PRG - ETSInf. TEORIA. Curs 2017-18. Parcial 1.
16 d'abril de 2018. Durada: 2 hores.

Nota: L'examen s'avalua sobre 10 punts, però el seu pes específic en la nota final de PRG és **de 3 punts**.

1. [4 punts] Donats un array `v` de `String` i un nombre natural `n`, escriu un mètode **recursiu** que retorne quants elements de l'array `v` tenen una longitud `n`.

   Per exemple, si l'array `v` continguera {`"barco"`, `"autobus"`, `"tren"`, `"moto"`, `"bici"`}, i `n` fóra 4, llavors el mètode retornaria 3. Si l'array fóra el mateix, i `n` fóra 5, llavors el mètode retornaria 1. Per al mateix array `v`, i `n` igual a 3, el mètode retornaria 0.

   Es demana:

   a) (0.75 punts) Perfil del mètode, amb els paràmetres adequats per a resoldre recursivament el problema, i precondició relativa a aquests paràmetres.

   b) (1.25 punts) Cas base i cas general.

   c) (1.50 punts) Implementació en Java.

   d) (0.50 punts) Crida inicial perquè es realitze el càlcul sobre tot l'array.

   Solució:

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

   ```
   /** Precondició: n >= 0 && 0 <= pos <= v.length */
   public static int comptarPar(String[] v, int n, int pos)
   ```

   de manera que retorne quants elements de longitud `n` hi ha a l'array `v[pos..v.length - 1]`, sent $0 \leq$ `pos` $\leq$ `v.length`.

   b)
   - Cas base, `pos = v.length`: Subarray buit. Retorna 0.
   - Cas general, `pos` $<$ `v.length`: Subarray amb un o més elements. Es calcula el nombre d'elements de longitud `n` al subarray `v[pos + 1..v.length - 1]` i a aquest nombre se li suma 1 si la longitud de `v[pos]` és `n`. Es retorna aquest nombre.

   c)
   ```
   public static int comptarPar(String[] v, int n, int pos) {
       if (pos == v.length) { return 0; }
       else {
           if (v[pos].length() == n) { return 1 + comptarPar(v, n, pos + 1); }
           else { return comptarPar(v, n, pos + 1); }
       }
   }
   ```

   d) Per a un array `v` i un nombre `n`, la crida `comptarPar(v, n, 0)` resol el problema enunciat.

2. [3 punts] Donada una matriu quadrada `m` d'enters, els components de la qual són tots positius, el següent mètode comprova per a quines files la suma de tots els seus components és menor que `limit`. Per a cadascuna d'aquestes files, escriu el valor de la suma dels seus components.

   ```
   /** Precondició: m és una matriu quadrada, i els seus elements
    *  són tots positius. El valor de limit és > 0. */
   public static void sumes(int[][] m, int limit) {
       int n = m.length;
       for (int i = 0; i < n; i++) {
           int suma = m[i][0];
           int j = 1;
           while (j < n && suma < limit) {
               suma += m[i][j];
   ```

```
            j++;
        }
        if (suma < limit) {
            System.out.println("Fila " + i + ": " + suma);
        }
    }
}
```

Es demana:

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

b) (0.75 punts) Indica, i justifica, si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identifica-les si és el cas.

c) (1.50 punts) Tria una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella obtín 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) (0.50 punts) Expressa el resultat anterior utilitzant notació asimptòtica.

Solució:

a) La talla del problema és la dimensió de la matriu `m` i l'expressió que la representa és `m.length`. D'ara endavant, anomenarem a aquest nombre n. Açò és, $n =$ `m.length`.

b) Sí que existeixen diferents instàncies. El cas millor es dóna quan totes les files tenen en el component 0 un valor major o igual que `limit`. El cas pitjor es dóna quan per a totes les files es compleix que la suma de tots els components de la fila no arriba a `limit`.

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

- En el cas millor: $T^m(n) = 1 + \sum_{i=0}^{n-1} 1 = 1 + n$ *p.p.*
- En el cas pitjor: $T^p(n) = 1 + \sum_{i=0}^{n-1}(1 + \sum_{j=1}^{n-1} 1) = 1 + \sum_{i=0}^{n-1} n = 1 + n^2$ *p.p.*

Si triem com a unitat de mesura la instrucció crítica i considerant com a tal, per exemple, l'avaluació de la guarda `j < n && suma < limit` (de cost unitari), es té:

- En el cas millor $T^m(n) = \sum_{i=0}^{n-1} 1 = n$ *i.c.*
- En el cas pitjor: $T^p(n) = \sum_{i=0}^{n-1} \sum_{j=1}^{n} 1 = \sum_{i=0}^{n-1} n = n^2$ *i.c.*

d) En notació asimptòtica: $T^m(n) \in \Theta(n)$ i $T^p(n) \in \Theta(n^2)$. Per tant, $T(n) \in \Omega(n)$ i $T(n) \in O(n^2)$.

3. [3 punts] Es desitja calcular el cost del següent mètode recursiu:

```
public static double testMethod(double[] v, int left, int right) {
    if (left > right) { return 1.0; }
    else {
        int middle = (left + right) / 2;
        return v[middle]
            * testMethod(v, left, middle - 1)
            * testMethod(v, middle + 1, right);
    }
}
```

Es demana:

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

b) (0.75 punts) Indica, i justifica, si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identifica-les si és el cas.

c) (1.50 punts) Escriu l'equació de recurrència del cost temporal en funció de la talla per a cadascun dels casos si hi ha més d'un, o una única equació si únicament hi haguera un cas. Ha de resoldre's per substitució.

d) (0.50 punts) Expressa el resultat anterior utilitzant notació asimptòtica.

Solució:

a) La talla del problema és el nombre d'elements del subarray `v[right..left]`, donat per l'expressió `right - left + 1` $= n$.

b) Es tracta d'un recorregut i, per tant, no hi ha instàncies significatives.

c) Plantegem l'equació de recurrència considerant les talles correponents al cas base i al cas general. Per simplicitat, aproximem a $n/2$ la talla de les dues crides del cas general.

$$T(n) = \begin{cases} 2 \cdot T(n/2) + 1 & \text{si } n > 0 \\ 1 & \text{si } n = 0 \end{cases}$$

Resolent per substitució:

$T(n) = 2 \cdot T(n/2) + 1 = 2^2 \cdot T(n/2^2) + 2 + 1 = 2^3 \cdot T(n/2^3) + 4 + 2 + 1 = 2^3 \cdot T(n/2^3) + 2^3 - 1 = \ldots = 2^i \cdot T(n/2^i) + 2^i - 1$. Si $1 \leq n/2^i < 2 \rightarrow i = \lfloor log_2 n \rfloor$, amb el que $T(n) = 2^{\lfloor log_2 n \rfloor} \cdot T(n/2^{\lfloor log_2 n \rfloor}) + 2^{\lfloor log_2 n \rfloor} - 1$.

Prenent $2^{\lfloor log_2 n \rfloor} \approx n$, es té que $T(n) = n \cdot T(1) + n - 1 = n \cdot (2 \cdot T(0) + 1) + n - 1$. S'arriba al cas base en el que $T(0) = 1$, amb el que $T(n) = 4n - 1$ *p.p.*

d) En notació asimptòtica, el cost temporal serà $T(n) \in \Theta(n)$.