# Algorithmen & Datenstrukturen

**Blatt 6**

Dr. Matthias Thimm
Tina Walber, Leon Kastler, Martin Leinberger und Maximilian Strauch
Fachbereich Informatik, Universität Koblenz-Landau

7. Dezember 2013

# 1 O-Notation (3 Punkte)

Beweisen oder widerlegen Sie folgende Aussage:

1. $g(n) \in \Theta(f(n))$ gilt genau dann wenn $\Theta(g(n)) = \Theta(f(n))$

## 2 Rekursionsbaum erstellen (4 Punkte)

Gegeben ist folgende Rekursionsgleichung:

$$T(a,b) = \begin{cases} \Theta(1) & \text{für } a \leq 1 \text{ oder } b \leq 1 \\ 2T(\frac{a}{2}, \frac{b}{2}) + \Theta(ab) & \text{sonst} \end{cases}$$

a) **Rekursionsbaum skizzieren (1 Punkt)**
Skizzieren sie für $T(a,b)$ die ersten drei Ebenen des Rekursionsbaums. Bestimmen Sie mit Hilfe des Rekursionsbaums folgende Informationen:

i) Welche Tiefe hat der Rekursionsbaum in Abhängigkeit von $a$ und $b$?

ii) Wieviele Knoten hat der Rekursionsbaum auf der Ebene $i$?

iii) Wieviele Blattknoten hat der Rekursionsbaum in Abhängigkeit von $a$ und $b$?

b) **Summenformel aufstellen (2 Punkt)**
Benutzen Sie die Informationen aus der vorherigen Teilaufgabe um die Rekursionsgleichung $T(a,b)$ mit Hilfe einer Summenformel auszudrücken.

c) **Summenformel lösen (1 Punkt)**
Lösen Sie die Summenformeln von $T(a,b)$ aus der vorherigen Teilaufgabe. Welche Aufwandsklassen ergeben sich?

### Hinweis:

Bei der Aufgabe muss eine Fallunterscheidung vorgenommen werden, je nachdem wie groß $a$ und $b$ sind.

# 3 Binäre Suchbäume (4 Punkte)

a) Fügen Sie die Zahlen der unten stehenden Folge der Reihe nach in einen zu Anfang leeren binären Suchbaum ein. Zeichen Sie den Zustand des Baumes nachdem alle Zahlen eingefügt wurden. (1 Punkt)
$[4, 15, 9, 17, 18, 2, 5, 20]$

b) Schreiben Sie die Reihenfolge auf, in der die Zahlen in Ihrem Baum behandelt werden, wenn sie Ihn mit den folgenden Traversierungsansätzen durchlaufen:

- Preorder
- Inorder
- Postorder
- Levelorder

(2 Punkte)

c) Wie viele Schritte müssen maximal ausgeführt werden, um ein bestimmtes Element in Ihrem Baum zu finden? Was wäre der maximale Aufwand, wenn der Baum ausgeglichen wäre? (1 Punkt)

# 4 Programmieraufgabe: Traversieren (9 Punkte)

Im Folgenden sollen binäre Bäume dafür genutzt werden, um mathematische Terme zu repräsentieren und deren Wert zu berechnen. Gegeben ist ein Term in der so genannten Polnischen Notation[1]. Operationen werden in der Form `Operator Operand1 Operand2` aufgeschrieben. Das bedeutet, dass statt der üblichen Infixnotation[2] `(1 + 2)`, die Polnische Notation folgende Schreibweise vorsieht: `+ 1 2`.

Für unseren Anwendungsfall existieren zwei unterschiedliche Elemente eines Terms, mathematische Operatoren und konkrete Werte. Wir verwenden die mathematischen Operatoren $+$, $-$, $*$ und $/$. Diese verrechnen immer zwei Unterterme miteinander. Unterterme sind entweder ein konkreter, ganzzahliger Wert oder ein Operator mit ebenfalls zwei Untertermen. Aus den einzelnen Operatoren können kompliziertere Terme zusammengebaut werden, die rekursiv ausgewertet werden können.

Wenn man Terme mit Hilfe von binären Bäumen repräsentiert, dann enthält jeder Knoten entweder einen mathematischen Operator oder einen konkreten Wert. Ein Knoten der einen mathematischen Operator enthält, hat immer einen linken und rechten Nachfolger. Nur Knoten mit konkreten Werten sind Blattknoten. Mathematische Operatoren werden ausgewertet, indem der linke und rechte Teilbaum ausgewertet werden und deren Ergebnisse entsprechend verrechnet werden.

a) **Beispiel nachvollziehen (2 Punkte)** Vollziehen sie folgendes Beispiel:

```
+ / + 3 1 2 * - 5 4 7
```

i) Zeichnen Sie zu dem gegebenen Beispiel den zugehörigen binären Baum auf. (1 Punkt)

ii) Rechnen Sie das Ergebnis aus, indem sie mit der Wurzel anfangen und sich in die Äste des Baums vorarbeiten. (1 Punkt)

b) **Baum aufbauen (2 Punkt)**
Implementieren Sie die Methode `Node createTree(String term)` des Interfaces `TreeCreator`. Der übergebene String entspricht dem oben beschriebenen Aufbau und repräsentiert einen Term in Polnischer Notation. Die einzelnen Elemente werden jeweils durch Leerzeichen getrennt.

**Hinweis:** Es empfiehlt sich zuerst den String in einzelne Elemente zu zerlegen bevor aus diesen dann der Baum konstruiert wird.

c) **Baum traversieren (3 Punkte)**

Im Folgenden wird der oben erstellte Baum auf verschiedene Weisen traversiert. (i) Implementieren Sie die Methode `String printTree(Node root)` des Interfaces `TreePrinter`. Dieser Methode wird der Wurzelknoten eines Binärbaums übergeben

[1] http://de.wikipedia.org/wiki/Polnische_Notation
[2] http://de.wikipedia.org/wiki/Infixnotation

und liefert einen String zurück, der den Term in Infixnotation repräsentiert, inklusive korrekter Klammerung. Aus der Schreibweise `+ 3 + 2 1` wird beispielsweise `(3 + (2 + 1))`. (1 Punkt)
(ii) Danach implementieren Sie die Methode `int calculateTree(Node root)` des Interfaces `TreeCalculator`. Diese Methode wird der Wurzelknoten eines Binärbaums übergeben und liefert das Ergebnis des Terms, der durch den Baum repräsentiert wird. Beispiel: durch die Auswertung des Terms `+ 3 + 2 1` erhält man das Ergebnis `6`. (1 Punkt)

d) **Lösung überprüfen (2 Punkt)** Erstellen Sie einen JUnit[3] Test Case, um das Beispiel aus Aufgabe a mit Ihrem Programm zu berechnen und somit Ihre Implementation zu überprüfen.

[3] http://de.wikipedia.org/wiki/JUnit