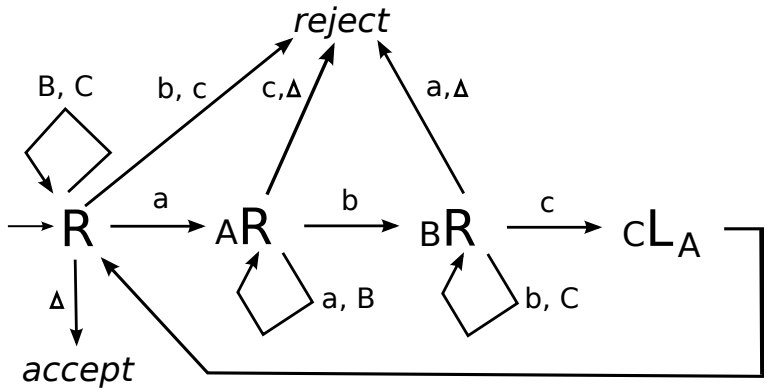
2. května 2010

1. úkol z předmětu Složitost

Petr Zemek
xzemek02@stud.fit.vutbr.cz
Fakulta Informačních Technologií, Brno

Příklad 1

Následující TS M, reprezentovaný kompozitním diagramem[1], rozhoduje jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$.



M funguje následujícím způsobem. Pokud se na pásce nenachází žádný vstup (resp. vstup nulové délky), tak M přijímá. V opačném případě postupuje iteračně, a to tak, že v každé iteraci přepíše nejlevější symbol a na velké A, nejlevější b na B a nejlevější c na C. V případě, že stroj při této činnosti narazí na nesrovnalost (symboly nejsou v pořadí $a \ldots ab \ldots bc \ldots c$ či počet symbolů a je větší než počet b či c), tak zamítne. Jakmile takto přepíše nejpravější symbol a, tak projde páskou až nakonec, přičemž kontroluje, zda byly přepsány všechny symboly na jejich velké verze (počet symbolů a není menší než počet symbolů b či c). Pokud tomu tak je, stroj přijímá, jinak zamítá.

Pro potřebu analýzy složitosti, nechť n značí délku vstupního řetězce.

Analýza časové složitosti

M provede $O(n)$ iterací (za každý symbol a se provede jedna iterace), kde časová složitost každé iterace je $O(n)$ (prochází se všechny nezpracované symboly a, poté všechny symboly b či B a všechny symboly C). Nakonec dojde k ověření, že ve vstupním řetězci už nezbyly žádné symboly a, b, či c, což má časovou složitost $O(n)$ (prochází ve vstupní řetězec až po první prázdné políčko). Celková časová složitost je tedy $O(n)O(n) + O(n) = O(n^2)$.

Analýza prostorové složitosti

Jelikož M během své činnosti nic nezapisuje mimo oblast, na které se nachází vstupní řetězec, je jeho prostorová složitost $O(n)$.

[1] Použitá notace je podle předmětu TIN; *accept* značí přijetí vstupu, *reject* značí zamítnutí vstupu.

Příklad 2

Následující RAM program[2] provádí násobení n čísel $x_1, x_2, \ldots, x_n$, kde $n > 0$, reprezentovaných vstupním vektorem $I = (n, x_1, x_2, \ldots, x_n)$. Výsledek je na konci uložen v registru r_0.

```
1             READ   1        ; r_0 := n
2             STORE  1        ; r_1 := r_0
3             READ   2        ; r_0 := x_1
4             JZERO  end0     ; pokud je na vstupu 0, pak je výsledek 0
5             STORE  2        ; r_2 := r_0
6    next:                    ; zpracování následujícího čísla
7             LOAD   1        ; r_0 := r_1
8             SUB    =1       ; r_0 := r_0 - 1
9             JZERO  end      ; žádná další čísla k násobení
10            STORE  1        ; r_1 := r_0
11            ADD    =2       ; r_0 := r_0 + 2 (kvůli indexaci vstupu)
12            READ   ↑0       ; r_0 := I_{r_0} (načtení dalšího čísla ze vstupu)
13            JZERO  end0     ; pokud je na vstupu 0, pak je výsledek 0
14            STORE  4        ; r_4 := r_0 (následující 4 instrukce slouží k r_3 := r_2)
15            LOAD   2        ; r_0 := r_2
16            STORE  3        ; r_3 := r_0
17            LOAD   4        ; r_0 := r_4
18   mult2:                   ; vynásobení dvou čísel pomocí sčítání
19            SUB    =1       ; r_0 := r_0 - 1
20            JZERO  next     ; násobení bylo dokončeno, jdeme na další číslo
21            STORE  4        ; r_4 := r_0
22            LOAD   2        ; r_0 := r_2
23            ADD    3        ; r_0 := r_0 + r_3
24            STORE  2        ; r_2 := r_0
25            LOAD   4        ; r_0 := r_4
26            JPOS   mult2
27   end0:                    ; konec násobení (výsledkem je 0)
28            LOAD   =0       ; r_0 := 0
29            HALT
30   end:                     ; konec násobení (výsledek může být nenulový)
31            LOAD   2        ; r_0 := r_2
32            HALT
```

Program používá registr r_1 pro uložení počtu čísel, která je ještě třeba vynásobit (vždy před zahájením násobení dvou čísel se dekrementuje a pokud je jeho obsah nulový, tak se program ukončí). Průběžný výsledek násobení je uložen v registru r_2. Registry r_3 a r_4 slouží jako pracovní registry při násobení dvou čísel. Dokud je co násobit, tak program postupně násobí vždy dvě čísla (aktuální výsledek uložený v registru r_2 s dalším číslem na vstupu) pomocí opakovaného sčítání (přičítání obsahu registru r_3 k r_2).

Pro potřebu analýzy složitosti, nechť:

- $l(x)$ značí délku zápisu čísla x v binárním zápise,
- n je počet čísel, které se násobí (první hodnota vstupu),
- $w = l(n)l(x_1)l(x_2)\ldots l(x_n)$ je délka vstupního vektoru v binárním zápise,
- $k = max(\{x_1 \cdot x_2 \cdot \ldots \cdot x_n, n, 4\})$ je největší číslo, se kterým se v programu pracuje,
- $m = max(\{x_i \mid 2 \leq i \leq n\} \cup \{0\})$ je maximální hodnota násobitele (0, pokud $n = 1$).

[2] Používám mírně pozměněnou notaci, kdy místo explicitních čísel řádků uvažuji návěští, aby to bylo přehlednější a udržovatelné. Znak ';' uvozuje komentář.

Analýza časové složitosti

Řádky 1–5 mají konstantní složitost $O(1)$. Na řádcích 7–26 se nachází vnější cyklus, který pro každé číslo na vstupu (kromě prvního) provede vynásobení tohoto čísla s mezivýsledkem násobení. Tento cyklus se tedy provede $O(n)$ krát. Dále se v tomto cyklu na řádcích 19–26 provádí vnitřní cyklus násobení dvou čísel, jehož složitost je $O(2^m)$ (v každém cyklu se sníží hodnota násobitele, což je počet sčítání, které se mají provést, o 1). Řádky 28–29 a 31–32 mají opět konstantní složitost $O(1)$. Celková jednotková časová složitost programu je tedy $O(1) + O(n)O(2^m) + O(1) = O(n \cdot 2^m)$. Logaritmická časová složitost je pak $O(n \cdot 2^m \cdot l(k))$.

Analýza prostorové složitosti

Program využívá registry r_0 až r_4, čili jednotková prostorová složitost je $O(5 + w) = O(w)$. Logaritmická prostorová složitost je pak $O(5 \cdot l(k) + w) = O(l(k) + w)$.