Institut für Informatik **Ludwig-Maximilians-Universität München**
Prof. Dr. Volker Heun

Formale Sprachen und Komplexität, SS 18
Tutoriumsblatt 11

Aufgabe 11-1 NP-Vollständigkeit

Das Problem *RUCKSACK* ist im Buch folgendermaßen definiert:

$$RUCKSACK = \{\ (a_1, \ldots, a_k, b) \in \mathbb{N}^{k+1} \mid \exists I \subset \{1, \ldots, k\} : \sum_{i \in I} a_i = b\ \}$$

Es ist bekannt, dass *RUCKSACK* NP-vollständig ist.

Das Problem *PARTITION* ist im Buch folgendermaßen definiert:

$$PARTITION = \{\ (a_1, \ldots, a_l) \in \mathbb{N}^{l} \mid \exists J \subset \{1, \ldots, l\} : \sum_{j \in J} a_j = \sum_{j \notin J} a_j\ \}$$

Zeigen Sie, dass *PARTITION* NP-vollständig ist.

Lösungsvorschlag:

Um zu zeigen, dass *PARTITION* NP-vollständig ist, müssen wir zeigen:

1. für alle Sprachen L in NP gilt: $L \leq_p PARTITION$ (*PARTITION* ist NP-hart)
2. $PARTITION \in \mathsf{NP}$

zu 1. Hätten wir eine NP-vollständige Sprache B, für die $B \leq_p PARTITION$ gilt, dann wüssten wir für alle Sprachen L in NP: $L \leq_p B \leq_p PARTITION$.

Wir suchen also eine NP-vollständige Sprache B mit $B \leq_p PARTITION$. Die Aufgabenstellung legt nahe, für B *RUCKSACK* zu wählen.

Zeige, dass $RUCKSACK \leq_p PARTITION$. Man braucht also eine Funktion von *RUCKSACK* nach *PARTITION*, bzw. von $\mathbb{N}^{k+1}$ nach $\mathbb{N}^{l}$.

Sei $(a_1, \ldots, a_k, b)$ ein *RUCKSACK*-Problem. Sei weiter $M := \sum_{i=1}^{k} a_i$.

Wir definieren die Abbildung:

$$\begin{array}{lll} \phi : \mathbb{N}^{k+1} & \to & \mathbb{N}^{k+2} \\ (a_1, \ldots, a_k, b) & \mapsto & (a_1, \ldots, a_k,\ M - b + 1,\ b + 1) \end{array}$$

Vom Himmel fallende Behauptung: ϕ vermittelt eine Reduktion von *RUCKSACK* auf *PARTITION*! (l ist jetzt $k+2$, das heißt wenn man ein *RUCKSACK*-Problem der Länge $k+1$ Lösen kann, dann kann man ein *PARTITION*-Problem der Länge $k+2$ lösen. Da *RUCKSACK*-Probleme mindestens 1 lang sind, und *PARTITION*-Probleme mindestens 2, hat man kein Problem mit den Dimensionen.)

Jetzt ist noch zu zeigen:
Für alle $x = (a_1, \ldots, a_k, b) \in \mathbb{N}^{k+1}$ gilt: $x \in \mathit{RUCKSACK} \Leftrightarrow \phi(x) \in \mathit{PARTITION}$.

'$\Rightarrow$' Sei $x = (a_1, \ldots, a_k, b)$ in $\mathit{RUCKSACK}$, mit der lösenden Menge $I \subset \{1, \ldots, k\}$.

Dann ist $\sum_{i \in I} a_i = b$, nach Definition.

Untersuche

$$\phi(x) = (a_1, \ldots, a_k, M - b + 1, b + 1)$$

Jetzt brauchen wir hierfür eine Lösung des $\mathit{PARTITION}$-Problems.
Wähle $J \subset \{1, \ldots, k+2\}$:

$$J = I \cup \{k+1\}$$

Bilde die Summen über alle Einträge $j \in J$ und $j \notin J$ in $\phi(x)$ und zeige, dass diese gleich sind.

$$\begin{aligned}
& \sum_{j \in J} \phi(x)_j = \sum_{j \notin J} \phi(x)_j \\
\Leftrightarrow \quad & \underbrace{\sum_{i \in I} a_i}_{=b} + \underbrace{(M - b + 1)}_{k+1ste\ Stelle} = \underbrace{\sum_{i \notin I} a_i}_{=M-b} + \underbrace{(b+1)}_{k+2te\ Stelle} \\
\Leftrightarrow \quad & M + 1 = M + 1
\end{aligned}$$

Also ist $\phi(x)$ in $\mathit{PARTITION}$.

'$\Leftarrow$' Sei $y = (a_1, \ldots, a_k,\ M - b + 1,\ b + 1) \in \mathit{PARTITION}$.
Dann gibt es eine Menge $J \subset \{1, \ldots, k+2\}$, sodass $\sum_{j \in J} y_j = \sum_{j \notin J} y_j$. Betrachte die Summe über alle Elemente von y. Es ist

$$a_1 + \cdots + a_k + (M - b + 1) + (b + 1) = 2M + 2$$

Also müssen die Summen über die Partitionen beide den Wert $M + 1$ haben.

$M - b + 1$ und $b + 1$ können nicht in der selben Partition liegen, denn dann wäre die Partitionssumme größer als $M + 2$. Angenommen $M - b + 1$ ist in der Partition J, $b + 1$ nicht.
Dann ist

$$\begin{aligned}
\sum_{j \in J} y_j &= \sum_{j \in J \setminus \{k+1\}} a_j + (M - b + 1) = M + 1 \\
\Rightarrow \sum_{j \in J \setminus \{k+1\}} a_j &= b
\end{aligned}$$

Also ist $J \setminus \{k+1\}$ eine Lösung des ursprünglichen $\mathit{RUCKSACK}$-Problems.

Da ϕ **total** und **mit polynomialem Aufwand berechenbar** ist ($\phi \in O(k)$, also linear), sind wir fertig. $\mathit{PARTITION}$ ist NP-hart.

zu 2. Wir zeigen, dass $PARTITION \in \mathsf{NP}$ ist durch einen nichtdeterministischen Entscheidungsalgorithmus für $PARTITION$:

Für Eingabe $(a_1, \ldots, a_l)$	
1. Wähle nichtdeterministisch eine Teilmenge $J \subseteq \{1, \ldots, l\}$;	$O(l)$
2. Prüfe, ob $\sum_{j \in J} a_j = \sum_{j \notin J} a_j$;	$O(l)$

Dieser informelle Algorithmus kann von einer nichtdeterministischen (Mehrband-)Turingmaschine ausgeführt werden.

Sie akzeptiert jedes $(a_1, \ldots, a_l) \in PARTITION$
mit einer Berechnung der Länge $O(2l)$.
Das ist proportional (und damit polynomial) zur Länge von $(a_1, \ldots, a_l)$.

Also ist $PARTITION \in \mathsf{NP}$.

Aufgabe 11-2 Minsum

MinSum:		*PARTITION*:	
gegeben:	$(a_1, ...a_k, b_1, ..., b_k) \in \mathbb{N}^{2k}$	*gegeben:*	$(a_1, ...a_l) \in \mathbb{N}^l$
gefragt:	$\exists\, J \subseteq \{1, ..., k\}$:	*gefragt:*	$\exists\, I \subseteq \{1, ..., l\}$:
	$\sum_{i \in J} \min(a_i, b_i) = \sum_{i \notin I} \min(a_i, b_i)$		$\sum_{i \in I} a_i = \sum_{i \notin I} a_i$

Zeigen Sie: *PARTITION* $\leq_p$ *MinSum*.

Lösungsvorschlag:

Betrachten wir die Funktion

$$\begin{aligned} \phi: \quad & \mathbb{N}^k \to \mathbb{N}^{2k} \\ & (a_1, ..., a_k) \mapsto (a_1, ..., a_k, a_1 + 1, ..., a_k + 1) \end{aligned}$$

Die Funktion ϕ hat die Eigenschaft:

$x \in$ *PARTITION* $\Leftrightarrow \phi(x) \in$ *MinSum*.

Beweis $\Rightarrow$: Angenommen $x = (a_1, ...a_k) \in$ *PARTITION*.
Dann existiert eine Indexmenge I, sodass $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$.

$\phi(x)$ hat nun die Form $(a_1, ...a_k, b_1, ..., b_k)$, wobei jeweils gilt: $b_i = a_i + 1$.

Als Lösungsmenge für $\phi(x)$ kann jetzt einfach $J = I$ gewählt werden, da $\min(a_i, a_i + 1) = a_i$, und damit:

$$\sum_{i \in J} \min(a_i, b_i) = \sum_{i \in J} a_i = \sum_{i \notin J} a_i = \sum_{i \notin J} \min(a_i, b_i)$$

Beweis $\Leftarrow$: Angenommen $\phi(x) = (a_1, ...a_k, a_1 + 1, ..., a_k + 1) \in$ *MinSum*.
Dann existiert eine Indexmenge J, sodass $\sum_{i \in J} \min(a_i, a_i + 1) = \sum_{i \notin J} \min(a_i, a_i + 1)$.

Wiederum kann als lösende Menge für $x = (a_1, ...a_k) \in$ *PARTITION* einfach $I = J$ gewählt werden, da

$$\sum_{i \in I} a_i = \sum_{i \in I} \min(a_i, a_i + 1) = \sum_{i \notin I} \min(a_i, a_i + 1) = \sum_{i \notin I} a_i$$

Die Funktion ϕ hat somit die gesuchte Eigenschaft. Außerdem ist sie offensichtlich total und in polynomieller Zeit berechenbar: Alle verwendeten Funktionen sind total, die Nachfolgerfunktion ist in konstanter Zeit berechenbar, und das konstruierte Tupel hat die Länge $2k$.

Damit ist gezeigt: *PARTITION* $\leq_p$ *MinSum*.