

## Ejercicio 9 - Sesión 3 (Recurrencias)

def algoritmo (x: array, n: int):

var i, j

var a, b, c: array

if n > 1:

    for i=1 to n/2:

        for j=1 to n/2:

            a[i,j] = x[i,j]

            b[i,j] = x[i,j+n/2]

            c[i,j] = x[i+n/2,j]

$$\sum_{i=1}^{n/2} \sum_{j=1}^{n/2} 3 = \frac{3}{4} n^2$$

Nos centramos en estas 3 instrucciones

algoritmo (a, n/2)

algoritmo (b, n/2)

algoritmo (c, n/2)

→ 3 llamadas recursivas

- Ecuación de recurrencia

$$\begin{cases} T(1) = 1 \\ T(n) = 3T\left(\frac{n}{2}\right) + \frac{3}{4}n^2 \end{cases}$$

\*1: Asumimos que se ejecuta una instrucción (ej. el if)

- Obtener una cota resolviendo la ecuación

1. Necesitamos cambiar la ecuación para que sea posible resolver con el método de la ecuación característica

2. Cambio de variable

$$\text{Hacemos que } n = 2^k \Rightarrow T(2^k) = 3T\left(\frac{2^k}{2}\right) + \frac{3}{4}(2^k)^2 \quad *_2 \quad *_2(2^k)^2 = (2^2)^k = 4^k$$

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

$$\Rightarrow *_3 S(k) = 3 \cdot S(k-1) + \frac{3}{4}4^k$$

3. Ecuación característica

$$S(k) - 3S(k-1) = \frac{3}{4}4^k$$

\*4: porque  $a_0t(n) + a_1 \cdot t(n-1) + \dots + a_k \cdot t(n-k) = b^n \cdot p(n)$

$$\Rightarrow *_4 (x-3) \cdot (x-4) \Rightarrow x \left\{ \begin{array}{l} 3 \\ 4 \end{array} \right.$$

$\downarrow$   
 $(a_0 \cdot x^{k-1} + a_1 \cdot x^{k-2} + \dots + a_k)(x-b)$  d grado

$$\begin{aligned} \Leftrightarrow S(k) &= C_1 \cdot 3^k + C_2 \cdot 4^k \\ \Rightarrow f(2^k) &= C_1 \cdot 3^k + C_2 \cdot 4^k \\ \Rightarrow f(n) &= C_1 \cdot 3^{\log_2 n} + C_2 \cdot 4^{\log_2 n} \end{aligned} \Rightarrow \Theta(n^2)$$

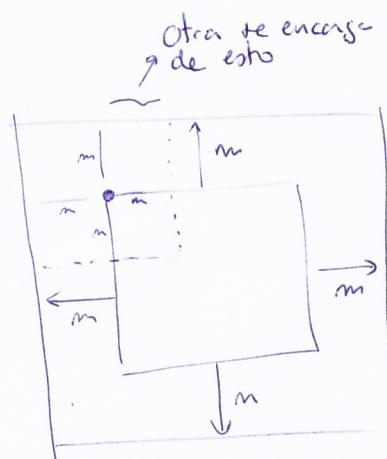
deshacer el cambio

## Ejercicio 1 - Sesión 4 (repaso)

operación Una ( $\text{Imagen}[n,n]$ , m)

```

 $i = m+1$ 
repetir
  para  $j = m+1$  hasta  $n-m$ :
    | otra ( $\text{Imagen}$ ,  $i, j, m$ )
    |    $i = i + 1$ 
hasta  $i > n-m$ 
```



operación Otra ( $\text{Imagen}[n,n]$ ,  $x, y, \text{tam}$ )

```

acum = 0
para  $i = y - \text{tam}$  hasta  $y + \text{tam}$ :
  para  $j = x - \text{tam}$  hasta  $x + \text{tam}$ :
    | acum = acum +  $\text{Imagen}[i, j]$   $\Leftarrow$  (I)
```

El algoritmo va moviendo una caja de  $m \times m$  por todo el array

- Calcular el número de veces que se ejecuta (I)

1. Empezamos por Otra

$$\sum_{i=y-\text{tam}}^{y+\text{tam}} \sum_{j=x-\text{tam}}^{x+\text{tam}} 1 = \sum_{i=y-\text{tam}}^{y+\text{tam}} (2\text{tam}+1) = (2\text{tam}+1) \cdot (2\text{tam}+1) = (2\text{tam}+1)^2$$

$\Downarrow$  es igual

~~$y+\text{tam} - y - \text{tam} + 1 = 2\text{tam} + 1$~~

$$2. Sustituimos \text{tam} \Rightarrow m \rightarrow t_{\text{OTRA}}(m) = (2m+1)^2$$

3. Resolvemos Una

$$\sum_{i=m+1}^{n-m} \sum_{j=m+1}^{n-m} t_{\text{OTRA}}(m) \Rightarrow \cancel{\sum_{i=m+1}^{n-m} \sum_{j=m+1}^{n-m} (2m+1)^2} \cdot (n-2m)^2 \cdot (2m+1)^2$$

$\Downarrow$  es igual

$n-m - m - 1 + 1$

$n-2m$

Ejercicio 2 - Sesión 4 (repasso)

	$t(n)$	dominante	$O(\cdot)$
1	$5 + 0.001n^3 + 0.025n$	$0.001n^3$	$O(n^3)$
2	$500n + 100n^{1.5} + 50n \cdot \log_{10}n$	$100n^{1.5}$	$O(n^{1.5})$
3	$0.3n + 5n^{1.5} + 2.5n^{1.75}$	$2.5n^{1.75}$	$O(n^{1.75})$
4	$n^2 \log_2 n + n(\log_2 n)^2$	$n^2 \log_2 n$	$O(n \cdot \log n)$
5	$n \cdot \log_3 n + n \cdot \log_2 n$	$n \cdot \log_3 n, n \cdot \log_2 n$	$O(n \cdot \log n)$
6	$3 \log_2 n + \log_2 \log_2 \log_2 n$	$3 \cdot \log_2 n$	$O(\log n)$
7	$100n + 0.01 \cdot n^2$	$0.01n^2$	$O(n^2)$
8	$0.01n + 100n^2$	$100n^2$	$O(n^2)$
9	$2n + n^{0.5} + 0.5n^{1.25}$	$0.5n^{1.25}$	$O(n^{1.25})$
10	$0.01n \cdot \log_2 n + n \cdot (\log_2 n)^2$	$n(\log_2 n)^2$	$O(n \cdot \log^2 n)$
11	$100n \cdot \log_3 n + n^3 + 100n$	$n^3$	$O(n^3)$
12	$0.003 \cdot \log_4 n + \log_2 \log_2 n$	$0.003 \cdot \log_4 n$	$O(\log n)$

$$* 2 \Rightarrow 100n^{1.5} \Rightarrow 100n \cdot n^{0.5}$$

• Ejercicios 3 - Sesión 4 (reparo)

• Propiedades de las notaciones asintóticas

1.  $O(f+g) = O(f) + O(g)$

FALSO

↳  $O(f+g) = \max(O(f), O(g))$

2.  $O(f \cdot g) = O(f) \cdot O(g)$

VERDAD

3. Transitividad

si  $g = O(f)$  y  $h = O(f)$

entonces  $g = O(h)$

FALSO

↳ si  $g = O(f)$  y  $f = O(h)$

entonces  $g = O(h)$

4.  $5n + 8n^2 + 100n^3 \in O(n^4)$

↳ VERDAD

5.  $5 + 8n^2 + 100n^3 \in O(n^2 \log n)$

↳ FALSO  $\rightarrow \in O(n^3)$

## Ejercicio 7 - Sesión 4 (repetición)

Tenemos dos paquetes software A y B.

El tamaño de los datos que se quieren procesar es  $10^9$  elementos.

$$T_A(n) = 0.001 \cdot n \text{ ms}$$

$$T_B(n) = 500 \sqrt{n} \text{ ms}$$

- ¿Cuál es el mejor en términos de O?

$$\del{T_A(n) \in O(n)}$$

$$T_B(n) \in O(n^{0.5})$$

En principio, solo con la información sobre O, elegiríamos el paquete B

- ¿Cuál elegiríamos?

Puesto que tenemos información más precisa ( $T_A$  y  $T_B$ ) podemos usarlo.

El paquete B comienza a superar a A cuando:

$$T_A(n) \geq T_B(n)$$

es decir,

$$0.001 \cdot n \geq 500 \sqrt{n}$$

$$\Rightarrow 10^{-3} \cdot n \geq 500 \sqrt{n}$$

$$\xrightarrow{\text{elevar al cuadrado}} 10^{-6} \cdot n^2 \geq 25 \cdot 10^4 \cdot n$$

$$\Rightarrow n \geq 25 \cdot 10^{10}$$

Para que tenga sentido usar B hay que procesar al menos  $25 \cdot 10^{10}$  elementos. Por tanto, para este caso ( $10^9$  elementos), es mejor A.

## Ejercicio 8 - Series 4 (repetición)

int doble (n) {

if (n == 1) {

return 1;

else {

return 2 \* doble (n - 1);

}

}

Válor

$$T(1) = 1$$

$$T(n) = 2 \cdot T(n-1)$$

$$x - 2 = 0 \rightarrow x = 2$$

$$f(n) = C_1 \cdot 2^n$$

$$f(1) = 1 = C_1 \cdot 2^1 = C_1 = \frac{1}{2}$$

$$\boxed{f(n) = 2^{n-1}}$$

Tiempo

$$f(1) = 1 + 1 = 2$$

$$f(n) = 4 * T(n-1) + 1$$

$$(x-4) \cdot (x-1) \Rightarrow x \leq 1$$

$$a_0 \cdot f(n) + a_1 \cdot f(n-1) + \dots + a_k \cdot f(n-k) = b^n \cdot p(n)$$

$$(ec. característica) (x-b)^{d+1}$$

grado d

$$f(n) = C_1 \cdot 4^n + C_2 \cdot 1^n$$

$$f(1) = 2 = C_1 \cdot 4 + C_2 \Rightarrow C_2 = 2 - C_1 \cdot 4$$

$$f(2) = 9 = C_1 \cdot 16 + C_2 \Rightarrow 9 = 16 \cdot C_1 + 2 - 4C_1$$

$$9 = 12C_1 + 2$$

$$7 = 12C_1$$

$$C_1 = \frac{7}{12}$$

$$C_2 = 2 - \frac{28}{12} = -\frac{1}{3}$$

$$\boxed{f(n) = \frac{7}{12} \cdot 4^n - \frac{1}{3}}$$