Gavrila Andreea Tema nr 1 – Structuri de Date 24 Aprilie 2020

Grupa 144

Exercitiul 1: Demonstrati ca orice algoritm care construieste un arbore binar de cautare cu n numere ruleaza in timp Ω(n log n).

Rezolvare:

- ⇨ " Operaţiile de bază pe arborii binari consumă un timp proporţional cu înălţimea arborelui.
- ⇨ **Pentru un arbore binar complet cu n noduri, aceste operaţii se execută în cazul cel mai defavorabil într-un timp ϴ(log n).**
- ⇨ Dacă însă arborele este un lanţ liniar de n noduri, atunci timpul consumat în cazul cel mai defavorabil este ϴ(n).
- ⇨ Înălţimea unui arbore binar de căutare construit aleator este O(log n), deci operaţiile de bază pe mulţimile dinamice vor consuma un timp de ϴ(log n). "
  _ Introduction to Algorithms, Third Edition. Autori: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Cliord Stein

- Se stie ca este nevoie de un timp ϴ(n) pentru a traversa un arbore binar de căutare cu n noduri.
- (Teorema) Algoritmul de sortare bazat pe comparatii intre chei are nevoie de un timp de rulare Ω(n log n).

Presupun ca nu este adevarata propozitia ce trebuie demonstrata: „In cazul cel mai defavorabil, orice algoritm care construieste un arbore binar de cautare cu n numere ruleaza in timp Ω(n log n)."

- Astfel, obtinem un algoritm de sortare prin modelul comparatiei care necesită un timp mai puțin de Ω(n log n).

Acest lucru este o contradictie cu prima teorema.

- Prin urmare, este adevarat ca orice algoritm care construieste un arbore binar de cautare cu n numere ruleaza in timp Ω(n log n), in cazul cel mai defavorabil. ■

Exercitiul 2: Demonstrati ca daca $f(n) = \Theta\, g(n)$si $g(n) = \Theta\, h(n)$ atunci $f(n) = \Theta\, h(n)$.

Rezolvare:

$f(n) = \Theta\, g(n) \Leftrightarrow g(n) = \Theta\, f(n) \quad \Leftrightarrow$

$\Leftrightarrow \quad \exists\, c_1, c_2, n_0 > 0\ astfel\ incat\ \forall\, n \geq n_0\ avem\ \ c_1\, g(n) \leq\ f(n) \leq\ c_2\, g(n)$

$g(n) = \Theta\, h(n) \Leftrightarrow h(n) = \Theta\, g(n) \ \Leftrightarrow$

$\Leftrightarrow \quad \exists\, c_3, c_4, n_0 > 0\ astfel\ incat\ \forall\, n \geq n_0\ avem\ \ c_3\, h(n) \leq\ g(n) \leq\ c_4\, h(n)\ \ /\ \ \cdot c_1$

⇨ $c_1\, c_3\, h(n) \leq\ c_1\, g(n) \leq\ c_1\, c_4\, h(n) \Leftrightarrow \boldsymbol{c_1\, c_3\, h(n)} \leq\ c_1\, g(n)\ \leq \boldsymbol{f(n)} \leq\ c_2\, g(n) \leq \boldsymbol{c_1\, c_4\, h(n)}$

⇨ $\boldsymbol{c_1\, c_3\, h(n)}\ \leq f(n) \leq \boldsymbol{c_1\, c_4\, h(n)}$

notez $\boldsymbol{c_1\, c_3 = c_5\,,\ \ c_1\, c_4 =\ c_6\,,\ \ c_5, c_6 > 0}$

⇨ $c_5\, h(n)\ \leq f(n) \leq c_6\, h(n)\,,\ c_5, c_6 > 0\ \ \Leftrightarrow\ \ h(n) = \Theta\, f(n) \Leftrightarrow \boldsymbol{f(n) = \Theta\, h(n)}$ ■

Exercitiul 3: Demonstrati ca $\log n = o\,(\sqrt{n}\,)$

Rezolvare:

Notez: $g(n) =\ \log n$ si $f(n) =\ \sqrt{n}$

$g(n) =\ o\,(f(n)) \Leftrightarrow$

$\Leftrightarrow\ \ \forall\, c > 0,\ \exists\, n_0 > 0\ astfel\ incat\ \ n \geq n_0\ avem\ g(n) < c\, h(n)\ \Rightarrow\ \ \log n\ < c\,\sqrt{n}$

Stim ca $\log x < x$, $\forall\, x > 0$.

Atunci $\log \sqrt{x} < \sqrt{x}$, $\forall\, x > 0$ / $\cdot 2$ $\Rightarrow$ $2 \log \sqrt{x} < 2\sqrt{x} \Rightarrow 2 \log x^{1/2} < 2\sqrt{x}$

$\Rightarrow 2\,\frac{1}{2} \log x < 2\sqrt{x} \Rightarrow \log x < 2\sqrt{x}$, $\forall\, x > 0$

$\forall\, c > 0,\ \exists\, n_0 > 0\ astfel\ incat\ n \geq n_0\ avem\ \log n < c\sqrt{n}$

⇨ Pentru $c = 2,\ n_0 = 0 \Rightarrow \log n < 2\sqrt{n}$, $\forall\, n \geq 0$ (Adevarat) ∎

<u>Exercitiul 4</u>:

Se da un sir cu n numere de la 1 la n, cu exceptia unui numar care apare de 2 ori. Determinati numarul care apare de doua ori. Pentru un algoritm de complexitate O(n^2) se acorda 0,5 puncte. Pentru un algoritm de complexitate O($n \log n$) veti primi 1 punct, iar pentru un algoritm de complexitate O(n) care foloseste doar O(1) spatiu suplimentar (adica fara vector de frecvente) veti primi 1,5 puncte.

Exemplul 1: 2 1 3 3 4 => Elementul duplicat este: 3

Exemplul 2: 4 1 5 5 2 3 => Elementul duplicat este: 5

Rezolvare:

```
{ int v[100],  n,  i,  suma,  elem_duplicat;
suma = 0;
cin>>n;
for( i =0; i < n; i++)
        cin>> v[ i ];

for( i =0; i < n; i++)
        suma = suma + v[ i ];      //adun suma tuturor numerelor

elem_duplicat = suma –   (n-1)*n / 2;           // 4+1+5+5+2+3 – (1+2+3+4+5) = 5
                 // din suma tuturor elementelor citite, scad suma elementelor de la 1 la (n-1). ( = (n−1) n / 2 ).
cout<< elem_duplicat;
return 0;
}
 // timpul de executie al programului este O(n) care foloseste doar O(1) spatiu suplimentar.
// (adica nu foloseste niciun vector de frecvente)
```

```cpp
#include <iostream>

using namespace std;

int main()
{

    int v[100], n, i, s=0, el;
    cout << "n=";  cin >> n;
    for (i = 0; i < n; i++)
    {
       cout << "v[" << i << "]= ";
       cin >> v[i];
    }

    for(i=0;i<n;i++)
        s=s+v[i];

    cout<<"\ns="<<s<<endl;
    el=s - (n-1)*n/2;

    cout<<"elementul_duplicat= "<<el;
    return 0;
}
```

```
n=6
v[0]= 4
v[1]= 1
v[2]= 5
v[3]= 5
v[4]= 2
v[5]= 3

s=20
elementul_duplicat= 5
Process returned 0 (0x0)   execution time : 4.803 s
Press any key to continue.
```

<u>Exercitiul 5</u>:

Fie X[1 :: n] si Y [1 :: n] doi vectori, _ecare continand n numere sortate. Prezentati un algoritm care sa gaseasca mediana celor 2n elemente. Mediana unei multimi de n elemente este elementul de pe pozitia [n/2] in sirul sortat. De exemplu, mediana multimii 3; 1; 7; 6; 4; 9 este 4.

In functie de timpul de rulare al algoritmului veti primi urmatoarele punctaje:
O(n log n) - (0,25 puncte); O(n) - (0,5 puncte); O(log^2 n) - (1 punct); O(log n) - (1,5 puncte).

Rezolvare:

```
def mediana ( X, Y, n):
{    daca n == 1 atunci
            return ( X[0] + Y[0] ) / 2 ;
     altfel
        daca n == 2 k + 1  atunci
            { m1 = X[k];
              m2 = Y[k];
              daca  m1 < m2  atunci
                    return mediana ( X[ k : n ], Y[ 0 : k+1],  k+1 ) ;
              daca  m1 > m2  atunci
                    return mediana ( X[ 0 : k+1 ], Y[ k : n],  k+1 ) ;
              altfel
                    return  m1 ;
            }

        daca n == 2 k   atunci
            { m1 = ( X[k − 1] + X[k] )/ 2 ;
              m2 = ( Y[k − 1] + Y[k] )/ 2 ;
              daca  m1 < m2  atunci
                    return mediana ( X[ k : n ], Y[ 0 : k],  k ) ;
              daca  m1 > m2  atunci
                    return mediana ( X[ 0 : k ], Y[ k : n],  k ) ;
              altfel
                    return  m1 ;
            }
}
```

Justificarea complexitatii:

T(n) = T(n/2) + 1   (a=1, b=2, f(n)=1)

$$\left.\begin{array}{l} f(n) \in O\left(n^{\log 1-\varepsilon}\right) = O(n^{-\varepsilon}) \ ,\ \varepsilon > 0 \\ f(n) \in O(1) \qquad\qquad n^{-\varepsilon} = \frac{1}{n^{\varepsilon}} \xrightarrow{n\to\infty} 0 \end{array}\right\} \Rightarrow f(n) \notin O\left(n^{\log 1-\varepsilon}\right)$$

$f(n) \in \Omega(n^{\varepsilon})$? ⇒ nu se incadreaza

$f(n) \in \theta(n^0 \cdot \log n^k) \Leftrightarrow f(n) \in \theta(\log n^k)$

pentru k=0 $\Leftrightarrow f(n) \in \theta(1)$ *(Adevarat)*

⇨ T(n) ∈ $\theta(\log n)$ ■

Exercitiul 6:

Sa presupunem urmatoarele. Ati castigat la loterie si v-ati cumparat o vila pe care doriti sa o mobilati. Deoarece Ferrari-ul dvs. are capacitate limitata, doriti sa faceti cat mai putine drumuri de la magazin la vila. Mai exact, Ferrari-ul are capacitate n, iar dumneavoastra aveti de cumparat k bunuri de dimensiune x1; x2, …, xk.

Fie urmatorul algoritm greedy. Parcurgem bunurile in ordinea 1, 2, …, k si incercam sa le punem in masina. In momentul in care un bun nu mai incape in masina, efecutam un transport si continuam algoritmul.

1. Demonstarti ca algoritmul prezentat mai sus nu este optim. (0.5 puncte)
2. Fie OPT, numarul de drumuri in solutia optima. Demonstrati ca algoritmul greedy prezentat mai sus efectueaza cel mult 2OPT drumuri. (1 punct).

Rezolvare:

1. Pentru a demonstra ca algoritmul prezentat nu este optim, trebuie sa gasesc un contraexemplu prin care se evidentiaza acest lucru.

Initial, presupun ca acest algoritm greedy prezentat in cerinta este optim.

- Pentru un exemplu in care capacitatea n = 7 , iar sirul de bunuri este (4, 2, 3, 2, 1, 2) aplic algoritmul.
- Se obtin: (4, 2), (3, 2, 1), (2) => 3 drumuri.
- Dar daca aranjez altfel cele k bunuri, pot obtine o solutie mai optima.
  Sirul de bunuri rearanjat: (4, 2, 1, 3, 2, 2)
  Se obtin: (4, 2, 1), (3, 2, 2) => 2 drumuri.

In concluzie, algoritmul greedy prezentat in cerinta nu este optim. ■

2. Voi folosi exemplul anterior in care capacitatea masinii n = 7 , iar sirul de bunuri este (4, 2, 3, 2, 1, 2) si aplic algoritmul greedy.

Obtin astfel:

- (4, 2), (3, 2, 1), (2) => 3 drumuri.
  ↓ ↓ ↓
  1 1 5 (pierderea din fiecare drum = Capacitatea masinii – Greutatea per drum)

- Greutatea Totala a bunurilor = 14 (= 4+2+3+2+1+2)
- Greutatea Pierderilor = 7 (= 1+1+5)
- Capacitatea masinii = n =7
- Nr de drumuri din solutia greedy, NrDGr = 3
- Nr de drumuri din solutia optima, OPT = 2 (demonstrata la pct 1)

(1+1+5) + (4+2+3+2+1+2) = 7 * 3

7 + 14 = 7 * 3

greutate pierderi greutate totala = capacitatea masinii NrDGr

$\Rightarrow$ $3 = \frac{7 \ + \ 14}{7}$ $\left(\text{NrDGr} = \frac{\text{greutate pierderi } + \text{ greutate totala}}{\text{capacitatea masinii}}\right)$

Se stie ca greutatea Pierderilor ≥ 0 (**cand pierderile sunt zero, avem atunci OPT, solutia optima**)

Consider acum cazul solutiei optime: nr de drumuri = OPT = 2

$\Rightarrow$ $2 \geq \frac{0 \ + \ 14}{7}$ $\left(OPT \geq \frac{\text{greutate pierderi } + \text{ greutate totala}}{\text{capacitatea masinii}}\right)$

Notez aceasta relatie cu (1).

(1+1+5) < (4+2+3+2+1+2)

7 < 14

greutate pierderi greutate totala

Doresc acum sa formez in membrul stang Numarul de drumuri din solutia greedy, NrDGr.

$7 < 14 \quad / +14 \quad \Rightarrow \quad 7 + 14 < 14 + 14 \quad / : 7 \quad \Rightarrow \quad \frac{7+14}{7} < \frac{2*14}{7} \quad \Rightarrow \quad 3 < 2 * \frac{14}{7}$ $\left(\text{NrDGr} < 2\frac{\text{greutate totala}}{\text{capacitatea masinii}} \right)$

Din relatia (1) $\Rightarrow \quad \frac{14}{7} \leq 2 \quad \left(\frac{\text{greutate totala}}{\text{capacitatea masinii}} \leq OPT \right)$

⇨ NrDGr $< 2 \frac{\text{greutate totala}}{\text{capacitatea masinii}} \leq 2 * OPT \qquad \left(3 < 2 * \frac{14}{7} \leq 2 * 2 \right)$

⇨ NrDGr $\leq 2 * OPT$ ■