TIN - složitost - demo

$L = \{a, bb, c\}$

(1) Uvažte známou operaci p na jazycích takovou, že

$$p(L) = \{w \in \Sigma^* \mid \exists w_p \in L : w_p \text{ je prefix } w\}$$

$p(L) = \{a\ldots,$
$bb\ldots,$
$c\ldots\}$

Dokažte, že pro každý jazyk $L \in \Sigma^*$ platí:

$L \in DTime[f(n)] \Rightarrow$
a) $p(L) \in DTime[n \cdot f(n)]$
b) $p(L) \in DSpace[f(n)]$
c) $p(L) \in NTime[f(n)]$

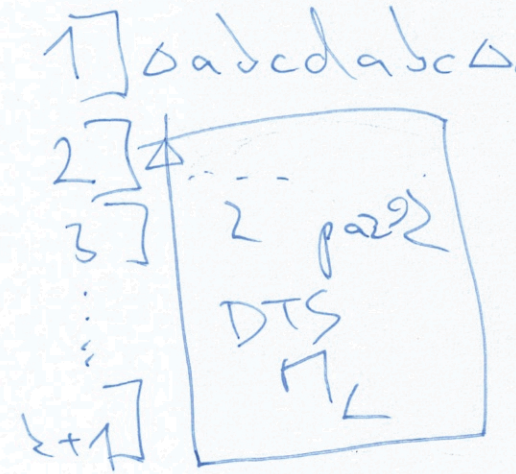
pro $f(n) \geq n$

Nechť Π_L je 2-pásková DTS rozhodující jazyk L v čase $O(f(n))$.

$\triangle abcdabc\triangle\ldots$

a) 1) Zkonstruujeme DTS $\Pi_{p(L)}$ - bude mít 2+1 pásek,
1. páska je vstupní.

2) $\Pi_{p(L)}$ bude postupně na 1. pásce z konce nazat symbolu a testovat, jestli zbývající podřetězec $w_p \in L(\Pi_L)$



3) vždy zkopíruje obsah 1. pásky na druhou a spustí simulaci Π_L na páskách $2-(2+1)$ (Π_L je obsaženo ve stavech řízení $\Pi_{p(L)}$).

- pokud Π_L přijme, $\Pi_{p(L)}$ taky přijme
- pokud Π_L odmítne a na 1. pásce jsou ještě nějaké symboly $\Pi_{p(L)}$ smaže poslední symbol na 1. pásce, smaže obsahy pásek $2-(2+1)$ a zkopíruje obsah 1. pásky na druhou

a zase spustí simulaci Π_L

- jinak $\Pi_{p(L)}$ odmítne

časová složitost $\Pi_{p(L)} = (n+1)\cdot\big(\underbrace{O(1)}_{\text{nalezení symbolu}} + \underbrace{O(n)}_{\text{kopírování + reset}} + \overbrace{f(n)}^{\text{běh } \Pi_L} + \underbrace{O(f(n))}_{\text{nalezení pásek pro } \Pi_L}\big) = (n+1)\cdot O(f(n)) = O(n\cdot f(n))$

---

4) Π_L mohl v čase $O(f(n))$ použít max. $O(f(n))$ buněk, velikost vstupu je n. Výsledná prostorová složitost $\Pi_{p(L)}$ je tedy $O(f(n)) + n = O(f(n))$

c) NTS $\Pi'_{p(L)}$ sestaví správný $\Pi_{p(L)}$ takže v 2. fázi "uhádne" správný prefix. Potom v analýze čas. složitosti zmizí člen $(n+1)$ a zůstane $O(f(n))$.

□

(2) Dokažte, že třída P je uzavřená na operace a) $\cdot$, b) *

a) $L_1 \in P, L_2 \in P \Rightarrow L_1 \cdot L_2 \in P$

Lze realizovat tak, že DTS rozhodující jazyk $L_1 \cdot L_2$ systematicky projde všechna rozdělení vstupu $w = a_1 a_2 \dots a_n$ na podřetězce w_1, w_2, kde $w = w_1 \cdot w_2$ a pro každé rozdělení ověří, zda $w_1 \in L_1$ a $w_2 \in L_2$. Pozdbplatí pro alespoň jedno rozdělení přijme, jinak odmítne. K testování $w_1 \in L_1$ a $w_2 \in L_2$ se použijí DTS rozhodující L_1 a L_2 s časovou složitostí $p_1(n)$ a $p_2(n)$.

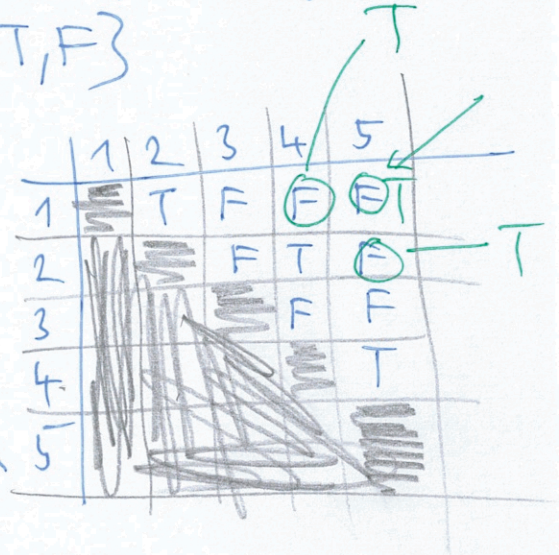
- složitost: $(n+1) \cdot (O(p_1(n)) + O(p_2(n)) + O(n)) \in O(n^2)$ pro $\forall \in \mathbb{N}$

$O(p_1(n))$: $w_1 \in L_1$ + výhod

$O(p_2(n))$: :

b) $L \in P \Rightarrow L^* \in P$ $\quad L = \{a, bc\}$ $\quad \Delta abca \Delta \ldots$

- nelze testit systematickým rozdělením vstupního řetězce na $1, 2, \ldots, n$ podřetězců
  → 2^{n-1} rozdělení na podřetězce
  - $\Omega(2^n)$
- pozorování: když narazíme na rozdělení „abc|de|fg" a „abc|def|g"
  - dostat výsledek $abc \in L$
- použijeme dynamické programování
- DTS Π, který bude rozhodovat L^* $(L \in P)$, sestrojen pro vstup $w = a_1 \ldots a_n$ matici $(n+1) \cdot (n+1)$ s hodnotami $\{T, F\}$



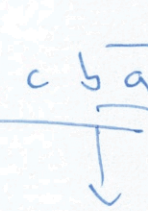
- na index (i,j) uloží $T \Leftrightarrow a_i \ldots a_{j-1} \in L$
- pro testování příslušnosti bude používat DTS Π' rozhodující L v poly. čase
- pole umíme zkonstruovat, jestli v grafu odpovídající matici existuje cesta z 1 do $(n+1)$

- spočítáme tranzitivní uzávěr matice pomocí Warshallova algoritmu: $O(n^3)$

Π pojeo $\Leftrightarrow$ v polici $(1, n+1)$ je T

(3) navrhněte efektivní reprezentaci fronty pomocí DS zásobníku.

- 1. myšlenka - budeme používat dva zásobníky a dle potřeby mezi nimi prvky přesouvat.




```
stack head ← []
stack tail ← []
enqueue(x) { tail.push(x) }
dequeue()
{   if (head = []){
        while(tail ≠ []){
            x ← tail.pop()
            head.push(x)
        }
    }
    return head.pop()
}
```

složitost (jednotlivá různá operace) a b c d] head

- v nejhorším případě:
  - enqueue : $1 \in O(1)$
  - dequeue : $\max(2, 3n+3) = 3n+3 \in O(n)$
- celkově n operací : $n \cdot O(n) = O(n^2)$
  → velmi nepraktické nadhodnocení

amortizovaná složitost – metoda účtů

operace	cena	kredit
enqueue	1	1+3
dequeue	head ≠ []: 2 head = []: 3n+3	3

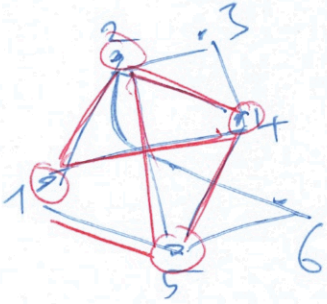
- Hlavní pozorování je, že cena operace dequeue záleží na počtu předchozích volání enqueue
- proto si každou operaci enqueue předplatíme 3 kredity (cena těla while cyklu)
- musíme ukázat, že kredit účtu nikdy nebude < 0.
- je snadné vidět, že platí invariant účet ≥ 3 · |tail|
- amortizovaná složitost sekvence n operací je pak $n \cdot \max(\text{kredit}) = 4n \in O(n)$

④ dokažte, že KLIKA $\leq_p$ SAT — CNF

LENP-complete

LENP ∧ LENP-hard

KLIKA: vrcholy neorientovaný graf $G=(V,E)$ a $k \in \mathbb{N}$. Existuje úplný podgraf s k uzly.



- nechť, že $V = \{v_1, \dots, v_n\}$ a očíslujeme uzly klíky $1, 2, \dots, k$.
- zavedeme proměnné x_i^j, kde pro $1 \leq i \leq n$, $1 \leq j \leq k$:

$$x_i^j = \begin{cases} \text{true pokud } v_i \text{ je } j\text{-tým uzlem klíky} \\ \text{false jinak} \end{cases}$$

- aby ohodnocení těchto proměnných odpovídalo klíce, musí platit:

1) každý uzel klíky musí odpovídat uzlu grafu

$$\varphi_1: \bigwedge_{1 \leq j \leq k} \bigvee_{1 \leq i \leq n} x_i^j$$

2) dva různé uzly klíky se musí navzájem na sebe uzel grafu

$$\varphi_2: \bigwedge_{1 \leq j_1 < j_2 \leq k} \bigwedge_{1 \leq i \leq n} \left(\neg x_i^{j_1} \vee \neg x_i^{j_2} \right)$$

3) dva různé uzly grafu se nemají reprezentovat stejným uzlem sítě

$$\varphi_3 : \bigwedge_{1 \le i_1 < i_2 \le n} \bigwedge_{1 \le j \le 2} \left(\neg X^j_{i_1} \vee \neg X^j_{i_2} \right)$$

4) každé dva uzly sítě musí být navzájem na uzly grafu, které jsou spojeny hranou, tj. nemá existovat dva různé uzly sítě nepropojené hranou

$$\varphi_4 : \bigwedge_{1 \le j_1 < j_2 \le 2} \bigwedge_{\substack{1 \le i_1 < i_2 \le n \\ \{v_{i_1}, v_{i_2}\} \notin E}} \left(\neg X^{j_1}_{i_1} \vee \neg X^{j_2}_{i_2} \right)$$

- výsledná formule je $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$ (je v CNF)
- konstrukce formule proběhne v poly. čase a zachovává pravdivost:

G má síť $\Leftrightarrow$ el. $\Leftrightarrow \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$ je splnitelná

(5) SUBSET-SUM: Nechť $S \subseteq \mathbb{N}$ a $w \in \mathbb{N}$. Existuje $R \subseteq S$ t.ž. $\sum_{r \in R} r = w$?

Dokažte, že SUBSET-SUM $\in$ NP-complete.

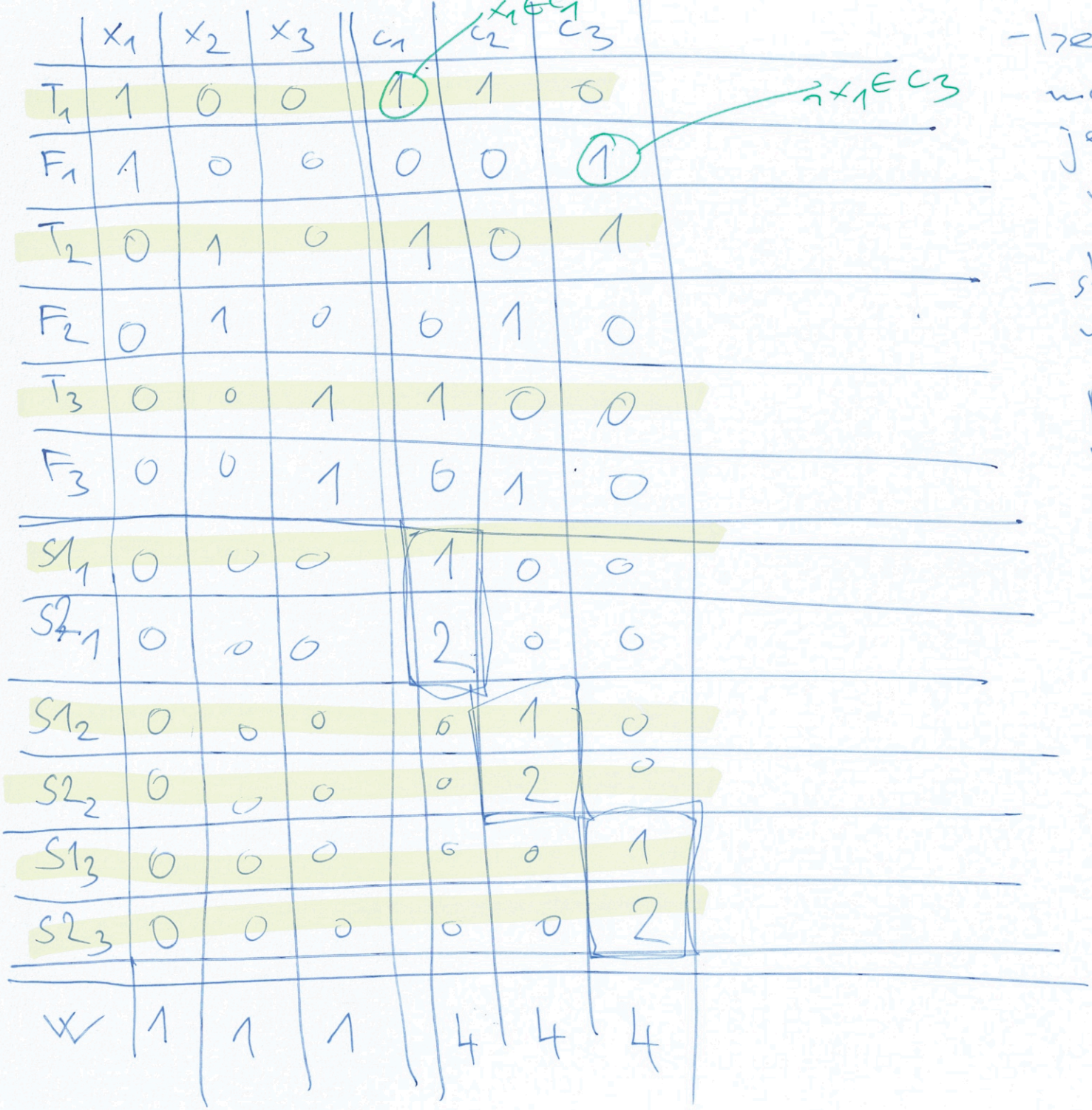
a) SUBSET-SUM $\in$ NP: jednoduše

b) SUBSET-SUM $\in$ NP-hard: redukce z 3SAT $\qquad$ 3SAT $\leq_p$ SUBSET-SUM

- nejprve formule φ: $\overbrace{(x_1 \vee x_2 \vee x_3)}^{c_1} \wedge \overbrace{(x_1 \vee \neg x_2 \vee \neg x_3)}^{c_2} \wedge \overbrace{(\neg x_1 \vee x_2)}^{c_3}$

s n proměnnými a k klauzulemi

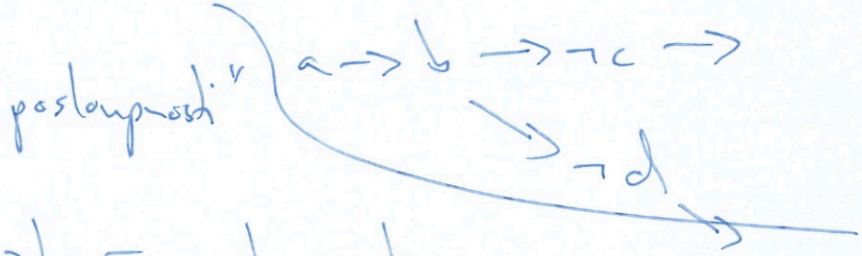
- vytvoříme množinu $S = \{T_1, \dots, T_n, F_1, \dots, F_n, S1_1, \dots, S1_k, S2_1, \dots, S2_k\}$ čísel, jejichž délka v desítkovém zápisu je $n+k$, takto:



- lze předpokládat, že každý model v odpovídá právě jedné množině
např.: $\{x_1 \to T, x_2 \to T, x_3 \to T\}$
- strategie S a v vyhrává, že pro každou proměnnou x_i musí být v R právě jedna z T_i/F_i a pro každou klauzuli c_j musí být v R alespoň jedna T_i/F_i s číslem 1 na pozici dané klauzule.

⑥ Určete speciální případ SAT, kde každá klauzule má právě dva literály: 2SAT. Triviálně platí, že 2SATENP. Dokažte, že 2SATEP.

- Klíčové pozorování: pokud máme klauzuli $(a \vee b)$, pak platí, že $(\neg a \to b) \wedge (\neg b \to a)$
- tedy množina klauzulí generuje „posloupnosti" implikací



- pokud existuje proměnná x t.ž. existují posloupnosti $x \to \ldots \to \neg x$ a $\neg x \to \ldots \to x$, tak víme, že φ je nesplnitelná. Hledání těchto posloupností realizuje hledání cesty v grafu.
- nechť $X = \{x_1, \ldots, x_n\}$ je množina proměnných v φ a C je množina klauzulí
- sestrojme graf $G = (V, E)$, kde $V = \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$ a

$$(a,b) \in E \Leftrightarrow (\overline{a} \vee b) \in C \quad \text{kde} \quad \overline{a} = \begin{cases} x & \text{pro } a = \neg x \\ \neg x & \text{pro } a = x \end{cases}$$

tj. pro $(a \vee b) \in C$ přidáme hrany $(\overline{a}, b)$ a $(\overline{b}, a)$

φ je nesplnitelná $\Leftrightarrow \exists x \in X$ t.ž. v G existují cesty $x \rightsquigarrow \neg x$ a $\neg x \rightsquigarrow x$

$\Leftarrow$ - zřejmé na základě pozorování

$\Rightarrow$ lze ukázat obdobně - pokud neexistuje „dostatečně" cesty v G_φ, tak lze sestavit model

- dále, velikost G je v $O(|\varphi|)$ a hledání cesty je v $O(n^2)$
- lze rozšířit na splnitelnost Hornových klauzulí - (max. 1 pozitivní literál)

$(x \vee \neg y \vee \neg z) \Leftrightarrow ((y \wedge z) \rightarrow x)$