Tema 2 - Structuri de date

Borza Maria - Cristina

May 16, 2020

1

Demonstrati ca un arbore binar care nu este plin nu poate corespunde unui cod optim.

Solutie:
Fie T un arbore binar care nu este plin, corespunzator unui cod de prefixe.
Fie C alfabetul peste care lucram.
Fie $c.freq$ frecventa unui caracter c, $c \in C$, iar $d_T(c)$ adancimea lui c in arbore. Costul lui T este:

$$Cost(T) = \sum_{c \in C} c.freq \cdot d_T(c)$$

Fie N un nod care are un singur fiu (stim ca exista un astfel de nod deoarece T nu este plin).
Fie X singurul fiu al lui N.
Construim un nou arbore T', prin stergerea nodului N si inlocuirea lui cu nodul X. Fie m o frunza din subarborele lui X (m corespunde unui caracter din alfabet). Costul arborelui nou obtinut este:

$$Cost(T') = \sum_{c \in C} c.freq \cdot d_{T'}(c)$$

$$Cost(T') \leq (\sum_{c \in C - \{m\}} c.freq \cdot d_T(c)) + m.freq \cdot d_{T'}(c)$$

$$Cost(T') \leq (\sum_{c \in C - \{m\}} c.freq \cdot d_T(c)) + m.freq \cdot (d_T(c) - 1)$$

$$Cost(T') \leq (\sum_{c \in C - \{m\}} c.freq \cdot d_T(c)) + m.freq \cdot d_T(c) - m.freq$$

$$Cost(T') \leq (\sum_{c \in C} c.freq \cdot d_T(c)) - m.freq$$

$$Cost(T') \leq Cost(T) - m.freq$$

Cum m apare cel putin odata in fisier $\Rightarrow m.freq \geq 1 \Rightarrow$

$$Cost(T') < Cost(T)$$

Asadar, costul arborele T' este strict mai mic decat costul arborelui $T \Rightarrow$ Codul de prefixe caruia ii corespunde arborele T nu este optim $\Rightarrow$ Un arbore binar care nu este plin nu poate corespunde unui cod optim.

2

Explicati cum se poate modifica metoda de sortare quicksort pentru ca aceasta sa ruleze in cazul cel mai defavorabil (i.e., worst-case) in timp $O(nlogn)$, presupunand ca toate numerele ce trebuie sortate sunt distincte.

Solutie:
Algoritmul de sortare quicksort functioneaza in felul urmator: foloseste tehnica divide et impera. La fiecare pas al recursiei:

1. Se alege un pivot p.
2. Se partitioneaza vectorul in functie de p, in felul urmator: elementele mai mici decat p se pun la stanga lui p in vector, iar cele mai mari decat el se pun in dreapta lui.
3. Se sorteaza cele doua secvente, prin apelearea recursiva a functiei. (se apeleaza pentru subsecventa din vector cu elemente mai mici decat p si pentru cea cu elemente mai mari decat p).

Pentru ca algoritmul sa ruleze in timp $O(nlogn)$ in cazul cel mai defavorabil, vom alege pivotul ca fiind mediana vectorului pe care vrem sa il sortam. Atunci complexitatea algoritmului va fi:

$$T(n) = 2T(\frac{n}{2}) + T'(n) + cn$$

unde $T'(n)$ reprezinta timpul necesar pentru gasirea medianei unui vector cu n elemente, iar c este o constanta, $c > 0$.
(Complexitatea este aceasta deoarece la fiecare pas a recursiei, avem exact $\frac{n}{2}$ elemente mai

mici decat mediana si $\frac{n}{2}$ elemente mai mari decat mediana, iar timpul necesar pentru a partitiona un vector in functie de pivotul ales este $O(n)$).

Fie urmatorul algoritm pentru gasirea medianei unui vector cu n elemente:

1. Se imparte vectorul in grupe a cate 5 elemente.
2. Pentru fiecare din aceste grupe se calculeaza in timp constanat mediana.
3. Se calculeaza mediana medianelor, prin apelarea recursiva a algoritmului.
4. Se alege mediana medianelor ca pivot, si se partitioneaza vectorul in functie de ea (asemanator cu algoritmul quicksort).
5. Daca sunt mai putin de k elemente mai mici decat p in vector,se apeleaza recursiv functia pentru subsecventa din vector cu elemente mai mici decat p (pivotul ales) si tot al k-lea element. Altfel, se apeleaza recursiv functie pentru subsecventa cu elemente mai mari ca p, unde ne intereseaza sa gasim al $k - x$ -lea element (am notat cu k al catalea element din vector vreau il gasesc si cu x numarul de elemente mai mici decat p).

Deoarece am ales pivotul p ca fiind mediana medianelor stim urmatorul lucru: sunt $\frac{n}{10}$ mediane mai mici decat p, iar pentru ca fiecare dintre aceste mediane este mai mare ca jumatate din numere din grupa ei (3 numere), atunci cu siguranta sunt cel putin $\frac{3n}{10}$ elemente mai mici decat $p \Rightarrow$ exista cel mult $\frac{7n}{10}$ elemente mai mari decat p.
Similar se arata ca sunt si cel mult $\frac{7n}{10}$ elemente mai mici decat p. Asadar, complexitatea algoritmului pentru gasirea medianei unui vector cu n elemente va fi, in cazul cel mai defavorabil:

$$T'(n) = T'(\frac{n}{5}) + T'(\frac{7n}{10}) + dn$$

(Complexitatea e aceasta deoarece partionarea vectorului necesita timp $O(n)$).
Vom demonstra acum prin metoda substitutiei ca $T'(n) \in O(n)$.
Presupunem ca $T'(m) \leq c'm, \forall m < n$. Vrem sa aratam ca $T'(n) \leq c'n$.

$$T'(n) \leq c'\frac{n}{5} + c'\frac{7n}{10} + cn$$

$$T'(n) \leq c'(\frac{n}{5} + \frac{7n}{10}) + cn$$

$$T'(n) \leq c'\frac{9n}{10} + cn$$

$$T'(n) \le c'n - c'\frac{n}{10} + cn$$

$$T'(n) \le c'n - n(\frac{c'}{10} - c)$$

$$T'(n) \le c'n \text{ pentru } c' \ge 10c \Rightarrow T'(n) \in O(n).$$

Asadar, complexitatea algoritmului quicksort, cu alegerea pivotului ca fiind mediana vectorului va fi, in cel mai defavorabil caz:

$$T(n) = 2T(\frac{n}{2}) + c'n + cn$$

Vom arata ca $T(n) \in O(nlogn)$.
Prespunem ca $T(m) < c''mlogm$, $\forall m < n$. (Vom nota cu $d = c + c'$). Vrem sa demonstram ca $T(n) < c''nlogn$.

$$T(n) \le 2c''\frac{n}{2}log(\frac{n}{2}) + dn$$

$$T(n) \le c''nlog(\frac{n}{2}) + dn$$

$$T(n) \le c''n(logn - log2) + dn$$

$$T(n) \le c''nlogn - c''nlog2 + dn$$

$$T(n) \le c''nlogn - c''n + dn$$

$$T(n) \le c''nlogn - n(c'' - d)$$

$$T(n) \le c''nlogn, \text{ pentru } c'' > d \Rightarrow T(n) \in O(nlogn)$$

Asadar, daca alegem pivotul in acest fel, complexitatea algoritmului quicksort va fi $O(nlogn)$ in cel mai defavorabil caz.

3

Fie T un arbore binar de cautare si x un nod din arbore care are doi copii. Demonstrati ca succesorul nodului x nu are fiu stang, iar predecesorul lui x nu are fiu drept.

Solutie:
Fie $x.right$ fiul drept al nodului x. Atunci succesorul nodului x este minimul din subarborele lui $x.right$ - sa notam acest element cu y.
Presupunem prin absurd ca y are un fiu stang, fie el $y.left$. Asta ar insemna ca $y.left < y$, dar totodata $y.left$ se afla in subarborele drept al lui x, de unde rezulta ca $y.left > x$.

Asadar, $x < y.left$ si $y.left < y \Rightarrow x < y.left < y$. Ceea ce ar insemna ca nu y este succesorul lui x, ci $y.left$ (contraditie) $\Rightarrow$ Presupunerea facuta este falsa $\Rightarrow$ Succesorul lui x nu are fiu stang. (1)

Fie $x.left$ fiul stang al nodului x. Atunci predecesorul nodului x este maximul din subarborele lui $x.left$ - sa notam acest element cu z.
Presupunem prin absurd ca z are un fiu drept, fie el $z.right$. Asta ar insemna ca $z.right > z$, dar totodata $z.right$ se afla in subarborele stang al lui x, de unde rezulta ca $z.right < x$.
Asadar, $x > z.right$ si $z.right > z \Rightarrow x > z.right > z$. Ceea ce ar insemna ca nu z este predecesorul lui x, ci $z.right$ (contraditie) $\Rightarrow$ Presupunerea facuta este falsa $\Rightarrow$ Predecesorul lui x nu are fiu drept. (2)

Din (1) si (2) $\Rightarrow$ Succesorul lui x nu are fiu stang, iar predecesorul lui x nu are fiu drept.

4

Rezolvati recurenta: $T(n) = T(n/2) + T(n/3) + 1$.

Solutie:
Vom arata prin metoda substitutiei ca $T(n) \in O(n^{0.79})$. Prespunem ca $T(m) \leq cm^{0.79} - d$, $\forall m < n$. Vrem sa demonstram ca $T(n) < cn^{0.79} - d$.

$$T(n) \leq c(\frac{n}{2})^{0.79} - d + c(\frac{n}{3})^{0.79} - d + 1$$

$$T(n) \leq c\frac{n^{0.79}}{2^{0.79}} + c\frac{n^{0.79}}{3^{0.79}} - 2d + 1$$

$$T(n) \leq cn^{0.79}(\frac{1}{2^{0.79}} + \frac{1}{3^{0.79}}) - 2d + 1$$

$$T(n) \leq cn^{0.79} - 2d + 1$$

$$T(n) \leq cn^{0.79} - d, \text{ pentru } d > 1 \Rightarrow T(n) \in O(n^{0.79})$$