

# Lógica, teoría de grafos y complejidad computacional



**Universidad**  
Internacional  
de Valencia  
**Máster Universitario**  
**en Inteligencia Artificial**

**02MIAR | Matemáticas:**  
Matemáticas para la Inteligencia Artificial

**Profesor:**  
David Zorío Ventura

- ▶ Recordemos que  $p \rightarrow q \equiv \bar{p} + q$ .
- ▶  $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$ , luego  $p \leftrightarrow q \equiv (\bar{p} + q) \cdot (p + \bar{q})$ , por lo que simplificando  $p \leftrightarrow q \equiv p \cdot q + \bar{p} \cdot \bar{q}$ .
- ▶ Cualquier fórmula lógica puede expresarse empleando como únicos conectores la conjunción (producto), la disyunción (suma) y la negación (**álgebras de Boole**).

## Definición

Una **función booleana** es una aplicación  $f : \mathbb{B}^n \rightarrow \mathbb{B}$ , donde  $\mathbb{B} = \{0, 1\}$  es el **conjunto binario**. Se denota  $\mathcal{F}_n$  al conjunto formado por todas las funciones booleanas de  $n$  variables, es decir:

$$\mathcal{F}_n = \{f : \mathbb{B}^n \rightarrow \mathbb{B} \mid f \text{ aplicación}\}.$$

## Ejemplos

1.  $f : \mathbb{B} \rightarrow \mathbb{B}, f(x) = x.$
2.  $f : \mathbb{B} \rightarrow \mathbb{B}, f(x) = \bar{x}.$
3.  $f : \mathbb{B}^2 \rightarrow \mathbb{B}, f(x, y) = x + y.$
4.  $f : \mathbb{B}^2 \rightarrow \mathbb{B}, f(x, y) = x \cdot y.$
5.  $f : \mathbb{B}^2 \rightarrow \mathbb{B}, f(x, y) = \bar{x} + y.$
6.  $f : \mathbb{B}^2 \rightarrow \mathbb{B}, f(x, y) = x \cdot y + \bar{x} \cdot \bar{y}.$
7.  $f : \mathbb{B}^3 \rightarrow \mathbb{B}, f(x, y, z) = \bar{x} \cdot z + y.$

## Definición

*Una **puerta lógica** es un operador que transforma uno o dos valores de entrada binarios,  $x, y$ , en un valor binario de salida, en función de  $x, y$ . Por tanto, una puerta lógica puede verse como elemento de  $\mathcal{F}_n$ .*

Existen un total de siete puertas lógicas:

- ▶ OR
- ▶ AND
- ▶ XOR
- ▶ NOT
- ▶ NOR
- ▶ NAND
- ▶ XNOR

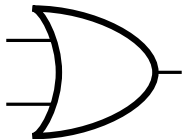
**Puerta OR:** se trata de la puerta lógica que combina un par de valores  $x, y \in \mathbb{B}$  según la tabla de operaciones siguiente:

OR	$y = 0$	$y = 1$
$x = 0$	0	1
$x = 1$	1	1

Esta puerta lógica puede interpretarse como la siguiente función booleana de  $\mathcal{F}_2$ :

$$f : \mathbb{B}^2 \rightarrow \mathbb{B}, \quad f(x, y) = x + y.$$

Se representa mediante el siguiente símbolo:





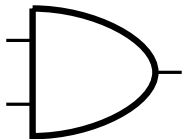
**Puerta AND:** se trata de la puerta lógica que combina un par de valores  $x, y \in \mathbb{B}$  según la tabla de operaciones siguiente:

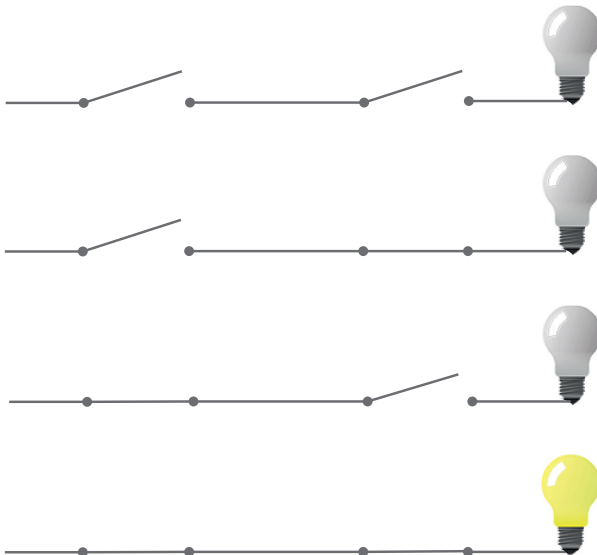
AND	$y = 0$	$y = 1$
$x = 0$	0	0
$x = 1$	0	1

Esta puerta lógica puede interpretarse como la siguiente función booleana de  $\mathcal{F}_2$ :

$$f : \mathbb{B}^2 \rightarrow \mathbb{B}, \quad f(x, y) = x \cdot y.$$

Se representa mediante el siguiente símbolo:







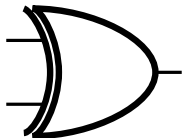
**Puerta XOR:** se trata de la puerta lógica que combina un par de valores  $x, y \in \mathbb{B}$  según la tabla de operaciones siguiente:

XOR	$y = 0$	$y = 1$
$x = 0$	0	1
$x = 1$	1	0

Esta puerta lógica puede interpretarse como la siguiente función booleana de  $\mathcal{F}_2$ :

$$f : \mathbb{B}^2 \rightarrow \mathbb{B}, \quad f(x, y) = (\bar{x} + \bar{y}) \cdot (x + y).$$

Se representa mediante el siguiente símbolo:



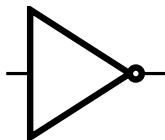
**Puerta NOT:** se trata de la puerta lógica que invierte el estado de un valor binario  $x \in \mathbb{B}$ :

NOT	
$x = 0$	1
$x = 1$	0

Esta puerta lógica puede interpretarse como la siguiente función booleana de  $\mathcal{F}_1$ :

$$f : \mathbb{B} \rightarrow \mathbb{B}, \quad f(x) = \bar{x}.$$

Se representa mediante el siguiente símbolo:



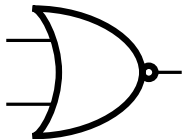
**Puerta NOR:** se trata de la puerta lógica que combina un par de valores  $x, y \in \mathbb{B}$  según la tabla de operaciones siguiente:

NOR	$y = 0$	$y = 1$
$x = 0$	1	0
$x = 1$	0	0

Esta puerta lógica puede interpretarse como la siguiente función booleana de  $\mathcal{F}_2$ :

$$f : \mathbb{B}^2 \rightarrow \mathbb{B}, \quad f(x, y) = \bar{x} \cdot \bar{y}.$$

Se representa mediante el siguiente símbolo:



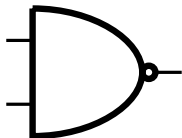
**Puerta NAND:** se trata de la puerta lógica que combina un par de valores  $x, y \in \mathbb{B}$  según la tabla de operaciones siguiente:

NAND	$y = 0$	$y = 1$
$x = 0$	1	1
$x = 1$	1	0

Esta puerta lógica puede interpretarse como la siguiente función booleana de  $\mathcal{F}_2$ :

$$f : \mathbb{B}^2 \rightarrow \mathbb{B}, \quad f(x, y) = \bar{x} + \bar{y}.$$

Se representa mediante el siguiente símbolo:



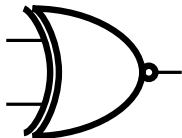
**Puerta XNOR:** se trata de la puerta lógica que combina un par de valores  $x, y \in \mathbb{B}$  según la tabla de operaciones siguiente:

XNOR	$y = 0$	$y = 1$
$x = 0$	1	0
$x = 1$	0	1

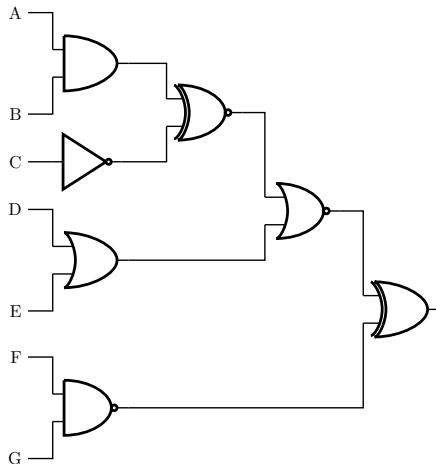
Esta puerta lógica puede interpretarse como la siguiente función booleana de  $\mathcal{F}_2$ :

$$f : \mathbb{B}^2 \rightarrow \mathbb{B}, \quad f(x, y) = x \cdot y + \bar{x} \cdot \bar{y}.$$

Se representa mediante el siguiente símbolo:

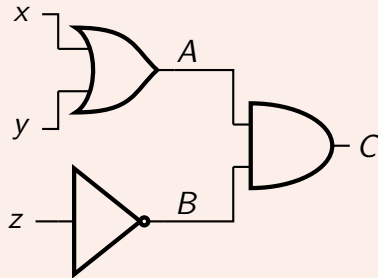
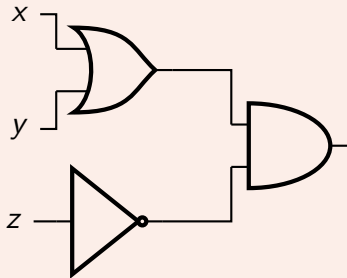


Las puertas lógicas pueden combinarse entre sí para formar **circuitos lógicos**, los cuales pueden interpretarse como elementos de  $\mathcal{F}_n$ .



## Ejemplo

Consideremos el circuito lógico siguiente:



La función booleana asociada a este circuito es  $f \in \mathcal{F}_3$  dada por

$$f(x, y, z) = (x + y) \cdot \bar{z}.$$

## Definición

Un **grafo**  $G$  es una terna  $(V, E, \gamma)$ , donde  $V$  y  $E$  son conjuntos y  $\gamma$  es una aplicación

$$\gamma : E \rightarrow \mathcal{E},$$

donde  $\mathcal{E} = \{\{v, v'\} \mid v, v' \in V\}$

- ▶ Los elementos del conjunto  $V$  reciben el nombre de **vértices**.
- ▶ Los elementos del conjunto  $E$  reciben el nombre de **lados** o **aristas**.
- ▶ La aplicación  $\gamma$  se conoce como **aplicación de incidencia**.
- ▶ Dicha aplicación identifica las aristas con el par correspondiente de vértices que unen.

Todo grafo tiene su correspondiente representación gráfica, como veremos en los siguientes ejemplos.



## Ejemplos

1. Consideremos el grafo  $G = (V, E, \gamma)$ , con  $V = \{v_1, v_2, v_3\}$ ,  $E = \{e_1, e_2\}$ , siendo  $\gamma : E \rightarrow \mathcal{E}$  dada por:

$$\gamma(e_1) = \{v_1, v_2\}$$

$$\gamma(e_2) = \{v_2, v_3\}$$



## Ejemplos

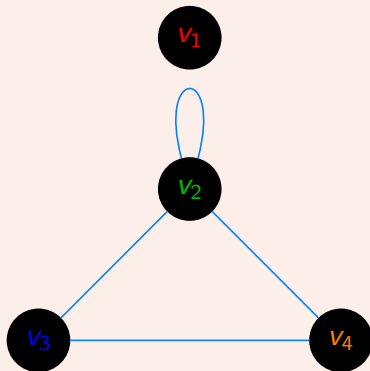
2. Consideremos el grafo  $G = (V, E, \gamma)$ , con  $V = \{v_1, v_2, v_3, v_4\}$ ,  $E = \{e_1, e_2, e_3, e_4\}$ , siendo  $\gamma : E \rightarrow \mathcal{E}$  dada por:

$$\gamma(e_1) = \{v_2, v_2\}$$

$$\gamma(e_2) = \{v_2, v_3\}$$

$$\gamma(e_3) = \{v_3, v_4\}$$

$$\gamma(e_4) = \{v_4, v_2\}$$



## Ejemplos

3. Consideremos el grafo  $G = (V, E, \gamma)$ , con  $V = \{v_1, v_2, v_3, v_4\}$ ,  $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$ , siendo  $\gamma : E \rightarrow \mathcal{E}$  dada por:

$$\gamma(e_1) = \{v_1, v_2\}$$

$$\gamma(e_2) = \{v_1, v_2\}$$

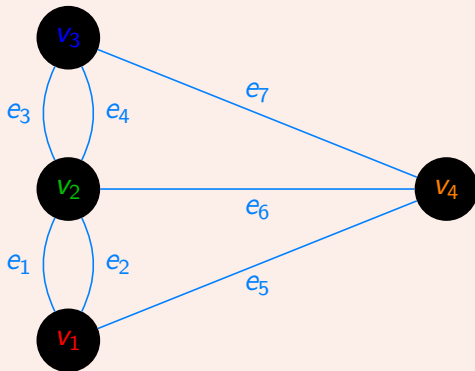
$$\gamma(e_3) = \{v_2, v_3\}$$

$$\gamma(e_4) = \{v_2, v_3\}$$

$$\gamma(e_5) = \{v_1, v_4\}$$

$$\gamma(e_6) = \{v_2, v_4\}$$

$$\gamma(e_7) = \{v_3, v_4\}$$



## Definición

Un **grafo dirigido**  $G$  es una terna  $(V, E, \gamma)$ , donde  $V$  y  $E$  son conjuntos (de vértices y de aristas o lados, respectivamente) y  $\gamma$  es una aplicación de la forma

$$\gamma : E \rightarrow V \times V.$$

Los grafos dirigidos habitualmente se representan gráficamente dibujando las aristas con una flecha en la dirección válida para recorrerla, siendo la primera componente del par ordenado el punto de partida y la segunda componente el punto de llegada.

## Ejemplo

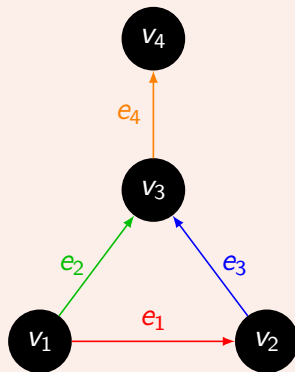
Sea  $G = (V, E, \gamma)$  el grafo dirigido dado por el conjunto de vértices  $V = \{v_1, v_2, v_3, v_4\}$ , el conjunto de aristas  $E = \{e_1, e_2, e_3, e_4\}$  y  $\gamma : E \rightarrow V \times V$  dada por:

$$\gamma(e_1) = (v_1, v_2),$$

$$\gamma(e_2) = (v_1, v_3),$$

$$\gamma(e_3) = (v_2, v_3),$$

$$\gamma(e_4) = (v_3, v_4).$$



## Definición

Un **grafo simple**  $G = (V, E, \gamma)$  es un grafo que cumple dos restricciones adicionales:

1. No hay aristas diferentes que unan el mismo par de vértices, es decir, si  $e \neq e'$ , entonces  $\gamma(e) \neq \gamma(e')$  ( $\gamma$  es inyectiva).
2. No contiene **bucles** o **loops**, es decir, no tiene aristas que unen un vértice consigo mismo; equivalentemente,  $\gamma(e) = \{v_e, v'_e\}$ , con  $v_e \neq v'_e, \forall e \in E$ .

## Nota

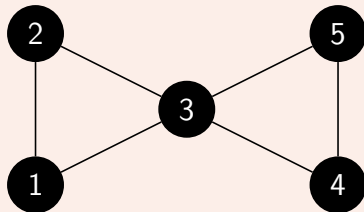
En adelante, y por comodidad, en caso de trabajar con grafos simples, omitiremos la existencia de la aplicación  $\gamma$  y denotaremos las aristas directamente por el correspondiente conjunto o par ordenado de vértices.

## Definición

Un grafo  $G$  es **conexo** si para cada par de vértices diferentes existe algún **camino** (sucesión de aristas adyacentes) que los une.

## Ejemplos

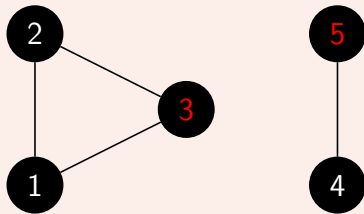
1. Sea  $G$  un grafo de 5 vértices dado por las aristas  $\{1, 2\}, \{1, 3\}, \{2, 3\}, \{3, 4\}, \{3, 5\}, \{4, 5\}$ . Gráficamente:



$G$  es conexo.

## Ejemplos

2. Sea  $G$  un grafo de 5 vértices cuyas aristas son  $\{1, 2\}$ ,  $\{1, 3\}$ ,  $\{2, 3\}$ ,  $\{4, 5\}$ .  
Gráficamente:



Entonces  $G$  no es conexo.



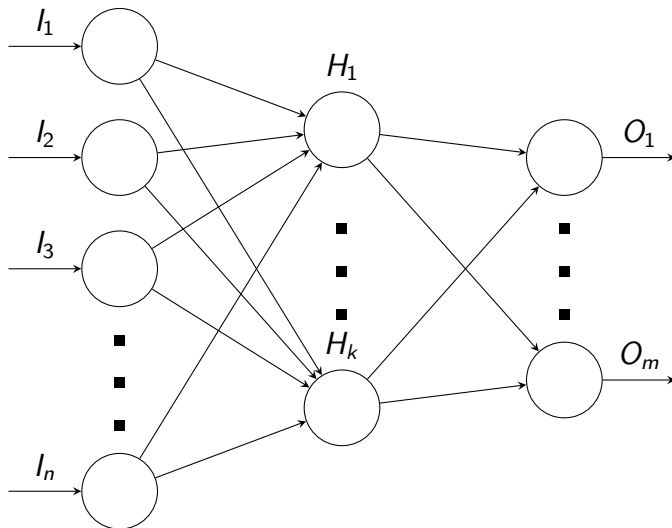
## Definición

Una **red neuronal** es un grafo dirigido  $(V, E)$ , con  $V = \{1, 2, \dots, n\}$  y  $E \subseteq V \times V$  con los elementos adicionales siguientes:

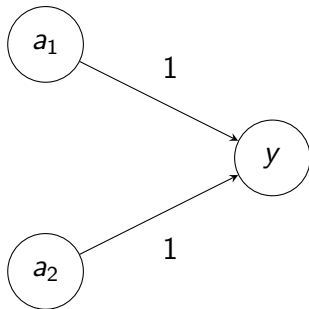
- ▶  $\forall i \in V$  tiene asociada una variable de estado  $a_i \in [0, 1]$ .
- ▶  $\forall (i, j) \in E$  tiene asociado un peso  $\omega_{i,j} \in \mathbb{R}$  (**grafo ponderado**).
- ▶  $\forall i \in V$  tiene asociado un **sesgo**  $b_i$ .
- ▶  $\forall i \in V$  le corresponde una **función de activación**  $f_i : \mathbb{R} \rightarrow [0, 1]$  de la forma

$$f_i \left( \sum_{j=1}^n \omega_{j,i} a_j + b_i \right).$$

Capa de entrada    Capas ocultas    Capa de salida

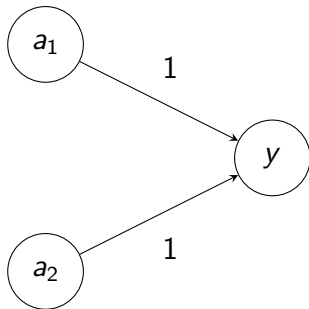


## Ejemplo 1: neurona AND



$$y = f(a_1 + a_2 - 1.5), \quad f(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$$

## Ejemplo 2: neurona OR



$$y = f(a_1 + a_2 - 0.5), \quad f(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$$

La teoría de **complejidad computacional** estudia y clasifica los algoritmos y problemas computacionales en función de su coste y dificultad de resolver.

---

**Algorithm** Compute  $p = u \cdot v$  for  $u, v \in \mathbb{R}^n$

---

**Require:**  $\text{length}(u) = \text{length}(v)$

**Ensure:**  $p = u \cdot v$

$p \leftarrow u_1 v_1$

**for**  $k \leftarrow 2$  to  $n$  **do**

$p \leftarrow p + u_k v_k$

**end for**

**return**  $p$

---

- ▶ **Operaciones:**  $n - 1$  sumas,  $n$  productos.
- ▶ **Total operaciones:**  $2n - 1$  operaciones (coste **lineal**).

---

**Algorithm** Compute  $C = A + B$  for  $A, B \in \mathbb{R}^{n \times n}$

---

**Require:**  $\text{size}(A) = \text{size}(B)$

**Ensure:**  $C = A + B$

$C \leftarrow 0_{n \times n}$

**for**  $i \leftarrow 1$  to  $n$  **do**

**for**  $j \leftarrow 1$  to  $n$  **do**

$C_{i,j} \leftarrow A_{i,j} + B_{i,j}$

**end for**

**end for**

**return**  $C$

---

- ▶ **Operaciones:**  $n^2$  sumas.
- ▶ **Total operaciones:**  $n^2$  operaciones (coste **cuadrático**).

---

**Algorithm** Compute  $C = A \cdot B$  for  $A, B \in \mathbb{R}^{n \times n}$

---

**Require:**  $\text{size}(A) = \text{size}(B)$

**Ensure:**  $C = A \cdot B$

$C \leftarrow 0_{n \times n}$

**for**  $i \leftarrow 1$  to  $n$  **do**

**for**  $j \leftarrow 1$  to  $n$  **do**

**for**  $k \leftarrow 1$  to  $n$  **do**

$C_{i,j} \leftarrow C_{i,j} + A_{i,k}B_{k,j}$

**end for**

**end for**

**end for**

**return**  $C$

---

- ▶ **Operaciones:**  $n^3$  sumas,  $n^3$  productos.
- ▶ **Total operaciones:**  $2n^3$  operaciones (coste **cúbico**).

Resolución de un sistema de ecuaciones lineal  $n \times n$  con matriz de coeficientes triangular inferior:

$$\left. \begin{array}{l} a_{1,1}x_1 = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 = b_2 \\ \vdots \\ a_{n-1,1}x_1 + a_{n-1,2}x_2 + \cdots + a_{n-1,n-1}x_{n-1} = b_{n-1} \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n-1}x_{n-1} + a_{n,n}x_n = b_n \end{array} \right\}$$

Matricialmente:  $AX = B$ , donde  $A \in \mathbb{R}^{n \times n}$ ,  $X \in \mathbb{R}^{n \times 1}$  y  $B \in \mathbb{R}^{n \times 1}$ :

$$A = \begin{pmatrix} a_{1,1} & 0 & \cdots & 0 & 0 \\ a_{2,1} & a_{2,2} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,n-1} & 0 \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n-1} & a_{n,n} \end{pmatrix}, X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}, B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}.$$



---

**Algorithm** Solve  $AX = B$  for  $A \in \mathbb{R}^{n \times n}$ ,  $X, B \in \mathbb{R}^{n \times 1}$

---

**Require:**  $\det(A) \neq 0$

**Ensure:**  $AX = B$

```
for  $i \leftarrow 1$  to  $n$  do
   $X_i \leftarrow 0$ 
  for  $j \leftarrow 1$  to  $i - 1$  do
     $X_i \leftarrow X_i - A_{i,j}X_j$ 
  end for
   $X_i \leftarrow \frac{X_i}{A_{i,i}}$ 
end for
return  $X$ 
```

- 
- ▶ **Operaciones:**  $(n - 1)n$  sumas,  $(n - 1)n$  productos,  $n$  divisiones.
  - ▶ **Total operaciones:**  $2n^2 - n$  operaciones (coste **cuadrático**).

## Definición

Diremos que el coste computacional de un algoritmo es **polinomial** si el coste en función del tamaño de la entrada,  $n$ , es un polinomio.

## Definición

Diremos que el coste computacional de un algoritmo con  $c(n)$  operaciones es  $\mathcal{O}(n^k)$ ,  $k \in \mathbb{N}$ , si

$$\exists \lim_{n \rightarrow +\infty} \frac{c(n)}{n^k} = C, \quad C \in ]0, +\infty[.$$

## Ejemplos

1. El coste computacional del producto escalar, con  $2n - 1$  operaciones, es  $\mathcal{O}(n)$ :

$$\lim_{n \rightarrow +\infty} \frac{2n - 1}{n} = 2.$$

## Ejemplos

2. El coste computacional de la suma de matrices cuadradas, con  $n^2$  operaciones, es  $\mathcal{O}(n^2)$ :

$$\lim_{n \rightarrow +\infty} \frac{n^2}{n^2} = 1.$$

3. El coste computacional del producto de matrices cuadradas, con  $2n^3$  operaciones, es  $\mathcal{O}(n^3)$ :

$$\lim_{n \rightarrow +\infty} \frac{2n^3}{n^3} = 2.$$

4. El coste computacional de la resolución de un sistema triangular, con  $2n^2 - n$  operaciones, es  $\mathcal{O}(n^2)$ :

$$\lim_{n \rightarrow +\infty} \frac{2n^2 - n}{n^2} = 2.$$

---

**Algorithm** Compute  $\det(A)$  for  $A \in \mathbb{R}^{n \times n}$

---

**Require:**  $A$  is a square matrix

**Ensure:**  $d = \det(A)$

$n \leftarrow \dim(A)$

**if**  $n = 1$  **then**

$d \leftarrow A_{1,1}$

**else**

$d \leftarrow 0$

**for**  $j \leftarrow 1$  to  $n$  **do**

$d \leftarrow d + (-1)^{1+j} A_{1,j} \det(A_{\overline{\{1\}}, \overline{\{j\}}})$

**end for**

**end if**

**return**  $d$

---

$2 \cdot n!$  operaciones.

- ▶ Se tiene que  $\forall k \in \mathbb{N}$

$$\lim_{n \rightarrow +\infty} \frac{2 \cdot n!}{n^k} = +\infty.$$

- ▶ Necesitamos ampliar la noción de  $\mathcal{O}$ .

## Definición

Sea  $f : \mathbb{N} \rightarrow \mathbb{R}$  y  $c(n)$  el número de operaciones requeridas por un algoritmo. Diremos que el coste computacional del algoritmo es  $\mathcal{O}(f(n))$  si

$$\lim_{n \rightarrow +\infty} \frac{c(n)}{f(n)} = C, \quad C \in ]0, +\infty[.$$

## Ejemplos

1. El coste computacional del algoritmo que calcula el determinante es  $\mathcal{O}(n!)$ :

$$\lim_{n \rightarrow +\infty} \frac{2 \cdot n!}{n!} = 2.$$

2. El coste computacional de un algoritmo con  $2^n + n^6 - 2n^4 + 3n + 7$  operaciones es  $\mathcal{O}(2^n)$ :

$$\lim_{n \rightarrow +\infty} \frac{2^n + n^6 - 2n^4 + 3n + 7}{2^n} = 1.$$

3. Algoritmos de ordenación eficientes:  $\mathcal{O}(n \log(n))$ .

## Definición

Un **problema de decisión** consiste en un problema cuya respuesta se puede plantear como un “sí” o un “no” en función de sus entradas.

## Ejemplos

1. Algoritmo que determina si un número dado es par o impar.
2. Algoritmo que determina si un número dado es primo o no.
3. Algoritmo para determinar si un grafo es o no es conexo.
4. Algoritmo que determina si un Sudoku tiene solución o no.
5. Algoritmo para determinar si existe un subconjunto de un determinado subconjunto numérico cuyos elementos suman 0.

## Definición

*Definimos como **P** (polynomial time) el conjunto formado por todos los problemas de decisión que pueden resolverse de forma determinista con un coste computacional polinomial.*

## Ejemplos

- 1. Algoritmo que determina si un número dado es par o impar. Coste computacional:  $\mathcal{O}(1)$ .*
- 2. Algoritmo que determina si una matriz es o no es simétrica. Coste computacional:  $\mathcal{O}(n^2)$ .*
- 3. Algoritmo que determina si se puede encontrar un camino en un grafo simple que recorre todos sus vértices pasando por todas sus aristas una única vez (caminos/ciclos eulerianos). Coste computacional:  $\mathcal{O}(n^2)$ .*



## Definición

Definimos como **NP** (*non-deterministic polynomial time*) el conjunto formado por todos los problemas de decisión que pueden resolverse de forma no determinista con un coste computacional polinomial.

Equivalentemente, **NP** es el conjunto formado por todos los problemas de decisión cuyas instancias pueden verificarse con un coste computacional polinomial.

## Ejemplos

1. *Sudoku.*
2. *Hallar el subconjunto de un determinado subconjunto numérico cuyos elementos suman 0.*
3. *Cualquier problema en **P** ( $\mathbf{P} \subseteq \mathbf{NP}$ ).*

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

- ▶ Sea  $A$  el conjunto dado por

$$A = \{-11, 8, 7, 1, -2, 52, -14, 0, -10, 23\}.$$

- ▶ Entonces el subconjunto  $B \subseteq A$  dado por

$$B = \{1, -14, -10, 23\}$$

verifica que la suma de sus elementos es 0.

- ▶ Coste computacional de la verificación:  $\mathcal{O}(n)$ .
- ▶ En el caso general, un conjunto  $A$  verificando  $|A| = n$ , posee  $|\mathcal{P}(A)| = 2^n$  subconjuntos, luego si se procede por fuerza bruta, en el peor de los casos habría que realizar

$$\sum_{k=1}^n (k-1) \binom{n}{k} \text{ sumas} \sim \mathcal{O}(n \cdot 2^n) \text{ operaciones.}$$

## PROBLEMA DEL MILLÓN:

**$\text{¿P} = \text{NP?}$**

# ¡Muchas gracias!



**Universidad  
Internacional  
de Valencia**

**Contacto:**

[david.zorio@campusviu.es](mailto:david.zorio@campusviu.es)

