Examen la Analiza Algoritmilor
Set 2 – 04/02/2019

Timp de rezolvare: 90 de minute

1. (1,5p) Problema turnurilor din Hanoi este:
   a) NP-hard; b) NP-completă; c) semi-decidabilă
2. (1,5p) Dacă am găsit un algoritm determinist și polinomial pentru a rezolva problema q-sume (*subset sum*), care dintre următoarele afirmații este adevărată?
   a) Acest lucru este imposibil; b) P==NP; c) P!=NP
3. (1,5p) Fie problema q-sume (*subset sum*) modificată astfel încât vectorul de la intrare conține doar numerele `-1` sau `+1`. (notăm problema unary-q-sume):
   a) unary-q-sume∈NP-complete; b) unary-q-sume∈NP\(P U NP-complete); c) unary-q-sume∈P
4. (1,5p) Fie o problemă A∈NSPACE(`f(n)`), ∀ `f(n)`∈Ω(`log n`). Care din următoarele afirmații este adevărată:
   a) A∈NTIME(`f(n)`); b) A∈DSPACE(`f(n)**2`); c) A∈NPSPACE
5. (1,5p) Fie o structură de date pentru care se execută o secvență de `n` operații cu următoarele costuri: dacă `i==2**k`, costul operației este `10*i`; altfel, costul este `1`. Care este complexitatea amortizată per operație?
   a) Θ(n); b) Θ(log n); c) Θ(1)

6. (4p) Demonstrați următoarea afirmație: Fie A⊆N (mulțimea numerelor naturale), dacă A este recursiv-numărabilă și complementul său este recursiv-numărabil, atunci A este recursivă.

7. (4p) Pornind de la enunțul problemei k-acoperire cu vârfuri, prezentați un algoritm de aproximare pentru aceasta care să aibă factorul de aproximare constant. Demonstrați/justificați intuitiv acest factor de aproximare.

8. (3p) a) Cum se poate verifica satisfiabilitatea unei formule logice în forma normală disjunctivă? Dați un exemplu de formulă FND cu minim 3 variabile și 2 clauze conjunctive care să nu fie satisficabilă și justificați pe scurt.
   (3.5p) b) Prezentați schema de reducere polinomială SAT $\leq_P$ 3-SAT.

9. (5p) Scrieţi un algoritm nedeterminist pentru următoarea problemă și determinați complexitatea sa:
Se dă un graf neorientat `G(V,E)`, și două numere naturale `p` și `k` (`p ≤ k ≤ n`). Să se determine dacă există `k` submulțimi cu câte `1, 2, 3,…,` respectiv `k` noduri astfel încât orice muchie `(u,v)∈E` să aibă (cel puțin) unul din capete de minim `p` ori în cele `k` mulțimi. Submulțimile pot avea elemente (noduri) comune între ele.

10. (3p) Verificaţi posibilitatea aplicării teoremei master şi calculaţi limite asimptotice de complexitate pentru recurenţa: `T(n) = 8 * T(n/3) + n**2 * log(n)`

11. (10p) Se consideră tipurile de date LIST<N> și TREE<N>, pentru care avem definiți constructorii:

```
[] : → LIST<N>                        leaf : → TREE<N>
x:xs : N x LIST<N> → LIST<N>          node(t1, a, t2) : TREE<N> x N x TREE<N> → TREE<N>
```

și axiomele:

```
mirror: TREE<N> → TREE<N>
(M1) mirror(leaf) = leaf
(M2) mirror(node(t1, x, t2)) =
  node(mirror(t2), x, mirror(t1))
inorder: TREE<N> → LIST<N>
(I1) inorder(leaf) = []
(I2) inorder(node(t1, x, t2)) =
append(inorder(t1), x:inorder(t2))

reverse: LIST<N> → LIST<N>
(R1) reverse([]) = []
(R2) reverse(x:xs) = append(reverse(xs),
                            x:[])

append: LIST<N> x LIST<N> → LIST<N>
(A1) append([], l) = l
(A2) append(x:xs, l) = x:append(xs, l))
```

Știind că proprietatea `prop: append(append(l1, l2), l3)) == append(l1,append(l2, l3))` este adevărată ∀ `l1, l2, l3` ∈ `LIST<N>`, verificați prin inducție structurală următoarea proprietate:

```
P(t) = ( inorder(t) == reverse(inorder(mirror(t))) ) ∀ t ∈ TREE<N>
```

Total: 40p