Universitatea Alexandru Ioan Cuza, Iaşi   `Numele:`
Facultatea de Informatică   `Grupa:`

Proiectarea Algoritmilor - Test Scris (24 iunie 2019)

---

1. (18p) **Proiectare și analiză, baza.**
Se consideră următorul proces, notat cu P, care poate fi aplicat recursiv oricărui număr întreg pozitiv n:
   - dacă $n = 1$ nu se face nimic și procesul se oprește;
   - dacă n este divizibil prin 7, se împarte cu 7;
   - altfel se adugă 1 la n.

Se pune problema de a calcula numărul de 1 ce trebuie adăugați înainte ca procesul să se termine. De exemplu, dacă $n = 20$, atunci se adună 5 de 1. Notăm această problemă cu ONEBYONE.

(a) (4p) Să se formuleze ONEBYONE ca pereche (*input*,*output*). Se vor da formulări cât mai precise și riguroase.
(2p) *Input.* $n \in \mathbb{Z}$, $n > 0$;
Output. (1p) $f(n)$ = numărul de 1 ce trebuie adunați înainte ca procesul P să se termine.
(1p) $f(n) = \begin{cases} 0 & \text{, dacă } n = 1 \\ f(n/7) & \text{, dacă } n \text{ se împarte cu } 7 \\ f(n) + 1 & \text{, altfel.} \end{cases}$

(b) (4p) Să se scrie un algoritm determinist și nerecursiv care rezolvă ONEBYONE.

```
ONEBYONE(n) {
  f = 0;
  while (n != 1) {
    if (n % 7 == 0) n = n/7;
    else {
      n = n + 1;
      f = f + 1;
    }
  }
  return f;
}
```

Observație. Nu s-au punctat descrierile algoritmice recursive deoarece nu corespund specificației.

(c) (4p) Să se arate că algoritmul se termină totdeauna.
Fie N_i valoarea lui n exact înainte de a-l împărți la 7 a i-a oară, $i = 1, 2, \ldots$. Avem $n+7 > N_1 > N_2 > \cdots > 0$. De unde rezultă că secvența strict descrescătoare și pozitivă $N_1, N_2, \ldots$ este finită. Deoarece numărul de incrementări ale lui n (resp. f) între două împărțiri este cel mult 6, rezultă că bucla while se va termina după un număr finit de pași.
Observație. Formulările incomplete au primit 1 punct.

(d) (6p) Să se calculeze complexitatea în cazul cel mai nefavorabil.
(1p) *Dimensiunea unei instanțe.* n sau $m = \log_2 n$.

(1p) *Operații numărate.* incrementări ale lui n (sau f).

(2p) *Cazul cel mai nefavorabil.* Pentru un n dat:
– dacă s-a ales dimensiunea n, există o singură instanță cu această dimnesiune, care este și cea mai nefavorabilă.
– dacă s-a ales dimnsinea m, atunci cazul cel mai nefavorabil este un n cu proprietatea $m = \log_2 n$ și că $n \,\%, 7 = 1$, $((n + 6)/7) \,\%\, 7 = 1$, $\ldots$.

(2p) *Timpul pentru cazul cel mai nefavorabil.* $O(6 \cdot \log_7(n + 6))$ sau $O(6 \cdot m)$ $(m = O(\log_7(n + 6)))$.

Ciornă.

2. (18p) **Algoritmi probabiliști, complexitate medie**.
Se consideră următorul algoritm probabilist, descris informal:
randSearch(a, n)
2. repetă
  2.1 alege aleatoriu uniform un x din a;
  2.2 dacă $x = 1$ atunci întoarce poziția i pe care a fost găsit x;
3. până când găsește un x egal cu 1;
unde a este un tablou cu n elemente, dintre care $\frac{n}{2}$ sunt egale cu 1 și celelalte egale cu 0.

(a) (5p) Să se descrie în Alk algoritmul, ca o funcție.

```
randsearch(a, n) {
  i = random(n);
  x = a[i];
  while (x != 1) {
    i = random(n);
    x = a[i];
  }
  return i;
}
```

Observație. Utilizarea instrucțiunii `choose` în loc de `random` s-a penalizat cu 2 puncte.

(b) (5p) Care e probabilitatea ca algoritmul să întoarcă $i \in \{0, 1, \ldots, n-1\}$?
Probabilitatea ca algorimul să nu să se termine este $\lim_{j\to\infty} \dfrac{1}{2^j} = 0$ (3p).
Rezultă că algoritmul se termină și întoarce $i \in \{0, 1, \ldots, n-1\}$ cu probabilitatea 1. (2p).
Observație. S-au acordat 2 puncte dacă s-a menționat că probabilitatea este 1 cu justificare incompletă/imprecisă.

(c) (6p) Fie X variabila aleatorie care întoarce numărul de execuții ale instrucțiunii 2.1. Calculați care sunt valorile posibile ale lui X și probabilitățile aferente.
Valorile posibile sunt:
  i. $X = 1$, dacă corpul buclei while s-a executat de 0 ori (x este ales egal cu 1 după prima alegere probabilistă);
  $P(X = 1) = \frac{1}{2}$ (= probabilitatea să fie ales un 1 din a).
  ii. $X = 2$, dacă corpul buclei este executat exact o dată (x este 0 după prima alegere și egal cu 1 după a doua alegere probabilistă);
  $P(X = 2) = \frac{1}{2} \times \frac{1}{2}$ (= probabilitatea să fie ales un 0 din a înmulțită cu probabilitatea ca apoi să fie ales un 1 din a)
  iii. ...
  iv. $X = j$, dacă corpul buclei este executat exact de $j - 1$ ori (la primele $j - 2$ alegeri este ales 0, apoi la a $j - 1$-a alegere 1);
  $P(X = j) = \frac{1}{2^{j-1}} \times \frac{1}{2} = \frac{1}{2^j}$
  v. ...

Observație. S-au acordat 2 puncte dacă s-a menționat doar un caz particular sau numai valorile (fără probabilități).

(d) (2p) Care ese timpul mediu de execuție a algoritmului? $M(X) = \sum_{j>0} j \cdot P(X = j) = \sum_{j>0} \dfrac{j}{2^j}$.

Ciornă.

3. (18p) **Programare dinamică**.
După o reformă financiară, în România dispar bancnotele actuale și apar bancnote de $1, 5, 9, 14$ și 17 RON. Scopul este să achităm o suma S cu cât mai puține bancnote.
De exemplu, suma de 20 RON poate fi achitată cu 3 bancnote: $20 = 14 + 1 + 5$.

(a) (4p) Să se formuleze problema de mai sus ca pereche (*input*,*output*). Se vor da formulări cât mai precise şi riguroase.
(2p) *Input*: $S \in \mathbb{N}$ - suma de achitat
(2p) *Output*: cel mai mic număr natural l a.î. $\exists y_1, \ldots, y_l \in \{1, 5, 9, 14, 17\}$ cu proprietatea că $y_1 + \ldots + y_l = S$.

(b) (4p) Să se găsească un contraexemplu (altul decât $S = 20$), care arată că strategia de a alege la fiecare pas o bancnotă cât mai mare nu conduce la soluția optimă.
$S = 19$.
Strategia produce soluția $19 = 17 + 1 + 1$ (trei bancnote).
Optim ar fi $19 = 14 + 5$.
Observație. S-au acordat 1 punct pentru contra-exemplu și 3 puncte pentru justificare.

(c) (4p) Fie subproblemele notate $d(x) =$ numărul minim de bancnote necesare pentru a achita suma $0 \le x \le S$. Calculați $d(0)$.
$d(0) = 0$. E nevoie de 0 bancnote pentru a achita suma 0.
Observație. S-au acordat 3 puncte pentru expresie și 1 punct pentru justificare. O justificare greșită s-a penalizat cu 1 punct.

(d) (6p) Pentru $x \ge 1$, exprimați $d(x)$ în funcție de valorile $d(y)$ $(0 \le y < x)$.
$d(x) = 1 + min\{d(x-1), d(x-5), d(x-9), d(x-14), d(x-17)\}$.
Deoarece e posibil ca $d(x-1) < 0$ sau $d(x-5) < 0$ sau $\ldots$, funcția d este extinsă astfel încât $d(-1) = d(-2) = \ldots = \infty$. Neconsiderarea acestor cazuri s-a penalizat cu 1 punct.

Ciornă.

4. (18p) **Backtracking**.
Context: proiectarea unui algoritm de tip backtracking pentru problema `CLIQUE`. Presupunem că nodurile grafului sunt numerotate de la 0 la $n-1$. Vom reprezenta o soluție a problemei sub forma unui vector $v[0..n-1]$, unde:

$$v[i] = \begin{cases} 1 & \text{, dacă nodul } i \text{ aparține mulțimii } V' \\ 0 & \text{, altfel.} \end{cases}$$

Cerințe:

(a) (4p) Definiți problema CLIQUE ca pereche input-output.
(2p) *Input*: un graf $G = (V, E)$, un număr întreg $k > 0$. Presupunem că V este $\{0, 1, \ldots, n-1\}$.
(2p) *Output*: *true* – dacă există o clică cu cel puțin k vârfuri, i.e., există $V' \subseteq V$ a.î. $|V'| \geq k$ și $\forall u, v \in V' : u \neq v \implies \{u, v\} \in E$ (există muchie între oricare două vârfuri din V').

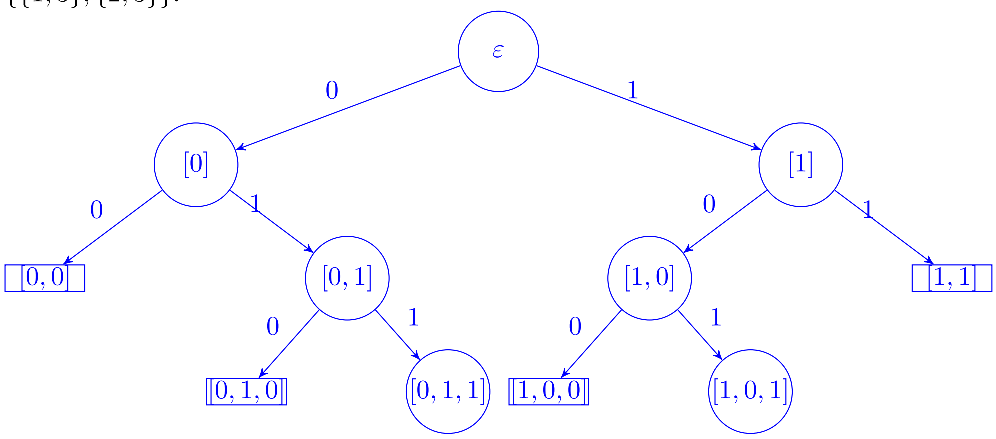
(b) (4p) Ce este o *soluție parțială* în contextul problemei de mai sus?
O soluție parțială este dată de un prefix al unei soluții complete, adică de un vector $v[0..i-1]$, unde $0 \leq i \leq n$ și $v[j] \in \{0, 1\}$ $(0 \leq j \leq i-1)$.

(c) (4p) Ce este o *soluție parțială viabilă* în contextul problemei de mai sus?
O soluție parțială $v[0..i-1]$ este viabilă dacă:

i. pentru orice $0 \leq x, y \leq i-1$ a.î. $v[x] = v[y] = 1$, avem că există muchie între nodurile x și y;
ii. numărul de 1 din vectorul $v[0..i-1]$ este $\geq k - (n - i)$ (trebuie să existe șansa de a selecta cel puțin k noduri).

(d) (4p) Să se definească *succesorii* unei soluții parțiale (în contextul problemei de mai sus).
Dacă $i < n$, succesorii soluției parțiale $v[0..i-1]$ sunt vectorii $v_0[0..i]$ și $v_1[0..i]$ a.î. $v_0[0..i-1] = v_1[0..i-1] = v[0..i-1]$ și $v_0[i] = 0$, $v_1[i] = 1$.

(e) (2p) Să se reprezinte arborele de căutare $k = 2$ și pentru graful $G = (V, E)$, $V = \{1, 2, 3\}$, $E = \{\{1, 3\}, \{2, 3\}\}$.



Ciornă.