

Algorithmen und Datenstrukturen

Vorlesung #00 – Organisatorisches



Benjamin Blankertz

Lehrstuhl für Neurotechnologie, TU Berlin

benjamin.blankertz@tu-berlin.de

10 · Apr · 2019



Willkommen zu “Algorithmen und Datenstrukturen”

Algorithmen

Verfahren zum
Lösen von Problemen
als schrittweise
Anleitung

Willkommen zu “Algorithmen und Datenstrukturen”

Algorithmen

Verfahren zum
Lösen von Problemen
als schrittweise
Anleitung

Datenstrukturen

Organisation von
Daten zur effizienten
Verarbeitung

Willkommen zu “Algorithmen und Datenstrukturen”

Algorithmen

Verfahren zum
Lösen von Problemen
als schrittweise
Anleitung

**Implementation durch Computerprogramm:
Objekt-orientierte Programmierung in Java**

Datenstrukturen

Organisation von
Daten zur effizienten
Verarbeitung

Willkommen zu “Algorithmen und Datenstrukturen”

- ▶ Spannende und wichtige Anwendungsgebiete der Informatik:
 - ▶ Maschinelles Lernen/ Künstliche Intelligenz
 - ▶ Robotics
 - ▶ Kommunikationssysteme
 - ▶ Computergrafik
 - ▶ Bildverarbeitung
 - ▶ Datenbanken
 - ▶ ...

Willkommen zu “Algorithmen und Datenstrukturen”

- ▶ Spannende und wichtige Anwendungsgebiete der Informatik:
 - ▶ Maschinelles Lernen/ Künstliche Intelligenz
 - ▶ Robotics
 - ▶ Kommunikationssysteme
 - ▶ Computergrafik
 - ▶ Bildverarbeitung
 - ▶ Datenbanken
 - ▶ ...
- ▶ In allen diesen Bereichen spielen **Algorithmen** eine entscheidende Rolle.
- ▶ Daher hat diese Vorlesung als Ausgangspunkt einen sehr wichtigen Stellenwert im Informatikstudium.

Willkommen zu “Algorithmen und Datenstrukturen”

- ▶ Die Kenntnisse über Algorithmen und Datenstrukturen sind für die Informatik so wichtig, dass sie häufig bei Bewerbungsgesprächen getestet werden.
- ▶ Dabei geht es nicht um auswendig gelernte Verfahren, sondern die Fähigkeit, algorithmische Lösungsstrategien zu unbekannten Problemvarianten zu entwickeln.

Willkommen zu “Algorithmen und Datenstrukturen”

- ▶ Die Kenntnisse über Algorithmen und Datenstrukturen sind für die Informatik so wichtig, dass sie häufig bei Bewerbungsgesprächen getestet werden.
- ▶ Dabei geht es nicht um auswendig gelernte Verfahren, sondern die Fähigkeit, algorithmische Lösungsstrategien zu unbekannten Problemvarianten zu entwickeln.
- ▶ Resulte für **Java** in der Umfrage von *hired.com* in der Rubrik 2019 State of Software Engineers – The Hottest Coding Languages:
- ▶ *Which programming languages do you primarily work with?*

Willkommen zu “Algorithmen und Datenstrukturen”

- ▶ Die Kenntnisse über Algorithmen und Datenstrukturen sind für die Informatik so wichtig, dass sie häufig bei Bewerbungsgesprächen getestet werden.
- ▶ Dabei geht es nicht um auswendig gelernte Verfahren, sondern die Fähigkeit, algorithmische Lösungsstrategien zu unbekannten Problemvarianten zu entwickeln.
- ▶ Resulte für **Java** in der Umfrage von *hired.com* in der Rubrik 2019 State of Software Engineers – The Hottest Coding Languages:
 - ▶ *Which programming languages do you primarily work with?*
2. Platz (nach JavaScript)

Willkommen zu “Algorithmen und Datenstrukturen”

- ▶ Die Kenntnisse über Algorithmen und Datenstrukturen sind für die Informatik so wichtig, dass sie häufig bei Bewerbungsgesprächen getestet werden.
- ▶ Dabei geht es nicht um auswendig gelernte Verfahren, sondern die Fähigkeit, algorithmische Lösungsstrategien zu unbekannten Problemvarianten zu entwickeln.
- ▶ Resulte für **Java** in der Umfrage von *hired.com* in der Rubrik 2019 State of Software Engineers – The Hottest Coding Languages:
 - ▶ *Which programming languages do you primarily work with?*
2. Platz (nach JavaScript)
 - ▶ *Most Loved Programming Languages:*

Willkommen zu “Algorithmen und Datenstrukturen”

- ▶ Die Kenntnisse über Algorithmen und Datenstrukturen sind für die Informatik so wichtig, dass sie häufig bei Bewerbungsgesprächen getestet werden.
- ▶ Dabei geht es nicht um auswendig gelernte Verfahren, sondern die Fähigkeit, algorithmische Lösungsstrategien zu unbekannten Problemvarianten zu entwickeln.
- ▶ Resulte für **Java** in der Umfrage von *hired.com* in der Rubrik 2019 State of Software Engineers – The Hottest Coding Languages:
 - ▶ *Which programming languages do you primarily work with?*
2. Platz (nach JavaScript)
 - ▶ *Most Loved Programming Languages:*
3. Platz (nach Python und JavaScript)

Willkommen zu “Algorithmen und Datenstrukturen”

- ▶ Die Kenntnisse über Algorithmen und Datenstrukturen sind für die Informatik so wichtig, dass sie häufig bei Bewerbungsgesprächen getestet werden.
- ▶ Dabei geht es nicht um auswendig gelernte Verfahren, sondern die Fähigkeit, algorithmische Lösungsstrategien zu unbekannten Problemvarianten zu entwickeln.
- ▶ Resulte für **Java** in der Umfrage von *hired.com* in der Rubrik 2019 State of Software Engineers – The Hottest Coding Languages:
 - ▶ *Which programming languages do you primarily work with?*
2. Platz (nach JavaScript)
 - ▶ *Most Loved Programming Languages:*
3. Platz (nach Python und JavaScript)
 - ▶ *Most Hated Programming Languages:*

Willkommen zu “Algorithmen und Datenstrukturen”

- ▶ Die Kenntnisse über Algorithmen und Datenstrukturen sind für die Informatik so wichtig, dass sie häufig bei Bewerbungsgesprächen getestet werden.
- ▶ Dabei geht es nicht um auswendig gelernte Verfahren, sondern die Fähigkeit, algorithmische Lösungsstrategien zu unbekannten Problemvarianten zu entwickeln.
- ▶ Resulte für **Java** in der Umfrage von *hired.com* in der Rubrik 2019 State of Software Engineers – The Hottest Coding Languages:
 - ▶ *Which programming languages do you primarily work with?*
2. Platz (nach JavaScript)
 - ▶ *Most Loved Programming Languages:*
3. Platz (nach Python und JavaScript)
 - ▶ *Most Hated Programming Languages:*
2. Platz (nach PHP)

Die Wichtigkeit von Laufzeitanalyse und -optimierung

- ▶ Die Vorlesung erklärt, wie komplexe Probleme **effizient** gelöst werden können.
- ▶ Dies ist auch angesichts immer schneller werdender Computer essenziell in der Informatik.

Die Wichtigkeit von Laufzeitanalyse und -optimierung

- ▶ Die Vorlesung erklärt, wie komplexe Probleme **effizient** gelöst werden können.
- ▶ Dies ist auch angesichts immer schneller werdender Computer essenziell in der Informatik.
- ▶ Das folgende Gedankenexperiment illustriert die Wichtigkeit, sich über Wachstumsverhalten von Laufzeit Gedanken zu machen und effiziente Implementationsmöglichkeiten zu kennen.

Die Wichtigkeit von Laufzeitanalyse und -optimierung

- ▶ Die Vorlesung erklärt, wie komplexe Probleme **effizient** gelöst werden können.
- ▶ Dies ist auch angesichts immer schneller werdender Computer essenziell in der Informatik.
- ▶ Das folgende Gedankenexperiment illustriert die Wichtigkeit, sich über Wachstumsverhalten von Laufzeit Gedanken zu machen und effiziente Implementationsmöglichkeiten zu kennen.
- ▶ Wir nehmen an, dass Sie ein Programm zur Lösung eines komplexeren Problems geschrieben haben.
- ▶ Durch geschicktes Programmieren ist es Ihnen gelungen, eine Laufzeit von **einer Minute** bei einer Eingabegröße von $N = 1000$ zu erreichen.

Die Wichtigkeit von Laufzeitanalyse und -optimierung

- ▶ Die Vorlesung erklärt, wie komplexe Probleme **effizient** gelöst werden können.
- ▶ Dies ist auch angesichts immer schneller werdender Computer essenziell in der Informatik.
- ▶ Das folgende Gedankenexperiment illustriert die Wichtigkeit, sich über Wachstumsverhalten von Laufzeit Gedanken zu machen und effiziente Implementationsmöglichkeiten zu kennen.
- ▶ Wir nehmen an, dass Sie ein Programm zur Lösung eines komplexeren Problems geschrieben haben.
- ▶ Durch geschicktes Programmieren ist es Ihnen gelungen, eine Laufzeit von **einer Minute** bei einer Eingabegröße von $N = 1000$ zu erreichen.
- ▶ Was passiert, wenn Ihr Programm in der Praxis eingesetzt wird, und die Datenmenge wächst, die das Programm verarbeiten muss?

Die Wichtigkeit von Laufzeitanalyse und -optimierung

- ▶ Die Vorlesung erklärt, wie komplexe Probleme **effizient** gelöst werden können.
- ▶ Dies ist auch angesichts immer schneller werdender Computer essenziell in der Informatik.
- ▶ Das folgende Gedankenexperiment illustriert die Wichtigkeit, sich über Wachstumsverhalten von Laufzeit Gedanken zu machen und effiziente Implementationsmöglichkeiten zu kennen.
- ▶ Wir nehmen an, dass Sie ein Programm zur Lösung eines komplexeren Problems geschrieben haben.
- ▶ Durch geschicktes Programmieren ist es Ihnen gelungen, eine Laufzeit von **einer Minute** bei einer Eingabegröße von $N = 1000$ zu erreichen.
- ▶ Was passiert, wenn Ihr Programm in der Praxis eingesetzt wird, und die Datenmenge wächst, die das Programm verarbeiten muss?
- ▶ Hierfür ist die Kenntnis der Wachstumsordnung der Laufzeit relevant.

Die Wichtigkeit von Laufzeitanalyse und -optimierung (2)

- ▶ Annahme: Die Laufzeit für eine Eingabe der Größe $N = 1000$ ist eine Minute.
- ▶ Als mögliche Wachstumsordnung der Laufzeit Ihres Programmes, betrachten wir die typischen Fälle: konstant, logarithmisch, linear, leicht überlinear, quadratisch, kubisch, und exponentiell.

Die Wichtigkeit von Laufzeitanalyse und -optimierung (2)

- ▶ Annahme: Die Laufzeit für eine Eingabe der Größe $N = 1000$ ist eine Minute.
- ▶ Als mögliche Wachstumsordnung der Laufzeit Ihres Programmes, betrachten wir die typischen Fälle: konstant, logarithmisch, linear, leicht überlinear, quadratisch, kubisch, und exponentiell.
- ▶ Nun betrachten wir, wie sich die Laufzeit für Eingabegrößen von $10N$, $100N$ und $1000N$ in Abhängigkeit der Wachstumsordnung verhält.

Wachstum	W-Ordnung	$N = 1000$	$10N$	$100N$	$1000N$
konstant	1	1 Minute			
logarithmisch	$\log N$	1 Minute			
linear	N	1 Minute			
leicht überlinear	$N \log N$	1 Minute			
quadratisch	N^2	1 Minute			
kubisch	N^3	1 Minute			
exponentiell	2^N	1 Minute			

Die Wichtigkeit von Laufzeitanalyse und -optimierung (2)

- ▶ Annahme: Die Laufzeit für eine Eingabe der Größe $N = 1000$ ist eine Minute.
- ▶ Als mögliche Wachstumsordnung der Laufzeit Ihres Programmes, betrachten wir die typischen Fälle: konstant, logarithmisch, linear, leicht überlinear, quadratisch, kubisch, und exponentiell.
- ▶ Nun betrachten wir, wie sich die Laufzeit für Eingabegrößen von $10N$, $100N$ und $1000N$ in Abhängigkeit der Wachstumsordnung verhält.

Wachstum	W-Ordnung	$N = 1000$	$10N$	$100N$	$1000N$
konstant	1	1 Minute	1 Minute	1 Minute	1 Minute
logarithmisch	$\log N$	1 Minute			
linear	N	1 Minute			
leicht überlinear	$N \log N$	1 Minute			
quadratisch	N^2	1 Minute			
kubisch	N^3	1 Minute			
exponentiell	2^N	1 Minute			

Die Wichtigkeit von Laufzeitanalyse und -optimierung (2)

- ▶ Annahme: Die Laufzeit für eine Eingabe der Größe $N = 1000$ ist eine Minute.
- ▶ Als mögliche Wachstumsordnung der Laufzeit Ihres Programmes, betrachten wir die typischen Fälle: konstant, logarithmisch, linear, leicht überlinear, quadratisch, kubisch, und exponentiell.
- ▶ Nun betrachten wir, wie sich die Laufzeit für Eingabegrößen von $10N$, $100N$ und $1000N$ in Abhängigkeit der Wachstumsordnung verhält.

Wachstum	W-Ordnung	$N = 1000$	$10N$	$100N$	$1000N$
konstant	1	1 Minute	1 Minute	1 Minute	1 Minute
logarithmisch	$\log N$	1 Minute			
linear	N	1 Minute	10 Minuten	2 Stunden	1 Tag
leicht überlinear	$N \log N$	1 Minute			
quadratisch	N^2	1 Minute			
kubisch	N^3	1 Minute			
exponentiell	2^N	1 Minute			

Die Wichtigkeit von Laufzeitanalyse und -optimierung (2)

- ▶ Annahme: Die Laufzeit für eine Eingabe der Größe $N = 1000$ ist eine Minute.
- ▶ Als mögliche Wachstumsordnung der Laufzeit Ihres Programmes, betrachten wir die typischen Fälle: konstant, logarithmisch, linear, leicht überlinear, quadratisch, kubisch, und exponentiell.
- ▶ Nun betrachten wir, wie sich die Laufzeit für Eingabegrößen von $10N$, $100N$ und $1000N$ in Abhängigkeit der Wachstumsordnung verhält.

Wachstum	W-Ordnung	$N = 1000$	$10N$	$100N$	$1000N$
konstant	1	1 Minute	1 Minute	1 Minute	1 Minute
logarithmisch	$\log N$	1 Minute	1, $\bar{3}$ Minuten	1, $\bar{6}$ Minuten	2 Minuten
linear	N	1 Minute	10 Minuten	2 Stunden	1 Tag
leicht überlinear	$N \log N$	1 Minute			
quadratisch	N^2	1 Minute			
kubisch	N^3	1 Minute			
exponentiell	2^N	1 Minute			

Die Wichtigkeit von Laufzeitanalyse und -optimierung (2)

- ▶ Annahme: Die Laufzeit für eine Eingabe der Größe $N = 1000$ ist eine Minute.
- ▶ Als mögliche Wachstumsordnung der Laufzeit Ihres Programmes, betrachten wir die typischen Fälle: konstant, logarithmisch, linear, leicht überlinear, quadratisch, kubisch, und exponentiell.
- ▶ Nun betrachten wir, wie sich die Laufzeit für Eingabegrößen von $10N$, $100N$ und $1000N$ in Abhängigkeit der Wachstumsordnung verhält.

Wachstum	W-Ordnung	$N = 1000$	$10N$	$100N$	$1000N$
konstant	1	1 Minute	1 Minute	1 Minute	1 Minute
logarithmisch	$\log N$	1 Minute	$1, \bar{3}$ Minuten	$1, \bar{6}$ Minuten	2 Minuten
linear	N	1 Minute	10 Minuten	2 Stunden	1 Tag
leicht überlinear	$N \log N$	1 Minute	13 Minuten	3 Stunden	$1 \frac{1}{2}$ Tage
quadratisch	N^2	1 Minute			
kubisch	N^3	1 Minute			
exponentiell	2^N	1 Minute			

Die Wichtigkeit von Laufzeitanalyse und -optimierung (2)

- ▶ Annahme: Die Laufzeit für eine Eingabe der Größe $N = 1000$ ist eine Minute.
- ▶ Als mögliche Wachstumsordnung der Laufzeit Ihres Programmes, betrachten wir die typischen Fälle: konstant, logarithmisch, linear, leicht überlinear, quadratisch, kubisch, und exponentiell.
- ▶ Nun betrachten wir, wie sich die Laufzeit für Eingabegrößen von $10N$, $100N$ und $1000N$ in Abhängigkeit der Wachstumsordnung verhält.

Wachstum	W-Ordnung	$N = 1000$	$10N$	$100N$	$1000N$
konstant	1	1 Minute	1 Minute	1 Minute	1 Minute
logarithmisch	$\log N$	1 Minute	$1, \bar{3}$ Minuten	$1, \bar{6}$ Minuten	2 Minuten
linear	N	1 Minute	10 Minuten	2 Stunden	1 Tag
leicht überlinear	$N \log N$	1 Minute	13 Minuten	3 Stunden	$1 \frac{1}{2}$ Tage
quadratisch	N^2	1 Minute	2 Stunden	5 Tage	23 Wochen
kubisch	N^3	1 Minute			
exponentiell	2^N	1 Minute			

Die Wichtigkeit von Laufzeitanalyse und -optimierung (2)

- ▶ Annahme: Die Laufzeit für eine Eingabe der Größe $N = 1000$ ist eine Minute.
- ▶ Als mögliche Wachstumsordnung der Laufzeit Ihres Programmes, betrachten wir die typischen Fälle: konstant, logarithmisch, linear, leicht überlinear, quadratisch, kubisch, und exponentiell.
- ▶ Nun betrachten wir, wie sich die Laufzeit für Eingabegrößen von $10N$, $100N$ und $1000N$ in Abhängigkeit der Wachstumsordnung verhält.

Wachstum	W-Ordnung	$N = 1000$	$10N$	$100N$	$1000N$
konstant	1	1 Minute	1 Minute	1 Minute	1 Minute
logarithmisch	$\log N$	1 Minute	$1, \bar{3}$ Minuten	$1, \bar{6}$ Minuten	2 Minuten
linear	N	1 Minute	10 Minuten	2 Stunden	1 Tag
leicht überlinear	$N \log N$	1 Minute	13 Minuten	3 Stunden	$1 \frac{1}{2}$ Tage
quadratisch	N^2	1 Minute	2 Stunden	5 Tage	23 Wochen
kubisch	N^3	1 Minute	1 Tag	2 Jahre	2000 Jahre
exponentiell	2^N	1 Minute			

Die Wichtigkeit von Laufzeitanalyse und -optimierung (2)

- ▶ Annahme: Die Laufzeit für eine Eingabe der Größe $N = 1000$ ist eine Minute.
- ▶ Als mögliche Wachstumsordnung der Laufzeit Ihres Programmes, betrachten wir die typischen Fälle: konstant, logarithmisch, linear, leicht überlinear, quadratisch, kubisch, und exponentiell.
- ▶ Nun betrachten wir, wie sich die Laufzeit für Eingabegrößen von $10N$, $100N$ und $1000N$ in Abhängigkeit der Wachstumsordnung verhält.

Wachstum	W-Ordnung	$N = 1000$	$10N$	$100N$	$1000N$
konstant	1	1 Minute	1 Minute	1 Minute	1 Minute
logarithmisch	$\log N$	1 Minute	$1, \bar{3}$ Minuten	$1, \bar{6}$ Minuten	2 Minuten
linear	N	1 Minute	10 Minuten	2 Stunden	1 Tag
leicht überlinear	$N \log N$	1 Minute	13 Minuten	3 Stunden	$1 \frac{1}{2}$ Tage
quadratisch	N^2	1 Minute	2 Stunden	5 Tage	23 Wochen
kubisch	N^3	1 Minute	1 Tag	2 Jahre	2000 Jahre
exponentiell	2^N	1 Minute	inf	inf	inf

Die Wichtigkeit von Laufzeitanalyse und -optimierung (2)


- ▶ Annahme: Die Laufzeit für eine Eingabe der Größe $N = 1000$ ist eine Minute.
- ▶ Als mögliche Wachstumsordnung der Laufzeit Ihres Programmes, betrachten wir die typischen Fälle: konstant, logarithmisch, linear, leicht überlinear, quadratisch, kubisch, und exponentiell.
- ▶ Nun betrachten wir, wie sich die Laufzeit für Eingabegrößen von $10N$, $100N$ und $1000N$ in Abhängigkeit der Wachstumsordnung verhält.


Wachstum	W-Ordnung	$N = 1000$	$10N$	$100N$	$1000N$
konstant	1	1 Minute	1 Minute	1 Minute	1 Minute
logarithmisch	$\log N$	1 Minute	$1, \bar{3}$ Minuten	$1, \bar{6}$ Minuten	2 Minuten
linear	N	1 Minute	10 Minuten	2 Stunden	1 Tag
leicht überlinear	$N \log N$	1 Minute	13 Minuten	3 Stunden	$1 \frac{1}{2}$ Tage
quadratisch	N^2	1 Minute	2 Stunden	5 Tage	23 Wochen
kubisch	N^3	1 Minute	1 Tag	2 Jahre	2000 Jahre
exponentiell	2^N	1 Minute	inf	inf	inf

Der konstante Faktor in den Laufzeitfunktionen ist so gewählt, dass sich für $N = 1000$ eine Minute ergibt. Die angegebenen Laufzeiten für $10N$, $100N$, $1000N$ sind gerundet.

Kurzvorstellung – Benjamin Blankertz

- ▶ Studium: Mathematik mit Nebenfächern Informatik und Mathematische Logik
- ▶ Seit 2012 Professor an der TU Berlin

- ▶ Studium: Mathematik mit Nebenfächern Informatik und Mathematische Logik
- ▶ Seit 2012 Professor an der TU Berlin
- ▶ Forschungsschwerpunkt Neurotechnologie mit Methoden des Maschinellen Lernens
- ▶ Speziell: *Brain-Computer Interfaces* (BCI) 
- ▶ BCIs müssen komplexe, hoch-dimensionale Merkmale der Gehirnsignale in Echtzeit analysieren können.

- ▶ Studium: Mathematik mit Nebenfächern Informatik und Mathematische Logik
- ▶ Seit 2012 Professor an der TU Berlin
- ▶ Forschungsschwerpunkt Neurotechnologie mit Methoden des Maschinellen Lernens
- ▶ Speziell: *Brain-Computer Interfaces* (BCI) 
- ▶ BCIs müssen komplexe, hoch-dimensionale Merkmale der Gehirnsignale in Echtzeit analysieren können.

Mögliche Anwendungen:

- ▶ Medizinisch: Kommunikation für gelähmte Patienten
- ▶ Therapie: Effektivere Rehabilitationsverfahren
- ▶ Produktentwicklung: Beleuchtung und Videokodierung besser an Menschen angepasst
- ▶ *Human Factor*: Unterstützung in sicherheitsrelevanten Situationen

- ▶ **Benjamin Blankertz**
- ▶ Fachgebiet Neurotechnologie, Fakultät IV der TU Berlin
- ▶ Sprechstunde: Mi 11:00 bis 12:00 Uhr, Raum 4.041
- ▶ Anmeldung über Sekretariat MAR 4-3:
- ▶ Imke Weitkamp <imke.weitkamp@tu-berlin.de>

- ▶ **Vera Röhr**
- ▶ **Kolja Stahl** (Fachgebiet *Robotics and Biology Laboratory*)
- ▶ Kontakt: `algodat@neuro.tu-berlin.de`
- ▶ Kontakt zu den Tutorinnen und Tutoren: `algodat.tutors@neuro.tu-berlin.de`

Das Wichtigste für die Organisation

- ▶ Die wichtigste Quelle für Informationen: ISIS Kurs “Algorithmen und Datenstrukturen”:

<https://isis.tu-berlin.de/course/view.php?id=15622>

- ▶ Vorlesungsfolien und ggf. andere Materialien
 - ▶ Aktuelle Ankündigung
 - ▶ Termine, Räume (Tutorien, Klausuren, ...)
 - ▶ Diskussionsforen
 - ▶ Kontakte und Antworten auf häufige Fragen
- ▶ Melden Sie sich **möglichst bald** an sofern nicht schon geschehen, mit einer Email Adresse, deren Nachrichten Sie sicher empfangen und häufig lesen.
Ohne TU Account: “Als Gast anmelden” und dann als Gastschlüssel *AlgoDat19* eingeben.

Das Wichtigste für die Organisation

- ▶ Die wichtigste Quelle für Informationen: ISIS Kurs “Algorithmen und Datenstrukturen”:

<https://isis.tu-berlin.de/course/view.php?id=15622>

- ▶ Vorlesungsfolien und ggf. andere Materialien
 - ▶ Aktuelle Ankündigung
 - ▶ Termine, Räume (Tutorien, Klausuren, ...)
 - ▶ Diskussionsforen
 - ▶ Kontakte und Antworten auf häufige Fragen
- ▶ Melden Sie sich **möglichst bald** an sofern nicht schon geschehen, mit einer Email Adresse, deren Nachrichten Sie sicher empfangen und häufig lesen.
Ohne TU Account: “Als Gast anmelden” und dann als Gastschlüssel *AlgoDat19* eingeben.
- ▶ **Besonders dringend:** Bis **heute um 18 Uhr** in MOSES für die Tutorien anmelden.
Falls dies nicht möglich ist, E-Mail mit Angabe von Matr.Nr., Studiengang/Uni und blockierten Zeitslots an algotat@neuro-tu-berlin.de.

Modul “Algorithmen und Datenstrukturen”

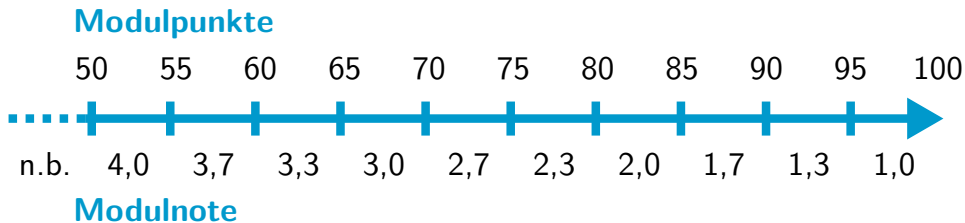
- ▶ Modul “Algorithmen und Datenstrukturen”: QISPOS #6140, und MOSES #40022.
- ▶ Anmeldung zur Modulprüfung #6145 in QISPOS **am besten direkt**. Anmeldungen und Abmeldungen sind bis **25.05.2019** möglich.

Modul “Algorithmen und Datenstrukturen”

- ▶ Modul “Algorithmen und Datenstrukturen”: QISPOS #6140, und MOSES #40022.
- ▶ Anmeldung zur Modulprüfung #6145 in QISPOS **am besten direkt**. Anmeldungen und Abmeldungen sind bis **25.05.2019** möglich.
- ▶ Die Prüfungselemente des Moduls sind (Portfolioprüfung):
- ▶ **50% Übungen:** 10 Aufgabenblätter, je 5 Modulpunkte
- ▶ **50% Klausur:** Schriftlicher Test (85 Minuten): 50 Modulpunkte

Modul “Algorithmen und Datenstrukturen”

- ▶ Modul “Algorithmen und Datenstrukturen”: QISPOS #6140, und MOSES #40022.
- ▶ Anmeldung zur Modulprüfung #6145 in QISPOS **am besten direkt**. Anmeldungen und Abmeldungen sind bis **25.05.2019** möglich.
- ▶ Die Prüfungselemente des Moduls sind (Portfolioprüfung):
- ▶ **50% Übungen:** 10 Aufgabenblätter, je 5 Modulpunkte
- ▶ **50% Klausur:** Schriftlicher Test (85 Minuten): 50 Modulpunkte
- ▶ Benotung nach Notenschlüssel 2 der Fakultät IV:



- ▶ **als Pflichtmodul:**

- ▶ Anmeldung über QISPOS, siehe vorige Seite

- ▶ **als Zusatzmodul:**

- ▶ Anmeldung über gelben Zettel mit vorheriger Genehmigung, siehe ISIS

- ▶ **für Nebenhörer/innen und Gasthörer/innen:**

- ▶ Anmeldung über gelben Zettel mit vorheriger Genehmigung, siehe ISIS
siehe auch <http://www.tu-berlin.de/?id=76326>

- ▶ **im Orientierungsstudium MINTgrün oder Erasmus:**

- ▶ Anmeldung per Email an imke.weitkamp@tu-berlin.de, siehe ISIS

Komponenten der Veranstaltung in der Übersicht

- ▶ **Vorlesung**
- ▶ **Übungsaufgaben**
 - ▶ Programmieraufgaben
 - ▶ In Einzelabgabe

- ▶ **Vorlesung**
- ▶ **Übungsaufgaben**
 - ▶ Programmieraufgaben
 - ▶ In Einzelabgabe
- ▶ **Tutorien**
 - ▶ Erläuterungen und Übungen zum Vorlesungsstoff
 - ▶ Besprechung und Vorbereitung der Aufgabenzettel
 - ▶ Fragen und Antworten
 - ▶ Betreuung beim Programmieren (wenn noch Zeit ist)

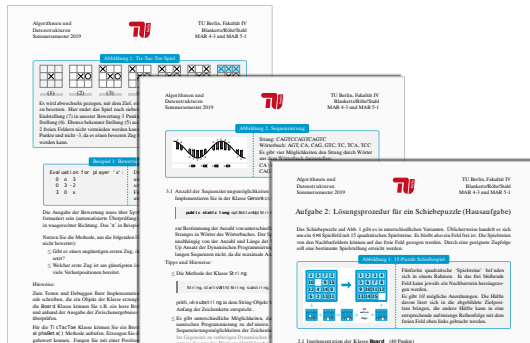
- ▶ **Vorlesung**
- ▶ **Übungsaufgaben**
 - ▶ Programmieraufgaben
 - ▶ In Einzelabgabe
- ▶ **Tutorien**
 - ▶ Erläuterungen und Übungen zum Vorlesungsstoff
 - ▶ Besprechung und Vorbereitung der Aufgabenzettel
 - ▶ Fragen und Antworten
 - ▶ Betreuung beim Programmieren (wenn noch Zeit ist)
- ▶ **Rechnerübungen** in Rechnerräumen
 - ▶ Betreuung beim Programmieren
 - ▶ nur wenige Plätze (daher auch Tutorien nutzen)
- ▶ **Fachmentorien**

- ▶ Vorlesungsfolien werden auf ISIS und in einem *git* bereitgestellt.
- ▶ Korrekturen bitte an mich `<benjamin.blankertz@tu-berlin.de>`

- ▶ Anmeldung in MOSES **bis heute um 18 Uhr!**
- ▶ Mitteilung der zugeordneten Termine über ISIS
- ▶ Beginn der Tutorien: nächste Woche Mittwoch (17.04.)

- ▶ Alle Aufgaben sind Programmieraufgaben mit Einzelabgabe.
- ▶ Aufgabenzettel gibt es jeweils am **Mittwoch**.
- ▶ Abgabe in der übernächsten Woche am **Dienstag bis 20 Uhr**.

- ▶ Alle Aufgaben sind Programmieraufgaben mit Einzelabgabe.
- ▶ Aufgabenzettel gibt es jeweils am **Mittwoch**.
- ▶ Abgabe in der übernächsten Woche am **Dienstag bis 20 Uhr**.
- ▶ Heute kein Übungszettel, aber Informationen zur Software Installation, siehe ISIS!
- ▶ In den folgenden zehn Wochen gibt es Übungszettel, die für die Modulnote zählen.



- ▶ Die Abgabe geschieht über ein Versionsverwaltungssystem
 - ▶ Alle Versionen werden mit Zeitstempel und Benutzerkennung gesichert
 - ▶ Versionen können später wiederhergestellt werden
 - ▶ Versionsverwaltungssysteme werden u.a. in der Softwareentwicklung zur Quelltextverwaltung eingesetzt
- ▶ Wir verwenden GIT, siehe heutiges Blatt

- ▶ Die Abgabe geschieht über ein Versionsverwaltungssystem
 - ▶ Alle Versionen werden mit Zeitstempel und Benutzerkennung gesichert
 - ▶ Versionen können später wiederhergestellt werden
 - ▶ Versionsverwaltungssysteme werden u.a. in der Softwareentwicklung zur Quelltextverwaltung eingesetzt
- ▶ Wir verwenden GIT, siehe heutiges Blatt
- ▶ Die Abgaben werden automatisch über spezielle Tests bewertet.
- ▶ Angebot (ohne Garantie): Bis zum Abgabetermin wird pro Tag ein Lösungsversuch vorab getestet. (Hochladen bis 23:59 Uhr, Bewertung am nächsten Morgen, wenn möglich). Diese Bewertung kann zur Kontrolle benutzt werden.

- ▶ Die Abgabe geschieht über ein **Versionsverwaltungssystem**
 - ▶ Alle Versionen werden mit Zeitstempel und Benutzerkennung gesichert
 - ▶ Versionen können später wiederhergestellt werden
 - ▶ Versionsverwaltungssysteme werden u.a. in der Softwareentwicklung zur Quelltextverwaltung eingesetzt
- ▶ Wir verwenden GIT, siehe heutiges Blatt
- ▶ Die Abgaben werden **automatisch** über spezielle Tests **bewertet**.
- ▶ Angebot (ohne Garantie): Bis zum Abgabetermin wird **pro Tag ein Lösungsversuch** vorab getestet. (Hochladen bis 23:59 Uhr, Bewertung am nächsten Morgen, wenn möglich). Diese Bewertung kann zur Kontrolle benutzt werden.
- ▶ Das wichtigste Mittel zur Verbesserung der Implementationen sollte **sorgfältiges Debugging** und ggf. selbstgeschriebene Tests sein.
- ▶ Die letzte Einsendung vor dem Abgabetermin zählt. Endredaktion der Korrektur durch WiMis; die Punktzahl kann vom automatischen Testergebnis abweichen.

- ▶ Für Erfolg und das richtige Verständnis: Üben, Üben, Üben
- ▶ Es gibt viele Angebote zur Unterstützung beim Programmieren: nutzt sie!

- ▶ Für Erfolg und das richtige Verständnis: Üben, Üben, Üben
- ▶ Es gibt viele Angebote zur Unterstützung beim Programmieren: nutzt sie!
- ▶ Macht die Programmierung der Übungsaufgaben **selbstständig**:
- ▶ Kein Quelltext von anderen Personen oder Quellen.
- ▶ Die Lösungen werden mit Plagiatserkennungssoftware geprüft.

- ▶ Für Erfolg und das richtige Verständnis: Üben, Üben, Üben
- ▶ Es gibt viele Angebote zur Unterstützung beim Programmieren: nutzt sie!
- ▶ Macht die Programmierung der Übungsaufgaben **selbstständig**:
- ▶ Kein Quelltext von anderen Personen oder Quellen.
- ▶ Die Lösungen werden mit Plagiatserkennungssoftware geprüft.
- ▶ **Plagiate** werden als Betrugsversuch gewertet und führen zu **Nichtbestehen** des Kurses für Geber und Nehmer.
- ▶ Diskussionen über Lösungswege sind **ausdrücklich erlaubt**,
- ▶ die Implementierung muss selbstständig gemacht werden.

- ▶ Es sind jeweils zwei Tutor/inn/en anwesend.
- ▶ Mo 12-16 Uhr, Raum MAR 6.001
- ▶ Fr 14-18 Uhr, Raum MAR 6.057
- ▶ Anfang schon ab dieser Woche Freitag

- ▶ Hilfe vornehmlich für ausländische, aber auch deutsche Studierende:
 - ▶ Abbau fachlicher Schwierigkeiten
 - ▶ Orientierung am Studienanfang
 - ▶ Einführung in die Lern- und Lehrformen an der Hochschule
 - ▶ Vorbereitung auf Klausuren und Prüfungen
- ▶ Nicolai Skutsch
 - ▶ Mi 12-14 Uhr, Raum MAR 4.044 Fachmentorium
 - ▶ Mi 14-16 Uhr, Raum FH 523 Sprechstunde
 - ▶ Mi 16-18 Uhr, Raum MAR 4.044 Fachmentorium

Überblick über den Kurs (momentaner Planungsstand)

1. Einführung Java: Objektorientierte Programmierung
2. Einführung Java: Vererbung, Generics, Debugging; Wachstumsordnungen
3. Einführung Java: Ausnahmenbehandlung, Unit Tests; Vorrangwarteschlangen
4. *Backtracking*, P und NP, Gierige Algorithmen
5. *Branch-and-Bound*, Alpha-Beta Suche, Randomisierte Algorithmen
6. Dynamische Programmierung
7. Graphenalgorithmen: Tiefen- und Breitensuche
8. Minimale Spannbäume
9. Kürzeste Wege, Topologische Sortierung
10. Flussgraphen, maximaler Fluss, minimaler Schnitt
11. Heuristische Algorithmen, Approximative Algorithmen
12. Effektive Symboltabellen mit *Hashing*

- ▶ **Vorlesungen**
 - ▶ Einführung in die Programmierung (*IntroProg*)
 - ▶ Rechnerorganisation

- ▶ **Vorlesungen**

- ▶ Einführung in die Programmierung (*IntroProg*)
- ▶ Rechnerorganisation

- ▶ **Kenntnisse**

- ▶ Elementare Datenstrukturen: verkettete Listen, Arrays, Bäume, binäre Halden
- ▶ Komplexitätsanalyse von Algorithmen
- ▶ Sortier- und Suchverfahren
- ▶ Aufbau und Funktionsweise eines Computers

- ▶ **Vorlesungen**

- ▶ Einführung in die Programmierung (*IntroProg*)
- ▶ Rechnerorganisation

- ▶ **Kenntnisse**

- ▶ Elementare Datenstrukturen: verkettete Listen, Arrays, Bäume, binäre Halden
- ▶ Komplexitätsanalyse von Algorithmen
- ▶ Sortier- und Suchverfahren
- ▶ Aufbau und Funktionsweise eines Computers

- ▶ **Fähigkeiten**

- ▶ Imperative Programmierung am Beispiel C
- ▶ Versionskontrollsysteme mit svn oder git

Kenntnis

- ▶ von komplexere Datenstrukturen (Graphen, Hashtabellen, ...)
- ▶ der wichtigsten elementaren Algorithmen (kürzeste Wege, minimaler Spannbaum, Hashing, ...)
- ▶ der Verfahren zur Entwicklung effizienter Algorithmen (Dynamisches Programmieren, *Branch-and-Bound*, Alpha-Beta Suche, Heuristische Verfahren, ...)

Kenntnis

- ▶ von komplexere Datenstrukturen (Graphen, Hashtabellen, ...)
- ▶ der wichtigsten elementaren Algorithmen (kürzeste Wege, minimaler Spannbaum, Hashing, ...)
- ▶ der Verfahren zur Entwicklung effizienter Algorithmen (Dynamisches Programmieren, *Branch-and-Bound*, Alpha-Beta Suche, Heuristische Verfahren, ...)

Fähigkeit

- ▶ für ein Anwendungsproblem passende Algorithmen und Datenstrukturen zu identifizieren
- ▶ Rechenzeit und Speicherbedarf von Algorithmen abzuschätzen

Kenntnis

- ▶ von komplexere Datenstrukturen (Graphen, Hashtabellen, ...)
- ▶ der wichtigsten elementaren Algorithmen (kürzeste Wege, minimaler Spannbaum, Hashing, ...)
- ▶ der Verfahren zur Entwicklung effizienter Algorithmen (Dynamisches Programmieren, *Branch-and-Bound*, Alpha-Beta Suche, Heuristische Verfahren, ...)

Fähigkeit

- ▶ für ein Anwendungsproblem passende Algorithmen und Datenstrukturen zu identifizieren
- ▶ Rechenzeit und Speicherbedarf von Algorithmen abzuschätzen
- ▶ verständliche Programme in Java zu schreiben
- ▶ ... unter Verwendung von Objektorientierung und
- ▶ ... mit Debugging, Unit Tests und Kommentierstandards

Programmierung in Java

- ▶ Ullendoom C, **Java ist auch eine Insel**. 13. Auflage, Rheinwerk Computing, 2018. Onlinefassung: <http://openbook.rheinwerk-verlag.de/javainsel>
- ▶ Programming Guide – Java: <https://programming.guide/java/>
- ▶ Sedgewick R & Wayne K, **Introduction to Programming in Java: An Interdisciplinary Approach**. 2. Auflage, Addison-Wesley Professional, 2017. Onlinefassung: <https://introcs.cs.princeton.edu/java>
- ▶ MOOC Platform Hyperskill: <https://hyperskill.org>

Algorithmen und Datenstrukturen

- ▶ Sedgewick R & Wayne K, **Algorithmen: Algorithmen und Datenstrukturen**. Pearson Studium, 4. Auflage, 2014. ISBN: 978-3868941845
- ▶ Ottmann T & Widmayer P **Algorithmen und Datenstrukturen**. Springer Verlag, 5. Auflage; 2011. ISBN: 978-3827428042
- ▶ Cormen TH, Leiserson CE, Rivest R, Stein C, **Algorithmen - Eine Einführung**. De Gruyter Oldenbourg, 4. Auflage; 2013. ISBN: 978-3486748611