Facultad de Informática

Ingeniería en Informática de Gestión

Estructuras de Datos y de la Información

Grupo A, Octubre 2002

Ejercicios de órdenes de complejidad y recurrencias

Ejercicio 1 Demostrar por inducción sobre $n \geq 0$ las siguientes igualdades:

1. $\sum_{i=1}^{n} i = n(n+1)/2$.
2. $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$.
3. $\sum_{i=1}^{n} 2^i i = (n-1)2^{n+1} + 2$.

Ejercicio 2 Suponiendo que todas las funciones que aparecen en los enunciados están definidas de $\mathbb{N}$ en $\mathbb{R}^*$, demostrar:

1. $f(n) \in O(f(n))$.
2. Si $f(n) \in O(g(n))$ y $g(n) \in O(h(n))$, entonces $f(n) \in O(h(n))$.
3. $O(f(n)) \subseteq O(g(n))$ si y sólo si $f(n) \in O(g(n))$.
4. $O(f(n)) = O(g(n))$ si y sólo si $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$.
5. $O(f(n)) \subset O(g(n))$ si y sólo si $f(n) \in O(g(n))$ y $g(n) \notin O(f(n))$.
6. Si $f_1(n) \in O(g_1(n))$ y $f_2(n) \in O(g_2(n))$, entonces $f_1(n)f_2(n) \in O(g_1(n)g_2(n))$.
7. $f(n) \in O(g(n))$ si y sólo si $g(n) \in \Omega(f(n))$.
8. $\Omega(f(n)) \subset \Omega(g(n))$ si y sólo si $f(n) \in \Omega(g(n))$ y $g(n) \notin \Omega(f(n))$.
9. $\Omega(f(n) + g(n)) = \Omega(\max(f(n), g(n)))$.
10. Si $\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$, entonces
    $\Theta(f(n)) = \{g : \mathbb{N} \to \mathbb{R}^* \mid \exists c, d \in \mathbb{R}^+ . \exists n_0 \in \mathbb{N} . \forall n \geq n_0 . cf(n) \leq g(n) \leq df(n)\}$.
11. $\Theta(f(n) + g(n)) = \Theta(\max(f(n), g(n)))$.
12. $f(n) \in \Theta(g(n))$ si y sólo si $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$.
13. $f(n) \in \Theta(g(n))$ si y sólo si $g(n) \in \Theta(f(n))$.
14. Si $f(n) \in \Theta(g(n))$ y $g(n) \in \Theta(h(n))$, entonces $f(n) \in \Theta(h(n))$.
15. $O(f(n)) = O(g(n))$ si y sólo si $\Theta(f(n)) = \Theta(g(n))$ si y sólo si $f(n) \in \Theta(g(n))$.

Ejercicio 3 Encontrar dos funciones $f, g : \mathbb{N} \to \mathbb{R}^*$ tales que $f(n) \notin O(g(n))$ y $g(n) \notin O(f(n))$.

Ejercicio 4 Encontrar dónde está el error en el siguiente razonamiento:

$$\sum_{i=1}^{n} i = 1 + 2 + \ldots + n \in O(1 + 2 + \ldots + n) = O(max(1, 2, \ldots, n)) = O(n)$$

Ejercicio 5 Demostrar el siguiente teorema, utilizando la versión de este teorema demostrada en clase y las propiedades del ejercicio anterior que hagan falta. Supongamos que $\lim_{n\to\infty} f(n)/g(n) = k \in \mathbb{R}^* \cup \{\infty\}$.

1. Si $k \in \mathbb{R}^+$, entonces $f \in \Theta(g)$.
2. Si $k = 0$, entonces $f \in O(g(n))$ pero $f \notin \Theta(g)$.
3. Si $k = \infty$, entonces $f \in \Omega(g)$ pero $f \notin \Theta(g)$.

Ejercicio 6 Demostrar que $log\ n \in O(\sqrt{(}n))$ pero que $\sqrt{(}n) \notin O(log\ n)$.

Ejercicio 7 Demuestra que $O(f(n)) \subset O(g(n))$ no implica que $lim_{n\to\infty} f(n)/g(n) = 0$.

Ejercicio 8 Demostrar la siguiente cadena de inclusiones estrictas:

$$O(1) \subset O(\log n) \subset O(n) \subset O(n^2) \subset \ldots \subset O(n^k) \subset \ldots \subset O(2^n) \subset O(n!)$$

Ejercicio 9 Comparar con respecto a O y Ω los siguientes pares de funciones:

1. $2^{n+1}, 2^n$.
2. $(n+1)!, n!$.
3. $\log n, \sqrt{n}$.
4. Para cualquier $a \in \mathbb{R}^+$, $\log n, n^a$.

Ejercicio 10 Demostrar o refutar: si $f \in O(n)$ entonces $2^{f(n)} \in O(2^n)$. Hacer lo mismo para Ω.

Ejercicio 11 ¿Verdadero o falso?

1. $2^n + n^{99} \in O(n^{99})$.
2. $2^n + n^{99} \in \Omega(n^{99})$.
3. $2^n + n^{99} \in \Theta(n^{99})$.
4. Si $f(n) = n^2$, entonces $f(n)^3 \in O(n^5)$.
5. Si $f(n) \in O(n^2)$ y $g(n) \in O(n)$, entonces $f(n)/g(n) \in O(n)$.
6. Si $f(n) = n^2$, entonces $3f(n) + 2n \in \Theta(f(n))$.
7. Si $f(n) = n^2$ y $g(n) = n^3$, entonces $f(n)g(n) \in O(n^6)$.

Ejercicio 12 Hallar la solución *exacta* de la siguiente recurrencia:

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(n-1) + n - 1 & n \geq 2. \end{cases}$$

Ejercicio 13 Hallar la solución *exacta* de la siguiente recurrencia para n potencia de 2:

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(n/2) + 6n - 1 & n \geq 2. \end{cases}$$

Ejercicio 14 Hallar la solución *exacta* de la siguiente recurrencia:

$$T(n) = \begin{cases} 1 & n = 1 \\ 6 & n = 2 \\ T(n-2) + 3n + 4 & n \geq 3. \end{cases}$$

Ejercicio 15 Hallar la solución *exacta* de la siguiente recurrencia:

$$T(n) = \begin{cases} 1 & n = 1 \\ \sum_{i=1}^{n-1} T(n-i) + 1 & n \geq 2. \end{cases}$$

Ejercicio 16 Hallar la solución *exacta* de la siguiente recurrencia:

$$T(n) = \begin{cases} 1 & n = 1 \\ 2\sum_{i=1}^{n-1} T(i) + 1 & n \geq 2. \end{cases}$$

Ejercicio 17 ¿Cuántas multiplicaciones realiza el siguiente algoritmo para calcular potencias?

```
fun potencia1(y : num, z : nat) dev p : nat
    p := 1 ;
    mientras z > 0 hacer
        p := p * y ;
        z := z - 1
    fmientras
ffun
```

Ejercicio 18 ¿Cuántas multiplicaciones realiza el siguiente algoritmo para calcular potencias (en el caso peor)?

```
fun potencia2(y : num, z : nat) dev p : nat
    p := 1 ;
    mientras z > 0 hacer
        si impar(z) entonces p := p * y fsi ;
        z := z div 2 ;
        y := y * y
    fmientras
ffun
```

Ejercicio 19 ¿Cuántas multiplicaciones realiza el siguiente algoritmo para calcular el factorial?

```
fun factorial(y : nat) dev f : nat
    f := 1 ;
    mientras y > 1 hacer
        f := f * y ;
        y := y - 1
    fmientras
ffun
```

Ejercicio 20 Supongamos que el siguiente algoritmo se ejecuta sobre un número natural n que es potencia de 5. Dar una relación de recurrencia para el número exacto de divisiones que se realizan en función de n y resolverla de forma exacta.

```
proc mickey(n : nat)
    si n = 1 entonces nada
    si no mouse := n/5 ;
        para i = 1 hasta 4 hacer
            mickey(mouse)
        fpara
    fsi
fproc
```

Ejercicio 21 ¿Qué valor devuelve la siguiente función? Dar la respuesta como función en términos de n, y calcular su complejidad.

```
fun valor1(n : nat) dev r : nat
var i, j, k : nat
    r := 0 ;
    para i = 1 hasta n − 1 hacer
        para j = i + 1 hasta n hacer
            para k = 1 hasta j hacer
                r := r + 1
            fpara
        fpara
    fpara
ffun
```

Ejercicio 22 ¿Qué valor devuelve la siguiente función? Dar la respuesta como función en términos de n, y calcular su complejidad.

```
fun valor2(n : nat) dev r : nat
var i, j, k : nat
    r := 0 ;
    para i = 1 hasta n hacer
        para j = i + 1 hasta n hacer
            para k = i + j − 1 hasta n hacer
                r := r + 1
            fpara
        fpara
    fpara
ffun
```

Ejercicio 23 Un algoritmo emplea 2 segundos en resolver un ejemplar de tamaño 1000 de un cierto problema. Cuánto tardará en resolver un problema de tamaño 3000,

- si el coste del algoritmo está en $\Theta(n^2)$?
- si el coste del algoritmo está en $\Theta(n\ log\ n)$?
- si el coste del algoritmo está en $\Theta(2^n)$?

Ejercicio 24 Para los siguientes programas, calcular manualmente los polinomios correspondientes a sus tiempos de ejecución en el caso mejor y en el caso peor. Decir el orden de complejidad al que pertenece la función del tiempo en el caso peor. Repetir el cálculo utilizando las técnicas aprendidas incluyendo la de las instrucción crítica:

- Cálculo del producto de dos matrices de dimensiones $n \times n$:

```
fun producto(a, b : vector[1..n, 1..n] de nat) dev c : vector[1..n, 1..n] de nat
var i, j, k, s : nat
    i := 1 ;
    mientras i ≤ n hacer
        j := 1 ;
        mientras j ≤ n hacer
            k := 1 ;
            s := 0 ;
            mientras k ≤ n hacer
                s := s + a[i, k] * b[k, j] ;
                k := k + 1 ;
            fmientras
            c[i, j] := s ;
            j := j + 1 ;
        fmientras
        i := i + 1 ;
    fmientras
ffun
```

- Ordenación de un vector $a[1..n]$ con $n \geq 0$ por el método de inserción:

```
proc seleccion(ent/sala : vector[1..n] de nat)
var i, p, x : nat ;  seguir : bool ;
    para i = 2 hasta n hacer
        p := i ;  x := a[i] ;  seguir := cierto ;
        mientras p > 1 ∧ seguir hacer
            si x < a[p−1] entonces a[p] := a[p−1]si no seguir := falso fsi ;
            p := p − 1
        fmientras
        a[p] := x
    fpara
fproc
```

- El siguiente programa averiga si una matriz es simétrica:

```
fun simetrica(v : vector[1..n, 1..n] de nat) dev b : bool
var i, j : nat ;
    b := cierto ;  i := 1 ;
    si i ≤ n ∧ b
        b := b ∧ (v[i, j] == v[j, i]) ;
        j := j + 1
    fsi
    i := i + 1
ffun
```