Script **generated by TTT**

Title: Seidl: Theoretische_Informatik (15.07.2013)

Date: Mon Jul 15 10:15:35 CEST 2013

Duration: 89:34 min

Pages: 108

Weitere Anwendungen von SAT:

Bounded Model Checking

Weitere Anwendungen von SAT:

Bounded Model Checking

Hardware Entscheide, ob eine Schaltung mit Zustand für alle Eingaben innerhalb von 10 Zyklen ein bestimmtes Verhalten (nicht) hat.

Weitere Anwendungen von SAT:

Bounded Model Checking

Hardware Entscheide, ob eine Schaltung mit Zustand für alle Eingaben innerhalb von 10 Zyklen ein bestimmtes Verhalten (nicht) hat.

Software Entscheide, ob ein Programm für alle Eingaben innerhalb von 10 Schritten ein bestimmtes Verhalten (nicht) hat.

Weitere Anwendungen von SAT:

Bounded Model Checking

Hardware Entscheide, ob eine Schaltung mit Zustand für alle Eingaben innerhalb von 10 Zyklen ein bestimmtes Verhalten (nicht) hat.

Software Entscheide, ob ein Programm für alle Eingaben innerhalb von 10 Schritten ein bestimmtes Verhalten (nicht) hat. Variablen müssen auf sehr kleine Wertebereiche eingeschränkt werden!

5.4 Weitere NP-vollständige Probleme

5.4 Weitere NP-vollständige Probleme

Wie zeigt man, dass ein (weiteres) Problem B NP-vollständig ist?

- Zeige $B \in$ NP (meist trivial)

5.4 Weitere NP-vollständige Probleme

Wie zeigt man, dass ein (weiteres) Problem B NP-vollständig ist?

- Zeige $B \in$ NP (meist trivial)
- Zeige $A \leq_p B$ für ein NP-vollständiges Problem A.

5.4 Weitere NP-vollständige Probleme

Wie zeigt man, dass ein (weiteres) Problem B NP-vollständig ist?

- Zeige $B \in$ NP (meist trivial)
- Zeige $A \leq_p B$ für ein NP-vollständiges Problem A.

Lemma 5.23
Ist A NP-vollständig, so ist $B \in$ NP ebenfalls NP-vollständig, falls $A \leq_p B$.

5.4 Weitere NP-vollständige Probleme

Wie zeigt man, dass ein (weiteres) Problem B NP-vollständig ist?

- Zeige $B \in$ NP (meist trivial)
- Zeige $A \leq_p B$ für ein NP-vollständiges Problem A.

Lemma 5.23
Ist A NP-vollständig, so ist $B \in$ NP ebenfalls NP-vollständig, falls $A \leq_p B$.

Beweis:
Folgt direkt aus der Transitivität von $\leq_p$:
B ist NP-hart, denn für $C \in$ NP gilt $C \leq_p A \leq_p B$. □

Warum will man wissen, dass ein Problem B NP-vollständig ist?
Um sicher zu sein, dass

- ein polynomieller Algorithmus für B ein Durchbruch wäre

Warum will man wissen, dass ein Problem B NP-vollständig ist?
Um sicher zu sein, dass

- ein polynomieller Algorithmus für B ein Durchbruch wäre
- und daher wahrscheinlich nicht existiert.

Warum will man wissen, dass ein Problem B NP-vollständig ist?
Um sicher zu sein, dass

- ein polynomieller Algorithmus für B ein Durchbruch wäre
- und daher wahrscheinlich nicht existiert.

Praktische Instanzen von B könnten trotzdem (zB mit SAT) „effizient“ lösbar sein.

Definition 5.24

- Eine Formel ist in Konjunktiver Normalform (KNF) gdw sie eine Konjunktion von Klauseln ist: $K_1 \wedge \cdots \wedge K_n$

Definition 5.24

- Eine Formel ist in Konjunktiver Normalform (KNF) gdw sie eine Konjunktion von Klauseln ist: $K_1 \wedge \cdots \wedge K_n$
- Eine Klausel ist eine Disjunktion von Literalen: $L_1 \vee \cdots \vee L_m$
- Ein Literal ist eine (evtl. negierte) Variable.

Definition 5.24

- Eine Formel ist in Konjunktiver Normalform (KNF) gdw sie eine Konjunktion von Klauseln ist: $K_1 \wedge \cdots \wedge K_n$
- Eine Klausel ist eine Disjunktion von Literalen: $L_1 \vee \cdots \vee L_m$
- Ein Literal ist eine (evtl. negierte) Variable.
- Eine Formel ist in 3KNF gdw jede Klausel ≤ 3 Literale enthält.

Definition 5.24

- Eine Formel ist in Konjunktiver Normalform (KNF) gdw sie eine Konjunktion von Klauseln ist: $K_1 \wedge \cdots \wedge K_n$
- Eine Klausel ist eine Disjunktion von Literalen: $L_1 \vee \cdots \vee L_m$
- Ein Literal ist eine (evtl. negierte) Variable.
- Eine Formel ist in 3KNF gdw jede Klausel ≤ 3 Literale enthält.

Dh eine KNF ist ein Konjunktion von Disjunktionen von (evtl negierten) Variablen.
Bsp: $(x_9 \vee \neg x_2) \wedge (\neg x_7 \vee x_1 \vee x_6)$

Definition 5.24

- Eine Formel ist in Konjunktiver Normalform (KNF) gdw sie eine Konjunktion von Klauseln ist: $K_1 \wedge \cdots \wedge K_n$
- Eine Klausel ist eine Disjunktion von Literalen: $L_1 \vee \cdots \vee L_m$
- Ein Literal ist eine (evtl. negierte) Variable.
- Eine Formel ist in 3KNF gdw jede Klausel ≤ 3 Literale enthält.

Dh eine KNF ist ein Konjunktion von Disjunktionen von (evtl negierten) Variablen.
Bsp: $(x_9 \vee \neg x_2) \wedge (\neg x_7 \vee x_1 \vee x_6)$

3KNF-SAT

Gegeben: Ein Formel in 3KNF

Definition 5.24

- Eine Formel ist in Konjunktiver Normalform (KNF) gdw sie eine Konjunktion von Klauseln ist: $K_1 \wedge \cdots \wedge K_n$
- Eine Klausel ist eine Disjunktion von Literalen: $L_1 \vee \cdots \vee L_m$
- Ein Literal ist eine (evtl. negierte) Variable.
- Eine Formel ist in 3KNF gdw jede Klausel ≤ 3 Literale enthält.

Dh eine KNF ist ein Konjunktion von Disjunktionen von (evtl negierten) Variablen.
Bsp: $(x_9 \vee \neg x_2) \wedge (\neg x_7 \vee x_1 \vee x_6)$

3KNF-SAT

Gegeben: Ein Formel in 3KNF

Problem: Ist die Formel erfüllbar?

Satz 5.25

3KNF-SAT ist NP-vollständig.

Satz 5.25
3KNF-SAT ist NP-vollständig.

Beweis:
Wir zeigen SAT $\leq_p$ 3KNF-SAT mit einer polynomiellen Reduktion $F \mapsto F'$ so dass F' in 3KNF ist

Satz 5.25
3KNF-SAT ist NP-vollständig.

Beweis:
Wir zeigen SAT $\leq_p$ 3KNF-SAT mit einer polynomiellen Reduktion $F \mapsto F'$ so dass F' in 3KNF ist und

$$F \text{ ist erfüllbar } \Leftrightarrow \ F' \text{ ist erfüllbar}$$

Satz 5.25
3KNF-SAT ist NP-vollständig.

Beweis:
Wir zeigen SAT $\leq_p$ 3KNF-SAT mit einer polynomiellen Reduktion $F \mapsto F'$ so dass F' in 3KNF ist und

$$F \text{ ist erfüllbar } \Leftrightarrow \ F' \text{ ist erfüllbar}$$

NB F und F' sind erfüllbarkeitsäquivalent, aber nicht notwendigerweise auch äquivalent.

Satz 5.25
3KNF-SAT ist NP-vollständig.

Beweis:
Wir zeigen SAT $\leq_p$ 3KNF-SAT mit einer polynomiellen Reduktion $F \mapsto F'$ so dass F' in 3KNF ist und

$$F \text{ ist erfüllbar } \Leftrightarrow \ F' \text{ ist erfüllbar}$$

NB F und F' sind erfüllbarkeitsäquivalent, aber nicht notwendigerweise auch äquivalent.

1. Transformiere F in Negations-Normalform (NNF) durch fortgesetze Anwendung der de Morganschen Gesetze

Satz 5.25
3KNF-SAT ist NP-vollständig.

Beweis:
Wir zeigen SAT $\leq_p$ 3KNF-SAT mit einer polynomiellen Reduktion $F \mapsto F'$ so dass F' in 3KNF ist und

$$F \text{ ist erfüllbar } \Leftrightarrow \ F' \text{ ist erfüllbar}$$

NB F und F' sind erfüllbarkeitsäquivalent, aber nicht notwendigerweise auch äquivalent.

1. Transformiere F in Negations-Normalform (NNF) durch fortgesetze Anwendung der de Morganschen Gesetze

$$\begin{aligned} \neg(A \wedge B) &= \neg A \vee \neg B \\ \neg(A \vee B) &= \neg A \wedge \neg B \end{aligned}$$

Satz 5.25
3KNF-SAT ist NP-vollständig.

Beweis:
Wir zeigen SAT $\leq_p$ 3KNF-SAT mit einer polynomiellen Reduktion $F \mapsto F'$ so dass F' in 3KNF ist und

$$F \text{ ist erfüllbar } \Leftrightarrow \ F' \text{ ist erfüllbar}$$

NB F und F' sind erfüllbarkeitsäquivalent, aber nicht notwendigerweise auch äquivalent.

1. Transformiere F in Negations-Normalform (NNF) durch fortgesetze Anwendung der de Morganschen Gesetze

$$\begin{aligned} \neg(A \wedge B) &= \neg A \vee \neg B \\ \neg(A \vee B) &= \neg A \wedge \neg B \end{aligned}$$

$$\neg\neg A = A$$

Satz 5.25
3KNF-SAT ist NP-vollständig.

Beweis:
Wir zeigen SAT $\leq_p$ 3KNF-SAT mit einer polynomiellen Reduktion $F \mapsto F'$ so dass F' in 3KNF ist und

$$F \text{ ist erfüllbar } \Leftrightarrow \ F' \text{ ist erfüllbar}$$

NB F und F' sind erfüllbarkeitsäquivalent, aber nicht notwendigerweise auch äquivalent.

1. Transformiere F in Negations-Normalform (NNF) durch fortgesetze Anwendung der de Morganschen Gesetze

$$\begin{aligned} \neg(A \wedge B) &= \neg A \vee \neg B \\ \neg(A \vee B) &= \neg A \wedge \neg B \\ \neg\neg A &= A \end{aligned}$$

von links nach rechts.

Satz 5.25
3KNF-SAT ist NP-vollständig.

Beweis:
Wir zeigen SAT $\leq_p$ 3KNF-SAT mit einer polynomiellen Reduktion $F \mapsto F'$ so dass F' in 3KNF ist und

$$F \text{ ist erfüllbar } \Leftrightarrow \ F' \text{ ist erfüllbar}$$

NB F und F' sind erfüllbarkeitsäquivalent, aber nicht notwendigerweise auch äquivalent.

1. Transformiere F in Negations-Normalform (NNF) durch fortgesetze Anwendung der de Morganschen Gesetze

$$\begin{aligned} \neg(A \wedge B) &= \neg A \vee \neg B \\ \neg(A \vee B) &= \neg A \wedge \neg B \\ \neg\neg A &= A \end{aligned}$$

von links nach rechts. F_1 ist Resultat.

Beweis (Forts.):

Für F_1 gilt: $\neg$ nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.
Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.

Beweis (Forts.):

Für F_1 gilt: $\neg$ nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.
Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.
Ordne dabei der Wurzel von F_1 die Variable y_0 zu.

3. Betrachte die y_i als Abkürzung für die Teilbäume, an deren Wurzeln sie stehen

$$y_i = \begin{array}{c} \circ_i \\ / \quad \backslash \\ l_i \quad r_i \end{array}$$

Beweis (Forts.):

Für F_1 gilt: $\neg$ nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.
Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.
Ordne dabei der Wurzel von F_1 die Variable y_0 zu.

3. Betrachte die y_i als Abkürzung für die Teilbäume, an deren Wurzeln sie stehen

$$y_i = \begin{array}{c} \circ_i \\ / \quad \backslash \\ l_i \quad r_i \end{array}$$

wobei $\circ_i \in \{\wedge, \vee\}$ und l_i, r_i ein Literal oder eine Variable y_j ist.

Beweis (Forts.):

Für F_1 gilt: $\neg$ nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.
Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.
Ordne dabei der Wurzel von F_1 die Variable y_0 zu.

3. Betrachte die y_i als Abkürzung für die Teilbäume, an deren Wurzeln sie stehen

$$y_i = \begin{array}{c} \circ_i \\ / \quad \backslash \\ l_i \quad r_i \end{array}$$

wobei $\circ_i \in \{\wedge, \vee\}$ und l_i, r_i ein Literal oder eine Variable y_j ist.
Beschreibe F_1 Knoten für Knoten:

$$y_0 \wedge (y_0 \leftrightarrow (l_0 \circ_0 r_0))$$

Beweis (Forts.):

Für F_1 gilt: $\neg$ nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.
Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.
Ordne dabei der Wurzel von F_1 die Variable y_0 zu.

3. Betrachte die y_i als Abkürzung für die Teilbäume, an deren Wurzeln sie stehen

$$y_i = \begin{array}{c} \circ_i \\ / \quad \backslash \\ l_i \quad r_i \end{array}$$

wobei $\circ_i \in \{\wedge, \vee\}$ und l_i, r_i ein Literal oder eine Variable y_j ist.
Beschreibe F_1 Knoten für Knoten:

$$y_0 \wedge (y_0 \leftrightarrow (l_0 \circ_0 r_0)) \wedge (y_1 \leftrightarrow (l_1 \circ_1 r_1)) \dots$$

Beweis (Forts.):

Für F_1 gilt: $\neg$ nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.
Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.
Ordne dabei der Wurzel von F_1 die Variable y_0 zu.

3. Betrachte die y_i als Abkürzung für die Teilbäume, an deren Wurzeln sie stehen

$$y_i = \begin{array}{c} \circ_i \\ / \quad \backslash \\ l_i \quad r_i \end{array}$$

wobei $\circ_i \in \{\wedge, \vee\}$ und l_i, r_i ein Literal oder eine Variable y_j ist.
Beschreibe F_1 Knoten für Knoten:

$$y_0 \wedge (y_0 \leftrightarrow (l_0 \circ_0 r_0)) \wedge (y_1 \leftrightarrow (l_1 \circ_1 r_1)) \dots =: F_2$$

Beweis (Forts.):

F_1 erf. $\implies$ F_2 erf.: y_i bekommt Wert seines Teilbaums.

Beweis (Forts.):

Für F_1 gilt: $\neg$ nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.
Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.
Ordne dabei der Wurzel von F_1 die Variable y_0 zu.

3. Betrachte die y_i als Abkürzung für die Teilbäume, an deren Wurzeln sie stehen

$$y_i = \begin{array}{c} \circ_i \\ / \quad \backslash \\ l_i \quad r_i \end{array}$$

wobei $\circ_i \in \{\wedge, \vee\}$ und l_i, r_i ein Literal oder eine Variable y_j ist.
Beschreibe F_1 Knoten für Knoten:

$$y_0 \wedge (y_0 \leftrightarrow (l_0 \circ_0 r_0)) \wedge (y_1 \leftrightarrow (l_1 \circ_1 r_1)) \dots =: F_2$$

Beweis (Forts.):

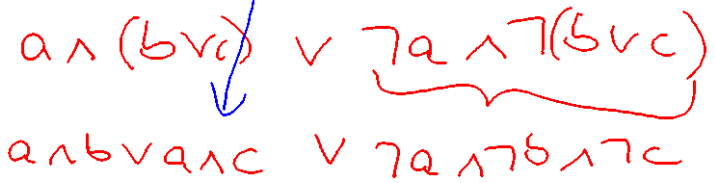
F_1 erf. $\Longrightarrow$ F_2 erf.: y_i bekommt Wert seines Teilbaums.
F_2 erf. $\Longrightarrow$ F_1 erf.: klar

Beweis (Forts.):

F_1 erf. $\Longrightarrow$ F_2 erf.: y_i bekommt Wert seines Teilbaums.
F_2 erf. $\Longrightarrow$ F_1 erf.: klar

4. Transformiere jede Äquivalenz in 3KNF:

$$(a \leftrightarrow (b \vee c)) \quad \mapsto \quad (a \vee \neg b) \wedge (a \vee \neg c) \wedge (\neg a \vee b \vee c)$$
$$(a \leftrightarrow (b \wedge c)) \quad \mapsto \quad (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c)$$



Beweis (Forts.):

F_1 erf. $\Longrightarrow$ F_2 erf.: y_i bekommt Wert seines Teilbaums.
F_2 erf. $\Longrightarrow$ F_1 erf.: klar

4. Transformiere jede Äquivalenz in 3KNF:

$$(a \leftrightarrow (b \vee c)) \quad \mapsto \quad (a \vee \neg b) \wedge (a \vee \neg c) \wedge (\neg a \vee b \vee c)$$
$$(a \leftrightarrow (b \wedge c)) \quad \mapsto \quad (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c)$$

Ergebnis ist F'.

Jeder Schritt ist in polynomieller Zeit in $|F|$ berechenbar.
Bei Transformation in NNF nimmt mit jedem Schritt

Summe der $|G|$ für alle Teilformeln $\neg G$ von F

ab.

Beweis (Forts.):

F_1 erf. $\Longrightarrow$ F_2 erf.: y_i bekommt Wert seines Teilbaums.
F_2 erf. $\Longrightarrow$ F_1 erf.: klar

4. Transformiere jede Äquivalenz in 3KNF:

$$(a \leftrightarrow (b \vee c)) \quad \mapsto \quad (a \vee \neg b) \wedge (a \vee \neg c) \wedge (\neg a \vee b \vee c)$$
$$(a \leftrightarrow (b \wedge c)) \quad \mapsto \quad (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c)$$

Ergebnis ist F'.

Jeder Schritt ist in polynomieller Zeit in $|F|$ berechenbar.
Bei Transformation in NNF nimmt mit jedem Schritt

Summe der $|G|$ für alle Teilformeln $\neg G$ von F

ab. Daher erreicht man die NNF in $\leq |F|$ Schritten. $\square$

Da jede Formel in 3KNF auch in KNF ist:

Korollar 5.26
KNF-SAT ist NP-vollständig.

Da jede Formel in 3KNF auch in KNF ist:

Korollar 5.26
KNF-SAT ist NP-vollständig.

Kann man wie folgt die NP-Vollständigkeit von KNF-SAT zeigen?

> Man zeigt SAT $\leq_p$ KNF-SAT
> indem man jede Formel in KNF transformiert.

Da jede Formel in 3KNF auch in KNF ist:

Korollar 5.26
KNF-SAT ist NP-vollständig.

Kann man wie folgt die NP-Vollständigkeit von KNF-SAT zeigen?

> Man zeigt SAT $\leq_p$ KNF-SAT
> indem man jede Formel in KNF transformiert.

Satz 5.27
2KNF-SAT $\in P$

Ohne Beweis

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M
und eine Zahl $k \leq n$.

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \ldots, i_k \in \{1, \ldots, n\}$ mit $M = T_{i_1} \cup \cdots \cup T_{i_k}$?

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \ldots, i_k \in \{1, \ldots, n\}$ mit $M = T_{i_1} \cup \cdots \cup T_{i_k}$?

Beispiel 5.28

$$\begin{array}{lllllll} T_1 & = & \{1,2\} & & T_2 & = & \{1,3\} \\ T_3 & = & \{3,4\} & & T_4 & = & \{3,5\} \\ M & = & \{1,2,3,4,5\} & & k & = & 3 \end{array}$$

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \ldots, i_k \in \{1, \ldots, n\}$ mit $M = T_{i_1} \cup \cdots \cup T_{i_k}$?

Beispiel 5.28

$$\begin{array}{lllllll} T_1 & = & \{1,2\} & & T_2 & = & \{1,3\} \\ T_3 & = & \{3,4\} & & T_4 & = & \{3,5\} \\ M & = & \{1,2,3,4,5\} & & k & = & 3 \end{array}$$

Lösung:

Anwendung: Zulieferer

M Menge der Teile, die eine Firma einkaufen muss

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \ldots, i_k \in \{1, \ldots, n\}$ mit $M = T_{i_1} \cup \cdots \cup T_{i_k}$?

Beispiel 5.28

$$\begin{array}{lllllll} T_1 & = & \{1,2\} & & T_2 & = & \{1,3\} \\ T_3 & = & \{3,4\} & & T_4 & = & \{3,5\} \\ M & = & \{1,2,3,4,5\} & & k & = & 3 \end{array}$$

Lösung:

Anwendung: Zulieferer

M Menge der Teile, die eine Firma einkaufen muss

T_i Menge der Teile, die Zulieferer i anbietet

Kann die Firma ihre Bedürfnisse mit k Zulieferern abdecken?

Fakt 5.29
MÜ $\in$ *NP.*

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \ldots, i_k \in \{1, \ldots, n\}$ mit $M = T_{i_1} \cup \cdots \cup T_{i_k}$?

Beispiel 5.28

$$\begin{array}{lllllll} T_1 & = & \{1,2\} & & T_2 & = & \{1,3\} \\ T_3 & = & \{3,4\} & & T_4 & = & \{3,5\} \\ M & = & \{1,2,3,4,5\} & & k & = & 3 \end{array}$$

Lösung:

Anwendung: Zulieferer

M Menge der Teile, die eine Firma einkaufen muss

T_i Menge der Teile, die Zulieferer i anbietet

Kann die Firma ihre Bedürfnisse mit k Zulieferern abdecken?

Fakt 5.29

MÜ $\in$ NP.

Satz 5.30

MÜ ist NP-vollständig.

Beweis:

Wir zeigen KNF-SAT $\leq_p$ MÜ.

Satz 5.30

MÜ ist NP-vollständig.

Beweis:

Wir zeigen KNF-SAT $\leq_p$ MÜ.

Sei $F = K_1 \wedge \cdots \wedge K_m$ in KNF, mit Variablen $x_1, \ldots, x_k$.

Satz 5.30

MÜ ist NP-vollständig.

Beweis:

Wir zeigen KNF-SAT $\leq_p$ MÜ.

Sei $F = K_1 \wedge \cdots \wedge K_m$ in KNF, mit Variablen $x_1, \ldots, x_k$.

$$M := \{1, \ldots, m, m+1, \ldots, m+k\}$$

Satz 5.30
MÜ ist NP-vollständig.

Beweis:
Wir zeigen KNF-SAT $\leq_p$ MÜ.
Sei $F = K_1 \wedge \cdots \wedge K_m$ in KNF, mit Variablen $x_1, \ldots, x_k$.

$$\begin{aligned} M &:= \{1, \ldots, m, m+1, \ldots, m+k\} \\ T_i &:= \{j \mid x_i \text{ kommt in } K_j \text{ positiv vor}\} \cup \{m+i\} \end{aligned}$$

Satz 5.30
MÜ ist NP-vollständig.

Beweis:
Wir zeigen KNF-SAT $\leq_p$ MÜ.
Sei $F = K_1 \wedge \cdots \wedge K_m$ in KNF, mit Variablen $x_1, \ldots, x_k$.

$$\begin{aligned} M &:= \{1, \ldots, m, m+1, \ldots, m+k\} \\ T_i &:= \{j \mid x_i \text{ kommt in } K_j \text{ positiv vor}\} \cup \{m+i\} \end{aligned}$$

$$T'_i := \{j \mid x_i \text{ kommt in } K_j \text{ negativ vor}\} \cup \{m+i\}$$

Satz 5.30
MÜ ist NP-vollständig.

Beweis:
Wir zeigen KNF-SAT $\leq_p$ MÜ.
Sei $F = K_1 \wedge \cdots \wedge K_m$ in KNF, mit Variablen $x_1, \ldots, x_k$.

$$\begin{aligned} M &:= \{1, \ldots, m, m+1, \ldots, m+k\} \\ T_i &:= \{j \mid x_i \text{ kommt in } K_j \text{ positiv vor}\} \cup \{m+i\} \\ T'_i &:= \{j \mid x_i \text{ kommt in } K_j \text{ negativ vor}\} \cup \{m+i\} \end{aligned}$$

F ist erfülbar gdw
M wird durch k der Teilmengen $T_1, \ldots, T_k, T'_1, \ldots, T'_k$ überdeckt

□

Satz 5.30
MÜ ist NP-vollständig.

Beweis:
Wir zeigen KNF-SAT $\leq_p$ MÜ.
Sei $F = K_1 \wedge \cdots \wedge K_m$ in KNF, mit Variablen $x_1, \ldots, x_k$.

$$\begin{aligned} M &:= \{1, \ldots, m, m+1, \ldots, m+k\} \\ T_i &:= \{j \mid x_i \text{ kommt in } K_j \text{ positiv vor}\} \cup \{m+i\} \\ T'_i &:= \{j \mid x_i \text{ kommt in } K_j \text{ negativ vor}\} \cup \{m+i\} \end{aligned}$$

F ist erfülbar gdw
M wird durch k der Teilmengen $T_1, \ldots, T_k, T'_1, \ldots, T'_k$ überdeckt

□

Das **Minimierungsproblem**

Gegeben: Teilmengen $T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M

Problem: Finde das kleinste k, so dass M von k der Teilmengen überdeckt wird.

Das **Minimierungsproblem**

Gegeben: Teilmengen $T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M

Problem: Finde das kleinste k, so dass M von k der Teilmengen überdeckt wird.

Das **Minimierungsproblem**

Gegeben: Teilmengen $T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M

Problem: Finde das kleinste k, so dass M von k der Teilmengen überdeckt wird.

kann auf das Entscheidungsproblem reduziert werden:

Das **Minimierungsproblem**

Gegeben: Teilmengen $T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M

Problem: Finde das kleinste k, so dass M von k der Teilmengen überdeckt wird.

kann auf das Entscheidungsproblem reduziert werden:

Finde kleinstes k durch binäre Suche im Intervall $[1, n]$ mit $O(\log n)$ Aufrufen von MÜ.

Das **Minimierungsproblem**

Gegeben: Teilmengen $T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M

Problem: Finde das kleinste k, so dass M von k der Teilmengen überdeckt wird.

kann auf das Entscheidungsproblem reduziert werden:

> Finde kleinstes k durch binäre Suche im Intervall $[1, n]$ mit $O(\log n)$ Aufrufen von MÜ.

Kann man MÜ in Zeit $O(f(n))$ entscheiden,
dann kann man das kleinste k in Zeit $O(f(n) \cdot \log n)$ berechnen.

Das **Minimierungsproblem**

Gegeben: Teilmengen $T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M

Problem: Finde das kleinste k, so dass M von k der Teilmengen überdeckt wird.

kann auf das Entscheidungsproblem reduziert werden:

> Finde kleinstes k durch binäre Suche im Intervall $[1, n]$ mit $O(\log n)$ Aufrufen von MÜ.

Kann man MÜ in Zeit $O(f(n))$ entscheiden,
dann kann man das kleinste k in Zeit $O(f(n) \cdot \log n)$ berechnen.

$$f(n) \text{ polynomiell} \implies f(n) \cdot \log n \text{ polynomiell}$$
$$f(n) \text{ exponentiell} \implies f(n) \cdot \log n \text{ exponentiell}$$

Die Berechnung einer **Lösung**

Gegeben: Teilmengen $\vec{T} := T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M, und eine Zahl $k \leq n$.

Problem: Finde eine Überdeckung von M durch k der Teilmengen.

Die Berechnung einer **Lösung**

Gegeben: Teilmengen $\vec{T} := T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M, und eine Zahl $k \leq n$.

Problem: Finde eine Überdeckung von M durch k der Teilmengen.

kann auf das Entscheidungsproblem reduziert werden:

if $(\vec{T}, M, k) \notin$ MÜ **then** output("nicht lösbar")

Die Berechnung einer **Lösung**

Gegeben: Teilmengen $\vec{T} := T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M, und eine Zahl $k \leq n$.

Problem: Finde eine Überdeckung von M durch k der Teilmengen.

kann auf das Entscheidungsproblem reduziert werden:

if $(\vec{T}, M, k) \notin$ MÜ **then** output("nicht lösbar")
else
  $\ddot{U} := \emptyset$
  for $i := 1$ **to** n **do**

Die Berechnung einer **Lösung**

Gegeben: Teilmengen $\vec{T} := T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M, und eine Zahl $k \leq n$.

Problem: Finde eine Überdeckung von M durch k der Teilmengen.

kann auf das Entscheidungsproblem reduziert werden:

if $(\vec{T}, M, k) \notin$ MÜ **then** output("nicht lösbar")
else
  $\ddot{U} := \emptyset$
  for $i := 1$ **to** n **do**
    if $(\vec{T} - T_i, M, k) \in$ MÜ

Die Berechnung einer **Lösung**

Gegeben: Teilmengen $\vec{T} := T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M, und eine Zahl $k \leq n$.

Problem: Finde eine Überdeckung von M durch k der Teilmengen.

kann auf das Entscheidungsproblem reduziert werden:

if $(\vec{T}, M, k) \notin$ MÜ **then** output("nicht lösbar")
else
  $\ddot{U} := \emptyset$
  for $i := 1$ **to** n **do**
    if $(\vec{T} - T_i, M, k) \in$ MÜ
    then $\vec{T} := \vec{T} - T_i$

Die Berechnung einer **Lösung**

Gegeben: Teilmengen $\vec{T} := T_1, \ldots, T_n \subseteq M$ einer endlichen Menge M, und eine Zahl $k \leq n$.

Problem: Finde eine Überdeckung von M durch k der Teilmengen.

kann auf das Entscheidungsproblem reduziert werden:

if $(\vec{T}, M, k) \notin$ MÜ **then** output("nicht lösbar")
else
  $\ddot{U} := \emptyset$
  for $i := 1$ **to** n **do**
    if $(\vec{T} - T_i, M, k) \in$ MÜ
    then $\vec{T} := \vec{T} - T_i$
    else $\ddot{U} := \ddot{U} \cup \{T_i\}$

CLIQUE

Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.

Problem: Besitzt G eine „Clique“ der Größe mindestens k?

CLIQUE

Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.

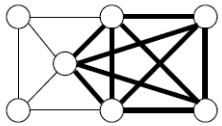
Problem: Besitzt G eine „Clique“ der Größe mindestens k?
Dh eine Teilmenge $V' \subseteq V$ mit $|V'| \geq k$
und alle $u, v \in V'$ mit $u \neq v$ sind benachbart.

CLIQUE

Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.

Problem: Besitzt G eine „Clique“ der Größe mindestens k?
Dh eine Teilmenge $V' \subseteq V$ mit $|V'| \geq k$
und alle $u, v \in V'$ mit $u \neq v$ sind benachbart.

Beispiel mit 5-Clique:

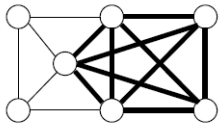


CLIQUE

Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.

Problem: Besitzt G eine „Clique“ der Größe mindestens k?
Dh eine Teilmenge $V' \subseteq V$ mit $|V'| \geq k$
und alle $u, v \in V'$ mit $u \neq v$ sind benachbart.

Beispiel mit 5-Clique:



Anwendung von Clique-Berechnung:

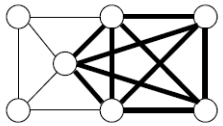
Knoten	=	Prozess
Kante	=	Zwei Prozesse sind parallel ausführbar

CLIQUE

Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.

Problem: Besitzt G eine „Clique“ der Größe mindestens k?
Dh eine Teilmenge $V' \subseteq V$ mit $|V'| \geq k$
und alle $u, v \in V'$ mit $u \neq v$ sind benachbart.

Beispiel mit 5-Clique:



Anwendung von Clique-Berechnung:

Knoten	=	Prozess
Kante	=	Zwei Prozesse sind parallel ausführbar

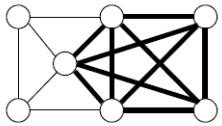
$\Longrightarrow$ Clique ist Gruppe von parallelisierbaren Prozessen

CLIQUE

Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.

Problem: Besitzt G eine „Clique“ der Größe mindestens k?
Dh eine Teilmenge $V' \subseteq V$ mit $|V'| \geq k$
und alle $u, v \in V'$ mit $u \neq v$ sind benachbart.

Beispiel mit 5-Clique:



Anwendung von Clique-Berechnung:

Knoten	=	Prozess
Kante	=	Zwei Prozesse sind parallel ausführbar

$\Longrightarrow$ Clique ist Gruppe von parallelisierbaren Prozessen

Fakt 5.31
CLIQUE $\in$ *NP*

Satz 5.32
CLIQUE ist NP-vollständig.

Satz 5.32
CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT $\leq_p$ CLIQUE:

Satz 5.32
CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT $\leq_p$ CLIQUE:
Eine Formel F in 3KNF-SAT (oE: *genau* 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \ldots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

Satz 5.32
CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT $\leq_p$ CLIQUE:
Eine Formel F in 3KNF-SAT (oE: *genau* 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \ldots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

wird auf einen Graphen abgebildet:

$$V = \{(1,1),(1,2),(1,3),\ldots,(k,1),(k,2),(k,3)\}$$

Satz 5.32
CLIQUE ist NP-vollständig.

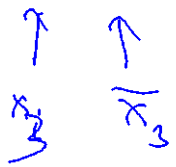
Beweis: 3KNF-SAT $\leq_p$ CLIQUE:
Eine Formel F in 3KNF-SAT (oE: *genau* 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \ldots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

wird auf einen Graphen abgebildet:

$$V = \{(1,1),(1,2),(1,3),\ldots,(k,1),(k,2),(k,3)\}$$

$$E = \{\{(i,j),(p,q)\} \mid i \neq p \text{ und } z_{ij} \neq \neg z_{pq}\}$$



Satz 5.32
CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT $\leq_p$ CLIQUE:
Eine Formel F in 3KNF-SAT (oE: *genau* 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \ldots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

wird auf einen Graphen abgebildet:

$$V = \{(1,1),(1,2),(1,3),\ldots,(k,1),(k,2),(k,3)\}$$

$$E = \{\{(i,j),(p,q)\} \mid i \neq p \text{ und } z_{ij} \neq \neg z_{pq}\}$$

F ist erfüllbar durch eine Belegung σ $\quad \Longleftrightarrow$

Satz 5.32
CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT $\leq_p$ CLIQUE:
Eine Formel F in 3KNF-SAT (oE: *genau* 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \ldots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

wird auf einen Graphen abgebildet:

$$\begin{aligned} V &= \{(1,1),(1,2),(1,3),\ldots,(k,1),(k,2),(k,3)\} \\ E &= \{\{(i,j),(p,q)\} \mid i \neq p \text{ und } z_{ij} \neq \neg z_{pq}\} \end{aligned}$$

F ist erfüllbar durch eine Belegung σ $\iff$
Es gibt in jeder Klausel i ein Literal z_{ij_i} mit $\sigma(z_{ij_i}) = 1$
$\iff$

Satz 5.32
CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT $\leq_p$ CLIQUE:
Eine Formel F in 3KNF-SAT (oE: *genau* 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \ldots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

wird auf einen Graphen abgebildet:

$$\begin{aligned} V &= \{(1,1),(1,2),(1,3),\ldots,(k,1),(k,2),(k,3)\} \\ E &= \{\{(i,j),(p,q)\} \mid i \neq p \text{ und } z_{ij} \neq \neg z_{pq}\} \end{aligned}$$

F ist erfüllbar durch eine Belegung σ $\iff$
Es gibt in jeder Klausel i ein Literal z_{ij_i} mit $\sigma(z_{ij_i}) = 1$
$\iff$
Die Literale $z_{1j_1}, \ldots, z_{kj_k}$ sind paarweise nicht komplementär
$\iff$

Satz 5.32
CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT $\leq_p$ CLIQUE:
Eine Formel F in 3KNF-SAT (oE: *genau* 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \ldots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

wird auf einen Graphen abgebildet:

$$\begin{aligned} V &= \{(1,1),(1,2),(1,3),\ldots,(k,1),(k,2),(k,3)\} \\ E &= \{\{(i,j),(p,q)\} \mid i \neq p \text{ und } z_{ij} \neq \neg z_{pq}\} \end{aligned}$$

F ist erfüllbar durch eine Belegung σ $\iff$
Es gibt in jeder Klausel i ein Literal z_{ij_i} mit $\sigma(z_{ij_i}) = 1$
$\iff$
Die Literale $z_{1j_1}, \ldots, z_{kj_k}$ sind paarweise nicht komplementär
$\iff$
Die Knoten $(1,j_1), \ldots, (k,j_k)$ sind paarweise benachbart

Satz 5.32
CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT $\leq_p$ CLIQUE:
Eine Formel F in 3KNF-SAT (oE: *genau* 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \ldots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

wird auf einen Graphen abgebildet:

$$\begin{aligned} V &= \{(1,1),(1,2),(1,3),\ldots,(k,1),(k,2),(k,3)\} \\ E &= \{\{(i,j),(p,q)\} \mid i \neq p \text{ und } z_{ij} \neq \neg z_{pq}\} \end{aligned}$$

F ist erfüllbar durch eine Belegung σ $\iff$
Es gibt in jeder Klausel i ein Literal z_{ij_i} mit $\sigma(z_{ij_i}) = 1$
$\iff$
Die Literale $z_{1j_1}, \ldots, z_{kj_k}$ sind paarweise nicht komplementär
$\iff$
Die Knoten $(1,j_1), \ldots, (k,j_k)$ sind paarweise benachbart
$\iff$ G hat eine Clique der Größe k. $\square$

RUCKSACK

Gegeben: Zahlen $a_1, \ldots a_n \in \mathbb{N}$ und $b \in \mathbb{N}$.

RUCKSACK

Gegeben: Zahlen $a_1, \ldots a_n \in \mathbb{N}$ und $b \in \mathbb{N}$.

Problem: Gibt es $R \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in R} a_i = b$?

RUCKSACK

Gegeben: Zahlen $a_1, \ldots a_n \in \mathbb{N}$ und $b \in \mathbb{N}$.

Problem: Gibt es $R \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in R} a_i = b$?

Satz 5.33

RUCKSACK ist NP-vollständig.

RUCKSACK

Gegeben: Zahlen $a_1, \ldots a_n \in \mathbb{N}$ und $b \in \mathbb{N}$.

Problem: Gibt es $R \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in R} a_i = b$?

Satz 5.33

RUCKSACK ist NP-vollständig.

Beweis:

3KNF-SAT $\leq_p$ RUCKSACK :

RUCKSACK

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$ und $b \in \mathbb{N}$.

Problem: Gibt es $R \subseteq \{1, \dots, n\}$ mit $\sum_{i \in R} a_i = b$?

Satz 5.33

RUCKSACK ist NP-vollständig.

Beweis:

3KNF-SAT $\leq_p$ RUCKSACK :

Sei $F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \dots \wedge (z_{m1} \vee z_{m2} \vee z_{m3})$, wobei

$$z_{ij} \in \{x_1, \dots, x_n\} \cup \{\neg x_1, \neg x_2, \dots, \neg x_n\}$$

D.h. $m =$ Anzahl der Klauseln und n Anzahl der vorkommenden Variablen.

Wir geben jetzt Zahlen $\vec{a} = a_1, \dots, a_k$ und b an. b ist (etwa im Dezimalsystem)

$$b = \underbrace{44 \dots 44}_{m} \underbrace{11 \dots 11}_{n}$$

□

PARTITION

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$.

PARTITION

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$.

Problem: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

PARTITION

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$.

Problem: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

Satz 5.34

PARTITION ist NP-vollständig.

PARTITION

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$.

Problem: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

Satz 5.34

PARTITION ist NP-vollständig.

Beweis: RUCKSACK $\leq_p$ PARTITION

PARTITION

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$.

Problem: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

Satz 5.34

PARTITION ist NP-vollständig.

Beweis: RUCKSACK $\leq_p$ PARTITION

Beispiel: $\vec{a} = 12, 7, 4, 9, 7, 3, 15$ und $b = 38$.

PARTITION

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$.

Problem: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

Satz 5.34

PARTITION ist NP-vollständig.

Beweis: RUCKSACK $\leq_p$ PARTITION

Beispiel: $\vec{a} = 12, 7, 4, 9, 7, 3, 15$ und $b = 38$.

Lösung: Die Zahlen $7, 9, 7, 15$

PARTITION

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$.

Problem: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

Satz 5.34

PARTITION ist NP-vollständig.

Beweis: RUCKSACK $\leq_p$ PARTITION

Beispiel: $\vec{a} = 12, 7, 4, 9, 7, 3, 15$ und $b = 38$.

Lösung: Die Zahlen $7, 9, 7, 15$, dh $R = \{2, 4, 5, 7\}$

PARTITION

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$.

Problem: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

Satz 5.34

PARTITION ist NP-vollständig.

Beweis: RUCKSACK $\leq_p$ PARTITION

Beispiel: $\vec{a} = 12, 7, 4, 9, 7, 3, 15$ und $b = 38$.

Lösung: Die Zahlen $7, 9, 7, 15$, dh $R = \{2, 4, 5, 7\}$

RUCKSACK $\rightarrow$ PARTITION:

$M := \sum_{i=1}^{n} a_i = 57$,

PARTITION

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$.

Problem: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

Satz 5.34

PARTITION ist NP-vollständig.

Beweis: RUCKSACK $\leq_p$ PARTITION

Beispiel: $\vec{a} = 12, 7, 4, 9, 7, 3, 15$ und $b = 38$.

Lösung: Die Zahlen $7, 9, 7, 15$, dh $R = \{2, 4, 5, 7\}$

RUCKSACK $\rightarrow$ PARTITION:

$M := \sum_{i=1}^{n} a_i = 57, \quad M - b + 1 = 20, \quad b + 1 = 39$

PARTITION

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$.

Problem: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

Satz 5.34

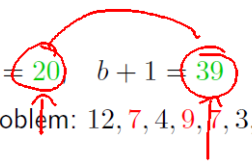
PARTITION ist NP-vollständig.

Beweis: RUCKSACK $\leq_p$ PARTITION

Beispiel: $\vec{a} = 12, 7, 4, 9, 7, 3, 15$ und $b = 38$.

Lösung: Die Zahlen $7, 9, 7, 15$, dh $R = \{2, 4, 5, 7\}$

RUCKSACK $\rightarrow$ PARTITION:



$M := \sum_{i=1}^{n} a_i = 57, \quad M - b + 1 = 20, \quad b + 1 = 39$

Das resultierende PARTITONS-Problem: $12, 7, 4, 9, 7, 3, 15, \ 20, 39$

PARTITION

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$.

Problem: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

Satz 5.34

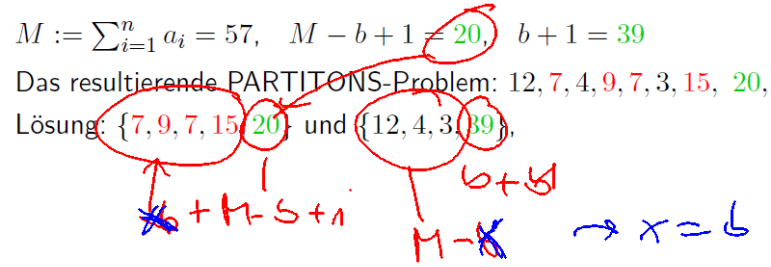
PARTITION ist NP-vollständig.

Beweis: RUCKSACK $\leq_p$ PARTITION

Beispiel: $\vec{a} = 12, 7, 4, 9, 7, 3, 15$ und $b = 38$.

Lösung: Die Zahlen $7, 9, 7, 15$, dh $R = \{2, 4, 5, 7\}$

RUCKSACK $\rightarrow$ PARTITION:



$M := \sum_{i=1}^{n} a_i = 57, \quad M - b + 1 = 20, \quad b + 1 = 39$

Das resultierende PARTITONS-Problem: $12, 7, 4, 9, 7, 3, 15, \ 20, 39$

Lösung: $\{7, 9, 7, 15, 20\}$ und $\{12, 4, 3, 39\}$.

PARTITION

Gegeben: Zahlen $a_1, \dots a_n \in \mathbb{N}$.

Problem: Gibt es $I \subseteq \{1, \dots, n\}$ mit $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

Satz 5.34
PARTITION ist NP-vollständig.

Beweis: RUCKSACK $\leq_p$ PARTITION

Beispiel: $\vec{a} = 12, 7, 4, 9, 7, 3, 15$ und $b = 38$.

Lösung: Die Zahlen $7, 9, 7, 15$, dh $R = \{2, 4, 5, 7\}$

RUCKSACK $\rightarrow$ PARTITION:

$M := \sum_{i=1}^{n} a_i = 57, \quad M - b + 1 = 20, \quad b + 1 = 39$

Das resultierende PARTITONS-Problem: $12, 7, 4, 9, 7, 3, 15, \ 20, 39$

Lösung: $\{7, 9, 7, 15, 20\}$ und $\{12, 4, 3, 39\}$, dh $I = \{2, 4, 5, 7, 8\}$.

Reduktion im Allgemeinen:

$$a_1, a_2, \dots, a_k, b \ \mapsto \ a_1, a_2, \dots, a_k, M - b + 1, b + 1$$ □

BIN PACKING

Gegeben: Eine „Behältergröße“ $b \in \mathbb{N}$, die Anzahl der Behälter $k \in \mathbb{N}$ und „Objekte“ $a_1, a_2, \dots a_n$.

BIN PACKING

Gegeben: Eine „Behältergröße“ $b \in \mathbb{N}$, die Anzahl der Behälter $k \in \mathbb{N}$ und „Objekte“ $a_1, a_2, \dots a_n$.

Problem: Können die Objekte so auf die k Behälter verteilt werden, dass kein Behälter überläuft?

BIN PACKING

Gegeben: Eine „Behältergröße“ $b \in \mathbb{N}$, die Anzahl der Behälter $k \in \mathbb{N}$ und „Objekte“ $a_1, a_2, \dots a_n$.

Problem: Können die Objekte so auf die k Behälter verteilt werden, dass kein Behälter überläuft?

Satz 5.35
BIN PACKING ist NP-vollständig.

BIN PACKING

Gegeben: Eine „Behältergröße“ $b \in \mathbb{N}$, die Anzahl der Behälter $k \in \mathbb{N}$ und „Objekte“ $a_1, a_2, \ldots a_n$.

Problem: Können die Objekte so auf die k Behälter verteilt werden, dass kein Behälter überläuft?

Satz 5.35

BIN PACKING ist NP-vollständig.

Beweis: PARTITION $\leq_p$ BIN PACKING

BIN PACKING

Gegeben: Eine „Behältergröße“ $b \in \mathbb{N}$, die Anzahl der Behälter $k \in \mathbb{N}$ und „Objekte“ $a_1, a_2, \ldots a_n$.

Problem: Können die Objekte so auf die k Behälter verteilt werden, dass kein Behälter überläuft?

Satz 5.35

BIN PACKING ist NP-vollständig.

Beweis: PARTITION $\leq_p$ BIN PACKING

$$(a_1, \ldots, a_n) \mapsto (b, k, 2a_1, \ldots, 2a_n)$$

wobei $b := \sum_{i=1}^{k} a_i$ und $k := 2$. □

HAMILTON

Gegeben: Ungerichteter Graph G

Problem: Enthält G einen Hamilton-Kreis, dh einen geschlossenen Pfad, der jeden Knoten genau einmal enthält?

HAMILTON

Gegeben: Ungerichteter Graph G

Problem: Enthält G einen Hamilton-Kreis, dh einen geschlossenen Pfad, der jeden Knoten genau einmal enthält?

Satz 5.36

HAMILTON ist NP-vollständig.

TRAVELLING SALESMAN (TSP)

Gegeben: Eine $n \times n$ Matrix $M_{ij} \in \mathbb{N}$ von „Entfernungen“
und eine Zahl $k \in \mathbb{N}$