TECHNISCHE UNIVERSITÄT DRESDEN

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 9: AVL-BÄUME & TOPOLOGISCHES SORTIEREN

Eric Kunze
eric.kunze@tu-dresden.de

TU Dresden, 6. Dezember 2021

letzte Änderung:
04.12.2021, 22:13

AVL-Bäume

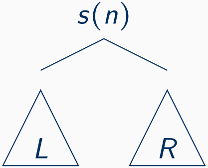
AVL-BÄUME

Wir betrachten einen Baum t und bezeichnen die *Schlüssel* an den Knoten n mit $s(n)$.

AVL-BÄUME

Wir betrachten einen Baum t und bezeichnen die *Schlüssel* an den Knoten n mit $s(n)$.

Suchbaum:

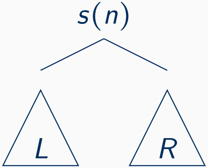


mit $L < s(n) < R$

AVL-BÄUME

Wir betrachten einen Baum t und bezeichnen die *Schlüssel* an den Knoten n mit $s(n)$.

Suchbaum:



mit $L < s(n) < R$

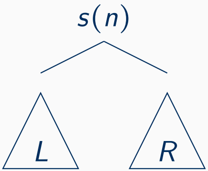
Die **Höhe** des Baumes bezeichnen wir mit $h(t)$. Wir ordnen jedem Knoten n einen **Balancefaktor** $b(n)$ zu:

$$b(n) := h(R) - h(L)$$

AVL-BÄUME

Wir betrachten einen Baum t und bezeichnen die *Schlüssel* an den Knoten n mit $s(n)$.

Suchbaum:



mit $L < s(n) < R$

Die **Höhe** des Baumes bezeichnen wir mit $h(t)$. Wir ordnen jedem Knoten n einen **Balancefaktor** $b(n)$ zu:

$$b(n) := h(R) - h(L)$$

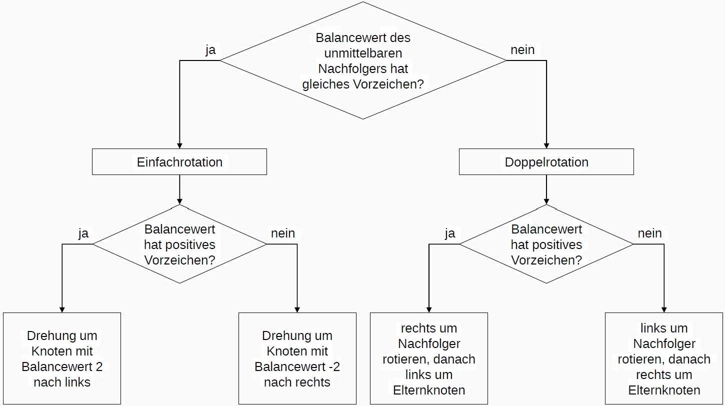
AVL-Baum: Suchbaum mit $b(n) \in \{-1, 0, 1\}$

BALANCIEREN

- Einfügen eines neuen Schlüssels s
- Berechne Balancefaktoren auf dem Pfad von s zur Wurzel bis zum ersten Auftreten von ± 2

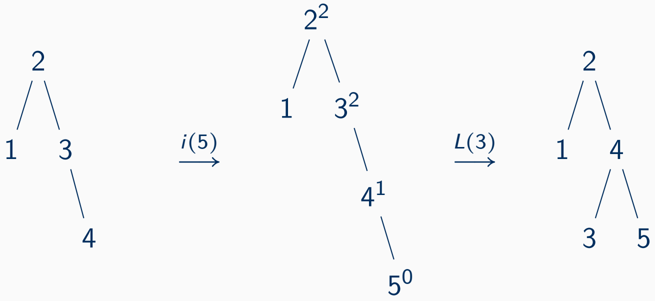
BALANCIEREN

- Einfügen eines neuen Schlüssels s
- Berechne Balancefaktoren auf dem Pfad von s zur Wurzel bis zum ersten Auftreten von ± 2
- **Balancierungsalgorithmus**:

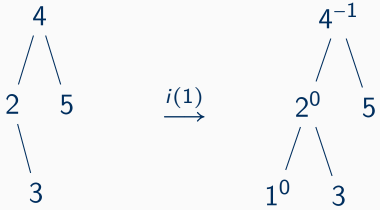


AUFGABE 1

Baum 1:

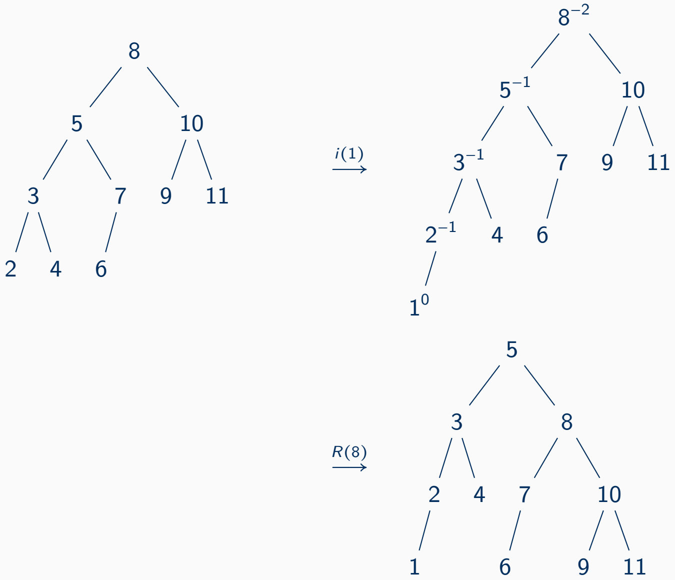


Baum 2:

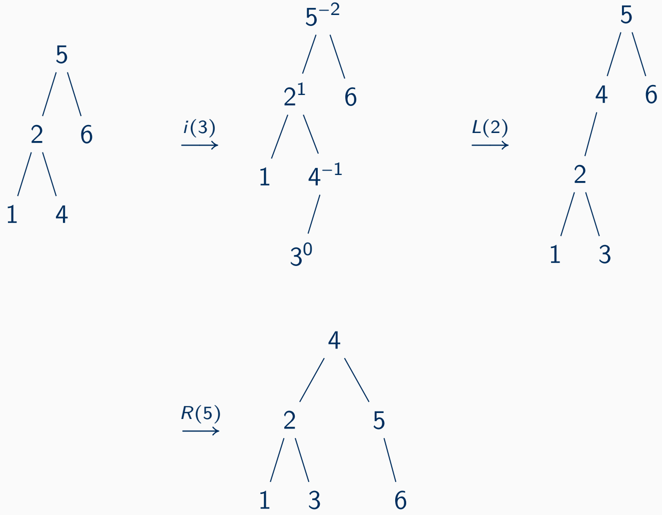


AUFGABE 1

Baum 3:

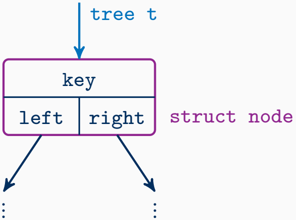


Baum 4:



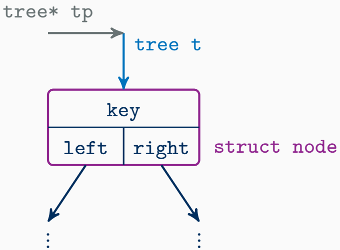
ERINNERUNG: BÄUME

```c
typedef struct node *tree;
struct node { int key; tree left, right; };
```



ERINNERUNG: BÄUME

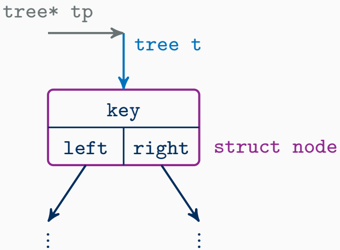
```
typedef struct node *tree;
struct node { int key; tree left, right; };
```

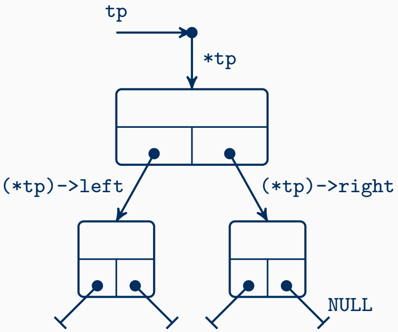


- Verarbeitung eines Baumes: Einfachreferenzen (`tree t`)
- Veränderung eines Baumes:

ERINNERUNG: BÄUME

```
typedef struct node *tree;
struct node { int key; tree left, right; };
```





- Verarbeitung eines Baumes: Einfachreferenzen (`tree t`)
- Veränderung eines Baumes:

Topologisches Sortieren

TOPOLOGISCHES SORTIEREN

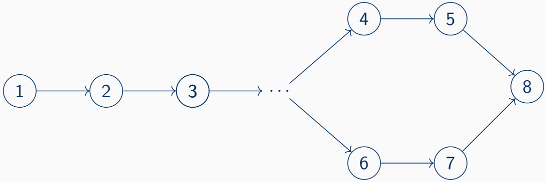
- Sortierung von *Beziehungen* zwischen Objekten
- **Bsp.:** Ablauf eines Bauvorhabens

TOPOLOGISCHES SORTIEREN

- Sortierung von *Beziehungen* zwischen Objekten
- **Bsp.:** Ablauf eines Bauvorhabens
  - Baugrube ausheben (1) vor Fundament gießen (2)
  - Fundament gießen (2) vor Wände setzen (3)
  - ...
  - Elektrik im Bad (4) vor Fliesen (5)
  - Wohnzimmer tapezieren (6) vor streichen (7)
  - Wände streichen (5) und Fliesen (7) vor Möbel aufstellen (8)

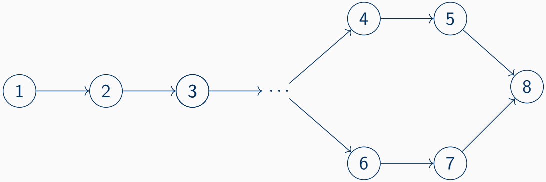
TOPOLOGISCHES SORTIEREN

- Sortierung von *Beziehungen* zwischen Objekten
- **Bsp.:** Ablauf eines Bauvorhabens
  - Baugrube ausheben (1) vor Fundament gießen (2)
  - Fundament gießen (2) vor Wände setzen (3)
  - ...
  - Elektrik im Bad (4) vor Fliesen (5)
  - Wohnzimmer tapezieren (6) vor streichen (7)
  - Wände streichen (5) und Fliesen (7) vor Möbel aufstellen (8)



TOPOLOGISCHES SORTIEREN

- Sortierung von *Beziehungen* zwischen Objekten
- **Bsp.:** Ablauf eines Bauvorhabens
  - Baugrube ausheben (1) vor Fundament gießen (2)
  - Fundament gießen (2) vor Wände setzen (3)
  - ...
  - Elektrik im Bad (4) vor Fliesen (5)
  - Wohnzimmer tapezieren (6) vor streichen (7)
  - Wände streichen (5) und Fliesen (7) vor Möbel aufstellen (8)



- In welcher Reihenfolge kann ich die Tätigkeiten abarbeiten?

TOPOLOGISCHES SORTIEREN

Gegeben sei ein gerichteter, azyklischer Graph $G = (V, E)$. Eine **topologische Sortierung** von G ist eine *bijektive* Abbildung $\mathrm{ord}\colon V \to \{1, \ldots, |V|\}$, sodass für alle $v, v' \in V$ mit $(v, v') \in E$ die Relation $\mathrm{ord}(v) < \mathrm{ord}(v')$ gilt.

TOPOLOGISCHES SORTIEREN

Gegeben sei ein gerichteter, azyklischer Graph $G = (V, E)$. Eine **topologische Sortierung** von G ist eine *bijektive* Abbildung $\mathrm{ord}\colon V \to \{1, \ldots, |V|\}$, sodass für alle $v, v' \in V$ mit $(v, v') \in E$ die Relation $\mathrm{ord}(v) < \mathrm{ord}(v')$ gilt.

Algorithmus:

`while` (Elemente übrig)

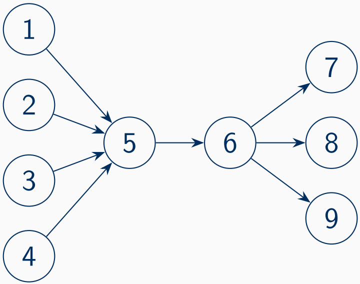
$\{$

- wähle Element v ohne Vorgänger
- dekrementiere Anzahl der Vorgänger in den Nachfolgern von v
- füge v der Ausgabeliste hinzu
- lösche v aus G

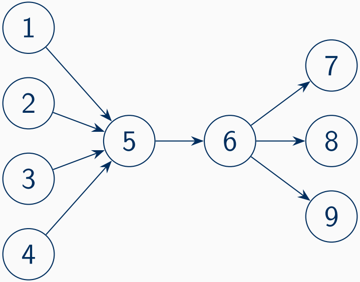
$\}$

AUFGABE 3

Teil (a)



Teil (a)



Es gibt $4! \cdot 1! \cdot 3! = 24 \cdot 1 \cdot 6 = 144$ viele topologische Sortierungen.

AUFGABE 3

Teil (b)

Es werden alle Möglichkeiten mit der 1 am Anfang gestrichen, d.h. im ersten Block gibt es nur noch $4! - 3! = 24 - 6$ viele Möglichkeiten — insgesamt also

$$\left(4! - 3!\right) \cdot 1! \cdot 3! = 18 \cdot 1 \cdot 6 = 108\,.$$

AUFGABE 3

Teil (b)

Es werden alle Möglichkeiten mit der 1 am Anfang gestrichen, d.h. im ersten Block gibt es nur noch $4! - 3! = 24 - 6$ viele Möglichkeiten — insgesamt also

$$(4! - 3!) \cdot 1! \cdot 3! = 18 \cdot 1 \cdot 6 = 108\,.$$

Teil (c)