

Introducción al análisis de algoritmos

$P \rightarrow n^n \leftarrow$

$f(n) \rightarrow$ Velocidad de convergencia

Por ejemplo:

$$S = 1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{6} + \frac{1}{8} - \frac{1}{10} + \frac{1}{12} - \dots$$

$\rightarrow \text{Do } [S = 1 + \sum_{i=1}^{n-1} \frac{(-1)^i}{2i}, \{n, 1000\}]$
 $\text{Print } [S] \leftarrow$

En Mathematica:

Time Constrained [expr , t ,
 "Falló el tiempo indicado"]

En el caso de S :

$\text{ClearSystemCache}[]$
 Time Constrained [$\text{Do } [1 + \sum_{i=1}^{n-1} \frac{(-1)^i}{2i}, \{n, 1000\}]$,
 1, "Falló el tiempo indicado"]

Falló el tiempo indicado

Otro algoritmo: $\text{Serie}[n] := \text{Module}[\{suma = 1, signo = -1\},$
 $\rightarrow \text{for } [i = 2, i \leq n, \text{suma} = \text{suma} + \text{signo}/(2i - 2);$
 $\text{signo} = -\text{signo}; i++];$
 $\text{If } [i = n + 1, \text{Print } [\text{suma}]]]$

En Mathematica:

Timing

ClearSystemCache[]

→ Timing [Module [dn = 10000, $1 + \sum_{i=1}^{n-1} \frac{(-1)^i}{2^i}$]]

↑ 0.078, 196642 %

ClearSystemCache[]

Timing [Module [suma = 1, signo = -1, n = 10000,

For [i = 2, i ≤ n,
 suma = suma + signo / (2ⁱ - 2);
 signo = -signo; i++;

If [i == n + 1, Return [suma]]]]]

0.125, Null

En el tema de complejidad de algoritmos:

f(n) → "orden"

Por ejemplo:

ClearSystemCache[]

→ T = Table [dn, First [Timing [Module [...]]],
 {n, 1, 10}]

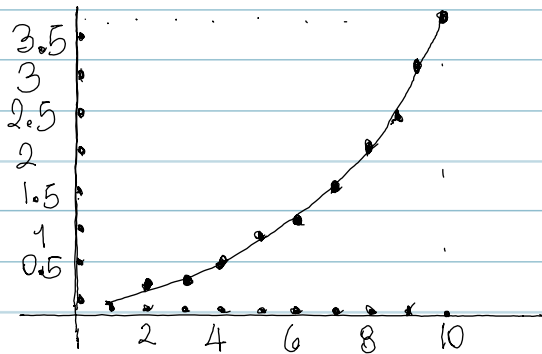
ListPlot [T]

Fit [T, {1, n, n²}, n]

% /. n → x;

h [x_] = %;

Show [ListPlot [T], Plot [h [x], {x, 1, 10}]]



1 → $n = 10\,000$
 2 → $n = 20\,000$
 ⋮
 10 → $n = 100\,000$.

Enfoque experimental,

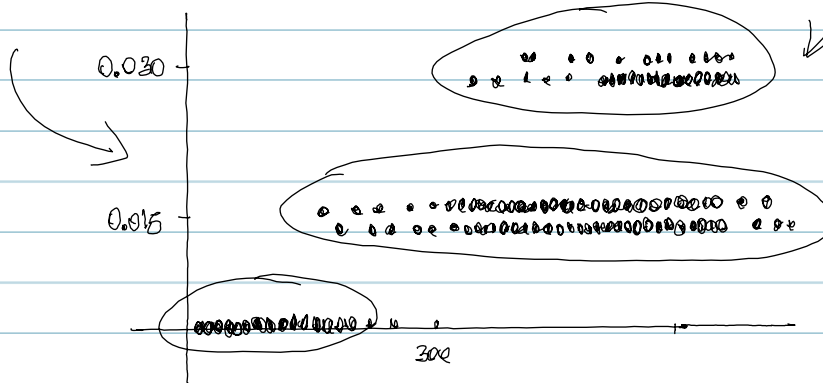
Por ejemplo:

→ Quicksort [begin, end] :=
 IF [begin < end, i = begin; j = end;
 piv = L [[floor [(begin + end) / 2]]];
 while [i ≤ j, while [L [[i]] < piv, i++];
 while [L [[j]] > piv, j--];
 IF [i ≤ j, aux = L [[i]]; L [[i]] = L [[j]]; L [[j]] = aux;
 i++; j--]; IF [begin < j, Quicksort [begin, j]];
 IF [i < end, Quicksort [i, end]];]

clear SystemCache []

→ L = L [RandomInteger [20, 10000]];
 T = Table [d[n, First [Timing [L = Append [L, RandomInteger [20, 1000]]]; Quicksort [1, Length [L]]]],
 d[n, 2, 500]];

ListPlot [T]



Notación asintótica O

$$\begin{array}{c} \downarrow \\ \boxed{f(n)} \end{array} \Rightarrow \left\{ \begin{array}{l} n! \\ n^t, t \in \mathbb{N} \\ a^n \\ \log a^n, a \in \mathbb{R}^+ \end{array} \right.$$

Si recordamos, la función que cuenta el número de pasos necesarios para resolver el juego de las torres de Hanoi:

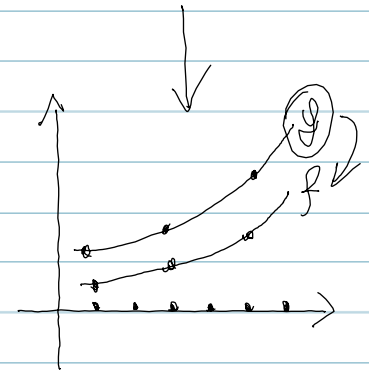
$$f(n) = 2^n - 1, \forall n, n \in \mathbb{N}$$

$$g(n) = 2^n$$

$$2^n - 1 < 2^n$$

Gráficamente:

show [Plot [2^n - 1, 2^n], {n, 0, 3},
 PlotRange -> {0, 10}, PlotStyle -> {Thick},
 Graphics [Text [Style ["f", 10], {1, 1.25},
 Text [Style ["g", 10], {1, 2.25}]]]



$O \rightarrow$ "o grande"

Definición Sean f y g funciones definidas sobre el conjunto de los números naturales. Se dice que $f(n)$ es de orden a lo sumo $g(n)$ y se denota $f(n) = O(g(n))$, si existe una constante, C , $C \in \mathbb{R}^+$, tal que:

$$|f(n)| \leq C |g(n)| \quad \forall n, n \in \mathbb{N}$$

O excepto un número finito de ellos. A la notación $O(g(n))$ se le llama notación asintótica "o grande".

$f(n) \rightarrow$ unidades de tiempo.

$$\begin{array}{c} \uparrow \\ f(n) = O(g(n)) \\ \uparrow \nearrow \\ g = (n) \end{array}$$

Al comparar: $\ln(\ln(n))$, $\ln(n)$, n (identidad), $n \ln(n)$, n^2
 n^3 y 2^n

Una forma: $\text{Table}[\{\text{Log}[\text{Log}[n]], \text{Log}[n], n, n \text{Log}[n],$
 $\left. \begin{matrix} n^2, n^3, 2^n, 2n, 2, 10 \end{matrix} \right\} // N$

Otra forma: $\text{Show}[\text{Plot}[\{\text{Log}[\text{Log}[n]], \text{Log}[n], n,$
 $\left. \begin{matrix} n \text{Log}[n], n^2, n^3, 2^n \end{matrix} \right\}, \{n, 0, 6\}, \text{PlotStyle}$
 $\rightarrow \{\text{Thick}\}]$

n : 2^n , n^3 , n^2 , $n \ln(n)$, n , $\ln(n)$ y $\ln(\ln(n))$

Veamos algunos ejemplos de la notación asintótica O grande
 Ejemplo Demuestre que $n^3 + n^2 + n + 2 = O(n^3)$.

Por definición: $c \in \mathbb{R}^+$

$$|n^3 + n^2 + n + 2| \leq c |n^3|, n \in \mathbb{N}$$

$$n^3 + n^2 + n + 2 \leq n^3 + n^3 + n^3 + 2n^3$$

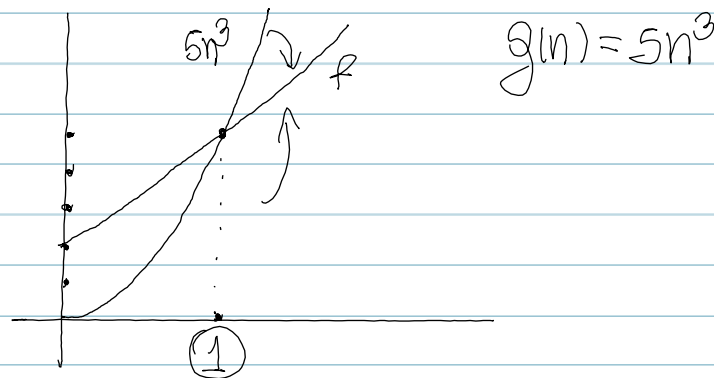
$$\Rightarrow n^3 + n^2 + n + 2 \leq 5n^3$$

$$\Rightarrow |n^3 + n^2 + n + 2| \leq 5 |n^3|, n \in \mathbb{N}$$

En Mathematica:

→ Show [Plot [{ $n^3 + n^2 + n + 2$, $5n^3$ }, {n, 0, 2}],
 Plot Range → {0, 10}, Plot Style → {Thick},
 Graphics [{Text [style ["f", 10], {0.5, 0.5³ +
 0.5² + 0.5 + 2 + 0.2}], Text [style ["5n³", 10],
 {1.02, 6}]}]]

Out []



En general:

$$f(n) = a_j n^j + a_{j-1} n^{j-1} + \dots + a_0$$

$$f(n) = O(n^j)$$

Ejemplo Pruebe que $1^j + 2^j + \dots + n^j = O(n^{j+1})$ con
 $j \in \mathbb{N}$! Enfatice utilizando el software Mathematica la cota
 superior para $j = 1, 2, \dots, 10$.

$$1^j + 2^j + \dots + n^j \leq \overbrace{1^j + n^j + \dots + n^j}^{j+1 \text{ términos}} \leq n \cdot n^j$$

$$\Rightarrow 1^j + 2^j + \dots + n^j \leq n^{j+1} \quad \uparrow \quad c = 1$$

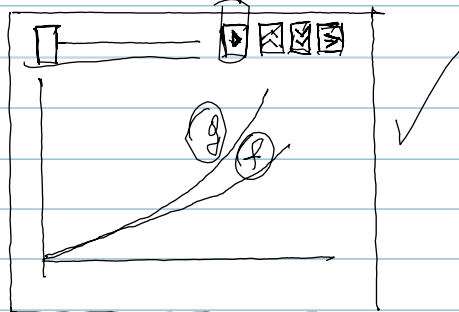
$$|1^j + \dots + n^j| = 1^j + \dots + n^j \quad \uparrow \quad c = 1 \quad \text{y} \quad |n^{j+1}| = n^{j+1}$$

In Mathematica:

→ For [j=1, j≤10, s[n_]:=Sum[i^8, {i, 1, n}], f[n_]:=s[n],
 g[n_]:=n^8+1; Print[show[Plot[f[n], g[n],
 {n, 0, 5}], PlotStyle->{Thick}],
 Graphics[Text[Style["f", 10], {4.6, f[4.7]+1}],
 Text[Style["g", 10], {3.9, g[4]+1}]]]; j++]

Using Animate:

Animate[show[Plot[s[n_]=Sum[i^8, {i, 1, n}],
 f[n_]:=s[n]; g[n_]:=n^8+1;
 {f[n], g[n]}, {n, 0, 5}], PlotStyle->{Thick}],
 Graphics[...], {j, 1, 10}], AnimationRunning->
 False]



Ejemplo Demuestre: $2 + 4 + \dots + 2^n = O(2^n)$.

$$2 + 4 + \dots + 2^n = 2^1 + 2^2 + \dots + 2^n$$

$$\sum_{j=0}^n r^j = \frac{r^{n+1} - 1}{r - 1}, \quad r \neq 1$$

$$2^1 + 2^2 + \dots + 2^n = \sum_{j=1}^n 2^j = \sum_{j=0}^n 2^j - 1 = \frac{2^{n+1} - 1}{2 - 1} - 1$$

$$= 2^{n+1} - 2$$

Luego:

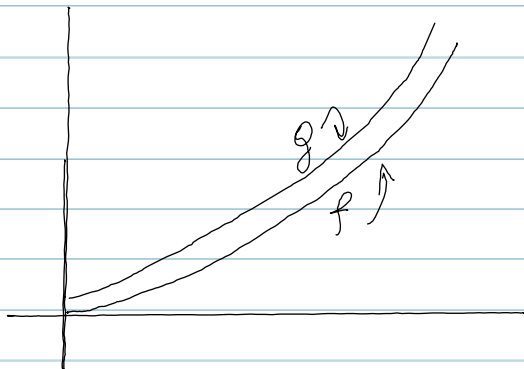
$$2^{n+1} - 2 \leq 2^{n+1}$$

$$\Rightarrow |2^{n+1} - 2| \leq 2 |2^n| \quad \forall n, n \in \mathbb{N}.$$

$$2 + 4 + \dots + 2^n = O(2^n)$$

Gráficamente con Mathematica:

Show & Plot $[f[n_] := 2^{n+1} - 2; g[n_] := 2^{n+1}; \{f[n], g[n]\}, \{n, 0, 5\}, \text{PlotStyle} \rightarrow \{\text{Thick}\}]$



↓
Ejemplo Pruebe que $\ln(n!) = O(n \ln n)$. Verifique esta igualdad por medio del software Mathematica.

$$\begin{aligned} \ln(n!) &= \ln(1 \cdot 2 \cdot \dots \cdot n) \\ &= \sum_{j=1}^n \ln j = \ln(1) + \ln(2) + \dots + \ln(n) \\ \Rightarrow \ln(n!) &\leq \sum_{j=1}^n \ln(n) = n \ln n \end{aligned}$$

$$\ln 1 \leq \ln n$$

$$\ln 2 \leq \ln n$$

$$\vdots$$

$$\ln n \leq \ln n$$

$$|\ln(n)| = \ln(n) \text{ y } |\ln \ln n| = n \ln n \quad \forall n, n \in \mathbb{N}.$$

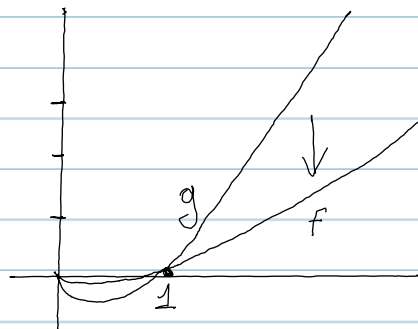
$$|\ln(n!)| \leq |\ln \ln(n)|$$

↑
C = 1.

$$\therefore \ln(n!) = O(n \ln n)$$

usando Mathematica:

↓
`Show[Plot[f[n_]:=Log[n!];
g[n_]:=nLog[n]; {f[n], g[n]} &, {n, 0, 5}, PlotStyle->{Thick}]]`



$(f(n)) \quad (g(n)) \downarrow$

Teorema Sean f y g funciones definidas sobre el conjunto de los números naturales.

1. $f(n) = O(f(n))$
- 2. $\lim_{n \rightarrow +\infty} \frac{f(n)}{g(n)} = C, C \in \mathbb{R}^+ \Rightarrow f(n) = O(g(n)) \leftarrow$
3. $n^a = O(n^b) \Leftrightarrow a \leq b, a, b \in \mathbb{R}^+$
4. $a^n = O(b^n) \Leftrightarrow a < b, a, b \in \mathbb{R}^+$
5. $\ln(n) = O(n^a) \forall a, a \in \mathbb{R}^+$

Ejemplo Pruebe que $\frac{(n+1)(4n+1)}{7n+1} = O(n)$.

$$f(n) = \frac{(n+1)(4n+1)}{7n+1}, \quad g(n) = n$$

$$\lim_{n \rightarrow +\infty} \frac{f(n)}{g(n)}$$

En Mathematica:

$$\text{Limit} \left[\frac{(n+1)(4n+1)}{7n+1}, n \rightarrow \infty \right]$$

$$\left[\frac{4}{7} \right]$$

$$f(n) = O(g(n)) \checkmark$$

Apie:

$$\begin{aligned} \rightarrow \lim_{n \rightarrow +\infty} \frac{(n+1)(4n+1)}{7n+1} &= \lim_{n \rightarrow +\infty} \frac{(n+1)(4n+1)}{n(7n+1)} \\ &= \lim_{n \rightarrow +\infty} \frac{4n^2 + 5n + 1}{7n^2 + n} = \lim_{n \rightarrow +\infty} \frac{4n^2}{7n^2} = \frac{4}{7} \end{aligned}$$

Ejemplo Verifique a través del software Mathematica: $2^n = O(n!)$.

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \rightarrow 2^n$$
$$\lim_{n \rightarrow \infty} g(n) \rightarrow n!$$

Luego:

$$\text{Limit} \left[\frac{2^n}{n!}, n \rightarrow \infty \right]$$
$$\rightarrow 0 \checkmark$$
$$f(n) = O(g(n)) \Rightarrow 2^n = O(n!)$$

Ejemplos de análisis de algoritmos

Ejemplo El algoritmo de la burbuja es un método que ordena una lista L de datos de forma ascendente o descendente. Se basa en tomar inicialmente la lista L y comparar el primer elemento con el segundo, si el primero es mayor que el segundo se intercambian y se continúa ahora comparando el segundo elemento de la nueva lista con el tercero. Se sigue de esta misma forma hasta comparar con el último elemento de L . El proceso permite dejar el número mayor, o el que posee mayor peso, al final de la lista L . Lo anterior completa una iteración del algoritmo de la burbuja. La segunda iteración se produce al repetir el mismo procedimiento con la lista L actual, exceptuando su último elemento. En la tercera iteración, se continúa de manera similar con la lista L menos sus dos últimos términos. El procedimiento continúa hasta obtener una lista de longitud uno. Al finalizar, L contiene los datos ordenados ascendentemente (similar a una burbuja en ascenso).

En Mathematica:

$L = \{k\}$ ✓

$$\rightarrow \begin{cases} n = \text{Input}["\text{Digite el tamaño de la lista:}"]; \\ \text{For } i = 1, i \leq n, m = \text{Input}["\text{Digite el dato:}"]; \\ L = \text{Append}[L, m]; i++ \end{cases}$$

(* Algoritmo de la burbuja *)

```

For [i = n, i ≥ 1,
  For [j = 0, j < i, If [L[Ej]] > L[Ej+1],
    aux = L[Ej];
    L[Ej] = L[Ej+1]; L[Ej+1] = aux; j++]; i--];
  
```

Determine su orden de convergencia O en el peor caso.

$$a_n = \underbrace{a_{n-1}} + \underbrace{n-1}, \quad a_1 = 1 \leftarrow$$

Can 'RSolve':

$$\text{RSolve}[a[n] == a[n-1] + n - 1, a[1] == 1, \\ a[n], n]$$

$$\rightarrow \{ \{ a[n] \rightarrow \frac{1}{2} (2 - n + n^2) \} \}$$

$$f(n) = \frac{n^2 - n + 2}{2} = O(n^2).$$

Ejemplo. Considere el programa recursivo que calcula el factorial de un número natural o cero. Determine para el peor y mejor caso, una notación asintótica O .

$$\rightarrow \text{Factoriales}[n_] := \text{If}[\text{Or}[n == 0, n == 1], \text{Return}[1], \\ n * \text{Factoriales}[n-1]]$$

Peor caso: $n \neq 0$ y $n \neq 1$

$$\begin{array}{c} n \rightarrow 0 \text{ o } 1 \\ n! \rightarrow \underbrace{1 + 1 + \dots + 1}_{n \text{ veces}} = n \\ \downarrow \\ n = O(n) \end{array}$$

En el mejor caso: $n = 0$, o bien, $n = 1 \rightarrow 1$
 $O(1)$

Ejemplo Analice a través de la notación asintótica O , el tiempo que tarda la peor situación del algoritmo:

CalculaSuma[n] := Module [p ← 0, ↙
↪ For [i = 1, i ≤ n, For [j = 1, j ≤ n, p ← p + 2; j++],
i++]; Print [p]]
↑

Solución

2 → For ↓
↪ p ← p + 2 → $O(1)$

→ For → n → $O(n)$.

$$\underline{O(n) \cdot O(n) = O(n \cdot n) = O(n^2)}$$

$$O(f(n)) \cdot O(g(n)) = O(f(n) \cdot g(n)).$$

↓
Ejemplo Determine O para el peor caso en:

→ { CalculaProducto [n-] :=
 Module [p=1, While [p ≤ n, p = p*3]; Print [p]] ✓

Solución:

"while" → p → 1

↓
 3^k

→ $3^k > n \Rightarrow k > \log_3 n$

$\log_3 n \rightarrow$ entero superior

$r, c \in \mathbb{R}^+, r = \log_3 n + c$

En Mathematica:

Limit $\left[\frac{\log [3, n] + c}{\log [n]}, n \rightarrow \infty \right]$

→ $\frac{1}{\log [3]} \in \mathbb{R}$

$r = O(\ln n)$

↑
 n ↗

Ejemplo El siguiente código devuelve la misma salida del programa expuesto en el ejemplo anterior. ¿Cuál de los dos algoritmos proporciona los mejores resultados en tiempo de ejecución?

→ } Calcula Otro Producto [n-] :=
 { Module [dp=1, For [i=1, i≤n, while [p≤i, p=p*3];
 i++], Print [p]]

Solución

For → $O(n)$

while → $O(\ln n)$

$$O(n) \cdot O(\ln n) = O(n \ln n)$$

←

Una comparación en Mathematica usando Timing verifica esta interpretación:

```

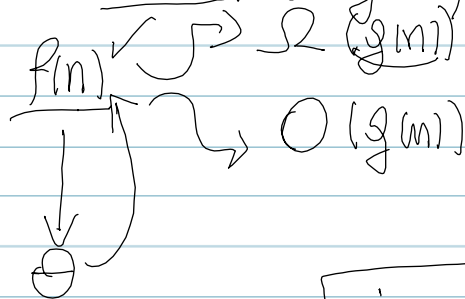
clearSystemCache[]
v = Table[First[Timing[Module[dp=1, while[p≤n,
p=p*3]; Return[p]]], First[Timing[Module[dp=1,
For[i=1, i≤n, while[p≤i, p=p*3]; i++]; Return[p]]]],
{n, 100, 2000}];
For[i=1, i≤Length[v], If[Min[v[[i]]] == v[[i, 2],
Print["Funciona mejor el algoritmo 1"], Print["Funciona mejor
el algoritmo 2"]; i++];
  
```

Otras notaciones asintóticas

→ Definición Se dice que una función $f(n)$ con dominio en el conjunto de los números naturales es de orden al menos $g(n)$ y se denota $f(n) = \Omega(g(n))$ si existe una constante real positiva c , tal que:

$$|f(n)| \geq c|g(n)| \quad \forall n, n \in \mathbb{N}$$

O excepto un número finito de ellos. A la notación $\Omega(g(n))$ se le llama notación asintótica "omega". Por otra parte, si $f(n) = \Omega(g(n))$ y $f(n) = O(g(n))$ entonces se dice que $f(n)$ es de orden "theta" de $g(n)$ y se representa $f(n) = \Theta(g(n))$.



Ejemplo Demuestre que $1^j + 2^j + \dots + n^j = \Theta(n^{j+1})$ con $j \in \mathbb{N}$. Mediante el comando Animate en Mathematica, visualice esta identidad.

$$f(n) = 1^j + 2^j + \dots + n^j = O(n^{j+1})$$

$$f(n) = \Omega(n^{j+1})$$

$$1^j + 2^j + \dots + n^j \geq 1^j + \dots + 1^j = n$$

Eliminar la mitad de los términos:

$$1^j + 2^j + \dots + n^j \geq \left(\left\lceil \frac{n}{2} \right\rceil\right)^j + \dots + (n-1)^j + n^j$$

$$\left(\left\lceil \frac{n}{2} \right\rceil\right)^j; \text{ luego: } 1^j + 2^j + \dots + n^j \geq \left(\left\lceil \frac{n}{2} \right\rceil\right)^j + \dots + \left(\left\lceil \frac{n}{2} \right\rceil\right)^j$$

$$\Rightarrow 1^j + 2^j + \dots + n^j \geq \frac{n+1}{2} \left(\left\lceil \frac{n}{2} \right\rceil\right)^j$$

Donc:

$$\frac{n+1}{2} \left(\left\lceil \frac{n}{2} \right\rceil \right)^j \geq \frac{n}{2} \left(\left\lceil \frac{n}{2} \right\rceil \right)^j \geq \frac{n}{2} \cdot \left(\frac{n}{2} \right)^j = \frac{n^{j+1}}{2^{j+1}}$$

$$|1^j + 2^j + \dots + n^j| \geq \frac{1}{2^{j+1}} |n^{j+1}|$$

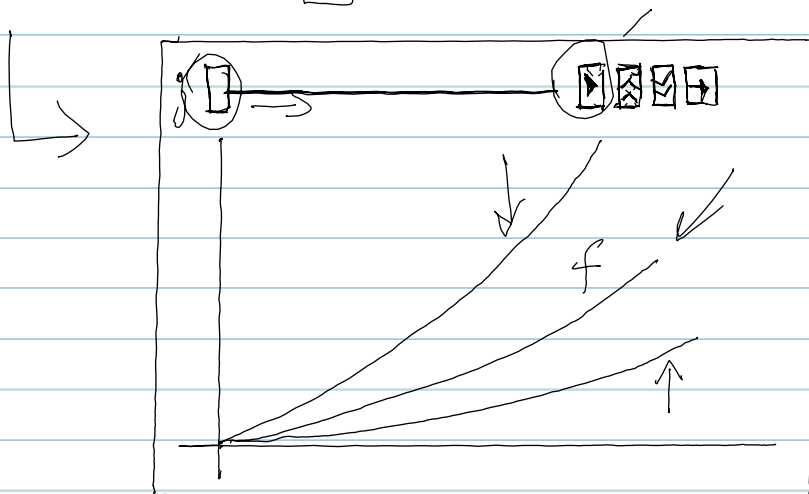
$$f(n) = \Omega(n^{j+1})$$

$$\Rightarrow f(n) = \Theta(n^{j+1})$$

En Mathematica:

Animate [show [Plot [s[n_] = $\sum_{i=1}^n i^j$; f[n_] := s[n]; g[n_] := $\frac{n^{j+1}}{2^{j+1}}$; h[n_] := $\frac{n^{j+1}}{2^{j+1}}$; d [f[n], g[n], h[n]]], {n, 0, 4}], PlotStyle -> {Thick}],

Graphics [d [Text [style ["f", 10], {2.6, f[2.7] + 1}], {j, 1, 10}], AnimationRunning -> {False}]



Ejemplo Prueba: $\ln(n!) = \Theta(n \ln n)$

$$\ln(n!) = \Theta(n \ln n), \quad \ln(n!) = \Omega(n \ln n)$$

$$\ln(n!) = \sum_{j=1}^n \ln j = \ln 1 + \ln 2 + \dots + \ln n$$

$$a = \ln\left(\left\lceil \frac{n}{2} \right\rceil\right)$$

$$\Rightarrow \ln(n!) \geq \ln\left(\left\lceil \frac{n}{2} \right\rceil\right) + \dots + \ln(n-1) + \ln(n) \checkmark$$

$$\Rightarrow \ln(n!) \geq \underbrace{\ln\left(\left\lceil \frac{n}{2} \right\rceil\right) + \ln\left(\left\lceil \frac{n}{2} \right\rceil\right) + \dots + \ln\left(\left\lceil \frac{n}{2} \right\rceil\right)}_{\frac{n+1}{2} \cdot a}$$

Luego:

$$\ln(n!) \geq \left(\frac{n+1}{2}\right) a \geq \frac{n}{2} a \geq \frac{n}{2} \ln\left(\frac{n}{2}\right) = \frac{n}{2} (\ln n - \ln 2)$$

Por convergencia:

$$\begin{aligned} & \frac{n}{2} (\ln n - \ln 2) \checkmark \\ &= \frac{n}{2} \left(\frac{\ln n}{2} + \frac{\ln n}{2} - \ln 2 \right) \\ &\geq \frac{n}{2} \frac{\ln n}{2} \checkmark \quad n \geq 4 \end{aligned}$$

$$\Rightarrow |\ln(n!)| \geq \frac{n \ln n}{4} \quad \forall n, n \in \mathbb{N}$$

$$\ln(n!) = \Omega(n \ln n) \xrightarrow{c = 1/4} \ln(n!) = \Theta(n \ln n)$$

Ejemplo Encuentre una notación Θ para el siguiente algoritmo:

$\{$ CalculaOtraSuma[n-] :=
 Module [dp=0, For [i=1, i ≤ n, For [j=1, j ≤ i, p=p+2;
 j++]; i++]; Print [p]]

For
 i=1 \Rightarrow p=p+2 \rightarrow 1
 i=2 \Rightarrow p=p+2 \rightarrow 2
 ...

$$p = p+2 \rightarrow 1+2+\dots+n$$

$$1+2+\dots+n = \Theta(n^{j+1}) \quad \nearrow j=1.$$

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

Ejemplo Halle Θ en el programa:

$\{$ CalculaMasProductos[n-] :=
 Module [dp=1, i=n, While [i > 1, p=p*3; i=i/2];
 Print [p]]

$$i \rightarrow \frac{n}{2^k}, k \in \mathbb{N}$$

$$\text{While} \rightarrow \frac{n}{2^k} < 1 \Rightarrow n < 2^k$$

$$\Rightarrow k > \log_2 n$$

$$c \in \mathbb{R}^+; \lceil \log_2 n \rceil - c = \log_2 n$$

$$\log_2 n + c = O(\ln n)$$

$$\frac{\log_2 n + c}{\ln n} \rightarrow \text{En Mathematica:}$$

$$\lim_{n \rightarrow \infty} \left[\frac{\log_2 n + c}{\log n} \right]$$

$$\rightarrow \frac{1}{\log 2} \in \mathbb{R}$$

Debermos probar ahora que:

$$\log_2 n + c = \Omega(\ln n)$$

$$\log_2 n + c \geq \log_2 n = \frac{\ln n}{\ln 2}$$

$$|\log_2 n + c| \geq \left(\frac{1}{\ln 2} \right) |\ln n|$$



$$\therefore \log_2 n + c = \Omega^c(\ln n)$$

$$\Theta(\ln n).$$