Algorithmen und Berechnungskomplexität I

Prof. Dr. Heiko Röglin

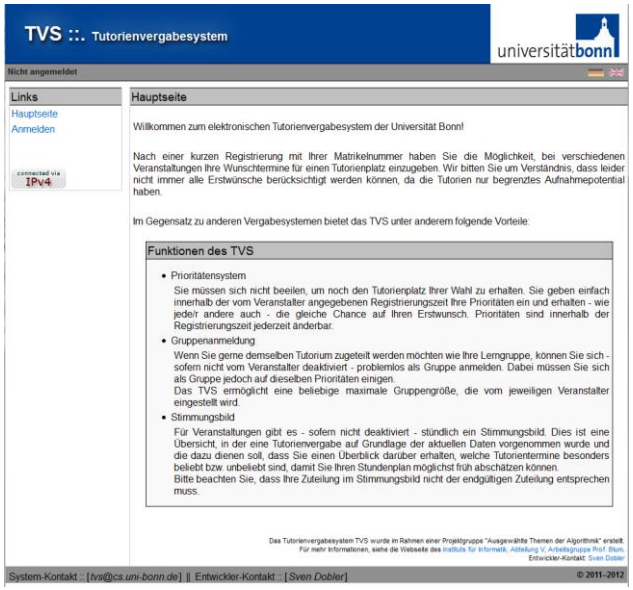
Institut für Informatik

Wintersemester 2013/14

Organisatorisches

- **Vorlesung**
  Dienstag und Donnerstag, 12:30 – 14:00 Uhr (HS 1)

- **Übungen**
  16 Übungsgruppen
  Anmeldung über Tutorienvergabesystem
  https://puma.cs.uni-bonn.de/
  bis Freitag, den 18.10., um 23:59 Uhr.



universitätbonn

Dozenten und Tutoren

- **Vorlesung**
  Heiko Röglin
  Professor für Theoretische Informatik
  E-Mail: roeglin@uni-bonn.de
  Web: http://www.roeglin.org/
  Sprechstunde: nach Vereinbarung
  und nach der Vorlesung
  Büro: Friedrich-Ebert-Allee 144, E.63

- **Übungen**
  Organisation: Tobias Brunsch, Clemens Rösner
  studentische Tutoren: Philipp Bruckschen, Patrick Loka, Yvonne Omlor, Sonja Schäfer, Patrick Seume

universitätbonn

Übungsblätter und Studienleistung

Übungsblätter

- Ausgabe: Dienstag in der Vorlesung & auf Webseite
- Abgabe: Dienstag darauf, 12:30 Uhr, Briefkasten Römerstr.
- Besprechung: in der Woche nach der Abgabe

Studienleistung

- insgesamt mind. 50% der möglichen Punkte
  (Abgabe in Gruppen mit max. 3 Studenten möglich)
- Präsentation von 3 (Teil)aufgaben in den Übungen

universitätbonn

Literatur


Skript zur Vorlesung

Algorithmen und Berechnungskomplexität I

Prof. Dr. Heiko Röglin
Institut für Informatik

universitätbonn

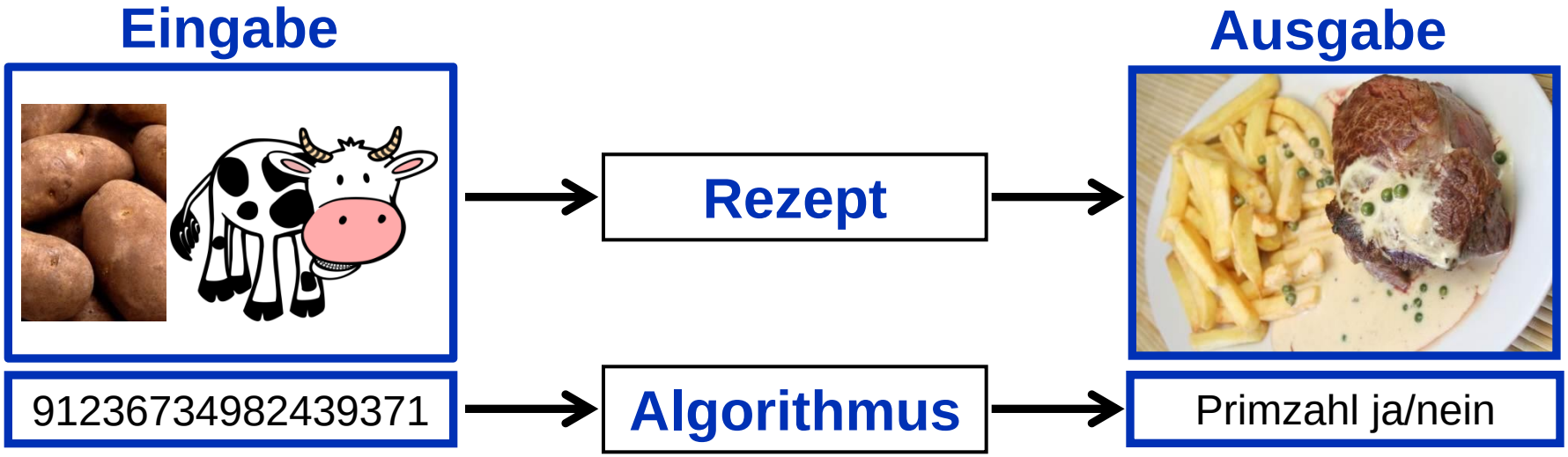
Wintersemester 2013/14
11. Oktober 2013

- Zu dieser Vorlesung gibt es ein Skript: http://www.roeglin.org/teaching/WS2013/AlgoI/AlgoI.pdf
- Dieses wird während des Semesters kontinuierlich erweitert.
- Fallen Ihnen Fehler auf oder haben Sie Anregungen und Verbesserungs-vorschläge zum Skript, so sagen Sie mir bitte Bescheid.
- Sekundärliteratur: siehe Skript

universitätbonn

Algorithmik

Algorithmen sind formal definierte Handlungsvorschriften, die von Computern wie ein Rezept zur Lösung eines Problems ausgeführt werden. Dabei wird eine Eingabe in eine Ausgabe transformiert.



Algorithmus zum Primzahltest:
Bei Eingabe n
teste alle Zahlen 2,3,4,…,n-1, ob sie Teiler von n sind.
Ausgabe: Nein falls Teiler gefunden, sonst Ja.

universitätbonn

Algorithmik (2)

Algorithmen begegnen uns täglich.

Navigationsgerät
Was ist der kürzeste Wege von A nach B?

Online Banking, Einkaufen im Internet
Wie werden die Daten ver- und entschlüsselt?

Google

Suchmaschinen
Wie sucht man in einer großen Datenmenge?

Paketdienste, Logistikdienstleister
In welcher Reihenfolge werden Kunden beliefert?

universitätbonn

Datenstrukturen

Ziel: Korrekte und **schnelle Algorithmen**.

Wichtiges Hilfsmittel: **Datenstrukturen**.

Wie sollen Daten gespeichert werden, um schnelle Ausführung wichtiger Operationen zu ermöglichen?

Beispiel: Suchen in einer großen Datenmenge.

universitätbonn

Lernziele

- Kenntnis grundlegender Datenstrukturen, Methoden und Algorithmen
- Fähigkeit, ein gegebenes Problem zu analysieren:
  - Einordnung der Schwierigkeit des Problems
  - Auswahl geeigneter Datenstrukturen und Algorithmen
- Entwurf von Algorithmen
- Analyse von Algorithmen

universitätbonn

Berechenbarkeitstheorie

Welche Probleme können von Computern gelöst werden?

```
void foo (int a) {
  int i = a;
  while (i != 10) {
    i++;
  }
}
```

Es wäre schön, wenn …

Compilerwarnung: Eine Endlosschleife kann auftreten.

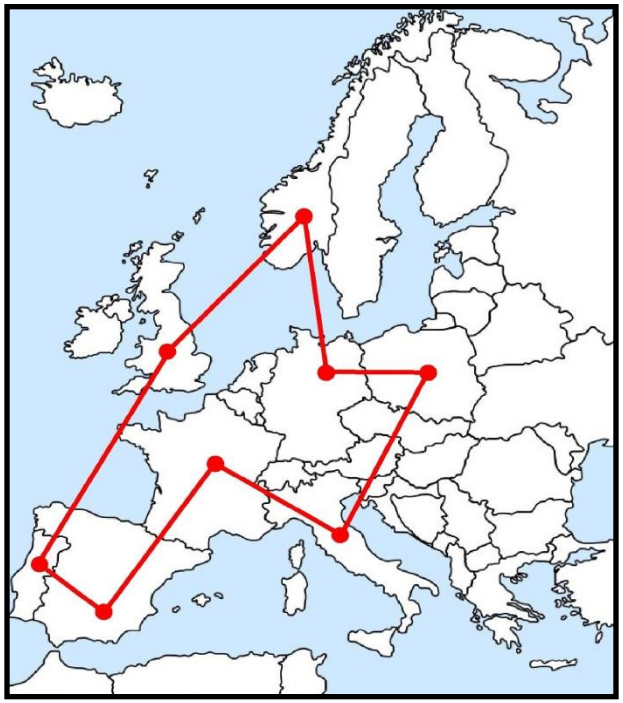
Ein Resultat der Berechenbarkeitstheorie besagt:
Es gibt keinen Compiler, der automatisch testet, ob ein gegebenes Programm in eine Endlosschleife geraten kann; egal, wie leistungsstark unsere Rechner auch sein mögen.

→ Algorithmen und Berechnungskomplexität II

universitätbonn

Komplexitätstheorie

Welche Ressourcen (z.B. Rechenzeit und Speicherplatz) sind nötig, um ein Problem zu lösen?



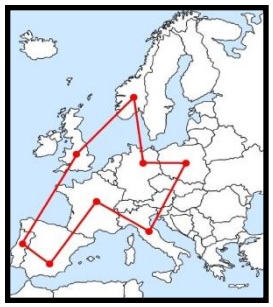
Problem des Handlungsreisenden:
Finde kürzeste Tour durch alle Städte.

Algorithmus: Teste alle Möglichkeiten und wähle die kürzeste Tour.
Keine gute Idee: Bei 20 Städten mehr als $6 \cdot 10^{16}$ Möglichkeiten.

Das Problem ist **NP-hart**, d.h. vermutlich, gibt es keinen effizienten Algorithmus.

universitätbonn

Komplexitätstheorie (2)



Es gibt zahlreiche natürliche **NP-harte Probleme**.
Entweder: Für alle gibt es effiziente Algorithmen.
Oder: Für keines gibt es effizienten Algorithmus.

Dies ist eine große ungelöste Frage (**P≠NP-Vermutung**).
Lösung ist 1 Million Dollar Wert.

Negative Ergebnisse sind hilfreich:
Existiert kein (effizienter) Algorithmus, so kann die Suche eingestellt werden.

→ Algorithmen und Berechnungskomplexität II

universitätbonn

Automatentheorie und formale Sprachen

- Was ist ein gültiges Java/C++-Programm?
- Wie funktioniert ein Compiler?
- Wie muss eine formale Sprache (Programmiersprache) aussehen, damit ein effizienter Compiler existiert?

Grammatik:
Regelsystem zur Erzeugung syntaktisch korrekter Programme.

```
WhileStatement:
    while ( Expression ) Statement
BasicForStatement:
    for ( ForInit_opt ; Expression_opt ; ForUpdate_opt ) Statement
```

Automat:
Einfaches Rechnermodell, das bei Compilern zum Einsatz kommt, z.B. für die lexikalische Analyse (Zerlegung des Quelltextes in Tokens).

universitätbonn

Viel Spaß und Erfolg!

© Dr. Thomas Mauersberg / Uni Bonn

universitätbonn