NP-Vollständigkeit

Dominik Fakner, Richard Hentschel, Hamid Tabibian,
den 20.01.2012

Inhalt

- **Was ist NP?**
  - Definitionen
- **NP-Vollständigkeit**
  - Definition
  - Nachweis
- **Beispiel Reduktion**
- **Quellen**

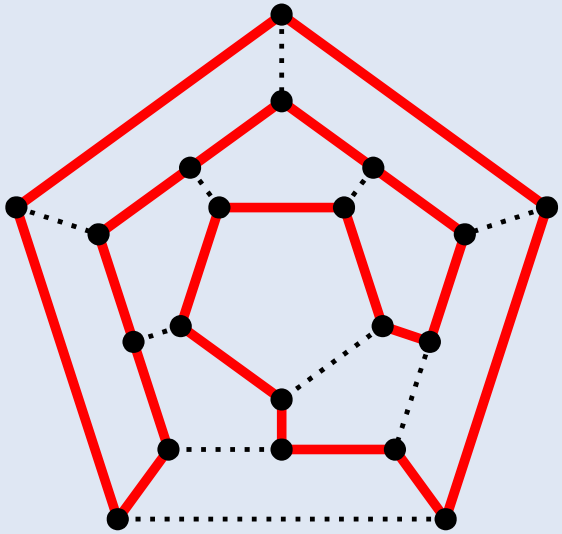
Was ist NP?

- Komplexitätsklasse
- Befasst sich mit Komplexität von algorithmisch behandelbaren Problemen
- Enthält Probleme, die von einer _____ Maschine in _____ Rechenzeit gelöst werden können

Definition Nichtdeterminismus

- Berechnung einer Eingabe mittels Algorithmen oder Maschinen
- Verschiedene Möglichkeiten an das Ergebnis zu kommen
- **Beispiel:**

  Enthält ein gegebener Graph einen Hamiltonkreis?



- Folge $i_1,\ldots,i_{|V|}$ von $|V|$ Zahlen aus Menge $\{1,\ldots, |V|\}$
- $\{i_1,i_2\},\{i_2,i_3\},\ldots,\{i_{|V|},i_1\}$

Definition Polynomialzeit

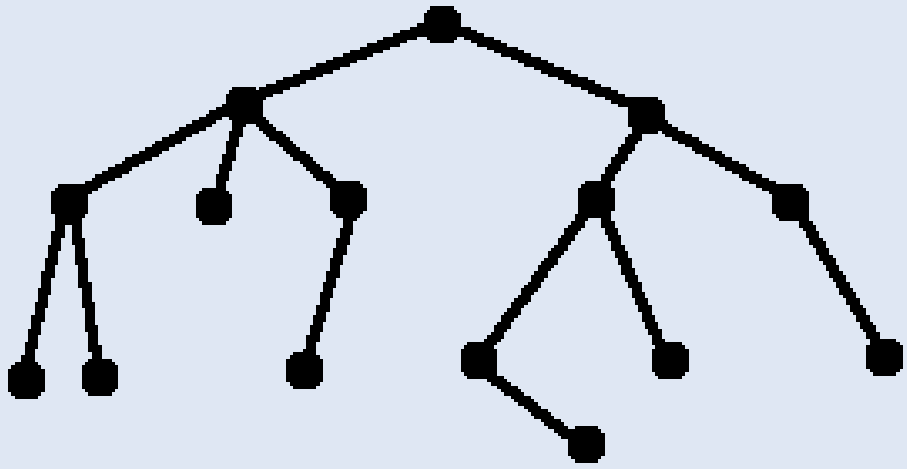
- Ein Problem wird als in Polynomialzeit lösbar bezeichnet, wenn die benötigte Rechenzeit in Bezug auf die Problemgröße nicht stärker als mit einer Polynomfunktion wächst

- Grenze zwischen praktisch lösbar und nicht lösbar

Definition Polynomialzeit

- Ein Polynomialzeitalgorithmus spaltet die Arbeit in viele parallele Rechenwege

- Überprüft jedes potenzielle Ergebnis auf Korrektheit



NP-Klasse

- **Enthält Probleme, die von einer nichtdeterministischen Maschine in Polynomialzeit gelöst werden können**
- **Polynomialzeit als obere Grenze**
- **Alternative Definition:**
  - eine gegebene Lösung kann von einer deterministischen TM in Polynomialzeit überprüft werden
- **Satz von Fagin:**
  - L ∈ NP gdw ∃ Satz der existentiellen Logik zweiter Stufe, der L beschreibt

Sprachakzeptanz

- **L ∈ NP gdw ∃ nichtdeterministische Maschine M und ein Polynom p gibt, sodass gilt:**
  - Bei Eingabe von x hält M nach höchstens p(|x|) Schritten
  - x ∈ L gdw es mindestens einen akzeptierenden Lauf von M auf x gibt

Suchproblemdefinition

- $L \in NP$ gdw $\exists R_L \subseteq \{0,1\}^* \times \{0,1\}^*$ und ein Polynom p, sodass gilt:
  - R_L wird von einer deterministischen und polynomiell zeitbeschränkte Maschine erkannt
  - $x \in L$ gdw $\exists y$ mit $|y| \leq p(|x|)$ und $(x,y) \in R_L$

Äquivalenz der Definitionen

- Existiert eine nichtdeterministische Maschine M, die L erkennt, so gibt es zu jedem $x \in L$ eine akzeptierende Rechnung von M, welche sich in einen String $a_M(x)$ kodieren lässt. $\forall x \in L$ gilt $R_L = (x, a_M(x))$ und erfüllt obige Eigenschaften.

- Gibt es umgekehrt eine Relation R_L nach obiger Definition, so kann eine nichtdeterministische Maschine M konstruiert werden, die ein entsprechendes y zunächst nichtdeterministisch rät, und dann mittels einer deterministischen Maschine überprüft, ob $(x,y) \in L$ also $x \in L$

NP-Klasse

- **Abgeschlossen unter:**
  - Vereinigung
  - Durchschnitt
  - Konkatenation
  - Kleene-Stern
  - Epsilon-freien Homomorphismen
  - inversen Homomorphismen

NP-Vollständigkeit

- **Satz von Cook**
  - Es existiert eine Teilmenge der Klasse NP, auf die sich alle Probleme aus NP polynomiell reduzieren lassen. Diese Teilmenge wird als NP-vollständig bezeichnet.
- **Definition**
  - Ein Entscheidungsproblem L heißt NP-vollständig gdw.:
    - L in der Klasse NP liegt, dh. $L \in NP$
    - L NP-schwierig ist, dh. $\forall L' \in NP: L' \leq_p L$

Polynomielle Reduktion

- Lösung eines Problems mit Hilfe eines Algorithmus für ein anderes Problem
- **Reduzierbarkeit** → Relation zwischen zwei Problemen
- **Bsp:**
  - $L_Q := \{(a, b) : b = a^2, a \in Z\}$
  - $L_M := \{(a_1,a_2,b) \in Z^3 : b = a_1 * a_2\}$
  - $f : Z^2 \rightarrow Z^3, (a,b)$
- f muss in Polynomialzeit berechenbar sein

Nachweis

- **Eigenschaften:**
  - $L \in NP$

    Raten der Lösung und zeigen, dass diese in Polynomialzeit verifizierbar ist

  - $\forall L' \in NP: L' \leq_p L$

    Durch Reduktion von einem ähnlichen, bekannten auf das aktuelle Problem

Starke und schwache NP-Vollständigkeit

- Sei p ein Polynom und n die Eingabelänge
- Die Zahlenwerte der Eingabe werden durch p(n) beschränkt

- **L ist immer noch NP-vollständig**
  - Stark NP-vollständig
- **L ist nicht mehr NP-vollständig**
  - Schwach NP-vollständig
    - Es existiert ein pseudopolynomieller Algorithmus

Polynomialreduktion

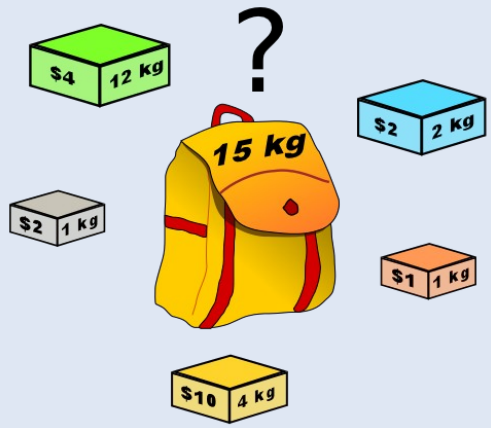
- **SAT**
  - Erfüllbarkeitsproblem der Aussagenlogik
  - Frage ob eine Aussagelogische Formel erfüllbar ist
- **3SAT**
  - SAT mit Disjunktion von höchstens 3 Literalen einer Klausel
  - $F=(v_1 \vee \neg v_3 \vee v_5) \wedge (\neg v_1 \vee v_5 \vee v_4) \wedge (\neg v_2 \vee \neg v_2 \vee \neg v_5)$
  - **Annahme:** 3SAT ist NP-Vollständig

Reduktion

- **Rucksack:** $=\{(b,a_1,\ldots,a_k) \in N^{k+1} \mid \exists I \subseteq \{1,\ldots k\} \text{ mit } \Sigma_{i \in I} a_i = b\}$
- **Annahme:** Rucksack ist in NP
- **Zz:** Funktion f: 3KNF → N*
- b = 444…44 111…111

  m n
- m=Klauseln und n=Variablen
- **Also:** b=444 11111
- **Außerdem sei:**

$$F = (v_1 \vee \neg v_3 \vee v_5) \wedge (\neg v_1 \vee v_5 \vee v_4) \wedge (\neg v_2 \vee \neg v_2 \vee \neg v_5)$$



Reduktion

- **Nun definieren wir für jedes v ein x und ein x' in der Form mmm nnnnn:**
  - Die i-te Position gibt an, ob die jeweilige Variable in der i-ten Klausel vorkommt.
  - Für die positiven Vorkommen werden x definiert und für die negativen x'
  - falls i = Index v, m+i-te Position = 1
  - Für jedes v_i wird x_i oder x_i' gewählt

Reduktion

- $F = (v_1 \vee \neg v_3 \vee v_5) \wedge (\neg v_1 \vee v_5 \vee v_4) \wedge (\neg v_2 \vee \neg v_2 \vee \neg v_5)$
  - x1 = 100 10000 x1' = 010 10000
  - x2 = 000 01000 x2' = 002 01000
  - x3 = 000 00100 x3' = 100 00100
  - x4 = 010 00010 x4' = 000 00010
  - x5 = 110 00001 x5' = 001 00001

  - c_1 = 100 00000 d_1 = 200 00000
  - c_2 = 010 00000 d_2 = 020 00000
  - c_3 = 001 00000 d_3 = 002 00000

Reduktion

- **Eine Lösung raten und auf Korrektheit prüfen**

$v_1 \rightarrow 1 \quad \rightarrow x_1 \quad 100\ 10000$

$v_2 \rightarrow 0 \quad \rightarrow x_2' \quad 002\ 01000$

$v_3 \rightarrow 0 \quad \rightarrow x_3' \quad 100\ 00100$

$v_4 \rightarrow 1 \quad \rightarrow x_4 \quad 010\ 00010$

$v_5 \rightarrow 0 \quad \rightarrow x_5' \quad \underline{001\ 00001}$

$213\ 11111$

Reduktion

- **Durch hinzufügen von c und d auf b bringen**

213 11111

c_2 = 010 00000

c_3 = 001 00000

d_1 = 200 00000

d_2 = 020 00000

b = 444 11111

also $b = x_1 + x_2' + x_3' + x_4 + x_5' + c_2 + c_3 + d_1 + d_2$

- falls Problem in polynomieller Zeit lösbar --> Polynomialzeitreduktion --> Rucksack NP-vollständig

Quellen

- Einführung in die Theoretische Informatik
  Klaus W. Wagner, Springer-Verlag, 1994
- www.inf.fh-brs.de/informatikmedia/downloads/Personen/witt/3SAT_RUCKSACK.pdf
- http://lrb.cs.uni-dortmund.de/~martens/data/GTI10/19-Reduktionenw.pdf
- de/en.wikipedia.org