Übungsblatt06 - Berechnungsprobleme ... - Musterlösung

TH Mittelhessen, FB MNI, Berechenbarkeit und Komplexität, Prof. Dr. B. Just

Aufgabe 1

Im Folgenden finden Sie eine Liste von Sprachen, die Berechnungsprobleme darstellen. Bilden Sie sich eine Meinung, ob diese jeweils in P, NP, co-NP oder außerhalb dieser Komplexitätsklassen liegen.

- Falls Sie der Meinung sind, ein Berechnungsproblem liegt in P, geben Sie einen Polynomialzeitalgorithmus mit Laufzeitanalyse an.
- Falls das nicht gelingt, und Sie glauben, das Problem liegt in NP, geben Sie einen Nichtdeterministischen Polynomialzeitalgorithmus (mit Laufzeitanalyse) sowie einen Verifier an.
- Falls Sie glauben, das Problem liegt in co-NP, geben Sie einen Polynomialzeitalgorithmus (mit Laufzeitanalyse) und einen Verifier für das Komplement an.
- Falls Sie glauben, das Problem liegt außerhalb von NP $\cup$ co-NP, begründen Sie dies bitte :).

a.) EULER-GRAPHS = $\{G = (V, E) : G$ ist Euler-Graph$\}$.
Erinnerung: Ein Graph ist ein Euler-Graph, wenn er einen Kreis enthält, der jede Kante genau einmal durchläuft (nicht notwendig alle Knoten, isolierte Knoten werden nicht durchlaufen).

b.) HAMPATH = $\{G = (V, E) : G$ ist Hamilton-Graph$\}$.
Erinnerung: Ein Graph ist ein Hamilton-Graph, wenn er einen Kreis enthält, der jeden Knoten genau einmal durchläuft (nicht notwendig alle Kanten).

c.) CLIQUE = $\{(G, k) : G$ ist ungerichteter Graph, $k \in \mathbb{N}$, und G enthält eine k-Clique$\}$.
Definition: Eine $k-$Clique ist eine k-elementige Teilmenge der Knoten, deren Knoten paarweise adjazent sind.

d.) ANTI-CLIQUE = $\{(G, k) :$ $k \in \mathbb{N}$, G ungerichteter Graph mit k-Anticlique$\}$.
Definition: Eine k-Anticlique ist eine k-elementige Teilmenge der Knoten, deren Knoten paarweise nicht adjazent sind.

e.) Die Menge aller ungerichteter Graphen ohne CLIQUE.
Bemerkung: Das ist natürlich nicht ANTI-CLIQUE :).

f.) VERTEX-COVER = $\{(G, k) : k \in \mathbb{N}$, G ungerichteter Graph, mit k-Vertex-Cover$\}$.
Definition: Ein Vertex-Cover ist eine Teilmenge der Knoten (vertex) des Graphen, mit nichtleerem Schnitt zu jeder Kante.

g.) SUBSET-SUM = $\{(S, t) : S = \{x_1, ..., x_n\} \subset \mathbb{N}$ und $\exists\ y_1, ..., y_s \in S$ mit $\Sigma y_i = t\}$.

h.) SAT=$\{$erfüllbare boolesche Formeln$\}$.
Hinweis: Eine Boolesche Formel heißt „erfüllbar“, wenn es eine Belegung der Variablen mit 0 und 1 gibt, sodass ihr Wahrheitswert bei dieser Belegung 1 ist.

i.) TQBF=$\{$wahre voll quantifizierte boolesche Formeln$\}$.
Definition: Eine „voll quantifizierte boolesche Formel“ ist eine boolesche Formel, der für jede Variable ein Quantor vorausgestellt ist. Sie hat einen eindeutigen Wahrheitswert.
Beispiel: $\exists x_1 : \forall x_2 : (x_1 \vee x_2)$
hat den Wahrheitswert 1, denn wählt man $x_1 = 1$, so ist der Ausdruck für alle x_2 wahr.

j.) COMPOSITES = $\{n \in \mathbb{N} : n = p \cdot q$ für $p, q \in \mathbb{N}$, $p, q > 1\}$

k.) PRIMES = $\{p \in \mathbb{N} : p$ prim$\}$.
Hinweis: Überlegen Sie sich etwas über COMPOSITES und PRIMES, aber die Lösung werden Sie ohne Internet-Recherche kaum finden. Und lassen Sie sich von dem, was Sie finden, nicht abschrecken ;).

Lösung Aufgabe 1:

a.) EULER-GRAPHS ist in P. Denn aus der diskreten Mathematik ist bekannt, dass ein Graph G genau dann ein Euler-Graph ist, wenn er die folgenden beiden Bedingungen erfüllt:

i.) G ist bis auf isolierte Punkte zusammenhängend, und

ii.) Jeder Knoten von G hat geraden Grad.

Beides kann man in Polynomialzeit überprüfen. (Den Zusammenhang durch einen Breadth-First-Search oder Depth-First-Search Algorithmus, die Bedingung mit dem graden Grad durch Abzählen).

b.) Für HAMPATH ist bisher kein Polynomialzeit-Algorithmus bekannt. Es handelt sich um eines der NP-vollständigen Probleme (kommt später in der Vorlesung).
HAMPATH ist in NP. Die nichtdeterministische Turingmaschine $T_{HAMPATH}$ für HAMPATH errät in Zeit O(# Knoten) einen Pfad.
Der Verifier erhält als Input einen Graphen $G = (V, E)$ und (als Zertifikat) eine Permutation der Knoten. Er prüft, ob alle Kanten im Graphen vorhanden sind, um die Permutation abzulaufen. Dies ist in $O(|V| \cdot |E|)$ Schritten, also in einer Laufzeit, die polynomiell in der Länge des Inputs ist, möglich.
Man überlegt zur Korrektheit des Verifiers:
Es gibt ein Zerifikat, das der Verifier mit einem Graphen G akzeptiert, genau dann, wenn G ein Hamilton-Graph ist.

c.) Für CLIQUE ist bisher ebenfalls kein Polynomialzeit-Algorithmus bekannt. Es handelt sich wie HAMPATH um eines der NP-vollständigen Probleme, die später in der Vorlesung behandelt werden.

CLIQUE ist in NP. Die nichtdeterministische Turingmaschine T_{CLIQUE} für CLIQUE errät bei Input $G = (V, E)$ und einer Zahl $k \in \mathbb{N}$ von in Zeit $O(|V|)$ eine $k-$Clique. Wir nehmen im Folgenden an, dass k kleiner als die Anzahl der Knoten des Graphen ist, sonst ist (G, k) trivialerweise nicht in CLIQUE.
Der Verifier erhält als Input einen Graphen $G = (V, E)$, eine Zahl $k \leq |V|$ und (als Zertifikat) eine k-elementige Teilmenge von V. Er prüft, ob alle Kanten zwischen den Knoten der Teilmenge vorhanden sind.
Dies ist in Polynomialzeit möglich, denn es müssen höchstens $O(|V|^2)$ Kanten überprüft werden, und das Überprüfen einer dauert höchstens $O(|V| + |E|)$ Schritte, insgesamt also höchstens $O(|V|^2 \cdot (|V| + |E|))$ Schritte.
Man überlegt:
Es gibt ein Zerifikat, das der Verifier mit einem Graphen G und einer Zahl k akzeptiert, genau dann, wenn (G, k) in CLIQUE ist.

d.) ANTI-CLIQUE ist ebenfalls NP-vollständig, es ist also kein Poynomialzeit-Algorithmus bekannt. ANTI-CLIQUE ist von der Berechnugskomplexität mit CLIQUE äquivalent.
Für einen Graphen $G = (V, E)$ mit Knotenmenge $V = \{v_1, ..., v_n\}$ sei der Komplementärgraph $\overline{G} = (V, (V \times V) \setminus (E \cup \{(v_i, v_i) : i = 1, .., n\}))$ der Graph mit derselben Knotenmenge wie G, der genau die Kanten enthält, die G nicht enthält.
Dann ist offenbar (G, k) in CLIQUE, genau dann, wenn $(\overline{G}, k)$ in ANTI-CLIQUE ist.
Die nichtdeterministische Polynomialzeit-Turingmaschine errät die Anticlique, und der Verifier verifiziert, dass tatsächlich keine Kante zwischen ihren Knoten vorhanden ist. Die Laufzeit ist, wie für CLIQUE in Teil c. analysiert wurde, höchstens $O(|V|^2 \cdot (|V| + |E|))$ Schritte.

e.) Die Menge aller ungerichteten Graphen ohne CLIQUE ist in co-NP, denn sie ist das Komplement von CLIQUE.
Es ist keine Polynomialzeit-nichtdeterministische Turingmaschine bekannt.
Es ist auch nicht klar, wie ein polynomieller Verifier aussehen könnte. Denn das Überprüfen aller k-elementigen Teilgraphen benötigt z.B. für $k = |V|/2$ mehr als polynomial viele Schritte, nämlich $O(|V|^{|V|/2})$ Schritte für einen Graphen $G = (V, E)$.

f.) Auch für VERTEX-COVER ist bisher kein Polynomialzeit-Algorithmus bekannt. Es handelt sich ebenfalls um eines der NP-vollständigen Probleme, wie HAMPATH und CLIQUE.
VERTEX-COVER ist in NP. Die nichtdeterministische Turingmaschine $T_{VERTEX-COVER}$ für VERTEX-COVER errät bei Input $G = (V, E)$ und einer Zahl $k \in \mathbb{N}$ von in Zeit $O(|V|)$ ein $k-$Vertex-Cover, falls eines existiert. Wir nehmen im Folgenden an, dass k kleiner als die Anzahl der Knoten des Graphen ist, sonst ist (G, k) trivialerweise nicht in VERTEX-COVER.
Der Verifier erhält als Input einen Graphen $G = (V, E)$, eine Zahl $k \leq |V|$ und (als Zertifikat) eine k-elementige Teilmenge von V. Er prüft, ob jede Kante einen der Knoten des Zertifikats enthält.
Dies ist in Polynomialzeit möglich, denn es müssen höchstens $O(|E|)$ Kanten überprüft werden, und das Überprüfen einer dauert höchstens $O(|V|)$ Schritte, insgesamt also höchstens $O(|V| \cdot |E|)$ Schritte. Man überlegt leicht die folgende Äquivalenz:
Es gibt ein Zerifikat, das der Verifier mit einem Graphen G und einer Zahl k akzeptiert, genau dann, wenn (G, k) in VERTEX-Cover ist.

g.) Auch für SUBSET-SUM ist bisher kein Polynomialzeit-Algorithmus bekannt, es handelt sich ebenfalls um eines der NP-vollständigen Probleme (spannend, oder ;)).
SUBSET-SUM ist in NP. Die nichtdeterministische Turingmaschine $T_{\text{SUBSET-SUM}}$ errät bei Input $(x_1, ..., x_n, t)$ in Zeit $O(n)$ eine Teilmenge $I \subseteq \{1, ..., n\}$. Ist der Input in SUBSET-SUM, gibt es eine Teilmenge mit $\Sigma_{i \in I}\ x_i = t$, also erkennt $T_{\text{SUBSET-SUM}}$ die Sprache SUBSET-SUM.
Der Verifier erhält als Input $(x_1, ..., x_n, t)$, und (als Zertifikat) eine Teilmenge von $I \subseteq \{1, ..., n\}$. Er prüft, ob $\Sigma_{i \in I}\ x_i = t$.
Dies ist in Polynomialzeit möglich, denn es müssen höchstens $O(n)$ Zahlen addiert werden, die alle bereits im Input vorkommen. Also ist die Gesamtrechenzeit $O(n \cdot$Länge Input$) = 0($(Länge Input$)^2)$.
Man überlegt:
Es gibt ein Zerifikat, das der Verifier mit einem Input $(x_1, ..., x_n, t)$ akzeptiert, genau dann, wenn Input $(x_1, ..., x_n, t)$ in SUBSET-SUM ist.

h.) Für SAT ist ebenfalls kein Polynomialzeitalgorithmus bekannt, SAT ist ebenfalls NP-vollständig.

SAT ist in NP. Die nichtdeterministische Turingmaschine T_{SAT} errät bei Input einer booleschen Formel mit n booleschen Variablen Zeit $O(n)$ eine Belegung der Variablen. Ist der Input in SAT, gibt es eine erfüllende Belegung, also erkennt T_{SAT} die Sprache SAT.
Der Verifier erhält als Input eine boolesche Formel, und (als Zertifikat) eine Belegung ihrer Variablen. Er prüft, ob die Belegung die Formel erfüllt.
Dies ist in Polynomialzeit möglich, eine grobe Schätzung ist $0($(Länge Input$)^2)$ Rechenschritte, um die Formel von innen nach außen auszuwerten. Man überlegt:
Es gibt ein Zerifikat, das der Verifier mit einem Input akzeptiert, genau dann, wenn der Input in SAT ist.

i.) Es ist nicht bekannt, ob TQBF in NP oder co-NP ist, man vermutet aber, in beiden nicht - es scheint schwerer zu sein.
Es ist ein Algorithmus bekannt, der TQBF in polynomialem SPEICHERPLATZ entscheidet, aber dieser hat exponentielle Laufzeit. Auch ein Zertifikat scheint im Allgemeinen exponentielle Länge zu haben.

j.) COMPOSITES ist in P.
Der Beweis dafür wurde erst 2002 von Agrawal, Kayal und Saxena gefunden. Der Algorithmus weist aber nur nach, ob ein Input n zusammengesetzt ist oder nicht. Er liefert KEINEN Faktor, sondern nur eine Eigenschaft der Gruppe $\mathbb{Z}/n\mathbb{Z}^*$, die bezeugt, ob n zusammengesetzt ist oder nicht.
Vorher galt COMPOSITES lange als ein Kandidat für eine Menge, die in NP ist, aber nicht in P und auch nicht NP-vollständig ist.
Man beachte, dass für eine Zahl $n \in \mathbb{N}$ die Inputlänge nicht $O(n)$ ist, sonst könnte man in Polynomialzeit ausprobieren, ob man einen Primfaktor findet. Die Inputlänge ist nur $O(\log n)$. Ausprobieren dauert also exponentiell lang in der Länge des Inputs.

COMPOSITES $\in$ NP sieht man leicht: Die nichtdeterministische TM errät Bit für Bit die Binärdarstellung eines Faktors - dies ist in Zeit $O(\log n)$ möglich. Der Verifier prüft, ob der Faktor tatsächlich weder 1 noch n ist, und ob er Teiler von n ist.

k.) Da COMPOSITES in P ist, ist das Komplement PRIMES ebenfalls in P. (Tatsächlich heißt die bahnbrechende Arbeit von Agrawal, Kayal und Saxena „PRIMES is in P".).

Ohne tiefer in die Struktur von $\mathbb{Z}/n\mathbb{Z}^*$ einzusteigen, sieht man, dass PRIMES in co-NP ist, weil das Complement COMPOSITES in NP ist.

Das PRIMES auch in NP ist, ist nicht so offensichtlich. Denn wie kann ein Zertifikat aussehen, das besagt, dass es KEINEN echten Primteiler gibt? Man benutzt dazu einen Satz aus der Zahlentheorie. Dieser besagt:
p ist prim genau dann, wenn $\mathbb{Z}/p\mathbb{Z}^*$ sowohl zyklisch als auch von der Ordnung $p-1$ ist.
Mehr findet man, wenn man „primes in np proof" googelt. Aber die benötigten Grundlagen aus der Gruppentheorie sprengen den Rahmen dieser Vorlesung. Bitte nicht von diesem Teil der Aufgabe abschrecken lassen - hier war der Weg das Ziel :).