Theoretische Informatik - Übungsblatt 5
Übungsgruppe 7 - Mi, 16-18 Uhr
Elena Kvassova

Jan Hendrik Dithmar (Matrikelnummer 2031259)
Dirk Heine (Matrikelnummer 2030941)

Aufgabe 1

Konstruiere zunächst eine 2-Band-Turingmaschine M, die entscheidet, ob n eine Primzahl ist.

$$M = (Q, \Sigma, \Gamma, q_0, \delta, F) \quad \text{mit}$$

$$Q = \{q_0, \ldots, q_8\}, \quad \Sigma = \{1\}, \quad \Gamma = \{1, B\}, \quad F = \{q_7\}$$

δ	$(\mathbf{B}, \mathbf{B})$	$(\mathbf{B}, \mathbf{1})$	$(\mathbf{1}, \mathbf{B})$	$(\mathbf{1}, \mathbf{1})$
q_0	$(q_8, (B, B), (N, N))$	$(q_8, (B, 1), (N, N))$	$(q_1, (1, 1), (N, R))$	–
q_1	–	–	$(q_2, (1, 1), (N, L))$	–
q_2	$(q_7, (B, B), (N, N))$	$(q_7, (B, 1), (N, N))$	$(q_3, (1, B), (N, L))$	$(q_2, (1, 1), (R, R))$
q_3	–	–	$(q_4, (1, B), (N, R))$	$(q_3, (1, 1), (N, L))$
q_4	$(q_8, (B, B), (N, N))$	$(q_6, (B, 1), (L, N))$	$(q_5, (1, B), (N, L))$	$(q_4, (1, 1), (R, R))$
q_5	–	–	$(q_4, (1, B), (N, R))$	$(q_5, (1, 1), (N, L))$
q_6	$(q_2, (B, 1), (R, N))$	$(q_6, (B, 1), (N, L))$	$(q_6, (1, B), (L, N))$	$(q_6, (1, 1), (L, L))$
q_7	–	–	–	–
q_8	–	–	–	–

Hinweis: auf Band 1 steht die zu prüfende Zahl n, auf Band 2 steht die Laufvariable i in unärer Kodierung

q_0 Startzustand & Initialisierung, schreibe auf Band 2 eine 1

q_1 weitere Initialisierung, schreibe zweite 1 auf Band 2 (somit ist $i = 2$ entsprechend dem Wort „11“ auf Band 2)

q_2 taste Band 2 ab (solange 1 gelesen) und bewege Lesekopf auf Band 1 gleichzeitig nach rechts (erste Abtastung)

q_3 wenn zuvor Wortende auf Band 1 nicht erreicht, gehe an Anfang von Band 2 und zu Zustand q_4; wenn zuvor Wortende auf Band 1 erreicht, keine Teiler gefunden $\Rightarrow$ Primzahl

q_4 taste Band 2 ab (solange 1 gelesen) und bewege Lesekopf auf Band 1 gleichzeitig nach rechts (zweite und folgende Abtastungen)

q_5 wenn zuvor Wortende auf Band 1 nicht erreicht, gehe an Anfang von Band 2 und zu Zustand q_5; wenn zuvor Wortende auf Band 1 erreicht, diese Zahl kein Teiler $\Rightarrow$ Nachfolger prüfen (q_6)

q_6 gehe jeweils an Anfang von Band 1 und 2 zurück, inkrementiere Wert i auf Band 2 (eine 1 vorne anhängen), anschließend mit dieser Zahl Prüfung wiederholen

q_7 akzeptierender Endzustand, n ist eine Primzahl

q_8 nicht akzeptierender Endzustand, n ist *keine* Primzahl

Nun müssen wir noch beweisen, dass o.g. TM in ihrer Rechenzeit polynomiell beschränkt ist. Die äußere Schleife unserer TM wird höchstens $(n-1)$-mal ausgeführt und die innere Schleife höchstens $\lceil \frac{n}{i} \rceil$-mal. Jeder dieser Durchläufe der inneren Schleife besteht jeweils aus maximal $i \leq n$ Schritten. Somit ist die Laufzeit in Abhängigkeit von n in $\mathcal{O}(n^3)$ und damit polynomiell beschränkt. □

Aufgabe 2

Folgende Wörter können nach maximal 5 Schritten auf dem Band stehen:

$$0, 1, 00, 10, 01, 000, 100, 010, 001, 0000, 1000, 0100, 0010, 0001$$

Aufgabe 3

Wir konstruieren eine nicht-deterministische Turingmaschine wie folgt. Die Bandinhalte werden niemals geändert, es wird immer genau das geschrieben was auch gelesen wurde. In Zustand q_0 wird nach Übereinstimmung zwischen dem aktuellen Zeichen aus Wort w_1 und dem ersten Zeichen von w_2 gesucht.

- Wird eine Übereinstimmung $(0, 0)$ oder $(1, 1)$ gefunden, gehen wir zu Zustand q_1 über (dabei gehen wir in w_1 *und* w_2 ein Zeichen nach rechts) und vergleichen, ob wirklich ein Vorkommen von w_2 in w_1 vorliegt. Dazu wird geprüft, ob das aktuelle Zeichen aus w_1 und das aktuelle Zeichen aus w_2 gleich sind.
  - Wird $(0, 0)$ oder $(1, 1)$ gelesen, gehen wir in beiden Worten jeweils ein Zeichen nach rechts (immer noch Übereinstimmung).
  - Wird $(0, B)$, $(1, B)$ oder (B, B) gelesen, haben wir ein zusammenhängendes Vorkommen von w_2 in w_1 gefunden (Ende von w_2 erreicht, bis dahin nur Übereinstimmung).
  - Wird jedoch $(0, 1)$ oder $(1, 0)$ (keine weitere Übereinstimmung) bzw. $(B, 0)$ oder $(B, 1)$ (Ende von w_1 aber noch nicht Ende von w_2 erreicht) gelesen, dann haben wir auf diesem Rechenweg keine Übereinstimmung gefunden.
  - Wird schließlich $(B, 0)$ oder $(B, 1)$ gelesen, haben wir das Ende von w_1 erreicht, *ohne* hier eine vollständige Übereinstimmung mit w_2 zu finden.

  Zusätzlich können auch in Zustand q_0 bleiben, gehen in Wort w_1 auf Band 1 ein Zeichen nach rechts und lassen den Lesekopf auf Band 2 überverändert stehen. Damit prüfen wir auf Vorkommen von w_2 in w_1 jedoch beginnend *hinter* der aktuellen Lesekopf-Position auf Band 1.
- Wird in Zustand q_0 bereits (B, B), $(0, B)$ oder $(1, B)$ gelesen, dann heißt das, dass das Wort w_2 leer ist (in q_0 steht der Lesekopf von Band 2 immer auf dem ersten Zeichen von w_2). Das leere Wort ist immer ein zusammenhängendes Teilwort von w_1.
- Wird in Zustand q_0 entweder $(B, 0)$ oder $(B, 1)$ gelesen, ist das Ende von Wort w_1 erreicht und wir können keine Vorkommen von w_2 in w_1 ab dieser Position mehr finden.
- In den Fällen $(0, 1)$ bzw. $(1, 0)$ liegt keine mögliche Startposition für ein Vorkommen von w_2 in w_1 vor und wir bewegen den Lesekopf von Band 1 ein Zeichen nach rechts. Somit prüfen wir auf Vorkommen *hinter* der aktuellen Position in w_1.

Gelangt ein Rechenweg in den Zustand q_2, so wurde auf diesem Rechenweg *kein* Vorkommen von w_2 in w_1 gefunden. Gelangt ein Rechenweg jedoch in den Zustand q_3, dann wurde ein Vorkommen gefunden.

$$M = (Q, \Sigma, \Gamma, q_0, \delta, F) \quad \text{mit}$$

$$Q = \{q_0, \ldots, q_3\}, \quad \Sigma = \mathbb{B}, \quad \Gamma = \{0, 1, B\}, \quad F = \{q_3\}$$

$$\begin{aligned}
\delta = \{ & \\
& (q_0, (B,B), q_3, (B,B), (N,N)), && \Rightarrow w_2 \text{ ist leer} \\
& (q_0, (0,B), q_3, (0,B), (N,N)), && \Rightarrow w_2 \text{ ist leer} \\
& (q_0, (1,B), q_3, (1,B), (N,N)), && \Rightarrow w_2 \text{ ist leer} \\
& (q_0, (B,0), q_2, (B,0), (N,N)), && \Rightarrow \text{ Ende von } w_1 \text{ erreicht} \\
& (q_0, (B,1), q_2, (B,1), (N,N)), && \Rightarrow \text{ Ende von } w_1 \text{ erreicht} \\
& (q_0, (0,0), q_1, (0,0), (R,R)), && \Rightarrow \text{ mögl. Startposition gef.} \\
& (q_0, (1,1), q_1, (1,1), (R,R)), && \Rightarrow \text{ mögl. Startposition gef.} \\
& (q_0, (0,0), q_0, (0,0), (R,N)), && \Rightarrow \text{ weiter hinten suchen} \\
& (q_0, (0,1), q_0, (0,1), (R,N)), && \Rightarrow \text{ weiter hinten suchen} \\
& (q_0, (1,0), q_0, (1,0), (R,N)), && \Rightarrow \text{ weiter hinten suchen} \\
& (q_0, (1,1), q_0, (1,1), (R,N)), && \Rightarrow \text{ weiter hinten suchen} \\
& && \\
& (q_1, (0,0), q_1, (0,0), (R,R)), && \Rightarrow \text{ weiterhin Übereinstimmung} \\
& (q_1, (1,1), q_1, (1,1), (R,R)), && \Rightarrow \text{ weiterhin Übereinstimmung} \\
& (q_1, (0,1), q_2, (0,1), (N,N)), && \Rightarrow \text{ keine weitere Übereinst.} \\
& (q_1, (1,0), q_2, (1,0), (N,N)), && \Rightarrow \text{ keine weitere Übereinst.} \\
& (q_1, (0,B), q_3, (0,B), (N,N)), && \Rightarrow \text{ Ende von } w_2\text{, Erfolg} \\
& (q_1, (1,B), q_3, (1,B), (N,N)), && \Rightarrow \text{ Ende von } w_2\text{, Erfolg} \\
& (q_1, (B,B), q_3, (B,B), (N,N)), && \Rightarrow \text{ Ende von } w_2\text{, Erfolg} \\
& (q_1, (B,0), q_2, (B,0), (N,N)), && \Rightarrow \text{ vorzeitiges Ende von } w_1 \\
& (q_1, (B,1), q_2, (B,1), (N,N)) && \Rightarrow \text{ vorzeitiges Ende von } w_1 \\
\} &
\end{aligned}$$

Aufgabe 4

(a) Nach Voraussetzung ist der Algorithmus A für das BPP gegeben. Wir haben also einen Algorithmus, um für gegebene $a_1, \ldots, a_n, b$ das minimale k zu ermitteln. Jetzt haben wir jedoch ein k' gegeben und suchen das minimale b'. Dazu durchlaufen wir mit dem Algorithmus A alle Möglichkeiten von

$$b \in \left\{ \min_i(a_i), \ldots, \sum_i a_i \right\}$$

in aufsteigender Reihenfolge. Ein $b < \min_i(a_i)$ ist nicht sinnvoll, da sonst die Bedingung $\sum_{f(i)=j} a_i \leq b$ verletzt würde, wenn k minimal ist (es also keine leeren Behälter gibt). Wir finden auch spätestens bei $b = \sum_i a_i$ eine Lösung, wenn nämlich alle Objekte in einen Behälter gelegt werden, was dem kleinsten sinnvollen Wert $k' = 1$ entspricht. Wir finden auch zuerst das minimale b', da wir die Möglichkeiten von b in aufsteigender Reihenfolge durchlaufen. Das gesuchte b' ist gleich dem ersten b für das $k \leq k'$ ist. Die Gleichheit $k = k'$ ist *nicht* erforderlich, da für das modifizierte BPP keine Minimierung der Behälteranzahl k' gefordert ist und wir somit die ,,überflüssigen" Behälter einfach leer lassen könnnen.

(b) Nach Voraussetzung ist der Algorithmus A' für das modifizierte BPP gegeben. Wir haben also einen Algorithmus, um für gegebene $a_1, \ldots, a_n, k'$ das minimale b' zu ermitteln. Jetzt haben wir jedoch ein b gegeben und suchen das minimale k. Wir durchlaufen mit dem Algorithmus A' alle Möglichkeiten von

$$k \in \{1, \ldots, n\}$$

in aufsteigender Reihenfolge. Ein $k < 1$ ist nicht sinnvoll, es sei denn $n = 0$, für diesen Sonderfall ist die Lösung jedoch trivial. Ein $k > n$ ist ebenfalls nicht sinnvoll, da wir nur n Objekte haben und wir diese nicht auf einen größere Anzahl von Behältern aufteilen können (leere Behälter würden der Minimierung von k widersprechen). Wir finden in diesem Bereich auch auf jeden Fall eine Lösung, sofern die Vorgabe von b sinnvoll ist, also $b \geq \max_i a_i$ (jeder Behälter muss mindestens so groß sein, um das größte Objekt aufnehmen zu können). Wir erhalten also spätestens bei $k = n$ eine Lösung. Wir finden auch hier mit Sicherheit zuerst das minimale k, da wir k' in aufsteigender Reihenfolge betrachten. Das gesuchte k ist gleich dem ersten k' für das $b' \leq b$ ist. Auch hier ist die Gleichheit $b' = b$ *nicht* erforderlich, da die potentiell kleineren Behälter zwar eine stärkere Aussage geben, jedoch ein kleineres k' mit einem größeren $b' \leq b$ bereits vorher gefunden worden wäre.