Complejidad

- ¿Cuántos recursos se necesitan para resolver un problema?

  En un sentido amplio.
  - Tiempo.
  - Espacio (memoria).
  - (Consumo de energía).

- Nos van a interesar los problemas decidibles.

- Si tenemos un lenguaje A
  Por ejemplo $A = \{ a^k b^k \mid k \in \mathbb{N}_0 \}$
  es decidible.

  $\cancel{a}\cancel{a}\cancel{a}\cancel{a}\cancel{b}\cancel{b}\cancel{b}\cancel{b}$

  Existe una M.T. M tq. $M(w) = \begin{cases} \text{acepta} & \text{si } w \in A \\ \text{rechaza} & \text{si } w \notin A. \end{cases}$

- ¿Cuánto tarda la M.T. M en aceptar o rechazar una cadena?

- Nos va a interesar la cantidad de pasos $T(w)$ que requiere una M.T. para decidir si acepta o rechaza una cadena w.

- En general vamos a expresar T en función de la longitud de la cadena de entrada.

- $\boxed{1}\boxed{0}\boxed{2}\boxed{4}$ $\quad\quad$ $T\cancel{(n)} \cancel{=} n$

  $\Sigma = \{0, 1, 2, \dots, 9, \dots\}$

- En particular, si escribimos un número en una base, el tamaño de la entrada no es el valor del número sino la cantidad de dígitos del número.

- Por ejemplo, un número n se escribe con $\lfloor \log_{10} n \rfloor + 1$.

- Entonces

$$T(d) \approx 10^d$$

$$\log_{10} 1 = 0$$
$$\log_{10} 10 = 1$$
$$\log_{10} 100 = 2$$
$$\log_{10} 1000 = 3$$

$$d = \lfloor \log_{10} n \rfloor + 1$$
$$\approx \log_{10} n$$

"$T(n)$: Tiempo que tarda la M.T. M en terminar cuando se la alimenta con una cadena w de longitud n."

aaba
abaa
abab

Def. Sea M una M.T. que siempre termina.

- Definimos el tiempo de ejecución en mejor caso para M así:

$$T_{mejor} : \mathbb{N} \longrightarrow \mathbb{N}$$

$$T_{mejor}(n) = \text{mín} \left\{ \begin{array}{l} \text{Cantidad de pasos que tarda } M(w) \\ \text{en terminar} \end{array} \;\middle|\; |w| = n \right\}$$

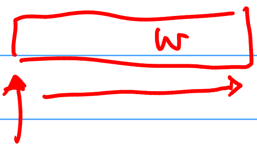
- Definimos el tiempo de ejecución en peor caso de M así:

$$T_{peor} : \mathbb{N} \longrightarrow \mathbb{N}$$

$$T_{peor}(n) = \text{máx} \left\{ \begin{array}{l} \text{cantidad de pasos que tarda } M(w) \\ \text{en terminar} \end{array} \;\middle|\; |w| = n \right\}.$$

Ejemplo.

- $A = \{ w \in \{0,1\}^* \mid w \text{ no tiene ningún } 1 \}$.
- M M.T. que decide A.



$$T_{mejor}(n) = 1$$

$$T_{peor}(n) = n$$

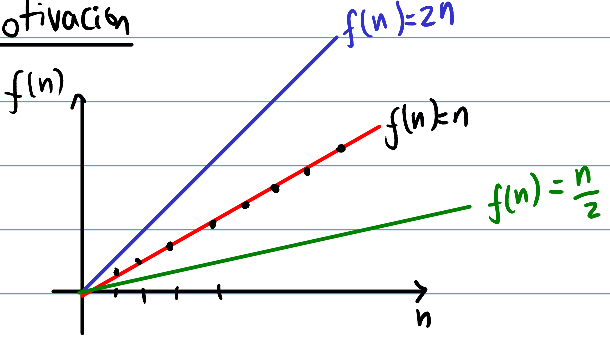
$$T_{peor}(n) = 7n^2 - 3n + 28 \in O(n^2)$$

Definimos el tiempo de ejecución en caso promedio de M así:

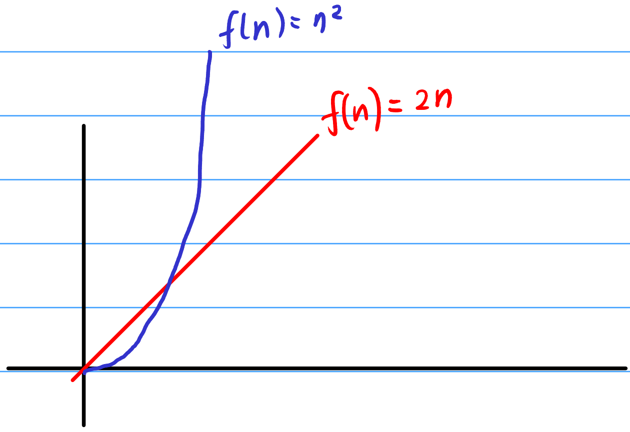
$$T_{\text{promedio}}(n) = \frac{\displaystyle\sum_{\substack{w \in \Sigma^* \\ |w|=n}} \left(\text{Cantidad de pasos que tarda } M(w) \text{ en terminar}\right)}{\underbrace{\#\{w \in \Sigma^* \mid |w| = n\}}_{|\Sigma|^n}}$$

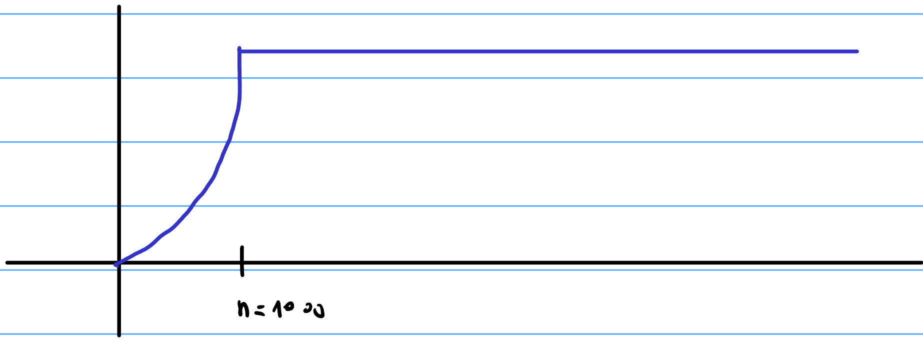
$$f : \mathbb{N} \longrightarrow \mathbb{R}^+$$

Motivación



$2n,\ \frac{n}{2},\ n \in O(n)$





Resumen:

1) No nos interesan los factores constantes.
2) No nos interesan los valores que toma la función cuando n es chico.

Def. Sean $f, g : \mathbb{N} \longrightarrow \mathbb{R}^+$.

Decimos que $f \in O(g)$ "f es más chica que g". "g domina a f" asintóticamente

si existen una constante $c > 0$ y un $n_0 \in \mathbb{N}$ tales que para todo $n \geq n_0$ vale:

$$f(n) \leq c \cdot g(n).$$

$f(n) = n$
$g(n) = n+5$
$f \in O(g)$

<u>Ejemplo</u>. $n \in O(n+5)$

Tomando $C := 1$, $n_0 := 1$

$$n \leq \overset{1}{\cancel{c}} \cdot (n+5) \quad \forall n \geq \overset{1}{\cancel{n_0}}.$$

<u>Ejemplo</u>. $n+5 \in O(n)$.

Tomando $c := 2$ y $n_0 := 5$

$$n+5 \leq n+n = 2 \cdot n \qquad \forall n \geq 5$$

<u>Lema</u>. Si $f_1, f_2 \in O(g)$
entonces $\underbrace{f_1 + f_2}_{= f} \in O(g)$.

$f(n) = f_1(n) + f_2(n)$

<u>Dem</u>. Supongamos que $f_1, f_2 \in O(g)$.
Es decir,

- Como $f_1 \in O(g)$, existen $c_1 > 0$ y un $n_1 \in \mathbb{N}$ tales que

$$f_1(n) \leq c_1 \cdot g(n) \qquad \forall n \geq n_1.$$

- Como $f_2 \in O(g)$, existen $c_2 > 0$ y un $n_2 \in \mathbb{N}$ tales que

$$f_2(n) \leq c_2 \cdot g(n) \qquad \forall n \geq n_2.$$

- Queremos hallar una constante $c > 0$ y un $n_0 \in \mathbb{N}$ tales que

$$f_1(n) + f_2(n) \leq c \cdot g(n) \qquad \forall n \geq n_0.$$

- Podemos tomar $c := c_1 + c_2$ y $n_0 := \text{máx}\{n_1, n_2\}$.

Entonces si $n \geq n_0 = \text{máx}\{n_1, n_2\}$, sabemos que $n \geq n_1$ y $n \geq n_2$.
Por lo tanto:

$$f_1(n) + f_2(n) \leq c_1 \cdot g(n) + f_2(n) \leq c_1 \cdot g(n) + c_2 \cdot g(n)$$

$$= (c_1 + c_2) \cdot g(n)$$
$$= c \cdot g(n)$$
$$\forall n \geq n_0.$$

<u>Ejemplo</u>. $3n^2 + n \in O(n^2)$.

<u>Dem</u>. Por el lema anterior, bastaría ver que $3n^2 \in O(n^2)$ ✓ y $n \in O(n^2)$. ✓

- Efectivamente,

$$3n^2 \in O(n^2) \quad \text{pues} \quad 3n^2 \leq \underbrace{3}_{c} \cdot n^2 \quad \forall n \geq \underbrace{1}_{n_0}$$

Y

$$n \in O(n^2) \quad \text{pues} \quad n \leq n \cdot n = \underbrace{1}_{c} \cdot n^2 \quad \forall n \geq \underbrace{1}_{n_0}$$

Lema. Si $f : \mathbb{N} \to \mathbb{R}^+$
es una función polinomial,
es decir

$$f(n) = a_0 + a_1 \cdot n + a_2 \cdot n^2 + \dots + a_d \cdot n^d$$

Suponiendo que $a_d \neq 0$.

Entonces $f \in O(n^d)$.

Dem. Por el lema de la suma,
basta ver que

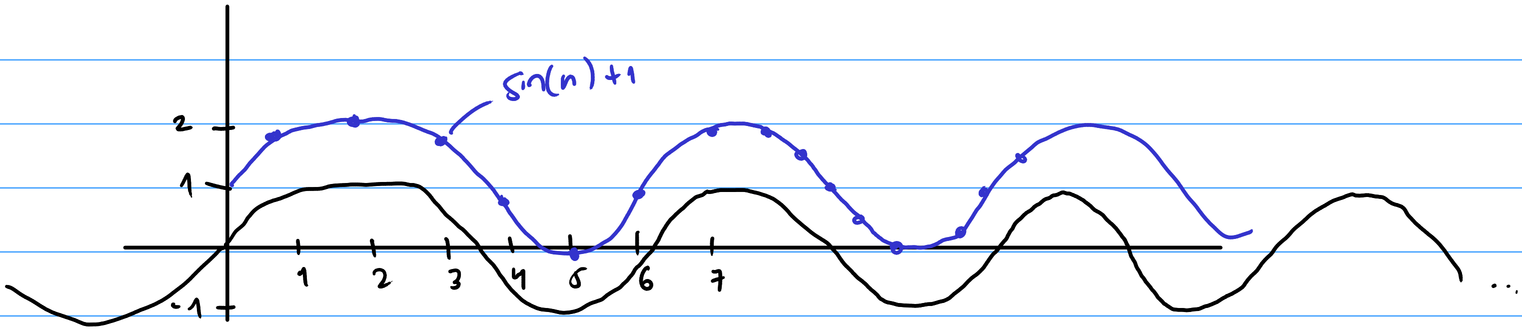
$$a_0 \cdot n^0 \in O(n^d)$$
$$a_1 \cdot n^1 \in O(n^d)$$
$$\vdots$$
$$a_d \cdot n^d \in O(n^d).$$

Veamos que para cualquier $i \in 0..d$ vale $a_i \cdot n^i \in O(n^d)$.

Efectivamente:

$$a_i \cdot n^i \leq a_i \cdot n^i \cdot \underbrace{n^{d-i}}_{\substack{\in \mathbb{N} \\ \geq 1}} = \underbrace{a_i}_{c} \cdot n^d \qquad \forall n \geq \underbrace{1}_{n_0}$$

Ejemplo. $\sin(n)+1 \in O(1)$



$$\sin(n)+1 \leq 2 = 2 \cdot f(n) \qquad \forall n \geq 1$$

$$f(n) = 1$$

$$\sin(n)+1 \in O(f) = O(1)$$

Ejemplo. $n \notin O(1)$.

Dem. Por el absurdo.

Supongamos que $n \in O(1)$.

- Entonces existen $C > 0$ y un $n_0 \in \mathbb{N}$ tq.

$$n \leq C \qquad \forall\, n \geq n_0.$$

- Esto es absurdo. Por ejemplo si tomamos

$$n := \max\{C, n_0\} + 1$$

tendríamos que $C+1 \leq n \leq C$.

Ejemplo. $n^2 \notin O(n)$.

Dem. Por el absurdo, supongamos que

$$n^2 \in O(n).$$

Entonces existen una $c>0$ y un $n_0 \in \mathbb{N}$ tq.

$$n^2 \leq c.n \qquad \forall n \geq n_0.$$

Entonces tendríamos que

$$n \leq c \qquad \forall n \geq n_0.$$

Absurdo.

Propiedad (transitividad).

Si $f \in O(g)$ y $g \in O(h)$

entonces

$$f \in O(h).$$

Dem. Supongamos que $f \in O(g)$ y $g \in O(h)$.

- Como $f \in O(g)$, existen $c_1 > 0$ y $n_1 \in \mathbb{N}$ tq.

$$f(n) \leq c_1 \cdot g(n) \qquad \forall n \geq n_1$$

- Como $g \in O(h)$, existen $c_2 > 0$ y $n_2 \in \mathbb{N}$ tq.

$$g(n) \leq c_2 \cdot h(n) \qquad \forall n \geq n_2$$

- Queremos ver que existen $c > 0$ y $n_0 \in \mathbb{N}$ tq.

$$f(n) \leq c \cdot h(n) \qquad \forall n \geq n_0.$$

Tomando $n_0 := \max\{n_1, n_2\}$ y $c := c_1 \cdot c_2$.

tenemos que:

$$f(n) \leq c_1 \cdot g(n) \leq \underbrace{c_1 \cdot c_2}_{c} \cdot h(n) \qquad \forall n \geq n_0.$$

✓

• $\log_b(n) = x$ donde $b^x = n$

$\log_3(n) \overset{?}{\in} O(\log_2(n))$ ✓ $\log_3 n \leq \log_2 n$

$\log_2(n) \overset{?}{\in} O(\log_3(n))$?

Lema (Orden de los logaritmos).

Si $a, b > 1$ entonces $\log_a(n) \in O(\log_b(n))$.

Dem.

Recordemos que $\log_a(n) = \dfrac{\log_b(n)}{\log_b(a)}$

pues $\log_b(n) = x$

$\log_b(a) = y$

tenemos que $b^x = n$, $b^y = a$.

Por lo tanto $a^{1/y} = (b^y)^{1/y} = b$.

Por lo tanto $a^{x/y} = (a^{1/y})^x = b^x = n$.

Por lo tanto $\log_a(n) = x/y = \dfrac{\log_b(n)}{\log_b(a)}$.

Notemos que:

$$\log_a(n) \leq \frac{\log_b(n)}{\log_b(a)} = \underbrace{\frac{1}{\log_b(a)}}_{c} \cdot \log_b(n) \qquad \forall n \geq 1.$$

Por lo tanto $\log_a(n) \in O(\log_b(n))$.

Ejemplo. $2^n \in O(3^n)$

$$2^n \leq 3^n \quad \forall n.$$

Pero $3^n \notin O(2^n)$.

Dem. Por el absurdo, supongamos que $3^n \in O(2^n)$.

Es decir existen $c>0$ y $n_0 \in \mathbb{N}$ tq.

$$3^n \leq c \cdot 2^n \quad \forall n \geq n_0.$$

Entonces

$$\frac{3^n}{2^n} \leq c \quad \forall n \geq n_0$$

Entonces

$$\left(\frac{3}{2}\right)^n \leq c \quad \forall n \geq n_0.$$

Absurdo. $\frac{3}{2} > 1$ $\quad \frac{3}{2}^n \xrightarrow[n \to +\infty]{} +\infty$

Ejemplo. Si $f_1, f_2 \in O(g)$

Ya sabíamos que $f_1 + f_2 \in O(g)$.

¿Qué pasará con $f_1 \cdot f_2$?

No siempre $f_1 \cdot f_2 \in O(g)$.

Por ejemplo, $f_1(n) = n$ $f_2(n) = n$

$g(n) = n$

$f_1, f_2 \in O(g)$

pero $f_1 \cdot f_2 \notin O(g)$.

Lema. Si $f_1 \in O(g_1)$ y $f_2 \in O(g_2)$

entonces $\underbrace{f_1 \cdot f_2}_{f(n)=f_1(n)\cdot f_2(n)} \in O(\underbrace{g_1 \cdot g_2}_{g(n)=g_1(n)\cdot g_2(n)})$.

Dem. Supongamos que $f_1 \in O(g_1)$
y $f_2 \in O(g_2)$.

Es decir:

- Como $f_1 \in O(g_1)$, existen $c_1 > 0$ y $n_1 \in \mathbb{N}$ tq.

$$f_1(n) \leq c_1 \cdot g_1(n) \quad \forall n \geq n_1$$

- Como $f_2 \in O(g_2)$, existen $c_2 > 0$ y $n_2 \in \mathbb{N}$ tq.

$$f_2(n) \leq c_2 \cdot g_2(n) \quad \forall n \geq n_2$$

- Queremos ver que existen $c > 0$ y $n_0 \in \mathbb{N}$ tq.

$$f_1(n) \cdot f_2(n) \leq c \cdot g_1(n) \cdot g_2(n) \quad \forall n \geq n_0$$

- En efecto, tomando $c := c_1 \cdot c_2$, $n_0 := \max\{n_1, n_2\}$,

tenemos que si $n \geq n_0$

$$\begin{aligned} f_1(n) \cdot f_2(n) &\leq c_1 \cdot g_1(n) \cdot f_2(n) \\ &\leq c_1 \cdot g_1(n) \cdot c_2 \cdot g_2(n) \\ &= \underbrace{c_1 \cdot c_2}_{c} \cdot g_1(n) \cdot g_2(n) \end{aligned}$$

✓

Lema. "Exponencial domina a lineal": $n \in O(2^n)$.

Dem. Veamos que $n \leq 2^n \quad \forall n \geq 1$

Por inducción en n:

- $\underline{n=1}$.

$$1 \leq 2^1 = 2 \quad \checkmark$$

- $\underline{P(n) \Rightarrow P(n+1)}$:

Supongamos, por HI, que $n \leq 2^n$.

Entonces $n+1 \leq 2^n + 1 \leq 2^n + 2^n = 2 \cdot 2^n = 2^{n+1}$

Lema. ("Lineal domina a logarítmica").

$$\log_2(n) \in O(n).$$

Dem. Veamos que $\log_2(n) \leq n$.

Basta notar que $n \leq 2^n$ (lo vimos en el lema anterior).

porque

$$\log_2(n) \leq \log_2(2^n) = n \quad \checkmark$$

Teorema ("Exponencial domina a polinomial").

Si $p \geq 0$, $n^p \in O(2^n)$.

Dem.

Afirmo: $\forall k \in \mathbb{N} \ \forall n \geq p$ vale:

$$\left(\frac{n}{p}\right)^k \leq \left(2^{n/p}\right)^k$$

Dem.

$$\frac{n}{p} \leq 2^{n/p}$$

Por lo tanto

$$\left(\frac{n}{p}\right)^k = \underbrace{\left(\frac{n}{p}\right) \cdot \ldots \cdot \left(\frac{n}{p}\right)}_{k \text{ veces}} \leq \underbrace{\left(2^{n/p}\right) \cdot \ldots \cdot \left(2^{n/p}\right)}_{k \text{ veces}} = \left(2^{n/p}\right)^k$$

En particular, si $k = p$, $\forall n \geq p$ tendríamos

$$\frac{n^p}{p^p} = \left(\frac{n}{p}\right)^p \leq \left(2^{n/p}\right)^p = 2^{\frac{\cancel{p}n}{\cancel{p}}} = 2^n$$

Por lo tanto:

$$n^p \leq \underbrace{p^p}_{c} \cdot 2^n \quad \forall n \geq \underbrace{p}_{n_0}$$

Luego $n^p \in O(2^n)$.

$n^p \in O(2^n)$

$n^p \in O(b^n)$ si $b > 1$

$$8n^2 + 7n + 3 \in O(1.03^n)$$

Notación.

- $3n^2 + O(n)$ significa $3n^2 + f(n)$ donde $f \in O(n)$.

- $2^{O(n)}$ significa $2^{f(n)}$ donde $f \in O(n)$.

- $n^{O(1)}$ significa $n^{f(n)}$ donde $f \in O(1)$.

$n^{\sin(n)}$

Notación

- $\boxed{f \in O(g)}$ "$f \leq g$"
- $f \in \Omega(g)$ $\leftrightarrow$ $g \in O(f)$ $\quad n^3 \in \Omega(n)$ "$f \geq g$"
- $f \in \Theta(g) \longleftrightarrow f \in O(g) \wedge g \in O(f)$
- $f \in o(g)$ "$f < g$"
- $f \in \omega(g) \longleftrightarrow g \in o(f)$ "$f > g$"

"$f < g$"

Def. Si $f, g : \mathbb{N} \longrightarrow \mathbb{R}^+$

Decimos que $f \in o(g)$ si

$$\lim_{n \to +\infty} \frac{f(n)}{g(n)} = 0$$

Es decir, $\forall \varepsilon > 0 \ \exists n_0 \in \mathbb{N} \ \forall n \geq n_0$.

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

Ejemplo: $n \in o(n^2)$ pues $\frac{n}{n^2} = \frac{1}{n} \xrightarrow[n \to +\infty]{} 0$

$n \in O(2n)$ ✓ pero $n \notin o(2n)$ pues $\frac{n}{2n} = \frac{1}{2} \not\xrightarrow[n \to +\infty]{} 0$

3	5	~~9~~	~~2~~	~~1~~	4
		2	1	4	9

Bubblesort

Entrada: una lista L de enteros de longitud n.

Salida: la lista L ordenada de menor a mayor.

Dada una lista L de longitud n

- en mejor caso hace $O(n)$ comparaciones
- en peor caso hace $O(n^2)$ comparaciones.

Algoritmo:

```
for i=1 to n {
    hiceSwaps := false
    for j= 0 .. n-2 {
        if(L[j] > L[j+1]) {
            Swap ( L[j], L[j+1])
            hiceswaps := true
        }
    }
    if !hiceSwaps { return; }
}
```

Quicksort

Entrada: Una lista L de enteros de longitud n.

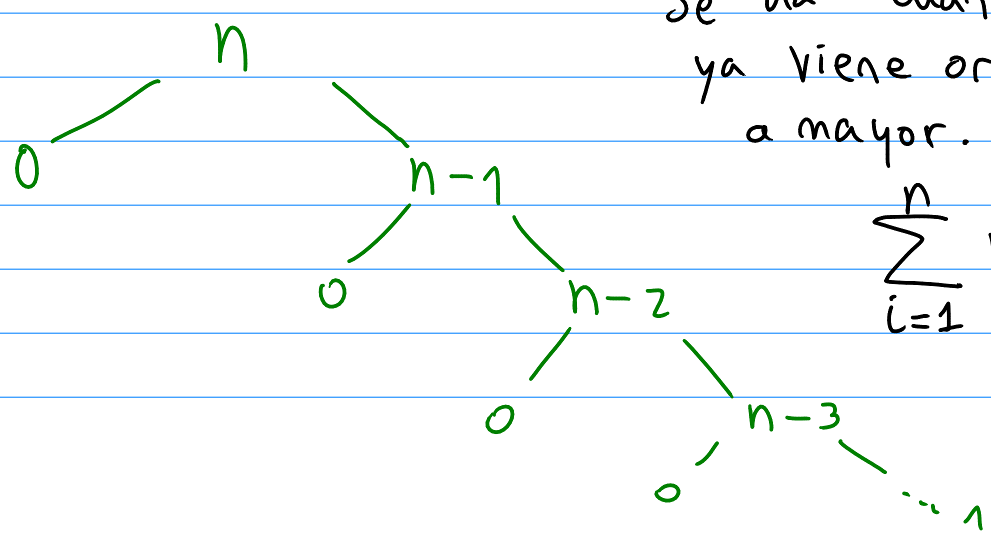
Salida: La lista L ordenada de menor a mayor.

Algoritmo:

```
if n ≤ 1 then return;
p := L[0]
i := 1
j := n
while i < j {
        if L[i] ≤ p {
                i := i+1
        } else {
                swap(L[i], L[j-1])
                j := j-1
        }
}
swap(L[0], L[i-1])
Quicksort(L[0..i-1])
Quicksort(L[i..n])
```

hace $O(n)$ comparaciones

Peor caso



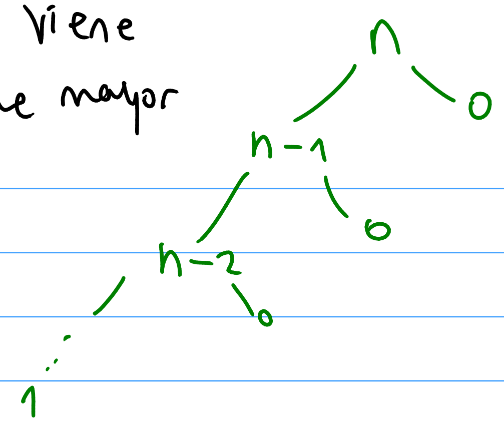
Se da cuando la lista ya viene ordenada de menor a mayor. Se hacen

$$\sum_{i=1}^{n} n-i \in O(n^2)$$

comparaciones.

De hecho, Quicksort en caso promedio hace $O(n \cdot \log n)$ comparaciones.

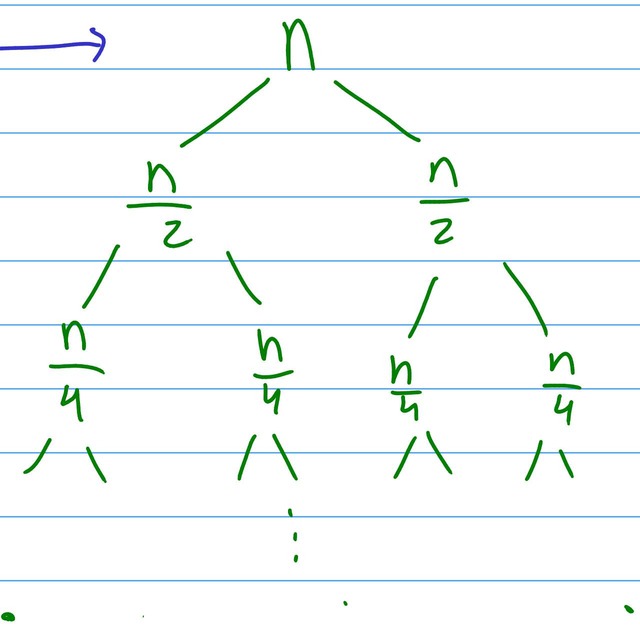
Si la lista viene ordenada de mayor a menor



también vamos a alcanzar el peor caso, y se hacen $O(n^2)$ comparaciones.

El mejor caso se alcanza cuando el pivote elegido parte la lista en dos partes iguales (es decir de tamaños $\lfloor \frac{n-1}{2} \rfloor$ y $n-1-\lfloor \frac{n-1}{2} \rfloor$).

Cada nivel hace n comparaciones



Este árbol tiene altura $\log_2(n)$.

En total se hacen $O(n \cdot \log(n))$ comparaciones.

$$n \cdot \log(n) \in o(n^2)$$

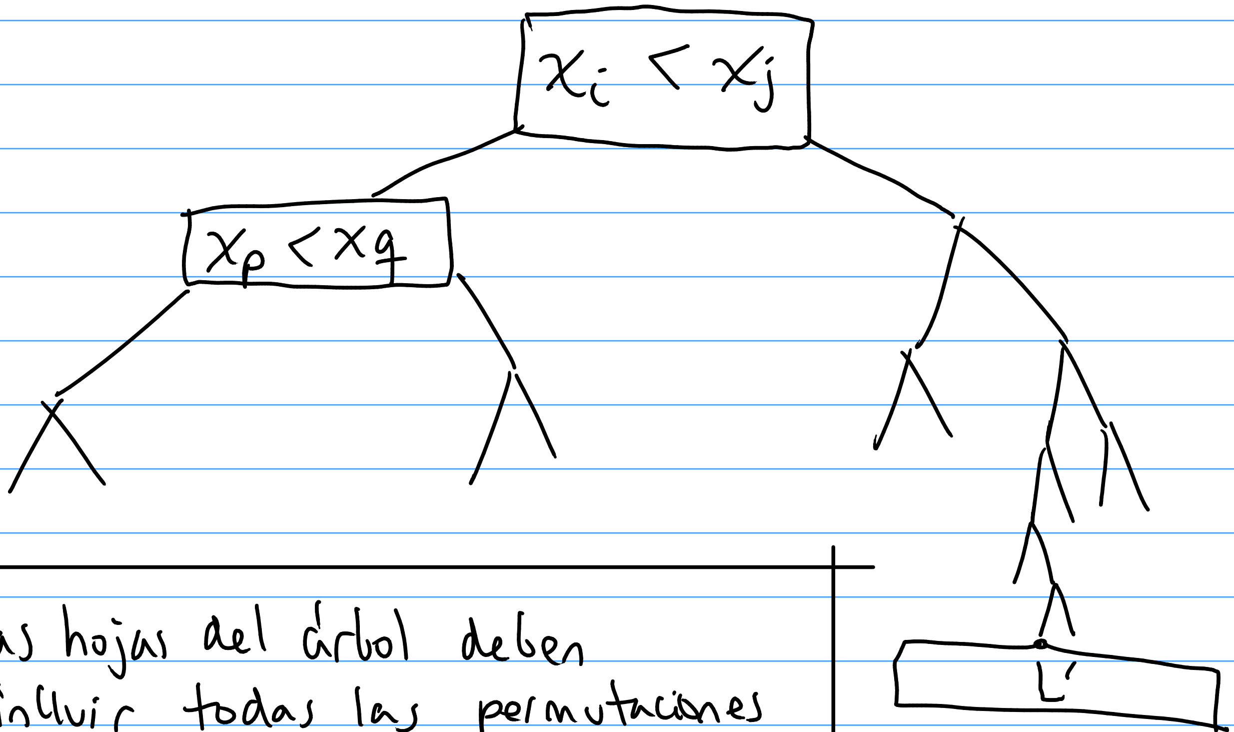
Teorema. Si A es un algoritmo que ordena listas de menor a mayor, y A está basado en comparaciones, entonces en peor caso hace

$$\Omega(n \cdot \log(n)) \quad \text{comparaciones.}$$

donde n es el tamaño de la lista.

Dem. Cuando el algoritmo A procesa una lista L.

$$L = [x_1, x_2, \ldots, x_n]$$



Las hojas del árbol deben incluir todas las permutaciones de la lista $[x_1, x_2, \ldots, x_n]$.

L' es una permutación de la lista de entrada.

La cantidad de comparaciones que se hacen en peor caso están dadas por la altura del árbol.

$[1,2,3]$

123
132
213
231
312
321

$3! = 6$

- La altura mínima que podría tener este árbol es

$$\log_2(\#hojas) \geq \log_2(n!)$$
$$= \log_2\left(\prod_{i=1}^{n} i\right)$$
$$= \sum_{i=1}^{n} \log_2(i)$$
$$\geq \sum_{i=1}^{n/2} \log_2(n/2)$$
$$= (n/2) \cdot \log_2(n/2)$$
$$= \frac{n}{2} \cdot (\log_2(n) - \log_2(2))$$
$$= \underbrace{\frac{n}{2} \cdot \log_2(n)}_{\Omega(n \cdot \log n)} - \underbrace{\frac{n}{2} \cdot \log_2(2)}_{\Omega(n)}$$
$$\in \Omega(n \cdot \log n)$$