Tema 2 – Structuri de Date

Student: Nimară Dan Gabriel
Grupa 141

1 2 puncte

Demonstrati ca un arbore binar care nu este plin nu poate corespunde unui cod optim. Pentru definitia unui cod optim, va rog sa consultati cursul despre coduri Huffman. Va reamintesc ca intr-un arbore binar plin, orice nod cu exceptia frunzelor are exact 2 fii.

Arborele optim are cost minim.

cost(arbore) = $\sum_{i \in frunzelor} frecvență(i) \times adâncime(i)$

Presupunem prin reducere la absurd că un arbore Huffman A incomplet corespunde unui cod optim. Dacă arborele binar nu este plin, atunci ∃ un nod x care are un singur fiu. Dacă eliminăm x din A, adâncimea tuturor nodurilor din subarborele lui x va scădea.

Luăm A' mulțimea frunzelor din subarborele lui x(dacă x nu are frunze, A'= $\{x\}$).

$cost(A - \{x\}) = \sum_{i=frunza \notin A'}(frecvență(i) \times adâncime(i)) + \sum_{i=frunza \in A'}(frecvență(i) \times (adâncime(i-1))$

$cost(A) - cost(A - \{x\}) = \sum_{i=frunza \in A'}(frecvență(i) \times adâncime(i) - frecvență(i) \times (adâncime(i) - 1)) = \sum_{i=frunza \in A'}(frecvență(i) \times [adâncime(i) - (adâncime(i) - 1)]) > 0 => cost(A) - cost(A - \{x\}) > 0 => cost(A - \{x\}) < cost(A)$

cost($A - \{x\}$)<cost A, contradicție deoarece am presupus că A este optim. Așadar, un arbore binar care nu este plin nu poate corespunde unui cod optim.

2 2 puncte

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 $O(n \log n)$, presupunand ca toate numerele ce trebuie sortate sunt distincte.

Algoritmul de sortare quicksort are în cazul cel mai defavorabil complexitatea $O(n^2)$ atunci când vectorul este deja sortat crescător sau descrescător, iar pivotul este primul sau ultimul element din vector.

Pentru ca metoda să ruzele în cazul cel mai defavorabil în timp $O(n \log n)$, trebuie mai întâi să ne legăm de mediana vectorului. Mediana unui vector nesortat poate fi găsită în timp liniar $O(n)$ (algoritm prezentat și demonstrat la curs, inclusiv complexitatea). Mai întâi găsim

mediana, apoi partiționăm vectorul în jurul medianei(aflând ce pivot trebuie luat), iar pe baza partiționării aplicăm metoda quicksort.

Utilizând algoritmul acesta, relația de recurență va fi:

$$T(n) = 2 \times T\left(\frac{n}{2}\right) + \Theta(n)$$

Aplicând Teorema Master, avem: a=2, b=2,

$$f(n) = \Theta(n) = \Theta(n^{\log_2 2})$$

$T(n) \in \Theta(n^{\log_b a} \log n), T(n) \in \Theta(n^{\log_2 2} \log n) <=> T(n) \in \Theta(n \log n)$

În concluzie, $T(n) \in O(n\ log\ n)$.

3 2 puncte

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.

Primul caz:

Într-un arbore binar de căutare, succesorul nodului x este cel mai mic nod mai mare decât x.

Presupunem prin reducere la absurd că succesorul nodului x ar avea fiu stâng. Deoarece în acest tip de arbore fiul stâng al unui nod este mai mic decât acel nod, fiul stâng al succesorului nodului x este mai mic decât succesorul nodului x, dar mai mare decât nodul x(pentru că x are 2 copii din ipoteză). **Contradicție** cu faptul că succesorul lui x este cel mai mic nod mai mare decât x.

Al doilea caz:

Într-un arbore binar de căutare, predecesorul nodului x este cel mai mare nod mai mic decât x.

Presupunem prin reducere la absurd că predecesorul nodului x ar avea fiu drept. Deoarece în acest tip de arbore fiul drept al unui nod este mai mare decât acel nod, fiul drept al predecesorului nodului x este mai mare decât predecesorul nodului x, dar mai mic decât x(pentru că x are 2 copii). **Contradicție** cu faptul că predecesorul lui x este cel mai mare nod mai mare decât x.

4 2 puncte

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

$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{3}\right) + n$

Vrem să arătăm că T(n)=O(n).

Presupunem că $T(\frac{n}{2}) \leq c\frac{n}{2}$ și $T(\frac{n}{3}) \leq c\frac{n}{3}$

$T(\frac{n}{2})+ T(\frac{n}{3}) \leq c\frac{n}{2} + c\frac{n}{3} \quad |+n$

$T(n) \leq c\frac{n}{2} + c\frac{n}{3} + n$

$T(n) \leq \frac{3cn+2cn+6n}{6}$

$\frac{n(5c+6)}{6} \leq cn$

$\frac{5c+6}{6} \leq c \ |* 6$

$5c + 6 \leq 6c \leftrightarrow c \geq 6$, de unde rezultă că T(n) = O(n) $\forall c \geq 6$.