Universitatea Al.I.Cuza Iasi   **Nume:**
Facultatea de Informatică   **Grupa:**

Proiectarea algoritmilor – Test scris 06.04.2015, A

Observații:
1. Nu este permisă consultarea bibliografiei.
2. Toate întrebările sunt obligatorii.
3. Fiecare întrebare/item este notată cu un număr de puncte indicat în paranteză. Descrieți conceptele utilizate în răspunsuri.
4. Algorimii vor fi descriși în limbajul Alk(cel utilizat la curs). Se admit extensii cu sintaxă inspirată din C++ (de exemplu, for, do-while, repeat-until, etc.). Pentru structurile de date utilizate se vor preciza operațiile (fără implementare daca nu se cere explicit) și complexitățile timp și spațiu ale acestora.
5. Nu este permisă utilizarea de foi suplimentare.
6. Timp de răspuns: 1 oră.

1. Se considera problema testării dacă un tablou dat de numere întregi este ordonat strict crescător, notată cu ISINC.
a) [1.5p] Să se formuleze problema ISINC ca pereche (input, output).
b) [1.5p] Să se descrie un algoritm care rezolvă ISINC.
c) [1.5p] Să se arate ca algoritmul de la b) rezolvă corect problema descrisă la a). Se va menționa care este invariantul instrucțiunii repetive.
d) [1.5p] Să se precizeze care este cazul cel mai nefavorabil pentru timpul de execuție.
e) [2p] Se consideră ca dimensiune a intrării $n = m \log M$, unde m este numărul de elemente din tablou, M este numărul maxim memorat în tablou. Se presupune că orice operație asupra elementelor din tablou se execută în timpul $\log M$. Să se determine timpul de execuție pentru cazul cel mai nefavorabil.

Răspuns.
a)
Input: a = (a[0],...,a[m-1]), a[i] = a_i in Z
Output: *true* daca $a_0 < \ldots < a_{m-1}$, false in caz contrar
b)
```
isinc(a, m) {
  i = 0;
  while ( i < m-1) {
    if (a[i] >= a[i+1]) return false;
    i = i + 1;
}
  return true;
}
```
c)
Invariantul lui for: a[0..i] este ordonata strict crescator si i <= m-1. Pentru i = 0 este evidente adevarat. Este pastrat de bucla while: incrementarea lui i se face numai daca a[i] < a[i+1]. Dupa executia lui while are loc invariantul si i >= m-1, care implica i = m-1 si a[0..m-1] ordonat strict crecator.
d)
Cazul cel mai nefavorabil este cand tabloul este deja ordonat crescator, cu exceptia eventual a ultimelor doua.
e)
Se numara numai operatiile cu elemente din tablou.
O bucla while executa o singura operatie, comparatia, in timpul log M.
Numarul de iteratii in vazul cel mai nefavorabil: m-1.
Rezulta timpul total = (m-1) log M = n – log M = O(n).

2. Se consideră mulțimea S cu următoarele puncte în plan:

$$P[0]=(1,1), P[1]=(2,6), P[2]=(3,3), P[3]=(4,2), P[4]=(5,4), P[5]=(6,5)$$

Scopul exercițiului este de a descrie cum se aplică algoritmul lui Graham pentru a determina înfășuratoarea convexă.

a) [2p] Descrieți cum se determina un punct interior $CH(S)$.

b) [3p] Scrieți lista punctelor sortate după coordonate polare.

c) [4p] Descrieți cum este aplicată scanarea Graham pentru a determina $CH(S)$.

d) [2p] Descrieți invariantul menținut de scanarea Graham și exemplificați acesta pentru pașii descriși la c).

Răspuns.

a)

Se poate lua centrul de greutate G = (xG, yG), unde xG si yG sunt mediile aritmetice al absciselor respectiv ordonatelor:
xG = (1+2+3+4+5+6)/ 6 = 3.5; yG = (1+6+3+2+4+5) = 3.5

b)

Trebuia desente punctele P[i] si G pe un sistem de coordonate carteziene, si trasate apoi razele polare (detalii pe tabla, necessita desen).
Rezulta Lista circulara L = (P[4], P[5], P[1], P[2], P[0], P[3]).

c)

Se parcurge L si se calculeaza ccw pentru trei puncte succesive:
ccw(P[4], P[5], P[1]) > 0 (sens invers Arcelor de ceasornic), ok.
ccw(P[5], P[1], P[2]) > 0 ok.
ccw(P[1], P[2], P[0]) < 0 => sensul arcelor de ceasornic, se elimina P[2].
ccw(P[5], P[1], P[0]) > 0 ok.
ccw(P[1], P[0], P[3]) > 0 ok.
ccw(P[0], P[3], P[4]) > 0 ok.
ccw(P[3], P[4], P[5]) < 0 => se elimina P[4].
ccw(P[0], P[3], P[5]) > 0 ok.
ccw(P[3], P[5], P[1]) > 0 ok.
La pasul urmator se ajunge in punctul de start P[5] (P[4] a fost eliminat si P[5] a devenit punct de start).
Infasuratoarea convexa este data de poligonul convex CH = (P[5], P[1], P[0], P[3]).

d)

Lista punctelor din L pana la varful curent formeaza un poligon convex:
(P[4], P[5], P[1]) ; (P[4], P[5], P[1], P[2]); (P[4], P[5], P[1]); (P[4], P[5], P[1], P[0]); (P[5], P[1], P[0]); (P[5], P[1], P[0], P[3]).

3. a) [2p] Descrieți problemele NEAREST NEIGHBOR și ALL NEAREST NEIGHBOR.
b) [3p] Să se arate că NEAREST NEIGHBOR se reduce la ALL NEAREST NEIGHBOR. Să se precizeze ce tip de reducere este (Turing sau Karp).
c) [3p] Să se descrie relațiile dintre complexitățile celor două probleme, pe baza reducerii de la b).

Răspuns.

a)

NEAREST NEIGHBOR
Intrare: O multime S cu n puncte, un punct P, toate in plan.
Iesire: Cel mai apropiat punct de P din S.

ALL NEAREST NEIGHBOR
Intrare: O multime S cu n puncte in plan.
Iesire: Cel mai apropiat vecin din S pentru fiecare punct din S.

b)

1. instanta (S, P) se transforma in instanta S.
2. Se apeleza algoritmul care calculeaza lista L cu L[i] cel mai apropiat de S[i].
3. Intoarce L[k] pentru k cu proprietatea ca S[k] = P.

Reducerea se face in O(n) (daca P este precizat direct prin K, atunci in O(1).
Este o reducere Karp deoarece se apeleza o singura data algoritmul pentru ALL NEAREST NEIGHBOR (s-a punctat ca fiind corect si raspunsul cu reducere Turing datorita post-procesarii).

c)

Daca ALL NEAREST NEIGHBOR are complexitatea O(f(n)) atunci NEAREST NEIGHBOR are complexitaea O(f(n) + n) (sau O(f(N) daca reducerea e in O(1)).
Daca NEAREST NEIGHBOR are complexitatea Ω(f(n)) atunci ALL NEAREST NEIGHBOR are complexitatea Ω(f(n) - n) (sau Ω(f(n)) daca reducerea e in O(1)).