Berechenbarkeit und Komplexität
Simon Wagner - 343756
Timotheus Jochum - 304222

Übungsblatt 9

December 22, 2016

9. Übung

Aufgabe 9.1

Zeigen Sie, jeweils mit Hilfe eines Polynomialzeitverifizierers, dass die folgenden Entscheidungsprobleme in NP sind. Beschreiben Sie dazu im Detail die Kodierung und die Länge das Zertifikats sowie die Arbeitsweise und die Laufzeit des Verifizierers (eine Beschreibung ähnlich zu Folie 418 ist **nicht** hinreichend detailliert)

(a) MaxSpanTree $= \{(G, c) \mid c \in \mathbb{N}$ und G ist ein Graph mit gewichteten Kanten und hat einen Spannbaum mit Kosten $\geq c\}$

MaxSpanTree ist in NP:
Es kodiert das Zertifikat $z \in \{0,1\}^{|E|}$ welche Kanten $z_i \in E = \{e_1, \ldots, e_n\}$ in unserem Spannbaum enthalten sind. Dabei gilt $z_i = 1$, wenn z_i im Spannbaum enthalten ist.
Es gilt auch $|z| = |E|$, die Eingabelänge ist somit polynomiell.
Der Verifizierer prüft nun ob der Graph zusammenhängend und $|V| - 1$ Stellen gleich 1 sind, also ob der durch z kodierte Baum T ein Spannbaum ist. Sollten diese Bedingungen nicht erfüllt sein wird die Eingabe verworfen.
Anschließend prüfen wir, ob T V überdeckt.
Ist auch das erfüllt, berechnen wir die Summe der Kantengewichte von T $w(T)$, und prüfen $w(T) \geq c$. Ist dies erfüllt akzeptieren wir die Eingabe, ansonsten wird sie verworfen.

Laufzeit:
Prüfen der Einträge von z: $\mathcal{O}(|E|\log(|E|))$
Zusammenhang von z: Nach VL $\mathcal{O}(|V|^2)$
Überdeckungsprüfung: Wir prüfen für alle Knoten alle sonstigen $|V| - 1$ Kanten, erhalten also $\mathcal{O}(|V|^2)$
Addieren der Kantengewichte und Vergleich mit c: Wir addieren $|V| - 1$ Werte und erhalten $\mathcal{O}(|V||E|)$ als Laufzeit der Summierung und $\mathcal{O}(|V|c)$ als Komplexität des Vergleiches.
Da alle Schritte in polynomieller Zeit (nach Eingabelänge) ausgeführt werden folgt die Behauptung.

(b) Composite $= \{w \in \{0,1\}^* \mid w$ ist die binäre Kodierung einer Zahl $k \in \mathbb{N}$ und k ist keine Primzahl$\}$

Composite ist in NP:
Wir wählen als Zertifikat eine binäre Kodierung von $a \geq 2$ und $b \geq 2$ mit $k = ab$, zB also indem wir eine 0 vor jede Stelle der Binärdarstellung von a schreiben, und eine 1 vor jede Stelle von b. Die Eingabelänge ist also polynomiell in der Eingabelänge.
Ist $w = k$ keine (binär codierte) Primzahl, so können wir sie entsprechend in a und b zerlegen und so ein wo oben beschriebenes Zertifikat erstellen.
Der Verifizierer prüft $a, b \neq 0$, und vergleicht danach $k = ab$.

Laufzeit:
Mit Ü7.1 können wir die Multiplikation in polynomieller Zeit ausführen.
Zu prüfen ob $k = ab$ braucht nur $\mathcal{O}(n \cdot \log(k))$ (mit n als Eingabelänge).
Da alle Schritte in polynomieller Zeit (nach Eingabelänge) ausgeführt werden gilt die Behauptung.

(c) GRAPHISOMORPHIE $= \{G_1 \# G_2 \mid G_1, G_2$ sind Graphen, G_1 ist isomorph $G_2\}$
Zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ sind isomorph, falls es eine bijektive Abbildung $f : V_1 \to V_2$ gibt, so dass $(v_i, v_j) \in E_1 \Leftrightarrow (f(v_i)), f(v_j)) \in E_2$ für alle $v_i, v_j \in V$ gilt.

GRAPHISOMORPHIE ist in NP:
Wir schaffen ein Zertifikat z, welches die Abbildung f von $V_1 = \{v_{1,1}, \ldots, v_{1,n}\}$ nach $V_2 = \{v_{2,1}, \ldots, v_{2,n}\}$ als Funktion g kodiert, mit $g(a) = b \Leftrightarrow f(v_{1,a}) = v_{2,b}$.
Die Kodierung sieht dabei wie folgt aus:
Für einen Graphen $G = (V, E)$ schreiben wir

$$\text{bin}(n)\#\text{code}(e_1)\#...\#\text{code}(e_m)$$

(wobei # für ein einzigartig genutztes Symbol steht).
Hierbei gilt

$$\text{code}(e_i) = \text{code}((v_s, v_t)) = (\text{bin}(s), \text{bin}(t)) \text{ und } s \leq t.$$

Die Abbildung f wird nun als

$$\text{bin(1), bin(g(1))\#bin(2), bin(g(2))\#...\#bin(n), bin(g(n))}$$

kodiert.
Wie in (b) können wir die einzelnen Paare durch 0 und 1 als Präfix leicht voneinander trennen. Wir erhalten (maximal) $|V|$ Zuordnungen mit je $2\log(|V|)$ Stellen, demnach ist das Zertifikat polynomiell in der Eingabelänge.
Unser Verifizierer prüft nun $|V_1| = |V_2|$ indem er den ersten Eintrag von $\text{code}(G_1)$ und $\text{code}(G_2)$ vergleicht ($\mathcal{O}|(\text{gesamte Eingabe}|^2)$).
Außerderm werden alle Knoten $v_{1,k}$ durch $f(v_{1,k}), k \in [1, n]$ in der Kodierung G_1 ersetzt, indem k durch $g(k)$ ersetzt wird. Die Suche nach der entsprechenden Kodierung kann dabei in $\mathcal{O}(|z| \log\ k)$ geschehen.
Da alle Kanten welche inzident sind ersetzt werden müssen erhalten wir eine Laufzeit von $\mathcal{O}(|E| \cdot |z| \cdot \log |V|), \log k \leq \log |V|$.
Wir prüfen, ob im modifizierten Graphen der Index des ersten Knotens $\leq$ Index des zweiten Knotens ist, und tauschen diese ansonsten (in $\mathcal{O}(|E| \cdot \log |V|)$).
Schlussendlich wird überprüft, ob alle Kanten des modifizierten Graphen auch in E_2 liegen (und umgekehrt), indem wir für jede Kante die modifizierte Eingabe durchlaufen (in $\mathcal{O}(|E|^2 \cdot |\text{Eingabe}| \cdot \log |V|)$)
Da alle Schritte in polynomieller Zeit (nach Eingabelänge) erfolgen gilt die Behauptung.

Aufgabe 9.2

Seien BPP und TSP die in der Vorlesung vorgestellten Optimierungsprobleme. Zeigen Sie die folgenden Aussagen:

(a) Falls BPP-E in P ist, so kann auch BPP in polynomieller Zeit gelöst werden.

(b) Falls TSP-E in P ist, so kann auch TSP in polynomieller Zeit gelöst werden.

Aufgabe 9.3

Betrachten Sie folgendes Entscheidungsproblem:

DoubleSat
Eingabe: Eine aussagenlogische Formel φ in KNF.
Ausgabe: Ja gdw. es mindestens **zwei** erfüllende Belegungen für φ gibt.

Zeigen Sie, dass DoubleSat NP-vollständig ist.

Wir reduzieren DoubleSat auf Sat:
Wir fügen für eine Eingabe $X(x_1, \ldots, x_n)$ eine Variable y ein, wobei für die Ausgabe gilt

$$X'(x_1, \cdots, x_n, y) = X(x_1, \cdots, x_n) \wedge (y \vee \bar{y}).$$

Wenn X Sat $\notin$ Sat, X also nicht erfüllbar ist, kann offensichtlicherweise auch X' nicht erfüllbar sein und somit gilt auch $X' \notin$ DoubleSat.
Wenn aber $X \in$ Sat, dann kann man X' lösen, indem man $y = 0$ oder $y = 1$ setzt. Man hat also zwei erfüllende Belegungen, und somit ist $X' \in$ DoubleSat.
$\Rightarrow$ Sat $\leq_p$ DoubleSat
$\Rightarrow$ DoubleSat ist NP-vollständig.