Metodología de la Programación. Análisis de Algoritmos

Curso 98–99

Ejercicio 1.–La sucesión de Fibonacci está caracterizada por las condiciones siguientes:

$$\begin{cases} F(0) = 0, \ F(1) = 1 \\ F(n) = F(n-1) + F(n-2), \quad \text{si } n \geq 2 \end{cases}$$

Probar por inducción débil que para cada $n \geq 1$ se verifica que:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} F(n+1) & F(n) \\ F(n) & F(n-1) \end{pmatrix}$$

Ejercicio 2.–Consideremos la ecuación de recurrencia $x_{n+1} = x_n + n$ $(\forall n \in \mathbb{N})$. Se pide:

(a) Probar que dicha ecuación de recurrencia no posee soluciones polinómicas de primer grado.

(b) Hallar una solución particular del tipo $x_n = n \cdot (a \cdot n + b)$.

(c) Resolver dicha ecuación de recurrencia.

Ejercicio 3.–Hallar el orden exacto de la función f definida por recursión como sigue: $f(0) = 0$ y $f(n) = 2f(n-1) + n$, para cada $n \geq 1$.

Ejercicio 4.–Resolver la ecuación de recurrencia

$$\left.\begin{aligned} x_{n+1} &= x_n + 2y_n \\ y_{n+1} &= -x_n + 4y_n \end{aligned}\right\}$$

con las condiciones iniciales $x_1 = 2$ e $y_1 = 1$.

Ejercicio 5.–Sea x_n el número de subconjuntos que posee un conjunto con n elementos. Probar que se verifica $x_n = 2x_{n-1}$, para cada $n \geq 1$, y hallar x_n.

Ejercicio 6.–Probar que

$$\sum_{i=0}^{n} (-1)^i \cdot \frac{1}{i+1} \binom{n}{i} = \frac{1}{n+1}$$

Ejercicio 7.–Utilizando el ejercicio anterior, probar que

$$\sum_{i=1}^{n} \frac{(-1)^{i-1}}{i} \binom{n}{i} = \sum_{i=1}^{n} \frac{1}{i}$$

Ejercicio 8.–Probar que

$$\sum_{i=1}^{n} i^k \in \theta(n^{k+1})$$

Ejercicio 9.–Probar que

$$\sum_{i=1}^{n} \frac{1}{i} \in \theta(\lg n)$$

Ejercicio 10.–Probar que para cada $k \in \mathbb{R}^+$ se verifica:

$$\sum_{i=1}^{n} i^k \cdot \lg i \in \theta(n^{k+1} \cdot \lg n)$$

Ejercicio 11.–Probar que para cada $n \geq 2$ se tiene que

$$\sum_{i=0}^{n} (-1)^i \cdot \binom{n}{i} = 0$$

Ejercicio 12.–Probar que

$$O(1) \subsetneqq O(\lg n) \subsetneqq O(n) \subsetneqq O(n \cdot \lg n) \subsetneqq O(n^k) \subsetneqq O(\alpha^n) \subsetneqq O(n!)$$

para cada $k > 1$ y $\alpha > 1$.

Ejercicio 13.–Demostrar que:

(1) $\lg_a n \in \theta(\lg_b n)$, siendo $a, b > 1$.

(2) $2^{\lg_a n} \notin \theta(2^{\lg_b n})$, siendo $0 < a, b \neq 1$ y $a \neq b$.

(3) $\lg n! \in \theta(n \cdot \lg n)$.

Ejercicio 14.–Ordenar en forma creciente, según sus órdenes asintóticos, las siguientes funciones:

$$\left.\begin{array}{l} f(n) = n^3 \\ g(n) = n \cdot \lg n \end{array}\right\} \quad \left.\begin{array}{l} h(n) = \sqrt{2^n} \\ k(n) = 4596 \end{array}\right\} \quad \left.\begin{array}{l} \varphi(n) = n^{\frac{7}{3}} \\ \psi(n) = 2^n \end{array}\right\}$$

Interpretar algorítmicamente los resultados obtenidos.

Ejercicio 15.–Consideremos la función $f : \mathbb{N} \to \mathbb{R}^+$ definida por recursión como sigue:

$$f(n) = \begin{cases} 0 & \text{si} \quad n = 0 \\ f(n-1) + 2n - 1 & \text{si} \quad n \geq 1 \end{cases}$$

Se pide:

1. Probar por inducción débil que $\forall n \in \mathbb{N}\ (f(n) = n^2)$.
2. Resolver la ecuación de recurrencia que se obtiene de la definición de f, y comprobar que $\forall n \in \mathbb{N}\ (f(n) = n^2)$.

Ejercicio 16.–Consideremos la función $f : \mathbb{N} \to \mathbb{R}^+$ definida por recursión como sigue:

$$f(n) = \begin{cases} 0 & \text{si} \quad n = 0, 1 \\ 2f(\lfloor \frac{n}{2} \rfloor + n & \text{si} \quad n > 1 \end{cases}$$

Se pide:

1. Probar por inducción débil sobre k que $\forall n = 2^k\ (f(n) = n \cdot \lg n)$.
2. Deducir del apartado anterior un orden asintótico condicional de $f(n)$ y justificar que es posible eliminar el condicional.

Ejercicio 17.–Hallar el orden exacto de la función f definida como sigue:

$$f(n) = \begin{cases} 1 & \text{si} \quad n = 1 \\ \displaystyle\sum_{i=0}^{\lg\lfloor \frac{n}{2} \rfloor} f(2^i) + n & \text{si} \quad n > 1 \end{cases}$$

Ejercicio 18.–Hallar el orden exacto de la función f definida como sigue:

$$f(n) = \begin{cases} 0 & \text{si} \quad n = 1 \\ 1 + \frac{1}{n} \cdot \displaystyle\sum_{i=1}^{n-1} i \cdot f(i) & \text{si} \quad n > 1 \end{cases}$$

Indicación: Efectúese el cambio de variables $z_n = n \cdot f(n)$.

Ejercicio 19.–Hallar el coste en tiempo de los siguientes algoritmos que calculan la función factorial.

```
procedimiento A(n)
    si n = 0 entonces
        devolver 1
    si no
        devolver n · A(n − 1)
```

```
procedimiento B(n)
    si n = 0 entonces
        devolver 1
    si no
        j ← 1
        para k ← 1 hasta n hacer
            j ← k ∗ j
        devolver j
```

Ejercicio 20.–Hallar el coste en tiempo de los siguientes algoritmos que calculan la sucesión de Fibonacci.

```
procedimiento A(n)
    si n ≤ 1 entonces
        devolver n
    si no
        devolver A(n − 1) + A(n − 2)
```

```
procedimiento B(n)
    si n = 0 entonces
        devolver 0
    si no
        i ← 0; j ← 1
        para k ← 1 hasta n − 1 hacer
            j ← i + j; i ← j − i
        devolver j
```

Ejercicio 21.–(**El problema de las torres de Hanoi**) Se dan tres varillas y n discos que tienen radios distintos dos a dos. Suponiendo que inicialmente todos los discos están colocados en una varilla i, dispuestos de "menor"a "mayor", según su radio, diseñar un algoritmo que permita colocar los n discos en otra varilla $j \neq i$ en orden decreciente (de "menor"a "mayor"). Hallar el coste en tiempo del algoritmo diseñado.