Reglas generales del análisis de algoritmos

Las siguientes reglas resultan ser muy útiles probando enunciados sobre el comportamiento asintótico de funciones y la complejidad computacional de algoritmos.

A continuación suponemos que Z representa cualquiera de los conjuntos O, Ω, Θ, o, ω y que

$$T, f, g : [0, \infty) \to [0, \infty),$$

son funciones de N, donde N mide el tamaño de los datos de entrada de un algoritmo. Las reglas son:

1. En cuanto a estilo y notación.
   a) **Notación**: Técnicamente debería escribirse $f(N) \in Z(g(N))$; sin embargo, resulta más útil el abuso de notación $f(N) = Z(g(N))$.
   b) **Estilo**: Al expresar el comportamiento asintótico note que
      - Es inapropiado escribir $f(N) \leqslant O(g(N))$ porque la desigualdad es implícita en la definición.
      - Es incorrecto escribir $f(N) \geqslant O(g(N))$ porque no tiene sentido y contradice la definición.
      - Un criterio similar se puede aplicar para $f(N) \geqslant \Omega(g(N))$, $f(N) < o(g(N))$ y $f(N) > \omega(g(N))$.
2. Sea $a \in \mathbb{R}$ una constante y $k, p \geqslant 0$, entonces
   a) **Constantes**: No escriba $T(N) = Z(aN^k)$, escriba $T(N) = Z(N^k)$; las constantes son ignoradas asintóticamente.
   b) **Órdenes**: No escriba $T(N) = Z(N^k + N^p)$ donde $k > p$, escriba $T(N) = Z(N^k)$; los órdenes inferiores son ignorados asintóticamente.
3. Si $T_1(N) = O(f(N))$ y $T_2(N) = O(g(N))$, entonces
   a) **Regla de la suma**: $T_1(N) + T_2(N) = O(f(N) + g(N))$
   b) **Regla del producto**: $T_1(N) \cdot T_2(N) = O(f(N) \cdot g(N))$
4. Dados $f(N)$ y $g(N)$, las siguiente propiedades siempre se satisfacen:
   a) **Reflexividad**: $f(N) = \Theta(f(N))$, $f(N) = \Omega(f(N))$, $f(N) = O(f(N))$
   b) **Simetría**: $f(N) = \Theta(g(N))$ si y solo si $g(N) = \Theta(f(N))$
   c) **Transposición**: $f(N) = O(g(N))$ si y solo si $g(N) = \Omega(f(N))$
5. Si $T(N) = \sum_{i=0}^{k} a_i N^i$, es un polinomio de grado k, donde $a_k > 0$ y $k \geqslant 0$, entonces $T(N) = \Theta(N^k)$. Esto implica que $T(N) = O(N^k)$ y $T(N) = \Omega(N^k)$.
6. Sea k una constante, entonces $\lg^k(N) = O(N)$ para todo $k \geqslant 0$; es decir, los logaritmos, y sus potencias, crecen muy lentamente.

Definiciones límite de los conjuntos de funciones

Las siguientes definiciones de los conjuntos de funciones usados en el análisis de algoritmos en términos de límites, resultan útiles en la prueba de enunciados. Asumiendo las mismas hipótesis que en la sección anterior y suponiendo que

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

para el límite del cociente de la funciones $f(N)$ y $g(N)$, se tienen las siguiente definiciones:

Notación	**Definición Formal**	**Definición Límite**
$f(N) = \omega(g(N))$	$\forall c > 0,\ \exists\, n_0 > 0, \forall N \geqslant n_0,\ f(N) > cg(N)$	$L \to \infty$
$f(N) = \Omega(g(N))$	$\exists\, c, n_0 > 0, \forall N \geqslant n_0,\ f(N) \geqslant cg(N)$	$L > 0$
$f(N) = \Theta(g(N))$	sii $f(N) = \Omega(g(N))$ y $f(N) = O(g(N))$	$0 < L < \infty$
$f(N) = O(g(N))$	$\exists\, c, n_0 > 0, \forall N \geqslant n_0,\ f(N) \leqslant cg(N)$	$L < \infty$
$f(N) = o(g(N))$	$\forall c > 0,\ \exists\, n_0 > 0, \forall N \geqslant n_0,\ f(N) < cg(N)$	$L = 0$

Las definiciones límite suponen que $g(N) \neq 0$ para N suficientemente grande. Note que los conjuntos están listados en una suerte de "orden asintótico ascendente".

Ejemplos de análisis de algoritmos

Los siguientes ejemplos usan las reglas, las definiciones formales o las definiciones límite en la prueba de enunciados sobre análisis asintótico.

En los siguientes enunciados suponemos que las funciones en cuestión tienen dominio y rango en el intervalo $[0, \infty)$, es decir, son funciones siempre positivas.

Enunciado 1. *Pruebe que el tiempo de ejecución* $T(N) = N^3 + 20N + 1$ *es* $O(N^3)$.

Solución: Usando definición formal. De acuerdo a la definición del conjunto de funciones $O(N^3)$, debemos encontrar un par $c, n_0 > 0$ tales que $T(N) \leqslant cN^3$ a partir de $N \geqslant n_0$. Observe que

$$T(N) = N^3 + 20N + 1 \leqslant N^3 + 20N^3 + 1$$

para $N \geqslant 1$; adicionalmente,

$$N^3 + 20N^3 + 1 \leqslant N^3 + 20N^3 + N^3 = 22N^3,$$

y por lo tanto, $T(N) \leqslant 22N^3$ si $N \geqslant 1$. Es decir, existen

$$c = 22,\ n_0 \geqslant 1$$

tales que se satisface la desigualdad $T(N) \leqslant cN^3$ para todo $N \geqslant n_0$.

Por lo tanto, por virtud de la definición de $O(N^3)$ se tiene que $T(N) = O(N^3)$. End

Enunciado 2. *Pruebe que la siguiente función satisface* $2N^3 - 7N + 1 = \Omega(N^3)$.

Solución: Usando definición formal. Para probar que la función pertenece al conjunto $\Omega(N^3)$, debemos encontrar un par $c, n_0 > 0$ tales que $2N^3 - 7N + 1 \geqslant cN^3$ para todo $N \geqslant n_0$. Observe que

$$2N^3 - 7N + 1 = (2N^2 - 7)N + 1 \geqslant N^3 + 1 \geqslant N^3,$$

para $N \geqslant \sqrt{7}$. Por lo tanto, tenemos que $2N^3 - 7N + 1 \geqslant N^3$ siempre y cuando $N \geqslant \sqrt{7}$. Es decir, existen

$$c = 1,\ n_0 \geqslant \sqrt{7}$$

tales que se cumple que $2N^3 - 7N + 1 \geqslant cN^3$ para todo $N \geqslant n_0$. Por lo tanto, de acuerdo a la definición de $\Omega(N^3)$, hemos mostrado que $2N^3 - 7N + 1 = \Omega(N^3)$.

End

Enunciado 3. *Pruebe que si* $T_1(N) = O(f(N))$ *y* $T_2(N) = O(f(N))$*, entonces* $T_1(N) + T_2(N) = O(f(N))$.

Solución: Usando reglas. Supongamos que $T_1(N) = O(f(N))$ y $T_2(N) = O(f(N))$. Luego, por la regla de la suma $3a$ sabemos que

$$T_1(N) + T_2(N) = O(f(N) + f(N)) = O(2f(N)).$$

Observe, además, que por la regla de las constantes $2a$ sabemos que $O(2f(N)) = O(f(N))$. Por lo tanto, combinando estos dos resultados tenemos que $T_1(N) + T_2(N) = O(f(N))$, que es justamente lo que queríamos probar.

End

Enunciado 4. *Pruebe que dados* $f(N)$ *y* $g(N)$*, entonces* $f(N) = O(g(N))$ *si y solo si* $g(N) = \Omega(f(N))$.

Solución: Usando definición formal. Esta es una prueba de un si y solo si.

$\Rightarrow$) Supongamos, primero, que $f(N) = O(g(N))$. Luego, existen $c, n_0 > 0$ tales que para todo $N \geqslant n_0$, $f(N) \leqslant cg(N)$. Observe que $\frac{1}{c}f(N) \leqslant g(N)$, para todo $N \geqslant n_0$. Sea $c_\Omega = 1/c$, entonces para todo $N \geqslant n_0$

$$g(N) \geqslant c_\Omega f(N).$$

Es decir, existen $c_\Omega, n_0 > 0$ tales que para todo $N \geqslant n_0$ la desigualdad de arriba se cumple. Luego, $g(N) = \Omega(f(N))$.

$\Leftarrow$) Supongamos, ahora, que $g(N) = \Omega(f(N))$. Luego, existen $c', n_0' > 0$ tales que para todo $N \geqslant n_0'$, $g(N) \geqslant c'f(N)$. Observe que $\frac{1}{c'}g(N) \geqslant f(N)$, para todo $N \geqslant n_0$. Sea $c_O = 1/c'$, entonces para todo $N \geqslant n_0'$

$$f(N) \leqslant c_O g(N).$$

Es decir, existen $c_O, n_0' > 0$ tales que para todo $N \geqslant n_0'$ la desigualdad de arriba se cumple. Luego, $f(N) = O(g(N))$.

Por lo tanto, $f(N) = O(g(N))$ si y solo si $g(N) = \Omega(f(N))$. End

Enunciado 5. *Con las definiciones de los conjuntos de funciones O y Θ, pruebe que:*

Si $f(N) = N \lg N + O(N)$ *entonces* $f(N) = \Theta(N \lg N)$.

Esta proposición es útil en el estudio de algoritmos de ordenamiento "rápidos".

Solución: Usando definición límite. Supongamos que $f(N) = N \lg N + h(N)$ y $g(N) = N \lg N$, donde $h(N) = O(N)$. Observe que dado que $h(N) = O(N)$ podemos escribir asintóticamente $h(N) = aN^p$, donde $0 \leqslant p \leqslant 1$ y $a > 0$ es una constante finita. Entonces,

$$f(N) = N \lg N + aN^p.$$

Evaluemos, ahora, el límite

$$\begin{aligned}\lim_{N\to\infty} \frac{f(N)}{g(N)} &= \lim_{N\to\infty} \frac{N \lg N + aN^p}{N \lg N} = 1 + a \lim_{N\to\infty} \frac{N^p}{N \lg N} \\ &= 1 + a \lim_{N\to\infty} \frac{N^{p-1}}{\lg N} = 1 + a \lim_{N\to\infty} \frac{1}{N^{1-p} \lg N} \\ &= 1 + 0 = 1.\end{aligned}$$

Observe que, sin importar el valor de p en el intervalo $[0, 1]$, el numerador del límite (en el segundo sumando) siempre crece más lentamente que el denominador, dado que el logaritmo es una función monótona creciente. Entonces, el límite de esta fracción es 0 y, por lo tanto, $\lim_{N\to\infty} f(N)/g(N) = 1$.

Es decir, de acuerdo a la definición límite de $\Theta(g(N))$, si $f(N) = N \lg N + O(N)$ y $g(N) = N \lg N$, entonces $f(N) = \Theta(g(N))$. End

Enunciado 6. *Muestre que* 2^{2N} *no es* $O(2^N)$*; es decir,* $2^{2N} \neq O(2^N)$.

Solución: Usando definición límite. Observe que el siguiente límite

$$\lim_{N\to\infty} \frac{2^{2N}}{2^N} = \lim_{N\to\infty} 2^N = \infty.$$

Es decir, el valor del límite contradice la definición límite del conjunto O. Por lo tanto, 2^{2N} no es $O(2^N)$; es decir, $2^{2N} \neq O(2^N)$. End

Enunciado 7. *Sean* $f, g : [0, \infty) \to [0, \infty)$ *dos funciones de* N*, pruebe verdadero o falso que* $f(N) = O(g(N))$ *implica* $2^{f(N)} = O(2^{g(N)})$

Solución: Usando contraejemplo. El enunciado es falso porque existen $f(N)$, $g(N)$ tales que $f(N) = O(g(N))$, pero $2^{f(N)} \neq O(2^{g(N)})$.

Sean $f(N) = 2N$ y $g(N) = N$. Observe que si $c = 2$ y $n_0 = 1$, entonces $f(N) \leqslant cg(N)$ para todo $N \geqslant n_0$. Es decir, $f(N) = O(g(N))$; sin embargo, por el Enunciado anterior $2^{2N} \neq O(2^N)$. End