

Análisis de algoritmos

Parte 2: Crecimiento de funciones

Notación asintótica

- En este tema veremos algunos conjuntos de funciones con las que se suele representar la forma de crecimiento asintótico de un algoritmo
- Practicaremos su uso y calcularemos unas pocas de sus relaciones, pero sobre todo, veremos su significado gráfico

¿Qué es un conjunto?

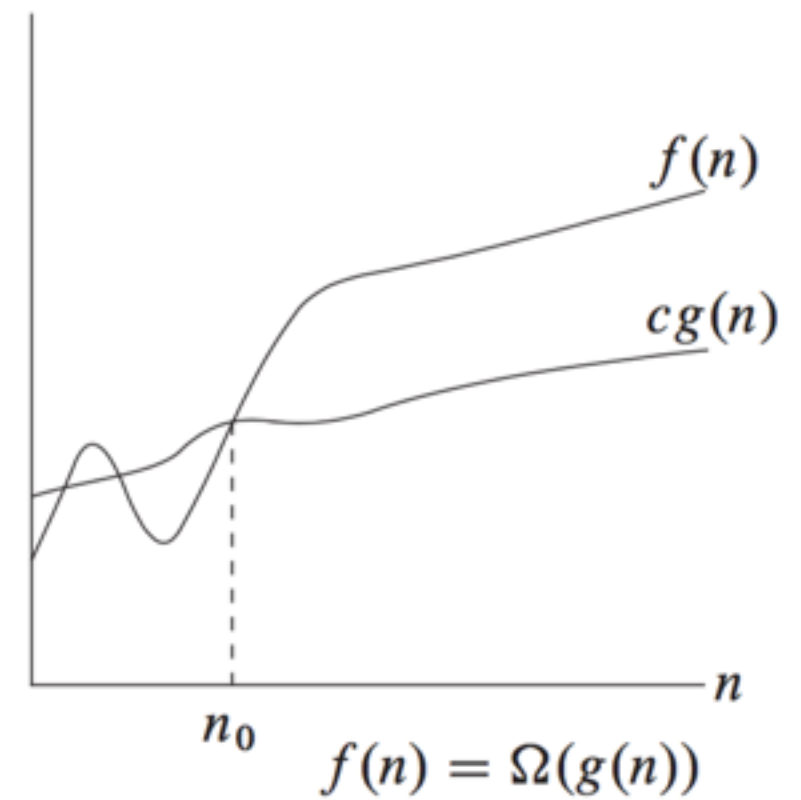
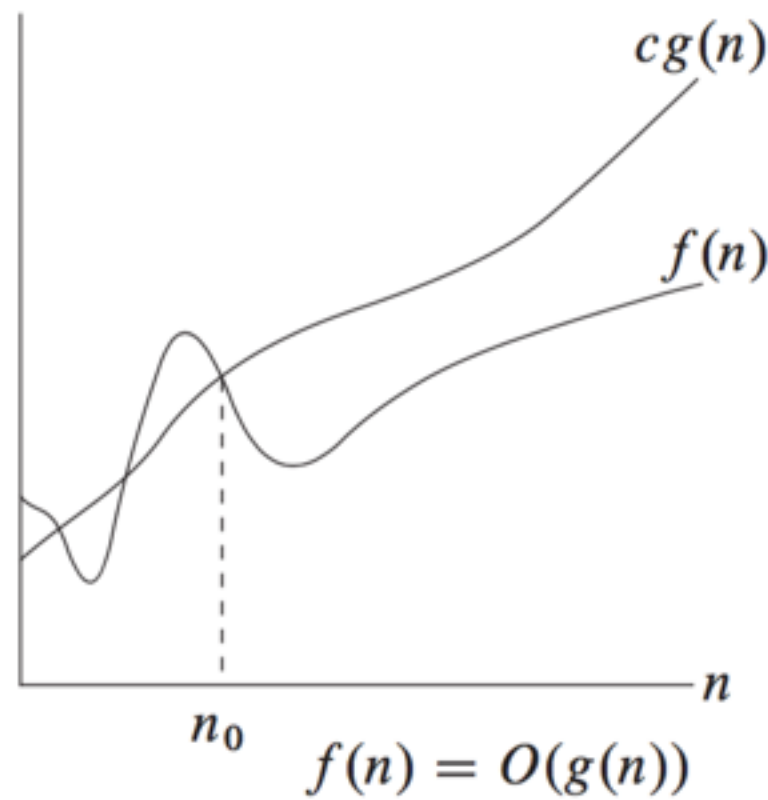
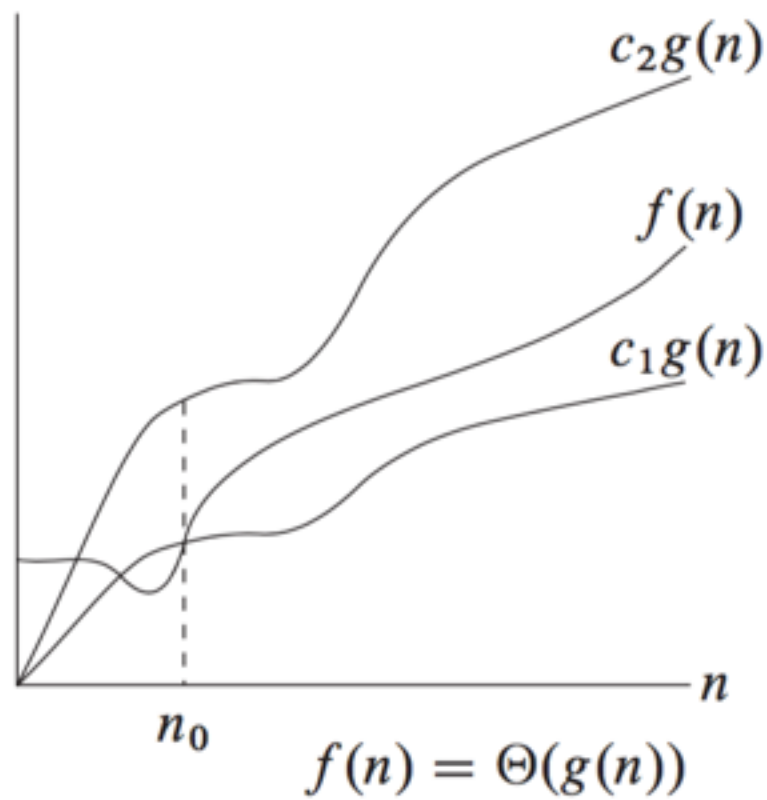
- Colección de elementos u objetos
- Con el nombre del conjunto en común
- No se admiten elementos repetidos

Algunos conjuntos de funciones

- Rectas: $\{Ax+B : A, B \in \mathbb{R}\}$
- Parábolas: $\{Ax^2+Bx+C : A, B, C \in \mathbb{R}\}$
- Exponenciales: $\{Ae^{Bx} : A, B \in \mathbb{R}\}$
- Logaritmos: $\{A \ln(Bx) : A \in \mathbb{R}, B \in \mathbb{R}^+\}$
- Áreas de triángulos con la misma base:
 $\{bx/2 : b \in \mathbb{R}^+\}$

- $O(g(n)) = \{f(n) : \text{existe } n_0 > 0 \text{ y } c > 0, \text{ tal que } 0 \leq f(n) \leq cg(n) \text{ para toda } n \geq n_0\}$
- $\Omega(g(n)) = \{f(n) : \text{existe } n_0 > 0 \text{ y } c > 0, \text{ tal que } 0 \leq cg(n) \leq f(n) \text{ para toda } n \geq n_0\}$
- $\Theta(g(n)) = \{f(n) : \text{existe } c_1 > 0, c_2 > 0, n_0 > 0, \text{ tal que } 0 \leq c_1g(n) \leq f(n) \leq c_2g(n) \text{ para toda } n \geq n_0\}$

gráficamente



- El análisis asintótico se refiere a ver el comportamiento de las funciones para valores del parámetro n grandes
- Este análisis marca la tendencia de la función, o sea, lo que se puede esperar de ella
- Es el factor **más importante** y en general, **el más útil para analizar un algoritmo**, sin embargo no es el único y a veces las constantes que se “desprecian” son demasiado significativas para no considerarlas.
- Sin embargo, es más o menos fácil identificar cuando estas constantes van a ser significativas.

Abusando de la notación

¿Cómo denotamos una función que está en estos conjuntos?

$f(n) \in O(g(n))$ se escribe $f(n) = O(g(n))$

$f(n) \in \Omega(g(n))$ se escribe $f(n) = \Omega(g(n))$

$f(n) \in \Theta(g(n))$ se escribe $f(n) = \Theta(g(n))$

El “orden” de una función

- Toda función se puede escribir $f(x) = O(f(x))$
- En efecto, si $f(x) \geq 0$, tenemos que para $c=1$ y $n_0=1$ tenemos:

$$0 \leq f(x) \leq cf(x)$$

- De manera similar $f(x) = \Omega(f(x))$ y $f(x) = \Theta(f(x))$

- Así, si $f(x) = \frac{1}{2}x + 3$
- entonces $\frac{1}{2}x + 3 = O(\frac{1}{2}x + 3)$
- pero la importancia de esta notación asintótica es su poder para simplificar la notación de funciones
- Por ejemplo, podemos también escribir $\frac{1}{2}x + 3 = O(x)$
- lo que demostraremos a continuación...

- Por definición, si queremos demostrar

$$\frac{1}{2}x + 3 = O(x)$$
- entonces tenemos que mostrar que existen

$$c > 0, n_0 > 0$$

- tal que

$$0 \leq f(x) \leq cg(x)$$

- donde

$$f(x) = \frac{1}{2}x + 3, g(x) = x$$

- Así tenemos

$$0 \leq \frac{1}{2}x + 3 \leq cx$$

- De la desigualdad derecha tenemos

$$\frac{1}{2}x + 3 \leq cx$$

- pasamos las x del lado mayor y lo demás del lado menor:

$$3 \leq cx - \frac{1}{2}x$$

- factorizamos x

$$3 \leq (c - \frac{1}{2})x$$

- lo del paréntesis no puede ser cero ni negativo (pues el producto no podría ser mayor que 3), para poder dividir por lo de la izquierda, así escogemos un valor $c > \frac{1}{2}$ por ejemplo $c = 1$, así:

$$x \geq \frac{3}{1 - \frac{1}{2}} = \frac{3}{\frac{1}{2}} = 6$$

- Con lo que el valor para n_0 se escoge

$$n_0 = 6$$

- con lo que demostramos que efectivamente

$$\frac{1}{2}x + 3 = O(x)$$

- De la misma manera podemos demostrar que

$$\frac{1}{2}x + 3 = \Omega(x)$$

- así, queremos demostrar que existen

$$c > 0, n_0 > 0$$

- tales que

$$0 \leq cx \leq \frac{1}{2}x + 3$$

- Con la desigualdad derecha observamos que

$$\begin{aligned} cx &\leq \frac{1}{2}x + 3, \\ -3 &\leq \frac{1}{2}x - cx, \\ -3 &\leq \left(\frac{1}{2} - c\right)x \end{aligned}$$

- Lo del paréntesis no puede ser cero ni negativo, o al pasar dividiendo, la x estará del lado menor, y queremos que esté del lado mayor. Además la c debe ser positiva, así $0 < c < 1/2$, tomemos $c = 1/4$.

- Con lo que tenemos que $x \geq \frac{-3}{\frac{1}{2} - \frac{1}{4}} = \frac{-3}{\frac{1}{4}} = -12$
- Si $n_0 = 1$, el primer positivo mayor que -12, todo lo anterior se cumplirá para $x \geq n_0$
- con lo que efectivamente $\frac{1}{2}x + 3 = \Omega(x)$
- Con estas dos pruebas y el siguiente teorema tendremos que $\frac{1}{2}x + 3 = \Theta(x)$
- En efecto, el **Teorema 1** relaciona O , Ω y Θ

$$f(n) = O(g(n)), f(n) = \Omega(g(n)) \iff f(n) = \Theta(g(n))$$

- Probemos primero que

$$f(n) = O(g(n)), f(n) = \Omega(g(n)) \implies f(n) = \Theta(g(n))$$
- Así sabemos que existen constantes

$$n_0 > 0, m_0 > 0, c_1 > 0, c_2 > 0$$
- tales que

$$0 \leq c_1 g(n) \leq f(n)$$
- y

$$0 \leq f(n) \leq c_2 g(n)$$
- tomemos

$$k_0 = \max\{n_0, m_0\}$$
- con lo que para toda $x > k_0$ se tiene

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$



- Probemos ahora que

$$f(n) = \Theta(g(n)) \implies f(n) = O(g(n)), f(n) = \Omega(g(n))$$
- así, sabemos que existen

$$n_0 > 0, c_1 > 0, c_2 > 0$$
- tales que para toda $x > n_0$ se tiene

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$
- para toda $n > n_0$, por lo que
- de la 1a y 2a desigualdad (desde la izquierda) tenemos que

$$f(n) = \Omega(g(n))$$
- para de la 1a y la 3a desigualdad tenemos que

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



- Antes de ver más ejemplos y resultados, veamos la interpretación de estos conjuntos de funciones, en adelante simplemente las llamaremos “notación asintótica” en lo que se ha visto hasta ahora de la complejidad espacio-temporal de algoritmos, en particular veremos los ejemplos de inserción y mezclas.

Complejidad temporal del algoritmo de inserción

- En el algoritmo de ordenamiento por inserción vemos que tiene el ciclo (que empieza en la línea 1) que controla la variable j y dentro tiene un ciclo (que empieza en la línea 4), que controla la variable i .
- La variable j toma valores desde 1 hasta $\text{len}(A) - 1$, por lo que produce $n - 1$ iteraciones.
- A su vez, el ciclo de la variable i produce entre 1 y $j - 1$ iteraciones.
- El **mejor caso** (la lista está ordenada) es cuando el ciclo de i produce 1 iteración por cada iteración de j , o sea **$n - 1$ iteraciones en total**.
- Mientras que en el peor caso es cuando se producen $j - 1$ iteraciones por cada iteración de j .
- Como la cantidad total de iteraciones se puede calcular con la suma de los enteros desde 1 hasta $n - 1$, esta cuenta nos da **$(n - 2)(n - 1) / 2$ iteraciones en total** en el **peor caso** (cuando la lista está ordenada al revés).

`inserción(A):`

1. Para j de 1 a $\text{len}(A)-1$
2. $\text{pivote} = A[j]$ # carta a acomodar
3. $i = j - 1$ # posición a la izquierda del pivote
4. Mientras $i \geq 0$ y $A[i] > \text{pivote}$
5. $A[i+1] = A[i]$
6. $i = i - 1$
7. $A[i + 1] = \text{pivote}$

- Así, podemos escribir las funciones para los casos mejor y peor de esta forma:

$$f_{\text{mejor}}(n) = n-1$$

$$f_{\text{peor}}(n) = n(n-1)/2 = (n^2 - n)/2$$
- Ej. Ver que $f_{\text{mejor}}(n) = \Theta(n)$ y $f_{\text{peor}}(n) = \Theta(n^2)$
- Para la mayoría de los algoritmos se prefiere hablar de una sólo complejidad temporal, en este caso la peor. Sin embargo en este caso, no podríamos decir que $\Theta(n^2)$ es lo que inserción se va a tardar para **todas las entradas posibles** (si el arreglo está ordenado, esto sería $\Theta(n)$)
- Sin embargo, con estas notaciones, podemos decir que inserción se tarda entre $\Omega(n)$ y $O(n^2)$. Es decir, existen entradas para las cuales se tarda n y otras para las cuales se tarda n^2
- Así, Ω nos dice la **cota inferior**, o el tiempo en el **mejor caso**, mientras que O nos dice la **cota superior**, o el tiempo en el **peor caso**. Como O nos dice el tiempo en el peor caso, por lo general ese es el que se suele decir que cubre este algoritmo.

Demostrar que $n^3 \neq \Theta(n^2)$:

- Supongamos que si existe $c_1 > 0$, $c_2 > 0$ y $n_0 > 0$ tales que $0 \leq c_1 n^2 \leq n^3 \leq c_2 n^2$, para toda $n > n_0$
- Dividiendo entre n^2 en la 2a desigualdad tenemos que $n \leq c_2$
- Pero como c_2 es constante, esto claramente no puede ser cierto para toda $n > n_0$



Para cotas menos justas existen...

- $o(g(n)) = \{f(n) : \text{para toda } c > 0, \text{ existe } n_0 > 0 \text{ tal que } 0 \leq f(n) < cg(n) \text{ para toda } n \geq n_0\}$
- $\omega(g(n)) = \{f(n) : \text{para toda } c > 0, \text{ existe } n_0 > 0 \text{ y, tal que } 0 \leq cg(n) < f(n) \text{ para toda } n \geq n_0\}$
- En $o(g(n))$, todo múltiplo de $g(n)$ será eventualmente más grande que $f(n)$, de manera que se separan cada vez mas conforme n crece. Dicho de otra forma, $f(n)$ se vuelve insignificante respecto a $g(n)$ para valores grandes de n .

- Por ejemplo, demostrar que $n=o(n^2)$:
- Hay que mostrar que para toda $c>0$, existe una $n_0>0$ tales que $0 \leq n < cn^2$, para toda $n>n_0$
- Sea $c>0$, y dividamos la 2a desigualdad entre cn , resultando $1/c < n$
- Por lo que si tomamos $n_0=1/c$, obtendremos lo que buscamos.



- Antes vimos que $n^2 = O(n^2)$, de hecho esta es una cota justa, y escribiamos $n^2 = \Theta(n^2)$, ahora veamos que $n^2 \neq o(n^2)$:
- Mostraremos que existe $c > 0$, tal que no existe $n_0 > 0$ tal que $0 \leq n^2 < cn^2$, para toda $n > n_0$
- Dividamos la segunda desigualdad por n^2 , resultando $1 < c$, así si tomamos c menor que 1, por ejemplo $1/2$, nunca n^2 será menor que $n^2/2$, por lo que no puede existir tal $n_0 > 0$



Propiedades

Transitivity:

$$\begin{aligned} f(n) = \Theta(g(n)) \text{ and } g(n) = \Theta(h(n)) &\text{ imply } f(n) = \Theta(h(n)) , \\ f(n) = O(g(n)) \text{ and } g(n) = O(h(n)) &\text{ imply } f(n) = O(h(n)) , \\ f(n) = \Omega(g(n)) \text{ and } g(n) = \Omega(h(n)) &\text{ imply } f(n) = \Omega(h(n)) , \\ f(n) = o(g(n)) \text{ and } g(n) = o(h(n)) &\text{ imply } f(n) = o(h(n)) , \\ f(n) = \omega(g(n)) \text{ and } g(n) = \omega(h(n)) &\text{ imply } f(n) = \omega(h(n)) . \end{aligned}$$

Reflexivity:

$$\begin{aligned} f(n) &= \Theta(f(n)) , \\ f(n) &= O(f(n)) , \\ f(n) &= \Omega(f(n)) . \end{aligned}$$

Propiedades

Symmetry:

$$f(n) = \Theta(g(n)) \text{ if and only if } g(n) = \Theta(f(n)) .$$

Transpose symmetry:

$$f(n) = O(g(n)) \text{ if and only if } g(n) = \Omega(f(n)) ,$$

$$f(n) = o(g(n)) \text{ if and only if } g(n) = \omega(f(n)) .$$

Propiedades

$f(n) = O(g(n))$ is like $a \leq b$,

$f(n) = \Omega(g(n))$ is like $a \geq b$,

$f(n) = \Theta(g(n))$ is like $a = b$,

$f(n) = o(g(n))$ is like $a < b$,

$f(n) = \omega(g(n))$ is like $a > b$.

We say that $f(n)$ is *asymptotically smaller* than $g(n)$ if $f(n) = o(g(n))$, and $f(n)$ is *asymptotically larger* than $g(n)$ if $f(n) = \omega(g(n))$.