

Einführung in die Theoretische Informatik

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2017/18

Die Laufzeit einer NTM M bei Eingabe x ist die maximale Anzahl an Rechenschritten, die $M(x)$ ausführt.

Definition

- Die **Laufzeit** einer NTM M bei Eingabe x ist definiert als

$$time_M(x) = \sup\{t \geq 0 \mid \exists K : K_x \vdash^t K\},$$

wobei $\sup \mathbb{N} = \infty$ ist.

- Sei $t : \mathbb{N} \rightarrow \mathbb{N}$ eine monoton wachsende Funktion.
- Dann ist M **$t(n)$ -zeitbeschränkt**, falls für alle Eingaben x gilt:

$$time_M(x) \leq t(|x|).$$

Die Zeitschranke $t(n)$ beschränkt also die Laufzeit bei allen Eingaben der Länge n (**worst-case** Komplexität).

Zeitkomplexitätsklassen

Wir fassen alle Sprachen und Funktionen, die in einer vorgegebenen Zeitschranke $t(n)$ entscheidbar bzw. berechenbar sind, in folgenden **Komplexitätsklassen** zusammen.

Definition

- Die in deterministischer Zeit $t(n)$ entscheidbaren Sprachen bilden die Sprachklasse

$$\text{DTIME}(t(n)) = \{L(M) \mid M \text{ ist eine } t(n)\text{-zeitbeschränkte DTM}\}.$$

- Die in nichtdeterministischer Zeit $t(n)$ entscheidbaren Sprachen bilden die Sprachklasse

$$\text{NTIME}(t(n)) = \{L(M) \mid M \text{ ist eine } t(n)\text{-zeitbeschränkte NTM}\}.$$

- Die in deterministischer Zeit $t(n)$ berechenbaren Funktionen bilden die Funktionenklasse

$$\text{FTIME}(t(n)) = \left\{ f \mid \begin{array}{l} \text{es gibt eine } t(n)\text{-zeitbeschränkte} \\ \text{DTM } M, \text{ die } f \text{ berechnet} \end{array} \right\}.$$

Die wichtigsten Zeitkomplexitätsklassen

- Die wichtigsten deterministischen Zeitkomplexitätsklassen sind

$$\text{LINTIME} = \bigcup_{c \geq 1} \text{DTIME}(cn + c) \quad \text{„Linearzeit“}$$

$$\text{P} = \bigcup_{c \geq 1} \text{DTIME}(n^c + c) \quad \text{„Polynomialzeit“}$$

$$\text{E} = \bigcup_{c \geq 1} \text{DTIME}(2^{cn+c}) \quad \text{„Lineare Exponentialzeit“}$$

$$\text{EXP} = \bigcup_{c \geq 1} \text{DTIME}(2^{n^c+c}) \quad \text{„Exponentialzeit“}$$

- Die nichtdeterministischen Klassen **NLINTIME**, **NP**, **NE**, **NEXP** und die Funktionenklassen **FLINTIME**, **FP**, **FE**, **FEXP** sind analog definiert.
- Für eine Klasse \mathcal{F} von Funktionen sei $\text{DTIME}(\mathcal{F}) = \bigcup_{t \in \mathcal{F}} \text{DTIME}(t(n))$ (die Klassen **NTIME**(\mathcal{F}) und **FTIME**(\mathcal{F}) sind analog definiert).

Asymptotische Laufzeit und Landau-Notation

Definition

Seien f und g Funktionen von \mathbb{N} nach $\mathbb{R}^+ \cup \{0\} = [0, \infty)$.

- Wir schreiben $f(n) = \mathcal{O}(g(n))$, falls es Zahlen n_0 und c gibt mit

$$\forall n \geq n_0 : f(n) \leq c \cdot g(n).$$

Bedeutung: „ f wächst **nicht wesentlich schneller** als g .“

- Formal bezeichnet der Term $\mathcal{O}(g(n))$ die Klasse aller Funktionen f , die obige Bedingung erfüllen, d.h.

$$\mathcal{O}(g(n)) = \{f: \mathbb{N} \rightarrow [0, \infty) \mid \exists n_0, c \in \mathbb{N} \forall n \geq n_0 : f(n) \leq c \cdot g(n)\}.$$

- Die Gleichung $f(n) = \mathcal{O}(g(n))$ drückt also in Wahrheit eine **Element-Beziehung** $f \in \mathcal{O}(g(n))$ aus.
- \mathcal{O} -Terme können auch auf der linken Seite vorkommen. In diesem Fall wird eine **Inklusionsbeziehung** ausgedrückt.
- So steht $n^2 + \mathcal{O}(n) = \mathcal{O}(n^2)$ für die Aussage

$$\{n^2 + f \mid f \in \mathcal{O}(n)\} \subseteq \mathcal{O}(n^2).$$

Beispiel

- $7 \log(n) + n^3 = \mathcal{O}(n^3)$ ist **richtig**.
- $7 \log(n)n^3 = \mathcal{O}(n^3)$ ist **falsch**.
- $2^{n+\mathcal{O}(1)} = \mathcal{O}(2^n)$ ist **richtig**.
- $2^{\mathcal{O}(n)} = \mathcal{O}(2^n)$ ist **falsch** (siehe Übungen).

Mit der \mathcal{O} -Notation lassen sich die wichtigsten deterministischen Zeitkomplexitätsklassen wie folgt charakterisieren:

LINTIME = DTIME($\mathcal{O}(n)$) „Linearzeit“

P = DTIME($n^{\mathcal{O}(1)}$) „Polynomialzeit“

E = DTIME($2^{\mathcal{O}(n)}$) „Lineare Exponentialzeit“

EXP = DTIME($2^{n^{\mathcal{O}(1)}}$) „Exponentialzeit“

Das P-NP-Problem

- Wie wir gesehen haben, sind NTMs nicht mächtiger als DTMs, d.h. jede NTM kann von einer DTM simuliert werden.
- Die Frage, wieviel Zeit eine DTM zur Simulation einer NTM benötigt, ist eines der wichtigsten offenen Probleme der Informatik.
- Wegen $\text{NTIME}(t) \subseteq \text{DTIME}(2^{\mathcal{O}(t)})$ erhöht sich die Laufzeit im schlimmsten Fall exponentiell.
- Insbesondere die Klasse NP enthält viele für die Praxis überaus wichtige Probleme, für die kein Polynomialzeitalgorithmus bekannt ist.
- Für viele dieser Probleme A konnte folgende Implikation gezeigt werden:
$$A \in P \Rightarrow P = NP$$
- Da jedoch nur Probleme in P als effizient lösbar angesehen werden, hat das **P-NP-Problem**, also die Frage, ob $NP = P$ ist, eine immense praktische Bedeutung.

Die Polynomialzeitreduktion

Definition

- Eine Sprache $A \subseteq \Sigma^*$ ist auf $B \subseteq \Gamma^*$ **in Polynomialzeit reduzierbar** ($A \leq^P B$), falls eine Funktion $f : \Sigma^* \rightarrow \Gamma^*$ in FP existiert mit

$$\forall x \in \Sigma^* : x \in A \Leftrightarrow f(x) \in B.$$

- Eine Sprache A heißt **\leq^P -hart** für eine Sprachklasse \mathcal{C} (kurz: **\mathcal{C} -hart** oder **\mathcal{C} -schwer**), falls gilt:

$$\forall L \in \mathcal{C} : L \leq^P A.$$

- Eine \mathcal{C} -harte Sprache A , die zu \mathcal{C} gehört, heißt **\mathcal{C} -vollständig** (bzgl. \leq^P).
- **NPC** bezeichnet die Klasse aller NP-vollständigen Sprachen.

Lemma

- Aus $A \leq^P B$ folgt $A \leq B$.
- Die Reduktionsrelation \leq^P ist reflexiv und transitiv (s. Übungen).

Die Polynomialzeitreduktion

Satz

Die Klassen P und NP sind unter \leq^P abgeschlossen.

Beweis

- Sei $B \in P$ und gelte $A \leq^P B$ mittels einer Funktion $f \in FP$.
- Seien M und T DTMs mit $L(M) = B$ und $T(x) = f(x)$.
- Weiter seien p und q polynomielle Zeitschranken für M und T .
- Betrachte die DTM M' , die bei Eingabe x zuerst T simuliert, um $f(x)$ zu berechnen, und danach M bei Eingabe $f(x)$ simuliert. Dann gilt

$$x \in A \Leftrightarrow f(x) \in B \Leftrightarrow f(x) \in L(M) \Leftrightarrow x \in L(M').$$

- Also ist $L(M') = A$ und wegen

$$\text{time}_{M'}(x) \leq \text{time}_T(x) + \text{time}_M(f(x)) \leq q(|x|) + p(q(|x|))$$

ist M' polynomiell zeitbeschränkt und somit A in P.

- Der Abschluss von NP unter \leq^P folgt analog. □

Satz

- 1 $A \leq^P B$ und A ist NP-hart $\Rightarrow B$ ist NP-hart.
- 2 Falls ein NP-vollständiges Problem A in P enthalten ist, folgt $P = NP$.

Beweis

- 1 Da A NP-hart ist, ist jede NP-Sprache L auf A reduzierbar. Da zudem $A \leq^P B$ gilt und \leq^P transitiv ist, folgt $L \leq^P B$.
- 2 Sei A eine NP-vollständige Sprache in P . Dann ist jede NP-Sprache L auf A reduzierbar und da P unter \leq^P abgeschlossen ist, folgt $L \in P$. \square

Platzkomplexität von Turingmaschinen

- Als nächstes definieren wir den Platzverbrauch von NTMs.
- Intuitiv ist dies die Anzahl aller besuchten Bandfelder.
- Wollen wir auch sublinearen Platz sinnvoll definieren, so dürfen wir hierbei das erste Band offensichtlich nicht berücksichtigen.
- Um sicherzustellen, dass eine NTM M das erste Band nur zum Lesen der Eingabe und nicht auch zum Speichern von weiteren Informationen benutzt, verlangen wir, dass M
 - die Felder auf dem Eingabeband nicht verändert und
 - sich höchstens ein Feld von der Eingabe entfernt.

Definition

Eine NTM M heißt **offline-NTM** (oder NTM mit **Eingabeband**), falls für jede von M bei Eingabe x erreichbare Konfiguration

$$K = (q, u_1, a_1, v_1, \dots, u_k, a_k, v_k)$$

gilt, dass $u_1 a_1 v_1$ ein Teilwort von $\sqcup x \sqcup$ ist.

Definition

- Der **Platzverbrauch** einer offline-NTM M bei Eingabe x ist definiert als

$$space_M(x) = \sup \left\{ s \geq 1 \left| \begin{array}{l} \exists K = (q, u_1, a_1, v_1, \dots, u_k, a_k, v_k) \\ \text{mit } K_x \vdash^* K \text{ und } s = \sum_{i=2}^k |u_i a_i v_i| \end{array} \right. \right\}.$$

- Sei $s : \mathbb{N} \rightarrow \mathbb{N}$ eine monoton wachsende Funktion.
- M heißt **$s(n)$ -platzbeschränkt**, falls für alle Eingaben x gilt:

$$space_M(x) \leq s(|x|) \text{ und } time_M(x) < \infty.$$

Wir fassen alle Sprachen, die in einer vorgegebenen Platzschranke $s(n)$ entscheidbar sind, in folgenden **Platzkomplexitätsklassen** zusammen.

Definition

- Die auf deterministischem Platz $s(n)$ entscheidbaren Sprachen bilden die Klasse

$$\text{DSPACE}(s(n)) = \{L(M) \mid M \text{ ist eine } s(n)\text{-platzb. offline-DTM}\}.$$

- Die auf nichtdeterministischem Platz $s(n)$ entscheidbaren Sprachen bilden die Klasse

$$\text{NSPACE}(s(n)) = \{L(M) \mid M \text{ ist eine } s(n)\text{-platzb. offline-NTM}\}.$$

- Die wichtigsten deterministischen Platzkomplexitätsklassen sind

$L = \text{DSPACE}(\mathcal{O}(\log n))$ „Logarithmischer Platz“

$\text{Linspace} = \text{DSPACE}(\mathcal{O}(n))$ „Linearer Platz“

$\text{PSPACE} = \text{DSPACE}(n^{\mathcal{O}(1)})$ „Polynomieller Platz“

- Die nichtdeterministischen Klassen NL , NLinspace und NPSPACE sind analog definiert.

Frage

Welche elementaren Beziehungen gelten zwischen den verschiedenen Zeit- und Platzklassen?

Satz

- Für jede Funktion $t(n) \geq n + 2$ gilt

$$\text{DTIME}(t) \subseteq \text{NTIME}(t) \subseteq \text{DSPACE}(\mathcal{O}(t)).$$

- Für jede Funktion $s(n) \geq \log n$ gilt

$$\text{DSPACE}(s) \subseteq \text{NSPACE}(s) \subseteq \text{DTIME}(2^{\mathcal{O}(s)}) \text{ und}$$

$$\text{NSPACE}(s) \subseteq \text{DSPACE}(s^2). \quad (\text{Satz von Savitch})$$

Korollar

$$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} = \text{NSPACE} \subseteq \text{EXP} \subseteq \text{NEXP} \subseteq \text{EXPSPACE}.$$

$\text{REG} = \text{DSPACE}(\mathcal{O}(1)) = \text{NSPACE}(\mathcal{O}(1)) \not\subseteq L,$

$\text{DCFL} \not\subseteq \text{LINTIME},$

$\text{CFL} \not\subseteq \text{NLINTIME} \cap \text{DTIME}(\mathcal{O}(n^3)) \not\subseteq P,$

$\text{DCSL} = \text{LINSPACE} \subseteq \text{CSL},$

$\text{CSL} = \text{NLINSPACE} \subseteq \text{PSPACE} \cap E,$

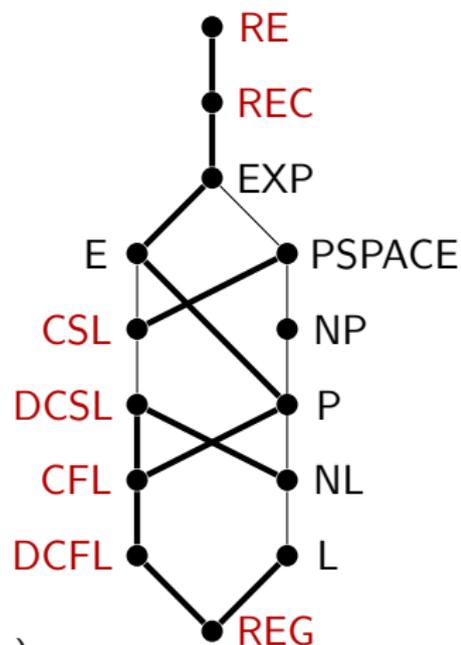
$\text{REC} = \bigcup_f \text{DSPACE}(f(n))$

$= \bigcup_f \text{NSPACE}(f(n))$

$= \bigcup_f \text{DTIME}(f(n))$

$= \bigcup_f \text{NTIME}(f(n)),$

wobei f alle (oder äquivalent: alle berechenbaren)
Funktionen $f : \mathbb{N} \rightarrow \mathbb{N}$ durchläuft.



Aussagenlogische Formeln

- Die Menge der **booleschen** (oder **aussagenlogischen**) **Formeln** über den Variablen x_1, \dots, x_n , $n \geq 0$, ist induktiv wie folgt definiert:
 - Die Konstanten 0 und 1 sind boolesche Formeln.
 - Jede Variable x_i ist eine boolesche Formel.
 - Mit G und H sind auch die **Konjunktion** ($G \wedge H$) und die **Disjunktion** ($G \vee H$) von G und H sowie die **Negation** $\neg G$ von G Formeln.
- Eine **Belegung** von x_1, \dots, x_n ist ein Wort $a = a_1 \dots a_n \in \{0, 1\}^n$.
- Der **Wert** $F(a)$ von F unter a ist induktiv wie folgt definiert:

F	0	1	x_i	$\neg G$	$(G \wedge H)$	$(G \vee H)$
$F(a)$	0	1	a_i	$1 - G(a)$	$G(a)H(a)$	$G(a) + H(a) - G(a)H(a)$

- Durch die Formel F wird also eine **n -stellige boolesche Funktion** $F : \{0, 1\}^n \rightarrow \{0, 1\}$ definiert, die wir ebenfalls mit F bezeichnen.

Aussagenlogische Formeln

Notation

Wir benutzen die **Implikation** $G \rightarrow H$ als Abkürzung für die Formel $\neg G \vee H$ und die **Äquivalenz** $G \leftrightarrow H$ als Abkürzung für $(G \rightarrow H) \wedge (H \rightarrow G)$.

Beispiel (Wahrheitstabelle)

Die Formel $F = (G \rightarrow H)$ mit den Teilformeln

- $G = (x_2 \wedge x_3)$ und
- $H = (\neg x_1 \vee \neg x_2)$

berechnet nebenstehende boolesche Funktion $F : \{0, 1\}^3 \rightarrow \{0, 1\}$.

a	$G(a)$	$H(a)$	$F(a)$
000	0	1	1
001	0	1	1
010	0	1	1
011	1	1	1
100	0	1	1
101	0	1	1
110	0	0	1
111	1	0	0



Aussagenlogische Formeln

Definition

- Zwei Formeln F und G heißen **(logisch) äquivalent** (kurz $F \equiv G$), wenn sie dieselbe boolesche Funktion berechnen.
- Eine Formel F heißt **erfüllbar**, falls es eine Belegung a mit $F(a) = 1$ gibt.
- Gilt sogar für alle Belegungen a , dass $F(a) = 1$ ist, so heißt F **Tautologie**.

Beispiel

- Die Formel $F = (G \rightarrow H)$ mit $G = (x_2 \wedge x_3)$ und $H = (\neg x_1 \vee \neg x_2)$ ist erfüllbar, da $F(000) = 1$ ist.
- F ist aber keine Tautologie, da $F(111) = 0$ ist.



Aussagenlogische Formeln

Präzedenzregeln zur Klammerersparnis

- Der Junktor \wedge bindet stärker als der Junktor \vee und dieser wiederum stärker als die Junktoren \rightarrow und \leftrightarrow .
- Formeln der Form $(F_1 \circ (F_2 \circ (F_3 \circ \dots \circ F_n) \dots))$, $\circ \in \{\wedge, \vee\}$, kürzen wir durch $(F_1 \circ \dots \circ F_n)$ ab und schreiben dafür auch $\bigwedge_{1 \leq i \leq n} F_i$ bzw. $\bigvee_{1 \leq i \leq n} F_i$.

Beispiel (Formel für die mehrstellige Entweder-Oder Funktion)

- Folgende Formel nimmt unter einer Belegung $a = a_1 \dots a_n$ genau dann den Wert 1 an, wenn $\sum_{i=1}^n a_i = 1$ ist:

$$G(x_1, \dots, x_n) = (x_1 \vee \dots \vee x_n) \wedge \bigwedge_{1 \leq i < j \leq n} \neg(x_i \wedge x_j)$$

- D.h. es gilt genau dann $G(a) = 1$, wenn genau eine Variable x_i mit dem Wert $a_i = 1$ belegt ist.
- Diese Formel wird im Beweis des nächsten Satzes benötigt. ◀

Erfüllbarkeitsproblem für boolesche Formeln (*satisfiability*, SAT):

Gegeben: Eine boolesche Formel F .

Gefragt: Ist F erfüllbar?

Dabei kodieren wir boolesche Formeln F durch Binärstrings w_F und ordnen umgekehrt jedem Binärstring w eine Formel F_w zu.

Um die Notation zu vereinfachen, werden wir zukünftig jedoch F anstelle von w_F schreiben.

Satz (Cook, Karp, Levin)

SAT ist NP-vollständig.

SAT ist NP-vollständig

SAT \in NP

Eine NTM kann bei Eingabe einer booleschen Formel F zunächst eine Belegung a nichtdeterministisch raten und dann in Polynomialzeit testen, ob $F(a) = 1$ ist (*guess and verify* Strategie).

SAT ist NP-hart

- Sei L eine beliebige NP-Sprache und sei $M = (Z, \Sigma, \Gamma, \delta, q_0)$ eine durch ein Polynom p zeitbeschränkte k -NTM mit $L(M) = L$.
- Da sich jede $t(n)$ -zeitbeschränkte k -NTM in Zeit $O(t^2(n))$ durch eine 1-NTM simulieren lässt, können wir $k = 1$ annehmen.
- Zudem können wir $Z = \{q_0, \dots, q_m\}$, $E = \{q_m\}$ und $\Gamma = \{a_1, \dots, a_l\}$ sowie $\delta(q_m, a) = \{(q_m, a, N)\}$ für alle $a \in \Gamma$ annehmen.
- Unsere Aufgabe besteht nun darin, zu jedem Wort $w = w_1 \dots w_n \in \Sigma^*$ eine Formel F_w mit folgenden Eigenschaften zu konstruieren:
 - $w \in L(M) \iff F_w \in \text{SAT}$,
 - die Reduktionsfunktion $w \mapsto F_w$ ist in FP berechenbar.

Idee:

Konstruiere F_w so, dass F_w unter einer Belegung a genau dann wahr wird, wenn a eine akzeptierende Rechnung von $M(w)$ beschreibt.

- Wir bilden F_w über den Variablen

$$x_{t,i}, \quad \text{für } 0 \leq t \leq p(n), 0 \leq i \leq m$$

$$y_{t,j}, \quad \text{für } 0 \leq t \leq p(n), -p(n) \leq j \leq p(n)$$

$$z_{t,j,a}, \quad \text{für } 0 \leq t \leq p(n), -p(n) \leq j \leq p(n), a \in \Gamma$$

- Diese Variablen stehen für folgende Aussagen:

$x_{t,i}$: zum Zeitpunkt t befindet sich M im Zustand q_i

$y_{t,j}$: zur Zeit t besucht M das Feld mit der Nummer j

$z_{t,j,a}$: zur Zeit t steht das Zeichen a auf dem Feld mit der Nr. j

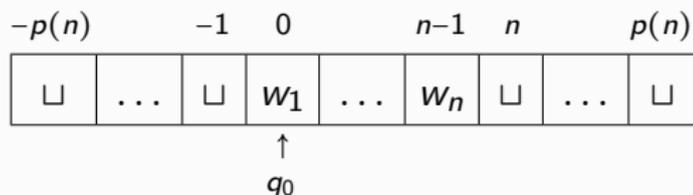
- Konkret sei $F_w = R \wedge S_w \wedge \dot{U}_1 \wedge \dot{U}_2 \wedge E$.

- Konkret sei $F_w = R \wedge S_w \wedge \ddot{U}_1 \wedge \ddot{U}_2 \wedge E$.
- Dabei stellt die Formel $R = \bigwedge_{t=0}^{p(n)} R_t$ (Randbedingungen) sicher, dass wir jeder erfüllenden Belegung von F_w eindeutig eine Folge von Konfigurationen $K_0, \dots, K_{p(n)}$ zuordnen können:

$$R_t = G(x_{t,0}, \dots, x_{t,m}) \wedge G(y_{t,-p(n)}, \dots, y_{t,p(n)}) \\ \wedge \bigwedge_{j=-p(n)}^{p(n)} G(z_{t,j,a_1}, \dots, z_{t,j,a_l}).$$

- Die Teilformel R_t sorgt also dafür, dass zum Zeitpunkt t
 - genau ein Zustand $q_i \in Z$ eingenommen wird,
 - genau ein Bandfeld $j \in \{-p(n), \dots, p(n)\}$ besucht wird und
 - auf jedem Feld j genau ein Zeichen $a_k \in \Gamma = \{a_1, \dots, a_l\}$ steht.

- Die Formel S_w (wie Startbedingung) stellt sicher, dass zum Zeitpunkt 0 tatsächlich die Startkonfiguration vorliegt:



$$S_w = x_{0,0} \wedge y_{0,0} \wedge \bigwedge_{j=-p(n)}^{-1} z_{0,j,\sqcup} \wedge \bigwedge_{j=0}^{n-1} z_{0,j,w_{j+1}} \wedge \bigwedge_{j=n}^{p(n)} z_{0,j,\sqcup}$$

- Die Formel \ddot{U}_1 sorgt dafür, dass der Inhalt von nicht besuchten Feldern beim Übergang von K_t zu K_{t+1} unverändert bleibt:

$$\ddot{U}_1 = \bigwedge_{t=0}^{p(n)-1} \bigwedge_{j=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} (\neg y_{t,j} \wedge z_{t,j,a} \rightarrow z_{t+1,j,a})$$

- \ddot{U}_2 achtet darauf, dass sich bei jedem Rechenschritt der Zustand, die Kopfposition und das gerade gelesene Zeichen gemäß einer Anweisung in δ verändern:

$$\ddot{U}_2 = \bigwedge_{t=0}^{p(n)-1} \bigwedge_{j=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} \bigwedge_{i=0}^m (x_{t,i} \wedge y_{t,j} \wedge z_{t,j,a} \rightarrow$$

wobei
$$\bigvee_{(q_{t'}, a', D) \in \delta(q_t, a)} x_{t+1, i'} \wedge y_{t+1, j+D} \wedge z_{t+1, j, a'}),$$

$$j + D = \begin{cases} j - 1, & D = L \\ j, & D = N \\ j + 1, & D = R \end{cases}$$

- Schließlich überprüft E , ob $M(w)$ nach (spätestens) $p(n)$ Schritten den Endzustand q_m erreicht hat:

$$E = x_{p(n), m}$$

- Da der Aufbau der Formel $f(w) = F_w$ einem einfachen Bildungsgesetz folgt und ihre Länge polynomiell in n ist, folgt $f \in \text{FP}$.
- Es ist klar, dass F_w im Fall $w \in L(M)$ erfüllbar ist, indem wir die Variablen von F_w gemäß einer akz. Rechnung von $M(w)$ belegen.
- Umgekehrt führt eine Belegung a mit $F_w(a) = 1$ wegen $R(a) = 1$ eindeutig auf eine Konfigurationenfolge $K_0, \dots, K_{p(n)}$.
- Für diese gilt:
 - K_0 ist Startkonfiguration von $M(w)$ (wegen $S_w(a) = 1$)
 - $K_i \vdash K_{i+1}$ für $i = 0, \dots, p(n) - 1$ (wegen $\ddot{U}_1(a) = \ddot{U}_2(a) = 1$)
 - der Zustand von $K_{p(n)}$ ist m (wegen $E(a) = 1$)
- Also gilt für alle $w \in \Sigma^*$ die Äquivalenz

$$w \in L(M) \Leftrightarrow F_w \in \text{SAT},$$

d.h. die FP-Funktion $f : w \mapsto F_w$ reduziert $L(M)$ auf SAT. □

Boolesche Schaltkreise

Als nächstes betrachten wir das Erfüllbarkeitsproblem für Schaltkreise.

Definition

- Ein **boolescher Schaltkreis** über den Variablen x_1, \dots, x_n ist eine Folge $S = (g_1, \dots, g_m)$ von **Gattern**

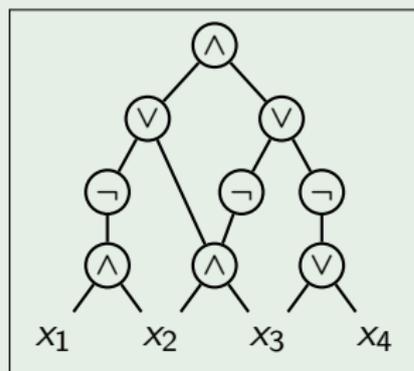
$$g_l \in \{0, 1, x_1, \dots, x_n, (\neg, j), (\wedge, j, k), (\vee, j, k)\} \text{ mit } 1 \leq j, k < l.$$

- Jedes Gatter g_l berechnet eine n -stellige boolesche Funktion $g_l: \{0, 1\}^n \rightarrow \{0, 1\}$.
- Für $a = a_1 \dots a_n \in \{0, 1\}^n$ ist $g_l(a)$ induktiv wie folgt definiert:

g_l	0	1	x_i	(\neg, j)	(\wedge, j, k)	(\vee, j, k)
$g_l(a)$	0	1	a_i	$1 - g_j(a)$	$g_j(a)g_k(a)$	$g_j(a) + g_k(a) - g_j(a)g_k(a)$

- S berechnet die boolesche Funktion $S(a) = g_m(a)$.
- S heißt **erfüllbar**, wenn eine Eingabe $a \in \{0, 1\}^n$ ex. mit $S(a) = 1$.

Beispiel



Graphische Darstellung
des Schaltkreises

$$S = (x_1, x_2, x_3, x_4, (\wedge, 1, 2),$$

$$(\wedge, 2, 3), (\vee, 3, 4), (\neg, 5),$$

$$(\neg, 6), (\neg, 7), (\vee, 6, 8),$$

$$(\vee, 9, 10), (\wedge, 11, 12)).$$

Bemerkung

- Die Anzahl der Eingänge eines Gatters g wird als **Fanin** von g bezeichnet,
- die Anzahl der Ausgänge von g (also die Anzahl der Gatter, die g als Eingabe benutzen) als **Fanout**.
- Boolesche Formeln entsprechen also genau den booleschen Schaltkreisen $S = (g_1, \dots, g_m)$, bei denen jedes Gatter g_i , $1 \leq i \leq m-1$, Fanout 1 hat.
- Eine boolesche Formel F kann somit leicht in einen äquivalenten Schaltkreis S mit $S(a) = F(a)$ für alle Belegungen a transformiert werden.

Erfüllbarkeitsproblem für boolesche Schaltkreise (CIRSAT):

Gegeben: Ein boolescher Schaltkreis S .

Gefragt: Ist S erfüllbar?

Satz

CIRSAT ist NP-vollständig.

Beweis

Klar, da $SAT \leq^P CIRSAT$ und $CIRSAT \in NP$ gilt. \square

Bemerkung

- Da SAT NP-vollständig ist, ist auch CIRSAT auf SAT reduzierbar.
- Dies bedeutet, dass sich jeder Schaltkreis S in Polynomialzeit in eine **erfüllbarkeitsäquivalente** Formel F_S überführen lässt.
 F_S und S müssen aber nicht logisch äquivalent sein.
- CIRSAT ist sogar auf eine spezielle SAT-Variante reduzierbar.

Formeln in konjunktiver Normalform (KNF)

- Ein **Literal** ist eine Variable x_i oder eine negierte Variable $\neg x_i$, die wir auch kurz mit \bar{x}_i bezeichnen.
- Eine **Klausel** ist eine Disjunktion $C = \bigvee_{j=1}^k l_j$ von Literalen l_1, \dots, l_k .
- Hierbei ist auch $k = 0$ zulässig, d.h. die **leere Klausel** repräsentiert die Konstante 0 und wird üblicherweise mit \square bezeichnet.
- Eine boolesche Formel F ist in **konjunktiver Normalform** (kurz **KNF**), falls $F = \bigwedge_{j=1}^m C_j$ eine Konjunktion von Klauseln C_1, \dots, C_m ist.
- Auch hier ist $m = 0$ zulässig, wobei die **leere Konjunktion** die Konstante 1 repräsentiert.
- Enthält jede Klausel höchstens k Literale, so heißt F in **k -KNF**.
- Klauseln werden oft als Menge $C = \{l_1, \dots, l_k\}$ ihrer Literale und KNF-Formeln als Menge $F = \{C_1, \dots, C_m\}$ ihrer Klauseln dargestellt.
- Enthält F die leere Klausel, so ist F unerfüllbar.
- Dagegen ist die leere Formel eine Tautologie.

Erfüllbarkeitsprobleme für KNF-Formeln

Erfüllbarkeitsproblem für k -KNF Formeln (k -SAT):

Gegeben: Eine boolesche Formel F in k -KNF.

Gefragt: Ist F erfüllbar?

Beispiel

- Der 3-KNF Formel $F = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4)$ entspricht die Klauselmengemenge $F = \{\{x_1, \bar{x}_2\}, \{\bar{x}_1, x_3\}, \{x_2, \bar{x}_3, x_4\}\}$.
- Offenbar ist $F(0000) = 1$, d.h. $F \in 3$ -SAT.

Satz

k -SAT ist für $k \leq 2$ in P entscheidbar und für $k \geq 3$ NP-vollständig.

3-SAT ist NP-vollständig

Reduktion von CIRSAT auf 3-SAT

- Wir transformieren einen Schaltkreis $S = (g_1, \dots, g_m)$ mit n Eingängen in eine Formel F_S über den Variablen $x_1, \dots, x_n, y_1, \dots, y_m$.
- F_S enthält die Klausel $\{y_m\}$ und für jedes Gatter g_i die Klauseln einer 3-KNF F_i , die zu folgender Formel G_i äquivalent ist:

Gatter g_i	G_i	Klauseln von F_i
0	$y_i \leftrightarrow 0$	$\{\bar{y}_i\}$
1	$y_i \leftrightarrow 1$	$\{y_i\}$
x_j	$y_i \leftrightarrow x_j$	$\{\bar{y}_i, x_j\}, \{\bar{x}_j, y_i\}$
(\neg, j)	$y_i \leftrightarrow \bar{y}_j$	$\{\bar{y}_i, \bar{y}_j\}, \{y_j, y_i\}$
(\wedge, j, k)	$y_i \leftrightarrow y_j \wedge y_k$	$\{\bar{y}_i, y_j\}, \{\bar{y}_i, y_k\}, \{\bar{y}_j, \bar{y}_k, y_i\}$
(\vee, j, k)	$y_i \leftrightarrow y_j \vee y_k$	$\{\bar{y}_j, y_i\}, \{\bar{y}_k, y_i\}, \{\bar{y}_i, y_j, y_k\}$

3-SAT ist NP-vollständig

Reduktion von CIRSAT auf 3-SAT

- Wir zeigen, dass für alle $a \in \{0, 1\}^n$ folgende Äquivalenz gilt:

$$S(a) = 1 \Leftrightarrow \exists b \in \{0, 1\}^m : F_S(ab) = 1.$$

- Ist nämlich $a \in \{0, 1\}^n$ eine Eingabe mit $S(a) = 1$. Dann erhalten wir mit

$$b_i = g_i(a) \text{ für } i = 1, \dots, m$$

eine erfüllende Belegung $ab_1 \dots b_m$ für F_S .

- Ist umgekehrt $ab_1 \dots b_m$ eine erfüllende Belegung für F_S , so muss

- $b_m = 1$ sein, da $\{y_m\}$ eine Klausel in F_S ist, und
- durch Induktion über $i = 1, \dots, m$ folgt

$$g_i(a) = b_i,$$

d.h. insbesondere folgt $S(a) = g_m(a) = b_m = 1$.

Reduktion von CIRSAT auf 3-SAT

- Wir wissen bereits, dass für alle $a \in \{0, 1\}^n$ die Äquivalenz

$$S(a) = 1 \Leftrightarrow \exists b \in \{0, 1\}^m : F_S(ab) = 1.$$

gilt.

- Dies bedeutet, dass der Schaltkreis S und die 3-KNF-Formel F_S erfüllbarkeitsäquivalent sind, d.h.

$$S \in \text{CIRSAT} \Leftrightarrow F_S \in \text{3-SAT}.$$

- Da zudem die Reduktionsfunktion $S \mapsto F_S$ in FP berechenbar ist, folgt $\text{CIRSAT} \leq^P \text{3-SAT}$. □

Notation – ungerichtete Graphen

- Ein (**ungerichteter**) **Graph** ist ein Paar $G = (V, E)$, wobei
 - V - eine endliche Menge von **Knoten/Ecken** und
 - E - die Menge der **Kanten** ist.

Hierbei gilt

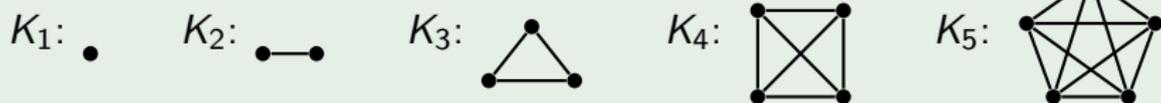
$$E \subseteq \binom{V}{2} := \{\{u, v\} \subseteq V \mid u \neq v\}.$$

- Die **Knotenzahl** von G ist $n(G) = \|V\|$.
- Die **Kantenzahl** von G ist $m(G) = \|E\|$.
- Die **Nachbarschaft** von $v \in V$ ist $N_G(v) = \{u \in V \mid \{u, v\} \in E\}$ und die Nachbarschaft von $U \subseteq V$ ist $N_G(U) = \bigcup_{u \in U} N_G(u)$.
- Der **Grad** von $v \in V$ ist $\deg_G(v) = \|N_G(v)\|$.
- Der **Minimalgrad** von G ist $\delta(G) := \min_{v \in V} \deg_G(v)$ und
- der **Maximalgrad** von G ist $\Delta(G) := \max_{v \in V} \deg_G(v)$.

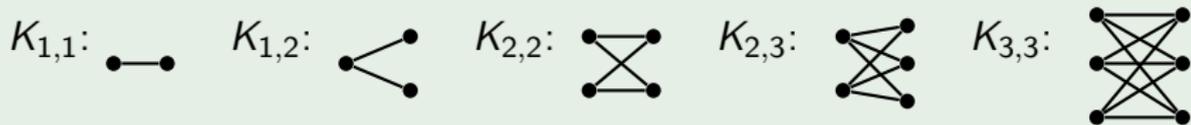
Falls G aus dem Kontext ersichtlich ist, schreiben wir auch einfach n , m , $N(v)$, $\deg(v)$, δ usw.

Beispiel

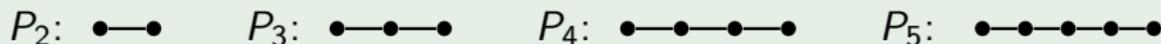
- Der **vollständige Graph** (V, E) mit $\|V\| = n$ und $E = \binom{V}{2}$ wird mit K_n und der **leere Graph** (V, \emptyset) wird mit E_n bezeichnet.



- Der **vollständige bipartite Graph** (A, B, E) auf $a + b$ Knoten, d.h. $A \cap B = \emptyset$, $\|A\| = a$, $\|B\| = b$ und $E = \{\{u, v\} \mid u \in A, v \in B\}$ wird mit $K_{a,b}$ bezeichnet.



- Der **Pfadgraph** (oder **lineare Graph**) mit n Knoten heißt P_n :



- Der **Kreisgraph** (kurz **Kreis**) mit n Knoten heißt C_n :



- Ein Graph $H = (V', E')$ heißt **Sub-/Teil-/Untergraph** von $G = (V, E)$, falls $V' \subseteq V$ und $E' \subseteq E$ ist.
- Ein **Weg** ist eine Folge von Knoten v_0, \dots, v_j mit $\{v_i, v_{i+1}\} \in E$ für $i = 0, \dots, j - 1$.

Ein Weg heißt **einfach** oder **Pfad**, falls alle durchlaufenen Knoten verschieden sind.

Die **Länge** des Weges ist die Anzahl der Kanten, also j .

Ein Weg v_0, \dots, v_j heißt auch **v_0 - v_j -Weg**.

- Ein **Zyklus** ist ein u - v -Weg der Länge $j \geq 2$ mit $u = v$.
- Ein **Kreis** ist ein Zyklus $v_0, v_1, \dots, v_{j-1}, v_0$ der Länge $j \geq 3$, für den v_0, v_1, \dots, v_{j-1} paarweise verschieden sind.

Cliquen, Stabilität und Matchings

- Eine Knotenmenge $U \subseteq V$ heißt **Clique**, wenn E alle Kanten enthält, die beide Endpunkte in U haben, d.h. es gilt $\binom{U}{2} \subseteq E$.

Die **Cliquenzahl** ist

$$\omega(G) = \max\{\|U\| \mid U \text{ ist Clique in } G\}.$$

- Eine Knotenmenge $U \subseteq V$ heißt **stabil** oder **unabhängig**, wenn keine Kante in G beide Endpunkte in U hat, d.h. es gilt $E \cap \binom{U}{2} = \emptyset$.

Die **Stabilitätszahl** ist

$$\alpha(G) = \max\{\|U\| \mid U \text{ ist stabile Menge in } G\}.$$

- Zwei Kanten $e, e' \in E$ heißen **unabhängig**, falls $e \cap e' = \emptyset$ ist.

Eine Kantenmenge $M \subseteq E$ heißt **Matching** in G , falls alle Kanten in M paarweise unabhängig sind.

Die **Matchingzahl** von G ist

$$\mu(G) = \max\{\|M\| \mid M \text{ ist ein Matching in } G\}.$$

Knotenüberdeckungen und Färbungen

- Eine Knotenmenge $U \subseteq V$ heißt **Knotenüberdeckung** (engl. *vertex cover*), wenn jede Kante $e \in E$ mindestens einen Endpunkt in U hat, d.h. es gilt $e \cap U \neq \emptyset$ für alle Kanten $e \in E$.

Die **Überdeckungszahl** ist

$$\beta(G) = \min\{\|U\| \mid U \text{ ist eine Knotenüberdeckung in } G\}.$$

- Eine Abbildung $f: V \rightarrow \mathbb{N}$ heißt **Färbung** von G , wenn $f(u) \neq f(v)$ für alle $\{u, v\} \in E$ gilt.

G heißt **k -färbbar**, falls eine Färbung $f: V \rightarrow \{1, \dots, k\}$ existiert.

Die **chromatische Zahl** ist

$$\chi(G) = \min\{k \in \mathbb{N} \mid G \text{ ist } k\text{-färbbar}\}.$$

Eulerlinien und -touren

Definition

Sei $s = (v_0, v_1, \dots, v_l)$ ein Weg in einem Graphen $G = (V, E)$, d.h. $\{v_i, v_{i+1}\} \in E$ für $i = 0, \dots, l-1$.

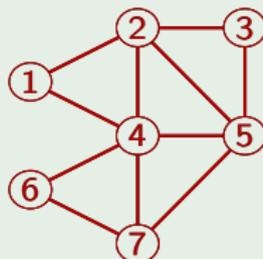
- Dann heißt s **Eulerlinie** (auch **Eulerzug** oder **Eulerweg**) in G , falls s jede Kante in E genau einmal durchläuft, d.h. es gilt $l = \|E\|$ und

$$\left\{ \{v_i, v_{i+1}\} \mid i = 0, \dots, l-1 \right\} = E.$$

- Ist s zudem ein Zyklus, d.h. es gilt $v_l = v_0$, so heißt s **Eulerkreis** (auch **Eulerzyklus** oder **Eulertour**).

Beispiel (Eulerlinie)

$$s = (4, 1, 2, 3, 5, 7, 6, 4, 5, 2, 4, 7)$$



Gerichtete Eulerlinien

Definition

Sei $s = (v_0, v_1, \dots, v_l)$ ein gerichteter Weg in einem Digraphen $G = (V, E)$, d.h. $(v_i, v_{i+1}) \in E$ für $i = 0, \dots, l-1$.

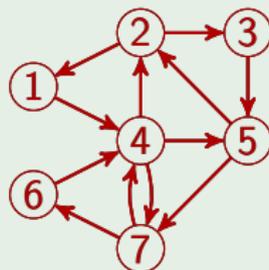
- Dann heißt s **Eulerlinie** (auch **Eulerzug** oder **Eulerweg**) in G , falls s jede Kante in E genau einmal durchläuft, d.h. es gilt $l = \|E\|$ und

$$\{(v_i, v_{i+1}) \mid i = 0, \dots, l-1\} = E.$$

- Ist s zudem ein Zyklus, d.h. es gilt $v_l = v_0$, so heißt s **Eulerkreis** (auch **Eulerzyklus** oder **Eulertour**).

Beispiel (Eulerkreis in einem Digraphen)

$$s = (1, 4, 5, 2, 3, 5, 7, 4, 7, 6, 4, 2, 1)$$



Definition

Sei $s = (v_0, v_1, \dots, v_l)$ ein Pfad in einem Graphen $G = (V, E)$, d.h. $\{v_i, v_{i+1}\} \in E$ für $i = 0, \dots, l-1$ und v_0, \dots, v_l sind paarweise verschieden.

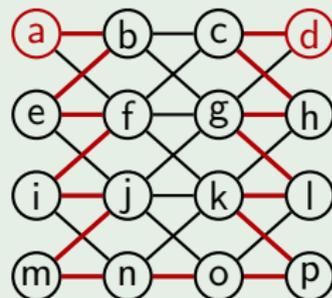
- Dann heißt s **Hamiltonpfad** in G , falls s jeden Knoten in V genau einmal durchläuft, d.h. es gilt

$$V = \{v_0, \dots, v_l\} \text{ und } l = \|V\| - 1.$$

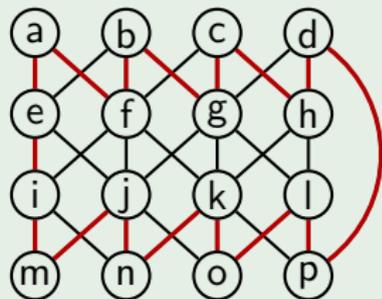
- Ist zudem $\{v_0, v_l\} \in E$, d.h. $s' = (v_0, v_1, \dots, v_l, v_0)$ ist ein Kreis, so heißt s' **Hamiltonkreis**.

Hamiltonpfade und -kreise

Beispiel (Hamiltonpfad)

$$s = (a, b, e, f, i, j, m, n, o, p, k, l, g, h, c, d)$$


Beispiel (Hamiltonkreis)

$$s = (a, f, b, g, c, h, d, p, l, o, k, n, j, m, i, e, a)$$


Gerichtete Hamiltonpfade

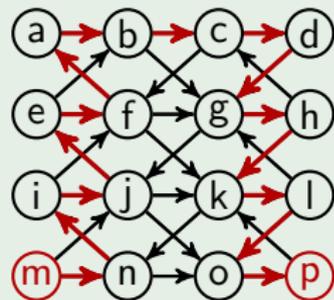
Definition

Sei $s = (v_0, v_1, \dots, v_l)$ ein gerichteter Pfad in einem Digraphen $G = (V, E)$, d.h. $(v_i, v_{i+1}) \in E$ für $i = 0, \dots, l-1$ und v_0, \dots, v_l sind alle verschieden.

- Dann heißt s **Hamiltonpfad** in G , falls s jeden Knoten in V genau einmal durchläuft, d.h. es gilt $V = \{v_0, \dots, v_l\}$ und $l = \|V\| - 1$.
- Ist zudem $(v_l, v_0) \in E$, d.h. $s' = (v_0, v_1, \dots, v_l, v_0)$ ist ein gerichteter Kreis, so heißt s' **Hamiltonkreis**.

Beispiel (Hamiltonpfad in einem Digraphen)

$s = (m, n, i, j, e, f, a, b, c, d, g, h, k, l, o, p)$



Für einen gegebenen Graphen G und eine Zahl $k \geq 1$ betrachten wir folgende Probleme:

- **CLIQUE:**
Hat G eine Clique der Größe k ?
- **MATCHING:**
Hat G ein Matching der Größe k ?
- **Independent Set (IS):**
Hat G eine stabile Menge der Größe k ?
- **Vertex Cover (VC):**
Hat G eine Knotenüberdeckung der Größe k ?
- **Färbbarkeit (COLORING):**
Ist G k -färbbar?

Weitere algorithmische Graphprobleme

Zudem betrachten wir für einen gegebenen Graphen G folgende Probleme:

- **k -FÄRBBARKEIT (k -COLORING)** für jede natürliche Zahl $k \geq 1$:
Ist G k -färbbar?
- **Das Eulerkreisproblem (EULERCYCLE)**:
Hat G einen Eulerkreis?
- **Das Hamiltonkreisproblem (HAMCYCLE)**:
Hat G einen Hamiltonkreis?

und für einen Graphen G und zwei Knoten s und t folgende Probleme:

- **Das Eulerlinienproblem (EULERPATH)**:
Hat G eine Eulerlinie von s nach t ?
- **Das Hamiltonpfadproblem (HAMPATH)**:
Hat G einen Hamiltonpfad von s nach t ?

Weitere algorithmische Graphprobleme

Zudem betrachten wir für einen gegebenen Digraphen G folgende Probleme:

- **Das gerichtete Eulerkreisproblem (DIEULERCYCLE):**
Hat G einen Eulerkreis?
- **Das gerichtete Hamiltonkreisproblem (DIHAMCYCLE):**
Hat G einen Hamiltonkreis?

und für einen gegebenen Digraphen G und zwei Knoten s und t :

- **Das gerichtete Eulerlinienproblem (DIEULERPATH):**
Hat G eine Eulerlinie von s nach t ?
- **Das gerichtete Hamiltonpfadproblem (DIHAMPATH):**
Hat G einen Hamiltonpfad von s nach t ?

Satz

- CLIQUE, IS, VC, COLORING, 3-COLORING, HAMCYCLE, HAMPATH, DIHAMPATH und DIHAMCYCLE sind NP-vollständig.
- 2-COLORING, MATCHING, EULERCYCLE, EULERPATH, DIEULERCYCLE und DIEULERPATH sind in P entscheidbar.

IS ist NP-vollständig

Reduktion von 3-SAT auf IS

- Sei $F = \{C_1, \dots, C_m\}$ mit $C_j = \{l_{j,1}, \dots, l_{j,k_j}\}$ für $j = 1, \dots, m$ eine 3-KNF-Formel über den Variablen x_1, \dots, x_n .

- Betrachte den Graphen $G = (V, E)$ mit

$$V = \{v_{ji} \mid 1 \leq j \leq m, 1 \leq i \leq k_j\} \text{ und}$$

$$E = \{\{v_{ji}, v_{j'i'}\} \in \binom{V}{2} \mid j = j' \text{ oder } l_{ji} = \bar{l}_{j'i'} \text{ oder } l_{j'i'} = \bar{l}_{ji}\}.$$

- Nun gilt

$F \in 3\text{-SAT} \Leftrightarrow$ es gibt eine Belegung, die in jeder Klausel C_j (mindestens) ein Literal l_{j,i_j} wahr macht

\Leftrightarrow es gibt m Literale $l_{1,i_1}, \dots, l_{m,i_m}$, die paarweise nicht komplementär sind (d.h. $l_{j,i_j} \neq \bar{l}_{j',i_{j'}}$ für $j \neq j'$)

\Leftrightarrow es gibt m Knoten $v_{1,i_1}, \dots, v_{m,i_m}$, die nicht durch Kanten verbunden sind

$\Leftrightarrow G$ besitzt eine stabile Menge von m Knoten. □

Korollar

CLIQUE ist NP-vollständig.

Beweis

- Es ist klar, dass jede Clique in einem Graphen $G = (V, E)$ eine stabile Menge in dem zu G komplementären Graphen $\bar{G} = (V, \bar{E})$ mit $\bar{E} = \binom{V}{2} \setminus E$ ist und umgekehrt.
- Daher lässt sich IS mittels

$$f : (G, k) \mapsto (\bar{G}, k)$$

auf CLIQUE reduzieren. □

Korollar

VC ist NP-vollständig.

Beweis

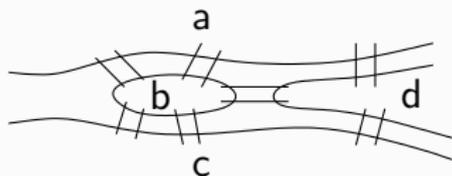
- Offensichtlich ist eine Menge I genau dann stabil, wenn ihr Komplement $V \setminus I$ eine Knotenüberdeckung ist.
- Daher lässt sich IS mittels

$$f : (G, k) \mapsto (G, n(G) - k)$$

auf VC reduzieren. □

Das Königsberger Brückenproblem

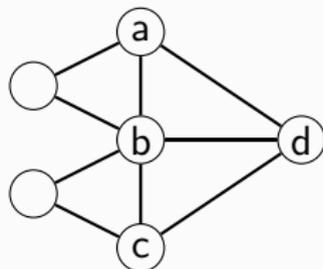
Die 7 Königsberger Brücken



Frage

Gibt es einen Spaziergang über alle 7 Brücken, bei dem keine Brücke mehrmals überquert wird und der zum Ausgangspunkt zurückführt?

Gelöst von Euler (1707 – 1783) durch Betrachtung des folgenden Graphen, der offenbar genau dann einen Eulerkreis hat, wenn die Antwort auf obige Frage „ja“ ist.



Satz (Euler, 1736)

Ein zusammenhängender Graph $G = (V, E)$ besitzt genau dann einen Eulerkreis, wenn all seine Knoten geraden Grad haben.

Satz (Euler, 1736)

Ein zusammenhängender Graph $G = (V, E)$ besitzt genau dann einen Eulerkreis, wenn all seine Knoten geraden Grad haben.

Beweis

- Falls G einen Eulerkreis s besitzt, existiert zu jeder Kante, auf der s zu einem Knoten gelangt, eine weitere Kante, auf der s den Knoten wieder verlässt.
- Daher muss jeder Knoten geraden Grad haben.
- Ist umgekehrt G zusammenhängend und hat jeder Knoten geraden Grad, so können wir wie folgt einen Eulerkreis s konstruieren.

Eulerkreise und -linien

- Ist umgekehrt G zusammenhängend und hat jeder Knoten geraden Grad, so können wir wie folgt einen Eulerkreis s konstruieren.

Algorithmus zur Berechnung eines Eulerkreises in $G = (V, E)$

- 1 Wähle $u \in V$ beliebig und initialisiere s zu $s = (u)$
- 2 Wähle einen beliebigen Knoten u auf dem Weg s , der mit einer unmarkierten Kante verbunden ist.
- 3 Folge ausgehend von u den unmarkierten Kanten auf einem beliebigen Weg z solange wie möglich und markiere dabei jede durchlaufene Kante. (Da von jedem erreichten Knoten $v \neq u$ ungerade viele markierte Kanten ausgehen, muss der Weg z zum Ausgangspunkt u zurückführen.)
- 4 Füge den Zyklus z an der Stelle u in s ein.
- 5 Wenn noch nicht alle Kanten markiert sind, gehe zu 2.
- 6 **Output:** s



Satz (Euler, 1736)

- i) Ein zusammenhängender Graph $G = (V, E)$ besitzt im Fall $s \neq t$ genau dann eine Eulerlinie von s nach t , wenn s und t ungeraden und alle übrigen Knoten geraden Grad haben.
- ii) Ein stark zusammenhängender Digraph besitzt genau dann einen Eulerkreis, wenn für jeden Knoten u der Eingangs- und der Ausgangsgrad übereinstimmen.

Beweis

- i) Da G im Fall $s \neq t$ genau dann eine Eulerlinie von s nach t hat, wenn der Graph $G' = (V \cup \{u_{neu}\}, E \cup \{\{t, u_{neu}\}, \{u_{neu}, s\}\})$ einen Eulerkreis hat, folgt dies aus dem vorigen Satz.
- ii) Dies folgt vollkommen analog zum ungerichteten Fall. □

Satz

HAMPATH, HAMCYCLE, DIHAMPATH und DIHAMCYCLE sind NP-vollständig.

Bemerkung

Bevor wir den Satz beweisen, betrachten wir ein weiteres Problem, das mit dem Hamiltonkreisproblem eng verwandt ist und große praktische Bedeutung hat.

Das Problem des Handlungsreisenden

- Gegeben sind die Entfernungen d_{ij} zwischen n Städten $i, j \in \{1, \dots, n\}$.
- Gesucht ist eine Rundreise (i_1, \dots, i_n) mit minimaler Länge $d_{i_1, i_2} + \dots + d_{i_{n-1}, i_n} + d_{i_n, i_1}$, die jede Stadt genau einmal besucht.
- Die Entscheidungsvariante dieses Optimierungsproblems ist wie folgt definiert.

Problem des Handlungsreisenden (TSP; *traveling salesman problem*)

Gegeben: Eine $n \times n$ Matrix $D = (d_{i,j}) \in \mathbb{N}^{n \times n}$ und eine Zahl k .

Gefragt: Existiert eine Permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$, so dass die Rundreise $(\pi(1), \dots, \pi(n))$ die Länge $\leq k$ hat?

Satz

3-SAT \leq^P DIHAMPATH \leq^P HAMPATH \leq^P HAMCYCLE \leq^P TSP.

Reduktion von HAMCYCLE auf TSP

- Sei $G = (V, E)$ ein Graph, wobei wir $V = \{1, \dots, n\}$ annehmen.
- Dann lässt sich G in Polynomialzeit auf die TSP Instanz (D, n) mit $D = (d_{i,j})$ und

$$d_{i,j} = \begin{cases} 1, & \text{falls } \{i,j\} \in E, \\ 2, & \text{sonst,} \end{cases}$$

transformieren.

- Diese Reduktion ist korrekt, da G genau dann einen Hamiltonkreis hat, wenn es in dem Distanzgraphen D eine Rundreise $(\pi(1), \dots, \pi(n))$ der Länge $L(\pi) \leq n$ gibt. □

Satz

$$3\text{-SAT} \leq^P \text{DIHAMPATH} \leq^P \text{HAMPATH} \leq^P \text{HAMCYCLE} \leq^P \text{TSP}.$$

Reduktion von HAMPATH auf HAMCYCLE

- Seien ein Graph $G = (V, E)$ und zwei Knoten $s, t \in V$ gegeben.
- Wir transformieren G in den Graphen $G' = (V', E')$ mit

$$V' = V \cup \{u_{\text{neu}}\} \text{ und}$$

$$E' = E \cup \left\{ \{t, u_{\text{neu}}\}, \{u_{\text{neu}}, s\} \right\}.$$

- Offenbar ist G' in Polynomialzeit aus G berechenbar und besitzt genau dann einen Hamiltonkreis, wenn G einen s - t -Hamiltonpfad besitzt. \square

Satz

$$3\text{-SAT} \leq^P \text{DIHAMPATH} \leq^P \text{HAMPATH} \leq^P \text{HAMCYCLE} \leq^P \text{TSP}.$$

Reduktion von HAMPATH auf HAMCYCLE

- Seien ein Graph $G = (V, E)$ und zwei Knoten $s, t \in V$ gegeben.
- Wir transformieren G in den Graphen $G' = (V', E')$ mit

$$V' = V \cup \{u_{\text{neu}}\} \text{ und}$$

$$E' = E \cup \left\{ \{t, u_{\text{neu}}\}, \{u_{\text{neu}}, s\} \right\}.$$

- Offenbar ist G' in Polynomialzeit aus G berechenbar und besitzt genau dann einen Hamiltonkreis, wenn G einen s - t -Hamiltonpfad besitzt. \square

Satz

$$3\text{-SAT} \leq^P \text{DIHAMPATH} \leq^P \text{HAMPATH} \leq^P \text{HAMCYCLE} \leq^P \text{TSP}.$$

Reduktion von HAMPATH auf HAMCYCLE

- Seien ein Graph $G = (V, E)$ und zwei Knoten $s, t \in V$ gegeben.
- Wir transformieren G in den Graphen $G' = (V', E')$ mit

$$V' = V \cup \{u_{\text{neu}}\} \text{ und}$$

$$E' = E \cup \left\{ \{t, u_{\text{neu}}\}, \{u_{\text{neu}}, s\} \right\}.$$

- Offenbar ist G' in Polynomialzeit aus G berechenbar und besitzt genau dann einen Hamiltonkreis, wenn G einen s - t -Hamiltonpfad besitzt. \square

Satz

$$3\text{-SAT} \leq^P \text{DIHAMPATH} \leq^P \text{HAMPATH} \leq^P \text{HAMCYCLE} \leq^P \text{TSP}.$$

Reduktion von HAMPATH auf HAMCYCLE

- Seien ein Graph $G = (V, E)$ und zwei Knoten $s, t \in V$ gegeben.
- Wir transformieren G in den Graphen $G' = (V', E')$ mit

$$V' = V \cup \{u_{\text{neu}}\} \text{ und}$$

$$E' = E \cup \left\{ \{t, u_{\text{neu}}\}, \{u_{\text{neu}}, s\} \right\}.$$

- Offenbar ist G' in Polynomialzeit aus G berechenbar und besitzt genau dann einen Hamiltonkreis, wenn G einen s - t -Hamiltonpfad besitzt. \square

Satz

$$3\text{-SAT} \leq^P \text{DIHAMPATH} \leq^P \text{HAMPATH} \leq^P \text{HAMCYCLE} \leq^P \text{TSP}.$$

Reduktion von HAMPATH auf HAMCYCLE

- Seien ein Graph $G = (V, E)$ und zwei Knoten $s, t \in V$ gegeben.
- Wir transformieren G in den Graphen $G' = (V', E')$ mit

$$V' = V \cup \{u_{\text{neu}}\} \text{ und}$$

$$E' = E \cup \left\{ \{t, u_{\text{neu}}\}, \{u_{\text{neu}}, s\} \right\}.$$

- Offenbar ist G' in Polynomialzeit aus G berechenbar und besitzt genau dann einen Hamiltonkreis, wenn G einen s - t -Hamiltonpfad besitzt. \square

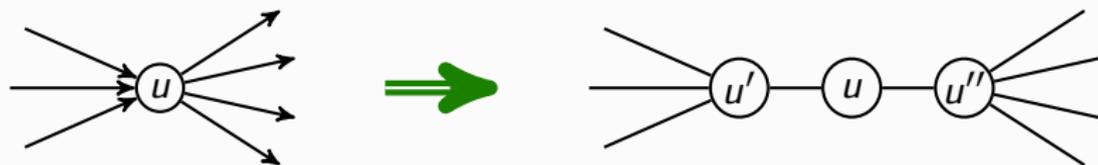
Hamiltonkreise, Hamiltonpfade und TSP

Satz

$$3\text{-SAT} \leq^P \text{DIHAMPATH} \leq^P \text{HAMPATH} \leq^P \text{HAMCYCLE} \leq^P \text{TSP}.$$

Reduktion von DIHAMPATH auf HAMPATH

- Seien ein Digraph $G = (V, E)$ und zwei Knoten $s, t \in V$ gegeben.
- Zuerst entfernen wir alle Kanten, die von t ausgehen oder in s enden.
- Dann konstruieren wir den Graphen G' , indem wir lokal für jeden Knoten $u \in V$ die folgende Ersetzung durchführen:



- Hierbei lassen wir u' (bzw. u'') weg, falls keine Kanten in u enden (bzw. von u ausgehen).
- Dann ist die Funktion $G \mapsto G'$ in FP berechenbar und G enthält genau dann einen Hamiltonpfad von s nach t , wenn dies auf G' zutrifft. \square

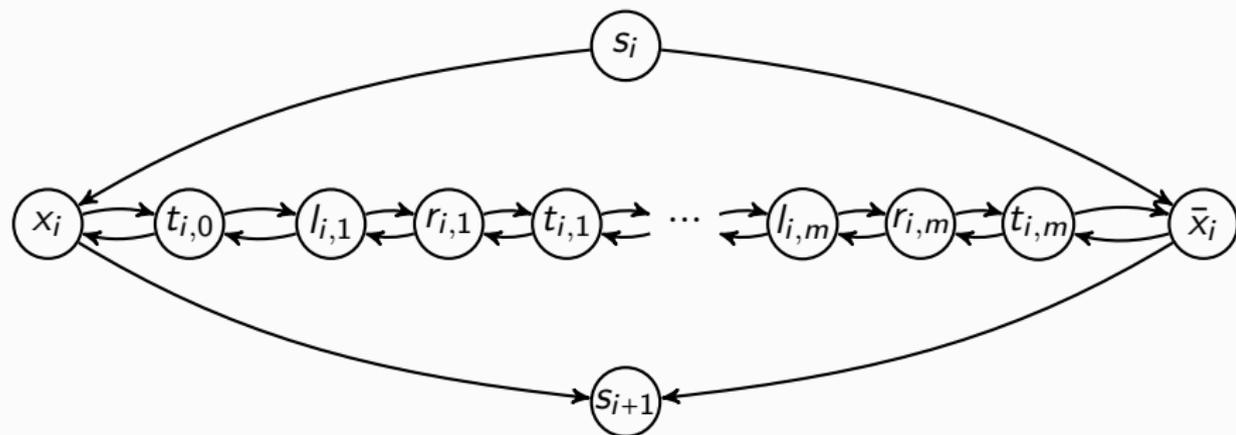
Satz

$$3\text{-SAT} \leq^P \text{DIHAMPATH} \leq^P \text{HAMPATH} \leq^P \text{HAMCYCLE} \leq^P \text{TSP}.$$

Reduktion von 3-SAT auf DIHAMPATH

- Sei $F = \{C_1, \dots, C_m\}$ mit $C_j = \{l_{j1}, \dots, l_{jk_j}\}$ für $j = 1, \dots, m$ eine 3-KNF-Formel über den Variablen x_1, \dots, x_n .
- Wir transformieren F in Polynomialzeit in einen gerichteten Graphen $G_F = (V, E)$ mit zwei ausgezeichneten Knoten s und t , der genau dann einen hamiltonschen s - t -Pfad besitzt, wenn F erfüllbar ist.
- Jede Klausel C_j repräsentieren wir durch einen Knoten c_j und jede Variable x_i repräsentieren wir durch folgenden Graphen X_i .

- Jede Klausel C_j repräsentieren wir durch einen Knoten c_j und jede Variable x_i repräsentieren wir durch folgenden Graphen X_i :



- Ein Pfad von s_1 nach s_{n+1} kann ausgehend von s_i ($i = 1, \dots, n$) entweder zuerst den Knoten x_i oder zuerst den Knoten \bar{x}_i besuchen.
- Daher können wir jedem s_1 - s_{n+1} -Pfad P eine Belegung $b_P = b_1 \dots b_n$ zuordnen mit $b_i = 1$ gdw. P den Knoten x_i vor dem Knoten \bar{x}_i besucht.

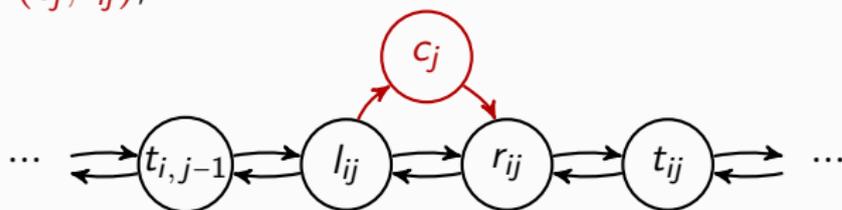
Reduktion von 3-SAT auf DIHAMPATH

- Die Knotenmenge V von G_F ist also

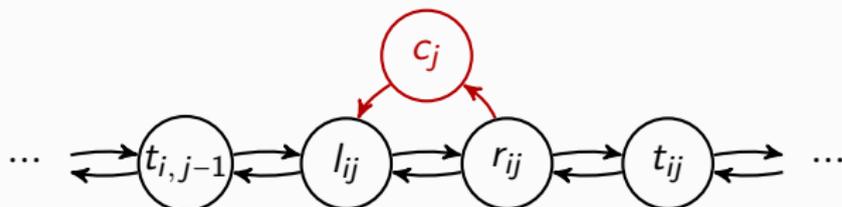
$$V = \{c_1, \dots, c_m\} \cup \{s_1, \dots, s_{n+1}\} \cup \{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n\} \cup \bigcup_{i=1}^n \{t_{i,0}, l_{i,1}, r_{i,1}, t_{i,1}, \dots, l_{i,m}, r_{i,m}, t_{i,m}\}.$$

- Dabei haben die Graphen X_{i-1} und X_i den Knoten s_i gemeinsam.
- Als Startknoten wählen wir $s = s_1$ und als Zielknoten $t = s_{n+1}$.
- Jetzt fehlen nur noch die Verbindungskanten zwischen den Teilgraphen X_i und den Klauselknoten c_j .
- Diese Kanten sollen einem s - t -Pfad P genau dann einen Abstecher nach c_j ermöglichen, wenn die Belegung b_P die Klausel C_j erfüllt.

- Für jedes Literal $l \in C_j$ fügen wir zu E im Fall $l = x_i$ die beiden Kanten (l_{ij}, c_j) und (c_j, r_{ij}) ,



und im Fall $l = \bar{x}_i$ die Kanten (r_{ij}, c_j) und (c_j, l_{ij}) hinzu:



- Man beachte, dass ein s - t -Pfad P über diese Kanten genau dann einen Abstecher zu c_j machen kann, wenn die Belegung b_P das Literal l wahr macht.

Reduktion von 3-SAT auf DIHAMPATH

- Zunächst ist klar, dass die Reduktionsfunktion $F \mapsto (G_F, s, t)$ in Polynomialzeit berechenbar ist.
- Es bleibt also zu zeigen, dass F genau dann erfüllbar ist, wenn in G_F ein Hamiltonpfad von $s = s_1$ nach $t = s_{n+1}$ existiert.
- Falls $F(b) = 1$ ist, so erhalten wir einen Hamiltonpfad, indem wir in jeder Klausel C_j ein wahres Literal $l = x_i$ bzw. $l = \bar{x}_i$ auswählen und in den zu b gehörigen s - t -Pfad P einen „Abstecher“ vom Knotenpaar l_{ij}, r_{ij} zum Klauselknoten c_j einbauen.
- Ist umgekehrt P ein s - t -Hamiltonpfad in G_F , so müssen der Vorgänger- und Nachfolgerknoten jedes Klauselknotens c_j ein Paar l_{ij}, r_{ij} bilden, da P andernfalls nicht beide Pufferknoten $t_{i,j-1}$ und $t_{i,j}$ besuchen kann.
- Da aber P alle Klauselknoten besucht und ausgehend von dem Paar l_{ij}, r_{ij} nur dann ein Abstecher zu c_j möglich ist, wenn die Belegung b_P die Klausel C_j erfüllt, folgt $F(b_P) = 1$. □

Beispiel

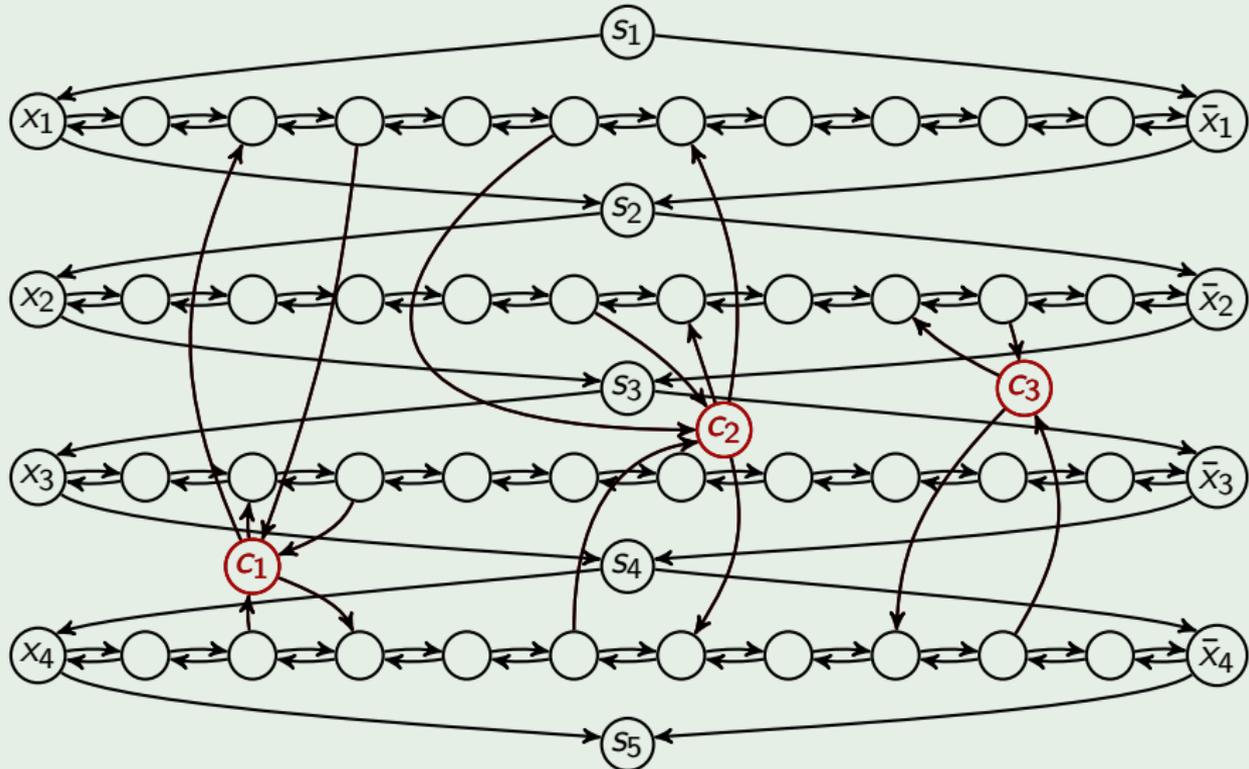
Die 3-SAT-Instanz

$$F = (\bar{x}_1 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_4).$$

lässt sich auf folgenden Digraphen G mit Startknoten $s = s_1$ und Zielknoten $t = s_5$ reduzieren:

Reduktion von 3-SAT auf DIHAMPATH

$$F = (\bar{x}_1 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_4).$$



Das Rucksack-Problem

- Wie schwierig ist es, einen Rucksack der Größe w mit einer Auswahl aus k Gegenständen der Größe u_1, \dots, u_k möglichst voll zu packen?
- Dieses Optimierungsproblem lässt sich leicht auf folgendes Entscheidungsproblem reduzieren.

RUCKSACK:

Gegeben: Eine Folge (u_1, \dots, u_k, v, w) von natürlichen Zahlen.

Gefragt: Ex. eine Auswahl $S \subseteq \{1, \dots, k\}$ mit $v \leq \sum_{i \in S} u_i \leq w$?

Beim SubsetSum-Problem möchte man dagegen nur wissen, ob der Rucksack randvoll gepackt werden kann.

SUBSETSUM:

Gegeben: Eine Folge (u_1, \dots, u_k, w) von natürlichen Zahlen.

Gefragt: Ex. eine Auswahl $S \subseteq \{1, \dots, k\}$ mit $\sum_{i \in S} u_i = w$?

Rucksack und SubsetSum sind NP-vollständig

Satz

RUCKSACK und SUBSETSUM sind NP-vollständig.

Beweis

- Es ist klar, dass beide Probleme in NP enthalten sind.
- Zum Nachweis der NP-Härte zeigen wir die folgenden Reduktionen:

$$3\text{-SAT} \leq^P \text{SUBSETSUM} \leq^P \text{RUCKSACK}.$$

Reduktion von SUBSETSUM auf RUCKSACK

Da SUBSETSUM einen Spezialfall des RUCKSACK-Problems darstellt, lässt es sich leicht darauf reduzieren:

$$(u_1, \dots, u_k, w) \mapsto (u_1, \dots, u_k, w, w).$$

□

SubsetSum ist NP-vollständig

Reduktion von 3-SAT auf SUBSETSUM

- Sei $F = \{C_1, \dots, C_m\}$ mit $C_j = \{l_{j1}, \dots, l_{jk_j}\}$ für $j = 1, \dots, m$ eine 3-KNF-Formel über den Variablen x_1, \dots, x_n .
- Betrachte die Reduktionsfunktion

$$f : F \mapsto (u_1, \dots, u_n, u'_1, \dots, u'_n, v_1, \dots, v_m, v'_1, \dots, v'_m, w),$$

mit folgenden Dezimalzahlen:

$$u_i = b_{i1} \dots b_{im} 0^{i-1} 10^{n-i}$$

$$v_j = v'_j = 0^{j-1} 10^{m-j-1} 0^n$$

$$u'_i = b'_{i1} \dots b'_{im} 0^{i-1} 10^{n-i}$$

$$w = \underbrace{3 \dots 3}_{m\text{-mal}} \underbrace{1 \dots 1}_{n\text{-mal}},$$

wobei

$$b_{ij} = \begin{cases} 1 & x_i \in C_j \\ 0 & \text{sonst} \end{cases} \quad \text{und} \quad b'_{ij} = \begin{cases} 1 & \bar{x}_i \in C_j \\ 0, & \text{sonst} \end{cases} \quad \text{ist.}$$

- Hierbei können führende Nullen natürlich auch weggelassen werden.

SubsetSum ist NP-vollständig

Beispiel

- Betrachte die 3-KNF Formel

$$F = (\bar{x}_1 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_4).$$

- Die zu F gehörige SUBSETSUM-Instanz $f(F)$ ist

$$(u_1, u_2, u_3, u_4, u'_1, u'_2, u'_3, u'_4, v_1, v_2, v_3, v'_1, v'_2, v'_3, w)$$

mit

$$u_1 = 010\ 1000 \quad u_2 = 010\ 0100 \quad u_3 = 000\ 0010 \quad u_4 = 110\ 0001$$

$$u'_1 = 100\ 1000 \quad u'_2 = 001\ 0100 \quad u'_3 = 100\ 0010 \quad u'_4 = 001\ 0001$$

und

$$v_1 = 100\ 0000 \quad v_2 = 010\ 0000 \quad v_3 = 001\ 0000$$

$$v'_1 = 100\ 0000 \quad v'_2 = 010\ 0000 \quad v'_3 = 001\ 0000$$

sowie

$$w = 333\ 1111.$$

- Der erfüllenden Belegung $a = 0100$ entspricht dann die Auswahl $(u'_1, u_2, u'_3, u'_4, v_1, v_2, v'_2, v_3, v'_3)$.



SubsetSum ist NP-vollständig

Beweis von $F \in 3\text{-SAT} \Rightarrow f(F) \in \text{SUBSETSUM}$

- Sei $a = a_1 \cdots a_n$ eine erfüllende Belegung für F .
- Da a in jeder Klausel mindestens ein und höchstens drei Literale wahr macht, hat die Zahl

$$\sum_{a_j=1} u_j + \sum_{a_j=0} u'_j$$

eine Dezimaldarstellung der Form $b_1 \cdots b_m 1 \cdots 1$ mit $1 \leq b_j \leq 3$ für $j = 1, \dots, m$.

- Durch Addition von

$$\sum_{b_j \leq 2} v_j + \sum_{b_j = 1} v'_j$$

erhalten wir w .

SubsetSum ist NP-vollständig

Beweis von $f(F) \in \text{SUBSETSUM} \Rightarrow F \in 3\text{-SAT}$

- Sei $S = P \cup N \cup I \cup J$ eine Auswahlmenge für $f(F)$ mit

$$\sum_{i \in P} u_i + \sum_{i \in N} u'_i + \sum_{j \in I} v_j + \sum_{j \in J} v'_j = \underbrace{3 \cdots 3}_{m\text{-mal}} \underbrace{1 \cdots 1}_{n\text{-mal}}.$$

- Da die Teilsumme $\sum_{j \in I} v_j + \sum_{j \in J} v'_j$ die Form $c_1 \cdots c_m 0 \cdots 0$ mit $c_j \leq 2$ hat, muss die Teilsumme $\sum_{i \in P} u_i + \sum_{i \in N} u'_i$ die Form $b_1 \cdots b_m 1 \cdots 1$ mit $b_j \geq 1$ haben.
- Da keine Überträge auftreten, muss also $P = \{1, \dots, n\} - N$ sein, und jede Klausel C_j muss mindestens ein Literal aus der Menge $\{x_i \mid i \in P\} \cup \{\bar{x}_i \mid i \in N\}$ enthalten.
- Folglich erfüllt folgende Belegung $a_1 \cdots a_n$ die Formel F :

$$a_i = \begin{cases} 1, & i \in P, \\ 0, & i \in N. \end{cases}$$

- Damit haben wir die Korrektheit von f gezeigt. □