Algunos Big O **ordenados por complejidad**
1. O(log(log(n)))
2. O(log(n))
3. O(n)
4. O(n*log(n))
5. O(n^2)
6. O(2^n)
7. O(n!)
8. O((n!)^2)
9. O((2n)!)

Búsqueda lineal

```
function busquedaLineal(A: Int[], e: Int)
    n = Long(A)
    for i = 0..n do
        if A[i] = e then
            return true
    return false
```

En el mejor de los casos asignamos una vez, entramos al loop una vez, chequeamos una vez y retornamos una vez

$$f_{\text{mejor}} = \Theta(1) + \Theta(1) + \Theta(1) + \Theta(1) = \Theta(\max\{1, 1, 1, 1\}) = \Theta(1)$$

En el peor de los casos, tengo que recorrer toodo el loop

$$f_{\text{peor}} = \Theta(1) + \sum_{i=0}^{i<n} (\Theta(1)) + \Theta(1)$$
$$f_{\text{peor}} = \Theta(1) + \Theta(n) + \Theta(1)$$
$$f_{\text{peor}} = \Theta(n)$$

Demostrar que
1. $n^2 + 5n + 3 \in \Omega(n)$

Facilon
$(\forall n \in \mathbb{N})(n^2 + 5n + 3 \geq n) \Rightarrow n^2 + 5n + 3 \in \Omega(n)$

2. $n^2 + 5n + 3 \in 0 = (n)$

También, re trivial
$(\forall n \in \mathbb{N})(n^2 + 5n + 3 \geq n) \Rightarrow (\nexists n \in \mathbb{N})(n^2 + 5n + 3 < n) \Rightarrow f \notin O(n)$

Juguemos con la propiedad de límites

$$\begin{aligned} l &= \lim_{n \to \infty} \frac{n^2 + 5n + 3}{n} \\ &= \lim_{n \to \infty} \left(n + 5 + \frac{3}{n} \right) \\ &= \infty \end{aligned}$$

De esto podemos afirmar que $f \in \Omega(n)$ y $f \notin O(n)$

Encontrar la peor y la mejor ejecución

1. Se tiene una matriz A, de $n \times n$ números naturales, de manera que $A[i, j]$ representa al elemento en la fila i y columna j ($1 \leq i, j \leq n$). Se sabe que el acceso a un elemento cualquiera se realiza en tiempo $O(1)$, así como la obtención de la dimensión de la matriz (función `Long`). Una matriz en degradé es una en la que todos los elementos de la matriz son distintos y que todas las filas y columnas de la matriz están ordenadas de forma creciente (es decir, $i < n \Rightarrow A[i, j] < A[i + 1, j]$ y $j < n \Rightarrow A[i, j] < A[i, j + 1]$), como se aprecia en los ejemplos A_1 y A_2 de más abajo.

```
function valorEnMatriz(Matriz de naturales A, Natural val)
  n = Long(A)
  i = 0
  while i < n and A[0, i] ≤ val do
    i = i + 1
  colLim = i - 1
  i = 0
  while i < n and A[i, 0] ≤ val do
    i = i + 1
  filLim = i - 1
  for i = 0 to filLim do
    for j = 0 to colLim do
      if A[i, j] = val then
        return true
  return false
```

Encontrar el mejor caso, y el peor caso.

En el mejor de los casos el número que quiero está en la primer celda de la matriz. Ergo, es $\Theta(1)$ En el peor de los casos debo recorrer dos listas de n elementos. Luego, tendría que

$$
\begin{aligned}
f_{\text{peor}} &= \sum_{i=0}^{i<n}(\Theta(1)) + \sum_{i=0}^{i<n}(\Theta(1)) + \sum_{i=0}^{i<n}\left(\sum_{i=0}^{i<n}(\Theta(1))\right) \\
&= 2\Theta(n) + \Theta(n^2) \\
&= \Theta(n^2)
\end{aligned}
$$

2.

```
function busquedaBinaria(Arreglo de Enteros A, Natural e)
    n = Long(A)
    i = 0
    j = n - 1
    while i ≠ j do
        m = (i + j) / 2
        if A[m] > e then
            j = m - 1
        else
            i = m
    devolver A[i] == e
```

Notemos que el peor de los casos ocurre cuando $i = j$, veamos un caso "elemental" de esto, a ver a dónde llegamos

$$\begin{aligned} i = j &\Leftrightarrow 0 = n - 1 \\ &\Leftrightarrow n = 1 \end{aligned}$$

La única matriz que cumple esto es $n = 1$, es decir, $A = [[k]]$. Bueno, si $n = 1 \rightarrow f \in \Theta(1)$, que además es $\Theta(n)$, para este caso particular. Sigan viendo

Supongamos un caso dónde esto no se cumpla, o sea, veamos $\forall n \in \mathbb{N}$

	m	i	j
0°	$\frac{i+j}{2}$	0	n-1
1°	$\frac{n-1}{2}$	$0 \mid \frac{n-1}{2}$	$\frac{n-3}{2} \mid n-1$
2°	$\frac{n-3}{4} \mid \frac{(n-1)+(n-3)}{4} = \frac{2n-4}{4} = \frac{n-2}{2} \mid n-1 \mid$	...	...

Bueno, me imagino que esto me diría algo, pero creo que un mejor razonamiento es este:

En la primer iteración, me va a quedar la mitad de la lista por revisar $\frac{|A|}{2}$. En la segunda $\frac{|A|}{2^2}$. En la tercera $\frac{|A|}{2^2}$ ... Bueno, así hasta que me al llegar al final (cuando queda un elemento), tengo $\frac{|A|}{2^k}$. Cómo debe quedarme un elemento, puedo pensar la siguiente ecuación.

$$\begin{aligned} \frac{|A|}{2^k} &= 1 \\ |A| &= 2^k \\ \log_2(|A|) &= k \end{aligned}$$

Es decir, en general, para una lista con n elementos, vamos a tener $\log(n)$ pasos. Podemos ignorar la base del logaritmo pues es mayor que uno, por propiedad.

Luego, $f \in \Theta(\log(n))$