

Homework 1

Jakub Senko, Štefan Uherčík

18. marca 2014

PRÍKLAD 1

Nech $X = [x_1, x_2, x_3, \dots, x_n]$ je pole čísel dĺžky $n, n > 0$ a platí že $\forall x, y \in X : x \neq y$. Každému $x_i \in X$ je priradené číslo w_i , pre ktoré platí:

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

Optimálny prvok postupnosti je číslo x_k pre ktoré platí:

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

Problémom je návrh algoritmu ktorý rieši nájdenie optimálneho prvku s časovou zložitou $\Theta(n)$ a poskytnutie dokazu jeho korektnosti a zložitosti.

Navrhovane riesenie je modifikovany algoritmus *Quick Select* ktorý rieši problém nájdenia medianu v poli čísel. Tento algoritmus má obecnú zložitost $\mathcal{O}(n^2)$ pri nevhodnej voľbe pivotu, avšak pomocou procedury *Median of Medians* je možné najst dostatočne dobrý pivot na to, aby mal algoritmus vždy lineárnu zložitost. *Quick Select* je popísaný v nasledujúcom texte iba neformálne, s odkazom na relevantné zdroje s dokazom zložitosti. Zadaná úloha je vyriešená ukázaním redukcie problému nájdenia optimálneho prvku na

problem riešený algoritmom *Quick Select + Median of Medians* [?] a dokazom, že táto procedura je vykonateľná v konštantnom čase. Výsledná zložitosť je teda $\mathcal{O}(n)$.

Quick Select

Vráti index n -teho najmenšieho prvku pola, rekurzívne hľadá v časti ohranicenej indexami *left* a *right* (vrátane).

```
1: function QUICK_SELECT(list, left, right, n)
2:   pivot  $\leftarrow$  RANDOM_BETWEEN(left, right)
3:   pivot  $\leftarrow$  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 beží v $\mathcal{O}(n^2)$.

Ako by bolo možné deterministicky vybrať dobrý *pivot* tak, aby bol výsledný algoritmus vždy lineárny.

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  $\leftarrow$  SPLIT_INTO_GROUPS_OF(list, left, right, 5)
3:   medians  $\leftarrow$  NEW_LIST
4:   for all group  $\in$  groups do
5:     median  $\leftarrow$  MEDIAN_OF_5(group)
6:     ADD(medians, median)
7:   end for
8:   size  $\leftarrow$  SIZE(medians)
9:   return QUICK_SELECT(medians, 0, size,  $\frac{n}{2}$ )
10: end function
```

Rozdelenie do skupín po 5 je $\mathcal{O}(n)$ ale nájdenie medianu z konštantného počtu prvkov je $\mathcal{O}(1)$. Nahradíme funkciu *random_between* použitú na získanie indexu *pivota* algoritmom *median_of_medians*. Nahradenie spôsobí problém pretože MOM vracia hodnotu a QS očakáva index, ale QS môžeme upraviť aby vracal index lineárnym vyhľadávaním danej hodnoty v poli. Takto upravený algoritmus bude obsahovať dve rekurzívne volania. Jedno vo funkcii MoM (riadok 9, zložitosť je $n/5$ pretože sme získali $n/5$ medianov) a druhé priamo v QS (riadok 7 alebo 8). Druhé volanie má zložitosť $7/10n$ zodpovedajúce hornému odhadu počtu prvkov, ktoré sú väčšie alebo naopak menšie ako median me-

dianov (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):

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + c.n \quad (0.3)$$

$$T(n) = 10.c.n \in \Theta(n)$$

Redukcia spociva v doplnujucej operacii ktora pre obidve particie zoznamu rozdeleného podľa pivota spocita sucet w_i . Tieto opracie su znovy vykonatelne v linearnom case. Vysledne sučty 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 nepotrebný (kazda postupnost ma unikatny optimalny prvok). Vysledny upraveny algoritmus je nasledovny:

```

1: function QUICK_SELECT_MOM(list, left, right)
2:   if left = right then
3:     return list[left]
4:   end if
5:   pivotIndex ← MEDIAN_OF_MEDIANS_INDEX(list, left, right)
6:   pivotIndex ← PARTITION(list, left, right, pivotIndex)
7:   sumBottom ← SUM_W(list, 0, pivotIndex - 1)
8:   sumTop ← SUM_W(list, pivotIndex + 1, length(list))
9:   if sumBottom ≥  $\frac{1}{2}$  then
10:    return QUICK_SELECT_MOM(list, left, pivotIndex - 1)
11:  else
12:    return QUICK_SELECT_MOM(list, pivotIndex + 1, right)
13:  end if
14: end function
15:
16: function QUICK_SELECT_MOM_START(list)
17:   return QUICK_SELECT_MOM(list, 1, length(list))
18: end function

```

Korektnost:

Vstupna podmienka $\phi(\langle X \rangle)$ je urcena vzťahom (0.1) v zadani problemu.

Vystupna podmienka $\psi(\langle X \rangle, x_k)$ pozaduje, ze vystup algoritmu, x_k , je *optimalny prvok* podľa vzťahu (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 $X = [x_0, x_1, x_2, \dots, x_n]$, $n > 0$ rozdelené na tri sekcie ($\text{left} < \text{right}$):

$X_{\text{left}} = [x_1, x_2, x_3, \dots, x_{\text{left}-1}]$

$X_{\text{mid}} = [x_{\text{left}}, \dots, x_{\text{right}}]$

$$X_{right} = [x_{right+1}, \dots, x_n]$$

Tvrdim, že pre každé volanie funkcie QSMOM platí, že:

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

podobne

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

Pokiaľ dokážeme, že veľkosť X_{mid} v každom rekurzívnom zavolaní funkcie klesne (konvergenciu dokážeme potom), dostaneme sa postupne k časti ktorá bude obsahovať práve jeden prvok. Pretože pre nás budú platiť vyššie uvedené podmienky, bude sa jednáť o hľadaný *optimalný prvok postupnosti*. Indukciu budeme teda viesť vzhľadom k veľkosti X_{mid} .

Pred samotným dôkazom je potrebné poznamenať, že predpokladáme, že procedúry median of median index, partition a um pokladáme za úplne korektné. Keďže upravený algoritmus vychádza z existujúcich algoritmov popísaných v predchádzajúcom texte ktoré tiež používajú dane procedúry, ich dôkaz je možné vyhľadať v existujúcich zdrojoch.

Základ indukcie: Pri zavolaní funkcie quick select môžeme začať (prvé volanie QSMOM) je rozdelenie na tri sekcie nasledovne: $X_{left} = []$

$$X_{mid} = [x_1, x_2, \dots, x_{n-1}, x_n]$$

$$X_{right} = []$$

Triválne tvrdenie platí.

Indukčný krok:

n Predpokladáme, že pre k -te rekurzívne volanie tvrdenie platí na začiatku funkcie. Analyzujeme prebeh funkcie aby sme dokázali, že to bude platiť aj pri nasledujúcom rekurzívnom volaní a zároveň $|X_{mid_k}| > |X_{mid_{k+1}}|$. V prvom kroku zvolíme pivot a získame jeho index. Funkcia vráti hodnotu, pre ktorú platí: $left \leq pivotIndex \leq right$. Prípadom, že implementácia tejto funkcie nie je podstatná pre korektnosť algoritmu ale len pre jeho efektívnosť. [todo PICK] V druhom kroku sa volá funkcia partition, ktorá rozdelí X_{mid} na dve partie:

$$X_{lower} = [x_{left}, \dots, x_{pivotIndex-1}]$$

$$X_{upper} = [x_{pivotIndex+1}, \dots, x_{right}]$$

Pre ktoré platí: $\forall x \in X_{lower}, x < x_{pivotIndex}$ a $\forall x \in X_{upper}, x \geq x_{pivotIndex}$. Táto funkcia je prebrána z existujúcich algoritmov Quick Select, Quick Sort apod. a teda nie je potrebné dokazovať jej korektnosť. Ďalšie dva kroky spočítajú sumu vah všetkých prvkov v zozname naľavo a potom napravo od pivotu, teda:

V tomto momente sú už k dispozícii všetky informácie a nasleduje podmienené vetvenie, ktoré v dvoch prípadoch rekurzívne zavola danú funkciu a v jednom vráti konkrétny výsledok. Budem sa teraz venovať prvým dvom možnostiam a dokážem, že v týchto prípadoch platí indukčná hypotéza a poslednej možnosti sa budem venovať v ďalšej sekcii. Ak platí, že $sumBottom \geq \frac{1}{2}$ potom bude rekurzívne nové rozdelenie nasledujúce:

$$X_{left_{k+1}} = X_{left_k}$$

$$X_{mid_{k+1}} = X_{lower_k}$$

$$X_{right_{k+1}} = [x_{pivotIndex}] \sqcup X_{upper_k} \sqcup X_{right_k}$$

V tomto prípade platí že všetky prvky v left sú menšie než všetky z mid triválne, pretože $X_{left_{k+1}} = X_{left_k}$ a $X_{mid_{k+1}} \sqsubset X_{mid_k}$ a teda to vyplýva priamo z ind. hypotézy. To isté platí pre sumu $X_{left_{k+1}} \cdot X_{right_{k+1}}$ sa skladá z pivota a X_{upper_k} ktoré sú väčšie ako $X_{mid_{k+1}}$ čo priamo vyplýva zo špecifikácie pre partition funkciu. Pre X_{right_k} to vyplýva z indukčnej hypotézy. To že je súčet menší ako jedna polovica vyplýva zo zadania ktoré požaduje aby bol súčet vah práve jedna. Keďže z podmienky ktorá spôsobila vetvenie platí že $sumBottom \geq \frac{1}{2} > 1/2$ a right tvorí zvyšok zoznamu, potom jeho suma musí byť menšia ako $1/2$. Indukcia platí.

V druhom prípade je nové rozdelenie nasledujúce:

$$X_{left_{k+1}} = X_{left_k} \sqcup X_{lower_k} \sqcup [x_{pivotIndex}]$$

$$X_{mid_{k+1}} = X_{upper_k}$$

$$X_{right_{k+1}} =$$

Platí že $sumBottom < \frac{1}{2}$, ale keďže X_{right} sa nemení, jej súčet je podľa indukčnej hypotézy $\leq \frac{1}{2}$. Analogicky ako pri predchádzajúcej možnosti, indukčný krok a teda celá indukcia platí.

Konvergenca a parciálna korektnosť:

Dokaz konvergenie spočíva v tom, že časť X_{mid} sa v každom kroku zmenší minimálne o prvok, ktorý bol v predchádzajúcom kroku pivotom. Týmto spôsobom sa v konečnom počte krokov veľkosť X_{mid} zredukuje na jediný prvok. V tomto prípade sa výpočet dostane do prvej podmienenej vetvy, a keďže pre tento prvok musí platiť indukčný krok a teda aj výstupná podmienka, algoritmus v konečnom case vráti správny výsledok. \square

PRÍKLAD 2

Riešenie tohto algoritmu môžeme zjednodušiť na hľadania najväčšieho prvku v poli

```

1: function SEARCH_ACE(matrix, startX, startY, size)
2:   midX  $\leftarrow$  startX +  $\lfloor \frac{size}{2} \rfloor$ 
3:   midY  $\leftarrow$  startY +  $\lfloor \frac{size}{2} \rfloor$ 
4:   maxX  $\leftarrow$  1
5:   maxY  $\leftarrow$  1
6:   for i = startX to startX + size - 1 do
7:     if matrix[i][midY] > matrix[maxX][maxY] then
8:       maxX  $\leftarrow$  i
9:       maxY  $\leftarrow$  midY
10:    end if
11:  end for
12:  for i = startY to startY + size - 1 do
13:    if matrix[midX][i] > matrix[maxX][maxY] then
14:      maxX  $\leftarrow$  midX
15:      maxY  $\leftarrow$  i
16:    end if
17:  end for
18:  if IS_ACE(maxX, maxY) then
19:    return matrix[maxX][maxY]
20:  end if
21:  if SELECT_QUADRANT(maxX, maxY) = 1 then
22:    SEARCH_ACE(matrix, startX, startY,  $\lfloor \frac{size}{2} \rfloor$ )
23:  else if SELECT_QUADRANT(maxX, maxY) = 2 then
24:    SEARCH_ACE(matrix, midX, startY,  $\lfloor \frac{size}{2} \rfloor$ )
25:  else if SELECT_QUADRANT(maxX, maxY) = 3 then
26:    SEARCH_ACE(matrix, startX, midY,  $\lfloor \frac{size}{2} \rfloor$ )
27:  else if SELECT_QUADRANT(maxX, maxY) = 4 then
28:    SEARCH_ACE(matrix, midX, midY,  $\lfloor \frac{size}{2} \rfloor$ )
29:  end if
30: end function
31:
32:
33: function IS_ESO(x,y)
34:                                     ▷ Implementacia vynechana, popis v texte
35: end function
36:
37:
38: function SELECT_QUADRANT(x,y)
39:                                     ▷ Implementacia vynechana, popis v texte
40: 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 kríži nájdeme 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 strednú kríž, v opačnom prípade, funkcia `IS_ACE` vráti `true` a k tejto funkcii sa vykonávanie nedostane. Túto funkcionality 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 určitou `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 podmaticy (`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`.

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

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 zložitosťnej 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 jednotlivé 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 $y \neq z$. 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 zložitost' $O(n)$.

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 odlišné 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>