Teoretická informatika, FIT, VUT Brno

# 3. domácí úloha

**David Mihola**
xmihol00

18. prosince 2023

# 1 NP-úplnost problému $LP$

Nejdříve si uvědomme, jaká část zadaného problému vede na to, že ho nejsme schopní řešit v deterministickém polynomiálním čase, a přibližme si jeho možné řešení. Problém můžeme rozdělit do následujících kroků:

1. Relaci $M$ bez ztráty obecnosti[1] doplníme o dvojice tak, aby byla reflexivní, symetrická a tranzitivní, tzn., aby byla relací ekvivalence. Následně dle této relace provedeme rozklad množiny $H$ na třídy ekvivalence $E$. Reflexivní, symetrický i tranzitivní uzávěr lze řešit v deterministickém polynomiálním čase a stejnou složitost má i rozklad množiny relací ekvivalence. Podotkneme, že nyní každá třída ekvivalence obsahuje hosty, kteří spolu musí sedět u jednoho stolu.

2. V rámci každé třídy z $E$ zkontrolujeme, že libovolní dva hosté spolu nejsou v relaci $N$. Pokud ano, úloha nemá řešení, pokud ne, je nutné pokračovat. Zřejmě tato kontrola má kvadratickou složitost, takže lze řešit také v deterministickém polynomiálním čase.

3. Nakonec musíme hosty rozsadit ke stolům tak, aby u jednoho stolu spolu seděli všichni hosté z třídy $e_i$ pro každou třídu $e_i$ z $E$, a pokud jsou u jednoho stolu posazení hosté z více tříd z $E$ tak, aby žádní dva hosté u takového stolu nebyli spolu v relaci $N$. Dále musí platit, že součet mohutností tříd $e_i$ hostů usazených u jednoho stolu je menší nebo roven $V$ a že toto rozsazení je možné realizovat pro $S$ stolů. Zřejmě jsme schopni v deterministickém polynomiálním čase ověřit, jestli daný zasedací pořádek vyhovuje zadaným podmínkám, ale již nejsme schopni v tomto čase takový zasedací pořádek najít. Tato část úlohy tedy způsobuje, že problém $LP$ je v NP, a současně dokazuje, že je maximálně v NP[2].

Všimněme si, že, pokud je relace $N$ prázdná, 3. část řešení problému spočívá v rozmístění skupin hostů o různých velikostech k $S$ stolům o $V$ místech. Toto zjednodušení se značně podobá *Partition problému* ($PP$), který je NP-úplný. Proveďme tedy redukci z $PP$ na $LP$ v polynomiálním čase ($PP \leq_P^m LP$) a dokažme takto, že problém $LP$ je NP-těžký. Deterministický Turingův stroj $T$ implementující redukční funkci zachovávající příslušnost ve tvaru $\sigma : \{0,1\}^* \rightarrow \{0,1,\#\}^*$ se vstupem $\langle X \rangle$ a výstupem $\langle H \rangle \# \langle M \rangle \# \langle N \rangle \# \langle S \rangle \# \langle V \rangle$, kde $\langle \cdot \rangle$ značí vhodné kódování jednotlivých elementů a $X$ je multimnožina prvků z $\mathbb{N}$, bude pracovat následovně:

1. $T$ zkontroluje, že instance problému $PP$ je korektně zadaná, pokud ne, vrací kód problému $LP$, kde $H = \{Alice, Bob\}$, $M = \{(Alice, Bob)\}$, $N = \emptyset$ a $S = V = 0$.

2. $T$ sečte všechny prvky z $X$ a uloží tuto hodnotu do $sum$, pokud $\exists half \in \mathbb{N} : 2 \cdot half = sum$, $T$ nastaví $V_T = half$ a pokračuje následujícím bodem, jinak $T$ vrací kód problému $LP$, kde $H = \{Alice, Bob\}$, $M = \{(Alice, Bob)\}$, $N = \emptyset$ a $S = V = 0$.

[1] Zřejmě platí, že hoste sedí vždy jen u jednoho stolu, tj. sami se sebou, pokud host $A$ musí sedět s hostem $B$, tak i host $B$ bude sedět s hostem $A$ u jednoho stolu a pokud host $A$ musí sedět s hostem $B$ a host $B$ musí sedět s hostem $C$, tak i host $A$ bude sedět s hostem $C$ u jednoho stolu, když daná instance problému bude mít řešení.

[2] Tzn., neleží v žádné nadmnožině NP, ze které odebereme problémy z NP.

3. $T$ převede $X$ na pole.
4. $T$ nastaví $H_T$ provedením algoritmu 1.
5. $T$ nastaví $M_T$ provedením algoritmu 2.
6. $T$ vrací kód problému $LP$, kde $H = H_T$, $M = M_T$, $N = \emptyset$, $S = 2$ a $V = V_T$.

Jinak řečeno, redukce bude vypadat tak, že každé přirozené číslo $n_i > 0$ z $X$ bude zakódováno na $n_i$ unikátních prvků do množiny $H$ a relace $M$ bude vybudována tak, aby po provedení jejího reflexivního, symetrického a tranzitivního uzávěru rozkládala $H$ na třídy $e_i$ o mohutnostech $n_i$. Jelikož relace $N$ bude vždy prázdná, problém rozsazení pak odpovídá rozdělení tříd $e_i$ mezi dva stoly o velikosti poloviny součtu prvků v $X$.

$H_T \leftarrow \emptyset$
**for** $i \in \{0, 1, ..., len(X) - 1\}$ **do**
    **if** $X[i] > 0$ **then**
        **for** $j \in \{1, ..., X[i]\}$ **do**
            $H_T \leftarrow H_T \cup \{i_j\}$
        **end for**
    **end if**
**end for**

Algoritmus 1: Určení množiny hostů s futuristickými jmény $H_T$

$M_T \leftarrow \emptyset$
**for** $i \in \{0, 1, ..., len(X) - 1\}$ **do**
    **if** $X[i] > 1$ **then**
        **for** $j \in \{2, 3, ..., X[i]\}$ **do**
            $M_T \leftarrow M_T \cup \{(i_{j-1}, i_j)\}$
        **end for**
    **end if**
**end for**

Algoritmus 2: Určení množiny musí sedět s $M_T$

Dokázali jsme, že problém $LP$ je maximálně v NP, a současně, že je NP-těžký. Z toho plyne, že tento problém je NP-úplný. □

(Pozn.: Omlouvám se za zbytečně komplikované řešení. Po sepsání mě napadla ještě redukce z problému *Graph coloring*, která je mnohem jednodušší. Nastíním zde pouze její hlavní myšlenku. Množina hran obarvovaného grafu by se převedla na relaci $N$ (vrcholy spojené hranou nesmí mít stejnou barvu, čili spolu "nesmí sedět u stolu"). Množina $M$ by byla prázdná. Počet stolů $S$ by se rovnal počtu barev (každý vrchol musí mít nějakou barvu, čili musí "sedět u nějakého stolu"). A velikost stolů $V$ by byla dostatečně velká, tj. například počet vrcholů v obarvovaném grafu (klidně všechny vrcholy mohou mít stejnou barvu, "mohou sedět u jednoho stolu", pokud obarvovaný graf neobsahuje hrany).)

# 2 Složitost operace `vlož_a_uřež`

Uvažujme, že oboustranný seznam implementuje běžné zásobníkové operace[3]. Rekurzivně pak lze operaci `vlož_a_uřež` implementovat následovně:

```
procedure InsertAndCut(k)
    if list.empty() or list.top() < k then          ▷ součet ceny: 1
        list.push(k)                                 ▷ součet ceny: 2
    else
        value ← list.pop()                           ▷ součet ceny: 2
        InsertAndCut(k)                              ▷ součet ceny: 3
        Print(value)                                 ▷ součet ceny: 4
    end if
end procedure
```

Algoritmus 3: Procedura implementující operaci `vlož_a_uřež`

## 2.1 Důkaz běhu sekvence operací `vlož_a_uřež` v čase $O(n)$

Tabulka cen a kreditů bude v $i$-té iteraci vypadat následovně:

| operace | cena | kredit |
|---|---|---|
| `vlož_a_uřež` | $2 + 4l_i$ | $2 + 4$ |

Dále bude platit, že začínáme s délkou oboustranného seznamu $n_0 = 0$ a se stavem účtu $s_0 = 4n_0 = 0$[4]. V každé iteraci dle algoritmu výše musí platit, že $0 \leq l_i \leq n_i$ a že $n_{i+1} = n_i - l_i + 1$. Indukcí dokažme, že pro stav účtu v $i$-té iteraci platí, že $s_i = 4n_i \geq 0$[5]. V každé iteraci může nastat jedna z následujících situací[6]:

- Délka oboustranného seznamu se v $i$-té iteraci zvětší, pak jistě platí, že $l_i = 0$. Cena operace je pouze $2 + 4 \cdot 0 = 2$ a na účet se tím pádem uloží 4 kredity. Můžeme tedy konstatovat, že v $i + 1$. iteraci bude stav účtu $s_{i+1} = 4n_i$[7] $+ 2 + 4 - 2 = 4(n_i - 0 + 1) = 4n_{i+1}$.

- Délka oboustranného seznamu se v $i$-té iteraci nezvětší, pak jistě platí, že $0 < l_i \leq n_i$. Cena operace bude $2 + 4l_i$. Pak stav účtu v $i + 1$. iteraci bude $s_{i+1} = 4n_i$[7] $+ 2 + 4 - (2 + 4l_i) = 4n_i - 4l_i + 4 = 4(n_i - l_i + 1) = 4n_{i+1}$.

Jelikož délka seznamu je vždy nezáporná, je zřejmé, že stav účtu v žádné iteraci nebude záporný. Z toho plyne, že složitost sekvence $n$ operací `vlož_a_uřež` je $6n$, což náleží do $O(n)$. Amortizovaná složitost jedné operace `vlož_a_uřež` je tím pádem konstantní. □

[3]Tzn., že operaci `vlož_a_uřež` lze implementovat se stejnou složitostí i nad zásobníkem.
[4]bázový případ
[5]indukční předpoklad
[6]indukční krok
[7]Plyne z indukčního předpokladu $s_i = 4n_i$.

# 3 Důkaz lineárního nárůstu počtu znaků ve formuli Tseytinovou transformací

Definujme Tseytinovou transformací rekurzivně algoritmem 4. Z algoritmu je zřejmé, že každá podformule $X_i$ transformované formule $\varphi$ obsahuje nově vytvořenou výrokovou proměnnou $x_i$. Nechť formule $X_{n+1}$[8] je výsledek volání $\textsc{Tse}(\varphi, 0)$, pak formule $\varphi$ je logicky ekvivalentní s formulí $(X_{n+1} \wedge x_n)$[9]. Dále pro vybrané formule $\psi$ definujme jejich převod do CNF dle tabulky 1[10].

**procedure** $\textsc{Tse}(\varphi, i)$
  **if** $\varphi = \neg\neg\varphi_a$ **then** ▷ odstranění dvojité negace, $\varphi_a$ je formule nebo výroková proměnná
    $X_j = \textsc{Tse}(\varphi_a, i)$
    **return** $X_j$
  **else if** $\varphi = (\varphi_a \vee \varphi_b)$ **then** ▷ $\varphi_a$ a $\varphi_b$ jsou libovolně formule nebo výrokové proměnné
    $X_j \leftarrow \textsc{Tse}(\varphi_a, i)$
    $X_k \leftarrow \textsc{Tse}(\varphi_b, j+1)$
    $l \leftarrow k+1$
    $X_l \leftarrow \text{CNF}(x_l \leftrightarrow (x_j \vee x_k))$
    **return** $X_j \wedge X_k \wedge X_l$
  **else if** $\varphi = \neg(\varphi_a \vee \varphi_b)$ **then** ▷ $\varphi_a$ a $\varphi_b$ jsou libovolně formule nebo výrokové proměnné
    $X_j \leftarrow \textsc{Tse}(\neg\varphi_a, i)$
    $X_k \leftarrow \textsc{Tse}(\neg\varphi_b, j+1)$
    $l \leftarrow k+1$
    $X_l \leftarrow \text{CNF}(x_l \leftrightarrow (x_j \wedge x_k))$ ▷ umělý krok pro zachování tvaru a číslování podformulí
    **return** $X_j \wedge X_k \wedge X_l$
  **else if** $\varphi = y$ **then** ▷ ukončující podmínka rekurze, kde $y$ je výroková proměnná
    $X_i \leftarrow \text{CNF}(x_i \leftrightarrow y)$
    **return** $X_i$
  **else if** $\varphi = \neg y$ **then** ▷ ukončující podmínka rekurze, kde $y$ je výroková proměnná
    $X_i \leftarrow \text{CNF}(x_i \leftrightarrow \neg y)$
    **return** $X_i$
  **end if** ▷ žádné další možné případy pro zadanou gramatiku nejsou
**end procedure**

Algoritmus 4: Procedura implementující rekurzivní Tseytinovu transformaci

[8] Pak podformule $X_{n+1}$ lze uspořádat a formuli lze zapsat jako $X_{n+1} = X_0 \wedge X_1 \wedge \cdots \wedge X_{n-1} \wedge X_n$. Čili číslování se mírně liší od zadání, ale to nic nemění na obecnosti řešení.

[9] Tento krok prodlouží transformovanou formuli o konstantní počet znaků, takže lineární nárůst znaků neporuší.

[10] Jedná se o bližší specifikování formulí $X \leftrightarrow \neg Y$ a $X \leftrightarrow (Y \vee Z)$ ze zadání, kde formule $x \leftrightarrow \neg y$, $x \leftrightarrow y \Leftrightarrow x \leftrightarrow \neg\neg y$ a $x \leftrightarrow (y \wedge z) \Leftrightarrow x \leftrightarrow \neg\neg(y \wedge z)$ odpovídají tvaru $X \leftrightarrow \neg Y$ a formule $x \leftrightarrow (y \vee z)$ odpovídá tvaru $X \leftrightarrow (Y \vee Z)$

| $\psi$ | **$\psi$ v CNF** | **$\lvert\psi$ v CNF$\rvert$** |
|---|---|---|
| $x \leftrightarrow y$ | $(x \vee \neg y) \wedge (y \vee \neg x)$ | 13 |
| $x \leftrightarrow \neg y$ | $(x \vee y) \wedge (\neg y \vee \neg x)$ | 13 |
| $x \leftrightarrow (y \vee z)$ | $(x \vee \neg y) \wedge (x \vee \neg z) \wedge (y \vee z \vee \neg x)$ | 22 |
| $x \leftrightarrow (y \wedge z)$ | $(x \vee \neg y \vee \neg z) \wedge (y \vee \neg x) \wedge (z \vee \neg x)$ | 23 |

Tabulka 1: Převod vybraných formulí do CNF

Nyní indukcí dokažme, že je $|\textsc{Tse}(\varphi, i)| \leq c|\varphi|$, kde $c \in \mathbb{N}^+$ je dostatečně velká konstanta. Předpokládejme, že tvrzení platí pro všechny syntakticky správné formule o délce $n = k$[11], kde $k \in \mathbb{N}^+$[12]. Pokud $n = |\varphi| = 1$, tak $\varphi = y$, kde $y$ je výroková proměnná, pak dle tabulky 1 je $(|\textsc{Tse}(\varphi, i)| = |\text{CNF}(x_i \leftrightarrow y)| = 13) \leq (c|\varphi| = c) \Leftrightarrow 13 \leq c$. Pokud $n = |\varphi| = 2$, tak $\varphi = \neg y$, kde $y$ je výroková proměnná, pak dle tabulky 1 je $(|\textsc{Tse}(\varphi, i)| = |\text{CNF}(x_i \leftrightarrow \neg y)| = 13) \leq (c|\varphi| = 2c) \Leftrightarrow 13 \leq 2c$[13]. Následně pro všechna $n = k + 1$[14] nastane jeden z následujících případů[15]:

- $\varphi = \neg\neg\varphi_a$, pak $|\textsc{Tse}(\varphi, i)| = |\textsc{Tse}(\varphi_a, i)|$. Dle indukčního předpokladu je $|\textsc{Tse}(\varphi, i)| \leq c|\varphi_a|$. Současně $|\varphi| = |\neg\neg\varphi_a| = |\varphi_a| + 2$. Zřejmě tedy platí, že je $(|\textsc{Tse}(\varphi, i)| \leq c|\varphi_a|) \leq (c|\varphi| = c(|\varphi_a| + 2) = c|\varphi_a| + 2c) \Leftrightarrow 0 \leq c$.

- $\varphi = (\varphi_a \vee \varphi_b)$, pak $|\textsc{Tse}(\varphi, i)| = |\textsc{Tse}(\varphi_a, i)| + |\textsc{Tse}(\varphi_b, j + 1)| + |\text{CNF}(x_l \leftrightarrow (x_j \vee x_k)| + 2$. Dle indukčního předpokladu a tabulky 1 je $|\textsc{Tse}(\varphi, i)| \leq c|\varphi_a| + c|\varphi_b| + 22 + 2$. Současně $|\varphi| = |(\varphi_a \vee \varphi_b)| = |\varphi_a| + |\varphi_b| + 3$. Zřejmě tedy platí, že je $(|\textsc{Tse}(\varphi, i)| \leq c|\varphi_a| + c|\varphi_b| + 24) \leq (c|\varphi| = c|\varphi_a| + c|\varphi_b| + 3c) \Leftrightarrow 8 \leq c$.

- $\varphi = \neg(\varphi_a \vee \varphi_b)$, pak $|\textsc{Tse}(\varphi, i)| = |\textsc{Tse}(\neg\varphi_a, i)| + |\textsc{Tse}(\neg\varphi_b, j+1)| + |\text{CNF}(x_l \leftrightarrow (x_j \wedge x_k)| + 2$. Dle indukčního předpokladu a tabulky 1 je $|\textsc{Tse}(\varphi, i)| \leq c|\neg\varphi_a| + c|\neg\varphi_b| + 23 + 2 = c|\varphi_a| + c|\varphi_b| + 2c + 25$. Současně $|\varphi| = |\neg(\varphi_a \vee \varphi_b)| = |\varphi_a| + |\varphi_a| + 4$. Zřejmě tedy platí, že je $(|\textsc{Tse}(\varphi, i)| \leq c|\varphi_a| + c|\varphi_b| + 2c + 25) \leq (c|\varphi| = c|\varphi_a| + c|\varphi_b| + 4c) \Leftrightarrow 2c + 25 \leq 4c \Leftrightarrow 25 \leq 2c$.

Dokázali jsme, že pro všechny možné průběhy výpočtu takové $c$ jistě existuje. Ze vzniklé soustavy nerovnic lze poté snadno odvodit, že v nejhorším případě je $c \geq 13$. Můžeme tedy říci, že $|\textsc{Tse}(\varphi, i)| \leq 13|\varphi|$, a tím pádem, že nárůst počtu znaků ve formuli $\varphi$ Tseytinovou transformací formule náleží do $O(|\varphi|)$. $\square$

[11] Důkaz by šel samozřejmě vést i vzhledem k počtu operátorů $\neg$ a $\vee$ ve formuli podobně jako na cvičení. Takto se mi ale postupovalo lépe.

[12] indukční předpoklad

[13] bázové případy pro $n = 1$ a $n = 2$

[14] Jistě existuje syntakticky správná formule pro libovolné $n > 1$, vždy je možné ve formuli mít $n - 1$ negací a následně výrokovou proměnnou.

[15] indukční krok

# 4 Nevyjádřitelnost množiny důsledků $T$ prvořádové teorie $T_C$

Množina důsledků je množina $T = \{\text{věta } \varphi \text{ jazyka } T_C \mid \varphi \text{ je platná v } T_C\} = \{G(e) \mid e \in \mathcal{T}\}$. Nevyjádřitelnost množiny v teorii $T_C = \langle F = \{C_{/2}\}, P = \emptyset\rangle$ dokážeme aplikací Tarského věty a tedy dokázáním, že pro množinu $A = \{n \in \mathbb{N} \mid H(n) \in \mathcal{T}\} \subseteq \mathbb{N}$ platí následující:

1. Množina $\overline{A} = \mathbb{N} \setminus A$ je vyjádřitelná pro každou množinu $A$.
2. Množina $A^* = \{n \in \mathbb{N} \mid G(E_n(n)) \in A\}$ je vyjádřitelná pro každou množinu $A$.

První bod zjevně platí v logice prvního řádu vždy, protože tato logika obsahuje negaci. Zřejmě $\overline{A}$ lze vyjádřit jako $\overline{A} = \{n \in \mathbb{N} \mid \neg(n \in A)\}$. Důkaz druhého bodu pak povedeme v následujícím výčtu:

1. Pro každý používaný symbol $a$ je nutné vhodně zvolit Gödlovo číslo $G(a)$, v našem případě symboly zakódujeme následovně na číslice dekadické soustavy:

$$\begin{aligned} ' &\longrightarrow 0 \\ 0 &\longrightarrow 1 \\ ( &\longrightarrow 2 \\ ) &\longrightarrow 3 \\ C &\longrightarrow 4 \\ v &\longrightarrow 5 \\ \neg &\longrightarrow 6 \\ \rightarrow &\longrightarrow 7 \\ \forall &\longrightarrow 8 \\ = &\longrightarrow 9 \end{aligned}$$

   Kde čísla kódujeme jako $0 = 0$, $1 = 0'$, $2 = 0''$, ..., $n = 0^{n \cdot \prime}$, jména proměnných jako $v$, $v'$, $v''$, ... Dále pro zjednodušení zápisu budeme uvažovat, že $\overline{n}$ značí kód $0^{n \cdot \prime}$ čísla $n$ s hodnotou $\overline{n} = G(0) \cdot 10^n + G(') \cdot 10^{n-1} + \ldots + G(') \cdot 10^1 + G(') \cdot 10^0 = 1 \cdot 10^n + 0 \cdot 10^{n-1} + \ldots + 0 \cdot 10^1 + 0 \cdot 10^0 = 10^n$.

2. Gödlovo číslo konkatenace dvou řetězců $u$ a $v$ lze vyjádřit za použití funkce $C$ tak, že $G(uv) = C(G(u), G(v))$, značme $G(u) \heartsuit G(v)$.

3. $G(E_e(n))$ lze vyjádřit za použití $e$ a $n$ tak, že, pokud $E_e(n)$ má volnou proměnnou $v$, $E_e(n) \Leftrightarrow \forall v(v = \overline{n} \rightarrow E_e)$. Tím pádem $G(E_e(n)) = 8 \heartsuit 5 \heartsuit 2 \heartsuit 5 \heartsuit 9 \heartsuit 10^n \heartsuit 7 \heartsuit e \heartsuit 3$.

4. Uvažujme, že $A = \{m \in \mathbb{N} \mid F(m) \in \mathcal{T}\}$, kde $F \in \mathcal{H}$. Pak $A^* = \{m \in \mathbb{N} \mid \exists v(v = G(F_m(m)) \wedge F(v)\}$. To jde ale na základě bodu 3 vyjádřit jako $A^* = \{m \in \mathbb{N} \mid \exists v(v = 8 \heartsuit 5 \heartsuit 2 \heartsuit 5 \heartsuit 9 \heartsuit 10^n \heartsuit 7 \heartsuit m \heartsuit 3 \wedge F(v)\}$

Dokázali jsme, že množina $\overline{A}$ i množina $A^*$ jsou vyjádřitelné pro každou množinu $A$. Z toho plyne, že množina $T$ nelze v prvořádové teorii $T_C$ vyjádřit. $\square$