IV003, Fakulta Informatiky, Masarykova Univerzita

Homework 1

Jakub Senko, Štefan Uherčík

18. 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)
    if left = right then
        return list[left]
    end if
    pivotIndex ← MEDIAN_OF_MEDIANS_INDEX(list, left, right)
    pivotIndex ← PARTITION(list, left, right, pivotIndex)
    sumBottom ← SUM_W(list, 0, pivotIndex − 1)
    sumTop ← SUM_W(list, pivotIndex + 1, length(list))
    if sumBottom ≥ 1/2 then
        return QUICK_SELECT_MOM(list, left, pivotIndex − 1)
    else
        return QUICK_SELECT_MOM(list, pivotIndex + 1, right)
    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):
$X_{left} = [x_1, x_2, x_3, \ldots, x_{left-1}]$
$X_{mid} = [x_{left}, \ldots, x_{right}]$

$X_{right} = [x_{right+1}, \dots, x_n]$
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: $X_{left} = []$
$X_{mid} = [x_1, x_2, \dots, x_{n-1}, x_n]$
$X_{right} = []$
Trivialne tvrdenie plati.

Indukcny krok:
n Predpokladame ze pre k-te rekurzivne volanie tvrdenie plati na zaciatku funkcie. Analyzujme prebeh funkcie aby sme dokazali ze to bude platit aj pri nasledujucom rekurzivnom volani a zaroven $|X_{mid_k}| > |X_{mid_{k+1}}|$ V prvom kroku zvolime pivot a ziskame jeho index. Funkcia vrati hodnotu pre ktoru plati: $left \leq pivotIndex \leq right$. Prpominam ze implementacia tejto funkcie nie je podstatna pre korektnost algortmu ale len pre jeho efektivitu. [todo PICK] V druhom kroku sa vola funkcia partition, ktora rozdeli X_{mid} na dve particie:
$X_{lower} = [x_{left}, \dots, x_{pivotIndex-1}]$
$X_{upper} = [x_{pivotIndex+1}, \dots, x_{right}]$
Pre ktore plati: $\forall x \in X_{lower}, x < x_{pivotIndex}$ a $\forall x \in X_{lower}, x \geq x_{pivotIndex}$ Tato funkcia je prebrana z existujucich algoritmov Quick Select, Quick Sort apod. a teda nie je potrebne dokazovat jej korektnost. Dalsie dva kroky spocitaju sumu vah vsetkych prvkov v zozname nalavo a potom napravo od pivota, teda:

V tomto momente su uz k dispozicii vsetky informacie a nasleduje podmienene vetvenie ktore v dvoch pripadoch rekurzivne zavola danu funkciu a v jednom vrati konkretny vysledok. Budem sa teraz venovat prvym dvom moznostiam a dokazem ze v tychto pripadoch plati indukcna hypoteza a poslednej moznosti sa budem venovat v dalsej sekcii.
Ak plati ze $sumBottom \geq \frac{1}{2}$ potom bude rekurzivne nove rozdelenie nasledujuce:
$X_{left_{k+1}} = X_{left_k}$
$X_{mid_{k+1}} = X_{lower_k}$

$X_{right_{k+1}} = [x_{pivotIndex}] \sqcup X_{upper_k} \sqcup X_{right_k}$
V tomto pripade plati ze vsetky prvky v left su mensie nez vsetky z mid trivialne, pretoze $X_{left_{k+1}} = X_{left_k}$ a $X_{mid_{k+1}} \sqsubset X_{mid_k}$ a teda to vyplyva priamo z ind. hypotezy. To iste plati pre sumu $X_{left_{k+1}}$. $X_{right_{k+1}}$ sa sklada z pivota a X_{upper_k} ktore su vacsie ako $X_{mid_{k+1}}$ co priamo vyplyva zo specifikacie pre partition funkciu. Pre X_{right_k} to vyplyva z indukcnej hypotezy. To ze je sucet mensi ako jedna polovica vyplyva zo zadania ktore pozaduje aby bol sucet vah prave jedna. Kedze z podmienky ktora sposobila vetvenie plati ze $sumBottom \geq \frac{1}{2} > 1/2$ a right tvori zvysok zoznamu, potom jeho suma musi byt mensia ako $1/2$. Indukcia plati.
V druhom pripade je nove rozdelenie nasledujuce:
$X_{left_{k+1}} = X_{left_k} \sqcup X_{lower_k} \sqcup [x_{pivotIndex}]$
$X_{mid_{k+1}} = X_{upper_k}$
$X_{right_{k+1}} =$
Plati ze $sumBottom < \frac{1}{2}$, ale kedze X_{right} sa nemeni, jej sucet je podla indukcnej hypotezy $\leq \frac{1}{2}$. Analogicky ako pri predchadzajucej moznosti, indukcny krok a teda cela indukcia plati.
Konvergencia a parcialna korektnost:

Dokaz konvergenie spociva v tom, ze cast X_{mid} sa v kazdom kroku zmensi minimalne o prvok, ktory bol v predchadzajucom kroku pivotom. Tymto sposobom sa v konecnom pocte krokov velkost X_{mid} zredukuje na jediny prvok. V tomto pripade sa vypocet dostane do prvej podmienej vetvy, a kedze pre tento prvok musi platit indukcny krok a teda aj vystupna podmienka, algoritmus v konecnom case vrati spravny vysledok. $\square$

Príklad 2

Riešenie tohto algoritmu môžeme zjednodušiť na hľadania najväčšieho prvku v poli

function SEARCH_ACE($matrix$, $startX$, $startY$, $size$)
    $midX \leftarrow startX + \lfloor \frac{size}{2} \rfloor$
    $midY \leftarrow startY + \lfloor \frac{size}{2} \rfloor$
    $maxX \leftarrow 1$
    $maxY \leftarrow 1$
    for $i = startX$ to $startX + size - 1$ **do**
        if $matrix[i][midY] > matrix[maxX][maxY]$ **then**
            $maxX \leftarrow i$
            $maxY \leftarrow midY$
        end if
    end for
    for $i = startY$ to $startY + size - 1$ **do**
        if $matrix[miX][i] > matrix[maxX][maxY]$ **then**
            $maxX \leftarrow midX$
            $maxY \leftarrow i$
        end if
    end for
    if IS_ACE($maxX$, $maxY$) **then**
        return $matrix[maxX][maxY]$
    end if
    if SELECT_QUADRANT($maxX$, $maxY$) $= 1$ **then**
        SEARCH_ACE($matrix$, $startX$, $startY$, $\lfloor \frac{size}{2} \rfloor$)
    else if SELECT_QUADRANT($maxX$, $maxY$) $= 2$ **then**
        SEARCH_ACE($matrix$, $midX$, $startY$, $\lceil \frac{size}{2} \rceil$)
    else if SELECT_QUADRANT($maxX$, $maxY$) $= 3$ **then**
        SEARCH_ACE($matrix$, $startX$, $midY$, $\lceil \frac{size}{2} \rceil$)
    else if SELECT_QUADRANT($maxX$, $maxY$) $= 4$ **then**
        SEARCH_ACE($matrix$, $midX$, $midY$, $\lceil \frac{size}{2} \rceil$)
    end if
end function

function IS_ESO(x,y)
    ▷ Implementacia vynechana, popis v texte
end function

function SELECT_QUADRANT(x,y)
    ▷ Implementacia vynechana, popis v texte
end function

Hlavnou funkciou tohto algoritmu je searchAce. V tejto funkcii najprv vyberieme stredný stĺpec matice a a jej stredný riadok. Tento stĺpec a riadok spolu vytvárajú kríž, ktorý rozdeľuje maticu na rovnaké, resp. skoro rovnaké kvadranty (pri sudom n).
V tomto ḱrížiňájdeme maximálnu hodnotu - zložitosť tejto časti algoritmu je 2n
Pred dôkazom korektnosti doplníme korektnosť funkcií, ktoré sme priamo nedefinovali:
IS_ACE Táto funkcia má za úlohu zistiť, či je prvok na zadaných súradniciach v matici eso. Na toto stačia maximálne 4 porovnania so susednými prvkami (ak tieto prvky existujú). Výsledná zložitosť patrí do $\mathcal{O}(1)$.
SELECT_QUADRANT Táto funkcia vráti číslo z množiny {1,2,3,4}, ktoré reprezentuje 1 zo štyroch možných kvadrantov matice, na ktorých sa funkcia SEARCH_ACE rekurzívne zavolá. Vstupné parametre sú súradnice maximálneho prvku na *kríži*. Funkcia vyhodnotí v ktorom kvadrante sa nachádza najväčší zo štyroch susediacich prvkov (nazveme ho *smerodajný prvok*. Tento najväčší prvok sa musí nachádzať mimo strednu kríža, v opačnom prípade, funkcia IS_ACE vráti true a k tejto funkcii sa vykonávanie nedostane. Túto funkcionalitu je možné implementovať pomocou konštantného počtu porovnaní, čo znamená že výsledná zložitosť patrí do $\mathcal{O}(1)$.
Korektnosť
Parciálna korektnosť. Vstupná aj výstupná podmienka sú devinované v zadaní TODO. Dôkaz korektnosti algoritmu vykonáme matematickou indukciou vzhľadom na počet rekurzívnych volaní. Ak argument funkcie SEARCH_ACE je štvorcová podmatica pôvodnej matice obsahujúca eso , potom kvadrant(podmatica), ktorý algoritmus vyberie a rekurzívne zavolá, tiež obsahuje eso.
bázový krok: funkciu SEARCH_ACE zavoláme na zadanej matici M nasledovným spôsobom SEARCH_ACE(M,1,1,$|M|$), M triviálne obsahuje eso. indukčný krok: Vnútri kvadrantu existuje prvok, ktorý je väčší ako všetky prvky na jeho hranici, konkrétne prvok, na základe ktorého bol kvadrant vybraný funkciou SELECT_QUADRANT. V závislosti na vybranom kvadrante môže byť časť tejto hranice súčasťou kvadrantu, tvrdenie ale platí stále pre zvyšné prvky.
Lemma 1: Vzhľadom k tomu, že matica je zaplnená rôznymi prirodzenými číslami, teda obsahuje aj maximálny prvok a tento prvok je s určitosťou eso.
Keby sa maximálny prvok mohol nachádzať na hranici v rámci kvadrantu mohli by sme ho označiť za eso, ale v skutočnosti by mohol susediť s prvkom mimo kvadrantu, ktorý by bol väčší a teda by sa o eso nejednalo. Maximálny prvok v kvadrante je väčší ako ktorýkoľvek prvok na hranici kvadrantu, pretože maximálny prvok musí byť väčší alebo rovný ako *smerodajný prvok*. To znamená, že kvadrant, nad ktorým zavoláme rekurzívne funkciu SEARCH_ACE obsahuje eso, čo znamená, že indukčný krok platí.
Konvergentnosť. Algoritmus skončí v konečnom počte rekurzívnych volaní, pretože veľkosť podmatice (size) sa v každom volaní zmenší $\lceil \frac{size}{2} \rceil$. V prípade, že size dosiahne hodnotu 1, algoritmus skončí, pretože metóda IS_ACE vráti určite true.
Zložitosť Na výpočet zložitosti využijeme Master Theorem. Zložitosť použitého algoritmu je zadaná ako počet operácii porovnania. Tieto závisia na veľkosti strany vstupnej matice. V každom volaní funkcie sa rekurzívne tento parameter zmenší na polovicu. Naviac určenie kvadrantu do ktorého sa algoritmus rekurzívne zavolá vyžaduje nájdenie maximálneho prvku z dvoch polí dĺžky n. A nakonie sa vykoná jedna s funkcii IS_ACE alebo

SELECT_QUADRANT, ktoré majú konštantnú zložitosť c. Zložistosť algoritmu je teda:

$$\begin{aligned} T(n) &= T\left(\frac{n}{2}\right) + 2.n + c \\ T(n) &\in \mathcal{O}(n) \end{aligned} \tag{0.4}$$

0.1 Bla Bla

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)

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

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

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