- *Insert*(k, T):

  Führe *IsElement*(k, T) aus;

  if k nicht in T **then**

  IsElement ergibt Blatt w (NIL-Pointer);

  Füge neuen internen Knoten v mit $k(v) = k$ an Stelle von w ein

  fi

- *Delete*(k, T):

  Führe *IsElement*(k, T) aus;

  if k in T enthalten **then**

  IsElement führt zu Knoten w mit $k = k(w)$

  Falls w keine internen Kinder hat: klar

  Falls w nur ein internes Kind hat: Ersetze w durch dieses

  Ansonsten ersetze w durch seinen In-Order-Vorgänger oder -Nachfolger

  fi

Problem bei natürlichen Suchbäumen:
Bei bestimmten Abfolgen der Wörterbuchoperationen entarten natürliche Suchbäume stark, z.B. bei Einfügen der Schlüssel in monoton aufsteigender bzw. absteigender Folge zu einer linearen Liste der Tiefe n.

Daraus ergibt sich für die Wörterbuch-Operationen *Insert*, *Delete*, *IsElement* eine *worst case*-Komplexität von

$$\Theta(n)\,.$$

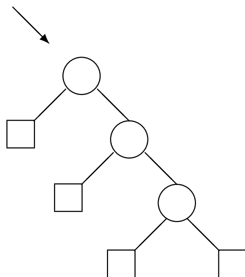
Falls alle Einfügefolgen gleichwahrscheinlich sind, gilt für die Höhe des Suchbaums

$$\mathbb{E}[h] = \mathcal{O}(\log n)\,.$$

Falls alle topologischen (von der Form her gleichaussehenden) Bäume der Größe n gleichwahrscheinlich sind, dann gilt

$$\mathbb{E}[h] = \Theta(\sqrt{n})\,.$$

Ohne Beweis!



3.2 Höhenbalancierte binäre Suchbäume (AVL-Bäume)

Definition 20

AVL-Bäume sind (interne) binäre Suchbäume, die die folgende Höhenbalancierung erfüllen:

Für jeden Knoten v gilt:

$$|\text{Höhe}(\text{linker UB}(v)) - \text{Höhe}(\text{rechter UB}(v))| \leq 1\,.$$

Bemerkung: AVL-Bäume sind nach ihren Erfindern G. Adelson-Velskii und Y. Landis (1962) benannt.

Satz 21

Ein AVL-Baum der Höhe h enthält mindestens $F_{h+2} - 1$ und höchstens $2^h - 1$ interne Knoten, wobei F_n die n-te Fibonacci-Zahl ($F_0 = 0, F_1 = 1$) und die Höhe die maximale Anzahl von Kanten auf einem Pfad von der Wurzel zu einem (leeren) Blatt ist.

Beweis:

Die obere Schranke ist klar, da ein Binärbaum der Höhe h höchstens

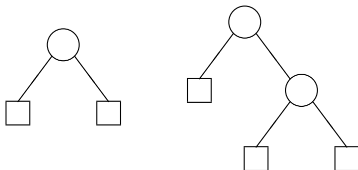
$$\sum_{j=0}^{h-1} 2^j = 2^h - 1$$

interne Knoten enthalten kann.

Beweis:

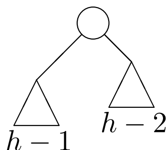
Induktionsanfang:

1. ein AVL-Baum der Höhe $h = 1$ enthält mindestens einen internen Knoten, $1 \geq F_3 - 1 = 2 - 1 = 1$
2. ein AVL-Baum der Höhe $h = 2$ enthält mindestens zwei Knoten, $2 \geq F_4 - 1 = 3 - 1 = 2$



Beweis:

Induktionsschluss: Ein AVL-Baum der Höhe $h \geq 2$ mit minimaler Knotenzahl hat als Unterbäume der Wurzel einen AVL-Baum der Höhe $h - 1$ und einen der Höhe $h - 2$, jeweils mit minimaler Knotenzahl.



Beweis:

Induktionsschluss: Ein AVL-Baum der Höhe $h \geq 2$ mit minimaler Knotenzahl hat als Unterbäume der Wurzel einen AVL-Baum der Höhe $h-1$ und einen der Höhe $h-2$, jeweils mit minimaler Knotenzahl. Sei

$$f_h := 1 + \text{minimale Knotenanzahl eines AVL-Baums der Höhe } h\,.$$

Dann gilt demgemäß

$$\begin{aligned} f_1 &= 2 && = F_3 \\ f_2 &= 3 && = F_4 \\ f_h - 1 &= 1 + f_{h-1} - 1 + f_{h-2} - 1\,, && \text{also} \\ f_h &= f_{h-1} + f_{h-2} && = F_{h+2} \end{aligned}$$

Bemerkung:

Da

$$F(n) \approx \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^n ,$$

hat ein AVL-Baum mit n internen Knoten eine Höhe

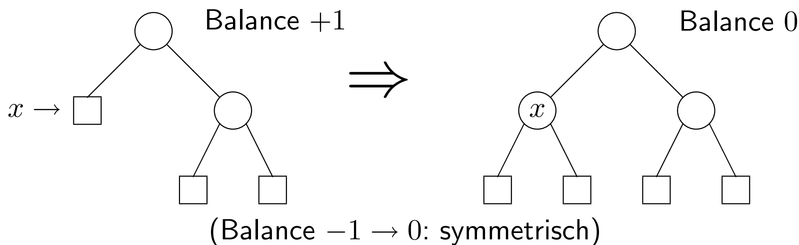
$$\Theta(\log n) .$$

Dies ist ein bedeutender Fortschritt gegenüber der bisweilen bei natürlichen Suchbäumen entstehenden, im *worst-case* linearen Entartung.

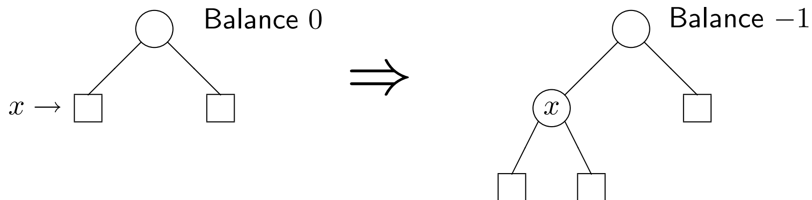
Die Operationen auf AVL-Bäumen:

1. *IsElement*: Diese Operation wird wie bei den natürlichen Suchbäumen implementiert. Wie oben ausgeführt, haben aber AVL-Bäume mit n Schlüsseln eine Höhe von $\mathcal{O}(\log n)$, woraus logarithmische Zeit für das Suchen folgt.
2. *Insert*: Zunächst wird eine *IsElement*-Operation ausgeführt, die für den Fall, dass das einzufügende Element nicht bereits enthalten ist, zu einem Blatt führt. Dies ist die Stelle, an der das neue Element einzufügen ist. Dabei ergeben sich 2 Fälle:

1. Fall:

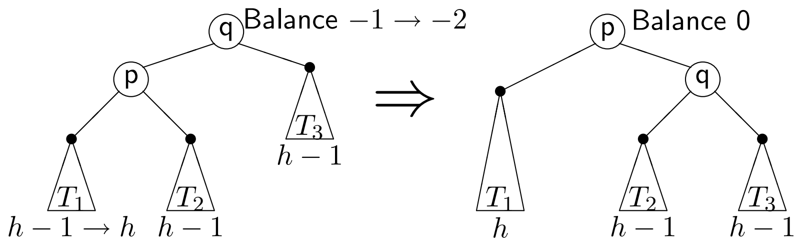


2. Fall:

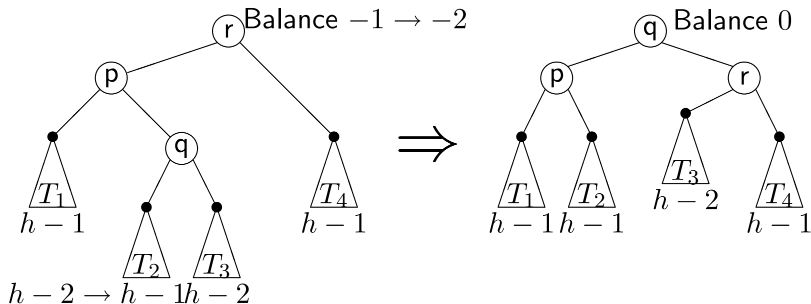


Hier ist eventuell eine Rebalancierung auf dem Pfad zur Wurzel notwendig, denn die Höhe des Unterbaums ändert sich.

Fall 2a:



Fall 2b:



3. *Delete*: Auch diese Operation wird wie bei natürlichen Suchbäumen implementiert. Jedoch hat am Ende ggf. noch eine Rebalancierung auf dem Pfad von dem Blatt, das an die Stelle des zu löschenden Elements geschrieben wird, zur Wurzel zu erfolgen.

Satz 22

Bei AVL-Bäumen sind die Operationen IsElement, Insert, und Delete so implementiert, dass sie die Zeitkomplexität $\mathcal{O}(\log n)$ haben, wobei n die Anzahl der Schlüssel ist.

Beweis:

Klar!

Im Grundsatz gelten folgende Bemerkungen für AVL-Bäume:

1. Sie haben in der Theorie sehr schöne Eigenschaften, auch zur Laufzeit.
2. Sie sind in der Praxis sehr aufwändig zu implementieren.

Weitere Informationen:

Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman:
The design and analysis of computer algorithms,
Addison-Wesley Publishing Company: Reading (MA), 1974

Robert Sedgewick, Philippe Flajolet:
An introduction to the analysis of algorithms,
Addison-Wesley Publishing Company, 1996

3.3 Gewichtsbalancierte Bäume

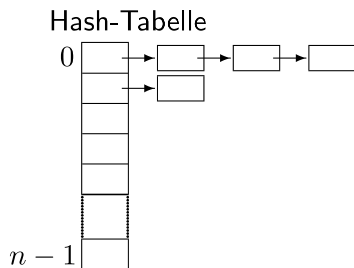
Siehe zu diesem Thema Seite 189ff in

Kurt Mehlhorn:
Data structures and algorithms 1: Sorting and searching,
EATCS Monographs on Theoretical Computer Science,
Springer Verlag: Berlin-Heidelberg-New York-Tokyo, 1984

4. Hashing

4.1 Grundlagen

- Universum U von Schlüsseln, z.B. $\subseteq \mathbb{N}_0$, $|U|$ groß
- Schlüsselmenge $S \subseteq U$, $|S| = m \leq n$
- array $T[0..n-1]$ Hashtabelle
- Hashfunktion $h : U \rightarrow [0..n-1]$



Problemstellung: Gegeben sei eine Menge M von Elementen, von denen jedes durch einen Schlüssel k aus der Menge U bezeichnet sei. Die Problemstellung ist: Wie muss die Speicherung der Elemente aus M bzw. der zugehörigen Schlüssel organisiert werden, damit jedes Element anhand seines Schlüssels möglich schnell lokalisiert werden kann?
Gesucht ist also eine Abbildung

$$h : K \to T$$

von der Menge aller Schlüssel in den Adressraum T der Maschine. Hierbei soll jedoch eine bisher nicht beachtete Schwierigkeit berücksichtigt werden: Die Menge U der möglichen Schlüsselwerte ist wesentlich größer als der Adressraum. Folglich kann die Abbildung h nicht injektiv sein, es gibt Schlüssel $k_1, k_2, \ldots$ mit $h(k_1) = h(k_2) = \ldots$.

Wir werden sehen, dass aufgrund dieses Umstandes die Speicherstelle eines Elements mit Schlüssel k von einem anderen Element mit einem anderen Schlüssel l besetzt sein kann und mit einer gewissen Wahrscheinlichkeit auch sein wird: Es treten Kollisionen auf.