PRG - ETSInf. TEORIA. Curs 2015-16. Parcial 1.
11 d'abril de 2016. Duració: 2 hores.

1. 4 punts Donat un array `a` de `int` ordenat ascendentment i un enter `x`, escriu un mètode **recursiu** que calcule la suma dels elements menors que `x` que hi ha a `a`. Per exemple, per `a` = {`2,2,7,8,10,10,12,23,34`} i `x`=10 ha de tornar 19; per al mateix array i `x`=1 ha de tornar 0. El mètode ha d'evitar fer comparacions supèrflues d'elements de `a` amb `x`.

   Es demana:

   a) (0.75 punts) Perfil del mètode, amb els paràmetres adequats per a resoldre recursivament el problema.
   b) (1.25 punts) Cas base i cas general.
   c) (1.5 punts) Implementació en Java.
   d) (0.5 punts) Crida inicial perquè es realitze el càlcul sobre tot l'array.

Solució:

a) Una possible solució consisteix en definir un mètode amb el següent perfil:

```
/** Precondició: a ordenat ascendentment i 0 <= pos. */
public static int sumaMenors(int[] a, int pos, int x)
```

de manera que torne la suma dels elements de `a[pos..a.length-1]` menors que `x`, sent 0≤`pos`.

b)
- Cas base, `pos`≥`a.length`: Subarray buit. Torna 0.
- Cas general, `pos<a.length`: Subarray d'un o més elements. Si `a[pos]` es menor que `x`, torna la suma de `a[pos]` més la suma dels elements menors que `x` en `a[pos+1..a.length-1]`; però si `a[pos]` ≥ `x`, i en estar l'array ordenat ascendentment, es retorna 0 sense necessitat de fer més comprovacions.

c)

```
/** Torna la suma dels elements de a [pos..a.length-1] menors que x.
 *  Precondició: a ordenat ascendentment i 0 <= pos. */
public static int sumaMenors(int[] a, int pos, int x) {
    if (pos < a.length) {
        if (a[pos] >= x) { return 0; }
        else { return a[pos] + sumaMenors(a, pos + 1, x); }
    } else { return 0; }
}
```

d) Per un array `a`, la crida `sumaMenors(a, 0, x)` resol el problema enunciat.

2. 3 punts El següent mètode comprova si un array d'enters es troba ordenat de manera ascendent entre les posicions `ini` i `fi` inclusivament:

```
/** Torna true si a[ini..fi] està ordenat ascendentment,
 *  false en cas contrari. */
public static boolean ordenat(int[] a, int ini, int fi) {
    if (ini >= fi) { return true; }
    else {
        if (a[ini] > a[ini + 1] || a[fi] < a[fi - 1]) { return false; }
        else { return ordenat(a, ini + 1, fi - 1); }
    }
}
```

   Es demana:

   a) (0.25 punts) Indiqueu quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.5 punts) Indiqueu si existeixen diferents instàncies significatives per al cost temporal de l'algorisme, identificant-les si és el cas.

c) (1.5 punts) Doneu la relació de recurrència per al cost, resolent-la per substitució, distingint el cost de les instàncies més significatives en cas d'haver-les.

d) (0.75 punts) Expresseu el resultat anterior utilitzant notació asimptòtica.

Solució:

a) La talla del problema és el nombre d'elements de l'array `a` en consideració, i l'expressió que la representa és `fi-ini+1`. D'ara endavant, anomenarem a aquest valor `n`. És a dir, `n=fi-ini+1`.

b) Es tracta d'un problema de cerca (ascendent i descendentment, de forma simultània) i, per tant, per a una mateixa talla presenta instàncies diferents. El cas millor és quan la primera o l'última parella d'elements en consideració no estan ordenades. El cas pitjor és quan `a[ini..fi]` està ordenat .

c) Per obtenir el cost del mètode, estudiem cadascuna de les dues instàncies significatives :

- Cas millor: $T^m(n) = 1$ *p.p.*
- Cas pitjor:

$$T^p(n) = \begin{cases} T^p(n-2)+1 & \text{si } n > 1 \\ 1 & \text{si } n \leq 1 \end{cases}$$

expressat en passos de programa (p.p.).
Resolent per substitució: $T^p(n) = T^p(n-2)+1 = T^p(n-4)+2 = \ldots = T^p(n-2\cdot i)+i$. S'arriba al cas base, talla 0 o 1, per a $i = n/2$. Amb el que

$$T^p(n) = 1 + \frac{n}{2} \; p.p.$$

d) En notació asimptòtica: $T^m(n) \in \Theta(1)$ y $T^p(n) \in \Theta(n)$. Per tant, $T(n) \in \Omega(1)$ y $T(n) \in O(n)$, és a dir, el cost temporal depèn com a molt linealment de la talla del problema.

3. $\boxed{\text{3 punts}}$ El següent mètode transposa una matriu quadrada:

```
/** Canvia la matriu m a la seua transposada.
 *  Precondició: m es una matriu quadrada. */
public static void transposada(int[][] m) {
    for (int i = 0; i < m.length; i++) {
        for (int j = 0; j < i; j++) {
            int aux = m[i][j];
            m[i][j] = m[j][i];
            m[j][i] = aux;
        }
    }
}
```

Es demana:

a) (0.25 punts) Indiqueu quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.5 punts) Indiqueu si existeixen diferents instàncies significatives per al cost temporal de l'algorisme, identificant-les si és el cas.

c) (1.5 punts) Escolliu una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella obtenir una expressió matemàtica, el més precisa possible, del cost temporal del mètode, distingint el cost de les instàncies més significatives en cas d'haver-les .

d) (0.75 punts) Expresseu el resultat anterior utilitzant notació asimptòtica.

Solució:

a) La talla del problema és la dimensió de la matriu `m`, és a dir, `m.length`. D'ara endavant, anomenarem a aquest valor `n`. És a dir, `n`=`m.length`.

b) No existeixen diferents instàncies.

c) Triant com a unitat de mesura el pas de programa, es té:

$$T(n) = 1 + \sum_{i=0}^{n-1}(1 + \sum_{j=0}^{i-1} 1) = 1 + \sum_{i=0}^{n-1}(1+i) = 1 + \frac{(1+n)\cdot n}{2} = 1 + \frac{n}{2} + \frac{n^2}{2}\ p.p.$$

Si es pren com a instrucció crítica l'avaluació de la guarda `j < i`, i es compta com a mesura del cost el nombre de vegades que s'executa aquesta guarda, s'arriba a l'expressió :

$$T(n) = \sum_{i=0}^{n-1}\sum_{j=0}^{i} 1 = \sum_{i=0}^{n-1}(1+i) = \frac{n}{2} + \frac{n^2}{2}\ i.c.$$

que és com l'expressió anterior, tot i que menyspreant el terme de menor ordre.

d) En notació asimptòtica: $T(n) \in \Theta(n^2)$.