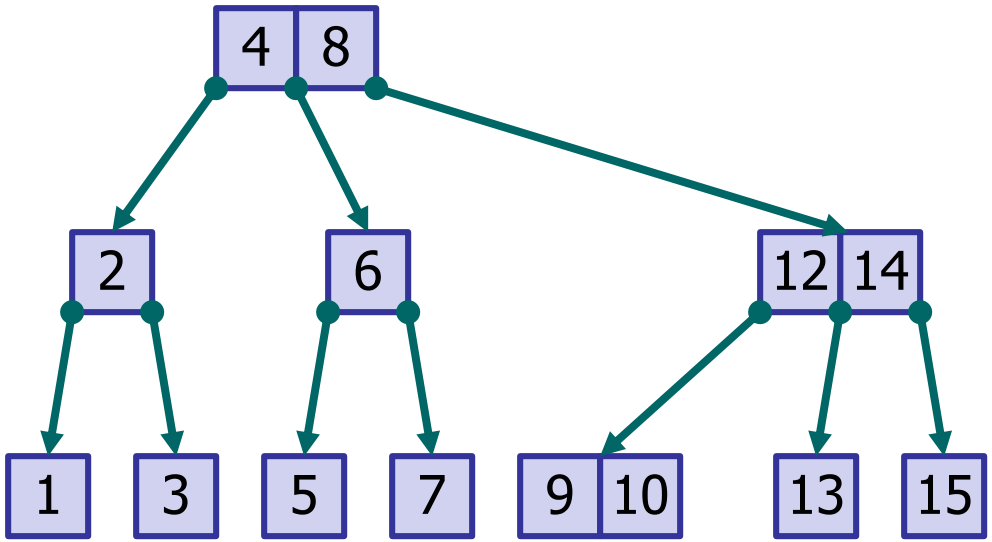
Gegenbeispiel für (2,3)-Bäume



Gegenbeispiel für (2,3)-Bäume

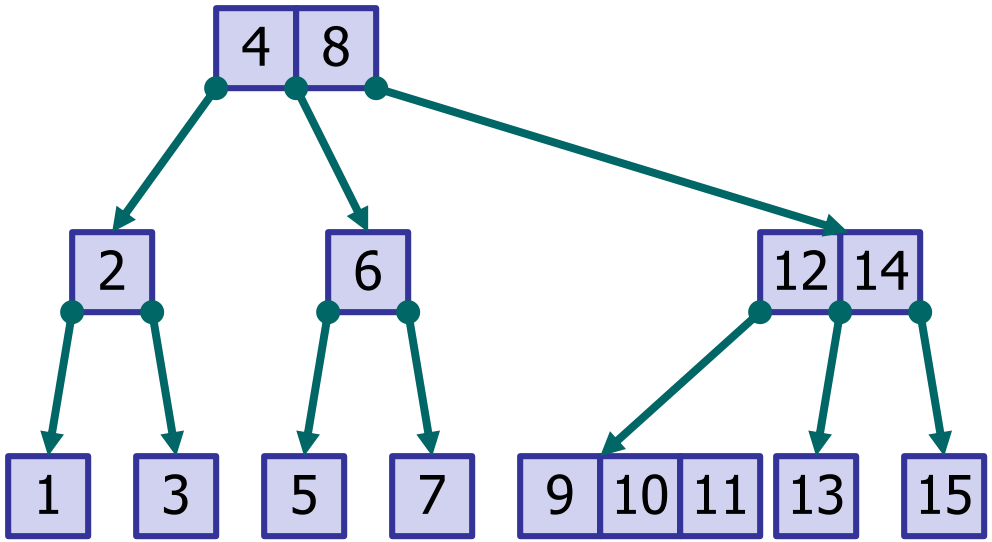


Gegenbeispiel für (2,3)-Bäume

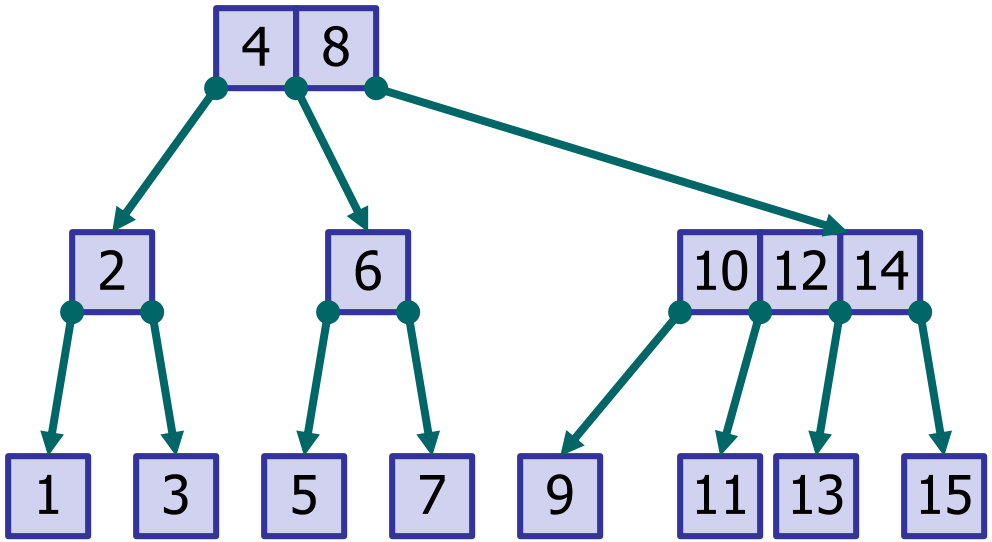


insert(11)

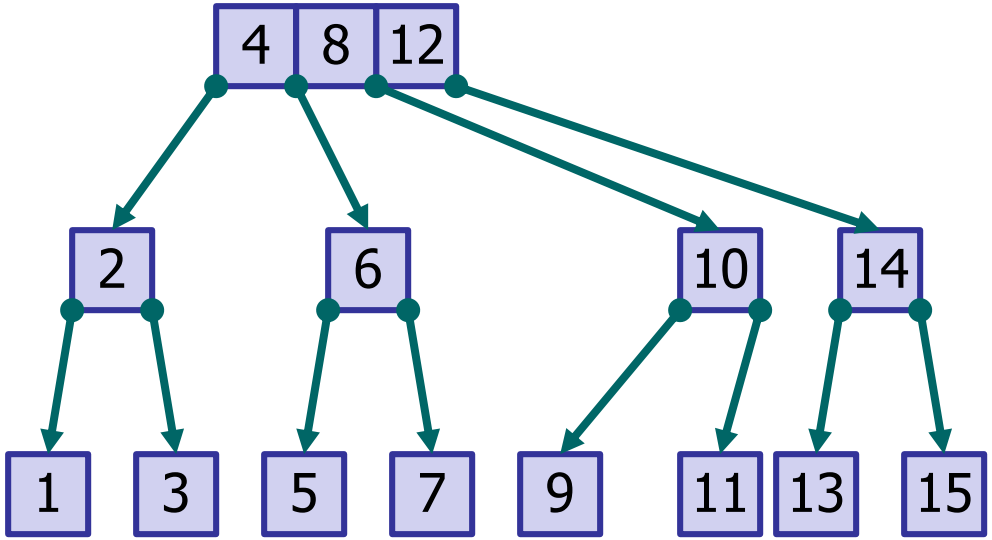
Gegenbeispiel für (2,3)-Bäume



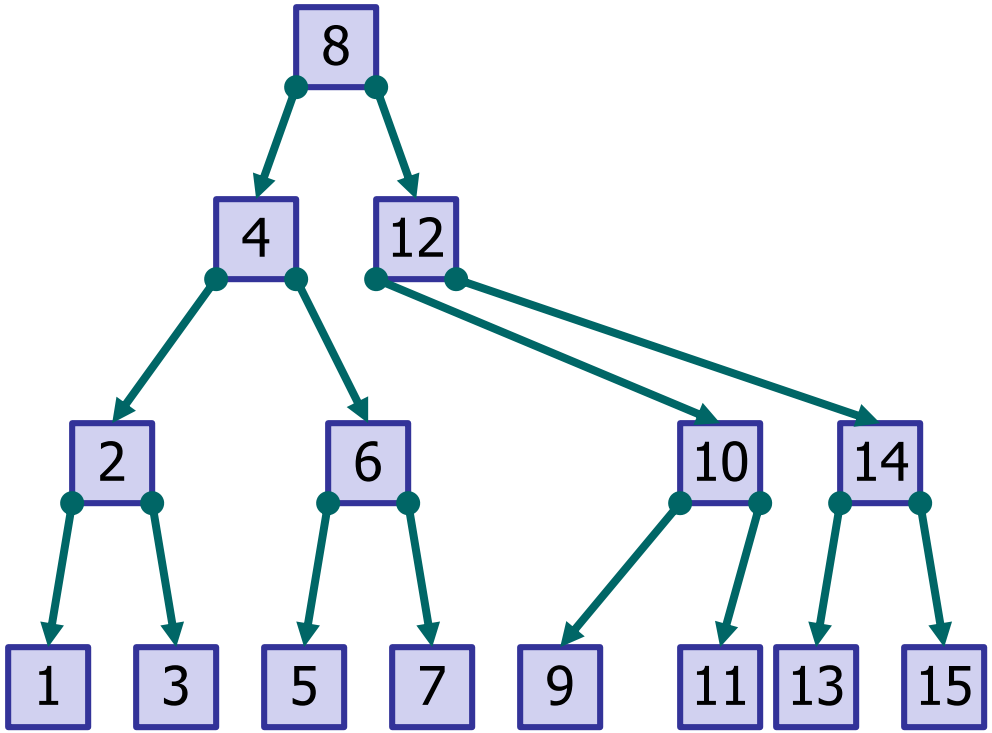
Gegenbeispiel für (2,3)-Bäume



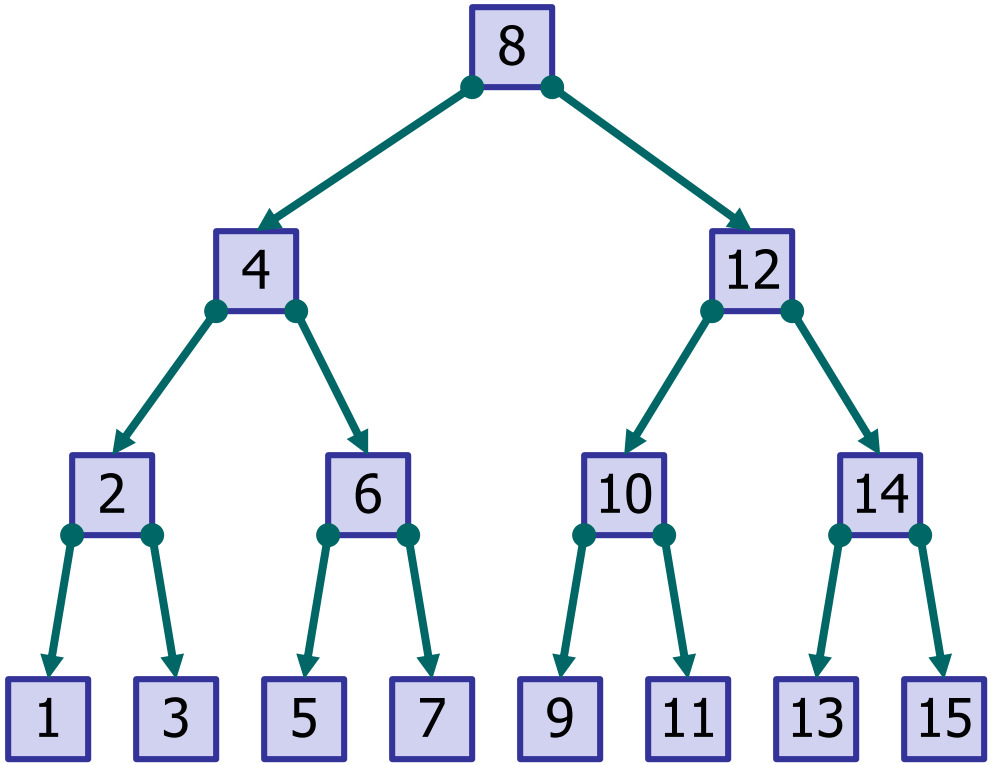
Gegenbeispiel für (2,3)-Bäume



Gegenbeispiel für (2,3)-Bäume



Gegenbeispiel für (2,3)-Bäume



- Wir sind wieder beim Anfangszustand
- Also: Wenn b = 2a-1, dann gibt es eine Folge von inserts und removes, bei der jede Operation log n kostet

Analyse (2,4)-Bäume: Intuition

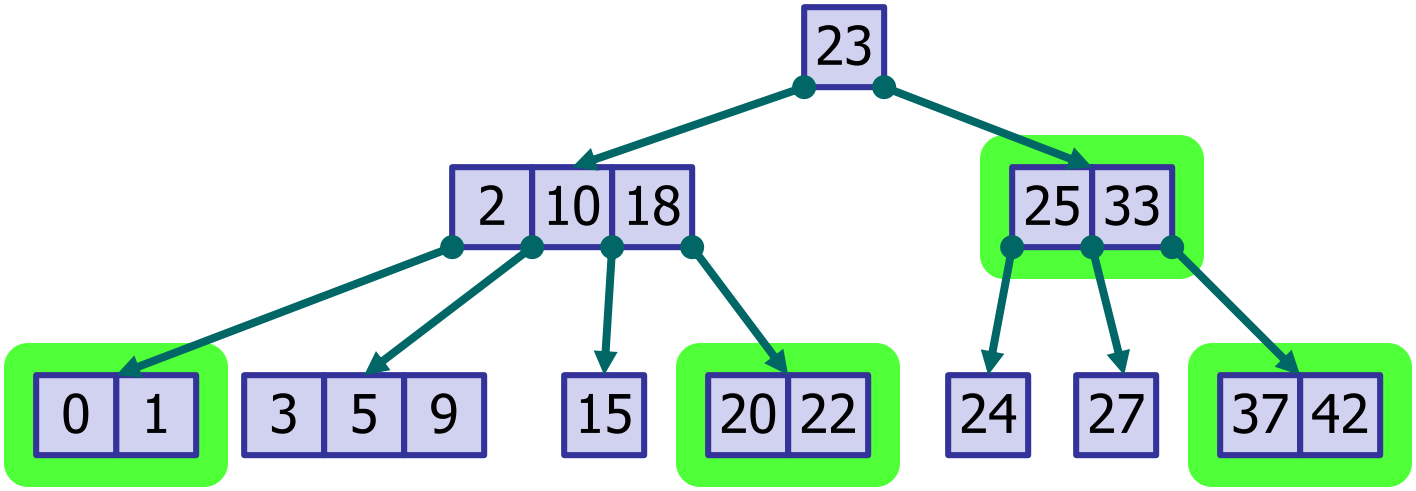
UNI FREIBURG

- 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
  - **Erkenntnis:** Die Grad-3-Knoten sind harmlos – weder ein insert, noch ein remove in einem solchen Knoten ziehen weitere Operationen nach sich!
  - 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: Potential

- Terminologie
  - Wir betrachten eine Folge von n Operationen
  - Sei Φ_i das Potential des Baumes nach der i-ten Operation
    = die Anzahl der harmlosen Knoten (mit Grad genau 3)

Beispiel: Baum mit Potential $\Phi = 4$, (Grad-3-Knoten umrandet)



  - Φ_0 = Potential am Anfang := 0 (leerer Baum)

Analyse (2,4)-Bäume: Beweis 1/7

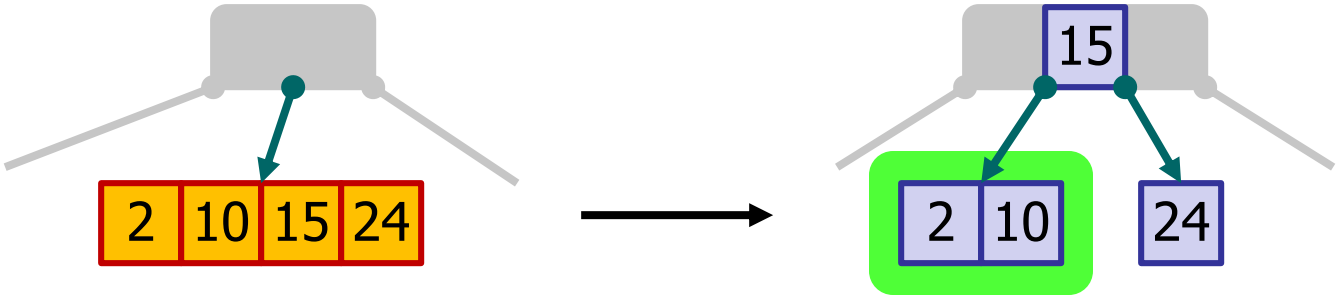
UNI FREIBURG

- Forsetzung: Terminologie
  - Seien c_i die Kosten = Laufzeit der i-ten Operation
- Wir werden zeigen:
  - Jede Operation kann maximal einen harmlosen Knoten zerstören
  - Für jeden weiteren Schritt, der Kosten verursacht, erzeugt die Operation jeweils einen neuen harmlosen Knoten
- D.h. die Kosten für Operation i sind mit der Potentialdifferenz gekoppelt:
  - Es gilt $c_i \leq A \cdot (\Phi_i - \Phi_{i-1}) + B$ für irgendwelche $A > 0$ und $B > A$

    Anzahl der harmlosen Knoten, die bei der i-ten Operation hinzukommen. Kann auch -1 sein, aber nicht kleiner als -1
- Dann zeigen wir, dass die Summe der Kosten $O(n)$ hat, und somit jede Operation $O(1)$ (amortisiert)

Analyse (2,4)-Bäume: Beweis 2/7

- Fall 1: i-te Operation ist ein insert auf einen vollen Knoten



  - Beobachtung: Für jedes Aufspalten hat man nachher einen Grad-3-Knoten mehr
  - Im Elternknoten kommt ein Element hinzu.
  - Wenn der auch voll war, ebenfalls aufspalten, usw. bis zu einem Elternknoten von Grad 2 oder 3. Dann fertig.

Analyse (2,4)-Bäume: Beweis 3/7

UNI FREIBURG

- Sei m = Anzahl der Aufspaltungen
- Das Potential erhöht sich um m und erniedrigt sich am Ende evtl. um 1, wenn der "Stop-Knoten" vom Grad 3 war.

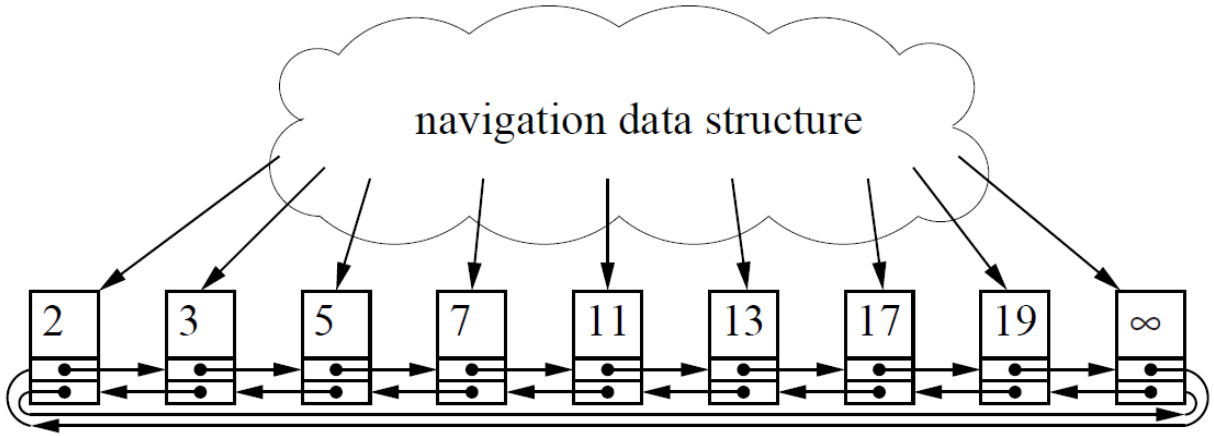
$$\Phi_i \geq \Phi_{i-1} + m - 1$$
$$\Rightarrow m \leq \Phi_i - \Phi_{i-1} + 1$$

Kosten: $c_i \leq A \cdot m + B$

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

Analyse (2,4)-Bäume: Beweis 4/7

- Fall 2: i-te Operation ist ein remove
- Fall 2.1: Innerer Knoten:
  - Finden des Nachfolgers im Baum ist O(d)
  - Aber: Üblicherweise Kombination mit doppelt verketteter Liste:

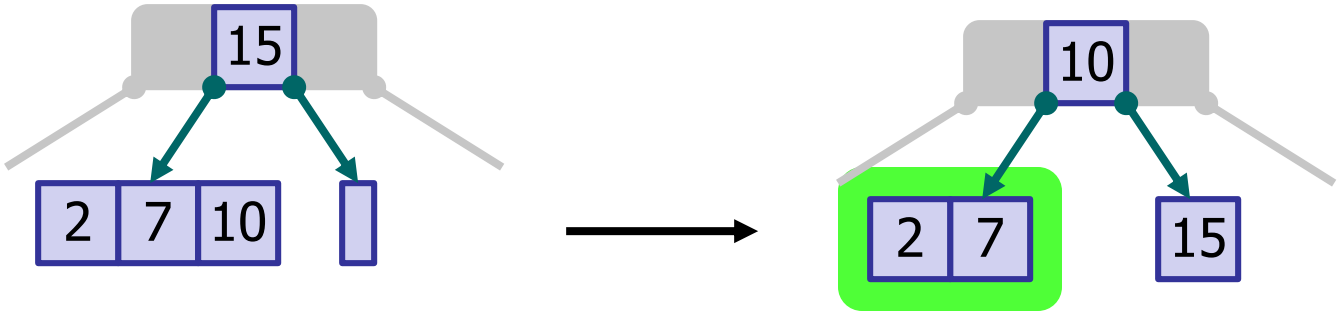


[Mehlhorn-Sanders, Fig. 7.1]

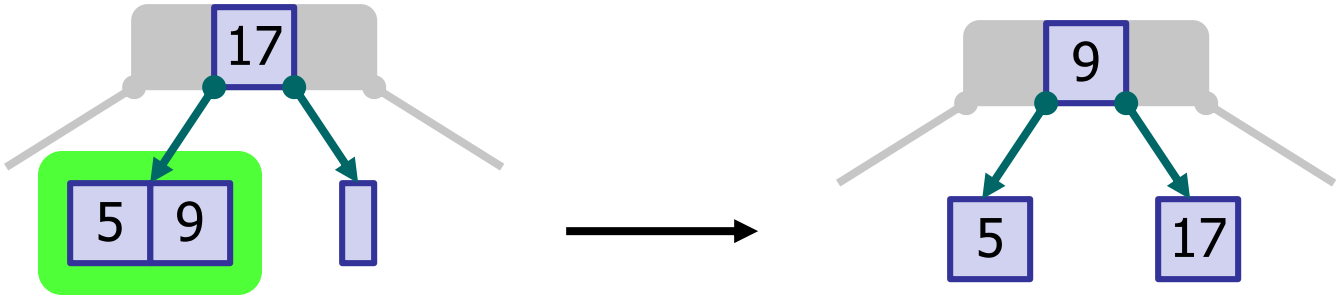
  - Damit findet man man den Nachfolger in O(1)

Analyse (2,4)-Bäume: Beweis 5/7

- Fall 2: i-te Operation ist ein remove
- Fall 2.1: Klauen: Potential erhöht sich um eins:



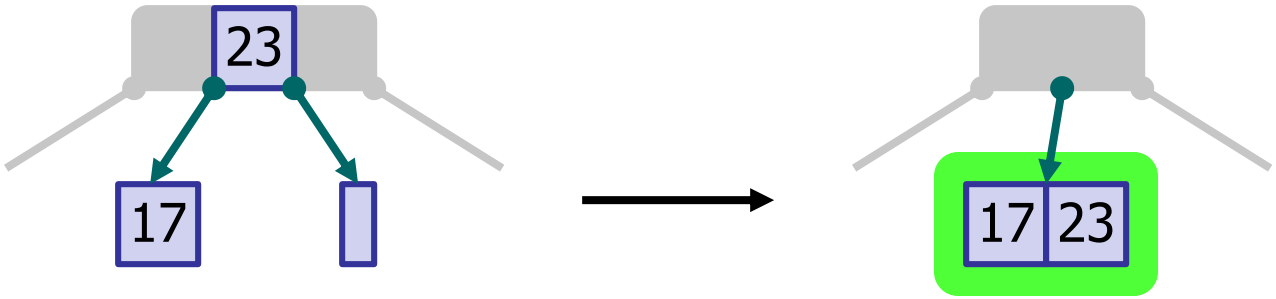
– oder Potential verringert sich um eins:



– aber nur 1mal (Operation pflanzt sich nicht nach oben fort)

Analyse (2,4)-Bäume: Beweis 6/7

- Fall 2.2: Verschmelzen: Potential erhöht sich um eins:



  - pflanzt sich so lange nach oben fort, bis zu einem Knoten von Grad > 2 oder einem Grad-2-Knoten, der etwas klauen kann.
  - Also auch hier m mal Potential erhöhen und am Ende evtl. 1 mal verringern:
  - → gleiche Kosten wie beim insert. □

Analyse (2,4)-Bäume: Beweis 7/7

- Lemma
  - Es gilt $c_i \leq A \cdot (\Phi_i - \Phi_{i-1}) + B$ für irgendwelche $A > 0$ und $B > A$

    Anzahl der Grad-3-Knoten, die bei der i-ten Operation hinzukommen
  - Daraus folgt dann $\sum_{i=1..n} c_i = O(n)$
  - Beweis:

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

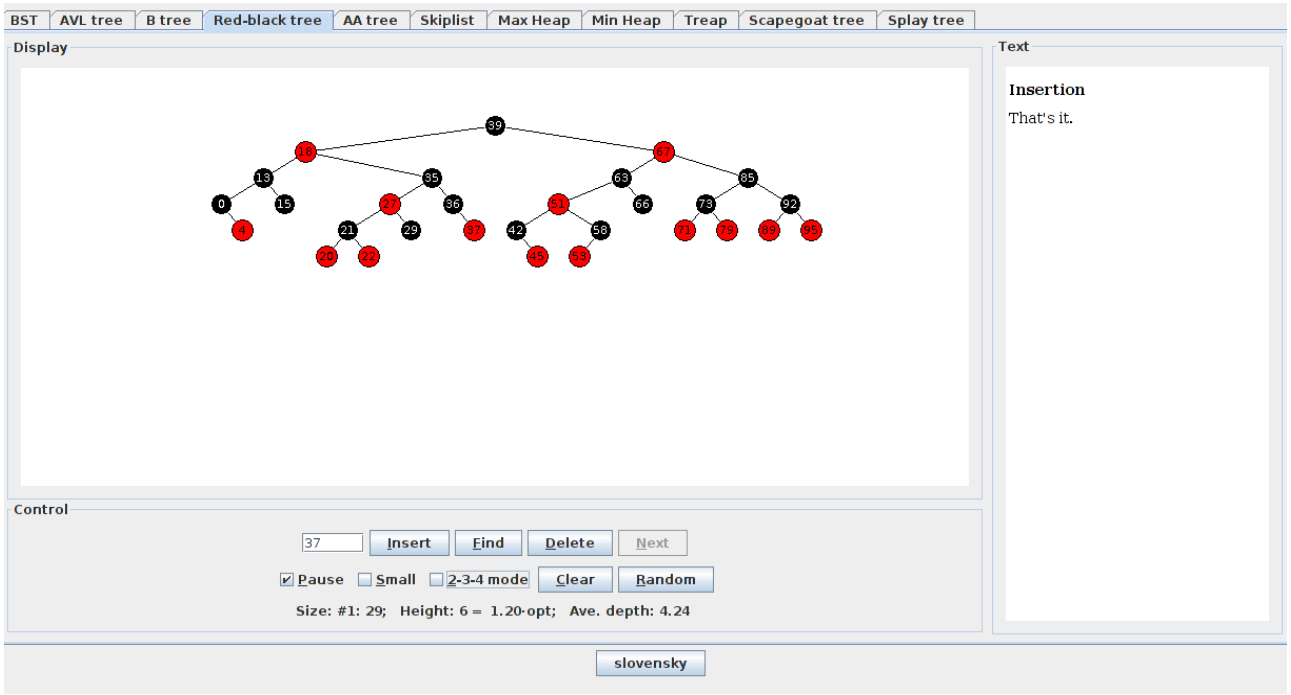
$$= A \cdot (\Phi_n - \Phi_0) + B \cdot n \quad \text{// Teleskopsumme}$$

$$= A \cdot \Phi_n + B \cdot n \quad \text{// leerer Baum am Anfang (kein Grad-3-Knoten)}$$

$$< A \cdot n + B \cdot n = \mathcal{O}(n) \quad \text{// Anzahl Grad-3-Knoten immer < Anzahl Elemente}$$

□

Rot-Schwarz-Bäume



- Binärbaum mit schwarzen und roten Knoten
- Anzahl der schwarzen Knoten auf Pfad zu jedem Blatt gleich
- Kann direkt als (2,4)-Baum (auch 2-3-4 Baum genannt) interpretiert werden
  - jeder (2,4)-Baum-Knoten ist ein kleiner Rot-schwarz-Baum mit schwarzem „Einstiegsknoten“

Übungsblatt 11

- Beweis amortisiertes O(1) für (4,9) Bäume

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
  - In Wikipedia

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

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