8. AuD Tafelübung T-C3

Simon Ruderich

15. Dezember 2010

Rekursion

Fragen zur Rekursion?

EBNF

Produktion $\rightarrow$ EBNF

A $\rightarrow$ T | A + T | A - T

A ::= T ('+' T)* ('-' T)*

EBNF

Produktion $\rightarrow$ EBNF

A $\rightarrow$ T | A + T | A - T

A ::= T ('+' T)* ('-' T)*

Nicht korrekt!

z. B. T + T - T + T nicht möglich

Vollständige Induktion

Vollständige Induktion

Induktionsanfang Beweise die Behauptung für ein n_0, meistens $n_0 = 1$

Induktionsvoraussetzung Man nimmt an, die Behauptung gilt für ein $n > n_0$

Induktionsschritt Man zeigt, dass die Annahme für $n + 1$ unter Verwendung der Voraussetzung gilt.
$\Rightarrow$ Behauptung gilt für alle $n \geq n_0$

Vollständige Induktion

„Kleiner“ Gauß

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2} \qquad \forall n \in \mathbb{N}$$

Beispiel

$$\sum_{k=1}^{n} (2k-1)^2 = \frac{n \cdot (2n-1) \cdot (2n+1)}{3} \qquad \forall n \in \mathbb{N}$$

Übungsblatt

Wichtig

Aufgabe *mit* Induktion lösen!

$\mathcal{O}$-Kalkül

Definition

$\mathcal{O}(f(n)) = \{g(n) | \exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0 : 0 \leq g(n) \leq c \cdot f(n)\}$

$\mathcal{O}$-Kalkül

- Ziel: Aufwandsabschätzung von Algorithmen
- Menge $\mathcal{O}(f(n))$ enthält alle Funktionen $g(n)$, die höchstens genauso schnell wachsen, wie $f(n)$
- Abkürzung: $\mathcal{O}(f(n)) =: \mathcal{O}(f)$
- erlaubt den Vergleich der Laufzeit von Algorithmen

$\mathcal{O}$-Kalkül

Verschiedene Aufwandsklassen

- $\mathcal{O}(\log n)$, z. B. binäre Suche (Ratespiel)
- $\mathcal{O}(\sqrt{n})$
- $\mathcal{O}(n)$, z. B. Maximumssuche
- $\mathcal{O}(n \cdot \log n)$, z. B. Merge-Sort
- $\mathcal{O}(n^2)$, z. B. Insertion-Sort
- $\mathcal{O}(2^n)$, z. B. Überdeckungsproblem (GTI)
- $\mathcal{O}(n!)$
- ...

$\mathcal{O}$-Kalkül

Rechenregeln

- $c \cdot \mathcal{O}(f) = \mathcal{O}(c \cdot f) = \mathcal{O}(f)$
  Konstante Faktoren können vernachlässigt werden
- $c \pm \mathcal{O}(f) = \mathcal{O}(c \pm f) = \mathcal{O}(f)$
  Konstanten können vernachlässigt werden
- $\mathcal{O}(f + g) = \mathcal{O}(\max(f, g))$
- $\mathcal{O}(f) \cdot \mathcal{O}(g) = \mathcal{O}(f \cdot g)$
- $\mathcal{O}(\log_2 n) = \mathcal{O}(\log_3 n) = \mathcal{O}(\log n)$
  Basis des Logarithmus kann vernachlässigt werden

$\mathcal{O}$-Kalkül Beispiele

Beispiele

- $\mathcal{O}(7n + 4) = ?$
- $\mathcal{O}(n(n-1)) = ?$
- $\mathcal{O}(10 \cdot \log_{24} n + n \cdot \log_2 n) = ?$
- $\mathcal{O}(500n + 100n^{1,5} + 50n \log_{10}(n)) = ?$
- $\mathcal{O}\left(\sum_{i=1}^{n} i\right) = ?$

$\mathcal{O}$-Kalkül Beispiele

Beispiele

- $\mathcal{O}(7n+4) = \mathcal{O}(7n) + \mathcal{O}(4) = \mathcal{O}(n)$
- $\mathcal{O}(n(n-1)) = ?$
- $\mathcal{O}(10 \cdot \log_{24} n + n \cdot \log_2 n) = ?$
- $\mathcal{O}(500n + 100n^{1,5} + 50n \log_{10}(n)) = ?$
- $\mathcal{O}\left(\sum_{i=1}^{n} i\right) = ?$

$\mathcal{O}$-Kalkül Beispiele

Beispiele

- $\mathcal{O}(7n+4) = \mathcal{O}(7n) + \mathcal{O}(4) = \mathcal{O}(n)$
- $\mathcal{O}(n(n-1)) = \mathcal{O}(n^2 - n) = \mathcal{O}(\max(n^2, n)) = \mathcal{O}(n^2)$
- $\mathcal{O}(10 \cdot \log_{24} n + n \cdot \log_2 n) = ?$
- $\mathcal{O}(500n + 100n^{1,5} + 50n \log_{10}(n)) = ?$
- $\mathcal{O}\left(\sum_{i=1}^{n} i\right) = ?$

$\mathcal{O}$-Kalkül Beispiele

Beispiele

- $\mathcal{O}(7n+4) = \mathcal{O}(7n) + \mathcal{O}(4) = \mathcal{O}(n)$
- $\mathcal{O}(n(n-1)) = \mathcal{O}(n^2 - n) = \mathcal{O}(\max(n^2, n)) = \mathcal{O}(n^2)$
- $\mathcal{O}(10 \cdot \log_{24} n + n \cdot \log_2 n) = \mathcal{O}(n \cdot \log n)$
- $\mathcal{O}(500n + 100n^{1,5} + 50n \log_{10}(n)) = ?$
- $\mathcal{O}\left(\sum_{i=1}^{n} i\right) = ?$

$\mathcal{O}$-Kalkül Beispiele

Beispiele

- $\mathcal{O}(7n+4) = \mathcal{O}(7n) + \mathcal{O}(4) = \mathcal{O}(n)$
- $\mathcal{O}(n(n-1)) = \mathcal{O}(n^2 - n) = \mathcal{O}(\max(n^2, n)) = \mathcal{O}(n^2)$
- $\mathcal{O}(10 \cdot \log_{24} n + n \cdot \log_2 n) = \mathcal{O}(n \cdot \log n)$
- $\mathcal{O}(500n + 100n^{1,5} + 50n \log_{10}(n)) = \mathcal{O}(n^{1,5})$
- $\mathcal{O}\left(\sum_{i=1}^{n} i\right) = ?$

$\mathcal{O}$-Kalkül Beispiele

Beispiele

- $\mathcal{O}(7n+4) = \mathcal{O}(7n) + \mathcal{O}(4) = \mathcal{O}(n)$
- $\mathcal{O}(n(n-1)) = \mathcal{O}(n^2 - n) = \mathcal{O}(\max(n^2, n)) = \mathcal{O}(n^2)$
- $\mathcal{O}(10 \cdot \log_{24} n + n \cdot \log_2 n) = \mathcal{O}(n \cdot \log n)$
- $\mathcal{O}(500n + 100n^{1,5} + 50n \log_{10}(n)) = \mathcal{O}(n^{1,5})$
- $\mathcal{O}\left(\sum_{i=1}^{n} i\right) = \mathcal{O}\left(\frac{n(n+1)}{2}\right) = \mathcal{O}(n^2)$

Weitere Landau-Symbole

Weitere Landau-Symbole

- $g(n) \in \Omega(f(n))$, $g(n)$ wächst mindestens so schnell
- $g(n) \in \Theta(f(n))$, $g(n)$ wächst genau so schnell

$\mathcal{O}$-Kalkül Beispiele

Beispiel

```
public void stuff(int n) {
    int[][] matrix = new int[n][n];
    for (int i = 0; i < 10000; i++) {
        matrix[0][0] = i;
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < 150; j++) {
            matrix[i][j] = Math.random();
        }
    }
}
```

$\mathcal{O}$-Kalkül Beispiele

Beispiel

```
public void stuff(int n) {
    int[][] matrix = new int[n][n];
    for (int i = 0; i < 10000; i++) {
        matrix[0][0] = i;
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < 150; j++) {
            matrix[i][j] = Math.random();
        }
    }
}
```

Aufwand

$\mathcal{O}(10000 + n \cdot 150) = \mathcal{O}(n)$

$\mathcal{O}$-Kalkül Beispiele

Beispiel

```
public int count(int[] array) {
    int result = 0;
    for (int i = 1; i < array.length; i *= 2) {
        result += array[i];
    }
    return result;
}
```

$\mathcal{O}$-Kalkül Beispiele

Beispiel

```
public int count(int[] array) {
    int result = 0;
    for (int i = 1; i < array.length; i *= 2) {
        result += array[i];
    }
    return result;
}
```

Aufwand

$\mathcal{O}(\log n)$

$\mathcal{O}$-Kalkül Beispiele

Beispiel

```
int k = 42 * 1000;
public int doThis(int n) {
    for (int i = 0; i < k; i++) {
        doSomeStuff(k, i);
    }
    return n * k;
}
```

$\mathcal{O}$-Kalkül Beispiele

Beispiel

```
int k = 42 * 1000;
public int doThis(int n) {
    for (int i = 0; i < k; i++) {
        doSomeStuff(k, i);
    }
    return n * k;
}
```

Aufwand

$\mathcal{O}(1)$

$\mathcal{O}$-Kalkül Beispiele

Beispiel

```
public void uselessLoop(int n) {
    for (int i = 1; i < n; i++) {
        doMyStuff();
    }
}
public void anotherLoop(int n) {
    for (int i = 1; i < n * n; i *= 2) {
        uselessLoop(i);
    }
}
```

$\mathcal{O}$-Kalkül Beispiele

Beispiel

```
public void uselessLoop(int n) {
    for (int i = 1; i < n; i++) {
        doMyStuff();
    }
}
public void anotherLoop(int n) {
    for (int i = 1; i < n * n; i *= 2) {
        uselessLoop(i);
    }
}
```

$$\sum_{k=0}^{\mathrm{ld}(n^2)} 2^k = 2n^2 - 1 \in \mathcal{O}(n^2)$$

$\mathcal{O}$-Kalkül Beispiele

Beispiel

```
public void uselessThing(int n) {
    for (int i = n; i > 0; i /= 2) {
        for (int j = 1; j < n; j *= 2) {
            for (int k = 0; k < n; k += 2) {
                System.out.println("...");
            }
        }
    }
}
```

$\mathcal{O}$-Kalkül Beispiele

Beispiel

```
public void uselessThing(int n) {
    for (int i = n; i > 0; i /= 2) {
        for (int j = 1; j < n; j *= 2) {
            for (int k = 0; k < n; k += 2) {
                System.out.println("...");
            }
        }
    }
}
```

Aufwand

$\mathcal{O}(\frac{n}{2} \cdot \log n \cdot \log n) = \mathcal{O}(n \cdot (\log n)^2) = \mathcal{O}(n \cdot \log^2 n)$

$\mathcal{O}$-Kalkül Beispiele

Beispiel

```
public void evilLoop(int n) {
    for (i = 0; i < n * n; i++) {
        simpleLoop(i);
    }
}
public void simpleLoop(int i) {
    for (j = 0; j < i; j++) {
        // O(1)
    }
}
```

$\mathcal{O}$-Kalkül Beispiele

Beispiel

```
public void evilLoop(int n) {
    for (i = 0; i < n * n; i++) {
        simpleLoop(i);
    }
}
public void simpleLoop(int i) {
    for (j = 0; j < i; j++) {
        // O(1)
    }
}
```

$$\sum_{i=0}^{n^2} i = \frac{1}{2} n^2 (n^2 + 1) \in \mathcal{O}(n^4)$$

Ausnahmen („Exceptions“)

Exceptions

- Exceptions melden einen aufgetreten Fehler
- zeigen genau an wo der Fehler aufgetreten ist (Stacktrace)
- werden automatisch bei Fehlern von der JVM geworfen
- Exceptions sind Objekte, Unterklassen von `Throwable`:

  Error schwerwiegender Fehler, meist nicht behandelbar
  z. B. `OutOfMemoryError`, `StackOverflowError`, ...

  Exception „normale“ Exception, behandelbar
  z. B. `IndexOutOfBoundsException`, `NumberFormatException`, ...

Exceptions

Exceptions deklarieren

- jede Methode die eine Exception wirft muss diese mit `throws` deklarieren
- aufrufende Methode muss diese Exception behandeln oder selbst wieder deklarieren
- falls `main()` eine Exception wirft bricht das Programm ab

Beispiel

```
public void test() throws Exception {
    throw new Exception();
}
```

Exceptions

Beispiel

```
public class Test {
    public static void main(String[] args)
                                        throws Exception {
        Exception up = new Exception();
        throw up;
    }
}

Exception in thread "main" java.lang.Exception
        at Test.main(Test.java:3)
```

Exceptions

Beispiel

```
import java.io.*;
public String readFromFile(String path)
                                    throws IOException {
    if (path == null) {
        throw new IOException();
    }
    return read(path);
}
// Exception nach oben "weiterreichen"
public void doStuff() throws IOException {
    String content = readFromFile("AuD.java");
    // ...
}
```

Exceptions behandeln

Exceptions behandeln

- Exceptions werden „gefangen“
- kritischer Code kommt in einen `try`-Block
- Fehlerbehandlung kommt in einen `catch`-Block
- Code der *immer* ausgeführt werden soll steht im `finally`-Block
- Exceptions einer Methoden stehen in der Java-API

Exceptions behandeln

Beispiel

```
public void doStuff() {
    try {
        String content = readFromFile("AuD.java");
    } catch (IOException e) {
        System.out.println("Can't open file: "
            + e.getMessage());
    } finally {
        // ...
    }
}
```

- `getMessage()`-Methode liefert weitere Fehlerdetails
- `printStackTrace()` gibt Stacktrace aus

RuntimeExceptions

RuntimeExceptions

- `RuntimeException`s treten zur Laufzeit auf
- Unterklasse von `Exception`
- müssen nicht mit `throws` deklariert werden
- können aber normal gefangen werden
- z. B. `ArrayIndexOutOfBoundsException`

RuntimeExceptions

Beispiel

```
public void doException() {
    int[] array = new int[10];
    // IndexOutOfBoundsException ...
    System.out.println(array[10]);
}
public void doNoException() {
    int[] array = new int[10];
    try {
        System.out.println(array[10]);
    } catch (IndexOutOfBoundsException e) {
        // who cares - schlechter Stil!
    }
}
```

Eigene Exceptions

Eigene Exceptions

- Konstruktor der Exceptions nimmt String, eigener Text möglich
- eigene Exceptions durch Vererbung von Exception-Klassen (meist `Exception`)

Beispiel

```
import java.io.*;
public void testRuntime() {
    throw new RuntimeException("Meine Exception");
}
public void testIO() throws IOException {
    throw new IOException("Meine IO Exception");
}
```

Übungsaufgabe

Zur Aufgabe . . .

$a \cdot b$ auf Überlauf prüfen:

```
if ((a * b) / b != a) { ... }
```

Fragen

Fragen?