Tema I- structuri de date

Mîndruleanu Matei Daniel -143.

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

Trebuie sa construm un arbore binar de cautare cu n elemente. Pentru fiecare dintre aceste elemente trebuie parcursa inaltumea arborelui. Inaltimea minima a acestuia este n log(n) => timpul de rulare este Ω(n log n).

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

f(n) = Θ(g(n)) => ∃ c1, c2 ai c1×g(n) <= f(n) <= c2×g(n)

g(n) = Θ(h(n)) => ∃ c3, c4 ai c3×h(n) <= g(n) <= c4×h(n)

=> c1×c3×h(n) <= c1×g(n) <= c1×c4×h(n)

=> c2×c3×h(n) <= c2×g(n) <= c2×c4×h(n)

=> c1×c3×h(n) <= c1×g(n) <= f(n) <= c2×g(n) <= c2×c4×h(n)

=> c1×c3×h(n) <= f(n) <= c2×c4×h(n)

=> f(n) = Θ(h(n))

3. Demonstrati ca log n = o($\sqrt{n}$).

log n = o($\sqrt{n}$) <=> $\lim_{n\to\infty} \left(\frac{\log n}{\sqrt{n}}\right)^1 = 0$

$\lim_{n\to\infty} \left(\frac{\log n}{\sqrt{n}}\right)^1 = \lim_{n\to\infty} \left(\frac{(\log n)'}{(\sqrt{n})'}\right)^1 = \lim_{n\to\infty} \left(\frac{\frac{1}{n \ln 10}}{\frac{1}{2\sqrt{n}}}\right)^1 = \lim_{n\to\infty} \left(\frac{2\sqrt{n}}{n \ln 10}\right)^1 = \frac{2}{\ln 10} \lim_{n\to\infty} \left(\frac{1}{\sqrt{n}}\right)^1 =$

$= \frac{2}{\ln 10} \times \frac{1}{\infty} = \frac{2}{\ln 10} \times 0 = 0$ => log n = o($\sqrt{n}$)

4. 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(n2) 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

```
max = a[0];
for ( int i=0; i<n; i++ ){
if ( a[i] > max ) max = a[i];
sum += a[i];
}
return ( sum - ( max*(max+1) ) /2 );
T = O(n)
```

5. 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. 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(log2 n) - (1 punct); O(log n) - (1,5 puncte).

```
int med(int v[], int c[], int n)
{
    if (n == 1)
        return ( v[0]>c[0] ? c[0] : v[0]  );
    if (n == 2)
        if ( v[0] > c[0] ){
          if ( v[0] > c[1] ) return c[1];
          else return v[0];
        }
        if ( v[0] < c[0] ){
          if ( c[0] > v[1] ) return v[1];
          else return c[0];
        }
    int m1 = mediana_vector (v, n);
    int m2 = mediana_vector (c, n);
    if (m1 == m2)  return m1;
    if ( m1 < m2 ){
      return med( v+n/2-1, c, n-n/2+1 );
    }
    return med( c+n/2-1, v, n-n/2+1);
}
T = O(log n)
```

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).

1. Algoritmul prezentat mai sus nu este optim intrucat nu utilizeaza masina la capacitate maxima la fiecare drum. Algoritmul ar trebui sa se ocupe mai intai de utilizarea potentialului masinii (in masina sa nu mai poata incapea nimic la momentul plecarii) pentru a fi optim.