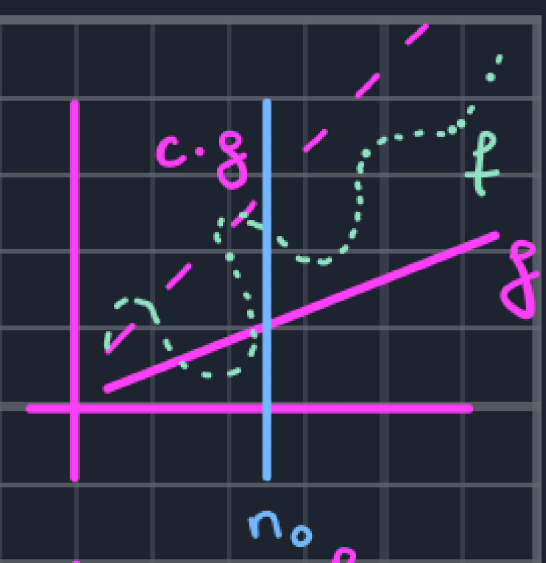
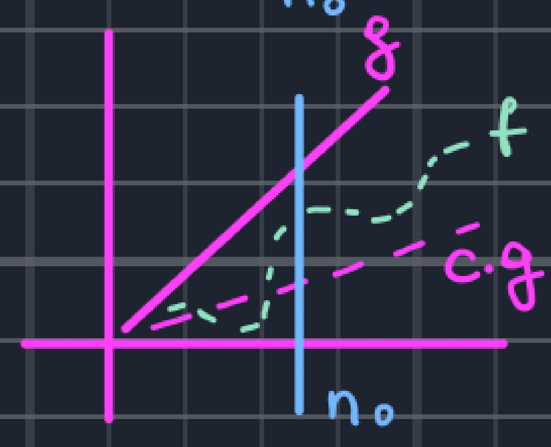
Definiciones Básicas

O (cota superior)



$f(n) \in O(g(n)) \leftrightarrow \exists\ c \in \mathbb{R}_{>0},\ n_0 \in \mathbb{N} \ / \ \forall\ n \geq n_0 : f(n) \leq c \cdot g(n)$

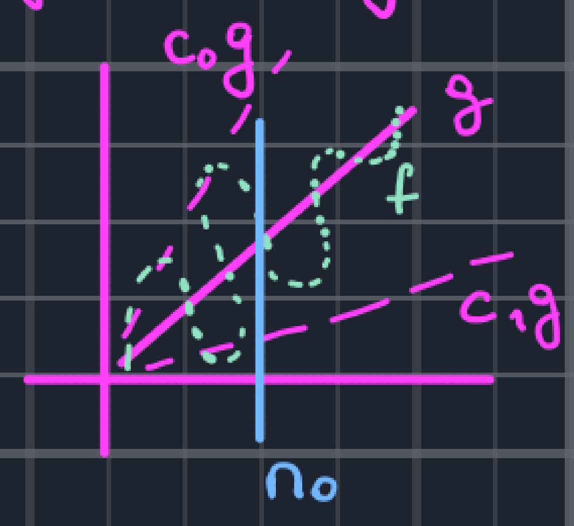
Ω (cota inferior)



$f(n) \in \Omega(g(n)) \leftrightarrow \exists\ c \in \mathbb{R}_{>0},\ n_0 \in \mathbb{N} \ / \ \forall\ n \geq n_0 : c \cdot g(n) \leq f(n)$

Θ (cota ajustada:

$f(n) \in \Theta(g(n)) \leftrightarrow f(n) \in O(g(n)) \wedge f(n) \in \Omega(g(n)) : \Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$



Álgebra de órdenes:
(vale para Ω y Θ)

- Suma: $O(f) + O(g) = O(f+g) = O(\max\{f, g\})$
- Producto: $O(f) \cdot O(g) = O(f \cdot g)$
- Reflexividad: $f \in O(f)$
- Transitividad: $f \in O(g)$ y $g \in O(h) \Rightarrow f \in O(h)$

Sólo para Θ vale:

Simetría: $f \in \Theta(g) \Rightarrow g \in \Theta(f)$

Θ define una rel. de equiv. entre funciones

❀ Siempre que podamos vamos a usar Θ para dar una cota ajustada

❀ NO fijar el tamaño de entrada para mejor/peor caso pues es un análisis asintótico

Propiedad del Límite:

Sean $f, g : \mathbb{N} \to \mathbb{R}_{>0}$. Si existe:

$$\lim_{n \to +\infty} \frac{f(n)}{g(n)} = \ell \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$$

$f \in \Theta(g) \leftrightarrow 0 < \ell < +\infty$

$f \in O(g)$ y $f \notin \Omega(g) \leftrightarrow \ell = 0$

$f \in \Omega(g)$ y $f \notin O(g) \leftrightarrow \ell = +\infty$

Ejercicio 1. Probar utilizando las definiciones que:

a) $n^2 - 4n - 2 = O(n^2)$.

b) Para todo $k \in \mathbb{N}$ y toda función $f : \mathbb{N} \to \mathbb{N}$, si $f \in O(n^k)$, entonces $f \in O(n^{k+1})$.

c) Si $f : \mathbb{N} \to \mathbb{N}$ es tal que $f \in O(\log n)$, entonces $f \in O(n)$.

a) $n^2 - 4n - 2 = O(n^2)$

$\exists\, n_0,\ c>0 \,/\, n \geq n_0 \Rightarrow n^2-4n-2 \leq c.n^2$ tomo $c=1$

$n^2-4n-2 \leq 1.n^2$

$-4n-2 \leq 0$ para cualquier valor

$0 \leq 4n+2$ de n, $\forall\, n \geq 0$

se podría usar la prop. de la suma en este caso? sí

b) $\forall\ k \in \mathbb{N} \wedge f : \mathbb{N} \to \mathbb{N}$, si $f \in O(n^k) \Rightarrow f \in O(n^{k+1})$

$f(n) \in O(g(n)) \iff \exists\, c \in \mathbb{R}_{>0},\ n_0 \in \mathbb{N} \,/\, \forall\, n \geq n_0,\ f(n) \leq c.g(n)$

Sé que $f \in O(n^k)$ y ~~qvq~~ $f \in O(n^{k+1})$

$g(n) = n^k$, como $g \in O(n^{k+1})$

entonces por transitividad: $f \in O(n^k)$ y $g \in O(n^{k+1}) \Rightarrow f \in O(n^{k+1})$

luego: $f \in O(n^k) \Rightarrow f \in O(n^{k+1})$ como quería probar.

Por def: $\exists\ n_0, c>0 \,/\, n \geq n_0 \Rightarrow n^k \leq c \cdot n^{k+1}$

$\cancel{n^k} \leq c.\ n.\cancel{n^k}$

$1 \leq \underbrace{c.n}$ para cualquier n y c

$c = 1$ y $n_0 \geq 1$

c) si $f : \mathbb{N} \to \mathbb{N}$ es tal que $f \in O(\log(n))$, entonces $f \in O(n)$

Como $O(\log(n)) \subset O(n)$, si $f \in O(\log(n)) \Rightarrow f \in O(n)$ por transitividad ✓

$\lim_{n\to\infty} \frac{\log n}{n} \underset{\text{L'Hopital}}{=} \frac{1}{n} \xrightarrow[n\to\infty]{} 0$ luego es cierto que $\log(n) \in O(n)$.

Ejercicio 2. Determinar la verdad o falsedad de cada una de las siguientes afirmaciones. **Justificar.**

a) $2^n = O(1)$.

b) $n = O(n!)$.

c) $n! = O(n^n)$.

d) $2^n = O(n!)$.

e) Para todo $i, j \in \mathbb{N}$, $i \cdot n = O(j \cdot n)$.

f) Para todo $k \in \mathbb{N}$, $2^k = O(1)$.

g) $\log n = O(n)$.

h) $n! = O(2^n)$.

i) $n^5 + bn^3 \in \Theta(n^5) \iff b = 0$.

j) Para todo $k \in \mathbb{R}$ $n^k log(n) \in O(n^{k+1})$.

k) Para toda función $f : \mathbb{N} \to \mathbb{N}$, $f = O(f)$.

a) $2^n = O(1)$ F

pues tomo $f = 2^n$ y $g = 1$ y $f \in O(g) \leftrightarrow \lim_{n\to+\infty} \frac{f(n)}{g(n)} = 0$

$\lim_{n\to+\infty} \frac{2^n}{1} = +\infty$ luego $2^n \notin O(1)$

Por def: $\exists\ c, n_0 \,/\, 2^n \leq 1.c\ \ \forall\ n \geq n_0$ pero 2^n es creciente y no acotada, por lo tanto $\nexists\, c$.

$\log_2 2^n \leq \log c$ como los naturales no tienen cota superior,

$n.\log_2 2 \leq \log_2 c$ no es cierto que $\forall\, n \in \mathbb{N}$ sean $\leq \log_2(c)$ para

$n \leq \log_2 c$ un c fijo.

b) $n = O(n!)$ V

$f = n$

$g = n!$

$\lim_{n\to+\infty} \frac{n}{n!} = \frac{\cancel{n}^{\,1}}{\cancel{n}_{1}.(n-1)!} = \frac{1}{(n-1)!} \xrightarrow[n\to\infty]{} 0$

$n \in O(n!) \leftrightarrow \{f : \mathbb{N} \to \mathbb{R} \,/\, (\exists\ n_0 \in \mathbb{N},\ c \in \mathbb{R}_{>0})\ \ n \leq c.\,n!\ \ \forall\ n \geq n_0\}$

$\exists\ c, n_0 \,/\, n \leq n!.c$

$1 \leq (n-1)!.c$ si tomo $c=1$

esto es verdadero

$\forall\, n$ y $n_0 > 1$

c) $n! = O(n^n)$ (V)

$n! \in O(n^n) \leftrightarrow \{\exists\ n_0 \in \mathbb{N},\ c \in \mathbb{R}_{>0} \ /\ n! \leq c.n^n,\ \forall n \geq n_0\}$

$n! \leq c\, n^n$

$n.(n-1)! \leq c.\ n.n\ldots n$

$n(n-1)\ldots 2.1 \leq c.n.n.n\ldots n$

esto es verdadero para cualquier n, pues multiplicar a un número por sí mismo n veces es mayor o a lo sumo igual (caso $n=1$) que multiplicar a un número por sus $n-1$ antecesores (factorial).

d) $2^n = O(n!) \leftrightarrow \{\exists\ n_0 \in \mathbb{N} \wedge c \in \mathbb{R}_{>0} \ /\ 2^n \leq c.n!,\ \forall n \geq n_0\}$ (V)

$2^n \leq c.\ n!$ Para un n lo suficientemente grande se cumple la desigualdad.

tomo $n = 4$ y $c = 1$

En estos casos donde no está claro si vale la desigualdad, es recomendable ver qué pasa para distintos valores de n, y si a partir de cierto n se cumple es porque ese es n_0.
Los $n > n_0$ se pueden demostrar por inducción.

$2^4 \leq 1.4!$

$16 \leq 24$ ✓

si hago inducción lo puedo probar para todo $n_0 > 4$ y $c=1$

HI: $2^n \leq n!$

qvq: $2^{n+1} \leq (n+1)!$

Caso base: $n_0 = 4$

$2^4 \leq 4!$

$16 \leq 24$ ✓

Paso inductivo:

$2^{h+1} \leq (h+1)!$

$\leftrightarrow 2.2^h \leq (h+1)\ h!$

HI $\leftrightarrow$ $2 \leq h+1$

$\leftrightarrow$ $1 \leq h \quad \forall h \geq n_0 = 4$

e) $\forall\ i,j \in \mathbb{N},\ i \times n = O(j \times n)$ (V)

$\leftrightarrow \{\exists\ c \in \mathbb{R}_{>0},\ n_0 \in \mathbb{N} \ /\ i.n \leq c.j.n \quad \forall n \geq n_0\}$

$i.\cancel{n}^{1} \leq c.j.\cancel{n}^{1}$

$i \leq c.j$

$\frac{i}{j} \leq c$

si tomo $\frac{i}{j} = c$ siempre vale la desigualdad $\forall n$.

f) $\forall\ k \in \mathbb{N},\ 2^k = O(1)$ (V)

como defino '$\forall k \in \mathbb{N}$' eso ya me indica que es una cte, no es un n cualquiera.

$\leftrightarrow \{\exists\ c, n_0,\ c \in \mathbb{R}_{>0} \wedge n_0 \in \mathbb{N} \ /\ 2^k \leq 1.c \quad \forall n \geq n_0\}$

$2^k \leq 1.c$

$2^k \leq c \rightarrow$ Alcanza con tomar $c = 2^k$ pues 2^k es una cte.

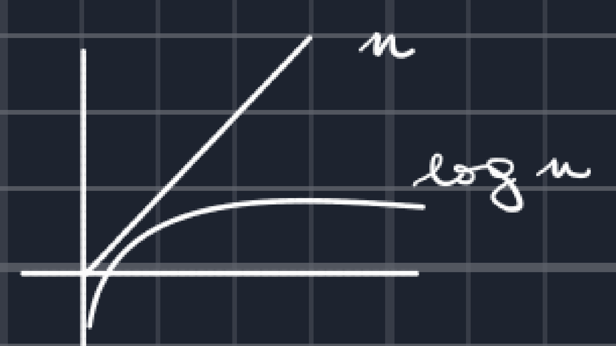
Luego, como $\exists$ c, y vale $\forall n$, entonces es verdadero

g) $\log(n) = O(n) \leftrightarrow \{\exists\ c, n_0,\ c \in \mathbb{R}_{>0},\ n_0 \in \mathbb{N} \ /\ \log(n) \leq c.n \quad \forall n \geq n_0\}$

$\log(n) \leq c.n$

tomo $c = 1$

$\frac{\log(n)}{n} \leq 1 \quad \forall n.$



verdadero para cualquier n

↳ También puedo probarlo con la prop. del lím.

h) $n! = O(2^n)$

Es cierto que $2^n \in O(n!)$ pero no vale que $n! \in O(2^n)$ pues $O(2^n) \subset O(n!)$ pero no al revés.

- $n! \leq c \cdot 2^n$ no vale para ningún n, ni c.

i) $n^5 + bn^3 \in \Theta(n^5) \leftrightarrow b = 0$

$n^5 + b\,n^3 = \Theta(\max\{n^5; bn^3\})$

(F) pues por ej: si $b = 1$ $\quad \Theta(\max\{n^5 : n^3\}) = \Theta(n^5)$ igualmente.

j) $\forall\, k \in \mathbb{R}$, $n^k \log(n) \in O(n^{k+1})$ (V)

$\leftrightarrow \quad n^k \log(n) \leq c \cdot n^{k+1}$

$\underbrace{\log(n) \leq n}$ $\quad$ tomo $c = 1$

ya lo había probado antes así que es verdadero!

k) $\forall\; f : \mathbb{N} \to \mathbb{N} \quad f = O(f)$ (V)

pues siempre vale que $f \leq c \cdot f$ $\quad c$ existe, por ej: tomando $c = 1$ ya vale.

Ejercicio 3. ¿Qué significa, intuitivamente, $O(f) \subseteq O(g)$? ¿Qué se puede concluir acerca del crecimiento de f y g cuando, simultáneamente, tenemos $O(f) \subseteq O(g)$ y $O(g) \subseteq O(f)$?

Significa que f está acotada superiormente por $O(g)$.
es decir, f crece a lo sumo lo que crece g. en el límite de $n \to \infty$

Si simultaneamente tenemos $O(f) \subseteq O(g)$ y $O(g) \subseteq O(f)$ es porque f y g crecen equitativamente, tienen la misma complejidad, lo que no quiere decir que sean iguales!

Ejercicio 4. Determinar el orden de complejidad temporal de peor caso de los siguientes algoritmos, asumiendo que todas las operaciones sobre arreglos y matrices toman tiempo $O(1)$. La complejidad se debe calcular en función de una medida de los parámetros de entrada, por ejemplo, la cantidad de elementos en el caso de los arreglos y matrices y el valor en el caso de parámetros naturales.

a) SUMATORIA, que calcula la sumatoria de un arreglo de enteros:

```
function SUMATORIA(arreglo A)
    int i, total;
    total := 0;
    for i := 0 ... Long(A) - 1 do
        total := total + A[i];
    end for
end function
```

Mejor caso: $|A| = 1$

$\Theta(1)$
$\Theta(1)$
$\Theta(1)$
$\Theta(1)$

$\Theta(1) \to \Omega(1)$ cota inf

Peor caso: $|A| = n$

$\Theta(1)$
$\Theta(1)$
$\Theta(n) \longrightarrow \sum_{i=0}^{n-1} O(1)$
$\Theta(1)$

$\Theta(n) \to O(n)$ cota sup.

$$f_{peor} = O(1) + \sum_{i=0}^{n-1} O(1) = O(n)$$

b) SUMATORIALENTA, que calcula la sumatoria de n, definida como la suma de todos los enteros entre 1 y n, de forma poco eficiente:

```
function SUMATORIALENTA(nat n)
    int i, total;
    total := 0;
    for i := 1 ... n do
        for j := 1 ... i do
            total := total + 1;
        end for
    end for
end function
```

Mejor caso $n = 1$

$O(1)$
$O(1)$
$O(1)$
$O(1)$
$O(1)$

$O(1)$

Peor caso:

$$f_{peor} = O(1) + \sum_{i=1}^{n} \sum_{j=1}^{i} O(1) =$$

$$= O(1) + \sum_{i=1}^{n} O(i) = O\left(\frac{n(n+1)}{2}\right) = O\left(\frac{n^2+n}{2}\right) = \frac{1}{2} O(n^2+n) = \frac{1}{2} O(\max\{n^2, n\}) = O(n^2)$$

entre parentesis va el resultado de la sumatoria

el $\frac{1}{2}$ como es cte. puede salir de O

c) ProductoMat, que dadas dos matrices A (de $p \times q$) y B (de $q \times r$) devuelve su producto AB (de $p \times r$):

```
function ProductoMat(matriz A, matriz B)
    int fil, col, val, colAFilB;
    matriz res(Filas(A), Columnas(B));
    for fil := 0 ... Filas(A) - 1 do
        for col := 0 ... Columnas(B) - 1 do
            val := 0;
            for colAFilB := 0 ... Columnas(A) - 1 do
                val := val + (A[fil][colAFilB] * B[colAFilB][col]);
            end for
            res[fil][col] := val;
        end for
    end for
    return res;
end function
```

→ la sumatoria va a ir desde 0 a col(A) - 1 (*)

$$f_{peor} = O(1) + \sum_{i=0}^{p-1} \sum_{j=0}^{r-1} \sum_{k=0}^{q-1} O(1) = O(p \cdot r \cdot q)$$

(*)

d) InsertionSort, que ordena un arreglo pasado como parámetro:

```
function InsertionSort(arreglo A)
    int i, j, valor;                 O(1)
    for i := 0 ... Long(A) - 1 do    O(n)
        valor := A[i];   i = 1 → j = 0      |A| = 3
        j := i - 1;      i = 2 → j = 1
        while j ≥ 0 ∧ a[j] > valor do
            A[j+1] := A[j];
            j := j - 1;
        end while
        A[j+1] := valor;
    end for
end function
```

$$\sum_{i=0}^{|A|-1} \sum_{j=0}^{i-1} O(1)$$

$|A| = n$

$$f_{peor} = \sum_{i=0}^{n-1} \sum_{j=0}^{i-1} O(1) = \sum_{i=0}^{n-1} O(i-1) = \sum_{i=1}^{n} O(i) = O\left(\frac{n(n+1)}{2}\right) = O(n^2+n) = O(n^2)$$

↓ trato de acomodar los bordes para que i vaya de 1 a n.

$|A| = 3$

$i = \{0, 1, 2\}$

$j = \{-1, 0, 1\}$

↓ while va desde $j \geq 0$ entonces los bordes quedan desde 0 a $i-1$

e) BúsquedaBinaria, que determina si un elemento se encuentra en un arreglo, que debe estar ordenado:

```
function BúsquedaBinaria(arreglo A, elem valor)
    int izq := 0, der := Long(A) - 1;
    while izq < der do
        int medio := (izq + der) / 2;
        if valor < A[medio] then
            der := medio;
        else
            izq := medio;
        end if
    end while
    return A[izq] = valor;
end function
```

$A = [0, 1, 2]$ Peor caso:

$izq = 0$ $der = 2$ valor $= 2$

1º) $0 < 2$ ✓ entra al while

medio $= 1$

$2 < 1$ ✗

⋮

$izq = 1$

2º) $1 < 2$

medio $= 2$

$2 < 2$ ✗

⋮

$izq = 2$

3º) $2 < 2$ ✗

Sé que Busqueda Binaria es $\log_2(n)$ pero porqué? como me doy cuenta mirando el código?

Rta: es porque en cada iteración divido o multiplico por n

En cada iteración divido a n por 2.
tomo k = # operaciones
$2^k = n$
$k \log_2 2 = \log_2 n$
$k = \log_2(n)$

$k = \log_2 2$ $n = 2$

$k = 1$ iteración

$k = \log_2 2^2$ $n = 4$

$k = 2$ iteraciones

⋮

$k = \log_2 2^3$ $n = 8$

$k = 3$ iteraciones

f) AlgoritmoQueHaceAlgo:

```
function AlgoritmoQueHaceAlgo(arreglo A)
    int i := 1; int j := 1;                      O(1)
    int suma := 1; int count := 0;               O(1)
    while i ≤ tam(A) do                          O(n)   n = |A|  Σ_{i=1}^{|A|} ...
        if i ≠ A[i] then
            count := count+1;
        end if
        j := 1;
        while j ≤ count do
            int k := 1;
            while k ≤ tam(A) do
                suma := suma + A[k];
                k := k * 2;
            end while
            j := j+1;
        end while
        i := i+1;
    end while
    return suma
```

como hago k.2 me doy cuenta que la compl. es logarítmica

$\sum_{j=1}^{i} \log(n)$ → 3° while

$A = [0,3,4]$

$i=1$, $|A|=3$

$1 \leq 3$? sí

$1 \neq A[1] \to 1 \neq 3$? sí!

count ++;

$j=1$, count $=1$

$j \leq$ count? sí!

$k=1$, $|A|=3$

$k \leq |A|$? sí!

$k = 1, 2, 4, 8, \dots$

suma := $1 + A[1] = 4$

$i=2$ $|A|=3$

$2 \leq 3$ sí!

$2 \neq A[2]$? sí

count $=2$ → como entro al if count se va equiparando con i, pues siempre le sumo 1 antes del while de j

$j=2$

$2 \leq 2$

$k=2$

$2 \leq |A|$

suma $= 4 + A[2]$

$k=4$

Rta: Porque en el peor caso entro siempre al if, y en ese caso le sumo 1 al count en cada iteración → luego me queda un count $= i$

$\sum_{j=1}^{count} = \sum_{j=1}^{i}$

$$f_{peor} = O(1) + \sum_{i=1}^{n} \sum_{j=1}^{i} \log(n)$$

Porqué el borde superior es i y no otra cosa, como por ej: el count

$$= \sum_{i=1}^{n} i \cdot \log(n) = \frac{n(n+1)}{2} \cdot \log(n) = \frac{1}{2}(n^2+n)\log(n)$$

$$= \log(n)\, n^2 + \log(n)\, n$$

$$= O(\max\{\log(n)\, n^2 ; \log(n)\, n\})$$

$$= O(\log(n)\, n^2)$$

Ejercicio 5. Para cada una de las siguientes afirmaciones, decida si son verdaderas o falsas y justifique su decisión.

a) $O(n^2) \cap \Omega(n) = \Theta(n^2)$

b) $\Theta(n) \cup \Theta(n \log n) = \Omega(n \log n) \cap O(n)$

c) $f \in O(g) \iff O(f) \subseteq O(g)$

d) Si $f \in \Omega(g)$, entonces $O(f) \cap \Omega(g) = O(g) \cap \Omega(f)$

e) Si $f(n) < g(n)$ para todo n, entonces $\Theta(f) != \Theta(g)$

f) Si $f \in O(g)$, entonces $f * g \in \Theta(g)$

$O(1) \subseteq O(\log(n)) \subseteq O(n) \subseteq O(n\log(n)) \subseteq O(n^2) \dots O(n^k)$

$\Omega(n^k) \subset \dots \subset \Omega(n^2) \subset \Omega(n\log(n)) \subset \Omega(n) \subset \Omega(\log(n)) \subset \Omega(1)$

a) $O(n^2) \cap \Omega(n) = \Theta(n^2)$ Falso

$\{n\}$ $n \in O(n^2)$ y $n \in \Omega(n)$. pero $n \notin \Theta(n^2)$.

$n \in (O(n^2) \cap \Omega(n))$ pues $O(n) \subset O(n^2)$ pero $O(n) \notin \Theta(n^2)$

¿por qué no entra $O(n)$ en $\Theta(n^2)$?

Porque $O(n) \not\subset \Theta(n^2)$

esto es porque Θ es una cota ajustada, nada fuera del orden de n^2 va a estar contenido en $\Theta(n^2)$

b) $\Theta(n) \cup \Theta(n\log(n)) \in \Omega(n\log(n)) \cap O(n)$ Falso

$\Theta(n) \cup \Theta(n\log(n)) \notin \emptyset$

porque todo lo que está en $O(n)$ $(\{1, \log(n), n\})$ no está en $\Omega(n\log(n))$ $(\{n^k, \dots, n^2\})$ entonces no hay intersección.

c) $f \in O(g) \leftrightarrow O(f) \subseteq O(g)$ V ✓

→) $f \in O(g) \to O(f) \subseteq O(g)$

$f \in O(g) = \{\exists\ c \in \mathbb{R}_{>0}, n_0 \in \mathbb{N} \ /\ f \leq c \cdot g(n)\ \forall\ n \geq n_0\}$ luego $O(f) \subseteq O(g)$

←) $O(f) \subseteq O(g) \to f \in O(g)$

como $f \in O(f)$ por reflexividad, entonces $f \in O(g)$ y luego $f \in O(g)$

d) Si $f \in \Omega(g) \to O(f) \cap \Omega(g) = O(g) \cap \Omega(f)$ Falso. Solo puede ser verdadero para $f=g$:(

·Nota: Antes de tratar de demostrarlo Probar cosas!

$f = n^2$ $g = n$

$f \in \Omega(g)$

$O(n^2) \cap \Omega(n) \neq O(n) \cap \Omega(n^2)$

$\{n\log(n)\} \neq \emptyset$

$O(1) \subseteq O(\log(n)) \subseteq O(n) \subseteq O(n\log(n)) \subseteq O(n^2) \dots O(n^k)$

$\Omega(n^k) \subset \dots \subset \Omega(n^2) \subset \Omega(n\log(n)) \subset \Omega(n) \subset \Omega(\log(n)) \subset \Omega(1)$

e) si $f(n) < g(n) \ \forall n \rightarrow \Theta(f) \neq \Theta(g)$ Falso

$f = n^2$ y $g = n^2+1$

$n^2 < n^2+1$

$O(f) \in O(g) \rightarrow \Theta(f) = n^2$ y $\Theta(g) = n^2$ a pesar de que $f<g$ pues sumar una cte. no afecta la complejidad

f) si $f \in O(g) \rightarrow f \cdot g \in \Theta(g)$

$f = n^2+n \in O(n^2) \rightarrow f \cdot g = (n^2+n)\, n^2 = n^4+n^3 \notin \Theta(n^2)$

Falso

Ejercicio 6. Para cada una de las siguientes afirmaciones, decida si son verdaderas o falsas y justifique su decisión.

a) $n+m = O(nm)$.

b) $n+m^5 = O(m^5)$.

c) $nm = O(n+m)$.

d) $m^5 = O(n+m^5)$.

a)

$f = n+m$ y $g = n.m$

Es cierto que $\exists\ c_0 \in \mathbb{R}_{>0},\ n_0 \in \mathbb{N} \ / \ f \leq c.g \ \forall n \geq n_0,\ m \geq m_0 \}$

$n+m \leq c.\ n.m$

entonces $n+m \leq c.\ n.\ m$

$\frac{1}{m} + \frac{1}{n} \leq c$ tomo $c=2$

$\frac{1}{m} + \frac{1}{n} \leq 2$ (V) como $\frac{1}{m} + \frac{1}{n}$ es a lo sumo $1+1$ entonces vale para $c=2$

b) $n+m^5 = O(m^5)$

$n+m^5 \leq c.\ m^5$

$\leftrightarrow n+m^5 \leq c\ m^5$

$n \leq c.m^5 - m^5$

$n \leq (c-1)\, m^5$

n no es acotada entonces (F) $n+m^5 \notin O(m^5)$.

c)

$n.m = O(n+m) \leftrightarrow \{\exists\ c_0 \in \mathbb{R}_{>0},\ n_0 \in \mathbb{N} \ / \ n.m \leq c.(n+m) \ \forall n \geq n_0,\ m \geq m_0\}$

$n.m \leq c\,(n+m)$

$1 \leq c\,(\frac{1}{m}+\frac{1}{n})$

$\frac{1}{c} \leq \frac{1}{m}+\frac{1}{n}$

d) $m^5 = O(n+m^5)$

$m^5 \leq c.\ (n+m^5)$ si tomo $c=1$

$m^5 \leq n+m^5$ es (V) $\forall n$

Ejercicio 7. Sean $f, g : \mathbb{N} \to \mathbb{N}$, y supongamos que está definido el límite $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = \ell \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$. Probar que:

a) $0 < \ell < +\infty$ si y sólo si $f \in \Theta(g)$.

b) $\ell = +\infty$ si y sólo si $f \in \Omega(g)$ y $f \notin O(g)$.

c) $\ell = 0$ si y sólo si $f \in O(g)$ y $f \notin \Omega(g)$.

Recordar las definiciones de límite:

- $\lim_{n\to+\infty} a_n = \ell \in \mathbb{R}$ si $\forall \varepsilon > 0.\ \exists n_0 \in \mathbb{N}.\ \forall n > n_0.\ |a_n - \ell| < \varepsilon$.
- $\lim_{n\to+\infty} a_n = +\infty$ si $\forall M > 0.\ \exists n_0 \in \mathbb{N}.\ \forall n > n_0.\ a_n > M$.

a) $0 < \ell < +\infty \leftrightarrow f \in \Theta(g)$

$\lim_{n\to\infty} a_n = \ell \in \mathbb{R}$ si $\forall \varepsilon > 0,\ \exists\ n_0 \in \mathbb{N}.\ \forall n > n_0\ |a_n - \ell| < \varepsilon$

$0 < \ell < +\infty \rightarrow f \in \Theta(g)$ si tomo $a_n = \frac{f(n)}{g(n)}$

$\ell \in \mathbb{R} \leftrightarrow \forall \varepsilon > 0,\ \exists\ n_0 \in \mathbb{N}\ \forall n > n_0\ |a_n - \ell| < \varepsilon$

$|\frac{f(n)}{g(n)} - \ell| < \varepsilon$

Es valido este paso ?

$\frac{f(n)}{g(n)} - \ell < \varepsilon$

$f(n) < \underbrace{(\varepsilon+\ell)}_{c}.g(n)$

$f(n) < c.\,g \Rightarrow$ Def $f \in \Theta(g)$

b) $\lim_{n\to\infty} a_n = +\infty$ si $\forall M>0, \exists n_0 \in \mathbb{N}, \forall n > n_0, a_n > M$

$a_n = \frac{f}{g}$

$\frac{f}{g} > M$

$f > M.g$

$M.g < f$ si $M = c$

$c.g < f \rightarrow f \in \Omega(g)$ por def.

y $f \notin O(g)$ pues es un menor estricto, si fuera menor o igual entonces ahí f podría llegar a pertenecer a $O(g)$.

c) $l = 0 \leftrightarrow f \in O(g)$ y $f \notin \Omega(g)$

$\lim_{n\to\infty} a_n = 0 \in \mathbb{R}$ si $\forall \varepsilon > 0, \exists n_0 \in \mathbb{N}. \forall n > n_0 \quad |a_n| < \varepsilon$

$a_n = \frac{f}{g}$: $\lim_{n\to\infty} \frac{f}{g} = 0 \leftrightarrow \forall c > 0, \exists n_0 \in \mathbb{N}, \forall n > n_0, \frac{f}{g} < c \Leftrightarrow f < c.g$ luego por def. $f \in O(g)$ como quería probar.

$c = \varepsilon$