NP-Vollständigkeit

Krautgartner Martin (9920077)
Markgraf Waldomir (9921041)
Rattensberger Martin (9921846)
Rieder Caroline (0020984)

Übersicht:

- Einleitung
- Einteilung in Klassen
- Die Klassen *P* und *NP*
- NP-Vollständigkeit (allgemein)
- Entscheidungsproblem des Pfades
- Verifikation in Polynomialzeit
- $P \stackrel{?}{=} NP$
- Reduzierbarkeit
- NP-Vollständigkeit (Definition)
- Wie beweist man, daß ein Problem NP-Vollständig ist
- Das Travelling-Salesman-Problem
- Lösung durch Approximation
- Schlußwort

NP-Vollständigkeit (NP Completeness)

- Zentrales Thema der Komplexitätstheorie
- Frage nach dem vertretbaren Aufwand
- KEINE exakte Bestimmung der Komplexität
- Praktisch gesehen heißt das: Ein NP-vollständiges Problem ist ein Problem, das auf dem Computer nur lösbar ist, wenn man gewillt ist, extrem lange auf die Lösung zu warten.

Einteilung in Klassen

- Probleme die mit polynomiellen Zeitaufwand gelöst werden können. O(n^k) für $k \geq 0$; noch vertretbarer Zeitaufwand

- Probleme die nur mit exponentiellen Zeitaufwand gelöst werden können. O(k^n) für $k \geq 2$; Zeitaufwand zu hoch unbrauchbar

Unterteilung der polynomiellen Probleme

- **P** - deterministisch polynomiell lösbaren Probleme
  Z.B. Sortieren

- **NP** - nichtdeterministisch lösbare Probleme
  Z.B. TSP, Minesweeper, ...

NP-Vollständigkeit (NP Completeness)

- NP-vollständige Probleme bilden die obere Schranke in der Klasse NP, also keine Problemstellung in NP ist schwerer als ein NP-vollständiges Problem.

- NP-Vollständigkeit besagt dass für das betrachtete Problem (bisher) kein polynomieller Algorithmus gefunden wurde. Die Lösung dieser Aufgabe erfordert also einen nicht vertretbaren Aufwand. (unter Annahme dass $P \neq NP$ wird das Problem nur deterministisch exponentiell lösbar sein)

Entscheidungsproblem des Pfades

- Aus dem Blickwinkel der formalen Sprache ist die Menge I der Instanzen eines konkreten Entscheidungsproblem Q gleich $\sum^{'}$ mit $\sum = \{0, 1\}$

- Ein Problem Q, das vollständig durch die Menge der Instanzen charakterisiert werden kann, wobei vorausgesetzt wird, dass es sich dabei um das Resultat 1 handelt, können wir Q auch als folgende Sprache interpretieren:

$$L = \{x \in \{0, 1\} \mid Q(x) = 1\}$$

- Beispiel: Das Entscheidungsproblem PFAD

$$PFAD = \{(G, u, v, k) \mid G = (V, E)\text{ungerichteter Graph.}$$

$$u, v \in Vk \geq 0\}$$

- Definition (6):
  Ein Algorithmus akzeptiert einen String $x \in \{0,1\}$, falls er für x $A(x) = 1$ berechnet. Die Sprache ist dann gleich.

$$L = \{x \in \{0,1\} \mid A(x) = 1\}$$

  Ein Algorithmus lehnt den String x ab, falls $A(x) = 0$

- Definition (7):
  Eine Sprache L wird durch einen Algorithmus A entschieden, wenn jeder String (der zu L gehört) von A akzeptiert wird und andere abgelehnt werden.

- Eine Sprache L wird in Polynomial-Zeit von einem Algorithmus A **akzeptiert** falls die Sprache von A akzeptiert wird und falls es eine Konstante k gibt, so daß jeder Eingabestring $x \in L$ der Länge n von A in der Zeit $O(n^k)$ akzeptiert wird.

- Eine Sprache L wird in Polynomial-Zeit von einem Algorithmus A **entschieden** falls die Sprache durch A entschieden wird und falls es eine Konstante k gibt, so daß jeder Eingabestring $x \in L$ der Länge n von A in der Zeit $O(n^k)$ korrekt bestimmt, ob x in L ist oder nicht.

- Beispiel:
  Die PFAD-Sprache kann in Polynomial-Zeit akzeptiert werden. Der Algorithmus arbeitet wie folgt:

  – Überprüfung ob G einen ungerichteten Graphen codiert

- Beispiel:
  Die PFAD-Sprache kann in Polynomial-Zeit akzeptiert werden. Der Algorithmus arbeitet wie folgt:

  – Überprüfung ob G einen ungerichteten Graphen codiert

  – Überprüfung ob u und v Knoten in G sind

- Beispiel:
  Die PFAD-Sprache kann in Polynomial-Zeit akzeptiert werden. Der Algorithmus arbeitet wie folgt:

  - Überprüfung ob G einen ungerichteten Graphen codiert
  - Überprüfung ob u und v Knoten in G sind
  - Berechne den kürzesten Pfad p von u nach v

- Beispiel:
  Die PFAD-Sprache kann in Polynomial-Zeit akzeptiert werden. Der Algorithmus arbeitet wie folgt:

  - Überprüfung ob G einen ungerichteten Graphen codiert
  - Überprüfung ob u und v Knoten in G sind
  - Berechne den kürzesten Pfad p von u nach v
  - Vergleiche die Zahl der Kanten [p] auf dem kürzesten Pfad mit k

- Beispiel:
  Die PFAD-Sprache kann in Polynomial-Zeit akzeptiert werden. Der Algorithmus arbeitet wie folgt:

  - Überprüfung ob G einen ungerichteten Graphen codiert
  - Überprüfung ob u und v Knoten in G sind
  - Berechne den kürzesten Pfad p von u nach v
  - Vergleiche die Zahl der Kanten [p] auf dem kürzesten Pfad mit k
  - Gib 1 aus, falls obige Bedingungen erfüllt sind und die Länge des kürzesten Pfades $\leq k$ ist.

- Definition (8):
  Eine Komplexitätsklasse P umfasst alle konkreten Entscheidungsprobleme, die in Polynomial-Zeit lösbar sind.

  $P = \{L \subseteq \{0,1\} \mid$
  es existiert ein Algorithmus A, der L in PZ **entscheidet**$\}$
  Satz(1):

  $P = \{L \subseteq \{0,1\} \mid$
  es existiert ein Algorithmus A, der L in PZ **akzeptiert**$\}$

- Beweis:
  Sei L eine Sprache, die durch einen PZ-Algorithmus A akzeptiert wird. Da es eine Konstante k gibt, so dass A die Sprache L in der Zeit $O(n^k)$ akzeptiert, gibt es auch eine Konstante c, so dass A die Sprache L in höchstens $T = cn^k$ Schritten akzeptiert.
  Der Algorithmus A' simuliert den Algorithmus A für einen Eingabe String x T Schritte lang. Am Ende dieser T Schritte inspiziert A' das Verhalten von A. Falls A akzeptiert hat, akzeptiert auch A' und gibt 1 aus.

Verifikation in Polynomial-Zeit:

- Es gibt Entscheidungsprobleme für die kein Polynomial-Zeit Algorithmus bekannt ist. Ist jedoch ein Zertifikat gegeben, so ist die Verifikation einfach.

- **Beispiel:**
  Gegeben sei ein ungerichteter Graph $G = (V, E)$. Es gibt einen Kreis in G, der jeden Knoten in V besucht. Graphen mit dieser Eigenschaft nennt man *Hamilton-Graphen*.

$$HAM - Kreis = \{(G) \mid \text{G ist ein Hamilton-Graph}\}$$

- Behauptung, dass ein bestimmter Graph ein Hamilton Graph ist. Um sie zu belegen gibt die Person, die es behauptet hat einen Pfad p als Zertifikat an.

- ob jeder Knoten genau einmal in p vorkommt. Zeit: $O(n)$

- ob jedes aufeinander folgendes Paar v_i, v_{i+1} von Knoten auf dem Pfad durch eine Kante in G miteinander verbunden ist. Zeit $O(n)$

- Definition (9):
  Ein Verifikations-Algorithmus ist ein Algorithmus, der zwei Argumente besitzt. Das erste Argument ist der Eingabestring x und das zweite Argument ist ein binärer String y das sogenannte Zertifikat.
  A verifiziert den Eingabestring x, falls es ein Zertifikat y gibt, so dass $A(x, y) = 1$

Bedeutende Frage in der Informatik

- $P \stackrel{?}{=} NP$
- bisher weder bewiesen noch widerlegt
- Clay Mathematics Institute: mit 1 Million US$ dotiert

- Annahme : $P \neq NP$
  - Grund:
    - **Existenz der NP-vollständigen Probleme**

      Existiert für ein NP-vollständiges Problem ein Polynomial-Zeit-Algorithmus,
      dann gibt es für alle Probleme in NP einen Polynomial-Zeit-Algorithmus, d.h., $NP = P$

Reduzierbarkeit

Eine Sprache L1 ist in Polynomial-Zeit auf eine Sprache L2 reduzierbar, falls es eine in Polynomial-Zeit berechenbare Funktion $f : \{0,1\} \rightarrow \{0,1\}$ gibt, so dass für alle $x \in \{0,1\}$ gilt:

$$x \in L1 \Longleftrightarrow f(x) \in L2$$

Schreibweise: $L1 \leq_p L2$

f.....Reduktions-Funktion
PZ-Algorithmus(f).....Reduktions-Algorithmus

NP-Vollständigkeit

Definition:
Eine Sprache $L \subseteq \{0,1\}$ ist NP-vollständig, wenn

(1)$L \in NP$
(2)$L' \leq_p L$ fr jede Sprache $L' \in NP$

NP-Vollständigkeit

Satz:

- Falls irgend ein NP-vollständiges Problem in PZ lösbar ist, dann ist $P = NP$.

- Falls irgend ein Problem in NP nicht in Polynomial-Zeit lösbar ist, dann ist kein NP-vollständiges Problem in PZ lösbar.

Beweis

Sei $L \in P$ und sei L NP-vollständig.
Fr jedes $L' \in NP$ existiert eine Reduktion auf L : $L' \leq_p L$.
Also ist auch $L' \in P$.

Wie beweist man, dass ein Problem NP-vollständig ist?

Grundlegendes Problem bzw. Menge von Problemen
(NP-vollständig)

↓

Prüfung, ob das neue Problem in NP enthalten ist

↓

Suche nach NP-vollständigem Problem,
das auf das neue Problem reduziehrt werden kann.

Das Travelling-Salesman-Problem (TSP)

- sehr bekanntes Problem der Graphentheorie

- Problembeschreibung:
  - n Städte und ein Handelsreisender
  - **Gesucht:** Die kürzeste Route durch alle Städte, wobei jede Stadt nur **einmal** besucht werden darf!
  - **Problem:** Bei z.B. 190 Städte gäbe es 190! mögliche Routen (= ca. $9,6803226752552491 5 * \underline{\underline{10^{353}}}$ Möglichkeiten!!)

Das TSP ist NP-vollständig:

- **Formaler:**

  Das **TSP** kann als ungerichteter Graph(G), (der min. einen Kreis haben muss, der alle Knoten beinhaltet) betrachtet werden.

  Die *Knoten*(V) entsprechen den Städten
  Die *Kanten*(E) entsprechen den Verbindungen zw. den Städten
  Die *Kantengewichte*(c) entsprechen der Reisezeit, Streckenlänge, o.ä.

- Somit ist das TSP äquivalent zur Suche nach kürzestem Hamiltonkreis

- Durch die Reduktion des TSP auf die Suche des kürzesten Hamiltonkreises (welche NP-vollständig ist), ist gezeigt, daß auch das Travelling-Salesman-Problem **NP-vollständig** ist.

Lösung des TSP durch Approximation

- Lösung von Problemen in NP durch Approximation
  - Annäherung an Lösung, bis Ergebnis gewünschte Genauigkeit hat
  - liefert eine *fast* optimale Lösung
- Konkret für unser TSP: Lösung mittels **Minimum-Spanning-Tree**
  - Wir benötigen dafür die *Dreiecksungleichung*, welche wir für Erfüllt annehmenn d.h.,für alle Knoten $u, v, w \in V$ gilt:

$$c(u, v) \leq c(u, w) + c(w, v)$$

Lösung des TSP mit Hilfe des Minimum-Spanning-Tree)

1. Mache einen Knoten $r \in V$ zum Wurzelknoten

2. Berechne einen Minimum-Spanning-Tree für G von der Wurzel r

3. Sei L die Liste von Knoten, die in einem Preorder-Tree-Weg besichtigt werden

4. Gib den Hamilton-Kreis zurück, der die Knoten in der Reihenfolge von L besucht (Beim Streichen der doppelten Punkte wird implizit die Dreiecksungleichung angewandt)

Fazit

- NP-Vollständigkeit: Ein ungelöstes Problem der Informatik, das auf seine Lösung wartet!

Quellenangaben

- Clay Mathematics Institute. Millennium Prize Problems. http://www.claymath.org/prizeproblems/index.htm.

- NP-vollstndige Probleme. Ein Artikel fr das PS Wissenschaftliches Arbeiten. WS 2001/02
  Hannes Eder

- Universitt Trier
  Fachbereich IV - Informatik
  Christoph Meinel: Komplexittstheorie
  Vorlesungsskript Sommersemester 2000

- Using Interactive Visualization for Teaching the Theory of NP-completeness
  Christian Pape
  Universitt Karlsruhe
  Institut fr Logik, Komplexitt und Deduktionssysteme

- Proseminar Genetische und Evolutionäre Algorithmen SS2002
  Anwendung: Das Travelling Salesman Problem
  Matthias Raab
  www.informatik.uni-ulm.de/ni/Lehre/SS02/Evosem/tsp.pdf