Berechenbarkeit und Komplexität

Vorlesung #1: Was ist ein Problem?

Prof. Berthold Vöcking
präsentiert durch Prof. Joost-Pieter Katoen

16. Oktober 2008

Berechenbarkeit – die absoluten Grenzen des Computers

Kann beispielsweise das folgende Problem durch einen Computer gelöst werden?

Halteproblem

- **Eingabe:** Programm Π in einer wohldefinierten, universellen Programmiersprache (z.B. Pascal, C, Java, Haskell)
- **Frage:** Hält Π auf jede Eingabe?

Wir werden beweisen, dass dieses Problem nicht durch einen Rechner gelöst werden kann.

Komplexität: Welche Probleme können effizient gelöst werden?

Kann beispielsweise das folgende Problem *effizient* durch einen Computer gelöst werden?

Traveling Salesperson (TSP)

- **Eingabe:** Graph G mit Kantengewichten
- **Ausgabe:** ein Hamiltonkreis mit minimalem Gewicht (= günstigste Rundreise)

Unter der Hypothese $P \neq NP$ werden wir zeigen, dass es keinen Polynomialzeitalgorithmus für TSP gibt.

Übersicht

Teil 1: Berechenbarkeit

- Kapitel 1: Was ist ein Problem?
- Kapitel 2: Computermodelle
- Kapitel 3: Nicht rekursive Probleme
- Kapitel 4: Semi-Entscheidbarkeit und Rekursive Aufzählbarkeit
- Kapitel 5: Eigenschaften rekursiver und rekursiv aufzählbarer Sprachen
- Kapitel 6: Die Technik der Reduktion
- Kapitel 7: Klassische Probleme aus der Rekursionstheorie
- Kapitel 8: Mächtigkeit von Programmiersprachen

Übersicht

Teil 2: Komplexität

- Kapitel 1: Die Komplexitätsklasse P
- Kapitel 2: Die Komplexitätsklasse NP
- Kapitel 3: P versus NP
- Kapitel 4: NP-Vollständigkeit
- Kapitel 5: Der Satz von Cook und Levin
- Kapitel 6: NP-Vollständigkeit einiger Graphprobleme
- Kapitel 7: NP-Vollständigkeit einiger Zahlprobleme
- Kapitel 8: Übersicht über die Komplexitätslandschaft
- Kapitel 9: Approximationsalgorithmen für NP-harte Probleme

Übungen

- 11 Übungsgruppen: verschiedene Zeiten
- Spezialübung für Lehramt und den Studiengang "Technik-Kommunikation": Mi 12:00 - 13:30 Uhr (5054)
- Anmeldung Übungsbetrieb über CAMPUS Office bis spätestens **Do 16.10. um 12 Uhr**
  - möglichst viele Prioritäten angeben
- Koordinatoren:
  - Carsten Kern, Stefan Rieger, Alexander Skopalik
- erster Übungszettel: ab Fr 17.10 spätestens 17 Uhr im Web
- Abgabe am folgenden Freitag um 12:00 Uhr im Sammelkasten vor dem Lehrstuhl i1 (oder in der VL)
- Beginn des Übungsbetriebs: Mo 20.10.2008

Wichtige Termine

Vorlesung: Di 8:15 - 9:45, Fr 12:00-13:30 (2-wöchentlich) im Eph

1. Präsenzübung: 12. Dezember 2008 (Vorlesungszeit)

2. Präsenzübung: 27. Januar 2009 (Vorlesungszeit)

Klausur: 16.2.2009 (vm) / 2. Versuch: 26.3.2009 (nm)

Zulassungskriterium Klausur: mindestens 40% der Summe der Punkte, die in den beiden Präsenzübungen erreichbar sind

Webseite: `moves.rwth-aachen.de/i2/248`

Was ist ein *Problem*?

Informelle Umschreibung des Begriffes *Problem*:

für gegebene Eingaben soll der Computer ...
... bestimmte Ausgaben produzieren.

Wir benötigen eine präzisere Definition ...

Alphabete und Wörter

- Ein- und Ausgaben sind Wörter über einem Alphabet Σ.
- Typischerweise $\Sigma = \{0, 1\}$.
- Σ^k ist die Menge aller Wörter der Länge k, z.B.

$$\{0,1\}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

- Das sogenannte *leere Wort*, also das Wort der Länge 0, bezeichnen wir mit ϵ, d.h. $\Sigma^0 = \{\epsilon\}$.
- $\Sigma^* = \bigcup_{k \in \mathbb{N}_0} \Sigma^k$ ist der sogenannte *Kleenesche Abschluss* von Σ und enthält alle Wörter über Σ, die wir z.B. der Länge nach aufzählen können

$$\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, \ldots$$

Problem als Relation

- Im Allgemeinen entspricht ein Problem einer *Relation* $R \subseteq \Sigma^* \times \Sigma^*$.
- Ein Paar (x, y) liegt in R, wenn y eine zulässige Ausgabe zur Eingabe x ist.

Beispiel: Primfaktorbestimmung

Zu einer natürlichen Zahl $q \geq 2$ suchen wir einen Primfaktor.

Wir einigen uns darauf Zahlen Binär zu kodieren. Die Binärkodierung einer natürlichen Zahl i bezeichnen wir mit $bin(i)$.

Die entsprechende Relation ist

$$R = \{(x, y) \in \{0,1\}^* \times \{0,1\}^* \mid x = bin(q), y = bin(p), p, q \in \mathbb{N}, q \geq 2, p \text{ prim}, p \text{ teilt } q\} .$$

Problem als Funktion

- Häufig gibt es zu jeder Eingabe eine eindeutige Ausgabe.
- Dann können wir das Problem als Funktion $f : \Sigma^* \to \Sigma^*$ beschreiben.
- Die zur Eingabe $x \in \Sigma^*$ gesuchte Ausgabe ist $f(x) \in \Sigma^*$.

Beispiel: Multiplikation

Zu zwei natürlichen Zahlen $i_1, i_2 \in \mathbb{N}$ suchen wir das Produkt.

Um die Zahlen i_1 und i_2 in der Eingabe voneinander trennen zu können, erweitern wir das Alphabet um ein Trennsymbol #, d.h. $\Sigma = \{0, 1, \#\}$.

Die entsprechende Funktion $f : \Sigma^* \to \Sigma^*$ ist

$$f(bin(i_1)\#bin(i_2)) \;=\; bin(i_1 \cdot i_2) \;.$$

Entscheidungsprobleme als Sprachen

- Viele Probleme lassen sich als Ja-Nein-Fragen formulieren.
- Derartige *Entscheidungsprobleme* sind von der Form $f : \Sigma^* \to \{0, 1\}$, wobei wir 0 als „Nein“ und 1 als „Ja“ interpretieren.
- Sei $L = f^{-1}(1) \subseteq \Sigma^*$ die Menge derjenigen Eingaben, die mit „Ja“ beantwortet werden.
- L ist eine Teilmenge der Wörter über dem Alphabet Σ. Eine solche Teilmenge wird allgemein als *Sprache* bezeichet.

Entscheidungsprobleme als Sprachen

Beispiel: Graphzusammenhang

Problemstellung: Für einen gegebenen Graphen G soll bestimmt werden, ob G zusammenhängend ist.

Der Graph G liege dabei in einer geeignete Kodierung $code(G) \in \Sigma^*$ vor, z.B. als Binär kodierte Adjazenzmatrix.

Die zu diesem Entscheidungsproblem gehörende Sprache ist

$$L = \{ w \in \Sigma^* \mid \exists \text{ Graph } G\text{: } w = code(G) \text{ und } G \text{ ist zusammenhängend} \} .$$