Test 2 - Analiza Algoritmilor

1. (3p) Fie tipul de date *List*, cu elemente de tipul T, având constructorii de bază:

$$[] : \rightarrow List \quad \text{(Lista vidă)}$$
$$(:) : T \times List \rightarrow List \quad \text{(Adăugare)}$$

Constructorul intern este reprezentat prin caracterul "două puncte", ca în expresia $(3 : [])$.

Fie operatorii:

$insert(x, L)$ insearează o valoare într-o listă, înaintea primului element mai mare sau egal cu ea.

$isort(L)$ utilizează operatorul *insert* pentru a sorta prin inserţie o listă.

   (a) Definiţi **tipurile** şi **axiomele** corespunzătoare operatorilor *insert* şi *isort*.

   (b) Se consideră cunoscută următoarea proprietate: dacă lista L este sortată, atunci $insert(x, L)$ este o *permutare sortată* a listei $x : L$. Demonstraţi, prin **inducţie structurală**, că, pentru orice listă L, $isort(L)$ este o *permutare sortată* a acesteia.

2. (1p) Fie urmatoarea multime $M = \{n \in \mathbb{N} : n \text{ este par si } P_n(n)^1 \text{ se termina}\}$. Ce complexitate angelica are cel mai bun algoritm nedeterminist care poate **decide** multimea M ?

3. (1p) Fie Q_1, Q_2, Q_3 trei probleme de decizie astfel incat $Q_1 \leq_p Q_2 \leq_p Q_3$. Daca $Q_2 \in P$, ce putem spune despre complexitatea problemelor Q_1 si Q_3 ?

4. (1p) Fie Q o problema NP-completa. Exista algoritmi deterministi tractabili de rezolvare pentru Q ? Justificati.

5. (1p) Fie $A, B \subset \mathbb{N}$ si Q_A, Q_B generatoare pentru multimile A si B. Daca $B = \mathbb{N} \setminus A$, atunci exista un program P_A care poate decide multimea A ? Exista un astfel de program si pentru multimea B ? Justificati.

6. (3p) Fie urmatoarea problema ***Acoperire-Para***: *Se da un graf $G = (V, E)$ in care gradul[2] fiecarui nod este par, si o valoare k. Are G o acoperire de dimensiune k ?*

   (a) (1p) Demonstrati ca ***Acoperire-Para*** $\in$ NP;

   (b) (0.5p) Demonstrati ca pentru orice graf G, numarul nodurilor avand grad impar, este par;

   (c) (1.5p) Folosind proprietatea de mai sus, demonstrati ca ***Acoperire-Para*** $\in$ NP-complete. Pentru aceasta, reduceti ***k-Vertex-Cover*** la ***Acoperire-Para***;

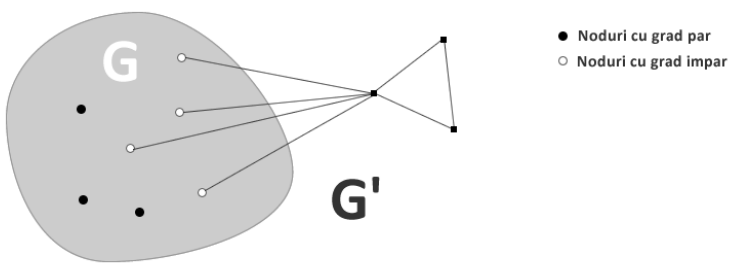

Figure 1: O posibila schema de reducere

[1] Prin P_n intelegem programul cu indicele n

[2] Gradul unui nod reprezinta numarul de muchii incidente acestuia.