Tema - Structuri de Date

Alexandra Nanu

April 2020

1 Problema 1

În următoarea demonstrație, vom ține cont de faptul că a sorta n elemente prin comparatie, înseamnă $\Omega(n \log n)$, atunci si a construi un BST inseamna tot $\Omega(n \log n)$.

Presupunem prin absurd ca a construi un BST in cel mai rau caz inseamna $o(n \log n)$. Atunci, ca sa sortam, doar am construi BST-ul si apoi am parcurge elementele in inordine. Acest pas este realizat in $\Theta(n)$, conform Teoremei 12.1 - Cormen. In plus, o traversare in inordine este realizata intr-o ordine sortata, intrucat elementele din subarborele stang sunt mai mici decat elementul curent si sunt printate inaintea elementului curent, iar elementele din subarborele drept sunt elemente mai mari decat elementul curent si sunt printate dupa acesta.

Acest fapt ne-ar permite sa sortam in $o(n \log n)$. - contradictie

O alta demonstratie este urmatoarea:
Cel mai bun caz este acela cand cheile sunt date in asa fel incat sa completeze, succesiv, toate nivelurile din arbore, i.e. sirul cheilor este $(k_i)_{1 \leq i \leq n}$ cu proprietatea ca pe primul nivel se afla k_1, pe al doilea k_2 si k_3 etc. (cu toate permutarile posibile pe un anumit nivel al arborelui, adica nu conteaza ordinea dintre k_2 si k_3, ci conteaza ca aceste doua chei sunt pe acelasi nivel).

Atunci, pentru a adauga o cheie pe nivelul j, sunt necesare $j-1$ comparatii (daca primul nivel este 1), respectiv j comparatii (daca primul nivel este 0).

Fiind arbore binar, un nod k_i se va afla pe nivelul $j = \log_2 k_i$. Atunci, pentru fiecare nod k_i vor fi necesare $\log_2 i \approx \log i$ comparatii, fiind cel mai bun timp ce poate fi obtinut. In acest caz, pentru adaugarea tuturor nodurilor (constructia arborelui binar de cautare), avem timpul:

$$\begin{aligned}T(n) &= T(k_1) + T(k_2) + ... + T(k_n)\\ &= \log 1 + \log 2 + ... + \log n\\ &= \log\left(\prod_{i=1}^{n} i\right)\\ &= \log(n!)\end{aligned}$$

Din inegalitatea lui Stirling, avem ca

$$n \cdot \log n - n < \log(n!) < (n+1) \cdot \log(n+1) - n \tag{1}$$

si, din definitia lui Ω

$$f(n) \in \Omega(g(n)) \Longleftrightarrow \exists c > 0, n_0 \geq 0 : \forall n > n_0 : 0 \leq cg(n) \leq f(n) \tag{2}$$

Din relatiile (1) si (2) avem ca

$$T(n) = \log(n!) > n \cdot \log n - n = g(n)$$

si exista $c \in (0,1)$ astfel incat $T(n) \geq cg(n)$, de unde rezulta ca $T(n) \in \Omega(n \cdot \log n)$.

2 Problema 2

Din definitia lui Θ:

$$f(n) \in \Theta(g(n)) \Longleftrightarrow \exists c_1, c_2 > 0, n_{01} \geq 0 : \forall n \geq n_{01} : 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \tag{1}$$

Aplicam si pentru g(n):

$$g(n) \in \Theta(h(n)) \Longleftrightarrow \exists c_3, c_4 > 0, n_{02} \geq 0 : \forall n \geq n_{02} : 0 \leq c_3 \cdot h(n) \leq g(n) \leq c_4 \cdot h(n) \tag{2}$$

Inmultim (2) cu c1:

$$0 \leq c_1 \cdot c_3 \cdot h(n) \leq c_1 \cdot g(n) \leq f(n) \qquad (3)$$

Inmultim (2) cu c2:

$$c_2 \cdot c_4 \cdot h(n) \geq c_2 \cdot g(n) \geq f(n)(4)$$

Tinand cont de (3) si (4), obtinem:

$$0 \leq c_1 \cdot c_3 \cdot h(n) \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \leq c_2 \cdot c_4 \cdot h(n)$$

Astfel,

$$0 \leq c_1 \cdot c_3 \cdot h(n) \leq f(n) \leq c_2 \cdot c_4 \cdot h(n)$$

Notand $c_1 \cdot c_3$ cu a_1 si $c_2 \cdot c_4$ cu a_2, obtinem:

$$0 \leq a_1 \cdot h(n) \leq f(n) \leq a_2 \cdot h(n)$$

Am considerat $n_0 \in max(n_{01}, n_{02})$.

3 Problema 3

Din definitia lui o:

$$f(n) \in o(g(n)) \iff \forall c > 0 \exists n_0 > 0 : \forall n > n_0 : 0 \leq f(n) < cg(n)$$

Adaptat la cazul nostru:

$$\log n \in o(\sqrt{n}) \iff \forall c > 0 \exists n_0 > 0 : \forall n > n_0 : 0 \leq \log n < c\sqrt{n} \tag{1}$$

Luam $c \geq 1$ si $n_0 = 1$, atunci $\forall n > n_0$ avem ca $\log n < \sqrt{n} = c\sqrt{n}$. Atunci $\log n \in o(\sqrt{n})$.

Pentru $c < 1$, aleg $n_0 = (1/c + 100)^{10}$.

Din (1) ne uitam la inegalitate in n_0:

$\log((1/c + 100)^{10}) < c\sqrt{(1/c + 100)^{10}} \iff 10\log(1/c + 100) < c(1/c + 100)^5$

Dar, $c(1/c + 100) > 1$ si $10\log(1/c + 100) < (1/c + 100)^4$ pentru ca $10 < 1/c + 100$ si $\log(1/c + 100) < 1/c + 100 =>$
$=>$ Inegalitatea este adevarata

Deci, pentru a arata ca $\log n \in o(\sqrt{n})$ putem alege n_0:

$$n_0 = \begin{cases} 1, c \geq 1 \\ (1/c + 100)^{10}, c < 1 \end{cases}$$

4 Problema 4

Vom calcula suma numerelor de la 1 la n ca fiind $s = n(n + 1)/2$, ceea ce presupune o complexitate $\Theta(1)$ Realizam suma S, a tuturor numerelor din sirul dat (parcurgerea unui sir cu n numere se realizeaza in $\Theta(n)$, iar apoi scadem din suma S a numerelor din sir, suma s a numerelor de la 1 la n, astfel obtinand elementul duplicat.

Fiind vorba despre o parcurgere a $n+1$ elemente, complexitatea algoritmului va fi $\Theta(n)$.

Pseudocod:

```
(1)     citeste n
(2)     s = n*(n+1)/2
(3)     S = 0
(4)     bec = 1
(5)     pentru i = 1, n + 1 executa
(6)         citeste x
(7)         //verific ca x sa nu fie mai mare decat n sau mai mic decat
decat 0
(8)         daca x>n sau x<1 atunci
(9)             i = n + 2
(10)             bec = 0
(11)         altfel
(12)             S = S + x
(13)     daca bec = 1 atunci
(14)         scrie S - s
```

Daca asociem fiecarei linii j cate un cost c_j, atunci observam ca

$$T(n) = c_1 + c_2 + c_3 + c_4 + c_5(n+2) + c_6(n+1) + c_8(n+1) + c_9(n+1) + c_{10}(n+1) + c_{11}(n+1)$$
$$+c_{13} + c_{14}$$

$$= (c_5 + c_6 + c_8 + c_9 + c_{10} + c_{11}) + \sum_{j=1}^{14} c_i$$

$$= an + b \in \Theta(n)$$

5 Problema 5

Voi folosi un algoritm ce se bazeaza pe metoda Divide et Impera astfel:

1) calculam medianele m_1 si m_2 pentru cei doi vectori

2) daca $m_1 = m_2$, atunci am gasit mediana celor doi vectori

3) daca $m_1 > m_2$, atunci mediana este prezenta intr-una dintre urmatoarele secvente:

secventa de la primul element al vectorului 1 pana la m_1 ($v_1[1]...v_1[\lfloor n/2 \rfloor]$)

secventa de la m_2 pana la ultimul element din vectorul 2 ($v_2[\lfloor n/2 \rfloor]...v_2[n]$)

4) daca $m_1 < m_2$, atunci mediana este prezenta intr-una dintre urmatoarele secvente:

secventa de la primul element al vectorului 2 pana la m_2 ($v_2[1]...v_2[\lfloor n/2 \rfloor]$)

secventa de la m_1 pana la ultimul element din vectorul 1 ($v_1[\lfloor n/2 \rfloor]...v_1[n]$)

5) repetam procesul descris pana cand lungimea vectorilor ajunge sa fie mai mica sau egala decat 2

6) interclasez cei doi vectori, iar output-ul va fi mediana rezultata

Pseudocod:

Mediana(v_1, v_2)
    daca $min(v_1.length, v_2.length)$ <= 2 atunci
        interclasez v_1, v_2
        returnez mediana vectorului interclasat
    altfel
        m_1 = mediana v_1
        m_2 = mediana v_2
        daca $m_1 = m_2$ atunci
            returnez m_1
        daca $m_1 < m_2$ atunci
            pastrez a doua jumatate din v_1
            pastrez prima jumatate din v_2
        altfel
            pastrez a doua jumatate din v_2
            pastrez prima jumatate din v_1

Asemanator cu o cautare binara, algoritmul va avea o complexitate de $\Theta(logn)$.
Consider recurenta:

$$T(n) = T(n/2) + 1$$

Aplicam Teorema Master pentru a demonstra acest fapt:
Fie $a = 1$, $b = 2$ si $f(n) = 1$ din recurenta de mai sus

$$n^{\log_b a} = n^{\log_2 1} = n^0 = 1$$

Asadar, deoarece f(n) = $\theta(n^{\log_b a}) = \theta(1)$, ne aflam in cel de-al doilea caz al Teoremei Master:

$T(n) \in \Theta(n^{\log_b a} \log n)$

Astfel, solutia recurentei este T(n) $\in \Theta(\log n)$

6 Problema 6.1

Pentru a prezenta ca algoritmul descris nu este optim, este de ajuns sa gasim un contraexemplu.
Fie n = 11, capacitatea Ferrari-ului, k = 10 bunuri, iar $x_1 = 1$, $x_2 = 2$, $x_3 = 3$, .., $x_{10} = 10$.
Algoritmul de tip Greedy prezentat in cerinta lucreaza in felul urmator:
1) va grupa obiectele x_1, x_2, x_3, x_4 care impreuna ocupa o capacitate de 10 din $n = 11$ posibil. Astfel, x_5 care ocupa un spatiu de 5 nu poate fi grupat cu aceasta prima diviziune, intrucat s-ar intrece maximul posibil de 11. Masina pleaca astfel cu primele 4 bunuri.

2) obiectele x_5 si x_6 ocupa impreuna un spatiu de 11, maximul posibil. Masina parcurge astfel a doua deplasare cu obiectele x_5 si x_6.
3) obiectul x_7 va fi transportat singur; la fel si obiectele x_8, x_9 si x_{10}, intrucat niciunul dintre aceste obiecte nu poate fi grupat cu succesorul sau.
4) Astfel, vor fi realizate 6 transporturi.

Contraexemplu:
Daca am grupa $x_{10} = 10$ cu $x_1 = 1$, $x_9 = 9$ cu $x_2 = 2$, $x_8 = 8$ cu $x_3 = 3$, $x_7 = 7$ cu $x_4 = 4$, $x_6 = 6$ cu $x_5 = 5$, obtinand astfel un numar de transporturi egal cu 5, numar de transporturi optim in situatia de fata.

Astfel, intrucat algoritmul de tip Greedy prezentat mai sus nu returneaza solutia optima pe orice exemplu, el nu este optim.

7 Problema 6.2

Daca raspunsul optim are OPT transporturi, atunci suma totala a elementelor este mai mica sau egala decat OPT*n. Algoritmul de tip Greedy prezentat va umple masina pana cand nu mai incape urmatorul obiect pe care il repartizeaza in urmatoarea masina. Astfel, daca grupam masinile doua cate doua (primul transport cu al 2-lea, al 3-lea cu al 4-lea etc.), atunci fiecare pereche are suma mai mare ca n. Altfel, am fi realizat un singur transport in loc de doua.

Deci, cum suma totala e maxim OPT*n si perechile au suma cel putin n, atunci avem maxim OPT perechi, adica maxim 2*OPT transporturi.

Suplimentar cu exemplu:

Consideram cazul in care avem urmatorul aranjament: intre doua obiecte care ar fi putut fi transportate impreuna inseram un obiect care ocupa volumul n.

De exemplu, cel mai defavorabil caz la care ne putem gandi este: n = 6, iar numerele noastre sunt 1 6 2 5 3 4

Se ajunge in situatia in care masina realizeaza cate un transport pentru fiecare obiect in parte, intrucat acestea nu pot fi grupate intre ele. Cautam sa grupam numerele a caror suma este cea mai apropiata de n (chiar egala cu n daca se poate). Chiar daca n este par, chiar daca este impar, obtinem $n/2 + 1$ drumuri ($(n-2)/2+2$ perechi in cazul unui n par: numarul n nu se poate grupa cu nimeni, si nici elementul aflat la jumatate si $(n-1)/2+1$ perechi in cazul imparelor pentru ca pe n nu il putem grupa cu nimeni). Astfel, algoritmul ce utilizeaza tehnica Greedy realizeaza cel mult 2*OPT drumuri (2*(n/2 +1)).