

66112 Herbst 2003

Automatentheorie / Komplexität / Algorithmen (vertieft)

Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Aufgabenübersicht

Thema Nr. 2	3
Aufgabe 5 [drei hoch]	3
Aufgabe 8 [Klasse „BinBaum“]	3



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.



Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

Thema Nr. 2

Aufgabe 5 [drei hoch]

Zeigen Sie mit Hilfe vollständiger Induktion, dass das folgende Programm bzgl. der Vorbedingung $x > 0$ und der Nachbedingung $\text{drei_hoch } x = 3^x$ partiell korrekt ist!

```
(define (drei_hoch x)
  (cond ((= x 0) 1)
        (else (* 3 (drei_hoch (- x 1)))))
)
```

Lösungsvorschlag

Induktionsanfang

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

$$\text{drei_hoch } 1 = 3 \cdot (\text{drei_hoch } 0) = 3 \cdot 1 = 3$$

Induktionsvoraussetzung

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

für alle $x < x_0$ gilt $\text{drei_hoch } x = 3^x$

Induktionsschritt

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

$x \rightarrow x+1$

$$\begin{aligned} \text{drei_hoch } (x + 1) &= 3 \cdot \text{drei_hoch } (-(x + 1)1) \\ &= 3 \cdot (\text{drei_hoch } x) \\ &= 3 \cdot 3^x \\ &= 3^{x+1} \end{aligned}$$

Aufgabe 8 [Klasse „BinBaum“]

- (a) Implementieren Sie in einer objektorientierten Sprache einen binären Suchbaum für ganze Zahlen! Dazu gehören Methoden zum Setzen und Ausgeben der Attribute `zahl`, `linker_teilbaum` und `rechter_teilbaum`. Design: eine Klasse `Knoten` und eine Klasse `BinBaum`. Ein Knoten hat einen linken und einen rechten Nachfolger. Ein Baum verwaltet die Wurzel. Er hängt neue Knoten an und löscht Knoten.

```
public class BinBaum {  
  
    private Knoten wurzel = null;
```

```

public void setzeWurzel(Knoten knoten) {
    wurzel = knoten;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java)

```

public class Knoten {
    private int zahl;
    private Knoten links = null;
    private Knoten rechts = null;

    public Knoten() {
    }

    public Knoten(int zahl) {
        this.zahl = zahl;
    }

    public void setzeZahl(int zahl) {
        this.zahl = zahl;
    }

    public int gibZahl() {
        return zahl;
    }

    public void setzeLinks(Knoten k) {
        links = k;
    }

    public Knoten gibLinks() {
        return links;
    }

    public void setzeRechts(Knoten k) {
        rechts = k;
    }

    public Knoten gibRechts() {
        return rechts;
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/Knoten.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2003/herbst/Knoten.java)

(b) Schreiben Sie eine Methode `fügeEin(...)`, die eine Zahl in den Baum einfügt!

Lösungsvorschlag

```

public void fügeEin(int zahl) {
    Knoten aktueller = wurzel;
    Knoten neuerKnoten = new Knoten(zahl);
    if (wurzel == null) {
        wurzel = neuerKnoten;
    }
}

```

```

    return;
}
while (aktueller != null) {
    // suche links
    if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() != null) {
        aktueller = aktueller.gibLinks();
        // fuege ein
    } else if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() == null) {
        aktueller.setzeLinks(neuerKnoten);
        break;
    }
    // suche rechts
    if (zahl > aktueller.gibZahl() && aktueller.gibRechts() != null) {
        aktueller = aktueller.gibRechts();
        // fuege ein
    } else if (zahl > aktueller.gibZahl() && aktueller.gibRechts() == null) {
        aktueller.setzeRechts(neuerKnoten);
        break;
    }
}
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java](https://github.com/orgs/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java)

- (c) Schreiben Sie eine Methode `void besuchePostOrder(...)`, die die Zahlen in der Reihenfolge postorder ausgibt!

Lösungsvorschlag

```

public static void besuchePostOrder(Knoten knoten) {
    // Sonderfall leerer (Teil-)Baum
    if (knoten == null) {
        System.out.println("Leerer Baum");
    } else {
        // Linker
        if (knoten.gibLinks() != null) {
            besuchePostOrder(knoten.gibLinks());
        }
        // Rechter
        if (knoten.gibRechts() != null) {
            besuchePostOrder(knoten.gibRechts());
        }
        System.out.println(knoten.gibZahl());
    }
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java](https://github.com/orgs/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java)

- (d) Ergänzen Sie Ihr Programm um die rekursiv implementierte Methode `int berechneSumme(...)`, die die Summe der Zahlen des Unterbaums, dessen Wurzel der Knoten ist, zurückgibt! Falls der Unterbaum leer ist, ist der Rückgabewert 0!

`int summe (Knoten x)...`

```

public int berechneSumme(Knoten knoten) {
    int ergebnis = 0;

    // Sonderfall: leerer Unterbaum
    if (knoten == null) {
        return 0;
    }
    // linker
    if (knoten.gibLinks() != null) {
        ergebnis = ergebnis + berechneSumme(knoten.gibLinks());
    }
    // rechter
    if (knoten.gibRechts() != null) {
        ergebnis = ergebnis + berechneSumme(knoten.gibRechts());
    }
    // Wurzel
    ergebnis = ergebnis + knoten.gibZahl();
    return ergebnis;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java](https://github.com/orgs/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java)

- (e) Schreiben Sie eine Folge von Anweisungen, die einen Baum mit Namen BinBaum erzeugt und nacheinander die Zahlen 5 und 7 einfügt! In den binären Suchbaum werden noch die Zahlen 4, 11, 6 und 2 eingefügt. Zeichnen Sie den Baum, den Sie danach erhalten haben, und schreiben Sie die eingefügten Zahlen in der Reihenfolge der Traversierungsmöglichkeit postorder auf!
- (f) Implementieren Sie eine Operation `isSorted(...)`, die für einen (Teil-)baum feststellt, ob er sortiert ist.

```

public boolean istSortiert(Knoten knoten) {
    // Baum leer
    if (knoten == null) {
        return true;
    }

    // linker Nachfolger nicht okay
    if (knoten.gibLinks() != null && knoten.gibLinks().gibZahl() >
→ knoten.gibZahl()) {
        return false;
    }

    // rechter Nachfolger nicht okay
    if (knoten.gibRechts() != null && knoten.gibRechts().gibZahl() <=
→ knoten.gibZahl()) {
        return false;
    }
    // sonst prüfe Teilbaeume
    return istSortiert(knoten.gibRechts()) && istSortiert(knoten.gibLinks());
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java](#)


```

public class BinBaum {

    private Knoten wurzel = null;

    public void setzeWurzel(Knoten knoten) {
        wurzel = knoten;
    }

    public void fügeEin(int zahl) {
        Knoten aktueller = wurzel;
        Knoten neuerKnoten = new Knoten(zahl);
        if (wurzel == null) {
            wurzel = neuerKnoten;
            return;
        }
        while (aktueller != null) {
            // suche links
            if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() != null) {
                aktueller = aktueller.gibLinks();
                // fuege ein
            } else if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() == null) {
                aktueller.setzeLinks(neuerKnoten);
                break;
            }
            // suche rechts
            if (zahl > aktueller.gibZahl() && aktueller.gibRechts() != null) {
                aktueller = aktueller.gibRechts();
                // fuege ein
            } else if (zahl > aktueller.gibZahl() && aktueller.gibRechts() == null) {
                aktueller.setzeRechts(neuerKnoten);
                break;
            }
        }
    }

    public static void besuchePostOrder(Knoten knoten) {
        // Sonderfall leerer (Teil-)Baum
        if (knoten == null) {
            System.out.println("Leerer Baum");
        } else {
            // Linker
            if (knoten.gibLinks() != null) {
                besuchePostOrder(knoten.gibLinks());
            }
            // Rechter
            if (knoten.gibRechts() != null) {
                besuchePostOrder(knoten.gibRechts());
            }
            System.out.println(knoten.gibZahl());
        }
    }
}

```

```

public int berechneSumme(Knoten knoten) {
    int ergebnis = 0;

    // Sonderfall: leerer Unterbaum
    if (knoten == null) {
        return 0;
    }
    // linker
    if (knoten.gibLinks() != null) {
        ergebnis = ergebnis + berechneSumme(knoten.gibLinks());
    }
    // rechter
    if (knoten.gibRechts() != null) {
        ergebnis = ergebnis + berechneSumme(knoten.gibRechts());
    }
    // Wurzel
    ergebnis = ergebnis + knoten.gibZahl();
    return ergebnis;
}

public boolean istSortiert(Knoten knoten) {
    // Baum leer
    if (knoten == null) {
        return true;
    }

    // linker Nachfolger nicht okay
    if (knoten.gibLinks() != null && knoten.gibLinks().gibZahl() > knoten.gibZahl()) {
        return false;
    }

    // rechter Nachfolger nicht okay
    if (knoten.gibRechts() != null && knoten.gibRechts().gibZahl() <= knoten.gibZahl())
→ {
        return false;
    }
    // sonst prüfe Teilbaeume
    return (istSortiert(knoten.gibRechts()) && istSortiert(knoten.gibLinks()));
}

public static void main(String[] args) {
    BinBaum baum = new BinBaum();

    baum.fügeEin(5);
    baum.fügeEin(7);
    baum.fügeEin(4);
    baum.fügeEin(11);
    baum.fügeEin(6);
    baum.fügeEin(2);

    besuchePostOrder(baum.wurzel);
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java)

```
public class Knoten {
    private int zahl;
    private Knoten links = null;
    private Knoten rechts = null;

    public Knoten() {
    }

    public Knoten(int zahl) {
        this.zahl = zahl;
    }

    public void setzeZahl(int zahl) {
        this.zahl = zahl;
    }

    public int gibZahl() {
        return zahl;
    }

    public void setzeLinks(Knoten k) {
        links = k;
    }

    public Knoten gibLinks() {
        return links;
    }

    public void setzeRechts(Knoten k) {
        rechts = k;
    }

    public Knoten gibRechts() {
        return rechts;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/Knoten.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2003/herbst/Knoten.java)