KMI/VCS1 – Vyčíslitelnost a složitost

Paměťová složitost: třídy L a NL

Jan Konečný

21. prosince 2013

Paměťová složitost

Zabýváme náročností výpočetních problémů z hlediska paměti, která je potřeba k jejich řešení.
Připomínka:

Definice

Paměťová složitost TS T je funkce $f : N \to N$, kde $f(n)$ je maximální počet políček použitých při výpočtu nad jakýmkoli vstupem délky n.

a pro NTS:

Definice

Paměťová složitost NTS T je funkce $f : N \to N$, kde $f(n)$ je maximální počet políček použitých při výpočtu nad jakýmkoli vstupem délky n v jakékoli větvi výpočtu.

Změna

Budeme uvažovat následující variantu TS:

TS s dvěma páskami:

- vstupní čtecí páska – je na ní zapsán vstup, nedá se zapisovat (read only)
- pracovní páska – lze číst i zapisovat.

Definice

Paměťová složitost TS T je funkce $f : N \to N$, kde $f(n)$ je maximální počet políček **pracovní pásky** použitých při výpočtu nad jakýmkoli vstupem délky n.

a pro NTS:

Definice

Paměťová složitost NTS T je funkce $f : N \to N$, kde $f(n)$ je maximální počet políček **pracovní pásky** použitých při výpočtu nad jakýmkoli vstupem délky n v jakékoli větvi výpočtu.

Paměťová složitost

Definice

Jazyk A nazveme *rozhodovaný v paměti* $f(n)$ pokud existuje TS, který má paměťovou složitost $f(n)$.

Analogicky pro NTS:

Definice

Jazyk A nazveme *nedeterministicky rozhodovaný v paměti* $f(n)$ pokud existuje NTS, který má paměťovou složitost $f(n)$.

Třídy paměťové složitosti:

Definice

$\mathrm{SPACE}(f(n)) = \{A \mid$ Jazyk A je rozhodovaný v paměti $\mathcal{O}(f(n))\}$.
$\mathrm{NSPACE}(f(n)) = \{A \mid$ Jazyk A je nedet. rozhodovaný paměti $\mathcal{O}(f(n))\}$.

Třídy L a NL

$$\mathrm{L} = \mathrm{SPACE}(\log(n))$$
$$\mathrm{NL} = \mathrm{NSPACE}(\log(n))$$

Poznámka

Omezujeme se na logaritmickou paměťovou složitost ze stejného důvodu, jako jsme se předtím omezili na polynomickou časovou složitost: problémy řešitelné v logaritmické paměti považujeme za řešitelné efektivně.

Příklad

$0^k 1^k \in \mathrm{L}$:

Sestrojíme TS T, který to řeší v log. paměti.

1. *Zapiš $n_0 = 0, n_1 = 0$*
2. *Na vstupní pásce jeď postupně doleva a za každou 0 zvyš n_0, dokud nenarazíš na $x \neq 0$.*
3. *Na vstupní pásce jeď postupně doleva a za každou 1 zvyš n_1, dokud nenarazíš na $x \neq 1$.*
4. *pokud je $x = 0$ zamítni.*
5. *pokud je $n_0 = n_1$ přijmi.*
6. *jinak zamítni.*

Potřebujeme si pamatovat pouze n_0, n_1 – log. paměť.

Připomínka:

$$\mathrm{PATH} = \{[G, s, t] \mid G \text{ je orientovaný graf, který má cestu z } s \text{ do } t\}$$

Příklad

$\mathrm{PATH} \in \mathrm{NL}$

Sestrojíme T_{PATH}, který řeší PATH v log. paměti:

TS T_{PATH} pro $[G, s, t]$:

1. *Nastaví $m = 0, u = s$ (= počet provedených kroků m, aktuální uzel u)*
2. *pokud $u = t$ přijmi*
3. *pokud $m =$ počet stavů G, zamítni.*
4. *nedeterministicky vyber souseda v, zapiš $u = v$, m++, opakuj od kroku 2.*

Stačí mít zapsáno m, u – logaritmická velikost.

Neví se, zda $\mathrm{PATH} \in \mathrm{L}$.

NL-úplnost

Definice

Přepisovač v logaritmické paměti je TS s

- vstupní čtecí páskou (read-only),
- pracovní čtecí/zapisovací páskou,
- výstupní zapisovací páskou (write-only),

pracovní páska může obsahovat $\mathcal{O}(\log n)$ symbolů.

Přepisovač v logaritmické paměti M vyčisluje funkci $f : \Sigma^* \to \Sigma^*$, kde $f(w)$ je řetězec, který je na výstupní pásce, jakmile M zastaví.

f nazýváme funkcí vyčislovanou v logaritmické paměti.

Jazyk A je redukovatelný v logaritmické paměti na jazyk B, pokud je redukovatelný na B s použitím funkce vyčislované v logaritmické paměti. Zapisujeme $A \leq_{\mathrm{L}} B$.

Definice

Jazyk B je NL-úplný, pokud

- $B \in \mathrm{NL}$,
- každý $A \in \mathrm{NL}$ je redukovatelný v logaritmické paměti na B.

Věta

Pokud $A \leq_{\mathrm{L}} B$ *a* $B \in \mathrm{L}$, *pak* $A \in \mathrm{L}$.

Důkaz na tabuli.

Důsledek

Pokud je jakýkoli NL-úplný jazyk v L, pak L=NL.

Věta

PATH *je* NL-*úplný jazyk.*

Důkaz na tabuli.

Důsledek

$$\mathrm{NL} \subseteq \mathrm{P}.$$

Důkaz na tabuli.