Complejidad

El tiempo de ejecución de un algoritmo T(n) indica el número de instrucciones a ejecutar. Ahora, hay que buscar medidas simples y abstractas, independientes del compilador, o de cualquier elemento de hardware de la computadora. Para ello, es necesario acotar de alguna forma la diferencia que se puede producir entre distintas implementaciones de un mismo algoritmo, ya sea del mismo código ejecutado por dos máquinas de distinta velocidad, como de dos códigos que implementen el mismo método. Esta diferencia es la que acota el principio de invarianza.

Principio de invarianza

Dado un algoritmo y dos implementaciones suyas, I_1 e I_2, que tardan $T_1(n)$ y $T_2(n)$ segundos respectivamente, el principio de invarianza afirma que existe una constante real $c > 0$ y un número natural n_0, tales que, para todo $n \geq n_0$, se verifica que $T_1(n) \leq cT_2(n)$.

Suponer ahora que se tienen los tiempos de ejecución de dos algoritmos que resuelven el mismo problema:

$T_1(n)=10n+4$

$T_2(n)=2n^2+2$

¿Cuál es el mejor?

Depende del tamaño del problema, ya que para n<6 es mejor el 1; pero, para n>=6, es mejor el 2. Entonces, ¿cuál se elige? Aquí es donde aparece el concepto de **complejidad**. La idea de la complejidad de un algoritmo es conocer cómo se comporta el tiempo de ejecución conforme la talla o el tamaño del problema vaya creciendo, especialmente para valores muy grandes (lo más grandes que se puedan imaginar en el peor de los casos).

Este estudio del algoritmo cuando el tamaño de entrada se vuelve muy grande se denomina **análisis asintótico del algoritmo**. Es necesario definir, entonces, la **notación asintótica**: dada una función *f*, se quieren estudiar aquellas funciones *g* que, a lo sumo, crecen tan rápido como *f*. Se denomina **cota superior o notación asintótica O(f)**. Conociendo esta cota, es posible asegurar que nunca una función será superior a esta cota.

Formalmente:

Sea f: $\mathbf{N} \rightarrow [0,\infty)$. Se define el conjunto de funciones de orden O (Omicron) de f como:

$$O(f) = \{g: \mathbf{N} \rightarrow [0,\infty) \mid \exists c \in \mathbf{R}, c>0, \exists n_0 \in \mathbf{N} \bullet g(n) \leq cf(n) \ \ \forall n \geq n_0\}.$$

Diremos que una función t: $\mathbf{N} \rightarrow [0,\infty)$ es de orden O de f si $t \in O(f)$.

¿Por qué se llama notación asintótica? Porque trata acerca del comportamiento de funciones en el límite; esto quiere decir para valores suficientemente grandes de su parámetro. Lo cual hace que los valores pequeños de las entradas no sean interesantes. Dicho de otra manera, el foco está puesto en las tasas de crecimiento, en lugar de los valores concretos.

La notación asintótica clasifica las funciones que determinan los tiempos de ejecución de los algoritmos, para que puedan ser comparadas.

Se pueden encontrar ciertas similitudes entre las funciones que definen la complejidad de los algoritmos. Por ejemplo, si se comparan todos los algoritmos cuya complejidad es lineal (es decir, una recta... por ejemplo: 10n+3, 32n+12, 56n+1) y se los compara con todos aquellos cuya complejidad es cuadrática (por ejemplo, $2n^2+3$, $4n^2+n$, $6n^2+4n+3$) y se dibujan sus funciones de complejidad en una gráfica, se observará que, conforme *n* se va haciendo grande, tienen un patrón de crecimiento bien diferenciado.

Entonces, se pueden agrupar todas las complejidades que crecen igual, y se las van a llamar **orden de complejidad**. Hablando más formalmente, para todas las funciones que se agrupen en un mismo orden, se encontrará una asíntota que, al multiplicarla por un valor, acote a la función superiormente cuando se esté tratando el peor caso.

Por ejemplo, todas las complejidades cuadráticas están acotadas asintóticamente por "n^2". Eso quiere decir que, para cualquiera de las complejidades cuadráticas, por ejemplo $6n^2+4n+3$, existe un valor real *c*, que hace que $6n^2+4n+3 \leq cn^2$ cuando *n* se hace muy grande, es decir, cuando $n \rightarrow \infty$

La notación para expresar esto es: por ejemplo, para decir que $6n^2+4n+3$ está acotado superiormente por la asíntota n^2, se dice "que $6n^2+4n+3$ pertenece al orden de n^2" o "que $6n^2+4n+3$ es del orden de n^2", y se lo escribe formalmente de esta manera:

$$\mathbf{6n^2+4n+3 \in O(n2)}$$

En lugar de hallar los tiempos exactos que tarda un algoritmo en solucionar los problemas, importa una asíntota que representa a todos los algoritmos cuyo tiempo crece de igual forma cuando la talla del problema tiende a infinito. Eso hace que el cálculo de la

complejidad se simplifique mucho. Lo que influye en la complejidad son principalmente los bucles y las sentencias de selección que tienen efecto sobre los bucles.

A continuación, los distintos órdenes de complejidad.

Jerarquía de órdenes de complejidad

O(1)	Constante	No depende del tamaño del problema	Eficiente
O(log n)	Logarítmica	Búsqueda binaria	
O(n)	Lineal	Búsqueda lineal	
O(n•log n)	Casi lineal	Quick-sort	
$O(n^2)$	Cuadrática	Algoritmo de la burbuja	Tratable
$O(n^3)$	Cúbica	Producto de matrices	
$O(n^k)$ $k>3$	Polinómica		
$O(k^n)$ $k>1$	Exponencial	Algunos algoritmos de grafos	Intratable
O(n!)	Factorial		

Propiedades

1. Para cualquier función *f* se tiene que $f \in O(f)$.
2. $f \in O(g) \Rightarrow O(f) \subset O(g)$.
3. $O(f) = O(g) \Leftrightarrow f \in O(g)$ y $g \in O(f)$.
4. Si $f \in O(g)$ y $g \in O(h) \Rightarrow f \in O(h)$.
5. Si $f \in O(g)$ y $f \in O(h) \Rightarrow f \in O(\min(g,h))$.

6. Regla de la suma: Si $f_1 \in O(g)$ y $f_2 \in O(h) \Rightarrow f_1 + f_2 \in O(\max(g,h))$.
7. Regla del producto: Si $f_1 \in O(g)$ y $f_2 \in O(h) \Rightarrow f_1.f_2 \in O(g.h)$.

Teorema

1- f es de menor orden que g sii $O(f(n)) \subset O(g(n))$.

2- f es de mayor orden que g sii $\Omega(f(n)) \subset \Omega(g(n))$.

Nota: $\subset$ denota en este tema la inclusión estricta.

Los siguientes teoremas son útiles para comparar tasas de crecimiento de funciones.

TEOREMA

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

(f y g son del mismo orden).

TEOREMA

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \rightarrow f(n)$ crece más rápidamente que $g(n)$,

o lo que es lo mismo: $O(g(n)) \subset O(f(n))$.

TEOREMA

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \rightarrow g(n)$ crece más rápidamente que $f(n)$.

o lo que es lo mismo: $O(f(n)) \subset O(g(n))$.

COROLARIO

De menor a mayor orden:

$O(1) \subset O(\log(n)) \subset O(n) \subset O(n\log(n)) \subset$

$O(n^2) \subset O(n^3) \subset ... \subset O(K^n) \subset O(n!) \subset O(n^n)$.