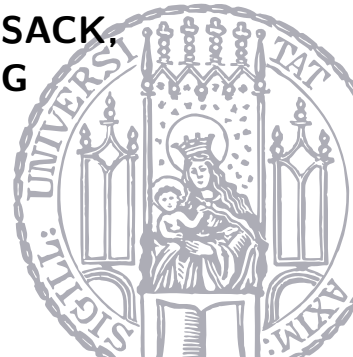
$\mathcal{NP}$-Vollständigkeit von SETCOVER, SUBSETSUM, KNAPSACK, PARTITION und BINPACKING

Prof. Dr. David Sabel

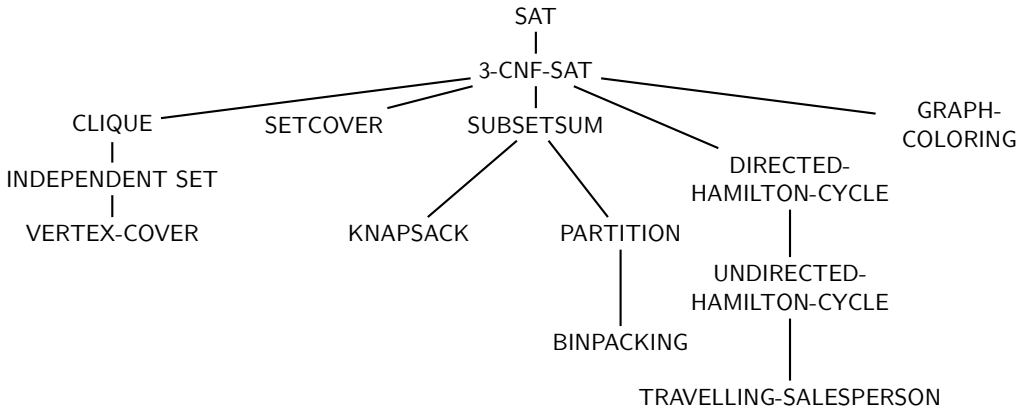
LFE Theoretische Informatik



Letzte Änderung der Folien: 24. Juli 2022

Inhalt der kommenden Vorlesungen

$\mathcal{NP}$-Vollständigkeitsbeweise für eine Auswahl an Problemen.



Heute: $\mathcal{NP}$-Vollständigkeit von SETCOVER, SUBSETSUM, KNAPSACK, PARTITION, BINPACKING

Das SETCOVER-Problem

Definition (SETCOVER-Problem)

Das SETCOVER-Problem lässt sich in der gegeben/gefragt-Notation formulieren durch:

gegeben: Mengen $T_1, \ldots, T_k$ mit $T_1, \ldots, T_k \subseteq M$, wobei M eine endliche Grundmenge ist und eine Zahl $n \leq k$.

gefragt: Gibt es eine Auswahl von n Mengen $T_{i_1}, \ldots, T_{i_n}$ $(i_j \in \{1, \ldots, k\})$ mit $T_{i_1} \cup \ldots \cup T_{i_n} = M$?

Das SETCOVER-Problem

Definition (SETCOVER-Problem)

Das SETCOVER-Problem lässt sich in der gegeben/gefragt-Notation formulieren durch:

gegeben: Mengen $T_1, \ldots, T_k$ mit $T_1, \ldots, T_k \subseteq M$, wobei M eine endliche Grundmenge ist und eine Zahl $n \leq k$.

gefragt: Gibt es eine Auswahl von n Mengen $T_{i_1}, \ldots, T_{i_n}$ $(i_j \in \{1, \ldots, k\})$ mit $T_{i_1} \cup \ldots \cup T_{i_n} = M$?

Beispiel:

$T_1 = \{1, 2, 3, 5\}, T_2 = \{1, 2\}, T_3 = \{3, 4\}, T_4 = \{3\}$ mit $M = \{1, 2, 3, 4, 5\}$ und $n = 2$

Das SETCOVER-Problem

Definition (SETCOVER-Problem)

Das SETCOVER-Problem lässt sich in der gegeben/gefragt-Notation formulieren durch:

gegeben: Mengen $T_1, \ldots, T_k$ mit $T_1, \ldots, T_k \subseteq M$, wobei M eine endliche Grundmenge ist und eine Zahl $n \leq k$.

gefragt: Gibt es eine Auswahl von n Mengen $T_{i_1}, \ldots, T_{i_n}$ $(i_j \in \{1, \ldots, k\})$ mit $T_{i_1} \cup \ldots \cup T_{i_n} = M$?

Beispiel:

$T_1 = \{1, 2, 3, 5\}, T_2 = \{1, 2\}, T_3 = \{3, 4\}, T_4 = \{3\}$ mit $M = \{1, 2, 3, 4, 5\}$ und $n = 2$

Lösung: T_1, T_3, da $T_1 \cup T_3 = M$.

$\mathcal{NP}$-Vollständigkeit von SETCOVER

Satz

SETCOVER ist $\mathcal{NP}$-vollständig.

Beweis, Teil 1: SETCOVER $\in \mathcal{NP}$

- Rate nichtdeterministisch die n Mengen $T_{i_1}, \ldots, T_{i_n}$.
- Verifiziere deterministisch, ob $T_{i_1} \cup \cdots \cup T_{i_n} = M$ gilt.
- Daher kann SETCOVER in Polynomialzeit auf einer NTM entschieden werden.

$\mathcal{NP}$-Vollständigkeit von SETCOVER (2)

Beweis, Teil 2: SETCOVER ist $\mathcal{NP}$-schwer.

- Ziel: 3-CNF-SAT $\leq_p$ SETCOVER.
- Sei $F = K_1 \wedge \ldots \wedge K_m$ eine 3-CNF.
- Seien $x_1, \ldots, x_n$ die in F vorkommenden aussagenlogischen Variablen.
- Setze $M = \{1, \ldots, m, m+1, \ldots, m+n\}$.
- Für $i = 1, \ldots, n$ sei

$$T_{i,a} = \{j \mid \text{Literal } x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+i\}$$
$$T_{i,b} = \{j \mid \text{Literal } \neg x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+i\}$$

- Das Mengensystem sei $T_{1,a}, \ldots, T_{n,a}, T_{1,b}, \ldots, T_{n,b} \subseteq M$.

$\mathcal{NP}$-Vollständigkeit von SETCOVER (3)

Zur Erinnerung:
$$T_{i,a} = \{j \mid \text{Literal } x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+i\}$$
$$T_{i,b} = \{j \mid \text{Literal } \neg x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+i\}$$

Sei I Belegung mit $I(F) = 1$.

- Wenn $I(x_i) = 1$, dann wähle $T_{i,a}$, sonst wähle $T_{i,b}$
- Ergibt n gewählte Mengen
- Jede Zahl aus M kommt vor:
  - Da jede Klausel durch I erfüllt, kommen alle 1,..., m vor
  - Da jede Variable x_i mit 0 oder 1 belegt wird, kommen alle $m+1, \ldots, m+n$ vor
- Vereinigung der n Mengen ergibt daher M.
- SETCOVER ist lösbar.

$\mathcal{NP}$-Vollständigkeit von SETCOVER (4)

Zur Erinnerung: $T_{i,a} = \{j \mid \text{Literal } x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+i\}$
$T_{i,b} = \{j \mid \text{Literal } \neg x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+i\}$

Seien $U_1, \ldots, U_n \subseteq T_{1,a}, \ldots, T_{n,a}, T_{1,b}, \ldots, T_{n,b}$ mit $U_1 \cup \ldots \cup U_n = M$

- Da $m+1, \ldots, m+n$ in der Vereinigung, muss für jedes $i = 1, \ldots, n$ genau entweder $T_{i,a}$ oder $T_{i,b}$ in der Vereinigung sein.
- O.B.d.A. $U_i \in \{T_{i,a}, T_{i,b}\}$
- Setze $I(x_i) = 1$ wenn $U_i = T_{i,a}$ und $I(x_i) = 0$ wenn $U_i = T_{i,b}$
- Da $1, \ldots, m$ in der Vereinigung, wird in jeder Klausel ein Literal durch I wahr gemacht
- F ist erfüllbar.

Da diese Übersetzung in Polynomialzeit berechenbar ist, gilt 3-SAT-CNF $\leq_p$ SETCOVER.

Beispiel

3-CNF:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$$

Beispiel

3-CNF:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$$

SET-COVER-Instanz dazu:

$$\begin{aligned}
T_{1,a} &= \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{1,3,5\} \\
T_{1,b} &= \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{2,4,5\} \\
T_{2,a} &= \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{1,4,6\} \\
T_{2,b} &= \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{2,3,6\}
\end{aligned}$$

Beispiel

3-CNF:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$$

SET-COVER-Instanz dazu:

$$\begin{aligned}
T_{1,a} &= \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{1,3,5\} \\
T_{1,b} &= \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{2,4,5\} \\
T_{2,a} &= \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{1,4,6\} \\
T_{2,b} &= \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{2,3,6\}
\end{aligned}$$

Gesucht: Vereinigung von $n = 2$ Mengen $T_{i,j}, T_{i',j'}$ mit $T_{i,j} \cup T_{i',j'} = \{1,2,3,4,5,6\}$

Beispiel

3-CNF:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$$

SET-COVER-Instanz dazu:

$$\begin{aligned}
T_{1,a} &= \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{1,3,5\} \\
T_{1,b} &= \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{2,4,5\} \\
T_{2,a} &= \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{1,4,6\} \\
T_{2,b} &= \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{2,3,6\}
\end{aligned}$$

Gesucht: Vereinigung von $n = 2$ Mengen $T_{i,j}, T_{i',j'}$ mit $T_{i,j} \cup T_{i',j'} = \{1,2,3,4,5,6\}$

Keine Lösung!

Beispiel (2)

3-CNF:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

Beispiel (2)

3-CNF:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

SET-COVER-Instanz dazu:

$$\begin{aligned}
T_{1,a} &= \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{1,2,4\} \\
T_{1,b} &= \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{3,4\} \\
T_{2,a} &= \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{1,5\} \\
T_{2,b} &= \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{2,3,5\} \\
T_{3,a} &= \{j \mid x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} &&= \{1,3,6\} \\
T_{3,b} &= \{j \mid \neg x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} &&= \{6\}
\end{aligned}$$

Beispiel (2)

3-CNF:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

SET-COVER-Instanz dazu:

$$\begin{aligned}
T_{1,a} &= \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{1,2,4\} \\
T_{1,b} &= \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{3,4\} \\
T_{2,a} &= \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{1,5\} \\
T_{2,b} &= \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{2,3,5\} \\
T_{3,a} &= \{j \mid x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} &&= \{1,3,6\} \\
T_{3,b} &= \{j \mid \neg x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} &&= \{6\}
\end{aligned}$$

Gesucht: Vereinigung von $n = 3$ Mengen $T_{i,j}, T_{i',j'}$ mit $T_{i,j} \cup T_{i',j'} = \{1,2,3,4,5,6\}$

Beispiel (2)

3-CNF:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

SET-COVER-Instanz dazu:

$$\begin{aligned}
T_{1,a} &= \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{1,2,4\} \\
T_{1,b} &= \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{3,4\} \\
T_{2,a} &= \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{1,5\} \\
T_{2,b} &= \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{2,3,5\} \\
T_{3,a} &= \{j \mid x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} &&= \{1,3,6\} \\
T_{3,b} &= \{j \mid \neg x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} &&= \{6\}
\end{aligned}$$

Gesucht: Vereinigung von $n = 3$ Mengen $T_{i,j}, T_{i',j'}$ mit $T_{i,j} \cup T_{i',j'} = \{1,2,3,4,5,6\}$

Lösung z.B. $T_{1,a}, T_{2,b}, T_{3,b}$

Beispiel (2)

3-CNF:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

SET-COVER-Instanz dazu:

$$\begin{aligned}
T_{1,a} &= \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{1,2,4\} \\
T_{1,b} &= \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} &&= \{3,4\} \\
T_{2,a} &= \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{1,5\} \\
T_{2,b} &= \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} &&= \{2,3,5\} \\
T_{3,a} &= \{j \mid x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} &&= \{1,3,6\} \\
T_{3,b} &= \{j \mid \neg x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} &&= \{6\}
\end{aligned}$$

Gesucht: Vereinigung von $n = 3$ Mengen $T_{i,j}, T_{i',j'}$ mit $T_{i,j} \cup T_{i',j'} = \{1,2,3,4,5,6\}$

Lösung z.B. $T_{1,a}, T_{2,b}, T_{3,b}$

Belegung dazu $I(x_1) = 1, I(x_2) = 0, I(x_3) = 0$

Das SUBSETSUM-Problem

Definition (SUBSETSUM-Problem)

Das SUBSETSUM-Problem lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Natürliche Zahlen $a_1, \ldots, a_k \in \mathbb{N}$ und $s \in \mathbb{N}$
gefragt: Gibt es eine Teilmenge $I \subseteq \{1, \ldots, k\}$,
sodass $\sum_{i \in I} a_i = s$?

Beispiel: $a_1, \ldots, a_6 = 1, 21, 5, 16, 12, 19$ und $s = 49$

Lösung: $2, 4, 5$, da $21 + 16 + 12 = 49$

$\mathcal{NP}$-Vollständigkeit von SUBSETSUM

Satz

Das SUBSETSUM-Problem ist $\mathcal{NP}$-vollständig.

Beweis, Teil 1: SUBSETSUM $\in \mathcal{NP}$:

- Rate nichtdeterministisch eine Teilmenge $I \subseteq \{1, \ldots, k\}$
- Prüfe (deterministisch) ob $\sum_{i \in I} a_i = s$ gilt
- Daher: SUBSETSUM in Polynomialzeit auf NTM entscheidbar.

$\mathcal{NP}$-Vollständigkeit von SUBSETSUM (2)

SUBSETSUM ist $\mathcal{NP}$-schwer: Zeige 3-CNF-SAT $\leq_p$ SUBSETSUM

- Sei $F = K_1 \wedge \cdots \wedge K_m$ eine 3-CNF
- Annahme: Alle Klauseln K_j bestehen aus genau 3 Literalen (Vermehrfache Literale anderenfalls)
- Seien $Var(F) = \{x_1, \ldots, x_n\}$ die Variablen in F

$\mathcal{NP}$-Vollständigkeit von SUBSETSUM (2)

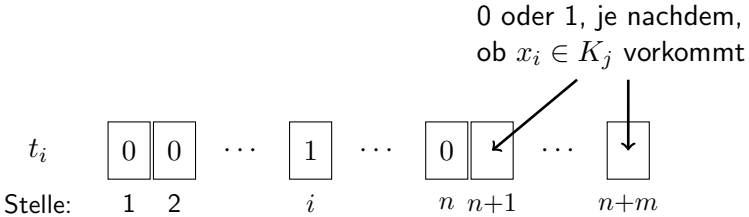
Konstruktion der SUBSETSUM-Instanz:

Erzeuge $n+m$-stellige Zahlen t_i, f_i für $i = 1, \ldots, n$:

$\mathcal{NP}$-Vollständigkeit von SUBSETSUM (2)

Konstruktion der SUBSETSUM-Instanz:

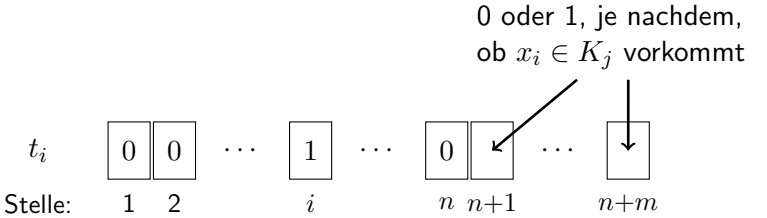
Erzeuge $n+m$-stellige Zahlen t_i, f_i für $i = 1, \ldots, n$:



$\mathcal{NP}$-Vollständigkeit von SUBSETSUM (2)

Konstruktion der SUBSETSUM-Instanz:

Erzeuge $n+m$-stellige Zahlen t_i, f_i für $i = 1, \ldots, n$:



$\mathcal{NP}$-Vollständigkeit von SUBSETSUM (3)

Weitere Zahlen:
Erzeuge $n+m$-stellige Zahlen c_j, d_j für $j = 1, \ldots, m$:

$\mathcal{NP}$-Vollständigkeit von SUBSETSUM (3)

Weitere Zahlen:
Erzeuge $n+m$-stellige Zahlen c_j, d_j für $j = 1, \ldots, m$:

c_j	0	0	$\cdots$	0	0	$\cdots$	1	$\cdots$	0
Stelle:	1	2		n	$n+1$		$n+j$		$n+m$
d_j	0	0	$\cdots$	0	0	$\cdots$	2	$\cdots$	0
Stelle:	1	2		n	$n+1$		$n+j$		$n+m$

$\mathcal{NP}$-Vollständigkeit von SUBSETSUM (3)

Weitere Zahlen:
Erzeuge $n+m$-stellige Zahlen c_j, d_j für $j = 1, \ldots, m$:

c_j	0	0	$\cdots$	0	0	$\cdots$	1	$\cdots$	0
Stelle:	1	2		n	$n+1$		$n+j$		$n+m$

d_j	0	0	$\cdots$	0	0	$\cdots$	2	$\cdots$	0
Stelle:	1	2		n	$n+1$		$n+j$		$n+m$

Zielsumme s:

s	1	1	$\cdots$	1	4	4	$\cdots$	4
Stelle:	1	2		n	$n+1$	$n+2$		$n+m$

$\mathcal{NP}$-Vollständigkeit von SUBSETSUM (4)

Sei also $f(F) = ((t_1, \ldots, t_n, f_1, \ldots, f_n, c_1, \ldots, c_m, d_1, \ldots, d_m), s)$.

Es gilt:

- Die Summe jeder Teilmenge der Zahlen erzeugt keine Überträge.
- Die n Einsen in s sorgen dafür, dass in I jeweils t_i oder f_i enthalten ist aber nicht beide.
- Die Wahl der t_i und f_i in I zählt gleichzeitig in den Stellen $n+1$ bis $n+m$, welche Klauseln durch Setzen von x_i auf 1 (bzw. x_i auf 0) wahr gemacht werden.
- In der Summe kann dies pro Stelle j eine Zahl zwischen 0 und 3 sein
- Durch Hinzunahme der c_j und/oder d_j kann die Zielsumme 4 pro Stelle j erreicht werden, wenn mindestens 1 Literal wahr ist.

$\mathcal{NP}$-Vollständigkeit von SUBSETSUM (5)

Sei I Lösung der SUBSETSUM-Instanz $((t_1, \ldots, t_n, f_1, \ldots, f_n, c_1, \ldots, c_m, d_1, \ldots, d_m), s)$.
Konstruiere Belegung B für F:

- Setze für $i \in I$ mit $1 \leq i \leq n$: $B(x_i) = 1$
- Setze für $i \in I$ mit $n+1 \leq n+i \leq n+n$: $B(x_i) = 0$.
- Macht in jeder Klausel mindestens 1 Literal wahr

$\mathcal{NP}$-Vollständigkeit von SUBSETSUM (5)

Sei I Lösung der SUBSETSUM-Instanz $((t_1, \ldots, t_n, f_1, \ldots, f_n, c_1, \ldots, c_m, d_1, \ldots, d_m), s)$.
Konstruiere Belegung B für F:

- Setze für $i \in I$ mit $1 \leq i \leq n$: $B(x_i) = 1$
- Setze für $i \in I$ mit $n+1 \leq n+i \leq n+n$: $B(x_i) = 0$.
- Macht in jeder Klausel mindestens 1 Literal wahr

Sei B erfüllende Belegung für F. Konstruiere Indexmenge I:

- I enthält den Index von t_i wenn $B(x_i) = 1$,
- I enthält den Index von f_i wenn $B(x_i) = 0$
- I enthält die Indizes der c_j, d_j, sodass sich die hinteren m Stellen zu 4 aufsummieren: Immer möglich, da für jede Stelle die Summe schon mindestens 1 ist (da B erfüllende Belegung).

$\mathcal{NP}$-Vollständigkeit von SUBSETSUM (5)

Sei I Lösung der SUBSETSUM-Instanz $((t_1, \ldots, t_n, f_1, \ldots, f_n, c_1, \ldots, c_m, d_1, \ldots, d_m), s)$.
Konstruiere Belegung B für F:

- Setze für $i \in I$ mit $1 \leq i \leq n$: $B(x_i) = 1$
- Setze für $i \in I$ mit $n+1 \leq n+i \leq n+n$: $B(x_i) = 0$.
- Macht in jeder Klausel mindestens 1 Literal wahr

Sei B erfüllende Belegung für F. Konstruiere Indexmenge I:

- I enthält den Index von t_i wenn $B(x_i) = 1$,
- I enthält den Index von f_i wenn $B(x_i) = 0$
- I enthält die Indizes der c_j, d_j, sodass sich die hinteren m Stellen zu 4 aufsummieren: Immer möglich, da für jede Stelle die Summe schon mindestens 1 ist (da B erfüllende Belegung).

Damit folgt: 3-CNF-SAT $\leq_p$ SUBSETSUM.

Beispiel

$F = (x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_1)$
wird übersetzt in:

$$
\begin{array}{lll}
a_1 & = t_1 & = 110 \\
a_2 & = f_1 & = 101 \\
a_3 & = c_1 & = 010 \\
a_4 & = c_2 & = 001 \\
a_5 & = d_1 & = 020 \\
a_6 & = d_2 & = 002 \\
 & s & = 144
\end{array}
$$

(Unlösbar)

Beispiel (2)

$F = (x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee x_4)$
wird übersetzt in

$$
\begin{array}{lll}
a_1 = t_1 = 100010100 & a_9 = c_1 = 000010000 & a_{14} = d_1 = 000020000 \\
a_2 = t_2 = 010011010 & a_{10} = c_2 = 000001000 & a_{15} = d_2 = 000002000 \\
a_3 = t_3 = 001000001 & a_{11} = c_3 = 000000100 & a_{16} = d_3 = 000000200 \\
a_4 = t_4 = 000100101 & a_{12} = c_4 = 000000010 & a_{17} = d_4 = 000000020 \\
a_5 = f_1 = 100001010 & a_{13} = c_5 = 000000001 & a_{18} = d_5 = 000000002 \\
a_6 = f_2 = 010000101 & & \\
a_7 = f_3 = 001001010 & & s = 111144444 \\
a_8 = f_4 = 000110000 & &
\end{array}
$$

Lösung $I = \{2, 4, 5, 7, 9, 10, 11, 12, 13, 14, 16, 18\}$.

Erfüllende Belegung $B(x_1) = 0, B(x_2) = 1, B(x_3) = 0, B(x_4) = 1$.

Das Rucksack-Problem

Definition (KNAPSACK-Problem)

Das KNAPSACK-Problem lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: k Gegenstände mit Gewichten $w_1, \ldots, w_k \in \mathbb{N}$ und
Nutzenwerten $n_1, \ldots, n_k \in \mathbb{N}$,
sowie zwei Zahlen $s_w, s_n \in \mathbb{N}$.

gefragt: Gibt es Teilmenge $I \subseteq \{1, \ldots, k\}$, sodass $\sum_{i \in I} w_i \leq s_w$ und $\sum_{i \in I} n_i \geq s_n$?

Das Rucksack-Problem

Definition (KNAPSACK-Problem)

Das KNAPSACK-Problem lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: k Gegenstände mit Gewichten $w_1, \ldots, w_k \in \mathbb{N}$ und
Nutzenwerten $n_1, \ldots, n_k \in \mathbb{N}$,
sowie zwei Zahlen $s_w, s_n \in \mathbb{N}$.

gefragt: Gibt es Teilmenge $I \subseteq \{1, \ldots, k\}$, sodass $\sum_{i \in I} w_i \leq s_w$ und $\sum_{i \in I} n_i \geq s_n$?

Beachte für $w_i = n_i$ und $s_n = s_w$ ergibt sich genau das SUBSETSUM-Problem!

$\mathcal{NP}$-Vollständigkeit von KNAPSACK

Satz

KNAPSACK ist $\mathcal{NP}$-vollständig.

$\mathcal{NP}$-Vollständigkeit von KNAPSACK

Satz

KNAPSACK ist $\mathcal{NP}$-vollständig.

Beweis:

KNAPSACK $\in \mathcal{NP}$:

- Rate eine Teilmenge $I \subseteq \{1, \ldots, k\}$ nichtdeterministisch
- Prüfe deterministisch, ob $\sum_{i \in I} w_i \leq s_w$ und $\sum_{i \in I} n_i \geq s_n$
- Daher kann KNAPSACK in Polynomialzeit auf einer NTM entschieden werden.

$\mathcal{NP}$-Vollständigkeit von KNAPSACK

Satz

KNAPSACK ist $\mathcal{NP}$-vollständig.

Beweis:

KNAPSACK $\in \mathcal{NP}$:

- Rate eine Teilmenge $I \subseteq \{1, \ldots, k\}$ nichtdeterministisch
- Prüfe deterministisch, ob $\sum_{i \in I} w_i \leq s_w$ und $\sum_{i \in I} n_i \geq s_n$
- Daher kann KNAPSACK in Polynomialzeit auf einer NTM entschieden werden.

KNAPSACK ist $\mathcal{NP}$-schwer:

- Sei $((a_1, \ldots, a_k), s)$ eine SUBSETSUM-Instanz
- Sei $f((a_1, \ldots, a_k), s) = ((w_1, \ldots, w_k), (n_1, \ldots, n_k), s_w, s_m)$ mit $w_i = a_i, n_i = a_i$ für $i = 1, \ldots, k$ und $s_w = s$ und $s_m = s$.
- $((a_1, \ldots, a_k), s)$ lösbar g.d.w. $f((a_1, \ldots, a_k), s)$ lösbar
- f ist in polynomieller Zeit von einer DTM berechenbar.
- SUBSETSUM $\leq_p$ KNAPSACK.

Das PARTITION-Problem

Definition (PARTITION-Problem)

Das PARTITION-Problem lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Natürliche Zahlen $a_1, \ldots, a_k \in \mathbb{N}$

gefragt: Gibt es eine Teilmenge $I \subseteq \{1, \ldots, k\}$, sodass $\sum_{i \in I} a_i = \sum_{i \in \{1,\ldots,k\} \setminus I} a_i$?

$\mathcal{NP}$-Vollständigkeit von PARTITION

Satz

PARTITION ist $\mathcal{NP}$-vollständig.

Beweis:

PARTITION $\in \mathcal{NP}$

- Rate nichtdeterministisch $I \subseteq \{1, \ldots, k\}$
- Prüfe deterministisch, ob $\sum_{i \in I} a_i = \sum_{i \in \{1,\ldots,k\} \setminus I} a_i$
- Daher kann PARTITION in Polynomialzeit auf einer NTM entschieden werden.

$\mathcal{NP}$-Vollständigkeit von PARTITION (2)

PARTITION ist $\mathcal{NP}$-schwer: Wir zeigen SUBSETSUM $\leq_p$ PARTITION

- Sei $f((a_1, \ldots, a_k), s) = (a_1, \ldots, a_k, a_{k+1}, a_{k+2})$
  mit $a_{k+1} = A + s$ und $a_{k+2} = 2A - s$, wobei $A = \sum_{i=1}^{k} a_k$.
- f kann von einer DTM in polynomieller Zeit berechnet werden.
- Wenn $(a_1, \ldots, a_k, a_{k+1}, a_{k+2})$ Lösung $I \subseteq \{1, \ldots, k+2\}$ hat,
  dann $\sum_{i \in I} a_i = 2A = \sum_{i \in \{1,\ldots,k+2\} \setminus I} a_i$.
  - Nicht $k+1$ und $k+2$ in I, da sonst die Summe zu groß ist.
  - Wenn $k+2 \in I$, dann $I' = I \setminus \{k+2\}$.
  - Wenn $k+1 \in I$, dann $I' = \{1, \ldots, k\} \setminus I$.
  - In beiden Fällen ist $\sum_{i \in I'} = s$ und I' ist Lösung von $((a_1, \ldots, a_k), s)$.

  Wenn $I \subseteq \{1, \ldots, k\}$ eine Lösung $((a_1, \ldots, a_k), s)$, dann ist $I \cup \{k+2\}$ eine Lösung für $(a_1, \ldots, a_k, a_{k+1}, a_{k+2})$.
- Daher gilt $((a_1, \ldots, a_k), s)$ lösbar g.d.w. $f((a_1, \ldots, a_k), s)$ lösbar
- SUBSETSUM $\leq_p$ PARTITION

Beispiel

- Sei ((1,2,3,4,5,6),14) eine SUBSETSUM-Instanz.
- PARTITION-Instanz dazu: (1,2,3,4,5,6,35,28)
- Lösung $I = \{1, 3, 4, 6, 8\}$ da $1 + 3 + 4 + 6 + 28 = 42 = 2 + 5 + 35$.
- Lösung der SUBSETSUM-Instanz $\{1, 3, 4, 6\}$

Das BINPACKING-Problem

Definition (BINPACKING-Problem)

Das BINPACKING-Problem lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Natürliche Zahlen $a_1, \ldots, a_k \in \mathbb{N}$, die Behältergröße $b \in \mathbb{N}$ und die Anzahl der Behälter m

gefragt: Kann man alle gegeben Zahlen so auf die Behälter aufteilen, sodass keiner der Behälter überläuft?
Formal: Gibt es eine totale Funktion $assign : \{1, \ldots, k\} \to \{1, \ldots, m\}$, sodass für alle $j \in \{1, \ldots, m\}$ gilt: $\sum_{assign(i)=j} a_i \leq b$?

$\mathcal{NP}$-Vollständigkeit von BINPACKING

Satz

BINPACKING ist $\mathcal{NP}$-vollständig.

Beweis: BINPACKING $\in \mathcal{NP}$

- Rate nichtdeterministisch für jede Zahl a_i in welchen Behälter sie gehört.
- Prüfe, dass die geratene Zuordnung keinen Behälter überlaufen lässt.
- BINPACKING kann daher in Polynomialzeit auf einer NTM entschieden werden.

$\mathcal{NP}$-Vollständigkeit von BINPACKING (2)

BINPACKING ist $\mathcal{NP}$-schwer

- Sei $(a_1, \ldots, a_k)$ eine PARTITION-Instanz.
- Sei $f(a_1, \ldots, a_k)$ die BINPACKING-Instanz mit Zahlen $a_1, \ldots, a_k$, Behältergröße $b = \left\lfloor \frac{\sum_{i=1}^{k} a_i}{2} \right\rfloor$ und $m = 2$ Behältern.
- Wenn $\sum_{i=1}^{k} a_i$ ungerade, dann ist die PARTITION-Instanz unlösbar und die BINPACKING-Instanz ebenso.
- Wenn $I \subseteq \{1, \ldots, k\}$ Lösung für PARTITION, dann ist $assign(i) = \begin{cases} 1, & \text{wenn } i \in I \\ 2, & \text{wenn } i \notin I \end{cases}$ eine Lösung für BINPACKING.
- Mit $I = \{i \mid 1 \leq i \leq k, assign(i) = 1\}$ kann aus Lösung für BINPACKING eine Lösung für PARTITION erstellt werden.
- f ist polynomiell berechenbar: PARTITION $\leq_p$ BINPACKING.