Universität Paderborn
Prof. Dr. Johannes Blömer

Gruppe 27:
Eli Kogan-Wang (7251030)
David Noah Stamm (7249709)
Bogdan Rerich (7248483)
Jan Schreiber(7253698)

Berechenbarkeit und Komplexität – WS 2022/2023

Heimübung 12
Abgabe: 30. Januar 2023 – 13:00 Uhr
(Dieser Übungszettel besteht aus 3 Aufgaben mit insgesamt 24 Punkten)

Aufgabe 1 (Approximationsalgorithmen) (6 Punkte)
Wir betrachten die Optimierungsvariante einer Einschränkung von HittingSet (siehe Heimübung 9 Aufgabe 2). Die Menge der Instanzen $\mathcal{I}$ ist die Menge aller Paare (X, S), wobei X eine endliche Menge ist und $S \subseteq \mathcal{P}(X)$, sodass für alle $T \in S : |T| \leq 3$. Die Menge der gültigen Lösungen ist

$$F(I) = \{H \mid H \in \mathcal{P}(X) \text{ und für alle } T \in S : H \cap T \neq \emptyset\} .$$

Für $H \in F(I)$ ist die Zielfunktion $w(H) = |H|$. Es handelt sich um ein Minimierungsproblem. Betrachten Sie folgenden Approximationsalgorithmus.

```
H ← ∅;
for T ∈ S do
    if T ∩ H = ∅ then
        H ← H ∪ T;
    end
end
return H;
```

Algorithm 1: kleinerSchnitt (X, S)

Zeigen Sie, dass kleinerSchnitt eine Approximationsgüte von höchstens 3 hat.

Lösung:

Da nach Aufgabenstellung bereits gilt, dass KleinerSchnitt (X,S) ein Approximationsalgorithmus ist, ist nur noch die Approximationsgüte zu Zeigen:

Sei H die Ausgabe von kleinerSchnitt bei Eingabe einer Beliebigen, aber festen Instanz (X,S).

Behauptung: w(H)= $|H| \leq 3 \cdot |O|$, wobei O die Optimale Lösungsmenge ist.

Da vor jedem hinzufügen eines T $\in S$ geprüft wird, ob H $\cap$ T = $\emptyset$ gilt:

$H = \dot{\bigcup}_{T \in S}$ T eine disjunkte Vereinigung von allen T $\in S$.

AuSSerdem gilt: $O \in F(I)$ gilt für alle T $\in$ S, dass $O \cap T \neq \emptyset$.

Folglich gilt, dass (mindestens) ein Element aus jedem der T, welches zu H hinzugefügt werden, in O liegen muss.

$\implies \frac{1}{3}|H| \leq |O|$, da für jede Teilmenge T $\in$ S gilt $|T| = 3$ gilt

Also gilt Insgesamt: H $\leq 3 \cdot |O| \iff \frac{w(H)}{opt(H)} \leq 3$

$\implies$ Kleiner Schnitt hat einen Approximationsfaktor von 3 $\quad\square$.

Aufgabe 2 (Such- und Optimierungsvarianten) (12 Punkte)
Betrachten Sie folgende, aus der Vorlesung bekannte, Sprache.

$$\textsc{Rucksack} = \left\{ \langle G, W, g, w \rangle \;\middle|\; \begin{array}{c} G = \{g_1, \ldots, g_n\},\, W = \{w_1, \ldots, w_n\},\, g, w \in \mathbb{N}, \\ \text{für } 1 \leq i \leq n \text{ ist } g_i, w_i \in \mathbb{N}, \text{ und es} \\ \text{gibt } S \subseteq \{1, \ldots, n\} \text{ mit } \sum_{i \in S} g_i \leq g \text{ und } \sum_{i \in S} w_i \geq w. \end{array} \right\}$$

Gehen Sie im folgenden davon aus, dass Sie Zugriff auf ein *Orakel* $\mathfrak{O}_{\mathrm{RS}}$ haben. Gegeben eine Instanz $I = \langle G, W, g, w \rangle$ kann Ihnen dieses Orakel *in einem Zeitschritt* beantworten, ob I in Rucksack liegt, oder nicht.
Zeigen Sie: Eine DTM, die $\mathfrak{O}_{\mathrm{RS}}$ verwenden darf, kann

1. gegeben $\langle G, W, g, w \rangle \in$ Rucksack in polynomieller Zeit ein $S \subseteq \{1, \ldots, n\}$ berechnen, sodass $\sum_{i \in S} g_i \leq g$ und $\sum_{i \in S} w_i \geq w$.
2. gegeben $\langle G, W, g \rangle$ in *polynomieller* Zeit ein $w \in \mathbb{N}$ berechnen, sodass $\langle G, W, g, w \rangle \in$ Rucksack, aber $\langle G, W, g, (w+1) \rangle \notin$ Rucksack. Hierbei können Sie den Algorithmus aus Teilaufgabe 1 benutzen, müssen sich aber klar machen, was polynomiell in Bezug auf die Eingaben bedeutet.

Lösung:

1. Wir zeigen eine DTM M, die unter verwendung von M ein $S \subseteq \{1, \ldots, n\}$ berechnet, sodass $\sum_{i \in S} g_i \leq g$ und $\sum_{i \in S} w_i \geq w$.

   Bei Eingabe $\langle G, W, g, w \rangle \in$ Rucksack geht die DTM wie folgt vor:

   a) $idx \leftarrow 1$, $S \leftarrow \{1, \ldots, n\}$
   b) Solange $idx \leq n$:
      i. Prüfe, ob $\langle \{g_i | g \in S \setminus \{idx\}\}, \{w_i | w \in S \setminus \{idx\}\}, g, w \rangle \in$ Rucksack mit $\mathfrak{O}_{\mathrm{RS}}$:

A. Falls ja, dann $S \leftarrow S \setminus \{idx\}$

ii. $idx \leftarrow idx + 1$

c) Schreibe S auf das Band

Schleifeninvariante: Nach jeden Durchlauf von 1b gilt:

Für jedes S', mit $S' = S \setminus \{n\}$ mit $n \in \{1, \dots, idx - 1\}$ gilt:

$$\langle \{g_i | g \in S'\}, \{w_i | w \in S'\}, g, w\rangle \notin \textsc{Rucksack}$$

Und

$$\langle \{g_i | g \in S\}, \{w_i | w \in S\}, g, w\rangle \in \textsc{Rucksack}$$

Nachdem $idx = n + 1$ gilt, ist S eine solche gesuchte Menge, da es keine strikten Untermengen von S gibt, die in Rucksack lägen, aber S selbst in Rucksack liegt.

Der Algorithmus geht n mal durch die Schleife, und für jeden Durchlauf wird $\mathfrak{O}_{\mathrm{RS}}$ genau einmal aufgerufen.

Der Algorithmus ist offensichtlich polynomiell in der Eingabe.

2. Wir beschreiben eine DTM, die unter Verwendung von M ein $w \in \mathbb{N}$ berechnet, sodass $\langle G, W, g, w\rangle \in \textsc{Rucksack}$, aber $\langle G, W, g, (w + 1)\rangle \notin \textsc{Rucksack}$.

   Bei Eingabe von $\langle G, W, g\rangle$ geht die DTM wie folgt vor:

   a) $max_w \leftarrow \sum_{i=1}^{n} w_i$

   b) Begehe binäre Suche auf $w \in \{1, \dots, max_w\}$, nach einem w mit der gesuchten Eigenschaft

   Für die binäre Suche verwenden wir die DTM M aus Teilaufgabe 1, mit 2 Aufrufen:

   Einmal mit w und einmal mit $w + 1$.

   Der Suchbereich ist max_w groSS, und $max_w \in O(2^n)$, da die Summe der Gewichte binär kodiert werden kann.

   Die binäre Suche aber benötigt $O(\log(max_w)) = O(n)$ Schritte, sodass der Algorithmus polynomiell ist.

Aufgabe 3 (Rekursive Aufzählbarkeit und Entscheidbarkeit) (6 Punkte)

a) *Beweisen* Sie, ob folgende Sprache rekursiv aufzählbar oder, durch geeignete Reduktion, nicht rekursiv aufzählbar ist

$$L_a = \left\{ \langle M\rangle \;\middle|\; \begin{array}{l} M \text{ ist DTM und es gibt } w, w' \in \{0,1\}^*\text{, sodass DTM } M \\ \text{bei Eingabe } w \text{ hält und für die Konkatenation } ww' \text{ nicht hält.} \end{array} \right\}$$

b) *Beweisen* Sie, ob folgende Sprache entscheidbar oder, durch geeignete Reduktion, nicht entscheidbar ist

$$L_b = \left\{ \langle k, x, M_1, M_2, \ldots, M_k \rangle \;\middle|\; \begin{array}{l} k \in \mathbb{N}, x \in \{0,1\}^*,\ M_i \text{ ist DTM für alle } 1 \leq i \leq k \\ \text{und es gibt } I \subseteq \{1, 2, \ldots, k\}, |I| \geq k/2, \text{ sodass} \\ \text{für alle } i \in I \text{ DTM } M_i \text{ bei Eingabe } x \text{ hält.} \end{array} \right\}$$

Lösung:
a)

Behauptung: $H^c \leq L_a$
Betrachte folgende Reduktionsfunktion:

$$f(w) = \begin{cases} \langle M^{(x)} \rangle & \text{wenn w} = \langle M \rangle \text{x, wobei } \langle M \rangle \\ & \text{die Kodierung einer DTM ist und x} \in \{0,1\}^* \\ \langle M^{\epsilon} \rangle & \text{sonst} \end{cases}$$

Wobei $\langle M^{(x)} \rangle$ eine Turingmaschine ist, welche wie folgt definiert ist:

$\langle M^{(x)} \rangle$ bei Eingabe z $\in \{0,1\}^*$

1. Berechne |z|. Falls |z|=0, so akzeptiere.
2. Simuliere M bei Eingabe x für |z| Schritte.
3. Falls M in |z| Schritten hält, so akzeptiere z.
4. Sonst gehe in eine Endlosschleife.

und $\langle M^{\epsilon} \rangle$ die Turingmaschine, welche nur das leere Wort akzeptiert und sonst in eine endlosschleife geht.

Da die Einzige Änderung von $f(w)$ bezüglich der Turingmaschine in Satz 2.10.1 ist, dass hinzufügen einer Bedingung in (1), welche trivialerweise berechbar ist. Folglich ist f(w) auch berechbar.

Angenommen w $\in H^c$

Entweder w $\neq \langle M \rangle$x

$\implies f(w) = \langle M^{\epsilon} \rangle$, welche in $L(L_a)$ liegt,da für $w = \epsilon$ und $w' \in \{0,1\}^+$, die DTM bei Eingabe w hält, aber nicht für ww'.
oder w = $\langle M \rangle$x, wobei M eine DTM ist, welche bei Eingabe x nicht hält.

$\implies$ f(w)= $\langle M^{(x)} \rangle$

Wähle w= ϵ (= das leere Wort) und $w^{*\prime}$ = x

$\implies ww' = w' = x.$

$\implies$ f(w) $\in L_a$, da nach (1) w von M $\in$ L(M) und nach (4) M bei Eingabe w' in eine Endlosschleife geht.

Angenommen w $\notin H^c$
$\implies$ w = $\langle M \rangle$x, wobei M eine DTM ist, welche bei Eingabe x hält.

$\implies$ M hält nach
b)
Behauptung: $H \leq L_b$
Betrachte folgende Reduktionsfunktion:

$$f(w) = \begin{cases} \langle 2, x, M, M \rangle & \text{wenn w} = \langle M \rangle \text{x, wobei } \langle M \rangle \\ & \text{ die Kodierung einer DTM ist und x} \in \{0,1\}^* \\ \epsilon(\text{das leere Wort}) & \text{,wenn w} \neq \langle M \rangle \text{x sonst} \end{cases}$$

f ist trivialerweise berechenbar, da f nur die Reihenfolge der Eingabe verändert und eine Kodierung einer 2 und M auf das bannt kopiert, welches alles Operationen sind, von denen wir wissen, dass sie berechenbar sind.

Behauptung: w $\in H \iff$ f(w) $\in L_b$

Angenommen w $\in$ H.
$\implies$ w = $\langle M \rangle$x und M hält bei Eingabe x.
$\implies$ f(w)= $\langle 2, x, M, M \rangle$. Da nach Voraussetzung beide Turingmaschinen bei Eingabe x halten.

Folglich halten mehr als $\frac{2}{2}$=1 Turingmaschine, womit gilt: f(w) $\in L_b$

Angenommen w $\notin$ H.

$\implies$ Entweder w $\neq \langle M \rangle$x, wird damit auf ϵ abgebildet, welches nicht in L_b liegt

oder w = $\langle M \rangle$x und M hält bei Eingabe x nicht.

$\implies$ f(w)= $\langle 2, x, M, M \rangle$, aber da M nicht bei Eingabe x hält, hält keine der Turingmaschinen.

$\implies$ f(w) $\notin L_b$
$\implies$ w $\in H \iff$ f(w) $\in L_b$

$\implies$ w $\in H \iff$ f(w) $\in L_b$ $\square$

Da nach 2.9.2 gilt, dass H nicht entscheidbar ist, folgt mit 2.9.1, dass L_b nicht entscheidbar ist.