1. Rezolvati urmatoarele exercitii:

a) Algoritmul A este O(n), algoritmul B este $o(n^2)$, algoritmul C este $\omega(n)$,iar algoritmul D este $\Theta(n^2)$. Ce relatii de ordine putem stabili intre acesti algoritmi din punct de vedere al complexitatii? Scurta justificare.

b) Algoritmul A are recurenta $T(n) = 10T(n/2) + n^3$, iar algoritmul B are recurenta $T(n) = aT(n/4) + 2 n^3$. care e valoarea intreaga maxima a lui a astfel incat B sa fie asimptotic mai rapid decat A ?

2. Pentru fiecare afirmatie de mai jos, scrieti valoarea de adevar impreuna cu o scurta justificare sau demonstratia completa:

a) $f(n) = \theta(\mathrm{n})$ si $\mathrm{g(n)} = \Omega(\mathrm{n})$ atunci $\mathrm{f(n)} * \mathrm{g(n)} = \Omega(\mathrm{n}^2)$
b) $f(n) = \theta(1)$ atunci $\mathrm{n}^{\mathrm{f(n)}} = O(\mathrm{n})$
c) $f(n) = \Omega(\mathrm{n})$ si $\mathrm{g(n)} = O(\mathrm{n}^2)$ atunci $\frac{\mathrm{f(n)}}{\mathrm{g(n)}} = O(\mathrm{n})$
d) $f(n) = O(n^2)\ si\ g(n) = O(n)\ atunci\ f\big(g(n)\big) = O(n^3)$
e) $f(n) = O(logn)$ atunci $2^{f(n)} = O(n)$
f) $f(n) = \Omega(\mathrm{n})$ atunci $2^{f(n)} = \Omega(n)$

3. Se da urmatoarea problema:

Fie n comutatoare ce corespund a n becuri (1..n), initial toate stinse. Se aplica urmatorii n pasi: la pasul k se modifica starea comutatoarelor din k in k incepand cu comutatorul 1 (1<=k<=n). Ne intereseaza cate becuri raman aprinse.

a) Scrieti pseudocodul algoritmului iterativ ce simuleaza cei n pasi descrisi anterior si determinati timpul sau de executie. Ce complexitate are acest algoritm (folositi notatia Theta) ?

b) Gasiti un algoritm cat mai eficient posibil pentru rezolvarea problemei. Indicatie: Ce pasi actioneaza asupra comutatorului k? Reduceti cat mai mult posibil complexitatea! Folositi de asemenea notatia Theta.

4. Se considera un tablou de n numere reale X[1..n]. Ne intereseaza sa raspundem la intrebari de genul: care este maximul subtabloului X[i..j]? Practic va trebui sa gasim un algoritm eficient per intrebare. Pentru aceasta vom imparti vectorul in n/k "bucati" de lungime k (eventual cu exceptia ultimei "bucati"). De exemplu pentru n = 8 si k = 3 vom avea [1..3], [4..6], [7, 8]. Pentru fiecare dintre aceste "bucati" vom calcula maximul si il vom retine intr-un vector max.

a) Scrieti in pseudocod un algoritm eficient ce gaseste maximul din subvectorul X[i..j]. Analizati complexitatea rezolvarii a m intrebari, luand in calcul si construirea vectorului max.

b) Cum ar trebui ales k astfel incat sa se obtina o complexitate cat mai mica ?

5. Intuiti o solutie si rezolvati prin metoda substitutiei recurenta: T(n) = T(n/2 + sqrt(n)) + n

6. Se da recurenta: T(n)= 2T(n/2) + n * log(n). Sa se rezolve prin metoda arborilor de recurenta si sa se confirme rezultatul obtinut prin metoda substitutiei.

7. Daca A si B sunt doua multimi recursive, demonstrati sa si A U B si A ∩ B sunt recursive.

8. Daca A si B sunt doua multimi recursiv-numarabile, demonstrati sa si A U B si A ∩ B sunt recursiv-numarabile.

9. Ce puteti spune despre decidabilitatea problemei urmatoare: "Se da o gramatica independenta de context, G(N, T, S, P). Este limbajul generat de G vid?" Limbajul generat de o gramatica este vid daca nu contine nici un cuvant.

Notare: Fiecare exercitiu valoreaza 1p => 9 * 1p + 1p (oficiu) = 10p