Informatik III - Tutorium XIII & XV (SR -107)

Weihnachtstutorium – Üb7, weitere Komplexitätsklassen

David Münch

Universität Karlsruhe (TH)
Institut für Informatik
ITI Wagner

18. Dezember 2007

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Inhaltsverzeichnis

1 Auftakt

Inhaltsverzeichnis

1. Auftakt
2. Lernziele

Inhaltsverzeichnis

1. Auftakt
2. Lernziele
3. Themen
   - Übungsblatt 7
   - Komplementsprachen
   - Weitere Komplexitätsklassen
   - $\mathcal{NP}$-schwer

Inhaltsverzeichnis

1. Auftakt
2. Lernziele
3. Themen
   - Übungsblatt 7
   - Komplementsprachen
   - Weitere Komplexitätsklassen
   - $\mathcal{NP}$-schwer
4. Abspann

Organisatorisches

Email: muenchdavid@gmail.com

https://www.stud.uni-karlsruhe.de/~uhbro/

Tutorium 13: Mittwochs 8:00 Uhr - Raum -107
Tutorium 15: Mittwochs 9:45 Uhr - Raum -107

Übungsblattabgabe Donnerstag.

Was wollen wir heute erreichen?

Was wollen wir heute erreichen?

- Zusammenhang zwischen den verschiedenen Komplexitätsklassen verstehen.

Was wollen wir heute erreichen?

- Zusammenhang zwischen den verschiedenen Komplexitätsklassen verstehen.
- $\mathcal{NP}$-schwere Probleme kennen lernen.


ICH HABE EINE MASCHINE ERFUNDEN, DIE DEN KOMMERZ AUS WEIHNACHTEN ENTFERNT.
LASS UNS GANZ VIELE DAVON BAUEN UND FÜR VIEL GELD VERKAUFEN.
WWW.NICHTLUSTIG.DE

$\mathcal{NP}$-vollständig

Definition: $\mathcal{NP}$-vollständig

Eine Sprache L heißt $\mathcal{NP}$-**vollständig**, falls gilt:

1. $L \in \mathcal{NP}$ und
2. für alle $L' \in \mathcal{NP}$ gilt: $L' \propto L$

Aufgabe 7

Zeige, dass TRAVELING SALESMAN $\mathcal{NP}$-vollständig ist.
Reduziere hierbei von HAMILTONIAN CIRCUIT.

Aufgabe 7

Problem **HAMILTONIAN CIRCUIT**:
Gegeben: Ein Graph $G = (V, E)$
Frage: Existiert ein einfacher Kreis in G, der alle Knoten in V enthält, also eine Folge $(v_1, \ldots, v_n)$ von paarweise verschiedenen Knoten $v_i \in V (i = 1, \ldots, n)$ mit $n := |V|$ und $\{v_j, v_{j+1}, \{v_n, v_1\} \in E (j = 1, \ldots, n-1)\}$?
Hinweis: HAMILTONIAN CIRCUIT ist $\mathcal{NP}$-vollständig.

Problem **TSP**:
Gegeben: Ein Graph $G = (V, E)$ und Parameter k
Frage: Existiert ein einfacher Kreis in G, der alle Knoten in V enthält, also eine Folge $(v_1, \ldots, v_n)$ von paarweise verschiedenen Knoten mit Kostenfunktion $c \leq k$?

1. Schritt: zu zeigen: **TSP** liegt in $\mathcal{NP}$

Rate Lösung (Folge von Knoten) und überprüfe. $\mathcal{O}(n)$.

1. Schritt: zu zeigen: **TSP** liegt in $\mathcal{NP}$
Rate Lösung (Folge von Knoten) und überprüfe. $\mathcal{O}(n)$.

2. Schritt: **HAMILTONIAN CIRCUIT** $\propto$ **TSP**
Instanz von **HC**: $G = (V, E), V = \{v_1, \ldots, v_n\}$
Die korrespondierende Instanz von **TSP** enthält n Orte, die Gewichte c_{ij} sind 0, wenn $(v_i, v_j) \in E$ und 1 sonst. Kostengrenze k ist n.
Offensichtlich ist diese Transformation polynomial.
Damit gilt:
G enthält einen **HC** genau dann, wenn c eine Rundreise mit Kosten kleiner n zulässt. Sei $(v_{i_1}, \ldots, v_{i_n}, v_{i_1})$ ein **HC** in G, dann sind dessen Kosten gleich n.
Sei $(v_{i_1}, \ldots, v_{i_n}, v_{i_1})$ eine Rundreise mit Kosten n, dann gilt für $i = 1, \ldots, n-1$, dass $(v_i, v_{i+1}) \in E$ und $(v_n, v_1) \in E$. Damit ist $(v_{i_1}, \ldots, v_{i_n}, v_{i_1})$ ein **HC** in G.

Aufgabe 8

Betrachte das Problem $\mathrm{HALFCLIQUE}$:
Gegeben: Graph $G = (V, E)$ mit $|V|$ gerade.
Frage: Gibt es eine Clique der Grösse $|V|/2$ in G?
Zeige, dass $\mathrm{HALFCLIQUE}$ $\mathcal{NP}$-vollständig ist.

Aufgabe 8

Betrachte das Problem $\mathrm{HALFCLIQUE}$:
Gegeben: Graph $G = (V, E)$ mit $|V|$ gerade.
Frage: Gibt es eine Clique der Grösse $|V|/2$ in G?
Zeige, dass $\mathrm{HALFCLIQUE}$ $\mathcal{NP}$-vollständig ist.

Problem **CLIQUE**:
Gegeben: Ein Graph $G = (V, E)$ und einen Parameter $K \leq |V|$
Frage: Gibt es in G eine Clique der Grösse mindestens k? Hinweis: CLIQUE ist $\mathcal{NP}$-vollständig.

Aufgabe 8

The problem is NP-complete.

To show that it is in NP, we give a non-deterministic polynomial-time algorithm for recognizing "yes" instances. This algorithm non-deterministically guesses a set of $n/2$ vertices, and then checks whether it is a clique.

To show that it is NP-hard, we give a reduction from the NP-complete problem **Clique**. Let (G, j) be an instance of **Clique**. (Check your lecture notes to remind yourself about this problem.) We will construct an instance G' of **HalfClique** such that (G, j) is a "yes" instance of **Clique** iff G' is a "yes" instance of **HalfClique**.

We do the construction in two cases. Let $V(G)$ be the set of vertices of G, so $|V(G)|$ is the number of vertices of G.

Aufgabe 8

Case 1: If $j \leq |V(G)|/2$ then we will construct a graph G' with $2|V(G)| - 2j$ vertices by adding $|V(G)| - 2j$ vertices to G. Connect each of these new vertices to every vertex in G' (except itself). We claim that G has a clique of size j iff G' has a clique of size $|V(G)| - j$. So (G, j) is a "yes" instance of **Clique** iff G' is a "yes" instance of **HalfClique**. Here is why the claim is true.

1. If G' has a clique of size $|V(G)| - j$ then at least j of these are from G and these are a clique.

2. If G has a clique of j then these plus the new vertices make a clique of size $|V(G)| - j$ in G'.

Case 2: If $j > |V(G)|/2$ then construct a graph G' with $2j$ vertices by adding $2j - |V(G)|$ new vertices and no new edges. We claim that G has a clique of size j iff G' has a clique of size j. So (G, j) is a "yes" instance of **Clique** iff G' is a "yes" instance of **HalfClique**.

Aufgabe 9

Ein Vertex-Cover eines Graphen $G = (V, E)$ ist eine Teilmenge $V' \subseteq V$, so dass jede Kante aus E zu mindestens einem Knoten aus V' inzident ist. Das Problem $\mathrm{VERTEX\ COVER}$ besteht nun darin zu entscheiden, ob es für einen Graphen $G = (V, E)$ und einen Parameter $k \leq |V|$ ein Vertex-Cover der Grösse höchstens k gibt.
Zeige, dass $\mathrm{VERTEX\ COVER}$ $\mathcal{NP}$-vollständig ist.

Aufgabe 9

Beweis. 1) Wir zeigen zunächst, dass VC in NP liegt. Gegeben sei hierzu ein Graph $G = (V, E)$ und ein $k \in \mathbb{N}$. Ein Zeuge ist $V' \subset V$. Der Verifikationsalgorithmus bestätigt, ob $|V'| = k$ und überprüft, ob für jeder Kante$(u, v) \in E$ gilt, dass $u \in V'$ oder $v \in V'$. Dies kann offenbar in $\mathcal{O}(|E|)$ durchgeführt werden. 2) Nun zeigen wir, dass VC NP-schwer ist, indem wir CLIQUE darauf reduzieren, d.h. CLIQUE$\leq_p$VC
(Zur Erinnerung: CLIQUE
Gegeben: Ein ungerichteter Graph $G = (V, E)$
Frage: Existiert ein vollständiger Teilgraph (Clique) der Grösse k?)

In der Reduktion konstruieren wir in polynomieller Zeit aus einer Eingabe (G, k) für CLIQUE eine Eingabe (G', k') für VC, sodass gilt: G enthält eine k-Clique $\Leftrightarrow$ G' hat einen VC der Grösse k'. Wir wählen als G' das Komplement $\bar{G}$ von G und $k' = n - k$ mit $n = |V|$. (Erinnerung: $\bar{G} = (V, \bar{E})$ mit $\bar{E} = \{(u, v) | (u, v) \notin E\}$). Dies ist in polynomieller Zeit möglich. Es bleibt nun also noch zu zeigen: G enthält eine k-Clique $\Leftrightarrow$ $\bar{G}$ hat einen VC der Grösse $n-k$. Wir betrachten hierzu $V' \subset V$, eine k-Clique von G. Per Definition heißt das: $\forall u, v \in V' : (u, v) \in E$ und $|V'| = k$. Da $\bar{E}$ das Komplement von E ist, bedeutet dies nichts anderes als: $\forall u, v \in V' : (u, v) \notin \bar{E}$ und $|V'| = k$. Das heißt, entweder u oder v liegen nicht in V', $\forall (u, v) \in \bar{E} : u \in V \setminus V'$ oder $v \in V \setminus V'$ (oder Beides) und $|V \setminus V'| = n-k$. Es wird also jede Kante aus $\bar{E}$ von $V \setminus V'$ bedeckt, was genau der Definition eines VC entspricht: $V \setminus V'$ ist $n - k$-VC in $\bar{G}$.


BULIMIE
FÜR
ANFÄNGER
WWW.NICHTLUSTIG.DE

Definition: Komplexitätsklasse $\mathcal{NPC}$

Klasse der $\mathcal{NP}$-vollständigen Sprachen bzw. Probleme.

Definition: Komplexitätsklasse $\mathcal{NPC}$

Klasse der $\mathcal{NP}$-vollständigen Sprachen bzw. Probleme.

Definition: Komplexitätsklasse **co** $-\mathcal{P}$

Klasse aller Sprachen $\Sigma^* \backslash L$ für $L \subseteq \Sigma^*$ und $L \in \mathcal{P}$.

Definition: Komplexitätsklasse $\mathcal{NPC}$

Klasse der $\mathcal{NP}$-vollständigen Sprachen bzw. Probleme.

Definition: Komplexitätsklasse **co** $-\mathcal{P}$

Klasse aller Sprachen $\Sigma^* \backslash L$ für $L \subseteq \Sigma^*$ und $L \in \mathcal{P}$.

Definition: Komplexitätsklasse $\mathcal{NPI}$

$\mathcal{NPI} := \mathcal{NP} \backslash (\mathcal{P} \cup \mathcal{NPC})$.

Definition: Komplexitätsklasse $\mathcal{NPC}$

Klasse der $\mathcal{NP}$-vollständigen Sprachen bzw. Probleme.

Definition: Komplexitätsklasse **co** $-\mathcal{P}$

Klasse aller Sprachen $\Sigma^* \backslash L$ für $L \subseteq \Sigma^*$ und $L \in \mathcal{P}$.

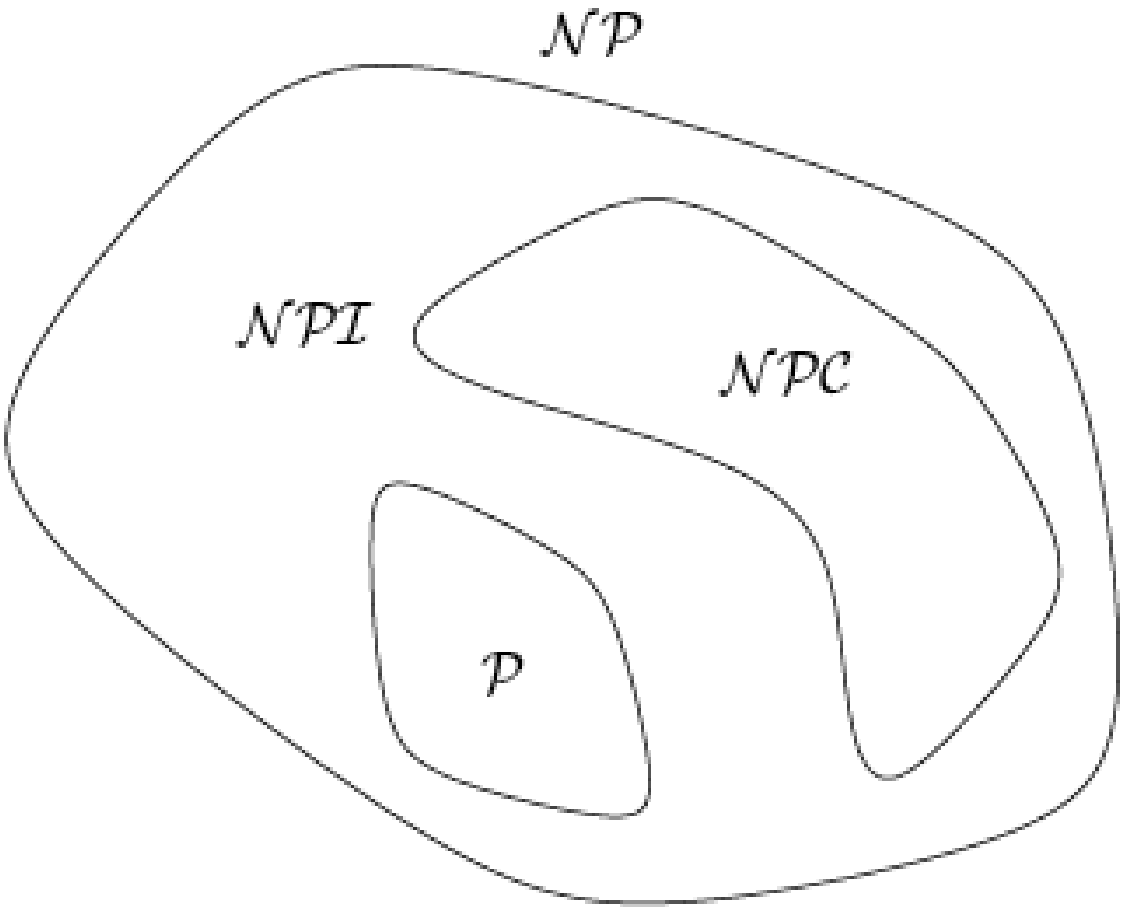
Definition: Komplexitätsklasse $\mathcal{NPI}$

$\mathcal{NPI} := \mathcal{NP} \backslash (\mathcal{P} \cup \mathcal{NPC})$.

Definition: Komplexitätsklasse **co** $-\mathcal{NP}$

Klasse aller Sprachen $\Sigma^* \backslash L$ für $L \subseteq \Sigma^*$ und $L \in \mathcal{NP}$.

Komplexitätsklassen:



Problem **FALSE SAT**:
Gegeben: Ein boolescher Ausdruck A, der falsch ist, wenn alle Variablen falsch sind.
Frage: Gibt es eine weitere Belegung der Variablen, sodass A falsch ist?

Problem **FALSE SAT**:
Gegeben: Ein boolescher Ausdruck A, der falsch ist, wenn alle Variablen falsch sind.
Frage: Gibt es eine weitere Belegung der Variablen, sodass A falsch ist?

Aufgabe 1

Formuliere das komplementäre Problem **co-FALSE SAT**.

Problem **FALSE SAT**:
Gegeben: Ein boolescher Ausdruck A, der falsch ist, wenn alle Variablen falsch sind.
Frage: Gibt es eine weitere Belegung der Variablen, sodass A falsch ist?

Aufgabe 1

Formuliere das komplementäre Problem **co-FALSE SAT**.

Aufgabe 2

In welcher der Klassen $\mathcal{NP}, \mathbf{co}-\mathcal{NP}$ liegt **FALSE SAT** mit Sicherheit? Begründe deine Antwort.
Können wir die Klasse noch weiter einschränken?

Wir wissen: **FALSE SAT** $\in \mathcal{NP}$
Vermutung: **FALSE SAT** $\in \mathcal{NPC}$
Also versuchen wir eine pol. Reduktion: **SAT** $\propto$ **FALSE SAT**
Für eine gegebene Instanz E von **SAT** konstruieren wir eine Instanz A für **FALSE SAT** wie folgt:

Wir wissen: **FALSE SAT** $\in \mathcal{NP}$
Vermutung: **FALSE SAT** $\in \mathcal{NPC}$
Also versuchen wir eine pol. Reduktion: **SAT** $\propto$ **FALSE SAT**
Für eine gegebene Instanz E von **SAT** konstruieren wir eine Instanz A für **FALSE SAT** wie folgt:

Test if E is false when all variables are false. If it is, convert E to $A = \ E * (x_1 + ... + x_n)$, a polynomial reduction. A is false when all variables are false:
$A = \ (?) * (false) = false$

If E is in **SAT**, then it is satisfiable with some truth assignment other than all-false, so A will be false with this non all-false assignment and hence be in **FALSE SAT**:
$A = \ (true) * (true) = false$

If E is not satisfiable, then A will be true with any non all-false assignment and hence not be in **FALSE SAT**:
$A = \ (false) * (true) = true$

Therefore A is in **FALSE SAT** if and only if E is satisfiable.

Otherwise, we know E is satisfiable, so let $A = y1 * y2$. A is in **FALSE SAT** since it is false when both variables are false or when one variable is true.

Quelle: http://tech.groups.yahoo.com/group/comp-sci-theory/message/802

If E is not satisfiable, then A will be true with any non all-false assignment and hence not be in **FALSE SAT**:
$A = \ (false) * (true) = true$

Therefore A is in **FALSE SAT** if and only if E is satisfiable.

Otherwise, we know E is satisfiable, so let $A = y1 * y2$. A is in **FALSE SAT** since it is false when both variables are false or when one variable is true.

Quelle: http://tech.groups.yahoo.com/group/comp-sci-theory/message/802

$\Rightarrow$ **FALSE SAT** $\in \mathcal{NPC}$

Vermutung: $\mathcal{NP} \neq \textbf{co}-\mathcal{NP}$

Nach Lemma 4.26 ist eine $\mathcal{NP}$-vollständige Sprache, die in $\textbf{co}-\mathcal{NP}$ liegt $\mathcal{NP} = \textbf{co}-\mathcal{NP}$.

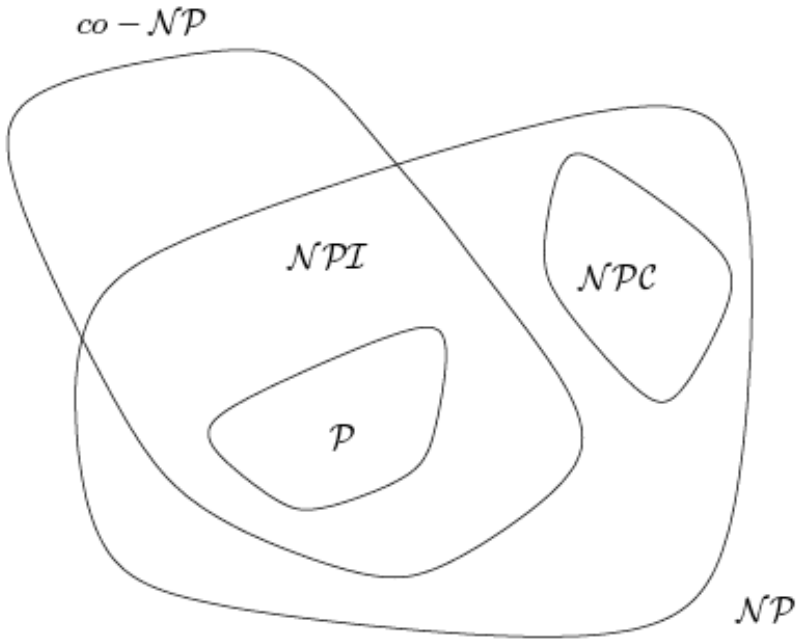
Solche Sprachen kennt man bisher aber nicht.

Vermutung: $\mathcal{NP} \neq \textbf{co}-\mathcal{NP}$

Nach Lemma 4.26 ist eine $\mathcal{NP}$-vollständige Sprache, die in $\textbf{co}-\mathcal{NP}$ liegt $\mathcal{NP} = \textbf{co}-\mathcal{NP}$.

Solche Sprachen kennt man bisher aber nicht.

Somit haben wir folgende Vermutung:



Problem **MISSING EDGE**:
Gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Konstante K.
Frage: Existieren für jede Teilmenge $V' \subseteq V$ mit $|V'| = K$ zwei Knoten $s, t \in V'$ mit $\{s, t\} \notin E$?

Problem **MISSING EDGE**:
Gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Konstante K.
Frage: Existieren für jede Teilmenge $V' \subseteq V$ mit $|V'| = K$ zwei Knoten $s, t \in V'$ mit $\{s, t\} \notin E$?

Aufgabe 3

Zeige: **MISSING EDGE** $\in$ **co** $-\mathcal{NP}$.

Definition
Ein **Suchproblem** Π *wird beschrieben durch*

1. *die Menge der Problembeispiele* D_Π *und*
2. *für* $I \in D_\Pi$ *die Menge* $S_\Pi(I)$ aller *Lösungen von* I.

Die „Lösung“ eines Suchproblems besteht in der Angabe einer Lösung aus $S_\Pi(I)$ *für ein Problembeispiel* $I \in D_\Pi$ *mit* $S_\Pi(I) \neq \emptyset$ *und „Nein“ sonst.*

Definition
Ein **Suchproblem** Π *wird beschrieben durch*

1. *die Menge der Problembeispiele* D_Π *und*
2. *für* $I \in D_\Pi$ *die Menge* $S_\Pi(I)$ aller *Lösungen von* I.

Die „Lösung" eines Suchproblems besteht in der Angabe einer Lösung aus $S_\Pi(I)$ *für ein Problembeispiel* $I \in D_\Pi$ *mit* $S_\Pi(I) \neq \emptyset$ *und „Nein" sonst.*

Problem TSP–Suchproblem
Gegeben: Graph $G = (V, E)$ vollständig und gewichtet mit c: $E \to \mathbb{R}$.

Frage: Gib eine optimale Tour zu G bezüglich c an.

$S_\Pi(I)$ ist die Menge aller optimalen Touren zu I. Die Angabe einer optimalen Tour löst also das Problem.

Definition
Ein **Aufzählungsproblem** Π *ist gegeben durch*

1. *die Menge der Problembeispiele* D_Π *und*
2. *für* $I \in D_\Pi$ *die Menge* $S_\Pi(I)$ *aller Lösungen von* I.

Die Lösung *eines Aufzählungsproblem* Π *besteht in der Angabe der Kardinalität von* $S_\Pi(I)$, *d.h.* $|S_\Pi(I)|$.

Definition
Ein **Aufzählungsproblem** Π *ist gegeben durch*

1. *die Menge der Problembeispiele* D_Π *und*
2. *für* $I \in D_\Pi$ *die Menge* $S_\Pi(I)$ *aller Lösungen von* I.

Die Lösung *eines Aufzählungsproblem* Π *besteht in der Angabe der Kardinalität von* $S_\Pi(I)$, *d.h.* $|S_\Pi(I)|$.

Problem Hamilton–Kreis (Aufzählungsproblem)
Gegeben: ein ungerichteter, ungewichteter Graph $G = (V, E)$.

Frage: Wieviele Hamilton–Kreise gibt es in G?

Zusammenhang Such-, Aufzählungs- und Entscheidungsprobleme

Problem: Wie vergleicht man Such-, Aufzählungs- und Entscheidungsprobleme bzgl. ihrer Komplexität?

Ansatz: Wir brauchen ein Konzept dafür, wann ein Problem Π mindestens so schwer ist, wie ein Problem Π'. Dazu benutzen wir, analog zur polynomialen Transformation bei Entscheidungsproblemen, das Konzept der Turingreduzierbarkeit.

Aufgabe 4

Formuliere ein weiteres Suchproblem.

Aufgabe 4

Formuliere ein weiteres Suchproblem.

Aufgabe 5

Formuliere ein weiteres Aufzählungsproblem.

Definition: Relation Π

Zu einem gegebenen Suchproblem defnieren wir eine Relation:
$R_\Pi = \{(x, s)|x \in D_\Pi, s \in S_\Pi(x)\}$

Ein Algorithms löst das durch die Relation R_Π beschriebene Suchproblem Π, wenn er eine Funktion berechnet, die R_Π realisiert.

Definition

Eine ***Orakel-Turing-Maschine*** *zum Orakel* $G\colon \Sigma^* \to \Sigma^*$ *ist eine deterministische Turing-Maschine mit einem ausgezeichnetem* ***Orakelband*** *und zwei zusätzlichen Zuständen* q_f *und* q_a*. Dabei ist* q_f *der* ***Fragezustand*** *und* q_a *der* ***Antwortzustand*** *des Orakels. Die Arbeitsweise ist in allen Zuständen* $q \neq q_f$ *wie bei der normalen Turing-Maschine. Wenn der Zustand* q_f *angenommen wird, der Kopf sich auf Position* i *des Orakelbandes befindet und der Inhalt des Orakelbandes auf Position* $1, \ldots, i$ *das Wort* $y = y_1 \ldots y_i$ *ist, so ist der Übergang:*

1. *Fehlermeldung, falls* $y \notin \Sigma^*$
2. *in einem Schritt wird* y *auf dem Orakelband gelöscht und* $g(y)$ *auf die Positionen* $1, \ldots, |g(y)|$ *des Orakelbandes geschrieben. Der Kopf des Orakelbandes springt auf Position 1 und der Folgezustand ist* q_a*.*

Definition

Seien R, R' Relationen über Σ^*. Eine **Turing-Reduktion** $\propto_T$ *von R auf R' ($R \propto_T R'$), ist eine Orakel-Turing-Maschine $\mathcal{M}$, deren Orakel die Relation R' realisiert und die selber in polynomialer Zeit die Funktion f berechnet, die R realisiert.*

Bemerkung:

- Falls R' in polynomialer Zeit realisierbar ist und $R \propto_T R'$, so ist auch R in polynomialer Zeit realisierbar.
- Falls $R \propto_T R'$ und $R' \propto_T R''$ so auch $R \propto_T R''$.

Definition: $\mathcal{NP}$-schwer

Ein Suchproblem Π heisst $\mathcal{NP}$-schwer, falls es eine $\mathcal{NP}$-vollständige Sprache L gibt mit $L \propto_T \Pi$

Definition: $\mathcal{NP}$-schwer

Ein Suchproblem Π heisst $\mathcal{NP}$-schwer, falls es eine $\mathcal{NP}$-vollständige Sprache L gibt mit $L \propto_T \Pi$

Hinweis: Ein Problem, welches $\mathcal{NP}$-schwer ist, muss nicht zwangsweise $\mathcal{NP}$-vollständig sein.

Definition: $\mathcal{NP}$-schwer

Ein Suchproblem Π heisst $\mathcal{NP}$-schwer, falls es eine $\mathcal{NP}$-vollständige Sprache L gibt mit $L \propto_T \Pi$

Hinweis: Ein Problem, welches $\mathcal{NP}$-schwer ist, muss nicht zwangsweise $\mathcal{NP}$-vollständig sein.

Wir nennen ein Problem $\mathcal{NP}$–schwer, wenn es mindestens so schwer ist, wie alle $\mathcal{NP}$–vollständigen Probleme. Darunter fallen auch

- Optimierungsprobleme, für die das zugehörige Entscheidungsproblem $\mathcal{NP}$–vollständig ist.
- Entscheidungsprobleme Π, für die gilt, dass für alle Probleme $\Pi' \in \mathcal{NP}$ gilt $\Pi' \propto \Pi$, aber für die nicht klar ist, ob $\Pi \in \mathcal{NP}$.

Klar ist, dass ein $\mathcal{NP}$–vollständiges Problem auch $\mathcal{NP}$–schwer ist.

Aufgabe

Zeige für das Optimierungsproblem CLIQUE, dass es NP-schwer ist.

Ein Beispiel für eine $\mathcal{NP}$-schwere Sprache ist **INTEGER PROGRAMMING**

Ein Beispiel für eine $\mathcal{NP}$-schwere Sprache ist **INTEGER PROGRAMMING**

Problem **INTEGER PROGRAMMING (ILP)**:
Gegeben: $A \in \mathbb{N}^{n \times m}, b \in \mathbb{N}^m, c \in \mathbb{N}^n, B \in \mathbb{N}$
Frage: Existieren $x \in \mathbb{N}^n$, so dass $\sum_{j=1}^{n} c_j x_j = B$ und $Ax \leq b$?

Problem **KNAPSACK**:
Gegeben: Endliche Menge M, Gewichtsfunktion $w : M \to \mathbb{N}$, Kostenfunktion $c : M \to \mathbb{N}$ und Konstanten $W, C \in \mathbb{N}$
Frage: Existiert Teilmenge $M' \subseteq M$ mit $\sum_{M'} w \leq W$ und $\sum_{M'} c \leq C$?

Problem **KNAPSACK**:
Gegeben: Endliche Menge M, Gewichtsfunktion $w : M \to \mathbb{N}$, Kostenfunktion $c : M \to \mathbb{N}$ und Konstanten $W, C \in \mathbb{N}$
Frage: Existiert Teilmenge $M' \subseteq M$ mit $\sum_{M'} w \leq W$ und $\sum_{M'} c \leq C$?

Aufgabe 6

Formuliere KNAPSACK als ILP.

$\mathcal{NP}$-schwer

Reflexion

Was haben wir heute gelernt?

Noch Fragen?

Vorschau

Vorschau

- Pseudopolynomiale Algorithmen

Vorschau

- Pseudopolynomiale Algorithmen
- verschiedene Approximationsalgorithmen

Bis zum nächsten Mal im neuen Jahr 2008