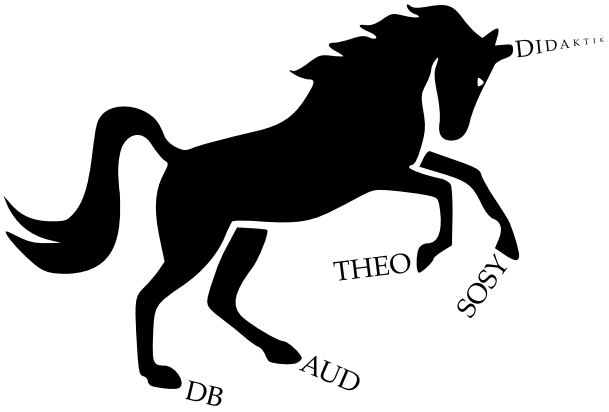
Die komplette Sammlung

Alle Aufgaben



Die Bschlangaul-Sammlung
Hermine Bschlangauland Friends

Inhaltsverzeichnis

46115 (Theoretische Informatik / Algorithmen / Datenstrukturen (nicht vertieft))

Halde (Heap)
Binärbaum
AVL-Baum
Min-Heap

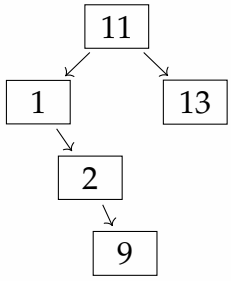
46115 / 2014 / Frühjahr / Thema 1 / Aufgabe 7

Fügen Sie nacheinander die Zahlen 11, 1, 2, 13, 9, 10, 7, 5

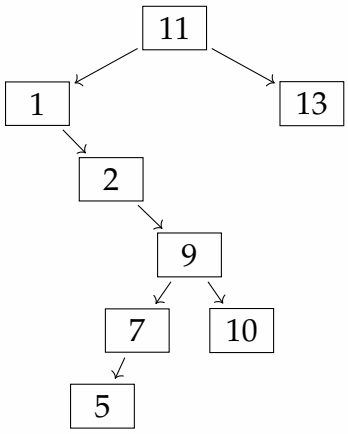
(a) in einen leeren binären Suchbaum und zeichnen Sie den Suchbaum jeweils nach dem Einfügen von „9“ und „5“

Lösungsvorschlag

Nach dem Einfügen von „9“:



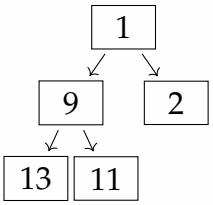
Nach dem Einfügen von „5“:



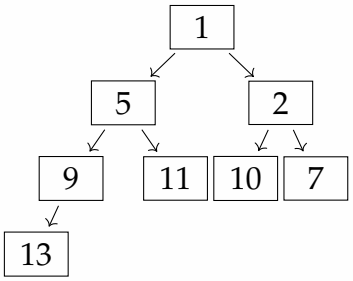
(b) in einen leeren Min-Heap ein, der bzgl. „$\leq$“ angeordnet ist und geben Sie den Heap nach „9“ und nach „5“ an

Lösungsvorschlag

Nach dem Einfügen von „9“:



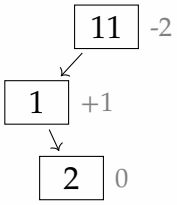
Nach dem Einfügen von „5“:



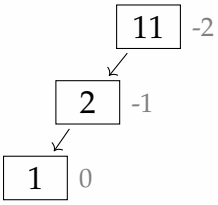
(c) in einen leeren AVL-Baum ein! Geben Sie den AVL Baum nach „2“ und „5“ an und beschreiben Sie die ggf. notwendigen Rotationen beim Einfügen dieser beiden Elemente!

Lösungsvorschlag

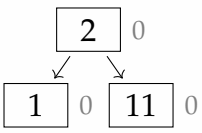
Nach dem Einfügen von „2“:



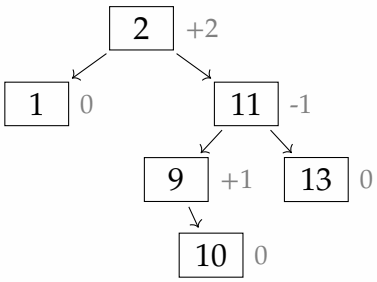
Nach der Linksrotation:



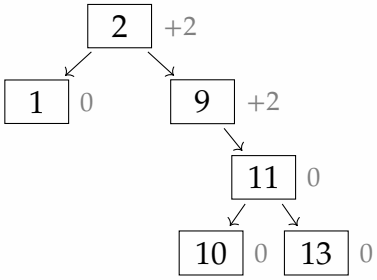
Nach der Rechtsrotation:



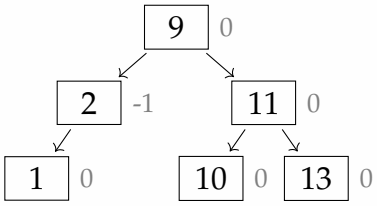
Nach dem Einfügen von „10“:



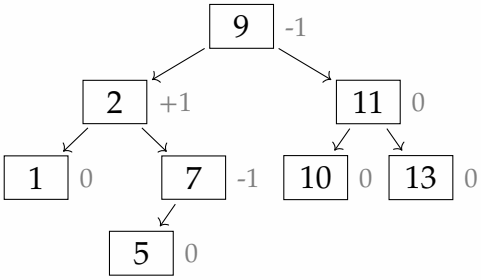
Nach der Rechtsrotation:



Nach der Linksrotation:



Nach dem Einfügen von „5“:



46115 / 2015 / Herbst / Thema 2 / Aufgabe 4

Gegeben sei die folgende Methode `function`:

```
double function(int n) {
  if (n == 1)
```

```
    return 0.5 * n;
  else
    return 1.0 / (n * (n + 1)) + function(n - 1);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/Induktion.java`

Beweisen Sie folgenden Zusammenhang mittels vollständiger Induktion:

$$\forall n \geq 1\colon \text{function}(n) = f(n) \text{ mit } f(n) := 1 - \frac{1}{n+1}$$

Hinweis: Eventuelle Rechenungenauigkeiten, wie z. B. in Java, bei der Behandlung von Fließkommazahlen (z. B. `double`) sollen beim Beweis nicht berücksichtigt werden - Sie dürfen also annehmen, Fließkommazahlen würden mathematische Genauigkeit aufweisen.

Lösungsvorschlag

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. ———

$f(1) := 1 - \frac{1}{1+1} = 1 - \frac{1}{2} = \frac{1}{2}$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———

$f(n) := 1 - \frac{1}{n+1}$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k+1)$ wahr sein muss. ——

zu zeigen:

$f(n+1) := 1 - \frac{1}{(n+1)+1} = f(n)$

Vorarbeiten (Java in Mathe umwandeln):

$\text{function}(n) = \frac{1}{n \cdot (n+1)} + f(n-1)$

$$
\begin{aligned}
f(n+1) &= \frac{1}{(n+1)\cdot((n+1)+1)} + f((n+1)-1) && n+1 \text{ eingesetzt}\\
&= \frac{1}{(n+1)\cdot(n+2)} + f(\mathrm{n}) && \text{vereinfacht}\\
&= \frac{1}{(n+1)\cdot(n+2)} + 1 - \frac{1}{n+1} && \text{für } f(n) \text{ Formel eingesetzt}\\
&= 1 + \frac{1}{(n+1)\cdot(n+2)} - \frac{1}{n+1} && \text{1. Bruch an 2. Stelle geschrieben}\\
&= 1 + \frac{1}{(n+1)\cdot(n+2)} - \frac{1\cdot(n+2)}{(n+1)\cdot(n+2)} && \text{2. Bruch mit } (n+2) \text{ erweitert}\\
&= 1 + \frac{1-(n+2)}{(n+1)\cdot(n+2)} && \text{die 2 Brüche subtrahiert}\\
&= 1 + \frac{1-n-2}{(n+1)\cdot(n+2)} && -+2=-2\\
&= 1 + \frac{-1-n}{(n+1)\cdot(n+2)} && 1-2=-1\\
&= 1 + \frac{-1\cdot(1+n)}{(n+1)\cdot(n+2)} && (n+1) \text{ ausgeklammert}\\
&= 1 + \left(-1\cdot\frac{(1+n)}{(n+1)\cdot(n+2)}\right) && \text{minus vor den Bruch bringen}\\
&= 1 - \frac{(1+n)}{(n+1)\cdot(n+2)} && \text{plus minus ist minus}\\
&= 1 - \frac{1}{n+2} && (n+1) \text{ gekürzt}\\
&= 1 - \frac{1}{(n+1)+1} && \text{Umformen zur Verdeutlichung}
\end{aligned}
$$

46116 (Softwaretechnologie / Datenbanksysteme (nicht vertieft))

46116 / 2014 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1

Gegeben sei folgende Methode zur Berechnung der Anzahl der notwendigen Züge beim Spiel *„Die Türme von Hanoi“*:

```
int hanoi(int nr, char from, char to) {
  char free = (char) ('A' + 'B' + 'C' - from - to);
  if (nr > 0) {
    int moves = 1;
    moves += hanoi(nr - 1, from, free);
    System.out.println("Move piece nr. " + nr + " from " + from + " to " + to);
```

```
    moves += hanoi(nr - 1, free, to);
    return moves;
  } else {
    return 0;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46116/jahr_2014/fruehjahr/Hanoi.java`

(a) Beweisen Sie formal mittels vollständiger Induktion, dass zum Umlegen von k Scheiben (z. B. vom Turm A zum Turm C) insgesamt $2^k - 1$ Schritte notwendig sind, also dass für $k \geq 0$ folgender Zusammenhang gilt:

$$\text{hanoi}(k, \text{'A'}, \text{'C'}) = 2^k - 1$$

Lösungsvorschlag

Zu zeigen:

$$\text{hanoi}(k, \text{'A'}, \text{'C'}) = 2^k - 1$$

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. ———————————

$k = 0$

$$\text{hanoi}(0, \text{'A'}, \text{'C'}) = 0$$

$$2^0 - 1 = 1 - 1 = 0$$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———————————

$$\text{hanoi}(k, \text{'A'}, \text{'C'}) = 2^k - 1$$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k + 1)$ wahr sein muss.

$$\text{hanoi}(k, \text{'A'}, \text{'C'}) = 1 + \text{hanoi}(k - 1, \text{'A'}, \text{'B'}) + \text{hanoi}(k - 1, \text{'B'}, \text{'C'})$$

Terminierungsfunktion
Kontrollflussgraph

$k \to k+1$

$$
\begin{aligned}
\text{hanoi}(k+1,\text{'A'},\text{'C'}) &= 1 + \text{hanoi}((k+1)-1,\text{'A'},\text{'B'}) + \\
&\quad\ \text{hanoi}((k+1)-1,\text{'B'},\text{'C'}) \\
&= 1 + \text{hanoi}(k,\text{'A'},\text{'B'}) + \\
&\quad\ \text{hanoi}(k,\text{'B'},\text{'C'}) && k+1-1=k \\
&= 1 + 2^k - 1 + 2^k - 1 && \text{Formeln eingesetzt} \\
&= 2^k + 2^k - 1 && 1-1-1=-1 \\
&= 2 \cdot 2^k - 1 && 2^k + 2^k = 2 \cdot 2^k \\
&= 2^{k+1} - 1 && 2 \cdot 2^k = 2^{k+1}
\end{aligned}
$$

(b) Geben Sie eine geeignete Terminierungsfunktion an und begründen Sie kurz Ihre Wahl!

Lösungsvorschlag

Betrachte die Argumentenfolge $k, k-1, k-2, \ldots, 0$. Die Terminierungsfunktion ist offenbar $T(k) = k$. $T(k)$ ist bei jedem Rekursionsschritt auf der Folge der Argumente streng monoton fallend. Bei der impliziten Annahme k ist ganzzahlig und $k \geq 0$ ist $T(k)$ nach unten durch 0 beschränkt.

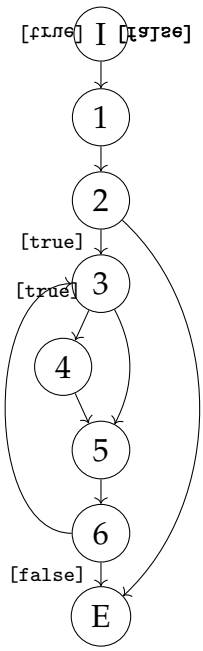
46116 / 2015 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2

Gegeben sei folgende Methode `isPalindrom` und ihr Kontrollflussgraph:
Abkürzungen: I = Import, E = Export

```
boolean isPalindrom(String s) {
  boolean yesItIs = true;
  if (s != null && s.length() > 1) {
    do {
      if (s.charAt(0) != s.charAt(s.length() -
      ↪  1)) {
        yesItIs = false;
      }
      s = s.substring(1, s.length() - 1);

    } while (yesItIs && s.length() > 1);
  }
  return yesItIs;
}
```

Code-Beispiel auf Github ansehen:
`src/main/java/org/bschlangaul/aufgaben/sosy/pu_5/Aufgabe2.java`

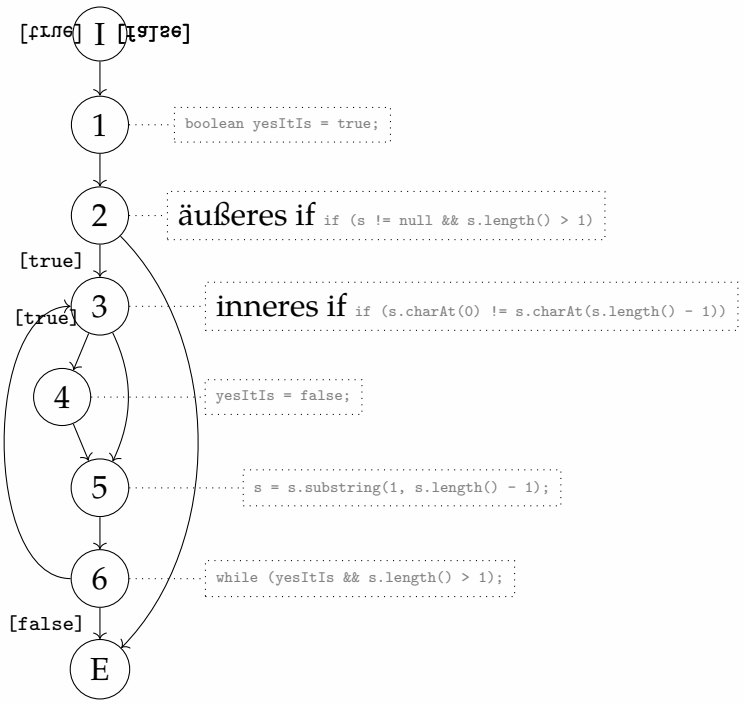


C1-Test Zweigüberdeckung (Branch Coverage)
C2b Schleife-Inneres-Pfadüberdeckung (Boundary-Interior Path Coverage)

(a) Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, die zum Erzielen einer vollständigen ... mit **minimaler** Testfallanzahl und **möglichst kurzen** Pfaden genügen würden.

Lösungsvorschlag

Bemerkung: In der Aufgabenstellung steht *„Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, [...] “*. Das bedeutet, dass es hier erstmal egal ist, ob ein Pfad im Code möglich ist oder nicht!



(i) Verzweigungsüberdeckung (Branch-Coverage, C_1)

Lösungsvorschlag

Pfad 1 (p1) (I) - (1) - (2) - (E)
(äußere `if`-Bedingung `false`)

Pfad 2 (p2) (I) - (1) - (2) - (3) - (5) - (6) - (3) - (4) - (5) - (6) - (E)
(äußere `if`-Bedingung `true`, innere `if`-Bedingung `false`, Wiederholung, innere `if`-Bedingung `true`, keine Wiederholung)

(ii) Schleife-Inneres-Überdeckung (Boundary-Interior-Coverage, $C_{\infty,2}$)

Lösungsvorschlag

ohne Ausführung der Wiederholung (äußere Pfade): p1 (siehe oben)
(I) - (1) - (2) - (E)

Boundary-Test: (alle Pfade, die die Wiederholung betreten, aber nicht wiederholen; innerhalb des Schleifenrumpfes alle Pfade!)

interior-Test: (alle Pfade mit *einer Wiederholung des Schleifenrumpfes*; innerhalb des Schleifenrumpfes wieder alle Pfade!)

Zyklomatische Komplexität nach Mc-Cabe

innere `if`-Bedingung `true`: ③ - ④ - ⑤ - ⑥
innere `if`-Bedingung `false`: ③ - ⑤ - ⑥

p5 ① - ① - ② - ③ - ④ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - Ⓔ
(innere `if`-Bedingung `true`, innere `if`-Bedingung `true`)

p2 (siehe oben) ① - ① - ② - ③ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - Ⓔ
(innere `if`-Bedingung `false`, innere `if`-Bedingung `true`)

p6 ① - ① - ② - ③ - ④ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - Ⓔ
(innere `if`-Bedingung `true`, innere `if`-Bedingung `false`)

p7 ① - ① - ② - ③ - ⑤ - ⑥ - ③ - ⑤ - ⑥ - Ⓔ
(innere `if`-Bedingung `false`, innere `if`-Bedingung `false`)

mit **minimaler** Testfallanzahl und **möglichst kurzen** Pfaden genügen würden.

(b) Welche der vorangehend ermittelten Pfade für die $C_{\infty,2}$-Überdeckung sind mittels Testfällen tatsächlich überdeckbar (*„feasible“*)? Falls der Pfad ausführbar ist, geben Sie den zugehörigen Testfall an - andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

Lösungsvorschlag

p1 `s = "a";`

p2 `s = "abaa";`

p3 `s = "ab";`

p4 `s = "aa";`

p5 nicht überdeckbar, da `yesItIs = false`, wenn innere `if`-Bedingung `true`) keine Wiederholung!

p6 nicht überdeckbar, da `yesItIs = false`, wenn innere `if`-Bedingung `true`) keine Wiederholung!

p7 `s = "abba";`

(c) Bestimmen Sie anhand des Kontrollflussgraphen des obigen Code-Fragments die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

Lösungsvorschlag

$M = b + p = 3 + 1 = 4$

(b: Anzahl Binärverzweigungen, p: Anzahl Zusammenhangskomponenten)

Alternativ

$M = e - n + 2p = 10 - 8 + 2 = 4$

(e: Anzahl Kanten, n: Anzahl Knoten, p: Anzahl Zusammenhangskomponenten)

C2a Vollständige Pfadüberdeckung (Full Path Coverage)
Datenfluss-annotierter Kontrollflussgraph
Vollständige Induktion

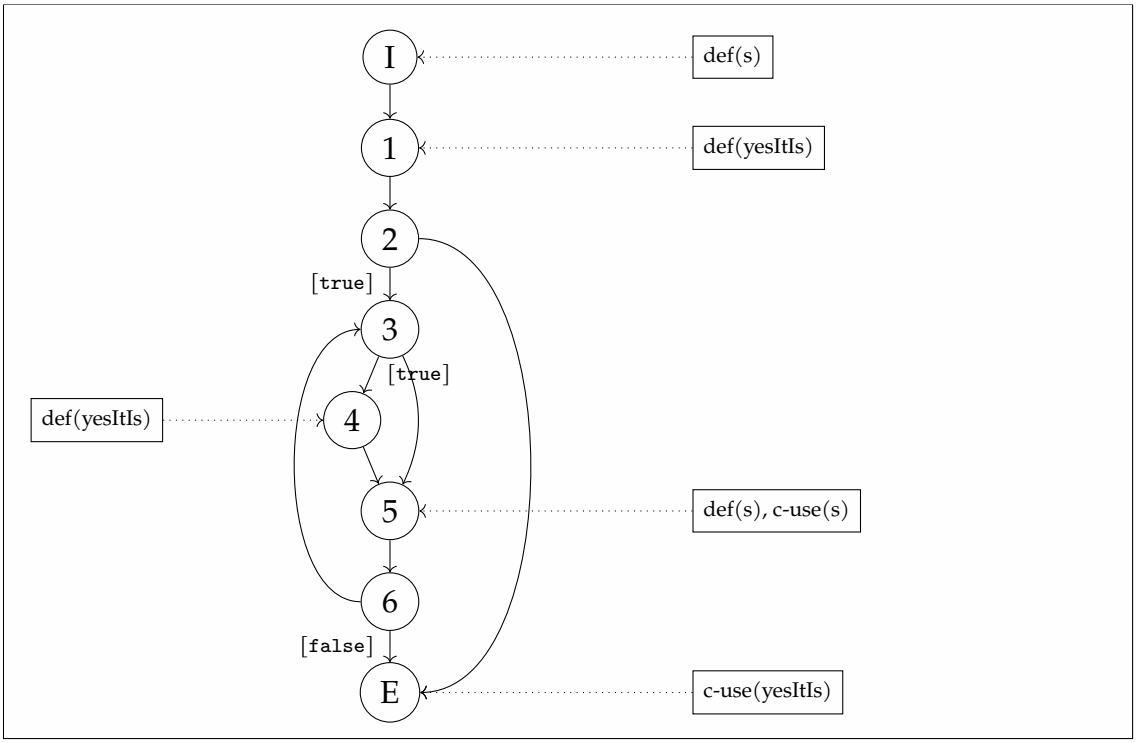
(d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

Lösungsvorschlag

Eine 100%-ige Pfadüberdeckung kann nicht erzielt werden, da es zum einen unüberdeckbare Pfade gibt (vgl. Teilaufgabe b). Zum anderen ist das Testen aller Testfälle nicht möglich, da die Anzahl an Zeichen des übergebenen Wortes nicht begrenzt ist und es somit eine unendliche Anzahl an Testfällen gibt.

(e) Übernehmen Sie den vorgegebenen Kontrollflussgraphen und annotieren Sie ihn mit allen relevanten Datenflussereignissen. Geben Sie jeweils an, ob die Verwendungen berechnend (`c-use`) oder prädikativ (`p-use`) sind.

Lösungsvorschlag



46116 / 2016 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 4

Gegeben sei folgende rekursive Methodendeklaration in der Sprache Java. Es wird als Vorbedingung vorausgesetzt, dass die Methode `cn` nur für Werte $n \geq 0$ aufgerufen wird.

```
int cn(int n) {
  if (n == 0)
    return 1;
  else
    return (4 * (n - 1) + 2) * cn(n - 1) / (n + 1);
}
```

Sie können im Folgenden vereinfachend annehmen, dass es keinen Überlauf in der Berechnung gibt, ðdass der Datentyp `int` für die Berechnung des Ergebnisses stets ausreicht.

(a) Beweisen Sie mittels vollständiger Induktion, dass der Methodenaufruf `cn(n)` für jedes $n \geq 0$ die n-te Catalan-Zahl C_n berechnet, wobei

$$C_n = \frac{(2n)!}{(n+1)! \cdot n!}$$

Exkurs: Fakultät

Für alle natürlichen Zahlen n ist

$$n! = 1 \cdot 2 \cdot 3 \cdots n = \prod_{k=1}^{n} k$$

als das Produkt der natürlichen Zahlen von 1 bis n definiert. Da das leere Produkt stets 1 ist, gilt

$$0! = 1$$

Die Fakultät lässt sich auch rekursiv definieren:

$$n! = \begin{cases} 1, & n = 0 \\ n \cdot (n-1)!, & n > 0 \end{cases}$$

Fakultäten für negative oder nicht ganze Zahlen sind nicht definiert. Es gibt aber eine Erweiterung der Fakultät auf solche Argumente [a]

[a] `https://de.wikipedia.org/wiki/Fakultät_(Mathematik)`

Exkurs: Catalan-Zahl

Die Catalan-Zahlen bilden eine Folge natürlicher Zahlen, die in vielen Problemen der Kombinatorik auftritt. Sie sind nach dem belgischen Mathematiker Eugène Charles Catalan benannt.

Die Folge der Catalan-Zahlen $C_0, C_1, C_2, C_3, \ldots$ beginnt mit $1, 1, 2, 5, 14, 42, 132, \ldots$ [a]

[a] `https://de.wikipedia.org/wiki/Catalan-Zahl`

Beim Induktionsschritt können Sie die beiden folgenden Gleichungen verwenden:

(i) $(2(n+1))! = (4n+2) \cdot (n+1) \cdot (2n)!$

(ii) $(a+2)! \cdot (n+1)! = (n+2) \cdot (n+1) \cdot (n+1)! \cdot n!$

Lösungsvorschlag

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. —

$$\begin{aligned} C_0 &= \frac{(2 \cdot 0)!}{(0+1)! \cdot 0!} \\ &= \frac{0!}{1! \cdot 0!} \\ &= \frac{1}{1 \cdot 1} \\ &= \frac{1}{1} \\ &= 1 \end{aligned}$$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———————

$$C_n = \frac{(2n)!}{(n+1)! \cdot n!}$$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k + 1)$ wahr sein muss.

Vom Code ausgehend

$$
\begin{aligned}
C_{n+1} &= \frac{(4 \cdot (n+1-1)+2) \cdot \mathrm{cn}(n+1-1)}{n+1+1} && \text{Java nach Mathe} \\
&= \frac{(4n+2) \cdot \mathrm{cn}(n)}{n+2} && \text{addiert, subtrahiert} \\
&= \frac{(4n+2) \cdot (2n)!}{(n+2) \cdot (n+1)! \cdot n!} && \text{für cn(n) Formel eingesetzt} \\
&= \frac{(4n+2) \cdot (2n)! \cdot (n+1)}{(n+2) \cdot (n+1)! \cdot n! \cdot (n+1)} && (n+1) \text{ multipliziert} \\
&= \frac{(4n+2) \cdot (n+1) \cdot (2n)!}{(n+2) \cdot (n+1)! \cdot (n+1) \cdot n!} && \text{umsortiert} \\
&= \frac{(2(n+1))!}{(n+2)! \cdot (n+1)!} && \text{Hilfsgleichungen verwendet} \\
&= \frac{(2(n+1))!}{((n+1)+1)! \cdot (n+1)!} && (n+1) \text{ verdeutlicht}
\end{aligned}
$$

Mathematische Herangehensweise

$$
\begin{aligned}
C_{n+1} &= \frac{(2(n+1))!}{((n+1)+1)! \cdot (n+1)!} && n+1 \text{ in } C_n \text{ eingesetzt} \\
&= \frac{(2(n+1))!}{(n+2)! \cdot (n+1)!} && \text{addiert} \\
&= \frac{(4n+2) \cdot (n+1) \cdot (2n)!}{(n+2) \cdot (n+1) \cdot (n+1)! \cdot n!} && \text{Hilfsgleichungen verwendet} \\
&= \frac{(4n+2) \cdot (2n)!}{(n+2) \cdot (n+1)! \cdot n!} && (n+1) \text{ gekürzt} \\
&= \frac{4n+2}{n+2} \cdot C_n && \text{Catalan-Formel ersetzt} \\
&= \frac{4((n+1)-1)+2}{(n+1)+1} \cdot C_{(n+1)-1} && (n+1) \text{ verdeutlicht}
\end{aligned}
$$

(b) Geben Sie eine geeignete Terminierungsfunktion an und begründen Sie, warum der Methodenaufruf `cn(n)` für jedes $n \geq 0$ terminiert.

SQL CREATE TABLE

Lösungsvorschlag

$T(n) = n$. Diese Funktion verringert sich bei jedem Rekursionsschritt um eins. Sie ist monoton fallend und für $T(0) = 0$ definiert. Damit ist sie eine Terminierungsfunktion für `cn(n)`.

46116 / 2018 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 3

Gegeben sei das folgende Datenbank-Schema, das für die Speicherung der Daten einer Schule entworfen wurde, zusammen mit einem Teil seiner Ausprägung. Die Primärschlüssel-Attribute sind jeweils unterstrichen.

Die Relation *Schüler* enthält allgemeine Daten zu den Schülerinnen und Schülern. Schülerinnen und Schüler nehmen an Prüfungen in verschiedenen Unterrichtsfächern teil und erhalten dadurch Noten. Diese werden in der Relation *Noten* abgespeichert. Prüfungen haben ein unterschiedliches Gewicht. Beispielsweise hat ein mündliches Ausfragen oder eine Extemporale das Gewicht 1, während eine Schulaufgabe das Gewicht 2 hat.

Schüler:

SchülerID	**Vorname**	**Nachname**	**Klasse**
1	Laura	Müller	4A
2	Linus	Schmidt	4A
3	Jonas	Schneider	4A
4	Liam	Fischer	4B
5	Tim	Weber	4B
6	Lea	Becker	4B
7	Emilia	Klein	4C
8	Julia	Wolf	4C

Noten:

SchülerID[Schüler]	**Schulfach**	**Note**	**Gewicht**	**Datum**
1	Mathematik	3	2	23.09.2017
1	Mathematik	1	1	03.10.2017
1	Mathematik	2	2	15.10.2017
1	Mathematik	4	1	11.11.2017

(a) Geben Sie die SQL-Befehle an, die notwendig sind, um die oben dargestellten Tabellen in einer SQL-Datenbank anzulegen.

Lösungsvorschlag

INSERT
ALTER TABLE

```
CREATE TABLE IF NOT EXISTS Schüler (
  SchülerID INTEGER PRIMARY KEY NOT NULL,
  Vorname VARCHAR(20),
  Nachname VARCHAR(20),
  Klasse VARCHAR(5)
);

CREATE TABLE IF NOT EXISTS Noten (
  SchülerID INTEGER NOT NULL,
  Schulfach VARCHAR(20),
  Note INTEGER,
  Gewicht INTEGER,
  Datum DATE,
  PRIMARY KEY (SchülerID, Schulfach, Datum),
  FOREIGN KEY (SchülerID) REFERENCES Schüler(SchülerID)
);
```

(b) Entscheiden Sie jeweils, ob folgende Einfügeoperationen vom gegebenen Datenbanksystem (mit der angegebenen Ausprägungen) erfolgreich verarbeitet werden können und begründen Sie Ihre Antwort kurz.

```
INSERT INTO Schüler
  (SchülerID, Vorname, Nachname, Klasse)
VALUES
  (6, 'Johannes', 'Schmied', '4C');
```

Lösungsvorschlag

Nein. Ein/e Schüler/in mit der ID 6 existiert bereits. Primärschlüssel müssen eindeutig sein.

```
INSERT INTO Noten VALUES (6, 'Chemie', 1, 2, '1.4.2020');
```

Lösungsvorschlag

Nein. Ein *Datum* ist zwingend notwendig. Da *Datum* im Primärschlüssel enthalten ist, darf es nicht NULL sein. Es gibt auch keine/n Schüler/in mit der ID 9. Der/die Schüler/in müsste vorher angelegt werden, da die Spalte *SchülerID* von der Tabelle *Noten* auf den Fremdschlüssel *SchülerId* aus der Schülertabelle verweist.

(c) Geben Sie die Befehle für die folgenden Aktionen in SQL an. Beachten Sie dabei, dass die Befehle auch noch bei Änderungen des oben gegebenen Datenbankzustandes korrekte Ergebnisse zurückliefern müssen.

- Die Schule möchte verhindern, dass in die Datenbank mehrere Kinder mit dem selben Vornamen in die gleiche Klasse kommen. Dies soll bereits auf Datenbankebene verhindert werden. Dabei sollen die Primärschlüssel nicht verändert werden. Geben Sie den Befehl an, der diese Änderung durchführt.

Lösungsvorschlag


```
ALTER TABLE Schüler
ADD CONSTRAINT eindeutiger_Vorname UNIQUE (Vorname, Klasse);
```

- Der Schüler *Tim Weber* (SchülerID: 5) wechselt die Klasse. Geben Sie den SQL-Befehl an, der den genannten Schüler in die Klasse *„4C“* überführt.

Lösungsvorschlag

```
UPDATE Schüler
SET Klasse = '4C'
WHERE
  Vorname = 'Tim' AND
  Nachname = 'Weber' AND
  SchülerID = 5;
```

- Die Schülerin *Laura Müller* (SchülerID: 1) zieht um und wechselt die Schule. Löschen Sie die Schülerin aus der Datenbank. Nennen Sie einen möglichen Effekt, welcher bei der Verwendung von Primär- und Fremdschlüsseln auftreten kann.

Lösungsvorschlag

Alle Noten von *Laura Müller* werden gelöscht, falls `ON DELETE CASCADE` gesetzt ist. Oder es müssen erst alle Fremdschlüsselverweise auf diese *SchülerID* in der Tabelle *Noten* gelöscht werden

```
DELETE FROM Noten
WHERE SchülerID = 1;
```

- Erstellen Sie eine View *„DurchschnittsNoten“*, die die folgenden Spalten beinhaltet: *Klasse, Schulfach, Durchschnittsnote*

  Hinweis: Beachten Sie die Gewichte der Noten.

Lösungsvorschlag

```
CREATE VIEW DurchschnittsNoten AS (
  (SELECT s.Klasse, n.Schulfach, (SUM(n.Note * n.Gewicht) /
↪  SUM(n.Gewicht)) AS Durchschnittsnote
  FROM Noten n, Schüler s
  WHERE s.SchülerID = n.SchülerID
  GROUP BY s.Klasse, n.Schulfach)
);

SELECT * FROM DurchschnittsNoten;
```

```
 klasse | schulfach  | durchschnittsnote
--------+------------+-------------------
 4A     | Mathematik |                 2
(1 row)
```

- Geben Sie den Befehl an, der die komplette Tabelle *„Noten“* löscht.

Lösungsvorschlag

```
DROP TABLE Noten;
```

(d) Formulieren Sie die folgenden Anfragen in SQL. Beachten Sie dabei, dass sie SQL-Befehle auch noch bei Änderungen der Ausprägung die korrekten Anfrageergebnisse zurückgeben sollen.

- Gesucht ist die durchschnittliche Note, die im Fach Mathematik vergeben wird.
  Hinweis: Das Gewicht ist bei dieser Anfrage nicht relevant

Lösungsvorschlag

```
SELECT AVG(Note)
FROM Noten
WHERE Schulfach = 'Mathematik';
```

- Berechnen Sie die Anzahl der Schüler, die im Fach Mathematik am 23.09.2017 eine Schulaufgabe (ðGewicht=2) geschrieben haben.

Lösungsvorschlag

```
SELECT COUNT(*) AS Anzahl_Schüler
FROM Noten
WHERE Datum = '23.09.2017' AND Gewicht = 2 AND Schulfach = 'Mathematik';
```

```
 anzahl_schüler
----------------
              1
(1 row)
```

- Geben Sie die *SchülerID* aller Schüler zurück, die im Fach Mathematik mindestens drei mal die Schulnote 6 geschrieben haben.

Lösungsvorschlag

```
SELECT SchülerID
FROM Noten
WHERE Schulfach = 'Mathematik' AND Note = 6
GROUP BY SchülerID
HAVING COUNT(*) >= 3;
```

```
 schülerid
-----------
(0 rows)
```

- Gesucht ist der Notendurchschnitt bezüglich jedes Fachs der Klasse „4A“.

Lösungsvorschlag

```
SELECT n.Schulfach, AVG(n.Note)
FROM Schüler s, Noten n
WHERE s.SchülerID = n.SchülerID AND s.Klasse = '4A'
GROUP BY n.Schulfach;
```

```
schulfach  |        avg
-----------+--------------------
 Mathematik | 2.5000000000000000
```

```
(1 row)
```

(e) Geben Sie jeweils an, welchen Ergebniswert die folgenden SQL-Befehle für die gegebene Ausprägung zurückliefern.

```
SELECT COUNT(DISTINCT Klasse)
FROM
Schüler NATURAL JOIN Noten;
```

Lösungsvorschlag

```
 count
-------
     1
(1 row)
```

4A von Laura Müller. Ohne `DISTINCT` wäre das Ergebnis 4.

```
SELECT COUNT(ALL Klasse)
FROM
Noten, Schüler;
```

Lösungsvorschlag

```
 count
-------
    32
(1 row)
```

Es entsteht das Kreuzprodukt $(8 \cdot 4 = 32)$.

```
SELECT COUNT(Note)
FROM
Schüler NATURAL LEFT OUTER JOIN Noten;
```

Lösungsvorschlag

```
SELECT * FROM Schüler NATURAL LEFT OUTER JOIN Noten;
```

ergibt:

```
 schülerid | vorname | nachname  | klasse | schulfach  | note | gewicht |   datum
-----------+---------+-----------+--------+------------+------+---------+------------
         1 | Laura   | Müller    | 4A     | Mathematik |    3 |       2 | 2017-09-23
         1 | Laura   | Müller    | 4A     | Mathematik |    1 |       1 | 2017-10-03
         1 | Laura   | Müller    | 4A     | Mathematik |    2 |       2 | 2017-10-15
         1 | Laura   | Müller    | 4A     | Mathematik |    4 |       1 | 2017-11-11
         2 | Linus   | Schmidt   | 4A     |            |      |         |
         5 | Tim     | Weber     | 4B     |            |      |         |
         8 | Julia   | Wolf      | 4C     |            |      |         |
         6 | Lea     | Becker    | 4B     |            |      |         |
         4 | Liam    | Fischer   | 4B     |            |      |         |
         3 | Jonas   | Schneider | 4A     |            |      |         |
         7 | Emilia  | Klein     | 4C     |            |      |         |
(11 rows)

 count
-------
     4
(1 row)
```

`COUNT` zählt die `{NULL}`-`Werte` nicht mit. Die *Laura Müller* hat 4 Noten. AVL-Baum

```
SELECT COUNT(*)
FROM
Schüler NATURAL LEFT OUTER JOIN Noten;
```

Lösungsvorschlag

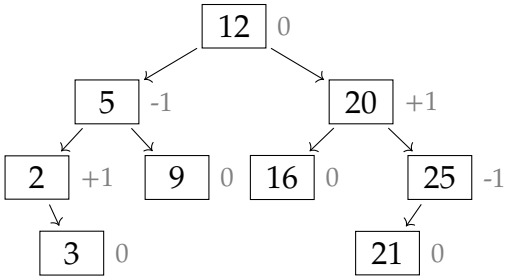
```
 count
-------
    11
(1 row)
```

Siehe Zwischenergebistabelle in der obenstehenden Antwort. Alle Schüler und die Laura 4-mal, weil sie 4 Noten hat.

66115 (Theoretische Informatik / Algorithmen (vertieft))

66115 / 2013 / Herbst / Thema 2 / Aufgabe 8

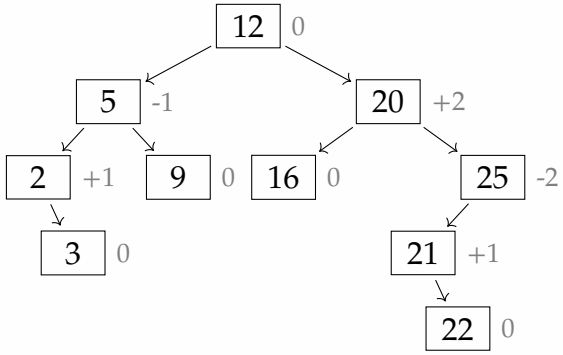
Gegeben sei der folgende AVL-Baum T. Führen Sie auf T folgende Operationen durch.



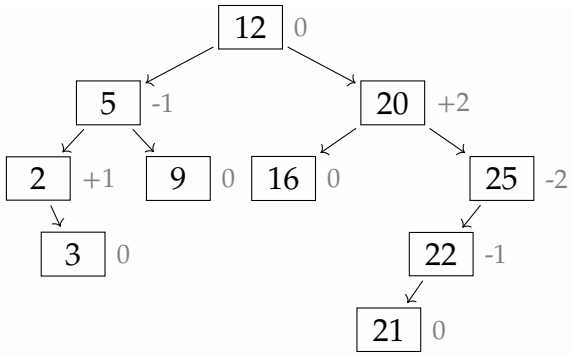
(a) Fügen Sie den Wert 22 in T ein. Balancieren Sie falls nötig und geben Sie den entstandenen Baum (als Zeichnung) an.

Lösungsvorschlag

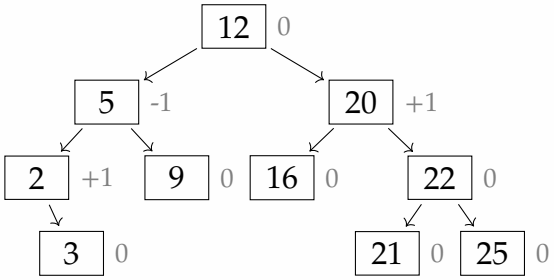
Nach dem Einfügen von „22“:



Nach der Linksrotation:



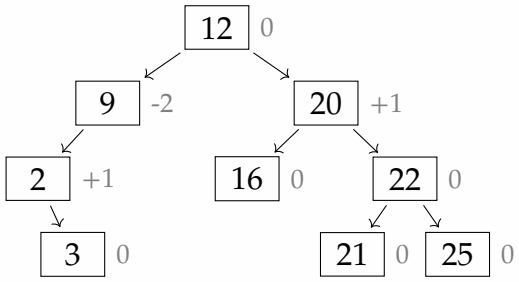
Nach der Rechtsrotation:



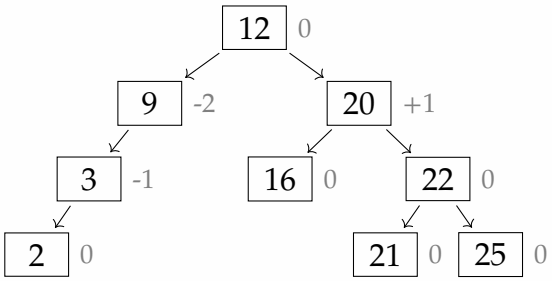
(b) Löschen Sie danach die 5. Balancieren Sie T falls nötig und geben Sie den entstandenen Baum (als Zeichnung) an.

Lösungsvorschlag

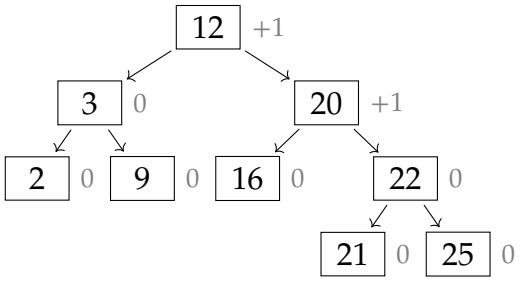
Nach dem Löschen von „5“:



Nach der Linksrotation:



Nach der Rechtsrotation:



66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 1

(a) Gegeben sei die Methode `BigInteger lfBig(int n)` zur Berechnung der eingeschränkten Linksfakultät:

$$!n := \begin{cases} n!(n-1) - (n-1)!(n-2) & \text{falls } 1 < n < 32767 \\ 1 & \text{falls } n = 1 \\ 0 & \text{sonst} \end{cases}$$

```
import java.math.BigInteger;
import static java.math.BigInteger.ZERO;
import static java.math.BigInteger.ONE;

public class LeftFactorial {

  BigInteger sub(BigInteger a, BigInteger b) {
    return a.subtract(b);
  }

  BigInteger mul(BigInteger a, BigInteger b) {
    return a.multiply(b);
  }

  BigInteger mul(int a, BigInteger b) {
    return mul(BigInteger.valueOf(a), b);
  }

  // returns the left factorial !n
  BigInteger lfBig(int n) {
    if (n <= 0 || n >= Short.MAX_VALUE) {
      return ZERO;
    } else if (n == 1) {
      return ONE;
    } else {
      return sub(mul(n, lfBig(n - 1)), mul(n - 1, lfBig(n - 2)));
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/LeftFactorial.java`

Implementieren Sie unter Verwendung des Konzeptes der *dynamischen Programmierung* die Methode `BigInteger dp(int n)`, die jede $!n$ auch bei mehrfachem

Aufrufen mit dem gleichen Parameter höchstens einmal rekursiv berechnet. Sie dürfen der Klasse `LeftFactorial` genau ein Attribut beliebigen Datentyps hinzufügen und die in `lfBig(int)` verwendeten Methoden und Konstanten ebenfalls nutzen.

Lösungsvorschlag

Wir führen ein Attribut mit dem Namen `store` ein und erzeugen ein Feld vom Typ `BigInteger` mit der Länge $n+1$. Die Länge des Feld $n+1$ hat den Vorteil, dass nicht ständig $n-1$ verwendet werden muss, um den gewünschten Wert zu erhalten.

In der untenstehenden Implementation gibt es zwei Methoden mit dem Namen `dp`. Die untenstehende Methode ist nur eine Hüllmethode, mit der nach außen hin die Berechnung gestartet und das `store`-Feld neu gesetzt wird. So ist es möglich `dp()` mehrmals hintereinander mit verschiedenen Werten aufzurufen (siehe `main()`-Methode).

```
BigInteger[] store;

BigInteger dp(int n, BigInteger[] store) {
  if (n > 1 && store[n] != null) {
    return store[n];
  }
  if (n <= 0 || n >= Short.MAX_VALUE) {
    return ZERO;
  } else if (n == 1) {
    return ONE;
  } else {
    BigInteger result = sub(mul(n, dp(n - 1, store)), mul(n - 1, dp(n - 2,
    ↪  store)));
    store[n] = result;
    return result;
  }
}

BigInteger dp(int n) {
  store = new BigInteger[n + 1];
  return dp(n, store);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/LeftFactorial.java`

(b) Betrachten Sie nun die Methode `lfLong(int)` zur Berechnung der vorangehend definierten Linksfakultät ohne obere Schranke. Nehmen Sie im Folgenden an, dass der Datentyp `long` unbeschränkt ist und daher kein Überlauf auftritt.

```
long lfLong(int n) {
  if (n <= 0) {
    return 0;
  } else if (n == 1) {
    return 1;
  } else {
```

```
    return n * lfLong(n - 1) - (n - 1) * lfLong(n - 2);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/LeftFactorial.java`

Beweisen Sie *formal* mittels *vollständiger Induktion*:

$$\forall n \geq 0 : \text{lfLong}(n) \equiv \sum_{k=0}^{n-1} k!$$

Lösungsvorschlag

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. ———————————

$$n = 1 \Rightarrow \texttt{lfLong(1)} = 1 = \sum_{k=0}^{n-1} k! = 0! = 1$$

$$\begin{aligned} n = 2 \Rightarrow & \text{lfLong}(2) \\ & = (n+1) \sum_{k=0}^{n-1} k! - n \sum_{k=0}^{n-2} k! \\ & = \texttt{2 * lfLong(1) - 1 * lfLong(0)} \\ & = 2 \\ & = \sum_{k=0}^{1} k! \\ & = 1! + 0! \\ & = 1 + 1 \\ & = 2 \end{aligned}$$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———————————

$$\texttt{lfLong(n)} = \sum_{k=0}^{n-1} k!$$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k + 1)$ wahr sein muss.

$A(n+1)$

$$
\begin{aligned}
&= \texttt{lfLong(n + 1)} \\
&= \texttt{(n + 1) * lfLong(n) - n * lfLong(n - 1)} \\
&= (n+1)\sum_{k=0}^{n-1} k! - n\sum_{k=0}^{(n-1)-1} k! && \text{Formel eingesetzt} \\
&= (n+1)\sum_{k=0}^{n-1} k! - n\sum_{k=0}^{n-2} k! && \text{subtrahiert} \\
&= n\sum_{k=0}^{n-1} k! + \sum_{k=0}^{n-1} k! - n\sum_{k=0}^{n-2} k! && \text{ausmultipliziert mit } (n+1) \\
&= \sum_{k=0}^{n-1} k! + n\sum_{k=0}^{n-1} k! - n\sum_{k=0}^{n-2} k! && \text{Reihenfolge der Terme geändert} \\
&= \sum_{k=0}^{n-1} k! + n\left((n-1)! + \sum_{k=0}^{n-2} k!\right) - n\sum_{k=0}^{n-2} k! && (n-1)! \text{ aus Summenzeichen entfernt} \\
&= \sum_{k=0}^{n-1} k! + n\left((n-1)! + \sum_{k=0}^{n-2} k! - \sum_{k=0}^{n-2} k!\right) && \text{Distributivgesetz } ac - bc = (a-b)c \\
&= \sum_{k=0}^{n-1} k! + n(n-1)! && +\Sigma - \Sigma = 0 \\
&= \sum_{k=0}^{n-1} k! + n! && \text{Fakultät erhöht} \\
&= \sum_{k=0}^{n} k! && \text{Element zum Summenzeichen hinzugefügt} \\
&= \sum_{k=0}^{(n+1)-1} k! && \text{mit } (n+1) \text{ an der Stelle von } n
\end{aligned}
$$

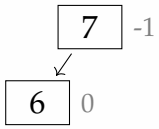
66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 3

(a) Fügen Sie die Zahlen (7, 6, 2, 1, 5, 3, 8, 4) in dieser Reihenfolge in einen anfangs leeren AVL Baum ein. Stellen Sie die AVL Eigenschaft ggf. nach jedem Einfügen mit geeigneten Rotationen wieder her. Zeichnen Sie den AVL Baum einmal vor und einmal nach jeder einzelnen Rotation.

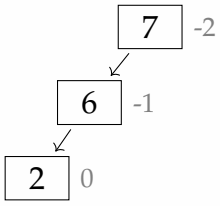
Lösungsvorschlag

Nach dem Einfügen von „7“:

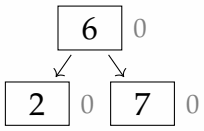
Nach dem Einfügen von „6“:



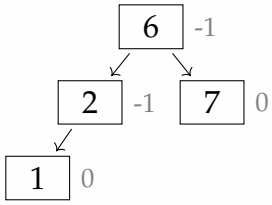
Nach dem Einfügen von „2“:



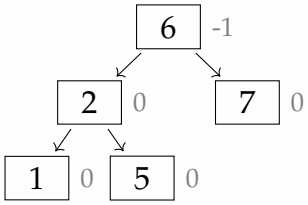
Nach der Rechtsrotation:



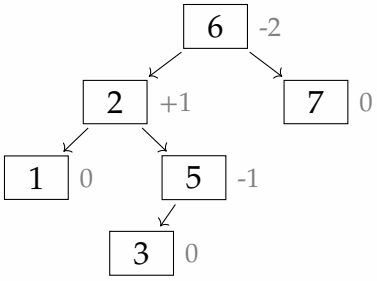
Nach dem Einfügen von „1“:



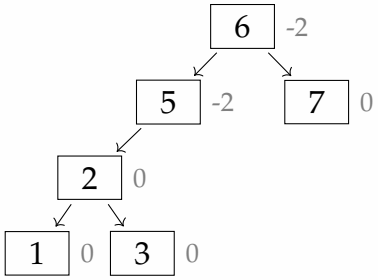
Nach dem Einfügen von „5“:



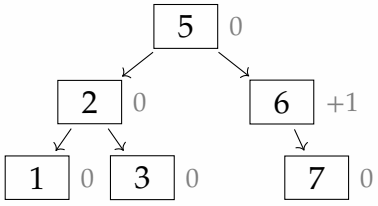
Nach dem Einfügen von „3“:



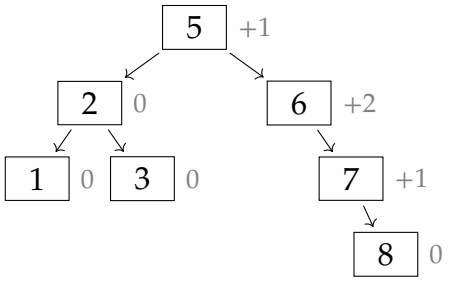
Nach der Linksrotation:



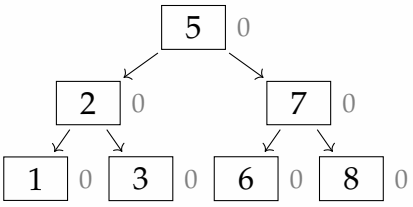
Nach der Rechtsrotation:



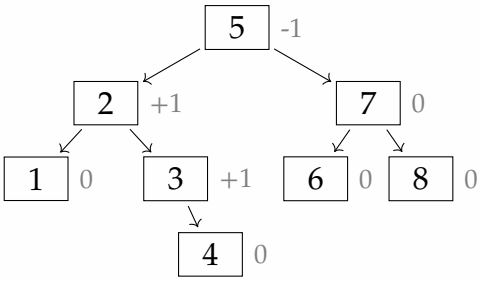
Nach dem Einfügen von „8“:



Nach der Linksrotation:

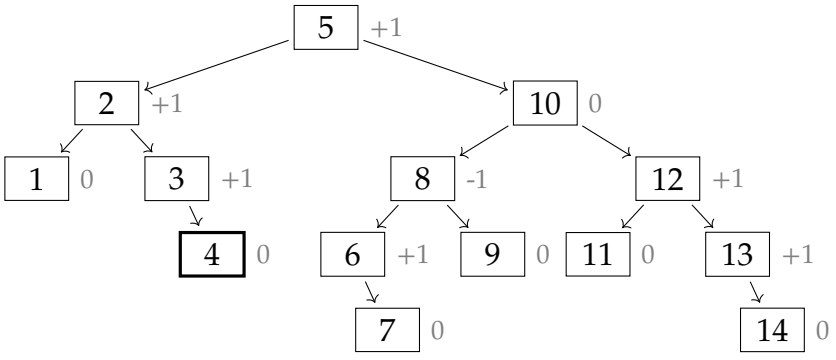


Nach dem Einfügen von „4“:



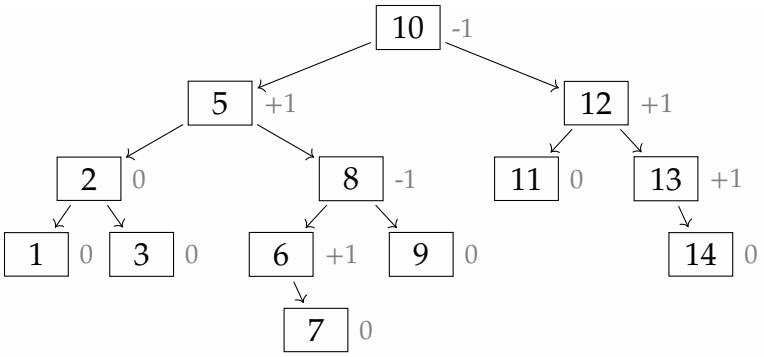
(b) Entfernen Sie den jeweils markierten Knoten aus den folgenden AVL-Bäumen. Stellen Sie die AVL-Eigenschaft ggf. durch geeignete Rotationen wieder her. Zeichnen Sie nur den resultierenden Baum.

(i) Baum 1:

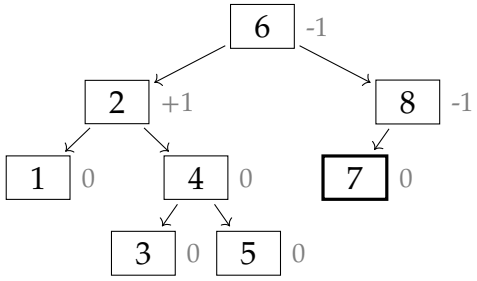


Lösungsvorschlag

Nach dem Löschen von „4“:

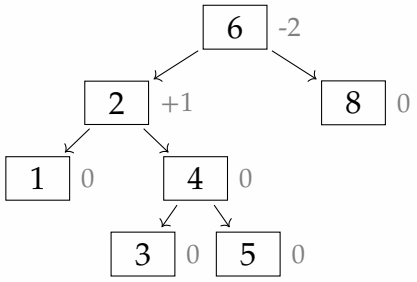


(ii) Baum 2:

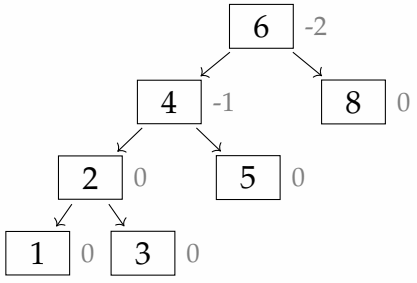


Lösungsvorschlag

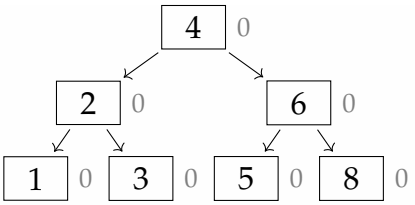
Nach dem Löschen von „7“:



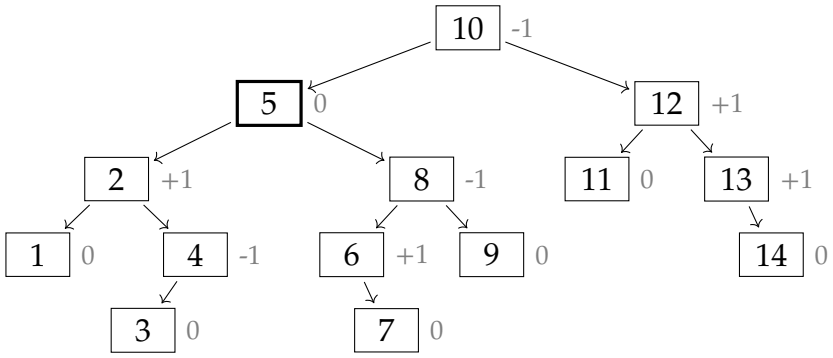
Nach der Linksrotation:



Nach der Rechtsrotation:

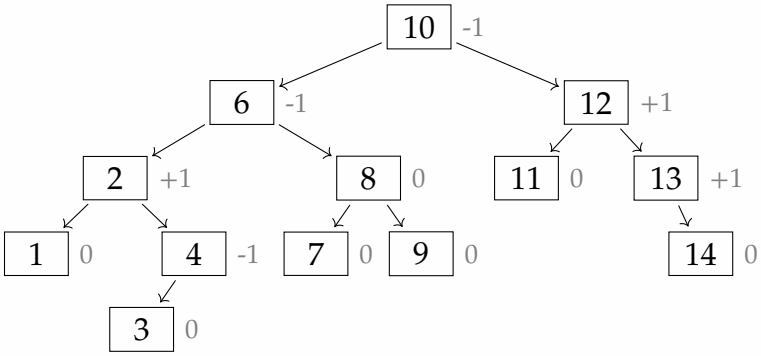


(iii) Baum 3:



Lösungsvorschlag

Nach dem Löschen von „5“:



66115 / 2017 / Frühjahr / Thema 1 / Aufgabe 4

Sie dürfen im Folgenden davon ausgehen, dass keinerlei Under- oder Overflows auftreten.

Gegeben sei folgende rekursive Methode für $n \geq 0$:

```
long sumOfSquares (long n) {
  if (n == 0)
    return 0;
  else
    return n * n + sumOfSquares(n - 1);
}
```

(a) Beweisen Sie formal mittels vollständiger Induktion:

$$\forall n \in \mathbb{N} : \texttt{sumOfSquares(n)} = \frac{n(n+1)(2n+1)}{6}$$

Lösungsvorschlag

Sei $f(n) : \frac{n(n+1)(2n+1)}{6}$

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. ———————————

Für $n = 0$ gilt:

`sumOfSquares(0)` $\overset{\texttt{if}}{=} 0 = f(0)$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. ———————————

Für ein festes $n \in \mathbb{N}$ gelte:

`sumOfSquares(n)` $= f(n)$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k + 1)$ wahr sein muss.

$n \to n + 1$

$$
\begin{aligned}
f(n+1) &= \texttt{sumOfSquares(n+1)} && \text{Java-Methode eingesetzt}\\
&\overset{\texttt{else}}{=} \texttt{(n+1)*(n+1) + sumOfSquares(n)} && \text{Java-Code der else-Verzweigung verwendet}\\
&\overset{\texttt{I.H.}}{=} (n+1)(n+1)+f(n) && \text{mathematisch notiert}\\
&= (n+1)(n+1)+\frac{n(n+1)(2n+1)}{6} && \text{Formel eingesetzt}\\
&= (n+1)^2+\frac{n(n+1)(2n+1)}{6} && \text{potenziert}\\
&= \frac{6(n+1)^2}{6}+\frac{n(n+1)(2n+1)}{6} && (n+1)^2 \text{ in Bruch umgewandelt}\\
&= \frac{6(n+1)^2+n(n+1)(2n+1)}{6} && \text{Addition gleichnamiger Brüche}\\
&= \frac{(n+1)6(n+1)+(n+1)n(2n+1)}{6} && n+1 \text{ ausklammern vorbereitet}\\
&= \frac{(n+1)(6(n+1)+n(2n+1))}{6} && n+1 \text{ ausgeklammert}\\
&= \frac{(n+1)(6n+6+2n^2+n))}{6} && \text{Klammern ausmultipliziert / aufgelöst}\\
&= \frac{(n+1)(2n^2+7n+6)}{6} && \text{umsortiert, addiert } 6n+n=7n\\
&= \frac{(n+1)(2n^2+3n+4n+6)}{6} && \text{Ausklammern vorbereitet}\\
&= \frac{(n+1)(n+2)(2n+3)}{6} && (n+2) \text{ ausgeklammert}\\
&= \frac{(n+1)((n+1)+1)(2(n+1)+1))}{6} && (n+1) \text{ verdeutlicht}
\end{aligned}
$$

[a]

[a] https://mathcs.org/analysis/reals/infinity/answers/sm_sq_cb.html

(b) Beweisen Sie die Terminierung von `sumOfSquares(n)` für alle $n \geq 0$.

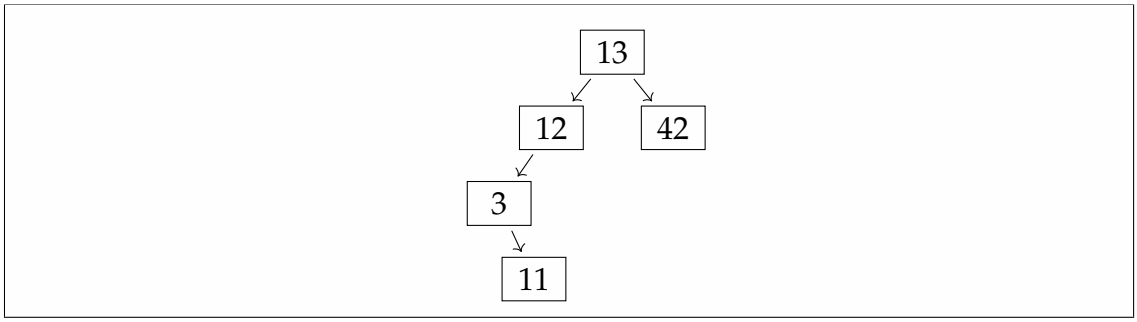
Lösungsvorschlag

Sei $T(n) = n$. Die Funktion $T(n)$ ist offenbar ganzzahlig. In jedem Rekursionsschritt wird n um eins verringert, somit ist $T(n)$ streng monoton fallend. Durch die Abbruchbedingung `n==0` ist $T(n)$ insbesondere nach unten beschränkt. Somit ist T eine gültige Terminierungsfunktion.

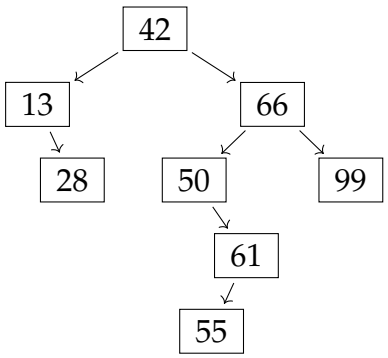
66115 / 2017 / Herbst / Thema 2 / Aufgabe 8

(a) Fügen Sie die Zahlen 13, 12, 42, 3, 11 in der gegebenen Reihenfolge in einen zunächst leeren binären Suchbaum mit aufsteigender Sortierung ein. Stellen Sie nur das Endergebnis dar.

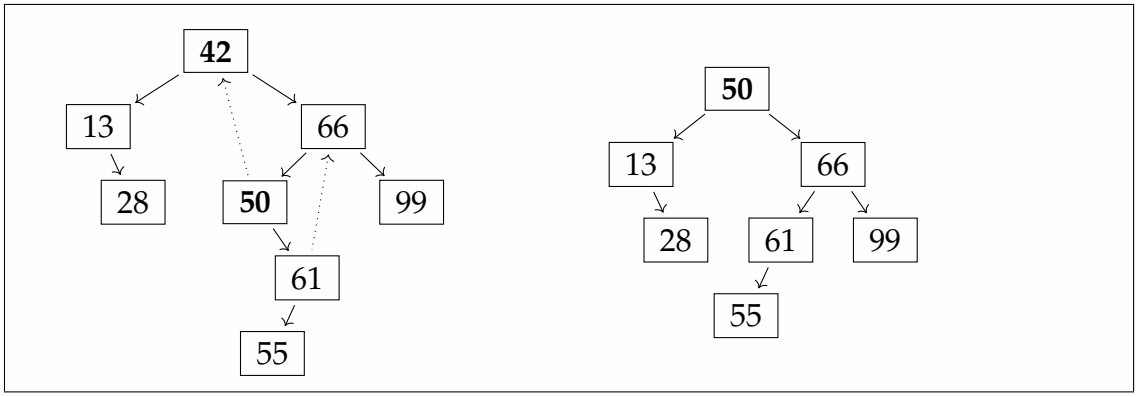
Lösungsvorschlag



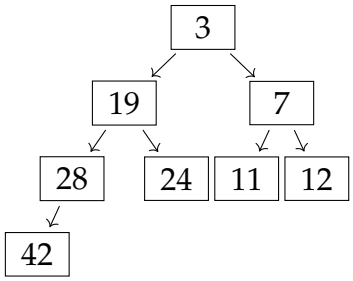
(b) Löschen Sie den Wurzelknoten mit Wert 42 aus dem folgenden *binären* Suchbaum mit aufsteigender Sortierung und ersetzen Sie ihn dabei durch einen geeigneten Wert aus dem *rechten* Teilbaum. Lassen Sie möglichst viele Teilbäume unverändert und erhalten Sie die Suchbaumeigenschaft.



Lösungsvorschlag

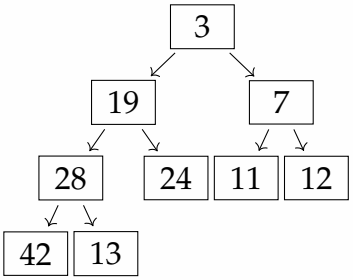


(c) Fügen Sie einen neuen Knoten mit dem Wert 13 in die folgende Min-Halde ein und stellen Sie anschließend die Halden-Eigenschaft vom neuen Blatt aus beginnend wieder her, wobei möglichst viele Knoten der Halde unverändert bleiben und die Halde zu jedem Zeitpunkt links-vollständig sein soll. Geben Sie nur das Endergebnis an.

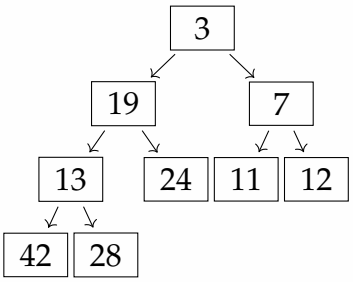


Lösungsvorschlag

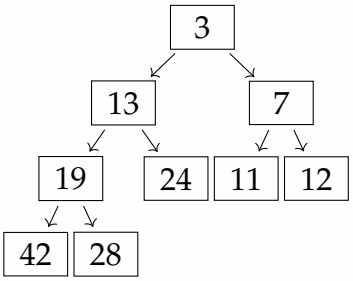
Nach dem Einfügen von „13“:



Nach dem Vertauschen von „13“ und „28“:



Nach dem Vertauschen von „13“ und „19“:

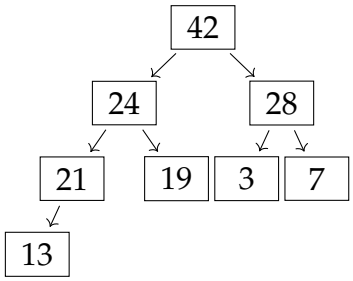


(d) Geben Sie für die ursprüngliche Min-Halde aus Teilaufgabe c)(ðohne den neu eingefügten Knoten mit dem Wert 13) die Feld-Einbettung (Array-Darstellung) an.

Lösungsvorschlag

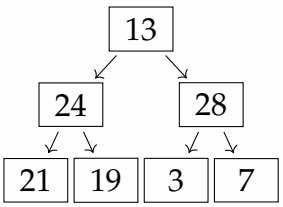
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
3	19	7	28	24	11	12	42

(e) Löschen Sie den Wurzelknoten mit Wert 42 aus der folgenden Max-Halde und stellen Sie anschließend die Halden-Eigenschaft ausgehend von einer neuen Wurzel wieder her, wobei möglichst viele Knoten der Halde unverändert bleiben und die Halde zu jedem Zeitpunkt links-vollständig sein soll. Geben Sie nur das Endergebnis an.

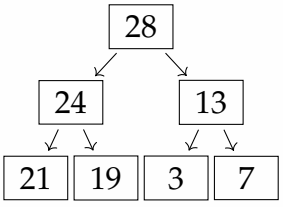


Lösungsvorschlag

Nach dem Ersetzen von „42“ mit „13“:

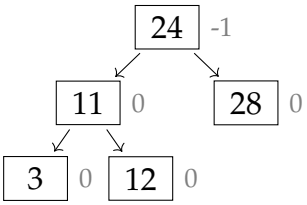


Nach dem Vertauschen von „13“ und „28“:



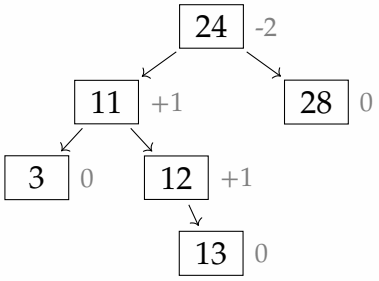
(f) Fügen Sie in jeden der folgenden AVL-Bäume mit aufsteigender Sortierung jeweils einen neuen Knoten mit dem Wert 13 ein und führen Sie anschließend bei Bedarf die erforderliche(n) Rotation(en) durch. Zeichnen Sie den Baum vor und nach den Rotationen.

(i) AVL-Baum A

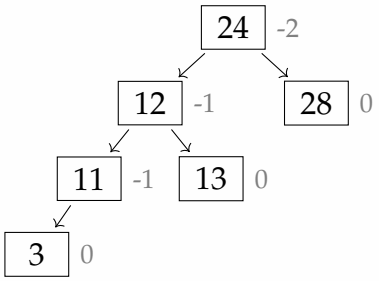


Lösungsvorschlag

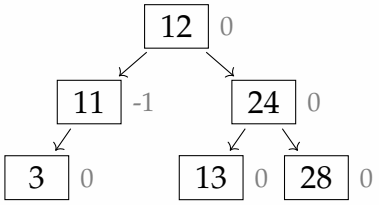
Nach dem Einfügen von „13“:



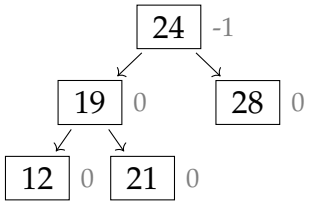
Nach der Linksrotation:



Nach der Rechtsrotation:

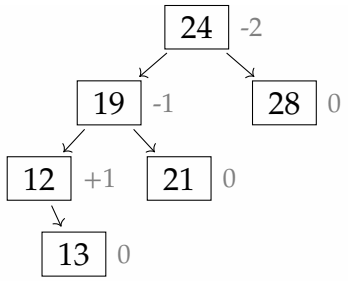


(ii) AVL-Baum B

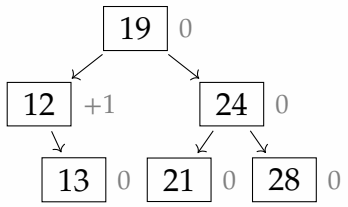


Lösungsvorschlag

Nach dem Einfügen von „13“:

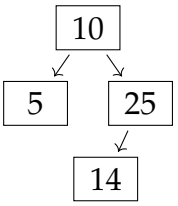


Nach der Rechtsrotation:



66115 / 2019 / Herbst / Thema 2 / Aufgabe 7

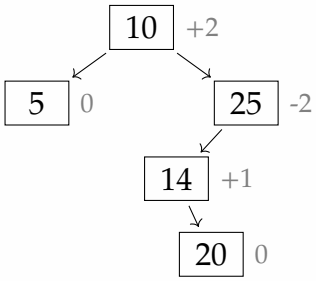
Fügen Sie (manuell) nacheinander die Zahlen $20, 31, 2, 17, 7$ in folgenden AVL-Baum ein. Löschen Sie anschließend aus dem entstandenen Baum nacheinander 14 und 25.



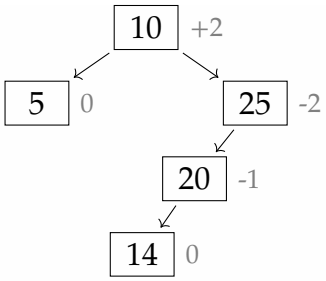
Zeichnen Sie jeweils direkt nach jeder einzelnen Operation zum Einfügen oder Löschen eines Knotens, sowie nach jeder elementaren Rotation den entstehenden Baum. Insbesondere sind evtl. anfallende Doppelrotationen in zwei Schritten darzustellen. Geben Sie zudem an jedem Knoten die Balancewerte an.

Lösungsvorschlag

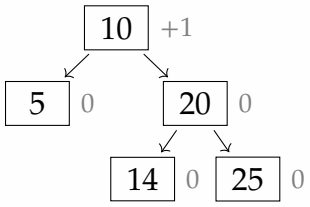
Nach dem Einfügen von „20“:



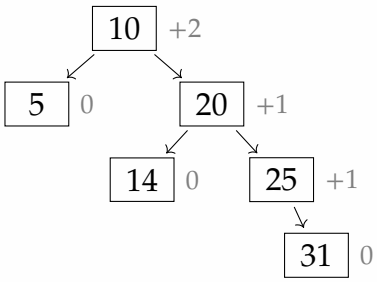
Nach der Linksrotation:



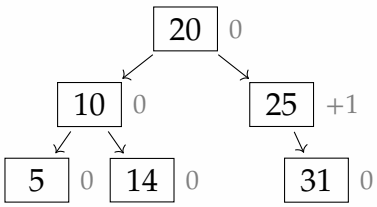
Nach der Rechtsrotation:



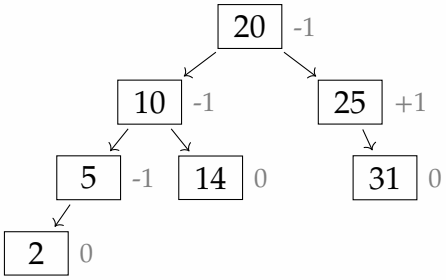
Nach dem Einfügen von „31“:



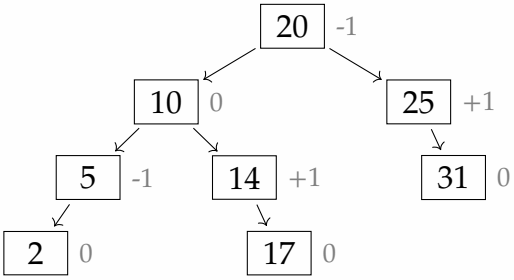
Nach der Linksrotation:



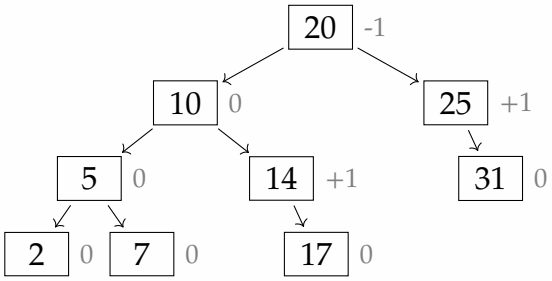
Nach dem Einfügen von „2“:



Nach dem Einfügen von „17“:

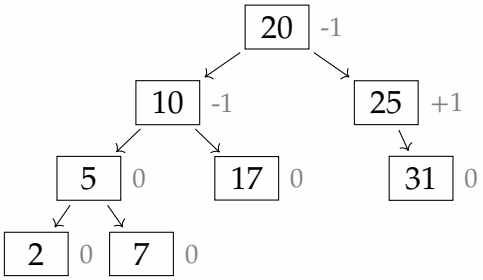


Nach dem Einfügen von „7“:

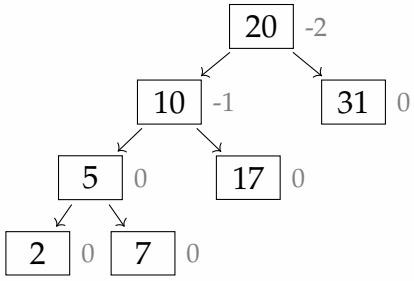


Löschen

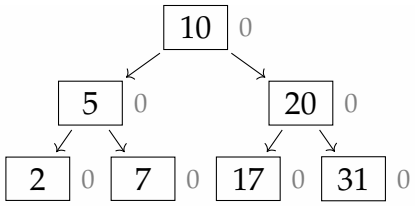
Nach dem Löschen von „14“:



Nach dem Löschen von „25“:



Nach der Rechtsrotation:



66116 (Datenbanksysteme / Softwaretechnologie (vertieft))

66116 / 2016 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Ordnen Sie die folgenden Aussagen entsprechend ihres Wahrheitsgehaltes in einer Tabelle der folgenden Form an:

Kategorie	WAHR	FALSCH
X	X1, X3	X2
Y	Y2	Y1
...	...	...

A Allgemein

A1 Im Software Engineering geht es vor allem darum qualitativ hochwertige Software zu entwickeln.

A2 Software Engineering ist gleichbedeutend mit Programmieren.

B Vorgehensmodelle

B1 Die Erhebung und Analyse von Anforderungen sind nicht Teil des Software Engineerings.

B2 Agile Methoden eignen sich besonders gut für die Entwicklung komplexer und sicherer Systeme in verteilten Entwicklerteams.

B3 Das Spiralmodell ist ein Vorläufer sogenannter Agiler Methoden.

C Anforderungserhebung

C1 Bei der Anforderungserhebung dürfen in keinem Fall mehrere Erhebungstechniken (z. B. Workshops, Modellierung) angewendet werden, weil sonst Widersprüche in Anforderungen zu, Vorschein kommen könnten.

C2 Ein Szenario beinhaltet eine Menge von Anwendungsfällen.

C3 Nicht-funktionale Anforderungen sollten, wenn möglich, immer quantitativ spezifiziert werden.

D Architekturmuster

D1 Schichtenarchitekturen sind besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.

D2 Das Black Board Muster ist besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.

D3 „Dependency Injection“ bezeichnet das Konzept, welches Abhängigkeiten zur Laufzeit reglementiert.

Schichtenarchitektur
Blackboard-Muster
Einbringen von Abhängigkeiten (Dependency Injection)
Sequenzdiagramm
Zustandsdiagramm Wissen
Komponentendiagramm
Modell-Präsentation-Steuerung (Model-View-Controller)
Einzelstück (Singleton)
Kommando (Command)
Validation
Verifikation

E UML

E1 Sequenzdiagramme beschreiben Teile des Verhaltens eines Systems.

E2 Zustandsübergangsdiagramme beschreiben das Verhalten eines Systems.

E3 Komponentendiagramme beschreiben die Struktur eines Systems.

F Entwurfsmuster

F1 Das MVC Pattern verursacht eine starke Abhängigkeit zwischen Datenmodell und Benutzeroberfläche.

F2 Das Singleton Pattern stellt sicher, dass es zur Laufzeit von einer bestimmten Klasse höchstens ein Objekt gibt.

F3 Im Kommando Enwurfsmuster (engl. „Command Pattern“) werden Befehle in einem sog. Kommando-Objekt gekapselt, um sie bei Bedarf rückgängig zu machen.

G Testen

G1 Validation dient der Überprüfung von Laufzeitfehlern.

G2 Testen ermöglicht sicherzustellen, dass ein Programm absolut fehlerfrei ist.

G3 Verifikation dient der Überprüfung, ob ein System einer Spezifikation entspricht.

Lösungsvorschlag

Kategorie	WAHR	FALSCH
A	A1	A2
B	B3	B1, B2
C	C3	C1, C2
D	D3	D1, D2
E	E1, E2, E3	
F	F2, F3	F1
G	G3	G1[a], G2[b]

[a] Validierung: Prüfung der Eignung beziehungsweise der Wert einer Software bezogen auf ihren Einsatzzweck: „Wird das richtige Produkt entwickelt?“

[b] Ein Softwaretest prüft und bewertet Software auf Erfüllung der für ihren Einsatz definierten

Anforderungen und misst ihre Qualität.

66116 / 2017 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Konstruieren Sie einen B-Baum, dessen Knoten maximal 4 Einträge enthalten können, indem Sie der Reihe nach diese Suchschlüsssel einfügen:

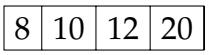
8, 10, 12, 20, 5, 30, 25, 11

Anschließend löschen Sie den Eintrag mit dem Suchschlüssel 8.

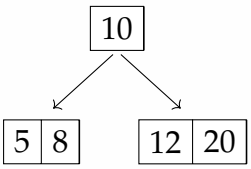
Zeigen Sie jeweils graphisch den entstehenden Baum nach relevanten Zwischenschritten; insbesondere nach Einfügen der 5 sowie nach dem Einfügen der 11 und nach dem Löschen der 8.

Lösungsvorschlag

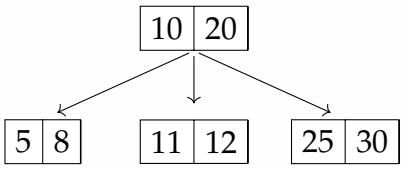
- Schlüsselwert 8 (einfaches Einfügen)
- Schlüsselwert 10 (einfaches Einfügen)
- Schlüsselwert 12 (einfaches Einfügen)
- Schlüsselwert 20 (einfaches Einfügen)



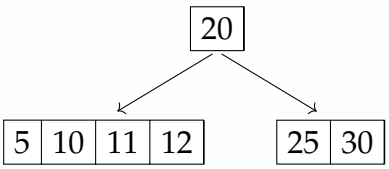
- Schlüsselwert 5 (Split)



- Schlüsselwert 30 (einfaches Einfügen)
- Schlüsselwert 25 (einfaches Einfügen)
- Schlüsselwert 11 (Split)



- Löschen des Schlüsselwerts 8 (Mischen/Verschmelzen)



Index