Complexity/Složitost (SLOa) – 2022/2023

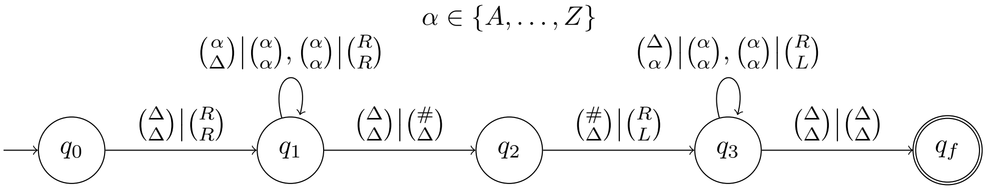
Homework assigment 1
Domácí úloha 1

Domink Nejedlý (xnejed09)

1. Navrhněte Turingův stroj, který implementuje reverzi vstupního řetězce nad latinskou abecedou $\Sigma = \{A, \ldots, Z\}$.

   - Vstup je ve tvaru $\Delta w \Delta^\omega$, kde w je vstupní řetězec.
   - Když Turingův stroj zastaví, první páska obsahuje $\Delta w \# w^R \Delta^\omega$, kde w^R je reverze w (např. pro $w = HELLO$, $w^R = OLLEH$)
   - Pokud navrhnete vícepáskový stroj, na obsahu ostatních pásek nezáleží.
   - Odhadněte horní meze jeho časové a prostorové složitost.

   Řešení: Navrhněme dvoupáskový deterministický Turingův stroj M implementující požadovanou reverzi vstupního řetězce nad latinskou abecedou Σ. Předpokládejme, že na počátku je vstup zapsán na 1. pásce M, druhá páska je prázdná a obě čtecí hlavy jsou na nejlevějších pozicích odpovídajících pásek. M lze definovat následovně:



Obrázek 1: Přechodový diagram dvoupáskového deterministického TS M

   1) M nejprve posune čtecí hlavy na obou páskách z prvního symbolu Δ o jednu pozici doprava.
   2) M v lineárním čase nakopíruje obsah své první pásky na druhou – znak po znaku postupně přepisuje symboly z první pásky na druhou, přičemž po každém přepisu vždy provede posun obou čtecích hlav o jednu pozici doprava, dokud se pod oběma z nich nenachází symbol Δ.
   3) M přepíše symbol Δ na první pásce za symbol # (na druhé pásce symbol Δ ponechá) a provede posun čtecích hlav. Tentokrát však posune čtecí hlavu na první pásce ze symbolu # o jednu pozici doprava a čtecí hlavu na druhé pásce ze symbolu Δ o jednu pozici doleva.
   4) M v lineárním čase postupně zprava doleva kopíruje obsah druhé pásky na první zleva doprava – znak po znaku přepisuje symboly z druhé pásky na první, přičemž po každém přepisu jednoho znaku posune čtecí hlavu na první pásce o jednu pozici doprava a na druhé pásce o jednu pozici doleva, dokud se pod oběma z nich nenachází symbol Δ, čímž běh stroje M úspěšně končí.

   Je zřejmé, že tímto způsobem má M na konci svého běhu pro každý vstup tvaru $\Delta w \Delta^\omega$, kde $w \in \Sigma^*$, na první pásce výstup tvaru $\Delta w \# w^R \Delta^\omega$. V případě špatného formátu vstupního řetězce (případně i části vstupu, jež během svého výpočtu M navštíví – např. za symbolem Δ za řetězcem w se nachází neočekávaný symbol, na který čtecí hlava přistoupí) pak běh stroje M abnormálně skončí. M je tedy úplný.

   Časová složitost výpočtu M je pak zřejmě již od pohledu v lineární závislosti k délce vstupního řetězce. Platí tedy $T_M(n) = 1 + 2n + 2 + 2n + 1 = 4n + 4 \in \mathcal{O}(n)$. Z hlediska prostoru pak M pro vstupní řetězec délky n navštíví nanejvýš $2n + 3$ polí první pásky a $n + 2$ polí pásky druhé. Prostorová složitost výpočtu M tedy odpovídá $S_M(n) = 2n + 3 + n + 2 = 3n + 5 \in \mathcal{O}(n)$.

2. Navrhněte RAM program, který pro vstupní vektor $I = (min, max, n)$ vypočítá číslo k takové, že $min \leq (k * n) \leq max$. Předpokládejme, že všechna vstupní čísla jsou větší než nula, tj. $min, max, n > 0$. Po instrukci $HALT$ bude registr r_0 obsahovat číslo k, nebo -1, pokud takové k neexistuje. (Poznámka: Není nutné implementovat optimální algoritmus.)

   - Analyzujte jednotkovou (uniformní) časovou a prostorovou složitost a odhadněte horní meze.
   - Analyzujte logaritmickou časovou a prostorovou složitost a odhadněte horní meze.

Nezapomeňte, že časová a prostorová složitost RAM programu je odhadována s ohledem na velikost jeho vstupních dat (tj. počet bitů vstupního vektoru (min, max, n)).

Řešení: RAM program Π řešící danou úlohu může pracovat následovně:

Algoritmus 1: RAM program Π pro výpočet k takového, že $min \leq (k * n) \leq max$

Vstup: vektor $I = (min, max, n)$, kde $min, max, n > 0$

Výstup: registr r_0 obsahující k, nebo -1 (v případě neexistence k)

```
1 READ 3
2 STORE 3
3 READ 2
4 STORE 2
5 READ 1
6 STORE 1
7 LOAD 4
8 ADD = 1
9 STORE 4
10 LOAD 2
11 SUB 3
12 STORE 2
13 LOAD 1
14 SUB 3
15 JPOS = 6
16 LOAD 2
17 JNEG = 20
18 LOAD 4
19 HALT
20 LOAD = −1
21 HALT
```

Nejprve jsou jednotlivé složky vstupního vektoru I (prostřednictvím jejich postupného načítání do registru r_0) uloženy do registrového pole stroje s náhodným přístupem – min do registru r_1, max do registru r_2 a n do registru r_3. Poté je od hodnot min a max (resp. jejich průběžných hodnot držených v registrech r_1 a r_2) opakovaně odečítána hodnota n (z registru r_3), dokud je průběžná hodnota min kladná ($min > 0$), přičemž je v registru r_4 uchováván celkový průběžný počet vykonaných odečtení. Nutno poznamenat, že před každým odečtem, porovnáním i inkrementací je daná hodnota vždy načtena do registru r_0, jelikož ten slouží jako akumulátor. Jakmile je pak průběžná hodnota $min \leq 0$, je do registru r_0 načtena průběžná hodnota max, u níž je zjišťováno, zdali je negativní ($max < 0$).

- Pokud ano ($max < 0$), potom žádné k takové, že $min \leq (k * n) \leq max$ neexistuje. Do registru r_0 je tedy načtena hodnota -1, která je následně při ukončení programu navrácena.
- Pokud ne ($max \geq 0$), pak je řešením k zřejmě hodnota uložená v registru r_4. Je tedy načtena do registru r_0 a při následném ukončení programu navrácena.

Z analýzy kódu RAM programu Π je zřejmé, že smyčka od instrukce 6 po instrukci 15 (včetně) je vždy provedena $\lceil \frac{min}{n} \rceil$-krát. V nejhorším případě pak, kdy $n = 1$, min-krát. Nechť $l = \text{len}(I) = \text{len}(min) + \text{len}(max) + \text{len}(n)$, kde $\text{len}(x)$ je délka binární reprezentace x, představuje délku binární reprezentace vstupu I. RAM program Π zcela zřejmě vždy provede $5 + 10\lceil \frac{min}{n} \rceil + 4 = 10\lceil \frac{min}{n} \rceil + 9$, v nejhorším případě $(n = 1)$ pak $10min + 9$, kroků, přičemž jistě platí $\lceil \frac{min}{n} \rceil \leq min \leq 2^l$. Z toho důvodu tedy jednotková časová složitost RAM programu Π náleží do $\mathcal{O}(2^l)$ $(T_{\Pi}^{uni}(l) \in \mathcal{O}(2^l))$. Vzhledem k tomu, že pro každý vstup RAM program Π používá 3 vstupní registry (i_1, i_2 a i_3) a 5 registrů ze svého registrového pole (r_0, r_1, r_2, r_3 a r_4) je jeho uniformní prostorová složitost v $\mathcal{O}(1)$ $(S_{\Pi}^{uni}(l) = 3 + 5 = 8 \in \mathcal{O}(1))$.

Jelikož RAM program Π dostává na vstupu pouze kladná čísla a ve smyčce pak postupně odečítá od hodnot min a max hodnotu n, přičemž zvyšuje průběžnou hodnotu potenciálního k, jež začíná na hodnotě 0, vždy o hodnotu 1, je zřejmé, že hodnotou s nejdelší binární reprezentací, se kterou program pracuje je právě $\max(min, max, n)$ (tedy maximální hodnota z hodnot min, max a n). Vzhledem ke způsobu, jakým totiž RAM program Π funguje, jistě vždy platí, že $k \leq min$ (pro $n = 1$ platí $k = min$). Současně pak ani odečtení n od hodnoty 1, což je nejnižší možná hodnota průběžných hodnot min a max před posledním odečtením n, nevyprodukuje záporné číslo s delší binární reprezentací než má n. Dále pak jistě platí, že $\text{len}(\max(min, max, n)) \leq l = \text{len}(I)$, jelikož ani délka binární reprezentace nejvyšší vstupní hodnoty nemůže přesáhnout délku celého vstupu, jehož je součástí. Z toho důvodu logaritmická prostorová složitost RAM programu Π náleží do $\mathcal{O}(l)$ $(S_{\Pi}^{log}(l) \in \mathcal{O}(l))$, neboť tento program pracuje s konečným počtem registrů a délka binární reprezentace žádného čísla v nich uloženého nepřesáhne délku binární reprezentace vstupu $l = \mathrm{len}(I)$. Logaritmická časová složitost pak vychází z maximálního celkového počtu instrukcí, které RAM program Π pro vstup délky l provede, a maximální délky binární reprezentace čísel, jimiž může program manipulovat (ta, jak je již výše zmíněno, nikdy nepřesáhne l). Proto tedy logaritmická časová složitost RAM programu Π náleží do $\mathcal{O}(n2^n) = \mathcal{O}(2^n 2^{log_2(n)}) = \mathcal{O}(2^{n+log_2(n)})$ ($T_\Pi^{log}(l) \in \mathcal{O}(2^{n+log_2(n)})$).

Jednotková časová složitost	$T_\Pi^{uni}(l) \in \mathcal{O}(2^l)$
Jednotková prostorová složitost	$S_\Pi^{uni}(l) \in \mathcal{O}(1)$
Logaritmická časová složitost	$T_\Pi^{log}(l) \in \mathcal{O}(n2^n) = \mathcal{O}(2^{n+log_2(n)})$
Logaritmická prostorová složitost	$S_\Pi^{log}(l) \in \mathcal{O}(l)$

Tabulka 1: Shrnutí analýzy složitosti RAM programu Π

3. Nechť L je regulární jazyk (tj. jazyk přijímaný konečným automatem). Odhadněte funkce $f(n)$ a $g(n)$ tak, že $L \in DTIME(f(n))$ a $L \in DSPACE(g(n))$. Dokažte své tvrzení.

Řešení: Víme, že pro každý regulární jazyk existuje úplně definovaný deterministický konečný automat – konečný automat bez ε-přechodů, jež může z každého svého stavu přes každý symbol vstupní abecedy přejít do nanejvýš jednoho následujícího stavu, tzn. že z jednoho stavu nemůže prostřednictvím jednoho symbolu vstupní abecedy přejít do dvou a více různých stavů, přičemž je jeho přechodová funkce totální. Takový konečný automat přijímající jazyk L pak každý řetězec w jedním průchodem vstupní pásky (tj. přečtením celého vstupního řetězce w) zleva doprava buď přijme ($w \in L$), či odmítne ($w \notin L$). V každém případě však tento automat provede pro každý řetězec délky n právě n přechodů, přičemž navštíví právě n polí vstupní pásky. Vyvoďme tedy následující tvrzení.

Věta: Pro každý regulární jazyk L platí, že $L \in DTIME(n)$ a $L \in DSPACE(n)$.

Odhadujme tedy, že pro funkce $f(n)$ a $g(n)$ ze zadání platí $f(n) = g(n) = n$. Nyní tohle tvrzení dokažme.

Důkaz. Mějme libovolný regulární jazyk L. Pak víme, že jistě existuje úplně definovaný deterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$ takový, že $L(M) = L$ (přijímající jazyk L). Bez újmy na obecnosti předpokládejme, že $q_0', q_f' \notin Q$ a $\Delta \notin \Sigma$. Nyní sestrojme (jednopáskový) úplný deterministický Turingův stroj

$$M' = (Q \cup \{q_0', q_f'\}, \Sigma, \Sigma \cup \{\Delta\}, \delta', q_0', q_f'),$$

kde

$$\delta' = \{((q_0', \Delta), (q_0, R))\} \cup \{((p, a), (q, R)) \mid ((p, a), q) \in \delta\} \cup \{((f, \Delta), (q_f', \Delta))\}.$$

M' pracuje následujícím způsobem:

1) M' nejprve posune čtecí hlavu z počátečního symbolu Δ o jednu pozici doprava, čímž se ve svém stavovém řízení dostane do bodu, kdy je schopen začít zpracovávat vstup způsobem odpovídajícímu čtení řetězce pomocí automatu M.
2) M' postupně zpracovává vstup znak po znaku zleva doprava – po každém čtení symbolu posune čtecí hlavu na pásce o jednu pozici doprava – stejně jako konečný automat M, dokud se pod jeho čtecí hlavou nenachází symbol Δ. V tomto případě pak M' abnormálně zastaví (nelze provést žádný přechod – zpracovávaný řetězec nenáleží do jazyka L), nebo přejde do koncového stavu q_f' a úspěšně skončí.

Vidíme, že platí $L(M') = L(M) = L$ a také, že M' je opravdu úplný (vždy zastaví). Dále lze vypozorovat, že na vstupním řetězci délky n, provede M' při jeho přijetí $n + 2$ kroků a při jeho odmítnutí pak kroků $n + 1$. V obou případech pak navštíví vždy $n+2$ polí pásky. Zřejmě tedy platí, že $L(M') = L(M) = L \in DTIME(n)$ a $L(M') = L(M) = L \in DSPACE(n)$. $\square$