Metodología y tecnología de la programación. Grupo C

Ingeniería Informática (UCM) Curso 2008/2009

Hoja 1: Análisis de la complejidad de algoritmos

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

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

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

(c) $\sum_{i=1}^{n} i2^i = (n-1)2^{n+1} + 2$

Ejercicio 2 Comparar con respecto a O y Ω los siguientes pares de funciones:

(a) $2^{n+1}, 2^n$.

(b) $(n+1)!, n!$.

(c) $n + \log n, \sqrt{n}$.

(d) Para cualquier $a \in \mathbb{R}^+$, $\log n, n^a$.

Ejercicio 3 Demostrar o refutar las siguientes afirmaciones:

(a) $2^n + n^{99} \in O(n^{99})$.

(b) $2^n + n^{99} \in \Omega(n^{99})$.

(c) $2^n + n^{99} \in \Theta(n^{99})$.

(d) Si $f(n) = n^2$, entonces $f(n)^3 \in O(n^5)$.

(e) Si $f(n) \in O(n^2)$ y $g(n) \in O(n)$, entonces $f(n)/g(n) \in O(n)$.

(f) Si $f(n) = n^2$, entonces $3f(n) + 2n \in \Theta(f(n))$.

(g) Si $f(n) = n^2$ y $g(n) = n^3$, entonces $f(n)g(n) \in O(n^6)$.

(h) Si $f \in O(n)$ entonces $2^{f(n)} \in O(2^n)$.

Ejercicio 4 ¿Cuántas multiplicaciones realizan los siguientes algoritmos para calcular potencias (en el caso peor)?

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

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

Ejercicio 5 Calcular la complejidad del siguiente algoritmo considerando como operación básica la suma.

```
fun mult(y, z : nat) dev m : nat
   si z = 1 entonces m := y
   si no si par(z) entonces m := mult(2 * y, z div 2)
                   si no     m := mult(2 * y, z div 2) + y
        fsi
   fsi
ffun
```

Ejercicio 6 ¿Cuántas multiplicaciones realizan los siguientes algoritmos para calcular el factorial?

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

(b)
```
fun factorial-rec(x : nat) dev f : nat
   si x ≤ 1 entonces f := 1
   si no f := x * factorial-rec(x − 1)
   fsi
ffun
```

Ejercicio 7 Calcular la complejidad de los algoritmos siguientes en términos de n.

```
fun función1(n : nat) dev r : 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
```

```
fun función2(n : nat) dev r : nat
   r := 0
   k := 10
   para i = 1 hasta k hacer
      para j = i hasta n hacer
         r := r + 1
      fpara
   fpara
ffun
```

Ejercicio 8 Calcular la complejidad temporal en los casos mejor y peor de los siguientes algoritmos:

(a) Cálculo del producto de dos matrices de dimensiones $n \times n$:

```
fun producto(a[1..n, 1..n], b[1..n, 1..n] de ent) dev c[1..n, 1..n] de ent
   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
```

(b) El siguiente programa averigua si una matriz es simétrica:

```
fun simétrica(v[1..n, 1..n] de ent) dev b : bool
    b := cierto ;  i := 1
    mientras i ≤ n ∧ b hacer
        j := i + 1
        mientras j ≤ n ∧ b hacer
            b := v[i, j] = v[j, i]
            j := j + 1
        fmientras
        i := i + 1
    fmientras
ffun
```

Ejercicio 9 Hallar la solución *exacta* de las siguientes recurrencias:

(a) $T(1) = 1$, y para todo $n \geq 2$, $T(n) = 2T(n-1) + n - 1$.

(b) $T(1) = 1$, y para toda potencia de 2 $n \geq 2$, $T(n) = 2T(n/2) + 6n - 1$.

(c) $T(1) = 1$, $T(2) = 6$, y para todo $n \geq 3$, $T(n) = T(n-2) + 3n + 4$.

(d) $T(1) = 1$, y para todo $n \geq 2$, $T(n) = \sum_{i=1}^{n-1} T(i) + 1$.

(e) $T(1) = 1$, y para todo $n \geq 2$, $T(n) = 2\sum_{i=1}^{n-1} T(n-i) + 1$.

(f) $T(1) = 1$, y para toda potencia de 2 $n \geq 2$, $T(n) = 2T(n/2) + n \log n$.

Ejercicio 10 Hallar la solución *exacta* de las siguiente recurrencia:

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

Ejercicio 11 ¿Qué valor devuelve la siguiente función? Dar la respuesta como función en términos de n y calcular su complejidad.

```
fun valor(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
```