ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd

Programovací techniky

Randomizovaný algoritmus

Autor: **Antonín NEUMANN**
Akademický rok: 2012/2013

Zadání

Vymyslete nebo najděte v dostupné literatuře randomizovaný algoritmus jako efektivní náhradu deterministického algoritmu pro libovolný problém.

Problém

Jako problém, na kterém bych chtěl prezentovat randomizovaný algoritmus jako efektivní náhradu deterministického algoritmu jsem zvolil řadící algoritmus. Konkrétně řadící algoritmus Quicksort, česky též známý jako třídění rozdělováním.

Popis algoritmu

Myšlenka tohoto algoritmu vychází ze skutečnosti, že nejefektivnější jsou výměny prvků v poli na velké vzdálenosti. Jestliže např. vezmeme pole s n prvky seřazenými v obráceném pořadí, můžeme je utřídit pomocí pouhých n/2 výměn. Porovnáme prvky na obou koncích pole a zaměníme je; pak postoupíme o jedno pole směrem ke středu a zopakujeme porovnání.[(1)]

Tento algoritmus je typickým příkladem D&C (District and Conquer) neboli rozděl a panuj algoritmu. Který funguje na principu rozdělování problému na menší a menší části až dojdeme na elementární, který má triviální (snadné, jednoznačné) řešení. D&C algoritmus je klasickým představitelem rekurzivního algoritmu.

Algoritmus vybere číslo, které určí za pivota. Všechny ostatní prvky zařadí buď nalevo od pivota, pokud jsou menší, nebo napravo pokud jsou větší. Následně vezme například levou menší část a celý postup se opakuje. Konec nastává pokud narazíme na jeden prvek, ten je totiž jednoznačně seřazený.

Volba pivota
Tím se dostáváme k nejsložitější části tohoto algoritmu a tím je volba pivota. Nejsnazší se zdá býti volba například posledního prvku pole, pokud ale algoritmu předáme pole již seřazené dostaneme složitost $O(n^2)$.

Nejefektivnější je volit takový prvek, který je medián právě řazené pod-posloupnosti. Tím získáme složitost O(n log n). Pro zjednodušení se často volí jako pivot medián z určitého výběru. Například medián z prvního, prostředního a posledního prvku.

For any deterministic method of pivot selectin, there exists a worst-case input instance which will domm us to quadratic time. „Quicksort runs in O(n log n) time, with high probability, if you give me randomly ordered data to sort“ [(2)]

Volně přeloženo. Pro každou deterministickou metodu volby pivota existuje vstup na kterém bude algoritmus řadit se složitostí $O(n^2)$.

Randomizace
Možnost jak se tomuto problému vyhnout je volit pivota náhodně nebo si každou posloupnost napřed náhodně rozházet. Potom docílíme vysokého zvýšení pravděpodobnosti řazení se složitostí O(n log n).

We can say: „Randomized quicksort runs in O(n log n) time on any input, with high probability.“ Alternately, we could get same guarantee by selecting a random element to be the pivot at each step. [(2)]

Pokud by byla volba náhodného pivota zajištěna pomocí nějakého hardwarového generátoru náhodných čísel (tedy skutečně náhodně), řadil by algoritmus vždy (tedy s takovou pravděpodobností, kterou lze považovat za vždy) se složitostí O(n log n). To je ale většinou velmi těžké, takže si musíme vystačit s pseudonáhodnými generátory. I ty nám ovšem zvýší pravděpodobnost, že quicksort bude řadit se složitostí O(n log n).

Ukázka algoritmu v pseudokódu [(4)]

```
procedure quicksort(List values)
    if values.size <= 1 then
        return values

    pivot = náhodný prvek z values

    Rozděl seznam values do 3 seznamů
        seznam1 = { prvky větší než pivot }
        seznam2 = { pivot }
        seznam3 = {  prvky menší než pivot }

    return quicksort(seznam1) + seznam2 +
quicksort(seznam3)
```

Závěr

Pokud tedy potřebujeme řadit co nejrychleji ale zároveň se chceme vyvarovat toho aby nám algoritmus běžel v jeho nejhorší složitosti je vhodnou volbou použít Randomizovaný Quicksort.

Bibliografie

1. EDELMANNOVÁ, Bc. Věra a Bc. Iva HENZLOVÁ. ČVUT. Projekt JERA: Základy algoritmizace [online]. Praha, 2003, 29-05-2003 [cit. 2012-11-23].
   Dostupné z: http://tjn.fjfi.cvut.cz/~virius/jera/binary/index.htm

2. SKIENA, Steven S. *The algorithm design manual*. 2nd ed. Santa Clara, CA: TELOS--the Electronic Library of Science, c1998, xvi, 486 p. ISBN 03-879-4860-0.
3. CORMEN, Thomas H. *Introduction to algorithms*. 3rd ed. Cambridge: MIT Press, c2009, xix, 1292 s. ISBN 978-0-262-03384-8.
4. MIČKA, Pavel. *Algoritmy.net: Příručka vývojáře* [online]. 2010 [cit. 2012-11-23]. Dostupné z: http://www.algoritmy.net/article/10/Quicksort