PRG - ETSInf. TEORÍA. Curso 2017-18. Parcial 1. Recuperación
19 de junio de 2018. 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. 4 puntos Sea `a` un array de `double` cuyas componentes representan valores de la coordenada Y de una enumeración de puntos pertenecientes a una recta de pendiente positiva, de manera que el array está ordenado ascendentemente.

   Se debe escribir un método recursivo que busque el punto de corte de la recta con el eje X, o que devuelva -1 si el punto de corte no se encuentra entre los de `a`.

   Por ejemplo: si el array es {`-7.4, -1.3, 0.0, 1.8, 2.3, 3.6`} el método debe devolver 2. Si el array es {`-7.4, -1.3, 1.8, 2.6, 3.6`} el método debe devolver -1.

   Se pide:

   a) (0.75 puntos) Perfil del método, con los parámetros adecuados para resolver recursivamente el problema, y precondición relativa a dichos parámetros.
   b) (1.25 puntos) Caso base y caso general.
   c) (1.50 puntos) Implementación en Java.
   d) (0.50 puntos) Llamada inicial para que, dado un cierto array `a`, se realice el cálculo sobre todo el array.

Solución:

1. Una posible solución consiste en definir un método con el siguiente perfil, y que sigue una estrategia de búsqueda dicotómica:

```
/** Busca el valor 0 en a[left..right]. Precondición: 0 <= left, right < a.length,
 *  a[left..right] ordenado ascendentemente. */
public static int intercepts(double[] a, int left, int right) {
    if (left > right) { return -1; }
    else {
        int middle = (left + right) / 2;
        if (a[middle] == 0) { return middle; }
        else if (a[middle] > 0) {
            return intercepts(a, left, middle - 1);
        } else {
            return intercepts(a, middle + 1, right); }
    }
}
```

   La llamada inicial para resolver el problema sobre un cierto array `a` debería ser `intercepts(a, 0, a.length - 1)`.

2. Otra posible solución consiste en definir un método con el siguiente perfil, que sigue una estrategia de búsqueda lineal:

```
/** Busca el valor 0 en a[ini..a.length - 1].
 *  Precondición: 0 <= ini, a[ini..a.length - 1] ordenado ascendentemente. */
public static int intercepts(double[] a, int ini) {
    if (ini >= a.length) { return -1; }
    else if (a[ini] == 0) { return ini; }
    else if (a[ini] > 0) {return -1;}
    else { return intercepts(a, ini + 1); }
}
```

   La llamada inicial para resolver el problema sobre un cierto array `a` debería ser `intercepts(a, 0)`.

Ambos métodos son $\Omega(1)$, aunque el primero es $O(logn)$, mientras que el segundo es $O(n)$, siendo $n =$ `a.length`.

2. 3 puntos Dada una matriz cuadrada `m` de caracteres y un carácter `c`, el siguiente método escribe las palabras o secuencias de caracteres que aparecen en cada fila, eliminando de ellas cada aparición de `c`.

```
/** Precondición: m es una matriz cuadrada. */
public static void escribeSin(char[][] m, char c) {
    int dim = m.length;
    for (int i = 0; i < dim; i++) {
        for (int j = 0; j < dim; j++) {
            if (m[i][j] != c) {
                System.out.print(m[i][j]);
            }
        }
        System.out.println();
    }
}
```

Por ejemplo, si `m` = {{'e','e','l','e'}, {'m','e','m','e'}, {'n','u','l','l'}, {'c','a','s','e'}}, y `c` = 'e', entonces el método escribe:

```
l
mm
null
cas
```

Se pide:

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

b) (0.75 puntos) Indicar, y justificar, si existen diferentes instancias significativas para el coste temporal del algoritmo e identificarlas si es el caso.

c) (1.50 puntos) Elegir una unidad de medida para la estimación del coste (pasos de programa, instrucción crítica) y de acuerdo con ella obtener 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) Expresar el resultado anterior utilizando notación asintótica.

Solución:

a) La talla del problema es la dimensión de la matriz `m` y la expresión que la representa es `m.length`. De ahora en adelante, llamaremos a este número n. Esto es, $n =$ `m.length`.

b) No existen diferentes instancias. El método examina todos los caracteres de todas las filas de `m`.

c) Si elegimos como unidad de medida el paso de programa, se tiene:

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

Si elegimos como unidad de medida la instrucción crítica y considerando como tal, por ejemplo, la evaluación de la condición `m[i][j] != c` (de coste unitario), se tiene:

$T(n) = \sum_{i=0}^{n-1}(\sum_{j=0}^{n-1} 1) = \sum_{i=0}^{n-1} n = n^2$ *i.c.*, es decir, n^2 *p.p.* despreciando términos de orden inferior.

d) En notación asintótica $T(n) \in \Theta(n^2)$.

3. 3 puntos Se desea calcular el coste del siguiente método recursivo, que dados `x` > 1 y `d` $\leq$ `x`, comprueba si `x` no tiene ningún divisor propio en el rango [`d`, `x`[:

```
/** Precondición: x > 1 && 1 < d <= x */
public static boolean sinDivisores(int x, int d) {
    if (x == d) { return true; }
    else {
        if (x % d == 0) { return false; }
        else { return sinDivisores(x, d + 1); }
    }
}
```

Se pide:

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

b) (0.75 puntos) Indicar, y justificar, si existen diferentes instancias significativas para el coste temporal del algoritmo e identificarlas si es el caso.

c) (1.50 puntos) Escribir la ecuación de recurrencia del coste temporal en función de la talla para caso mejor y peor si los hubiera, o una única ecuación si no hubiera diferentes casos. Debe resolverse por sustitución.

d) (0.25 puntos) Expresar el resultado anterior utilizando notación asintótica.

e) (0.25 puntos) ¿Cuál sería el coste asintótico en función del valor de `x` de la llamada `sinDivisores(x, 2)`, es decir, de averiguar si `x` es primo?

Solución:

a) La talla del problema es la diferencia entre los valores de los argumentos `x - d`. Llamaremos n a este valor de aquí en adelante.

b) Se trata de una búsqueda del primer divisor propio de `x` mayor o igual que `d` y, por tanto, hay instancias significativas. En el caso mejor, `d` es divisor de `x`. En el caso peor, `x` no tiene ningún divisor mayor o igual que `d`.

c) Considerando el coste expresado en pasos de programa, en el caso mejor $T^m(n) = 1$ *p.p.* En el caso peor, planteamos la ecuación de recurrencia considerando las tallas correspondientes al caso base y al caso general:

$$T^p(n) = \begin{cases} 1 + T^p(n-1) & \text{si } n > 0 \quad (\text{cuando } \texttt{x} > \texttt{d}) \\ 1 & \text{si } n = 0 \quad (\text{cuando } \texttt{x} = \texttt{d}) \end{cases}$$

Resolviendo por sustitución:

$T^p(n) = 1 + T^p(n-1) = 2 + T^p(n-2) = 3 + T^p(n-3) = \ldots = k + T^p(n-k)$ después de k pasos de sustitución. Se llega al caso base $T^p(0) = 1$ después de $k = n$ pasos de sustitución, con lo que $T^p(n) = n + 1$ *p.p.*

d) En notación asintótica, el coste temporal será: $T^m(n) \in \Theta(1)$, $T^p(n) \in \Theta(n)$. Es decir, $T(n) \in \Omega(1)$ y $T(n) \in O(n)$.

e) En el caso particular de la llamada `sinDivisores(x, 2)`, la talla es $n =$ `x` $- 2$ y, por lo tanto, el coste en función de `x` es $\Omega(1), O(\texttt{x})$.