Příklady důkazů NP-úplnosti

NP-úplnost problému SAT

Připomeňme problém SAT:

SAT (splnitelnost booleovských formulí)

Vstup: Booleovská formule φ.

Otázka: Je φ splnitelná?

Ukázat, že SAT patří do třídy NPTIME je snadné:

Nedeterministický algoritmus řešící SAT v polynomiálním čase pracuje následovně:

- Nedeterministicky zvolí ohodnocení ν, které přiřazuje booleovskou hodnotu každé proměnné vyskytující se ve formuli φ.
- Vyhodnotí φ při ohodnocení ν, tj. spočítá hodnotu $[\varphi]_\nu$.
- Pokud $[\varphi]_\nu = 1$, vrátí algoritmus odpověď ANO.
  Jinak vrátí odpověď NE.

NP-úplnost problému SAT

Ukázat, že problém SAT je NP-těžký je složitější.

Je třeba ukázat, že pro libovolný problém $P \in \mathrm{NPTIME}$ existuje polynomiální redukce z problému P na problém SAT,
tj. ukázat, že existuje algoritmus, který:

- dostane na svůj vstup (libovolnou) instanci problému P,
- k této instanci vyrobí booleovskou formuli φ takovou, že φ bude splnitelná právě tehdy, když pro danou instanci problému P bude odpověď ANO,
- bude mít polynomiální časovou složitost.

NP-úplnost problému SAT

Jestliže $P \in \text{NPTIME}$, musí existovat **nedeterministický** Turingův stroj $\mathcal{M}$ a polynom $p(n)$ takový, že:

- Pro libovolnou instanci w problému P (reprezentovanou slovem v nějaké abecedě Σ) platí, že:
  - Pokud je odpověď pro w ANO, pak existuje alespoň jeden výpočet stroje $\mathcal{M}$ nad slovem w, při kterém stroj $\mathcal{M}$ vydá odpověď ANO.
  - Pokud je odpověď pro w NE, pak všechny výpočty stroje $\mathcal{M}$ nad slovem w skončí s odpovědí NE.
- Stroj $\mathcal{M}$ provede při libovolném výpočtu nad slovem w nejvýše $p(|w|)$ kroků.

NP-úplnost problému SAT

Ukážeme, jak pro daný nedeterministický Turingův stroj $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F)$, polynom $p(n)$ a slovo $w \in \Sigma^*$ vyrobit booleovskou formuli φ takovou, že:

- φ bude splnitelná právě tehdy, když existuje výpočet stroje $\mathcal{M}$ nad slovem w, při kterém $\mathcal{M}$ udělá nejvýše $p(|w|)$ kroků a vydá odpověď Ano.
- Formuli φ bude možné vyrobit algoritmem v čase polynomiálním vzhledem k délce slova w.

NP-úplnost problému SAT

Připomeňme, že v definici Turingova stroje $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F)$:

- Q – množina stavů řídící jednotky
- Σ – vstupní abeceda $(\Sigma \subseteq \Gamma)$
- Γ – pásková abeceda
- $\delta : (Q - F) \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{-1, 0, +1\})$ – přechodová funkce
- $q_0 \in Q$ – počáteční stav řídící jednotky
- $F \subseteq Q$ – množina koncových stavů

Dále předpokládáme, že Γ obsahuje speciální prázdný symbol $\square$ (blank), přičemž $\square \notin \Sigma$.

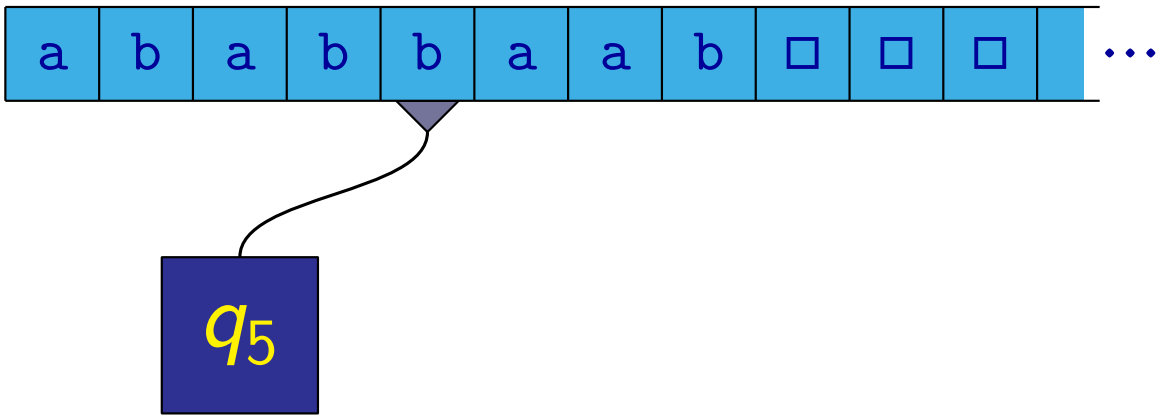
NP-úplnost problému SAT

Vzhledem k tomu, že se v následující konstrukci budeme zabývat pouze stroji, které řeší rozhodovací problémy, budeme předpokládat, že

$$F = \{q_{acc}, q_{rej}\}$$

- Pokud stroj skončí ve stavu q_{acc}, znamená to, že vydal odpověď ANO.
- Pokud stroj skončí ve stavu q_{rej}, znamená to, že vydal odpověď NE.

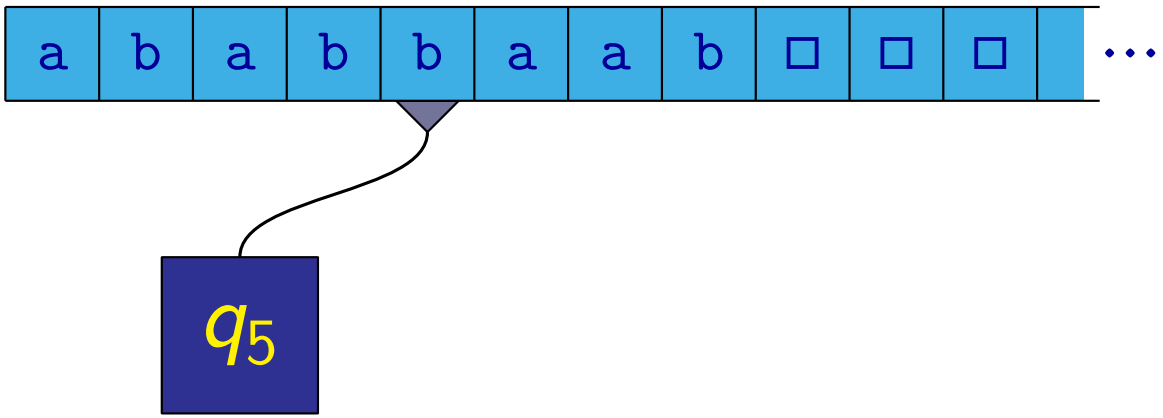
NP-úplnost problému SAT



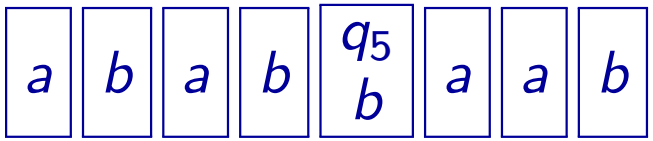
Konfigurace Turingova stroje je dána:

- stavem řídící jednotky
- obsahem pásky
- pozicí hlavy

NP-úplnost problému SAT

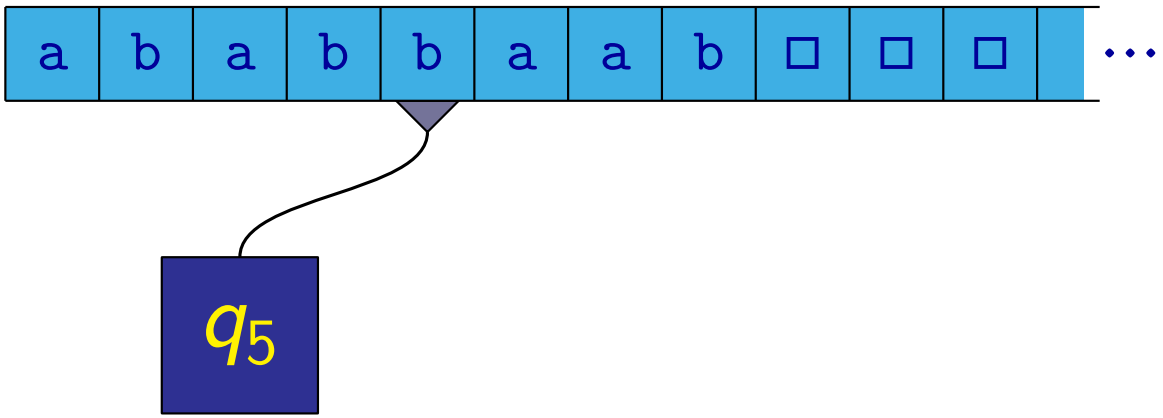


Konfiguraci můžeme zapsat jako slovo v abecedě $\Gamma \cup (Q \times \Gamma)$:

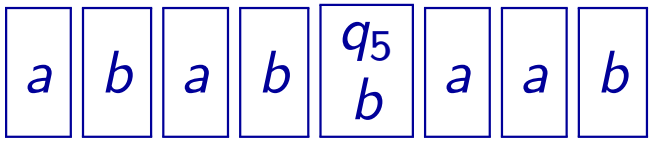


Toto slovo, vždy obsahuje právě jeden znak z $(Q \times \Gamma)$, který vyznačuje stav řídící jednotky i pozici hlavy.

NP-úplnost problému SAT

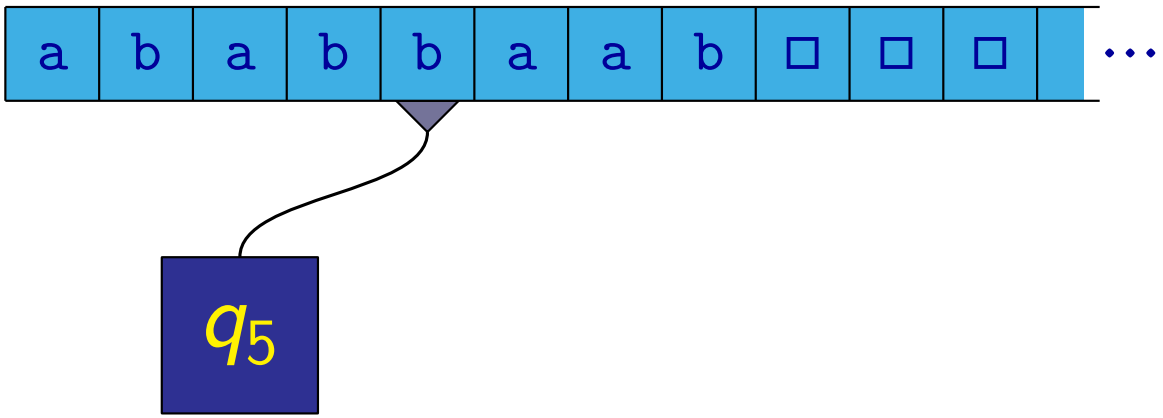


Konfiguraci můžeme zapsat jako slovo v abecedě $\Gamma \cup (Q \times \Gamma)$:

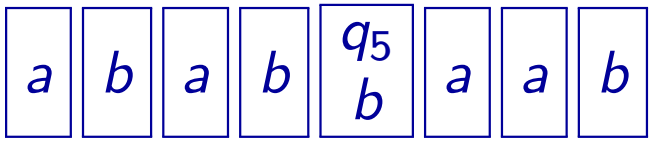


Poznámka: Znaky z $(Q \times \Gamma)$ píšeme jako $\boxed{\begin{matrix} q \\ a \end{matrix}}$ místo $\boxed{(q, a)}$.

NP-úplnost problému SAT

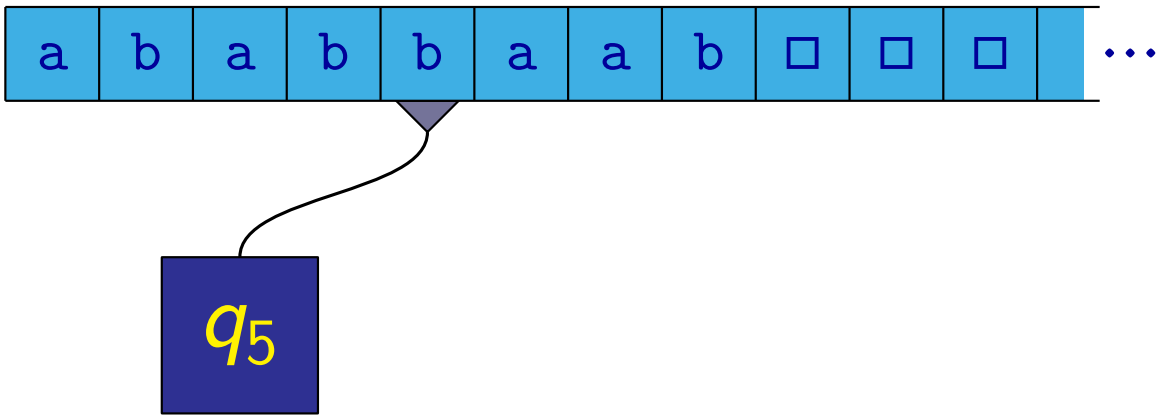


Konfiguraci můžeme zapsat jako slovo v abecedě $\Gamma \cup (Q \times \Gamma)$:

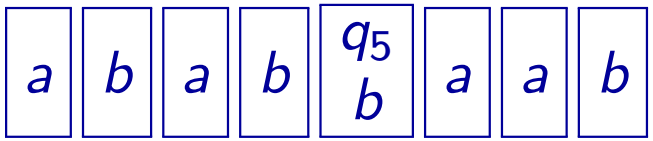


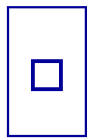
Ostatní symboly (z Γ) reprezentují obsah pásky.

NP-úplnost problému SAT



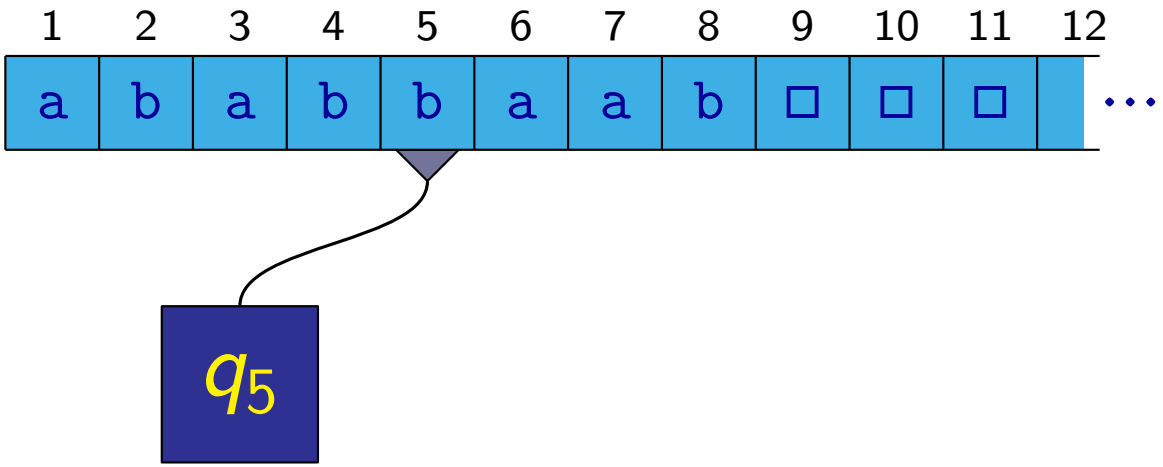
Konfiguraci můžeme zapsat jako slovo v abecedě $\Gamma \cup (Q \times \Gamma)$:



Políčka pásky, která nejsou ve slově vyznačena, obsahují symbol 

.

NP-úplnost problému SAT



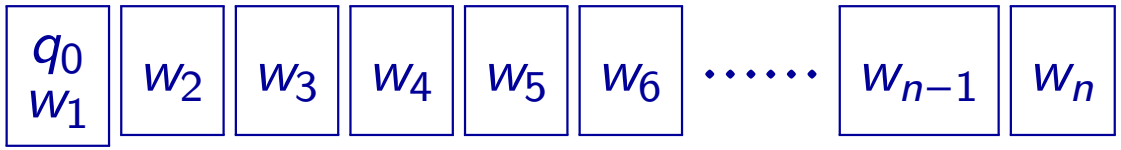
Budeme předpokládat, že Turingův stroj používá jednostranně omezenou pásku.

Políčka pásky si můžeme očíslovat čísly $1, 2, 3, \ldots$.

NP-úplnost problému SAT

Výpočet Turingova stroje je posloupnost konfigurací, kde:

- První konfigurace je **počáteční konfigurace**



  kde $w_1 w_2 \ldots w_n$ jsou jednotlivé symboly slova w, které je vstupem Turingova stroje $\mathcal{M}$.

- Každá další konfigurace v posloupnosti je konfigurace, do které se může stroj dostat z předchozí konfigurace provedením jednoho kroku podle přechodové funkce δ.

- Pokud je výpočet konečný, v poslední konfiguraci musí být řídící jednotka v některém koncovém stavu z množiny F.

NP-úplnost problému SAT

Předpokládejme nyní, že časová složitost stroje $\mathcal{M}$ je shora omezena nějakou funkcí $p(n)$.

(Bez ztráty na obecnosti budeme předpokládat, že pro všechna n je $p(n) \geq n$.)

Pokud stroj $\mathcal{M}$ dostane jako vstup slovo w délky n, provede během výpočtu nanejvýš $p(n)$ kroků.

Protože stroj začíná s hlavou na políčku číslo 1, může se během toho výpočtu dostat hlava nanejvýš na políčko číslo $p(n) + 1$ (v každém kroku se posune nanejvýš o jedno políčko).

NP-úplnost problému SAT

Pokud je tedy časová složitost stroje $\mathcal{M}$ shora omezena funkcí $p(n)$, můžeme všechny konfigurace ve výpočtu nad vstupem velikosti n zapsat jako slova délky $p(n) + 1$

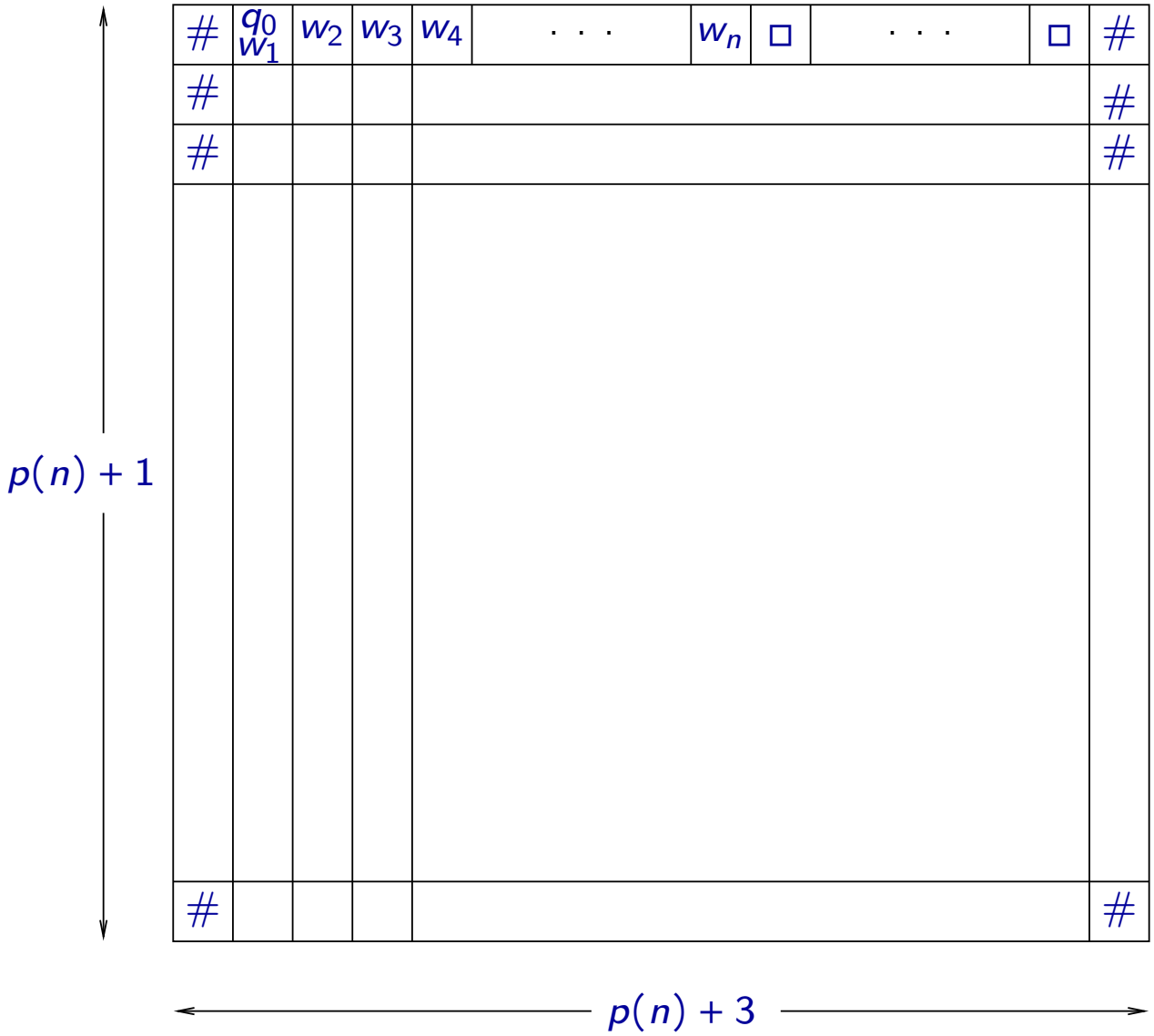
Na políčka s čísly většími než $p(n) + 1$ se stroj během výpočtu hlavou nedostane a tato políčka budou obsahovat symbol $\square$ (připomeňme, že předpokládáme $p(n) \geq n$).

NP-úplnost problému SAT

Slova popisující jednotlivé konfigurace ve výpočtu stroje $\mathcal{M}$ nad slovem $w = w_1 w_2 \cdots w_n$ tedy můžeme zapsat pod sebe do tabulky, kde:

- Řádky odpovídají jednotlivým konfiguracím (zapsaným jako slova v abecedě $\Gamma \cup (Q \times \Gamma)$).
- Sloupce odpovídají políčkům pásky s čísly $1, 2, \ldots, p(n) + 1$.
- Z technických důvodů přidáme ještě zleva a zprava sloupce obsahující jen speciální oddělovací znaky $\#$ (přičemž $\# \notin Q \cup \Gamma$).

NP-úplnost problému SAT



NP-úplnost problému SAT

- Jednotlivá políčka tabulky tedy budou obsahovat symboly z abecedy
$$\Delta = \Gamma \cup (Q \times \Gamma) \cup \{\#\}$$
- Řádky tabulky budou mít čísla 0 až $p(n) + 1$.
(Řádek 0 bude obsahovat počáteční konfiguraci).
- Sloupce budou mít čísla 0 až $p(n) + 2$.
(Sloupce 0 a $p(n) + 2$ budou obsahovat znaky $\#$.)

NP-úplnost problému SAT

Poznámka: Výpočet může být kratší než $p(n)$ kroků a v takovém případě by nebyla vyplněna celá tabulka.

Abychom i v tomto případě vyplnili celou tabulku, můžeme udělat to, že poslední konfiguraci, ve které se výpočet zastavil, zopakujeme ve všech zbylých řádcích tabulky.

NP-úplnost problému SAT

Pokud tedy máme dán (nedeterministický) Turingův stroj $\mathcal{M}$ s časovou složitostí omezenou shora polynomem $p(n)$ řešící problém P a instanci tohoto problému zapsanou jako slovo w, je pro tuto instanci odpověď ANO právě tehdy, když existuje nějaký přijímající výpočet stroje $\mathcal{M}$ nad slovem w.

Takový přijímající výpočet můžeme výše popsaným způsobem zapsat do tabulky o $p(n) + 1$ řádcích a $p(n) + 3$ sloupcích.

Poznámka: Všimněte si, že velikost tabulky je polynomiální vzhledem k n.

NP-úplnost problému SAT

Pro daný stroj $\mathcal{M}$, polynom $p(n)$ a slovo w vytvoříme formuli φ takovou, že:

- Různá ohodnocení ν booleovských proměnných ve formuli φ budou reprezentovat všechny možné (i nesmyslné) obsahy dané tabulky.
- $[\varphi]_\nu = 1$ bude platit právě pro ta ohodnocení ν, která reprezentují takový obsah tabulky, který je zápisem přijímajícího výpočtu stroje $\mathcal{M}$ nad vstupem w.

Formule φ tedy bude splnitelná právě tehdy, pokud bude existovat přijímající výpočet stroje $\mathcal{M}$ nad vstupem w.

NP-úplnost problému SAT

Formule φ bude složena pomocí booleovských spojek z atomických tvrzení typu:

$$\text{„}\textit{Políčko } (i,j) \textit{ obsahuje symbol } a.\text{“}$$

kde i, j, a budou konkrétní konstanty, například:

$$\text{„}\textit{Políčko } (9,4) \textit{ obsahuje symbol } \boxed{\begin{matrix} q_5 \\ b \end{matrix}}.\text{“}$$

Poznámka: Když mluvíme o políčku (i,j), máme tím na mysli políčko v i-tém řádku a j-tém sloupci tabulky.

NP-úplnost problému SAT

Formule φ tedy bude obsahovat booleovské proměnné $x_{i,j}^{a}$, kde:

- $0 \leq i \leq p(n) + 1$
- $0 \leq j \leq p(n) + 2$
- $a \in \Delta$ (kde $\Delta = \Gamma \cup (Q \times \Gamma) \cup \{\#\}$)

s tím, že $\nu(x_{i,j}^{a}) = 1$ znamená, že ν reprezentuje obsah tabulky, při kterém políčko (i, j) obsahuje symbol a,

a $\nu(x_{i,j}^{a}) = 0$ znamená, že ν reprezentuje obsah tabulky, při kterém políčko (i, j) neobsahuje symbol a.

NP-úplnost problému SAT

Příklad: Proměnná $x_{9,4}^{q_5,\mathtt{b}}$ reprezentuje tvrzení:

„*Políčko* $(9,4)$ *obsahuje symbol* $\boxed{\begin{matrix} q_5 \\ \mathtt{b} \end{matrix}}$.“

Poznámka: U hodnot z $(Q \times \Gamma)$ budeme v indexech raději používat zápis q, a místo $\boxed{\begin{matrix} q \\ a \end{matrix}}$.

Pokud tedy $\nu(x_{9,4}^{q_5,\mathtt{b}}) = 1$, znamená to, že při obsahu tabulky reprezentovaném ν políčko $(9,4)$ obsahuje symbol $\boxed{\begin{matrix} q_5 \\ \mathtt{b} \end{matrix}}$,

a pokud $\nu(x_{9,4}^{q_5,\mathtt{b}}) = 0$, tak při ν políčko $(9,4)$ neobsahuje $\boxed{\begin{matrix} q_5 \\ \mathtt{b} \end{matrix}}$.

NP-úplnost problému SAT

Celá formule φ, která jako celek bude říkat:

Tabulka obsahuje přijímající výpočet stroje $\mathcal{M}$ nad slovem w,

bude složena z mnoha podformulí, kde každá z těchto podformulí bude formulovat nějakou jednoduchou podmínku, která musí být splněna, aby obsah tabulky byl přijímajícím výpočtem stroje $\mathcal{M}$ nad slovem w.

Tyto podformule budou spojeny pomocí konjunkce.

Pokud tedy bude při daném ohodnocení ν některá z těchto podmínek porušena, bude celá formule φ při ν nabývat hodnoty 0, tj. $[\varphi]_\nu = 0$.

NP-úplnost problému SAT

V následujícím výkladu si popíšeme tyto jednotlivé podformule.

Pokud v dalším výkladu o nějaké formuli ψ řekneme, že

„ψ *přidáme do* φ,“

máme tím na mysli, že ψ spojíme pomocí konjunkce ($\wedge$) s dosud vytvořenou částí formule φ.

NP-úplnost problému SAT

Aby tabulka obsahovala přijímající výpočet stroje $\mathcal{M}$ nad vstupem w, musí být splněny následující podmínky:

1. Každé políčko tabulky obsahuje právě jeden symbol z Δ.
2. Řádek číslo 0 obsahuje počáteční konfiguraci se slovem w.
3. Každý řádek tabulky (kromě řádku 0) obsahuje buď:
   - konfiguraci, která je jedním krokem (podle přechodové funkce δ) dosažitelná z konfigurace zapsané v předchozím řádku, nebo
   - koncovou konfiguraci shodnou s konfigurací v předchozím řádku.
4. Poslední řádek tabulky obsahuje konfiguraci se stavem q_{acc}.

NP-úplnost problému SAT

Je zřejmé, že pokud tabulka obsahuje přijímající výpočet, tak jsou tyto čtyři podmínky splněny.

Na druhou stranu je rovněž zřejmé, že pokud budou tyto čtyři podmínky splněny, tak tabulka opravdu obsahuje přijímající výpočet stroje $\mathcal{M}$ nad slovem w.

NP-úplnost problému SAT

Podívejme se nejprve na první podmínku:

Každé políčko tabulky obsahuje právě jeden symbol z Δ.

Tu zajistíme tak, že pro každé políčko (i, j) přidáme do φ podformuli, která bude říkat:

Políčko (i, j) *obsahuje právě jeden symbol z* Δ,

což můžeme formulovat též takto:

Právě jedna z proměnných $x_{i,j}^{a_1}, x_{i,j}^{a_2}, \ldots, x_{i,j}^{a_k}$ *má hodnotu* 1,

kde $\{a_1, a_2, \ldots, a_k\}$ je množina všech symbolů z Δ.

NP-úplnost problému SAT

Vyjádřit tvrzení, že právě jedna z proměnných $x_1, x_2, \ldots, x_k$ má hodnotu 1 (kde $x_1, x_2, \ldots, x_k$ jsou nějaké libovolné booleovské proměnné) není složité.

Ukážeme si to na příkladě, kde pro jednoduchost budeme mít jen čtyři booleovské proměnné A, B, C, D:

$$\begin{array}{l} (\ \ A \wedge \neg B \wedge \neg C \wedge \neg D) \ \vee \\ (\neg A \wedge \ \ B \wedge \neg C \wedge \neg D) \ \vee \\ (\neg A \wedge \neg B \wedge \ \ C \wedge \neg D) \ \vee \\ (\neg A \wedge \neg B \wedge \neg C \wedge \ \ D) \ \vee \end{array}$$

Není těžké ověřit, že tato formule nabývá hodnoty 1 právě při těch ohodnoceních, při kterých má právě jedna z proměnných A, B, C, D hodnotu 1.

NP-úplnost problému SAT

Obecně pro množinu proměnných $X = \{x_1, x_2, \ldots, x_k\}$ můžeme tuto podmínku zapsat následovně:

$$\bigvee_{x_i \in X} \left(x_i \ \wedge \bigwedge_{x_j \in X - \{x_i\}} \neg x_j \right)$$

Všimněme si, že pro k proměnných má tato formule velikost $\mathcal{O}(k^2)$.

V našem případě je $k = |\Delta|$, takže velikost podformule, kterou přidáváme pro každé políčko, je $\mathcal{O}(|\Delta|^2)$ a je to tedy konstanta nezávislá na velikosti vstupu w.

Poznámka: Existuje způsob, jak zapsat výše uvedenou podmínku tak, aby velikost vytvořené formule byla $\mathcal{O}(k \log k)$, ale my to zde nepotřebujeme.

NP-úplnost problému SAT

Další podmínkou, která musí být splněna, je:

Řádek 0 obsahuje počáteční konfiguraci se slovem w.

Pokud tedy $w_1 w_2 \dots w_n$ jsou jednotlivé symboly slova w, přičemž $n \geq 1$, musí platit následující:

- Políčko $(0, 1)$ obsahuje symbol (q_0, w_1), kde q_0 je počáteční stav.
- Políčka $(0, 2)$, $(0, 3)$, $\dots$, $(0, n)$ obsahují symboly $w_2, w_3, \dots, w_n$.
- Políčka $(0, n+1)$, $(0, n+2)$, $\dots$, $(0, p(n+1))$ obsahují symboly $\square$.
- Políčka $(0, 0)$ a $(0, p(n)+2)$ obsahují symboly $\#$.

NP-úplnost problému SAT

Tuto podmínku tedy můžeme zapsat následující formulí, kterou přidáme do φ:

$$x_{0,1}^{q_0,w_1} \wedge \left(\bigwedge_{i=2}^{n} x_{0,i}^{w_i} \right) \wedge \left(\bigwedge_{i=n+1}^{p(n)+1} x_{0,i}^{\square} \right) \wedge x_{0,0}^{\#} \wedge x_{0,p(n)+2}^{\#}$$

Velikost této formule je $\mathcal{O}(p(n))$.

Poznámka: Případ, kdy $n = 0$, by se lišil jen v tom, že místo $x_{0,1}^{q_0,w_1}$ by formule obsahovala $x_{0,1}^{q_0,\square}$.

NP-úplnost problému SAT

Asi nejsložitější je zajištění třetí podmínky:

> *Každý řádek (kromě řádku 0) obsahuje konfiguraci, která vznikne z předchozí provedením jednoho kroku (a nebo je kopií předchozí koncové konfigurace).*

Vezměme si nějaké dvě po sobě jdoucí konfigurace.

Tyto dvě konfigurace se vždy liší nanejvýš na dvou pozicích:

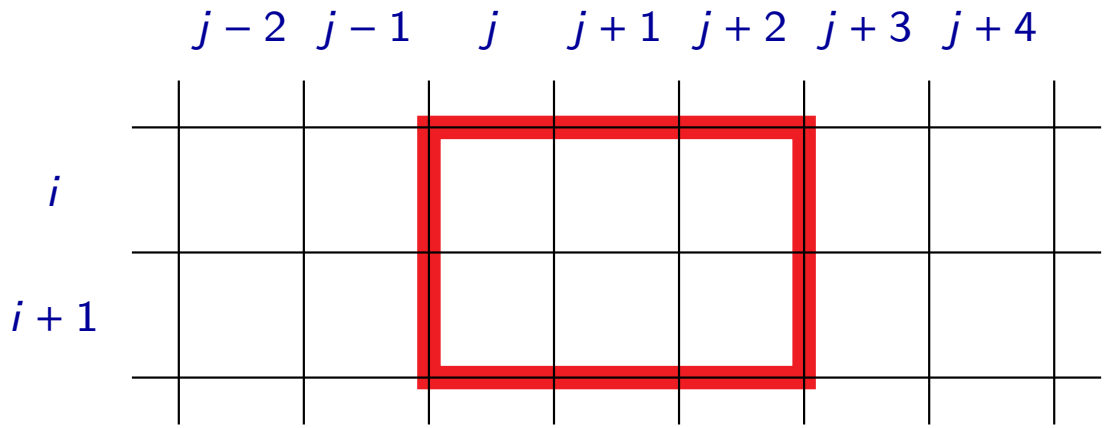
- na pozici, kde se v první z nich nachází hlava,
- a na jedné z pozic, které s ní bezprostředně sousedí.

Obsahy dvojic řádků i a $i + 1$ v tabulce jsou tedy velice úzce svázány.

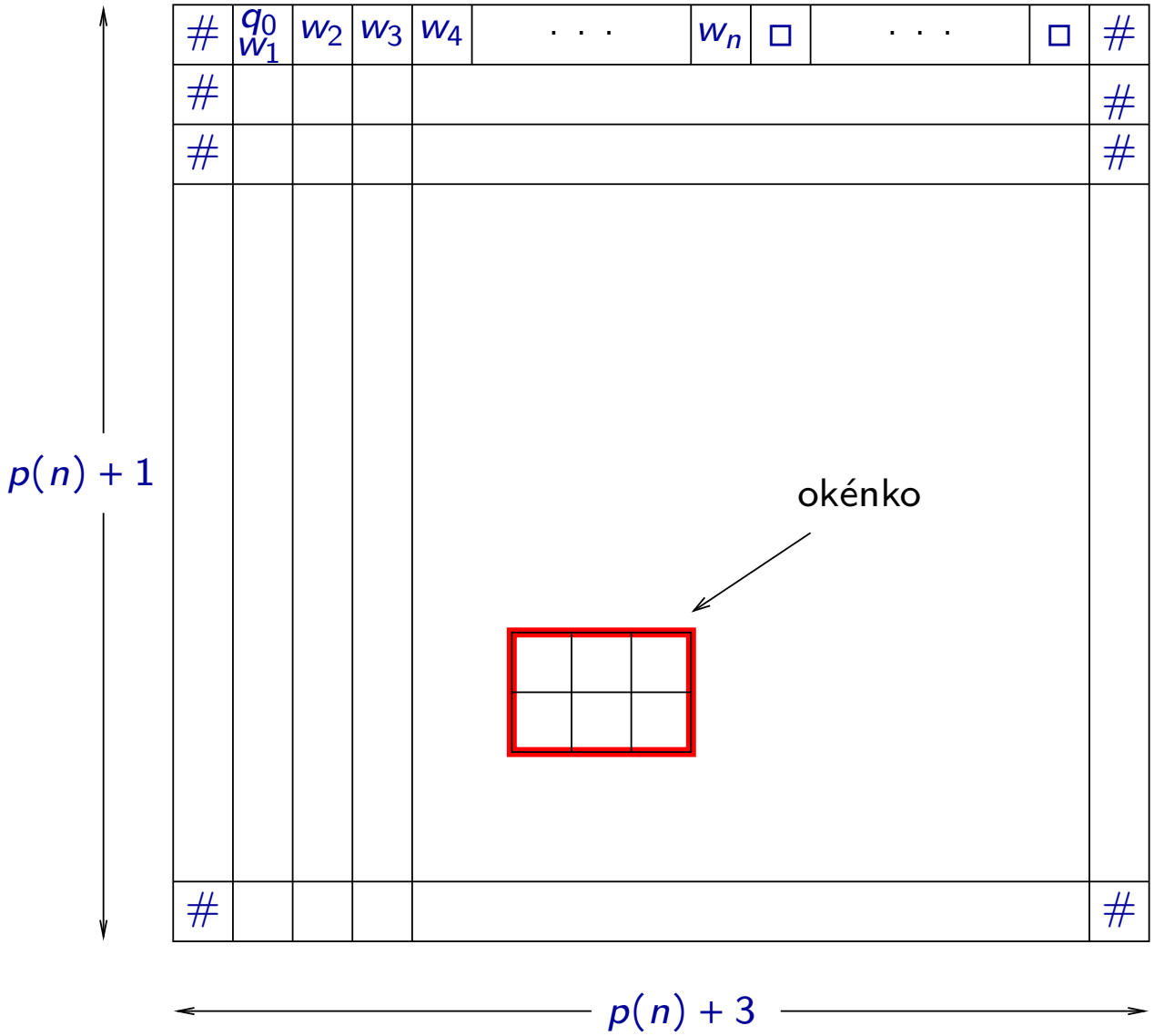
NP-úplnost problému SAT

Pokud obsah řádku $i+1$ neodpovídá konfiguraci dosažitelné jedním krokem z konfigurace na řádku i, pak se o tom můžeme přesvědčit tak, že najdeme konkrétní pozici, kde konfigurace do sebe „nepasují“.

Můžete si rozmyslet, že v takovém případě můžeme vždycky najít v dané dvojici řádků „okénko“ velikosti 2×3 takové, že o tom, že řádky **neobsahují** dvě po sobě jdoucí konfigurace, se můžeme přesvědčit jen prozkoumáním obsahu tohoto okénka (tj. aniž bychom se museli dívat na obsah ostatních políček).



NP-úplnost problému SAT



NP-úplnost problému SAT

Obsahům okének, které se mohou vyskytnout ve dvou po sobě jdoucích konfiguracích budeme říkat **korektní** a těm, které se nemohou vyskytnout ve dvou po sobě jdoucích konfiguracích (a které tedy dosvědčují, že dané dva řádky neobsahují dvě po sobě jdoucí konfigurace) budeme říkat **nekorektní**.

Přesné konkrétní podmínky, které musí korektní obsahy okének splňovat, zde nebudeme uvádět.

Místo toho si ukážeme konkrétní příklady korektních a nekorektních okének.

Zkuste si však (po prohlédnutí příkladů) tyto podmínky sami zformulovat.

NP-úplnost problému SAT

Příklady **nekorektních** okének, přičemž předpokládáme, že $\delta(q_5, \mathtt{a}) = \{(q_8, \mathtt{b}, -1), (q_3, \mathtt{a}, +1)\}$:

a	b	a
b	#	a

a	q_5 a	a
b	a	a

a	q_5 a	b
q_4 □	b	q_7 a

q_5 a	b	a
b	q_6 b	a

q_{acc} a	a	b
a	q_8 a	b

a	b	b
a	b	a

NP-úplnost problému SAT

Příklady **korektních** okének, přičemž předpokládáme, že $\delta(q_5, \mathtt{a}) = \{(q_8, \mathtt{b}, -1), (q_3, \mathtt{a}, +1)\}$:

a	b	a
a	b	a

a	q_5 a	b
a	a	q_3 b

b	a	□
q_3 b	a	□

q_5 a	b	a
b	b	a

a	a	q_{acc} b
a	a	q_{acc} b

□	□	#
□	□	#

NP-úplnost problému SAT

Množinu všech šestic znaků, které tvoří korektní obsah okének (pro daný konkrétní stroj $\mathcal{M}$) si označíme *Corr*.

Tj. $(a, b, c, d, e, f) \in Corr$ právě když

a	b	c
d	e	f

je korektní obsah okénka.

Celkový počet všech možných obsahů okének je $|\Delta|^6$, což je konstanta nezávislá na velikosti vstupu w, takže i počet prvků množiny *Corr* je konstanta nezávislá na velikosti vstupu.

NP-úplnost problému SAT

Pro každé okénko v tabulce přidáme do φ podformuli, která tvrdí, že obsah daného okénka je korektní (neboli jinak řečeno, že obsahuje některou z korektních šestic).

Tj. pro každé i takové, že $0 \leq i < p(n)$, a každé j takové, že $0 \leq j \leq p(n)$, přidáme do φ formuli

$$\bigvee_{\substack{(a,b,c,d,e,f) \\ \in Corr}} \left(x^a_{i,j} \wedge x^b_{i,j+1} \wedge x^c_{i,j+2} \wedge x^d_{i+1,j} \wedge x^e_{i+1,j+1} \wedge x^f_{i+1,j+2} \right)$$

Každá z těchto podformulí má velikost maximálně $\mathcal{O}(|\Delta|^6)$, tj. omezenou nějakou konstantou nezávislou na velikosti vstupu w.

Celkový počet těchto podformulí je $\mathcal{O}(p(n)^2)$

NP-úplnost problému SAT

Nyní zbývá zaručit poslední podmínku:

> *Poslední řádek obsahuje koncovou konfiguraci, kde stav řídící jednotky je* q_{acc}.

To je opět jednoduché – stačí přidat do φ podformuli, která tvrdí, že na některé pozici v posledním řádku (tj. řádku $p(n)$) se nachází dvojice (q_{acc}, a), kde a je nějaký symbol z Γ.

Tato podformule vypadá takto:

$$\bigvee_{j=1}^{p(n)+1} \bigvee_{a \in \Gamma} x^a_{p(n),j}$$

Velikost této formule je $\mathcal{O}(p(n))$.

NP-úplnost problému SAT

Z předchozího výkladu vidíme, že velikost formule φ vytvořené k danému vstupu w velikosti n je $\mathcal{O}(p(n)^2)$.

Jestliže je $p(n)$ polynom, tak i $p(n)^2$ je polynom, takže velikost φ je polynomiální vzhledem k n.

Vzhledem k jednoduché a pravidelné struktuře formule φ je zřejmé, že časová složitost algoritmu, který ke slovu w formuli φ vyrobí, je v podstatě úměrná velikosti formule φ, tedy také $\mathcal{O}(p(n)^2)$.

NP-úplnost problému SAT

Ukázali jsme tedy, že konstrukce je polynomiální, a nyní stručně shrneme, proč je korektní:

- Předpokládejme, že pro w je v problému P odpověď ANO, což znamená, že existuje výpočet nedeterministického stroje $\mathcal{M}$ (který řeší problém P) nad slovem w vedoucí k odpovědi ANO.

  Tento výpočet můžeme zapsat do odpovídající tabulky a proměnným ve formuli φ můžeme přiřadit booleovské hodnoty podle obsahu této tabulky.

  Je zřejmé, že při tomto ohodnocení bude mít φ hodnotu 1, protože všechny podmínky testované ve formuli φ budou splněny.

NP-úplnost problému SAT

- Předpokládejme nyní, že formule φ je splnitelná, tj. pro nějaké ohodnocení ν platí $[\varphi]_\nu = 1$.

  Podle ohodnocení ν nyní můžeme vyplnit tabulku.

  Z toho, že při ohodnocení ν musí být splněny všechny podmínky popsané ve formuli φ, vyplývá, že takto vyplněná tabulka obsahuje popis výpočtu stroje $\mathcal{M}$ nad slovem w, při kterém stroj vydá odpověď ANO, a že tedy takovýto výpočet existuje.

Vidíme tedy, že formule φ je splnitelná právě tehdy, když existuje výpočet stroje $\mathcal{M}$ vedoucí k přijetí slova w, tj. právě tehdy, když odpověď pro w je ANO.

Převod SAT na 3-SAT

Viděli jsme, že problém SAT je NP-úplný.

Ukážeme si, že zůstává NP-úplný, i když se omezíme jen na formule určitého speciální typu:

3-SAT

Vstup: Booleovská formule φ v konjunktivní normální formě, kde každá klauzule obsahuje právě 3 literály.

Otázka: Je φ splnitelná?

Převod SAT na 3-SAT

Popíšeme polynomiální algoritmus, který k zadané formuli φ vyrobí formuli φ' takovou, že:

- φ' bude v KNF a každá její klauzule bude obsahovat právě 3 literály,
- φ' bude splnitelná právě tehdy, když φ je splnitelná.

Poznámka: Jednoduchá myšlenka — převést φ do KNF — nefunguje. Problém je v tom, že výsledná formule by mohla být exponenciálně větší než φ (a tedy by ji nebylo možné sestrojit v polynomiálním čase).

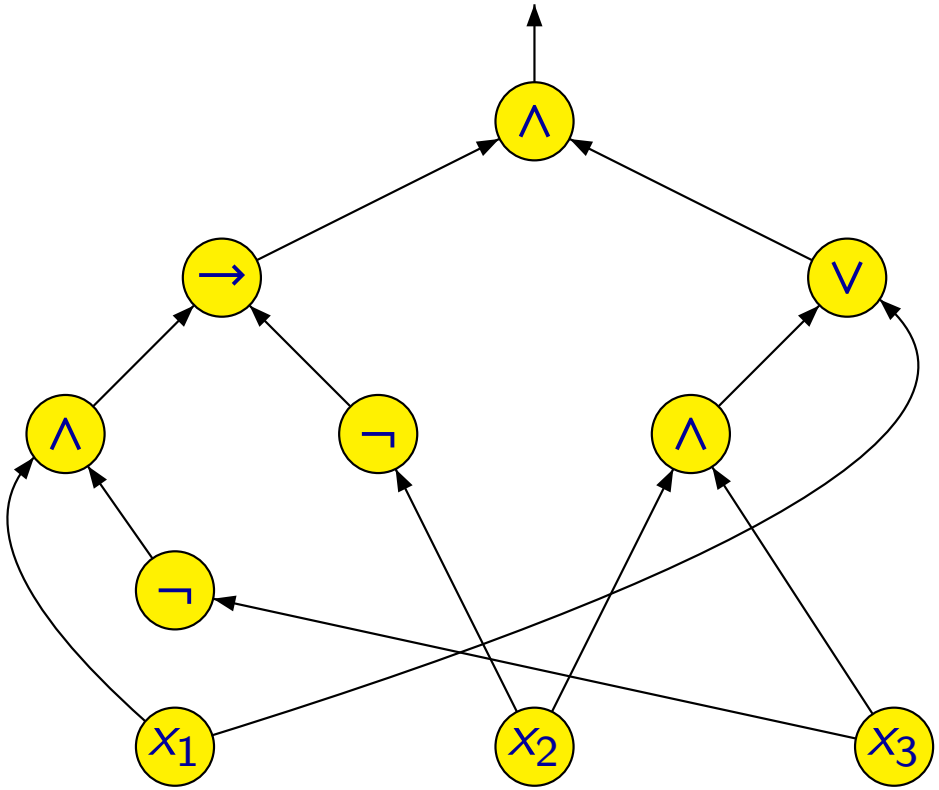
Převod SAT na 3-SAT

Algoritmus rozdělíme do dvou částí:

- Nejprve vyrobíme formuli φ_1, která bude v KNF a která bude obsahovat **nejvýše** 3 literály v každé klauzuli
  (a která bude splnitelná právě tehdy, když φ je splnitelná).
- Poté z φ_1 vyrobíme φ', která bude v KNF a která bude obsahovat **právě** 3 literály v každé klauzuli
  (a která bude splnitelná právě tehdy, když φ_1 bude splnitelná).

Převod SAT na 3-SAT

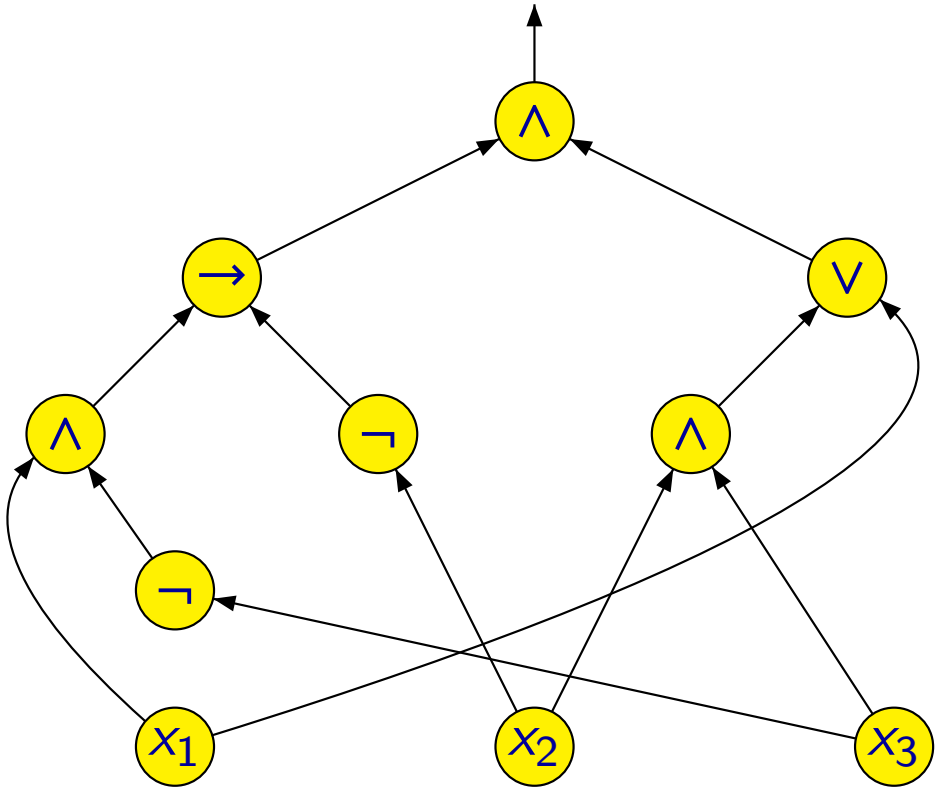
Formuli φ si můžeme znázornit jako booleovský obvod, jehož struktura je dána (abstraktním) syntaktickým stromem dané formule:



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

Převod SAT na 3-SAT

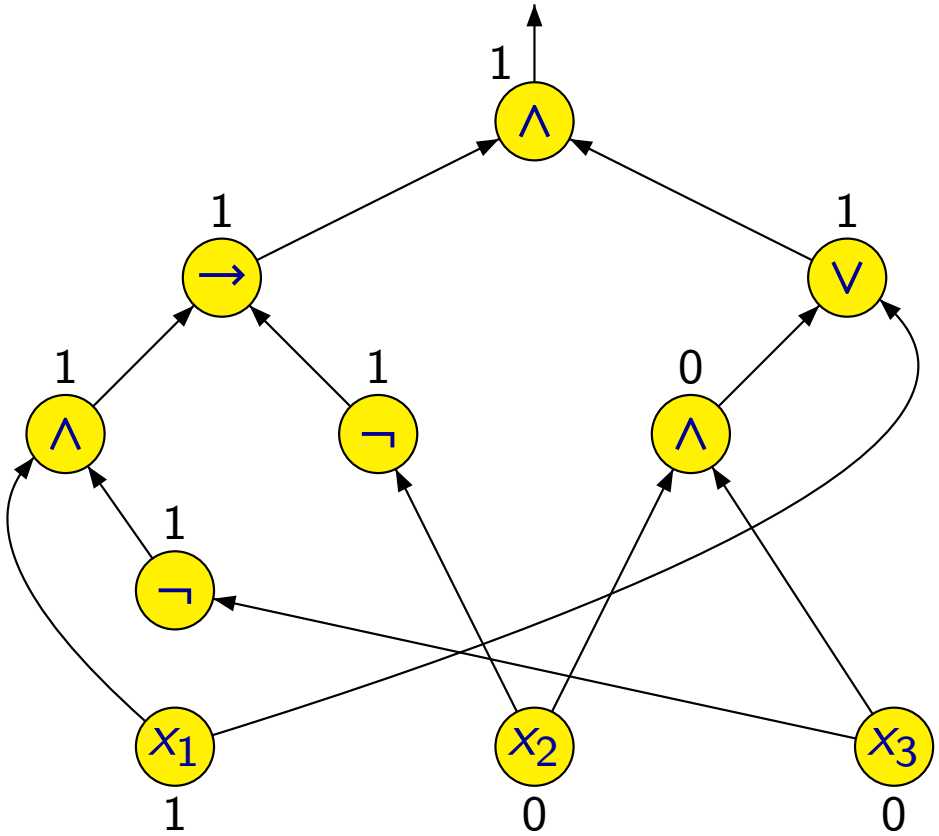
Formule φ je splnitelná právě tehdy, jestliže existuje nějaký vstup, pro který dostaneme na výstupu 1.



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

Převod SAT na 3-SAT

Formule φ je splnitelná právě tehdy, jestliže existuje nějaký vstup, pro který dostaneme na výstupu 1.



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

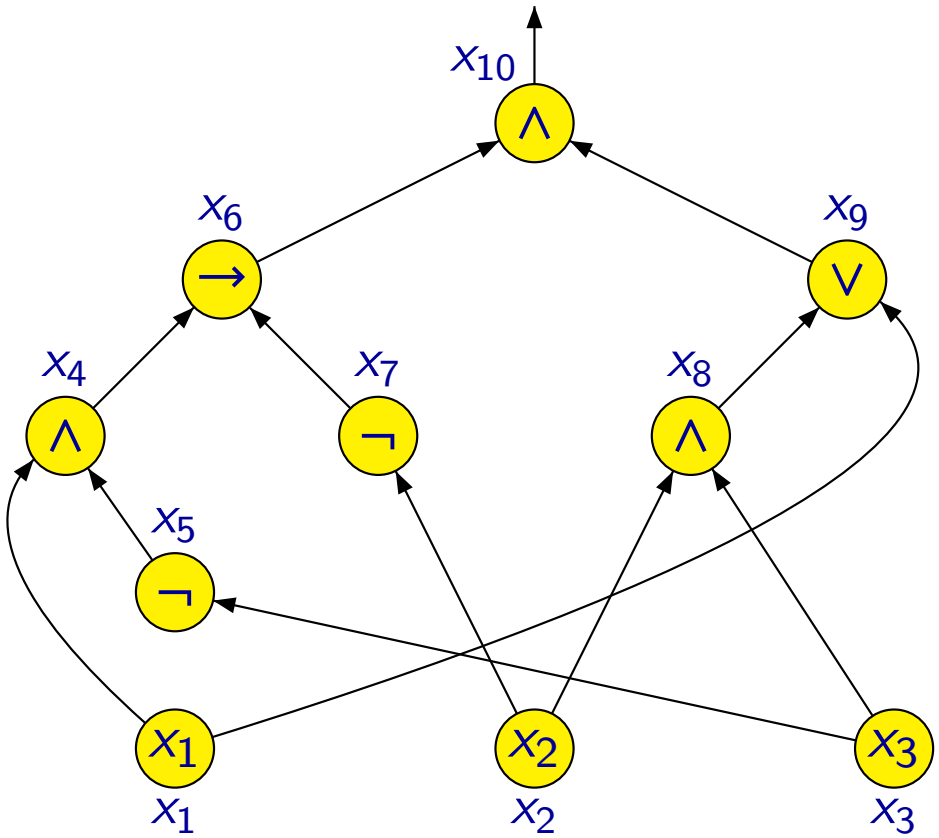
Převod SAT na 3-SAT

Ve formuli φ_1, kterou sestrojíme k dané formuli φ, se budou vyskytovat následující proměnné:

- všechny proměnné, které se vyskytují ve formuli φ
  (tj. jedna proměnná pro každý vstup obvodu),
- jedna proměnná pro každý výskyt booleovského operátoru ve φ
  (tj. jedna proměnná pro každé hradlo obvodu).

Převod SAT na 3-SAT

Příklad: Formule φ_1 bude obsahovat proměnné $x_1, x_2, \ldots, x_{10}$.



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

Převod SAT na 3-SAT

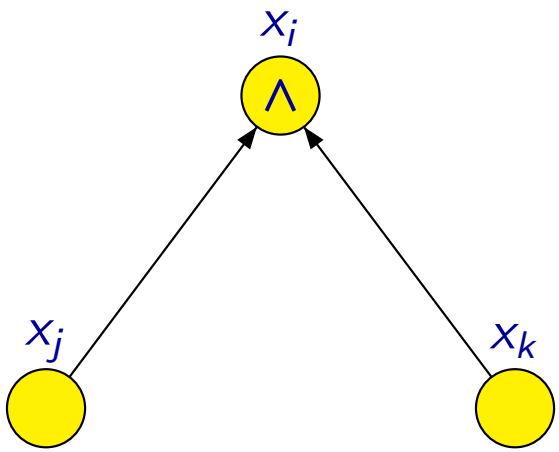
Formule φ_1 bude sestrojena tak, aby pro libovolné ohodnocení ν platilo, že $[\varphi_1]_\nu = 1$ právě tehdy, pokud:

- ν reprezentuje korektní přiřazení booleovských hodnot všem vstupům a výstupům jednotlivých hradel a
- na výstupu obvodu je hodnota 1.

(Pokud bude některá některá z těchto podmínek porušena, bude $[\varphi_1]_\nu = 0$.)

Převod SAT na 3-SAT

Zaměřme se nyní na jednotlivé hradlo (např. typu $\wedge$), jehož výstupu je přiřazena proměnná x_i a jehož vstupy jsou reprezentovány proměnnými x_j a x_k.



Převod SAT na 3-SAT

Možná (korektní) přiřazení hodnot vstupů a výstupu daného hradla (typu $\wedge$) jsou popsána následující tabulkou:

x_j	x_k	x_i
0	0	0
0	1	0
1	0	0
1	1	1

Obsah této tabulky je možné reprezentovat pomocí následující formule ψ:

$$
\begin{array}{lllllll}
(& \neg x_j & \wedge & \neg x_k & \rightarrow & \neg x_i &) \wedge \\
(& \neg x_j & \wedge & x_k & \rightarrow & \neg x_i &) \wedge \\
(& x_j & \wedge & \neg x_k & \rightarrow & \neg x_i &) \wedge \\
(& x_j & \wedge & x_k & \rightarrow & x_i &)
\end{array}
$$

Převod SAT na 3-SAT

Formule ψ reprezentuje výše uvedenou tabulku v tom smyslu, že ψ nabývá hodnoty 1 právě pro ta přiřazení, která se v této tabulce vyskytují (a pro ta, která se tam nevyskytují, nabývá hodnoty 0).

Převod SAT na 3-SAT

Libovolnou formuli tvaru

$$A \wedge B \rightarrow C$$

je možné přepsat na ekvivalentní formuli tvaru

$$\neg(A \wedge B) \vee C$$

a tu je dále možné přepsat na ekvivalentní formuli tvaru

$$\neg A \vee \neg B \vee C$$

Převod SAT na 3-SAT

Formuli

$$
\begin{array}{lllllll}
(& \neg x_j & \wedge & \neg x_k & \rightarrow & \neg x_i &) \wedge \\
(& \neg x_j & \wedge & x_k & \rightarrow & \neg x_i &) \wedge \\
(& x_j & \wedge & \neg x_k & \rightarrow & \neg x_i &) \wedge \\
(& x_j & \wedge & x_k & \rightarrow & x_i &)
\end{array}
$$

tedy můžeme přepsat na ekvivalentní formuli

$$
\begin{array}{lllllll}
(& x_j & \vee & x_k & \vee & \neg x_i &) \wedge \\
(& x_j & \vee & \neg x_k & \vee & \neg x_i &) \wedge \\
(& \neg x_j & \vee & x_k & \vee & \neg x_i &) \wedge \\
(& \neg x_j & \vee & \neg x_k & \vee & x_i &)
\end{array}
$$

Převod SAT na 3-SAT

Pokud by typ hradla byl $\vee$, postupovali bychom analogicky.

K tabulce

x_j	x_k	x_i
0	0	0
0	1	1
1	0	1
1	1	1

bychom dostali formuli

$$
\begin{array}{lllllllll}
(& x_j & \vee & x_k & \vee & \neg x_i &) & \wedge \\
(& x_j & \vee & \neg x_k & \vee & x_i &) & \wedge \\
(& \neg x_j & \vee & x_k & \vee & x_i &) & \wedge \\
(& \neg x_j & \vee & \neg x_k & \vee & x_i &) &
\end{array}
$$

Převod SAT na 3-SAT

Podobně bychom mohli reprezentovat i další booleovské operace ($\rightarrow$, $\leftrightarrow$, ...).

Pro ilustraci si ukážeme ještě konstrukci pro hradla typu $\neg$ (tentokrát máme jen jeden vstup x_j):

Tabulce

x_j	x_i
0	1
1	0

odpovídá formule

$$(\neg x_j \rightarrow x_i) \ \wedge \ (x_j \rightarrow \neg x_i)$$

kterou můžeme přepsat na tvar

$$(x_j \vee x_i) \ \wedge \ (\neg x_j \vee \neg x_i)$$

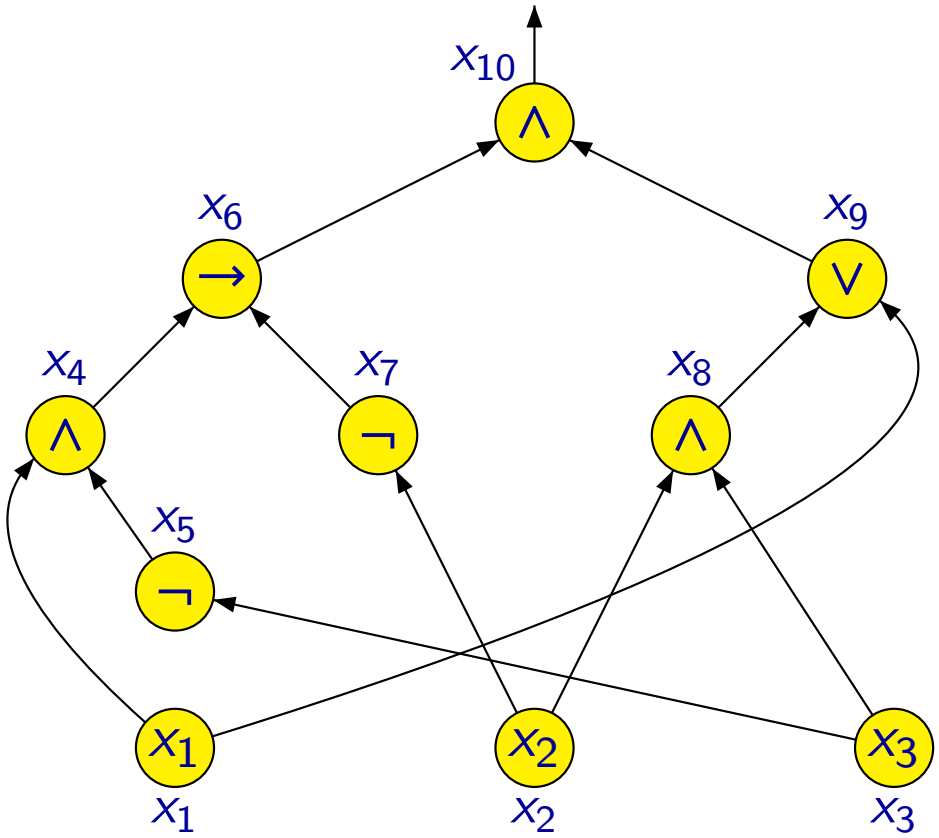
Převod SAT na 3-SAT

Nyní přistoupíme k vlastní konstrukci formule φ_1, kterou vytvoříme jako konjunkci následujících formulí:

- Pro každé hradlo přidáme jemu odpovídající formuli sestrojenou výše popsaným způsobem.
- Přidáme formuli x_{out}, kde x_{out} je proměnná reprezentující výstup obvodu.

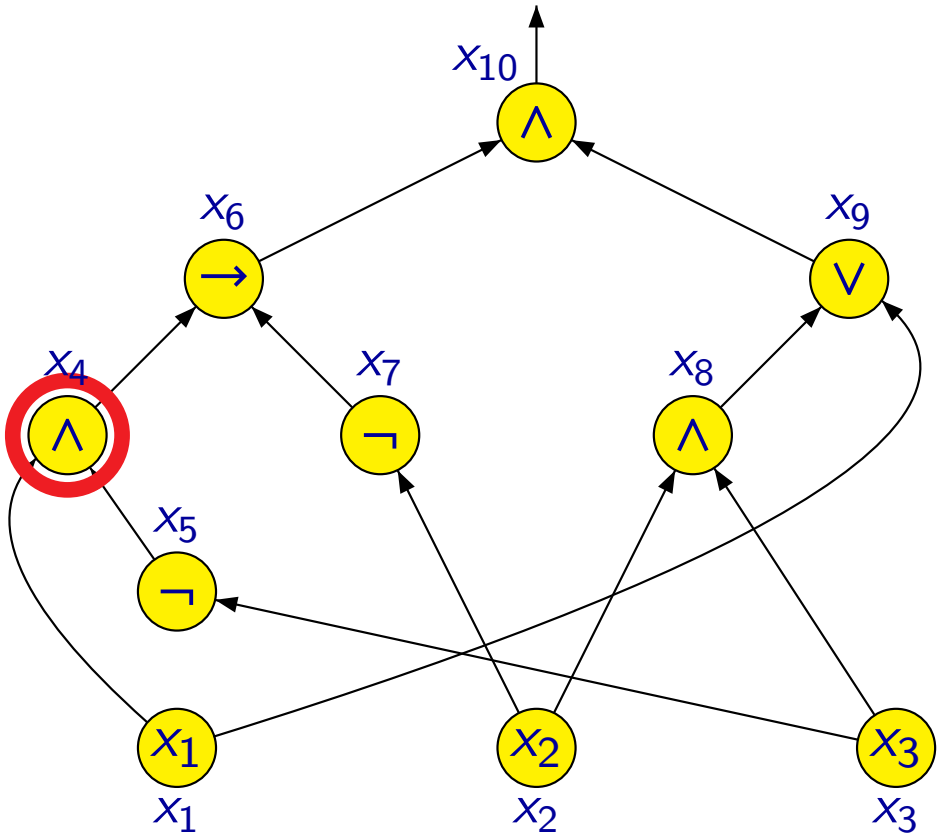
Převod SAT na 3-SAT

Příklad:



Převod SAT na 3-SAT

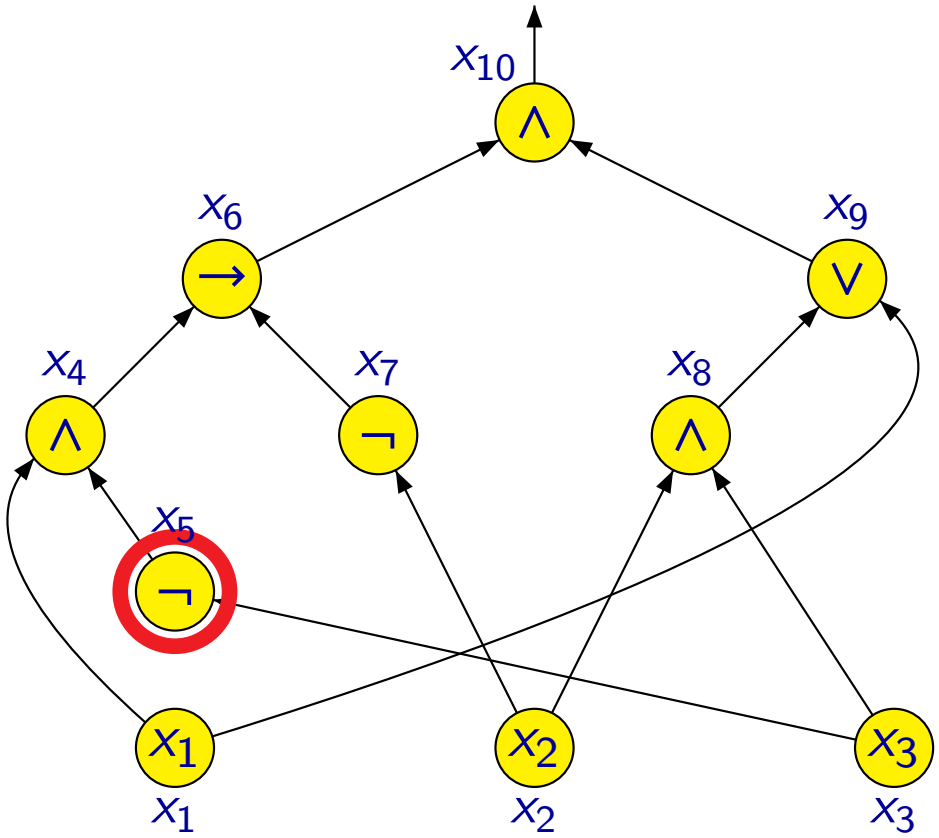
Příklad:



Pro x_4 přidáme do formule φ_1 tyto klauzule:
$(x_1 \vee x_5 \vee \neg x_4)$, $(x_1 \vee \neg x_5 \vee \neg x_4)$, $(\neg x_1 \vee x_5 \vee \neg x_4)$, $(\neg x_1 \vee \neg x_5 \vee x_4)$

Převod SAT na 3-SAT

Příklad:

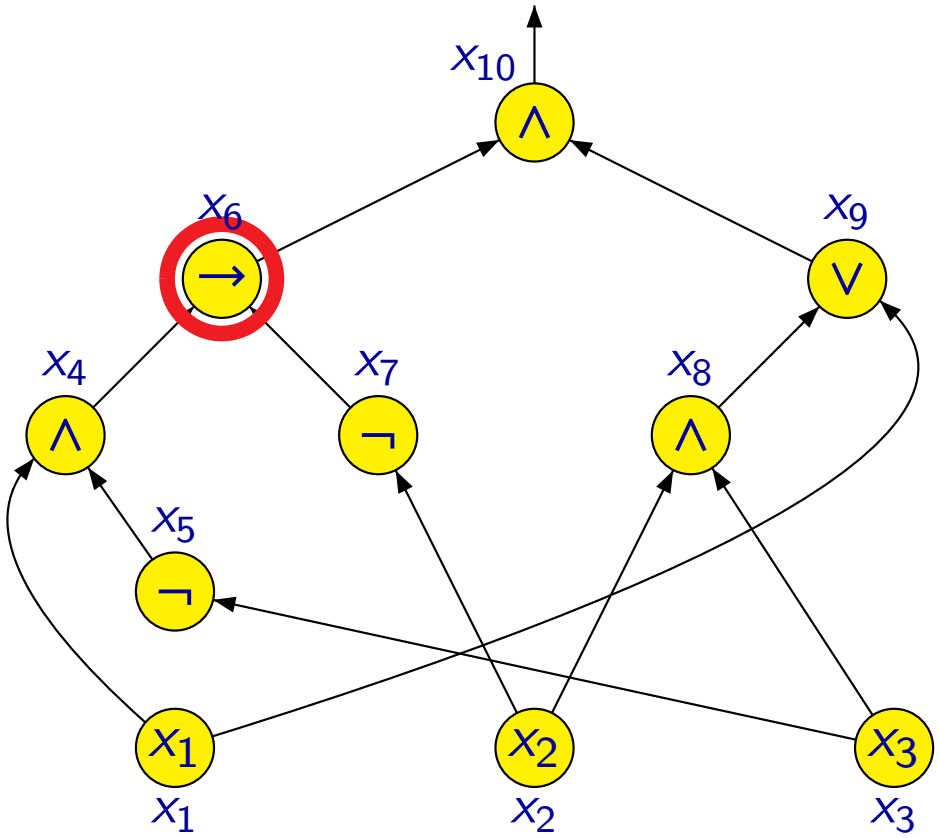


Pro x_5 přidáme do formule φ_1 tyto klauzule:

$$(x_3 \vee x_5), (\neg x_3 \vee \neg x_5)$$

Převod SAT na 3-SAT

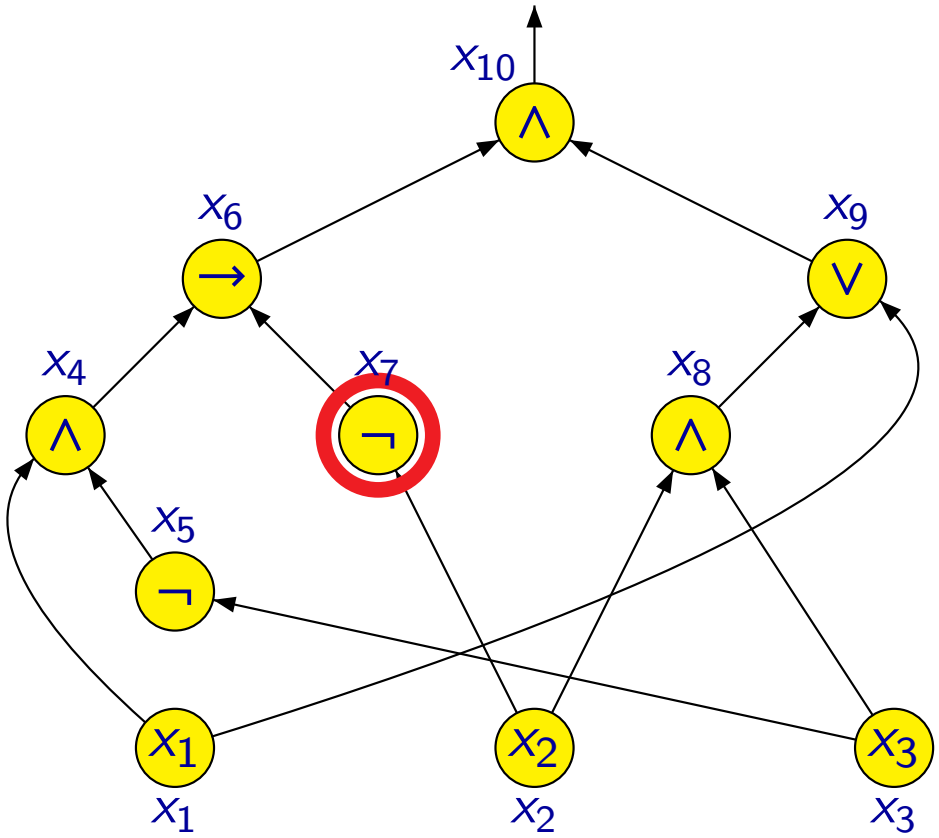
Příklad:



Pro x_6 přidáme do formule φ_1 tyto klauzule:
$(x_4 \vee x_7 \vee x_6)$, $(x_4 \vee \neg x_7 \vee x_6)$, $(\neg x_4 \vee x_7 \vee \neg x_6)$, $(\neg x_4 \vee \neg x_7 \vee x_6)$

Převod SAT na 3-SAT

Příklad:

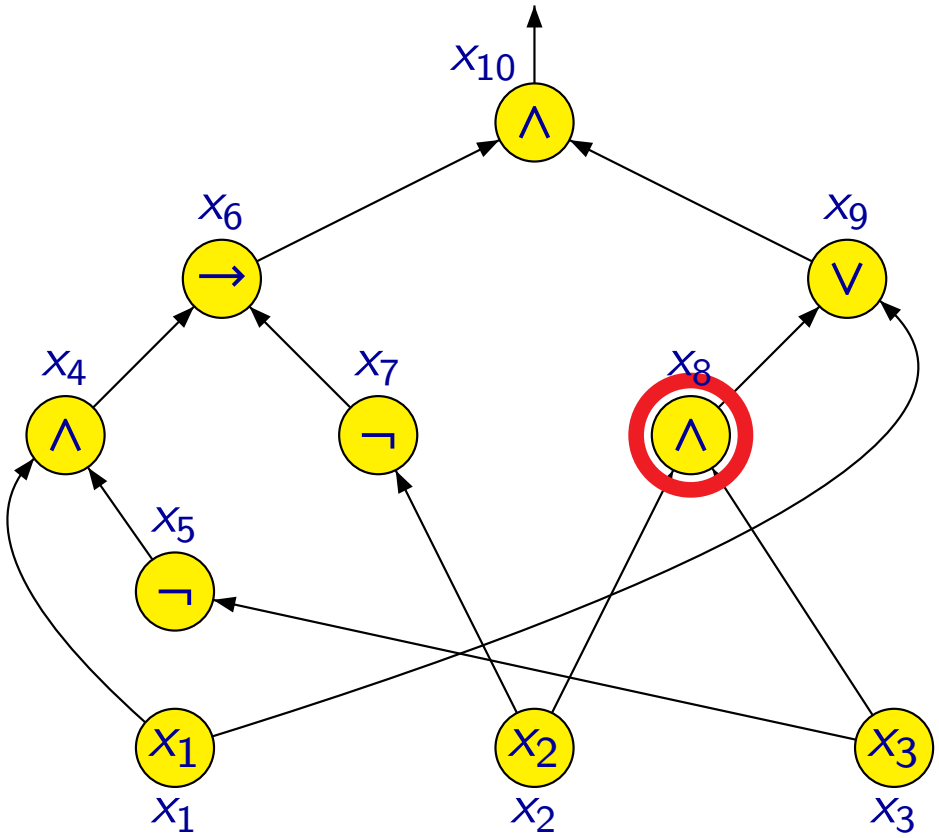


Pro x_7 přidáme do formule φ_1 tyto klauzule:

$$(x_2 \vee x_7), (\neg x_2 \vee \neg x_7)$$

Převod SAT na 3-SAT

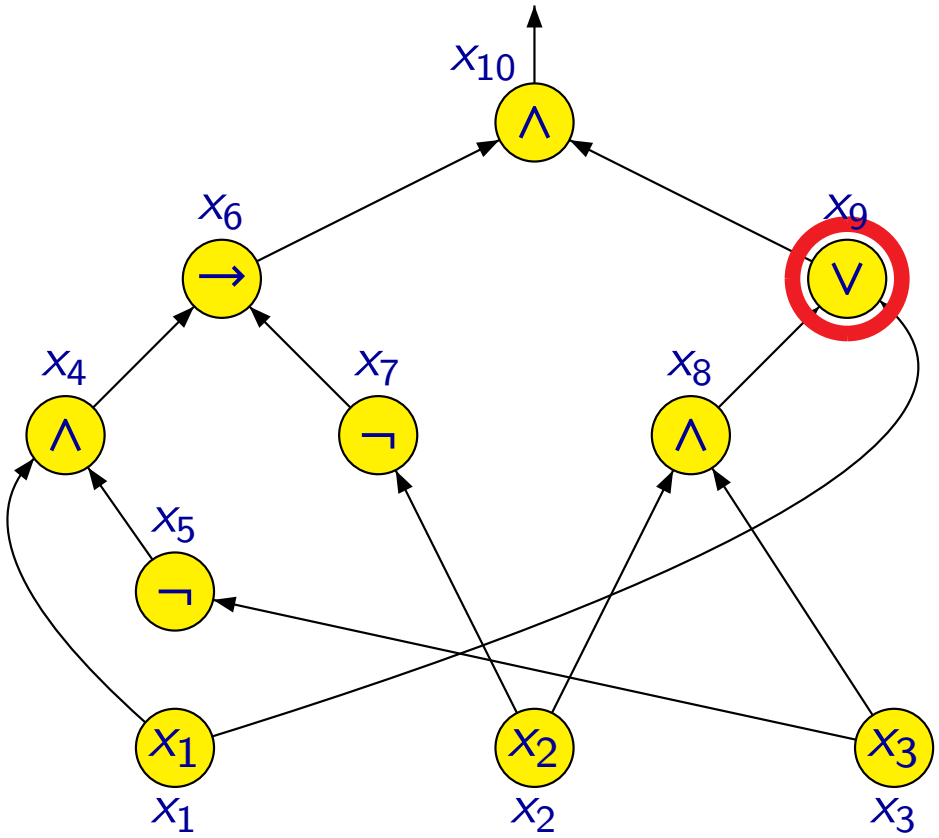
Příklad:



Pro x_8 přidáme do formule φ_1 tyto klauzule:
$(x_2 \vee x_3 \vee \neg x_8)$, $(x_2 \vee \neg x_3 \vee \neg x_8)$, $(\neg x_2 \vee x_3 \vee \neg x_8)$, $(\neg x_2 \vee \neg x_3 \vee x_8)$

Převod SAT na 3-SAT

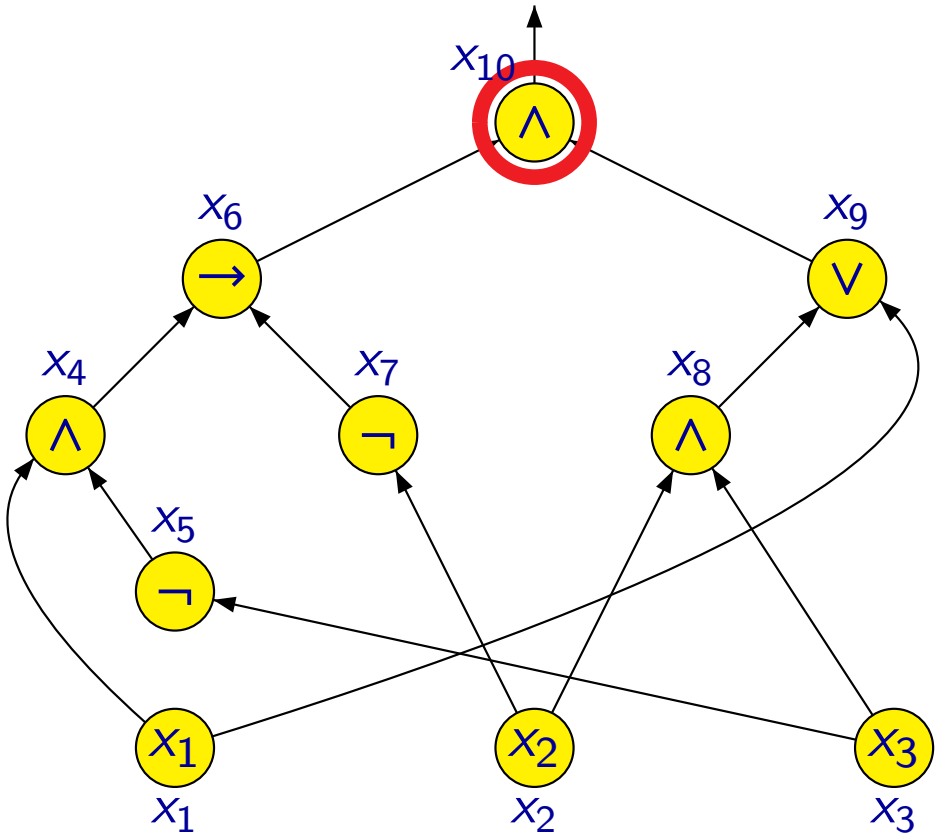
Příklad:



Pro x_9 přidáme do formule φ_1 tyto klauzule:
$(x_8 \vee x_1 \vee \neg x_9)$, $(x_8 \vee \neg x_1 \vee x_9)$, $(\neg x_8 \vee x_1 \vee x_9)$, $(\neg x_8 \vee \neg x_1 \vee x_9)$

Převod SAT na 3-SAT

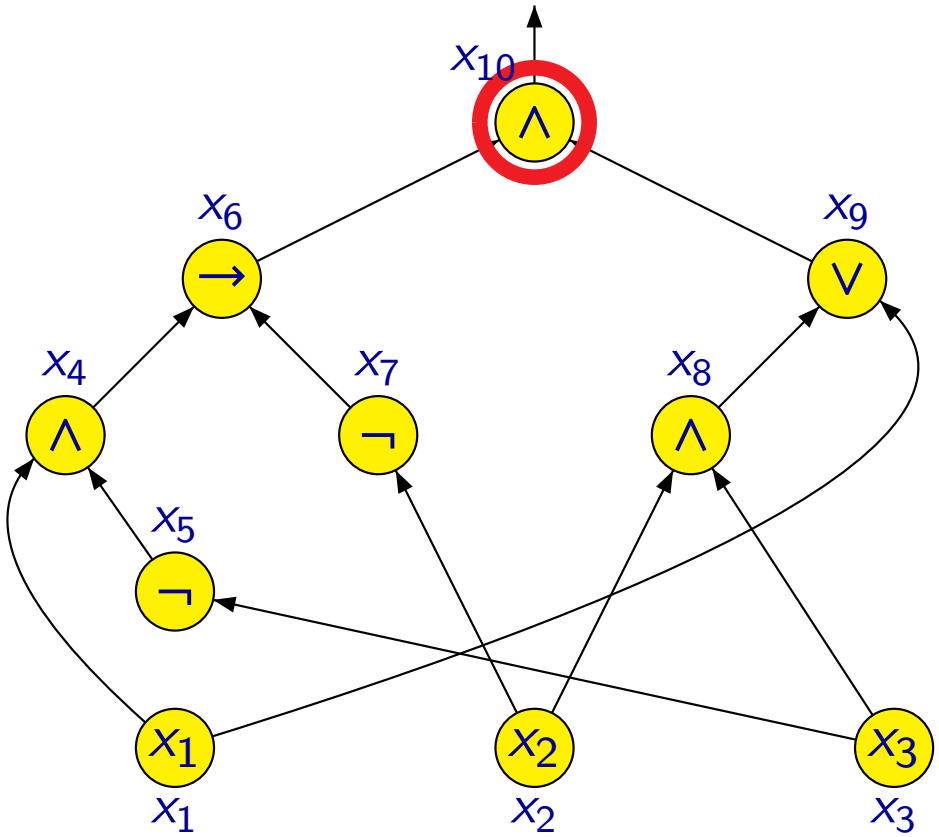
Příklad:



Pro x_{10} přidáme do formule φ_1 tyto klauzule:
$(x_6 \vee x_9 \vee \neg x_{10})$, $(x_6 \vee \neg x_9 \vee \neg x_{10})$, $(\neg x_6 \vee x_9 \vee \neg x_{10})$, $(\neg x_6 \vee \neg x_9 \vee x_{10})$

Převod SAT na 3-SAT

Příklad:



Nakonec přidáme do φ_1 klauzuli reprezentující hodnotu na výstupu:

$$(x_{10})$$

Převod SAT na 3-SAT

Celá formule φ_1 pak vypadá takto:

$$(x_1 \vee x_5 \vee \neg x_4) \wedge (x_1 \vee \neg x_5 \vee \neg x_4) \wedge (\neg x_1 \vee x_5 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_5 \vee x_4) \wedge$$
$$(x_3 \vee x_5) \wedge (\neg x_3 \vee \neg x_5) \wedge$$
$$(x_4 \vee x_7 \vee x_6) \wedge (x_4 \vee \neg x_7 \vee x_6) \wedge (\neg x_4 \vee x_7 \vee \neg x_6) \wedge (\neg x_4 \vee \neg x_7 \vee x_6) \wedge$$
$$(x_2 \vee x_7) \wedge (\neg x_2 \vee \neg x_7) \wedge$$
$$(x_2 \vee x_3 \vee \neg x_8) \wedge (x_2 \vee \neg x_3 \vee \neg x_8) \wedge (\neg x_2 \vee x_3 \vee \neg x_8) \wedge (\neg x_2 \vee \neg x_3 \vee x_8) \wedge$$
$$(x_8 \vee x_1 \vee \neg x_9) \wedge (x_8 \vee \neg x_1 \vee x_9) \wedge (\neg x_8 \vee x_1 \vee x_9) \wedge (\neg x_8 \vee \neg x_1 \vee x_9) \wedge$$
$$(x_6 \vee x_9 \vee \neg x_{10}) \wedge (x_6 \vee \neg x_9 \vee \neg x_{10}) \wedge (\neg x_6 \vee x_9 \vee \neg x_{10}) \wedge (\neg x_6 \vee \neg x_9 \vee x_{10}) \wedge$$
$$(x_{10})$$

Převod SAT na 3-SAT

Nyní se přesvědčíme, že φ_1 je splnitelná právě tehdy, když φ je splnitelná.

Převod SAT na 3-SAT

Nejprve předpokládejme, že φ je splnitelná.

Existuje tedy ohodnocení ν takové, že $[\varphi]_\nu = 1$. Definujme ohodnocení ν' následujícím způsobem:

- $\nu'(x_i) = \nu(x_i)$ pokud x_i je proměnná ve formuli φ
- Pokud x_i reprezentuje výstup hradla, $\nu'(x_i)$ nastavíme na hodnotu, která bude na tomto výstupu při ohodnocení ν.

Vzhledem k tomu, že $[\varphi]_\nu = 1$, musí platit $\nu'(x_{out}) = 1$.

Je tedy zřejmé, že bude platit $[\varphi_1]_{\nu'} = 1$, neboť x_{out} i všechny klauzule odpovídající jednotlivým hradlům budou mít při ohodnocení ν' hodnotu 1.

Převod SAT na 3-SAT

Předpokládejme nyní, že φ_1 je splnitelná,
tj. $[\varphi_1]_{\nu'} = 1$ pro nějaké ohodnocení ν'.

Snadno ověříme, že $[\varphi]_{\nu'} = 1$, neboť ν' musí odpovídat nějakému přiřazení hodnot na výstupech jednotlivých hradel, při kterém je na výstupu celého obvodu hodnota 1.

Tím jsme ověřili, že konstrukce formule φ_1 je opravdu korektní.

Převod SAT na 3-SAT

Nyní k formuli φ_1 sestrojíme formuli φ' takovou, že:

- φ' bude v KNF,
- každá klauzule formule φ' bude obsahovat právě 3 literály,
- v žádné klauzuli formule φ' se žádná proměnná nebude vyskytovat více než jednou,
- φ' bude splnitelná právě tehdy, když φ_1 je splnitelná.

Převod SAT na 3-SAT

Nejprve se zbavíme nadbytečných literálů a klauzulí:

- Pokud se v nějaké klauzuli vyskytuje nějaký literál více než jednou, odstraníme z této klauzule všechny jeho výskyty kromě jednoho.
- Pokud nějaká klauzule obsahuje současně literály x_i a $\neg x_i$ (kde x_i je nějaká proměnná), odstraníme celou tuto klauzuli (taková klauzule by měla hodnotu 1 při libovolném ohodnocení).

Je zjevné, že upravená formule je ekvivalentní s původní formulí.

Převod SAT na 3-SAT

Přidáme dvě nové proměnné y a z.

- Klauzule se třemi literály ponecháme beze změny.
- Každou klauzuli tvaru $(A \vee B)$ (tj. klauzuli se dvěma literály A a B) nahradíme následující dvojicí klauzulí:

$$(A \vee B \vee y) \ \wedge \ (A \vee B \vee \neg y)$$

- Každou klauzuli tvaru (A) (tj. klauzuli s jedním literálem) nahradíme následující čtveřicí klauzulí:

$$(A \vee y \vee z) \ \wedge \ (A \vee y \vee \neg z) \ \wedge \ (A \vee \neg y \vee z) \ \wedge \ (A \vee \neg y \vee \neg z)$$

Není těžké ověřit, že výsledná formule φ' je splnitelná právě tehdy, když je splnitelná původní formule.

Převod SAT na 3-SAT

Jestliže velikost formule φ je n, velikost formulí φ_1 i φ' bude v $\mathcal{O}(n)$.

Formule φ_1 i φ' snadno sestrojíme v čase $\mathcal{O}(n)$.

Popsaná redukce je tedy polynomiální.

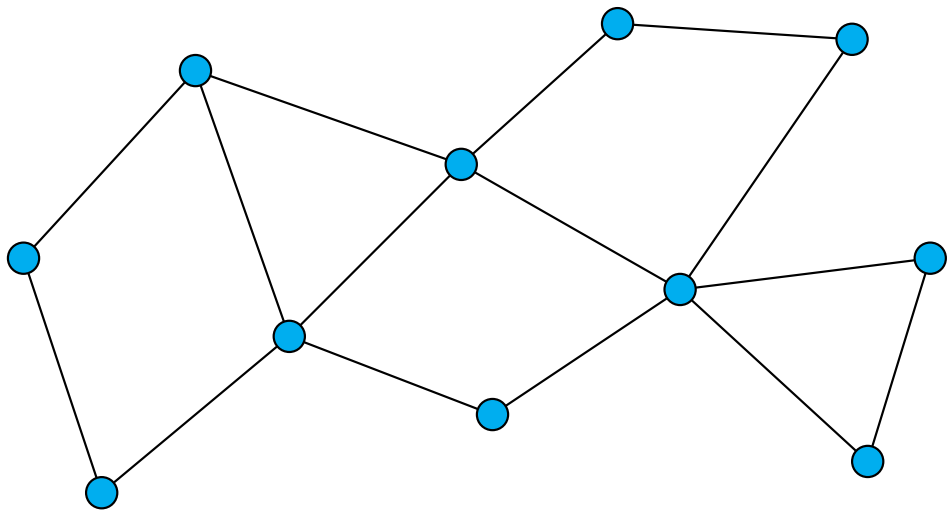
Barvení grafu 3 barvami

3-CG - Problém „Barvení grafu 3 barvami“

Vstup: Neorientovaný graf G

Výstup: Lze vrcholy grafu obarvit 3 barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Příklad:



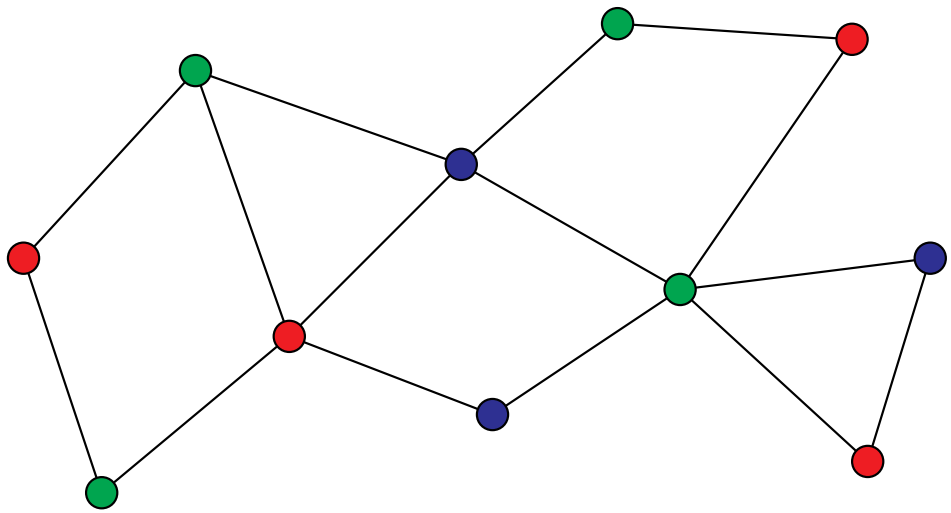
Barvení grafu 3 barvami

3-CG - Problém „Barvení grafu 3 barvami“

Vstup: Neorientovaný graf G

Výstup: Lze vrcholy grafu obarvit 3 barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Příklad:



Odpověď: Ano

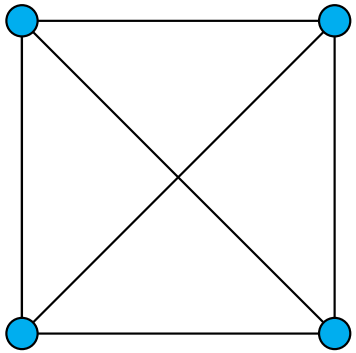
Barvení grafu 3 barvami

3-CG - Problém „Barvení grafu 3 barvami“

Vstup: Neorientovaný graf G

Výstup: Lze vrcholy grafu obarvit 3 barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Příklad:



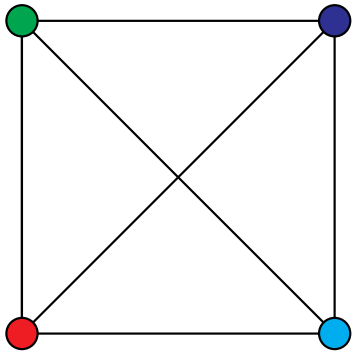
Barvení grafu 3 barvami

3-CG - Problém „Barvení grafu 3 barvami“

Vstup: Neorientovaný graf G

Výstup: Lze vrcholy grafu obarvit 3 barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Příklad:



Odpověď: NE

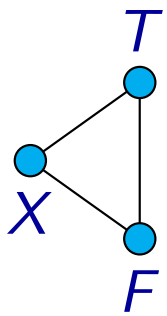
Převod 3-SAT na 3-CG

Hledáme algoritmus, který sestrojí k zadané formuli v konjunktivní normální formě s právě 3 literály v každé klauzuli graf tak, že formule bude splnitelná právě tehdy, když půjde graf obarvit korektně 3 barvami.

- Graf bude obsahovat vrcholy X, F, T propojené do trojúhelníku
- Pro každou proměnnou bude graf mít dvojici vrcholů $x, \neg x$ popojené hranou. Každý z nich bude propojen hranou s X.
- Pro každou klauzuli přidáme 6 vrcholů $c_1, c_1', c_2, c_2', c_3, c_3'$ a hrany $(c_1, c_2), (c_1, c_3), (c_2, c_3), (c_1, c_1'), (c_2, c_2'), (c_3, c_3')$
- Přidáme hrany mezi c_i' a x_j nebo $\neg x_j$ podle toho, jaký literál se nachází na i-té pozici v dané klauzuli

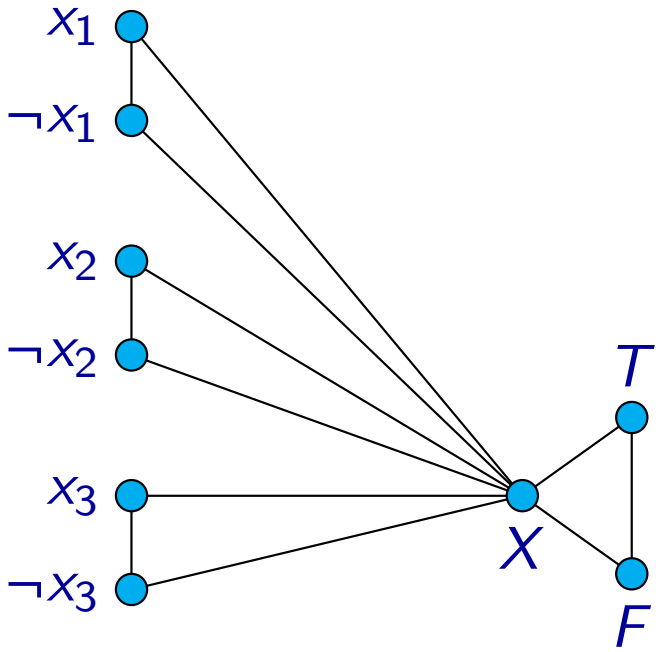
Převod 3-SAT na 3-CG

Příklad: Uvažujme formuli $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$



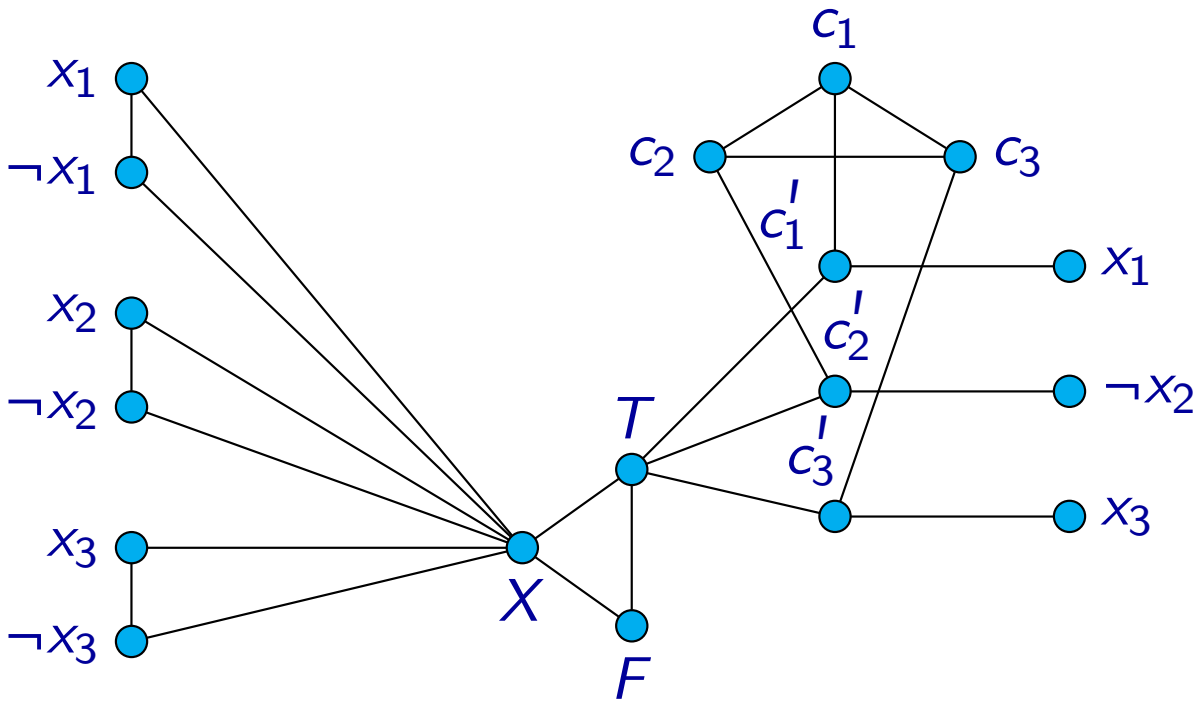
Převod 3-SAT na 3-CG

Příklad: Uvažujme formuli $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$



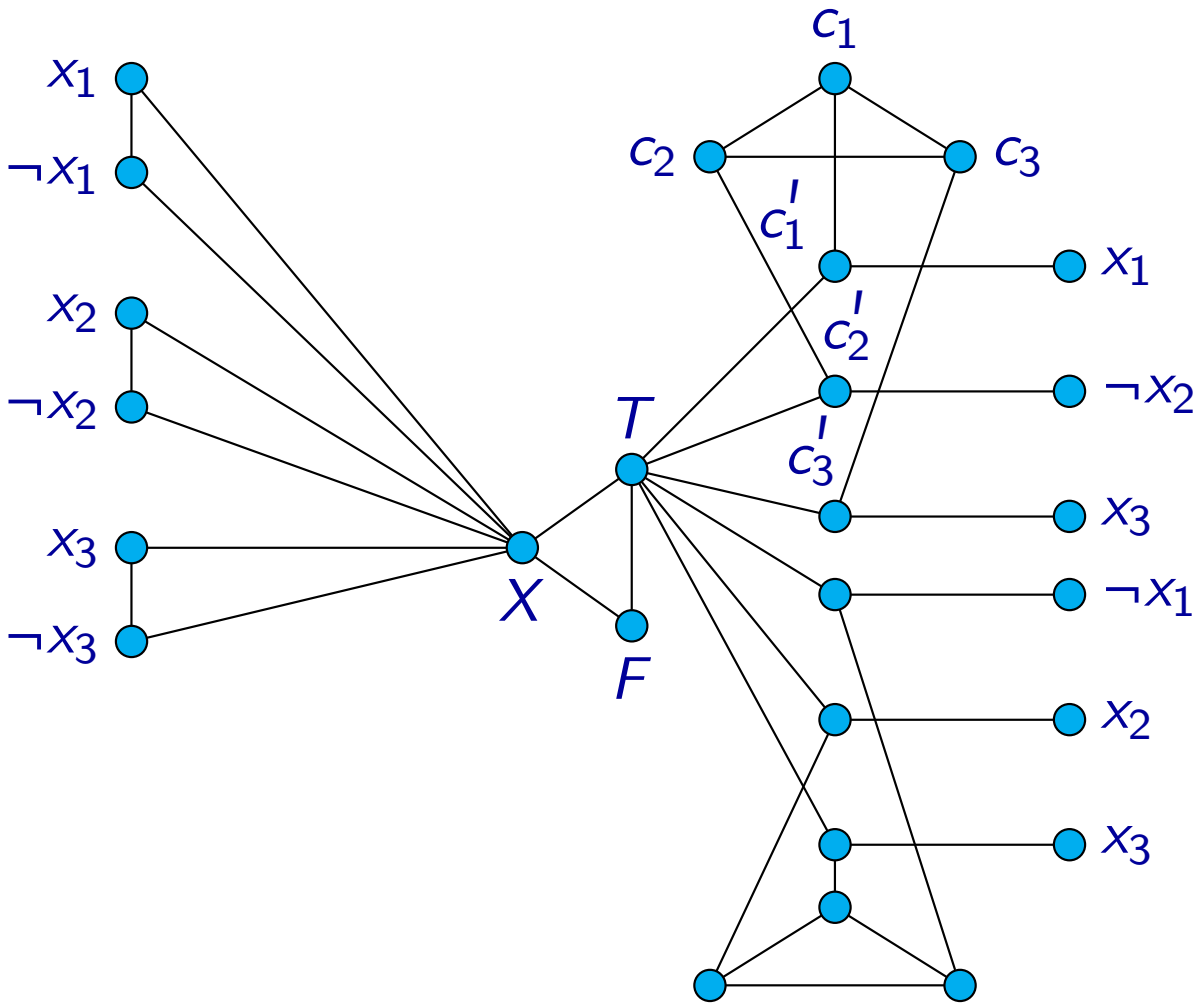
Převod 3-SAT na 3-CG

Příklad: Uvažujme formuli $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$



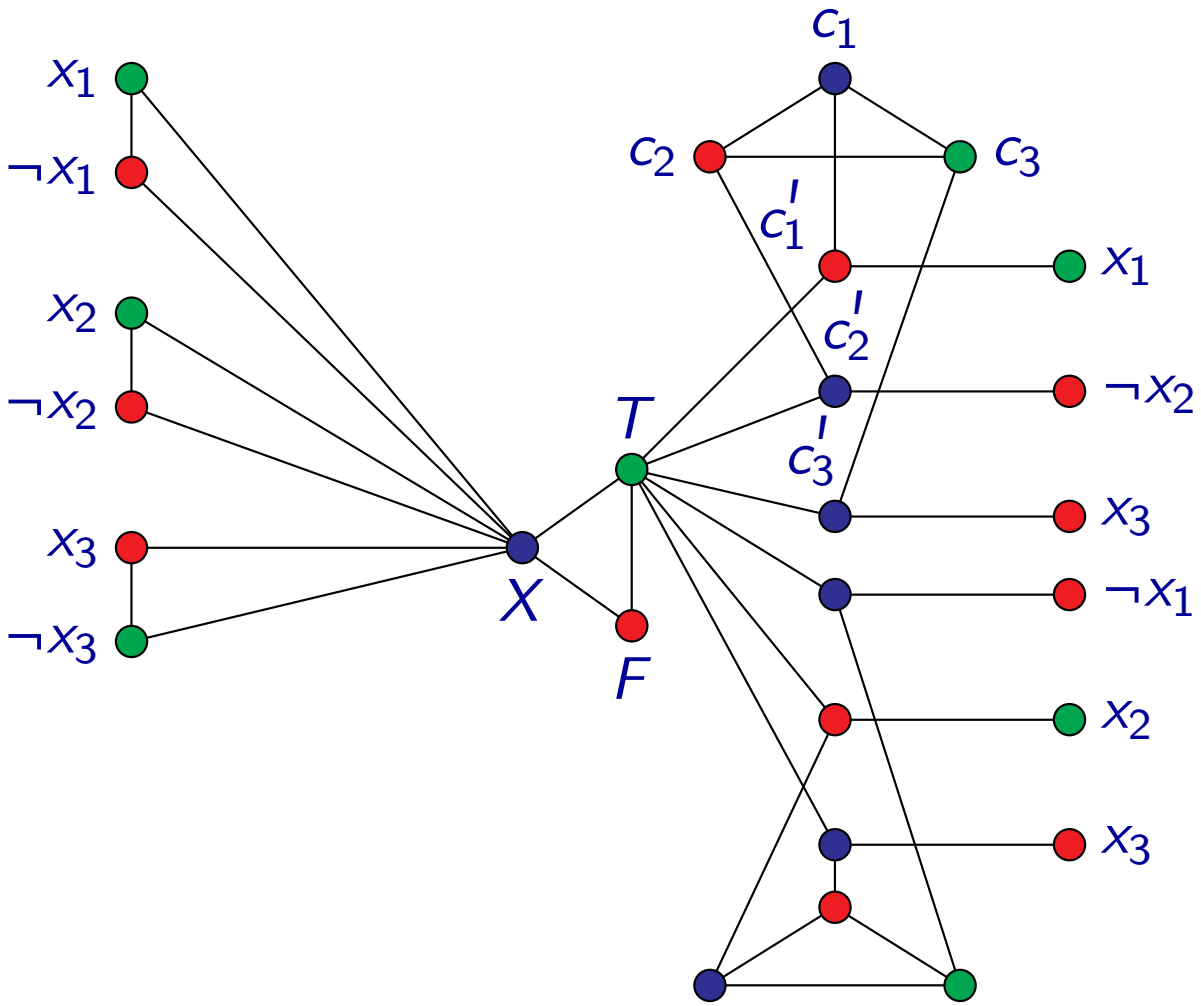
Převod 3-SAT na 3-CG

Příklad: Uvažujme formuli $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$



Převod 3-SAT na 3-CG

Příklad: Uvažujme formuli $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$



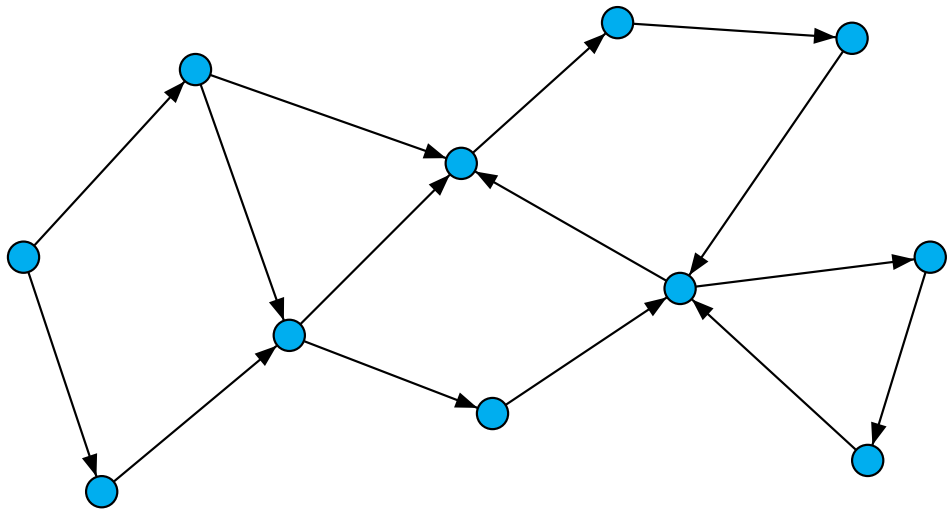
Hamiltonovský cyklus

HC - Problém „Hamiltonovský cyklus“

Vstup: Orientovaný graf G

Výstup: Existuje v grafu Hamiltonovský cyklus (orientovaná kružnice procházející každý vrchol právě jednou)?

Příklad:



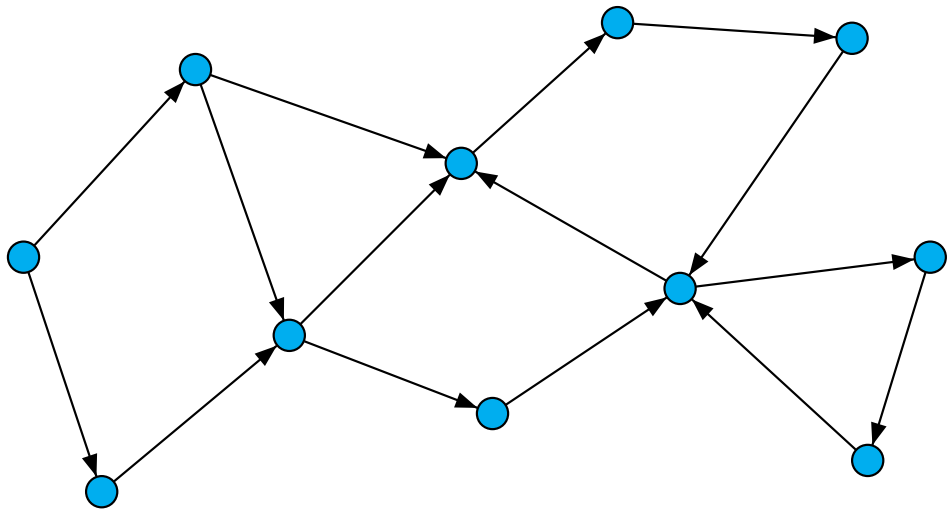
Hamiltonovský cyklus

HC - Problém „Hamiltonovský cyklus“

Vstup: Orientovaný graf G

Výstup: Existuje v grafu Hamiltonovský cyklus (orientovaná kružnice procházející každý vrchol právě jednou)?

Příklad:



Odpověď: NE

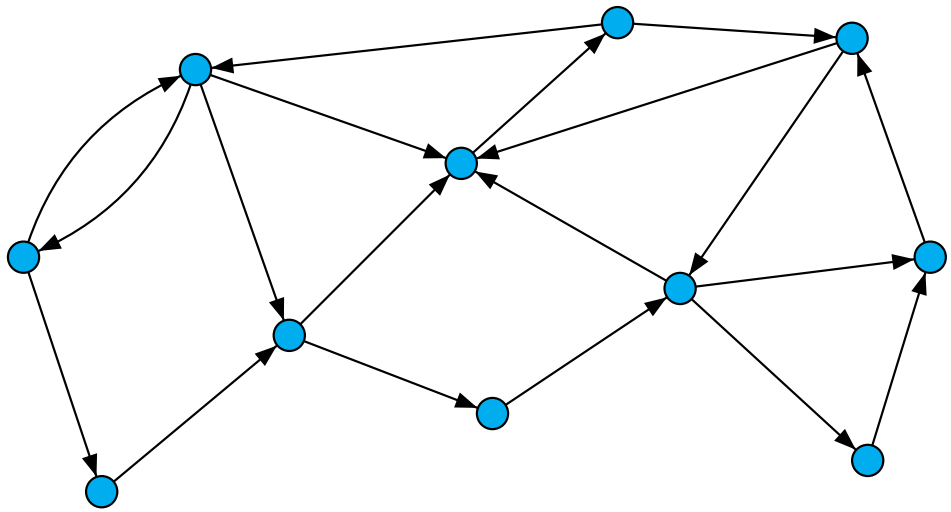
Hamiltonovský cyklus

HC - Problém „Hamiltonovský cyklus“

Vstup: Orientovaný graf G

Výstup: Existuje v grafu Hamiltonovský cyklus (orientovaná kružnice procházející každý vrchol právě jednou)?

Příklad:



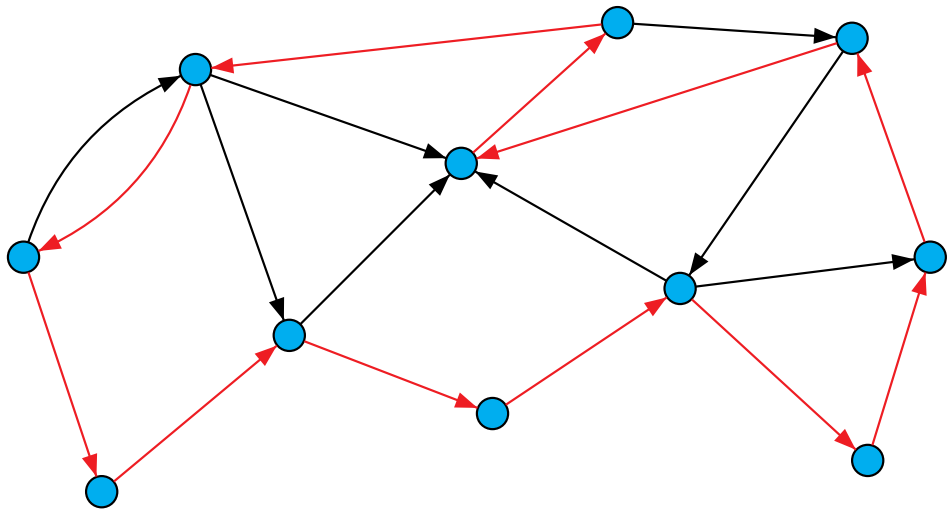
Hamiltonovský cyklus

HC - Problém „Hamiltonovský cyklus“

Vstup: Orientovaný graf G

Výstup: Existuje v grafu Hamiltonovský cyklus (orientovaná kružnice procházející každý vrchol právě jednou)?

Příklad:



Odpověď: ANO

Hamiltonovský cyklus

- Problém patří do třídy NP. Stačí vzít jako svědka množinu hran a ověřit, že tyto hrany tvoří cyklus a každý vrchol má právě jednu vstupní a výstupní hranu.
- NP-obtížnost můžeme ukázat například převodem z problému VC nebo SAT
- Oba důkazy jsou složitější na popsání, proto je vynecháme

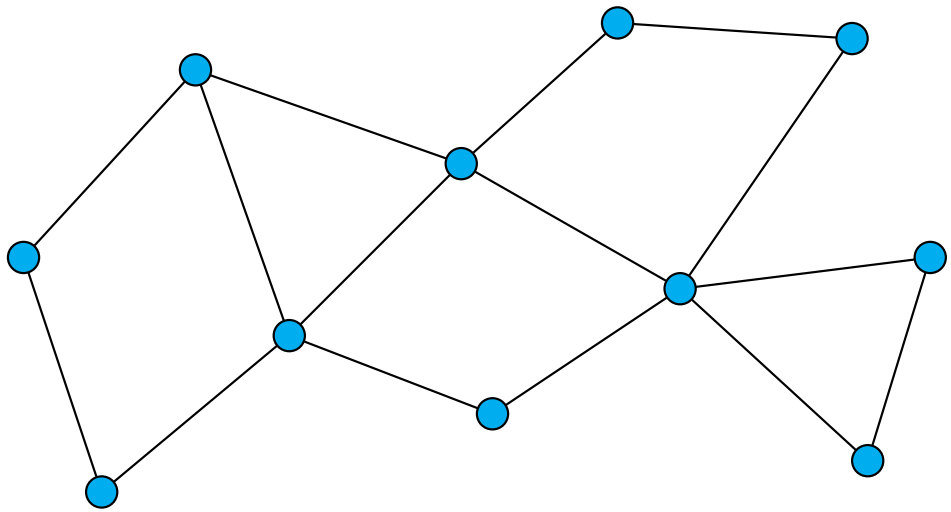
Hamiltonovská kružnice

HK - Problém „Hamiltonovská kružnice“

Vstup: Neorientovaný graf G

Výstup: Existuje v grafu Hamiltonovská kružnice (neorientovaná kružnice procházející každý vrchol právě jednou)?

Příklad:



Odpověď: NE

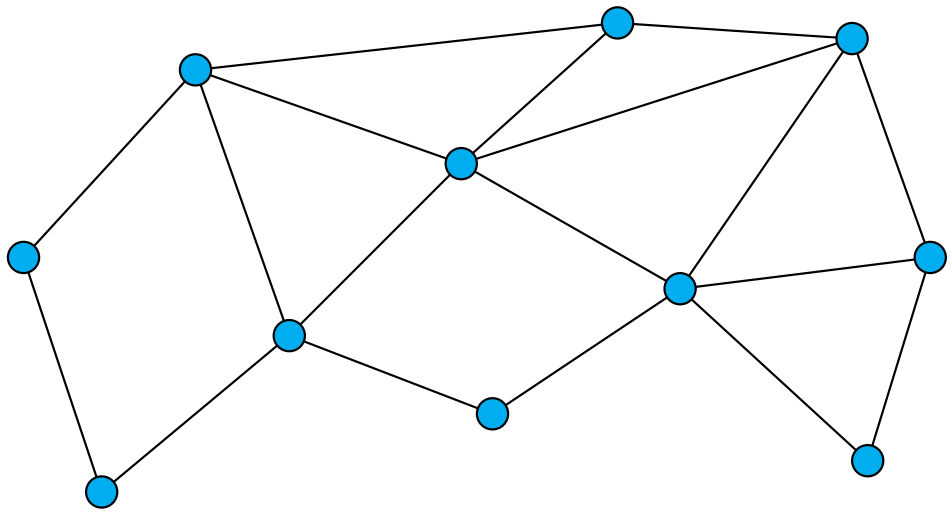
Hamiltonovská kružnice

HK - Problém „Hamiltonovská kružnice“

Vstup: Neorientovaný graf G

Výstup: Existuje v grafu Hamiltonovská kružnice (neorientovaná kružnice procházející každý vrchol právě jednou)?

Příklad:



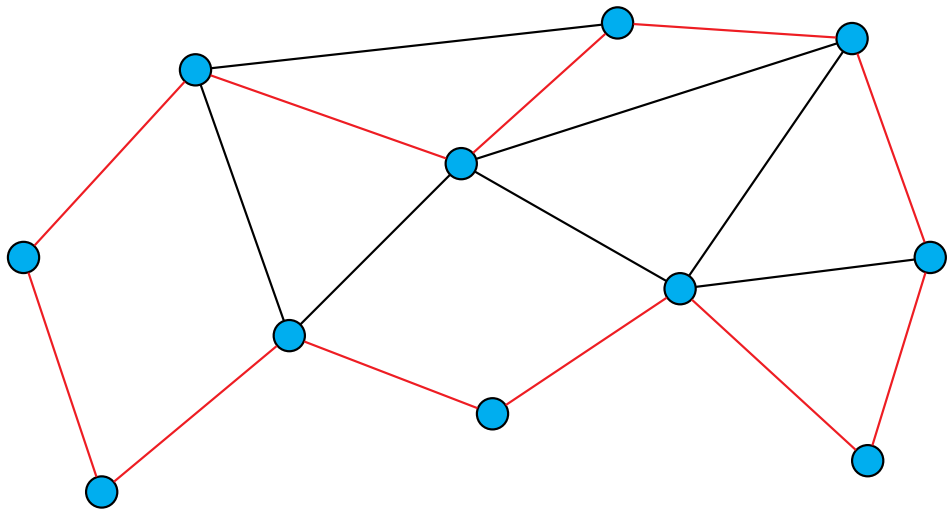
Hamiltonovská kružnice

HK - Problém „Hamiltonovská kružnice“

Vstup: Neorientovaný graf G

Výstup: Existuje v grafu Hamiltonovská kružnice (neorientovaná kružnice procházející každý vrchol právě jednou)?

Příklad:

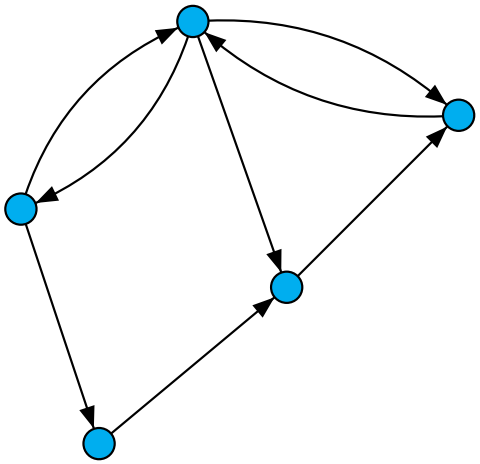


Odpověď: ANO

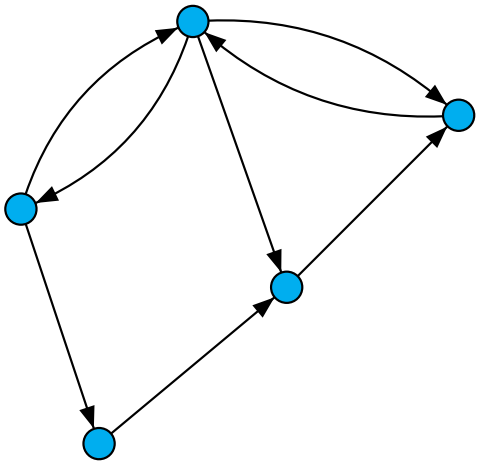
Hamiltonovská kružnice

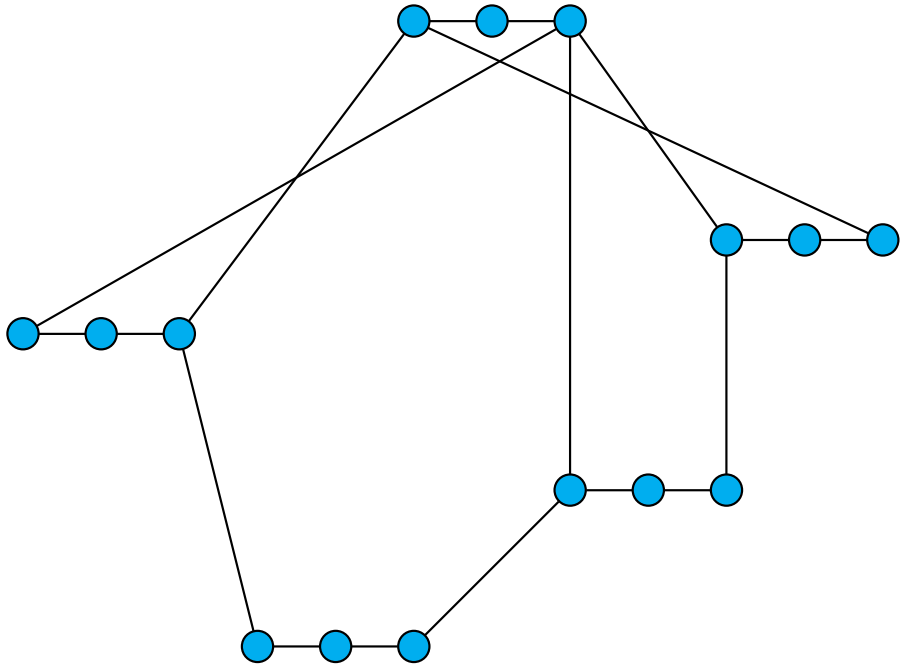
- Problém patří do třídy NP. Stačí vzít jako svědka množinu hran a ověřit, že tyto hrany tvoří kružnici a každý vrchol má právě dvě hrany.
- NP-obtížnost můžeme ukázat převodem z problému HC
- Ke každému vrcholu x orientovaného grafu dáme do neorientovaného tři vrcholy x_1, x_2, x_3, které spojíme hranami $(x_1, x_2), (x_2, x_3)$.
- Každá orientovaná hrana (x, y) bude reprezentována hranou (x_3, y_1)

Převod HC na HK

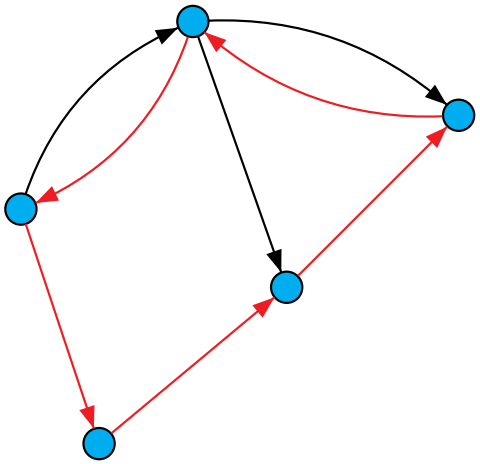


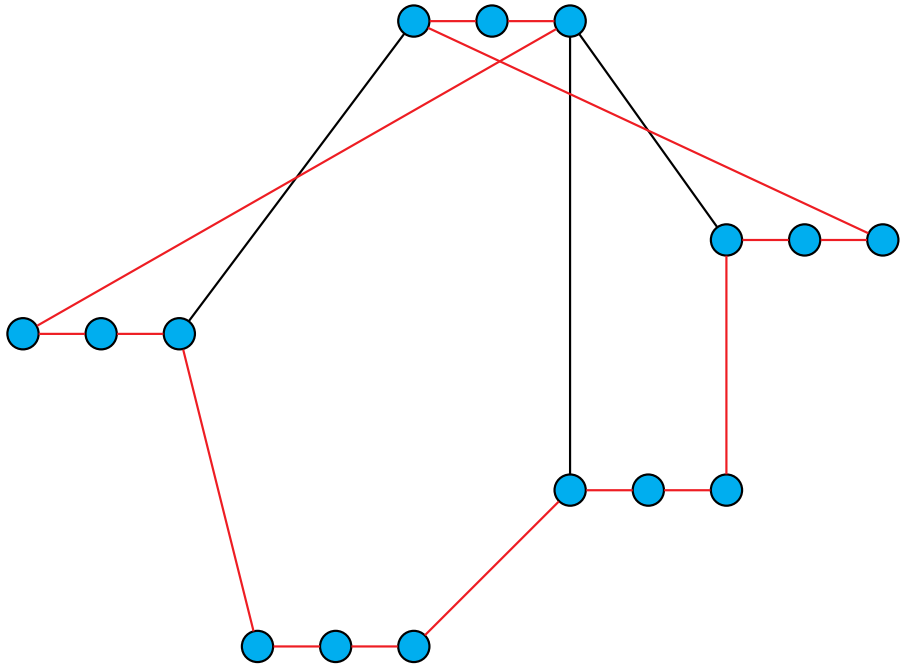
Převod HC na HK





Převod HC na HK





Převod 3-SAT na ILP

Problém ILP (celočíselné lineární programování)

Vstup: Celočíselná matice A a celočíselný vektor b.

Otázka: Existuje celočíselný vektor x, takový že $Ax \leq b$?

Příklad instance problému:

$$A = \begin{pmatrix} 3 & -2 & 5 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix} \qquad b = \begin{pmatrix} 8 \\ -3 \\ 5 \end{pmatrix}$$

Ptáme se tedy, zda existuje celočíselné řešení následující soustavy nerovnic:

$$\begin{aligned} 3x_1 - 2x_2 + 5x_3 &\leq 8 \\ x_1 + x_3 &\leq -3 \\ 2x_1 + x_2 &\leq 5 \end{aligned}$$

Převod 3-SAT na ILP

Jedním z řešení soustavy

$$\begin{aligned} 3x_1 - 2x_2 + 5x_3 &\leq 8 \\ x_1 + x_3 &\leq -3 \\ 2x_1 + x_2 &\leq 5 \end{aligned}$$

je například $x_1 = -4$, $x_2 = 1$, $x_3 = 1$, tj.

$$x = \begin{pmatrix} -4 \\ 1 \\ 1 \end{pmatrix}$$

neboť

$$\begin{aligned} 3 \cdot (-4) - 2 \cdot 1 + 5 \cdot 1 &= -9 \leq 8 \\ -4 + 1 &= -3 \leq -3 \\ 2 \cdot (-4) + 1 &= -7 \leq 5 \end{aligned}$$

Pro tuto instanci je tedy odpověď Ano.

Převod 3-SAT na ILP

Věta

Problém ILP je NP-těžký.

NP-obtížnost problému ILP dokážeme tak, že ukážeme polynomiální redukci z problému 3-SAT.

Převod 3-SAT na ILP

Předpokládejme, že máme dánu nějakou konkrétní instanci problému 3-SAT, například následující formuli φ:

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Naším úkolem je vyrobit k formuli φ soustavu lineárních nerovnic takovou, že tato soustava bude mít řešení v oboru celých čísel právě tehdy, když je formule φ splnitelná.

Převod 3-SAT na ILP

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Krok 1:
Každé booleovské proměnné x_i ve formuli φ bude v soustavě nerovnic odpovídat neznámá x_i'.

Například pro formuli φ uvedenou výše bude soustava nerovnic obsahovat neznámé x_1', x_2', x_3', x_4'.

Převod 3-SAT na ILP

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Krok 2:

Nejprve do soustavy přidáme pro každou neznámou x_i' dvojici nerovnic $x_i' \geq 0$ a $x_i' \leq 1$:

Převod 3-SAT na ILP

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Krok 2:

Nejprve do soustavy přidáme pro každou neznámou x_i' dvojici nerovnic $x_i' \geq 0$ a $x_i' \leq 1$:

$$\begin{array}{lcl} x_1' & \geq & 0 \\ x_1' & \leq & 1 \\ x_2' & \geq & 0 \\ x_2' & \leq & 1 \end{array} \qquad \begin{array}{lcl} x_3' & \geq & 0 \\ x_3' & \leq & 1 \\ x_4' & \geq & 0 \\ x_4' & \leq & 1 \end{array}$$

Převod 3-SAT na ILP

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Krok 2:
Nejprve do soustavy přidáme pro každou neznámou x_i' dvojici nerovnic $x_i' \geq 0$ a $x_i' \leq 1$:

$$\begin{array}{lcl} x_1' & \geq & 0 \\ x_1' & \leq & 1 \\ x_2' & \geq & 0 \\ x_2' & \leq & 1 \end{array} \qquad \begin{array}{lcl} x_3' & \geq & 0 \\ x_3' & \leq & 1 \\ x_4' & \geq & 0 \\ x_4' & \leq & 1 \end{array}$$

Poznámka: Tyto nerovnice zaručují, že v libovolném řešení výsledné soustavy bude muset pro všechna x_i' platit $x_i' \in \{0, 1\}$.

Převod 3-SAT na ILP

Krok 3:
Pro každou klauzuli tvaru $(\ell_1 \vee \ell_2 \vee \ell_3)$, kde ℓ_i jsou jednotlivé literály, přidáme do soustavy nerovnic nerovnici

$$f_1 + f_2 + f_3 \geq 1$$

kde

$$f_i = \begin{cases} x_i' & \text{pokud } \ell_i = x_i \\ (1 - x_i') & \text{pokud } \ell_i = \neg x_i \end{cases}$$

Převod 3-SAT na ILP

Krok 3:
Pro každou klauzuli tvaru $(\ell_1 \vee \ell_2 \vee \ell_3)$, kde ℓ_i jsou jednotlivé literály, přidáme do soustavy nerovnic nerovnici

$$f_1 + f_2 + f_3 \geq 1$$

kde

$$f_i = \begin{cases} x_i' & \text{pokud } \ell_i = x_i \\ (1 - x_i') & \text{pokud } \ell_i = \neg x_i \end{cases}$$

Příklad: Pro klauzuli $(x_1 \vee \neg x_3 \vee \neg x_4)$ přidáme nerovnici

$$x_1' + (1 - x_3') + (1 - x_4') \geq 1$$

Převod 3-SAT na ILP

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Takto vypadá celá odpovídající soustava nerovnic:

$$\begin{aligned}
x_1' &\geq 0 \\
x_1' &\leq 1 \\
x_2' &\geq 0 \\
x_2' &\leq 1 \\
x_3' &\geq 0 \\
x_3' &\leq 1 \\
x_4' &\geq 0 \\
x_4' &\leq 1 \\
x_1' + (1 - x_2') + x_3' &\geq 1 \\
x_2' + (1 - x_3') + x_4' &\geq 1 \\
x_1' + (1 - x_3') + (1 - x_4') &\geq 1 \\
(1 - x_1') + (1 - x_2') + x_4' &\geq 1
\end{aligned}$$

Převod 3-SAT na ILP

Krok 4:
Soustavu nerovnic převedeme pomocí jednoduchých aritmetických úprav do požadovaného maticového tvaru tak, aby všechny nerovnice byly tvaru

$$c_1 \cdot x_1' + c_2 \cdot x_2' + \cdots + c_n \cdot x_n' \leq d$$

kde $c_1, c_2, \ldots, c_n$ a d jsou konstanty.

Převod 3-SAT na ILP

Krok 4:
Soustavu nerovnic převedeme pomocí jednoduchých aritmetických úprav do požadovaného maticového tvaru tak, aby všechny nerovnice byly tvaru

$$c_1 \cdot x_1' + c_2 \cdot x_2' + \cdots + c_n \cdot x_n' \leq d$$

kde $c_1, c_2, \ldots, c_n$ a d jsou konstanty.

Poznámka:
Pokud se v nerovnici vyskytuje nerovnost '$\geq$' místo '$\leq$', můžeme využít toho, že $x \geq y$ právě tehdy, když $-x \leq -y$.

Převod 3-SAT na ILP

Příklad:

$$x_1' + (1 - x_3') + (1 - x_4') \geq 1$$

Převod 3-SAT na ILP

Příklad:

$$\begin{aligned} x_1' + (1 - x_3') + (1 - x_4') &\geq 1 \quad \text{// sečteme jednotlivé členy} \\ x_1' - x_3' - x_4' + 2 &\geq 1 \end{aligned}$$

Převod 3-SAT na ILP

Příklad:

$$
\begin{aligned}
x_1' + (1 - x_3') + (1 - x_4') &\geq 1 && \text{// sečteme jednotlivé členy} \\
x_1' - x_3' - x_4' + 2 &\geq 1 && \text{// odečteme } 2 \text{ od obou stran} \\
x_1' - x_3' - x_4' &\geq -1
\end{aligned}
$$

Převod 3-SAT na ILP

Příklad:

$$
\begin{aligned}
x_1' + (1 - x_3') + (1 - x_4') &\geq 1 && \text{// sečteme jednotlivé členy} \\
x_1' - x_3' - x_4' + 2 &\geq 1 && \text{// odečteme } 2 \text{ od obou stran} \\
x_1' - x_3' - x_4' &\geq -1 && \text{// vynásobíme obě strany } -1 \\
-x_1' + x_3' + x_4' &\leq 1
\end{aligned}
$$

Převod 3-SAT na ILP

Příklad:

$$
\begin{aligned}
x_1' + (1 - x_3') + (1 - x_4') &\geq 1 && \text{// sečteme jednotlivé členy} \\
x_1' - x_3' - x_4' + 2 &\geq 1 && \text{// odečteme } 2 \text{ od obou stran} \\
x_1' - x_3' - x_4' &\geq -1 && \text{// vynásobíme obě strany } -1 \\
-x_1' + x_3' + x_4' &\leq 1
\end{aligned}
$$

Po doplnění chybějících členů (s koeficienty 0) tedy výsledná nerovnice vypadá takto:

$$(-1) \cdot x_1' + 0 \cdot x_2' + 1 \cdot x_3' + 1 \cdot x_4' \leq 1$$

Převod 3-SAT na ILP

$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$

Po úpravě všech nerovnic tedy dostaneme soustavu:

$$
\begin{array}{rcrcrcrcl}
(-1)\cdot x_1' & + & 0\cdot x_2' & + & 0\cdot x_3' & + & 0\cdot x_4' & \leq & 0 \\
1\cdot x_1' & + & 0\cdot x_2' & + & 0\cdot x_3' & + & 0\cdot x_4' & \leq & 1 \\
0\cdot x_1' & + & (-1)\cdot x_2' & + & 0\cdot x_3' & + & 0\cdot x_4' & \leq & 0 \\
0\cdot x_1' & + & 1\cdot x_2' & + & 0\cdot x_3' & + & 0\cdot x_4' & \leq & 1 \\
0\cdot x_1' & + & 0\cdot x_2' & + & (-1)\cdot x_3' & + & 0\cdot x_4' & \leq & 0 \\
0\cdot x_1' & + & 0\cdot x_2' & + & 1\cdot x_3' & + & 0\cdot x_4' & \leq & 1 \\
0\cdot x_1' & + & 0\cdot x_2' & + & 0\cdot x_3' & + & (-1)\cdot x_4' & \leq & 0 \\
0\cdot x_1' & + & 0\cdot x_2' & + & 0\cdot x_3' & + & 1\cdot x_4' & \leq & 1 \\
(-1)\cdot x_1' & + & 1\cdot x_2' & + & (-1)\cdot x_3' & + & 0\cdot x_4' & \leq & 0 \\
0\cdot x_1' & + & (-1)\cdot x_2' & + & 1\cdot x_3' & + & (-1)\cdot x_4' & \leq & 0 \\
(-1)\cdot x_1' & + & 0\cdot x_2' & + & 1\cdot x_3' & + & 1\cdot x_4' & \leq & 1 \\
1\cdot x_1' & + & 1\cdot x_2' & + & 0\cdot x_3' & + & (-1)\cdot x_4' & \leq & 1
\end{array}
$$

Převod 3-SAT na ILP

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Tuto soustavu můžeme zapsat maticovým zápisem jako:

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \\ -1 & 1 & -1 & 0 \\ 0 & -1 & 1 & -1 \\ -1 & 0 & 1 & 1 \\ 1 & 1 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} x_1' \\ x_2' \\ x_3' \\ x_4' \end{pmatrix} \leq \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Převod 3-SAT na ILP

Je zřejmé, že tuto konstrukci můžeme provést v čase $\mathcal{O}(n^2)$, kde n je velikost formule φ.

Poznámka:
Ve skutečnosti nejvíce času zabere vyplňování matice A nulami.
Není těžké ověřit, že vše ostatní (vyplnění nenulových prvků v matici A a vytvoření vektoru b) je možné provést v čase $\mathcal{O}(n)$.

Převod 3-SAT na ILP

Je zřejmé, že tuto konstrukci můžeme provést v čase $\mathcal{O}(n^2)$, kde n je velikost formule φ.

Poznámka:
Ve skutečnosti nejvíce času zabere vyplňování matice A nulami.
Není těžké ověřit, že vše ostatní (vyplnění nenulových prvků v matici A a vytvoření vektoru b) je možné provést v čase $\mathcal{O}(n)$.

Poznámka:
Není těžké si rozmyslet, jak by vypadal algoritmus, který by vytvářel matici A a vektor b přímo, bez mezikroku s úpravami nerovnic.
Tento mezikrok zavádíme pro lepší pochopení konstrukce.

Převod 3-SAT na ILP

Nyní ještě zbývá ukázat korektnost konstrukce.

Nejprve si všimněme, že vzhledem k tomu, že vytvořená soustava obsahuje pro každé x_i' nerovnice

$$x_i' \geq 0 \qquad\qquad x_i' \leq 1$$

musí jakékoliv řešení celé soustavy (pokud vůbec nějaké existuje) být toho typu, že jednotlivá x_i' nabývají pouze hodnot 0 nebo 1.

Převod 3-SAT na ILP

Každému ohodnocení ν booleovských proměnných $x_1, x_2, \ldots, x_k$ ve formuli φ jednoznačně odpovídá přiřazení hodnot neznámým $x'_1, x'_2, \ldots, x'_k$ ve vytvořené soustavě nerovnic:

$$x'_i = \begin{cases} 0 & \text{když } \nu(x_i) = 0 \\ 1 & \text{když } \nu(x_i) = 1 \end{cases}$$

Převod 3-SAT na ILP

Každému ohodnocení ν booleovských proměnných $x_1, x_2, \ldots, x_k$ ve formuli φ jednoznačně odpovídá přiřazení hodnot neznámým $x'_1, x'_2, \ldots, x'_k$ ve vytvořené soustavě nerovnic:

$$x'_i = \begin{cases} 0 & \text{když } \nu(x_i) = 0 \\ 1 & \text{když } \nu(x_i) = 1 \end{cases}$$

Tento vztah je vzájemně jednoznačný.

Ke každému přiřazení celočíselných hodnot neznámým $x'_1, x'_2, \ldots, x'_k$ takovému, že pro všechna x'_i platí $x'_i \in \{0, 1\}$, existuje odpovídající přiřazení booleovských hodnot ν.

Převod 3-SAT na ILP

Vezměme si nyní nějaké ohodnocení booleovských proměnných ν a jemu odpovídající přiřazení hodnot 0 a 1 neznámým $x'_1, x'_2, \ldots, x'_k$.

Připomeňme, že ve vytvořené soustavě nerovnic odpovídá každé klauzuli $(\ell_1 \vee \ell_2 \vee \ell_3)$ vyskytující se ve formuli φ nerovnice tvaru

$$f_1 + f_2 + f_3 \geq 1$$

kde f_i je tvaru x'_j, pokud $\ell_i = x_j$, nebo $(1 - x'_j)$, pokud $\ell_i = \neg x_j$.

Převod 3-SAT na ILP

Vezměme si nyní nějaké ohodnocení booleovských proměnných ν a jemu odpovídající přiřazení hodnot 0 a 1 neznámým $x'_1, x'_2, \ldots, x'_k$.

Připomeňme, že ve vytvořené soustavě nerovnic odpovídá každé klauzuli $(\ell_1 \vee \ell_2 \vee \ell_3)$ vyskytující se ve formuli φ nerovnice tvaru

$$f_1 + f_2 + f_3 \geq 1$$

kde f_i je tvaru x'_j, pokud $\ell_i = x_j$, nebo $(1 - x'_j)$, pokud $\ell_i = \neg x_j$.

Vidíme, že ať už je literál ℓ_i tvaru x_j nebo $\neg x_j$, platí, že:

- $f_i = 1$, pokud $[\ell_i]_\nu = 1$
- $f_i = 0$, pokud $[\ell_i]_\nu = 0$

Převod 3-SAT na ILP

Hodnota výrazu $f_1 + f_2 + f_3$ při daném přiřazení je tedy počtem literálů v klauzuli $(\ell_1 \vee \ell_2 \vee \ell_3)$, které mají při ohodnocení ν hodnotu 1.

Vzhledem k tomu, že $[\ell_1 \vee \ell_2 \vee \ell_3]_\nu = 1$ právě tehdy, když pro alespoň jeden z literálů ℓ_1, ℓ_2, ℓ_3 platí $[\ell_i]_\nu = 1$, je očividné, že nerovnost

$$f_1 + f_2 + f_3 \geq 1$$

platí při daném přiřazení právě tehdy, když

$$[\ell_1 \vee \ell_2 \vee \ell_3]_\nu = 1\,.$$

Převod 3-SAT na ILP

Tvrzení

Pokud je formule φ splnitelná, pak existuje celočíselné řešení vytvořené soustavy nerovnic.

Důkaz: Jestliže je φ splnitelná, existuje nějaké ohodnocení ν takové, že $[\varphi]_\nu = 1$, tj. takové, kde $[C_i]_\nu = 1$ pro všechny klauzule C_i formule φ.

Z předchozího je zřejmé, že pokud vezmeme jemu odpovídající přiřazení hodnot 0 a 1 neznámým $x'_1, x'_2, \ldots, x'_k$, budou při tomto přiřazení platit všechny vytvořené nerovnice:

- Nerovnice tvaru $x'_i \geq 0$ a $x'_i \leq 1$ proto, že $x_i \in \{0, 1\}$.
- Nerovnice odpovídající klauzulím proto, že při ohodnocení ν má každá klauzule hodnotu 1.

Převod 3-SAT na ILP

Tvrzení

Jestliže existuje řešení vytvořené soustavy nerovnic, pak je formule φ splnitelná.

Důkaz: Je zřejmé, že pokud má soustava nerovnic řešení, tak musí toto řešení pro všechna x_i' splňovat podmínku $x_i' \in \{0, 1\}$.

Tomuto řešení tedy jednoznačně odpovídá nějaké ohodnocení ν a z předchozího je zřejmé, že každá klauzule formule φ při tomto ohodnocení nabývá hodnoty 1, takže platí

$$[\varphi]_\nu = 1$$

a formule φ je tedy splnitelná.

Převod 3-SAT na ILP

Vidíme, že formule φ je splnitelná právě tehdy, když existuje celočíselné řešení k ní vytvořené soustavy nerovnic.

Tím je důkaz korektnosti konstrukce hotov.

Převod 3-SAT na SUBSET-SUM

Problém SUBSET-SUM

Vstup: Sekvence přirozených čísel $a_1, a_2, \ldots, a_n$ a přirozené číslo s?

Otázka: Existuje množina $I \subseteq \{1, 2, \ldots, n\}$ taková, že $\sum_{i \in I} a_i = s$?

Jinak řečeno, ptáme se zda z dané (multi)množiny čísel je možné vybrat podmnožinu, jejíž součet je s.

Příklad: Pro vstup tvořený čísly $3, 5, 2, 3, 7$ a číslem $s = 15$ je odpověď Ano, neboť $3 + 5 + 7 = 15$.

Pro vstup tvořený čísly $3, 5, 2, 3, 7$ a číslem $s = 16$ je odpověď Ne, neboť žádná podmnožina těchto čísel nedává součet 16.

Převod 3-SAT na SUBSET-SUM

Poznámka:
Pořadí čísel $a_1, a_2, \ldots, a_n$ na vstupu není důležité.

Všimněte si však určitého rozdílu oproti tomu, kdybychom problém formulovali tak, že vstupem je množina $\{a_1, a_2, \ldots, a_n\}$ a číslo s:

V množině se čísla neopakují, zatímco v sekvenci se může totéž číslo vyskytnout vícekrát.

Převod 3-SAT na SUBSET-SUM

Problém SUBSET-SUM je speciálním případem **problému batohu** (knapsack problem):

Knapsack problem

Vstup: Sekvence dvojic přirozených čísel $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$ a dvě přirozená čísla s a t.

Otázka: Existuje množina $I \subseteq \{1, 2, \ldots, n\}$ taková, že $\sum_{i \in I} a_i \leq s$ a $\sum_{i \in I} b_i \geq t$?

Převod 3-SAT na SUBSET-SUM

Neformálně můžeme problém batohu formulovat takto:

Máme n předmětů, kde i-tý předmět váží a_i gramů a má cenu b_i Kč. Do batohu se vejdou předměty o maximální celkové váze s gramů.
Otázka zní, zda můžeme z předmětů vybrat podmnožinu, která by se vešla do batohu a měla celkovou cenu alespoň t Kč.

Poznámka:
Zde jsme problém batohu formulovali jako rozhodovací problém.
Běžnější je formulovat tento problém jako optimalizační problém, kde je cílem najít takovou množinu $I \subseteq \{1, 2, \ldots, n\}$, kde hodnota $\sum_{i \in I} b_i$ je maximální, přičemž ovšem musí být dodržena podmínka $\sum_{i \in I} a_i \leq s$,
tj. vybrat předměty s maximální celkovou cenou tak, aby nebyla překročena kapacita batohu.

Převod 3-SAT na SUBSET-SUM

To, že SUBSET-SUM je speciálním případem problému batohu, vidíme z následující (téměř triviální) redukce:

Řekněme, že $a_1, a_2, \ldots, a_n$, s_1 je instance problému SUBSET-SUM. Je očividné, že pro instanci problému batohu, kde máme sekvenci $(a_1, a_1), (a_2, a_2), \ldots, (a_n, a_n)$, $s = s_1$ a $t = s_1$, je odpověď stejná jako pro původní instanci SUBSET-SUM.

Převod 3-SAT na SUBSET-SUM

Pokud chceme studovat složitost problémů jako jsou SUBSET-SUM nebo problém batohu, je dobré si nejprve ujasnit, co považujeme za velikost vstupu.

Asi nejpřirozenější je definovat velikost vstupu jako celkový počet bitů, který potřebujeme k zápisu instance.

Musíme však určit, jakým způsobem jsou na vstupu zadána přirozená čísla – zda binárně (případně v jiné číselné soustavě o základu alespoň 2, např. desítkové nebo šestnáctkové) nebo unárně.

V dalším výkladu budeme předpokládat, že čísla jsou na vstupu zadána **binárně**, tj. že velikost vstupu je úměrná součtu délek binárních zápisů jednotlivých čísel na vstupu.

Převod 3-SAT na SUBSET-SUM

Není těžké se přesvědčit, že SUBSET-SUM i problém batohu (jeho rozhodovací varianta) patří do třídy NPTIME:

- Nedeterministický algoritmus nejprve nedeterministicky zvolí podmnožinu prvků sekvence na vstupu a poté (deterministicky) ověří, zda splňuje splňuje danou podmínku (resp. podmínky).

  Je zřejmé, že toto ověření je možné provést v čase polynomiálním vzhledem k velikosti instance.

Ukážeme si, že problém SUBSET-SUM (a tím pádem i problém batohu) je NP-těžký.

NP-obtížnost problému SUBSET-SUM ukážeme pomocí redukce z problému 3-SAT.

Převod 3-SAT na SUBSET-SUM

Předpokládejme, že máme dánu instanci problému 3-SAT, tj. formuli φ tvaru

$$C_1 \wedge C_2 \wedge \cdots \wedge C_k$$

kde $C_1, C_2, \ldots, C_k$ jsou jednotlivé **klauzule**, přičemž každá klauzule C_j je tvaru

$$(L_{j,1} \vee L_{j,2} \vee L_{j,3})$$

kde $L_{j,1}, L_{j,2}, L_{j.3}$ jsou jednotlivé **literály** v této klauzuli, přičemž každý z těchto literálů je buď tvaru x_i nebo tvaru $\neg x_i$, kde x_i je nějaká booleovská **proměnná**.

Předpokládejme dále, že množina všech proměnných, které se vyskytují ve formuli φ, je

$$\{x_1, x_2, \ldots, x_m\}$$

Převod 3-SAT na SUBSET-SUM

K dané formuli φ sestrojíme instanci problému SUBSET-SUM, která bude obsahovat multimnožinu čísel X a číslo s.

Čísla v multimnožině X i číslo s budeme popisovat tak, že popíšeme, jak budou vypadat jednotlivé číslice daného čísla zapsaného v číselné soustavě o základu $d = 10$.

Poznámka: Číselný základ $d = 10$ jsme zvolili pro větší názornost, konstrukce bude stejně dobře fungovat i s libovolným jiným základem $d \geq 7$.

Převod 3-SAT na SUBSET-SUM

Ještě si připomeňme, že $b_k b_{k-1} b_{k-2} \cdots b_1 b_0$ je v číselné soustavě o základu d zápisem čísla

$$\sum_{i=0}^{k} b_i \cdot d^i$$

V tomto zápise $b_k, b_{k-1}, \ldots, b_0$ reprezentují jednotlivé číslice, přičemž pro všechna b_i platí $0 \leq b_i < d$.

Budeme-li dále mluvit o číslici na pozici i, máme tím na mysli hodnotu b_i ve výše uvedeném zápise.

Příklad: V čísle 732594 je na pozici 5 číslice 7, na pozici 4 číslice 3 a na pozici 0 číslice 4.

Převod 3-SAT na SUBSET-SUM

Připomeňme, že předpokládáme, že formule φ obsahuje proměnné $x_1, x_2, \ldots, x_m$ a skládá se z klauzulí $C_1, C_2, \ldots, C_k$.

V instanci problému SUBSET-SUM, kterou budeme vytvářet, budeme jednotlivá čísla zapisovat jako $(m + k)$-místná čísla v číselné soustavě o základu $d = 10$ s tím, že:

- Každé proměnné x_i přidělíme jednu pozici v těchto číslech (každé jinou).
- Každé klauzuli C_j přidělíme jednu ze zbylých pozic (každé jinou).

Převod 3-SAT na SUBSET-SUM

Jak konkrétně toto přiřazení provedeme, není pro další konstrukci příliš podstatné.

Jednou z možností je například:

- Přiřadit proměnným $x_1, x_2, \ldots, x_m$ pozice $0, 1, \ldots, m-1$ (tj. proměnné x_i je přiřazena pozice $i-1$).
- Přiřadit klauzulím $C_1, C_2, \ldots, C_k$ pozice $m, m+1, \ldots, m+k-1$ (tj. klauzuli C_j je přiřazena pozice $j+m-1$).

V dalším výkladu budeme tedy hovořit o pozici odpovídající proměnné x_i či o pozici odpovídající klauzuli C_j.

Převod 3-SAT na SUBSET-SUM

Pro přehlednější výklad zavedeme následující značení:
Řekněme, že y je nějaké číslo, a že toto číslo zapíšeme v číselné soustavě o základu d.

- $y[x_i]$ bude označovat číslici na pozici odpovídající proměnné x_i.
- $y[C_j]$ bude označovat číslici na pozici odpovídající klauzuli C_j.

Ve výše popsaném konkrétním přiřazení pozic bude tedy platit, že číslo y bude zapsáno jako

$$y[C_k]\, y[C_{k-1}] \cdots y[C_1]\, y[x_m]\, y[x_{m-1}] \cdots y[x_1]$$

neboli

$$y = \sum_{i=1}^{m} y[x_i] \cdot d^{i-1} + \sum_{j=1}^{k} y[C_j] \cdot d^{j+m-1}$$

Převod 3-SAT na SUBSET-SUM

Nyní popíšeme, jak vytvořit instanci problému SUBSET-SUM s multimnožinou X a číslem s:

- Pro každou proměnnou x_i z množiny $\{x_1, x_2, \ldots, x_m\}$ přidáme do X dvojici čísel y_i a y_i', jejichž číslice jsou definovány následovně:
  - $y_i[x_i] = y_i'[x_i] = 1$
  - Pro $x_j \neq x_i$ je $y_i[x_j] = y_i'[x_j] = 0$
  - Pro každou klauzuli C_j je hodnota $y_i[C_j]$ rovna počtu výskytů literálu x_i v klauzuli C_j.
  - Pro každou klauzuli C_j je hodnota $y_i'[C_j]$ rovna počtu výskytů literálu $\neg x_i$ v klauzuli C_j.

Převod 3-SAT na SUBSET-SUM

- Pro každou klauzuli C_j z množiny $\{C_1, C_2, \ldots, C_k\}$ přidáme do X dvojici čísel z_j a z_j', jejichž číslice jsou definovány následovně:
  - $z_j[C_j] = 1$
  - $z_j'[C_j] = 2$
  - Všechny ostatní číslice v z_j i z_j' jsou 0.
- Vytvoříme hodnotu s, kde:
  - $s[x_i] = 1$ pro všechny proměnné x_i.
  - $s[C_j] = 4$ pro všechny klauzule C_j.

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Příklad instance problému SUBSET-SUM odpovídající níže uvedené formuli:

	C_4	C_3	C_2	C_1	x_4	x_3	x_2	x_1	
y_1	0	1	0	1	0	0	0	1	x_1
y_1'	1	0	0	0	0	0	0	1	$\neg x_1$
y_2	0	0	1	0	0	0	1	0	x_2
y_2'	1	0	0	1	0	0	1	0	$\neg x_2$
y_3	0	0	0	1	0	1	0	0	x_3
y_3'	0	1	1	0	0	1	0	0	$\neg x_3$
y_4	1	0	1	0	1	0	0	0	x_4
y_4'	0	1	0	0	1	0	0	0	$\neg x_4$
z_1	0	0	0	1	0	0	0	0	C_1
z_1'	0	0	0	2	0	0	0	0	C_1
z_2	0	0	1	0	0	0	0	0	C_2
z_2'	0	0	2	0	0	0	0	0	C_2
z_3	0	1	0	0	0	0	0	0	C_3
z_3'	0	2	0	0	0	0	0	0	C_3
z_4	1	0	0	0	0	0	0	0	C_4
z_4'	2	0	0	0	0	0	0	0	C_4
s	4	4	4	4	1	1	1	1	

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

Převod 3-SAT na SUBSET-SUM

Výsledná instance vytvořená k formuli

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$$

bude tedy vypadat takto:

$$X = \{1010001, 10000001, 100010, 10010010, 10100, 1100100,$$
$$10101000, 1001000, 10000, 20000, 100000, 200000, 1000000,$$
$$2000000, 10000000, 20000000\}$$

$$s = 44441111$$

Převod 3-SAT na SUBSET-SUM

Všimněme si ještě, že výše popsanou konstrukci můžeme stručně popsat pomocí následujících vztahů:

$$y_i = d^{i-1} + \sum_{j=1}^{k} c(i,j) \cdot d^{j+m-1} \qquad y_i' = d^{i-1} + \sum_{j=1}^{k} c'(i,j) \cdot d^{j+m-1}$$

$$z_j = d^{j+m-1} \qquad z_j' = 2 \cdot d^{j+m-1}$$

$$s = \sum_{i=1}^{m} d^{i-1} + \sum_{j=1}^{k} 4 \cdot d^{j+m-1}$$

kde $c(i,j)$ udává počet výskytů literálu x_i v klauzuli C_j a $c'(i,j)$ počet výskytů literálu $\neg x_i$ v klauzuli C_j.

Převod 3-SAT na SUBSET-SUM

Je zřejmé, že konstrukci je možné provést v polynomiálním čase.

Zbývá ukázat, že konstrukce je korektní, tedy, že z X je možné vybrat podmožinu, jejíž prvky dávají součet s, právě tehdy, když je původní formule φ splnitelná.

Převod 3-SAT na SUBSET-SUM

Než přistoupíme k vlastnímu důkazu korektnosti, všimněme si nejprve, jak vypadají jednotlivé sloupce číslic, když zapíšeme čísla v X pod sebou:

- Ve sloupcích odpovídajících proměnným x_i jsou vždy právě dvě jedničky (v y_i a y_i').
- Ve sloupcích odpovídajících klauzulím C_j je součet v daném sloupci přes všechna y_i a y_i' vždy roven 3 (tj. je to celkový počet literálů v klauzuli C_j), a dále z_j a z_j' obsahují vždy číslice 1 a 2
- Všechny ostatní hodnoty jsou 0.

Vidíme, že součet v každém sloupci je maximálně 6, a že vzhledem k tomu, že $d \geq 7$, nikdy při součtu žádné podmnožiny nedojde k přenosu z jednoho sloupce do druhého.

Převod 3-SAT na SUBSET-SUM

Předpokládejme nejprve, že formule φ je splnitelná.

Pak tedy existuje nějaké ohodnocení booleovských proměnných ν, při kterém $[\varphi]_\nu = 1$.

Budeme vytvářet (multi)množinu $Y \subseteq X$ takovou, aby součet hodnot v Y byl roven s:

- Pro každou proměnnou x_i takovou, že $[x_i]_\nu = 1$, přidáme do Y číslo y_i.
- Pro každou proměnnou x_i takovou, že $[x_i]_\nu = 0$, přidáme do Y číslo y_i'.

Převod 3-SAT na SUBSET-SUM

Když není sečteme hodnoty čísel, které jsme zatím přidali do Y, budou součty v jednotlivých sloupcích vypadat takto:

- Ve sloupcích odpovídajících proměnným x_i bude součty vždy 1, neboť do Y jsme dali vždy právě jednu z hodnot y_i, y_i'.

  V těchto sloupcích tedy součet souhlasí s požadovanou hodnotou s.

- Ve sloupcích odpovídajících klauzulím C_j bude vždy součet roven počtu literálů v dané klauzuli, které mají při ohodnocení ν hodnotu 1.

  Při ohodnocení ν pro všechny klauzule C_j platí, že $[C_j]_\nu = 1$ a tedy alespoň jeden literál v klauzuli C_j musí mít při ohodnocení ν hodnotu 1.

  Součet v daném sloupci tedy bude číslo v intervalu 1 až 3.

Převod 3-SAT na SUBSET-SUM

Pro každý sloupec odpovídající nějaké klauzuli C_j přidáme do Y jedno nebo případně obě z čísel z_j, z_j' tak, abychom dorovnali součet v tomto sloupci na 4.

(Všiměte si, že vzhledem k tomu, že součet je v intervalu 1 až 3, tak je toto dorovnání na součet 4 vždy možné.)

Čísla z_j i z_j' mají na všech ostatních pozicích hodnoty 0, takže součty v ostatních sloupcích neovlivní.

Vidíme tedy, že pokud je formule φ splnitelná, je možné z X vybrat podmnožinu, která dává součet s.

Převod 3-SAT na SUBSET-SUM

Předpokládejme nyní, že existuje nějaká (multi)množina $Y \subseteq X$, jež dává součet s.

Je zřejmé, že pro každé $i \in \{1, 2, \ldots, m\}$ musí Y obsahovat právě jednu z hodnot y_i, y_i'.

Zvolme tedy následující ohodnocení ν:

- Pokud Y obsahuje y_i, položme $\nu(x_i) = 1$.
- Pokud Y obsahuje y_i', položme $\nu(x_i) = 0$.

Podobně jako v předchozím případě udávají součty ve sloupcích odpovídajících klauzulím po odečtení případných z_j a z_j', která se mohou v Y vyskytovat, počty literálů v dané klauzuli C_j, které mají při ohodnocení ν hodnotu 1.

Převod 3-SAT na SUBSET-SUM

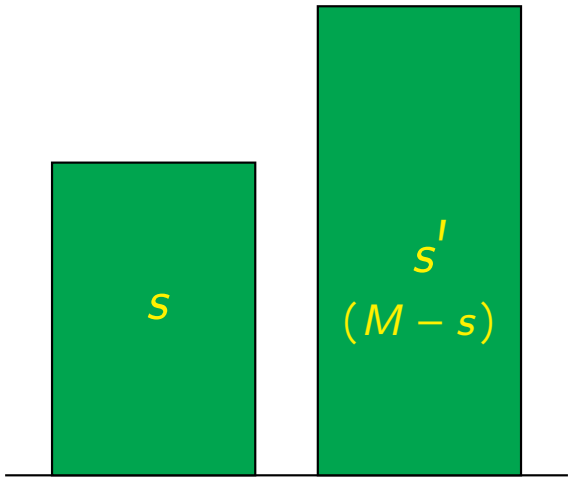
Pro každou klauzuli C_j musí být tento počet roven mimálně 1, protože pokud by byl 0, nebylo by ho možné pomocí z_i a z_i' dorovnat na 4.

V libovolné klauzuli C_j má tedy při ohodnocení ν alespoň jeden literál hodnotu 1, takže $[\varphi]_\nu = 1$ a φ je tedy splnitelná.

Tím je důkaz korektnosti hotov.

Převod 3-SAT na SUBSET-SUM

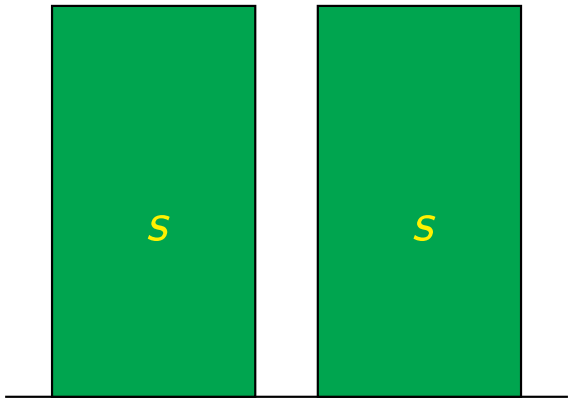
V problému SUBSET-SUM jde o to, rozdělit čísla v dané multimnožině do dvou podmnožin takových, že součet čísel v jedné je s a součet v druhé je $s' = M - s$, kde M je součet všech čísel v dané multimnožině.



Je očividné, že podmožinu se součtem s je možné vybrat právě tehdy, když je možné vybrat podmožinu se součtem s'.

Převod 3-SAT na SUBSET-SUM

Speciálním případem problému SUBSET-SUM je případ, kdy $s = s'$, kdy je cílem rozdělit zadanou multimnožinu do dvou podmnožin se stejně velkým součtem:



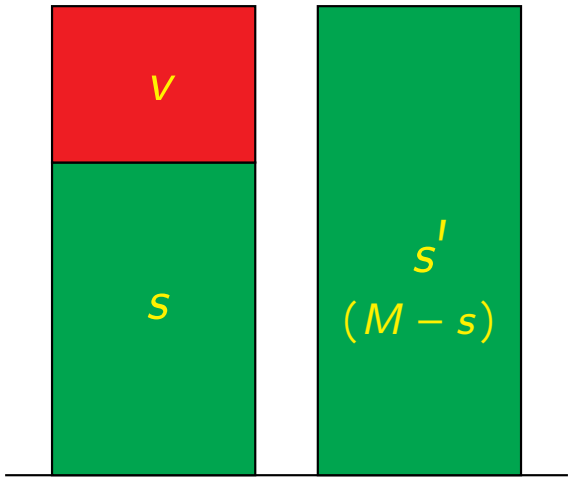
SUBSET-SUM-1/2

Vstup: Sekvence přirozených čísel $a_1, a_2, \ldots, a_n$.

Otázka: Existuje množina $I \subseteq \{1, 2, \ldots, n\}$ taková, že $\sum_{i \in I} a_i = \frac{1}{2} \sum_{i=1}^{n} a_i$?

Převod 3-SAT na SUBSET-SUM

I tento problém SUBSET-SUM-1/2 je NP-těžký, jak je vidět z následující snadné redukce ze SUBSET-SUM:



Do multimnožiny X přidáme nový prvek $v = s' - s$ (pokud $s < s'$) nebo $v = s - s'$ (pokud $s > s'$).
(Pokud $s = s'$, nemusíme dělat nic.)

Převod 3-SAT na SUBSET-SUM

Je zřejmé, že pokud multimnožinu X rozšířenou o nový prvek v rozdělíme na dvě podmnožiny se stejným součtem, musí tento prvek padnout do jedné z těchto podmnožin, a po jeho odebrání nám zbydou podmnožiny se součty s a s'.

Vidíme tedy, že řešení takto vytvořené instrance SUBSET-SUM-1/2 existuje právě tehdy, když existuje řešení původní instance SUBSET-SUM.

NP-úplné problémy

Známou knihou zabývající se problematikou NP-úplných problémů je

> Michael R. Garey, David S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.

Kniha mimo jiné obsahuje katalog různých NP-úplných problémů:

- Je v něm uvedeno přes 300 problémů.
- U jednotlivých problémů jsou uvedeny jejich definice, odkazy na literaturu a poznámky týkající se například toho, které varianty problému jsou řešitelné v polynomiálním čase a které už jsou NP-těžké, apod.

NP-úplné problémy

- Tento katalog je rozdělen do následujících sekcí podle oblastí, do kterých uvedené problémy spadají:
  - Graph Theory
  - Network Design
  - Sets and Partitions
  - Storage and Retrieval
  - Sequencing and Scheduling
  - Mathematical Programming
  - Algebra and Number Theory
  - Games and Puzzles
  - Logic
  - Automata and Language Theory
  - Program Optimization
  - Miscellaneous
  - Open Problems