Technische Universität Braunschweig

Institute of Operating Systems and Computer Networks

Algorithmen und Datenstrukturen
Große Übung #1

Christian Rieck, Arne Schmidt

26.10.2017

Organisatorisches

Homepage

Aktuelle Informationen, Hausaufgaben, Slides auf:

`https://www.ibr.cs.tu-bs.de/courses/ws1718/aud/webpages/index.html`

Anmeldung zu kleinen Übungen über:
`https://www.ibr.cs.tu-bs.de/courses/ws1718/aud/`

Wir nutzen kein StudIP!

Anmeldung

Group	Time (biweekly)	Place	Tutor	Participants
1	Montag, 08:00 - 09:30	IZ 305	Moritz Pfister	TBA
2	Dienstag, 08:00 - 09:30	IZ 305	Moritz Pfister	TBA
3	Dienstag, 08:00 - 09:30	IZ 358	Sabrina Hugo	TBA
4	Dienstag, 13:15 - 14:45	IZ 358	Mai Hellmann	TBA
5	Dienstag, 16:45 - 18:15	IZ 161	Mai Hellmann	TBA
6	Mittwoch, 08:00 - 09:30	IZ 358	Sabrina Hugo	TBA
7	Mittwoch, 08:00 - 09:30	IZ 161	Jakob Keller	TBA
8	Mittwoch, 13:15 - 14:45	IZ 305	Antje Mönch	TBA
9	Mittwoch, 13:15 - 14:45	IZ 161	Eva Bolle	TBA
10	Mittwoch, 15:00 - 16:30	IZ 160	Jakob Keller	TBA
11	Mittwoch, 15:00 - 16:30	IZ 358	Dennis Luck	TBA
12	Mittwoch, 16:45 - 18:15	IZ 358	Dennis Luck	TBA
13	Mittwoch, 16:45 - 18:15	IZ 305	Alexander Hill	TBA
14	Donnerstag, 15:00 - 16:30	PK 3.1	Alexander Hill	TBA
15	Donnerstag, 16:45 - 18:15	IZ 161	Eva Bolle	TBA
16	Freitag, 11:30 - 13:00	PK 3.1	Maurice Semren	TBA
17	Freitag, 13:15 - 14:45	PK 3.1	Antje Mönch	TBA
18	Freitag, 13:15 - 14:45	IZ 305	Mohammad Mahhouk	TBA
19	Freitag, 15:00 - 16:30	IZ 305	Mohammad Mahhouk	TBA
20	Freitag, 15:00 - 16:30	IZ 358	Maurice Semren	TBA

Anmeldung bis nächsten Donnerstag 20 Uhr.

Semesterplan

Semesterplan AuD WS17/18

Woche (KW)	Vorlesung (Di.+Mi.)	Gr. Übung (Do.)	Kl. Übung (Mo.-Fr.)	HA Ausgabe (Mo.)	HA Abgabe (Mo. bis 11:30 Uhr)	HA Rückgabe (in kl. UE)
42	-,-					
43	0,1	1				
44	-,2			HA0a*, HA0b*		
45	3,4	2	1	HA1		HA0a
46	5,6		2			HA0b
47	7,8	3		HA2	HA1	
48	9,10		3			HA1
49	11,12	4		HA3	HA2	
50	13,-		4			HA2
51	14,15	5		HA4	HA3	
52 1	Brace yourself: Winter is here!					
2	16,17	VL 18**	5			HA3
3	19,20	6		HA5	HA4	
4	21,22	VL 23***	6			HA4
5	24,25	7****			HA5	
*: Mündliche Aufgaben, Bearbeitung in ersten kleinen Übungen; keine Bewertung						
**: Die Vorlesung vom 31.10.2017 wird am 11.01.2018 nachgeholt (Übungsslot).						
***: Die Vorlesung vom 13.12.2017 wird am 25.01.2018 nachgeholt (Übungsslot).						
****: HA-Blatt 5 wird besprochen. Rückgabe wird über die Mailingliste bekannt gegeben.						

Anmeldung zu den kleinen Übungen: **Mo., 23.10.17 (0:00 Uhr) - Do., 02.11.17 (20:00 Uhr)**

Große Übungen

- Aufarbeitung des Inhalts aus der Vorlesung
- Mehr interessanter Inhalt
- Beantwortung von Fragen
- Interaktionen!

Ihr könnt die Übung mitgestalten! Einfach eine Mail mit Fragen/Wünschen an Christian oder Arne. Wir versuchen dann, diese einfließen zu lassen.

Hausaufgaben

- 7 Blätter
  - 2 unbewertet (Blatt0a/Blatt0b)
  - 5 bewertet (Blatt1-5)
- Je 60 Punkte $\Rightarrow$ Insgesamt 300 Punkte
- Studienleistung: 50% aller Punkte, also 150 Punkte
  - Studienleistung ist **keine** Vorraussetzung, um an der Prüfung teilzunehmen
  - Studienleistung ist **eine** Vorraussetzung, um das Modul abzuschließen

Hausaufgaben – Pt. 2

- Zu späte Abgaben: 0 Punkte
- Falscher Abgabeschrank: 0 Punkte
- Mit Bleistift oder Rot geschriebene Teile werden nicht gewertet
- Zusammen arbeiten, **ABER:** einzeln aufschreiben und abgeben

Wichtig: Die Hausaufgaben dienen Euch (und nicht uns) zur Vorbereitung für die Klausur. Abschreiben bringt nichts, da es nicht Eure Fehler sind. Die eigenen Fehler passieren dann in der Klausur!

Hausaufgaben – Pt. 3

Der Hausaufgabenschrank

Klausur

Findet statt am 23.02.2018, zwischen 11:00 Uhr und 14:00 Uhr.
Nähere Information wie

- Raumaufteilung
- Beginn der Klausur

werden wenige Tage vorher bekanntgegeben.

Mailingliste

- Anmeldung über Homepage.
- Sämtliche Ankündigungen werden dort verkündet:
  - Raumänderungen,
  - Ausfälle,
  - etc.
- Bietet Möglichkeit Fragen zu stellen.
  - Diskussion mit anderen Teilnehmern $\Rightarrow$ Gegenseitige Hilfe
- Hiwis können über die Mailingliste erreicht werden.

Fragen!

Stellt Eure Fragen:

- über die Mailingliste
- in: Vorlesung, Gr. Übung, Kl. Übungen
- Euren Tutoren
- Sprechstunde von Christian Rieck (Di. 08:30 - 09:30 Uhr)
- Sprechstunde von Arne Schmidt (Mo. 09:30 - 10:30 Uhr)
- Per Mail an Christian oder Arne
- Sprechstunde von Prof. Sándor Fekete (Mi. 13:15 - 14:00 Uhr)

Fragen?

Problem

Allgemeine Fragestellung. Meistens formuliert mit:

Eingabe: Was ist gegeben?

Ausgabe: Was ist gesucht?

Lösung eines Problems:
Angabe eines Algorithmus.

Instanz

Eine Problemstellung mit konkreter Eingabe.

Lösung einer Instanz:
Angabe einer konkreten Ausgabe.

Ein Beispiel - Größter Gemeinsamer Teiler

Problem ggT:

Gegeben: Zwei Zahlen a, b.

Gesucht: Der größte gemeinsame Teiler q von zwei Zahlen a und b.

Lösung: Euklidischer Algorithmus (dazu gleich mehr)

Instanzen:

- $\text{ggT}(5, 102)$; Lösung: 1
- $\text{ggT}(8, 64)$; Lösung: 8

Beispiel 2 - Fibonacci-Zahlen

Definition der Fibonacci-Zahlen:
$F(0) := 0,\ F(1) := 1,\ F(n) := F(n-1) + F(n-2)$

Problem Fibonacci:

Gegeben: Eine Zahl n.
Gesucht: Die n-te Fibonacci-Zahl

Lösung: Rekursion aufschlüsseln. (Siehe Blatt0a)

Instanzen:

- $F(6)$; Lösung: 8
- $F(17)$; Lösung: 1597
- $F(200)$; Lösung: 280571172992510140037611932413038677189525

Algorithmus

- Beschreiben durch:
  - Prosatext
  - *Pseudocode*
- Ausführbarkeit
- Endlichkeit
- Endliche Ausführzeit
- Endlicher Speicherbedarf

Also nicht:

- „Gib 5 oder 1 aus.“
- „Sei $F(0) = 0$, $F(1) = 1$, $F(2) = 1$, $F(3) = 2$, $F(4) = 3$, $F(5) = 5$, $F(6) = 8, \ldots$“

Warum Pseudocode?

Euklidischer Algorithmus als Text:
Solange die Zahl b nicht 0 ist, wählen wir $h = a \mod b$, setzen a auf b und b auf h. Nachdem $b = 0$, geben wir a zurück

Euklidischer Algorithmus als Pseudocode:

function $\textsc{euklid}(a, b)$
  while $b \neq 0$ **do**
    $h \leftarrow a \bmod b$
    $a \leftarrow b$
    $b \leftarrow h$
  end while
  return a
end function

(Für modulo gilt $r = a \mod b$, sodass $a = q * b + r$ für ein $q \in \mathbb{Z}$ gilt mit $0 \leq r < b$.)

Pseudocode - Bausteine

Ausgaben/Rückgaben:

- **print** *a (Gibt a aus und beendet die Funktion nicht)*
- **return** *a (Gibt a zurück und beendet die Funktion)*

Pseudocode - Bausteine

Zuweisung:

- $a \leftarrow b$ *(Weist a den Wert b zu.)*
- $a \leftarrow b + c$ *(Weist a die Summe von b und c zu.)*
- $A \leftarrow B$ *(Weist der Menge A die Menge B zu.)*

Beispiel:

function $\textsc{Sum}(a, b, c)$
  $sum \leftarrow a + b + c$
  $A \leftarrow \{a, b, c, sum\}$
  return A
end function

Pseudocode - Bausteine

Schleifen:

- **for** a **in** start, ..., ende **do**
  ...
  end for
- **for** $a \leftarrow b$ **(down)to** c **do**
  ...
  end for
- **while** Bedingung **do**
  ...
  end while
- **repeat**
  ...
  until Bedingung

Beispiel:

```
function SomeFunction(a, b, c)
    for a ← 0 to b do
        d ← c
        repeat
            a ← a + b
            d ← d − 1
        until d = 0
    end for
    return a
end function
```

Pseudocode - Bausteine

Bedingte Anweisung:

if Bedingung **then**
    Aktionen, falls Bedingung wahr ist
else
    Aktionen, falls Bedingung falsch ist
end if

Beispiel:

function ABSOLUTE(a)
    if $a < 0$ **then**
        $a \leftarrow -a$
    end if
    return a
end function

Pseudocode - Bausteine

Methoden:

- Funktion-Körper:

  function NAME(Param1, Param2,...)

  ...

  end function

- Aufruf:

  NAME(a, b,...)

Beispiel:

function ABSOLUTE(a)
  if $a < 0$ **then**
    $a \leftarrow -a$
  end if
  return a
end function

function ABSOLUTEDIFF(a, b)
  return ABSOLUTE($a - b$)
end function

Pseudocode - Beispiel

Problem SUM:

Gegeben Zwei Zahlen s und e

Gesucht Die Summe von s bis e, oder 0 falls $s > e$.

```
function SUM(s, e)
    if s > e then
        return 0
    end if
    for i ← s + 1 to e do
        s ← s + i
    end for
    return s
end function
```

Instanz:
Berechne die Summe von 4 bis 8, also SUM(4,8)
Lösungsschritte:

- $4 > 8$? Nein.
- Betrachte For-Schleife nach jeder Iteration:

i	s
5	9
6	15
7	22
8	30

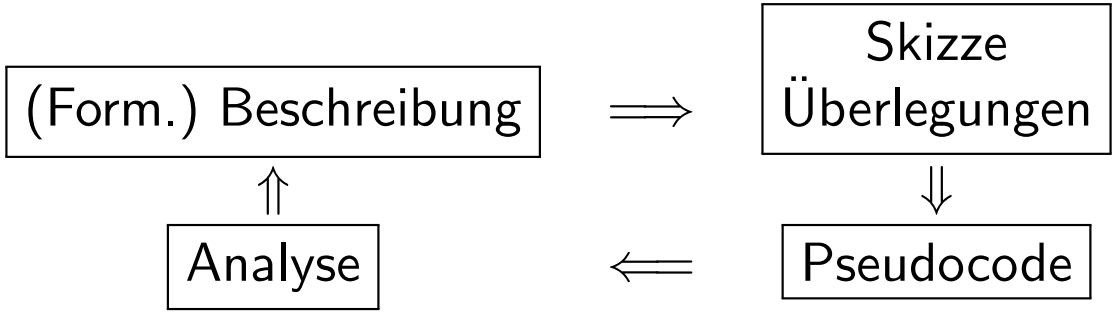
- Gib 30 zurück.

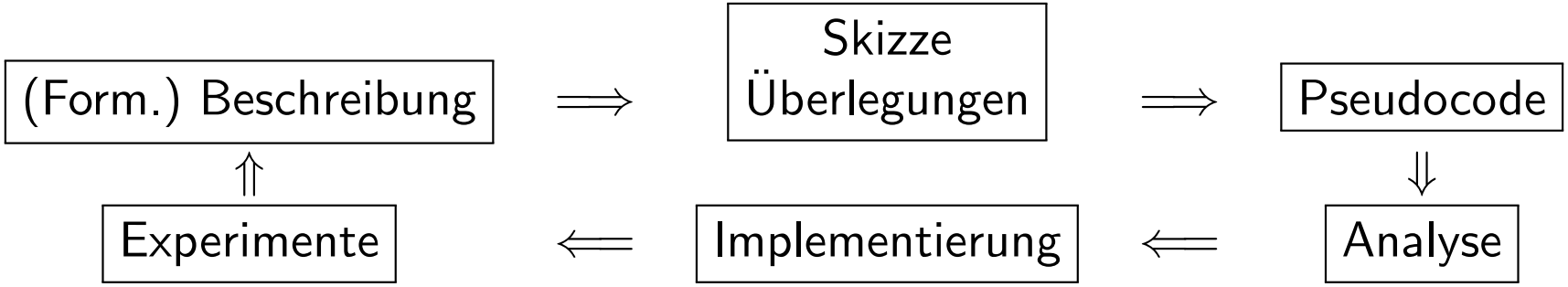
Pseudocode - Variationen

- Auf Deutsch
  - Wenn ... Dann ... Sonst ...
  - Solange ...
  - Für $i \leftarrow 1, \ldots, k$...
- Anstatt $\leftarrow$ schreibt man gelegentlich $=$
- Mischung aus Prosa und Pseudocode:

```
function ProsaCode(s)
    if (Es existiert k ∈ ℤ, sodass 0 < k < s) then
        Wähle h = 1
    else
        h = 2
    end if
    return s − h
end function
```

Algorithmenentwurf in Theorie und Praxis





Ein einfacher Algorithmus für den Logarithmus

Betrachten wir den Logarithmus:

Gegeben Zwei (ganzzahlige) Zahlen n und b

Gesucht Eine (nicht-ganzzahlige) Zahl x, sodass $n = b^x$. Oder kurz: $x = \log_b(n)$

Einfach ausgedrückt: *Wie oft muss man n durch b teilen, damit man auf 1 kommt?* Das ist einfach, wenn x ganzzahlig ist:

```
function LOG(n, b)
    log ← 0
    while n > 1 do
        Erhöhe log um 1
        n ← n/b
    end while
    return log
end function
```

Beispiel: LOG(64,2)

log	*n*
0	64
1	32
2	16
3	8
4	4
5	2
6	1

Ein Algorithmus für den Logarithmus

Was passiert, wenn x nicht ganzzahlig sein wird? Problem: Das Ergebnis kann irrational sein.$\Rightarrow$ Bestimme den Logarithmus auf k Stellen genau.
Exponentieren erlaubt? Wir beschränken uns auf das Exponentieren mit ganzen Zahlen!
Damit können wir folgendes tun:
Betrachte $10^k \log_b(n)$. Das gibt uns k Stellen mehr vor dem Dezimalkomma.
Die bekommen wir mit dem alten Algorithmus! Berechne also $\log_b(n^{(10^k)})$:

function $\textsc{Log}(n, b, k)$
  $n \leftarrow n^{(10^k)}$
  $log \leftarrow \textsc{Log}(n, b)$
  return $log \cdot 10^{-k}$
end function

Problem 1: Schon für kleine k treten große Zahlen auf.
Problem 2: Der Algorithmus benötigt exponentiell viele Schritte.

Mehr zu Logarithmen

Algorithmen:
Logarithmen lassen sich deutlich schneller präziser berechnen. Beispiele:

- Taylor-Entwicklung (gibt es in Analysis)
- Newton-Verfahren (gibt es in Numerik)

Diese Verfahren werden wir nicht brauchen.
Wir brauchen den Logarithmus beispielsweise

- für Laufzeiten, z.B. beim Sortieren (Kapitel 5)
- zum Bestimmen der Höhe eines Baumes (Kapitel 4)

Dazu sind folgende **Rechenregeln** hilfreich:

- $\log_c(a \cdot b) = \log_c(a) + \log_c(b)$
- $\log_c(a^b) = b \cdot \log_c(a)$
- $\log_c(a) = \frac{\log_b(a)}{\log_b(c)}$