PRG - ETSInf. TEORIA. Curs 2014-15. Parcial 1.
27 d'abril de 2015. Duració: 2 hores.

1. 3 punts Donat un array `a` de `int` amb almenys un element, escriure un mètode **recursiu** que comprove si tots els valors de l'array són parells i estan emmagatzemats en ordre creixent.

   Es demana:

   a) (0.5 punts) Perfil del mètode, amb els paràmetres adequats per tal de resoldre recursivament el problema.

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

   ```
   public static boolean parellsICreixents(int[] a, int pos)
   ```

   sent `a.length`>0 i 0≤`pos`<`a.length`.

   b) (1.2 punts) Cas base i cas general.

   Solució:

   - Cas base, `pos==a.length-1`: Subarray d'un element. Retorna `true` si `a[pos]` és parell i `false` en cas contrari.
   - Cas general, `pos<a.length-1`: Subarray de més d'un element. Retorna `true` sii `a[pos]` és parell i `a[pos] <= a[pos+1]` i els elements del subarray `a[pos+1..a.length-1]` són parells i estan ordenats creixentment. En un altre cas, retorna `false`.

   c) (1 punt) Implementació en Java.

   Solució:

   ```
   /** Comprova si tots els elements de l'array a són parells
    *  i estan ordenats creixentment.
    *  Precondició: a.length>0 i 0<=pos<a.length.
    */
   public static boolean parellsICreixents(int[] a, int pos) {
       if (pos == a.length-1) return a[pos]%2 == 0;
       else return a[pos]%2 == 0 && a[pos] <= a[pos+1] && parellsICreixents(a, pos+1);
   }
   ```

   d) (0.3 punts) Crida inicial perquè es verifique la propietat sobre tot l'array.

   Solució: Per a un array `a`, la crida `parellsICreixents(a,0)` resol el problema de l'enunciat.

2. 7 punts Per tal de determinar quants elements d'un array `a` són menors que un valor donat `x`, es proposen les dues solucions següents en Java, on la primera d'elles suposa que l'array està ordenat ascendentment.

   - Solució 1

   ```
   /** Torna el nombre d'elements de l'array a menors que x
    *  Precondició: a està ordenat ascendentment
    */
   public static int comptarMenorsX1(int[] a, int x) {
       int i = a.length - 1;
       while (i >= 0 && a[i] >= x) i--;
       return i + 1;
   }
   ```

- Solució 2

```
/** Torna el nombre d'elements de l'array a menors que x
 *  Precondició: 0 <= pos <= a.length
 *  Crida inicial: int res = comptarMenorsX2(a, x, 0);
 */
public static int comptarMenorsX2(int[] a, int x, int pos) {
    if (pos == a.length) return 0;
    else if (a[pos] < x) return 1 + comptarMenorsX2(a, x, pos + 1);
    else return comptarMenorsX2(a, x, pos + 1);
}
```

Es demana:

i. (3 punts/proposta) Per a cada solució proposada:

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

b) (*0.5 punts*) Indicar si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identificar-les si és el cas.

c) (*1.5 punts*) En el cas del mètode iteratiu, triar una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella, obtindre una expressió matemàtica, el més precisa possible, del cost temporal del programa, a nivell global o en les instàncies més significatives si n'hi ha.

En el cas del mètode recursiu, escriure l'equació de recurrència del cost temporal en funció de la talla per a cada un dels casos si n'hi ha diversos, o una única equació si només hi hagués un cas. Cal resoldre-la per substitució.

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

Solució:

Anàlisi per al mètode de la Solució 1:

a) La grandària o talla del problema és el nombre d'elements de l'array `a` i l'expressió que la representa és `a.length`. D'ara endavant, anomenarem a aquest número `n`. Açò és, `n = a.length`.

b) Es tracta d'un problema de cerca (iterativa descendent) i, per tant, per a una mateixa talla sí que presenta instàncies distintes. Com l'array `a` està ordenat ascendentment, el cas millor es dóna quan en l'última posició de l'array trobem un valor menor que `x` (`a[a.length-1]<x`), açò és, tots els elements de l'array són menors que `x`. El cas pitjor ocorre quan tots els elements de l'array són majors o iguals que `x` ($\forall$`i`, `0`$\leq$`i`$<$`a.length`, `a[i]`$\geq$`x`).

c) Triant com unitat de mesura el pas de programa, i considerant una passada del bucle com un pas de programa i la resta d'operacions com un únic pas de programa, la funció de cost temporal en el cas millor serà $T^m(n) = 1$ *p.p.* i en el cas pitjor $T^p(n) = 1 + \sum_{i=0}^{n-1} 1 = n + 1$ *p.p.*

Triant com unitat de mesura la instrucció crítica, l'única instrucció que podem considerar com crítica és la guarda del bucle: `i >= 0 && a[i] >= x` (s'executa sempre una vegada més que qualsevol de les del cos del bucle). En el cas millor només s'executarà una vegada. En el cas pitjor es repetirà tantes vegades com elements tinga l'array més una (quan es fa falsa). La funció de cost temporal, considerant la instrucció crítica de cost unitari, en el cas millor serà $T^m(n) = 1$ *i.c.* i en el cas pitjor $T^p(n) = \sum_{i=0}^{n} 1 = n + 1$ *i.c.*

d) En notació asimptòtica: $T^m(n) \in \Theta(1)$ i $T^p(n) \in \Theta(n)$. Per tant, $T(n) \in \Omega(1)$ i $T(n) \in O(n)$, és a dir, el cost temporal està fitat inferiorment per una funció constant i superiorment per una funció lineal amb la talla del problema.

Anàlisi per al mètode de la Solució 2:

a) La grandària o talla del problema és el nombre d'elements de l'array `a` en consideració i l'expressió que la representa és `a.length - pos`. D'ara endavant, anomenarem a aquest número `n`. Açò és, `n = a.length - pos`.

b) Es tracta d'un problema de recorregut (recursiu ascendent) i, per tant, per a una mateixa talla no presenta instàncies distintes.

c) Per obtenir el cost del mètode, plantegem l'equació de recurrència:

$T(n) = \begin{cases} T(n-1)+k & \text{si } n > 0 \\ k' & \text{si } n = 0 \end{cases}$ sent k i k' constants positives, en alguna unitat de temps.

Resolent per substitució: $T(n) = T(n-1)+k = T(n-2)+2k = \ldots = T(n-i)+ik$. En arribar al cas base, per a talla 0, $n-i=0 \rightarrow n=i$. Amb el que $T(n) = k' + nk$.

d) En notació asimptòtica: $T(n) \in \Theta(n)$, és a dir, el cost temporal depèn linealment de la talla del problema.

ii. (1 punt) Tenint en compte que l'algorisme d'ordenació per *inserció directa* vist en classe té un cost lineal en el millor cas i quadràtic en el pitjor, i que per a la primera solució caldria ordenar l'array prèviament, comparar el cost temporal total de comptar els valors menors que `x` en un array `a` no necessariament ordenat, quan es resol el problema mijançant els següents algorismes:

- Algorisme 1: Primer ordenar `a` per inserció directa i després aplicar el mètode `comptarMenorsX1(int[], int)`.
- Algorisme 2: Aplicar directament a l'array `a` el mètode `comptarMenorsX2(int[], int, int)`.

Solució:

La resolució del problema aplicant l'Algorisme 1 tindrà les següents fites de complexitat:

- En el cas millor, l'array està ordenat ascendentment i tots els seus elements són menors que `x`. Aplicar l'ordenació per inserció directa tindrà un cost $\Theta(n)$ i comptar el nombre d'elements menors que `x` serà $\Theta(1)$. Per tant, el cost en el cas millor serà lineal amb la talla del problema.
- En el cas pitjor, els elements de l'array estan ordenats descendentment i tots ells són majors o iguals que `x`. El cost de l'ordenació per inserció directa serà $\Theta(n^2)$ i comptar el nombre d'elements menors que `x` serà $\Theta(n)$. Per tant, el cost en el cas pitjor serà quadràtic amb la talla del problema.

La resolució del problema aplicant l'Algorisme 2, siga quin siga l'array `a`, tindrà sempre un cost $\Theta(n)$. Per tant, podem concloure que l'Algorisme 2 és més eficient quan el problema no està restringit a arrays ordenats.