Technische Universität Braunschweig



Algorithmen und Datenstrukturen
Große Übung 0

Ramin Kosfeld und Chek-Manh Loi

30.10.25

Programm heute

- Organisation
- Einstieg
  - Algorithmen und Pseudocode

Organisation

Formate



Tafelübungen im Hörsaal (mit allen)

- Ca. alle 2 Wochen
- Wiederholung und Vertiefung der Inhalte, Beispiele, Ergänzungen
- Fragen klären

Kleingruppen (Seminarräume)

- Alle 2 Wochen
- Betreut von Übungsleiter*in
- Besprechung von Hausaufgaben
- Fragen klären und Inhalte üben

Wie besteht man AuD?

Prüfungsleistung
→ Klausur

Studienleistung
→ Hausaufgaben

- Beide unabhängig voneinander
  (kann an Klausur teilnehmen ohne bestandene HA und andersherum)
- Beide bestanden: Veranstaltung geschafft :)

Klausur

- Voraussichtlich 18.02.26
- Dauer: 2 Stunden
- Inhalt: Prinzipiell alles aus VL und Übung
- *…mehr, wenn die Klausur näher rückt :)*



Technische Universität Braunschweig Wintersemester 2023/2024
Institut für Betriebssysteme und Rechnerverbund
Abteilung Algorithmik

Prof. Dr. Sándor P. Fekete
Ramin Kosfeld
Chek-Manh Loi

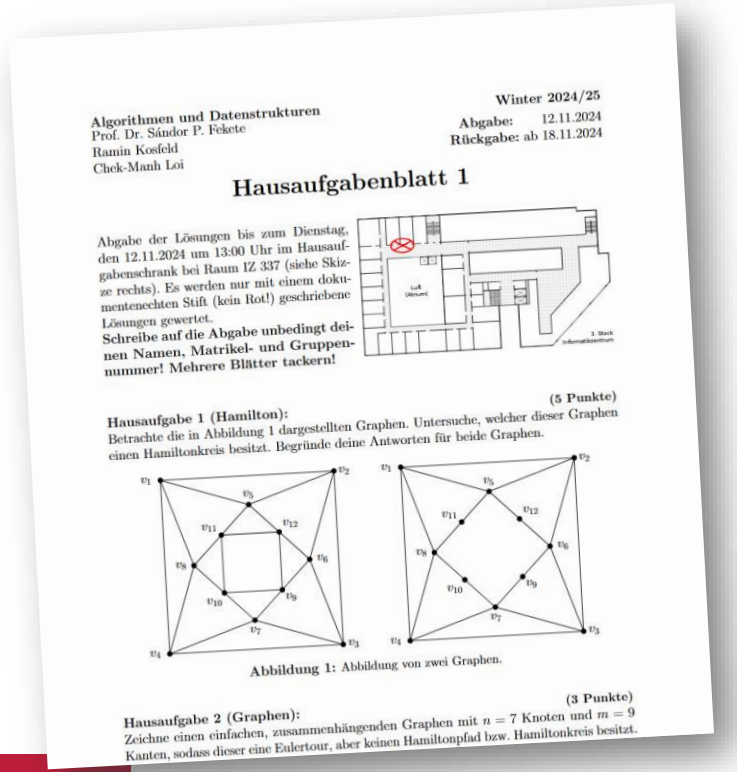
Klausur
Algorithmen und Datenstrukturen
13.02.2024

Muster

Name: .....
Vorname: .....
Matr.-Nr.: .....
Studiengang: .....
☐ Bachelor ☐ Master ☐ Andere

Klausurcode:
Dieser wird benötigt, um das Ergebnis der Klausur abzurufen.

Hausaufgaben



Algorithmen und Datenstrukturen
Prof. Dr. Sándor P. Fekete
Ramin Kosfeld
Chek-Manh Loi

Winter 2024/25
Abgabe: 12.11.2024
Rückgabe: ab 18.11.2024

Hausaufgabenblatt 1

Abgabe der Lösungen bis zum Dienstag, den 12.11.2024 um 13:00 Uhr im Hausaufgabenschrank bei Raum IZ 337 (siehe Skizze rechts). Es werden nur mit einem dokumentenechten Stift (kein Rot!) geschriebene Lösungen gewertet.
Schreibe auf die Abgabe unbedingt deinen Namen, Matrikel- und Gruppennummer! Mehrere Blätter tackern!

Hausaufgabe 1 (Hamilton): (5 Punkte)
Betrachte die in Abbildung 1 dargestellten Graphen. Untersuche, welcher dieser Graphen einen Hamiltonkreis besitzt. Begründe deine Antworten für beide Graphen.

Abbildung 1: Abbildung von zwei Graphen.

Hausaufgabe 2 (Graphen): (3 Punkte)
Zeichne einen einfachen, zusammenhängenden Graphen mit $n = 7$ Knoten und $m = 9$ Kanten, sodass dieser eine Eulertour, aber keinen Hamiltonpfad bzw. Hamiltonkreis besitzt.

Einzelabgaben

5 Zettel

Aufwand: Überschaubar

Neues Blatt: Alle 2 Wochen

Bearbeitungszeit: 2 Wochen

Pro Blatt: 20 Punkte

Unbenotet

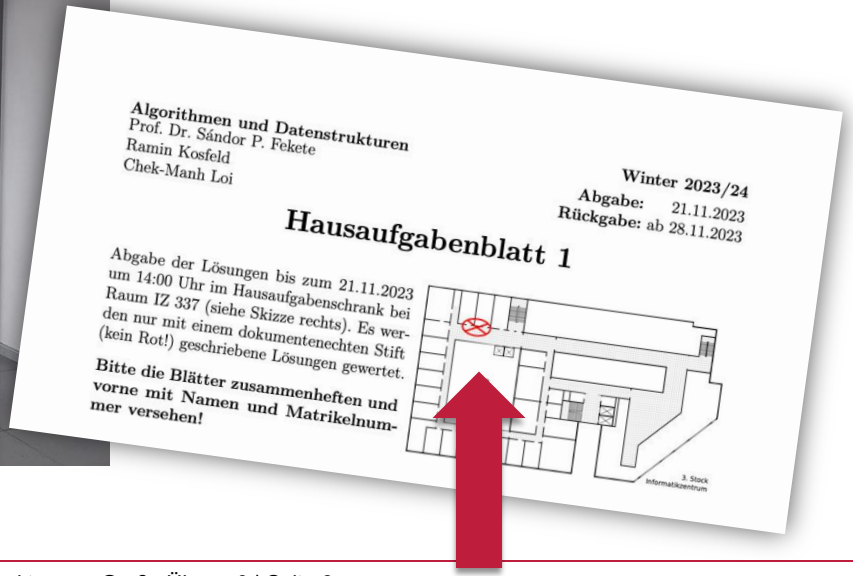
Benötigt: 50 Punkte
zum Bestehen der Studienleistung

Wie gebe ich meine Hausaufgaben ab?



Hausaufgabenschrank

- Abgabe in Papierform
- Einwurf in HA-Schrank, 3. Stock IZ
- Vorsicht: Bis Donnerstag 11:00 Uhr!
  - Danach: Korrektur, aber keine Punkte.



Algorithmen und Datenstrukturen
Prof. Dr. Sándor P. Fekete
Ramin Kosfeld
Chek-Manh Loi

Winter 2023/24
Abgabe: 21.11.2023
Rückgabe: ab 28.11.2023

Hausaufgabenblatt 1

Abgabe der Lösungen bis zum 21.11.2023 um 14:00 Uhr im Hausaufgabenschrank bei Raum IZ 337 (siehe Skizze rechts). Es werden nur mit einem dokumentenechten Stift (kein Rot!) geschriebene Lösungen gewertet.

Bitte die Blätter zusammenheften und vorne mit Namen und Matrikelnummer versehen!

Hausaufgaben und Präsenzblätter

5 verpflichtende Hausaufgabenblätter

- Studienleistung
- Rückgabe & Besprechung in den KÜ
- Sie sind korrigiert:
  Ihr seht eure Fehler und eure Punkte :)

6 freiwillige Präsenzblätter

- Müssen nicht abgegeben werden :)
- Gemeinsam bearbeiten in den KÜ
- Zusätzliche Vertiefung
- Prüfungsvorbereitung

Hausaufgaben

Wozu Hausaufgaben?
Die Hausaufgaben dienen euch zur Vorbereitung auf die Klausur.

- Ideale Nachbereitung der Vorlesungsinhalte
- Zeitersparnis bei der Prüfungsvorbereitung
- Direktes Feedback über euren Lernstand

- Zusammen überlegen ist gut, aber: *Einzeln* aufschreiben und abgeben. Sonst: **Plagiat** = keine Punkte.

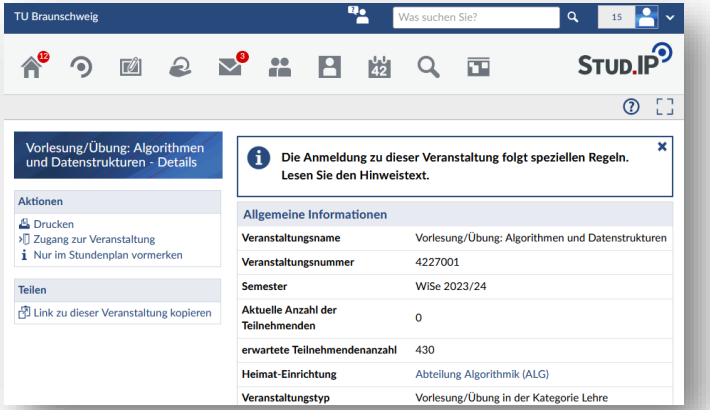
Hausaufgaben

- Alle wichtigen Infos stehen auch …
  - Teilweise auf den HA-Blättern selber
  - Der Website

  → „Hausaufgaben-Merkblatt“

- Bei Fragen / Problemen mit den HA und den Abgaben:
  - Schreibt *direkt* euren Tutor*innen eine Mail, die können oft helfen :)

Homepage und Anmeldung



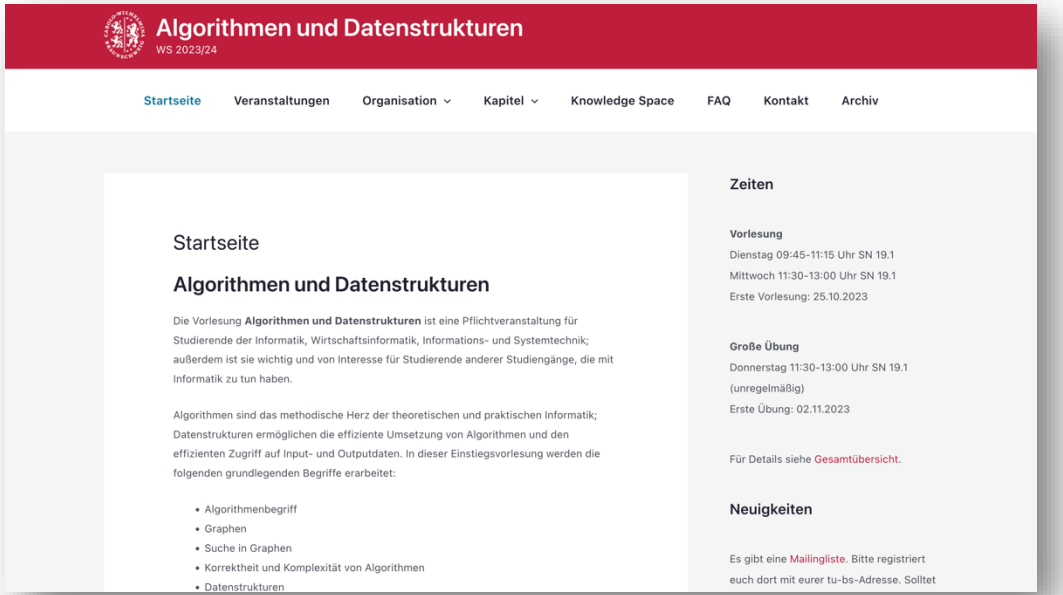
Stud.IP?



Homepage und Anmeldung

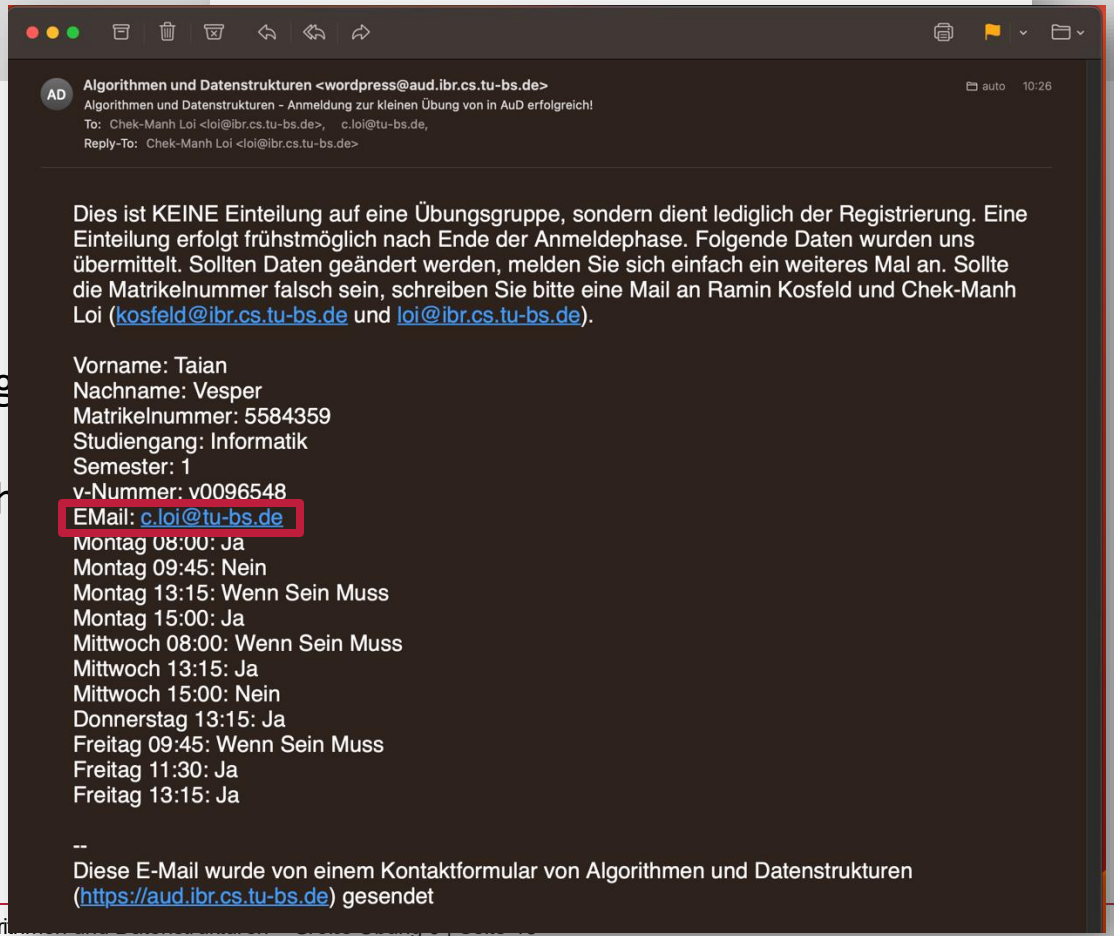
Unsere Kursseite: *aud.ibr.cs.tu-bs.de*

… Folien (ja, auch diese hier), Hausaufgaben, Altklausuren, **Übungsanmeldungen**:



Homepage und Anmeldung

- Anmeldung für die kleinen Übung
  auf der Kursseite
- Zuweisung erfolgt (voraussichtlich
  06.11.25



AD **Algorithmen und Datenstrukturen <wordpress@aud.ibr.cs.tu-bs.de>** auto 10:26
Algorithmen und Datenstrukturen - Anmeldung zur kleinen Übung von in AuD erfolgreich!
To: Chek-Manh Loi <loi@ibr.cs.tu-bs.de>, c.loi@tu-bs.de,
Reply-To: Chek-Manh Loi <loi@ibr.cs.tu-bs.de>

Dies ist KEINE Einteilung auf eine Übungsgruppe, sondern dient lediglich der Registrierung. Eine Einteilung erfolgt frühstmöglich nach Ende der Anmeldephase. Folgende Daten wurden uns übermittelt. Sollten Daten geändert werden, melden Sie sich einfach ein weiteres Mal an. Sollte die Matrikelnummer falsch sein, schreiben Sie bitte eine Mail an Ramin Kosfeld und Chek-Manh Loi (kosfeld@ibr.cs.tu-bs.de und loi@ibr.cs.tu-bs.de).

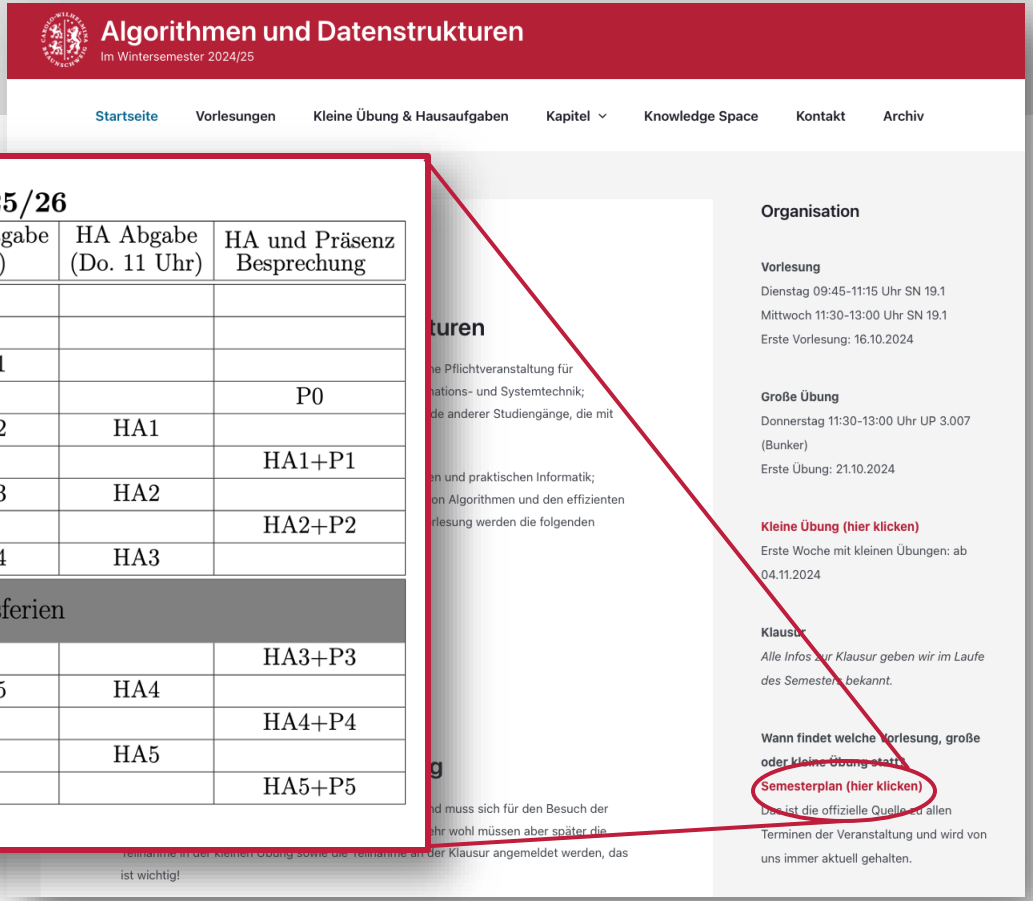
Vorname: Taian
Nachname: Vesper
Matrikelnummer: 5584359
Studiengang: Informatik
Semester: 1
v-Nummer: v0096548
EMail: c.loi@tu-bs.de
Montag 08:00: Ja
Montag 09:45: Nein
Montag 13:15: Wenn Sein Muss
Montag 15:00: Ja
Mittwoch 08:00: Wenn Sein Muss
Mittwoch 13:15: Ja
Mittwoch 15:00: Nein
Donnerstag 13:15: Ja
Freitag 09:45: Wenn Sein Muss
Freitag 11:30: Ja
Freitag 13:15: Ja

--
Diese E-Mail wurde von einem Kontaktformular von Algorithmen und Datenstrukturen (https://aud.ibr.cs.tu-bs.de) gesendet

Semesterplan

Semesterplan AuD WS25/26

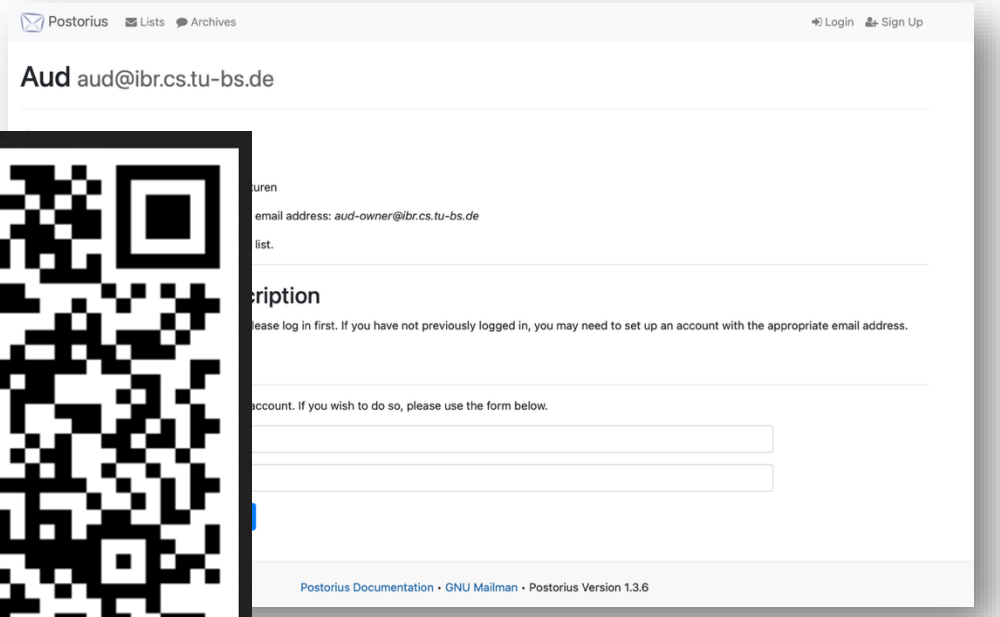
Woche (KW)	Woche (Datum)	Vorlesung (Di./Mi.)	Gr. Übung (Do.)	Kl. Übung	HA Ausgabe (Fr.)	HA Abgabe (Do. 11 Uhr)	HA und Präsenz Besprechung
43	20.10.	/,0					
44	27.10.	1,2	0				
45	03.11.	3,/	1		HA1		
46	10.11.	4,5	VL 6	0			P0
47	17.11.	7,8			HA2	HA1	
48	24.11.	9,10	2	1			HA1+P1
49	01.12.	11,12			HA3	HA2	
50	08.12.	13,14	3	2			HA2+P2
51	15.12.	15,16	4		HA4	HA3	
52	22.12.	Weihnachtsferien					
1	29.12.						
2	05.01.	17,18		3			HA3+P3
3	12.01.	19,20	5		HA5	HA4	
4	19.01.	21,22		4			HA4+P4
5	26.01.	23,24	6			HA5	
6	02.02.	25,26	7	5			HA5+P5



Mailingliste

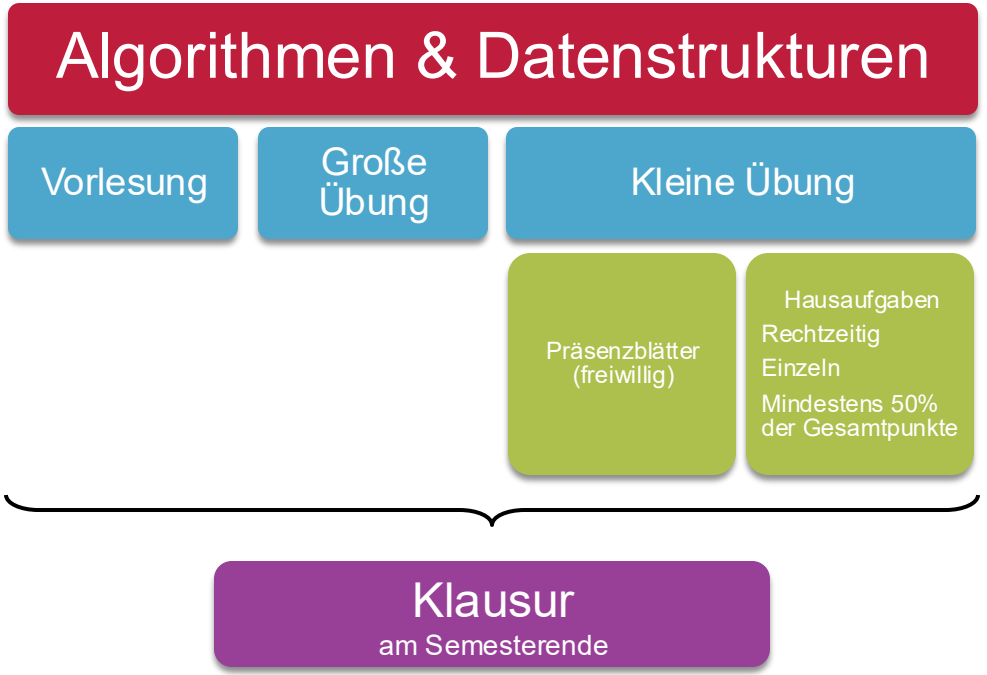
- Anmeldungslink auf der Homepage
- Für Informationen wie
  - Raumänderungen,
  - Ausfälle,
  - etc.
- Möglichkeit für Fragen

- Checkt regelmäßig eure
  (Generelles Ding an der



Zusammenfassung

- Kursseite
  - Semesterplan
  - Anmeldung kleine Übung
- Kommunikation
  - In Person
  - Mailingliste!
  - Mail an uns beide



Fragen

Stellt Eure Fragen …

- Während oder nach den Vorlesungen und Übungen
- Euren Tutoren in den kleinen Übungen!
- Über die Mailingliste (ins Plenum)
- Per Mail (kosfeld@ibr.cs.tu-bs.de & loi@ibr.cs.tu-bs.de)
  - Sprechstunde von Ramin und Chek-Manh (nach Vereinbarung)



Fragen?

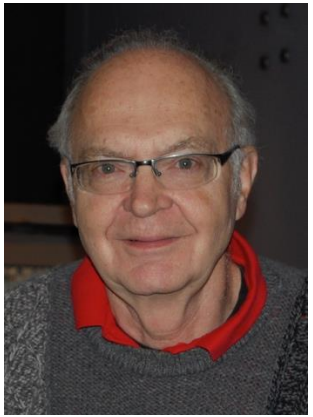
Algorithmen & Pseudocode

Algorithmus

Eindeutige Handlungsvorschrift zur Lösung eines Problems

- Eigenschaften
  - Ausführbarkeit
  - Endlichkeit
  - Endliche Ausführzeit
  - Endlicher Speicherbedarf

- Beschrieben durch
  - Prosatext
  - **Pseudocode**



Donald Knuth



Alan Turing

Pseudocode

Wörterbuch

pseu·do

/pseúdo/

Adjektiv UMGANGSSPRACHLICH

nicht echt, nur nachgemacht, nachgeahmt

Wörterbuch

Quell·code

/Quéllcode/

Substantiv, maskulin [der] EDV

in einer [höheren] Programmiersprache geschriebene Abfolge von Programmanweisungen, die vom Menschen gelesen, aber erst nach einer elektronischen Übersetzung vom Computer verarbeitet werden können

Warum Pseudocode?

```java
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

Ist dieser Code effizient zu lesen?

Was können wir weglassen, wenn ein Mensch (kein Computer) diesen Code verstehen soll?

Warum Pseudocode?

Java

```java
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

Pseudocode

```
function main(args)
  print Hello World!
end function
```

Beispiel: Euklidischer Algorithmus

Algorithmus zur Berechnung des größten gemeinsamen Teilers zwei ganzer Zahlen

Als Prosatext

Solange die Zahl b nicht 0 ist, wählen wir $h = a \bmod b$, setzen a auf b und b auf h. Nachdem $b = 0$, geben wir a zurück.

Als Pseudocode

function 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

$$7 \bmod 3 = 1, \qquad 7 = 2 * 3 + 1$$

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

Beispiel: Euklidischer Algorithmus

Pseudocode

function 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



Schlüsselwörter

- Anlehnung an höhere Programmiersprachen
- Vereinfachung zur Übersichtlichkeit (keine Klammern, Typen, etc.)
- ABER: **end** statements
- Können sowohl auf Deutsch als auch auf Englisch benutzt werden

Pseudocode - Bausteine

Methoden

function NAME (Param1, Param2, …)
  …
end function

Zuweisungen

$a := b$
$a \leftarrow b$ (Weist a den Wert b zu)

$a := b + c$ (Weist a die Summe aus b und c zu)
$A := B$ (Weist der Menge A die Menge B zu)

Bedingungen

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

Ausgaben / Rückgaben

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

Pseudocode - Bausteine

Schleifen - 1

while Bedingung **do**

Führe Aktion aus, *solange* eine Bedingung wahr ist.

end while

Schleifen - 2

repeat

Führe Aktion aus, *bis* eine Bedingung wahr ist.

until Bedingung

Schleifen - 3

for a **in** $b, \dots, c$ **do**

Starte mit $a \coloneqq b$

a wird nach jeder Iteration um 1 erhöht. ($a \coloneqq a + 1$)

Wiederhole bis a den Wert c erreicht hat.

end for

Schleifen - 4

for $a \coloneqq b$ **down to** c

Starte mit $a \coloneqq b$

a wird in jeder Iteration um 1 verringert. ($a \coloneqq a - 1$)

Wiederhole bis a den Wert c erreicht hat.

end for

Pseudocode - Varianten

- Auf Deutsch
  - Wenn … Dann … Sonst …
  - Solange …
  - Für $i \leftarrow 1, \ldots, k$ …
- Statt $\leftarrow$ schreibt man gelegentlich $:=$ oder sogar $=$

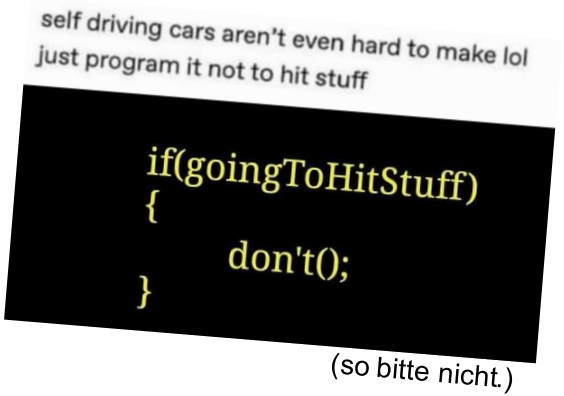
- Mischung aus Prosa und Pseudocode:

1. **function** Verhältnis(*liste*)
2. Sortiere die Zahlen in *liste* aufsteigend
3. **return** *liste*[1] / *liste*[länge(*liste*)]
4. **end function**

- end… statements dürfen weggelassen werden, aber Einrückung muss korrekt sein!

Wichtig
Pseudocode muss immer verständlich und eindeutig sein!

… und konsistent!



Problem und Instanz

Problem

- Allgemeine Fragestellung
- Lösung: Angabe eines Algorithmus
- (Meist) Formuliert durch
  - Eingabe: Was ist gegeben?
  - Ausgabe: Was ist gesucht?



Instanz

- Konkrete Fragestellung (Zahlen etc.)
- Lösung: Angabe einer konkreten Ausgabe
- Formuliert durch eine konkrete Eingabe eines bestimmten Problems

Problem und Instanz Beispiel

Problem → Instanz

Größter gemeinsamer Teiler zweier Zahlen

Eingabe: Zwei Zahlen a und b.

Ausgabe: Der größte gemeinsame Teiler q von a und b.

Lösung: Euklidischer Algorithmus (gerade gesehen)

Größter gemeinsamer Teiler zweier Zahlen

- $ggT(5, 102)$; **Lösung:** 1
- $ggT(8, 64)$; **Lösung:** 8
- …

Pseudocode – In Aktion

```
1.  function Mult(a, b)
2.      c ← 0
3.      for d ← b downto 1
4.          c ← c + a
5.      end for
6.      return c
7.  end function
```

Instanz: Berechne das Produkt von 9 und 3
Aufruf: Mult(9, 3)

Variablen nach jeder Iteration der for-Schleife:

Iteration	a	b	c	d

Pseudocode – In Aktion

```
1.  function Mult(a, b)
2.      c ← 0
3.      for d ← b downto 1
4.          c ← c + a
5.      end for
6.      return c
7.  end function
```

Instanz: Berechne das Produkt von 9 und 3

Aufruf: Mult(9, 3)

Rückgabe: 27



Variablen nach jeder Iteration der for-Schleife:

Iteration	a	b	c	d
1	9	3	9	3
2	9	3	18	2
3	9	3	27	1

*Das ist hier nur eine Demo für Pseudocode, natürlich dürft ihr in eurem Pseudocode einfach direkt Multiplikationen hinschreiben.

Pseudocode – In Aktion

1. **function** Euklid(a, b)
2. **while** b $\neq 0$ **do**
3. $h \leftarrow a \; mod \; b$
4. $a \leftarrow b$
5. $b \leftarrow h$
6. **end while**
7. **return** a
8. **end function**

Instanz: Berechne ggT von $a = 49, b = 21$
Aufruf: Euklid(49, 21)
Rückgabe: 7

Zeile	Iteration	a	b	h
1	-	49	21	-
2	-	49	21	-
3	1	49	21	7
4	1	21	21	7
5	1	21	7	7
6	1	21	7	7
2	1	21	7	7
3	2	21	7	0
4	2	7	7	0
5	2	7	0	0
6	2	7	0	0
2	2	7	0	0
7	2	7	0	0

Pseudocode – In Aktion

Der absolute Wert einer Zahl ist sein positiver Wert:
$\text{abs}(-5) = 5, \quad \text{abs}(11) = 11, \quad \text{abs}(0) = 0$

1. **function** Absolute(a)
2. **if** $a < 0$ **then**
3. $a \leftarrow -a$
4. **end if**
5. **return** a
6. **end function**

1. **function** AbsoluteSum(a, b)
2. **return** Absolute(a) + Absolute(b)
3. **end function**

Aufruf: AbsoluteSum(-10, 3)

Rückgabe: 13

Zeile (AbsoluteSum)	Zeile (Absolute)	a (AbsoluteSum)	a (Absolute)	b
1	-	-10	-	3
2	1	-10	-10	3
2	2	-10	-10	3
2	3	-10	10	3
2	4	-10	10	3
2	5	-10	10	3
2	1	-10	3	3
2	2	-10	3	3
2	4	-10	3	3
2	5	-10	3	3

Merkzettel Pseudocode

(Auf der Kursseite verlinkt)



Algorithmik

Technische Universität Braunschweig | Institut für Betriebssysteme und Rechnerverbund

Merkzettel ▷ Pseudocode

Version – 12. April 2021

Pseudocode dient der übersichtlichen und eindeutigen Darstellung von Algorithmen. Er ist meist an moderne höhere Programmiersprachen angelehnt, kann aber auch mit natürlicher Sprache gemischt werden. Pseudocode verzichtet auf technische Besonderheiten realer Programmiersprachen und ist dadurch kompakter, sollte sich aber dennoch ohne viel Aufwand in eine reale Implementierung überführen lassen.

Methoden

Funktionale Einheiten von Code werden zu benannten Blöcken (Funktionen) zusammengefasst.

- Funktions-Körper:
  function NAME(Parameter1, Parameter2,...)
  ...
  end function

Ein einfacher Algorithmus für den Logarithmus

log	n

Betrachten wir den Logarithmus:

Gegeben Zwei (ganzzahlige) Zahlen n und b

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

Einfach ausgedrückt: Wie oft muss man n durch b teilen, damit man auf 1 kommt?

Betrachten wir zunächst den einfachen Fall: x ist ganzzahlig

```
1.  function Log(n, b)
2.      log ← 0
3.      while n > 1 do
4.          n ← n/b
5.          log ← log + 1
6.      return log
```

Beispiel: Log(64, 2)

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: $x = \log_b n$)

Einfach ausgedrückt: Wie oft muss man n durch b teilen, damit man auf 1 kommt?

Betrachten wir zunächst den einfachen Fall: x ist ganzzahlig

```
1.  function Log(n, b)
2.      log ← 0
3.      while n > 1 do
4.          n ← n/b
5.          log ← log + 1
6.      return log
```

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 wird?

Problem: Irrationale Ergebnisse

Lösung: Bestimme den Logarithmus nur auf k Stellen genau.

Idee: Nutze den alten Algorithmus, indem wir die Stellen vor das Dezimalkomma verschieben.
$10^k \log_b n$ gibt uns k Stellen mehr vor dem Komma.

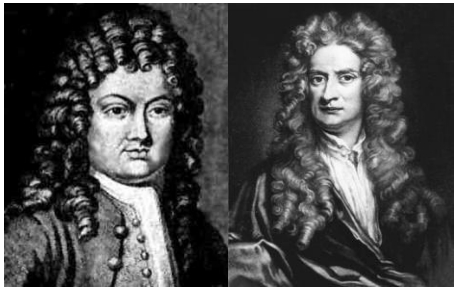
1. **function** Log(n, b, k)
2. $n \leftarrow n^{(10^k)}$
3. $\log \leftarrow \text{Log}(n, b)$
4. **return** $\log \cdot 10^{-k}$
5. **end function**

$$10^k \log_b n = \log_b(n^{(10^k)})$$

Problem 1: Schon für kleine k wird n sehr groß

Problem 2: Der Algorithmus benötigt exponentiell viele Schritte

Mehr zu Logarithmen



Logarithmen lassen sich deutlich schneller und präziser berechnen.

- Taylor-Entwicklung (aus der Analysis)
- Newton-Verfahren (aus der Numerik)

Für diese Vorlesung ist die genaue Berechnung von (nicht ganzzahligen) Logarithmen nicht relevant.

Logarithmen werden aber dennoch benötigt, beispielsweise

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

Wichtig: Rechenregeln für Logarithmen!

Fragen?

… nächstes Mal

Beweistechniken



… nächste Woche , am 06.11.!