Analiza Algoritmilor – Seria CA

Test 2

17.01.2013

1. (2p) Găsiți un invariant util pentru a explica funcționarea următorului algoritm și demonstrați corectitudinea acestuia:

```
Algoritm1(c[0..k], q)
        i = k
        n = c[i]
        WHILE (i > 0)
                i--
                n = q * n + c[i]
        RETURN n
```

Rezolvare:

Prelucrând câțiva pași din iterație:

i = k => n = c[k] (initializare)
i = k-1 => n = c[k-1] + q * c[k]
i = k-2 => n = c[k-2] + q * c[k-1] + q^2 * c[k]

se observa urmatorul invariant:

$$\mathrm{n} = \sum_{j=i}^{k} \mathrm{c[j]} * q^{j-i}$$

Demonstrarea corectitudinii

Inițializare

$\mathrm{i} = \mathrm{k};\ \mathrm{n} = \sum_{j=i}^{k} \mathrm{c[j]} * q^{j-i} = c[k]$. Adevărat, conform liniei 3 din pseudocod, înainte de intrarea în buclă.

Menținere

Înainte de iterația i:

$$\mathrm{n} = \sum_{j=i+1}^{k} \mathrm{c[j]} * q^{j-i-1}$$

În iterația i, n devine:

$$n = q * \sum_{j=i+1}^{k} c[j] * q^{j-i-1} + c[i] = \sum_{j=i+1}^{k} c[j] * q^{j-i} + c[i] =$$

$$= c[k] * q^{k-i} + c[k-1] * q^{k-1-i} + \cdots + c[i+1] * q^{i+1-i} + c[i] * q^{i-i} =$$

$$= \sum_{j=i}^{k} c[j] * q^{j-i}$$

Terminare

După ieșirea din buclă:

$i = 0; n = \sum_{j=0}^{k} c[j] * q^{j}$

2. (3p) Fie tipul de date *LIST* care reprezintă mulțimea listelor cu numere întregi definit prin următorii constructori de bază:

```
[ ] : → LIST
(x : l) : INT x LIST → LIST
```

Fie *FUN* mulțimea funcțiilor definite astfel: *INT x INT ⟶ INT*. Se definesc funcțiile *fold: FUN x INT x LIST ⟶ INT*, *len: LIST ⟶ INT*, *inc(x, y) = y+1*, precum și axiomele:

```
fold(f, z, [ ]) = z
fold(f, z, x : l) = f(x, fold(f, z, l))
len([ ]) = 0
len(x : l) = 1 + len(l)
```

Să se demonstreze prin inducție structurală că *P(l)* este adevarată pentru *∀l ∈ LIST*:

```
P(l) = ( len(l) = fold(inc, 0, l) )
```

Rezolvare:

Notăm:

```
(F1) fold(f, z, [ ]) = z
(F2) fold(f, z, x : l) = f(x, fold(f, z, l))
(L1) len([ ]) = 0
(L2) len(x : l) = 1 + len(l)
(INC) inc(x, y) = y+1
```

Cazul de bază

l = []. Vrem sa arătăm P([]) adevărată, unde:

```
P([]) = ( len([]) = fold(inc, 0, []) ).
```

Conform (L1), len([]) = 0, iar conform (F1), fold(inc, 0, []) = 0. Rezulta P([]) adevărată.

Ipoteza inductivă

l = xs. Presupunem P(xs) adevărată, unde:

P(xs) = (len(xs) = fold(inc, 0, xs)).

Pasul de inducție

l = x : xs. Vrem să demonstrăm P(x : xs) adevărată, unde:

P(x:xs) = (len(x:xs) = fold(inc, 0, x:xs)).

```
len(x:xs) = fold(inc, 0, x:xs) <=(L2)=>
1 + len(xs) = fold(inc, 0, x:xs) <=(F2)=>
1 + len(xs) = inc(x, fold(inc, 0, xs)) <=(INC)=>
1+len(xs) = fold(inc, 0, xs) +1 <=>
```

`len(xs) = fold(inc, 0, xs)` , care, conform ipotezei inductive, este adevărată.

Deci, P(x:xs) adevărată.

3. (2.5p) Construiți un algoritm nedeterminist pentru rezolvarea problemei *k-colorare*: "Fiind dat un graf neorientat *G(V, E)* și *k* culori diferite, se pot colora nodurile grafului folosind cele *k* culori astfel încât pentru *∀(u, v) ∈ E , color(u) != color(v)*?"

Rezolvare:

```
N_color(G(V, E), k)
    color[1..n] = new array(n)
    FOR (i=1..n)
        color[i] = CHOICE(1..k)

    FOREACH ((u,v)∈E)
        IF (color(u) == color(v))
            FAIL
    SUCCESS
```

4. (2.5p) Știind enunțul anterior pentru problema *k-colorare*, demonstrați că:

$$\texttt{3-colorare} \leq_p \texttt{4-colorare}$$

Rezolvare:

a) Pentru o construi o reducere polinomiala intre 3-colorare si 4-colorare, trebuie sa construim intai functia de transformare a datelor de intrare de la 3-colorare la 4-colorare.

F: G(V, E) → G'(V', E')

Cum ambele problem primesc un graf, gasirea acestei functii ar trebui sa fie destul de simpla. Mai mult, stim ca pentru orice graf G care este 3-colorabil, G' trebuie sa fie si el 4-colorabil (si pentru orice G care nu este 3-colorabil, nici G' nu trebuie sa fie 4-colorabil). Din cauza aceasta:

V' = V ∪ {vnew}
E' = E ∪ {(v, vnew) | ∀ v∈V}

Deci, in G' adaugam un nod nou, vnew, pe care il conectam cu muchii la toate celelalte varfuri din V.

Evident, F este o functie de transformare ce se poate implementa determinist si in timp polynomial.

b) Dupa ce am construit functia de transformare F, trebuie sa aratam ca 3-colorare(G) == 1 ⇔ 4-colorare(F(G)) == 1

Aratam intai ca 3-colorare(G) == 1 → 4-colorare(F(G)) == 1

Stim ca 3-colorare(G) == 1 => G este colorabil in 3 culori astfel incat oricare doua varfuri adiacente din G au culori diferite. Dar atunci G' va deveni colorabil in 4 culori astfel: pastram culorile asignate de 3-colorare ∀ v∈V, iar pe vnew il coloram in a 4a culoare pentru ca el este conectat cu toate varfurile din V asadar nu poate avea aceeasi culoare cu nici unul dintre ele (dar poate avea a 4a culoare) => G' = F(G) este 4-colorabil.

Acum aratam ca 4-colorare(F(G)) == 1 → 3-colorare(G) == 1

Stim ca G' = F(G) este 4-colorabil => Cum vnew este conectat cu toate varfurile din V, vnew nu poate fi colorat cu niciuna dintre culorile cu care sunt colorate varfurile din V => vnew foloseste o culoare, iar varfurile din V celalalte 3 culori din 4-colorare => G (care contine doar varfurile din V) este 3-colorabil.

5. (1.5p)
   a) Dați un exemplu de formulă în logica booleană, scrisă în forma normală conjunctivă, care să conțină minim 3 sub-clauze în conjuncție și minim 2 variabile per clauză și să nu fie satisfiabilă.
   b) Fie problemele de decizie *Q1 ∈ P* și *Q2 ∈ NP*. Pe baza lor construim o problemă nouă de decizie *Q3* astfel:

   ```
   Q3(i3) == (Q1(i1) && Q2(i2))
   ```

   Puteți presupune că datele de intrare *i1* pentru *Q1* și *i2* pentru *Q2* se pot contrui pe baza datelor de intrare *i3* ale lui *Q3*. Ce puteți spune despre o problemă oarecare *Q* despre care știm că $Q \leq_p Q3$?

Rezolvare:

a) Sunt multe solutii posibile. Una dintre ele este F(x1, x2) = (x1 v !x2) ^ (x1 v x2) ^ (!x1 v x2) ^ (!x1 v !x2). Se poate construe un tabel de valori pentru x1 si x2 daca vreti sa va convingeti ca F este intr-adevar nesatisfiabila.

b) Din modul in care este definite Q3, ea trebuie sa astepte atat terminarea lui Q1, cat si a lui Q2. Cum Q1 ∈ P si Q2 ∈ NP => Q3 ∈ NP .

In situatia aceasta, daca o problema oarecare $Q \leq_p Q3$, nu putem spune mai mult decat ca Q ∈ NP (poate fi in P, dar poate fi in in NP \ P).