Examen la Analiza Algoritmilor

1. (1,5p) Construirea unui heap binar se poate face optim in:
   a) $O(n^2)$; b) $O(n)$; c) $O(n \lg n)$
2. (1,5p) O problemă P dată este şi în NPTIME şi NP-completă este o afirmaţie care este:
   a) întotdeauna adevărată, b) niciodată adevărată, c) uneori adevărată
3. (1,5p) Care problemă din cele următoare are sigur o complexitate exponenţială:
   a) Ciclu hamiltonian; b) Problema opririi; c) Turnurile din Hanoi
4. (1,5p) Problema gasirii unui subgraf de acoperire minim k-arc conectat poate fi rezolvată de algoritmul Khuller-Vîșkin cu un factor de aproximare mai mic de:
   a) 3*k/2; b)(2*k-1)/k; c) k/2
5. (1,5p) GAP aparține mulțimii:
   a) LOGSPACE; b) SPACE((log n)**2); c) NP- completă

6. (4p) Definiți o mulţime recursiv enumerabilă care nu este recursivă, și demonstrați că așa este.

7. (5p) Enunțați teorema lui Cook și descrieți în câteva rânduri ideea demonstrației ei.

8. (2,5p) a) Construiți un subgraf de acoperire 3-arc-conectat optim al grafului complet cu 9 noduri
   (3p) b) Ce algoritm de aproximare cunoasteti care rezolva problema anterioara? Descrieti in cuvinte (nu in pseudocod) esenta acestui algoritm.

9. (5p) Scrieţi un algoritm nedeterminist pentru următoarea problemă.

Se consideră un graf neorientat G = (V, E), un întreg pozitiv k şi o funcţie de colorare a muchiilor fcol : E –> Culori, care asociază pentru fiecare muchie o culoare (deci colorarea muchiilor este știută). Se defineşte o partiţie a grafului G astfel: o submulţime E’ ⊆ E, astfel încât oricare două muchii din E’ nu au noduri comune şi toate muchiile din E’ au culori diferite. Să se decidă dacă există o partiție E’ a grafului G, cu |E’| ≥ k.

10. (3p) Verificaţi posibilitatea aplicării teoremei master şi calculaţi limite asimptotice de complexitate pentru recurenţa: $T(n) = 2*T(n/4) + sqrt(n)*log(n) + log^2(n)$

11. (10p) Se consideră tipul de date LIST<T>, pentru care avem definiți constructorii:

```
[] : → LIST<T>
[a] : T → LIST<T>
cons(a, l) : N x LIST<T> → LIST<T>
```

și axiomele:

```
size(l): LIST<T> → N
(S1)  size([]) = 0
(S2)  size([a]) = 1
(S3)  size(cons(a, x)) = 1 + size(x)
member(l, a): LIST<T> x T → BOOL
(M1)  member([], a) = false
(M2)  member([b], a) = (a==b)
(M3)  member(cons(b, x), a) = (a==b) || member(x, a)
filter(l, f): LIST<T> x FUNC1 → LIST<T>
(F1)  filter([], f) = []
(F2)  filter([a], f) = f(a) ? [a] : []
(F3)  filter(cons(a, x), f) =
              f(a) ? cons(a, filter(x, f)) : filter(x, f)
```

unde tipul FUNC1 reprezintă mulțimea funcțiilor cu antetul T → BOOLEAN (ex: `different_than_a(a1) = (a != a1)`). Verificați prin inducție structurală dacă următoarea proprietate este adevărată:

```
P(l) = ( member(l, a) → size(l) > size(filter(l, different_than_a)) )
              ∀ l ∈ LIST<T>
```

Total: 40p