Theoretische Informatik, Woche 9

Justin Brackemann (977398)

Aufgabe 9.1

(a) χ_E:

Gegeben: Rechnende TM M_f, $k \in \mathbb{N}$
Gesucht: Gibt es einen Ausgabewert von $M_f \geq k$?

(b) $\Sigma = \{0, 1\}$. Die Sprache ist die Menge der Binärkodierten Zahlen $L = \{x \in \mathbb{N} | \exists n \in \mathbb{N} : \ f(n) \geq x\} \subseteq \Sigma^*$. Das Wortproblem lautet also: Ist $k \in L$?

(c) Sei bspw. $f(x) = 2^{2^n}$. Folglich verbraucht das berechnen eines Wertes $\log_2 2^{2^n} = 2^n$, also exponentiell viel, Platz. Außerdem können wir nie sicher sein, dass $k \notin L$. Daher ist das Problem insbesondere unentscheidbar. Also $\chi_E \in$UNDEC.

Aufgabe 9.2

Nein, ist es nicht. (Beweis durch Widerspruch):

Sei H eine Hamilton Instanz über den Graphen G. Wir wählen jetzt ein beliebiges $v \in G$ und teilen es auf in v und v'. Der entstandenen Graph sei G' Daraus basteln wir eine Froschflucht Instanz indem wir $v = s$ und $v' = t$ setzen.

Angenommen Froschflucht wäre in deterministisch in polynomieller Zeit lösbar. Dann wüssten wir, wenn Froschflucht$(G') = |G|$ (größer geht nicht), dass es einen Hamiltonkreis gibt. Also wäre die Hamilton Instanz in poly Zeit gelöst. $\lightning$

Aufgabe 9.3

Eine Lösung ist $P' \subseteq P$, die alle Wünsche erfüllt. Das können wir einfach in polynomieller Zeit überprüfen: $\forall w_i \in W$: Ist $|(P' \cap P_i)| = k_i$?

Sei S eine 3-Sat Instanz. Diese wird zu folgendermaßen zur TSqP Instanz T: Für alle Variablen x_i aus S wird ein Wunsch genereriert, der die Form $(\{p_i, \bar{p}_i\}, 1)$. Für jede Klausel K_j wird ein Wunsch $(\{l_{i1}, l_{i2}, l_{i3}, o_{j1}, o_{j2}\}, 3)$ generiert, wobei $l_{ik} = p_{ik}$, wenn positiv oder $l_{ik} = \bar{p}_{ik}$ wenn negativ.

Eine Lösung für S erfüllt jeden der ersten Wünsche, da jede Variable wahr oder falsch ist und jeden der zweiten Wünsche, da in jeder Klausel mindestens ein Literal auf war ist, d.h. ein Baum gepflanzt wird. Um auf drei Orte zu kommen, können die extra Orte bepflanzt werden.

Eine Lösung für T ist eine Lösung für S, da auf jedem Literal, jeder Klausel mindestens ein Baum gepflanzt wurde und bei jeder Variablen entweder auf dem True oder dem False-Ort.

Aufgabe 9.4

Der Fehler ist, dass aus einem *false* von TSq geschlossen wird, dass es insgesamt keine Lösungen geben kann. Das ist aber nicht richtig.

Aufgabe 9.5

Algorithmus:

1. Füge für je zwei Kerzen k_1, k_2 eine Gerade $\{k_1, k_2\}$ zu G hinzu.

2. Iteriere für jede Gerade über alle anderen Geraden, wenn sie gleich sind verschmelze sie zu einer Gerade. Sobald eine Gerade mehr als k Kerzen trifft, lösche Sie und alle dazugehörigen Knoten.

Wissen: Jede Gerade hat maximal k Kerzen, d.h. selbst, wenn alle Geraden disjunkt sind, können wir maximal k^2 Kerzen auspusten. Die maximale Anzahl von Geraden ist dann k^4. Wenn $|G| \geq k^4$ finden wir keine Lösung. Jetzt können wir durch ausprobieren auf die Lösung kommen und die Laufzeit ist nur von k abhängig.

Aufgabe 8.6

Algoritmus:

for $i = 1, ..., k$:
    if $A(I) = true$: return *true*; break;
return *false*

Gegeben: q, p.

Gesucht: k, derart, dass $(1-p)^k > 1-q$

(a) $(1-\frac{1}{e})^k \geq (1-p)^k > 1-q$

Also wähle $k = \lceil \log_{(1-\frac{1}{e})} 1-p \rceil + 1$. Das ist weniger als konstant, also ist der Algorithmus auch polynomiell.

(b) $(1-\frac{1}{poly(n)})^k \geq (1-p)^k > 1-q$

Also wähle $k = \lceil \log_{(1-\frac{1}{poly(n)})} 1-p \rceil + 1$. k kann polynomiell von n abhängen, aber nicht mehr, da wenn p sehr klein wird, es durch ein Polynom trotzdem beliebig nah an $\frac{1}{e}$ herankommt. Also ist der Algorithmus auch polynomiell.