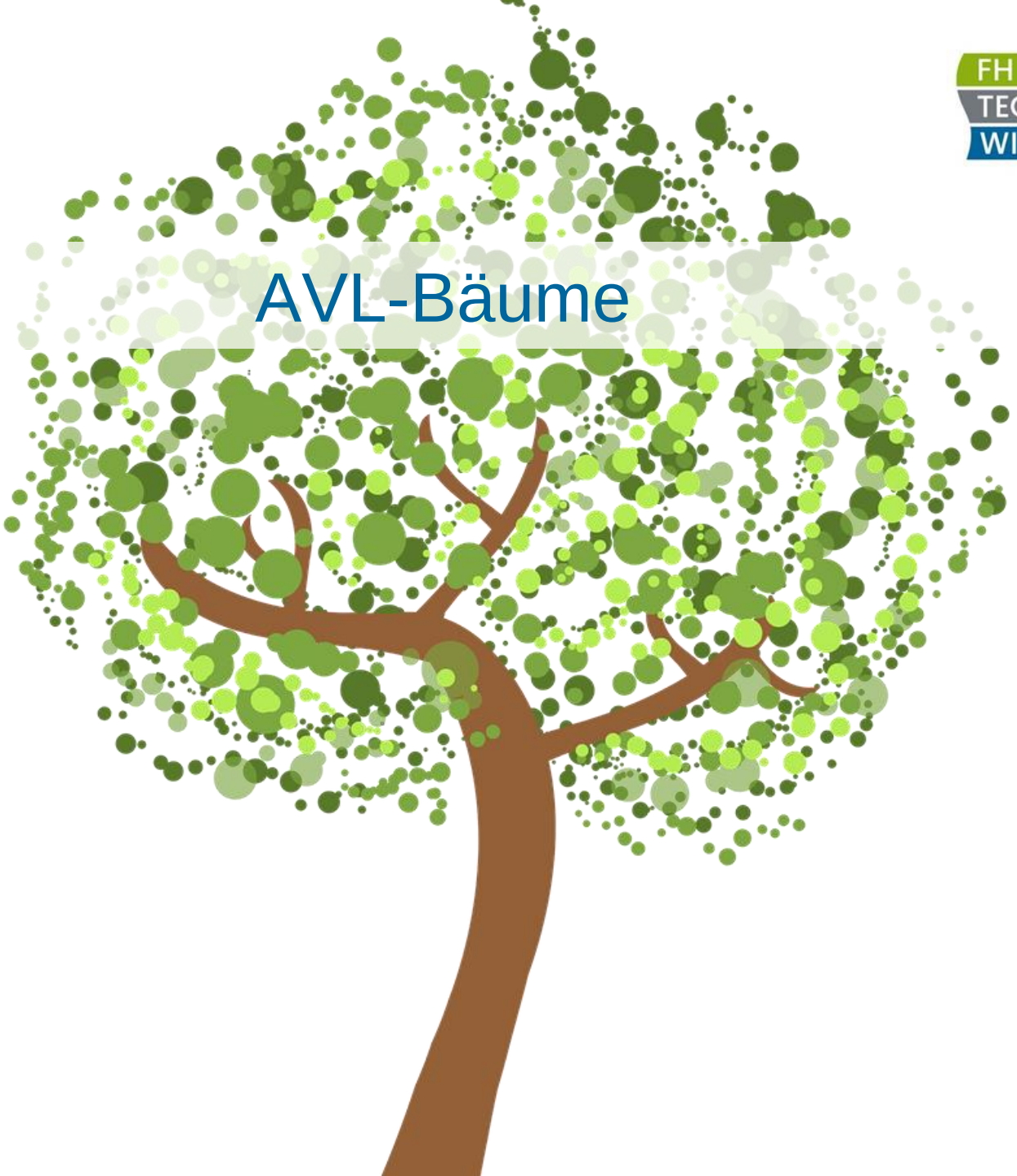
Alexander Nimmervoll - (A/B)
Eugen Kaltenegger - (C/D)

Algorithmen und Datenstrukturen
Sommersemester 2019

Agenda

- **AVL-Bäume**
- **B-Bäume**
- **B*-Bäume**

AVL-Bäume

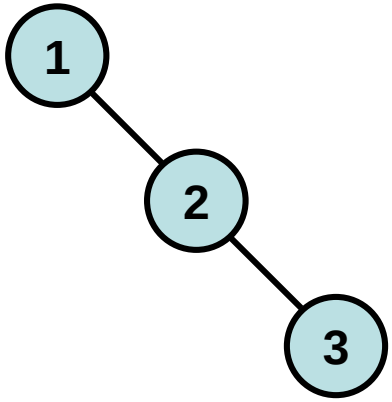


Motivation

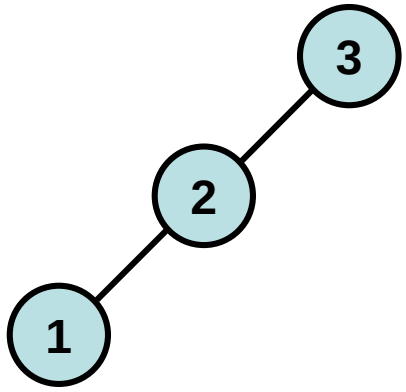
Problem:

Durch ungünstige Reihenfolge der Schlüssel kann ein Baum zu einer linearen Kette entarten daraus folgt, dass die Vorteile des Baumes verloren gehen.

Eingabefolge 1-2-3



Eingabefolge 3-2-1



Motivation

Problem:

Durch ungünstige Reihenfolge der Schlüssel kann ein Baum zu einer linearen Kette entarten daraus folgt, dass die Vorteile des Baumes verloren gehen.

Lösung:

Ausgeglichene bzw. Balancierte Bäume:

- Vollständig ausgeglichene Bäume nur schwer möglich
- Aufwändige Umordnung beim Einfügen und Löschen von Elementen

Praxis:

- Abgeschwächte Kriterien
- AVL-Bäume und B/B*-Bäume

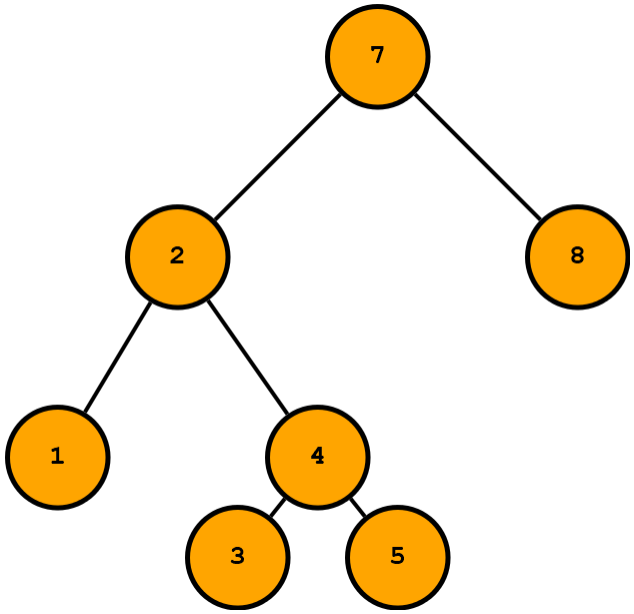
AVL-Bäume

Definition:

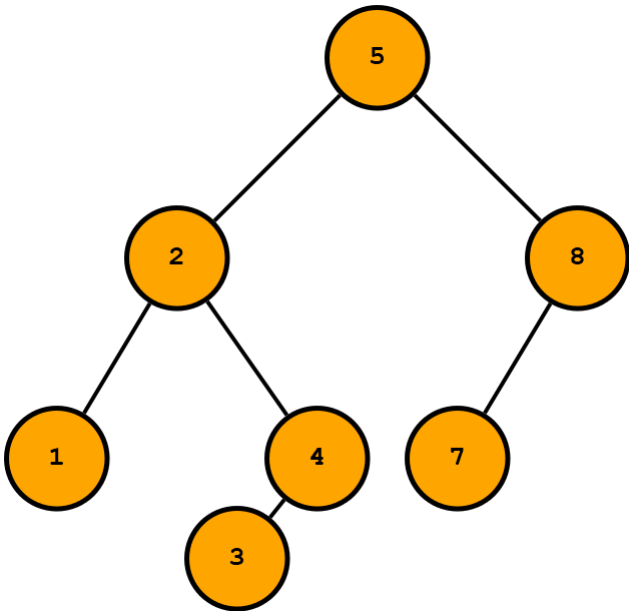
- Ein binärer Baum heißt ausgeglichener Baum oder AVL-Baum (nach **A**delson-**V**elskij und **L**andis), falls sich für jeden Knoten k die Höhen h der beiden Teilbäume um höchstens 1 unterscheiden
- Aufwand beim Suchen maximal $1{,}44 \log_2 n$

AVL-Bäume

kein AVL-Baum



AVL-Baum



AVL-Bäume

Balance-Faktor

Der Balance-Faktor eines Knoten k in einem binären Baum beschreibt die Höhendifferenz des rechten und linken Teilbaumes (TB):

$h_r(k)$ … **Höhe des rechten Teilbaumes von k**

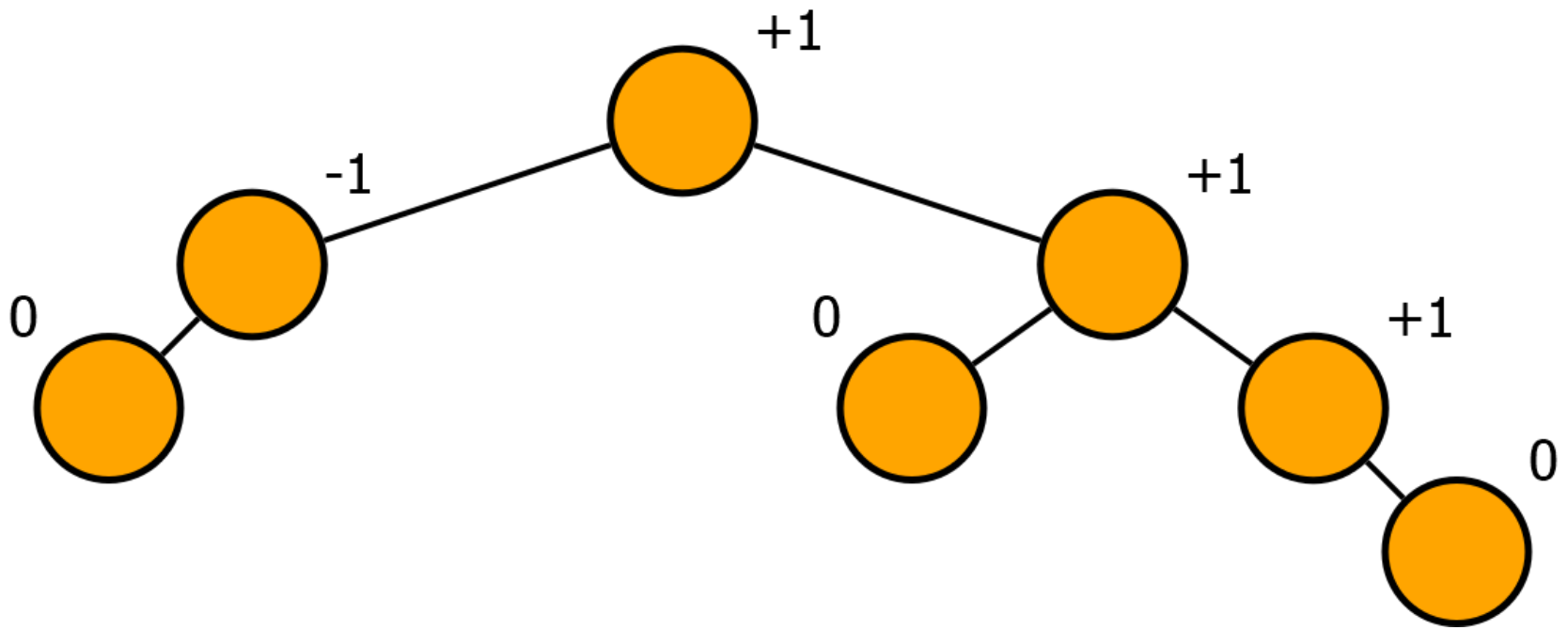
$h_l(k)$ … **Höhe des linken Teilbaumes von k**

$$\mathbf{bal(k)} = h_r(k) - h_l(k)$$

Für AVL-Bäume gilt:

$$\mathbf{bal(k)} \in \{-1, 0, +1\}$$

AVL-Bäume



AVL-Bäume

Rotation:

Es gibt zwei Arten von Rotationen:

- Einfachrotationen
  (R-Rotation, L-Rotation)
- Zweifachrotationen
  (RL-Rotation, LR-Rotation)

Diese dienen dazu einen Baum nach verlust der Blance (durch Einfügen oder Löschen eines Knotens) wieder herzustellen.

AVL-Bäume

Einfügen von Knoten:

Nach dem Einfügen eines Knotens muss die Balance des Baumes geprüft werden. Balance Werte vom eingefügten Knoten bis zur Wurzel müssen geprüft werden. Mit Hilfe von höchstens einer Rotation wird die Balance wieder hergestellt.

Worst-Case Laufzeit: $\mathcal{O}(\log n)$

Löschen von Knoten:

Nach dem Löschen eines Knotens muss die Balance des Baumes geprüft werden. Balance Werte vom eingefügten Knoten bis zur Wurzel müssen geprüft werden. Mit Hilfe von Rotationen (maximal für jeden Knoten vom gelöschten Knoten bis zur Wurzel) wird die Balance wieder hergestellt.

Worst-Case Laufzeit: $\mathcal{O}(\log n)$

AVL-Bäume

Durch das Einfügen und Löschen von Knoten kann ein Balance Faktor im Bereich

$$\mathbf{bal(k)} \in \{-\mathbf{2}, +\mathbf{2}\}$$

auftreten. Durch eine oder mehrere Rotationen soll dieses Problem behoben werden.

Fakten zur Rotation:

- Bei der Rotation kommt es zu einer vertikalen Verschiebung der involvierten Knoten
- Bei Einfachrotationen werden 3 Verknüpfungen geändert
- Bei Doppelrotationen werden 5 Verknüpfungen geändert
- Rotationen haben eine Konstante Laufzeit: $\mathcal{O}(1)$

AVL-Bäume

Überblick zu den möglichen Rotationen

$\mathbf{bal}(\boldsymbol{u}) = 2$ und $\boldsymbol{v}$ ist das rechte Kind von $\boldsymbol{u}$ und

- $\mathbf{bal}(\boldsymbol{v}) \in \{\mathbf{0}, \mathbf{1}\}$ Einfachrotation L
- $\mathbf{bal}(\boldsymbol{v}) = -\mathbf{1}$ Doppelrotation RL

$\mathbf{bal}(\boldsymbol{u}) = -2$ und $\boldsymbol{v}$ ist das linke Kind von $\boldsymbol{u}$ und

- $\mathbf{bal}(\boldsymbol{v}) \in \{-\mathbf{1}, \mathbf{0}\}$ Einfachrotation R
- $\mathbf{bal}(\boldsymbol{v}) = \mathbf{1}$ Doppelrotation LR

AVL-Bäume

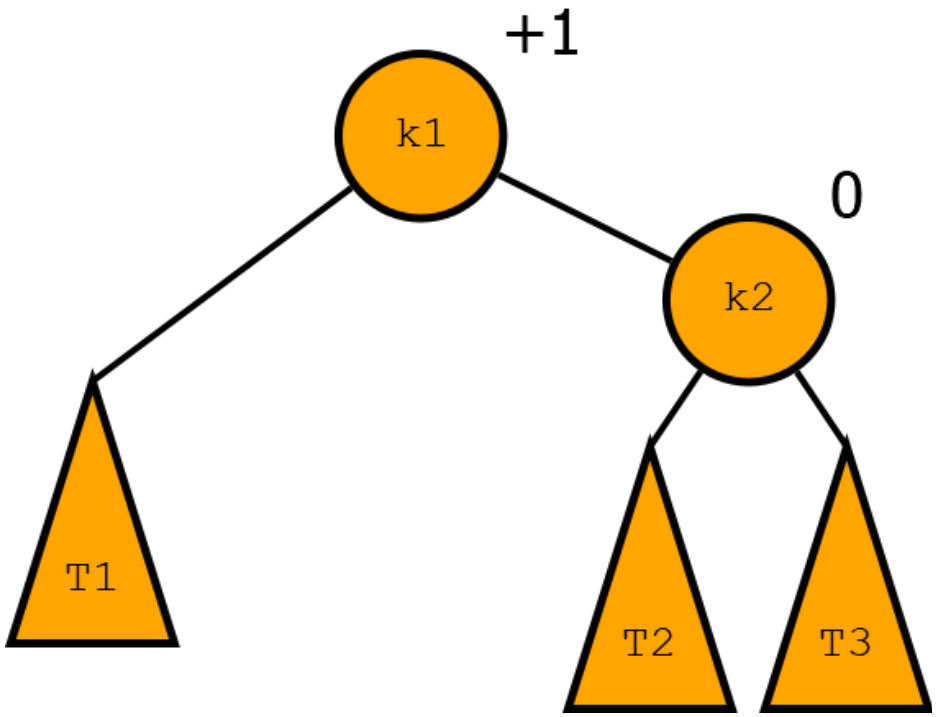
Beispiel L-Rotation

Im folgenden Beispiel wird ein Knoten x eingefügt und verletzt dadurch die Ausgeglichenheit des Baumes an einem höher gelegenem Knoten k_1.

Notwendige Korrektur: **L-Rotation**

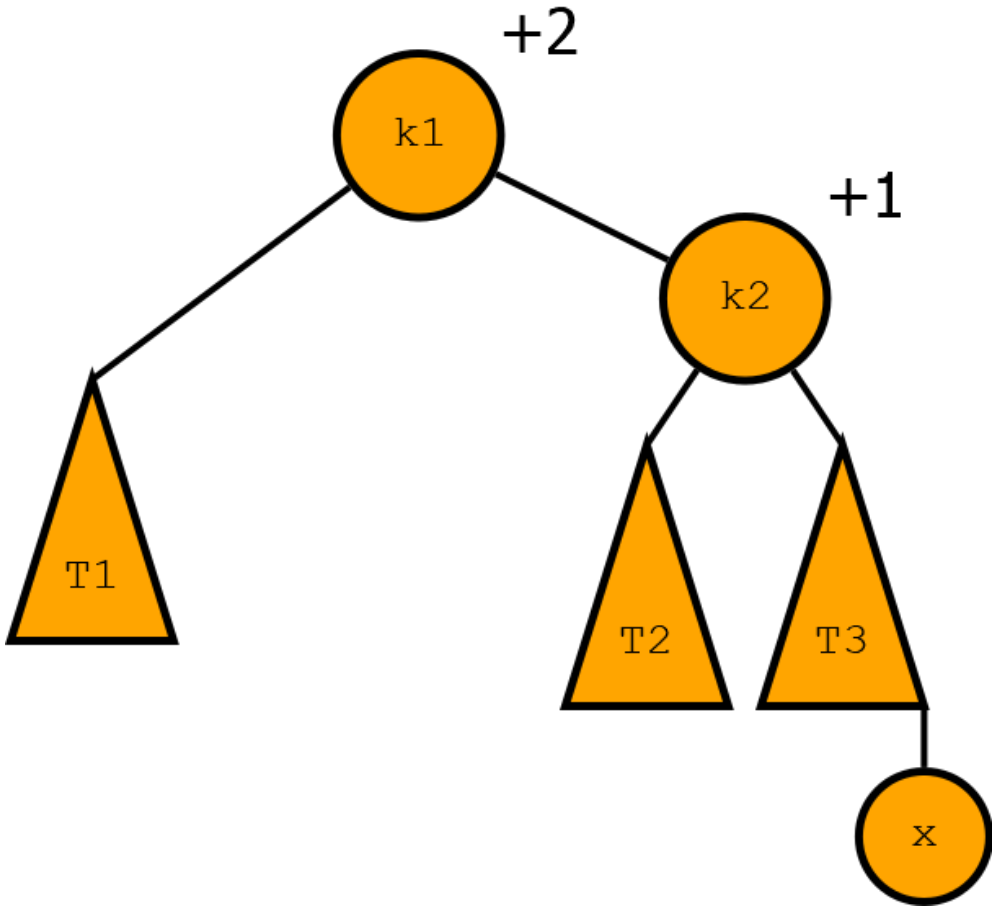
AVL-Bäume

L-Rotation



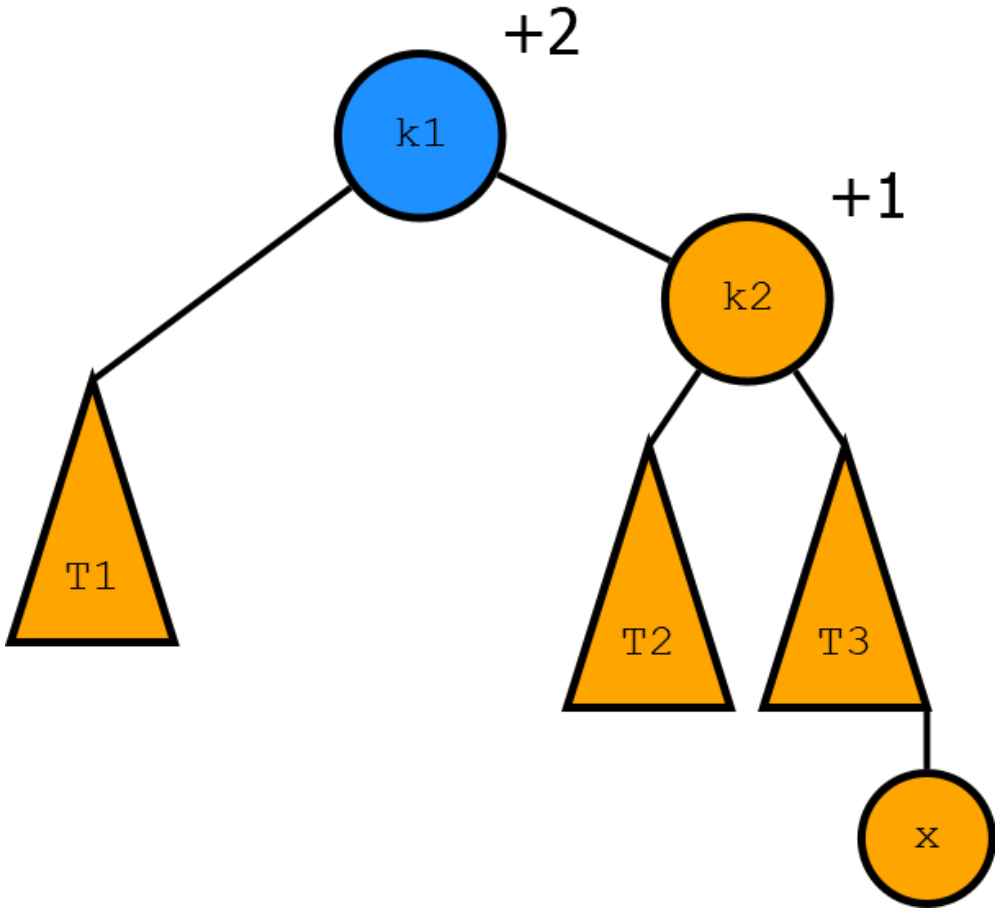
AVL-Bäume

L-Rotation



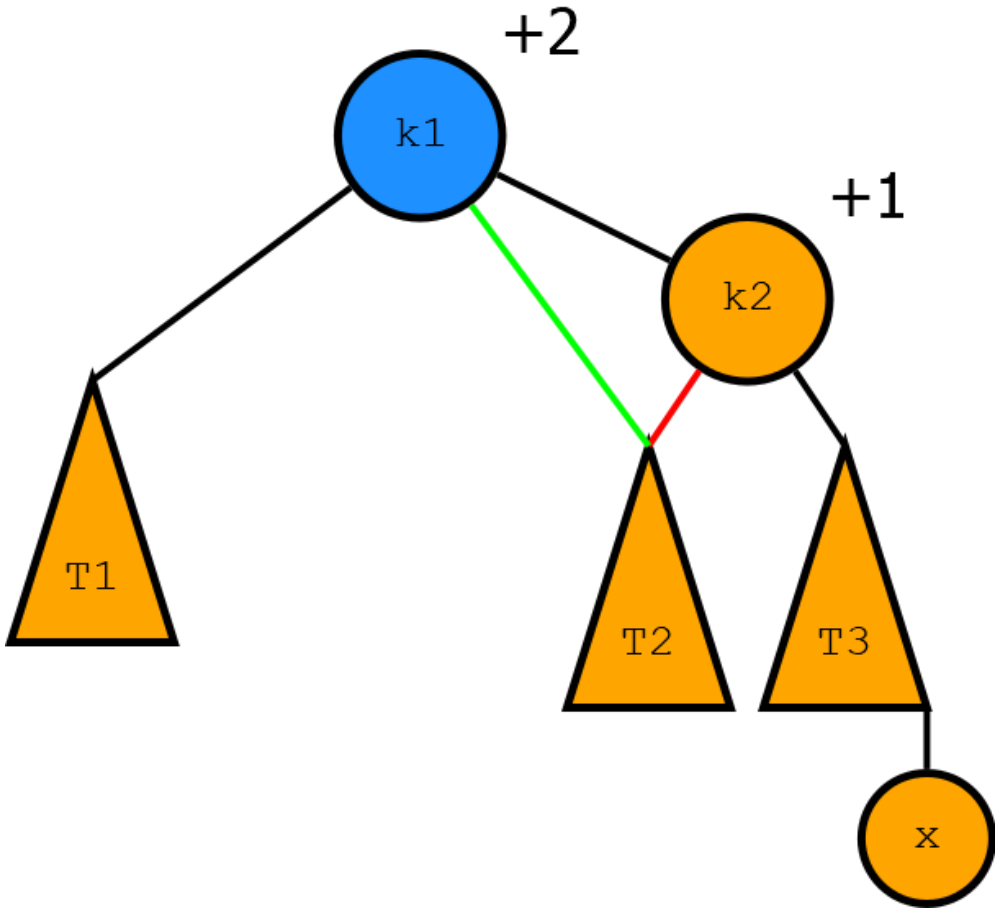
AVL-Bäume

L-Rotation



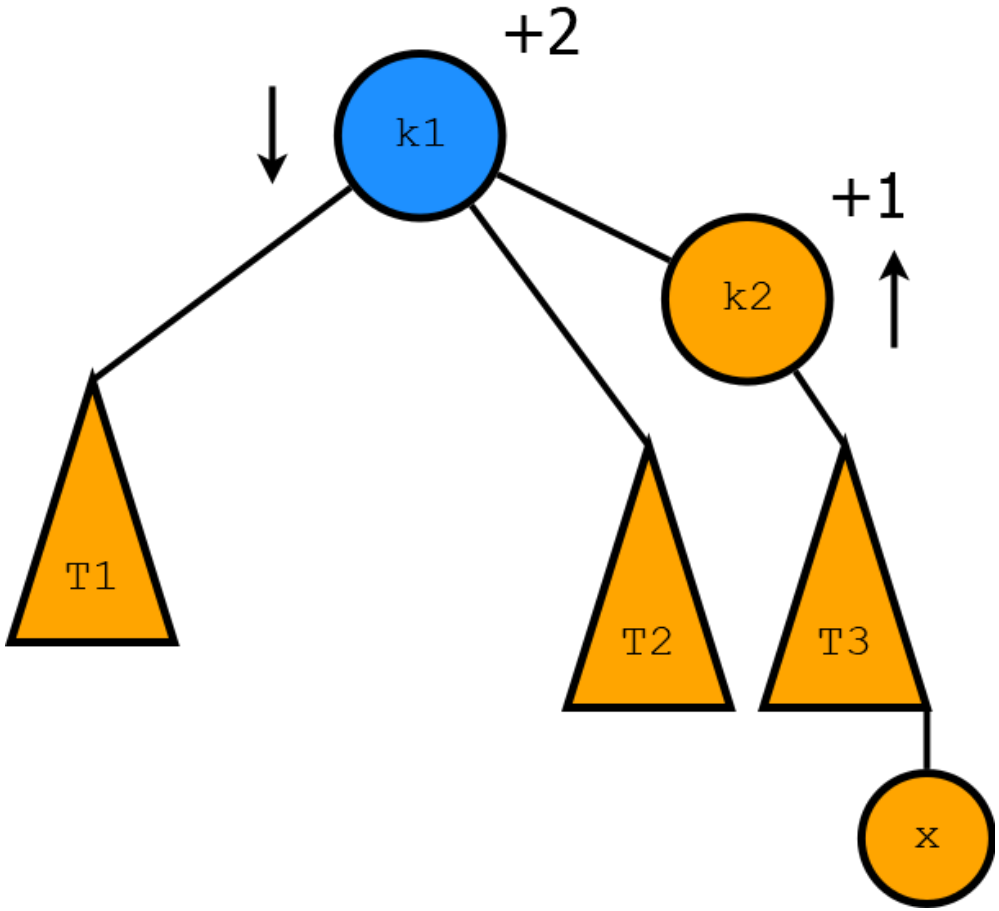
AVL-Bäume

L-Rotation



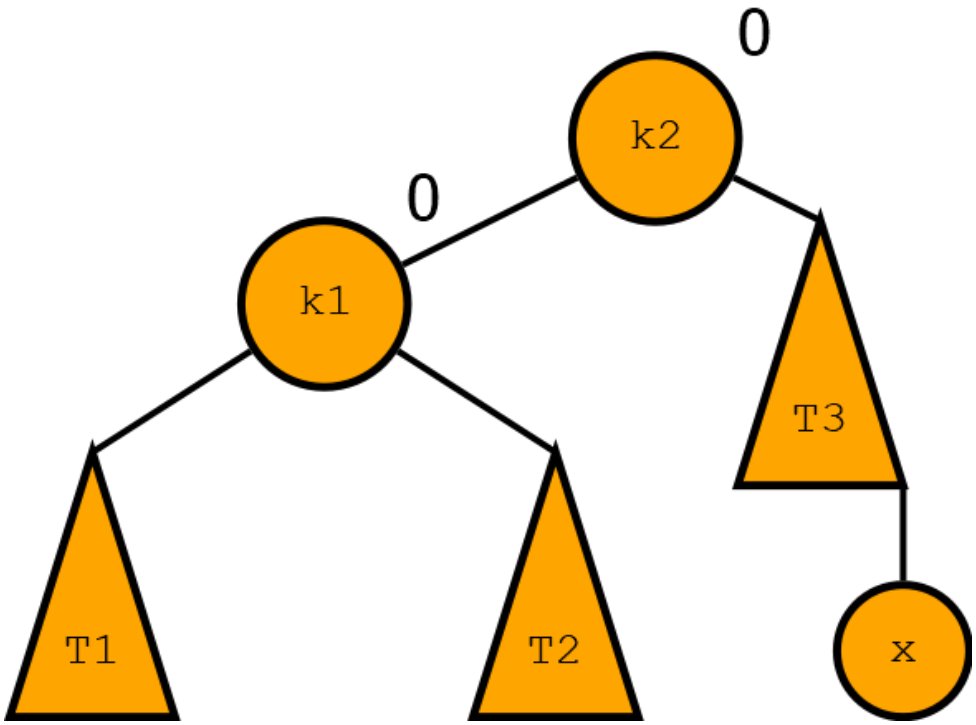
AVL-Bäume

L-Rotation



AVL-Bäume

L-Rotation



AVL-Bäume

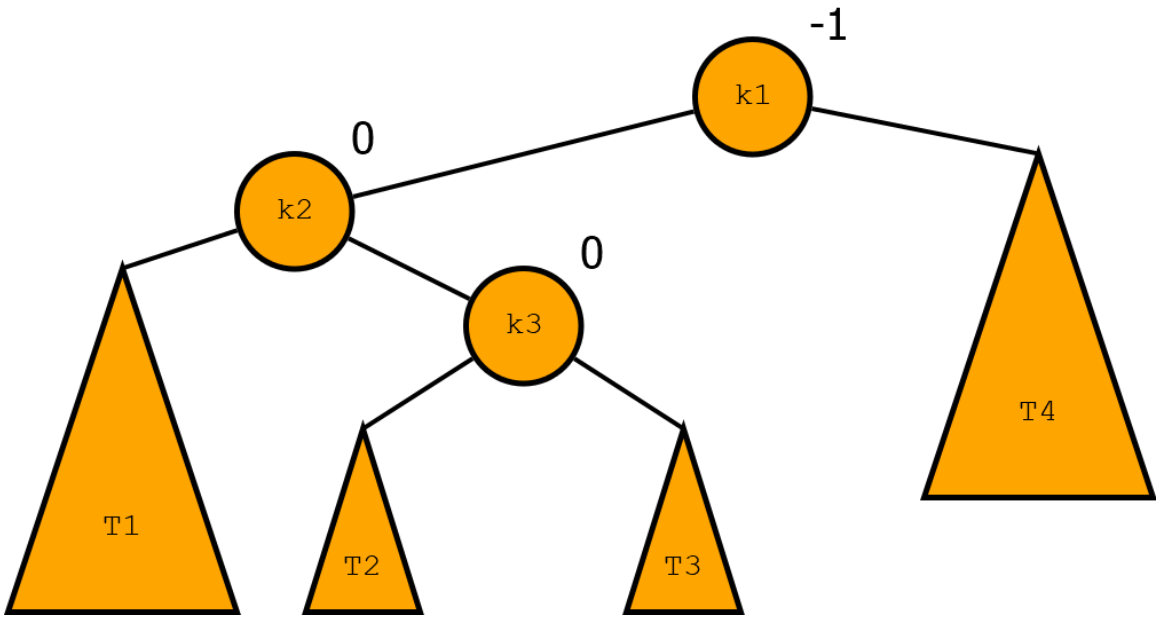
Beispiel LR-Rotation

Im folgenden Beispiel wird ein Knoten x eingefügt und verletzt dadurch die Ausgeglichenheit des Baumes an einem höher gelegenem Knoten k_1.

Notwendige Korrektur: **LR-Rotation**

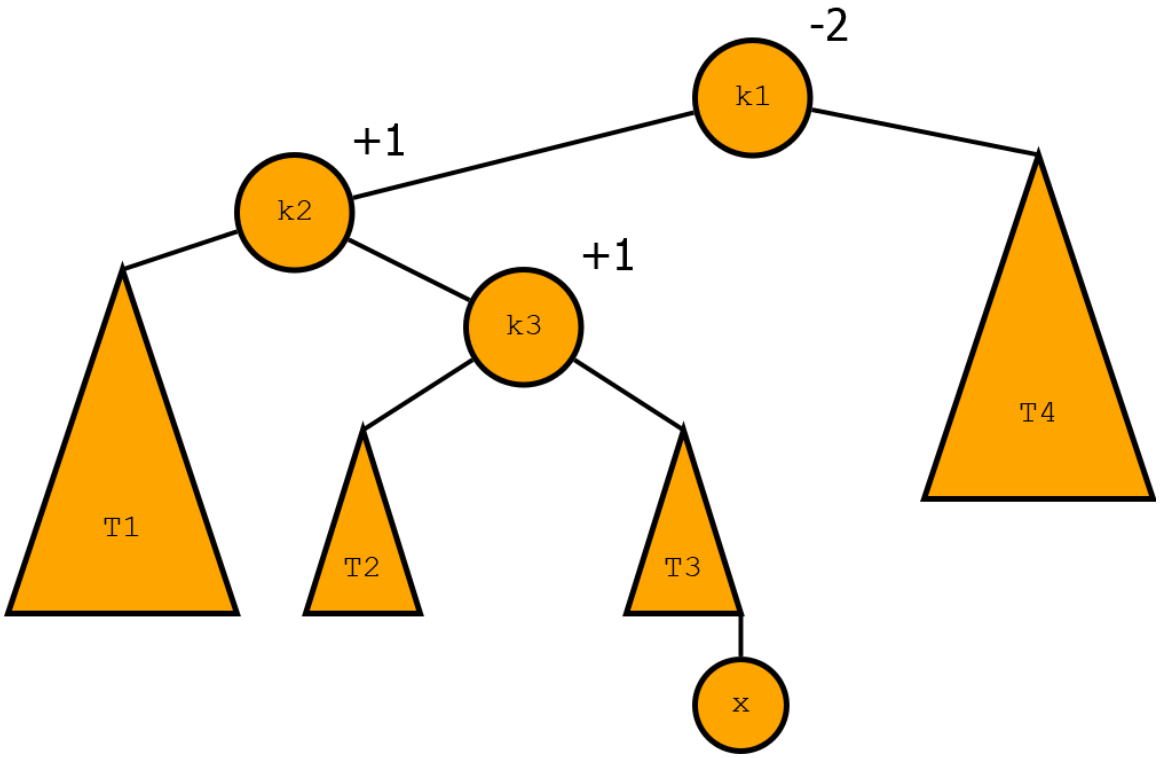
AVL-Bäume

LR-Rotation



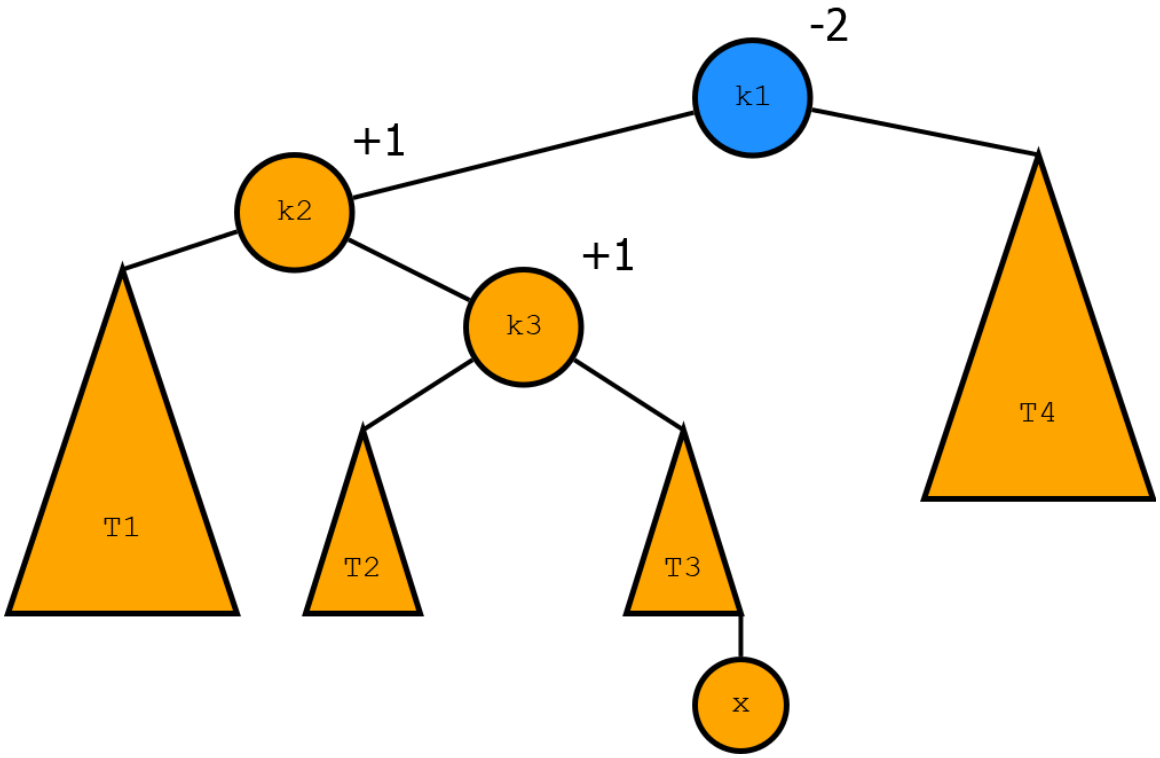
AVL-Bäume

LR-Rotation



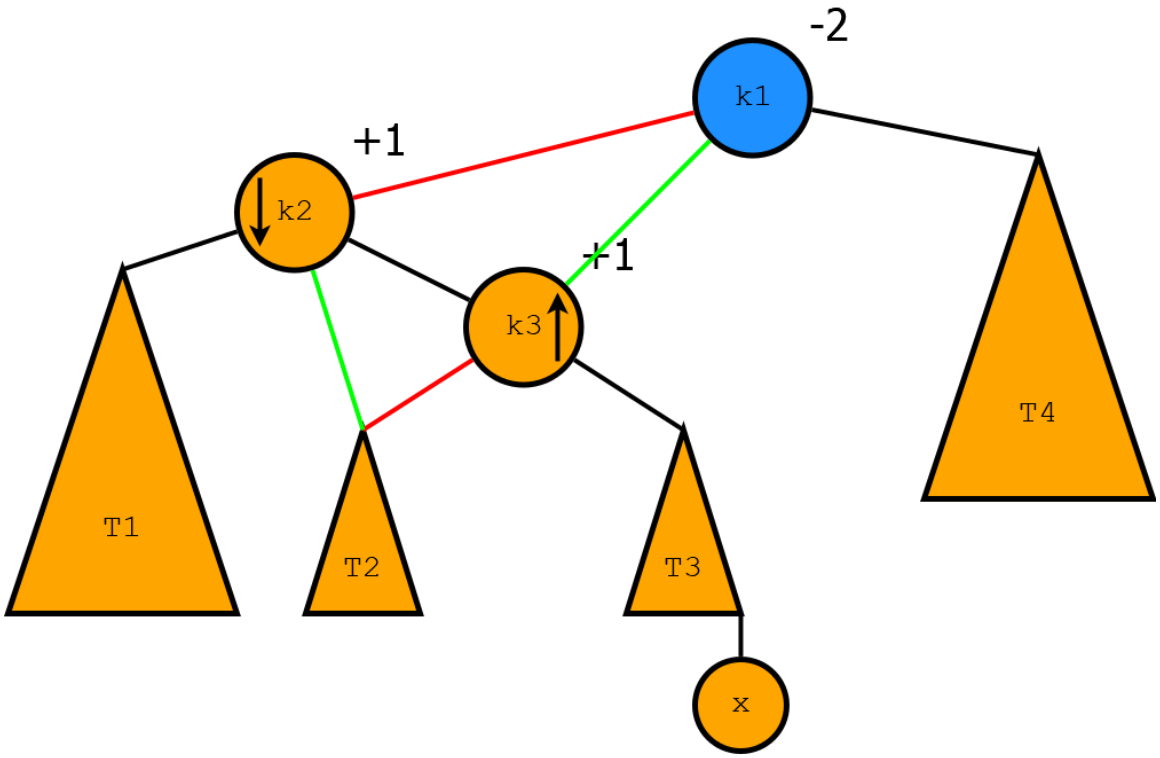
AVL-Bäume

LR-Rotation



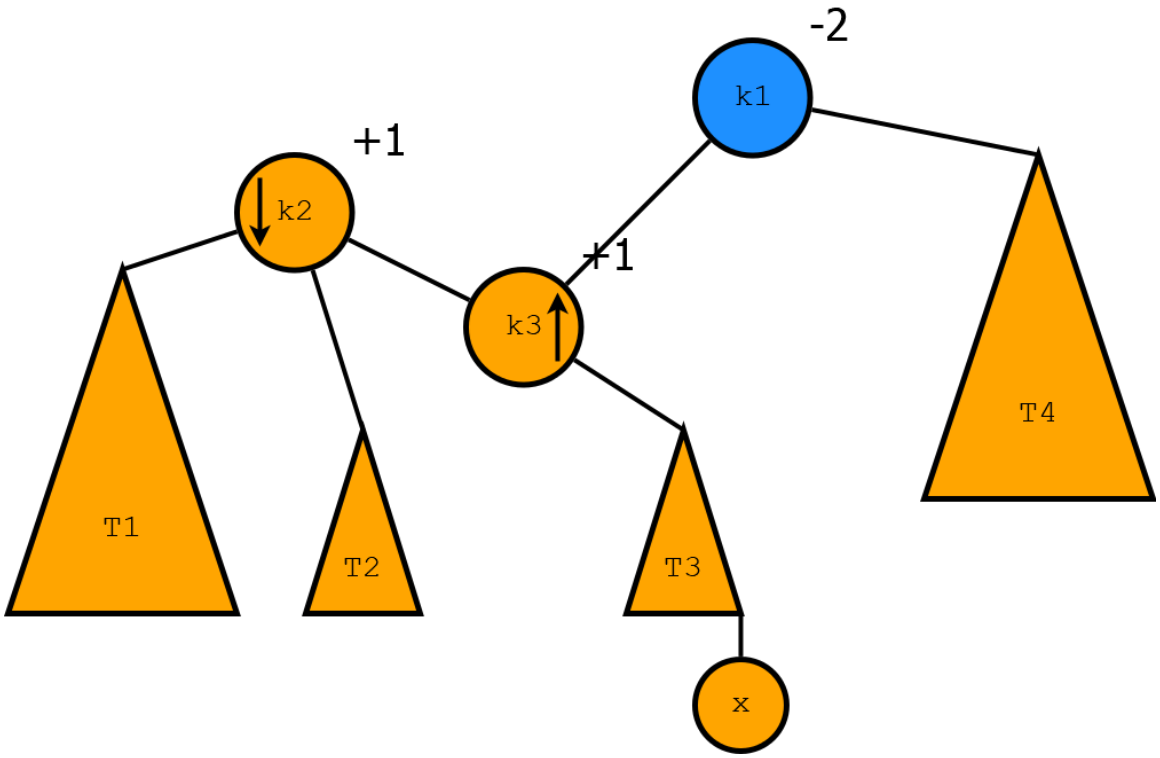
AVL-Bäume

LR-Rotation



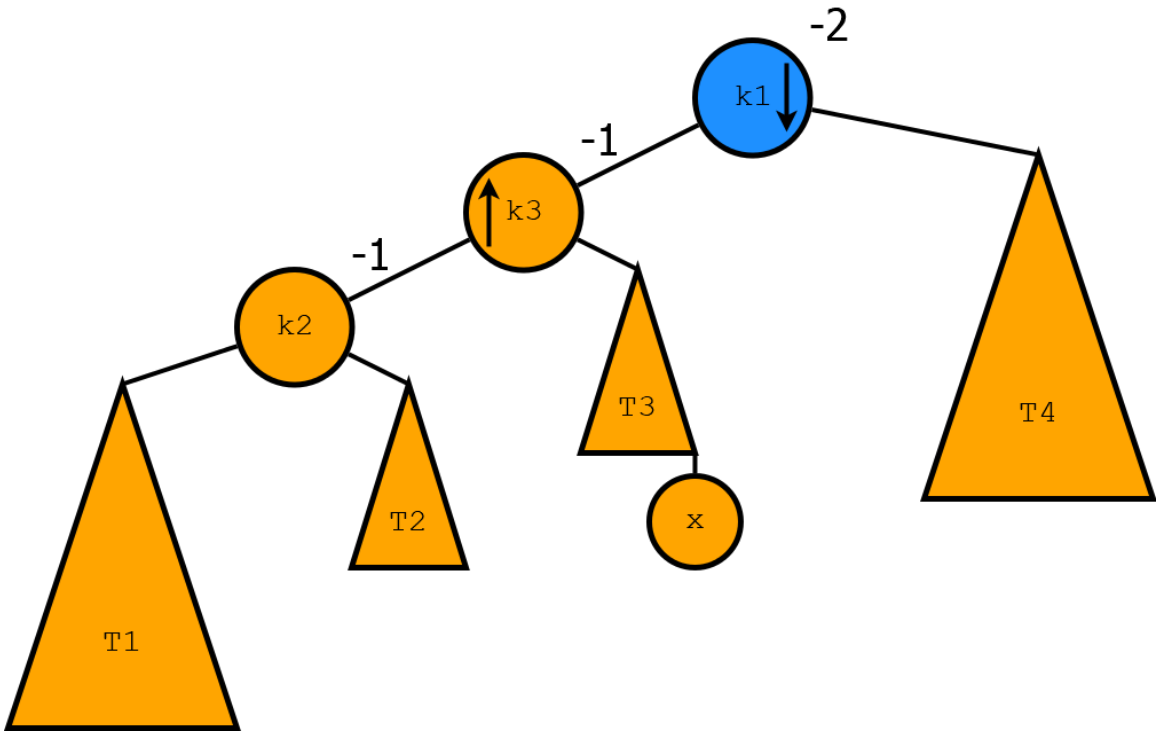
AVL-Bäume

LR-Rotation



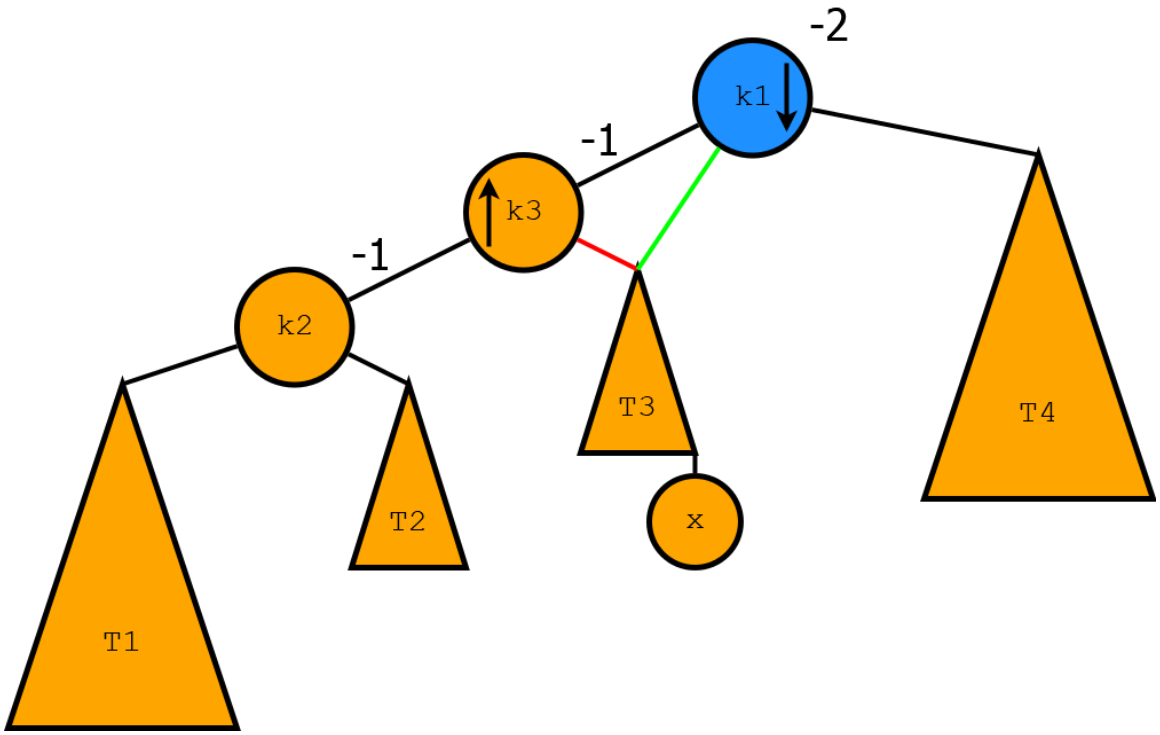
AVL-Bäume

LR-Rotation



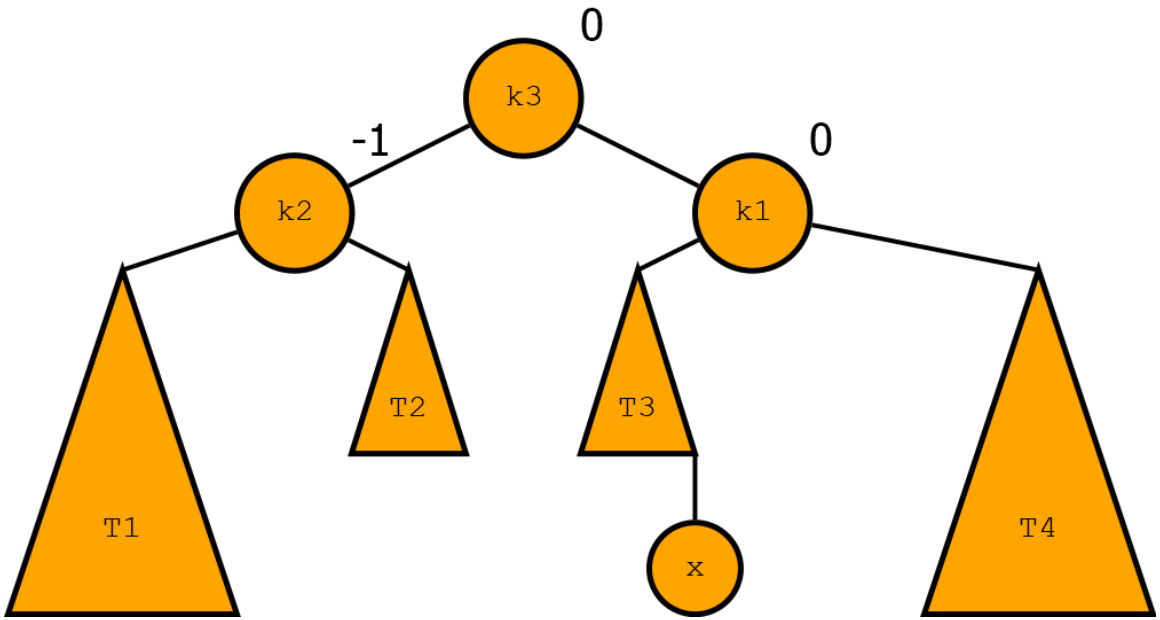
AVL-Bäume

LR-Rotation



AVL-Bäume

LR-Rotation



AVL-Bäume

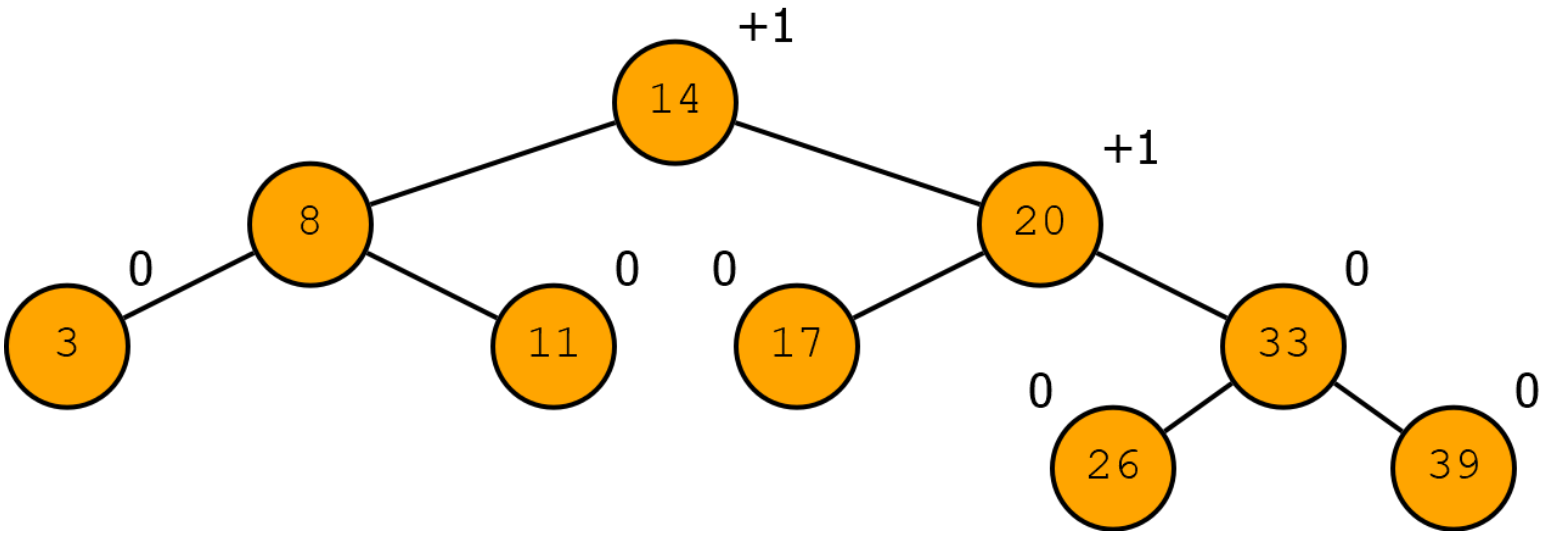
Es folgt ein Beispiel auf einem AVL-Tree:

- Hinzufügen von einem Element mit Key 30
- Entfernen des Element mit Key 8
- Entfernen des Element mit Key 11

Eigenstudium!

ERINNERUNG: VisuAlgo.net

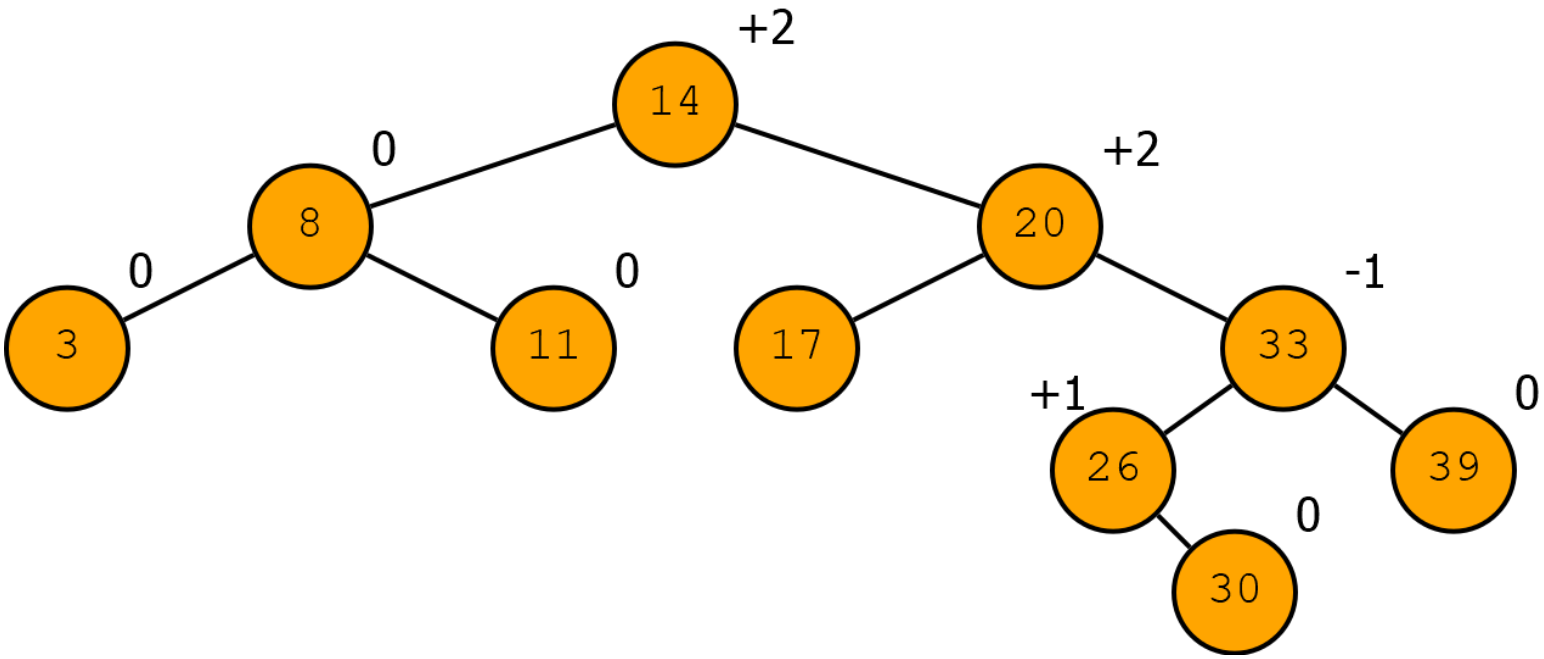
AVL-Bäume



Ausgangssituation

Ziel: Füge Knoten mit Key 30 ein

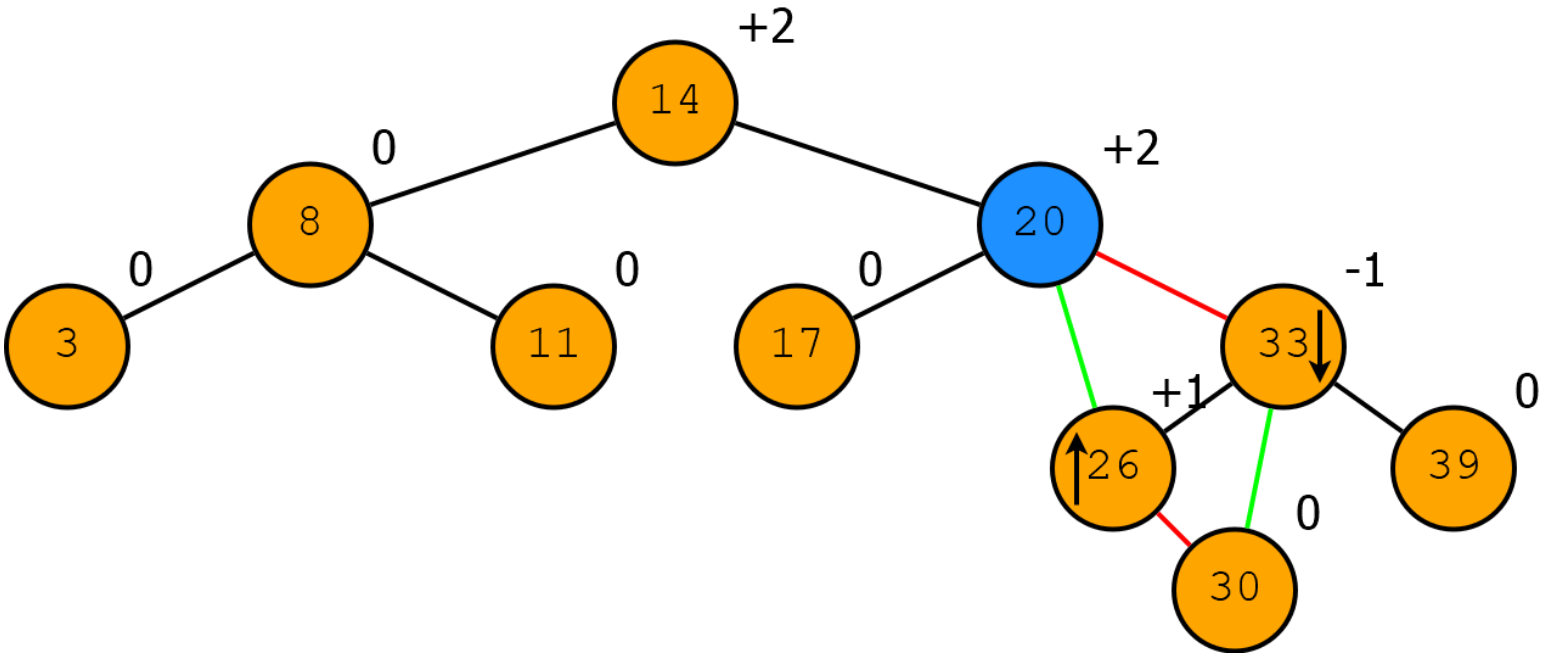
AVL-Bäume



Einfügen von Knoten mit Key 30

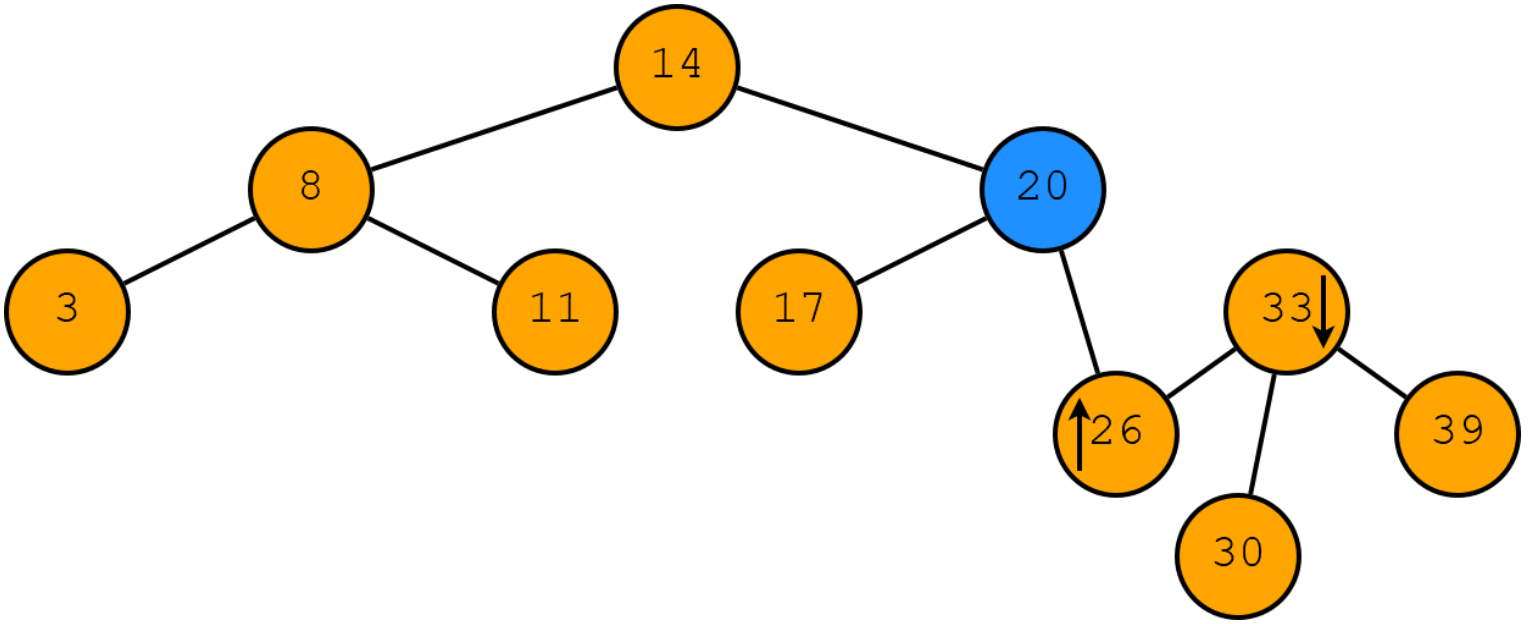
Rebalancing notwendig

AVL-Bäume



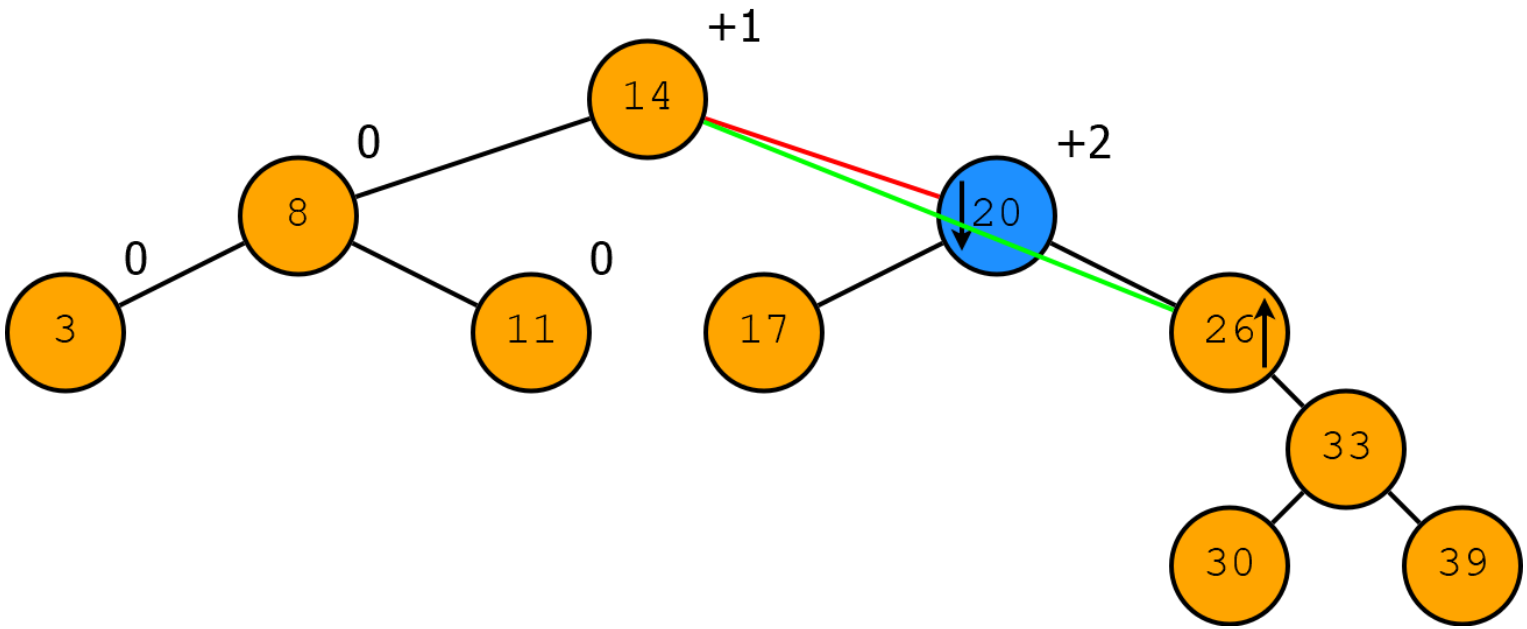
Erste Teilrotation

AVL-Bäume



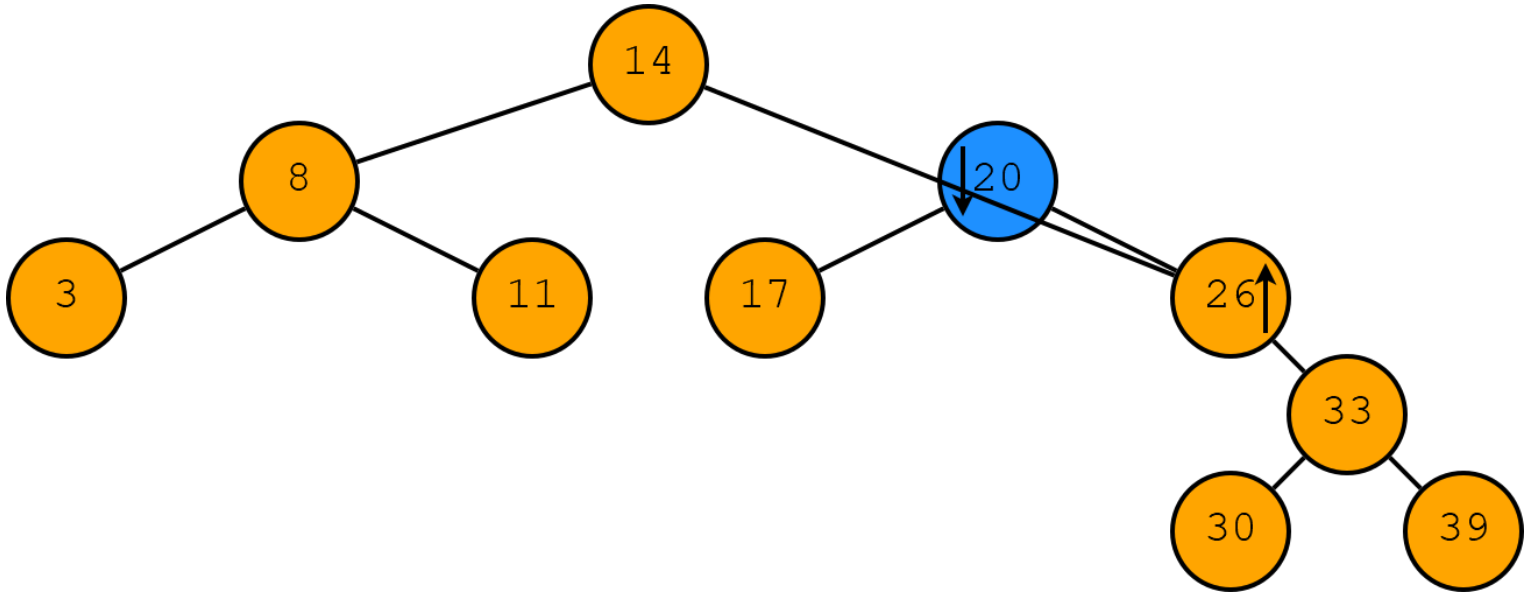
Erste Teilrotation

AVL-Bäume



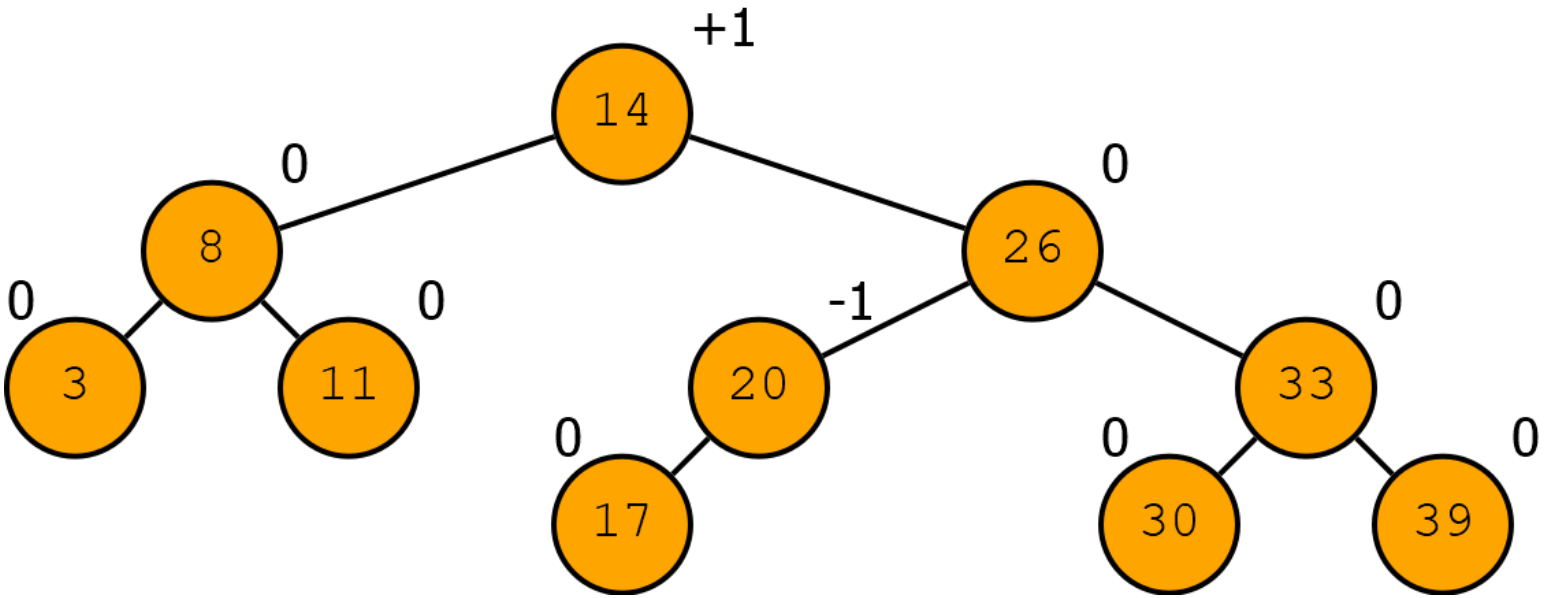
Zweite Teilrotation

AVL-Bäume



Zweite Teilrotation

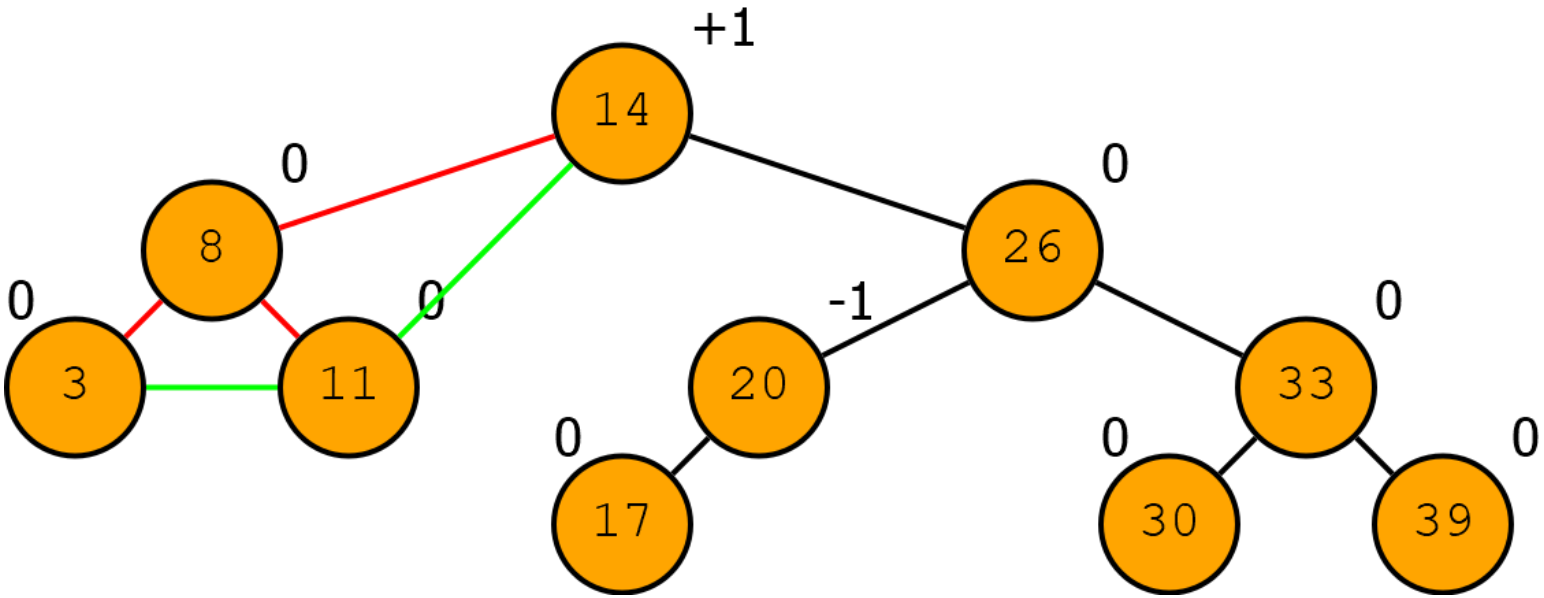
AVL-Bäume



Ausgangssituation

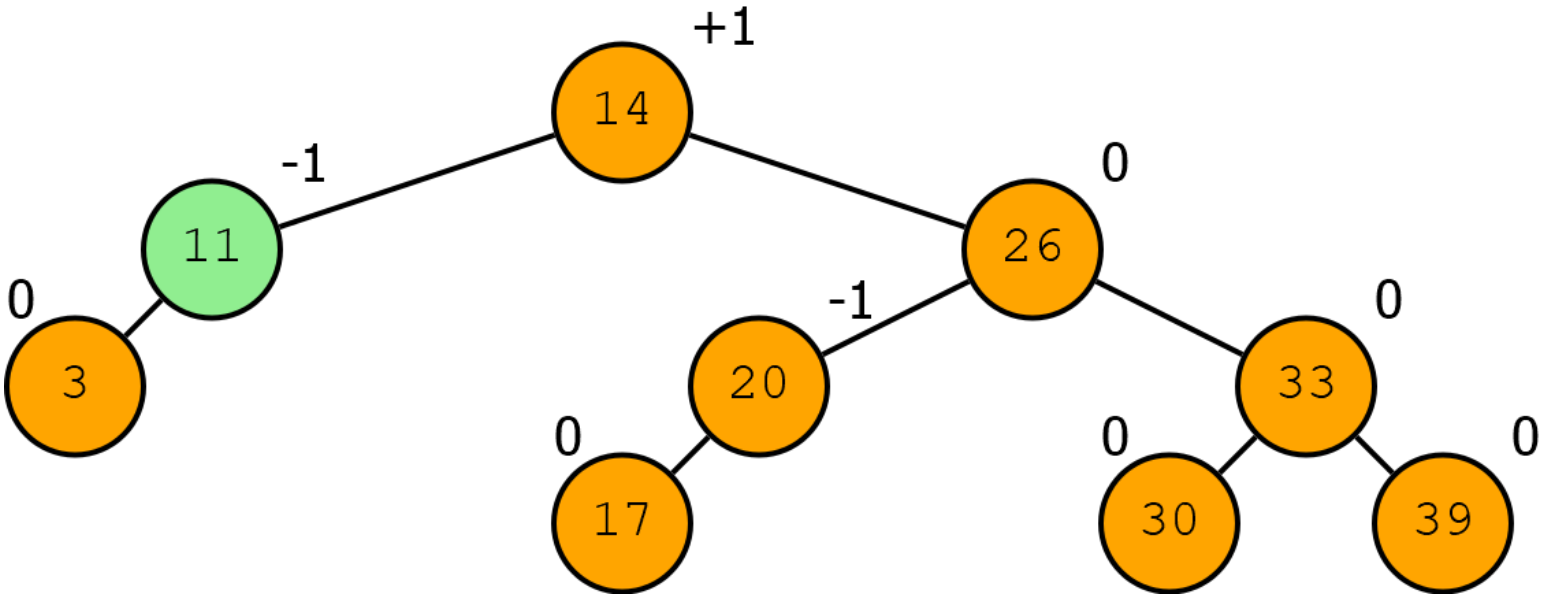
Ziel: Entferne Knoten mit Key 8

AVL-Bäume



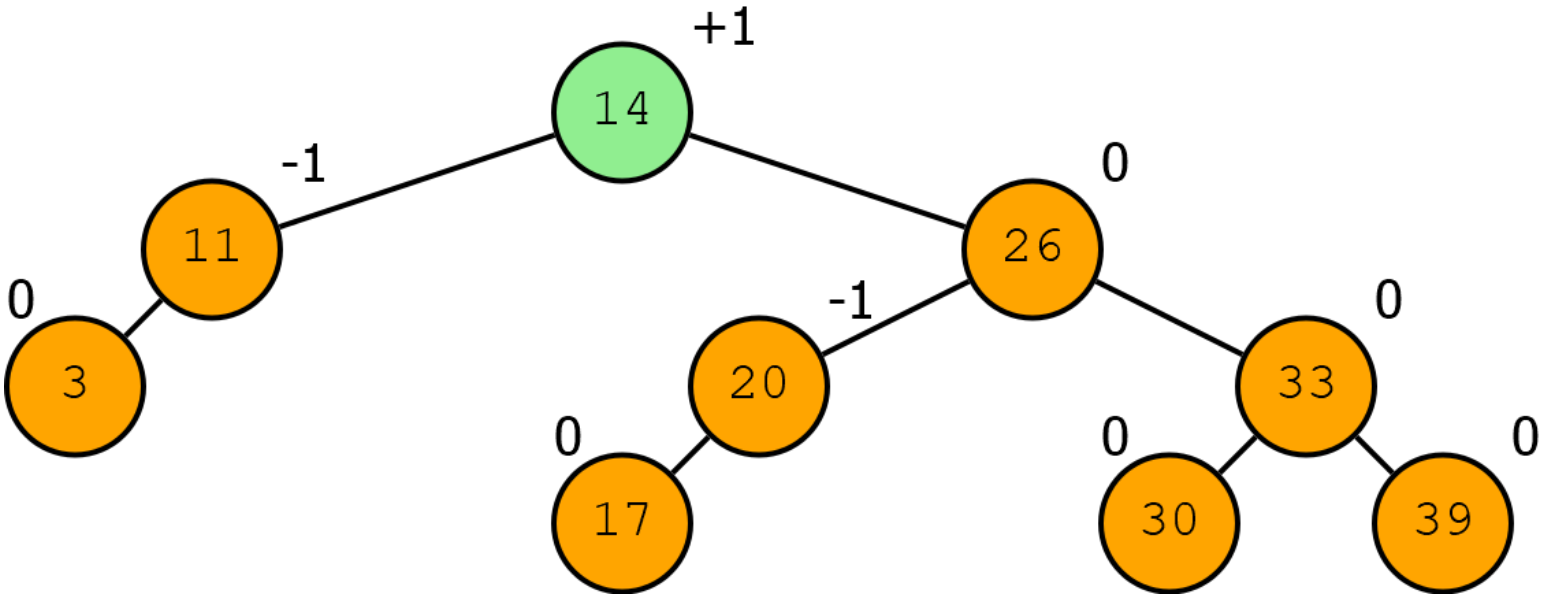
Entferne Knoten mit Key 8

AVL-Bäume



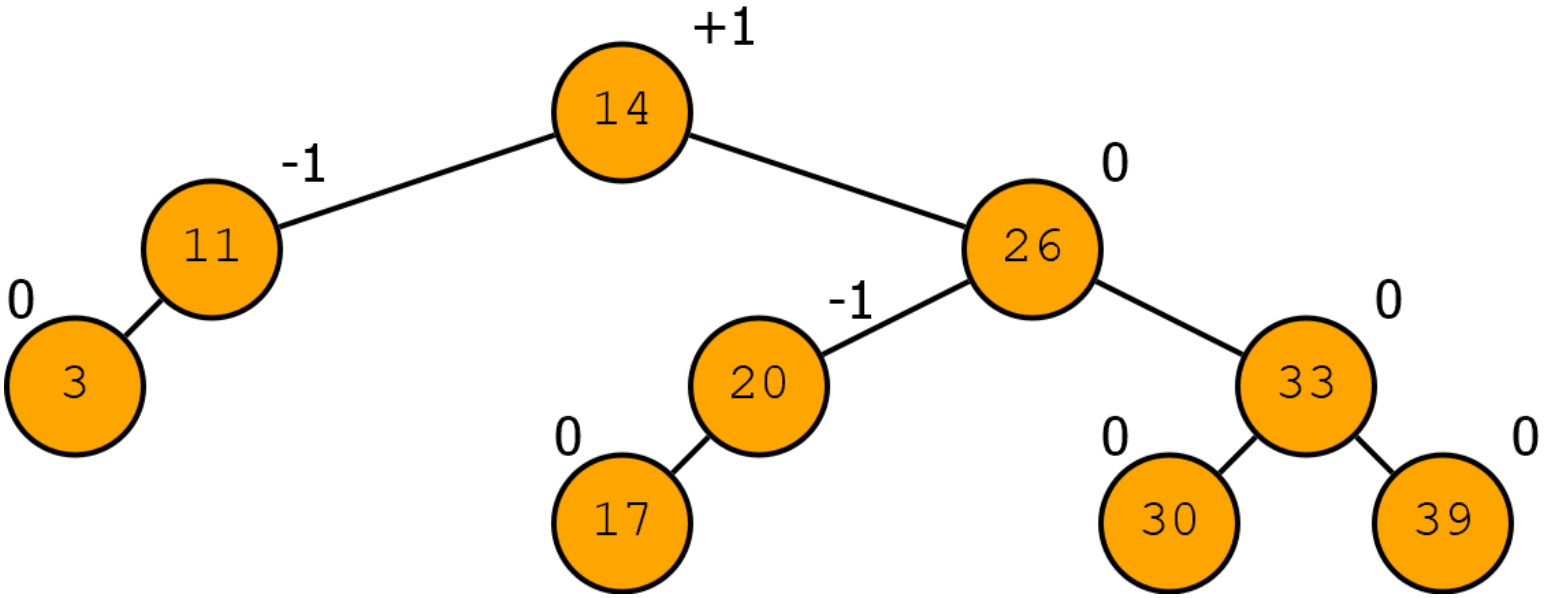
Überprüfe auf Verletzung der Balance

AVL-Bäume



Überprüfe auf Verletzung der Balance

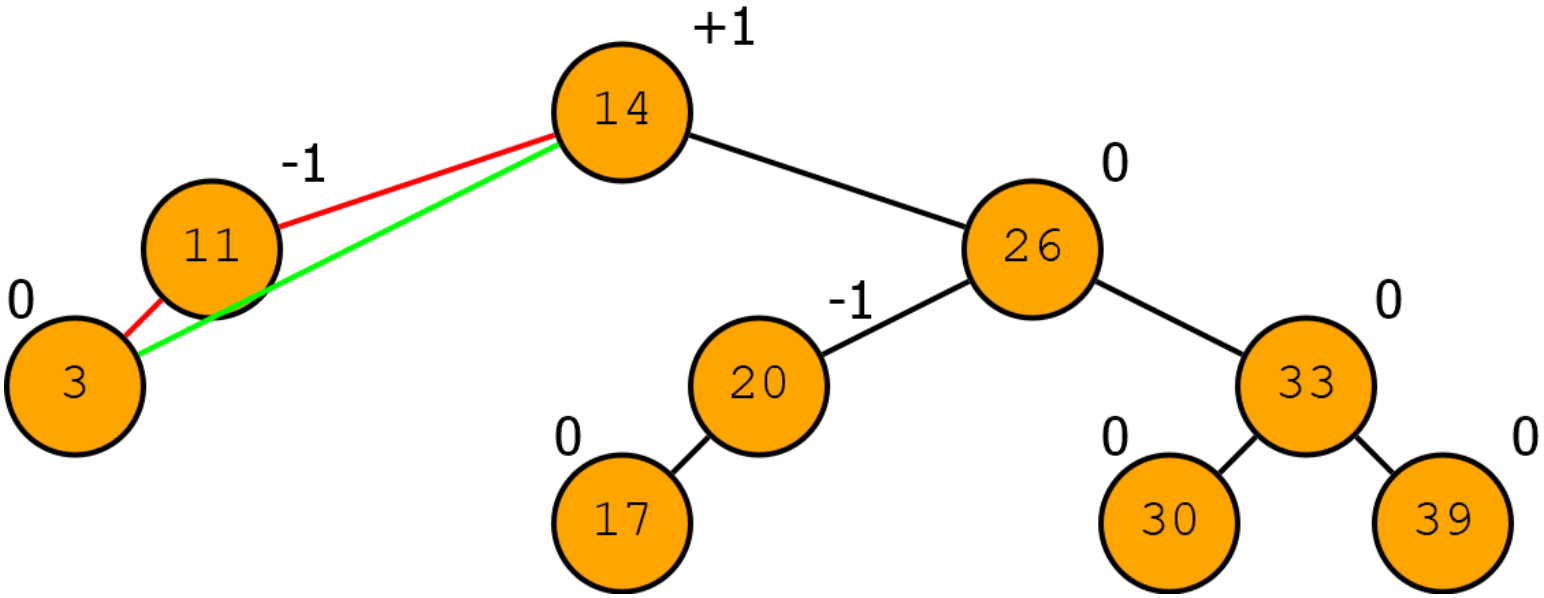
AVL-Bäume



Ausgangssituation

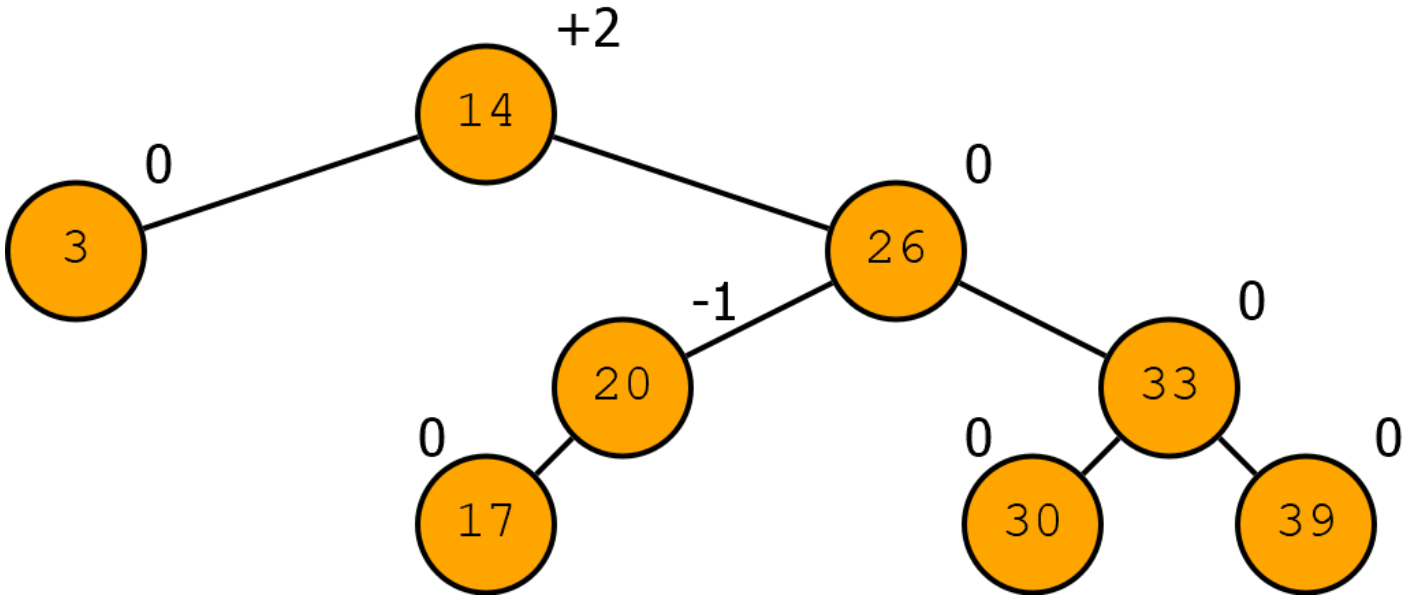
Ziel: Entferne Knoten mit Key 11

AVL-Bäume



Entferne Knoten mit Key 11

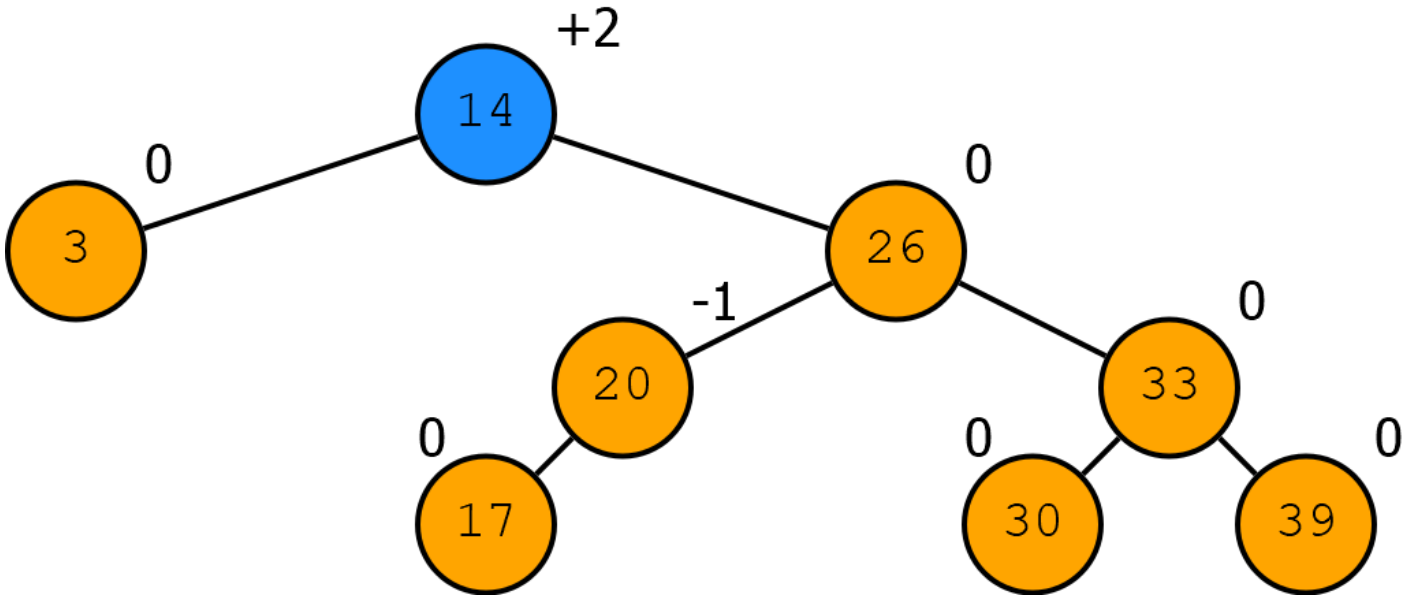
AVL-Bäume



Knoten entfernt

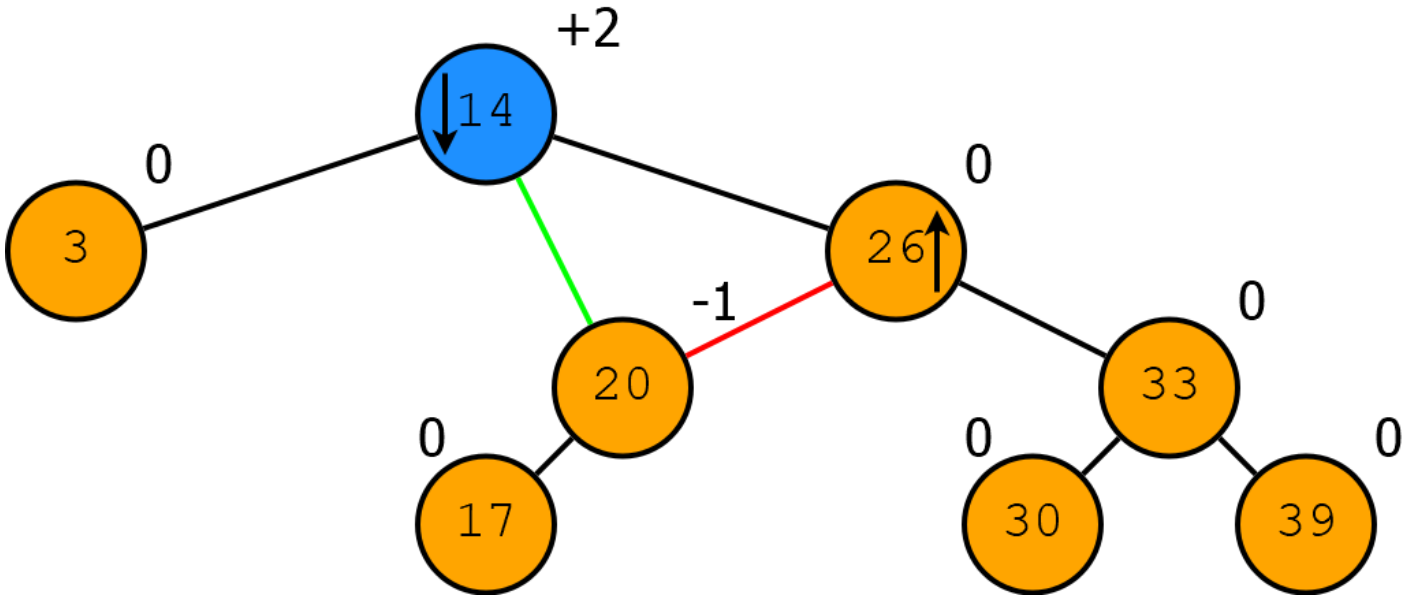
Rebalancing notwendig

AVL-Bäume



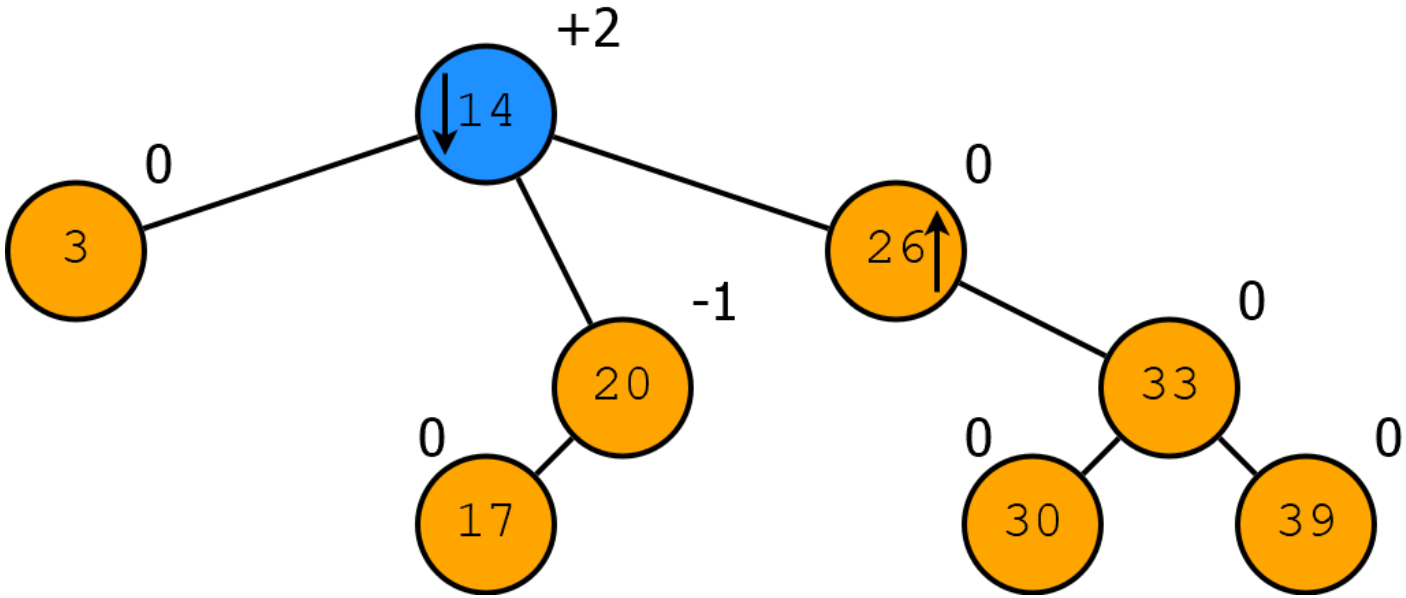
Rebalancing notwendig

AVL-Bäume



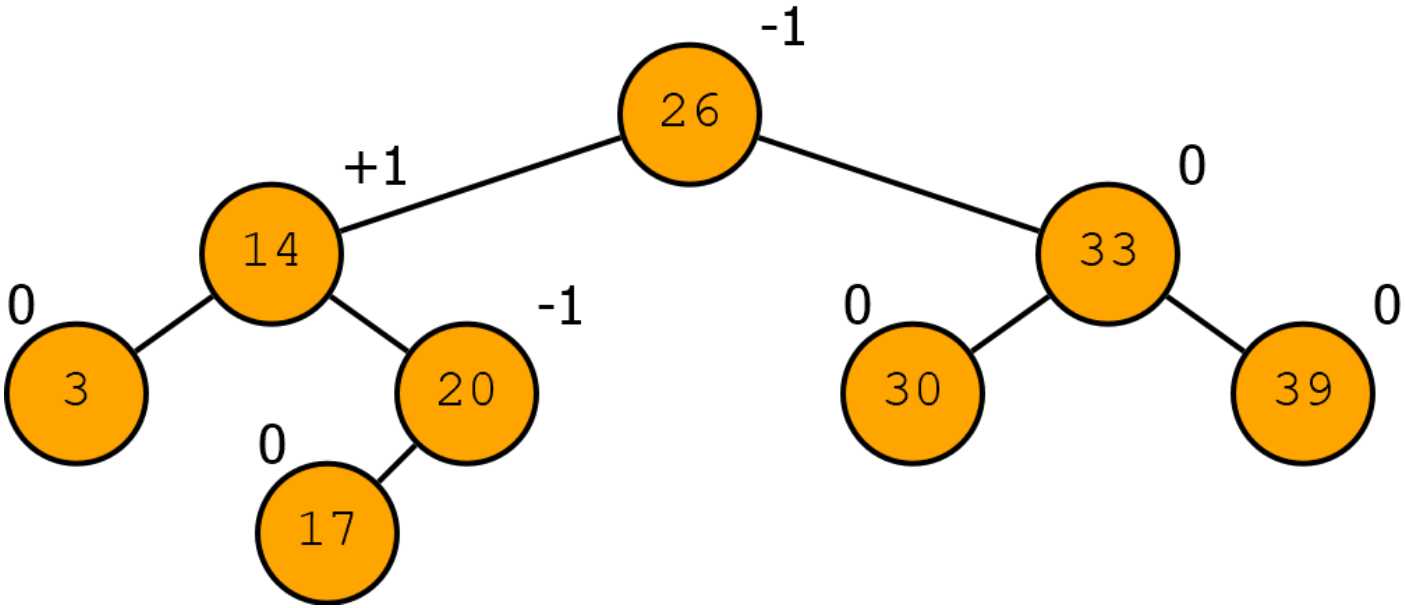
L-Rotation

AVL-Bäume



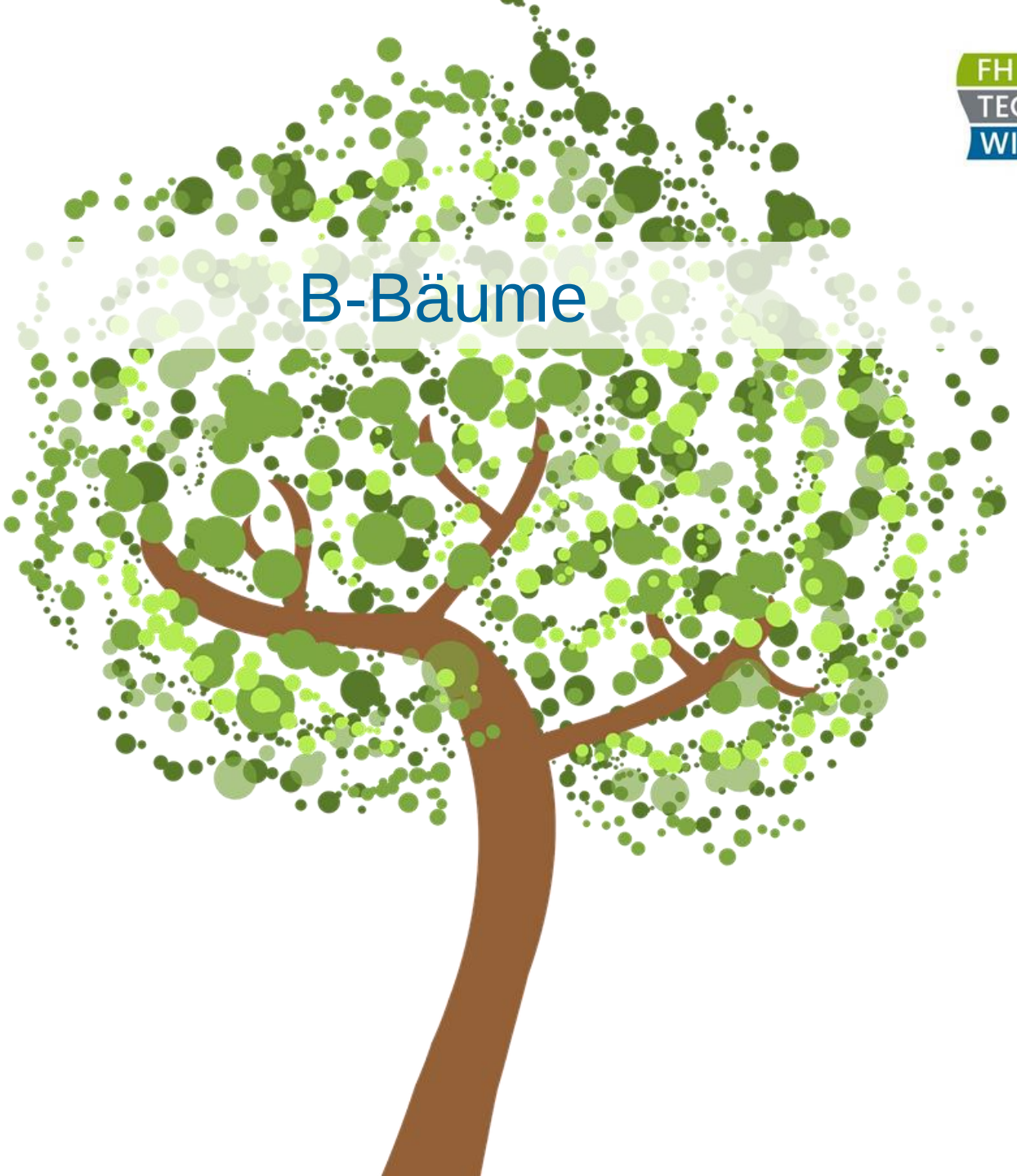
L-Rotation

AVL-Bäume



Rebalanced Tree

B-Bäume



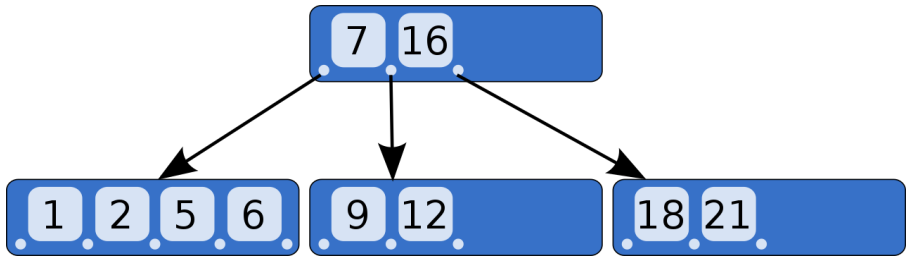
B-Bäume

Merkmale:

- B-Bäume – benannt nach Erfinder Rudolf Bayer
  (manchmal auch Mehrwegbaum)
- Vollständig ausgeglichene Höhe
- Variierender Verzweigungsgrad
  (kein Binär Bäum, pro Knoten sind mehrere Schlüssel und mehrere Nachfolger möglich)

Einsatzgebiet:

- Datenbanken: B-Baum als Indexdatenstruktur zur beschleunigten Suche der eigentlichen Datensätze
- Dateisysteme: Abbildung von Directory-Strukturen und schnelles Auffinden von Dateien (z.B. NTFS, EXT4, HFS+, ReiserFS)



B-Bäume

Definition für B-Baum der Ordnung m :

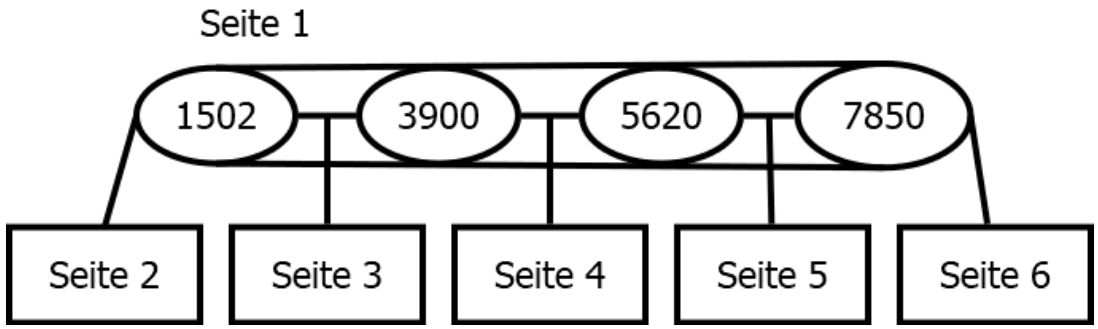
- Alle Blätter haben gleiche Tiefe und sind Leere Knoten (müssen in der Praxis nicht abgelegt werden)
- Jeder innere Knoten außer der Wurzel hat mindestens m und maximal $2m$ Kinder. Die Wurzel hat mindestens 2 Kinder. Damit sind alle inneren Knoten zu mindestens 50% gefüllt.
- Jeder Knoten mit l Schlüsseln hat $l + 1$ Kinder. Daraus folgt ein maximaler verzweigungsgrad von $2m + 1$
- Alle Schlüssel in Kindern liegen zwischen den Werten der beiden begrenzenden Schlüssel im Elternknoten.
- Höhe des Baumes maximal: $\log_m N$.
- Ein Koten wird auch Seite genannt und eignet sich um auf externe Medien ausgelagert zu werden.

B-Bäume

Beispiel:

- Kundendaten (Kundennummer als Schlüssel)
- Seite Enthält Kundendaten und Verweise

```
Wenn Kundennummer < 1502                  → Seite 2
Wenn Kundennummer > 1502 und < 3900       → Seite 3
Wenn Kundennummer > 3900 und < 5620       → Seite 4
```

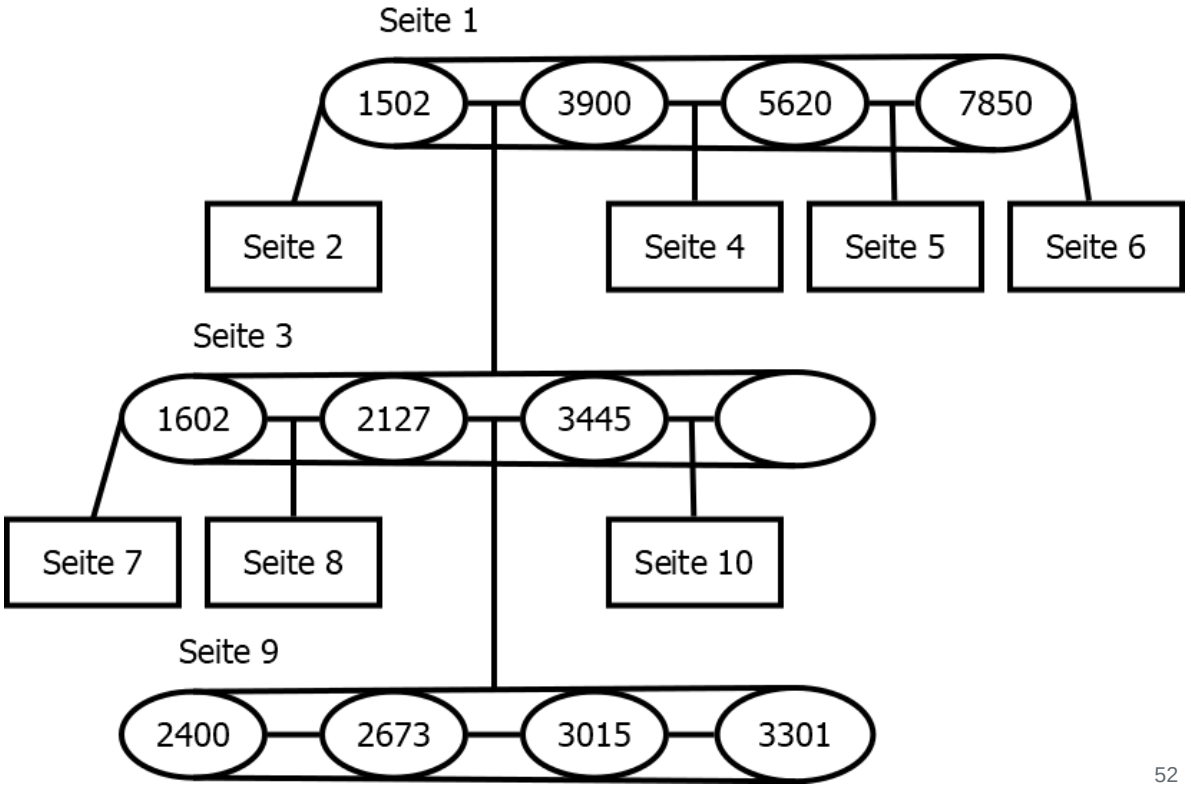


B-Bäume

Beispiel:

Suche Datensatz mit Schlüssel 3015

- `Lade Seite 1 (Wurzel)`
- `Lade Seite 3`
- `Lade Seite 9`



B-Bäume

Einfügen eines neuen Elements:

Anzahl der Elemente im Knoten muss zwischen m und $2m$ bleiben

- Suche von der Wurzel ausgehend die Seite, in die das neue Element eingefügt werden muss
- Ist genug Platz vorhanden (Anzahl der Elemente ist kleiner als $2m$), kann das Element einsortiert werden
- Besitzt die Seite bereits $2m$ Elemente, wird die Seite aufgespalten (siehe nächste Folie)

B-Bäume

Aufspalten einer Seite:

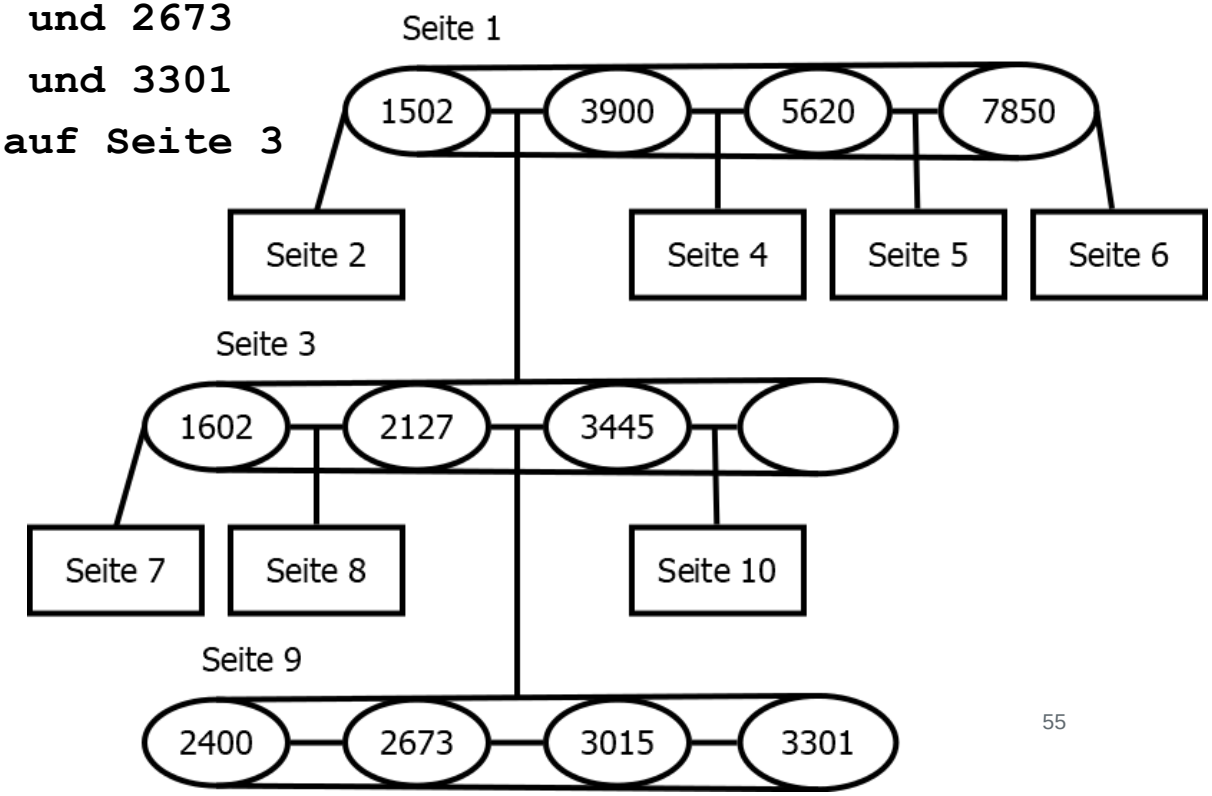
- Die ersten m Elemente (der insgesamt $2m + 1$ Elemente) bleiben in der Originalseite
- Die letzten m Elemente (der insgesamt $2m + 1$ Elemente) werden auf eine neu erstellte Seite verschoben
- Das mittlere Element wandert in den Vaterknoten nach oben
- Ist dort ebenfalls kein Platz muss rekursiv weiter aufgespalten werden
- Ist der Wurzelknoten erreicht und dort ebenfalls kein Platz, wird ein neuer Wurzelknoten erzeugt mit dem mittleren Element als Schlüssel und 2 Nachfolgern auf die 2 aufgespaltenen Seiten
  - Einzige Möglichkeit, wie der B-Baum in die Tiefe wachsen kann

B-Bäume

Beispiel ($m = 2$):

Füge Element 3100 ein

- **3100 gehört auf Seite 9 zwischen 3015 und 3301**
- **Seite 9 ist voll und muss aufgespalten werden**
- **Neue Seite mit 2400 und 2673**
- **Neue Seite mit 3100 und 3301**
- **Element 3015 kommt auf Seite 3**

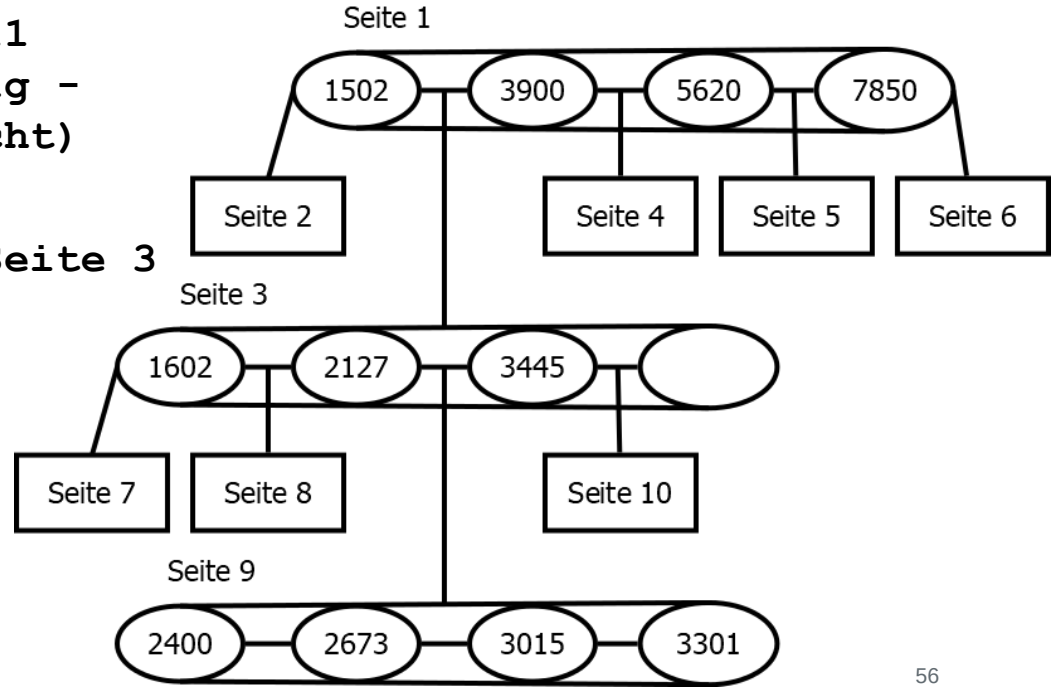


B-Bäume

Beispiel ($m = 2$):

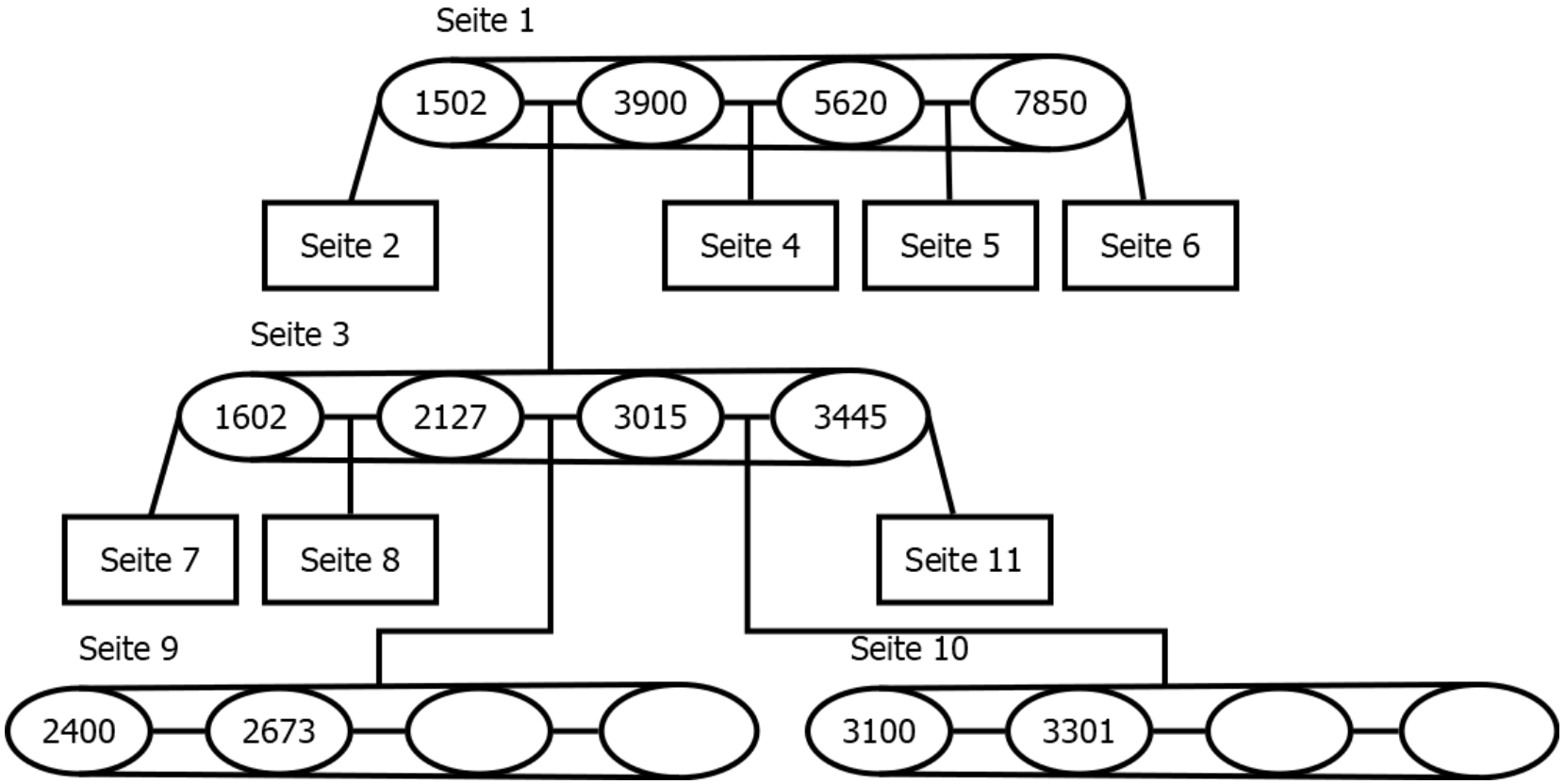
Füge Element 3100 ein

- **3100 gehört auf Seite 9 zwischen 3015 und 3301**
- **Seite 9 ist voll und muss aufgespalten werden**
- **Seite 9 mit 2400 und 2673**
- **Seite 10 wird zu Seite 11 (nicht zwingend notwendig - hier nur zwecks Übersicht)**
- **Seite 10 3100 und 3301**
- **Element 3015 kommt auf Seite 3**

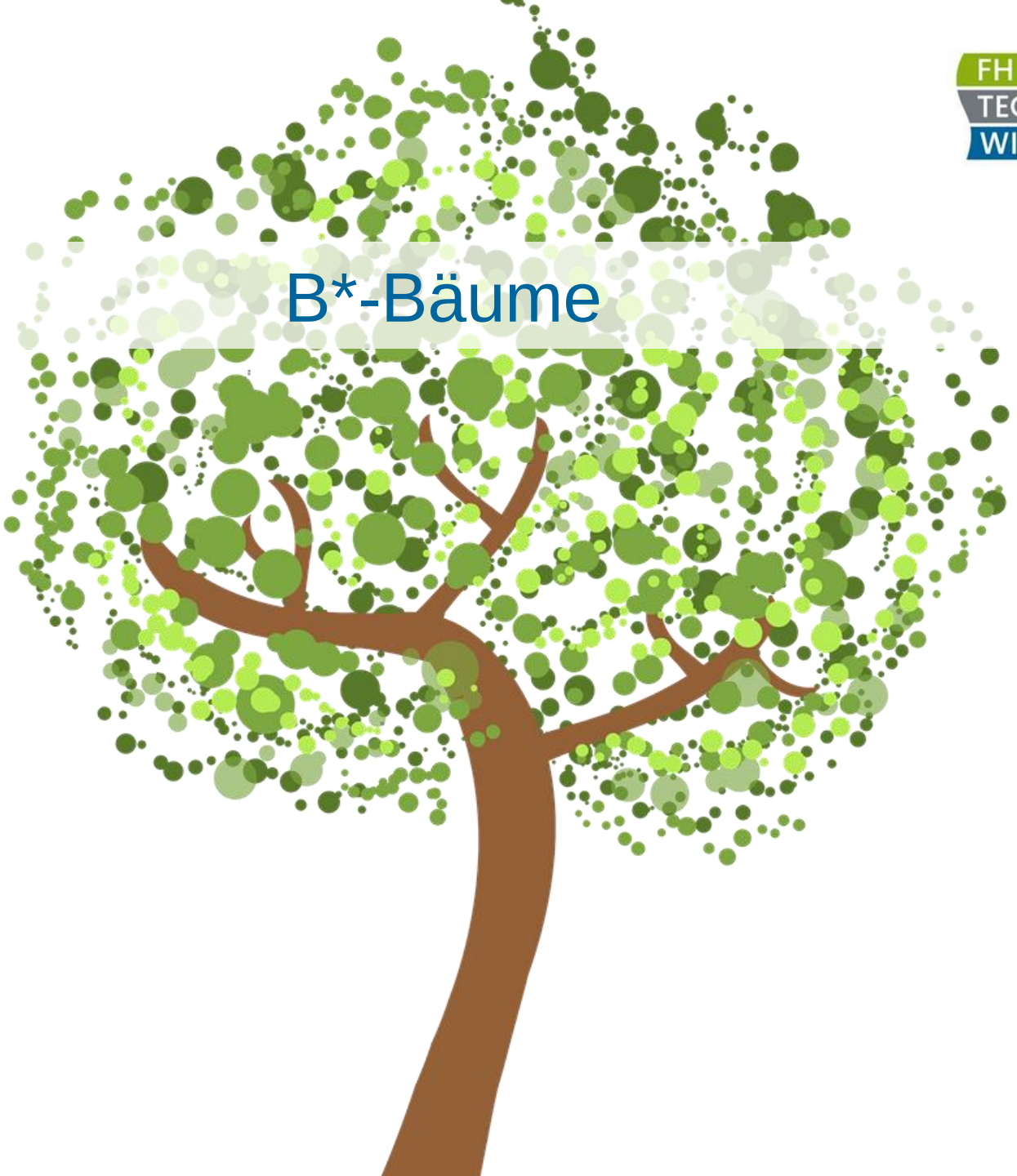


B-Bäume

Beispiel ($m = 2$): Fortsetzung



B*-Bäume

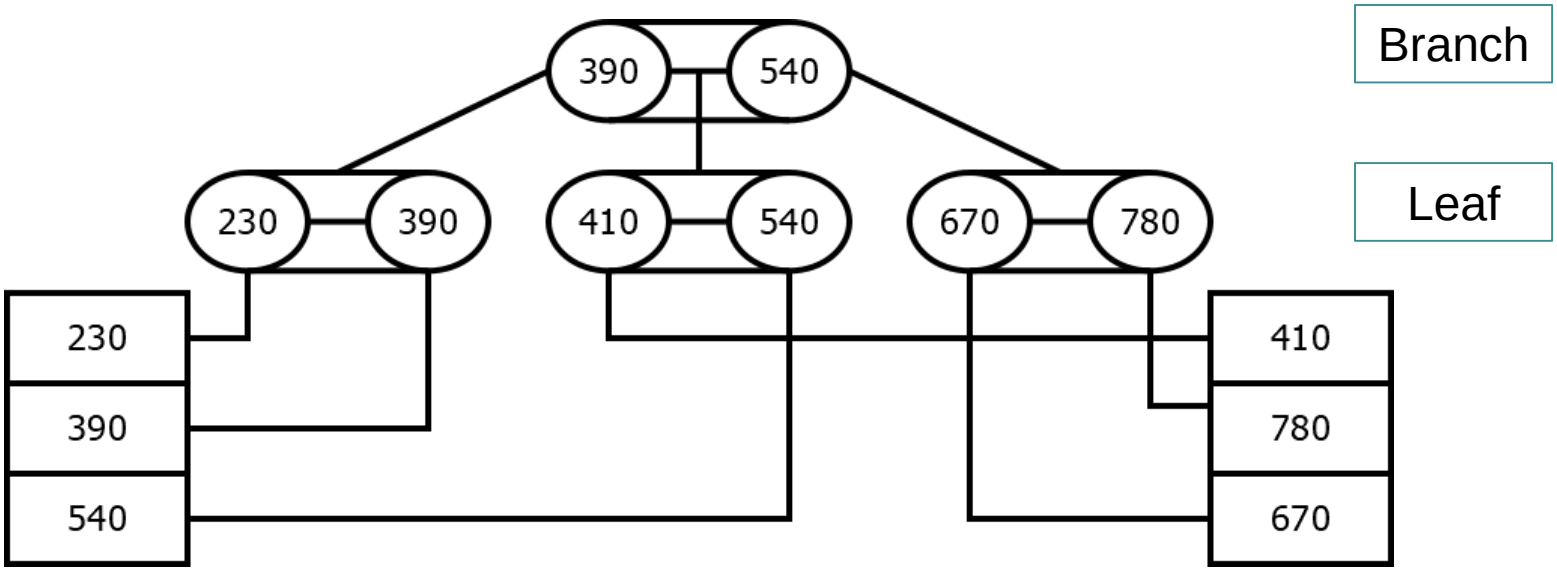


B*-Bäume

Merkmale: (im Bezug auf B-Baum)

- Datensätze stehen nur in den Blattknoten
- Keine trennenden Datensätze sondern nur trennende Schlüssel
- Zwischenknoten enthalten nur die Schlüssel zur Steuerung der Suche und die Verweise
  - Bei der Suche muss man daher immer bis zu einem Blattknoten suchen
- Die Blattknoten enthalten alle auftretenden Datensätze in geordneter Reihenfolge
  - Manchmal sind die Blattknoten miteinander verbunden, um die Datensätze linear durchwandern zu können
- in der Literatur oft auch B+Baum genannt

B*-Bäume



B-Bäume und B*-Bäume

Performance:

- Beispiel:
  - Seitengröße 4 KByte
  - Größe eines Elements: 32 Byte
  - Größe eines Verweises: 8 Byte
  - daher ca. 100 Einträge maximal pro Seite
  - $2m = 100$, $2m = 50$ also zwischen 50 und 100 Einträge pro Seite
  - Höhe des Baumes im schlimmsten Fall $\log_{50} N$
  - z.B. 1000000 Datensätze -> max. 4 Seitenzugriffe nötig!
- B*-Baum: noch höhere Verzweigung möglich, da Zwischenseiten nur Platz für Schlüssel und Verweise benötigen
  - Höhe des Baums und damit Seitenzugriffe sinken weiter

Bäume

Vielzahl an Varianten:

- **Binary Tree**
- **Binary Search Tree**
- **B -Tree**
- **B*-Tree**
- **Heaps**
- Red-Black Tree
- Dancing Tree
- und viele viele mehr!

ERINNERUNG: VisuAlgo.net

Fragen?

Feedback!