Cálculo del Orden de Complejidad de un Algoritmo Iterativo

Ejercicio

Véase el siguiente ejercicio que sirve como ejemplo del estudio que se lleva a cabo para el cálculo del orden exacto de funciones iterativas:

```
entero función fun_iter(E entero: n)
inicio
d ← 1
suma ← 0
mientras d ≤ n hacer
    x ← d
    mientras x ≥ 0 hacer
        suma ← suma + x
        x ← x − 1
    fin_ mientras
    d ← d + 1
fin_ mientras
devolver suma
fin_función
```

EL cálculo del coste total del algoritmo se puede simplificar haciendo uso de la operación crítica. Se llama operación crítica a la operación que más veces se ejecuta independientemente de los casos. La utilización de la operación crítica es un método sencillo de calcular el coste asintótico. Así, se puede considerar el coste en el peor caso como el número de veces que se repite esta operación. Calculando dicho número de veces, se obtiene directamente el orden de complejidad. La elección de la operación crítica ha de realizarse correctamente.

Siguiendo la definición de operación crítica en este ejemplo concreto, el cálculo del coste del algoritmo se reduciría a contar el número de veces que se repite la comparación $x \geq 0$, ya que no hay operación que se ejecute más que ella en el algoritmo. Sin embargo, se podría seleccionar cualquier otra operación, siempre y cuando el número de veces que ésta se ejecute sea del mismo orden. En este ejemplo se podría considerar cualquiera de las operaciones (incremento o decremento) que forman parte de las instrucciones del cuerpo del bucle interno ($suma \leftarrow suma + x$, $x \leftarrow x - 1$).

Los pasos a seguir serían, por tanto:

1. Seleccionar la categoría de operación crítica que mejor refleje la esencia del algoritmo.
2. Contar el número de veces que se ejecuta cualquier operación del algoritmo que se encuentre en dicha categoría.

Este algoritmo realiza una función mediante operaciones aritméticas. Se deduce que la esencia es la suma (el resto de operaciones aritméticas son necesarias para la inicialización de variables y control de bucles). De esta forma, se podría medir el tiempo simplemente en «número de sumas», y esa sería su operación crítica.

Por tanto, representando por $t(n)$ el tiempo que tarda en ejecutarse este algorimto para ejemplares de tamaño n, calculamos su valor a partir del número de veces que se repite la operación crítica, es decir, la operación de suma:

$t(n) = \sum_{d=1}^{n} (d+1) = \frac{n^2+3 \cdot n}{2}$

Es decir:

$t(n) \in \Theta(\frac{n^2+3 \cdot n}{2})$

Aplicando las propiedades elementales de simplificación:

$\Theta(c \cdot f) = \Theta(f)$ [Eliminación de las constantes]

$t(n) \in \Theta(n^2 + 3 \cdot n)$

$\Theta(f+g) = \Theta(\text{máx}(f,g))$ [Regla del máximo]

$t(n) \in \Theta(\text{máx}(n^2, 3 \cdot n))$

Se obtiene: $t(n) \in \Theta(n^2)$