

## Datenstrukturen und Algorithmen – Informatik I

### 2. Übung

Abgabe der Lösungen: 10. Mai 2004

---

**Hinweis:** Die Implementierungsaufgaben sind in Java oder C/C++ zu lösen. Der Quellcode sowie ein dokumentierter Beispiellauf sind beim Betreuer der Kleingruppe abzugeben.

#### Aufgabe 7: Komplexitäten

2 Punkte

Zeigen Sie, daß für beliebige Funktionen  $f, g: \mathbb{N} \rightarrow \mathbb{R}_+$  gilt:

$$g \in O(f) \iff f \in \Omega(g).$$

#### Aufgabe 8: Oh Oh Oh

2+2+2+2+1+1 Punkte

Beantworten Sie folgende Fragen und beweisen Sie jeweils Ihre Behauptung.

- Es sei  $g(n) := 4711n^2 - 42n + 5$  und  $f(n) := n^2$ . Gilt  $g \in \Theta(f)$ ?
- Es sei  $g(n) := \lceil \sqrt[3]{n} \rceil$  und  $f(n) := \lceil \sqrt{n} \rceil$ . Gilt  $g \in O(f)$ ? Gilt  $g \in \Omega(f)$ ?
- Es sei  $g(n) := \sum_{j=0}^n (n-j)$  und  $f(n) := n^2$ . Gilt  $g \in O(f)$ ?
- Es sei  $g(n) := \sum_{j=0}^n 2^j$  und  $f(n) := 3^n$ . Gilt  $g \in O(f)$ ? Gilt  $g \in \Theta(f)$ ?
- Es sei  $g(n) := \sum_{j=1}^n i \cdot 2^i$  und  $f(n) := n^2 \cdot 2^n$ . Gilt  $g \in O(f)$ ?
- Es sei  $g(n) := \lceil \log n \rceil$  und  $f(n) := \lceil \sqrt{n} \rceil$ . Gilt  $g \in O(f)$ ?

#### Aufgabe 9: Implementierung: Binärer Suchbaum

3+4+1+6 Punkte

Unter einem binären Suchbaum verstehen wir einen Baum vom Grad zwei mit folgenden Eigenschaften:

- Für die Werte der Knoten (Schlüsselemente) ist eine totale Ordnung definiert.
- Der Schlüssel eines Knotens  $n$  ist größer oder gleich seinem linken Nachfolger  $n_1$  und kleiner oder gleich seinem rechten Nachfolger  $n_2$ .

1. Implementieren Sie die Methoden der folgenden Klasse in Java:

```
class BinaryTree {
    BinaryTree() {} // leeren Baum initialisieren
    void Insert(int item) {} // Element einfügen
    boolean Search(int item) {} // feststellen ob Element im Baum ist
    void Print() {} // Elemente in aufsteigender Reihenfolge ausgeben
}
```

2. In einen anfangs leeren Suchbaum werden  $n$  zufällige Zahlen (in zufälliger Reihenfolge) eingefügt. Welche Zeitkomplexitäten haben die Funktionen Insert, Search und Print auf dem so entstandenen Baum im *best-*, *average-* und *worst-case*?

3. Verwenden Sie den Suchbaum um die unten angegebenen Beispieldaten zu sortieren. Welche Laufzeitkomplexität hat dieses Sortierverfahren im *best-*, *average-* und *worst-case*?

**Beispieldaten für die Aufgabe 9:**

18   -5   -8   -3   121   -30   100   -89   20   -100   10   105   -34   2  
-105   200   300   -611   500   -222   -299   -495   -125   501   -798   -30   450   13

**Aufgabe 10: Implementierung: McCarthy-Funktion**

**1+1+1+1 Punkte**

Die McCarthy-Funktion ist folgendermaßen definiert:

$$M(n) = \begin{cases} n - 10 & \text{if } n > 100 \\ M(M(n + 11)) & \text{if } n \leq 100 \end{cases}$$

1. Implementieren Sie eine *rekursive* Java Methode, die McCarthy-Zahlen gemäß obiger Definition berechnet.
2. Erweitern Sie die Implementierung um Memoization. Implementieren Sie dies auf zwei verschiedene Arten: a) der Lookup erfolgt in der aufgerufenen Funktion und b) der Lookup erfolgt vor dem Funktionsaufruf. Zählen Sie in beiden Fällen die Anzahl der Funktionsaufrufe und vergleichen Sie diese.
3. Wie groß sind die Laufzeit- und Speicherplatzkomplexitäten Ihrer beiden Methoden?