PRG - ETSInf. TEORIA. Curs 2018-19. Recuperació Parcial 1.
11 de juny de 2019. Duració: 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. $\boxed{\text{4 punts}}$ Donat un enter `n` > 0, escriu un mètode recursiu que mostre per l'eixida estàndard les xifres de `n` en sentit invers, seguides de les mateixes xifres en el sentit en que apareixen en `n`. Per exemple, si `n` és 4873, cal escriure 37844873; si `n` és 48, cal escriure 8448; si `n` és 4, cal escriure 44.

Es demana:

a) (0.75 punts) Perfil del mètode amb la seua precondició.
b) (1.25 punts) Cas base i cas general.
c) (2 punts) Implementació en Java.

Solució:

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

```java
/** Precondició: n > 0 */
public static void xifres(int n)
```

b)
- Cas base: `n` ≤ 9, només una xifra.
- Cas general: `n` ≥ 10, més d'una xifra.

c)
```java
/** Precondició: n > 0 */
public static void xifres(int n) {
    if (n <= 9) { System.out.print(n + "" + n); }
    else {
        int xifra = n % 10;
        System.out.print(xifra);
        xifres(n / 10);
        System.out.print(xifra);
    }
}
```

2. $\boxed{\text{3 punts}}$ Una *matriu triangular superior* és una matriu quadrada, els valors de la qual per baix de la diagonal principal són 0. El mètode iteratiu `isUpperTriangular` comprova si la matriu `m` compleix aquesta propietat.

```java
/** Precondició: m és una matriu quadrada d'enters */
public static boolean isUpperTriangular(int[][] m) {
    boolean res = true;
    int i = m.length - 1;
    while (i >= 0 && res) {
        int j = 0;
        while (j < i && m[i][j] == 0) { j++; }
        if (j < i) { res = false; }
        else { i--; }
    }
    return res;
}
```

Es demana:

a) (0.25 punts) Indica la talla del problema, i 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 calcula 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 utilizant notació asimptòtica.

Solució:

a) La talla és l'ordre de la matriu `m`, la seua expressió en Java és `m.length`, a la que d'ara endavant anomenarem n.

b) Per a una talla donada n sí existeixen instàncies significatives: el cas millor es dóna quan el primer element que s'analitza, `m[m.length - 1][0]` és distint de `0`; el cas pitjor correspon a una matriu triangular superior, ja que s'ha de recórrer tots els elements necessaris per a comprovar que efectivament són iguals a `0`.

c) Es pot considerar com instrucció crítica (de cost constant) la condició de cerca del bucle més intern (`m[i][j] == 0`). Així, les funcions de cost es poden expressar:

- Per al cas millor: en l'única iteració del bucle principal es compleix que `m[m.length - 1][0]` és distint de `0`, aleshores el bucle secundari no s'executa, la variable `res` es fica a `false`, acabant el bucle principal. Per tant, el cost en el cas millor és constant, i.e. $T^m(n) = 1$ i.c.
- Per al cas pitjor: la variable `res` sempre val `true` i tots els elements examinats de la matriu contenen un 0, per tant, els bucles de cerca es converteixen en bucles de recorregut.
  $T^p(n) = \sum_{i=n-1}^{0} \sum_{j=0}^{i-1} 1 = \sum_{i=0}^{n-1} i = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$ i.c.

d) Les funcions de cost dels casos millor i pitjor expressades en notació asimptòtica són $T^m(n) \in \Theta(1)$ i $T^p(n) \in \Theta(n^2)$ i, si es pren la funció de cost del cas millor com a quota inferior i la del cas pitjor com a quota superior, la funció de cost expressada en notació asimptòtica és: $T(n) \in \Omega(1), T(n) \in O(n^2)$.

3. $\boxed{\text{3 punts}}$ El següent mètode calcula recursivament la suma dels elements de la submatriu de grandària `n*n` amb origen en l'element `(0,0)`, d'una matriu quadrada `m`. Noteu que en el cas de voler sumar tots els elements de la matriu, la crida inicial seria `sumar(m, m.length)`.

```
/** Precondició: m és una matriu quadrada d'enters i 0 <= n <= m.length */
public static int sumar(int[][] m, int n) {
    if (n == 0) { return 0; }
    else {
        int s = m[n - 1][n - 1];
        for (int i = 0; i < n - 1; i++) {
            s = s + m[n - 1][i] + m[i][n - 1];
        }
        return s + sumar(m, n - 1);
    }
}
```

Es demana:

a) (0.25 punts) Indica quina és la talla del problema, i l'expressió que la representa.
b) (0.75 punts) Determina si existeixen instàncies significatives. Si n'hi ha, identifica les que representen els casos millor i pitjor de l'algorisme.
c) (1.50 punts) Escriu l'equació de recurrència del cost temporal en funció de la talla per a cadascun dels casos si n'hi hagueren, o una única equació si només hi haguera un cas. Ha de resoldre's per substitució.
d) (0.50 punts) Expressa el resultat anterior mitjançant notació asimptòtica.

Solució:

a) La talla del problema és el valor del paràmetre n.

b) No existeixen instàncies significatives perquè es tracta d'un problema de recorregut.

c) Equació de recurrència:

$$T(n) = \begin{cases} k' & n = 0 \\ T(n-1) + (n-1) * k & n > 0 \end{cases}$$

Resolent per substitució:

$T(n) = T(n-1) + (n-1) * k = T(n-2) + ((n-2) + (n-1)) * k = \ldots = T(n-i) + (\sum_{j=1}^{i}(n-j)) * k$.
S'arriba al cas base $T(0) = k'$ quan $n - i = 0$, això és, quan $i = n$.

Així, $T(n) = k' + (\sum_{j=1}^{n} n - \sum_{j=1}^{n} j) * k = k' + (n^2 - \dfrac{n * (n+1)}{2}) * k$

d) La funció de cost expressada en notació asimptòtica és $T(n) \in \Theta(n^2)$.