Complejidad algorítmica

Algoritmos y Estructuras de Datos II

2^{do} cuatrimestre 2019

Menú del día

¿Complejidad algorítmica?

- ¿Qué es?
- ¿Para qué se usa?
- ¿Cuál es el tamaño de la entrada?

Análisis de algoritmos

Algoritmo (parametro_de_entrada: α)	
operación elemental	$\triangleright$ c_0
otra operación elemental	$\triangleright$ c_1
OE restantes del algoritmo	$\triangleright$ costo

Análisis de algoritmos

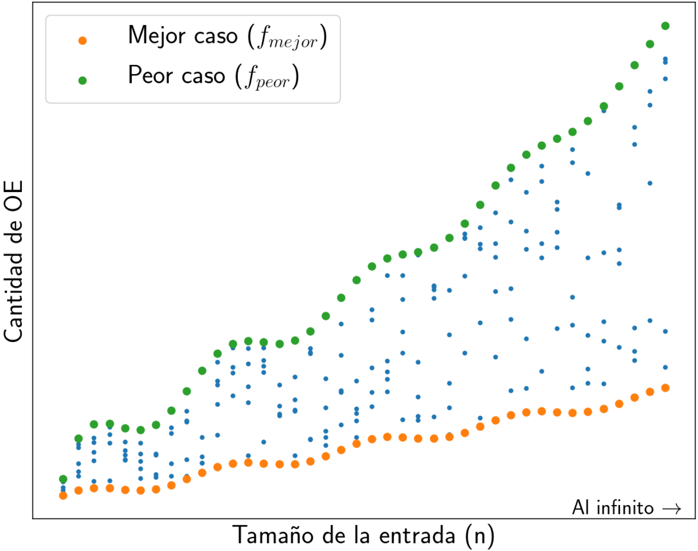
Algoritmo (parametro_de_entrada: α)

operación elemental	$\triangleright$ 1
otra operación elemental	$\triangleright$ 1
OE restantes del algoritmo	$\triangleright$ cantidad OE restantes

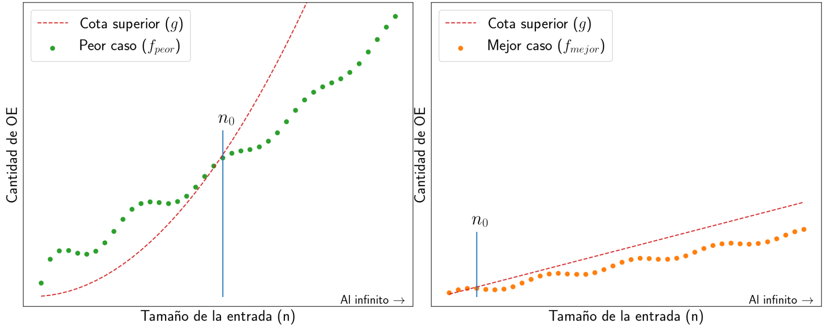
Recordar

- Advertencia: el gráfico es solo para ganar intuición.
- NO ANALIZAMOS ALGORITMOS CON GRÁFICOS.
- Nos centramos en el **análisis teórico** (que se realiza con demostraciones y aplicación de propiedades).

Cantidad de OE para distintas **instancias** de tamaño *n*



Función de peor caso y mejor casoBig $\mathcal{O}$ – Acotación superior



Definición

Sea $g : \mathbb{N} \to \mathbb{R}$. Entonces:

$$\mathcal{O}(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid (\exists\ n_0 \in \mathbb{N},\ c \in \mathbb{R}_{>0})\ \ f(n) \leq c \cdot g(n)\ \ \forall n \geq n_0\}$$

Big $\mathcal{O}$ – Acotación superior

Definición

Sea $g : \mathbb{N} \to \mathbb{R}$. Entonces:

$$\mathcal{O}(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid (\exists\, n_0 \in \mathbb{N},\ c \in \mathbb{R}_{>0})\ \ f(n) \le c \cdot g(n)\ \ \forall n \ge n_0\}$$

⚠ $\mathcal{O}(g)$ denota un **conjunto de funciones**

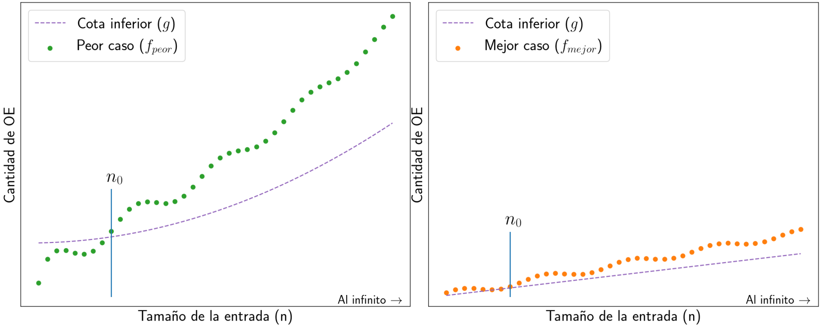
⚠ Para poder decir que una función pertenece a una clase de funciones (ej. $f_{peor} \in \mathcal{O}(g)$), hay que **demostrarlo**.

Propiedad 1: Dada $f : \mathbb{N} \to \mathbb{R}_{>0}$, $f \in \mathcal{O}(f)$.

Propiedad 2: Dada $f : \mathbb{N} \to \mathbb{R}_{>0}$ y k cte. (positiva). Si

$$f \in \mathcal{O}(g) \Rightarrow k \cdot f \in \mathcal{O}(g)$$

Función de peor caso y mejor casoΩ – Acotación inferior

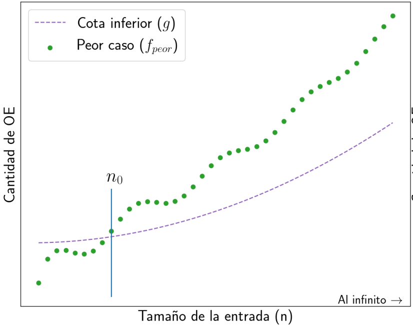


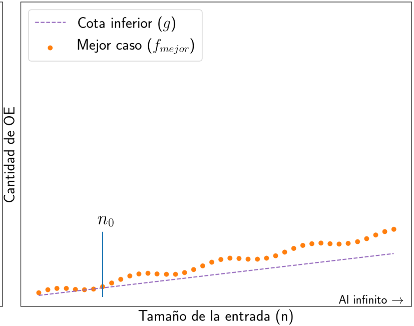
Definición

Sea $g : \mathbb{N} \to \mathbb{R}$. Entonces:

$$\Omega(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid (\exists\ n_0 \in \mathbb{N},\ c \in \mathbb{R}_{>0})\ \ f(n) \geq c \cdot g(n)\ \ \forall n \geq n_0\}$$

Función de peor caso y mejor casoΩ – Acotación inferior





Definición alternativa

Si $g : \mathbb{N} \to \mathbb{R}$, entonces:

$$\Omega(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid \ g \in O(f)\}$$

Ω – Acotación inferior

Definición

Sea $g : \mathbb{N} \to \mathbb{R}$. Entonces:

$$\Omega(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid (\exists\ n_0 \in \mathbb{N},\ c \in \mathbb{R}_{>0})\ \ f(n) \geq c \cdot g(n)\ \ \forall n \geq n_0\}$$

Definición alternativa

Si $g : \mathbb{N} \to \mathbb{R}$, entonces:

$$\Omega(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid\ g \in O(f)\}$$

Propiedad 1: Dada $f : \mathbb{N} \to \mathbb{R}_{>0}$, $f \in \Omega(f)$.

Propiedad 2: Dada $f : \mathbb{N} \to \mathbb{R}_{>0}$ y k cte. (positiva). Si

$$f \in \Omega(g) \Rightarrow k \cdot f \in \Omega(g)$$

Demos de tarea.

Θ – Acotación exacta

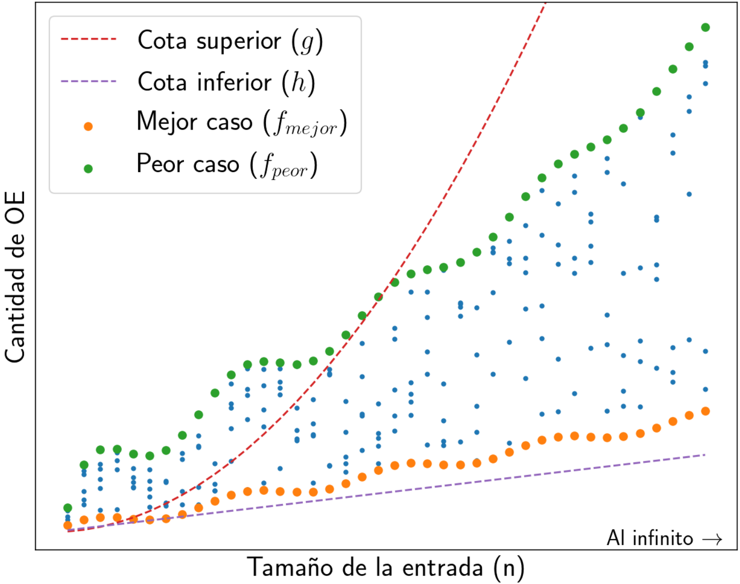
Definición

Sea $g : \mathbb{N} \to \mathbb{R}$. Entonces:

$$\Theta(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid f \in O(g) \wedge f \in \Omega(g)\}$$

- Si podemos acotar con funciones de la misma familia (solo variando el c) tanto superior como inferiormente tenemos que es Θ.
- Valen las propiedades mencionadas anteriormente para $\mathcal{O}$ y Ω.
- Observación: $\Theta(g) = O(g) \cap \Omega(g)$.

Análisis de un algoritmo – Observaciones



Análisis de un algoritmo

Para analizar algoritmos, podemos:

- **encontrar** las funciones de **peor caso** (f_{peor}) y **mejor caso** (f_{mejor})
- buscar y **proponer funciones para acotar** superior y/o inferiormente.
- **demostrar** que existe un múltiplo de la función propuesta que es efectivamente una cota asintótica.

Demostrando y usando **propiedades** simplificamos el análisis de funciones y por ende los algoritmos.

Primeros pasos – Análisis del **mejor** caso

Precondición: $|A| > 0$ (arreglo no vacío)

$\textsc{BusquedaSecuencial}(A : \texttt{arreglo(nat)}, e : \texttt{nat}) \longrightarrow \texttt{bool}$

```
var i : nat, n : nat
n ← tam(A)
i ← 0                                  ▷ línea 2+3: 2+1
mientras i < n ∧ A[i] ≠ e hacer        ▷ evaluar la guarda: 4
    i ← i + 1                          ▷ No ejecuta
devolver (i < n)                       ▷ 2
```

- $f_{mejor}(n)$ y $f_{peor}(n)$: **cantidad de operaciones** realizadas para un arreglo de tamaño n en mejor y peor caso respectivamente.
- $f_{mejor}(n) = 3 + 4 + 2 = 9 = k_{mejor}$

Análisis mejor caso

$$f_{mejor}(n) = 9 = k_{mejor}$$

- Acotemos inferiormente. $f_{mejor}(n) \in \Omega(1)$
- Acotemos superiormente. $f_{mejor}(n) \in \mathcal{O}(1)$
- $f_{mejor}(n) \in \Theta(1)$

Definición

Sea $g : \mathbb{N} \to \mathbb{R}$. Entonces:

$$\Omega(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid (\exists\ n_0 \in \mathbb{N},\ c \in \mathbb{R}_{>0})\ \ f(n) \geq c \cdot g(n)\ \ \forall n \geq n_0\}$$

Análisis mejor caso

$$f_{mejor}(n) = 9 = k_{mejor}$$

- Acotemos inferiormente. $f_{mejor}(n) \in \Omega(1)$
- Acotemos superiormente. $f_{mejor}(n) \in \mathcal{O}(1)$
- $f_{mejor}(n) \in \Theta(1)$

Definición

Sea $g : \mathbb{N} \to \mathbb{R}$. Entonces:

$$\mathcal{O}(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid (\exists\ n_0 \in \mathbb{N},\ c \in \mathbb{R}_{>0})\ \ f(n) \leq c \cdot g(n)\ \ \forall n \geq n_0\}$$

Análisis mejor caso

$$f_{mejor}(n) = 9 = k_{mejor}$$

- Acotemos inferiormente. $f_{mejor}(n) \in \Omega(1)$
- Acotemos superiormente. $f_{mejor}(n) \in \mathcal{O}(1)$
- $f_{mejor}(n) \in \Theta(1)$

Definición

Sea $g : \mathbb{N} \to \mathbb{R}$. Entonces:

$$\Theta(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid f \in O(g) \wedge f \in \Omega(g)\}$$

Primeros pasos – Análisis el **peor** caso

Precondición: $|A| > 0$ (arreglo no vacío)

$\textsc{BusquedaSecuencial}(A : \texttt{arreglo(nat)}, e : \texttt{nat}) \longrightarrow \texttt{bool}$

```
var i : nat, n : nat
n ← tam(A)
i ← 0                                  ▷ línea 2+3: 3
mientras i < n ∧ A[i] ≠ e hacer        ▷ ciclo:(n + 1) · 4 + 2 · n
    i ← i + 1                          ▷ 2
devolver (i < n)                       ▷ 2
```

- $f_{mejor}(n)$ y $f_{peor}(n)$: **cantidad de operaciones** realizadas para un arreglo de tamaño n en mejor y peor caso respectivamente.
- $f_{peor}(n) = 3 + (n+1)\ 4 + 2n + 2 = \ 3 + 4 + 2 + \ (4 + 2)\ n$
  $= 9 \ + \ 6 \ n$

Análisis peor caso

$$f_{peor}(n) = 9 \ + \ 6 \ n$$

- Acotemos superiormente. $f_{peor}(n) \in \mathcal{O}(n)$
- Acotemos inferiormente. $f_{peor}(n) \in \Omega(n)$
- $f_{peor}(n) \in \Theta(n)$

Definición

Sea $g : \mathbb{N} \to \mathbb{R}$. Entonces:

$$\mathcal{O}(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid (\exists \ n_0 \in \mathbb{N}, \ c \in \mathbb{R}_{>0}) \ \ f(n) \leq c \cdot g(n) \ \ \forall n \geq n_0\}$$

Análisis peor caso

$$f_{peor}(n) = 9 + 6\ n$$

- Acotemos superiormente. $f_{peor}(n) \in \mathcal{O}(n)$
- Acotemos inferiormente. $f_{peor}(n) \in \Omega(n)$
- $f_{peor}(n) \in \Theta(n)$

Definición

Sea $g : \mathbb{N} \to \mathbb{R}$. Entonces:

$$\Omega(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid (\exists\ n_0 \in \mathbb{N},\ c \in \mathbb{R}_{>0})\ \ f(n) \geq c \cdot g(n)\ \ \forall n \geq n_0\}$$

Análisis peor caso

$$f_{peor}(n) = 9 \ + \ 6 \ n$$

- Acotemos superiormente. $f_{peor}(n) \in \mathcal{O}(n)$
- Acotemos inferiormente. $f_{peor}(n) \in \Omega(n)$
- $f_{peor}(n) \in \Theta(n)$

Definición

Sea $g : \mathbb{N} \to \mathbb{R}$. Entonces:

$$\Theta(g) \stackrel{\text{def}}{=} \{f : \mathbb{N} \to \mathbb{R} \mid f \in O(g) \wedge f \in \Omega(g)\}$$

Análisis de BusquedaSecuencial

$$f_{mejor}(n) \in \Theta(1)$$
$$f_{peor}(n) \in \Theta(n)$$

¿Una función siempre es Θ de su cota?

Solamente si se trata de la cota (superior o inferior) más ajustada, es decir, del mismo orden asintótico que la función.

Análisis de BusquedaSecuencial

$\textsc{BusquedaSecuencial}(A : \texttt{arreglo(nat)}, e : \texttt{nat}) \longrightarrow \texttt{bool}$

Complejidad: $\Omega(1)$ y $\mathcal{O}(n)$ con $n = tam(A)$.

(Abusos de) notación

En lugar de "$f \in \mathcal{O}(g)$" a veces notamos:

$$f(n) = \mathcal{O}(g(n)) \qquad f(n) = \mathcal{O}(g) \qquad f \in \mathcal{O}(g(n))$$

En lugar de "$f \notin \mathcal{O}(g)$" a veces notamos:

$$f(n) \neq \mathcal{O}(g(n)) \qquad f(n) \neq \mathcal{O}(g) \qquad f \notin \mathcal{O}(g(n))$$

Álgebra de órdenes

1. **Suma.** $\mathcal{O}(f) + \mathcal{O}(g) \overset{\mathrm{def}}{=} \mathcal{O}(f+g)$ $\overset{\mathrm{prop}}{=} \mathcal{O}(\text{máx}\{f,g\})$.
2. **Producto.** $\mathcal{O}(f) \cdot \mathcal{O}(g) \overset{\mathrm{def}}{=} \mathcal{O}(f \cdot g)$.

Vale también para Ω, Θ.

Analicemos nuevamente el peor caso

$\textsc{BusquedaSecuencial}(\ A : \texttt{arreglo(nat)},\ e : \texttt{nat}) \longrightarrow \texttt{bool}$

```
1: var i : nat, n : nat
2: n ← tam(A)
3: i ← 0                                  ▷ Línea 2-3: 3
4: mientras i < n ∧ A[i] ≠ e hacer        ▷ (n + 1) · 4 + n · 2
5:     i ← i + 1                          ▷ 2
6: devolver (i < n)                       ▷ 2
```

Analicemos nuevamente el peor caso

BusquedaSecuencial(A : `arreglo(nat)`, e : `nat`) $\longrightarrow$ `bool`

1: **var** $i : nat, n : nat$
2: $n \leftarrow tam(A)$
3: $i \leftarrow 0$ ▷ Línea 2-3: $\Theta(1)$
4: **mientras** $i < n \wedge A[i] \neq e$ **hacer** ▷ $\Theta(n)$
5: $\quad i \leftarrow i + 1$ ▷ $\Theta(1)$
6: **devolver** $(i < n)$ ▷ $\Theta(1)$

$$f_{ciclo}(n) = (n+1) \cdot f_{guarda}(n) + n \cdot f_{interiorc}(n) =$$

$$\Theta(max\{n, 1\}) \cdot \Theta(1) + \Theta(n) \cdot \Theta(1) = \Theta(n) + \Theta(n) = \Theta(n)$$

Finalmente:

$$f_{peor}(n) = \Theta(1) + \Theta(n) + \Theta(1) = \Theta(n)$$

Busqueda secuencial en matrices – Análisis de **peor caso**

Precondición: M es matriz cuadrada de **n** filas por **n** columnas, $n > 0$.

$\textsc{BusquedaSecMatriz}(M : \texttt{arreglo(arreglo(nat))}, e : \texttt{nat})$

1: **var** $i : nat$, $j : nat$, $encontrado : bool$, $n : nat$
2: $n \leftarrow tam(M[0])$
3: $i \leftarrow 0$
4: $encontrado \leftarrow false$ $\quad\triangleright$ Líneas 2 a 4:$\Theta(1)$
5: **mientras** $i < n \ \wedge \neg encontrado$ **hacer** $\quad\triangleright\ n \cdot \Theta(n) = \Theta(n^2)$
6: $\qquad encontrado = \textsc{busquedaSecuencial}(\mathrm{M[i], e})$ $\quad\triangleright\ \Theta(n)$
7: $\qquad i \leftarrow i + 1$ $\quad\triangleright\ \Theta(1)$
8: **devolver** $encontrado$ $\quad\triangleright\ \Theta(1)$

Abstraemos y reusamos la función y análisis anterior.

$$f_{peor}(n) = \Theta(1) + \Theta(n^2) + \Theta(1) = \Theta(n^2)$$

Suma especial

SUMAESPECIAL(A: `arreglo(nat)`)

1: **var** $i : nat,\ n : nat,\ suma : nat$
2: $n \leftarrow tam(A)$
3: $i \leftarrow 1$
4: $suma \leftarrow 0$ ▷ Líneas 2 a 4:$\Theta(1)$
5: **mientras** $i < n$ **hacer** ▷ $\Theta(\log(n)) \cdot \Theta(1) = \Theta(\log(n))$
6: $\quad suma \leftarrow suma + A[i]$ ▷ $\Theta(1)$
7: $\quad i \leftarrow i \cdot 2$ ▷ $\Theta(1)$
8: **devolver** $suma$ ▷ $\Theta(1)$

¿Cuál es el mejor caso? ¿y el peor? Los casos coinciden.
¿Cuánto cuesta el ciclo?

$$f_{ciclo}(n) = \Theta(1) + \cdots + \Theta(1) = \Theta(\log(n))$$

$$f(n) = \Theta(1) + \Theta(\log(n)) + \Theta(1) = \Theta(\log(n))$$

Propiedades – $\Diamond$ es "comodín" de $\mathcal{O}, \Omega, \Theta$

1. Toda f cumple $f \in \Diamond(f)$. Reflexiva
2. $f \in \Diamond(g) \implies c \cdot f \in \Diamond(g)$ (con c cte.)
3. Regla de la suma:
   $f_1 \in \Diamond(g) \wedge f_2 \in \Diamond(h) \implies f_1 + f_2 \in \Diamond(g+h) = \Diamond(\max\{g, h\})$
4. Regla del producto:
   $f_1 \in \Diamond(g) \wedge f_2 \in \Diamond(h) \implies f_1 \cdot f_2 \in \Diamond(g \cdot h)$

3 y 4 corresponden al **álgebra de órdenes**. Además 4 implica 2.

- $f \in \Diamond(g) \wedge g \in \Diamond(h) \implies f \in \Diamond(h)$ Transitiva
- $f \in \Diamond(g) \implies \Diamond(f) \subseteq \Diamond(g)$
- $\Diamond(f) = \Diamond(g) \iff f \in \Diamond(g) \wedge g \in \Diamond(f)$

  Como $f \in \Theta(g) \implies g \in \Theta(f)$ Simétrica
  - $\Theta(f) = \Theta(g) \iff f \in \Theta(g)$

Ejercicios

Decidir si son verdaderas o falsas y justificar:

1. $2^n = \mathcal{O}(1)$ **F**
2. $n + \log^2 n \in \mathcal{O}(n + \log n)$
3. Dados $i, j \in \mathbb{N}$ fijos se tiene que $i \cdot n = \mathcal{O}(j \cdot n)$.
4. $\Omega(n) \subset \mathcal{O}(n^2)$
5. $\mathcal{O}(n^2) \subset \Omega(n)$
6. Si $f(n) = \mathcal{O}(n)$ entonces $2^{f(n)} = \mathcal{O}(2^n)$
7. $\Theta(n \log(n)) = \Theta(\log(n!))$

Propiedad

Dadas las funciones $f,\ g : \mathbb{N} \to \mathbb{R}$. Si existe

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = k$$

Podemos decir:

- Si $k \neq 0$ y $k < \infty$ entonces $\Theta(f) = \Theta(g)$.
- Si $k = 0$ entonces:
  - $f \in \mathcal{O}(g) \wedge g \notin \mathcal{O}(f)$ i.e. $\mathcal{O}(f) \subset \mathcal{O}(g)$ estrictamente.
  - $g \in \Omega(f) \wedge f \notin \Omega(g)$ i.e. $\Omega(g) \subset \Omega(f)$ estrictamente.
  - $\Theta(f) \neq \Theta(g)$.

TAREA: Convencerse de que los tres items para $k = 0$ son equivalentes.

Ejercicios

Decidir si son verdaderas o falsas y justificar:

1. $2^n = \mathcal{O}(1)$ **F**
2. $n + \log^2 n \in \mathcal{O}(n + \log n)$ **V**
3. Dados $i, j \in \mathbb{N}$ fijos se tiene que $i \cdot n = \mathcal{O}(j \cdot n)$. **V**
4. $\Omega(n) \subset \mathcal{O}(n^2)$ **F**
5. $\mathcal{O}(n^2) \subset \Omega(n)$ **F**
6. Si $f(n) = \mathcal{O}(n)$ entonces $2^{f(n)} = \mathcal{O}(2^n)$ **F**
7. $\Theta(n \log(n)) = \Theta(\log(n!))$ **Veamos que es V**

¿$\Theta(n\log(n)) = \Theta(\log(n!))$?

Para esto, basta probar que $n\log(n) \in \Theta(\log(n!))$ o que $\log(n!) \in \Theta(n\log(n))$. Veamos lo último, que consiste en:

$$\log(n!) \in \mathcal{O}(n\log(n)) \ \wedge \ \log(n!) \in \Omega(n\log(n))$$

$\log(n!) \in \mathcal{O}(n\log(n))$

Veamos la siguiente cota:

$$\log(n!) = \sum_{i=1}^{n} \log(i) \quad \leq \quad n \cdot \max_{1\leq i\leq n} \{\log(i)\} = n\log(n)$$

Entonces, tomando la definición de $\mathcal{O}(n\log(n))$:

$$\log(n!) \underset{cota}{\leq} n\log(n) \leq c \cdot n\log(n)$$

Tomamos $c = 1$ y vale para todo n natural en particular para $n_0 = 1$.

$\log(n!) \in \Omega(n\log(n))$

Miremos la siguiente cota:

$$\log(n!) = \log(1)+\log(2)+\cdots+\log\left(\frac{n}{2}\right)+\log\left(\frac{n}{2}+1\right)+\cdots+\log(n)$$

$$\geq \frac{n}{2}\log\left(\frac{n}{2}\right) = \frac{n}{2}(\log(n)-\log(2)) = \frac{n}{2}\log(n) - \frac{n}{2}\log(2)$$

$$\underset{q.v.q.}{\geq} \frac{n}{2}\log(n) - \frac{n}{4}\log(n) = \frac{n}{4}\log(n)$$

Podríamos tomar $c = 1/4$. Pero pusimos una condición sobre n. Para encontrar un n_0 recordemos que habíamos pedido

$$\frac{n}{2}\log(2) \leq \frac{n}{4}\log(n)$$

$\log(n!) \in \Omega(n \log(n))$ (continuación)

Recordemos que habíamos pedido $\frac{n}{2}\log(2) \leq \frac{n}{4}\log(n)$.
¿Para que n vale?

$$\frac{n}{2}\log(2) \overset{?}{\leq} \frac{n}{4}\log(n) = \frac{n}{2}\frac{1}{2}\log(n) = \frac{n}{2}\log(n^{1/2}) = \frac{n}{2}\log(\sqrt{n})$$

¿Para qué valores de n vale $\frac{n}{2}\log(2) \leq \frac{n}{2}\log(\sqrt{n})$?

Si tomamos $n = 4$ la desigualdad vale trivialmente y como log es creciente, vale para los $n \geq 4$.
Tomando $c = 1/4$ y $n_0 = 4$ probamos que vale la definición.

Ejercicios

Decidir si son verdaderas o falsas y justificar:

1. $2^n = \mathcal{O}(1)$ **F**
2. $n + \log^2 n \in \mathcal{O}(n + \log n)$ **V**
3. Dados $i, j \in \mathbb{N}$ fijos se tiene que $i \cdot n = \mathcal{O}(j \cdot n)$. **V**
4. $\Omega(n) \subset \mathcal{O}(n^2)$ **F**
5. $\mathcal{O}(n^2) \subset \Omega(n)$ **F**
6. Si $f(n) = \mathcal{O}(n)$ entonces $2^{f(n)} = \mathcal{O}(2^n)$ **F**
7. $\Theta(n\log(n)) = \Theta(\log(n!))$ **Veamos que es V**

Múltiples parámetros

Definición

$$O(g) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} f : \mathbb{N}^k \to \mathbb{R} \mid \exists \;\; \vec{n_0} \in \mathbb{N}^k, c \in \mathbb{R}_{>0} \\ \qquad\qquad\qquad\qquad f(\vec{n}) \leq c \cdot g(\vec{n}) \qquad \forall \vec{n} > \vec{n_0} \end{array} \right\}$$

Es decir, $f \in O(g)$ si y sólo si existen $\vec{n_0} \in \mathbb{N}^k$ y $c > 0$ tales que para todo $\vec{n} > \vec{n_0}$ se tiene:

$$f(\vec{n}) \leq c \cdot g(\vec{n})$$

Ejemplo: $m \log n = O(mn)$

¿Preguntas?