

Antes, ejercicio de repaso de la PRÁCTICA.

R1): Para Θ que quede menor $\Theta \xrightarrow{\text{Cn } O \text{ danáma los cota sup}}$ $\Theta \xrightarrow{\text{Cn } \Omega \text{ danáma los cota inf.}}$

• $i^* \times \epsilon O(n^2)$ $i^* \times \epsilon O(m^3)$.

• Of elementos $O(1)$.

• Contarán las complejidades.

• A veces $P_E = M_E$ o no si no hay cond. de if

Dar una cota de complejidad **ajustada** para el mejor y peor caso del siguiente algoritmo:

Búsqueda lineal

```
1: function busquedaLineal(Arreglo de Enteros A, Natural e)
2:   n = Long(A)  $\approx O(1) + O(1)$ 
3:   for i = 0...n - 1 do
4:     if A[i] = e then  $= O(1)$ 
5:       devolver true  $\approx O(1)$ 
6:   devolver false
```

MEJOR CASO: = PEOR CASO $\times 2$ HAY REPETIC.

MEJOR CASO: $\Theta(1)$

PEOR CASO: $\Theta(n)$

El for se ejecuta n veces

i return

MJOR CASO: $\Theta(1) + \Theta(1) = \Theta(\max\{1, 1\}) = \Theta(1)$ ✓

PEOR CASO:

$$\begin{aligned}
 \text{PEOR CASO: } & \Theta(1) + \sum_{i=0}^{m-1} \Theta(1) + \Theta(1) \\
 & = \Theta(1) + m \cdot \Theta(1) + \Theta(1) \\
 & = \Theta(1) + \Theta(m) \\
 & = \Theta(n \times \{1, m\}) \\
 & = \Theta(m)
 \end{aligned}$$

Valor en Matriz

Matrices en Degradé

```

1: function valorEnMatriz(Matriz de naturales A, Natural val)
2:   n := Long(A)  $\Rightarrow$  2 op  $\Theta(1)$ 
3:   i := 0  $\leq 10^6$ 
4:   while i < n  $\wedge$  A[0, i]  $\leq$  val do
5:     i = i + 1
6:   colLim := i - 1
7:   i := 0
8:   while i < n  $\wedge$  A[i, 0]  $\leq$  val do
9:     i = i + 1
10:    filLim := i - 1
11:    for i = 0...filLim do
12:      for j = 0...colLim do
13:        if A[i, j] = val then
14:          devolver true
15:    devolver false

```

MENOR CASO \neq PEOR CASO

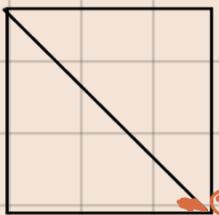
El menor caso de los cuantos $VAL > A[0, i] \wedge A[i, 0]$
llega al único numero de FOR.

$$f_{menor} = \Theta(1)$$

PEOR CASO:

LÍNEA 2 y 3 son $\Theta(1)$ xq son elementales.

El primer WHILE se ejecuta m veces (considerando q
los ítems i Val sea el mayor). ✓



$$\text{Max } m-1 \Rightarrow i \leq m-1$$

La función del while se ejecuta m veces y la O(n) es $\Theta(1)$.

$$\Theta(1) + m \cdot \Theta(1)$$

El algoritmo tiene en total el costo de los primeros m términos del bucle.

$$\Theta(1) + \sum_{i=0}^{m-1} \Theta(1) + \sum_{j=0}^{m-1} \Theta(1) = \sum_{i=1}^m \Theta(1) + \sum_{i=1}^n \Theta(1) = m \cdot \Theta(1) = \Theta(m)$$

Por lo tanto, el bucle se ejecuta m veces y se hace trabajo constante.

\Rightarrow Condición - 1 ✓ $\Rightarrow i \leq m-1$? ¿ $i < n-1$? Si el bucle es $i = n-1$ Max.

$$\sum_{i=0}^{n-1} \left(\sum_{j=0}^{m-1} \Theta(1) \right) = \sum_{i=0}^{n-1} m \cdot \Theta(1) = \sum_{i=0}^{n-1} \Theta(m) = m * \Theta(m) = \Theta(m^2)$$

$\hookrightarrow \sum_{j=1}^m \Theta(1) = m$

$$\text{Entonces, } \Theta(1) + \Theta(m) + \Theta(m) + \Theta(m^2)$$

$$F_{\text{PEON}}(m) = \Theta(\max(1, n, m^2)) = \Theta(m^2)$$

Enunciado

Demostrar que si $f \in O(g) \implies k * f \in O(g)$ para cualquier constante positiva k .

$$f \in O(g) \iff \exists C \in \mathbb{R}_{>0}, M_0 \in \mathbb{N} \text{ s.t. } \forall n > M_0 : f(n) \leq C * g(n)$$

$$\Rightarrow \underbrace{k * c * g(n)}_{\text{CONSTANTES}} \in O(g)$$

CONSTANTES \rightarrow Combinan complejidad

$$\Rightarrow f(n) \in O(g)$$

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

- a) $2^n = O(1)$. "2" crece más rápidamente que $O(1)$ " g) $\log n = O(n)$.
- b) $n = O(n!)$. $2 \not\in O(1)$ h) $n! = O(2^n)$.
- c) $n! = O(n^n)$. i) $n^5 + bn^3 \in \Theta(n^5) \iff b = 0$.
- d) $2^n = O(n!)$. j) Para todo $k \in \mathbb{R}$ $n^k \log(n) \in O(n^{k+1})$.
- e) Para todo $i, j \in \mathbb{N}$, $i \cdot n = O(j \cdot n)$. k) Para toda función $f : \mathbb{N} \rightarrow \mathbb{N}$, $f = O(f)$.
- f) Para todo $k \in \mathbb{N}$, $2^k = O(1)$.

a) FALSO. $O(1)$ es una operación constante.

Es la constante si $M=0$, si $M=1$ ya no es constante.

Además 2^n crece según el valor de n , x

la constante siempre será menor a la complejidad constante.

b) FALSO. Vale que $O(n) \subseteq O(n!)$ pero no es igual pues

M : número más grande que M .

$$c) \lim_{n \rightarrow \infty} \frac{M!}{n^M} \equiv \lim_{n \rightarrow \infty} \frac{(M-1)! \cdot M}{n^M} \equiv \lim_{n \rightarrow \infty} \frac{(n-1)!}{n^{n-1}}$$

$$e) f(i \cdot n) \leq O(j \cdot n) \quad \forall i, j \in \mathbb{N} \quad V.$$

$$f) \forall k, f(2^k) \leq O(1)$$

Concepto: Para $K \geq 1$ no vale?

$$g) \lim_{n \rightarrow \infty} \frac{\log n}{n} = 1$$

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 $O(g) \subseteq O(f)$?

f y g son logaritmos de funciones. Por lo tanto,

intuitivamente puede parecer que todas las f y g están

en $O(f)$ son un subconjunto de $O(g)$.

Si f está contenido en g , pero g no está contenido en f significa que es el mismo logaritmo función.

En decir $A \subset B \wedge B \subseteq A \Rightarrow A = B$

Por lo tanto se ve que un crecimiento "es igual?"

Y si es realmente igual es el mismo trabajo función \Rightarrow P.N. 6.

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:

```
1: function SUMATORIA(arreglo A)
2:   int i, total; 0 inclusivo
3:   total := 0;
4:   for i := 0 ... Long(A) - 1 do 3 2 2
5:     total := total + A[i]; 3
6:   end for
7: end function
```

• C.O

• P.C:

$$\Theta(1) + \sum_{i=0}^{m-1} \Theta(1) \equiv \Theta(1) + \sum_{i=1}^m \Theta(1) = \Theta(1) + m \cdot \Theta(1)$$

$$= \max(\{1, m\})$$

$$= \Theta(m)$$

• n.c: ~~if~~ no hay cond de confe. $\Theta(n)$

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

```

1: function SUMATORIALENTA(nat n)
2:   int i, total;
3:   total := 0;
4:   for i := 1 ... n do
5:     for j := 1 .. i do
6:       total := total + 1;
7:     end for
8:   end for
9: end function

```

$$\begin{aligned}
 \text{P.C.} &= \Theta(1) + \sum_{i=1}^n \sum_{j=1}^i \Theta(1) \equiv \Theta(1) + \sum_{i=1}^n i \cdot \Theta(1) \equiv \Theta(1) + \Theta(1) + \sum_{i=1}^n i \\
 &\quad \underbrace{\sum_{j=1}^i}_1, \quad \underbrace{\sum_{j=1}^2}_2, \quad \underbrace{\sum_{j=1}^3}_3 = i \\
 &\equiv \Theta(1) + \Theta(1) + \frac{n(n+1)}{2} \\
 &\equiv \Theta(1) + \Theta(1) + \frac{n^2+n}{2}
 \end{aligned}$$

by nome imp.

Luego, $\max(\{1, 1, m^2\}) \equiv m^2$.

n.c = 16va al peor caso q no hay cond se lanza q no pden limitar la lqg de ls lne.

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

```

1: function PRODUCTOMAT(matriz A, matriz B)
2:   int fil, col, val, colAFilB; 0
3:   matriz res(Filas(A), Columnas(B));
4:   for fil := 0 ... Filas(A) - 1 do
5:     for col := 0 ... Columnas(B) - 1 do
6:       val := 0;  Non fijo
7:       for colAFilB := 0 ... Columnas(A) - 1 do
8:         val := val + (A[fil][colAFilB] * B[colAFilB][col]);   $\Theta(1) \Rightarrow 7 \text{ op.}$ 
9:       end for
10:      res[fil][col] := val;   $\Rightarrow 3 \text{ op}$ 
11:    end for
12:  end for
13:  return res;

```

$MC = PC \wedge \text{No hay } (\omega \lambda) \in \text{CONT}$.

$$\text{PEOR: } \sum_{i=0}^{|C.A|-1} \Theta(1) \equiv \sum_{c=0}^{|C.B|-1} \sum_{i=0}^{|C.A|-1} \Theta(1) \equiv \sum_{F=0}^{|F.A|-1} \sum_{c=0}^{|C.B|-1} \sum_{i=0}^{|C.A|-1} \Theta(1) \equiv \sum_{F=0}^{|F.A|-1} \sum_{c=0}^{|C.B|-1} m$$

- $\sum_{i=1}^{|C.A|} \Theta(1) \equiv 1+1+1+1 = |C.A|$

- $\sum_{c=0}^{|C.B|-1} m \cdot \Theta(1) = \sum_{c=1}^{|C.B|} m \cdot \Theta(1) = m \cdot (m \cdot \Theta(1))$

- $\sum_{F=0}^{|F.A|-1} m \cdot (m \cdot \Theta(1)) = m_2 \cdot (m \cdot (m \cdot \Theta(1)))$

Llegó, el PEOR: $\Theta(m_2 \cdot (m \cdot (m \cdot \Theta(1))))$

¿CONVERGE ESTO?

Né se expresan los 3 bucles (que no están mal) y se suman todos en dirección.

RJA: $\Theta(m_2 \cdot m \cdot m)$ ✓

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

```
1: function INSERTIONSORT(arreglo A)
2:     int i, j, valor; 0
3:     for i := 0 ... Long(A) - 1 do
4:         valor := A[i]; 2
5:         j := i - 1; 2
6:         while j ≥ 0 ∧ a[j] > valor do
7:             A[j+1] := A[j];
8:             j := j - 1;
9:         end while
10:        A[j+1] := valor;
11:    end for
12: end function
```

POR CASO: Al cumplir la otra condición del WHILE siempre.

PEOR CASO: Cada j depende de i , no lo toca.

Si $A[j], A[j] < \text{VALOR}$ es el mejor caso, pero si el arreglo se está ordenando en $\Theta(n^2)$ siempre hace el for hasta el FIN.

P.C.: Siempre $i = -1$ (cuando $i=0$) para ese caso no hace nada y sigue en la parte de abajo todo o.p. ELEMENTAL.

Si $i=0$ solo O.P. Elementales.

PRIMER CASO (VAL): $i=1$ y $j=0$

ULTIMO CASO (VAL): $i=|A|-1$ $j=|A|-2$

$$\sum_{i=0}^{m-1} \sum_{j=0}^0 \Theta(1) = \sum_{i=1}^m \sum_{j=0}^{i-1} \Theta(1) = \sum_{i=1}^m i = \frac{\cancel{m^2} + m}{2} = \Theta(m^2)$$

↳ más operaciones

$$\text{Sugr}, \Theta(m \cdot m) = \Theta(m^2) \quad \checkmark$$

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

```

1: function BÚSQUEDABINARIA(arreglo A, elem valor)
2:   int izq := 0, der := Long(A) - 1; 3
3:   while izq < der do
4:     3 int medio := (izq + der) / 2; JAPORTA COMPLEJIDAD-LIMITA RANGO
5:     2 if valor < A[medio] then
6:       1 der := medio; he operar complejidad
7:     else
8:       1 izq := medio; he operar complejidad
9:     end if
10:   end while
11:   3 return A[izq] = valor;
12: end function

```

NESON CASO = PRIMERO CASO KQ NO HAY AND DE CONE.

TERMINA CUANDO izq = DER (alr 1 por m qdo x ver)

Como izq = 0 y der = Long(A) - 1 sin tener pines falso para del array A.

La lista de m revisa todo el array q no tiene elem.

$$\text{Es } \Theta(\log_2(|A|)) \equiv \log(|A|)$$

$$\text{Sugr, F_psol} = \Theta(1) + \sum_{i=0}^{\log(|A|)} \Theta(1) + \Theta(1)$$

$$\begin{aligned}
&= \Theta(1) + \log(|A|) \cdot \Theta(1) + \Theta(1) \\
&= \max(\{1, \log(|A|)\}) \\
&= \Theta(\log(|A|))
\end{aligned}$$

¿Puedes intuir que siempre hace algo? Verás que el algoritmo

operará en un (O)FICIONAL siempre la (COMPLEJIDAD) de?

f) ALGORITMOQUEHACEALGO:

```
1: function ALGORITMOQUEHACEALGO(arreglo A)
2:   int i := 1; int j := 1;
3:   int suma := 1; int count := 0;
4:   while  $i \leq tam(A)$  do i <= tam(A)? i = TAN[A] & INT
5:     if  $i \neq A[i]$  then
6:       count := count + 1;
7:     end if
8:     j := 1;
9:     while  $j \leq count$  do
10:      int k := 1;
11:      while  $k \leq tam(A)$  do summa := summa + A[k];
12:        k := k * 2;
13:      end while
14:      j := j + 1;
15:    end while
16:    i := i + 1;
17:  end while
18: return suma
```

$\ln A[TAN[A]]$

(OND) VARIAS (COMPLEJIDAD):

COUNT se incrementa si $i \neq A[i]$ x lo tanto (que) i no es de 1 a $tam(A)$ lo que significa que los numeros quedan corriendo.

En resumen, el PESON (AS) se da si $i \neq A[i]$ tiene A.

$i = 1$ [2, 3, 4, 5]

$i = 2$ [2, 3, 4, 5]

Entonces COUNT crece, y entra en el WHILE de $j \leq COUNT$.

No recuerda que el IND lo recorre con el valor que le COMPLEJIDAD en OTRA (MEJOR CASO)

$i = 1$ [0, 1, 2, 3, 4]

$i = 2$ [0, 1, 2, 3, 4]

POR CASO PASA POR:

NETO POR CASO PASA POR:

$$\text{MEJOR CASO: } \Theta(1) + \Theta(1) + \sum_{i=0}^{\lceil A/2 - 1 \rceil} \Theta(1) + \Theta(1) + \Theta(1) + \Theta(1)$$

5 8 9 11

$$+ \Theta(1)$$

$$T_{\text{MEJOR}} = \Theta(1) + \sum_{i=1}^m \Theta(1) + \Theta(1) \equiv \Theta(1) + m \cdot \Theta(1) + \Theta(1)$$

$$\equiv \max(\{1, m, 1\}) \equiv \Theta(m)$$

POR CASO:

C1:

$K := 1$ $1 \leq T_{\text{AM}}(A) \quad K > T_{\text{AM}}(A)$

WHILE $K \leq T_{\text{AM}}(A)$ DO:

$K=1$	$K=2$	$K=4$

$K := K * 2 \rightarrow$ SE VA SUSTITUYENDO VALORES DE K .

END WHILE

$$\sum_{K=1}^{\log_2(A)} \Theta(1) = \log_2(A) \cdot \Theta(1) \Rightarrow O(n)$$

C2:

fuerza, WHILE $j \leq \text{COUNT}$ DO:

INT $K := 1$

{0}

$j := j + 1$

END WHILE

i

$$\sum_{j=1}^i \Theta(1) + O_> + \Theta(1) \equiv i * (\Theta(1) + (\log(A) \cdot \Theta(1)) + \Theta(1))$$

DIFERENCIA:

En el peor caso, $i \neq A[i]$ y si, en el count tiene $|A| - 2$ qd el lgo $i=0$ no vale.

$$(3): \sum_{i=1}^{|A|} i * (\Theta(1) + (\log(A) \cdot \Theta(1)) + \Theta(1))$$

$$\equiv (\Theta(1) + (\log(A) \cdot \Theta(1)) + \Theta(1)) \cdot \sum_{i=1}^{|A|}$$

$$\equiv \log(A) \cdot \frac{A^2 + A}{2}$$

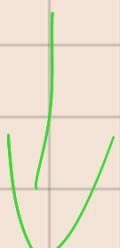
$$\equiv \log(A) \cdot \frac{A^2}{2} + \log(A) \cdot \frac{A}{2}$$

Dicho por ABAJO

$$\equiv \Theta(\log(A) \cdot A^2)$$

ME QUEDO CON
TERM MAYOR

(
SACO CONSTANTES Q
DIVIEN O MUL)



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 $f * g \in \Theta(g)$

a) Por def,

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

$$\Omega(n) = \{ c : \exists c > 0, n_0 \in \mathbb{N} / \forall n \geq n_0 : f(n) \geq c * g(n) \}$$

$$\Theta(n) = \Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$$

$O(n^2)$: funciones que crecen menor o igual a n^2 .

$$\text{ej: } n^2, n, \log n.$$

$\Omega(n)$: funciones que crecen más que n .

$$\text{ej: } n, n^2, n^3, 2^n$$

$\Theta(n^2)$: funciones exponencialmente más complicadas que n^2 o similares.

Folios.

$$O(n^2) \cap \Omega(n) = \{ n, n^2 \}$$



Nos quedamos igual
a n^2 .

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

$$f(n) \xrightarrow{2^n = O(1)} g(n)$$

- a) $2^n = O(1)$. g) $\log n = O(n)$.
- b) $n = O(n!)$. h) $n! = O(2^n)$.
- c) $n! = O(n^n)$. i) $n^5 + bn^3 \in \Theta(n^5) \iff b = 0$.
- d) $2^n = O(n!)$. j) Para todo $k \in \mathbb{R}$ $n^k \log(n) \in O(n^{k+1})$.
- e) Para todo $i, j \in \mathbb{N}$, $i \cdot n = O(j \cdot n)$. k) Para toda función $f : \mathbb{N} \rightarrow \mathbb{N}$, $f = O(f)$.
- f) Para todo $k \in \mathbb{N}$, $2^k = O(1)$.

a) $2^n \leq c \cdot 1$ $C \in \mathbb{R} > 0$ \wedge $n \geq n_0$ $n \in \mathbb{N}$

$\Rightarrow 2^n \leq c$ FALSO. Ej.: $C=4$, $4 \geq 2^4$

CONSIDERANDO ϵ fijo

$$b) M \leq C \cdot m!$$

$$\Rightarrow M \leq C \cdot (M-1)! \cdot M$$

$$\Rightarrow 1 \leq C \cdot (M-1)! \quad \text{VERDADERO.}$$

$$c) M! \leq C \cdot m^m$$

$$\Rightarrow \prod_{i=1}^M i \leq C \cdot m^m$$

$$\Rightarrow \prod_{i=1}^M \frac{i}{m^m} \leq C$$

\Rightarrow

$$i) m^5 + 6m^3 \in \Theta(m^5) \Leftrightarrow b=0$$

\Leftarrow) Ver si $b=0$?

$$\Rightarrow m^5 + 6m^3 \leq C * m^5$$

$$\exists C \in \mathbb{R} > 0$$

$$m \geq m_0 \quad (m_0 > 1)$$

$$\Rightarrow b n^3 \leq C * n^2 - n^2$$

$$\Rightarrow b n^3 \leq n^5 (-1 + c)$$

$$\Rightarrow b \leq \frac{n^5}{n^3} (-1 + c)$$

$$\Rightarrow b \leq n^2 (-1 + c)$$

$\Rightarrow 0 \leq -n^2 + n^2 \cdot c$ Porque $c > 0$ y la def dice que $\exists c$ en el ejmplo tomó ver $c=1$.

$$0 \leq 0$$

$\Rightarrow 1)$ ¿ Vale?

k) $f : \mathbb{N} \rightarrow \mathbb{N}$, $f = O(f)$ Es en $V \not\rightarrow$ la REFLEXIVIDAD.

Siempre \exists un C tal q $C * f$ sea f .

En otros palabras, siempre tiene función dada incluida en el O que si las funciones menores o iguales a ella misma.

j) $m^k \cdot \log(n) \in O(n^{k+1})$

$$\Rightarrow m^k \cdot \log(n) \leq C \cdot n^{k+1} \quad C \in \mathbb{R} > 0$$

$$\Rightarrow \log(n) \leq C * m \quad m \geq m_0 \quad m_0 \in \mathbb{N}$$

$$\Rightarrow \log(n) = O(m)$$

m^k siempre es \leq en n^{k+1} .

$\log(n)$ crece lento.

" $\log(n)$ " que $m^k \cdot \log(n)$ no llega a ser de n^{k+1}

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) $n + m = O(nm)$

$$n + m \leq C \cdot nm \quad n \geq n_0, m \in \mathbb{N}$$

$$C \in \mathbb{R} > 0$$

$$\Rightarrow \frac{n + m}{nm} \leq C$$

$$\Rightarrow \frac{n}{nm} + \frac{m}{nm} \leq C$$

$$\Rightarrow \frac{1}{m} + \frac{1}{n} \leq C$$

Ejercicio 1. Probar utilizando las definiciones que:

- $n^2 - 4n - 2 = O(n^2)$.
- Para todo $k \in \mathbb{N}$ y toda función $f : \mathbb{N} \rightarrow \mathbb{N}$, si $f \in O(n^k)$, entonces $f \in O(n^{k+1})$.
- Si $f : \mathbb{N} \rightarrow \mathbb{N}$ es tal que $f \in O(\log n)$, entonces $f \in O(n)$.

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

Por def de O,

$$n^2 - 4n - 2 \leq C * n^2 \text{ con } C \in \mathbb{R} > 0.$$

Usando límites. $\lim_{n \rightarrow +\infty} \frac{n^2 - 4n - 2}{n^2} = \frac{1 - \frac{4}{n} - \frac{2}{n^2}}{1} = \frac{1 - \frac{4}{\infty} - \frac{2}{\infty}}{1} = 1$

Luego, $f \in \Theta(n) \Leftrightarrow 0 < l < +\infty$ lmt (con)

$$O(n) \subseteq \Theta(n) \text{ mta qe } n^2 - 4n - 2 \leq O(n^2)$$

Por def, $n^2 - 4n - 2 = O(n^2) \equiv n^2 - 4n - 2 \leq C * n^2 \text{ con } C \in \mathbb{R} > 0$

Es qe $C > 0$, x los tants observa el factor determinante de la complejidad qd siempre qe $n^2 \leq C * n^2$ los qd sea el C, además de qe el $-4n - 2$ hace q esté incluido aún más.

b) $k \in \mathbb{N}$ $f : \mathbb{N} \rightarrow \mathbb{N}$, si

$$f \in O(n^k) \Rightarrow f \in O(n^{k+1}) \Rightarrow \exists c, F \text{ s.t. } A \in F \Rightarrow f \leq c * n^k$$

$$f \in O(n^k) \equiv f \leq c * n^k$$

$$f \in O(m^{k+1}) \Rightarrow f \leq C \cdot m^{k+1}$$

$$\begin{aligned} f \leq C \cdot m^k &\Rightarrow f \leq C \cdot m^{k+1} \\ &\Rightarrow C \cdot m^k \leq C \cdot m^{k+1} \\ &\Rightarrow C \cdot m^k \leq C \cdot n^k \cdot m \\ &\Rightarrow 1 \leq m \end{aligned}$$

$\hookrightarrow m \geq m_0$ (en $m_0 \in \mathbb{N}$)

c) $f \in \underbrace{O(\log n)}_{\text{ }} \Rightarrow f \in O(n)$

$$f \leq C * \log n \Rightarrow f \leq C * m \quad C > 0 \wedge C \in \mathbb{R}$$

$$m > m_0 \quad m_0 \in \mathbb{N}$$

Né que $\log m$ es mucho más chico que m a la larga, por lo tanto no tiene sentido que $f \leq O(\log n) \Rightarrow f \notin O(n)$

$$\begin{matrix} \text{y } O(\log n) \leq \underbrace{O(m)}_{\log m} & . \\ \vdots & \log m \\ 1 & \vdots \\ 1 & \end{matrix}$$

$$\Rightarrow C * \log m \leq C * m$$

$$\Rightarrow \log m \leq m \quad \text{y esto es V A } m \geq m_0 \text{ (suficientemente grande)}$$

$$\Rightarrow \log(n) = O(n)$$

$$f \leq \log m \leq C * m$$

$$m \log(m+k)$$

$\Theta(n) = \Omega(n)$

$n \cdot m$

$\Theta(n) = \Omega(m)$

$M + M$

f) ALGORITMOQUEHACEALGO:

```

1: function ALGORITMOQUEHACEALGO(arreglo A)
2:   int i := 1; int j := 1;
3:   int suma := 1; int count := 0;
4:   while  $i \leq tam(A)$  do
5:     if  $i \neq A[i]$  then
6:       count := count+1;
7:     end if
8:     j := 1;
9:     while  $j \leq count$  do
10:      int k := 1;
11:      while  $k \leq tam(A)$  do
12:        suma := suma + A[k];
13:        k := k * 2;
14:      end while
15:      j := j+1;
16:    end while
17:    i := i+1;
18:  end while
19:  return suma

```

$$\sum_{i=1}^M \sum_{j=1}^{i \cdot \log(m)} \sum_{k=1}^{m} \Theta(1)$$

$$\sum_{i=1}^M \sum_{k=1}^i \log(m) \cdot \Theta(1)$$

Como k es constante
aproxima log(m)

Límite dentro del ciclo.

$$= \sum_{i=1}^M i \cdot \log(m) \cdot \Theta(1)$$

$$= \log(m) \cdot \Theta(1) \cdot \sum_{i=1}^n i$$

$$= \log(m) \cdot \Theta(1) \cdot \frac{n(n+1)}{2}$$

$$= \log(n) \cdot \Theta(1) \cdot (n^2 + n)$$

$$= \Theta(\log(m)) \cdot \Theta(1) \cdot \Theta(n^2)$$

$$= \Theta(\log(m) \cdot n^2)$$

