Prof. T. Nipkow
K. Kappelmann, J. Rädle, L. Stevens **ABGABE 20.07.2020, 23:59 (Σ 6 P.)** Technische Universität München
Lehrstuhl für Logik und Verifikation

Einführung in die Theoretische Informatik

Sommersemester 2020 – Übungsblatt 12

AUFGABE 12.1.

0,5 + 0,5 + 1 P

Im Folgenden dürfen Sie zusätzlich zu den Basisfunktionen der primitiven Rekursion die erweiterte Komposition und das erweiterte rekursive Definitionsschema benutzen. Die in der Vorlesung behandelten primitiv rekursiven Funktionen und die Funktionen aus Tutoraufgabe 12.4 können ebenfalls verwendet werden. LOOP-Programme sind nicht erlaubt.
Zeigen Sie:

(a) Wenn die Funktion $f(x, y)$ primitiv rekursiv ist, dann ist auch die Funktion $g(x, z) = \sum_{i=0}^{z} f(x, i)$ primitiv-rekursiv.

(b) Die Funktion $\text{mod}(x, y)$ die den Rest der Ganzzahldivision $\frac{x}{y}$ auf natürlichen Zahlen berechnet, ist primitiv-rekursiv. (Im Fall $y = 0$ soll die Funktion 0 zurückgeben)

(c) Die Funktion

$$\begin{aligned} f(0) &= 1 \\ f(1) &= 1 \\ f(n+2) &= f(n+1) + f(n) \end{aligned}$$

ist primitiv-rekursiv. (*Hinweis:* Benutzen Sie die Cantorsche Paarungsfunktion)

AUFGABE 12.2.

1 + 1.5P

In der Vorlesung wurden einige Approximationsalgorithmen für NP-schwere Probleme gezeigt. Wir wollen in dieser Aufgabe zeigen, dass manche Probleme unter der Annahme $\mathsf{P} \neq \mathsf{NP}$ schwer zu approximieren sind. Um für ein Problem A zu zeigen, dass es keinen d-Approximationsalgorithmus geben kann, geht man folgendermaßen vor: Wir nehmen zunächst zum Widerspruch an, dass ein solcher Approximationsalgorithmus $\mathcal{A}$ existiert. Dann wählen wir ein geeignetes NP-schweres Entscheidungsproblem B, sodass wir für jede Probleminstanz b von B in polynomieller Zeit eine Probleminstanz a von A erzeugen können. Wenn wir nun $\mathcal{A}$ auf a anwenden und anhand des Lösungswerts ablesen können, ob die Antwort auf b "ja" oder "nein" ist, dann können wir B in polynomieller Zeit entscheiden. Das steht im Widerspruch zur NP-Schwere von B und damit kann es $\mathcal{A}$ nicht geben.

Wenden Sie diese Vorgehensweise an, um die folgenden Aussagen zu zeigen:

(a) Es kann keinen d-Approximationsalgorithmus mit $d < \frac{3}{2}$ für BIN PACKING geben. *Tipp:* Betrachten Sie die Reduktion von PARTITION auf BIN PACKING auf Folie 413.

(b) Wir betrachten das Problem TSP aus der Vorlesung mit der Modifikation, dass die Dreiecksungleichung nicht gilt. Zeigen Sie, dass es für kein $d \in \mathbb{N}$ einen d-Approximationsalgorithmus für TSP geben kann. *Tipp:* Betrachten Sie eine Reduktion von HAMILTONKREIS auf TSP.

AUFGABE 12.3. (*If you're lost, you can look and you will find me... time after time*)

1 + 0.5 + 0.5 + 0.5 + 0.5P

Wir führen nun Hausaufgabe 11.3 fort. Unser Ziel ist es zu zeigen, dass mehr Zeit/Speicher auch wirklich mehr Power bedeutet.

(a) Zeigen Sie: Es existieren unendlich viele, unterschiedliche Zeitkomplexitätsklassen. Verwenden Sie hierfür die Ergebnisse aus Hausaufgabe 11.3 (b).

(b) Zeigen Sie: Es existiert kein Polynom p, sodass $\mathsf{P} = \mathsf{TIME}(p(n))$ (man sagt auch, die Klasse P kollabiert nicht).

(c) (**Bonus**) In der Vorlesung wurde P unter anderem als die Menge der "leichten Probleme" bezeichnet. Diskutieren Sie diese Aussage im Licht des vorherigen Ergebnisses.

(d) (**Bonus**) Sei $f : \mathbb{N} \to \mathbb{N}$ berechenbar und total. Zeigen Sie: Es existiert eine entscheidbare Sprache $L' \notin \mathsf{SPACE}(f(n))$.

(e) (**Bonus**) Folgern Sie: Es existieren unendlich viele, unterschiedliche Speicherkomplexitätsklassen.

> Es gibt nichts praktischeres als eine gute Theorie.
>
> — Ludwig Boltzmann