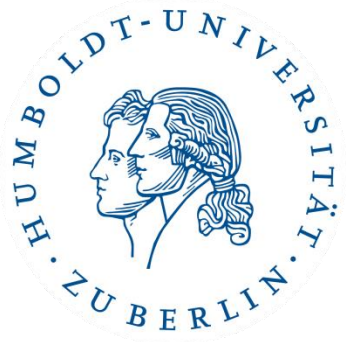
Übung Algorithmen und Datenstrukturen



Sommersemester 2016

Marc Bux, Humboldt-Universität zu Berlin

Agenda

- Hashing
- Binäre Suchbäume
- AVL-Bäume

Aufgabe zu Hashing

Sie sollen nun verschiedene Hashing Verfahren im Schreibtischtest anwenden. Es geht dabei nur darum, die Hashtabelle über eine Reihe von Einfügungen zu beobachten. Suchen oder Löschen wird nicht behandelt.
Verwenden Sie eine Hashtabelle mit 10 Plätzen (Index 0 bis 9) und fügen Sie in dieser Reihenfolge die folgenden Werte ein: 56, 34, 6, 13, 94, 27, 47. Geben Sie nach jedem Einfügeschritt die Hashtabelle an.

a) **Uniformes offenes Hashing.** Hier erhält jeder Schlüssel mit gleicher Wahrscheinlichkeit eine der 10! Permutationen von $\{0, 1, \ldots, 9\}$ als Sondierungsreihenfolge zugeordnet.
Als „zufällige“ Permutationen nutzen Sie:

$$h(56) = 2, 9, 3, 0, 8, 7, 6, 1, 4, 5 \qquad h(34) = 1, 5, 9, 3, 7, 4, 8, 6, 2, 0$$
$$h(6) = 0, 8, 5, 2, 1, 7, 9, 6, 3, 4 \qquad h(13) = 1, 4, 5, 2, 0, 7, 3, 6, 9, 8$$
$$h(94) = 6, 8, 4, 0, 9, 1, 5, 2, 3, 7 \qquad h(27) = 4, 2, 3, 5, 1, 8, 9, 0, 7, 6$$
$$h(47) = 1, 3, 6, 0, 4, 5, 2, 8, 7, 9$$

b) **Offenes Hashing mit linearem Sondieren.** Hashfunktion $h(k) = k \mod 10$. Bei Kollisionen wird hier das einzufügende Element an der nächsten freien Stelle links vom berechneten Hashwert eingefügt. Das heißt, die Sondierungsreihenfolge (die Reihenfolge, in der die Plätze im Array durchgegangen werden, bis erstmals ein freier Platz angetroffen wird) ist gegeben durch $s(k, j) = (h(k) - j) \mod 10$).

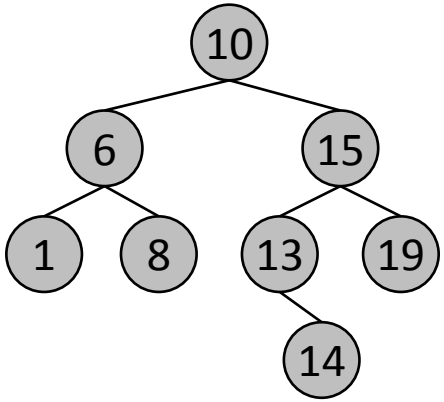
c) **Double Hashing.** Das doppelte Hashing ist ein offenes Hashing, bei dem die Sondierungsreihenfolge von einer zweiten Hashfunktion abhängt. Hashfunktion $h(k) = k \mod 10$, zweite Hashfunktion $h'(k) = 1 + (k \mod 8)$. Sondieren ist bestimmt durch die Funktion $s(k, j) = (h(k) - j \cdot h'(k)) \mod 10$.

Agenda

- Hashing
- Binäre Suchbäume
- AVL-Bäume

Binäre Suchbäume

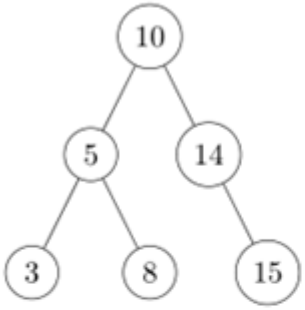
- Knoten
  - beinhaltet „Schlüssel“
  - hat Verweis auf linkes und rechtes Kind
- Sortierung/Sucheigenschaft
  - Schlüssel des linken Teilbaums eines Knotens sind kleiner als Schlüssel des Knotens selbst
  - Schlüssel des rechten Teilbaums eines Knotens sind größer als Schlüssel des Knotens selbst
  - Gleiche Schlüssel entweder links oder rechts



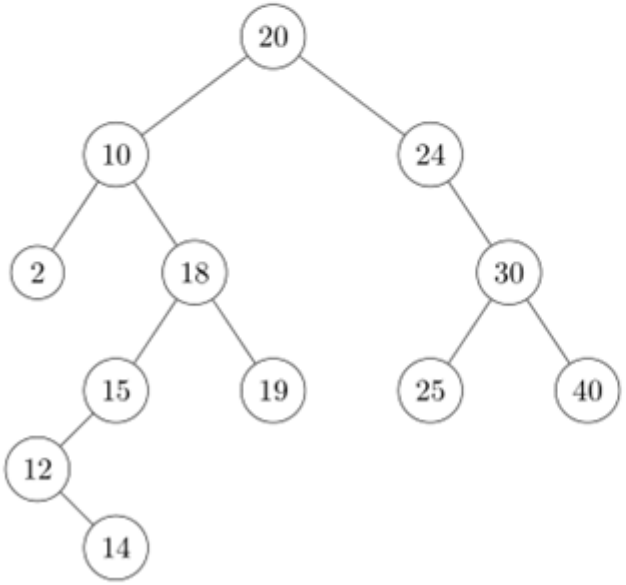
Aufgabe zu Binären Suchbäumen

In dieser Aufgabe sollen Sie einen Schreibtischtest mit binären Suchbäumen ausführen.

a) Führen Sie nun die Operationen `insert(16)`, `insert(11)` und `delete(8)` in dieser Reihenfolge aus. Geben Sie nach jeder Operation den neuen Baum an.

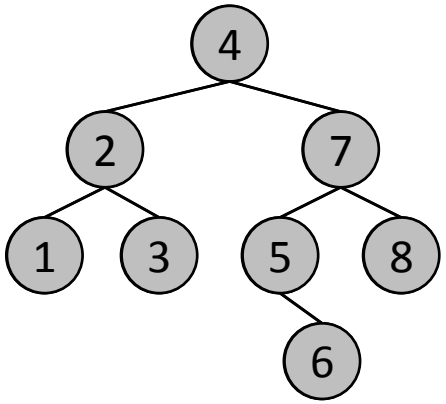


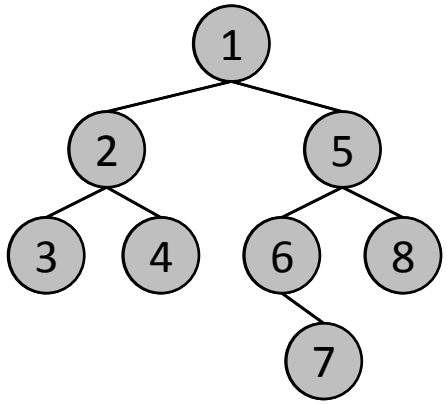
b) Führen Sie in dem folgenden Suchbaum die Operationen `delete(24)` und `delete(10)` in dieser Reihenfolge aus. Geben Sie nach jeder Operation den neuen Baum an.

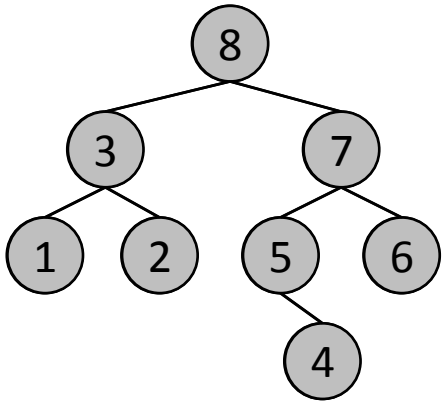


Exkurs: Traversierung binärer Bäume

- In-Order-Traversierung: Besuche linken Teilbaum, dann aktuellen Knoten, dann rechten Teilbaum
- Pre-Order-Traversierung: Besuche aktuellen Knoten, dann linken Teilbaum, dann rechten Teilbaum
- Post-Order-Traversierung: Besuche linken Teilbaum, dann rechten Teilbaum, dann aktuellen Knoten





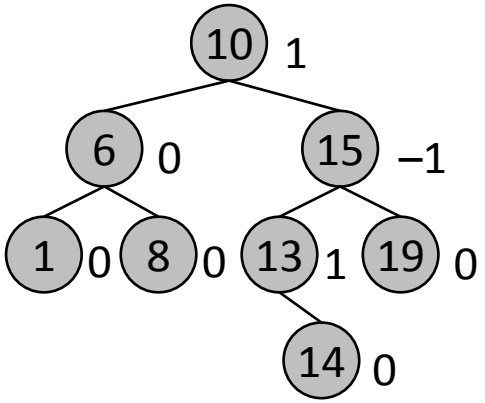


Agenda

- Hashing
- Binäre Suchbäume
- AVL-Bäume

AVL-Bäume

- Benannt nach Erfindern Adelson-Velsky und Landis (AVL)
- Älteste Datenstruktur für balancierte Bäume
- Binärer Suchbaum mit zusätzlicher Eigenschaft
  - In jedem Knoten unterscheidet sich die Höhe der beiden Teilbäume um höchstens eins
  - Höhe logarithmisch in Anzahl der Schlüssel
- Rebalancierung (Rotation) beim Einfügen und Löschen
- Definition:
  - Sei *u* Knoten in binärem Baum.
  - bal(*u*) = Differenz zwischen Höhe des rechten Teilbaums von *u* und Höhe des linkes Teilbaums von *u*
  - Ein binärer Baum heißt AVL-Baum, falls für alle Knoten *u* gilt: $|\text{bal}(u)| \leq 1$

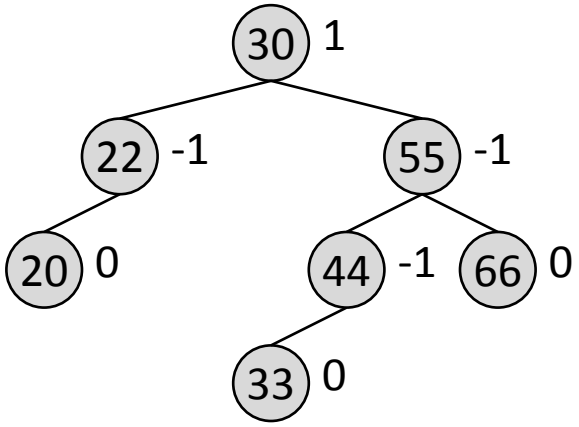


Rebalancierung von AVL-Bäumen

- 4 Rotationsoperationen auf AVL-Bäumen:
  - Sei *u* Knoten, *v* Kind von *u* im Teilbaum mit größerer Höhe
  - bal(*u*) = 2, bal(*v*) = 1: Einfachrotation Links(*u*)
  - bal(*u*) = 2, bal(*v*) = -1: Doppelrotation Rechts(*v*) + Links(*u*)
  - bal(*u*) = -2, bal(*v*) = 1: Doppelrotation Links(*v*) + Rechts(*u*)
  - bal(*u*) = -2, bal(*v*) = -1: Einfachrotation Rechts(*u*)
- Laufzeit: Rotationen sind lokale Operationen, die nur Umsetzen einiger Zeiger erfordern, und in Zeit $\mathcal{O}(1)$ erfolgen

Aufgabe zu AVL-Bäumen

Sei *T* folgender AVL-Baum:



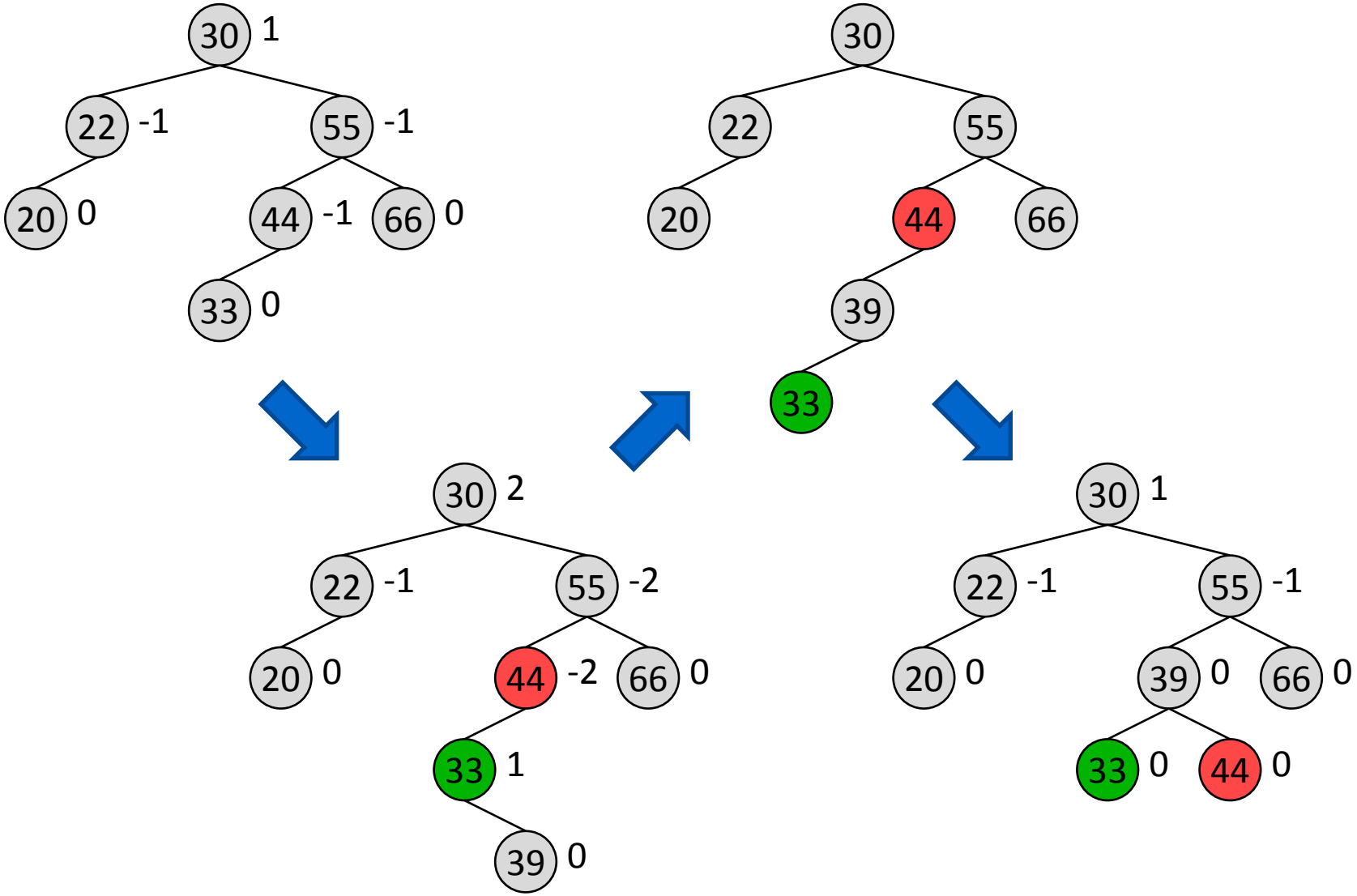
Fügen Sie nacheinander die Schlüssel

39, 42

in *T* ein und zeichnen Sie den jeweiligen AVL-Baum nach jeder insert-Operation.

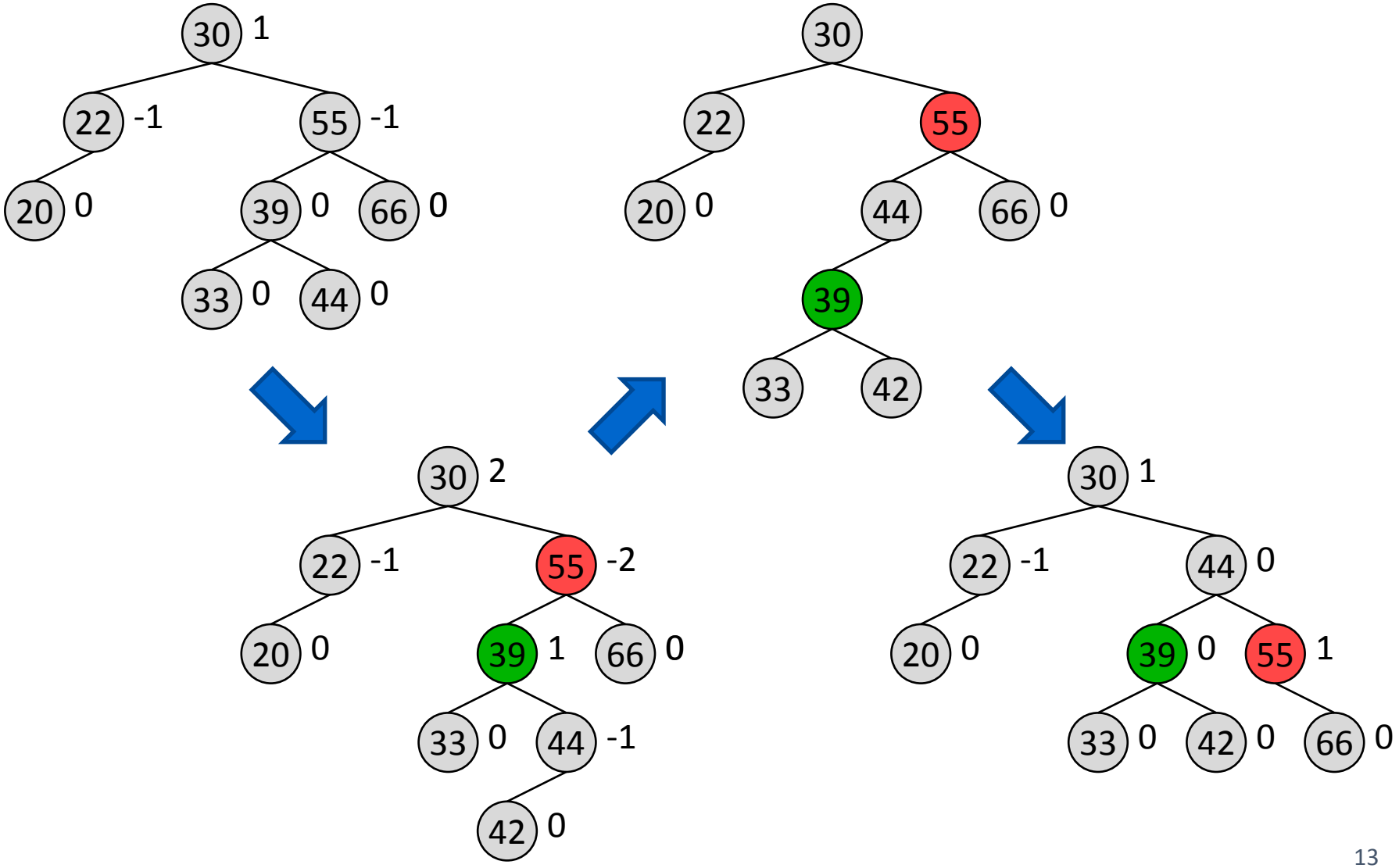
Einfügen von 39

bal(u) = -2, bal(v) = 1: Doppelrotation Links(v) + Rechts(u)



Einfügen von 42

bal(u) = -2, bal(v) = 1: Doppelrotation Links(v) + Rechts(u)



Ausblick: Nächste drei Wochen

- Besprechung der Lösungen des fünften Übungsblatts
- Besprechung der Lösungen der sechsten Übungsblatts
- Probeklausur
- Fragen?