Technische Universität München
Institut für Informatik I-16
Dr. Tobias Lasser
Matthias Wieczorek

SoSe 2015
Lösungsblatt 6
17. Juni 2014

Algorithmen und Datenstrukturen

Aufgabe 1 Binäre Suchbäume (Beispiellösung)

a) Ja, es handelt sich um einen binären Suchbaum (V,E), da für jeden inneren Knoten $v \in V$ gilt:

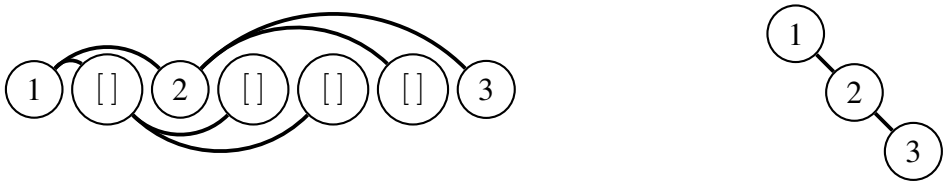
- für alle Knoten x im linken Teilbaum *v.left* gilt: $\text{key}(x) \leq \text{key}(v)$
- für alle Knoten y im rechten Teilbaum *v.right* gilt: $\text{key}(y) \geq \text{key}(v)$

Darüber hinaus gilt für jeden inneren Knoten $v \in V$, dass die Höhe des linken und des rechten Teilbaums gleich sind. Daher ist die AVL-Bedingung erfüllt und es handelt sich um einen AVL-Baum.

b) Bei dem Baum handelt es sich erneut um einen binären Suchbaum. Allerdings ist bei dem Knoten 4 die AVL-Bedingung verletzt (Höhe des linken Teilbaums ist 2, während die Höhe des rechten Teilbaums 0 beträgt). Daher stellt der Baum keinen AVL-Baum dar.

c) Es sich bei dem dargestellten Baum nicht um einen binären Suchbaum. Für den Knoten 8 ist das rechte Kind 7 kleiner. Da jeder AVL-Baum auch ein binärer Suchbaum sein muss, kann es sich somit auch nicht um einen AVL-Baum handeln.

d) Zuerst erzeugen wir aus der Liste den binären Baum entsprechend der Vorlesung:

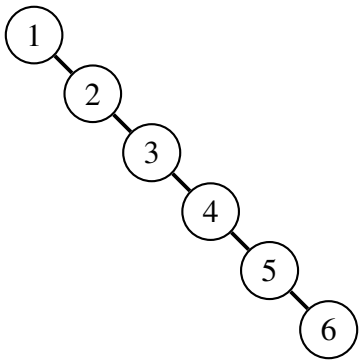


Wir stellen fest, dass es sich um einen binären Suchbaum handelt. Jedoch ist die AVL-Bedingung in der Wurzel verletzt, daher handelt es sich nicht um einen AVL-Baum.

e) Der Grund, warum es nicht sinnvoll ist, Suchbäume als Liste zu speichern ist der, dass diese nicht (zwangsläufig) vollständigen / fast vollständigen sind. Daher können die entsprechenden Listen großteils leer sein. Im Beispiel der vorherigen Teilaufgabe benötigen wir für den binären Suchbaum mit 3 Knoten eine Liste der Länge 7, wovon 4 Einträge leer sind.

Aufgabe 2 Binärer Suchbaum vs. AVL-Baum (Beispiellösung)

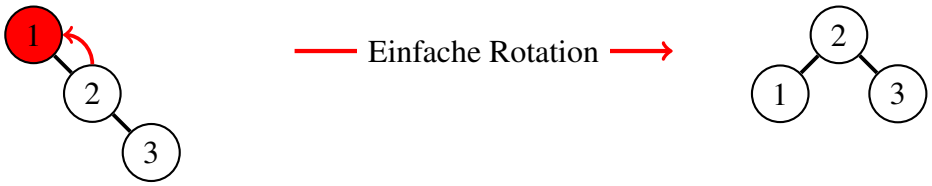
a) Fügen wir die Werte von 1 bis 6 nacheinander in einen binären Suchbaum ein, so erhalten wir einen entarteten Baum:



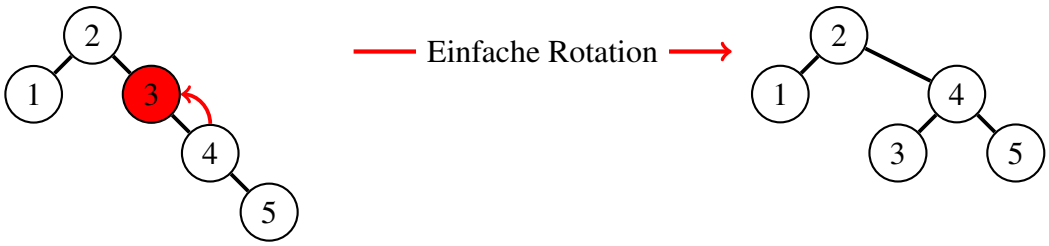
Um nun, ausgehend von der Wurzel, den Knoten 6 zu erreichen, benötigen wir 5 Schritte.

b) Der Einfachheit halber geben wir nur die Schritte an, bei welchen eine Rotation benötigt wird um die AVL-Bedingung wieder herzustellen.

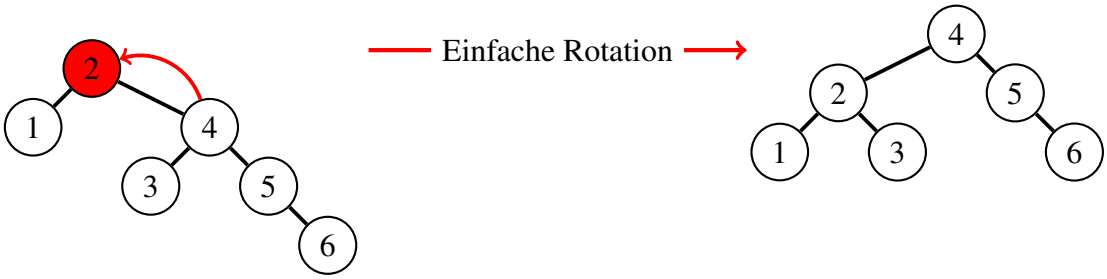
- ...
- **insert(3)**



- ...
- **insert(5)**



- ...
- **insert(6)**



Dieses Mal benötigen wir also 2 Schritte um zu dem Knoten 6.

Aufgabe 3 Hashing – Divisionsmethode (Beispiellösung)

a) Da der *modulo*-Operator Hashwerte im Interval $[0,\ldots,m-1]$ erzeugt, reicht es eine Hashtabelle der Grösse m zu wählen.

b) Für die Hashwerte erhalten wir:

- $key(4) = 4 \mod 4 = 0$
- $key(9) = 9 \mod 4 = 1$
- $key(14) = 14 \mod 4 = 2$
- $key(19) = 19 \mod 4 = 3$

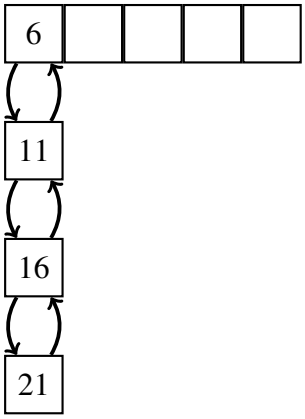
Damit ergibt sich folgende Hashtabelle:

4	9	14	19

c) Für die Hashwerte erhalten wir:

- $key(6) = 6 \mod 5 = 1$
- $key(11) = 11 \mod 5 = 1$
- $key(16) = 16 \mod 5 = 1$
- $key(21) = 21 \mod 5 = 1$

Es ergeben sich also Kollisionen bezüglich des Hashwertes 1, welche wir wie folgt mittels einer verketteten Liste auflösen. Damit ergibt sich folgende Hashtabelle:



Aufgabe 4 KMP-Präfix-Tabelle (Beispiellösung)

a)
```
 s t d : : s t r i n g
-1 0 0 0 0 0 1 2 0 0 0
```

b)
```
 i n t   s o r t ( i n t *   d a t a ,   i n t   n )
-1 0 0 0 0 0 0 0 0 0 1 2 3 0 0 0 0 0 0 0 0 1 2 3 4 0
```

c)
```
 T U   M ü n c h e n
-1 0 0 0 0 0 0 0 0 0
```