Universidad de Alcalá

Algoritmia y Complejidad

Ejercicios tema 1

Laboratorio Miércoles 15:00-17:00

Integrantes:

Carlos Javier Hellín Asensio
Diego Gutiérrez Marcos

Curso: 2020-2021

Ejercicio 3

Como n tiene más peso, contamos siempre que una línea de código o una condición vale 1, aunque en esa misma línea haya sumas, multiplicaciones, asignaciones, etc.… Como, por ejemplo, hacemos en valor <- valor + i

```
fun Calculo(x,y,z: entero) dev valor:entero
var i,j,t: entero
  valor ← 0                                          -> T(n)= 1
  Desde i ← x hasta y Hacer valor ← valor + i fdesde -> T(n)= Σ_{i=x}^{y} 1
  si (valor ÷ (x+y)) <= 1 entonces Devolver z        -> T(n)=1   -> T(n)=1
  si no
      t ← x + ((y-x) ÷ 2)  { ÷ es la división entera }  -> T(n)= 1
      Desde i ← x hasta y Hacer
            Desde j ← (3*x) hasta (3*y) Hacer
                  valor ← valor + Minimo(i,j)         } T(n)= Σ_{j=3x}^{3y} 1
            fdesde                                    } T(n)= Σ_{i=x}^{y} Σ_{j=3x}^{3y} 1
      fdesde
      valor ← valor + 4*Calculo(t,y,valor)           -> T(n)= 1+T(n/2)
      Devolver valor                                 -> T(n)= 1
  fsi
ffun
```

$$T(n) = 1 + \max\left\{1,\ 1 + \sum_{i=x}^{y}\left(\sum_{j=3x}^{3y} 1\right) + 1 + T(n/2) + 1\right.$$

$$T(n) = 1 + \sum_{i=x}^{y} 1 + 1 + \max\left\{1,\ 1 + \sum_{i=x}^{y}\left(\sum_{j=3x}^{3y} 1\right) + 1 + T(n/2) + 1\right\}$$

3

$$T(n) = 1 + \sum_{i=x}^{y} 1 + 1 + \max\left\{1, 1 + \sum_{i=x}^{y}\left(\sum_{j=3x}^{3y} 1\right) + 1 + T(n/2) + 1\right\} =$$

$$= 1 + n + 1 + 1 + n \cdot 3n + 1 + T(n/2) + 1 =$$

$$= 5 + n + 3n^2 + T(n/2)$$

Cambio variable $\Rightarrow$ $n = 2^k$

$$T(2^k) = 5 + 2^k + 3 \cdot 4^k + T(2^{k-1})$$

$$x^k = 5 + 2^k + 3 \cdot 4^k + x^{k-1}$$

$$x^k - x^{k-1} = 5 + 2^k + 3 \cdot 4^k$$

Homogénea $\Rightarrow$ $x^{(h)} \Rightarrow x^k - x^{k-1} = 0$

$$x^{k-1}\underbrace{(x-1)}_{\to x=1} = 0$$

$x = 1 \quad x^{(h)} = A$

Particular $\Rightarrow$ $x^{(p)} = B \cdot k + C \cdot 2^k + D \cdot 4^k$

General $\Rightarrow$ $x^{(g)} = x^{(h)} + x^{(p)}$

$$x^{(g)} = A + B \cdot k + C \cdot 2^k + D \cdot 4^k$$

Deshacer cambio variable $\Rightarrow$ $n = 2^k \Rightarrow k = \log_2 n$

$$x^{(g)} = A + B \cdot \log_2 n + C \cdot n + D \cdot n^2$$

En esta ecuación, el peor caso es $D \cdot n^2$. Por tanto, la complejidad es $\boxed{O(n^2)}$

Ejercicio 7

```
fun EsPrimo(n: entero) dev resultado:boolean
    si (n = 1) Devolver Falso fsi
    Desde i ← 2 hasta n Hacer
        si (n % i = 0) entonces
            Devolver Falso
        fsi
    fdesde
    Devolver Verdadero
ffun
```

si (n = 1): $T(n)=1$

Devolver Falso fsi: $T(n)=1$

$T(n)=1+1=2$

si (n % i = 0): $T(n)=1$

Devolver Falso: $T(n)=1$

$T(n)=1+1=2$

$T(n)=\sum_{i=2}^{n} 2$

Devolver Verdadero: $T(n)=1$

$$T(n)=2+\sum_{i=2}^{n} 2+1$$

```
fun EsPerfecto(n: entero) dev resultado:boolean
    var acumulador: entero
    acumulador ← 0
    Desde i ←1 hasta n Hacer
        si (n % i = 0) entonces
            acumulador ← acumulador + i
        fsi
    fdesde

    si (n = acumulador) entonces
        Devolver Verdadero
    si no
        Devolver Falso
    fsi
ffun
```

acumulador ← 0: $T(n)=1$

si (n % i = 0): $T(n)=1$

acumulador ← acumulador + i: $T(n)=1$

$T(n)=2$

$T(n)=\sum_{i=1}^{n} 2$

si (n = acumulador): $T(n)=1$

Devolver Verdadero: $T(n)=1$

Devolver Falso: $T(n)=1$

$T(n)=1+\max\{1,1\}=2$

$$T(n)=1+\sum_{i=1}^{n} 2+2$$

```
fun ContarPrimos(n: entero) dev contador:entero
    contador ← 0
    Desde i ← 1 hasta n Hacer
        si (EsPrimo(i)) entonces
            contador ← contador + 1
        fsi
    fdesde
    Devolver contador
ffun
```

contador ← 0: $T(n)=1$

si (EsPrimo(i)): $T(n)=T(n)_{EsPrimo}=2+\sum_{i=2}^{n} 2+1$

contador ← contador + 1: $T(n)=1$

fsi: $T(n)=1+2+\sum_{i=2}^{n} 2+1$

$T(n)=\sum_{i=1}^{n}\left(1+2+\sum_{i=2}^{n} 2+1\right)$

Devolver contador: $T(n)=1$

$$T(n)=1+\sum_{i=1}^{n}\left(4+\sum_{i=2}^{n} 2\right)+1$$

fun ContarPerfectos(n: entero) dev contador:entero
    contador ← 0 → $T(n) = 1$
    Desde i ← 1 hasta n Hacer:
        si (EsPerfecto(i)) entonces → $T(n) = T(n)_{EsPerfecto} = 1 + \sum_{i=1}^{n} 2 + 2$
            contador ← contador + 1 → $T(n) = 1$
        fsi
    fdesde → $T(n) = 1 + \sum_{i=1}^{n} 2 + 2 + 1$
    Devolver contador → $T(n) = 1$
ffun → $T(n) = \sum_{i=1}^{n} \left(1 + \sum_{i=1}^{n} 2 + 2 + 1\right)$

→ $T(n) = 1 + \sum_{i=1}^{n} \left(4 + \sum_{i=1}^{n} 2\right) + 1$

proc Programa(N: entero) → $T(n) = 1$
    Imprimir(N + " tiene " + ContarPrimos(N) + " primos") → $T(n)_{ContarPrimos}$
    Imprimir(N + " tiene " + ContarPerfectos(N) + " perfectos") → $T(n)$ Contar Perfectos
fproc

Para este ejercicio se ha dividido el problema en varias funciones. Los principales son EsPrimo y EsPerfecto para saber si un número es primo y si es perfecto respectivamente.

En el caso de EsPrimo como parámetros de entrada tiene número a comprobar que llamaremos n. Si n es igual a 1 devuelve Falso y después se hace un bucle desde i igual a 2 hasta n para comprobar si el resto de la división de n entre i es igual a cero (usando módulo) entonces se devuelve Falso. En caso contrario y cuando haya terminado el bucle, se devuelve Verdadero.

Para el caso de EsPerfecto como parámetros el número a comprobar que llamaremos n. Se crea una variable acumulador iniciada a 0, y se hace un bucle desde i igual a 1 hasta n para comprobar si el resto de la división de n entre i es igual a cero (usando módulo) entonces se le suma al acumulador el valor i.
Una vez terminado el bucle, si n es igual al acumulador se devuelve Verdadero y en caso contrario, se devuelve Falso.

Por último, tenemos las funciones ContarPrimos y ContarPerfectos que son llamados desde el procedimiento Programa para mostrarlo en pantalla. Estas dos funciones tienen los mismos parámetros de entrada que es n (el número positivo del usuario) y se crea una variable contador para saber cuántos hay empezando por 0.
Se crea un bucle desde i igual 1 hasta n y se llama a las funciones respectivas (ContarPrimos llama EsPrimo y ContarPerfectos llamada EsPerfecto) si es cierto, a la variable contador se le suma un 1 más, hasta terminar el bucle y devolver el contador.

7

Función EsPrimo:

$$T(n) = 2 + \sum_{i=2}^{n} 2 + 1 = 3 + \sum_{i=2}^{n} 2 = 3 + 2 \cdot n$$

Función ContarPrimos:

$$T(n) = 1 + \sum_{i=1}^{n} \left(4 + \sum_{i=2}^{n} 2\right) + 1 = 2 + \sum_{i=1}^{n} \left(4 + \sum_{i=2}^{n} 2\right) =$$

$$= 2 + 4 \cdot n + 2 \cdot n^2$$

Complejidad $\Rightarrow n^2$

Función EsPerfecto:

$$T(n) = 1 + \sum_{i=1}^{n} 2 + 2 = 3 + \sum_{i=1}^{n} 2 = 3 + 2 \cdot n$$

Funcion ContarPerfectos:

$$T(n) = 1 + \sum_{i=1}^{n} \left(4 + \sum_{i=1}^{n} 2\right) + 1 = 2 + \sum_{i=1}^{n} \left(4 + \sum_{i=2}^{n} 2\right) =$$

$$= 2 + 4 \cdot n + 2 \cdot n^2$$

Complejidad $\Rightarrow n^2$

La complejidad del programa, tras aplicar la regla de la suma entre ContarPrimos y ContarPerfectos, es $O(\max(n^2, n^2))$. Por tanto, la complejidad es de $\boxed{O(n^2)}$

Ejercicio 9

```
fun Sumatorio(n: entero) dev valor:entero
    si (n = 1) entonces Devolver 1
    si no Devolver n + Sumatorio(n - 1)
    fsi
ffun
```

$T(n) = 1$

$T(n) = 1$

$T(n) = 1 + T(n-1)$

$T(n) = 1 + max\{1, 1 + T(n-1)\}$

$T(n) = 1 + max\{1, 1 + T(n-1)\}$

Para esta función se ha hecho de forma recursiva. La condición de parada es si n es igual a 1 entonces devuelve 1, en caso contrario se llama de forma recursiva a Sumatorio para n - 1 y el resultado se suma a n para luego devolverlo.

De esta forma se consigue el sumatorio de 1 + 2 + 3 + 4 + … + n -1 + n

Como n tiene más peso, contamos siempre que una línea de código o una condición vale 1, aunque en esa misma línea haya sumas, multiplicaciones, asignaciones, etc…. Como, por ejemplo, hacemos en Devolver n + Sumatorio(n-1) que es 1 + T(n-1)

9

$$T(n) = 1 + max\{1, 1 + T(n-1)\}$$

$$T(n) = 2 + T(n-1)$$

$$X^n = 2 + X^{n-1}$$

$$X^n - X^{n-1} = 2$$

Homogénea $\Rightarrow X^{(h)} \Rightarrow X^n - X^{n-1} = 0$

$$X^{n-1}\underbrace{(x-1)}_{\to x=1} = 0 \qquad x^{(h)} = A$$

Particular $\Rightarrow X^{(p)} = B \cdot n$

General $\Rightarrow X^{(g)} = X^{(h)} + X^{(p)}$

$$X^{(g)} = A + B \cdot n$$

El peor caso de esta ecuación es $B \cdot n$. Por tanto, la complejidad es $\boxed{O(n)}$