Programación Científica y HPCI

Máster Universitario en Ingeniería Matemática y Computación

Docente: Jesús Cigales Canga

Tema 3

unir LA UNIVERSIDAD EN INTERNET

Universidad Internacional de La Rioja

¿Cómo estudiar este tema?

1. Introducción y objetivos
2. Complejidad en tiempo vs espacio
3. Notaciones asintóticas

Material Complementario

Complejidad algorítmica ejemplos de Python

Análisis de algoritmos y complejidad

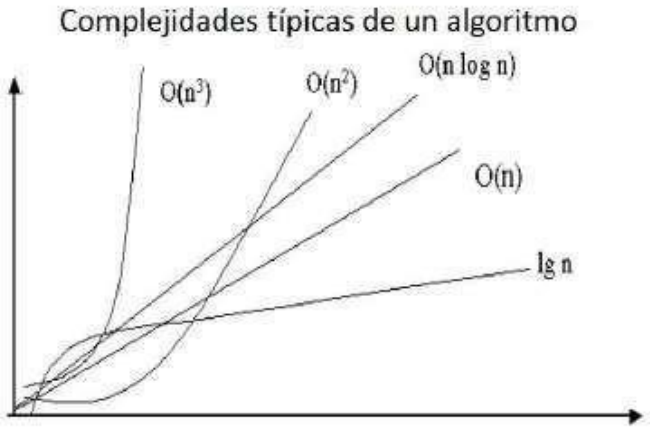
Introduction to Big O Notation and Time Complexity

Practicar conceptos avanzados de git en el navegador

Referencia rápida de git en español

Lecciones magistrales

La complejidad



- Comparar el rendimiento de diferentes algoritmos y elegir el mejor para resolver un problema concreto.
  - Complejidad temporal → cantidad de tiempo que tarda un algoritmo en ejecutarse en función de la longitud de la entrada
  - Complejidad espacial → cuantifica la cantidad de espacio o memoria que necesita un algoritmo para ejecutarse en función de la longitud de la entrada.

- Al analizar el algoritmo se considera solo un factor, el tiempo de ejecución

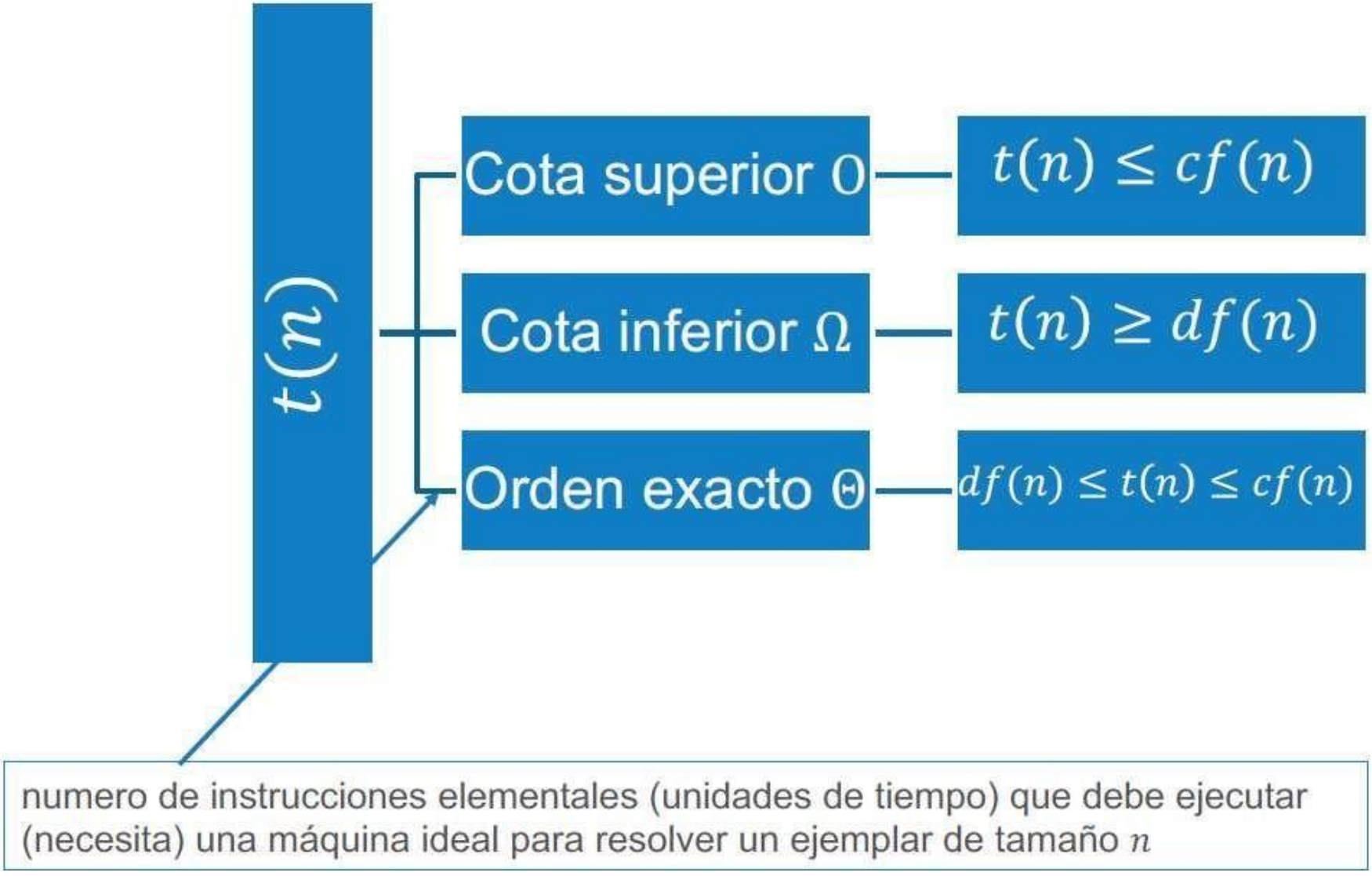
Fuente de la imagen: https://rootear.com/desarrollo/complejidades-algoritmos

Función de complejidad $t(n)$

numero de instrucciones elementales (unidades de tiempo) que debe ejecutar (necesita) una máquina ideal para resolver un ejemplar de tamaño n

Lecciones magistrales

Función de complejidad $t(n)$



numero de instrucciones elementales (unidades de tiempo) que debe ejecutar (necesita) una máquina ideal para resolver un ejemplar de tamaño n

Función de complejidad $t(n)$ y su orden

$¿n^2 + 2n \in O(n^3)?$

$f(n) = n^2 + 2n \;\; g(n) = n^3$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \leq cte \Rightarrow f(n) \in O(n^3) \;\; pero \; g(n) \notin O(f(n))$$

$¿n^2 + 2n \in O(n^2)?$

$f(n) = n^2 + 2n \;\; g(n) = n^2$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1 \leq cte \Rightarrow f(n) \in O(n^2) \;\; y \; g(n) \in O(f(n))$$

$¿n^2 + 2n \in O(n)?$

$f(n) = n^2 + 2n \;\; g(n) = n$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \not\leq cte \Rightarrow f(n) \notin O(n) \; y \;\; g(n) \in O(f(n))$$

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

Θ

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq cte$$

O

$¿n^2 + 2n \in \theta(n^3)?$

$f(n) = n^2 + 2n \;\; g(n) = n^3$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) \notin \theta(n^3)$$

$¿n^2 + 2n \in \theta(n^2)?$

$f(n) = n^2 + 2n \;\; g(n) = n^2$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1 > 0 \Rightarrow f(n) \in \theta(n^2)$$

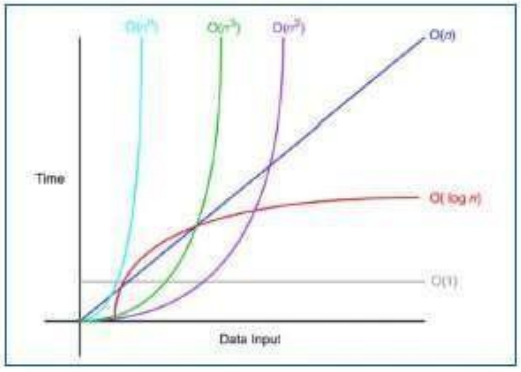
$¿n^2 + 2n \in \theta(n)?$

$t(n) = n^2 + 2n \;\; f(n) = n$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \not\leq cte \Rightarrow f(n) \notin \theta(n)$$

Ordenes de complejidad

Orden	Nombre
$O(1)$	constante
$O(\log n)$	logarítmica
$O(n)$	lineal
$O(nlog\ n)$	cuasi lineal
$O(n^2)$	cuadrática
$O(n^3)$	cúbica
$O(a^n)$	exponencial
$O(n!)$	factorial



Fuente: https://ichi.pro/es/entendiendo-la-notacion-big-o-2145609214246

Se descartan los coeficientes constantes y los términos menos significativos

Jerarquía

$$O(1) \subset O(\log n) \subset O(n) \subset O(nlog\ n) \subset O(n^2) \subset O(n^3) \subset O(a^n) \subset O(n!)$$

Orden de complejidad

`def cte:`
`  x=3`
`  x+=1`
	$O(1)$

`def func(n):`
`  suma = 0`
`  n=10`
`  for i in range(1,n+1):`
`  …`
	$O(n)$

`def func(a,x,menor,mayor):`
`  ...`
`  medio = (menor+((menor-mayor)//2))`
`  if ...: ans = func(a,x,menor,medio)`
`  elif ...: ans =func(a,x,medio+1,mayor)`
`  else: ...`
`  ...`
	$O(\log n)$
	$(2^k \geq n)$

`def func(n):`
`  suma = 0`
`  n=10`
`  for i in range(1,n+1):`
`  for j in range(1,n+1):`
`  ...`
	$O(n^2)$

`def func(n):`
`  if n==1 or n==2:`
`  return ...`
`  else func(n-1)+func(n-2)`
	$O(c^n)$

www.unir.net