Algorithmen und
Berechnungskomplexität II
Sommersemester 2020

Abgabe: 22.06.2020, 12.00 Uhr
Besprechung: KW 26

Prof. Dr. Heiko Röglin
Anna Großwendt, David Kübel
Institut für Informatik

Übungsblatt 9

Aufgabe 9.1: Polynomielle Verifizierer

In der Vorlesung haben wir gezeigt, dass wir NP über Verifizierer charakterisieren können. Ein Verifizierer arbeitet dabei auf Wörtern $x\#y$, wobei x die Eingabe und y ein beliebiges Zertifikat ist, dessen Länge polynomiell in $|x|$ beschränkt ist. Was können wir über Probleme aussagen, für die ein Verifizierer existiert, der auch auf Wörtern $x\#y$ arbeitet, wobei x wieder die Eingabe und y wieder ein Zertifikat ist, dessen Länge allerdings linear in $\log(|x|)$ beschränkt ist?

Aufgabe 9.2: Polynomielle Reduktion

Betrachten Sie die folgenden Probleme.

Problem P_1
Gegeben: Ungerichteter Graph $G = (V, E)$ und ein $k \in \mathbb{N}$.
Frage: Gibt es eine Teilmenge $T \subseteq V$ der Knoten mit $|T| \geq k$, sodass keine zwei Knoten aus T durch eine Kante von G verbunden sind? (d.h. $\forall u, v \in T$ gilt $(u, v) \notin E$)

Problem P_2
Gegeben: Ungerichteter Graph $G = (V, E)$ und ein $k' \in \mathbb{N}$.
Frage: Gibt es eine Teilmenge $T' \subseteq V$ der Knoten mit $|T'| \leq k'$, sodass für jede Kante von G mindestens einer der inzidenten Knoten in T' liegt? (d.h. $\forall (u, v) \in E$ gilt $u \in T'$ oder $v \in T'$)

Geben Sie eine polynomielle Reduktion von P_1 auf P_2 an. Welche Schlüsse lassen sich daraus für P_1 und P_2 ziehen?

Aufgabe 9.3: Codierung von Eingaben

Sei $G = (V, E)$ ein ungerichteter Graph mit $V = \{0, \dots, n-1\}$. Beschreiben Sie eine Codierung von G

(a) in Adjazenzmatrix-Darstellung über dem Alphabet $\Sigma = \{0, 1\}$,

(b) in Adjazenzlisten-Darstellung über dem Alphabet $\Sigma = \{0, 1, \#\}$,

(c) in Adjazenzlisten-Darstellung über dem Alphabet $\Sigma = \{0, 1\}$.

Aufgabe 9.4: NTM vs. DTM

Entwerfen Sie eine nichtdeterministische $p(n)$-zeitbeschränkte Turingmaschine N, welche das `VERTEXCOVER` - Problem entscheidet, wobei n die Eingabelänge bezeichne. Konstruieren Sie daraus weiter eine deterministische Turingmaschine D, welche N simuliert wobei die Laufzeit von D in $2^{O(p(n))}$ liege.