PRG - ETSInf. TEORÍA. Curso 2018-19. Recuperación Parcial 1.
11 de junio de 2019. Duración: 2 horas.

Nota: El examen se evalúa sobre 10 puntos, pero su peso específico en la nota final de PRG es de **3 puntos**.

1. $\boxed{\text{4 puntos}}$ Dado un entero `n` > 0, escribir un método recursivo que escriba en salida las cifras de `n` en sentido inverso, seguidas de dichas cifras en el mismo sentido en que aparecen en `n`. Por ejemplo, si `n` es 4873, se debe escribir 37844873, si `n` es 48 se debe escribir 8448, si `n` es 4 se debe escribir 44.

   Se pide:

   a) (0.75 puntos) Perfil del método con su precondición.

   b) (1.25 puntos) Caso base y caso general.

   c) (2 puntos) Implementación en Java.

Solución:

a) Una posible solución consiste en definir un método con el siguiente perfil:

```
/** Precondición: n > 0 */
public static void cifras(int n)
```

b)
- Caso base, `n` ≤ 9 , tiene solo una cifra.
- Caso general, `n` ≥ 10, tiene más de una cifra.

c)
```
public static void cifras(int n) {
    if (n <= 9) { System.out.print(n + "" + n); }
    else {
        int cifra = n % 10;
        System.out.print(cifra);
        cifras(n / 10);
        System.out.print(cifra);
    }
}
```

2. $\boxed{\text{3 puntos}}$ Una *matriz triangular superior* es una matriz cuadrada cuyos valores por debajo de la diagonal principal son todos iguales a 0. El método iterativo `isUpperTriangular` comprueba si la matriz `m` cumple esta propiedad.

```
/** Precondición: m es una matriz cuadrada de enteros */
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;
}
```

   Se pide:

   a) (0.25 puntos) Indica cuál es el tamaño o talla del problema, así como la expresión que la representa.

   b) (0.75 puntos) Indica, y justifica, si existen diferentes instancias significativas para el coste temporal del algoritmo e identifícalas si es el caso.

c) (1.50 puntos) Elige una unidad de medida para la estimación del coste (pasos de programa, instrucción crítica) y de acuerdo con ella obtén una expresión matemática, lo más precisa posible, del coste temporal del método, distinguiendo el coste de las instancias más significativas en caso de haberlas.

d) (0.50 puntos) Expresa el resultado anterior utilizando notación asintótica.

Solución:

a) La talla es el orden de la matriz `m`, su expresión Java es `m.length`, a la que desde ahora llamaremos n.

b) Para una talla dada n sí existen instancias significativas: el caso mejor se da cuando el primer elemento que se analiza, `m[m.length - 1][0]` es distinto de `0`; el caso peor corresponde a una matriz triangular superior, ya que debe recorrer todos los elementos necesarios para comprobar que efectivamente son iguales a `0`.

c) Se puede considerar como instrucción crítica (de coste constante) la condición de búsqueda del bucle más interno (`m[i][j] == 0`). Así, las funciones de coste se pueden expresar como sigue:

- Para el caso mejor: en la única iteración del bucle principal se cumple que `m[m.length - 1][0]` es distinto de `0`, por lo que el bucle secundario ni siquiera llega a ejecutarse, se pone la variable `res` a `false`, acabando el bucle principal. Por tanto, el coste en el caso mejor es constante, i.e. $T^m(n) = 1$ i.c.
- Para el caso peor: la variable `res` siempre vale `true` y todos los elementos examinados de la matriz valen 0, por lo que los bucles de búsqueda se convierten en bucles de recorrido. $T^p(n) = \sum_{i=n-1}^{0} \sum_{j=0}^{i-1} 1 = \sum_{i=0}^{n-1} i = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$ i.c.

d) Las funciones de coste de los casos mejor y peor expresadas en notación asintótica son $T^m(n) \in \Theta(1)$ y $T^p(n) \in \Theta(n^2)$ y, si se toma la función de coste del caso mejor como cota inferior y la del caso peor como cota superior, la función de coste expresado en notación asintótica es: $T(n) \in \Omega(1), T(n) \in O(n^2)$.

3. $\boxed{\text{3 puntos}}$ Dada `m`, una matriz cuadrada de números enteros, el siguiente método calcula recursivamente la suma de los elementos de la submatriz de tamaño `n*n` con origen en el elemento `(0,0)`. Nótese que en el caso de querer sumar todos los elementos de la matriz, la llamada inicial sería `sumar(m, m.length)`.

```
/** Precondición: m es una matriz cuadrada de enteros y 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);
    }
}
```

Se pide::

a) (0.25 puntos) Indica cuál es la talla del problema.

Solución: La talla del problema es el valor del parámetro n.

b) (0.75 puntos) Determina si existen instancias significativas. Si las hay, identifica las que representan los casos mejor y peor del algoritmo.

Solución: No existen instancia significativas porque se trata de un problema de recorrido.

c) (1.50 puntos) Escribe la ecuación de recurrencia del coste temporal en función de la talla para cada uno de los casos si hubiera varios, o una única ecuación si sólo hubiera un caso. Resuélvela por sustitución.

Solución:

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

$$\begin{aligned}
&1)T(n) = T(n-1) + (n-1) * k' \\
&2)T(n-2) + ((n-2) + (n-1)) * k' \\
&\ldots \\
&i)T(n-i) + (\textstyle\sum_{j=1}^{i}(n-j)) * k' \\
T(n) &= k + (n^2 - \tfrac{n*(n+1)}{2}) * k'
\end{aligned}$$

d) (0.50 puntos) Expresa el resultado anterior usando notación asintótica.

Solución:

$$T(n) \in \theta(n^2)$$