

# Algorítmica



**Los Del DGIIM**, [losdeldgiim.github.io](https://losdeldgiim.github.io)

Doble Grado en Ingeniería Informática y Matemáticas  
Universidad de Granada



Esta obra está bajo una Licencia Creative Commons Atribución-NoComercial-SinDerivadas 4.0 Internacional (CC BY-NC-ND 4.0).

Eres libre de compartir y redistribuir el contenido de esta obra en cualquier medio o formato, siempre y cuando des el crédito adecuado a los autores originales y no persigas fines comerciales.

# Algorítmica

Los Del DGIIM, `losdeldgiim.github.io`

Arturo Olivares Martos

Granada, 2023-24



# Índice general

<b>1. Relaciones de Problemas</b>	<b>5</b>
1.1. La eficiencia de los algoritmos . . . . .	5



# 1. Relaciones de Problemas

## 1.1. La eficiencia de los algoritmos

**Ejercicio 1.1.1.** Demostrar las siguientes propiedades:

a)  $k \cdot f(n) \in O(f(n)), \quad \forall k > 0.$

Hemos de ver que existe una constante  $c \in \mathbb{R}^+$ ,  $n_0 \in \mathbb{N}$  tal que  $k \cdot f(n) \leq c \cdot f(n)$  para todo  $n \geq n_0$ . En este caso, podemos tomar  $c = k$  y  $n_0 = 1$  y se tiene que  $k \cdot f(n) \leq k \cdot f(n)$  para todo  $n \in \mathbb{N}$ .

b)  $n^r \in O(n^k)$  si  $0 \leq r \leq k$ .

Hemos de ver que existe una constante  $c \in \mathbb{R}^+$ ,  $n_0 \in \mathbb{N}$  tal que  $n^r \leq c \cdot n^k$  para todo  $n \geq n_0$ .

Como  $0 \leq r \leq k$ , entonces  $n^r \leq n^k$  para todo  $n \in \mathbb{N}$ , por lo que podemos tomar  $c = 1$  y  $n_0 = 1$ .

c)  $O(n^k) \subset O(n^{k+1})$ .

Sea  $f(n) \in O(n^k)$ ; es decir, existe una constante  $c \in \mathbb{R}^+$ ,  $n_0 \in \mathbb{N}$  tal que  $f(n) \leq c \cdot n^k$  para todo  $n \geq n_0$ . Hemos de ver que  $f(n) \in O(n^{k+1})$ ; es decir, que existe una constante  $c' \in \mathbb{R}^+$ ,  $n'_0 \in \mathbb{N}$  tal que  $f(n) \leq c' \cdot n^{k+1}$  para todo  $n \geq n'_0$ .

Tomando  $c' = c$  y  $n'_0 = n_0$ , se tiene que  $f(n) \leq c \cdot n^k \leq c \cdot n^{k+1}$  para todo  $n \geq n_0$ , por lo que  $f(n) \in O(n^{k+1})$ .

d)  $n^k \in O(b^n) \quad \forall b > 1, k \geq 0$ .

Hemos de ver que existe una constante  $c \in \mathbb{R}^+$ ,  $n_0 \in \mathbb{N}$  tal que  $n^k \leq c \cdot b^n$  para todo  $n \geq n_0$ . Tomando  $c = 1$ , tenemos que dicho valor de  $n_0$  existe, ya que:

$$\lim_{n \rightarrow \infty} \frac{n^k}{b^n} = 0$$

e)  $\log_b n \in O(n^k) \quad \forall b > 1, k > 0$ .

Hemos de ver que existe una constante  $c \in \mathbb{R}^+$ ,  $n_0 \in \mathbb{N}$  tal que  $\log_b n \leq c \cdot n^k$  para todo  $n \geq n_0$ . Tomando  $c = 1$ , tenemos que dicho valor de  $n_0$  existe, ya que:

$$\lim_{n \rightarrow \infty} \frac{\log_b n}{n^k} = 0$$

- f) Si  $f(n) \in O(g(n))$  y  $h(n) \in O(g(n))$ , entonces  $f(n) + h(n) \in O(g(n))$ .

Tenemos que:

$$\begin{aligned} f(n) \in O(g(n)) &\implies \exists c_1 \in \mathbb{R}^+, n_1 \in \mathbb{N} \text{ tal que } f(n) \leq c_1 \cdot g(n) \quad \forall n \geq n_1, \\ h(n) \in O(g(n)) &\implies \exists c_2 \in \mathbb{R}^+, n_2 \in \mathbb{N} \text{ tal que } h(n) \leq c_2 \cdot g(n) \quad \forall n \geq n_2. \end{aligned}$$

Tomando  $c = c_1 + c_2$  y  $n_0 = \max\{n_1, n_2\}$ , se tiene que:

$$f(n) + h(n) \leq c_1 \cdot g(n) + c_2 \cdot g(n) = (c_1 + c_2) \cdot g(n) \quad \forall n \geq n_0,$$

- g) Si  $f(n) \in O(g(n))$ , entonces  $f(n) + g(n) \in O(g(n))$ .

Por el primer apartado, sabemos que  $g(n) \in O(g(n))$ . Por tanto, usando el apartado anterior, se tiene que  $f(n) + g(n) \in O(g(n))$ .

- h) *Reflexividad*:  $f(n) \in O(f(n))$ .

Se tiene de forma directa por el primer apartado tomando  $k = 1$ .

- i) *Transitividad*: Si  $f(n) \in O(g(n))$  y  $g(n) \in O(h(n))$ , entonces  $f(n) \in O(h(n))$ .

Tenemos que:

$$\begin{aligned} f(n) \in O(g(n)) &\implies \exists c_1 \in \mathbb{R}^+, n_1 \in \mathbb{N} \text{ tal que } f(n) \leq c_1 \cdot g(n) \quad \forall n \geq n_1, \\ g(n) \in O(h(n)) &\implies \exists c_2 \in \mathbb{R}^+, n_2 \in \mathbb{N} \text{ tal que } g(n) \leq c_2 \cdot h(n) \quad \forall n \geq n_2. \end{aligned}$$

Por tanto, tomando  $c = c_1 \cdot c_2$  y  $n_0 = \max\{n_1, n_2\}$ , se tiene que:

$$f(n) \leq c_1 \cdot g(n) \leq c_1 \cdot c_2 \cdot h(n) = c \cdot h(n) \quad \forall n \geq n_0,$$

- j) *Regla de la suma*: Si  $T1(n)$  es  $O(f(n))$  y  $T2(n)$  es  $O(g(n))$ , entonces:

$$T1(n) + T2(n) \in O(\max\{f(n), g(n)\}).$$

Tenemos que:

$$\begin{aligned} T1(n) \in O(f(n)) &\implies \exists c_1 \in \mathbb{R}^+, n_1 \in \mathbb{N} \text{ tal que } T1(n) \leq c_1 \cdot f(n) \quad \forall n \geq n_1, \\ T2(n) \in O(g(n)) &\implies \exists c_2 \in \mathbb{R}^+, n_2 \in \mathbb{N} \text{ tal que } T2(n) \leq c_2 \cdot g(n) \quad \forall n \geq n_2. \end{aligned}$$

Tomando  $c = \max\{c_1, c_2\}$  y  $n_0 = \max\{n_1, n_2\}$ , se tiene que:

$$T1(n) + T2(n) \leq c_1 \cdot f(n) + c_2 \cdot g(n) \leq c \cdot \max\{f(n), g(n)\} \quad \forall n \geq n_0,$$

- k) *Regla del producto*: Si  $T1(n)$  es  $O(f(n))$  y  $T2(n)$  es  $O(g(n))$ , entonces:

$$T1(n) \cdot T2(n) \in O(f(n) \cdot g(n)).$$

Tenemos que:

$$\begin{aligned} T1(n) \in O(f(n)) &\implies \exists c_1 \in \mathbb{R}^+, n_1 \in \mathbb{N} \text{ tal que } T1(n) \leq c_1 \cdot f(n) \quad \forall n \geq n_1, \\ T2(n) \in O(g(n)) &\implies \exists c_2 \in \mathbb{R}^+, n_2 \in \mathbb{N} \text{ tal que } T2(n) \leq c_2 \cdot g(n) \quad \forall n \geq n_2. \end{aligned}$$

Tomando  $c = c_1 \cdot c_2$  y  $n_0 = \max\{n_1, n_2\}$ , se tiene que:

$$T1(n) \cdot T2(n) \leq c_1 \cdot f(n) \cdot c_2 \cdot g(n) = c \cdot f(n) \cdot g(n) \quad \forall n \geq n_0,$$



**Ejercicio 1.1.2.** Expresar, en notación  $O(\cdot)$ , el orden que tendr  un algoritmo cuyo tiempo de ejecuci n fuera  $f_i(n)$ , donde:

1.  $f_1(n) = n^2$

En este caso se tiene que  $f_1(n) \in O(n^2)$ .

2.  $f_2(n) = n^2 + 1000n$

En este caso, por la regla de la suma, se tiene que  $f_2(n) \in O(n^2)$ .

3.  $f_3(n) = \begin{cases} n & \text{si } n \text{ es par} \\ n^3 & \text{si } n \text{ es impar} \end{cases}$

En este caso, como  $n^3 \geq n$  para todo  $n \geq 1$ , se tiene que  $f_3(n) \in O(n^3)$ .

4.  $f_4(n) = \begin{cases} n & \text{si } n \leq 100 \\ n^3 & \text{si } n > 100 \end{cases}$

En este caso, como se trata de comportamientos asint ticos, se tiene que  $f_4(n) \in O(n^3)$ .

5.  $f_5(n) = (n - 1)^3$

En este caso, por la regla de la suma, se tiene que  $f_5(n) \in O(n^3)$ .

6.  $f_6(n) = \sqrt{n^2 - 1}$ .

En este caso, como  $\sqrt{n^2 - 1} \leq n$  para todo  $n \geq 1$ , se tiene que  $f_6(n) \in O(n)$ .

7.  $f_7(n) = \log(n!)$

Por el Criterio de Stolz, se tiene que:

$$\left\{ \frac{\log(n!)}{n \log n} \right\} = \left\{ \frac{\log(n+1)}{(n+1) \log(n+1) - n \log n} \right\} = \left\{ \frac{1}{n+1 - n \cdot \frac{\log(n)}{\log(n+1)}} \right\} \rightarrow \frac{1}{n+1 - n} = 1$$

Por tanto, tenemos que  $\log(n!) \in O(n \log n)$ .

8.  $f_8(n) = n!$

Claramente,  $f_8(n) \in O(n!)$ .

**Ejercicio 1.1.3.** Usando la notaci n  $O(\cdot)$ , obtener el tiempo de ejecuci n de las siguientes funciones:

1. C digo Fuente 1 (ejemplo1).

```
1 void ejemplo1 (int n)
2 {
3     int i, j, k;
4
5     for (i = 0; i < n; i++)
6         for (j = 0; j < n; j++)
7             {
8                 C[i][j] = 0;
9                 for (k = 0; k < n; k++)
10                     C[i][j] += A[j][k] * B[k][j];
11             }
12 }
```

Código fuente 1: Función del Ejercicio 1.1.3 apartado 1.

2. Código Fuente 2 (ejemplo2).

```
1 long ejemplo2 (int n)
2 {
3     int i, j, k;
4     long total = 0;
5
6     for (i = 0; i < n; i++)
7         for (j = i+1; j <= n; j++)
8             for (k = 1; k <= j; k++)
9                 total += k*i;
10
11     return total;
12 }
```

Código fuente 2: Función del Ejercicio 1.1.3 apartado 2.

3. Código Fuente 3 (ejemplo3).

```
1 void ejemplo3 (int n)
2 {
3     int i, j, x=0, y=0;
4
5     for (i = 1; i <= n; i++)
6         if (i % 2 == 1)
7             {
8                 for (j = i; j <= n; j++)
9                     x++;
10                for (j = 0; j < i; j++)
11                    y++;
12            }
13 }
```

Código fuente 3: Función del Ejercicio 1.1.3 apartado 3.

4. Código Fuente 4 (ejemplo4).

```
1 int ejemplo4 (int n)
2 {
3     if (n <= 1)
4         return 1;
5     else
6         return (ejemplo4(n - 1) + ejemplo4(n-1));
7 }
```

Código fuente 4: Función del Ejercicio 1.1.3 apartado 4.

5. Código Fuente 5 (ejemplo5).

```
1 int ejemplo5 (int n)
2 {
3     if (n == 1)
4         return n;
5     else
6         return (ejemplo5(n/2) + 1);
7 }
```

Código fuente 5: Función del Ejercicio 1.1.3 apartado 5.

**Ejercicio 1.1.4.** Resolver las siguientes recurrencias:

$$\text{a) } T(n) = \begin{cases} 0 & \text{si } n = 0 \\ 2T(n-1) + 1 & \text{en otro caso} \end{cases}$$

$$\text{b) } T(n) = \begin{cases} 0 & \text{si } n = 0 \\ 2T(n-1) + n & \text{en otro caso} \end{cases}$$

$$c) \quad T(n) = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ T(n-1) + T(n-2) & \text{en otro caso} \end{cases}$$

$$d) \quad T(n) = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ 3T(n-1) + 4T(n-2) & \text{en otro caso} \end{cases}$$

$$e) \quad T(n) = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ 5T(n-1) - 8T(n-2) + 4T(n-3) & \text{en otro caso} \end{cases}$$

$$f) \quad T(n) = \begin{cases} 0 & \text{si } n = 0 \\ 36 & \text{si } n = 1 \\ 5T(n-1) + 6T(n-2) + 4 \cdot 3^n & \text{en otro caso} \end{cases}$$

$$g) \quad T(n) = 2T(n-1) + 3^n.$$

$$h) \quad T(n) = 2T(n-1) + n + 2^n.$$

$$i) \quad T(n) = 2T(n/2) + \log n.$$

$$j) \quad T(n) = 4T(n/2) + n.$$

$$k) \quad T(n) = 4T(n/2) + n^2.$$

$$l) \quad T(n) = 2T(n/2) + n \log n.$$

$$m) \quad T(n) = \begin{cases} 1 & \text{si } n = 2 \\ 2T(\sqrt{n}) + \log n & \text{si } n \geq 4 \end{cases}$$

$$n) \quad T(n) = \begin{cases} 1 & \text{si } n = 2 \\ 2T(\sqrt{n}) + \log \log n & \text{si } n \geq 4 \end{cases}$$

$$o) \quad T(n) = \begin{cases} 1 & \text{si } n = 1 \\ 5T(n/2) + (n \log n)^2 & \text{si } n \geq 2 \end{cases}$$

$$p) \quad T(n) = \sqrt{n}T(\sqrt{n}) + n, \quad n \geq 4.$$

$$q) \quad T(n) = \begin{cases} 6 & \text{si } n = 1 \\ nT^2(n/2) & \text{si } n > 1 \end{cases}.$$

$$r) \quad T(n) = \begin{cases} 1 & \text{si } n = 1 \\ 4 & \text{si } n = 2 \\ T(n/2) \cdot T^2(n/2) & \text{si } n \geq 4 \end{cases}$$

**Ejercicio 1.1.5.** El tiempo de ejecución de un Algoritmo  $A$  viene descrito por la recurrencia

$$T(n) = 7T(n/2) + n^2$$

Otro algoritmo  $B$  tiene un tiempo de ejecución descrito por la recurrencia

$$T'(n) = aT'(n/4) + n^2$$

¿Cuál es el mayor valor de la constante  $a \in \mathbb{R}$  que hace al algoritmo  $B$  asintóticamente más eficiente que  $A$ ?

**Ejercicio 1.1.6.** Resuelva la siguiente recurrencia:

$$T(n) = aT\left(\frac{n}{b}\right) + n^k$$

con  $a, b, k \in \mathbb{R}$ ,  $a \geq 1$ ,  $b \geq 2$ ,  $k \geq 0$ .