Algorithmen & Komplexität

Johannes Lengler
Institut für Theoretische Informatik

Zusammenfassung Teil 1

- Beispiele effizienter Algorithmen
  - Sortieren, Median, Matrixmult., Pfade, Spannbaum
- Grundlegende Paradigmen für Algorithmenentwurf
  - Divide & Conquer
  - Dynamische Programmierung
  - Greedy Algorithmen
- Datenstrukturen
  - Suchbäume
  - Union-Find-Strukturen
  - Hashing
  - Fibonacci Heaps

Teil 2

Berechenbarkeit und Komplexität

Berechenbarkeit und Komplexität

Darin enthalten: Welche Aussagen lassen sich mathematisch beweisen?

Berechenbarkeit:

Welche Probleme lassen sich überhaupt berechnen?

Komplexität:

Welche Probleme lassen sich effizient berechnen?

Berechenbarkeit

David Hilbert, 1900:

„Das Ziel, die Mathematik sicher zu begründen, ist auch das meinige; ich möchte der Mathematik den alten Ruf der unanfechtbaren Wahrheit, der ihr durch die Paradoxien der Mengenlehre verlorenzugehen scheint, wiederherstellen; aber ich glaube, dass dies bei voller Erhaltung ihres Besitzstandes möglich ist.“

Falsch

8.Sept. 1930:

"In der Mathematik gibt es kein Ignorabimus! (...) Wir müssen wissen, wir werden wissen!"

Hilberts Zweites Problem:

Sind die arithmetischen Axiome widerspruchsfrei?

Kurt Gödel, 7. Sept. 1930:
Wir werden es niemals wissen!

Berechenbarkeit

$$\{0,1\}^* := \bigcup_{c=0}^{\infty} \{0,1\}^c$$

- Wir betrachten Funktionen der Form $f : \{0,1\}^* \to \{0,1\}$.
- Äquivalent: Betrachten Sprachen $L \subseteq \{0,1\}^*$.
- Das ist etwas anders als in unserem Rechenmodell, dort arbeiten wir mit der Eingabe $\mathbb{Z}^*$ und der Ausgabe $\mathbb{Z}$ (bei Terminierung). Dies ist aber keine Einschränkung.
- Wir können nicht nur ganze Zahlen in $\{0,1\}^*$ kodieren, sondern auch:
  - Tupel
  - Programme
  - mathematische Aussagen (!)

Berechenbarkeit

Definition:

Eine Funktion $f : \{0,1\}^* \to \{0,1\}$ heisst *berechenbar*, falls es ein Programm A gibt, das immer terminiert, und das zu jedem Input $x \in \{0,1\}^*$ den Output $f(x)$ausgibt.

- Andere Modelle für Berechenbarkeit sind möglich. (Turingmaschinen, Lambda-Kalkül, ...)
- Offenbar hängt der Begriff „berechenbar“ vom Modell ab.
- Alle wichtigen Modelle sind äquivalent!

Church-Turing-These:

Die Klasse der intuitiv berechenbaren Funktionen stimmt mit der Klasse der RAM-berechenbaren Funktionen überein.

Berechenbarkeit

Gibt es Funktionen, die nicht berechenbar sind?

JA! (Abzählbarkeit)

Gibt es wichtige Funktionen, die nicht berechenbar sind?

wichtig = lassen sich überhaupt hinschreiben

Halteproblem

Haltefunktion:

$$H(A,x) := \begin{cases} 1, & A \text{ ist ein Programm und } A(x) \text{ terminiert.} \\ 0, & \text{sonst} \end{cases}$$

Theorem:

Die Haltefunktion ist nicht berechenbar.

Beweis:

Annahme: Es gibt ein Programm A_H, das H berechnet.

Wir definieren ein Programm C für Input y durch:

IF A_H(y,y) == 1 THEN gehe in Endlosschleife
ELSE terminate

Falls C(C) terminiert, so ist A_H(C,C) = H(C,C) = 1. 

Falls C(C) nicht terminiert, so ist A_H(C,C) = H(C,C) = 0. 

Erster Gödelscher Unvollständigkeitssatz

Ein Beweissystem ist eine Menge von Operationen (Axiomen), mit denen aus korrekten Aussagen weitere korrekte Aussagen gewonnen werden können.

Beweise sind automatisch verifizierbar!

Formal: Sei S eine Menge an mathematischen Aussagen.

Ein Beweissystem für S ist ein Programm V („Verifizierer“), sodass für jede Aussage $s \in S$ und jedes $p \in \{0,1\}^*$:

$$V(s,p) = \begin{cases} 1, & p \text{ ist ein gültiger Beweis für } s. \\ 0, & \text{sonst} \end{cases}$$

Wir nehmen im Folgenden an, dass V widerspruchsfrei ist:

$$\exists p : V(s,p) = 1 \implies \nexists p' : V(\neg s, p') = 1$$

Erster Gödelscher Unvollständigkeitssatz

Theorem (Erster Unvollständigkeitssatz):

Sei S „hinreichend mächtig“ und V widerspruchsfrei. Dann gibt es ein Programm A und eine Eingabe x, sodass entweder

- A(x) hält, aber die Aussage „A(x) hält“ ist nicht beweisbar, oder
- A(x) hält nicht, aber die Aussage „A(x) hält nicht“ ist nicht beweisbar.

Beweis durch Widerspruch:

Sonst könnten wir folgendes Programm definieren. Für Input (A,x):

FORALL $p \in \{0,1\}^*$

IF V(„A(x) hält“, p) == 1 THEN return 1

IF V(„A(x) hält nicht“, p) == 1 THEN return 0

Dieses Programm würde die Haltefunktion berechnen.

Zweiter Gödelscher Unvollständigkeitssatz

Theorem (Zweiter Unvollständigkeitssatz):

Sei S „mächtig“ und V widerspruchsfrei und „mächtig“. Dann lässt sich die Aussage „V ist widerspruchsfrei“ nicht mit V beweisen.

Beweis:

PROGRAMM **G** für Input A:

FORALL $p \in \{0,1\}^*$: IF V(„A(A) hält nicht“, p) == 1 THEN terminate

Zwei Annahmen über V:

1) „A(x) hält“ => V kann „A(x) hält“ beweisen.

2) V kann 1) beweisen.

Insbesondere kann V beweisen: („G(G) hält“ => V kann „G(G) hält“ beweisen).

Annahme: V kann die eigene Widerspruchsfreiheit beweisen.

$\Rightarrow$ V kann beweisen: („G(G) hält“ => V kann nicht „G(G) hält nicht“ beweisen)

$\Rightarrow$ V kann durch Analyse von G beweisen:
(„G(G) hält“ => „G(G) hält nicht“)

$\Rightarrow$ V kann beweisen: „G(G) hält nicht“

$\Rightarrow$ G(G) hält.

$\Rightarrow$ V kann „G(G) hält“ beweisen. 

Komplexitätstheorie

Wie entscheidet man, ob ein Problem effizient lösbar ist?

Wie beweist man, dass ein Problem nicht effizient lösbar ist?

P und NP

Wir betrachten Funktionen der Form $f : \{0,1\}^* \to \{0,1\}$.

Äquivalent: Betrachten Sprachen $L \subseteq \{0,1\}^*$.
Wir bezeichnen f (bzw. L) auch als *Entscheidungsproblem*.

Definition:

Eine Sprache $L \subseteq \{0,1\}^*$ ist in P, falls es einen Algorithmus A gibt, der in polynomieller Zeit mit A(x)=L(x) terminiert.

Eine Sprache L ist in NP (*nicht-deterministisch polynomiell*), falls es einen Verifizierer V gibt, sodass V in polynomieller Zeit terminiert, und sodass

$$L(x) = 1 \iff \exists p \in \{0,1\}^* \text{ mit } |p| = \text{poly}(|x|) : V(x,p) = 1$$

und

$$L(x) = 0 \iff \forall p \in \{0,1\}^* : V(x,p) = 0.$$

Probleme in P

Probleme in P:

- ZUSAMMENHANG
  $L = \{G \mid \ G \text{ zusammenhängender Graph}\}$
- DURCHMESSER
  $L = \{(G, b) \mid G \text{ Graph}, b \in \mathbb{R}, \text{ alle Knotenpaare haben Distanz} \leq b\}$
- PLANARITÄT
  $L = \{G \mid G \text{ kann “überkreuzungsfrei” in der Ebene gezeichnet werden}\}$
- PRIMZAHLEN
  $L = \{p \in \mathbb{N} \mid p \text{ prim}\}$

Probleme in NP

Probleme in NP:

- alle Probleme in P

- HAMILTONKREIS
  $L = \{G \mid G \text{ hat einen Kreis, der jeden Knoten genau einmal besucht}\}$
- CLIQUE
  $L = \{(G, m) \mid \text{Es gibt } m \text{ paarweise adjazente Knoten in } G\}$
- GRAPH-ISOMORHPHISMUS
  $L = \{(G, H) \mid G \text{ und } H \text{ sind isomorphe Graphen}\}$
- ERFÜLLBARKEIT
  $L = \{F \mid F \text{ ist eine Formel, die über } \mathbb{Z} \text{ erfüllbar ist}\}$
  z.B.: $x^2 + y^2 = z^2 \in L, \quad x^5 + y^5 = z^5 \notin L$
- SAT (SATISFIABILITY)
  $L = \{F \mid F \text{ ist erfüllbare aussagenlogische Formel}\}$
  z.B.: $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_1) \wedge (\overline{x}_2) \notin L$