IV003, Fakulta Informatiky, Masarykova Univerzita

Homework 1

Jakub Senko, Štefan Uherčík

17. marca 2014

Príklad 1

Nech $X = [x_1, x_2, x_3, \ldots, x_n]$ je pole cisel dlzky $n, n > 0$ a plati ze $\forall x, y \in X : x \neq y$. Kazdemu $x_i \in X$ je priradene cislo w_i, pre ktore plati:

$$\begin{gathered} w_i > 0 \\ \sum_{i=0}^{n} w_i = 1 \end{gathered} \tag{0.1}$$

Optimalny prvok postupnosti je cislo x_k pre ktore plati:

$$\begin{gathered} \sum_{x_i < x_k} w_i < \frac{1}{2} \\ \sum_{x_i > x_k} w_i \leq \frac{1}{2} \end{gathered} \tag{0.2}$$

Problemom je navrh algoritmu ktory riesi najdenie optimalneho prvku s casovou zlozitostou $\Theta(n)$ a poskytnutie dokazu jeho korektnosti a zlozitosti.

Navrhovane riesenie je modifikovany algoritmus *Quick Select* ktory riesi problem najdenia medianu v poli cisel. Tento algoritmus ma obecne zlozitost $\mathcal{O}(n^2)$ pri nevhodnej volbe pivota, avsak pomocou procedury *Median of Medians* je mozne najst dostatocne dobry pivot na to, aby mal algoritmus vzdy linearnu zlozitost. *Quick Select* je popisany v nasledujucom texte iba neformalne, s odkazom na relevantne zdroje s dokazom zlozitosti. Zadana uloha je vyriesena ukazanim redukcie problemu najdenia optimalneho prvku na problem rieseny algoritmom *Quick Select + Median of Medians* [1] a dokazom ze tato procedura je vykonatelna v konstantom case. Vysledna zlozitost je teda $\mathcal{O}(n)$.

Quick Select
Vrati index n-teho najmensieho prvku pola, rekurzivne hlada v casti ohranicenej indexami left a right (vratane).

```
1: function QUICK_SELECT(list, left, right, n)
2:     pivot ← RANDOM_BETWEEN(left, right)
3:     pivot ← PARTITION(list, left, right, pivot)
4:     if n = pivot then
5:         return list[n]
6:     else if n < pivot then
7:         return QUICK_SELECT(list, left, pivot − 1, n)
8:     else
9:         return QUICK_SELECT(list, pivot + 1, right, n)
10:    end if
11: end function
```

Quick Select bezi v $\mathcal{O}(n^2)$.

Ako by bolo mozne deterministicky vybrat dobry pivot tak, aby bol vysledny algoritmus vzdy linearny.

Median Of Medians
Median of medians: http://www.youtube.com/watch?v=QAbv_4ndfo4&list=PLLH73N9cB21W1TZ6zz1dL

```
1: function MEDIAN_OF_MEDIANS(list, left, right)
2:     groups ← SPLIT_INTO_GROUPS_OF(list, left, right, 5)
3:     medians ← NEW_LIST
4:     for all group ∈ groups do
5:         median ← MEDIAN_OF_5(group)
6:         ADD(medians, median)
7:     end for
8:     size ← SIZE(medians)
9:     return QUICK_SELECT(medians, 0, size, n/2)
10: end function
```

Rozdelenie do skupin po 5 je O(n) ale najdenie medianu z konstantneho poctu prvkov je O(1). Nahradime funkciu *random_between* pouzitu na ziskanie indexu pivota algoritmom *median_of_medians*. Nahradenie sposobi problem pretoze MOM vracia hodnotu a QS ocakava index, ale QS mozeme upravit aby vracal index linearnym vyhladanim danej hodnoty v poli. Takto upraveny algoritmus bude obsahovat dve rekurzivne volania. Jedno vo funkcii MoM (riadok 9, zlozitost je n/5 pretoze sme ziskali n/5 medianov) a druhe priamo v QS (riadok 7 alebo 8). Druhe volanie ma zlozitost 7/10n zodpovedajuce hornemu odhadu poctu prvkov ktore su vacsie alebo naopak mensie ako median medianov (TODO ref. slidy). Ostatne operacie su konstantne alebo linearne. Nasledujuca rekurentna rovnica vyjadruje celkovu zlozitost QS (kde c je konstanta - pocet operacii s linearnou zlozitostou):

$$\begin{gathered} T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + c.n \\ T(n) = 10.c.n \in \Theta(n) \end{gathered} \tag{0.3}$$

Redukcia spociva v doplnujucej operacii ktora pre obidve particie zoznamu rozdeleneho podla pivota spocita sucet w_i. Tieto opracie su znovy vykonatelne v linearnom case. Vysledne sucty urcia na ktorej particii sa algoritmus rekurzivne zavola, pripadne ak plati podmienka pre optimalny prvok postupnosti, vypocet skonci. To znamena ze parameter mozeme odstranit parameter n ktory je po tejto uprave nepotrebny (kazda postupnost ma unikatny optimalny prvok). Vysledny upraveny algoritmus je nasledovny:

```
function QUICK_SELECT_MOM(list, left, right)
    pivotIndex ← MEDIAN_OF_MEDIANS_INDEX(list, left, right)
    pivotIndex ← PARTITION(list, left, right, pivotIndex)
    sumLeft ← SUM(list, left, pivotIndex − 1)
    sumRight ← SUM(list, pivotIndex + 1, right)
    if sumLeft ≥ 1/2 then
        return QUICK_SELECT_MOM(list, left, pivotIndex − 1)
    else if sumRight > 1/2 then
        return QUICK_SELECT_MOM(list, pivotIndex + 1, right)
    else
        return list[n]
    end if
end function
function QUICK_SELECT_MOM_START(list)
    return QUICK_SELECT_MOM(list, 1, lenght(list))
end function
```

Korektnost:
Vstupna podmienka $\phi(\langle X \rangle)$ je urcena vztahom (0.1) v zadani problemu.
Vystupna podmienka $\psi(\langle X \rangle, x_k)$ pozaduje, ze vystup algoritmu, x_k, je *optimalny prvok* podla vztahu (0.2). Parcialna korektnost
Vyuzijeme dokaz matematickou indukciou. Zjednodusime ze funkciu median of medians budeme brat ako korektny sposob najdenia medianu (slidy) aj napriek tomu ze obsahuje rekurzivne volanie do tejto procedury. Ak dokazeme korektnost v tomto zjednodusenom pripade, dokazeme to aj pre MOMI.
Zakladom indukcie bude nasledujuca myslienka: Pocas priebehu algoritmu (na zaciatku) je pole $X = [x_0, x_1, x_2, \ldots, x_n], n > 0$ rozdelene na tri sekcie (left $<$ right):
$X_{left} = [x_1, x_2, x_3, \ldots, x_{left-1}]$
$X_{mid} = [x_{left}, \ldots, x_{right}]$
$X_{right} = [x_{right+1}, \ldots, x_n]$
Tvrdim, ze pre kazde volanie funkcie QSMOM plati, ze:

$\forall x_i \in X_{left}, \forall x_j \in X_{mid}, x_i < x_j$ and $\sum w_i < \frac{1}{2}$

podobne
$\forall x_j \in X_{mid}, \forall x_k \in X_{right}, x_j < x_k$ and $\sum w_k \leq \frac{1}{2}$

Pokial dokazeme, ze velkost X_{mid} v kazdom rekurzivnom zavolani funkcie klesne (konvergenciu dokazeme potom), dostaneme sa postupne k casti ktora bude obsahovat prave jeden prvok. Pretoze pren budu platit vyssie uvedene podmienky, bude sa jednat o hladany *optimalny prvok postupnosti*. Indukciu budeme teda viest vzhladom k velkosti X_{mid}. Pred samotnym dokazom je potrebne poznamenat, ze predpokladame ze procedury median of median index, partition a um pokladame za totalne korektne. Kedze upraveny algoritmus vychadza z existujucich algoritmov popisanych v predchadzajucom texte ktore tiez pouzivaju dane procedury, ich dokaz je mozne vyhladat v existujucich zdrojoch.
Zaklad indukcie: Pri zavolani funkcie quick select mom start (prve volanie QSMoM) je rozdelenie na tri sekcie nasledovne: $X_{left} = []$
$X_{mid} = [x_1, x_2, \ldots, x_{n-1}, x_n]$
$X_{right} = []$
Trivialne tvrdenie plati.

Indukcny krok: Predpokladame ze
Konvergencia
Zlozitost:

PRÍKLAD 2

Riešenie tohto algoritmu môžeme zjednodušiť na hľadania najväčšieho prvku v poli

```
function SEARCH_ACE(matrix)
    breakingIndex ← matrix.size/2
    maximum ← 0
    maxX ← 0
    maxY ← 0
    for i = 1 to n do
        if matrix[i][breakingIndex] > maximum then
            maximum ← matrix[i][breakingIndex]
            maxX ← i
            maxY ← breakingIndex
        end if
    end for
    for i = 1 to n do
        if matrix[breakingIndex][i] > maximum then
            maximum ← matrix[breakingIndex][i]
            maxX ← breakingIndex
            maxY ← i
        end if
    end for
     IS_ESO(maximum, maxX, maxY)
    if isEso then return maximum
    elseMAKE_SUBMATRIX(matrix,startX,startY,endX,endY) SEARCH_ACE(submatrix)
    end if
end function


function IS_ESO(matrix,positionX,positionX)
    if matrix[positionX][positionX] > something then
    end if
end function


function FIND_HIGHEST_NEIGHBOR(matrix,positionX,positionX)
    if matrix[positionX][positionX] > something then
    end if
end function


function MAKE_SUBMATRIX(matrix,startX,startY,endX,endY)
end function
```

0.1 Example of list (3*itemize)

- First item in a list
  - First item in a list
    - First item in a list
    - Second item in a list
  - Second item in a list
- Second item in a list

0.2 Example of list (enumerate)

1. First item in a list
2. Second item in a list
3. Third item in a list

«««< Updated upstream

Príklad 3

PRÍKLAD 4

Tvrdenie 1: Ľubovoľná postupnosť n operácií INSERT a MIN-ALL má zložitosť O(n).

Uvažujme prirodzené čísla n,k a l, pre ktoré platí n=k+l (n vyjadruje počet operácií)

$$l = \begin{cases} \frac{n}{2} & \text{ak } n \text{ je párne} \\ \frac{n-1}{2} & \text{ak } n \text{ is nepárne} \end{cases}$$

$$l = \begin{cases} \frac{n}{2} & \text{ak } n \text{ je párne} \\ \frac{n-1}{2} + 1 & \text{ak } n \text{ is nepárne} \end{cases}$$

Uvažujme k oprácií INSERT, každá z týchto operácií vloží do zoznamu rovnaké prirodzené číslo z. Po poslednej z týchto operácií bude mať zoznam dĺžku k.

Cena týchto operácií dohromady je k.

Po týchto operáciách nasleduje l operácií MIN-ALL. Všetky čísla v zozname sú rovnaké, teda všetky čísla v ňom sú minimálne. Znamená to, že pri žiadnom z volaní operácie MIN-ALL sa dĺžka zoznamu nezmení.
Cena týchto operácií bude

$$l * k = \begin{cases} \frac{n}{2} * \frac{n}{2} = \frac{n^2}{4} & \text{ak } n \text{ je párne} \\ \frac{n-1}{2} * (\frac{n}{2} + 1) = \frac{n^2}{4} + \frac{n}{4} - \frac{1}{2} & \text{ak } n \text{ is nepárne} \end{cases}$$

Z predošlého tvrdenia vyplýva, že špecifikovaná postupnosť operácií bude minimimálne v zložitostnej triede O(n), teda tvrdenie **neplatí**.

Tvrdenie 2: Ľubovoľná postupnosť n operácií INSERT a MIN-ONE má zložitosť O(n).
Príklad riešime pomocou metódy účtov, kredity pre jednotliv0 operácie stanovíme nasledovne

Operácia	Cena	Kredit
INSERT	1	2
MIN-ONE	|S|	1

Platí, že vždy počas výpočtu je veľkosť zoznamu rovná počtu kreditov na účte, teda počet kreditov nikdy nebude menší ako 0. Celkový kredit po vykonaní n operácií bude menší alebo rovný 2n, teda tvrdenie **platí**.

Tvrdenie 3: Ľubovoľná postupnosť n operácií INSERT a DELETE má zložitosť O(n).

Uvažujme prirodzené čísla n,k a l, pre ktoré platí n=k+l (n vyjadruje počet operácií Hodnotu čísel k a l stanovíme rovnako, ako pri tvrdení 1.
Uvažujme k oprácií INSERT, každá z týchto operácií vloží do zoznamu rovnaké prirodzené číslo z. Po poslednej z týchto operácií bude mať zoznam dĺžku k.

Cena týchto operácií dohromady je k.

Po týchto operáciách nasleduje l operácií DELETE(y), pričom platí, že y≠z. To má za dôsledok, že po žiadnej z týchto operácií sa dĺžka zoznamu nezmení. Cena týchto operácií bude rovnaká, ako v tvrdení 1. Tvrdenie preto **neplatí**.
Tvrdenie 4: Ľubovoľná postupnosť n operácií INSERT a DELETE taká, že pri každom volaní sa operácia DELETE volá s iným parametrom i, má zložitosť má zložitosť O(n).

Uvažujme prirodzené čísla n,k a l, pre ktoré platí n=k+l (n vyjadruje počet operácií). Hodnotu čísel k a l stanovíme rovnako, ako pri tvrdení 1.
Uvažujme k oprácií INSERT, každá z týchto operácií vloží do zoznamu rovnaké prirodzené číslo z. Po poslednej z týchto operácií bude mať zoznam dĺžku k.

Cena týchto operácií dohromady je k.

Špecifikujeme množinu M o veľkosti l, v ktorej sa nachádzajú prirodzené čísla odližné od z. Vykonáme l operácií DELETE, pričom pri každej jej volaní predložíme ako parameter iný prvok z množiny M. To bude mať za následok, že veľkosť zoznamu sa nezmení. Cena týchto operácií bude rovnaká, ako v tvrdení 1. Tvrdenie preto **neplatí**.

Príklad 5

Literatúra

[1] BLUM, Manuel, Robert W. FLOYD, Vaughan PRATT, Ronald L. RIVEST a Robert E. TARJAN. Time bounds for selection. Journal of Computer and System Sciences. 1973, vol. 7, issue 4, s. 448-461. DOI: 10.1016/S0022-0000(73)80033-9. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0022000073800339

[2] http://moonflare.com/code/select/select.pdf