Algoritmos y Estructuras de Datos II
Clase 4

Carreras:

Licenciatura en Informática

Ingeniería en Informática

2024

Notación Omega

- La notación Ω se usa para describir la *cota inferior* de un problema.

- Definición: la ***cota inferior*** de un problema es la complejidad temporal mínima requerida por cualquier algoritmo que pueda aplicarse para resolverlo.

- La mejor cota inferior es la más alta.

- Un ***algoritmo es óptimo*** si su complejidad temporal es igual a una cota inferior de este problema. Ya no es posible mejorar la cota inferior ni el algoritmo.

Notación Omega

Definición: Sea R* reales no negativos.

Considérese dos funciones t y f:N→R* de los números naturales en los números reales no negativos.

Se define Ω de f(n) como el conjunto de todas las aplicaciones:

$$\Omega(f(n)) = \{ t : N \rightarrow R^* / (\exists\, d \in R^+)\ (\exists\, n_0 \in N):\ t(n) \geq d \cdot f(n) \geq 0 \quad \forall\, n \geq n_0 \}$$

Por Ejemplo:

$$\sqrt{n}\ \epsilon \in \Omega(\log_2 n)$$

Para: $n_0 = 16$; $c = 1$

Notación Omega

Se dice que t(n) ***está en Omega de f(n),*** lo que se denota como:

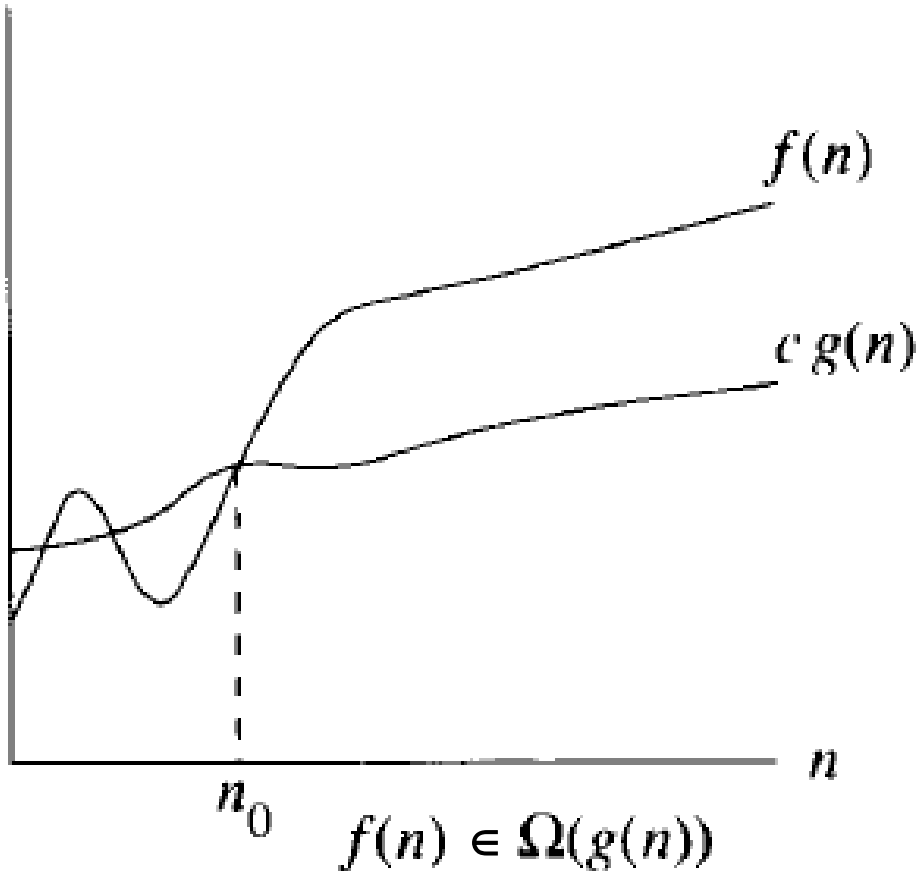
$$\mathbf{t(n) \in \Omega(f(n))}$$

si t(n) está acotada inferiormente por un múltiplo real positivo de f(n) para todo n suficientemente grande.

Matemáticamente, esto significa que existe una constante real positiva d y un umbral entero n_0 tal que $t(n) \geq d.f(n)$ siempre que $n \geq n_0$

Gráfica de la Notación Omega

En un gráfico:



Propiedades de la Notación Omega

- $f(n) \in \Omega(f(n))$
- $f(n) \in \Omega(g(n)) \Rightarrow \Omega(f(n)) \subset \Omega(g(n))$
- $f(n) \in \Omega(g(n))$ y $g(n) \in \Omega(f(n)) \Leftrightarrow \Omega(f(n)) = \Omega(g(n))$
- $f(n) \in \Omega(g(n))$ y $f(n) \in \Omega(h(n)) \Rightarrow f(n) \in \Omega(\max(g(n),h(n))$

- **Transitiva:**
  $f(n) \in \Omega(g(n))$ y $g(n) \in \Omega(h(n)) \Rightarrow f(n) \in \Omega(h(n))$

- **Regla de la suma:**
  $f1(n) \in \Omega(g1(n))$ y $f2(n) \in \Omega(g2(n)) \Rightarrow f1(n)+f2(n) \in \Omega(\max(g1(n),g2(n)))$

- **Regla del producto:**
  $f1(n) \in \Omega(g1(n))$ y $f2(n) \in \Omega(g2(n)) \Rightarrow f1(n).f2(n) \in \Omega(g1(n).g2(n))$

Ejemplo de la Notación Omega

$(n+3)^2 \in \Omega(n^2)$?

$(n+3)^2 \geq d.n^2$ para $n \geq n_0$

$n^2 + 6.n + 9 \geq d.n^2$ para $n \geq n_0$

Dividir por $n^2 > 0$ ambos miembros de la desigualdad:

$9 + 6/n + 1/n^2 \geq d$ para $n \geq n_0$

De modo que: si $n_0 = 1$, $d = 1$

$(n+3)^2 \geq 1.n^2$ para $n \geq 1$

➔ $(n+1)^2 \in \Omega(n^2)$

Notación Omega
Ejercicios:

Demostrar que:

- $3n^2+100\ n+4 \in \Omega(1)$
- $3n^2+100\ n+4 \in \Omega(n)$
- $3n^2+100\ n+4 \in \Omega(n^2)$
- $3n^2+100\ n+4 \notin \Omega(n^3)$

Notación Omega

Dado un problema:

- Si la cota inferior conocida actual del problema ***es más baja*** que la complejidad temporal del mejor algoritmo disponible, entonces es posible mejorar la cota inferior (hacia arriba) o el algoritmo puede mejorar su complejidad temporal (hacia abajo) o ambas cosas.
- Si la cota inferior conocida actual del problema ***es igual*** a la complejidad temporal del mejor algoritmo disponible, entonces no es posible mejorar ni el algoritmo ni la cota inferior. En este caso el *algoritmo es óptimo* y la cota inferior es la máxima posible.

Cota en problema de clasificación

- Una cota famosa es la cota del problema de clasificación, es igual a:

  $\Omega (n \log n)$

- Esta cota inferior *no es única*, se puede decir que una cota inferior es Ω (n) porque cada dato debe examinarse antes de terminar el ordenamiento, también se podría decir que es Ω (1) porque todo algoritmo realiza al menos un paso.

- Entonces se tienen tres cotas inferiores para ordenar:

  $\Omega (n \log n)$, $\Omega (n)$ y $\Omega (1)$,

  todas *son correctas* aunque la única *importante* es Ω (n log n), pues las otras *son cotas inferiores triviales.*

- Siempre hay que *dar la cota inferior más alta posible*.

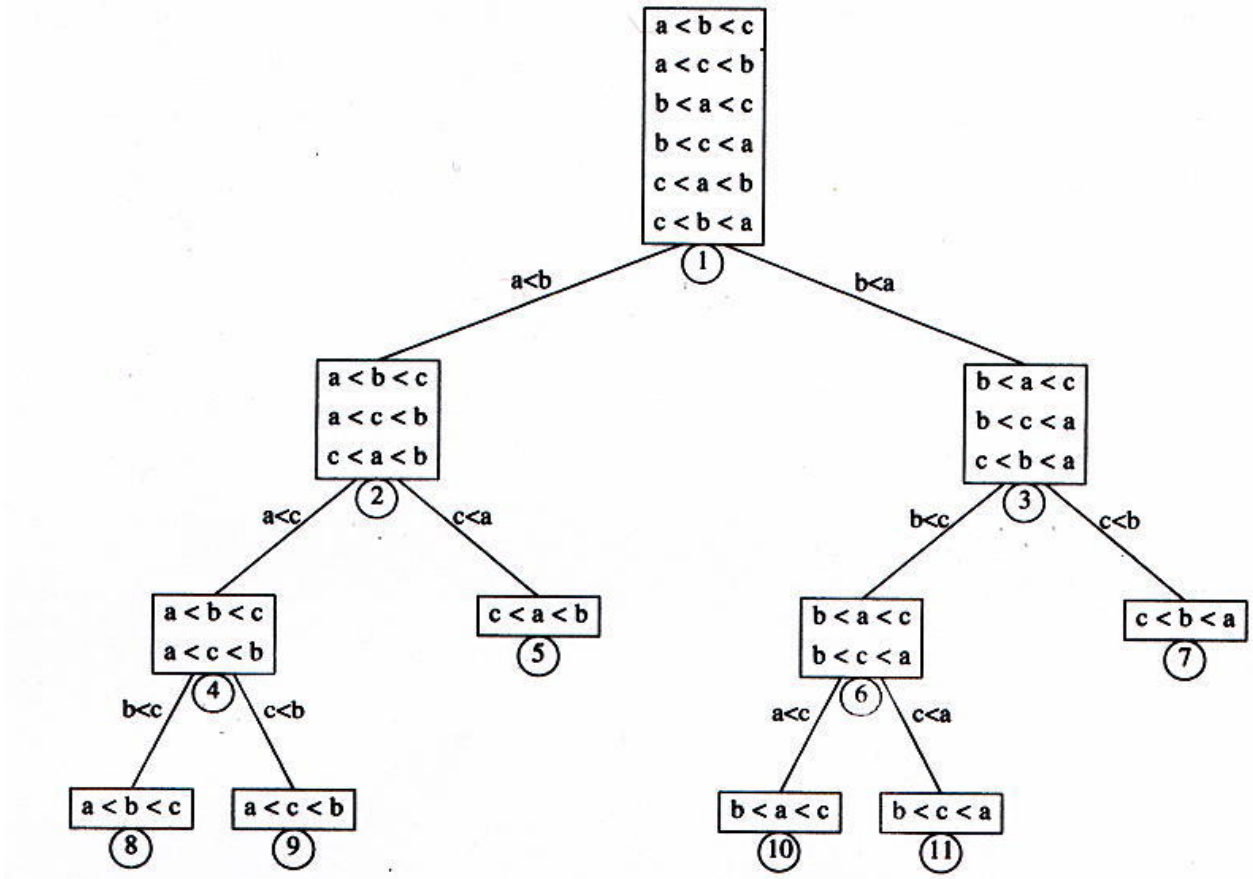
- Es importante saber que cada cota inferior más alta se encuentra mediante un análisis teórico, no por pura suposición.

Problema de clasificación

- Cualquier algoritmo de ordenamiento cuyas operaciones básicas sean de comparación e intercambio puede describirse mediante un árbol de decisión binario.
- La acción de un algoritmo basado en comparaciones y movimientos sobre un conjunto de entrada particular, corresponde a un camino que va desde la raíz del árbol hasta una hoja. En consecuencia cada nodo hoja corresponde a una permutación particular.
- *El camino más largo que va desde la raíz del árbol a un nodo hoja que se denomina altura del árbol, representa la complejidad temporal del peor caso de ese algoritmo.*

Arbol: problema de clasificación

Ejemplo*: Sea el caso de n=3 datos: a, b, c tales que: a ≠ b ≠ c*



Problema de clasificación

- Para encontrar la cota inferior del problema de ordenación es necesario encontrar la *altura menor* de algún árbol de entre todos los algoritmos posibles que ordenan con un árbol binario de decisión.

Algunos datos:

- Para todo algoritmo de ordenación, su árbol de decisión binario tendrá n! nodos hojas correspondientes a las n! permutaciones posibles.

- La altura del árbol de decisión será mínima si el mismo está balanceado

- Cuando el árbol está balanceado, la altura del árbol $h = \lceil \log_2 k \rceil$, donde k es el número de hojas

Problema de clasificación

- Entonces una cota inferior del problema de ordenación de n datos es $\lceil \log_2 n! \rceil$
- Por lo tanto el número de comparaciones necesarias para ordenar en el peor caso es por lo menos $\lceil \log_2 n! \rceil$
- Se puede demostrar que existen dos constantes:
  $n_0=4$ y $d=0.28$ para las que:

$$\log_2 n! \geq d.n.\ \log_2 n \quad \text{para } n \geq n_0$$

- *De modo que la cota inferior para el peor caso de ordenación es de* $\boldsymbol{\Omega(n.\log_2 n)}$

Notación Theta

- Cuando se analiza el comportamiento de un algoritmo, se necesitaría que su tiempo de ejecución esté acotado tanto por encima como por debajo mediante múltiplos de una misma función.

- Para ello se usa la notación Θ que se llama *orden exacto.*

- Se dice que t(n) está en Theta de f(n), o también que t(n) está en el orden exacto de f(n) y lo se denota con:

  $$\mathbf{t(n) \in \Theta(f(n))}$$

  si t(n) pertenece tanto a O(f(n)) como a Ω(f(n)).

Notación Theta

Definición formal:

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

$$\Theta(f(n)) = \{ t : N \rightarrow R^* / (\exists\, c,d \in R^+)\ (\exists\, n_0 \in N): \quad d \cdot f(n) \leq t(n) \leq c \cdot f(n) \quad \forall\, n \geq n_0 \}$$

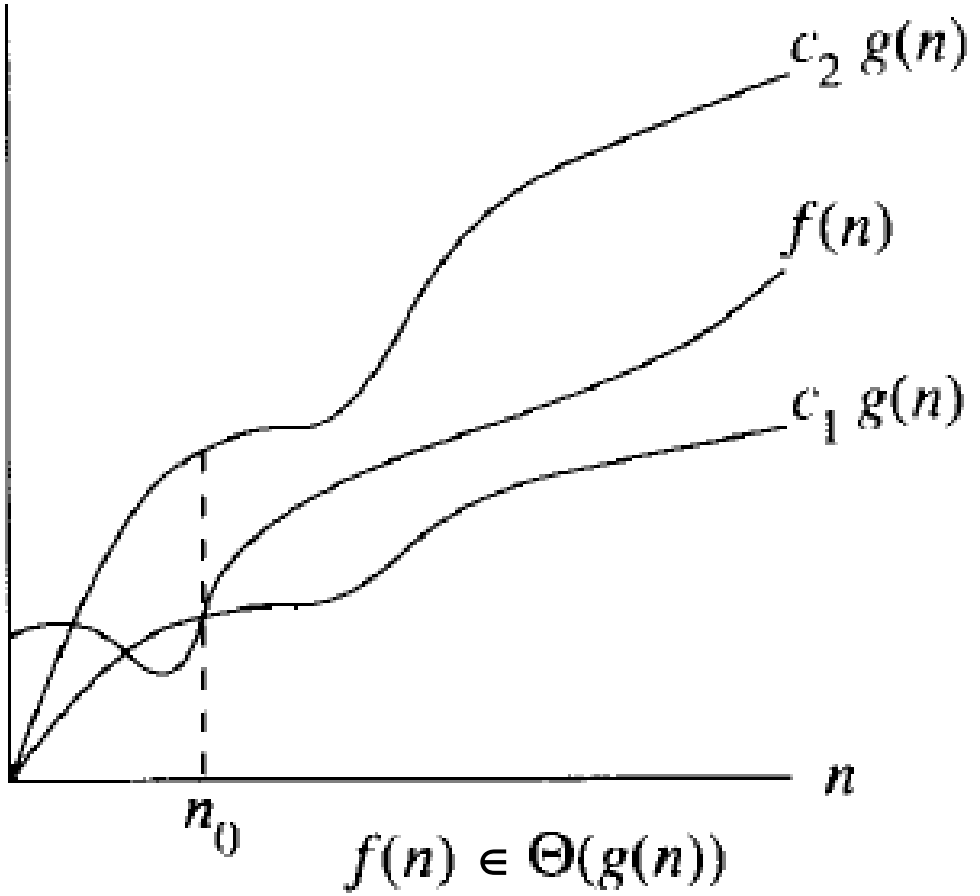
Por Ejemplo:

$$\tfrac{1}{2} n^2 - 2n \in \Theta(n^2)$$

Para: $n_0 = 8$; $d = 0.25$; $c = 1$

Gráfica de la Notación Theta

En un gráfico:



Propiedades de la Notación Theta

- $f(n) \in \Theta(f(n))$
- $f(n) \in \Theta(g(n)) \Rightarrow \Theta(f(n)) = \Theta(g(n))$
- $f(n) \in \Theta(g(n))$ y $g(n) \in \Theta(f(n)) \Leftrightarrow \Theta(f(n)) = \Theta(g(n))$

- **Transitiva:**
  $f(n) \in \Theta(g(n))$ y $g(n) \in \Theta(h(n)) \Rightarrow f(n) \in \Theta(h(n))$

- **Regla de la suma:**
  $f1(n) \in \Theta(g1(n))$ y $f2(n) \in \Theta(g2(n)) \Rightarrow f1(n)+f2(n) \in \Theta(\max(g1(n),g2(n)))$

- **Regla del producto:**
  $f1(n) \in \Theta(g1(n))$ y $f2(n) \in \Theta(g2(n)) \Rightarrow f1(n).f2(n) \in \Theta(g1(n).g2(n))$

Ejemplo de la Notación Theta

$(n+1)^2 \in \Theta(n^2)$?

$d.n^2 \leq (n+1)^2 \leq c.n^2$ para $n \geq n_0$

$d.n^2 \leq n^2 + 2.n + 1 \leq c.n^2$ para $n \geq n_0$

Dividir por $n^2 > 0$ la desigualdad:

$d \leq 1 + 2/n + 1/n^2 \leq c$ para $n \geq n_0$

De modo que: si $n_0 = 1$, $d = 1$, $c = 4$

$1.n^2 \leq (n+1)^2 \leq 4.n^2$ para $n \geq 1$

➔ $(n+1)^2 \in \Theta(n^2)$

Ejemplo de la Notación Theta

$n(n-1)/2 \in \Theta(n^2)$?

$d.n^2 \leq n(n-1)/2 \leq c.n^2$ para $n \geq n_0$

$d.n^2 \leq n^2/2 - n/2 \leq c.n^2$ para $n \geq n_0$

Dividir por $n^2 > 0$ la desigualdad:

$d \leq 1/2 - 1/(2n) \leq c$ para $n \geq n_0$

De modo que: si $n_0=2$, $d=1/4$, $c=1/2$

➔ $n(n-1)/2 \in \Theta(n^2)$

Notación Theta
Ejercicios:

Demostrar que:

- $3n^2+100\ n+4 \notin \Theta(1)$
- $3n^2+100\ n+4 \notin \Theta(n)$
- $3n^2+100\ n+4 \in \Theta(n^2)$
- $3n^2+100\ n+4 \notin \Theta(n^3)$

Resumen de Notaciones

- $f(n) \in O(g(n))$ significa que c.g(n) es una ***cota superior*** para f(n), de modo que $f(n) \leq c.g(n)$ para n suficientemente grande.
- $f(n) \in \Omega(g(n))$ significa que d.g(n) es una ***cota inferior*** para f(n), de modo que $f(n) \geq d.g(n)$ para n suficientemente grande.
- $f(n) \in \Theta(g(n))$ significa que c.g(n) es una ***cota superior*** para f(n) y que d.g(n) es una ***cota inferior*** para f(n), de modo que $d.g(n) \leq f(n) \leq c.g(n)$ para n suficientemente grande. Lo que implica que g(n) es una cota ajustada de f(n).

Notación o minúscula

Definición (Landau-1909):

Sea R* reales no negativos

Sea f: N → R* una función arbitraria

Se define o minúscula de f(n) como el conjunto de todas las aplicaciones:

$$\mathbf{o(f(n)) = \{\, t : N \rightarrow R^* \;/\; (\forall\, c \in R^+)\ (\exists\, n_0 \in N),\ 0 \leq t(n) < c \,.\, f(n)\ ,\quad \forall\, n \geq n_0 \,\}}$$

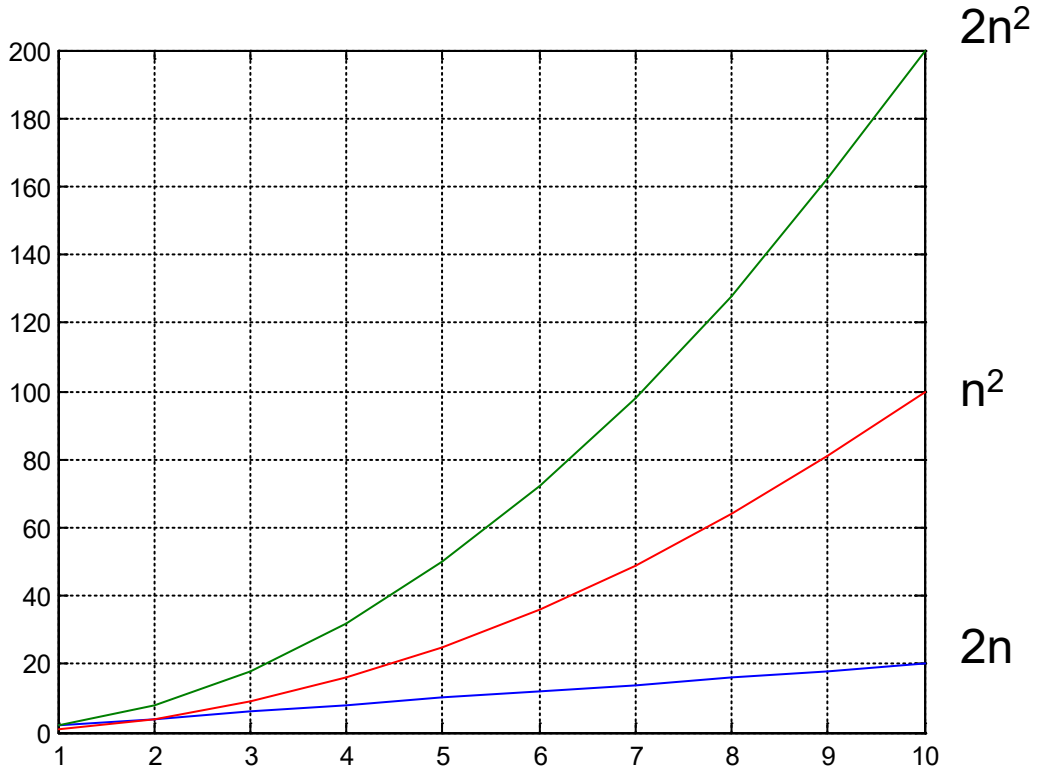
Por Ejemplo:

$2n^2 \in o(n^3)$ para: $n_0 = 2/c$

Notación o minúscula

Por ejemplo: $2n \in \mathbf{o}(n^2)$, pero $2n^2 \notin \mathbf{o}(n^2)$

n	2n	n^2	$2n^2$
1	2	1	2
2	4	4	8
3	6	9	18
4	8	16	32
5	10	25	50
6	12	36	72
7	14	49	98
8	16	64	128
9	18	81	162
10	20	100	200



Notación o minúscula

La notación o chica se usa para denotar cotas superior no ajustadas.

Existe una gran similitud en la definición O y o.

La principal diferencia es que:
$f(n) \in O(g(n))$ se cumple para alguna constante c real y positiva,

Mientras que:
$f(n) \in o(g(n))$ se cumple para toda constante c real y positiva.

Notación ω minúscula

Definición:

Sea R* reales no negativos

Sea f: N → R* una función arbitraria

Se define **ω** minúscula de f(n) como el conjunto de todas las aplicaciones:

$$\omega(f(n)) = \{ t : N \rightarrow R^* / (\forall\, c \in R^+)\ (\exists\, n_0 \in N),\ 0 \leq c \cdot f(n) < t(n),\ \forall\, n \geq n_0 \}$$

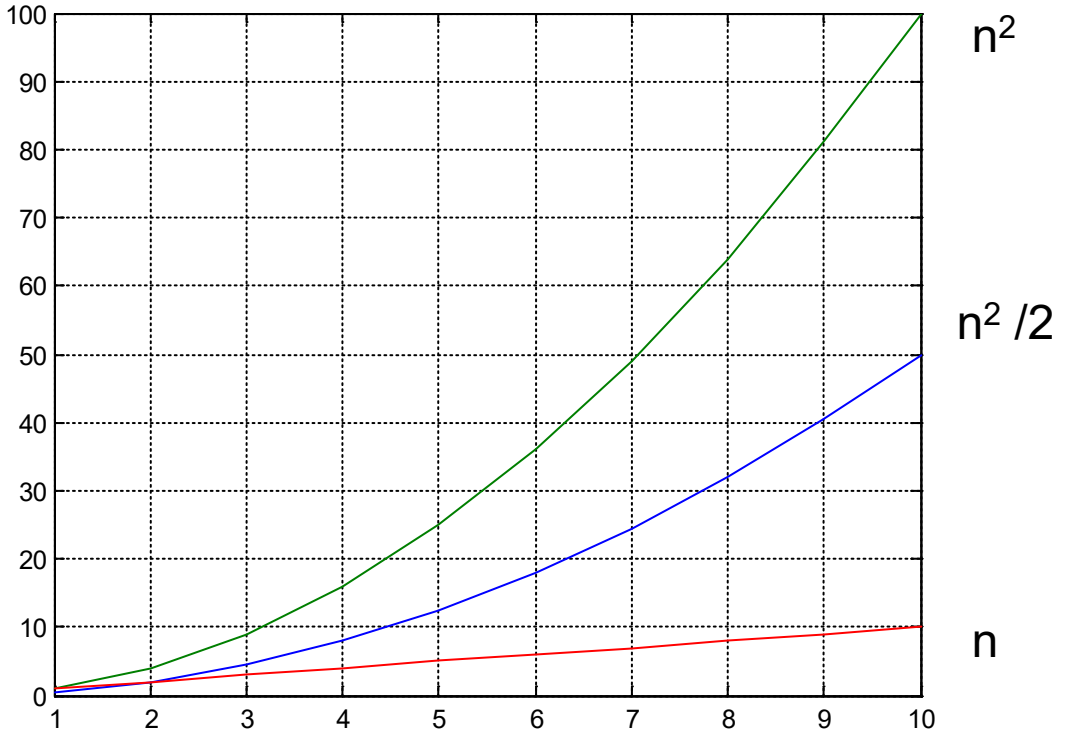
Por Ejemplo:

$$\sqrt{n} \in \omega(\log(n)) \quad \text{para: } n_0 = 1+1/c$$

Notación ω minúscula

Por ejemplo: $n^2/2 \in \boldsymbol{\omega}(n)$, pero $n^2/2 \notin \boldsymbol{\omega}(n^2)$

n	$n^2/2$	n^2
1.0	0.5	1.0
2.0	2.0	4.0
3.0	4.5	9.0
4.0	8.0	16.0
5.0	12.5	25.0
6.0	18.0	36.0
7.0	24.5	49.0
8.0	32.0	64.0
9.0	40.5	81.0
10.0	50.0	100.0



Notación ω minúscula

La notación ω chica se usa para denotar cotas asintóticas inferior no ajustadas.

Existe una gran similitud en la definición de Ω y de ω.

La principal diferencia es que:

$f(n) \in \Omega(g(n))$ se cumple para alguna constante c real y positiva,

Mientras que:

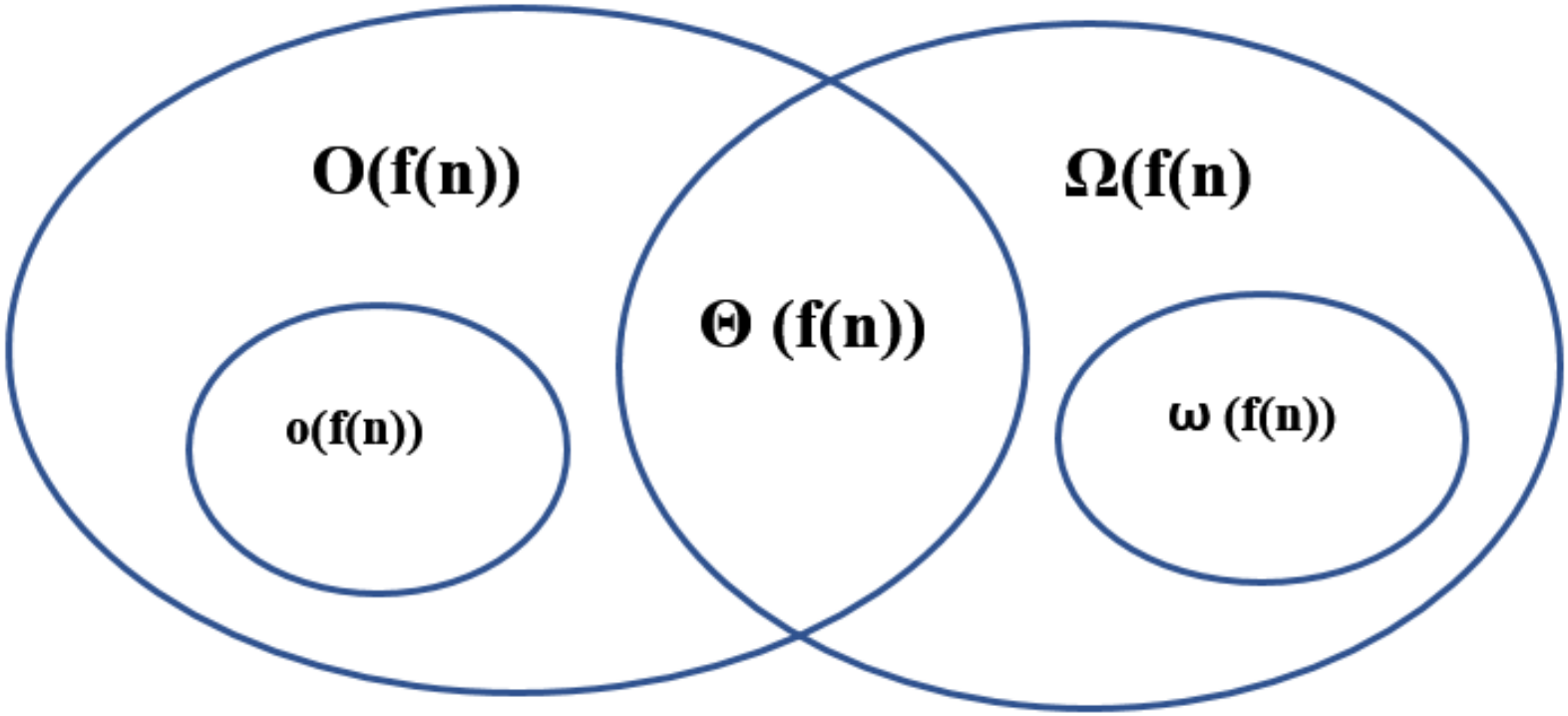
$f(n) \in \omega(g(n))$ se cumple para toda constante c real y positiva

Observaciones sobre las cotas asintóticas

De acuerdo a las propiedades de las notaciones asintóticas se puede hacer una analogía entre la comparación asintótica de dos funciones y la comparación de dos números reales:

$$f(n) \in O(g(n)) \approx a \leq b$$
$$f(n) \in \Omega(g(n)) \approx a \geq b$$
$$f(n) \in \Theta(g(n)) \approx a = b$$
$$f(n) \in o(g(n)) \approx a < b$$
$$f(n) \in \omega(g(n)) \approx a > b$$

Relación entre las cotas asintóticas



Observaciones sobre las cotas asintóticas

- La utilización de las cotas asintóticas para comparar funciones de tiempo de ejecución se basa en la hipótesis de que son suficientes para decidir el mejor algoritmo, prescindiendo de las constantes de proporcionalidad.
- Sin embargo, esta hipótesis puede no ser cierta cuando el tamaño de la entrada es pequeño.

Observaciones sobre las cotas asintóticas

- Para un algoritmo dado se pueden obtener tres funciones que miden su tiempo de ejecución, que corresponden a sus casos mejor, medio y peor, y que se denominan respectivamente:

$$T_m(n),\ T_{1/2}(n) \text{ y } T_p(n).$$

- Para cada una de ellas se puede dar tres cotas asintóticas de crecimiento, por lo que se obtiene un total de nueve cotas para el algoritmo.

Observaciones sobre las cotas asintóticas

Para simplificar, dado un algoritmo se dice que:

- Su orden de complejidad es O(f) si su tiempo de ejecución para el peor caso es de orden O de f, es decir:

$$T_p(n) \in O(f).$$

- De forma análoga se dice que su orden de complejidad para el mejor caso es Ω (g) si su tiempo de ejecución para el mejor caso es de orden Ω de g, es decir:

$$T_m(n) \in \Omega(g).$$

- Por último, un algoritmo es de orden exacto Θ(f) si su tiempo de ejecución en el caso medio es:

$$T_{1/2}(n) \in \Theta(f)$$

Relaciones dominantes

La *relación de dominancia* entre funciones es consecuencia de la teoría de límites.

Se dice que $g(n)$ *domina a* $f(n)$ si:

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

Esta relación se puede utilizar para determinar el orden relativo de crecimiento de dos funciones.

Regla del límite - Notación O

Sean $f, g: N \rightarrow R^*$ dos funciones arbitrarias de los números naturales en los números reales no negativos, y sea:

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

Si $L \in R^+$ entonces $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$

Si $L = 0$ entonces $f(n) \in O(g(n))$ y $g(n) \notin O(f(n))$

Si $L \rightarrow \infty$ entonces $f(n) \notin O(g(n))$ y $g(n) \in O(f(n))$

Regla del límite

Para 2 polinomios de distintos grados ***a*** y ***b***:

Entonces n^a domina n^b si es que el grado ***a > b***.

Se puede demostrar por el teorema del límite,

$$\lim_{n\to\infty} \frac{n^b}{n^a} = \lim_{n\to\infty} n^{b-a}$$

Además si $b<a$, b-a es un número negativo, de modo que:

$$\lim_{n\to\infty} \frac{n^b}{n^a} = \lim_{n\to\infty} n^{b-a} \to 0$$

Regla del límite

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

Si $L \in R^+$ entonces $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
Si $L = 0$ entonces $f(n) \in O(g(n))$ y $g(n) \notin O(f(n))$
Si $L \to \infty$ entonces $f(n) \notin O(g(n))$ y $g(n) \in O(f(n))$

Ejemplo 1: sean las funciones:

$$f(n) = \tfrac{1}{2} n(n-1) \qquad g(n) = n^2$$

Se quiere determinar si: $f \in O(g)$? y si: $g \in O(f)$?

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\frac{1}{2} n(n-1)}{n^2} = \frac{1}{2} \lim_{n\to\infty} \frac{n^2 - n}{n^2} = \frac{1}{2} \lim_{n\to\infty} (1 - \frac{1}{n}) = \frac{1}{2}$$

Por lo tanto: $\boxed{\tfrac{1}{2} n(n-1) \in O(n^2)}$ $\boxed{n^2 \in O(\tfrac{1}{2} n(n-1))}$

De modo que: $\boxed{\tfrac{1}{2} n(n-1) \in \Theta(n^2)}$

Regla del límite

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

Si $L \in R^+$ entonces $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
Si $L = 0$ entonces $f(n) \in O(g(n))$ y $g(n) \notin O(f(n))$
Si $L \to \infty$ entonces $f(n) \notin O(g(n))$ y $g(n) \in O(f(n))$

Ejemplo 2: sean las funciones:

$$f(n) = \log_2 n \qquad g(n) = \sqrt{n}$$

Se quiere determinar si: $f \in O(g)$? y si: $g \in O(f)$?

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} = \lim_{n\to\infty} \frac{(\log_2 n)'}{(\sqrt{n})'} = \lim_{n\to\infty} \frac{(\log_2 e)\frac{1}{n}}{\frac{1}{2\sqrt{n}}} = 2(\log_2 e) \lim_{n\to\infty} \frac{1}{\sqrt{n}} = 0$$

Por lo tanto: $\log_2 n \in O(\sqrt{n})$ $\sqrt{n} \notin O(\log_2 n)$

Además: $\log_2 n \in o(\sqrt{n})$

Regla del límite - Notación Ω

Para la notación Ω se puede ampliar la regla del límite, *mutatis mutandis*. *(del latín: Cambiando lo que se deba cambiar)*

Regla del límite: Sean $f, g: N \rightarrow R^*$ dos funciones arbitrarias de los números naturales en los números reales no negativos, y sea:

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

Se tiene que:

Si $L \in R^+$ entonces $f(n) \in \Omega(g(n))$ y $g(n) \in \Omega(f(n))$

Si $L = 0$ entonces $g(n) \in \Omega(f(n))$ y $f(n) \notin \Omega(g(n))$

Si $L \rightarrow \infty$ entonces $f(n) \in \Omega(g(n))$ y $g(n) \notin \Omega(f(n))$

Regla del límite - Notación Θ

Para la notación Θ se puede ampliar la regla del límite, *mutatis mutandis*. *(del latín: Cambiando lo que se deba cambiar)*

Regla del límite: Sean $f, g: N \rightarrow R^*$ dos funciones arbitrarias de los números naturales en los números reales no negativos, y sea:

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

Se tiene que:

Si $L \in R^+$ entonces $f(n) \in \Theta(g(n))$

Si $L = 0$ entonces $f(n) \in O(g(n))$ y $f(n) \notin \Theta(g(n))$

Si $L \rightarrow \infty$ entonces $f(n) \in \Omega(g(n))$ y $f(n) \notin \Theta(g(n))$

Regla del límite

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

Si $L \in R^+$ entonces $f(n) \in \Theta(g(n))$
Si $L = 0$ entonces $f(n) \in O(g(n))$ y $f(n) \notin \Theta(g(n))$
Si $L \to \infty$ entonces $f(n) \in \Omega(g(n))$ y $f(n) \notin \Theta(g(n))$

Ejemplo 3: sean las funciones:

$$f(n) = (n+1)^2 \qquad g(n) = n^2$$

Se quiere determinar si: $f \in \Theta(g)$?

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{(n+1)^2}{n^2} = \lim_{n\to\infty} \frac{n^2+2n+1}{n^2} = \lim_{n\to\infty} (1 + \frac{2}{n} + \frac{1}{n^2}) = 1$$

Por lo tanto:

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

Regla del límite - Resumen

Sean *f, g: N → R** dos funciones arbitrarias de los números naturales en los números reales no negativos, y sea:

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

Si $L \in R^+$ entonces $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$

Si $L = 0$ entonces $f(n) \in O(g(n))$ y $g(n) \notin O(f(n))$

Si $L \to \infty$ entonces $f(n) \notin O(g(n))$ y $g(n) \in O(f(n))$

Si $L \in R^+$ entonces $f(n) \in \Omega(g(n))$ y $g(n) \in \Omega(f(n))$

Si $L = 0$ entonces $g(n) \in \Omega(f(n))$ y $f(n) \notin \Omega(g(n))$

Si $L \to \infty$ entonces $f(n) \in \Omega(g(n))$ y $g(n) \notin \Omega(f(n))$

Si $L \in R^+$ entonces $f(n) \in \Theta(g(n))$

Si $L = 0$ entonces $f(n) \in O(g(n))$ y $f(n) \notin \Theta(g(n))$

Si $L \to \infty$ entonces $f(n) \in \Omega(g(n))$ y $f(n) \notin \Theta(g(n))$

Notación asintótica con varios parámetros

- Puede suceder cuando se analiza un algoritmo que su tiempo de ejecución dependa simultáneamente de más de un parámetro del objeto en cuestión.

- Esta situación es típica de ciertos algoritmos para problemas de grafos, por ejemplo, en los cuales el tiempo depende tanto del número de nodos como del número de aristas.

- En tales casos la noción de tamaño de entrada cambia su sentido. Por esta razón, se generaliza la notación asintótica para funciones de varias variables.

Notación asintótica con varios parámetros

Sea f : NxN→R* una función de pares de números naturales en los reales no negativos,

por ejemplo: f(a,b)=a log b.

Sea t : NxN→R* otra función de éstas.

Se dice que t(m,n) es de orden f(m,n) lo cual se denota:

$$t(m,n) \in O(f(m,n))$$

si t(m,n) está acotada superiormente por un múltiplo positivo de f(m,n) siempre que tanto m como n sean suficientemente grandes.

Notación asintótica con varios parámetros

Formalmente O(f(m,n)) se define como:

$$O(f(m,n)) = \{ t : NxN \rightarrow R^* / (\exists\, c \in R^+)\ (\exists\, n_0 , n_1 \in N)$$
$$t(m,n) \leq c \,.\, f(m,n),\ \forall\ m \geq n_0\ \ y\ \forall\ n \geq n_1\ \ \}$$

- La generalización de las otras notaciones se hace en forma similar.

Ejemplo: Algoritmo de Euclides

Entrada: m,n, dos numeros enteros positivos

Salida: m, entero positivo MCD de m y n
Auxiliar: r entero positivo o nulo

E1. Leer (m,n)
E2. Mientras n ≠ 0 Hacer
r ← resto (m/n)
m ← n
n ← r
E3. Escribir (m)
E4. Fin

Costo del algoritmo: $T(m,n) \in O(f(m,n))$

Ejemplo: subtira

Problema:

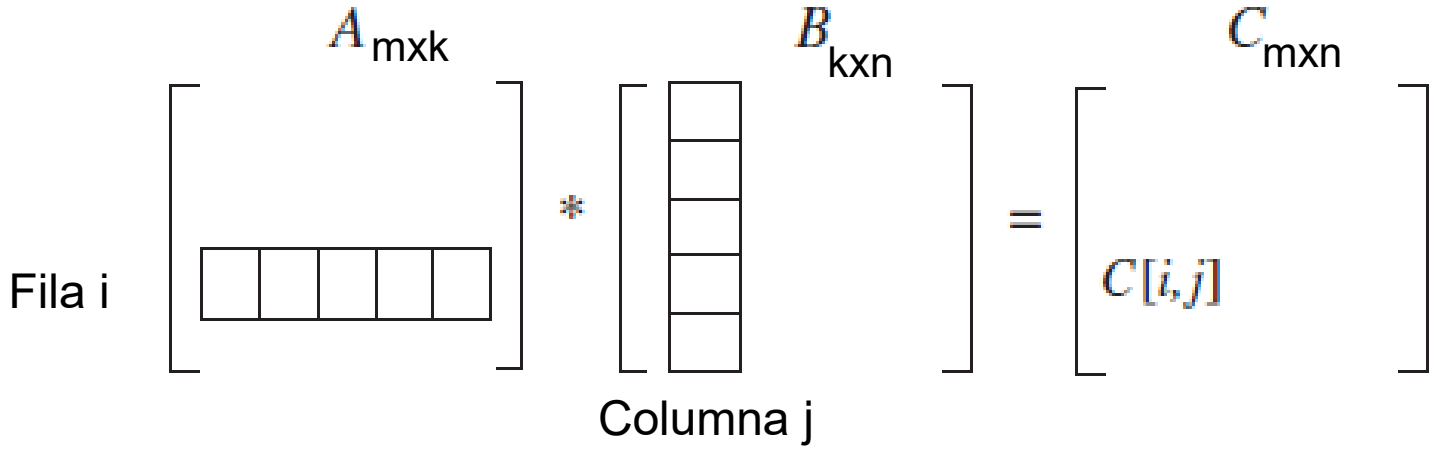
Dada una cadena de caracteres S de longitud n y una cadena de caracteres T de longitud m<n, se quiere determine la presencia de la cadena T en la cadena S, en caso de encontrarse donde se encuentra.

Costo del algoritmo T(m,n)

Ejemplo: multiplicación de matrices

Problema:

Dadas 2 matrices A (de dimensión mxk) y la matriz B (de dimensión kxn) se quiere multiplicarlas para obtener una matriz C (de dimesion mxn).



Costo del algoritmo T(m,k,n)

Algoritmos y Estructuras de Datos II

Trabajo Práctico 2