

Cvičení z Algoritmů a Datových Struktur 1

Matej Lieskovský

LS2324 - 2. cvičení

1 Levenstein bez rekurze

Na přednášce jste viděli, jak spočítat editační vzdálenost dvou řetězců pomocí rekurze a jak jej zrychlit pomocí kešování.

Navrhňte, jak tento problém řešit bez použití rekurze. (Bez asymptotického zpomalení algoritmu.)

2 Třídění výběrem na RAMu

Napište implementaci SelectionSortu v modelu RAM. Na vstupu je v buňce [0] délka posloupnosti celých čísel a samotná posloupnost se nachází v buňkách [1]-[[0]].

3 Příklad pro RAM

Budeme potřebovat si z RAMu udělat použitelný model počítače. Rozmyslete si, jak do RAMu přepisovat následující konstrukce:

3.1 Základy

- Složitější podmínky (AND a OR)
- Kontrola v konstantním čase, jestli je číslo mocninou dvojky

3.2 Více polí

Co když váš program potřebuje více různě velkých polí? Předpokládejte, že nevíte předem, jaká pole budete potřebovat, ale vždy při vytvoření pole víte jeho velikost.

3.3 Volání funkcí

Předpokládejte, že máte nějakou funkci (třeba třídění pole), kterou chcete ve vašem programu používat.

1. Jak upravit funkci třídění pole, aby uměla setřídřit libovolné pole, které budete chtít?
2. Jak to udělat, pokud chcete tuto funkci volat někde uprostřed vašeho programu?
3. Co když ji budete volat na několika různých místech ve vašem programu?
4. Co když je funkce samotná rekurzivní?

4 Zneužíváme RAM

Mějme model RAM s neomezenou velikostí čísel a s jednotkovou cenou instrukce s neomezenými čísly.

4.1 Komprese čísel

Vymyslete kódování (a následně dekódování), které zvládne do jednoho čísla uložit:

1. dvě libovolná celá čísla
 2. n libovolných celých čísel (kde n dekóder zná)
 3. n libovolných celých čísel (kde n dekóder nezná)
- (Kódování a dekódování nemusí být příliš efektivní.)

4.2 Konstantní paměť

Navrhněte postup, jak v případě neomezené kapacity paměťové buňky pozměnit libovolný program na RAMu tak, aby používal jen konstantně mnoho buňek. Program můžete libovolně zpomalit.

Bonus: Kolik nejméně buněk budete potřebovat?

Dá se dostat na 5

4.3 Rychlé násobení matic

Jak v čase $O(n)$ zakódovat dva vektory n celých čísel, abychom mohli v konstantním čase spočítat skalární součin dvou vektorů?

- Předpokládejte, že čísla jsou z rozsahu $[0,9]$
- Předpokládejte, že čísla mohou být libovolně velká

Jak z toho odvodit algoritmus pro násobení matic v čase $O(n^2)$?

5 Bonus: Rekurzivní hádanky

Co dělají následující funkce?

$f(x,y)$:

```
if x==0 => return y
else => return f((x&y) << 1, x^y)
```

$g(x,y)$:

```
if y==0 => return 0
else if even(y) => return 2*g(x, y/2)
else => return 2*g(x, y/2) + x
```