DIMA OANA-TEODORA
Grupa 141

TEMA 1: STRUCTURI DE DATE

+1 punct din oficiu

1 1,5 puncte

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

R:
Creearea unui arbore binar de cautare se face prin inserarea succesiva a n noduri.
Deci constructia oricarui arbore binar de cautare necisita o complexitate Ω(nH), unde:
H=inaltimea arborelui (in fiecare nivel va avea loc o comparatie)=log(n+1).
Demonstrez ca orice arbore de cautare cu n noduri are logn nivele:

Tratez cazul cand arborele este echilibrat (adica avem un numar minim de nivele)

Observ ca : Niv 1 …….. 1 nod
Niv 2……4 noduri
…….
Niv x ……2^{x-1} noduri

=> numarul total de noduri= suma puterilor lui 2 care depinde de inaltimea arborelui

$=\Sigma_i 2^{i-1}$ i=1,…H=>$\Sigma_i 2^{i-1} = 2^H-1$

Astfel am demonstrat ipoteza.

2 1,5 puncte

Demonstrati ca daca f(n) = Θ(g(n)) si g(n) = Θ(h(n)) atunci f(n) = Θ(h(n)).

R:
Fie a, b, c, d numere reale poztive (1)=> produsele ac, bd reale pozitive (5)
Daca f(n) = Θ(g(n)) , atunci $a < \frac{f(x)}{g(x)} <$ b (2)
Daca g(n) = Θ(h(n)), atunci $c < \frac{g(x)}{h(x)} <$ d (3)
Din (1) cu (2)*(3) => ac< $\frac{f(x)}{h(x)} <$ bd (6)
Din (5) si (6)=> concluzia

3 1,5 puncte

Demonstrati ca logn = o(√n)

R:
o(g(n)) = {f(n)| ∀ c>0 constant, ∃m >0 constant a.i. 0≤f(n)≤c g(n) , n≥m}
Identific:

f(n)=logn
g(n)= $\sqrt{n}$ => $0 \leq \log n \leq c\sqrt{n}$
Fie h(x)= $\frac{\log n}{\sqrt{n}}$
Vreau sa demonstrez ca $h(x) \leq c$.
Derivez h si obtin ca h'(x)= $\frac{2-\log n}{2n^{\frac{3}{2}}}$
Atasez ecuatia h'(x)=0 si obtin solutia : n=e^2
Observ ca h este strict crescatoare pentru $n\epsilon(0, e^2)$ si strict descrescatoare pentru $n\epsilon(e^2, \infty)$, iar pentru n=e^2 are valoarea 0.
Daca n=e^2 ⇨ log $e^2 \leq c\sqrt{e^2}$ ⇨ c=$\frac{2}{e}$ >0 de unde concluzia

4 1,5 puncte

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.
Exemplul 1: 2 1 3 3 4 Elementul duplicat este: 3
Exemplul 2: 4 1 5 5 2 3 Elementul duplicat este: 5

R:

```
Citeste n (numar natural)
Citeste v (sirul de n termeni)
Pentru i de la 0 la n-1 executa
        Daca v[modul( v[i])-1] <0  atunci
            Gasit=modul(v[i])
            Break
        v[modul(v[i])-1]*=-1 //trec peste un numar
Scrie Gasit (numarul duplicat)
```

Algoritmul se bazeaza pe faptul ca daca ajung sa trec peste un numar a doua oara (semnalizez prin transformarea lui in numar negative la prima trecere) inseamna ca am gasit duplicatul .
Complexitate timp : O(n)
Complexitate spatiu <u>suplimentar</u>: O(1) nu am folosit vectori *suplimentari* cum este cel de frecventa

5 1,5 puncte

Fie X[1 :: n] si Y [1 :: n] doi vectori, fiecare 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.
Exemplu, mediana multimii 3,1,7,6,4,9 este 4.

R:
Algoritmul are la baza urmatoarea idee:

Deoarece ambii vectori sunt de dimensiuni egale, ca renunt in a cauta mediana in prima jumatate a vectorului cu mediana mai mica si in a doua jumatate a vectorului cu mediana mai mare.

```
Mediana (v1,v2, n, m)
        s = 0
        d = n
         Cat timp s<=d executa :
                    i = (s+d) / 2
                    j = ((n+m+ 1) / 2) – i
              #i=n a doua jumatate a lui v1 e goala(1)
              #j=0 prima jumatate a lui v2 e goala(2)
              #i=0 prima jumatate a lui v1 e goala(3)
              #j=m a doua jumatate a lui v2 e goala(4)
              (1)+(2)=> caut in dreapata
              (3)+(4)=>caut in stanga

                    Daca i < n si  j >0 si v2[j - 1] > v1[i]  atunci :
                              s = i + 1
                    Altfel:
                          Daca i > 0 si  j < m si  v2[j] < v1[i - 1] atunci:
                                    d = i - 1
                          Altfel:
                              Daca i = 0 atunci:
                                        Returneaza v2[j - 1]
                              Daca j = 0 atunci:
                                        Returneaza v1[i - 1]
                              Altfel:
                                      Returneaza max(v1[i - 1], v2[j - 1])

              ------------------------------------------------
              Citeste n(numar natural)
              Citeste X, Y (vectori sortati de lungime n)
              M=Mediana(X,Y, n, n)
              Scrie M (mediana celor 2 vectori)
```

Complexitate algoritm prezentat: O(log n)
Demonstrarea complexitatii se face folosind Teorema Master
$T(n)=aT(n/b^k)+f(n)$
Aici: T(n)=T(n)+1
Conform Teoremei Master:
Identific a=1 si b=2 si k=0=> a=b => Algoritmul ruleaza in $O(n^k \log n)$=>O(logn)

6 1,5 puncte

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)

Fie O algoritmul prezentat .

R:

Presupun prin absurd ca O este optim.

Astfel presupunem ca masina efectueaza k drumuri pentru cele k bunuri.

Fie O' un algoritm care sorteaza intai dimensiunile bunurilor in ordine crescatoare.

Astfel poate rezulta un numar de k-m drumuri efectuate, unde m este numarul de bunuri care depasesc capacitatea n.

Astfel am gasit ca O' este un algoritm optim, deci presupunerea nu e buna, rezultand ca O nu e optim.

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

R:

Un algoritm optim ar efectua cat mai putine drumuri, adunand bunuri de dimensiune cat mai mare .

Fie s=suma dimensiunilor obiectelor pe care le transporta printr-un algoritm efficient

Studiez algoritmul prezentat de exercitiu:

Observ ca in orice transport, dimensiunile totale ale transportului trebuie sa fie $\leq$ n.

Exemplu caz 1:

Presupun ca as avea nevoie de 3 transporturi ca sa transport toata cantitatea s, ceea ce ma conduce la faptul ca suma a 2 transporturi este mai mica (strict) ca n. Cum s este clar mai mic ca n , ajung la concluzia ca sunt necesare doar 2 transporturi si nu 3. Deci are loc o contradictie.

Exemplu caz 2:

Presupun ca as avea nevoie de 4 transporturi ca sa transport toata cantitatea s, ceea ce ma conduce la faptul ca suma a 2 transporturi este mai mica (strict) ca n. Cum s este clar mai mic ca n , ajung la concluzia ca sunt necesare doar 2 transporturi si nu 4, iar pentru urmatoarele 2 algoritmul se repeta. Deci are loc o contradictie, avand nevoie in total de 2 transporturi .

Concluzie:

Orice transport al bunurilor poate fi efectuat in cel mult doua transporturi mici folosind algoritmul propus.