Gavrila Andreea Tema nr 2 – Structuri de Date 01 Iunie 2020
Grupa 144

Exercitiul 1: Demonstrati ca un arbore binar care nu este plin nu poate corespunde unui cod optim.

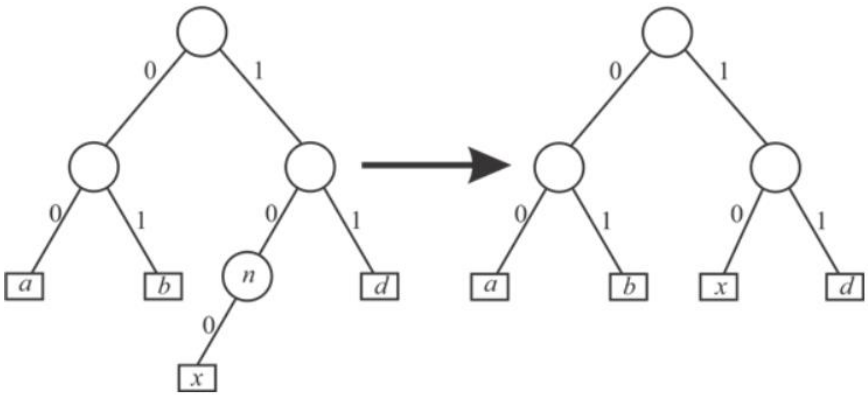
Rezolvare:

„Un *arborele binar plin* este un arbore în care orice nod cu excepția frunzelor are exact 2 fii.”
_ Introduction to Algorithms, Third Edition. Autori: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Cliord Stein

Presupun că există un arbore binar care nu este plin, notat T reprezentat de un cod optim C.
Frunzele sunt caracterele date, iar un cuvânt de cod binar pentru un caracter este drumul de la rădăcină până la caracterul respectiv, unde 0 reprezintă “mergi la fiul stâng” iar 1 “mergi la fiul drept”.

Din moment ce nu este plin, există un nod n care are un singur fiu notat x.
Acesta poate fi un fiu stâng sau un fiu drept, prin urmare muchia de la nodul n la nodul x va adăuga bitul 0, dacă x este fiu stâng sau bitul 1, dacă x este fiu drept, în codul prefix.

Deoarece „ în orice mod am alege două reprezentări a două elemente, niciuna din reprezentări nu este prefix pentru cealaltă. ” _ Introduction to Algorithms, Third Edition. Autori: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Cliord Stein
Pentru a obține codul optim, înlocuiesc nodul n cu x și elimin muchia de la n la x.
Astfel, lungimea codului prefix este redusă cu 1. Acest lucru înseamnă că noul arbore are un cost mai mic față de cel inițial. (contradictie)



$$B(T) = \sum_{c \in C} frecv[c] * d_T(c)$$

$$B(T_1) = \sum_{c \in C} frecv[c] * d_{T_1}(c) = \sum_{c \in C} frecv[c] * (d_T(c) - 1)$$

Cum $d_{T_1}(c) \ = \ d_T(c) - 1 \ < \ d_T(c)$

⇨ $B(T_1) \ < B(T)$, dar T era un arbore asociat unui cod optim C $\Longrightarrow contradicție$

În concluzie, orice arbore arbore binar asociat unui cod optim trebuie să fie plin. ■

<u>Exercitiul 2</u>: Explicati cum se poate modifca 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.

Rezolvare:

„ Algoritmul de sortare *Quicksort* rulează cu timp pătratic $O(n^2)$ pentru cel mai nefavorabil caz, dacă partiţionarea este total dezechilibrată la fiecare pas recursiv al algoritmului. " _ Introduction to Algorithms, Third Edition. Autori: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Cliord Stein

$T(n) = T(n-p-1) + T(p-1) + \theta(n)$, unde p este poziția pivotului, iar $\theta(n)$ timpul necesar partiţionării.

Justificarea complexitatii:

Dacă algoritmul de partiţionare ar produce doi vectori de n/2 elemente, și pivotul ar coincide cu mediana șirului, algoritmul de sortare rapidă lucrează mult mai repede. Formula de recurenţă în acest caz este:

$T(n) = 2\,T(n/2) + \theta(n)$

(aplicând Teorema Master) (a=2, b=2, f(n)= θ (n))

⇨ $T(n) \in O(\text{n} \log n)$ ■

Justificarea corectitudinii:

Pentru ca algoritmul să ruleze în cazul cel mai defavorabil în timp $O(\text{n} \log n)$, trebuie să se determine mediana unui șir nesortat în $O(n)$.

Pot aplica astfel algoritmul *Randomized-Select,* care este modelat pe baza algoritmului de sortare rapidă, deoarece ideea este de a partiţiona recursiv tabloul de intrare. Astfel dacă șirul se va sorta, se va selecta a *n*-a valoare, dar fără să se sorteze şirul. Spre deosebire de sortarea rapidă, care prelucrează recursiv ambele componente ale partiţiei, *Randomized-Select* lucrează numai cu o componentă.

Această diferenţă se evidenţiază la analiză: în timp ce *QuickSort* are un timp mediu de execuţie de $\theta(n \log n)$, timpul mediu al algoritmului *Randomized-Select* este $\theta(n)$.

„ Cazul cel mai defavorabil depinde de generatorul de numere aleatoare. Chiar dacă dorim, nu reuşim să generăm un vector de intrare nereuşit, deoarece permutarea aleatoare va face ca ordinea datelor de intrare să fie irelevantă. " _ Introduction to Algorithms, Third Edition. Autori: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Cliord Stein

Timpul de execuţie, în cazul cel mai defavorabil, pentru *Randomized-Select* $\theta(n^2)$, chiar şi pentru găsirea minimului, pentru că am putea fi extrem de nenorocoşi şi să partiţionăm în jurul celui mai mare element rămas.

Algoritmul lucrează bine în cazul mediu, dar fiind aleator, nu există date de intrare particulare care să provoace comportamentul celui mai defavorabil caz. Astfel, orice statistică de ordine şi în special mediana, poate fi determinată într-un timp mediu liniar.

În concluzie, se obține un *QuickSort* care rulează în cazul cel mai defavorabil în timp $O(\text{n} \log n)$. ■

Exercitiul 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.

Rezolvare:

- Demonstrez că dacă nodul x are fiu drept, atunci *succesorul său nu are fiu stâng:*

Presupun prin absurd ca (∃) un nod y, astfel încât *y = fiu stâng al Succesor(x)* .

$$\left.\begin{array}{l} y = \text{fiu stâng al Succesor}(x) \implies y < \text{Succesor}(x) \\ \Rightarrow\ y \text{ și Succesor}(x) \text{ sunt în subarborele drept al lui } x \implies y > x \text{ și Succesor}(x) > x \end{array}\right\} \begin{array}{l} \textit{(contradicție)} \\ y = \text{Succesor}(x) \end{array}$$

- Demonstrez că dacă nodul x are fiu stâng, atunci *predecesorul său nu are fiu drept:*

Presupun prin absurd ca (∃) un nod y, astfel încât *y = fiu drept al Predecesor(x)* .

$$\left.\begin{array}{l} y = \text{fiu drept al Predecesor}(x) \implies y > \text{Predecesor}(x) \\ \Rightarrow\ y \text{ și Predecesor}(x) \text{ sunt în subarborele stâng al lui } x \implies y < x \text{ și Predecesor}(x) < x \end{array}\right\} \begin{array}{l} \textit{(contradicție)} \\ y = \text{Predecesor}(x) \end{array}$$

În concluzie, succesorul nodului x nu are fiu stâng, iar predecesorul lui x nu are fiu drept. ■

Exercitiul 4: Rezolvati recurenta T(n) = T(n/2) + T(n/3) + 1. Demonstrati.

Rezolvare:

Pentru rezolvarea acestei recurențe aplic *metoda Akra-Bazzi* explicată în _ Introduction to Algorithms, Third Edition. Autori: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Cliord Stein

$$T(x) = \begin{cases} \sum_{i=1}^{k} a_i\, T(b_i x) + f(x) & \text{, dacă } x > x_0 \\ \theta(1) & \text{, dacă } 1 \le x \le x_0 \end{cases}$$

unde

- $x \ge 1$ *este un nr real*
- x_0 *este o constantă astfel încât* $x_0 \ge \frac{1}{b_i}$ *și încât* $x_0 \ge \frac{1}{1-b_i}$ *pentru* $i = 1, 2, \dots, k$
- a_i *este o constantă pozitivă pentru* $i = 1, 2, \dots, k$
- b_i *este o constantă astfel încât* $0 < b_i < 1$ *pentru* $i = 1, 2, \dots, k$
- $k \ge 1$ *este o constantă de tip întreg*
- *f(x) este o funcție non-negativă care satisface condiția de creștere polinomială:*
  (∃) *constante pozitive* c_1 *și* c_2 *astfel încât*
  pentru toți $x \ge 1$, *pentru* $i = 1, 2, \dots, k$ *și pentru toți u astfel încât* $b_i x \le u \le x$,
  avem $c_1 f(x) \le f(u) \le c_2 f(x)$
  $|f(x)| \in O(x^c)$, *unde c este o constantă*

Rezolv folosind *metoda Akra-Bazzi* următoarea recurență:

$$T(n) = T(n/2) + T(n/3) + 1$$

Avem:

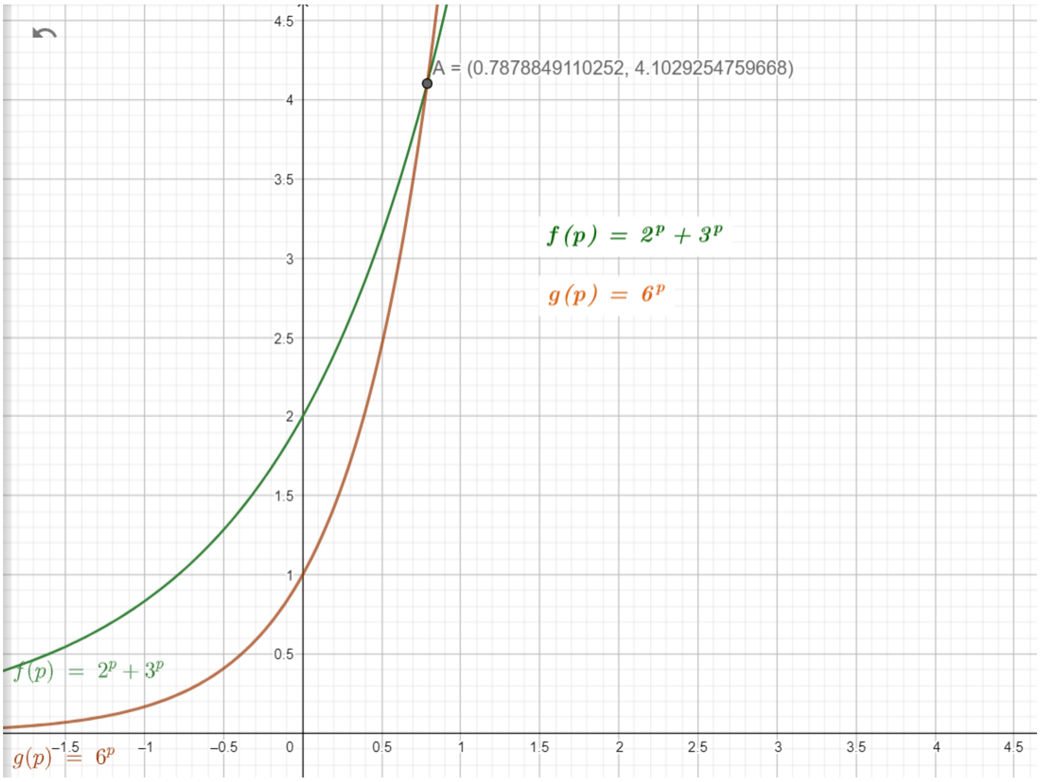
- $a_i = 1 \;\; pentru \; \forall \; i \in \mathbb{N}$
- $b_1 = \frac{1}{2} \;\; și \; b_2 = \frac{1}{3} \;\; , \; k = 2$
- $f(n) = 1 \;\in\; O(n^c) \;\; , unde \; c \; este \; o \; constantă$

Următorul pas este să găsim unicul nr real p care îndeplinește condiția $\sum_{i=1}^{k} a_i \; b_i^p = 1$
(„știm că p este un unic nr real care există întotdeauna”) _ Introduction to Algorithms, Third Edition. Autori: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Cliord Stein

$$\sum_{i=1}^{k} a_i \; b_i^p = 1 \;\Leftrightarrow\; \left(\frac{1}{2}\right)^p + \left(\frac{1}{3}\right)^p = 1$$

$$\Leftrightarrow \frac{1}{2^p} + \frac{1}{3^p} = 1 \;\Leftrightarrow\; 2^p + 3^p = 6^p$$

Rezolv această ecuație prin metoda grafică:



Astfel am obținut $p = 0{,}7878$

Conform *metodei Akra-Bazzi* calculez acum clasa asimptotică în care se încadrează T(n)

$$T(n) \; \epsilon \; \theta\left(\; n^p \; \left(1 + \int_1^n \frac{f(n)}{u^{p+1}} du \right)\right)$$

$$notez \quad I = \int_1^n \frac{f(n)}{u^{p+1}} du = \int_1^n \frac{1}{u^{p+1}} du = \int_1^n u^{-(p+1)}\, du = \left. \frac{u^{-p-1+1}}{-p-1+1} \right|_1^n = \left. \frac{u^{-p}}{-p} \right|_1^n = \left. \frac{1}{-p\, u^p} \right|_1^n$$

$$= \frac{1}{-p\, n^p} - \frac{1}{-p\, 1^p} = \frac{1}{-p\, n^p} + \frac{1}{p} \overset{n \to \infty}{=} 0 + \frac{1}{p} = \frac{1}{p}$$

Prin urmare,

$T(n)\, \epsilon\, \theta \left(n^p \left(1 + \int_1^n \frac{f(n)}{u^{p+1}} du \right) \right)$, devine:

$$T(n)\, \epsilon\, \theta \left(n^p * \left(1 + \frac{1}{p} \right) \right) \Leftrightarrow T(n)\, \epsilon\, \theta \left(n^p + n^p \frac{1}{p} \right) \Rightarrow T(n)\, \epsilon\, \theta\, (n^p)$$

Cum am obținut $p = 0{,}7878$, recurența dată:

⇨ $T(n)\, \epsilon\, \theta\, (n^{0,7878})$ ■