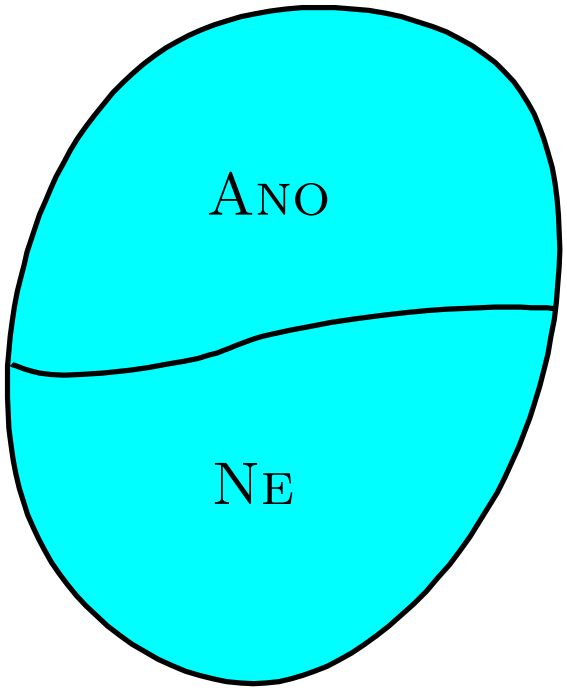
NP-úplné problémy

Polynomiální převody mezi problémy

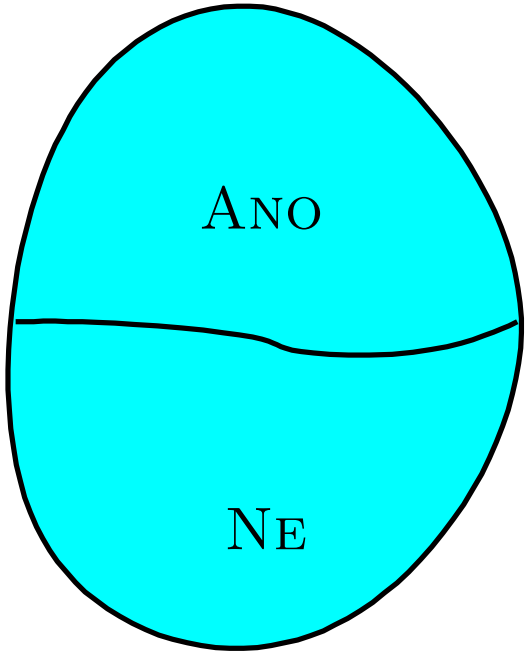
Problém P_1 je **polynomiálně převeditelný** na problém P_2, jestliže existuje algoritmus Alg s polynomiální časovou složitostí, který převádí problém P_1 na problém P_2.

Polynomiální převody mezi problémy

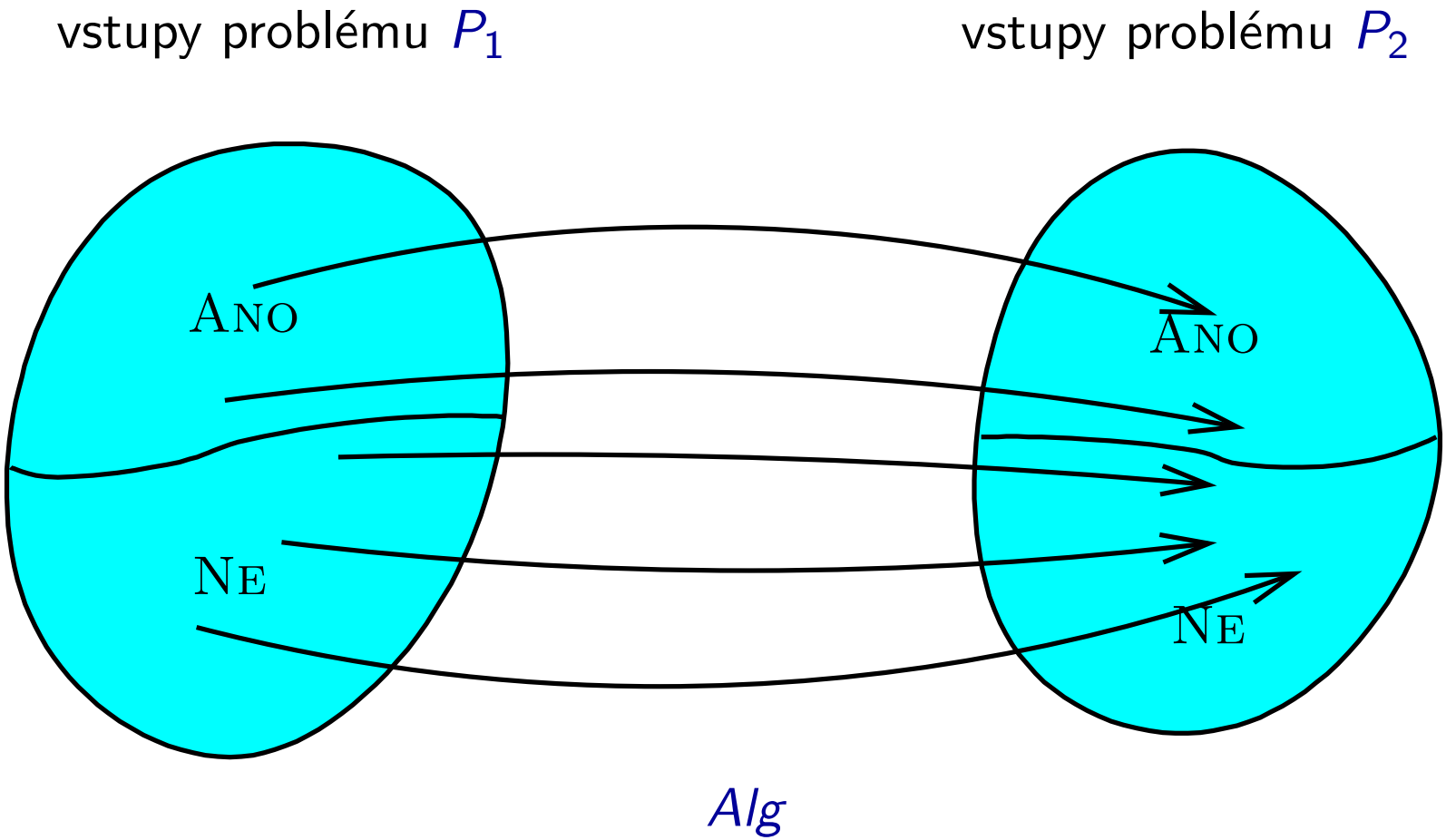
vstupy problému P_1



vstupy problému P_2



Polynomiální převody mezi problémy



Polynomiální převody mezi problémy

Řekněme, že problém P_1 je polynomiálně převeditelný na problém P_2, tj. existuje polynomiální algoritmus Alg realizující tento převod.

Pokud pro problém P_2 existuje polynomiální algoritmus, pak i pro problém P_1 existuje polynomiální algoritmus.

Řešení problému P_1 pro vstup w:

- Zavoláme Alg se vstupem w, vrátí nám hodnotu $Alg(w)$.
- Zavoláme algoritmus řešící problém P_2 se vstupem $Alg(w)$. Hodnotu, kterou nám vrátí, vypíšeme jako výsledek.

Z toho plyne:

Pokud neexistuje polynomiální algoritmus pro problém P_1, tak neexistuje ani polynomiální algoritmus pro problém P_2.

Polynomiální převody mezi problémy

Existuje velká skupina algoritmických problémů označovaných jako **NP-úplné** problémy, které:

- patří do třídy NPTIME, tj. jsou řešitelné v polynomiálním čase **nedeterministickým** algoritmem
- jsou tedy řešitelné v exponenciálním čase
- není pro ně znám žádný algoritmus s polynomiální časovou složitostí
- na druhou stranu není ani dokázáno, že daný pro daný problém nemůže algoritmus s polynomiální časovou složitostí existovat
- jsou všechny navzájem polynomiálně převeditelné

Poznámka: Toto není definice NP-úplných problémů. Ta bude uvedena později.

Problém SAT

Typickým příkladem NP-úplného problému je problém SAT:

SAT (splnitelnost booleovských formulí)

Vstup: Booleovská formule φ.

Otázka: Je φ splnitelná?

Příklad:

Formule $\varphi_1 = x_1 \wedge (\neg x_2 \vee x_3)$ je splnitelná:
např. při ohodnocení v, kde $v(x_1) = 1$, $v(x_2) = 0$, $v(x_3) = 1$, je formule φ_1 pravdivá.

Formule $\varphi_2 = (x_1 \wedge \neg x_1) \vee (\neg x_2 \wedge x_3 \wedge x_2)$ není splnitelná:
je nepravdivá při každém ohodnocení v.

Problém 3-SAT

3-SAT je varianta problému SAT, ve které se omezujeme na formule určitého speciálního typu:

3-SAT

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

Otázka: Je φ splnitelná?

Problém 3-SAT

Připomenutí některých pojmů:

- **Literál** je formule tvaru x nebo $\neg x$, kde x je atomický výrok.
- **Klauzule** je disjunkce literálů.

  Příklady: $x_1 \vee \neg x_2$ $\neg x_5 \vee x_8 \vee \neg x_{15} \vee \neg x_{23}$ x_6
- Formule je v **konjunktivní normální formě (KNF)**, jestliže je konjunkcí klauzulí.

  Příklad: $(x_1 \vee \neg x_2) \wedge (\neg x_5 \vee x_8 \vee \neg x_{15} \vee \neg x_{23}) \wedge x_6$

V případě problému 3-SAT tedy vyžadujeme, aby formule φ byla v KNF a navíc, aby každá klauzule obsahovala právě tři literály.

Příklad:

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

Problém 3-SAT

Následující formule je splnitelná:

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

Je pravdivá např. při ohodnocení v, kde

$$v(x_1) = 0$$
$$v(x_2) = 1$$
$$v(x_3) = 0$$
$$v(x_4) = 1$$

Naproti tomu následující formule není splnitelná:

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

Polynomiální převody mezi problémy

Ukážeme si příklad polynomiálního převodu problému 3-SAT na problém nezávislé množiny (IS).

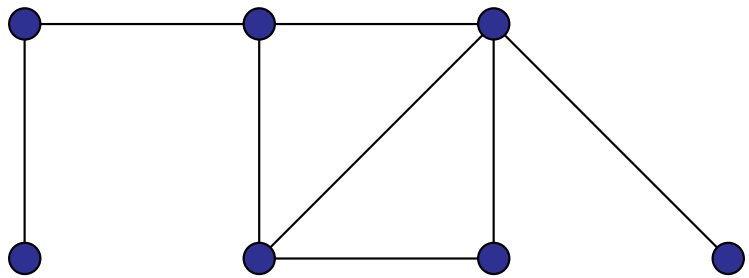
Poznámka: Jak 3-SAT, tak IS jsou příklady NP-úplných problémů.

Problém nezávislé množiny (IS)

Problém nezávislé množiny (IS)

Vstup: Neorientovaný graf G, číslo k.

Otázka: Existuje v grafu G nezávislá množina velikosti k?



$k = 4$

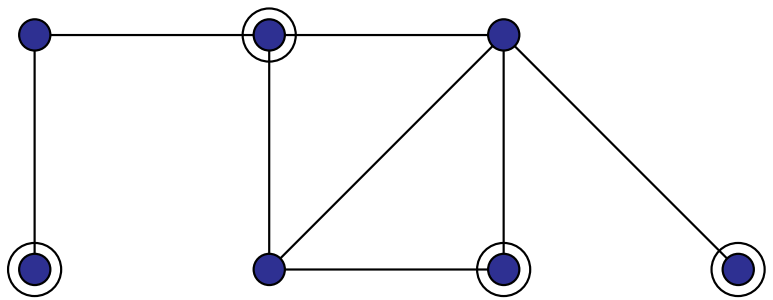
Poznámka: **Nezávislá množina** v grafu je podmnožina vrcholů grafu taková, že žádné dva vrcholy z této podmnožiny nejsou spojeny hranou.

Problém nezávislé množiny (IS)

Problém nezávislé množiny (IS)

Vstup: Neorientovaný graf G, číslo k.

Otázka: Existuje v grafu G nezávislá množina velikosti k?

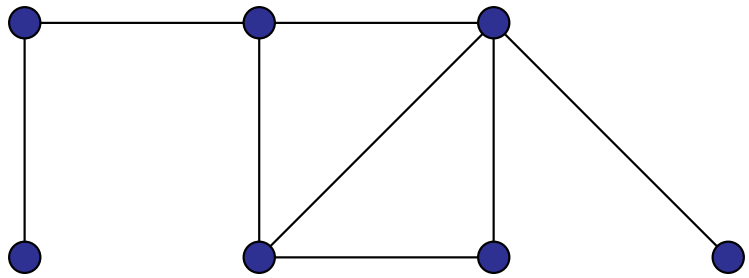


$k = 4$

Poznámka: **Nezávislá množina** v grafu je podmnožina vrcholů grafu taková, že žádné dva vrcholy z této podmnožiny nejsou spojeny hranou.

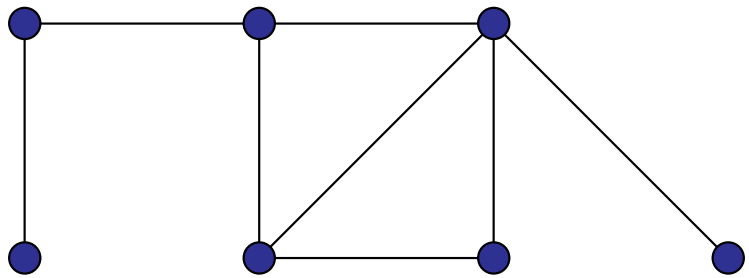
Problém nezávislé množiny (IS)

Příklad instance, kde je odpověď Ano:



$k = 4$

Příklad instance, kde je odpověď Ne:



$k = 5$

Převod 3-SAT na IS

Popíšeme (polynomiální) algoritmus, který bude mít následující vlastnosti:

- **Vstup:** Libovolná instance problému 3-SAT, tj. formule φ v konjunktivní normální formě, kde každá klauzule obsahuje právě tři literály.
- **Výstup:** Instance problému IS, tj. neorientovaný graf G a číslo k.
- Navíc bude pro libovolný vstup (tj. pro libovolnou formuli φ ve výše uvedeném tvaru) zaručeno následující:

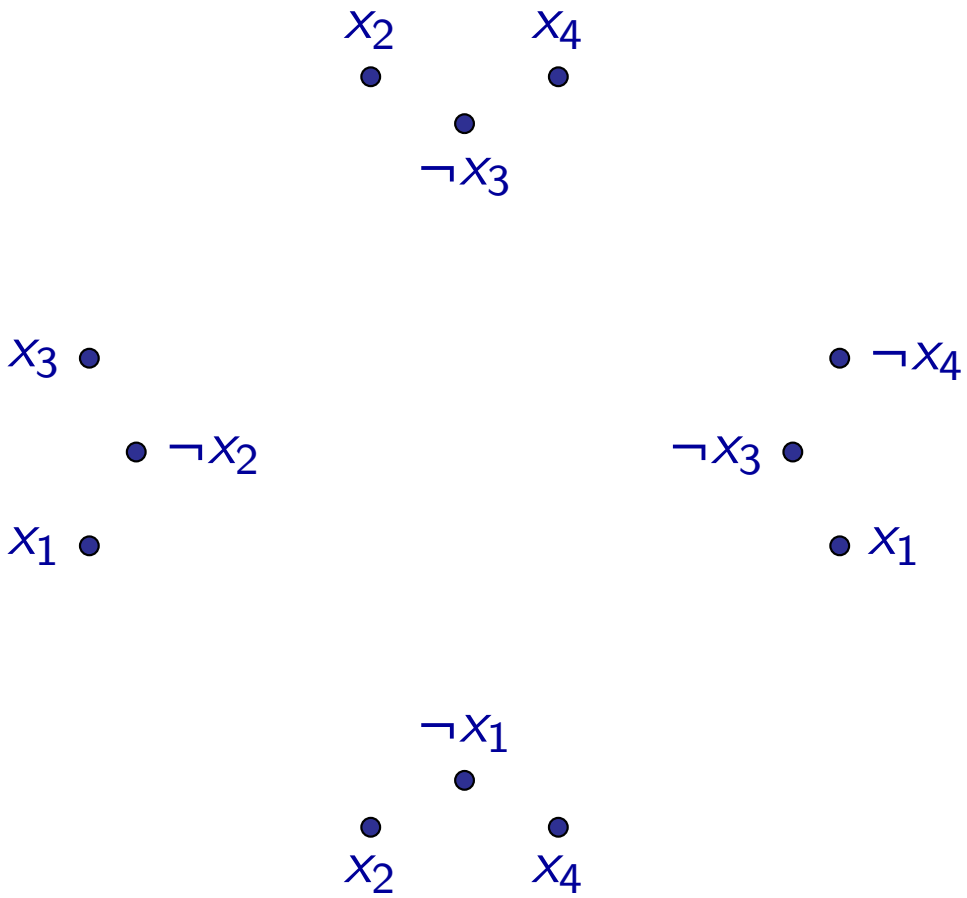
  V grafu G bude existovat nezávislá množina velikosti k právě tehdy, když formule φ bude splnitelná.

Převod 3-SAT na IS

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

Převod 3-SAT na IS

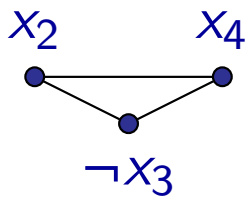
$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$

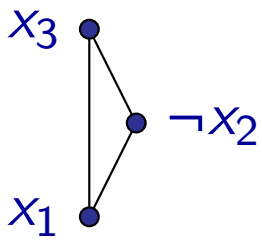


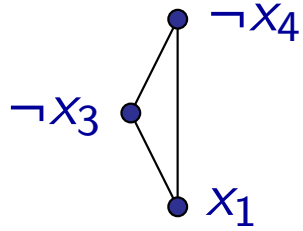
Pro každý výskyt literálu přidáme do grafu jeden vrchol.

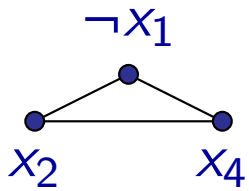
Převod 3-SAT na IS

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





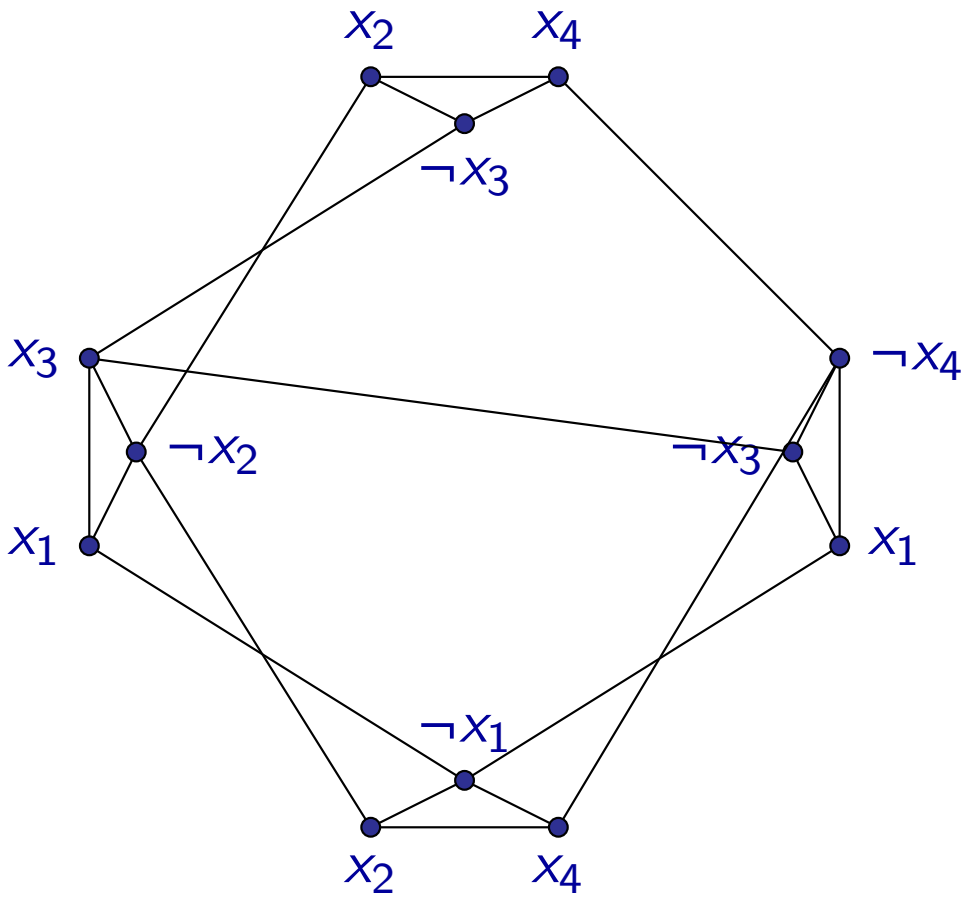




Vrcholy odpovídající výskytům literálů patřícím do stejné klauzule spojíme hranami.

Převod 3-SAT na IS

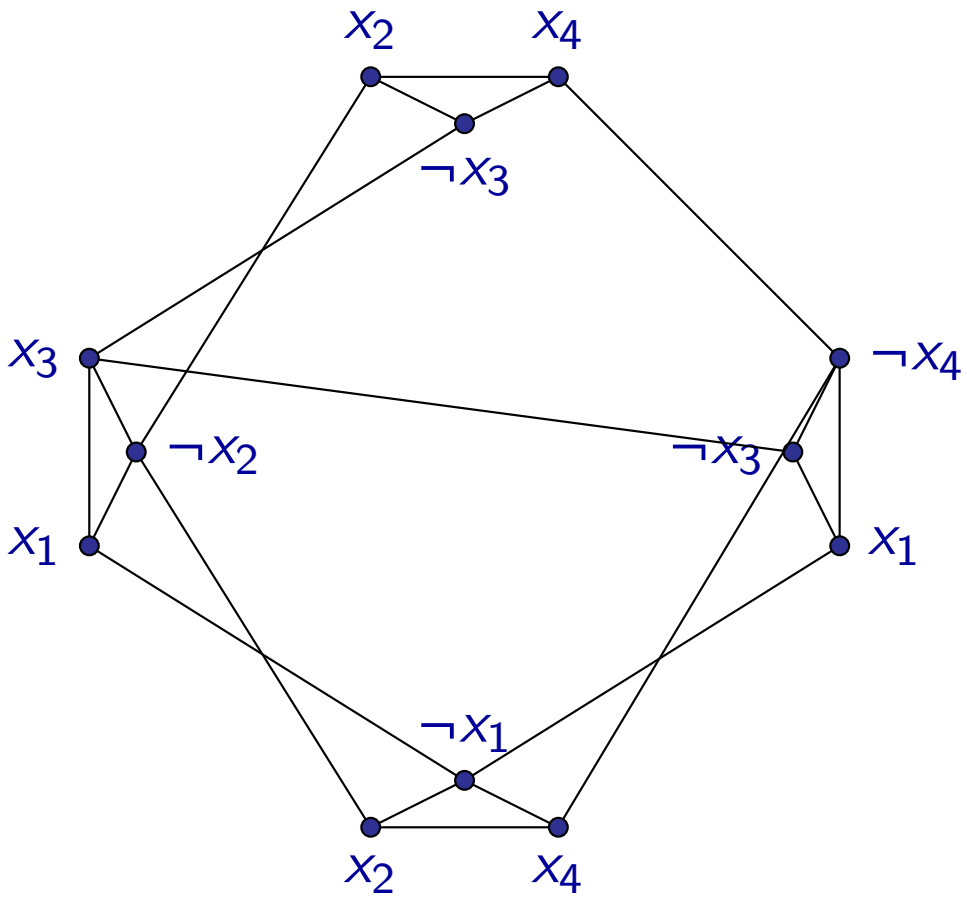
$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$



Dvojice vrcholů odpovídající literálům x_i a $\neg x_i$ spojíme hranami.

Převod 3-SAT na IS

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

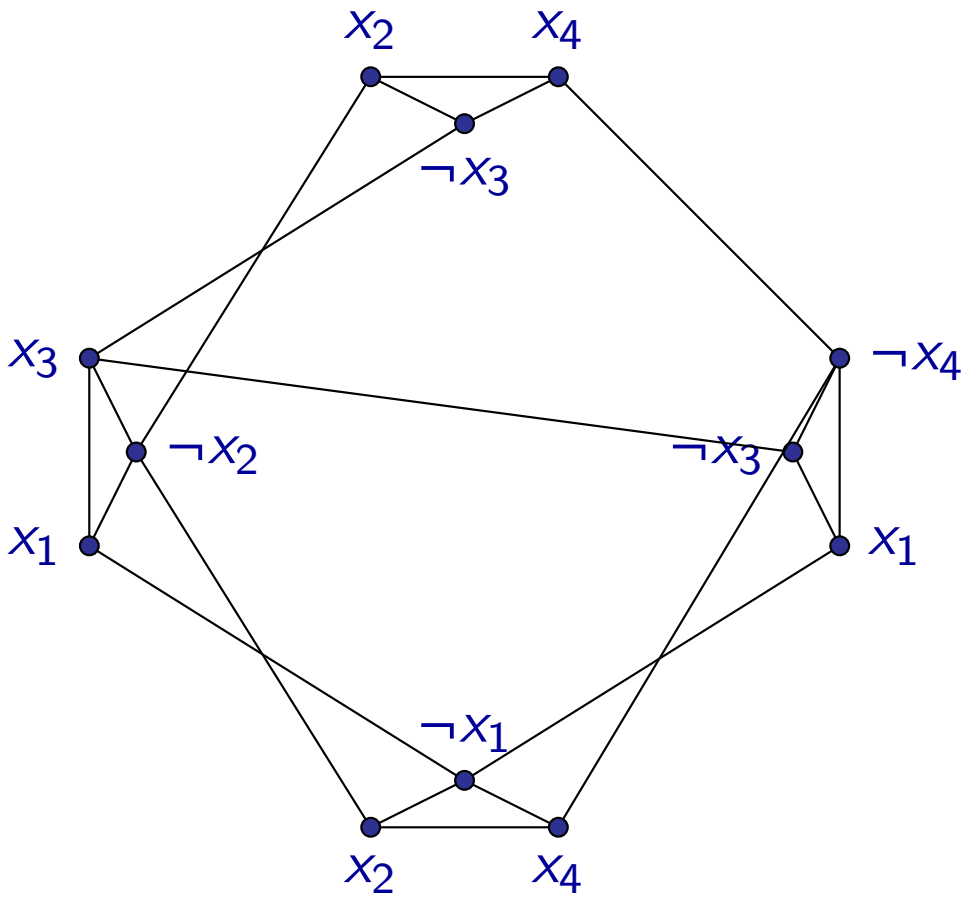


$k = 4$

Číslo k položíme rovno počtu klauzulí.

Převod 3-SAT na IS

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



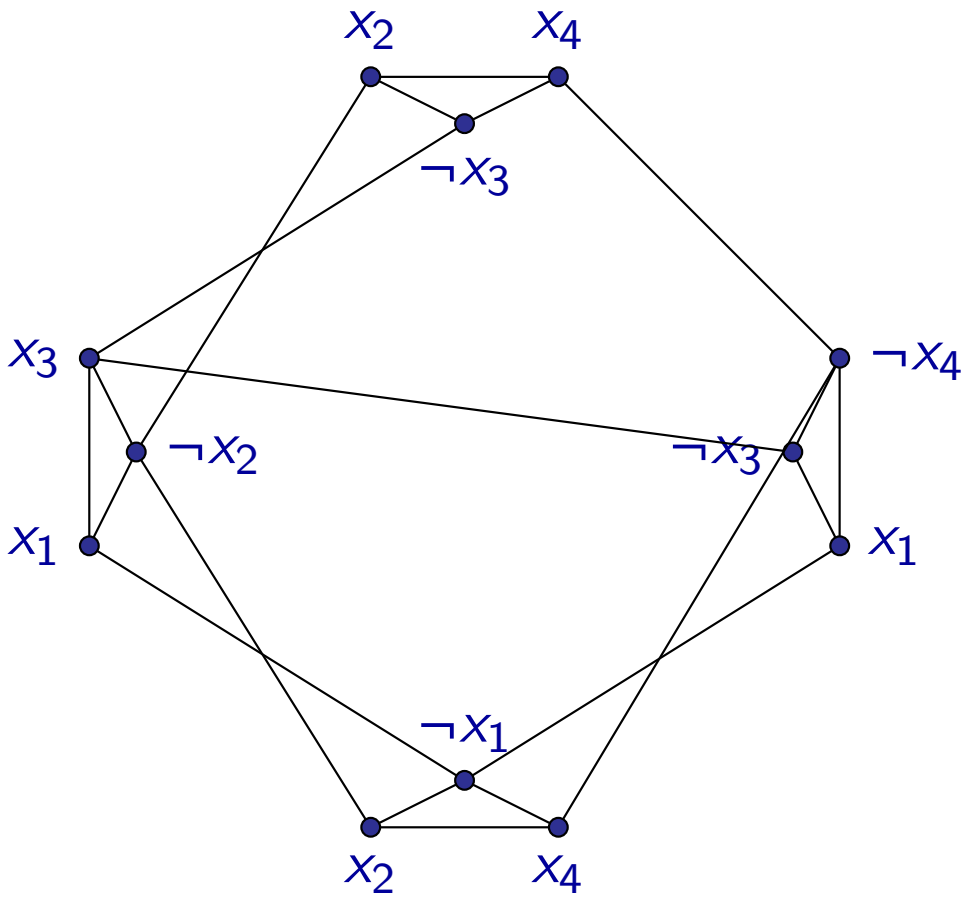
$k = 4$

Vytvořený graf a číslo k vydá algoritmus jako výstup.

Převod 3-SAT na IS

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

$v(x_1) = 1$
$v(x_2) = 1$
$v(x_3) = 0$
$v(x_4) = 1$



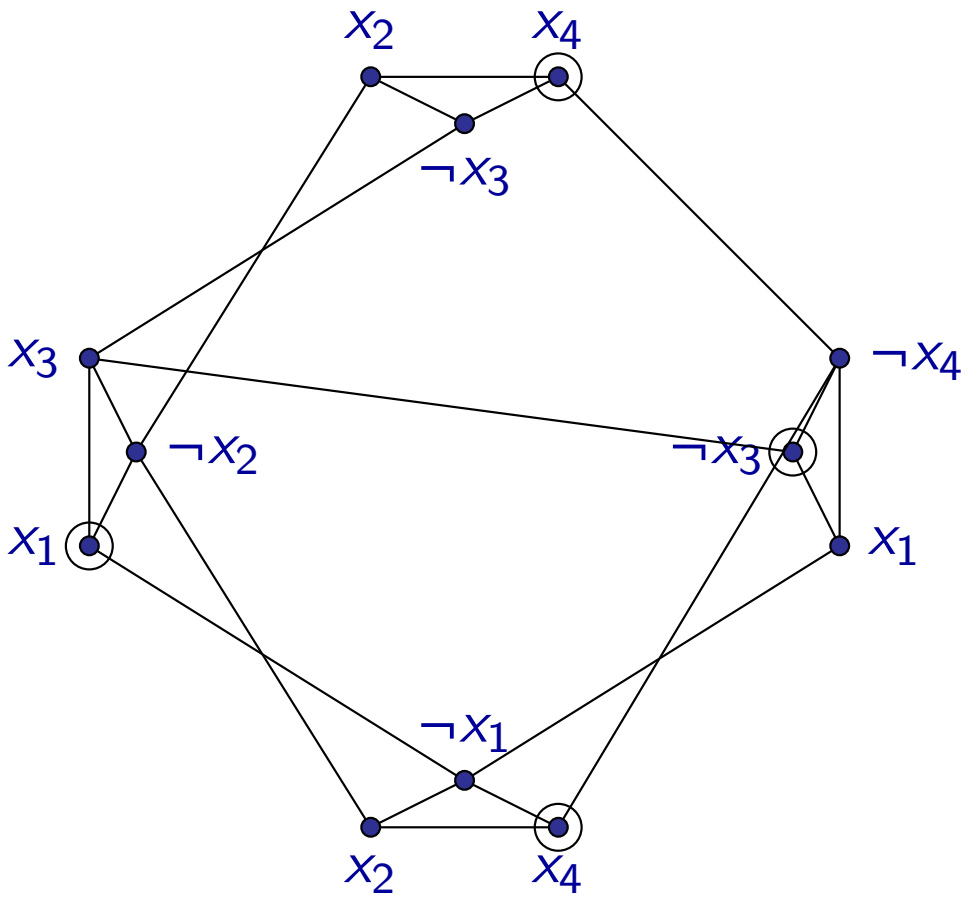
$k = 4$

Jestliže je formule φ splnitelná, existuje ohodnocení v, při kterém má v každé klauzuli alespoň jeden literál hodnotu 1.

Převod 3-SAT na IS

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

$v(x_1) = 1$
$v(x_2) = 1$
$v(x_3) = 0$
$v(x_4) = 1$



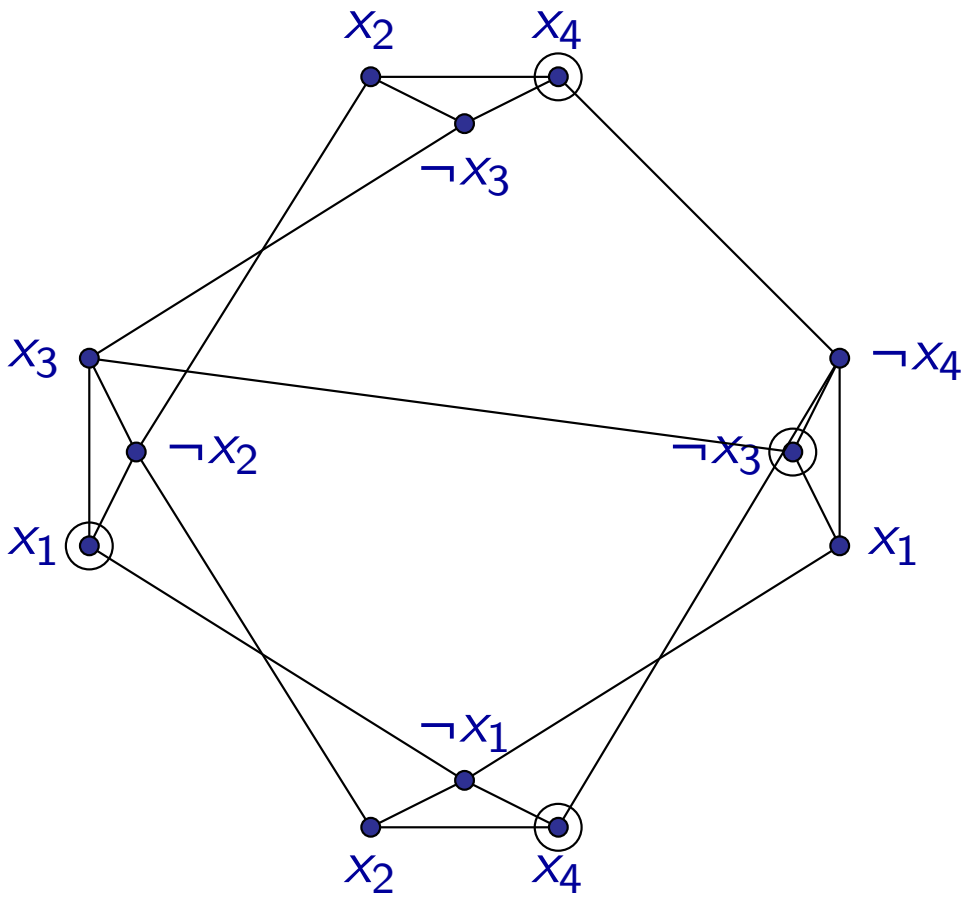
$k = 4$

Z každé klauzule vybereme jeden literál, který má při ohodnocení v hodnotu 1, a do nezávislé množiny přidáme odpovídající vrchol.

Převod 3-SAT na IS

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

$v(x_1) = 1$
$v(x_2) = 1$
$v(x_3) = 0$
$v(x_4) = 1$



$k = 4$

Lehce ověříme, že vybrané vrcholy tvoří nezávislou množinu.

Převod 3-SAT na IS

Vybrané vrcholy tvoří nezávislou množinu, protože:

- Z každé trojice vrcholů odpovídající jedné klauzuli byl vybrán jen jeden vrchol.
- Nemohly být současně vybrány vrcholy označené x_i a $\neg x_i$.
  (Při daném ohodnocení v má hodnotu 1 jen jeden z nich.)

Převod 3-SAT na IS

Na druhou stranu, pokud v grafu G existuje nezávislá množina velikosti k, musí určitě splňovat následující vlastnosti:

- Z každé trojice vrcholů odpovídající jedné klauzuli musí být vybrán nejvýše jeden vrchol.

  Protože je ale klauzulí k a je vybráno k vrcholů, musí být z každé takové trojice vybrán právě jeden.

- Nemohly být současně vybrány vrcholy označené x_i a $\neg x_i$.

Ohodnocení tedy zvolíme podle vybraných vrcholů, protože z předchozího vyplývá, že nehrozí, že by neexistovalo.
(Zbylým proměnným přiřadíme libovolné hodnoty.)

Při daném ohodnocení má formule φ určitě hodnotu 1, neboť v každé klauzuli má hodnotu 1 alespoň jeden literál.

Převod 3-SAT na IS

Popsaný algoritmus je určitě polynomiální:

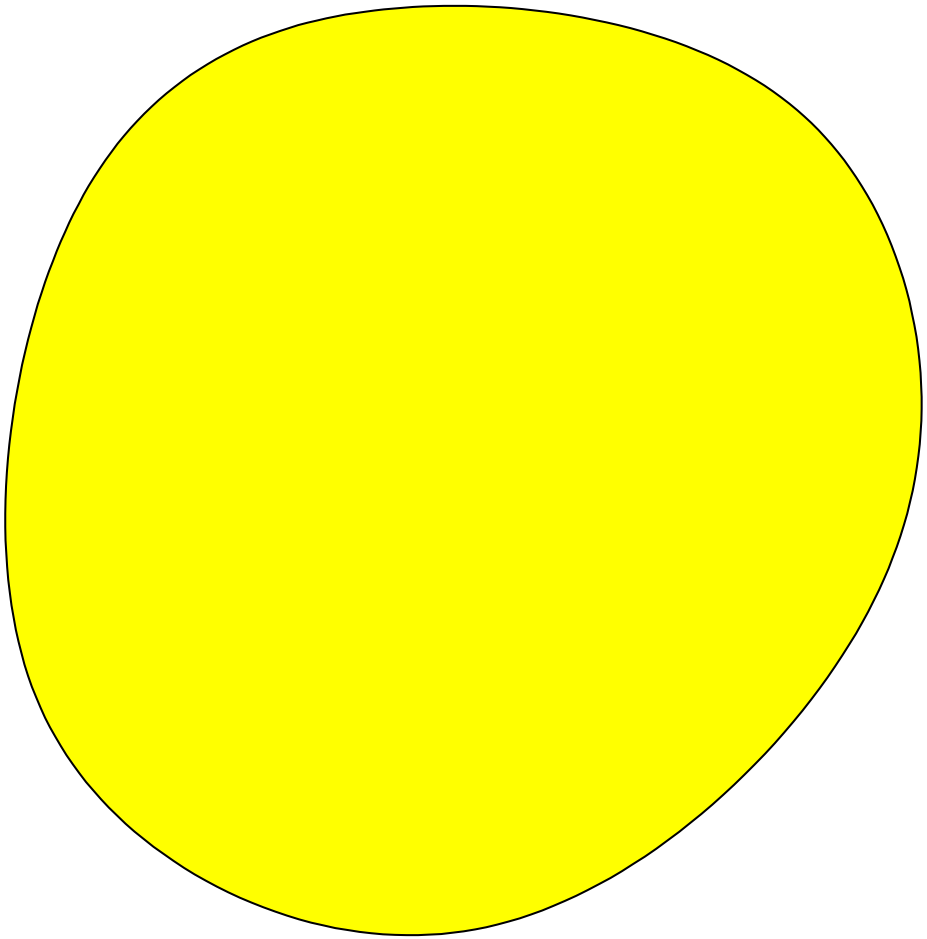
Graf G a číslo k je možné zkonstruovat k formuli φ v čase $O(n^2)$, kde n je velikost formule φ.

Navíc jsme viděli, že ve zkonstruovaném grafu G existuje nezávislá množina velikosti k právě tehdy, když formule φ je splnitelná.

Popsaný algoritmus tedy ukazuje, že problém 3-SAT je polynomiálně převeditelný na problém IS.

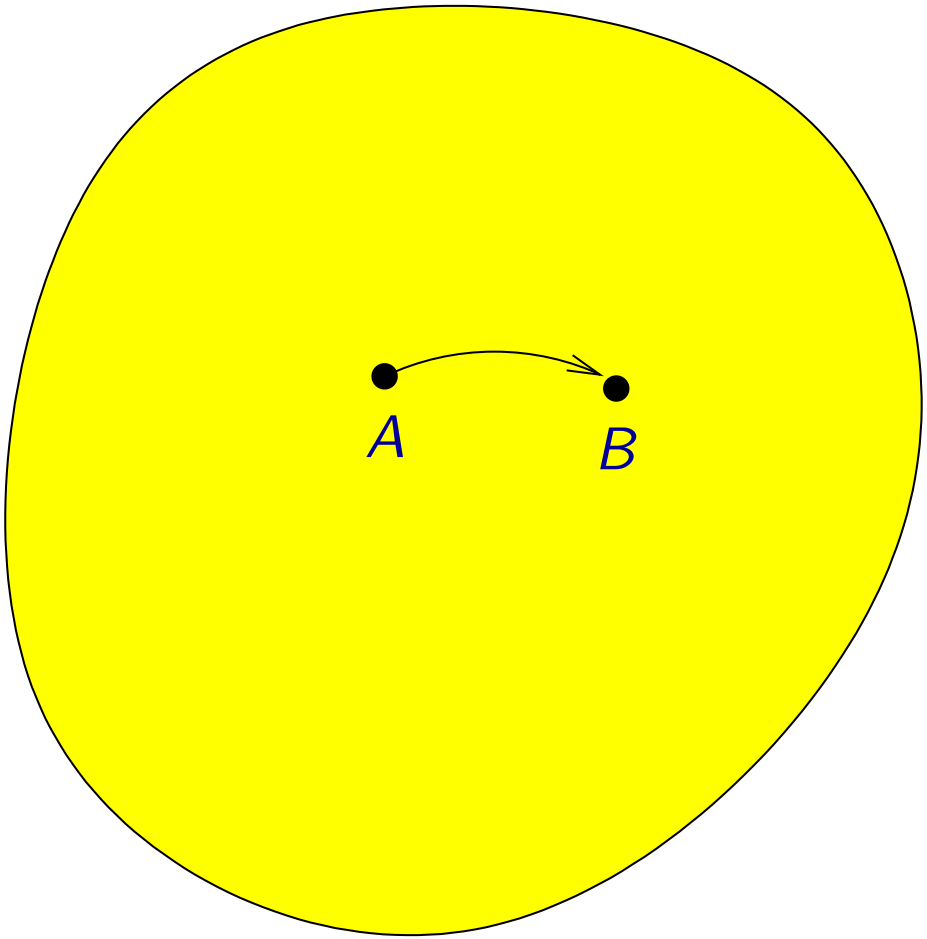
NP-úplné problémy

Vezměme si množinu všech možných rozhodovacích problémů.



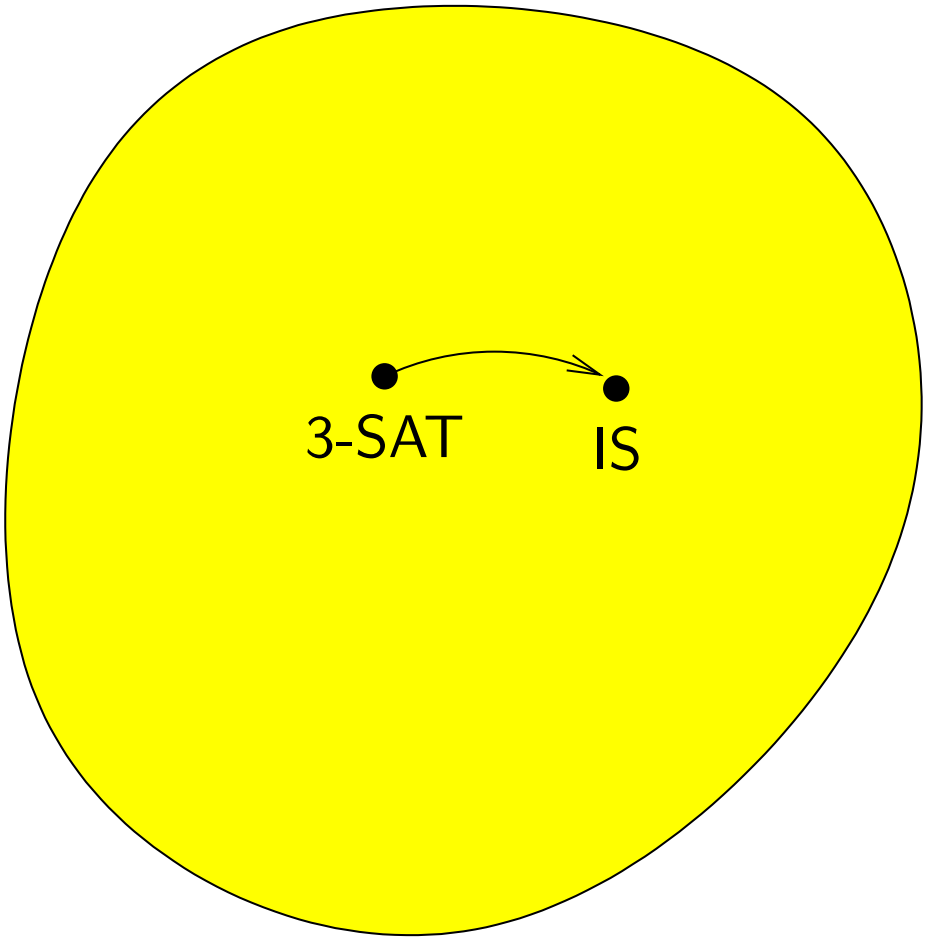
NP-úplné problémy

Šipkou si znázorníme, že problém A je polynomiálně převeditelný na problém B.



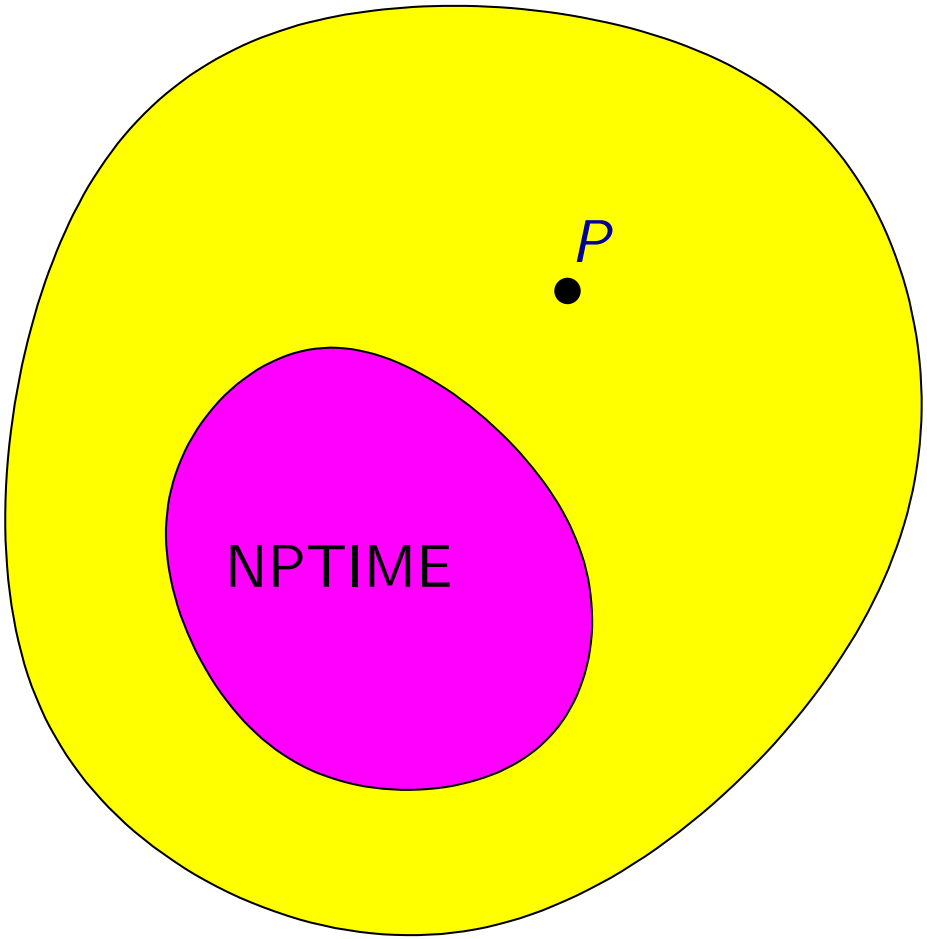
NP-úplné problémy

Například problém 3-SAT je polynomiálně převeditelný na problém IS.



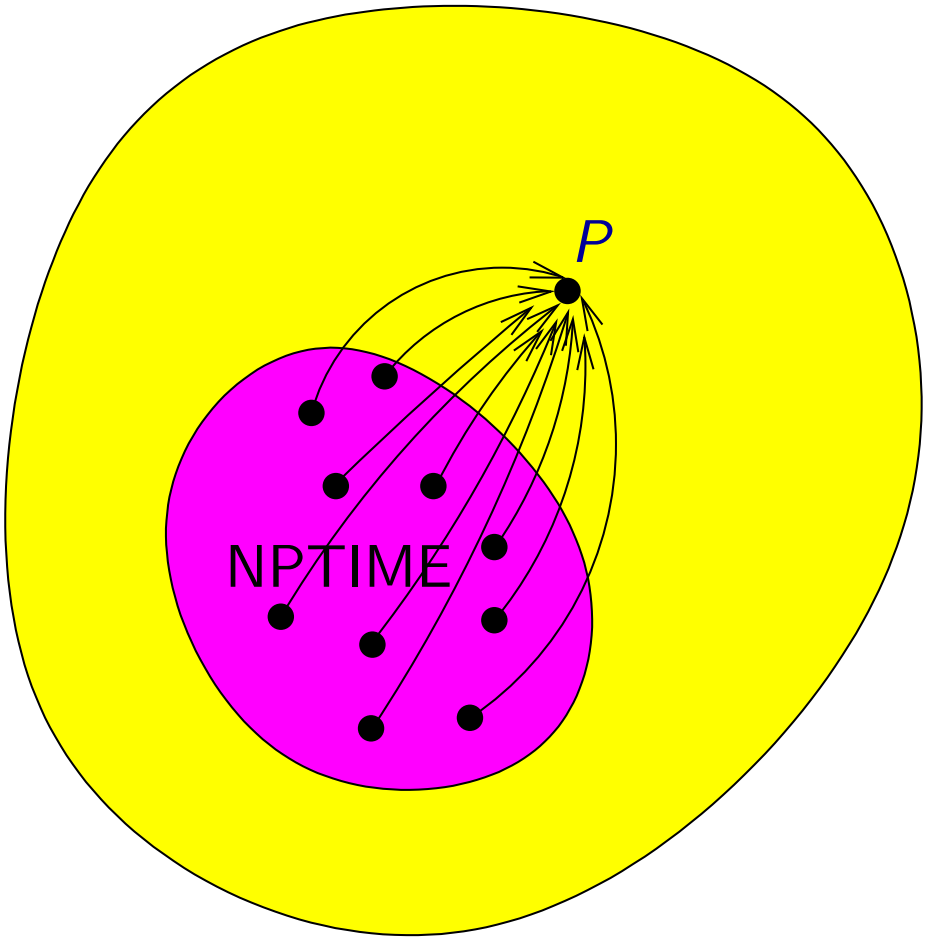
NP-úplné problémy

Vezměme si nyní třídu NPTIME a nějaký problém P.



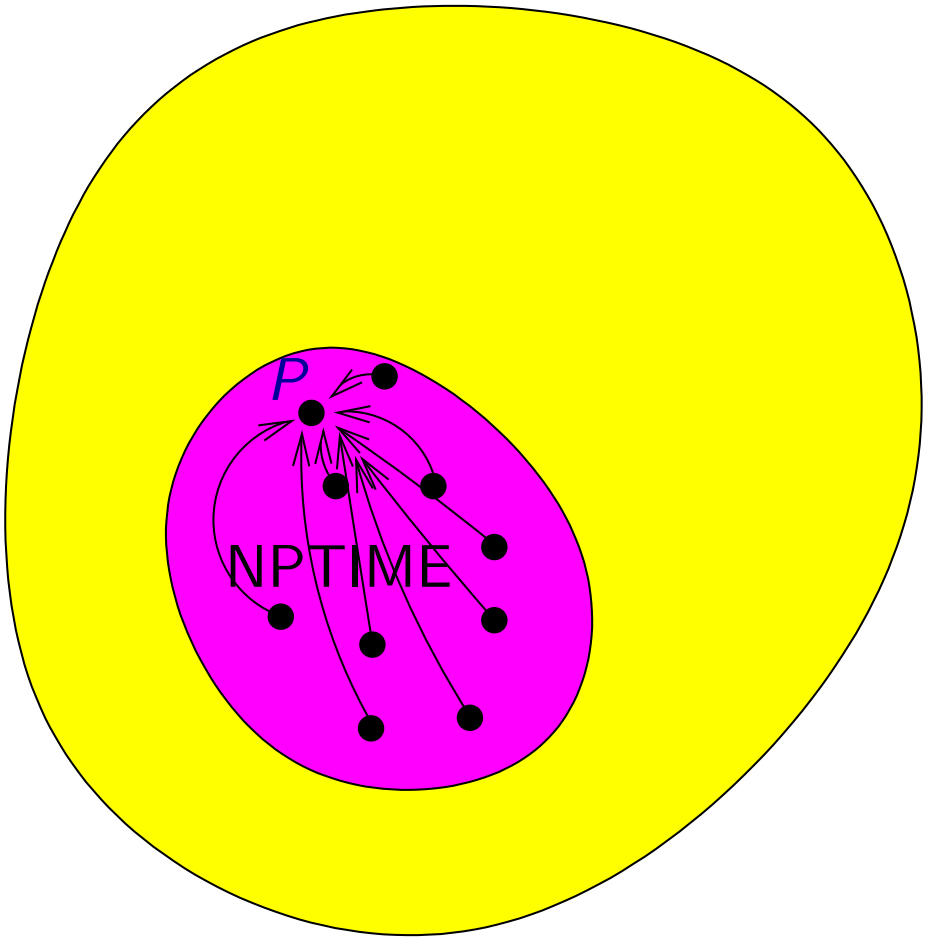
NP-úplné problémy

Problém P je NP-**těžký**, jestliže každý problém z NPTIME je polynomiálně převeditelný na P.



NP-úplné problémy

Problém P je NP-**úplný**, jestliže je NP-těžký a navíc sám patří do třídy NPTIME.



NP-úplné problémy

Pokud bychom pro nějaký NP-těžký problém P nalezli polynomiální algoritmus, získali bychom tím polynomiální algoritmus pro každý problém P' z NPTIME:

- Na vstup problému P' bychom nejprve aplikovali algoritmus realizující polynomiální převod z P' na P.
- Na vytvořenou instanci problému P bychom aplikovali polynomiální algoritmus řešící problém P a výsledek bychom vrátili jako odpověď pro danou instanci problému P'.

V takovém případě by tedy platilo PTIME = NPTIME, neboť pro každý problém z NPTIME by existoval polynomiální (deterministický) algoritmus.

NP-úplné problémy

Na druhou stranu, pokud existuje alespoň jeden problém z NPTIME, pro který neexistuje polynomiální algoritmus, tak z předchozího plyne, že pro žádný NP-těžký problém nemůže existovat polynomiální algoritmus.

Zda platí první nebo druhá možnost, je otevřený problém.

NP-úplné problémy

Není těžké si rozmyslet následující:

Pokud je problém A polynomiálně převeditelný na problém B a problém B je polynomiálně převeditelný na problém C, pak problém A je polynomiálně převeditelný na problém C.

Pokud tedy o nějakém problému P víme, že je NP-těžký a že P je polynomiálně převeditelný na problém P', pak víme, že i problém P' je NP-těžký.

NP-úplné problémy

Věta (Cook, 1971)

Problém SAT je NP-úplný.

Dá se ukázat, že SAT je polynomiálně převeditelný na 3-SAT a viděli jsme, že 3-SAT je polynomiálně převeditelný na IS.

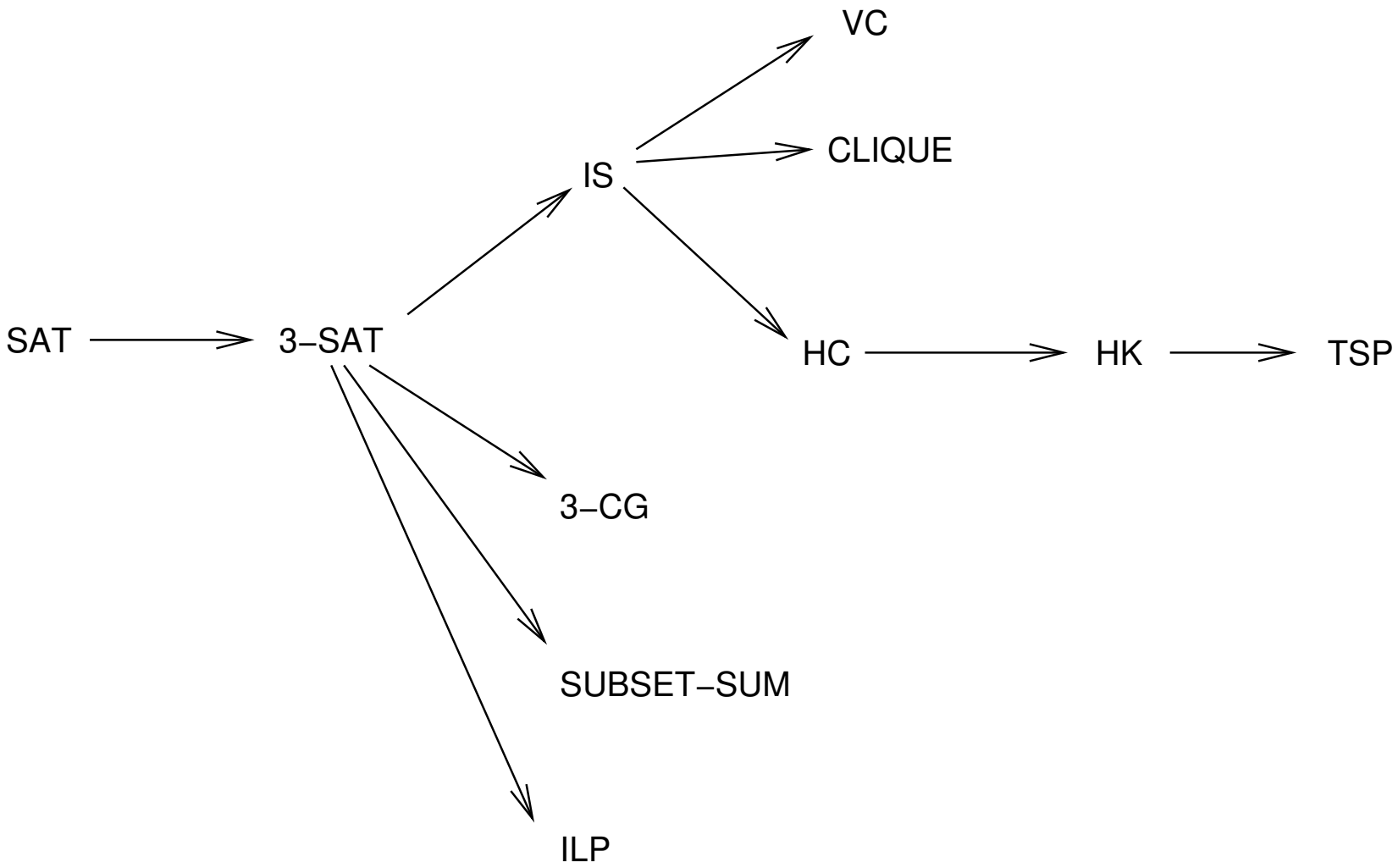
Z toho plyne, že problémy 3-SAT a IS jsou NP-těžké.

Není také těžké ukázat, že 3-SAT i IS patří do třídy NPTIME.

Problémy 3-SAT i IS jsou NP-úplné.

NP-úplné problémy

Polynomiálními převody z již známých NP-úplných problémů se dá ukázat NP-obtížnost celé řady různých dalších problémů:



Příklady některých NP-úplných problémů

Ze zatím uvedených problémů jsou NP-úplné následující tři problémy:

- SAT (splnitelnost booleovských formulí)
- 3-SAT
- IS — problém nezávislé množiny (independent set)

Na následujících slidech jsou uvedeny některé další NP-úplné problémy:

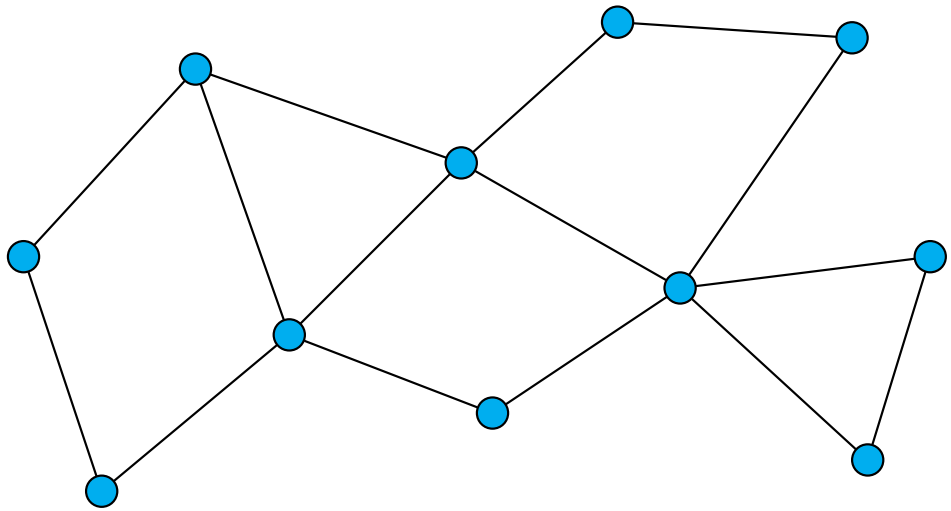
- CG — vrcholové barvení grafu (pozn.: je NP-úplný i ve speciálním případě, kdy máme právě 3 barvy)
- VC — vrcholové pokrytí grafu (vertex cover)
- CLIQUE — problém kliky
- HC — problém Hamiltonovského cyklu
- HK — problém Hamiltonovské kružnice
- TSP — problém obchodního cestujícího (traveling salesman problem)
- SUBSET-SUM
- ILP — celočíselné lineární programování (integer linear programming)

Barvení grafu

Vstup: Neorientovaný graf G, přirozené číslo k.

Otázka: Lze vrcholy grafu G obarvit k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Příklad: $k = 3$

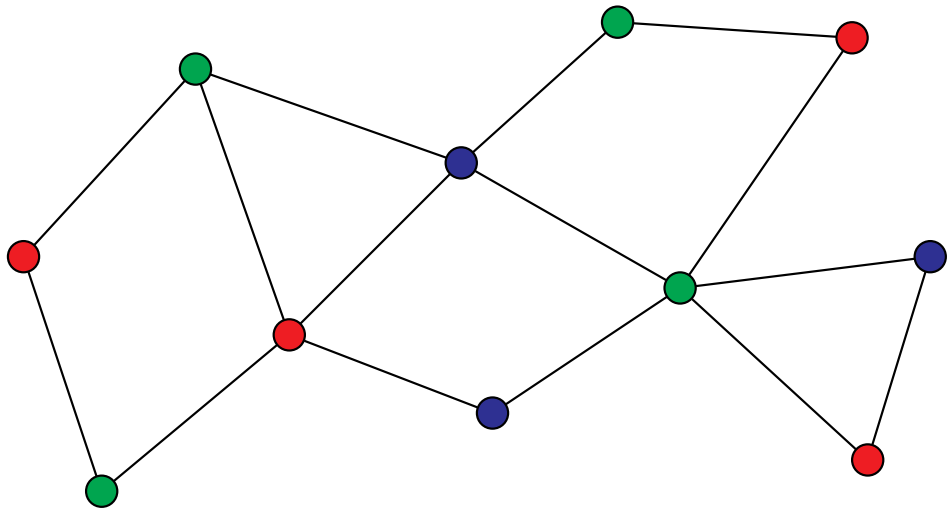


Barvení grafu

Vstup: Neorientovaný graf G, přirozené číslo k.

Otázka: Lze vrcholy grafu G obarvit k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Příklad: $k = 3$



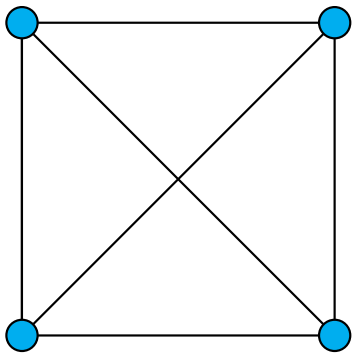
Odpověď: ANO

Barvení grafu

Vstup: Neorientovaný graf G, přirozené číslo k.

Otázka: Lze vrcholy grafu G obarvit k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Příklad: $k = 3$

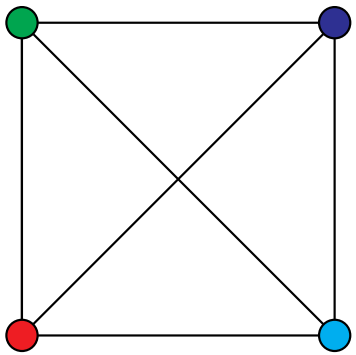


Barvení grafu

Vstup: Neorientovaný graf G, přirozené číslo k.

Otázka: Lze vrcholy grafu G obarvit k barvami tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu?

Příklad: $k = 3$



Odpověď: NE

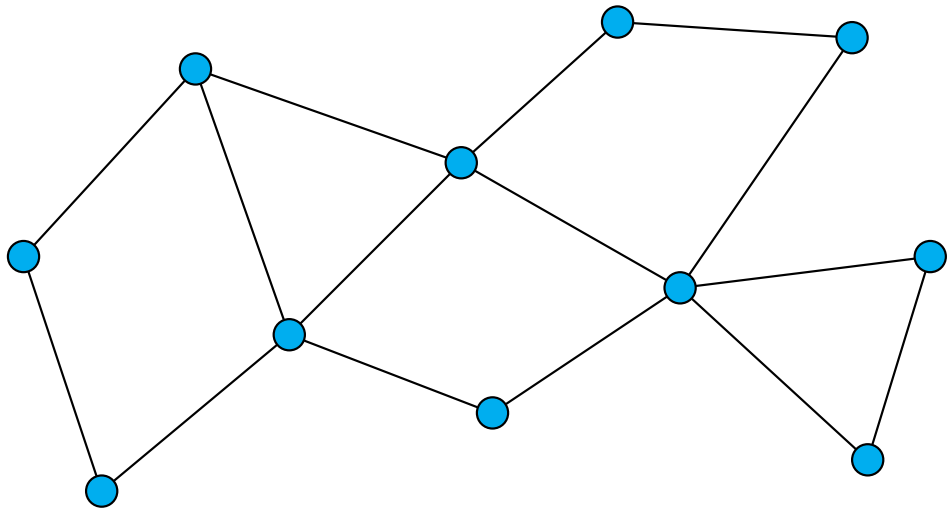
VC – Vrcholové pokrytí

VC – vrcholové pokrytí (vertex cover)

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Existuje v grafu G množina vrcholů velikosti k taková, že každá hrana má alespoň jeden svůj vrchol v této množině?

Příklad: $k = 6$



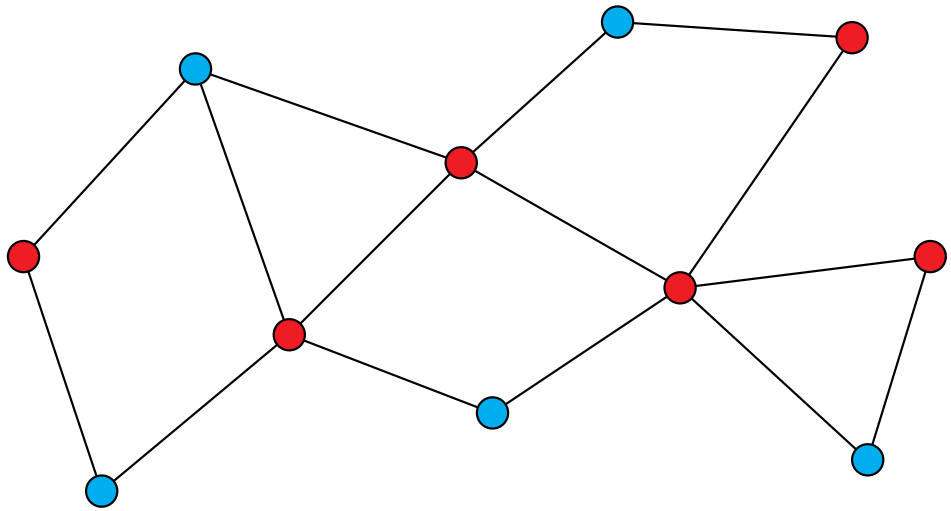
VC – Vrcholové pokrytí

VC – vrcholové pokrytí (vertex cover)

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Existuje v grafu G množina vrcholů velikosti k taková, že každá hrana má alespoň jeden svůj vrchol v této množině?

Příklad: $k = 6$



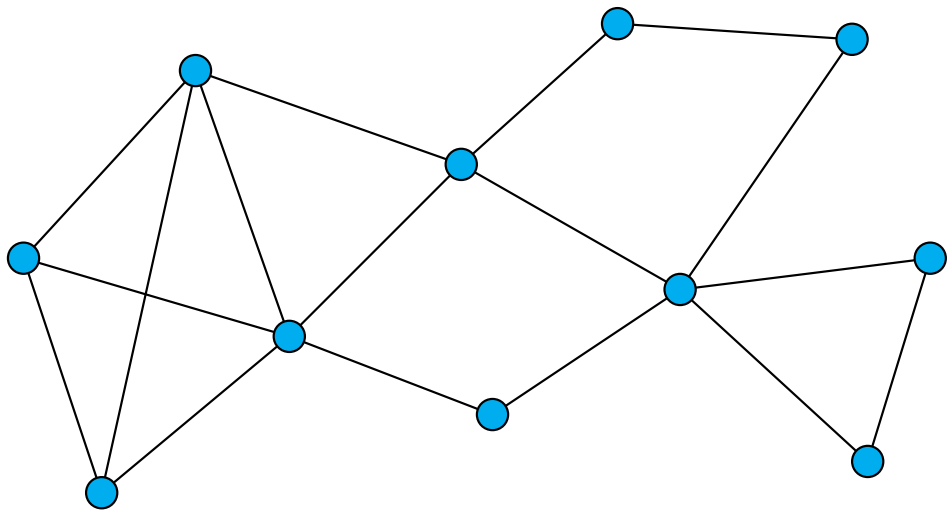
Odpověď: Ano

CLIQUE – problém kliky

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Existuje v grafu G množina vrcholů velikosti k taková, že každé dva vrcholy této množiny jsou spojeny hranou?

Příklad: $k = 4$

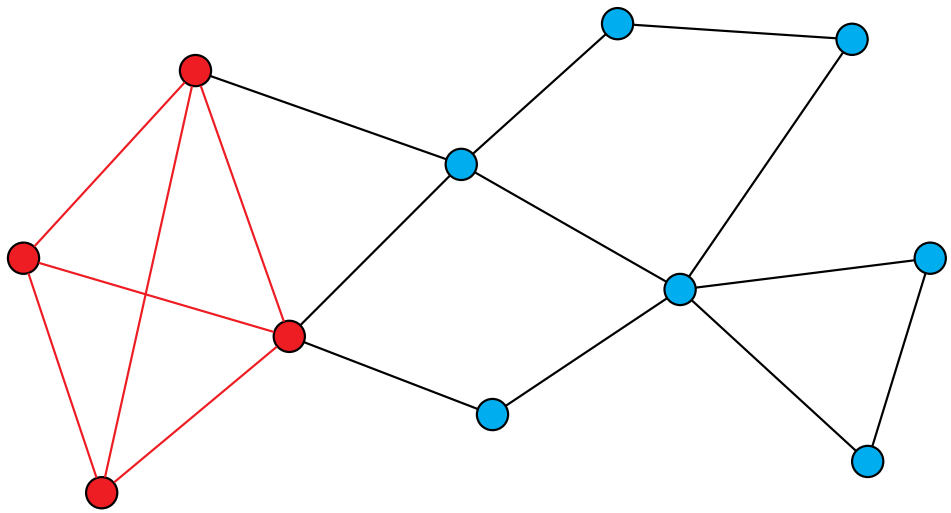


CLIQUE – problém kliky

Vstup: Neorientovaný graf G a přirozené číslo k.

Otázka: Existuje v grafu G množina vrcholů velikosti k taková, že každé dva vrcholy této množiny jsou spojeny hranou?

Příklad: $k = 4$



Odpověď: Ano

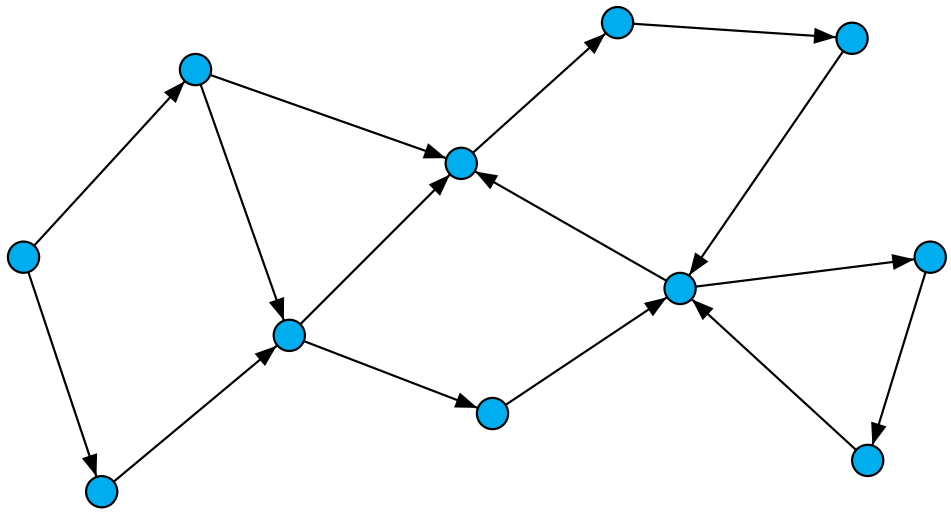
Hamiltonovský cyklus

HC – Problém „Hamiltonovský cyklus“

Vstup: Orientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovský cyklus (orientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



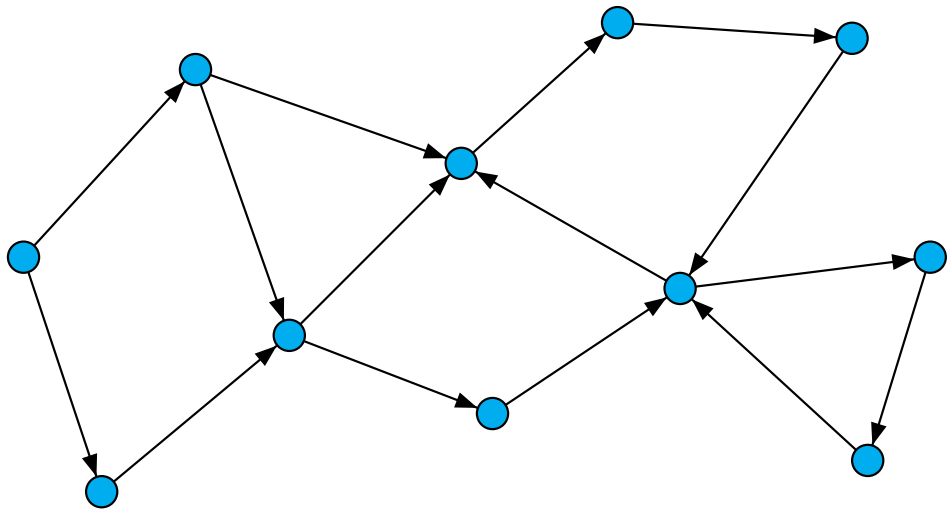
Hamiltonovský cyklus

HC – Problém „Hamiltonovský cyklus“

Vstup: Orientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovský cyklus (orientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



Odpověď: NE

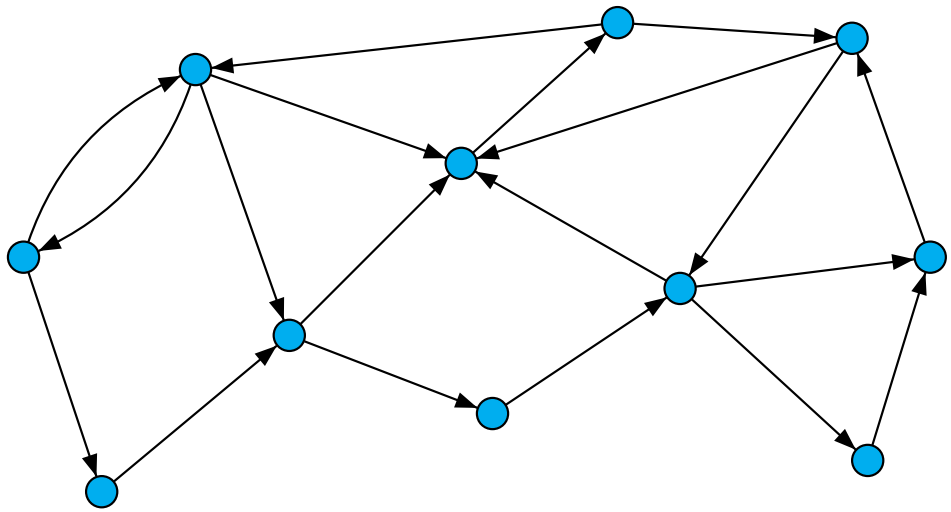
Hamiltonovský cyklus

HC – Problém „Hamiltonovský cyklus“

Vstup: Orientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovský cyklus (orientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



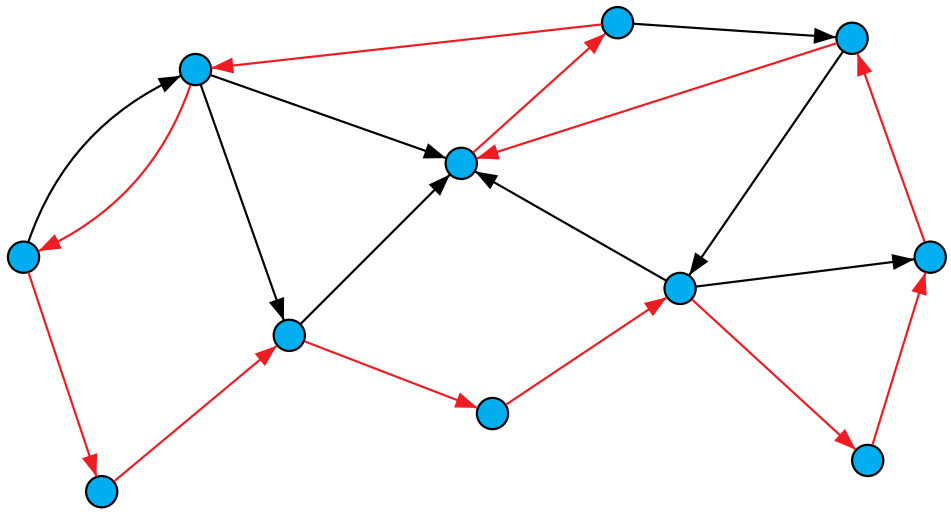
Hamiltonovský cyklus

HC – Problém „Hamiltonovský cyklus“

Vstup: Orientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovský cyklus (orientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



Odpověď: ANO

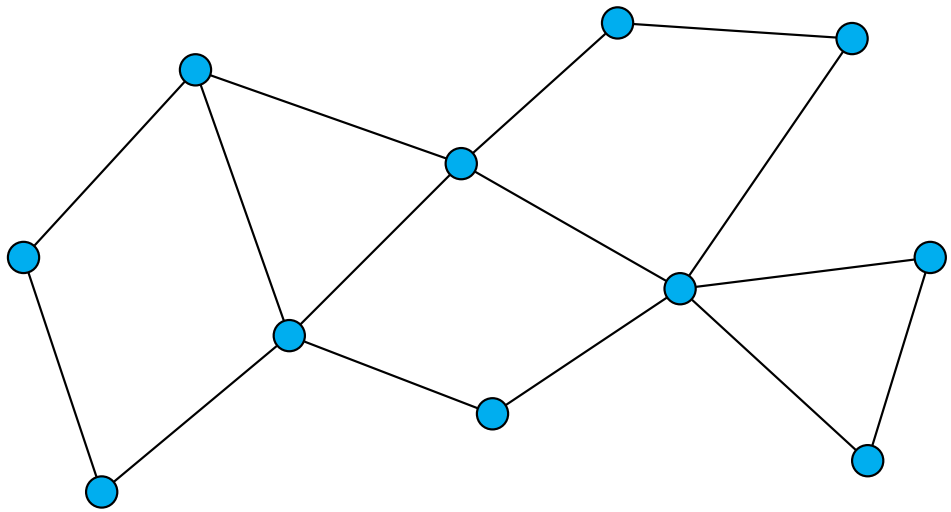
Hamiltonovská kružnice

HK – Problém „Hamiltonovská kružnice“

Vstup: Neorientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovská kružnice (neorientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



Odpověď: NE

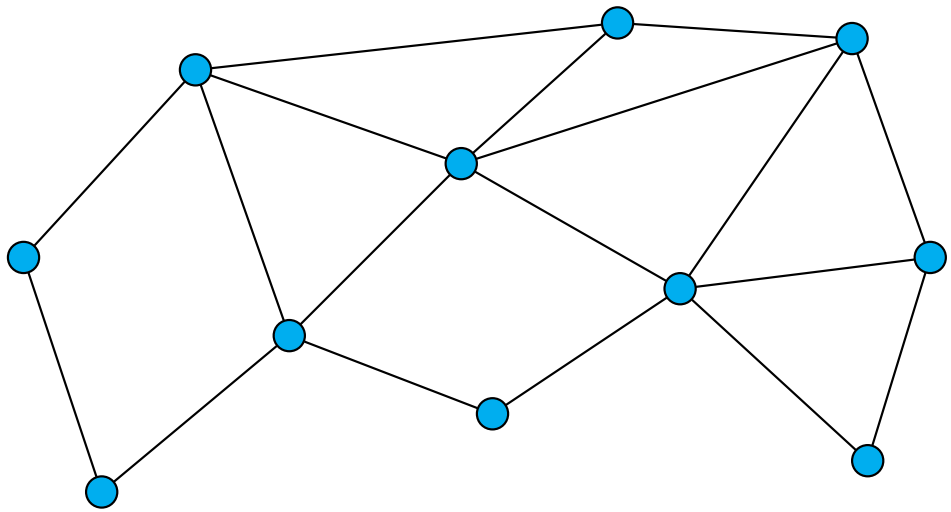
Hamiltonovská kružnice

HK – Problém „Hamiltonovská kružnice“

Vstup: Neorientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovská kružnice (neorientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



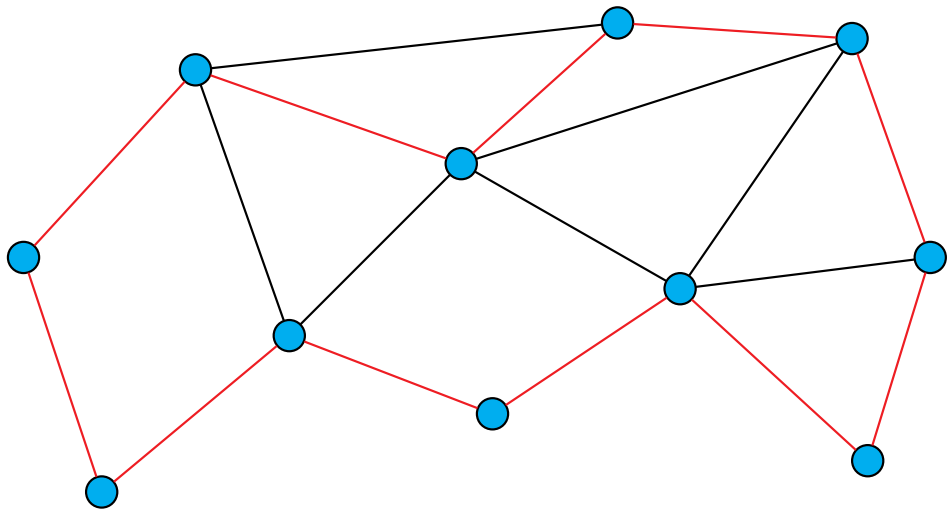
Hamiltonovská kružnice

HK – Problém „Hamiltonovská kružnice“

Vstup: Neorientovaný graf G.

Otázka: Existuje v grafu G Hamiltonovská kružnice (neorientovaný cyklus procházející každým vrcholem právě jednou)?

Příklad:



Odpověď: Ano

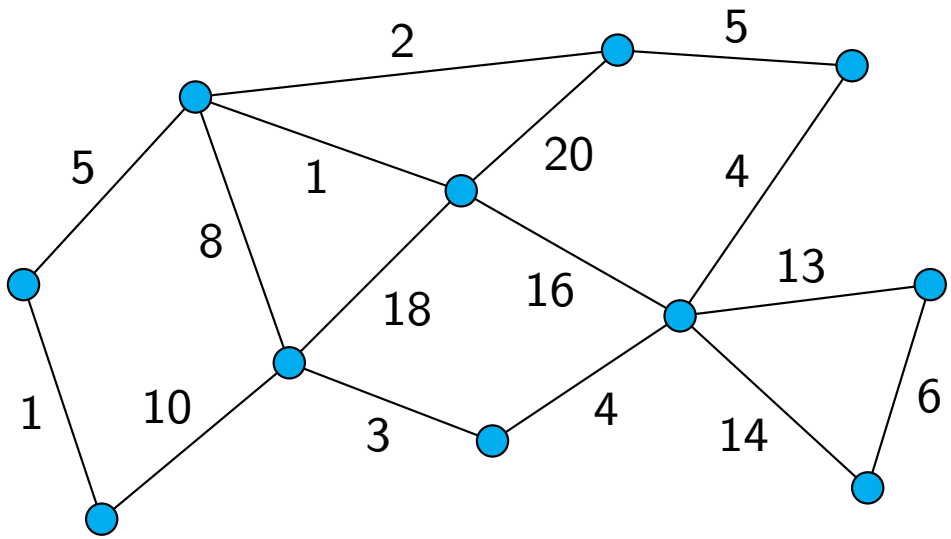
Problém obchodního cestujícího

TSP - Problém „obchodního cestujícího“

Vstup: Neorientovaný graf G s hranami ohodnocenými přirozenými čísly a číslo k.

Otázka: Existuje v grafu G uzavřená cesta procházející všemi vrcholy takový, že součet délek hran na této cestě (včetně opakovaných) je maximálně k?

Příklad: $k = 70$



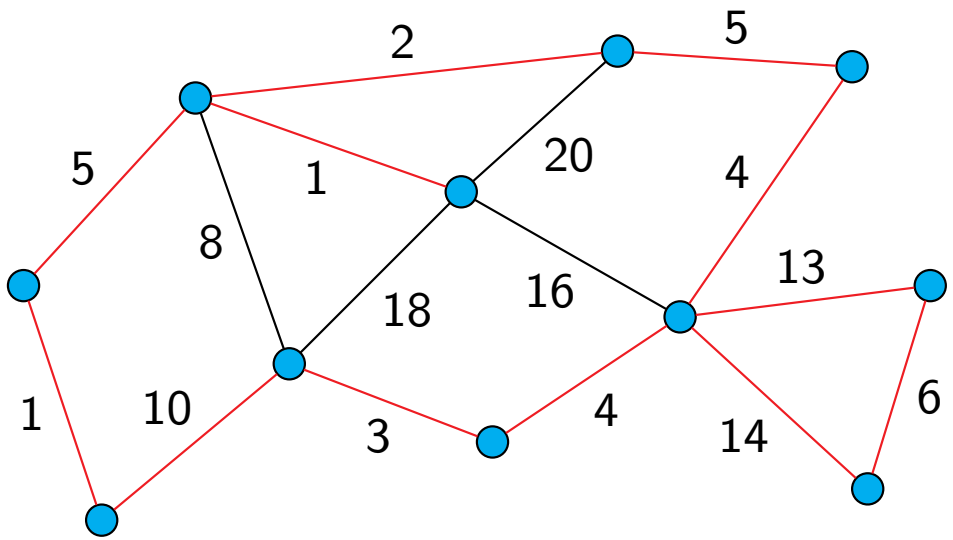
Problém obchodního cestujícího

TSP - Problém „obchodního cestujícího“

Vstup: Neorientovaný graf G s hranami ohodnocenými přirozenými čísly a číslo k.

Otázka: Existuje v grafu G uzavřená cesta procházející všemi vrcholy takový, že součet délek hran na této cestě (včetně opakovaných) je maximálně k?

Příklad: $k = 70$



Odpověď: Ano, protože byla nalezena cesta se součtem 69.

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.

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.

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 $\sum_{i \in I} b_i \geq t$?

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 vážila maximálně s gramů 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.

SUBSET-SUM

To, že SUBSET-SUM je speciálním případem problému batohu, vidíme z následující jednoduché konstrukce:

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.

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.

- Pokud počítáme velikost vstupu jako celkový počet bitů při použití **binárního** zápisu čísel, tak pro problém SUBSET-SUM není znám polynomiální algoritmus.
- Pokud počítáme velikost vstupu jako celkový počet bitů při použití **unárního** zápisu, tak existuje pro problém SUBSET-SUM algoritmus s polynomiální časovou složitostí.

ILP – celočíselné lineární programování

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:

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

ILP – celočíselné lineární programování

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.

ILP – celočíselné lineární programování

Poznámka: Analogický problém, kdy se pro danou soustavu lineálních nerovnic ptáme, zda existuje její řešení v oboru **reálných** čísel, je možné řešit v polynomiálním čase.

PSPACE-úplné problémy, EXPTIME-úplné problémy, . . .

- Problém P je PSPACE-**těžký**, jestliže je každý problém P' z PSPACE polynomiálně převeditelný na problém P.
- Problém P je PSPACE-**úplný**, jestliže je PSPACE-těžký a navíc sám patří do třídy PSPACE.

- Problém P je EXPTIME-**těžký**, jestliže je každý problém P' z EXPTIME polynomiálně převeditelný na problém P.
- Problém P je EXPTIME-**úplný**, jestliže je EXPTIME-těžký a navíc sám patří do třídy EXPTIME.

$\vdots$

PSPACE-úplné problémy, EXPTIME-úplné problémy, ...

Obecně pro libovolnou třídu složitosti $\mathcal{C}$ můžeme zavést třídy $\mathcal{C}$-**těžkých** a $\mathcal{C}$-**úplných** problémů:

Definice

- Problém P je $\mathcal{C}$-**těžký**, jestliže je každý problém P' ze třídy $\mathcal{C}$ polynomiálně převeditelný na problém P.
- Problém P je $\mathcal{C}$-**úplný**, jestliže je $\mathcal{C}$-těžký a navíc sám patří do třídy $\mathcal{C}$.

Kromě NP-úplných problémů tak máme PSPACE-úplné problémy, EXPTIME-úplné problémy, EXPSPACE-úplné problémy, 2-EXPTIME-úplné problémy, ...

Obecně se dá říci, že $\mathcal{C}$-úplné problémy jsou vždy ty nejtěžší problémy v dané třídě $\mathcal{C}$.

PTIME-úplné problémy, NL-úplné problémy, . . .

Poznámka: Výše uvedeným způsobem zavedené pojmy $\mathcal{C}$-těžkých a $\mathcal{C}$-úplných problémů, kdy byl v definici použit pojem **polynomiální převoditelnosti**, nedávají příliš smysl pro třídu PTIME a další třídy, které jsou jejími podmnožinami (jako třeba NLOGSPACE).

Pro takové třídy se zavádí pojmy $\mathcal{C}$-těžké a $\mathcal{C}$-úplné problémy podobným způsobem jako v předchozích definicích, ale místo polynomiální redukcí se používají, tzv. **logspace redukce**:

- algoritmus realizují daný převod musí být deterministický a mít logaritmickou prostorovou složitost

Tímto způsobem se zavádí například:

- PTIME-úplné a PTIME-těžké problémy
- NLOGSPACE-úplné a NLOGSPACE-těžké problémy (většinou se označují kratším názvem jako NL-úplné a NL-těžké)

Příklad NL-úplného problému

Typický příklad NL-úplného problému:

Dosažitelnost v grafu

Vstup: Orientovaný graf G a dva jeho vrcholy s a t.

Otázka: Existuje v grafu G cesta z vrcholu s do vrcholu t?

Příklad PTIME-úplného problému

Typický příklad PTIME-úplného problému:

Circuit Value Problem

Vstup: Acyklický booleovský obvod C skládající se z hradel a vodičů a booleovské hodnoty $x_1, x_2, \ldots, x_n$ na vstupech tohoto obvodu.

Otázka: Bude na výstupu obvodu C při daných hodnotách vstupů hodnota 1 ?

Příklady PSPACE-úplných problémů

Typickým příkladem PSPACE-úplného problému je problém **kvantifikovaných booleovských formulí** — **QBF** (Quantified Boolean Formulas):

QBF

Vstup: Kvantifikovaná booleovská formule tvaru

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdots \exists x_{n-1} \forall x_n : \varphi,$$

kde φ je (běžná) booleovská formule obsahující proměnné $x_1, x_2, \ldots, x_n$.

Otázka: Je daná formule pravdivá?

Příklady PSPACE-úplných problémů

EqNFA

Vstup: Nedeterministické konečné automaty $\mathcal{A}_1$ a $\mathcal{A}_2$.

Otázka: Je $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$?

Univerzalita NKA

Vstup: Nedeterministický konečný automat $\mathcal{A}$.

Otázka: Je $\mathcal{L}(\mathcal{A}) = \Sigma^*$?

Příklady PSPACE-úplných problémů

EqRE

Vstup: Regulární výrazy α_1 a α_2.

Otázka: Je $\mathcal{L}(\alpha_1) = \mathcal{L}(\alpha_2)$?

Univerzalita RV

Vstup: Regulární výraz α.

Otázka: Je $\mathcal{L}(\alpha) = \Sigma^*$?

Příklady PSPACE-úplných problémů

Uvažujme následující hru, kterou hrají dva hráči na orientovaném grafu G:

- Hráči střídavě přesunují po vrcholech grafu G jeden hrací kámen.
- Při tazích se označují vrcholy, které již byly kamenem navštíveny.
- Začíná se na specifikovaném vrcholu v_0.
- Řekněme, že kámen je momentálně na vrcholu v. Hráč, který je na tahu, vybere vrchol v' takový, že existuje hrana z v do v' a vrchol v' nebyl dosud navštíven.
- Hráč, který nemůže táhnout, prohrál a jeho protivník vyhrál.

Generalized Geografy

Vstup: Orientovaný graf G s vyznačeným počátečním vrcholem v_0.

Otázka: Má hráč, který táhne jako první, vyhrávající strategii ve hře hrané na grafu G, kde se začíná ve vrcholu v_0 ?

Příklady EXPTIME-úplných problémů

Typický příklad EXPTIME-úplného problému:

Vstup: Turingův stroj $\mathcal{M}$, slovo w a číslo k zapsané binárně.

Otázka: Zastaví se výpočet stroje $\mathcal{M}$ nad slovem w do k kroků?
(Tj. udělá stroj $\mathcal{M}$ při výpočtu nad slovem w
nejvýše k kroků?)

Příklady EXPTIME-úplných problémů

Další příklady EXPTIME-úplných problémů jsou například zobecněné varianty her jako jsou šachy, dáma nebo Go, hrané na hrací ploše libovolné velikosti (např. šachovnice velikosti $n \times n$):

- vstupem je pozice v dané hře (např. v šachu konkrétní rozestavení figurek na šachovnici a informace, který hráč je na tahu)
- otázka je, zda má hráč, který je momentálně na tahu, v dané pozici vyhrávající strategii

Příklady EXPSPACE-úplných problémů

Regulární výrazy s mocněním jsou definovány podobně jako běžné regulární výrazy, ale kromě operátorů $+$, $\cdot$ a * mohou navíc obsahovat unární operátor ^{2} s následujícím významem:

- α^2 je zkratkou pro $\alpha \cdot \alpha$.

Následující dva problémy jsou EXPSPACE-úplné:

Vstup: Regulární výrazy s mocněním α_1 a α_2.
Otázka: Je $\mathcal{L}(\alpha_1) = \mathcal{L}(\alpha_2)$?

Vstup: Regulární výraz s mocněním α.
Otázka: Je $\mathcal{L}(\alpha) = \Sigma^*$?

Presburgerova aritmetika

Příklad problému, který je sice rozhodnutelný, ale má velkou výpočetní složitost:

Problém

Vstup: Uzavřená formule predikátové logiky (prvního řádu), ve které mohou být použity jako predikátové symboly pouze $=$ a $<$, jako funkční symbol pouze $+$ a jako kostantní symboly pouze 0 a 1.

Otázka: Je daná formule pravdivá v oboru přirozených čísel (při přirozené interpretaci všech funkčních a predikátových symbolů)?

Pro tento problém je znám deterministický algoritmus s časovou složitostí $2^{2^{2^{O(n)}}}$ a je rovněž známo, že každý nedeterministický algoritmus řešící tento problém, musí mít časovou složitost nejméně $2^{2^{\Omega(n)}}$.